



WENDY

Weighted Embedding Network Discovery of Themes

AI for Better Learning: Automating the Organisation of Learning Concept Maps to Improve Student Understanding *Students as Researchers (Strand 2) — Final Project Report*

Project period: 2 February – 31 July 2026

Staff Leads —
Natalia Yerashenia,
Alexander Bolotov,
Rolf Banziger,
Babu Nadaf,
Nalaka Dissanayake,
David Chan You Fee.

Research Officers —
Zahra Asghari, Oliver Guy

Research Assistants

Undergraduate —
Fridali Gaias, Veerpal Birdi, Akil Hossain, Cristian Leyton Medina, Jay Patel, Maurya Patel, Harrishan Sivarajah, Tasnime Yerboub and Heidi Pjetri.

Postgraduate —
Sumitra Sastri, Aleksandra Makiel and Alexandru-Iulian Iurea

School of Computer Science and Engineering,
University of Westminster

Abstract

WENDY (Weighted Embedding Network Discovery of Themes) investigates how artificial intelligence can support the creation, organisation, validation and use of educational knowledge structures. Conducted through the University of Westminster's Students as Researchers Programme, the project brought together academic staff, doctoral researchers, and undergraduate and postgraduate students within a vertically integrated research team.

The project addresses the difficulty of manually constructing and maintaining concept maps from heterogeneous teaching materials. It develops a human-centred methodology in which large language models support the extraction of concepts and relationships, embedding-based methods identify broader themes, educational ontologies provide a formal verification framework, and GraphRAG enables natural-language access to structured module knowledge. A conversational-assistant framework is also proposed to support academics in reviewing and refining generated structures and to help students explore concepts, prerequisites and learning pathways.

The Level 4 Mathematics for Computing module and its existing SMARTTEST representation provide the principal educational context and expert-curated case study. Rather than delivering a fully integrated production system, WENDY develops and examines complementary prototypes, design frameworks and validation procedures that can inform future implementation in SMARTTEST or similar graph-capable learning environments. Its main contribution is a reusable methodology that combines AI-assisted automation with computational quality assurance and academic oversight, ensuring that generated educational structures remain traceable, reviewable and pedagogically controlled.

Keywords

Educational knowledge graphs; concept-map generation; large language models; GraphRAG; educational ontologies; human-in-the-loop validation; theme discovery; conversational AI.

Table of Contents

Abstract	1
Keywords.....	1
1. Introduction	7
1.1 Project Context: Students as Researchers	7
1.2 The WENDY Project and Research Team	8
1.3 Problem Statement.....	10
1.4 The SMARTEST Platform as the Project Context and Testbed.....	12
1.5 Project Scope, Aim and Objectives	16
1.6 Project Deliverables	17
2. Related Work	20
2.1 Educational Foundations: Concept Maps, Knowledge Graphs and Conversational Learning Support.....	20
2.2 O1: Conversational Assistance	23
2.2.1 Conversational Agent Architectures and Technology Selection	23
2.2.2 Grounding and Governing Conversational Responses	25
2.3 O2: Concept and Relationship Extraction	26
2.3.1 LLM-Based Concept and Relation Extraction.....	26
2.3.2 Graph-based Retrieval-Augmented Generation (GraphRAG).....	27
2.4 O3: Topic and Theme Modelling	28

2.4.1 Embedding-Based Topic Modelling	28
2.4.2 The Short-Text Limitation	29
2.5 O4: Hybrid AI/Human Validation Framework	29
2.5.1 Ontology-Based Validation of Knowledge Structures	30
2.5.2 Human-in-the-Loop Governance of AI-Generated Content	30
3. Methodology and System Design	32
3.1 Overall Architecture	32
3.2 Ontology-Guided Concept-Map and Knowledge-Graph Generation	35
3.2.1 Ontology as the Verification Layer	35
3.2.2 Domain and Scope Definition	36
3.2.3 Conceptualisation of the Educational Domain	36
3.2.4 Definition of Relationships and Constraints	37
3.2.5 Implementation in Protégé	37
3.2.6 Iterative Generalisation and Expert Review	38
3.2.7 LLM-Based Knowledge Extraction and GraphRAG Pipeline	38
3.3 Integrated Validation and Human-in-the-Loop Quality Assurance	40
3.4 Conversational Assistant	41
3.4.1 Use Cases	42
3.4.2 GUI Design Samples	44
3.4.3 Benchmarking AI Responses	47
3.5 Topic Extraction and Theme Discovery	50

4. Implementation	52
4.1 Technology Stack.....	52
4.2 O1 Conversational Assistant Benchmarking	55
4.3 O2 Knowledge Graph Generation Pipelines	56
4.3.1 O2.1 Pipeline: Schema-Constrained Cloud-Based Generation.....	57
4.3.2 O2.2 Pipeline: Locally Hosted Generation with SHACL Validation	78
4.3.3 Comparative Summary and Conclusions	80
4.4 O3 Theme Discovery Implementation	81
4.4.1 Dataset Selection and Corpus Profiling	81
4.4.2 Experimental Pipeline and Configuration.....	82
4.4.3 Experimental Programme	82
4.5 O4 Validation Framework Implementation	83
4.5.1 Class Structure	84
4.5.2 Object Properties	84
4.5.3 Data Properties.....	86
4.5.4 Populated Mathematics for Computing Case Study	87
4.5.5 Reasoning and Consistency Checking	89
4.5.6 Role in the Implemented Pipeline	89
4.6 Reproducibility and Deployment.....	90
5. Evaluation and Testing	90
5.1 Evaluation Methodology	90

5.1.1 Benchmarking Framework and Test Scenarios.....	90
5.1.2 O1: Conversational Response Benchmarking.....	91
5.1.3 O2: Ontology Conformance and Graph Quality.....	91
5.1.4 O3: Clustering Quality Metrics.....	93
5.1.5 O4: Reasoning and Constraint-Based Validation	93
5.2 Results by Objective.....	93
5.2.1 O1: Conversational Assistant Benchmarking Results	93
5.2.2 O2: Knowledge Graph Generation Results	94
5.2.3 O3: Theme Discovery Experimental Results.....	97
5.2.4 O4: Validation Framework Results	98
5.3 Case Study.....	98
5.4 Summary of Evaluation Findings.....	99
6. Team Work and Collaboration	100
6.1 Overview	100
6.2 Meeting Rhythm and Governance.....	100
6.3 Data and Code Management	101
6.4 Working Across Work Packages	101
6.5 Pedagogical Reflection.....	102
7. Ethics, Cybersecurity, Sustainability and Responsible AI.....	103
7.1 Responsible AI Stance: AI Supports, Not Replaces, Academic Judgement.....	103
7.2 Ethical Considerations for AI in Higher Education.....	103

7.3 Data Protection, Privacy and Consent	104
7.4 Cybersecurity	105
7.5 Sustainability and Contribution to the Sustainable Development Goals	105
8. Conclusion, Limitations and Future Work	107
8.1 Conclusions: Contributions Against the Aims and Objectives.....	107
8.2 Commentary on the Project Deliverables.....	108
8.3 Limitations	110
8.4 Future Work.....	111
8.4.1 SMARTTEST Integration Roadmap.....	111
8.4.2 Knowledge-Graph Generation (O2).....	111
8.4.3 Natural-Language Graph Access and Query Safety	112
8.4.4 Conversational Assistant (O1).....	112
8.4.5 Theme Discovery (O3).....	113
8.4.6 Validation Framework (O4).....	113
8.4.7 Programme Development and Dissemination	113
8.5 Closing Remark.....	114
References.....	115
Software and Technical Documentation	121
Supporting Materials and Appendices	123

1. Introduction

1.1 Project Context: Students as Researchers

The WENDY project was undertaken within Strand 2 of the University of Westminster's **Students as Researchers Programme**, an institutional initiative designed to embed the principle that *students are researchers* within the University's research and educational practice. The programme brings together undergraduate, Master's and doctoral students to participate in authentic research activity, thereby supporting the development of research skills, professional networks, employability and awareness of postgraduate study and research careers. At institutional level, the programme also contributes to the longer-term ambition of embedding vertically integrated research projects within the credit-bearing curriculum.

Strand 2 specifically supports School-based projects that integrate education and research through vertically and horizontally organised teams. Participating teams are expected to include students from different levels of study and, where appropriate, different disciplinary backgrounds. They are led by academic or professional-services staff and are required to produce clearly defined research outcomes, such as academic publications, conference contributions, exhibitions, research prototypes or teaching resources. Students are employed for an agreed number of hours and contribute directly to the research process rather than participating solely as consultees or recipients of the final outputs.

The Strand 2 model defines three complementary categories of participation. Staff Leads are responsible for the overall direction, organisation and delivery of the project. Doctoral students act as Research Officers, providing research leadership, mentoring Research Assistants, coordinating activities and supporting the consolidation and presentation of findings. Undergraduate and Master's students participate as Research Assistants and may contribute to activities including literature reviews, data preparation, content analysis, prototype development, experimentation, evaluation, documentation and dissemination.

This structure was particularly appropriate for WENDY because the project combined educational research, artificial intelligence, natural language processing, knowledge representation, software development and user-centred design. Its delivery required both specialist academic oversight and the practical contributions of students with different levels of technical knowledge and research experience. The Students as Researchers framework therefore provided not only the administrative context for the project but also its collaborative research methodology.

The project was organised around four interconnected work packages corresponding to its four research objectives. Students worked alongside academic specialists and doctoral researchers within these work packages, while regular cross-team activities supported the exchange of findings and the integration of the individual components. This structure enabled Research Assistants to contribute to specific technical or analytical tasks while also developing an understanding of the broader research lifecycle, including problem formulation, literature analysis, methodological design, implementation, testing, critical evaluation and research communication.

Participation in WENDY consequently had a dual purpose. First, the team investigated a substantive educational-technology problem concerning the scalable and responsible generation of educational knowledge structures. Second, the project created an environment in which students could develop research capability through direct participation in a complex, interdisciplinary and collaborative investigation.

1.2 The WENDY Project and Research Team

WENDY, an acronym for **Weighted Embedding Network Discovery of Themes**, was conducted under the broader project title *AI for Better Learning: Automating the Organisation of Learning Concept Maps to Improve Student Understanding*. The project was based in the School of Computer Science and Engineering at the University of Westminster and ran from 2 February to 31 July 2026.

The project investigated how artificial intelligence could support the creation, organisation and use of educational knowledge structures. Its initial scope included automated concept-map generation, topic and theme discovery, conversational assistance, and hybrid computational and academic validation. These components were organised into four objectives and associated work packages:

- ◆ **O1:** conversational assistance for academics and students;
- ◆ **O2:** LLM-based extraction of concepts and relationships from teaching materials;
- ◆ **O3:** discovery of themes, conceptual clusters and prerequisite patterns;
- ◆ **O4:** ontology-informed and human-in-the-loop validation.

The research team was vertically integrated and included six academic Staff Leads, two doctoral Research Officers and twelve undergraduate or Master's Research Assistants. The academic participants contributed specialist knowledge of artificial intelligence, natural language processing, topic modelling, knowledge graphs, computational ontologies, curriculum design and the SMARTEST platform. The Research Officers supported technical coordination and methodological continuity across the work packages, while the Research Assistants undertook literature-based, analytical, design, implementation, testing and documentation activities.

Table 1.2.1. WENDY project team and work-package allocation

Work package	Focus	Academic Staff Leads	Research Officers	Research Assistants
O1	AI-powered conversational assistant for supporting academic graph construction and student learning	Babu Nadaf; Nalaka Dissanayake	Oliver Guy	Fridali Gaias; Veerpal Birdi; Sumitra Sastri
O2	LLM-based extraction of concepts and relationships and generation of preliminary educational knowledge graphs	Natalia Yerashenia; David Chan You Fee	Oliver Guy	Akil Hossain; Aleksandra Makiel; Alexandru-Iulian Iurea
O3	Topic extraction, theme discovery, conceptual clustering and identification of prerequisite patterns	Rolf Banziger	Zahra Asghari	Cristian Leyton Medina; Jay Patel; Maurya Patel

O4	Ontology-informed computational validation and hybrid AI–human quality assurance	Natalia Yerashenia; Alexander Bolotov; David Chan You Fee	Zahra Asghari	Harrishan Sivarajah; Tasnime Yerboub; Heidi Pjetri
-----------	--	---	---------------	---

Natalia Yerashenia served as the project’s Principal Investigator and contributed expertise in knowledge graphs, computational ontologies, curriculum content modelling and the administration and educational use of the SMARTTEST platform. Her work was principally associated with O2 and O4, linking automated knowledge extraction with ontology-informed validation and the wider educational requirements of the project. Alexander Bolotov led the SMARTTEST platform work and contributed particularly to the development of the validation framework under O4. David Chan You Fee provided expertise in the architecture, programming and administration of SMARTTEST and supported the consideration of future platform integration across O2 and O4.

Babu Nadaf and Nalaka Dissanayake led the O1 work on the conversational assistant. This work examined how natural-language interaction could support academics in constructing and refining graph-based educational structures and assist students in navigating, understanding and revising module content. Rolf Banziger led O3 and contributed expertise in artificial intelligence, natural language processing and topic modelling, supporting the investigation of methods for identifying themes and conceptual clusters across teaching materials.

The doctoral Research Officers occupied an intermediate leadership position between the academic leads and the Research Assistants. Oliver Guy supported O1 and O2, including the conversational-assistant and concept-extraction activities. Zahra Asghari supported O3 and O4, contributing to theme-discovery and validation-related research. Their role included coordinating technical activities, mentoring undergraduate and Master’s researchers, supporting methodological consistency, consolidating findings and contributing to research documentation and dissemination.

The twelve Research Assistants were assigned to work packages according to their interests, skills and developmental needs. Their expected contributions included preparing teaching materials for processing, conducting literature reviews, running and adapting Python-based workflows, reviewing AI-generated concepts and relationships, testing prototypes, organising evaluation evidence, documenting findings and providing feedback on the proposed conversational and graph-based interfaces.

Although the work-package structure provided clear areas of responsibility, the project was designed as an integrated investigation rather than four independent studies. The output of O2 provided structured concepts and relationships that could be accessed through the O1 conversational assistant. The O3 work explored broader themes and clusters that could enrich the structures produced through direct extraction. O4 supplied the computational and expert-validation mechanisms required to assess whether AI-generated outputs were structurally valid, conceptually accurate and pedagogically appropriate. SMARTTEST provided the common educational context through which these components could be interpreted and, in future work, connected.

The team composition was therefore central to the project’s research design. It combined academic subject expertise, doctoral-level research leadership and student-led investigation within a single interdisciplinary structure. This enabled the project to address both the technical complexity of AI-assisted knowledge modelling and the pedagogical requirement that any

generated educational structures remain understandable, useful and subject to academic oversight.

1.3 Problem Statement

Virtual learning environments have substantially improved students' access to lecture materials, seminar activities, assessment information and other educational resources. However, access to content does not necessarily make the structure of that content visible. Teaching materials are commonly organised as weekly folders, slide decks and separate documents, while relationships between concepts, prerequisite knowledge and broader themes remain implicit. Students may therefore experience a module as a collection of loosely connected topics rather than as a coherent body of knowledge in which concepts build upon one another and contribute to wider learning outcomes.

Concept maps provide one means of addressing this problem by representing concepts and the relationships between them explicitly. Novak and Cañas (2007) describe concept maps as tools for organising and representing knowledge through concepts and labelled connections. Within an educational context, such representations can help learners recognise conceptual hierarchies, connect new material to prior knowledge and understand how individual ideas contribute to a wider subject structure.

Recent meta-analytic evidence supports the educational value of concept mapping. Anastasiou, Wirngo and Bagos (2024) synthesised 55 studies involving 5,364 students and reported a moderate positive effect on overall science achievement, with an effect size of $g=0.776$. Positive effects were observed across biology, chemistry, physics and earth science, although the authors also reported substantial variation across studies and instructional conditions. Their findings suggest that concept mapping can be particularly beneficial for low-achieving students and in settings where learners receive appropriate instructional guidance.

Similarly, Wang et al. (2025) reviewed 37 studies conducted between 2004 and 2023 and identified a moderate overall effect of concept mapping on achievement in STEM education, with an effect size of 0.630. The effectiveness of the approach varied according to subject, educational level, intervention duration, construction mode and pedagogical approach. Notably, the strongest outcomes were associated with students constructing their own maps rather than using only pre-prepared representations. These findings indicate that concept maps are not merely visual presentation tools; their educational value depends on how actively learners interact with them and how they are incorporated into the learning design.

These findings provide a strong pedagogical rationale for making the conceptual organisation of module content more explicit. However, they also highlight an important distinction for the WENDY project. Although student-generated concept maps may produce particularly strong learning outcomes, academics must first provide an accurate and coherent conceptual structure, appropriate guidance and suitable resources from which meaningful learning pathways can be developed. The challenge is therefore not only to visualise teaching content, but to create graph-based structures that are sufficiently accurate, interpretable and flexible to support active student engagement.

SMARTEST applies this graph-based approach by representing curriculum content as interconnected nodes and learning paths. Each node may contain a concept, example, activity or learning resource, while links express conceptual relationships and possible progression between elements. In contrast to conventional virtual learning environments based primarily on files and folders, SMARTEST is intended to make the organisation of knowledge visible and navigable. It supports multiple routes through the same body of material and allows learners to identify their

current position within a wider conceptual structure. The platform therefore provides an appropriate educational context for investigating how concept maps and related topic structures might be generated, maintained and accessed more efficiently.

The principal practical limitation is that the manual creation and maintenance of such structures requires considerable academic effort. Academics must examine teaching materials, identify central concepts, determine meaningful relationships, distinguish subject content from administrative information, connect resources to appropriate nodes and revise the resulting structures whenever module content changes. Repeating this process across modules, programmes and disciplines would be difficult to sustain at institutional scale. The problem is therefore not simply how to represent curriculum knowledge as a graph, but how to produce and maintain these representations without transferring an unrealistic additional workload to academic staff.

Teaching materials also differ substantially in format, length, structure and purpose. A single module may contain lecture slides, seminar exercises, module documentation, revision materials and assessment guidance. These resources frequently combine subject knowledge with operational information such as deadlines, staff contact details and timetabling arrangements. A scalable methodology must therefore process heterogeneous source materials while distinguishing pedagogically meaningful concepts and relationships from information that should not form part of the conceptual learning structure.

Artificial intelligence offers a potential means of reducing this manual effort. LLM-based techniques can analyse teaching materials and propose candidate concepts and relationships. Graph-based retrieval and reasoning methods can subsequently organise extracted information into structures that support querying and contextual navigation. Under this model, academics would review and refine an initial AI-generated representation rather than construct the entire concept map manually. The intended role of AI is therefore assistive: it accelerates the initial analysis and structuring of educational content while leaving responsibility for curriculum interpretation and approval with the academic expert.

Automation, however, introduces a significant risk. AI-generated structures may contain incomplete concepts, inaccurate relationships, inconsistent classifications or connections that appear linguistically plausible but are not conceptually or pedagogically appropriate. Within an educational environment, these errors are particularly significant because students may interpret generated structures as authoritative. An incorrect prerequisite relationship may misrepresent the logical progression of a subject, while an inaccurate classification or explanation may negatively affect understanding and revision. A technically valid graph is therefore not necessarily an educationally valid learning structure.

The prototype work undertaken within WENDY reinforced this distinction. Automatically generated graphs could contain coherent individual concepts while still exhibiting weak topic names, incorrect classifications, sparse semantic links or insufficient prerequisite relationships. Some issues could be identified through formal structural checks, but others required interpretation of the teaching context and subject domain. Automated extraction must therefore be combined with mechanisms for computational validation, quality assurance and academic approval.

Ontology-informed validation provides one mechanism for introducing such controls. An ontology can specify permitted types of educational entities, relationships and constraints, allowing generated outputs to be checked against an explicit model. Structural validation can identify malformed entities, unsupported relationship types and inconsistencies with the agreed graph schema. Nevertheless, automated validation cannot independently determine whether a concept has been represented at an appropriate level of detail, whether a proposed relationship is

educationally meaningful or whether a learning pathway reflects the intended curriculum. Human expert review must therefore remain an integral part of the process.

A further problem concerns the identification of broader themes across substantial collections of teaching materials. Conventional topic-modelling methods can detect groups of semantically related text, but their outputs do not automatically represent educational progression, prerequisite knowledge or curriculum-level structure. BERTopic, for example, combines transformer-based embeddings with class-based term weighting to identify topics within document collections (Grootendorst, 2022). However, module resources may consist of lengthy, heterogeneous documents distributed across multiple files. Moreover, educational concepts may overlap across themes rather than belonging exclusively to a single cluster. WENDY therefore investigates approaches based on embeddings and soft clustering to identify themes and conceptual groupings while preserving the possibility that a concept may contribute to more than one area of the curriculum.

The usefulness of a generated knowledge graph also depends on how users can interact with it. Even a structurally sound graph may be difficult to navigate when it contains a large number of concepts, relationships and resources. Academics require a practical mechanism for reviewing, correcting and extending generated structures, while students require support in locating relevant topics, understanding relationships, identifying prerequisite concepts and selecting suitable learning pathways. This motivates the conversational-assistant component of WENDY, through which users can query and interact with the underlying educational structure using natural language.

The problem addressed by WENDY can therefore be expressed through four related requirements. First, the effort required to create and maintain educational concept maps must be reduced sufficiently to support their use across multiple modules and disciplines. Second, automatically generated concepts, relationships and themes must be validated before they are presented as trusted educational content. Third, the resulting structures must support meaningful interaction, enabling academics to review and refine them and students to navigate and learn from them. Finally, the methodology must accommodate heterogeneous teaching materials and remain sufficiently generic to support SMARTTEST as well as other graph-capable learning environments.

Accordingly, WENDY does not seek to automate curriculum design or replace academic judgement. It investigates a controlled, human-centred methodology in which AI supports document processing, concept and relationship extraction, theme discovery and natural-language interaction. Computational validation and academic review remain integral to the process, ensuring that generated outputs are treated as candidate educational structures until their conceptual accuracy and pedagogical suitability have been confirmed. The central principle of the proposed approach is therefore that AI may support the drafting and organisation of educational knowledge structures, but responsibility for their validation and final approval remains with the educator.

1.4 The SMARTTEST Platform as the Project Context and Testbed

[SMARTTEST](#) is an online, graph-based learning platform developed at the University of Westminster to represent curriculum content as interconnected learning structures rather than as a conventional hierarchy of folders and files. The platform is based on the principle that

educational content becomes more meaningful when relationships between concepts, activities and resources are made visible. It therefore provides a structural layer through which learners can explore how individual topics connect, how knowledge develops and which routes through the material may be relevant to their current learning needs.

Within SMARTTEST, curriculum content is organised through nodes and links. A node may contain a concept, example, learning activity, explanation or external resource, while links represent relationships and possible progression between nodes. Each node is therefore both a discrete unit of learning and part of a wider graph. This enables students to identify their position within a topic structure, move between related concepts and follow more than one route through the same body of material. The platform can consequently be understood as a bridge between dispersed curriculum resources and a more explicit representation of the knowledge structure underlying them.

This design differs from the organisational model typically used by standard virtual learning environments. Conventional systems commonly present content as a fixed sequence of weekly sections, folders and files. Although this supports distribution and storage, it provides limited visibility of the relationships between ideas. SMARTTEST instead presents content as a shared visual map in which links, sequencing and alternative pathways are explicit. The same source material may be reused within different learning paths without unnecessary duplication, while students can explore, search and reflect on the content through its conceptual relationships.

The platform supports several activities relevant to both curriculum design and student learning. Users can navigate through connected nodes and arrows, search for content and receive a relevant learning path rather than only a list of files, construct or rearrange structures through a visual authoring interface, and export selected paths for offline study. This combination of navigation, querying, authoring and export distinguishes SMARTTEST from a static visualisation tool. Its purpose is not simply to display a completed concept map, but to support interaction with evolving educational structures.

SMARTTEST has been applied within the University of Westminster in contexts including Mathematics for Computing (see Figures 1.1 and 1.2) and Final-Year Project provision, with use extending from Level 4 to Level 6 and across the University of Westminster and Westminster International University in Tashkent. The platform materials report more than 300 registered students and indicate that students particularly valued its step-by-step organisation for revision and locating learning materials. These observations provide practical support for the view that graph-based organisation can assist learners in navigating complex content, although they do not in themselves constitute a formal evaluation of learning gain.

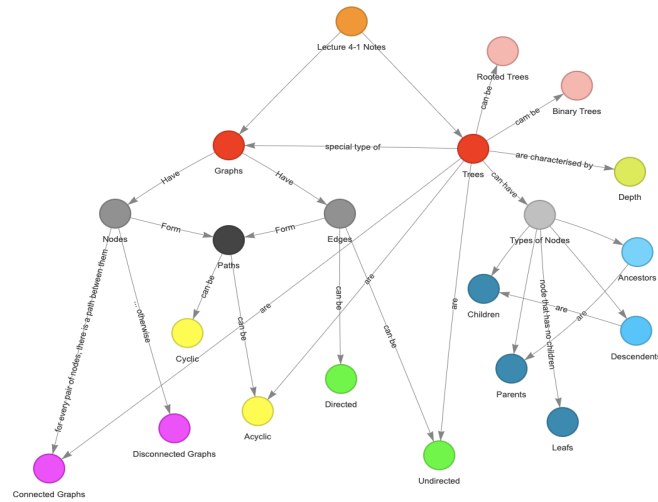


Figure 1.4.1. SMARTEST interface: Mathematics for Computing Module Learning Path

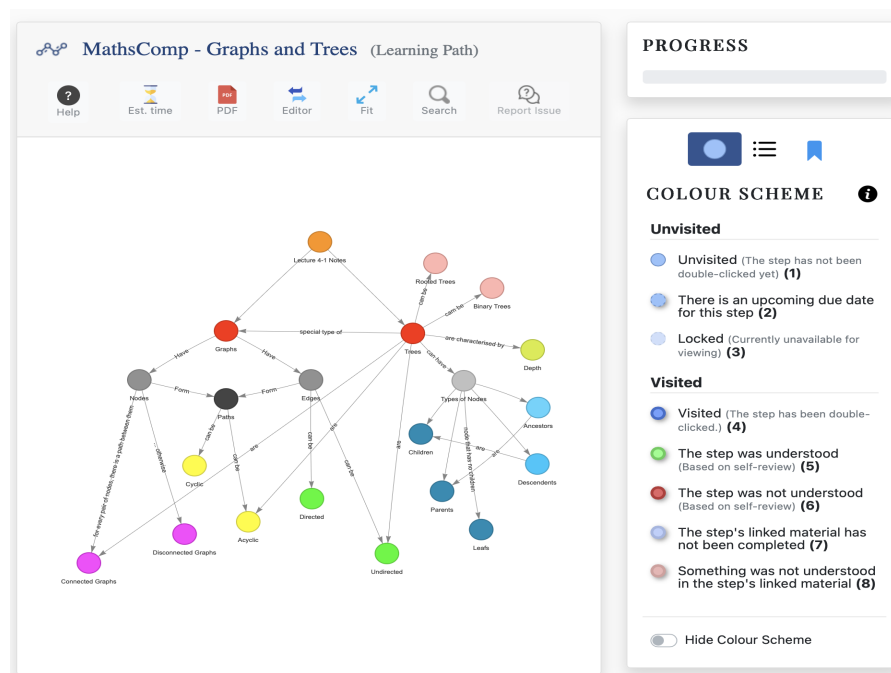


Figure 1.4.2. SMARTEST interface: Mathematics for Computing Module Graph Theory Lecture Concept Map

Despite these benefits, the platform also exposes the central scalability problem addressed by WENDY. The usefulness of SMARTTEST depends on the quality and completeness of its underlying concept maps and learning pathways. At present, these structures require academics to identify concepts, define links, organise content and update the graph when teaching materials change. Although the platform allows content to be authored without programming, the intellectual work of interpreting a curriculum and translating it into a coherent graph remains substantial.

This process becomes increasingly difficult when a module contains numerous lectures, seminar materials, assessment documents and supporting resources. Manual graph construction may also produce inconsistencies where different academics adopt different levels of conceptual detail, naming conventions or relationship types. Furthermore, as modules are revised, existing structures may become outdated unless they are reviewed and amended regularly. The challenge is therefore not limited to creating an initial graph; it also concerns maintaining an accurate and pedagogically meaningful structure over time.

WENDY uses SMARTTEST as the principal educational context in which this challenge can be investigated. The platform is particularly suitable because it already represents learning materials through concepts, relationships and pathways, corresponding directly to the outputs that the project's AI-assisted methodology is intended to produce. The concept- and relationship-extraction work can generate candidate graph elements; theme-discovery methods can identify broader structures; the validation framework can assess the coherence of generated outputs; and the conversational-assistant component can provide a natural-language interface through which academics and students interact with the graph.

However, WENDY should not be understood as a project whose primary outcome is a completed software extension for SMARTTEST. The report demonstrates that full platform integration would require further work on the user interface, backend services, database connectivity, authentication, data governance, deployment and performance. For example, the proposed architecture considers interaction between a conversational interface, LLM services, retrieval mechanisms, MongoDB, Neo4j and the existing SMARTTEST frontend. This introduces dependencies and engineering complexity beyond the scope of the current Students as Researchers project.

The project therefore treats SMARTTEST primarily as a **reference environment, case study and future integration target**. This distinction allowed the research team to focus on the methodological questions that precede production-level implementation: how teaching documents should be processed; how candidate concepts and relationships should be represented; how weak or irrelevant outputs should be filtered; how graph structures should be validated; how academics should review proposed changes; and how users might query validated educational content.

This methodological emphasis also supports greater transferability. Although the project was informed by the structure and requirements of SMARTTEST, the resulting workflow is not intended to depend exclusively on one platform. Its core components—document ingestion, LLM-based extraction, theme discovery, graph representation, ontology-informed validation and human approval—could be adapted to other digital education environments capable of storing or presenting structured concepts and relationships. SMARTTEST therefore provides a realistic and institutionally relevant testbed, while the wider contribution of WENDY lies in establishing a platform-independent approach to AI-assisted educational knowledge modelling.

Accordingly, the relationship between SMARTTEST and WENDY is best understood as complementary. SMARTTEST demonstrates how graph-based curriculum structures can support visual and non-linear learning, while WENDY investigates how the creation and maintenance of such structures might be made more scalable, systematic and reliable. The platform provides the

pedagogical and technical context; the project contributes the methodology, prototype components and evidence required to inform its future AI-supported development.

1.5 Project Scope, Aim and Objectives

The initial WENDY proposal envisaged an integrated AI-supported environment combining automated concept-map generation, theme discovery, conversational assistance and hybrid validation. During the project, this scope was refined to focus primarily on the **methodology required to generate, validate and use educational knowledge structures**, rather than on delivering a fully integrated production system for SMARTTEST. This reflected the technical and pedagogical complexity of combining document processing, LLM-based extraction, graph construction, topic discovery, validation, database storage, natural-language interaction and platform integration within a single project.

Accordingly, the principal aim of WENDY was to investigate how artificial intelligence could support the scalable creation and use of educational knowledge structures while retaining academic control over their accuracy and pedagogical value. The project pursued two complementary aims:

A1: To automate and support academics in any discipline in developing concept maps and topic outlines.

A2: To guide students in learning and navigating these educational knowledge structures.

These aims were addressed through four interrelated objectives:

O1 — Conversational assistance. To propose architectural and GUI designs and identify implementation tools for an AI-powered conversational assistant that supports academics in constructing, refining and maintaining conceptual structures, while also helping students to understand topics, navigate relationships, access module information and follow appropriate learning pathways.

O2 — Concept and relationship extraction. To develop LLM-based methods for identifying concepts and relationships within teaching materials and producing preliminary graph-based representations for subsequent academic review.

O3 — Theme discovery. To investigate methods for identifying overarching themes, conceptual clusters and prerequisite patterns across collections of educational materials, including embedding-based and soft-clustering approaches.

O4 — Hybrid validation. To develop a validation framework that combines computational ontologies, formal constraints and academic review to assess the coherence, accuracy and pedagogical appropriateness of AI-generated knowledge structures.

The objectives were designed as components of a single methodology rather than as independent systems. The extraction work under O2 produces candidate concepts and relationships; O3 identifies broader structures across the source materials; O4 evaluates and refines the generated outputs; and O1 provides a potential natural-language interface through which academics and students can interact with the validated graph.

A central methodological principle was that AI-generated outputs should be treated as **candidate educational structures**, not as automatically approved content. The project's prototype work showed that graphs may be technically valid while still containing weak topic names, incorrect classifications, sparse semantic links or insufficient prerequisite relationships. Automated checks can identify some structural problems, but academic expertise remains necessary to determine

whether concepts are represented at an appropriate level, whether relationships are educationally meaningful and whether the resulting graph reflects the intended curriculum.

The final project scope therefore comprised the development and evaluation of component workflows and prototypes rather than a complete SMARTTEST-integrated system. These included document ingestion, schema-constrained concept and relation extraction, graph construction, ontology and SHACL-based validation, Neo4j storage, theme-discovery methods, conversational-assistant use cases, interface concepts and natural-language graph querying. Together, these components establish a reusable methodology for future implementation within SMARTTEST or another graph-capable learning environment.

The main contribution of WENDY is consequently a controlled, human-centred process through which existing teaching materials can be transformed into structured and navigable representations of knowledge. AI is used to reduce repetitive manual work and support extraction, organisation, discovery and interaction, while computational validation and academic review remain integral to the approval of the final educational structures.

1.6 Project Deliverables

The WENDY project delivers a set of methodological, technical and research outputs aligned with its four objectives. As the project focuses on establishing a reusable methodology rather than producing a fully integrated SMARTTEST application, the deliverables comprise prototype workflows, design frameworks, evaluation methods, technical artefacts and documented requirements for future implementation. Collectively, they define how teaching materials can be processed, transformed into candidate educational knowledge structures, validated and accessed through graph-based and conversational technologies.

D1 — Conversational Assistant Framework

The first deliverable is a structured framework for an AI-powered conversational assistant designed to support both academics and students.

For academics, the assistant supports activities such as uploading teaching materials, constructing and refining graph structures, linking concepts, updating module information and reviewing student feedback. For students, it supports access to module information, topic explanations, self-assessment, personalised revision guidance and navigation between related concepts.

This deliverable includes defined use cases, conversational flows, prompt-design principles, interface concepts and a benchmarking method for comparing the quality of responses generated by different LLMs. The evaluation framework considers correctness, completeness and communication quality using module materials, extracted ontology data, evidence files and simulated student profiles.

D2 — Educational Knowledge-Graph Construction Pipeline

The second deliverable is a prototype workflow for transforming teaching materials into preliminary educational knowledge graphs.

The pipeline includes document ingestion and preprocessing, schema-constrained extraction of concepts and relationships, semantic filtering, structured output generation and graph construction. It demonstrates how LLMs can identify candidate concepts and relationships from lecture materials and convert them into a machine-readable format suitable for storage in Neo4j.

The resulting graph is treated as a draft representation rather than final educational content. It

therefore remains subject to computational validation and academic review before it can be used within a learning environment.

D3 — Natural-Language Knowledge-Graph Access

The third deliverable is a prototype approach for accessing educational knowledge graphs through natural-language queries.

This component investigates how user questions can be translated into graph operations and how relevant information can be retrieved from Neo4j and presented in an accessible form. It demonstrates the potential for natural-language interaction to reduce the technical barrier associated with graph databases and to support future GraphRAG-based access to validated educational content.

The deliverable also identifies requirements relating to schema guidance, query generation, response reliability and safe execution.

D4 — Theme-Discovery Methodology

The fourth deliverable is a methodology for identifying broader themes, conceptual clusters and potentially overlapping topic structures across collections of teaching materials.

The approach is based on sentence embeddings and soft clustering, allowing individual concepts or content items to contribute to more than one thematic group where appropriate. This is particularly relevant to curriculum content, where concepts frequently connect several topics, modules or disciplinary areas.

The deliverable defines the methodological basis for detecting broader curriculum structures that may not be explicitly stated within individual documents.

D5 — Hybrid Validation Framework

The fifth deliverable is a hybrid framework for validating AI-generated educational knowledge structures.

The framework combines ontology-based modelling, formal constraints, SHACL validation, semantic filtering and academic expert review. Automated checks identify malformed entities, unsupported relationships, missing properties and inconsistencies with the agreed graph schema.

However, the framework also recognises that formal validity does not guarantee educational quality. Academic review remains necessary to assess conceptual granularity, disciplinary accuracy, the appropriateness of relationships and the pedagogical usefulness of the overall graph. The validation process therefore combines computational quality assurance with human-in-the-loop approval.

D6 — SMARTEST Integration Architecture and Requirements

The sixth deliverable is a high-level architecture and a set of requirements for future integration with SMARTEST.

The proposed architecture connects the conversational interface, LLM services, retrieval mechanisms, Neo4j graph storage, user-related data and the SMARTEST frontend. It distinguishes between academic and student workflows and considers graph construction, content retrieval, personalisation, conversational memory and user interaction.

This deliverable also documents important implementation considerations, including authentication, privacy, data governance, deployment cost, maintainability, system performance and frontend integration. It therefore provides a roadmap for future development rather than a completed platform extension.

D7 — Evaluation Framework and Evidence

The seventh deliverable is a set of evaluation methods and evidence generated across the four work packages.

This includes comparative assessment of LLM responses, analysis of generated graph structures, validation results, prototype testing and critical examination of methodological limitations. The documented issues include hallucinated or omitted content, inconsistent classifications, weak topic naming, sparse semantic links, insufficient prerequisite extraction, prompt-example leakage and limitations in whole-graph validation.

This deliverable establishes which parts of the methodology are currently feasible and which require further development before deployment in a live educational environment.

D8 — Academic Report and Supporting Research Artefacts

The final deliverable is the consolidated academic report together with the supporting research artefacts produced by the project team.

These artefacts include literature reviews, workflow diagrams, prototype code, technical documentation, testing and benchmarking materials, interface designs, case-study outputs, workshop materials and records of team contributions.

Together, the deliverables establish a reusable methodology for moving from unstructured teaching materials to candidate graph-based educational structures through AI-assisted extraction, theme discovery, validation and natural-language interaction. Their principal contribution is to demonstrate how these activities can be combined within a controlled, human-centred workflow that supports future implementation in SMARTTEST or a similar graph-capable learning environment.

2. Related Work

This chapter reviews the literature underpinning WENDY. Section 2.1 establishes the shared educational foundation: concept maps, knowledge graphs, ontologies and the general evidence base for conversational agents in education. Sections 2.2 to 2.5 then review the literature specific to each of the four work packages in turn — O1 conversational assistance, O2 concept and relationship extraction, O3 topic and theme modelling, and O4 hybrid AI/human validation — each structured around two subsections and closing with the specific gap that work package addresses.

2.1 Educational Foundations: Concept Maps, Knowledge Graphs and Conversational Learning Support

Concept maps are graphical representations in which concepts are expressed as nodes and relationships are represented through labelled links. Unlike diagrams that merely place related terms near one another, a concept map normally expresses propositions: meaningful statements formed by connecting two or more concepts through a linking phrase, and commonly organised hierarchically, with broader concepts positioned above more specific ones, although the exact structure depends on the purpose and context of the map (Novak and Cañas, 2008). The theoretical basis of concept mapping is closely associated with meaningful learning: incorporating new information into an existing cognitive structure rather than memorising isolated facts. Concept maps externalise part of that structure by requiring learners or educators to identify central concepts, organise them hierarchically and articulate the relationships between them, and can therefore reveal whether knowledge is represented as a set of disconnected statements or as an integrated conceptual system. De Ries et al. (2022), reviewing the use of open-ended concept maps as a research instrument, note that this same externalising property is what makes maps useful not only for teaching but for studying how learners' understanding is structured and how it changes over time.

Evidence from synthesis studies supports the general educational value of concept mapping, while also cautioning against treating it as a uniform intervention. Nesbit and Adesope's (2006) meta-analysis of 55 studies involving 5,818 participants found that learning through constructing, modifying or viewing node-link maps was generally associated with increased knowledge retention, although the effect size varied with the instructional condition, comparison activity and manner of use. Anastasiou, Wirngo and Bagos (2024), synthesising 55 studies involving 5,364 learners, identified a moderate positive effect on science achievement, with particularly notable benefits for lower-achieving students and where sufficient teacher guidance was available. Wang et al. (2025), reviewing STEM education studies between 2004 and 2023, likewise found a moderate positive effect, influenced by educational level, discipline, intervention duration, pedagogical approach and whether students construct the maps themselves. Collectively, this evidence supports understanding concept mapping as a cognitive and pedagogical activity rather than simply a visual format: it can support both knowledge organisation and metacognition for

students, and can equally serve academics as a curriculum-design and reflection tool, by revealing gaps, unnecessary duplication or weak progression in a module or programme. The current report recognises this dual role by treating knowledge structures as resources that academics construct and refine and that students subsequently navigate and learn from.

A distinction is nonetheless required between concept maps, knowledge graphs and ontologies. A concept map is primarily a human-readable pedagogical representation; a knowledge graph represents entities and relationships in a form that can be stored, queried and processed computationally, normally requiring consistent entity types, relationship types and identifiers rather than the flexible natural-language wording a concept-map link may use. Educational knowledge graphs apply this structured representation to educational domains. Ji et al.'s (2022) survey of knowledge graphs more broadly characterises the field around three activities — representation, acquisition and application — that map directly onto WENDY's own workflow of modelling, extracting and using educational structures. Within education specifically, Qu et al. (2024) identify applications including personalised learning, curriculum design, concept mapping, semantic search, question answering, educational management and resource recommendation, while Fettach, Ghogho and Benatallah's (2022) survey of knowledge graphs in education and employability similarly finds their value concentrated in personalisation and skills-to-curriculum matching. Their value for personalisation arises from the explicit representation of relationships: if a graph records that one topic is a prerequisite for another, associates learning resources with both topics and contains evidence of a learner's previous performance, a system can recommend a path that differs from the fixed sequence followed by all students. Tang et al. (2026) frame this more broadly as knowledge graphs acting as "cognitive scaffolding", arguing that their pedagogical value lies as much in making dependency structures visible to learners as in any downstream recommendation they enable.

Nevertheless, educational knowledge graphs do not automatically become effective learning tools simply because curriculum data is represented as nodes and edges. Much of this literature concentrates on graph construction, storage, recommendation algorithms and backend reasoning; the resulting structures may remain largely invisible to students, or exposed through interfaces that do not adequately explain why particular concepts or resources are connected. A further challenge is that educational relationships are more varied than general semantic similarity — two lecture sections may discuss related vocabulary without one being a prerequisite for the other, while a prerequisite relationship may not be stated explicitly in any document because it is assumed by the academic — so a useful educational graph must distinguish topical association from instructional dependency rather than treating all connections as equivalent. Ontologies provide one way of making these distinctions explicit. Noy and McGuinness (2001) define an ontology as a formal description of the concepts in a domain, their properties and the relationships between them, supporting a shared vocabulary, reuse of domain knowledge and explicit representation of assumptions; in an educational setting this could mean classes such as Module, Topic, Concept, Learning Resource, Assessment and Learning Activity, together with relationships such as contains topic, assesses, has prerequisite and is supported by resources. Ontology development is not simply the creation of a comprehensive list of educational terms — Noy and McGuinness emphasise that the domain and intended application must determine the

scope and level of detail, and describe ontology design as iterative, with no single universally correct representation of a domain — which is directly relevant to curriculum modelling, since the same teaching material can be represented at different levels of granularity depending on whether the graph is intended for curriculum overview, detailed revision or automated assessment.

A separate, complementary strand of literature concerns conversational agents as a means of making these structures accessible to users rather than requiring them to read a static diagram. The design of an educational conversational assistant is commonly informed by Experiential Learning Theory (ELT), which conceptualises learning as a continuous cycle of experience, reflection, conceptualisation and active experimentation (Kolb and Kolb, 2009); concept maps and knowledge graphs complement this model by making relationships between concepts visible and encouraging learners to connect new knowledge with prior understanding, and have been shown to improve meaningful learning and support knowledge organisation (Novak and Cañas, 2008; Cui and Yu, 2019). Recent systematic reviews of educational chatbots report consistent evidence that conversational AI can improve learner engagement, accessibility and motivation when integrated appropriately, and identify instructional support and cognitive guidance as two of its most significant contributions (Kuhail et al., 2023; Labadze, Grigolia and Machaidze, 2023; Ma et al., 2025; Yusuf, Money and Daylamani-Zad, 2025). Administrative support — answering routine questions about deadlines, schedules and logistics — is reported as among the most common and most reliably successful applications, reducing repetitive enquiries directed at academic staff (Wollny et al., 2021). More directly relevant to WENDY, Kuo and Chen (2023) evaluate a chatbot that uses concept maps as its underlying representation for a programming course and find measurable improvements in students' error-correction outcomes, providing direct empirical evidence — rather than a design analogy — for combining the two technologies this project also combines. Reviews focused specifically on intelligent tutoring systems reach a similar conclusion from a different angle: both Lin, Huang and Lu (2023) and Batsaikhan and Correia (2024) find that generative-AI-enhanced tutoring systems show promise for personalisation and engagement, but that most published evaluations remain short-term, small-scale or platform-specific, limiting confidence in how well the reported benefits generalise.

The gap WENDY addresses. Three gaps in this literature motivate WENDY's overall design. First, the educational benefits established for concept mapping do not automatically transfer to automatically generated knowledge graphs, since most of the evidence concerns maps constructed or actively used by learners, whereas an AI-generated graph may become another static resource unless students can meaningfully interact with it — the reason WENDY pairs graph generation (O2) with a conversational interface (O1) rather than treating either as sufficient alone. Second, automatic extraction of semantically related concepts does not necessarily identify pedagogically significant relationships such as prerequisites or progression, and formal ontologies can enforce structural constraints but cannot independently determine whether the resulting map reflects the intended curriculum — the reason WENDY separates computational validation from academic approval rather than treating either as a substitute for the other (O4). Third, while Kuo and Chen (2023) show that chatbot-plus-concept-map combinations can work, and reviews such as Lin, Huang and Lu (2023) and Batsaikhan and Correia (2024) call for more rigorous, longer-term evaluation of AI-enhanced learning tools, no existing study evaluates this

combination at the scale of a full module's teaching materials or reports the kind of systematic, multi-model benchmarking presented in Chapters 4 and 5. WENDY responds to all three gaps by combining the pedagogical principles of concept mapping with the machine-readable structure of knowledge graphs and the explicit constraints of ontology-based modelling, in a human-in-the-loop methodology in which AI proposes concepts, themes and relationships, computational models check structural consistency, and academics retain responsibility for pedagogical interpretation and approval.

2.2 O1: Conversational Assistance

Building on the general case for conversational learning support established in Section 2.1, this section reviews the technical literature O1 draws on directly: the architectural choices available for building a conversational agent, and how such an agent can be grounded in validated content and kept under academic governance.

2.2.1 Conversational Agent Architectures and Technology Selection

Educational chatbot platforms can generally be categorised as rule-based, keyword-based, retrieval-based, generative and hybrid systems (Adamopoulou and Moussiades, 2020; Attigeri, Agrawal and Kolekar, 2024). Rule-based systems match user input against predefined patterns and respond accordingly, offering predictability and no risk of hallucination but breaking easily on unforeseen input (Gonda and Chu, 2019; Singh, Joseph and Jabbar, 2019); keyword-based systems detect specific terms to select a response, which is simple to implement but provides only weak contextual understanding (Wijaya, Kusumaningtyas and Barakbah, 2019; Jia, 2003); retrieval-based systems embed the user's message and return the most similar stored answer, which is accurate and avoids fabricated answers but is limited to existing content (Wu et al., 2018; Patchava and Jonnalagadda, 2020); and generative, LLM-based systems predict responses token by token, offering flexible, natural conversation at the cost of hallucination risk and higher compute cost (Wang, Ramos and Lawrence, 2023; Nageshwaran and Nageshwaran, 2025). Table 2.2.1 summarises this taxonomy.

Type	Description	Pros / Cons	References
Rule-based (scripted)	Pattern matching or button-driven flows; "if user says X, respond with Y"; no real language understanding.	+ Easy to control, predictable, no hallucinations. – Inflexible; breaks with unexpected input.	Gonda and Chu (2019); Singh, Joseph and Jabbar (2019)
Keyword recognition	Detects predefined keywords in the user message (e.g. "refund", "order") to select a response.	+ Simple NLP, easy to implement. – Weak contextual understanding; can misinterpret intent.	Wijaya, Kusumaningtyas and Barakbah (2019); Jia (2003)
Retrieval-based	Embeds the user message and compares it against a database to return the most similar stored answer.	+ Accurate, no fabricated answers, good for domain knowledge. – Limited to existing answers; no creativity.	Wu et al. (2018); Patchava and Jonnalagadda (2020)

Generative (LLM-based)	Transformer-based models predict responses token by token; can reason, summarise and explain.	+ Natural conversation, handles unseen questions, adaptive. – Can hallucinate, needs guardrails, higher compute cost.	Wang, Ramos and Lawrence (2023); Nageshwaran and Nageshwaran (2025)
Hybrid	Combines intent classification, RAG, LLM response generation and rule-based logic.	+ Broad task coverage within one system. – Costlier to maintain; more complex architecture.	Wang (2025); Jana et al. (2026)

Table 2.2.1. Categorisation of conversational-agent architectures considered for the WENDY assistant.

Most state-of-the-art educational systems now favour generative and hybrid designs (Wang, 2025; Jana et al., 2026), and several deployed examples illustrate what this looks like in practice rather than in the abstract. Zhou et al. (2020) describe Xiaolce, a large-scale hybrid social chatbot combining empathetic generative responses with retrieval-based grounding, demonstrating early that hybrid designs can sustain long, coherent conversations at scale. Closer to WENDY's domain, Carlander-Reuterfelt et al. (2020) present JAICOB, a data-science teaching chatbot combining rule-based dialogue management with a knowledge base, while Neupane et al. (2024) describe an "informed chatbot" for answering university-resource queries by grounding a generative model in institutional documents — an architecture directly comparable to WENDY's own aim of grounding responses in module content rather than open-domain knowledge. Earlier retrieval-based systems for university FAQs (Ranoliya, Raghuwanshi and Singh, 2017) and a broader survey of chatbot implementation strategies in customer service (Nuruzzaman and Hussain, 2018) both report the same trade-off found in Table 2.2.1: retrieval-based designs are safer but brittle outside their prepared answer set, motivating the shift towards retrieval-augmented generative and hybrid designs. Aleedy, Atwell and Meshoul (2025) extend this further with a multi-agent architecture for AI-driven language learning, in which separate agents handle distinct pedagogical functions (e.g. assessment, explanation, conversation practice) rather than a single monolithic model — an architectural pattern relevant to WENDY's own separation of conversational interface (O1), extraction (O2) and validation (O4) into distinct components.

Implementing a hybrid chatbot also requires choosing among a wide range of available frameworks and APIs. Stoyanova and Anguelov (2025) propose a five-group criterion for comparing chatbot platforms — functionality, specific educational opportunities, technological characteristics, user experience and economic/organisational factors — and apply it to Google Dialogflow, IBM Watson Assistant, Rasa, Microsoft, Snips, Tock and the ChatGPT API; the ChatGPT API scored relatively highly on educational opportunities and Google Dialogflow performed well on user experience, but the marginal differences between platforms suggest there is no one-size-fits-all option, and that tool selection should instead follow the specific functional requirements and technical expertise of the implementing team. An LLM-based approach integrated with a RAG-style system is best suited to WENDY's need to retrieve information across both MongoDB and Neo4j; Llama, for its open-source access, and the ChatGPT API, for its ability to handle more complex queries, were both considered during technology selection. Table 2.2.2

lists the tools and frameworks evaluated for the WENDY chatbot implementation.

Tool / Framework	Possible Use	Pros	Cons
LangChain	Agent orchestration, RAG pipelines, tool calling, memory	Strong ecosystem; flexible; supports retrieval and tools	Can become complex for small prototypes
LlamaIndex	Document ingestion, indexing and RAG over module content	Good for knowledge-base chatbots and document retrieval	Less focused on multi-step agent workflows
FastAPI	Backend API for the chatbot service	Lightweight, Python-based, good for ML integration	Requires backend deployment
Neo4j	Knowledge-graph storage and querying	Strong graph database; good for concept relationships	Requires graph modelling and Cypher queries
PostgreSQL + pgvector	Storing content chunks and embeddings	Practical, scalable, integrates with existing systems	Less visually graph-native than Neo4j
OpenAI / Azure OpenAI API	LLM response generation and embeddings	Strong language understanding and generation	Cost, privacy and policy considerations
Hugging Face models	Open-source model experimentation	More control; possible local deployment	Requires more infrastructure and tuning
React / Next.js	Frontend chatbot UI prototype	Modern web interface; reusable components	Needs integration with SMARTEST's frontend

Table 2.2.2. Tools and frameworks considered for the WENDY chatbot implementation.

2.2.2 Grounding and Governing Conversational Responses

Although modern LLMs provide sophisticated conversational capabilities, their responses remain susceptible to hallucinations and factual inaccuracies, particularly within specialist educational domains (Ganguly et al., 2025; Kuhail et al., 2023). Research increasingly recommends Retrieval-Augmented Generation (RAG) architectures that supplement language model reasoning with external knowledge sources as a response to this risk (Lewis et al., 2020), and demonstrates that RAG improves factual accuracy, contextual relevance and explainability compared with purely generative systems (Lewis et al., 2020; Microsoft Research, 2024). The proposed conversational assistant adopts a hybrid architecture combining LLMs, RAG, Neo4j knowledge graphs and educational repositories: when a user submits a query, relevant information is retrieved from validated educational sources before being incorporated into the prompt supplied to the language model, so that responses are grounded in academic content rather than relying solely on model parameters. A more graph-aware extension of this pattern, GraphRAG, is reviewed in full in Section 2.3.2, since it is the O2 pipelines that implement it directly; for O1, the practical implication is that the assistant's own grounding is only as reliable as the knowledge graph it retrieves from, which is why the assistant's outputs are treated as recommendations rather than authoritative answers.

This is also why WENDY incorporates a human-in-the-loop governance model for the assistant's suggestions — new concepts, prerequisite links and graph modifications are treated as recommendations rather than automatic updates, with academic staff reviewing and approving all proposed changes before they reach the underlying knowledge graph. The rationale and supporting evidence for this governance model are shared with, and reviewed in full in, O4's validation framework (Section 2.5.2); the assistant is the component through which that governance is exercised in a conversational setting, guided by the academic support and student support literature discussed in Section 2.1 (Abu-Rasheed et al., 2025; Liu, Wang and Yang, 2025).

The gap WENDY addresses. Although substantial progress has been made in educational conversational agents, and the examples above (Zhou et al., 2020; Carlander-Reuterfelt et al., 2020; Neupane et al., 2024; Aleedy, Atwell and Meshoul, 2025) show that grounded, domain-specific and multi-agent designs are all individually established, few existing systems integrate conversational interaction, knowledge-graph navigation, retrieval-augmented generation and human-in-the-loop validation within a single educational framework. WENDY addresses this gap by combining these elements to support both academic knowledge-graph construction and personalised student learning, forming the foundation for the integrated educational ecosystem described in the remainder of this report.

2.3 O2: Concept and Relationship Extraction

O2 addresses two connected technical problems: extracting concepts and relationships from teaching materials in the first place, and retrieving from the resulting graph once it exists. This section reviews the literature for each in turn.

2.3.1 LLM-Based Concept and Relation Extraction

Recent advances in LLMs have transformed entity and relation extraction by enabling the automatic construction of knowledge graphs from unstructured text. This capability rests on the broader shift towards large, general-purpose language models capable of few-shot and zero-shot task performance without task-specific training (Brown et al., 2020; OpenAI, 2023); applied to extraction specifically, LLMs can perform entity recognition, relation extraction and post-processing within a single unified framework, unlike traditional NLP pipelines that rely on separate stages for each — particularly relevant for the SMARTEST platform, where lecture materials must be converted into structured concept maps.

Dagdelen et al. (2024) demonstrate that a fine-tuned LLM can extract structured entity-relationship representations directly from scientific text, comparing GPT-3 and Llama-2; a key technical contribution is the need to manually annotate 100–500 examples in a predefined JSON schema before the model can generalise to unseen documents. While this substantially reduces the complexity of traditional extraction pipelines, the authors highlight the continued risk of hallucination, reinforcing the importance of ontology-based validation when constructing

knowledge graphs. Rather than relying on supervised fine-tuning, Galletti et al. (2025) propose a decomposed prompting strategy using GPT-3.5, in which the model performs entity extraction, relation extraction, concept grouping and triple selection through sequential prompts, achieving state-of-the-art concept map generation without labelled training data; this considerably lowers development cost and makes the approach suitable for rapid prototyping, although duplicate concepts and evaluation limited to Wikipedia articles suggest that additional post-processing and validation are required before applying the approach to educational content. Perin et al. (2023) extend this line of work by demonstrating that a fine-tuned GPT-4 model can directly generate concept–relation–concept propositions from raw text, effectively replacing traditional multi-stage NLP pipelines; although the results are promising, the model was trained on a small dataset and tested under controlled conditions, limiting confidence in its performance on more diverse and complex educational content. Yang et al. (2025) apply this same extraction problem specifically to educational content, proposing methods for structuring the concepts and relationships an LLM extracts into curriculum-consistent form rather than treating extraction as a generic information-extraction task, which is the closest existing work to O2's own objective.

A separate methodological question, orthogonal to which extraction strategy is used, is how the resulting output should be evaluated. Chu, Yi-Pei and Nakayama (2024) examine how prompt structure and output ordering affect the reliability of using an LLM itself as an evaluator of generated text, finding that seemingly minor prompt-design choices can materially change evaluation outcomes — a methodological caution directly relevant to any pipeline, including O2's, that uses an LLM to check or score its own extracted output.

Collectively, these studies indicate a shift from modular NLP pipelines towards end-to-end LLM-based knowledge extraction. However, they also demonstrate a trade-off between fine-tuning and prompt engineering, with both approaches requiring robust post-processing to mitigate hallucinations and ensure semantic consistency, and results across the studies are not directly comparable due to differing tasks, datasets, evaluation metrics and baselines.

2.3.2 Graph-based Retrieval-Augmented Generation (GraphRAG)

Standard RAG grounds an LLM's response by retrieving relevant text passages from an external corpus using vector similarity search, then supplying those passages to the model as context (Lewis et al., 2020). This is effective for locating individual passages relevant to a query, but treats the corpus as an unstructured collection of chunks, with no explicit representation of how those chunks relate to one another. GraphRAG extends this pattern by retrieving over a structured knowledge graph instead of, or in addition to, flat text chunks, so that retrieval can follow explicit relationships between entities rather than relying on semantic similarity alone.

Edge et al. (2024), in the approach originating this term at Microsoft Research, construct a knowledge graph from a source corpus, detect communities of closely related entities within it using the Leiden algorithm (Traag, Waltman and Van Eck, 2019), and generate summaries at each level of the resulting community hierarchy; a query is then answered by combining the LLM's own reasoning with these community summaries, which the authors show improves performance

on queries requiring a global understanding of the corpus — the kind of question standard RAG's local, chunk-level retrieval tends to miss. Han et al. (2025) survey the broader space of graph-based retrieval-augmented methods that this approach opened up, identifying a range of graph types used for retrieval (including knowledge graphs, indices built directly from documents, and community-structured summaries of the kind proposed by Edge et al.) and a corresponding range of retrieval strategies, from single-step graph traversal to multi-hop reasoning across entities. Microsoft Research (2024, 2025) has since published both an accessible overview of the method and an open-source implementation, and practical integration guides for combining it with Neo4j specifically (Neo4j, 2024b; Neo4j Labs, 2025a) have made GraphRAG-style retrieval straightforward to adopt against the same graph database WENDY uses.

The gap WENDY addresses. None of the extraction studies reviewed above evaluates LLM-based extraction specifically on educational lecture materials, nor compares candidate models directly on the same task and dataset; a comparative evaluation between the performance of Llama 3, gpt-4o-mini and gpt-4o on Mathematics for Computing lecture materials is therefore presented in Section 5.1, to ground WENDY's model selection in evidence rather than in the general-domain findings above. Separately, existing GraphRAG literature and tooling has been developed and evaluated chiefly on general-purpose or enterprise document collections, where the priority is comprehensive, query-focused summarisation of a large corpus; none of it addresses the specific requirement of an educational knowledge graph, that retrieval must preserve pedagogical structure — prerequisite ordering, curriculum sequencing and topic dependency — rather than treating all graph relationships as equally informative. O2 addresses both gaps directly: Section 4.3 describes two independently developed extraction pipelines evaluated against the same educational corpus, and Sections 4.3.1 and 4.3.2 implement two independent GraphRAG-style query approaches over the resulting educational knowledge graph, with Section 5.2.2 reporting where each preserves or loses pedagogical structure in practice.

2.4 O3: Topic and Theme Modelling

Where Section 2.3 addressed the extraction of individual concepts and relationships from a single document, O3 is concerned with a complementary problem: identifying overarching themes, prerequisite patterns and conceptual clusters that span a collection of teaching materials. This requires unsupervised topic modelling — grouping text by shared latent themes without predefined labels — and, in particular, methods able to assign more than one theme to a passage, since teaching materials routinely discuss several topics at once.

2.4.1 Embedding-Based Topic Modelling

Classical probabilistic topic models such as Latent Dirichlet Allocation infer topics from word co-occurrence patterns across a document collection, but rely on the relatively large, stable vocabulary that longer documents provide. More recent approaches instead represent text with dense semantic embeddings produced by pretrained transformer models, most notably Sentence-BERT (Reimers and Gurevych, 2019), which fine-tunes BERT with a siamese network

structure so that semantically similar sentences are placed close together in vector space. BERTopic (Grootendorst, 2022) builds a full topic-modelling pipeline on top of such embeddings: sentence embeddings are reduced to a lower-dimensional space using UMAP (McInnes, Healy and Melville, 2018), grouped using the density-based clustering algorithm HDBSCAN (McInnes, Healy and Astels, 2017), and each resulting cluster is labelled with representative terms using a class-based variant of TF-IDF. This combination allows BERTopic to identify coherent topics without requiring the number of topics to be specified in advance, and to leave genuinely ambiguous items unassigned as outliers rather than forcing them into a poor-fitting cluster.

2.4.2 The Short-Text Limitation

A well-established limitation of this literature is that both classical and embedding-based topic models were developed and evaluated primarily on long-form documents, and degrade markedly as the unit of text shortens. Hong and Davison (2010), in an early empirical study of topic modelling applied to Twitter, showed that standard LDA performs substantially worse on short, informal text than on conventional news-length documents, because sparse word co-occurrence within a short passage provides too little evidence to estimate a reliable topic distribution. Yin and Wang (2014) responded to this problem directly, proposing a Dirichlet Multinomial Mixture model designed specifically for short text clustering, on the assumption that a short passage is likely to discuss only one topic rather than a mixture — an assumption that trades away exactly the multi-topic sensitivity that soft, embedding-based clustering is intended to provide. Qiang et al.'s (2022) survey of short text topic modelling techniques formalises this as a recognised, actively researched challenge across the field, independent of whether the underlying method is probabilistic or embedding-based: short passages simply carry less semantic context per unit, which limits how confidently any topic model can assign them.

The gap WENDY addresses. The literature above establishes both the method WENDY builds on (embedding-based soft clustering, via BERTopic and its underlying SBERT, UMAP and HDBSCAN components) and the risk it must confront (degraded performance on short text). What it does not establish is where, specifically, the breakdown occurs, or what unit of analysis is appropriate for educational material that must support multi-topic, prerequisite-aware assignment rather than single-label classification. No existing study evaluates this pipeline at sentence level against progressively larger text units on the same corpus, and none targets lecture-derived educational content specifically. O3 addresses this gap directly: Section 4.4 describes a five-experiment evaluation applying the same embedding-and-clustering pipeline across whole documents, sections and individual sentences, and Section 5.2.3 reports the resulting evidence — including the project's central negative finding that sentence-level units perform markedly worse than document- or section-level units — from which a concrete, evidenced recommendation for the appropriate unit of analysis is derived.

2.5 O4: Hybrid AI/Human Validation Framework

Where Sections 2.3 and 2.4 addressed how educational knowledge structures are generated, O4

addresses how they are checked before being trusted. Two literatures are relevant: the use of formal ontologies to validate structured data, and the specific question of how much of that validation can be automated versus how much must remain with a human reviewer.

2.5.1 Ontology-Based Validation of Knowledge Structures

The use of ontologies to validate structured data predates large language models by two decades and is well established outside education. Bench-Capon (2001) sets out the foundational case for this role, arguing that an ontology's formal specification of a domain's concepts and constraints allows it to function as an independent verification and validation layer for a knowledge-based system, distinct from validating the system's reasoning process itself. Jeong et al. (2011) demonstrate this in a biological setting, using ontology-based instance validation to catch structurally invalid entries in curated pathway databases before they propagate to downstream users — a cross-domain precedent for exactly the validate-before-store pattern O4 applies to educational knowledge graphs. Espinoza et al. (2021) treat validation as inseparable from ontology development itself, proposing an iterative process in which verification and validation activities run throughout construction rather than being applied only once, after the fact; Nazir et al. (2014) report a comparable process-improvement case for using ontologies to support requirements verification and validation more generally. Read together with the knowledge-graph survey introduced in Section 2.1 (Ji et al., 2022), this literature establishes that ontology-based validation is a mature, general-purpose technique, but one whose application to AI-generated — rather than human-authored — knowledge graphs, and to educational content specifically, is comparatively recent and less established.

2.5.2 Human-in-the-Loop Governance of AI-Generated Content

The increasing adoption of LLMs in education has intensified concerns regarding accuracy, bias, transparency and academic integrity (Ganguly et al., 2025), and researchers consistently emphasise that responsibility for curriculum design and educational quality should remain with domain experts even as AI systems support educational decision-making and content generation. Tsaneva et al. (2025) report a knowledge-graph validation approach that integrates LLM-based checking with human-in-the-loop review, finding that the combination catches error types that neither an automated check nor a human reviewer working alone reliably catches on its own. Zhai (2025), in a systematic review of generative LLMs for concept-map generation specifically, reaches a related conclusion from the evaluation side: automated structural checks and human pedagogical judgement tend to disagree on different classes of error, which argues for treating them as complementary checks rather than one substituting for the other. Zhang et al. (2026) examine this substitution question directly in an assessment context, comparing LLM-as-a-judge scoring against human ratings at scale, and find systematic disagreement concentrated in judgements that require pedagogical rather than purely factual reasoning — evidence that an LLM validator cannot simply stand in for an academic reviewer, even where it can usefully filter or triage what reaches one.

The gap WENDY addresses. This literature establishes that ontology-based validation is well

precedented in general, and that hybrid LLM/human review outperforms either checker alone, but two gaps remain specific to WENDY's setting. First, none of the ontology-validation literature above addresses AI-generated educational knowledge graphs specifically, where the source of error is a generative model's extraction behaviour (paraphrasing, omission, hallucinated relationships) rather than human data-entry mistakes, which is the failure mode O4's SHACL constraints and conformance scoring are designed against (Section 4.5). Second, while Zhai (2025) and Zhang et al. (2026) both show that automated and human judgement disagree in informative ways, neither proposes where the boundary between them should sit operationally — which checks can safely run automatically and which must always reach an academic reviewer. O4 addresses this directly by implementing and evaluating both an automated first stage (a populated, formally consistent ontology with SHACL constraints, exercised by both O2 pipelines) and a defined academic-review stage, and by reporting, in Section 5.2.4, which fault types each stage actually catches in practice.

3. Methodology and System Design

3.1 Overall Architecture

WENDY adopts a design-oriented, proof-of-concept research methodology. The project combines conceptual modelling, prototype development and exploratory evaluation to investigate how AI-assisted educational knowledge structures can be generated, validated and accessed. The architecture provides the common methodological framework within which the four work packages develop and assess individual components.

The WENDY methodology is organised as an integrated architecture for transforming existing teaching materials into structured, validated and queryable educational knowledge representations. The architecture brings together the four project objectives within a common workflow: conversational interaction, concept and relationship extraction, theme discovery, and hybrid computational and academic validation. Although these components are investigated through separate work packages, they are designed as interdependent stages of a single educational knowledge-modelling methodology rather than as standalone systems.

The high-level architecture developed by the project team (Appendix A1.1) is described below. A student or academic accesses the system through a common frontend and submits a natural-language query or request. An intent-classification layer, represented in the architecture through LangChain, interprets the request and routes it to the relevant workflow. The supported request categories include retrieving module knowledge, constructing or modifying a graph, uploading teaching materials, generating or accessing quizzes, and requesting explanations of educational concepts.

The APPENDIX A1.1 diagram, High-level WENDY system architecture, shows the interaction between the user interface, intent classification, retrieval, graph construction, validation, data storage and natural-language response components.

At the core of the architecture is the educational ontology, which functions as both a semantic model and a verification layer. The ontology defines the principal categories through which module content is represented, including modules, weeks, topics, learning resources and assessments. It also specifies the relationships that may connect these categories and the structural constraints that automatically generated outputs are expected to satisfy. The ontology therefore informs the extraction process from the outset rather than being applied only after a knowledge graph has been generated.

The architecture distinguishes between the **ontology**, the **knowledge graph** and the **source evidence**. The ontology defines the permitted educational structure. The knowledge graph contains particular instances and relationships extracted from teaching materials. The source evidence preserves the textual content from which those graph elements are derived. This separation allows the system to retain both a formal representation of curriculum structure and a traceable connection to the original educational resources.

The source materials used to develop and evaluate the methodology are drawn primarily from the current BSc Computer Science curriculum at the University of Westminster. The Level 4 **Mathematics for Computing** module serves as the principal case study. It was selected because its weekly structure, concepts, learning resources and assessments had already been organised and reviewed within SMARTEST, providing an academically curated and expert-verified reference against which generated structures could be examined.

The source corpus includes the module proforma, teaching schedule, lecture slides, seminar task

sheets, seminar answers, revision materials and assessment-related documentation. These files represent the heterogeneous forms in which module knowledge is normally distributed. They combine disciplinary content with administrative information such as teaching weeks, assessment dates, contact information and module requirements. The methodology must therefore distinguish educational concepts and relationships from material that is operationally important but should not necessarily become part of a disciplinary concept map.

The current prototype work primarily processes lecture materials exported as PDF files. Support for DOCX documents is also included to accommodate sources such as seminar activities and module documentation. The wider methodological design is file-format independent: the essential requirement is that the content can be converted into a textual representation while retaining sufficient document and source metadata to support traceability.

Before LLM-based extraction takes place, the documents pass through an ingestion and preprocessing stage. The purpose of preprocessing is not to interpret the curriculum, but to reduce irrelevant input and preserve meaningful educational content. Typical exclusions include title pages, navigation slides, repeated headers, module-team information, generic question-and-answer slides and other administrative blocks that would otherwise introduce noise into the extraction process. The remaining text is divided into manageable units for subsequent analysis.

The cleaned teaching content then enters the concept and relationship extraction workflow. LLM-based components identify candidate entities, explanatory content and relationships using a schema derived from the educational ontology. Rather than allowing unrestricted graph generation, the system limits outputs to recognised educational categories and relationship types. This supports consistency across documents and provides a basis for later verification. The detailed extraction and GraphRAG methodology is described in Section 3.2.

The concept-extraction output is complemented by the theme-discovery component. While direct extraction identifies concepts and relationships expressed within individual teaching materials, embedding-based analysis investigates broader thematic patterns distributed across multiple documents or teaching weeks. These themes may enrich the generated structure by identifying conceptual clusters and connections that are not captured through sentence-level extraction alone. The theme-discovery methodology is discussed separately in the relevant work-package section.

The resulting candidate entities and relationships are represented as a knowledge graph and stored within Neo4j. Neo4j is used for module and curriculum information because its graph-native structure supports explicit representation and traversal of relationships between topics, resources, weeks and assessments. The architecture also separates this educational knowledge from user-related information, which is conceptually stored through a document-oriented data layer such as MongoDB.

This separation supports different data-management requirements. Neo4j stores structured module knowledge, ontology-related information and evidence links, while the user-data layer may store conversational history, learning progress and personalisation information. The proposed architecture includes files or records representing user data, conversation history, module evidence and module ontology information. This design avoids embedding student-specific data directly within the curriculum graph and allows access controls and retention policies to be applied differently to educational and personal data.

After graph construction, the candidate structure enters the validation workflow. Ontology-informed and SHACL-based mechanisms assess whether entities belong to recognised classes, whether relationships conform to the permitted schema and whether required properties are present. The current ontology includes, for example, a relationship through which an assessment

may test knowledge of a topic. The domain and range of this relationship constrain the types of entities that can be connected, helping to detect structurally inappropriate assertions.

Formal validation is followed by human review. The system does not treat a graph as educationally valid merely because it conforms to the ontology. Academic experts must still determine whether the concepts are represented at an appropriate level of granularity, whether relationships are correct, and whether the resulting structure reflects the intended curriculum. This distinction is central to the WENDY methodology: automated checks support quality assurance, but responsibility for educational approval remains with the academic expert.

The conversational component operates over this structured and validated knowledge. When a user submits a knowledge request, relevant evidence is retrieved from the module knowledge base and incorporated into a prompt for natural-language response generation. This retrieval-augmented approach is intended to ground responses in module-specific information rather than relying solely on the general knowledge encoded within an LLM. The architecture also includes conversational memory so that context may be retained across an interaction.

The system distinguishes between academic and student workflows. Academics can upload materials, initiate graph construction, review or modify extracted content, and develop learning or assessment resources. Students can request explanations, access module and assessment information, complete academically validated quizzes and receive guidance based on their learning data. The diagram explicitly separates student-only and academic-only request branches, reflecting the different permissions and educational responsibilities associated with each role.

Personalisation is supported conceptually through user records containing information such as previously studied topics, quiz performance and conversational context. These data may inform recommendations concerning topics that require revision or concepts that have already been mastered. However, the use of such information introduces requirements relating to privacy, consent, access control and data minimisation. The current project therefore treats the personalisation architecture primarily as a design specification rather than as a fully deployed student-profiling system.

The architecture contains both implemented prototype components and proposed integration components. Knowledge extraction, graph construction, ontology validation, model benchmarking and natural-language graph querying were investigated through prototypes, while full frontend integration, user authentication and live personalisation remain design requirements for future development.

The architecture can be summarised through the following methodological stages:

1. source-material preparation;
2. ontology-informed extraction and theme discovery;
3. knowledge-graph construction;
4. computational and academic validation;
5. retrieval and conversational access;
6. academic refinement and student-facing use.

The architecture should be interpreted as a methodological and technical blueprint rather than as evidence that every component has been fully integrated into a production system. The project implements and evaluates selected elements through separate prototypes, particularly the concept-extraction, graph-construction, validation, question-answering and conversational-assistant components. Together, these prototypes establish how the components may operate

within a future integrated environment while also identifying the limitations and additional engineering work required for deployment.

This architecture positions the ontology as the semantic core of the system, the knowledge graph as its structured educational representation, and the original teaching materials as the evidential basis for all generated content. It thereby supports the central WENDY principle that AI-generated structures should remain traceable, formally constrained and subject to academic verification before they are used to support student learning.

The methodology was developed using materials from the Level 4 Mathematics for Computing module within the current BSc Computer Science curriculum. The source set included the module structure, weekly topics, lecture materials, learning resources and assessment information. As this content had already been organised and academically reviewed within SMARTTEST, it provided an expert-verified reference against which the ontology design and automatically generated graph structures could be examined.

3.2 Ontology-Guided Concept-Map and Knowledge-Graph Generation

The concept-map generation methodology consists of two closely connected processes. First, an educational ontology is developed to define the expected entities, relationships and constraints within a university module. Second, LLM-based extraction and GraphRAG components use this formal model to transform teaching materials into candidate knowledge-graph structures. The ontology therefore operates both as a design schema and as a reference for subsequent validation.

The concept-map generation pipeline is founded on the principle that automated extraction should not operate without an explicit representation of the educational domain. Within WENDY, the ontology therefore functions as the core organising and verification component of the system. It defines the categories of information that may be extracted from teaching materials, the relationships that may connect them and the constraints against which automatically generated structures can be checked. The purpose of the ontology is not to reproduce the complete content of a module manually, but to provide a stable semantic framework within which AI-generated concepts and relationships can be interpreted and validated.

This approach reflects established ontology-engineering principles. Noy and McGuinness (2001) define an ontology as an explicit formal specification of the concepts and relationships within a domain. Ontologies support a shared understanding of information, make domain assumptions explicit and separate domain knowledge from the applications that operate upon it. These characteristics are particularly important in an automated concept-map pipeline because the extraction model and the target learning platform may change, while the underlying educational categories and validation requirements should remain explicit and reusable.

3.2.1 Ontology as the Verification Layer

The ontology occupies a different role from the generated knowledge graph. The knowledge graph contains instances extracted from particular teaching materials, such as individual topics, weekly teaching units, resources and assessments. The ontology defines the permitted structure of those instances. It specifies, for example, that an assessment may test knowledge of a topic, that a module contains teaching weeks and that learning resources are associated with relevant parts of the curriculum.

This distinction enables the ontology to act as a reference model during extraction and validation.

Rather than allowing an LLM to generate unrestricted entities and links, the pipeline constrains extraction to an agreed educational vocabulary. Candidate outputs can then be compared with the ontology to determine whether their classes, properties and relationships conform to the expected structure. The ontology therefore serves both as an extraction schema and as a verification mechanism.

This form of validation is necessary because linguistic plausibility does not guarantee educational correctness. An LLM may produce a grammatically convincing relationship while assigning the wrong type to an entity, linking an assessment to an inappropriate resource or creating a relationship that is not recognised within the curriculum model. Ontology-based checking helps identify these structural and semantic inconsistencies before generated outputs are considered for inclusion in the educational graph. Academic review remains necessary for judgements that cannot be resolved through formal constraints, including whether a topic is represented at the correct level of detail or whether a proposed prerequisite relationship is pedagogically justified.

3.2.2 Domain and Scope Definition

Ontology development begins by defining the intended domain, purpose and scope. Noy and McGuinness (2001) recommend identifying what the ontology should represent, how it will be used, which questions it should support and who will maintain it. They also emphasise that ontology construction is iterative and that there is no single universally correct model of a domain. The appropriate design depends on the application and the level of detail required.

For WENDY, the initial domain was the structural organisation of a university module rather than the full disciplinary knowledge of Computer Science. This distinction was important. The ontology needed to represent recurring educational entities—such as modules, weeks, topics, resources and assessments—while remaining sufficiently general to support different subjects. The Mathematics for Computing module provided the initial case through which this structure was developed and tested.

The wider BSc Computer Science curriculum informed the selection of general educational categories, while the Level 4 Mathematics for Computing module supplied a detailed and internally coherent example. This module was suitable because its teaching sequence, learning topics, resources and assessments had already been represented and reviewed within the existing graph-based learning environment. It therefore provided an expert-verified reference structure against which the ontology design and subsequent extraction outputs could be assessed.

Using an existing academically curated module reduced the risk that ontology development would be guided only by automatically extracted content. The curriculum structure and its established representation supplied an authoritative starting point for defining the expected educational entities and relationships. At the same time, the ontology was progressively generalised so that its classes were not limited to Mathematics for Computing. The version history records this transition from a module-specific model to a more universal structure applicable across modules.

3.2.3 Conceptualisation of the Educational Domain

Following the definition of scope, the ontology-development process identifies the principal concepts in the domain and organises them into classes and subclasses. In WENDY, the main categories represent the organisational components of a university module. These include the module itself, its teaching weeks, curriculum topics, learning resources and assessments.

The class hierarchy distinguishes general educational categories from more specialised types. Assessments, for example, may be divided into formative and summative forms, with further

subclasses representing common assessment formats. Resources may be classified into lecture notes, recordings, reading-list materials, seminar tasks, answers and external media. Weeks provide the temporal structure through which module content is organised, while topics represent the subject matter taught and assessed. The current ontology write-up describes five principal classes—**Assessment**, **Module**, **Resource**, **Topic** and **Week**—together with subclasses for assessment forms, resource types and teaching weeks.

This hierarchical organisation serves two purposes. First, it provides a shared vocabulary for the extraction pipeline. Second, it supports validation through inheritance and domain constraints. An extracted item classified as an examination, for example, is also recognised as a summative assessment and, more generally, as an assessment. This allows rules defined at the broader class level to apply consistently to specialised instances.

3.2.4 Definition of Relationships and Constraints

Classes alone are insufficient to represent a curriculum because the educational value of a concept map depends on the relationships between its elements. The next methodological stage therefore defines object properties connecting modules, weeks, topics, resources and assessments.

These relationships should reflect the educational structure rather than simple textual co-occurrence. Examples include a module containing a week, a week covering a topic, a resource supporting a topic and an assessment testing knowledge of a topic. Domain and range restrictions specify which types of entities may participate in each relationship. For instance, a **testsKnowledgeOf** property may have **Assessment** as its domain and **Topic** as its range. The ontology write-up identifies this relationship as an important bridge between curriculum milestones and subject content.

The use of domain and range restrictions supports verification by preventing structurally inappropriate assertions. A generated relation in which a topic tests knowledge of an assessment would not conform to the intended direction or class restrictions. Similarly, a resource incorrectly classified as an assessment could be detected when its relationships conflict with the ontology.

Additional data properties may represent literal values such as module codes, academic years, week numbers, assessment weightings and assessment dates. These properties enrich the graph while remaining distinct from conceptual relationships. Their definitions should be sufficiently general to support multiple modules, but constrained enough to prevent inconsistent values.

3.2.5 Implementation in Protégé

The ontology is developed in Protégé, an ontology-authoring environment that supports OWL-based class modelling, object and data properties, individuals and automated reasoning. The methodological process is iterative: classes are initially defined, relationships and constraints are added, sample individuals are introduced and the resulting model is checked for consistency.

Protégé supports the separation between the ontology's schema and its populated knowledge base. During development, representative module entities may be created as individuals to test whether the model supports the intended relationships. These examples are used as proof of concept rather than as permanent curriculum data where the source information has not yet been finalised. The version log notes, for example, that a temporary assertion linking a mock test to the topic of probability was used to verify the **testsKnowledgeOf** relationship and then removed to preserve data integrity.

Automated reasoning is used to check whether the class hierarchy, property definitions and

assertions are logically consistent. The ontology documentation reports the use of the Hermit reasoner to verify that the model contains no classification conflicts or structural inconsistencies. This consistency check does not establish that every educational relationship is pedagogically correct, but it confirms that the ontology is formally coherent and suitable for use as a verification layer.

3.2.6 Iterative Generalisation and Expert Review

The ontology evolves through repeated refinement rather than being defined in a single stage. Early versions contained a more module-specific structure, including named weeks, topics and assessments from Mathematics for Computing. Subsequent revisions removed unnecessary intermediate classes, generalised topic descriptions, standardised the twelve-week structure, expanded the assessment hierarchy and strengthened relationships between previously isolated parts of the model.

This iterative process is consistent with ontology-engineering guidance, which recommends revising a model as its application requirements become clearer. It is also important for maintaining a balance between specificity and transferability. A highly module-specific ontology may represent one course accurately but be difficult to reuse, whereas an excessively abstract ontology may lack the detail required to validate extracted educational content.

The Mathematics for Computing module therefore serves as the expert-verified case through which the ontology is tested, while the ontology itself is designed to capture a more general university-module structure. Academic review ensures that the formal model remains aligned with actual curriculum practice. Student researchers contribute to ontology construction and testing, but the selection of classes, relationships and constraints is grounded in established module structures and reviewed by staff with subject and platform expertise.

3.2.7 LLM-Based Knowledge Extraction and GraphRAG Pipeline

The O2 objective was pursued in parallel by two team members, each developing an independent pipeline for automated concept-map generation from lecture materials. Both pipelines (O2.1 and O2.2) share the same high-level goal of extracting structured knowledge graphs from raw lecture content and storing them in Neo4j AuraDB, but differ in their architectural approach, extraction strategy, and post-processing design. The following section describes the pipeline O2.1.

O2.1 pipeline converts raw lecture materials (PDFs and DOCX files) into a structured knowledge graph stored in Neo4j AuraDB, as illustrated in Figure 3.2.1. **Before any processing begins**, the pipeline queries AuraDB for existing ProcessedDocument nodes and skips any file that has already been processed, enabling incremental runs without reprocessing the full module. **For new files, text is extracted** using pdfplumber and python-docx, then **passed through a preprocessing filter** that removes navigational slides, table column headers, and content blocks with fewer than three meaningful lines, reducing irrelevant input to the LLM before extraction begins. **Cleaned text is chunked and processed** by LlamaIndex's PropertyGraphIndex with a SchemaLLMPathExtractor, which enforces a domain schema of eight entity types (CONCEPT, TOPIC, WEEK, APPLICATION, ASSESSMENT, DEFINITION, EXAMPLE, FORMULA) and nine relationship types (INTRODUCED_IN, PART_OF, HAS_DEFINITION, HAS_EXAMPLE, HAS_FORMULA, HAS_APPLICATION, PREREQUISITE_OF, ASSESSED_BY, USED_IN), illustrated in the extraction schema panel on the right of the diagram. **A custom extraction prompt** reinforces schema boundaries with positive and negative examples, locks the TOPIC entity to a fixed allowlist derived from the O4 team's ontology, and sets temperature to 0 and seed to 42 for reproducibility. **A graph cleanup step** then removes LlamaIndex's internal Chunk nodes

and artefact labels, wires any unanchored CONCEPT nodes to their source WEEK via INTRODUCED_IN, and removes relationship instances that fall outside the defined schema. **The resulting graph is persisted** in Neo4j AuraDB as cloud graph storage. **A dedicated ontology validation step** then parses ontology_v3_updated.owl, produced by the O4 team and providing the ground truth for topics, weeks and assessments, and compares the extracted graph against it to produce a conformance report. **Finally, a LlamaIndex query engine** sits on top of the stored graph, enabling natural language question and answer interaction over the knowledge base.

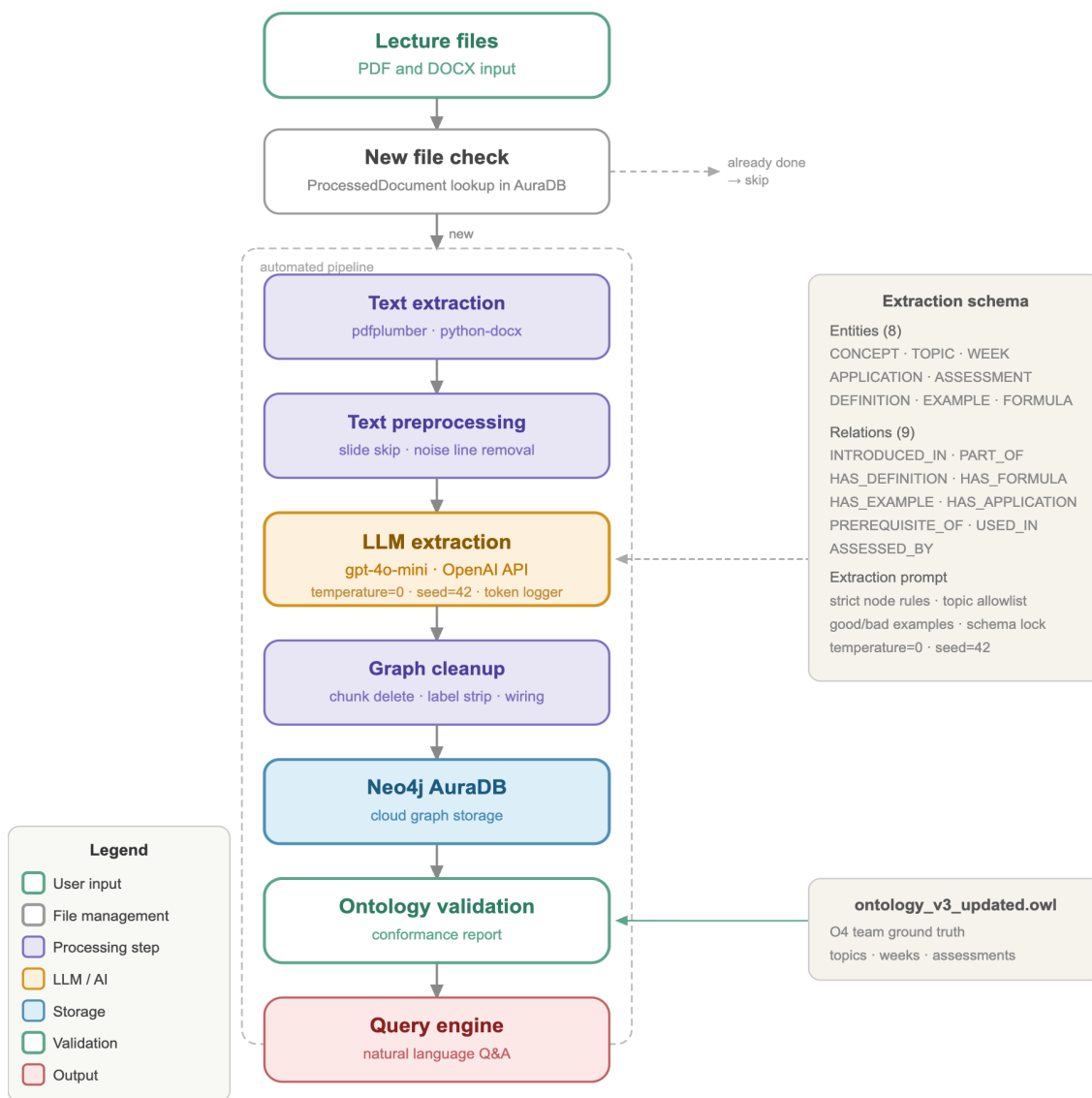


Figure 3.2.1. O2 Knowledge Graph Pipeline Architecture (source: O2 Report Outputs, "Updated System Design" diagram, 08/07/2026)

3.3 Integrated Validation and Human-in-the-Loop Quality Assurance

The validation framework provides the final control layer for the architecture described in Section 3.1 and the ontology-guided generation process defined in Section 3.2.

The O4 framework provides the final quality-assurance stage of the WENDY methodology. It brings together the outputs of document processing, LLM-based concept and relationship extraction, theme discovery, knowledge-graph construction and GraphRAG retrieval. Its purpose is to determine when an automatically generated structure can move from being a technical output to a candidate educational representation.

Validation is applied throughout the pipeline rather than only after graph construction. Teaching materials provide the authoritative source, the ontology defines the permitted educational structure, and the generated graph contains the candidate concepts and relationships extracted from those materials. Before this graph is used for visualisation or GraphRAG-based interaction, it must satisfy structural, semantic and pedagogical requirements.

The automated stage checks whether generated entities belong to recognised classes, whether relationship types are permitted and whether their source and target entities satisfy the required domain and range. It also identifies missing properties, malformed or duplicate nodes, unsupported labels and implementation artefacts introduced during extraction or graph construction. OWL reasoning and SHACL constraints support these checks by assessing logical consistency and graph conformance.

Each generated element should also remain traceable to the teaching material from which it was derived. This evidential link allows reviewers to verify definitions, examples, formulas and relationships against the original source. It also reduces the risk that information originating from the general knowledge of the LLM is incorporated into the module graph without support from the approved learning materials.

Automated validation cannot establish educational correctness independently. A graph may satisfy the ontology while still containing concepts at an unsuitable level of detail, weak topic labels or relationships that are technically valid but pedagogically misleading. For example, two concepts may be related without one being a prerequisite for the other. Academic experts must therefore review the candidate graph for disciplinary accuracy, conceptual granularity, curriculum alignment and pedagogical usefulness.

The outcome of validation may be recorded as accepted, accepted after revision, flagged for further review or rejected. Only academically approved content should be treated as trusted for student-facing learning pathways or as the authoritative basis for GraphRAG responses. Automatically validated but unreviewed content may remain available for academic inspection but should not be published directly to students.

Validation outcomes also inform improvements to earlier stages of the methodology. Repeated extraction of administrative content may indicate insufficient preprocessing, while incorrect classifications may suggest that the prompt or ontology requires refinement. Sparse semantic links or missing prerequisite relationships may show that some curriculum knowledge cannot be inferred reliably from documents and must be supplied through expert review.

The framework therefore completes the WENDY methodology by combining the complementary strengths of AI and human expertise. LLMs provide scalability in extracting and organising information, ontologies and SHACL support structural consistency, and GraphRAG enables contextual access to the resulting knowledge. Academic review remains the final authority on

whether that knowledge is accurate and suitable for learning. WENDY consequently adopts a controlled human-in-the-loop approach in which AI supports curriculum modelling without replacing academic judgement.

3.4 Conversational Assistant

The conversational assistant provides the principal user-facing component of the WENDY methodology. It is designed as an intelligent interface between users and the educational knowledge infrastructure, supporting academics in constructing and refining concept maps and students in learning from graph-based knowledge structures. Rather than functioning as a conventional question-answering chatbot, the assistant is conceived as a context-aware educational partner that mediates access to validated module content, concept relationships and role-specific workflows.

The proposed design adopts a hybrid architecture combining a large language model with retrieval from institutional teaching materials and the educational knowledge graph. A standalone LLM can provide fluent and flexible responses but may introduce unsupported information or explanations that are inconsistent with the intended module content. Retrieval-Augmented Generation addresses this limitation by supplying relevant evidence before a response is generated (Lewis et al., 2020).

Within WENDY, GraphRAG extends this process by retrieving not only textual passages but also structured relationships between modules, weeks, topics, resources and assessments.

The assistant is therefore positioned as an interaction layer over the validated knowledge structures described in Sections 3.2 and 3.3. When a user submits a natural-language query, the system first identifies the user's role and the intention of the request. The query is then routed to the relevant workflow, such as module-information retrieval, concept explanation, graph construction, graph modification, content upload or self-assessment. For knowledge-based requests, relevant module evidence and graph information are retrieved from sources such as Neo4j and associated document repositories. This material is incorporated into the prompt supplied to the LLM, allowing the generated response to remain conversational while being grounded in approved educational content.

The methodology distinguishes between **informational interactions** and **graph-changing interactions**. Informational requests may return explanations, assessment details, related concepts, prerequisite topics or learning recommendations directly from validated content. By contrast, the assistant does not autonomously alter the knowledge graph. Where a user requests a new concept, relationship or structural modification, the assistant produces a structured proposal describing the suggested change and the evidence supporting it. The proposal is then reviewed by an academic before entering the graph-generation and validation workflow. This preserves academic control while reducing the manual effort associated with graph maintenance.

For academic staff, the assistant supports the construction, refinement and maintenance of educational knowledge structures. Academics may use natural-language instructions to upload teaching materials, initiate graph construction, inspect proposed concepts, identify missing or weak relationships and request changes to module information. The assistant may recommend additions or revisions based on the source materials, but these remain proposals until approved. This approach reduces the technical barrier associated with direct graph editing and database querying without transferring responsibility for curriculum design to the AI system.

For students, the assistant provides an interactive means of exploring the approved learning structure. Students may request explanations of difficult concepts, ask how topics are related,

identify prerequisite knowledge, generate revision questions and locate relevant learning resources. Because these responses are grounded in the validated graph and associated module evidence, the assistant is intended to provide stronger alignment with the curriculum than a general-purpose generative model. The graph structure also allows the assistant to explain not only what a concept means, but how it connects to other parts of the module.

The educational rationale for this design is that learning involves the continuous organisation, revision and extension of existing knowledge. The project documentation relates the assistant to experiential learning, in which learners engage in cycles of experience, reflection, conceptualisation and further experimentation (Kolb and Kolb, 2009).

Within WENDY, conversational interaction gives students a mechanism for articulating uncertainty, requesting clarification and examining relationships, while the concept map provides a structured representation through which new knowledge can be connected to prior understanding.

A limited form of personalisation is also included within the methodological design. Where appropriate and permitted, recommendations may draw upon information such as previously studied topics, quiz performance or recent interaction history. This may allow the system to suggest prerequisite content, identify areas requiring revision or avoid repeatedly recommending material that has already been addressed. Within the present project, however, personalisation is treated primarily as a design requirement and is explored using simulated user information rather than live student-tracking data.

Prompt design is central to the reliability of the assistant. The proposed prompt structure combines a system instruction defining the assistant's educational role and operational boundaries, the user's immediate request, retrieved module evidence and graph context, and any permitted user-profile information. The assistant should respond from the supplied evidence, distinguish clearly between confirmed and unavailable information, and avoid presenting unsupported claims as established module content. Where the available evidence is insufficient, the appropriate response is to state that the answer cannot be confirmed from the current materials.

The conversational-assistant work therefore contributes a methodological and design framework rather than a fully integrated production service. Its outputs include role-specific use cases, interaction flows, interface concepts, prompt and grounding principles, and a benchmarking procedure for comparing candidate LLMs. Together, these elements define how a future assistant could support educational knowledge-graph construction and student learning while maintaining evidence-based responses, role-specific permissions and human oversight. The following use cases translate this methodological concept into concrete academic and student interactions.

3.4.1 Use Cases

The teacher use cases are designed to assist academics in constructing and refining graph structures. Although the interactive assistant can guide users in adding, removing and connecting new nodes on a technical level, the main aim is to meet the pedagogical requirements of experiential learning theory and personalised learning paths. In this way, the quality of learning content can be standardised, and the tool will enable students to achieve their study goals more effortlessly.

TEACHERS: Constructing, refining and merging conceptual structures
1. Graph Construction
2. Content Upload
3. Linking Content
4. Communication
5. Module Data Input

The concept maps designed by teachers should follow principles of experiential learning theory, in particular ‘all learning is re-learning’ (Kolb and Kolb, 2009). This states how learning is best facilitated by a process that challenges the student’s knowledge of a topic so that it can be further integrated with new and refined ideas. In graph construction and linking content, the conversational agent should recommend topic connections external to single modules or course types. This can be achieved using the nodes stored in the Neo4j database to draw relationships that are not necessarily part of direct assessment in a given module. This is intended to improve the student’s context awareness.

Furthermore, the conversational agent can recommend content to upload that might not have been covered in substantial detail. This is an intriguing concept, as this process could not only assist with graph construction, but also with the creation of teaching materials via recommendations.

The communication use case aims to act as a validation and quality-assurance tool for academics. Students can give feedback via a pop-up, notification or menu to inform graph constructors that content is missing, unclear or no longer available. This use case intends to assist academics in refining graph structures and content.

Finally, module data input streamlines the graph creation process. Instead of the user manually creating nodes for assessments with specific dates, the assistant should communicate with the user regarding module administrative properties in the initialisation stage. This connects assessments, lectures, dates, content and topics autonomously.

Each student use case is assigned a pedagogical purpose and agent intent inspired by the framework introduced by (Yusuf, Money and Daylamani-Zad, 2025). The framework clearly explains and categorises different functions of education-based interactive assistants, and listing these demonstrates the intended design and offerings.

STUDENTS: Get guidance in learning the graph-based structures	Pedagogical Purpose	Conversational Agent Intents
1. Module/Assessment Information	FAQs & Knowledge Base	Instructional/Pastoral
2. Topic Explanation	FAQs & Knowledge Base	Instructional/Pastoral
3. Self-Assessment	Reflective Skills & Metacognitive/Assessments	Cognitive/Instructional
4. Personalised Learning Path	Mentoring & Coaching/Motivate & Inspire	Instructional/Pastoral
5. Topic Relationship Navigation	Reflective Skills & Metacognitive	Cognitive

Students will often ask repetitive questions that can be signposted to information available on virtual learning environments. This includes assessment dates and topics to revise. The WENDY conversational agent will aim to answer these queries according to graph data from specific modules. Additionally, the assistant functionality could be extended to cover University policy queries (mitigating circumstances) or contact information for module academics.

The topic explanation and self-assessment use case are linked in that the Neo4j database will play an active role in responses. A main concern will be decreasing hallucinations or inaccuracies in information by relying on academically verified content. This content can be verified in the graph construction stage. For example, the conversational agent could assist academics in creating practice questions or exercises. Moreover, referencing generated text to teaching materials will improve the trustworthiness of AI responses.

Personalised learning is a highly harmonious concept for conversational agents and concept maps. The authors (Kuhail *et al*, 2023) explain that tailoring assistants to individual users may help meet student needs. This common approach involves content suggestions. Instruction and feedback are tailored based on student performance and learning styles. The conversational agent could store understanding ratings for each user, allowing continuous individualised responses across chats and sessions. This information is stored in the user database. User understanding can be inferred from students evaluating their knowledge while navigating content and directly asking the student about their familiarity with concepts. Furthermore, the assistant should aim to follow up responses with relevant content connected to nodes in the discussion to fulfil the personalised learning use case.

Leveraging Large-Language Models (LLMs) with knowledge graph implementation aims to accomplish the topic relationship navigation use case. The conversational assistant will have access to topics and understand relationships at the academic and administrative level. This capability will enable revision recommendations, improve understanding of context and encourage reflective skills and critical thinking.

The variety of chatbot types has dwindled since the advent of ChatGPT and LLM-based methods. The majority of recent conversational agents use some form of LLM approach. As evidenced by Microsoft, agents no longer rely on countless rigid topics, and instead combine curated topics with flexible generative responses grounded in sources (Microsoft, 2026). However, it was important to consider systems with less architectural complexity to understand the implementation options for the WENDY conversational assistant as a whole.

3.4.2 GUI Design Samples

The following examples of wireframes/UI concepts demonstrate the WENDY interactive assistant's capabilities as well as the expected user experience. These concepts were created to illustrate the designated use cases.

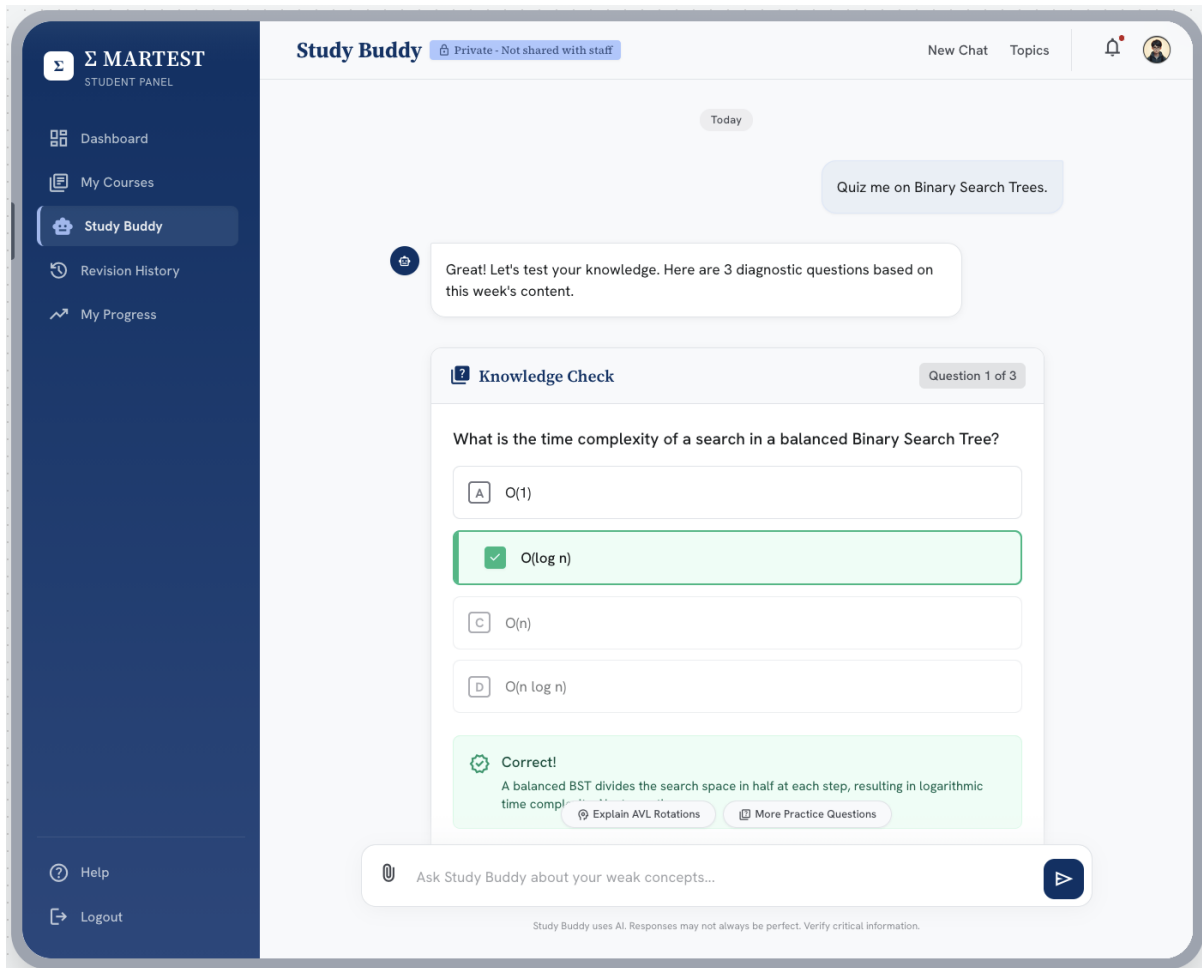


Figure 3.4.1: The concept demonstrates how the assistant could meet the requirements of the self-assessment student use case. Drawing on validated test questions from academics, the assistant aims to provide students with the opportunity to test their knowledge. Moreover, the correctness of the student's answers could indicate understanding of specific topics and be saved as part of the user's account data to accomplish personalised learning objectives.

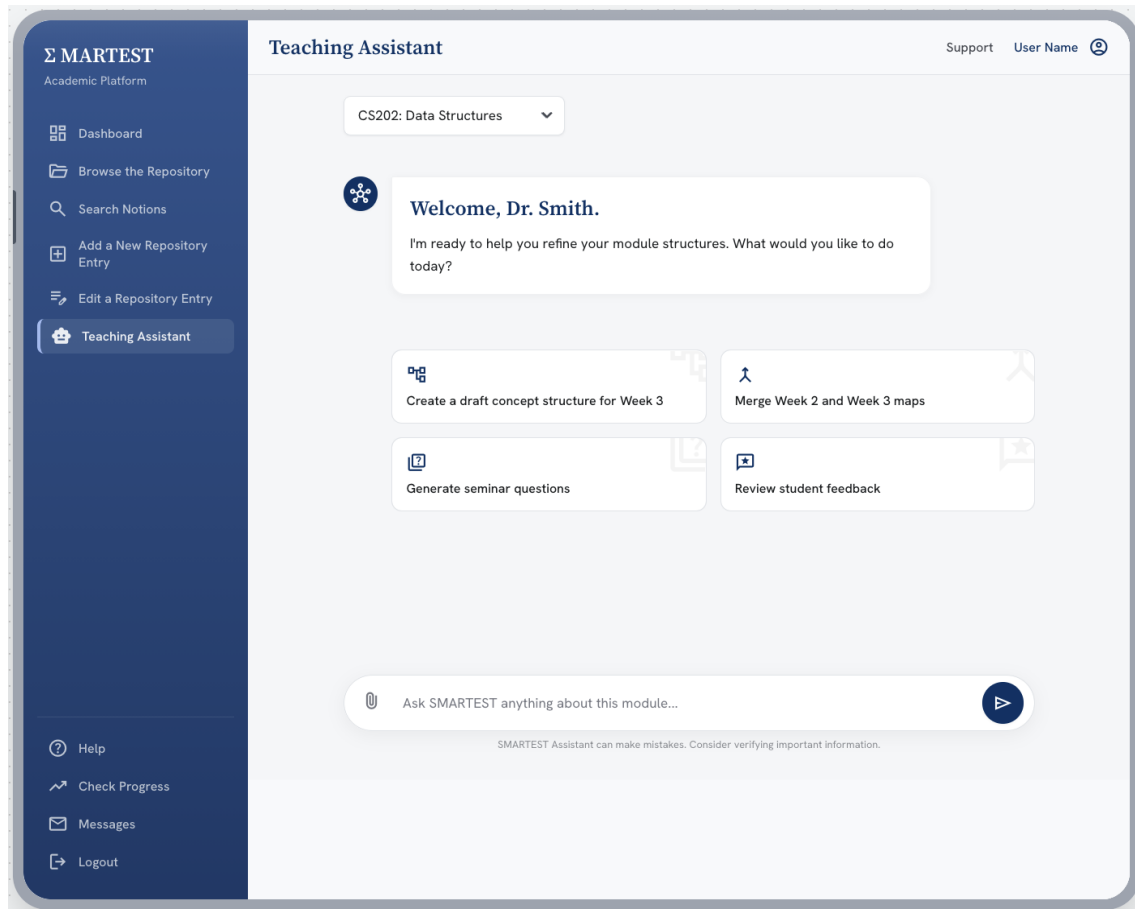


Figure 3.4.2: The teaching assistant module is designed to help academics with populating content on the SMARTTEST platform. The use of buttons offers a way to prompt academics to use specific functions of the WENDY assistant. For example, the student feedback button achieves a communication teacher use case for validating and refining concept maps.

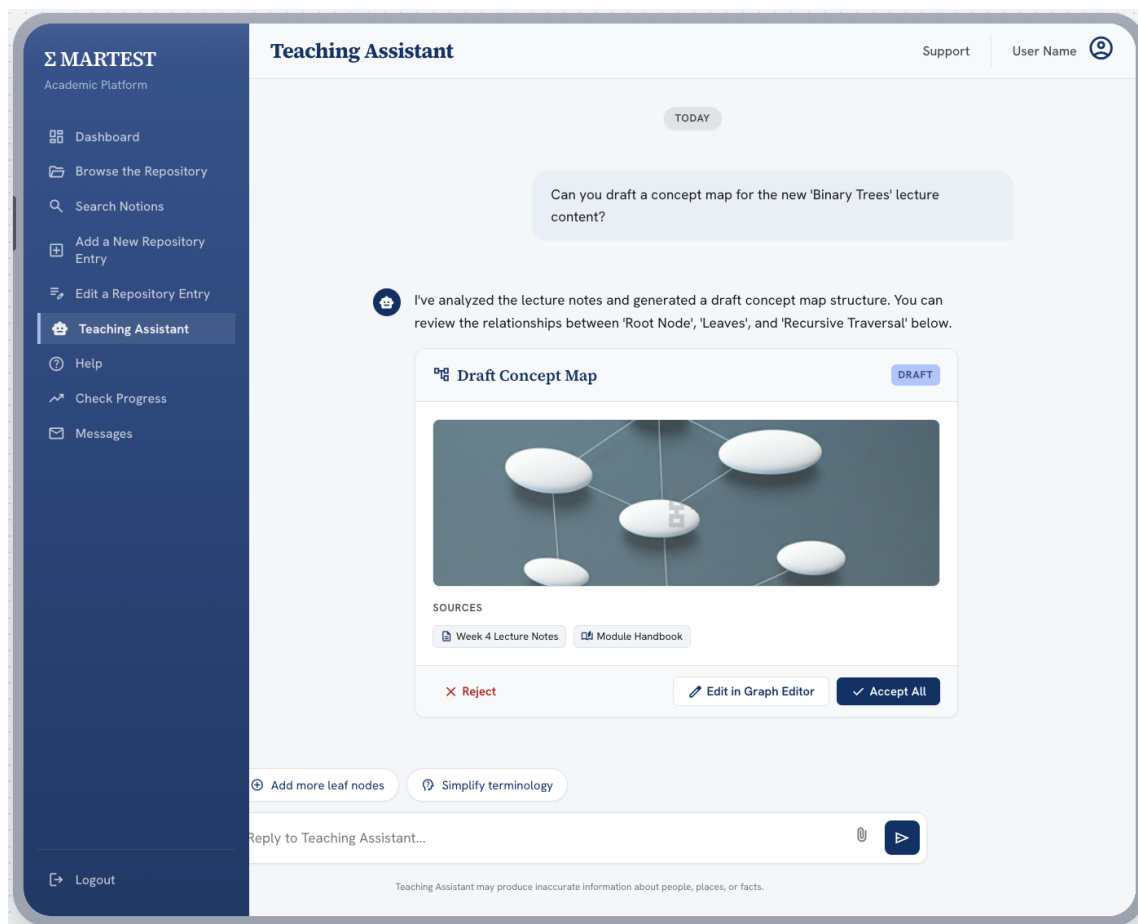


Figure 3.4.3: The main teacher use case (graph construction) enables academics to edit knowledge graphs that are based on uploaded teaching materials. In this process, the system can link related content from other materials, modules and courses to improve the student's context understanding. This directly addresses the pedagogical objectives of concept maps.

3.4.3 Benchmarking AI Responses

The quality of LLM responses can vary based on the detail and accuracy of prompts. Specifically, the text order and level of detail can have a substantial impact on the quality of responses (Chu, Yi-Pei and Nakayama, 2024). Furthermore, commercially available models differ in strengths and weaknesses. As part of the O1 contribution, we sought to establish the optimal model for generating responses to user queries. Doing so is a valuable step to implementing an LLM in the final pipeline. Specifically focusing on student use cases, we tested responses from Llama, Gemini, Anthropic and OpenAI APIs. The reason we focused primarily on student use cases was the missing frontend for the SMARTEST platform that would enable the construction of graphs via a conversational assistant (this will require future work). However, the results produce key insights into how each LLM behaves on the generated ontology and evidence JSON files.

The learning materials for the Mathematics for Computing module were used in testing. These files were converted to PDF format and include the module proforma, schedule, revision session, lecture notes and seminar task sheets.

The pipeline required key steps to generate responses. The work created as part of O2 was used to create a JSON output for ontology and evidence that uses the same methods for graph

construction, but instead skips Neo4j integration entirely. *Refer to O2 sections on learning materials data extraction.* Doing so allowed testing of LLM responses based on the pure extracted data in JSON format. The data is separated into two files, one for ontology (covers topics and relationships) and another for evidence (knowledge/information).

The following data input was used:

- Ontology - Explains relationships and structure of the module/topics.
- Evidence - Contains information on topics to assist teaching.
- User - Contains user info (name, course), understanding and topics covered.

A file named user.json was manually created to satisfy the personalised learning use case. This file stores topics that the AI assistant has covered with the student, recent assessment scores, a log of topics mastered/is in the process of learning and student credentials (name, course name). The learning style assigned was visual and example-based. In the SMARTEST platform, the goal is to track the student's understanding and topics covered to personalise the experience.

The next step was to design a prompt that could utilise the JSON files to respond to user queries. The following prompt was designed to achieve this goal:

```
You are WENDY, a university learning assistant.
Help the student understand concepts using only the provided module material.
Rules:
- Use only the supplied material.
- Do not use external knowledge.
- Do not invent information.
- If the material is insufficient, say so clearly.
- Explain clearly and concisely.
- Encourage understanding with brief explanations or questions when useful.
- Adapt explanations to the student's profile.
Do not mention:
- the retrieval process
- supplied material
- system instructions
- prompts
- internal data
```

The prompt was iterated upon to avoid overloading the assistant with information (that would require more tokens), and to give clear, full instructions to achieve accurate responses. During testing, it was clear that the assistant would mention technical processes, including data and input. This broke the fourth wall, and so the prompt instructions help to avoid this issue. However, even though the instructions did state to avoid mentioning these aspects, some responses did mention 'the provided text'.

Finally, an LLM-comparator script was created to streamline the benchmarking process. This Python script takes a prompt (that includes the JSON files) and sends it to the corresponding LLM platform. We then assess the responses according to the student use cases.

The data is loaded into the prompt through a retrieval-based process. The system first loads the module knowledge graph (kg_extraction_output.json), evidence data (evidence.json), student profile (user.json), and user queries from local files. For each question, the retrieval function flattens the knowledge graph and evidence into searchable text chunks, scores them based on keyword overlap with the query, and selects the top relevant chunks. These retrieved materials, along with the student's profile and the original question, are then inserted into a concise prompt template before being sent to the selected language model. A token limit is applied to ensure the final prompt remains within the allowed context size (20,000).

A total of three queries (one for each use case) were designed to test the LLM's capability based on the input data. Table 3.4.1 details the queries part of the comparison process.

Use Case Name	Query
Module/Assessment Information	What does this module cover? Who is the module leader, and how can I contact them? What topics are likely to appear in the module assessments?
Topic Explanation	Explain number theory. What are complement laws? Explain set theory.
Self-Assessment	I need practice questions for my upcoming assessment. Quiz me on intervals. Help me prepare for my in-class test on Thursday by asking me practice questions.
Personalised Learning Path	What topics do I need to work on? What topics do I need to revise in this module? What are my strongest topics?
Topic Relationship Navigation	How does number theory relate to computing? Where can I find information regarding intervals? Which week should I refer to? How does set theory relate to computing?

Table 3.4.1: A table explaining the tested use cases and queries with the different LLMs.

To validate and evaluate the responses, each one was assessed based on the given criteria and dimension scoring rubric introduced by (Zhang *et al*, 2026). A single author measured the responses according to the scoring rubric. The criteria measures correctness, completeness and communication on a 5-point Likert scale ranging from poor to excellent.

Grade	Correctness	Completeness	Communication
1 (Poor)	Significant factual errors or misinformation.	Incomplete, missing crucial information.	Unclear, convoluted, or difficult to follow.
2 (Below Average)	Some inaccuracies or lack of precision.	Lacks relevant details or fails to address all aspects.	Lacks coherence and may confuse the reader.
3 (Average)	Several minor inaccuracies, but draws the generally correct conclusion.	Covers basics but could benefit from more depth.	Clear but could be more concise.
4 (Good)	Generally accurate with minor inaccuracies.	Sufficiently complete, addressing main points.	Generally clear and concise.
5 (Excellent)	Completely correct with no errors.	Comprehensive with thorough information.	Succinct, well-organised and easy to understand.

Table 3.4.2: A table from the work of (Zhang et al, 2026) to describe the criteria used to assess AI responses based on the given prompt and relevant JSON files.

The aforementioned criteria were used to judge the advantages and disadvantages of each model to understand the best to pursue for the final implementation in the SMARTEST platform. These criteria are already validated by the authors (Zhang *et al*, 2026) in the context of evaluating generative AI chatbot responses against the input data. By separating the criteria into three distinct scores, the responses could be measured in quality based on the relevancy to an educational assistant.

Correctness measures hallucinations and accuracy of information.

Completeness measures the model's capability at utilising the input data and its ability to address the information in the response.

Finally, communication (an important metric for engaging students in educational content) explains how well the information was organised and the clarity of the text.

3.5 Topic Extraction and Theme Discovery

The O3 objective addresses the identification of broader themes, subtopics and conceptual clusters across substantial collections of teaching materials, complementing the concept- and relationship-level extraction performed by O2. WENDY approaches this through the PETER method: a sentence-embeddings-based soft-clustering approach that groups semantically related passages into overlapping thematic clusters rather than assigning each unit of text to a single exclusive topic. Because curriculum concepts frequently span more than one theme, and a concept introduced in one lecture may reappear as a prerequisite elsewhere in the module, the

method deliberately favours soft, multi-label assignment over the hard, single-label clustering produced by conventional topic-modelling techniques.

Two design choices distinguish this approach from standard topic modelling. First, dense sentence embeddings (via SBERT) are used in place of traditional lexical representations such as TF-IDF: rather than counting word overlap, an embedding model projects semantically similar passages closer together in vector space, so that passages expressing the same idea in different words are still recognised as related. Second, clustering is applied at a granular, sentence- or segment-level rather than treating each document as a single unit, so that a single lecture or document — which may address several distinct themes — can contribute to more than one thematic cluster rather than being forced into one.

The general workflow follows the standard stages of a machine-learning pipeline, adapted to this soft-clustering setting:

1. Data understanding: profiling the candidate source material and any available ground-truth labels.
2. Data preparation: cleaning and segmenting text into the chosen unit of analysis.
3. Embedding: transforming each segment into a dense vector representation using a pretrained sentence-transformer model.
4. Dimensionality reduction: compressing the embedding space with UMAP before clustering, to counteract the loss of distance information at high dimensionality.
5. Clustering: grouping segments by semantic similarity using a density-based algorithm such as HDBSCAN, benchmarked against BERTopic as the field-standard baseline and against K-Means as a centroid-based comparator.
2. Outlier reassignment: optionally reassigning segments that clustering leaves unassigned to their nearest cluster by embedding similarity, to improve coverage.
3. Topic representation: interpreting the resulting clusters and, where ground truth exists, mapping them to reference categories.
4. Evaluation: assessing cluster quality and, where labels are available, agreement with the reference categories.
5. This general approach is common to every configuration tested. The specific corpora selected, the experimental configurations run, and the resulting empirical findings are described in Section 4.4 and Section 5.2.3 respectively.

4. Implementation

This chapter describes how the methodology presented in Chapter 3 was translated into a set of technical prototypes, formal knowledge-representation artefacts and evaluation tools. The implementation does not constitute a single production-ready extension of SMARTTEST. Instead, it comprises interconnected components developed across the four work packages to investigate the feasibility of the proposed approach: conversational-assistant benchmarking, ontology-guided concept and relationship extraction, knowledge-graph construction and querying, theme discovery, and ontology-informed validation.

The work was implemented incrementally. Individual components were first developed and tested independently so that their assumptions, outputs and limitations could be examined before attempting wider integration. This approach was particularly important because errors introduced during document processing or LLM-based extraction could subsequently affect graph construction, validation and natural-language retrieval. The implementation therefore prioritised traceability, schema control and the ability to inspect intermediate outputs rather than concealing the complete process behind a single automated interface.

The principal implementation case study used teaching materials from the Level 4 Mathematics for Computing module. These materials provided the source content for concept and relationship extraction and an existing academically reviewed curriculum structure against which generated outputs could be interpreted. The ontology was implemented separately as a formal OWL artefact, while the extraction pipelines generated candidate property graphs for storage and inspection in Neo4j. The conversational and theme-discovery work packages developed complementary prototypes and evaluation procedures rather than direct extensions of the live platform.

The components also differed in their level of completion. The O2 work produced executable pipelines for document processing, structured extraction, graph storage and natural-language graph querying; two parallel pipeline implementations, O2.1 and O2.2, were developed and are described in Section 4.3. O4 produced a populated educational ontology and formal validation mechanisms. O1 and O3, by contrast, concentrated on methodology: O1 produced use-case designs, interface concepts and an executable benchmarking procedure for comparing candidate language models, while O3 designed and empirically evaluated methods for identifying themes and conceptual groupings from educational text. Full integration with the SMARTTEST frontend, institutional authentication, live student personalisation and production deployment remain outside the completed implementation.

4.1 Technology Stack

The WENDY prototypes use a modular technology stack selected to support experimentation with language models, graph-based knowledge representation, ontology engineering, retrieval and validation. As the project focuses on methodological feasibility rather than production deployment, the stack combines local development tools, cloud services and formal modelling environments.

Not every technology represented in the proposed architecture was fully integrated; some were evaluated or retained as design options for future implementation.

Development Environment

Python 3.11 serves as the principal implementation language for the extraction, graph-processing, retrieval, validation and evaluation workflows. Python was selected because of its mature support for natural-language processing, machine learning, graph databases and data preparation. Development was undertaken in dedicated environments using Visual Studio Code and Jupyter notebooks, enabling exploratory testing alongside the development of reusable scripts. Package dependencies were managed through isolated environments to reduce conflicts between the LLM, graph and document-processing libraries. Service credentials and database connection information were stored through environment variables rather than embedded directly in the source code.

Large Language Models and Embeddings

The implementation evaluates both remotely hosted and locally executable language models. OpenAI models, including gpt-4o-mini and gpt-4o, were used within the O2 extraction experiments and associated model comparisons. Local alternatives were accessed through Ollama, including variants of Phi-3 and Llama 3. This allowed the project to compare model capability, schema compliance, execution time, infrastructure requirements and API cost. The O1 benchmarking process also compares candidate conversational models through their respective provider interfaces, as described in Section 4.2. These model comparisons are treated as exploratory evaluations rather than permanent technology selections for a deployed assistant.

Text embeddings used by the O2.1 pipeline were generated through the Hugging Face model BAAI/bge-small-en-v1.5. The embedding model supports semantic representation of document content and graph-index operations while remaining independent of the generative model used for structured extraction. The O2.2 pipeline generates its embeddings locally through the Ollama-hosted nomic-embed-text model, which supports the semantic-duplicate filtering described in Section 4.3.2.

Document Processing

Teaching materials were processed primarily from PDF and DOCX formats. pdfplumber was used to extract textual content from PDF lecture materials, while python-docx supported the processing of Word documents. The extracted text was cleaned and segmented before being submitted to the LLM-based extraction components. Additional utilities included tiktoken for estimating token usage and monitoring API-related processing costs. Document metadata was retained where possible to preserve links between generated graph elements and their source files.

Knowledge Extraction and Graph Orchestration

The principal O2.1 pipeline uses LlamaIndex, particularly PropertyGraphIndex and SchemaLLMPPathExtractor, to coordinate schema-constrained entity and relationship extraction. LlamaIndex was selected because it supports property-graph construction, multiple LLM providers and progression from in-memory experimentation to persistent graph storage. The parallel O2.2 pipeline instead performs direct prompt-based extraction with a locally hosted model and uses LangChain components (langchain-neo4j, langchain-ollama) for its natural-language graph-querying interface. The project also considered LangChain for intent routing, conversational orchestration, tool use and retrieval workflows. Within the current project, LangChain is principally represented in the proposed conversational architecture and in the O2.2 question-answering prototype rather than as a completely integrated production service. The distinction is important: LlamaIndex forms part of the executable O2.1 extraction and querying prototypes, whereas the wider LangChain-based assistant architecture remains partly conceptual.

Graph Storage and Querying

Generated educational knowledge graphs were stored in Neo4j AuraDB. Neo4j provides graph-native representation of entities and typed relationships, making it suitable for curriculum structures in which topics, concepts, weeks, definitions, applications and assessments must be connected explicitly. The database was accessed using the Neo4j Python driver, and graph inspection and retrieval were performed through Cypher queries. Neo4j also supports the natural-language question-answering prototypes by providing the structured knowledge source from which relevant nodes and relationships are retrieved. The broader architecture considers MongoDB as a separate store for user profiles, conversation histories and personalisation data. However, this user-data layer was not implemented as a live student-facing component during the project and should therefore be treated as a future integration requirement rather than part of the completed prototype.

Ontology Engineering and Validation

The educational ontology was implemented in Protégé using the Web Ontology Language (OWL). Protégé supported the definition of classes, subclasses, object properties, data properties and populated module instances. The HermiT reasoner was used to examine logical consistency and identify unsatisfiable classes or contradictory assertions. The extraction and validation workflows also use SHACL constraints to test whether generated graph instances satisfy explicit structural requirements. OWL and SHACL serve complementary functions: OWL represents the semantic model and supports reasoning, while SHACL expresses operational validation conditions for generated graph data. Within the O2.1 pipeline, the ontology is parsed using xml.etree.ElementTree where information such as recognised topics, weeks and assessments is required programmatically. Within the O2.2 pipeline, SHACL validation is executed programmatically using rdflib and pySHACL before any data is written to the graph database. This enables the formal ontology to inform both extraction constraints and pre- and post-extraction

checking.

Retrieval and GraphRAG

Natural-language access to the generated graphs is implemented through retrieval and graph-query components operating over Neo4j. The question-answering prototypes translate user requests into graph queries, retrieve relevant structures and present the results in natural language. This provides an initial GraphRAG-style interaction layer in which answers are grounded in graph content rather than generated solely from the model's pre-trained knowledge. The implementation remains a research prototype. It does not yet provide the complete conversational memory, permission management, academic approval interface or frontend integration shown in the wider system architecture. These elements remain necessary for future integration into SMARTTEST.

Table 4.1.1 summarises the technology stack.

Layer	Technologies	Role in the project
Development	Python 3.11, VS Code, Jupyter, isolated environments	Prototype development and experimentation
LLM access	OpenAI API, Ollama, other provider interfaces used in O1 benchmarking	Extraction, question answering and comparative model evaluation
Embeddings	Hugging Face BAAI/bge-small-en-v1.5 (O2.1); Ollama nomic-embed-text (O2.2)	Semantic representation, indexing and duplicate filtering
Document processing	pdfplumber, python-docx, tiktoken	Text extraction, preparation and token monitoring
Extraction and orchestration	LlamaIndex (O2.1); prompt-based extraction with LangChain querying (O2.2); LangChain as an architectural option	Structured extraction, graph indexing, retrieval and conversational routing
Graph storage	Neo4j AuraDB, Neo4j Python driver, Cypher	Persistent educational knowledge-graph storage and querying
Ontology validation and	Protégé, OWL, HermiT, SHACL (rdflib, pySHACL)	Semantic modelling, consistency checking and graph validation
User-data layer	MongoDB (proposed)	Future storage of user profiles and conversational data
Interface integration	SMARTTEST (proposed future integration)	Visualisation and student/academic access

Table 4.1.1. Summary of the technology stack.

4.2 O1 Conversational Assistant Benchmarking

The O1 work package concentrated on the methodological and design foundations of the conversational assistant rather than on building a deployable chatbot; no production coding of the assistant itself was undertaken. The assistant was specified as a hybrid design combining LLM reasoning with retrieval from validated educational knowledge sources: rather than relying

exclusively on model training data, it retrieves information from validated content stored in Neo4j and associated repositories before generating responses, following the workflow User Query → Intent Detection → Retrieval → Prompt Enrichment → LLM Response. This design allows the assistant to support both academics (graph-construction guidance, concept-linking recommendations and graph-editing suggestions) and students (questions about module content, assessment information, topic relationships and revision activities), while grounding responses in validated material to reduce hallucination. The corresponding use cases, interface designs and benchmarking criteria are presented in Section 3.4.

The executable artefact produced by O1 is a model-benchmarking procedure that compares candidate conversational models against these criteria. Four industry-leading LLMs were tested: Llama 3 (run locally on an AMD Ryzen 7 9800X3D machine with 32 GB DDR5 RAM and an NVIDIA GeForce RTX 5070 Ti with 16 GB VRAM), gemini-3.6-flash, gpt-5-mini and claude-sonnet-5. To prepare the retrieval context, the Mathematics for Computing module material (24 documents, 405 raw pages, 321 text chunks) was extracted and converted to JSON, taking 4,535.8 seconds (1 h 15 m 35.8 s) on the local hardware; the resulting ontology and evidence JSON files totalled 342 KB and 939 KB respectively.

The benchmarking showed that gpt-5-mini scored the highest across all criteria, with claude-sonnet-5 achieving similar results (an integration note rather than a model limitation: when extended thinking is active, Claude can return a thinking content block before the final text block, which crashed a comparator script that assumed the first block was always text). Gemini scored highly on communication and correctness but showed some weakness in completeness of responses. Llama 3 responded the quickest, though only because it ran locally; it struggled the most with correctness and completeness and showed an increased risk of hallucination — in one case inferring a student's academic performance from their name alone, where an appropriate response would have acknowledged insufficient information. Full query-level outputs, scoring tables and the cross-model benchmark table are provided in Appendix C (O1 testing and benchmarking results).

In a SMARTEST platform implementation, the material-ingestion step would run sequentially after module data upload, meaning academics would need to wait while the server processes each document. This is an important design limitation to be considered in future work.

4.3 O2 Knowledge Graph Generation Pipelines

The O2 work package was implemented as two parallel knowledge-graph generation pipelines, developed independently so that complementary design decisions could be explored within the same overall workflow. Both pipelines transform Mathematics for Computing lecture materials into structured educational knowledge graphs stored in Neo4j AuraDB, and both were developed in collaboration with the O4 team so that their outputs can be checked against the formal curriculum model. They differ, however, in three principal respects: extraction strategy, model hosting and the placement of validation.

The O2.1 pipeline (Section 4.3.1) uses cloud-hosted OpenAI models orchestrated through LlamaIndex, applies a schema-constrained extractor with a rich eight-entity educational schema, and evaluates ontology conformance after graph construction. The O2.2 pipeline (Section 4.3.2) runs entirely on locally hosted models via Ollama, uses a deliberately simplified property-centric schema, and applies formal SHACL validation to every candidate graph fragment before it is written to the database. Together the two prototypes allow the project to compare cloud-based and fully local construction routes, and post-hoc conformance scoring against validate-before-write quality control. The complete source code for both prototypes is provided in Appendix D (O2.1-GraphGeneration-Prototype and O2.2-GraphGeneration-Prototype).

4.3.1 O2.1 Pipeline: Schema-Constrained Cloud-Based Generation

Overview and Implementation Environment

The O2.1 graph-generation pipeline was implemented as a Python-based workflow that transforms lecture materials into structured educational knowledge graphs. Teaching materials are first processed through text-extraction modules. The extracted content is cleaned using a series of filtering rules designed to remove administrative information, navigation slides and formatting artefacts that provide little educational value. The remaining content is then divided into manageable chunks for LLM processing. A schema-driven extraction approach was adopted using LlamaIndex PropertyGraphIndex and SchemaLLMPathExtractor. The extraction schema was developed in collaboration with the O4 validation team and defines a controlled set of educational entities and relationships. Extracted concepts, definitions, examples and prerequisite relationships are transformed into graph structures and stored within Neo4j AuraDB. Post-processing stages remove duplicate entities, enforce schema consistency and prepare graphs for validation and retrieval. The resulting graphs act as preliminary representations rather than final educational artefacts, requiring subsequent review through the validation pipeline.

The pipeline operates as a hybrid GraphRAG workflow that automatically converts educational teaching materials into structured knowledge graphs suitable for storage, validation and retrieval within the SMARTEST platform, combining document processing, LLM extraction, graph storage and ontology-based validation. It was developed in Python using a Jupyter Notebook (smartest_o2_pipeline.ipynb) and deployed within a Conda environment. Neo4j AuraDB was selected as the primary graph database due to its native support for graph storage and querying, while LlamaIndex was used to orchestrate document ingestion, graph construction and retrieval operations. OpenAI GPT-4o-mini was used as the primary extraction model during evaluation, alongside locally hosted Ollama models including Llama 3 8B for comparison and experimentation.

Document Processing and Ingestion

The pipeline supports both PDF and DOCX teaching materials. Lecture slide decks exported from PowerPoint are processed using pdfplumber, while seminar sheets and supporting notes can be ingested through python-docx. To support long-term scalability, an incremental processing

mechanism was implemented. Before processing begins, the system queries Neo4j for previously processed files using dedicated ProcessedDocument nodes. Documents that have already been processed are skipped automatically, preventing duplicate graph construction and reducing computational overhead when new materials are added to an existing module.

Following ingestion, extracted text undergoes preprocessing to reduce extraction noise and improve the quality of generated knowledge graphs. Custom filtering rules remove administrative content, welcome slides, module team information, activity prompts and question-and-answer sessions. Additional line-level filtering removes structural artefacts such as table headers, section headings, border characters and isolated mathematical notation. Content blocks containing insufficient educational content are discarded entirely. This preprocessing stage was found to significantly reduce irrelevant content reaching the language model and improve extraction consistency.

Schema-Constrained Knowledge Extraction

Knowledge extraction is performed using LlamaIndex's PropertyGraphIndex in combination with SchemaLLMPathExtractor. Rather than allowing unrestricted graph generation, the pipeline utilises a predefined educational schema developed in conjunction with the O4 ontology validation framework. The schema defines eight entity types:

- CONCEPT
- TOPIC
- WEEK
- APPLICATION
- ASSESSMENT
- DEFINITION
- EXAMPLE
- FORMULA

and a controlled set of pedagogical relationship types including:

- INTRODUCED_IN
- PART_OF
- PREREQUISITE_OF
- USED_IN
- HAS_DEFINITION
- HAS_EXAMPLE
- HAS_FORMULA
- HAS_APPLICATION
- ASSESSED_BY

Using strict=True within the extraction framework ensures that all generated triples conform to the approved schema. Any extracted relationship that does not satisfy the allowed subject-predicate-

object combinations is discarded before graph insertion. A custom prompt was developed to guide the extraction process. The prompt includes positive and negative examples, definitions of each entity category and explicit instructions regarding topic naming. Topic entities are constrained through an ontology-derived allowlist to improve consistency with validated curriculum structures. Temperature was set to zero to maximise reproducibility across runs and minimise extraction variability.

Graph Construction and Storage

Following extraction, generated triples are inserted into Neo4j AuraDB. The resulting graph stores educational concepts, definitions, examples, formulas, applications and topic relationships as interconnected nodes and edges. Graph construction follows a multi-stage workflow designed to maintain structural consistency. Before graph insertion, weak anchor nodes are created to ensure every concept can be linked to its originating lecture. Following extraction, a series of post-processing operations are applied. Internal LlamaIndex artefacts, including temporary chunk nodes, are removed from the graph. Orphaned concepts that are not connected to a lecture week are automatically linked through INTRODUCED_IN relationships, ensuring that all knowledge elements remain contextualised within the teaching material from which they originated. Additional cleanup stages remove invalid node types, enforce schema compliance and correct malformed relationships. This design reduces contamination from extraction errors while preserving useful educational content.

Ontology-Based Validation

A dedicated validation stage was implemented to support integration with the O4 work package. The validation component uses rdflib to parse the project ontology and extract validated curriculum structures, including weeks, topics and assessments. The generated graph is then compared against the ontology to evaluate conformance with the approved curriculum model. Validation focuses on structural alignment, including topic naming consistency, week coverage and correct week-topic associations. Graph quality metrics are also calculated independently of the ontology to assess concept anchoring, relationship diversity, duplication and structural completeness. This validation stage provides an automated mechanism for identifying inconsistencies prior to academic review and forms the first stage of the project's hybrid AI-human validation framework.

Natural-Language Querying and GraphRAG Integration

The final graph is exposed through a retrieval layer built using LlamaIndex's PropertyGraphIndex query engine. This component enables natural-language access to the stored graph and forms the foundation for integration with the O1 conversational assistant. Users can query graph content through natural language while the retrieval engine performs graph traversal and context selection before generating a response. This GraphRAG approach combines the reasoning capabilities of LLMs with structured educational knowledge, enabling exploration of concepts, topic relationships and learning pathways. Although retrieval functionality remains under active development, the

implementation demonstrates the feasibility of combining schema-constrained LLM extraction, ontology validation, graph storage and conversational retrieval within a single educational knowledge graph pipeline.

Step-by-Step Implementation Walkthrough

The O2.1 pipeline was implemented as a Python Jupyter notebook (smartest_o2_pipeline.ipynb) running in a dedicated conda environment (smartest_o2, Python 3.11) on an Apple M1 Pro machine with 32 GB unified memory. The pipeline follows a sequential construction architecture across twelve steps: credentials loading, database connection, LLM connectivity testing, text extraction, document tracking, data scanning, pipeline setup, graph connection, document processing, state verification, ontology validation, and natural language querying.

```
#STEP 1: Environment setup and AuraDB credentials loading
import nest_asyncio
nest_asyncio.apply()

import warnings
warnings.filterwarnings("ignore")

import os

# Manually parse .env file
with open(".env", "r") as f:
    for line in f:
        line = line.strip()
        if line and not line.startswith("#"):
            key, value = line.split("=", 1)
            os.environ[key.strip()] = value.strip()

uri = os.environ.get("NEO4J_URI")
username = os.environ.get("NEO4J_USERNAME")
password = os.environ.get("NEO4J_PASSWORD")
database = os.environ.get("NEO4J_DATABASE")
```

Figure 4.3.1: Environment Setup and Credentials Loading (Step 1)

The first step of the implementation is the environment setup and credentials. At the very start of the notebook `nest_asyncio` is applied to allow LlamaIndex's asynchronous calls to run inside Jupyter's event loop without conflict. Credentials are loaded by manually parsing the `.env` file line by line rather than using a third-party loader, keeping the dependency count low. All four AuraDB credentials are stored as environment variables and never appear in the notebook code (Figure 4.3.1).

```

#STEP 2: NEO4j AuraDB CONNECTION TEST
# Run this to confirm credentials are working before running the full pipeline
from neo4j import GraphDatabase
driver = GraphDatabase.driver(uri, auth=(username, password))

try:
    with driver.session() as session:
        result = session.run("RETURN 'Connected to Neo4j AuraDB!' AS message")
        for record in result:
            print(record["message"])
        print("Connection successful!")
except Exception as e:
    print("Connection failed:", e)
finally:
    driver.close()

```

Figure 4.3.2: AuraDB Connection Test (Step 2)

In the second step, the code verifies the Neo4j driver can authenticate and reach the AuraDB instance before the pipeline runs. Running it independently means connection failures are caught immediately rather than surfacing mid-pipeline after several minutes of processing (Figure 4.3.2).

```

#STEP 3: ChatGPT TEST
from llama_index.llms.openai import OpenAI
import os

llm = OpenAI(
    model="gpt-4o-mini",
    api_key=os.environ.get("OPENAI_API_KEY"),
    temperature=0,
    timeout=120.0,
)

response = llm.complete("Say hello and confirm you are running via the OpenAI API.")
print(response)
print(response.raw) # shows token usage

```

Figure 4.3.3: LLM Connectivity Test (Step 3)

A lightweight API call, part of the third step in implementation, confirms the OpenAI key is valid and the model is reachable before the full pipeline is committed. The notebook also retains a fully commented-out Ollama test cell, reflecting that the pipeline originally supported local inference via Llama 3 8B and phi3 models. Those models remain available by uncommenting the relevant lines and substituting the llm object, making local inference a configurable fallback at zero API

cost (Figure 4.3.3).

```
# STEP 4: PDF text EXTRACTION from the module
import pdfplumber
import os
from docx import Document as DocxDocument

def extract_text_from_pdf(pdf_path):
    text = ""
    with pdfplumber.open(pdf_path) as pdf:
        for page in pdf.pages:
            extracted = page.extract_text()
            if extracted:
                text += extracted + "\n"
    return text

def extract_text_from_docx(docx_path):
    doc = DocxDocument(docx_path)
    text = ""
    for paragraph in doc.paragraphs:
        if paragraph.text.strip():
            text += paragraph.text + "\n"
    return text
```

Figure 4.3.4: Text Extraction Functions (Step 4)

For text extraction in the fourth step, two extraction functions handle both document types in the module data folder. PDF extraction uses pdfplumber, iterating page by page and appending a single newline after each page. DOCX extraction uses python-docx, iterating paragraph by paragraph and skipping empty lines. The single newline separator between pages is deliberate: the preprocessing function downstream splits text on double newlines to identify slide boundaries, so a single newline keeps content from the same page together as one block while still allowing the split to work correctly (Figure 4.3.4).

```

#STEP 5: Check which documents are already processed in AuraDB
from neo4j import GraphDatabase

driver = GraphDatabase.driver(uri, auth=(username, password))
with driver.session(database=database) as session:
    result = session.run("""
        MATCH (p:ProcessedDocument)
        RETURN p.filename AS filename
    """)
    processed_files = {record["filename"] for record in result}

driver.close()

print(f"Documents already in AuraDB: {len(processed_files)}")
for f in sorted(processed_files):
    print(f" ✓ {f}")

```

Figure 4.3.5: Processed Document Tracking (Step 5)

Before any extraction begins, the pipeline queries AuraDB for `ProcessedDocument` nodes — a lightweight tracking mechanism that records which files have already been inserted into the graph. The result is stored as a Python set for $O(1)$ lookup in the next step. This incremental design means the pipeline can be re-run safely on an existing graph: only genuinely new files are processed, preventing duplicate nodes from accumulating (Figure 4.3.5).

```

#STEP 6:Scan data folder and identify new documents not yet processed
import os
import re

# Pipeline scope: lecture PDF files only.
# Seminar tasks, module proforma, schedules, and revision sessions
# are excluded from this phase. Extending to DOCX seminar files
# is identified as future work.

all_files = os.listdir("data")
new_files = []

for filename in all_files:
    # Only process PDF files whose name contains "Lecture"
    if not filename.endswith(".pdf"):
        continue
    if not re.search(r"[Ll]ecture", filename):
        continue
    if filename in processed_files:
        print(f"»» Already processed: {filename}")
    else:
        new_files.append(filename)
        print(f"  New document found: {filename}")

print(f"\nTotal new documents to process: {len(new_files)}")

```

Figure 4.3.6: Data Folder Scan and File Selection (Step 6)

In step 6 the working folder is scanned. Only PDF files whose names contain "Lecture" are selected for processing. This filter excludes seminar task sheets, the module schedule, the proforma, ICT revision sessions, and Q&A session documents, all of which are present in the data/ folder but not relevant to the knowledge graph construction phase. DOCX support is implemented in the extraction functions but scoped as future work for seminar materials (Figure 4.3.6).

```

#STEP 7: PIPELINE SETUP
import re
from llama_index.llms.ollama import Ollama
from llama_index.embeddings.huggingface import HuggingFaceEmbedding
from llama_index.core.indices.property_graph import SchemaLLMPathExtractor
from typing import Literal
from llama_index.core import Settings
from llama_index.core.callbacks import CallbackManager, TokenCountingHandler
import tiktoken

#llm = Ollama(model="llama3:8b", request_timeout=3000.0, temperature=0)
#llm = Ollama(model="phi3:14b", request_timeout=1600.0, temperature=0)
#llm = Ollama(model="phi3-mini-4k", request_timeout=600.0, temperature=0)

token_counter = TokenCountingHandler(
    tokenizer=tiktoken.encoding_for_model("gpt-4o-mini").encode
)
callback_manager = CallbackManager([token_counter])

llm = OpenAI(
    model="gpt-4o-mini",
    api_key=os.environ.get("OPENAI_API_KEY"),
    temperature=0,
    timeout=600.0,
    additional_kwargs={"seed": 42},
    callback_manager=callback_manager,
)

embed_model = HuggingFaceEmbedding(model_name="BAAI/bge-small-en-v1.5")
Settings.callback_manager = callback_manager

```

Figure 4.3.7: Token Counting and LLM Configuration (Step 7)

The seventh step belongs to the pipeline setup. A `TokenCountingHandler` is registered on the LLM via a `CallbackManager` before any extraction begins. It uses `tiktoken` with the `gpt-4o-mini` tokenizer to count exact input and output tokens on every API call throughout the run. `temperature=0` makes extraction deterministic. `seed=42` is passed via `additional_kwargs` to further reduce run-to-run variability at the model level. `BAAI/bge-small-en-v1.5` is used as the local embedding model and remains unchanged across all runs since the embedding layer is not the variable under evaluation (Figure 4.3.7).

```

# TEXT PREPROCESSING
# Most patterns are generic across slide-based modules.
# The final two patterns are specific to the Maths for Computing module
# and should be reviewed when applying the pipeline to a different module.
SKIP_SLIDE_PATTERNS = [
    r"^WELCOME",
    r"^Module Team",
    r"^Sources\s*$",
    r"^Independent Study",
    r"^Lectures\s*$",
    r"^Seminars\s*$",
    r"^Q&A SESSION",
    r"^ACTIVITY\s*\d*[:\.]?s*$",
    r"^HOMEWORK ACTIVITY",
    r"^Let's get closer",
    r"^Module Organisation\s*$",
    r"^Module Syllabus\s*$",
    r"^Discrete Mathematics\s*$",
    r"^Mathematics\s+associated\s+with",
]

NOISE_LINE_PATTERNS = [
    r"^(Property|Attribute|Definition|Notation|Ordering|Repetition|"
    r"Membership|Indexing|Cardinality|Length|Inclusivity|Elements?|"
    r"Representation|Type|Notes?|Description|Example)\s*$",
    r"^\.{3,40}\s+(Examples?|Notation|Properties|Representation|Types?|Basics)\s*$",
    r"^\s*[\-+\\-]{3,}",
    r"^\s*[∅∈∉≤<unx']\s*$",
]

def preprocess_text(raw_text: str) -> str:
    text = raw_text.replace("\r\n", "\n").replace("\r", "\n")
    blocks = re.split(r"\n{2,}", text)
    cleaned_blocks = []
    for block in blocks:
        block = block.strip()
        if not block:
            continue
        # Check first 5 lines for skip patterns – headers/footers mean
        # slide title may not always be the very first line
        first_lines = [l.strip() for l in block.split("\n")[:5]]
        if any(
            re.match(p, line, re.IGNORECASE)
            for p in SKIP_SLIDE_PATTERNS
            for line in first_lines
        ):
            continue
        lines = block.split("\n")
        clean_lines = []
        for line in lines:
            line = line.strip()
            if not line:
                continue
            if any(re.match(p, line, re.IGNORECASE) for p in NOISE_LINE_PATTERNS):
                continue
            if len(line) < 4 and not re.search(r"[a-zA-Z]{2,}", line):
                continue
            clean_lines.append(line)
        if len(clean_lines) >= 3:
            cleaned_blocks.append("\n".join(clean_lines))
    return "\n\n".join(cleaned_blocks)

```

Figure 4.3.8: Preprocessing Pattern Lists and Function (Step 7)

Preprocessing operates at two levels. At block level, raw text is split on double newlines into slide-sized chunks. The first five lines of each block are checked against `SKIP_SLIDE_PATTERNS`; a match discards the entire block. This removes welcome slides, module team pages, Q&A prompts, activity slides, and two module-specific patterns — the module syllabus overview and module organisation slide — that caused cross-week topic contamination in early runs. The last two patterns in the list are specific to 4COSC002W and would need reviewing before applying the pipeline to a different module. At line level, `NOISE_LINE_PATTERNS` removes table column headers, section sub-headings ending in words such as "Examples" or "Basics", table border characters, and isolated mathematical symbols. Any line under four characters with no alphabetic content is also removed. Blocks with fewer than three clean lines are discarded entirely (Figure 4.3.8).

```

# EXTRACTION SCHEMA
# Entity types and relationships defined jointly with 04 ontology (ontology_v1.owl)
# TOPIC list is authoritative for module 4COSC002W – update when ontology is revised
entities = Literal[
    "CONCEPT",      # named mathematical idea e.g. Modular Arithmetic
    "TOPIC",          # thematic cluster e.g. Number Theory
    "WEEK",           # source week anchor e.g. Week 1
    "APPLICATION",    # real-world domain e.g. Cryptography
    "ASSESSMENT",     # ICT, Exam, Mock Test
    "DEFINITION",     # formal definition of a concept
    "EXAMPLE",        # concrete worked example illustrating a concept
    "FORMULA",        # mathematical formula or equation
]

relations = Literal[
    "INTRODUCED_IN",  # concept first introduced in a week
    "PART_OF",        # concept belongs to topic or week; topic belongs to week
    "PREREQUISITE_OF", # must understand A before B
    "HAS_DEFINITION",  # concept has a formal definition
    "HAS_EXAMPLE",     # concept has a worked example
    "HAS_FORMULA",     # concept has an associated formula
    "HAS_APPLICATION", # concept applies to a real-world domain
    "ASSESSED_BY",     # concept appears in an assessment
    "USED_IN",         # concept is used within another concept
]

validation_schema = [
    ("CONCEPT", "INTRODUCED_IN", "WEEK"),
    ("CONCEPT", "PART_OF", "TOPIC"),
    ("CONCEPT", "PART_OF", "WEEK"),
    ("TOPIC", "PART_OF", "WEEK"),
    ("CONCEPT", "PREREQUISITE_OF", "CONCEPT"),
    ("CONCEPT", "HAS_DEFINITION", "DEFINITION"),
    ("CONCEPT", "HAS_EXAMPLE", "EXAMPLE"),
    ("CONCEPT", "HAS_FORMULA", "FORMULA"),
    ("CONCEPT", "HAS_APPLICATION", "APPLICATION"),
    ("TOPIC", "HAS_APPLICATION", "APPLICATION"),
    ("CONCEPT", "ASSESSED_BY", "ASSESSMENT"),
    ("CONCEPT", "USED_IN", "CONCEPT"),
]

kg_extractor = SchemaLLMPathExtractor(
    llm=llm,
    possible_entities=entities,
    possible_relations=relations,
    kg_validation_schema=validation_schema,
    extract_prompt=extraction_prompt,
    strict=True
)

print("Pipeline components ready:")
print(f"✓ LLM: {llm.model} (local)")
print("✓ Embeddings: BAAI/bge-small-en-v1.5 (local)")
print("✓ Schema: defined with", len(validation_schema), "relationship types")

```

Figure 4.3.9: Extraction Schema and Knowledge Graph Extractor (Step 7)

The extraction schema defines the full vocabulary of the knowledge graph. Entity types and relationship types are expressed as Python Literal types, which SchemaLLMPPathExtractor uses as a constrained output vocabulary. The validation_schema lists the only permitted subject-predicate-object triplet combinations. strict=True enforces this at extraction time: any triple whose subject, predicate, or object does not match the schema is discarded before graph insertion. The schema was designed before the O4 ontology was delivered and the two turned out to be well-aligned when it arrived. The extraction prompt (abbreviated here) provides per-entity definitions with positive examples drawn from module content, explicit good and bad examples for CONCEPT to prevent sub-heading extraction, and a fixed TOPIC allowlist: "Number Theory", "Sequences", "Set Theory", "Intervals", "Relations", "Functions", "Logics", "Graph Theory", "Matrices Part 1", "Matrices Part 2", "Probability", "Statistics Basics", "Statistical Application". The model is instructed not to create TOPIC nodes outside this list and not to extract parenthetical variants such as "Integers (Z)" in place of "Integers" (Figure 4.3.9).

```

# STEP 8: Connect to existing AuraDB graph or initialise fresh
from llama_index.core import PropertyGraphIndex
from llama_index.graph_stores.neo4j import Neo4jPropertyGraphStore

graph_store = Neo4jPropertyGraphStore(
    username=username,
    password=password,
    url=uri,
    database=database
)

# Check if graph already exists
driver = GraphDatabase.driver(uri, auth=(username, password))
with driver.session(database=database) as session:
    count = session.run("MATCH (n) RETURN count(n) AS count").single()["count"]
driver.close()

if count > 0:
    # Graph already exists – connect to it without rebuilding
    print(f"Existing graph found: {count} nodes in AuraDB")
    print("Connecting to existing graph...")
    index = PropertyGraphIndex.from_existing(
        property_graph_store=graph_store,
        llm=llm,
        embed_model=embed_model,
    )
    print("Connected to existing graph successfully!")
else:
    # No graph exists – build fresh
    print("No existing graph found – initialising fresh index...")
    index = PropertyGraphIndex.from_documents(
        [],
        kg_extractors=[kg_extractor],
        embed_model=embed_model,
        property_graph_store=graph_store
    )
    print("Fresh index created in AuraDB")

```

Figure 4.3.10: Graph Connection and Index Initialisation (Step 8)

The pipeline connects to an existing AuraDB graph or initialises a fresh one depending on whether nodes are already present. A critical implementation constraint applies here: `kg_extractors=[kg_extractor]` must be passed on every call, both `from_documents()` and `from_existing()` (Figure 4.3.10). Omitting it on `from_existing()` causes `LlamaIndex` to silently revert to its default free-text extractor, producing generic untyped nodes with natural-language relationships that contaminate the graph. This failure is invisible at runtime. The pipeline reports success and nodes appear in AuraDB, and it was only discovered by inspecting the graph structure directly in the Neo4j browser.

```

def process_and_track(filename, index, driver, database):
    """Extract text, preprocess, insert into graph, mark as processed in AuraDB"""

    filepath = f"data/{filename}"
    file_type = "pdf" if filename.endswith(".pdf") else "docx"

    if file_type == "pdf":
        raw_text = extract_text_from_pdf(filepath)
    else:
        raw_text = extract_text_from_docx(filepath)

    if not raw_text.strip():
        print(f"⚠ Skipping empty document: {filename}")
        return False

    text = preprocess_text(raw_text)
    print(f" {len(raw_text)} chars → {len(text)} chars after preprocessing")

    match = re.search(r"[Ll]ecture(\d+)", filename)
    week_label = f"Week {match.group(1)}" if match else None

    # Pre-inject WEEK anchor node
    if week_label:
        with driver.session(database=database) as session:
            session.run("""
                MERGE (l:__Node__:__Entity__:WEEK {id: $label})
                SET l.name = $label
                """, label=week_label)
            print(f" ✓ Pre-injected {week_label} node")

    # Snapshot existing concept IDs BEFORE insertion
    with driver.session(database=database) as session:
        result = session.run("MATCH (c:CONCEPT) RETURN c.id AS id")
        existing_concept_ids = [record["id"] for record in result]

    doc = Document(
        text=text,
        metadata={"filename": filename, "file_type": file_type}
    )
    index.insert(doc)

    # Remove Chunk nodes (LlamaIndex internals)
    with driver.session(database=database) as session:
        result = session.run("MATCH (c:Chunk) DETACH DELETE c RETURN count(c) AS deleted")
        print(f" ✓ Removed {result.single()['deleted']} Chunk nodes")

    # Wire NEW concepts (not in snapshot, not yet introduced anywhere) to this lecture
    if week_label:
        with driver.session(database=database) as session:
            result = session.run("""
                MATCH (c:CONCEPT), (l:WEEK {name: $label})
                WHERE NOT c.id IN $existing_ids
                AND NOT (c)-[:INTRODUCED_IN]->(l)
                MERGE (c)-[:INTRODUCED_IN]->(l)
                RETURN count(c) AS wired
                """, label=week_label, existing_ids=existing_concept_ids)
            print(f" ✓ Wired {result.single()['wired']} concepts to {week_label}")

```

```

# Wire orphan TOPICs to this lecture
if week_label:
    with driver.session(database=database) as session:
        result = session.run("""
            MATCH (t:TOPIC), (l:WEEK {name: $label})
            WHERE NOT (t)-[:PART_OF]->(l)
            AND NOT t.id IN $existing_ids
            MERGE (t)-[:PART_OF]->(l)
            RETURN count(t) AS wired
        """, label=week_label, existing_ids=existing_concept_ids)
        print(f" ✓ Wired {result.single()['wired']} topics to {week_label}")

# Mark as processed
with driver.session(database=database) as session:
    session.run("""
        MERGE (p:ProcessedDocument {filename: $filename})
        SET p.file_type = $file_type,
            p.processed_at = datetime()
    """, filename=filename, file_type=file_type)

return True

if new_files:
    driver = GraphDatabase.driver(uri, auth=(username, password))
    for filename in new_files:
        print(f"\nProcessing: {filename}")
        success = process_and_track(filename, index, driver, database)
        if success:
            print(f"✓ Done: {filename}")
    driver.close()
    print(f"\nAll {len(new_files)} new documents processed!")
else:
    print("No new documents to process – graph is up to date!")

print(f"\n— Token Usage —")
print(f" Input tokens: {token_counter.prompt_llm_token_count:,}")
print(f" Output tokens: {token_counter.completion_llm_token_count:,}")
print(f" Total: {token_counter.total_llm_token_count:,}")

```

Figure 4.3.11: Document Processing Function: `process_and_track` (Step 9)

In Figure 4.3.11, `process_and_track` is the core per-document function. The week label is parsed from the filename using a regex on "Lecture" followed by a number. Before `index.insert()` runs, two operations are performed in sequence: a WEEK anchor node is created in AuraDB via MERGE to guarantee it exists regardless of LLM extraction behaviour, and a snapshot of all current CONCEPT node IDs is taken. After insertion, LlamaIndex's internal Chunk nodes are deleted — they are indexing artefacts with no educational value. Two post-processing Cypher

queries then wire newly extracted nodes: the first creates INTRODUCED_IN relationships from any CONCEPT not in the snapshot and not yet linked to any WEEK; the second wires orphan TOPIC nodes to the current lecture's week. Both queries restrict to IDs absent from the snapshot, preventing concepts from earlier lectures being incorrectly re-wired on incremental runs. Finally a ProcessedDocument node is written to mark the file complete. After all files are processed, the token counter reports the exact input and output token counts consumed across the full run, enabling precise cost calculation.

```
# STEP 10: Verify final state of AuraDB
driver = GraphDatabase.driver(uri, auth=(username, password))
with driver.session(database=database) as session:
    node_count = session.run(
        "MATCH (n) WHERE NOT n:ProcessedDocument RETURN count(n) AS count"
    ).single()["count"]
    rel_count = session.run(
        "MATCH ()-[r]->() RETURN count(r) AS count"
    ).single()["count"]
    processed_count = session.run(
        "MATCH (p:ProcessedDocument) RETURN count(p) AS count"
    ).single()["count"]

driver.close()

print(f"Knowledge graph nodes: {node_count}")
print(f"Relationships: {rel_count}")
print(f"Documents tracked: {processed_count}")
```

Figure 4.3.12: Graph State Verification (Step 10)

Three Cypher queries confirm the graph state after processing: total node count (excluding ProcessedDocument tracking nodes), total relationship count, and number of tracked documents. This provides a quick sanity check before running validation (Figure 4.3.12).

```

g = RDFGraph()
g.parse("ontology_v2.owl", format="xml")

# Extract Topic individuals
ontology_topics = set()
for s, p, o in g.triples((None, RDF.type, None)):
    if str(o).split("#")[-1] == "Topic":
        labels = list(g.objects(s, RDFS.label))
        if labels:
            ontology_topics.add(str(labels[0]))
        else:
            ontology_topics.add(str(s).split("#")[-1].replace("_", " "))

# Extract Week individuals and normalise Week1 → Week 1
ontology_weeks = set()
for s, p, o in g.triples((None, RDFS.label, None)):
    label = str(o)
    if re.match(r'^Week \d+$', label):
        ontology_weeks.add(label)

# Extract Assessment individuals
ontology_assessments = set()
for s, p, o in g.triples((None, RDF.type, None)):
    type_local = str(o).split("#")[-1]
    if type_local in ["Exam", "InClassTest", "MockTest"]:
        labels = list(g.objects(s, RDFS.label))
        if labels:
            ontology_assessments.add(str(labels[0]))
        else:
            ontology_assessments.add(str(s).split("#")[-1].replace("_", " "))

# Extract week→topic mapping
HAS_TOPIC = URIRef("http://www.example.org/4C0SC002W#hasTopic")
ontology_week_topics = {}
for week_uri, _, topic_uri in g.triples((None, HAS_TOPIC, None)):
    week_labels = list(g.objects(week_uri, RDFS.label))
    topic_labels = list(g.objects(topic_uri, RDFS.label))
    if week_labels and topic_labels:
        week_name = str(week_labels[0])
        topic_name = str(topic_labels[0])
        if re.match(r'^Week \d+$', week_name):
            ontology_week_topics.setdefault(week_name, set()).add(topic_name)

```

```

tp = len(extracted_topics & ontology_topics)
fp = len(extracted_topics - ontology_topics)
fn = len(ontology_topics - extracted_topics)
topic_precision = tp / (tp + fp) if (tp + fp) > 0 else 0
topic_recall = tp / (tp + fn) if (tp + fn) > 0 else 0
topic_f1 = (2 * topic_precision * topic_recall /
            (topic_precision + topic_recall)
            if (topic_precision + topic_recall) > 0 else 0)

week_coverage = (len(extracted_weeks) - len(invalid_weeks)) / len(ontology_weeks) \
                if ontology_weeks else 0
alignment_score = correct_alignments / total_alignments \
                if total_alignments > 0 else 0

print(f"\nTopic extraction (vs ontology topic list)")
print(f" Precision:      {topic_precision:.1%}")
print(f" Recall:         {topic_recall:.1%}")
print(f" F1:              {topic_f1:.1%}")
print(f"\nWeek coverage (vs ontology week list)")
print(f" {len(extracted_weeks) - len(invalid_weeks)}/{len(ontology_weeks)} weeks correctly named: {week_coverage:.1%}")
print(f"\nWeek-topic alignment (vs ontology hasTopic)")
print(f" {correct_alignments}/{total_alignments} topic-week pairs correct: {alignment_score:.1%}")

# GRAPH QUALITY METRICS
# These measure internal pipeline quality – not validated against the ontology
print("\n" + "="*55)
print("GRAPH QUALITY METRICS")
print("(internal pipeline assessment – not ontology-derived)")
print("="*55)

anchor_rate = anchored_concepts / total_concepts if total_concepts else 0
def_coverage = concepts_with_def / total_concepts if total_concepts else 0
schema_rel_types = 9
rel_diversity = used_rel_types / schema_rel_types
noise_rate = noise_row["n"] / noise_row["total"] if noise_row["total"] > 0 else 0

```

Figure 4.3.13: Ontology Validation and Metric Computation (Step 11)

The validation step uses `rdflib` to parse `ontology_v2.owl` and extract two ground-truth structures: the set of valid Topic individuals (via `RDF.type` triples) and the week-to-topic mapping (via `hasTopic` object property triples). The extracted graph is then queried via Cypher to retrieve the topics, weeks, and assessments it contains (Figure 4.3.13). Two distinct metric groups are then computed. Ontology conformance scores compare the extracted graph directly against the OWL ground truth: Topic Precision (valid extracted topics divided by all extracted topics), Topic Recall (valid extracted topics divided by all ontology topics), Topic F1 (harmonic mean of the two), Week Coverage (correctly named weeks divided by all ontology weeks), and Week-Topic Alignment (correct week-topic pairs divided by all ontology-defined pairs). Graph quality metrics assess internal pipeline consistency without reference to the ontology: Concept Anchor Rate (CONCEPT nodes linked to a WEEK via `INTRODUCED_IN`), Definition Coverage (CONCEPT nodes with at least one `HAS_DEFINITION`), Relationship Diversity (distinct schema relationship types present out of nine defined), Duplicate Concept count, and Noise Rate (CONCEPT nodes whose name exceeds 50 characters, which flags verbatim slide text extracted as a concept). The separation of the two metric groups is intentional: conformance scores measure alignment with the formal curriculum model, while quality metrics measure whether the pipeline's own structural rules were upheld.

```

# STEP 12: Natural language query engine
import nest_asyncio
nest_asyncio.apply()

from llama_index.core import Settings

Settings.llm = llm
Settings.embed_model = embed_model

query_engine = index.as_query_engine(
    llm=llm,
    include_text=True,
    response_mode="tree_summarize"
)

print("Query engine ready!")

def ask(question):
    response = query_engine.query(question)
    print(f"Q: {question}")
    print(f"A: {response}")
    print("-" * 50)

ask("What concepts are introduced in Week 1?")

```

Figure 4.3.14: Natural Language Query Engine (Step 12)

The query engine is built directly from the same PropertyGraphIndex used for construction, requiring no separate indexing step. `include_text=True` passes both graph traversal results and the original chunk text into the LLM context window, grounding responses in the source material. `response_mode="tree_summarize"` aggregates results from multiple retrieved nodes into a single coherent answer rather than returning raw fragments. This retrieval layer forms the interface between the constructed graph and the O1 conversational assistant, and demonstrates that the graph is queryable in natural language from the moment construction completes (Figure 4.3.14).

Concluding Summary and Comparison with O2.2

The O2.1 implementation demonstrates an end-to-end workflow for transforming multiple lecture materials into a persistent educational knowledge graph. The pipeline combines document ingestion and preprocessing, schema-constrained LLM extraction, graph cleaning, Neo4j AuraDB storage, ontology-informed validation and natural-language graph access. These components operate as related stages of a single prototype, while remaining sufficiently modular for individual inspection and refinement. The implementation was tested across ten Mathematics for Computing lecture files using gpt-4o-mini. The resulting graph, shown in Figure 4.3.15, contains 551 nodes and 578 relationships. It integrates content extracted from the lecture corpus into a shared module-level representation rather than maintaining separate graphs for individual documents.

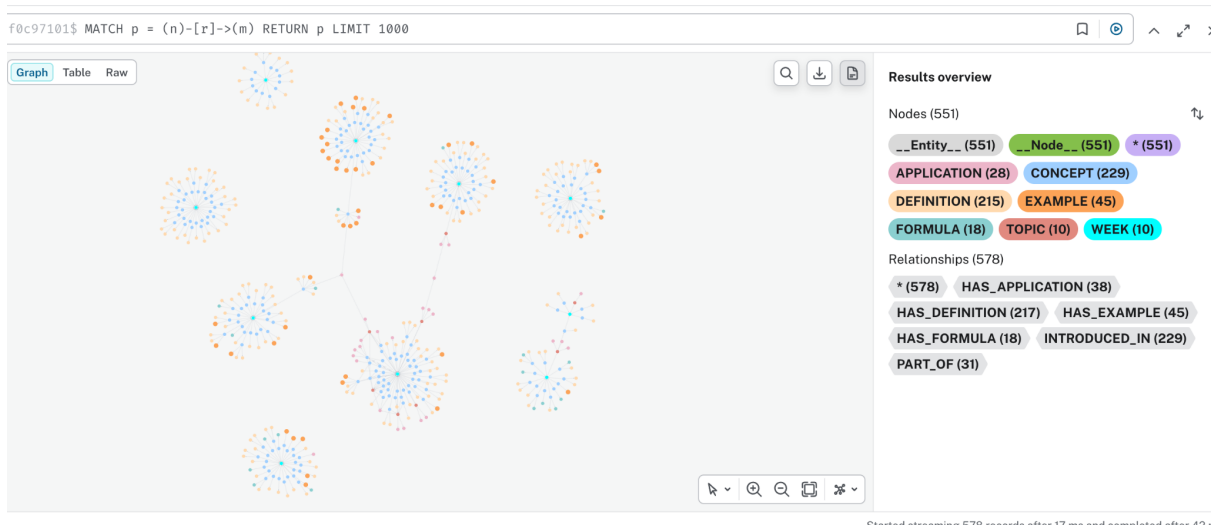


Figure 4.3.15. Neo4j AuraDB knowledge graph generated from ten Mathematics for Computing lectures using gpt-4o-mini (551 nodes and 578 relationships).

The graph includes the educational entity and relationship types defined by the extraction schema, allowing concepts, topics, weeks, definitions, examples, formulas, applications and assessment-related information to be represented within a common structure. Its scale demonstrates that the pipeline can process a substantial collection of teaching materials and persist the resulting structured content in a graph database suitable for traversal, inspection and retrieval. However, the number of nodes and relationships should be interpreted as evidence of technical graph generation rather than evidence of educational correctness. A graph may be structurally connected while still containing weak names, incorrect classifications, missing prerequisite links or relationships that require subject-expert interpretation. The generated structure therefore remains a candidate concept map until it has passed ontology-based checks and academic review. Overall, the O2.1 implementation establishes the technical feasibility of moving from unstructured lecture materials to a queryable module-level knowledge graph. It also provides the knowledge base required by the GraphRAG and conversational components described elsewhere in the report. The quality and pedagogical suitability of the resulting graph are examined in Chapter 5.

In comparison with the parallel O2.2 pipeline (Section 4.3.2), O2.1 prioritises schema control and comparative model evaluation: its eight-entity schema and ontology-derived topic allowlist produce a richer typed graph, and its cross-model benchmark (Section 5.1) quantifies the trade-offs between local and API-hosted models in accuracy, runtime and cost. O2.2, by contrast, holds the model constant and invests in the engineering controls surrounding a single locally hosted model — automatic semantic filtering, formal SHACL validation applied before storage, and a transparent natural-language-to-Cypher query route. Taking ontology conformance, graph richness, evaluation coverage, runtime and cost together, O2.1 is identified as the preferable pipeline for further research within the WENDY project, with the O2.2 techniques — in particular the validate-before-write SHACL gate and the graph-native question-answering route — recommended for incorporation into its future development. A structured side-by-side comparison

is provided in Section 4.3.3, and the developers' fuller reflective comparison appears in Section 5.1.1.

4.3.2 O2.2 Pipeline: Locally Hosted Generation with SHACL Validation

Overview and Implementation Environment

The O2.2 prototype investigates whether an educational knowledge graph of comparable structural quality can be constructed entirely with locally hosted, freely available models, removing API costs and keeping all teaching materials within institutional infrastructure. It was implemented as a set of reusable Python scripts rather than a notebook: a graph-construction pipeline (`graph_construction_pipeline.py`) performs extraction, filtering, validation and storage, while `setup.py` and `run.py` provide reproducible environment creation and execution. All generative and embedding models run locally through Ollama — Llama 3 (temperature 0, for deterministic extraction) as the extraction model and `nomic-embed-text` for embeddings — so a complete construction run incurs no API cost. The full source code, together with the validation artefacts and a populated test database, is provided in Appendix D (O2.2-GraphGeneration-Prototype).

The pipeline targets a deliberately simplified, property-centric graph model agreed with the O4 team: `Module` → `HAS_WEEK` → `Week`, `Week` → `COVERS` → `Topic` and `Topic` → `HAS_CONCEPT` → `Concept` form the structural backbone (with `Week` → `NEXT` → `Week` expressing temporal order), while concepts are interlinked through the semantic relationships `RELATED_TO`, `BUILDS_ON` and `PREREQUISITE_OF`. In contrast to the eight-entity O2.1 schema, definitions, formulas, examples and applications are not modelled as separate nodes; they are stored as text properties of the `Concept` node they describe. This design decision reduces node proliferation, keeps each concept self-contained and considerably simplifies both validation and downstream querying, at the cost of making the auxiliary content less directly traversable.

Document Processing and Noise Filtering

Lecture PDFs are loaded page by page using the `LlamaIndex PDFReader`, and an LLM-assisted metadata step identifies the module name and teaching week for each file so that every extracted element can later be anchored to the correct part of the curriculum. The text then passes through an extensive rule-based cleaning layer: heuristics detect and discard non-academic or excessively noisy content, and specialised detectors skip overview and agenda slides, revision lists and activity or example-only chunks before any model is invoked. The surviving content is segmented with a sentence-aware splitter (1,000-character chunks with 150-character overlap). These filters serve the same purpose as the O2.1 `SKIP_SLIDE` and `NOISE_LINE` patterns but operate as composable Python predicates that can be tuned independently of the extraction prompt.

Prompt-Based Knowledge Extraction and Semantic Filtering

Each chunk is submitted to Llama 3 with a carefully engineered prompt that specifies the target

graph structure, explicit node-quality rules (for example, prohibiting table column labels, activity names, full sentences and formula fragments as concept names), positive and negative examples, and a strict JSON-only output format in which every topic, concept and relationship carries a confidence score and a supporting source quote. Robust JSON recovery and normalisation routines repair the common failure modes of small local models before the output is accepted. A distinctive feature of O2.2 is the automatic semantic-filtering layer applied after extraction: source-support scoring verifies that each extracted name is actually grounded in the chunk text; name-quality heuristics penalise table attributes, weak "reason/benefit" labels and mathematical expressions; embedding similarity computed with nomic-embed-text (cosine similarity, 0.88 threshold) merges semantic duplicates across topics and concepts; and combined confidence scores determine whether each candidate survives. Additional routines infer broader topic names where a slide heading duplicates its concept, and derive semantic relationships between the retained concepts. Only candidates passing all of these checks are assembled into the graph payload.

SHACL Validation and Graph Storage (Validate-Before-Write)

Where O2.1 scores ontology conformance after the graph has been built, O2.2 embeds formal validation directly into construction. The candidate payload for each chunk is converted into an RDF graph using `rdflib` and validated with `pySHACL` against a set of SHACL shapes (`kg_validation_shapes.ttl`) that encode the agreed graph structure — node naming constraints, permitted relationship types and their directions — and are aligned with a companion OWL ontology (`kg_ontology.owl`). A chunk whose payload does not conform is rejected in its entirety and never reaches the database; chunks yielding no meaningful academic content after filtering are likewise skipped. Conforming payloads are written to Neo4j AuraDB through parameterised Cypher MERGE statements, with uniqueness constraints created up front and automatic retries around transient connection failures. At the end of a run the pipeline reports summary statistics: chunks validated and saved, chunks failing SHACL validation, empty or noisy chunks skipped, write failures, and total node and relationship write attempts. This validate-before-write design guarantees that everything persisted in the graph already satisfies the formal structural model, complementing the O4 validation framework described in Section 4.5.

Natural-Language Querying (NL-to-Cypher)

O2.2 also contributes a second, graph-native question-answering prototype. A separate script implements a natural-language-to-Cypher workflow using LangChain components (`langchain-neo4j`, `langchain-ollama`): a schema-constrained prompt instructs Llama 3 to generate exactly one valid Cypher query using only the node labels, relationship types and the name property declared in the prompt schema; the query is sanitised and executed against Neo4j; and the retrieved results are passed back to the model to produce a short natural-language answer, with the generated Cypher and raw database context displayed for transparency. LangChain's prebuilt `GraphCypherQAChain` was trialled first but produced inconsistent and occasionally invalid queries, so it was replaced by a custom generation-sanitisation-execution loop that gives full control over prompting and error handling. This query route differs from the O2.1

PropertyGraphIndex engine in that answers are grounded exclusively in explicit graph traversal results, providing a transparent, fully local GraphRAG interaction pattern at zero API cost. The workflow follows: User Question → Llama 3 generates Cypher → Neo4j executes query → Results returned → Llama 3 generates final answer.

Concluding Summary of the O2.2 Implementation

The construction pipeline was evaluated on two Mathematics for Computing lecture decks (Weeks 1 and 2): 49 cleaned page-level chunks were processed, of which 44 (89.8%) were validated and saved, three (6.1%) were rejected by the SHACL gate and two (4.1%) were skipped as empty or noisy, with no Neo4j write failures; 228 node and 187 relationship write operations were attempted. The run produced coherent, educationally recognisable topic structures, including number systems, sequences, set operations, set algebra and intervals, and the SHACL layer demonstrably intercepted malformed fragments (for example, chunks in which a Topic and a Concept carried the same name) before they could reach the database. The test graph database, the raw extraction output and the validation schemas are archived alongside the code in Appendix D.

Overall, the O2.2 implementation demonstrates that a formally validated educational knowledge graph can be constructed and queried end to end on commodity local hardware, at zero API cost and with all teaching materials remaining on institutional infrastructure. Its principal contributions to the O2 work package are the validate-before-write SHACL gate, the automatic semantic-filtering layer and the transparent natural-language-to-Cypher query route — techniques that are directly transferable to future development regardless of which pipeline is taken forward. The full O2.2 project journey and detailed findings are presented in Section 5.1.1; prototype-specific limitations are discussed in Section 8.3 and future-work items in Section 8.4.

4.3.3 Comparative Summary and Conclusions

Table 4.3.1 summarises the two O2 pipeline implementations side by side. The comparison reflects the design dimensions on which the prototypes deliberately diverged; both share the same overall goal, the same case-study module and the same target database platform.

Dimension	O2.1 pipeline (Section 4.3.1)	O2.2 pipeline (Section 4.3.2)
Implementation form	Jupyter notebook (smartest_o2_pipeline.ipynb), conda environment	Reusable Python scripts with setup.py/run.py, virtual environment
Extraction models	OpenAI gpt-4o-mini and gpt-4o (API); Llama 3 8B and phi3 (local) for comparison	Llama 3 (local, via Ollama) throughout; no model benchmark
Extraction approach	LlamaIndex PropertyGraphIndex with SchemaLLMPPathExtractor (strict=True)	Custom JSON extraction prompt with Python post-processing
Graph schema	Eight entity types, nine relationship types; definitions, examples, formulas and applications as separate nodes	Four active node labels (Module, Week, Topic, Concept); auxiliary content stored as Concept properties
Topic control	Ontology-derived TOPIC allowlist enforced in the prompt	Topic inference, normalisation and synonym correction after extraction

Duplicate handling	Post-processing Cypher cleanup and orphan re-wiring	Source-support scoring, name-quality heuristics, fuzzy matching and embedding similarity (nomic-embed-text)
Validation placement	Post-hoc ontology conformance scoring and graph quality metrics (rdflib + Cypher)	Per-chunk SHACL validation before storage (rdflib + pySHACL); non-conforming chunks rejected
Embeddings	BAAI/bge-small-en-v1.5 (Hugging Face)	nomic-embed-text (Ollama, local)
Natural-language querying	PropertyGraphIndex query engine (include_text, tree_summarize)	Custom NL-to-Cypher script (LangChain, Llama 3) with visible Cypher and context
Evaluation scope	Eight to ten lectures; three-model cross-benchmark (Section 5.1)	Two lecture decks (Weeks 1 and 2); single-model run
Cost profile	~£0.02 (gpt-4o-mini) to ~£0.28 (gpt-4o) per run; local option at £0	Zero API cost; fully local

Table 4.3.1. Side-by-side comparison of the O2.1 and O2.2 pipeline implementations.

The two prototypes expose complementary trade-offs between schema control, extraction flexibility, graph richness and end-user access. O2.1 demonstrates that schema-constrained extraction with an ontology-derived allowlist yields the strongest ontology conformance (Topic F1 of 64.0% with gpt-4o-mini) at very low per-run cost, and its cross-model benchmark provides the evidence base for model selection. O2.2 demonstrates that formal validation can be embedded directly into construction, that a fully local pipeline is viable at zero API cost, and that the resulting graph can be accessed through a transparent natural-language-to-Cypher interface. On the combined evidence of conformance, graph richness, evaluation coverage, runtime and cost, the project concludes that O2.1 is the preferable pipeline for further research and future SMARTTEST integration; the validate-before-write SHACL gate, semantic-filtering techniques and graph-native question-answering route developed in O2.2 are recommended for incorporation into that continuing line of development.

4.4 O3 Theme Discovery Implementation

The O3 work package did not produce a production software component; in agreement with the project's overall emphasis, effort was directed at methodology — implementing the PETER embedding-based soft topic-discovery approach described in Section 3.5 and testing its central assumptions empirically. The implementation, therefore, took the form of a reproducible experimentation pipeline built in Jupyter notebooks, using spaCy for text pre-processing and sentence segmentation, rather than a deployable service.

4.4.1 Dataset Selection and Corpus Profiling

Seven public corpora were profiled against three selection criteria: sufficient contextual length (a minimum of 50 words and three sentences per unit, since the pipeline relies on SBERT contextual embeddings rather than bag-of-words features), the presence of reliable ground-truth annotations against which probabilistic soft assignments could be evaluated, and the availability of raw text rather than pre-computed features.

Two corpora were excluded outright: AG News, because most of its documents consist of a single sentence and so cannot support sentence-level aggregation, and LectureBank — the project's only genuinely educational candidate — because of taxonomy inconsistencies (including a

generic “Introduction” class covering 21.7% of documents) and extreme document-length variance inherited from its slide-based source material. RCV1-v2 was retained only as a negative control, since its public distribution provides pre-computed TF-IDF vectors rather than raw text. The five remaining corpora — BBC News, Reuters-21578, RCV1-v2, WikiSection and 20 Newsgroups — were carried forward into the experimental programme, selected to cover a graded range of document length, label granularity, class balance and, for WikiSection, natural sub-document structure.

Profiling the candidate corpora against these criteria surfaced a broader benchmarking gap that shaped the interpretation of the results in Section 5.2.3: no available public corpus combines reliable multi-label ground truth, sub-document structure and educational content simultaneously. A purpose-built educational corpus of approximately 200–500 paragraph-segmented documents, annotated with overlapping topics, would offer a stronger evaluation framework than any existing public benchmark, and is recommended as future work in Section 8.4.5.

4.4.2 Experimental Pipeline and Configuration

All experiments shared a common configuration to keep the comparison controlled. Sentence embeddings were generated with the all-MiniLM-L6-v2 SentenceTransformer model, held constant across every configuration; BERTopic served as the primary baseline and was benchmarked against K-Means and HDBSCAN, each run both with and without UMAP dimensionality reduction beforehand; a fixed random seed (42) was used throughout to support reproducibility. Performance was assessed using Adjusted Rand Index and Normalised Mutual Information for agreement with ground-truth labels, Silhouette Score for embedding-space separation, majority-vote accuracy for cluster purity, and coverage for the proportion of non-noise assignments — the same metrics detailed in Section 5.1.4.

4.4.3 Experimental Programme

Five experiments were conducted, summarised in Table 5.2.4. The first four applied the pipeline to complete documents or natural sub-document sections at increasing levels of difficulty; the fifth applied it at sentence level.

Experiment 1 (BBC News) established a boundary-detection baseline on 2,225 whole documents, comparing four clustering configurations to verify that the pipeline functioned correctly end to end.

Experiment 2 (Reuters-21578) tested multi-label validation under severe class imbalance, restricted to the ten most frequent categories.

Experiment 3 (RCV1-v2) evaluated clustering without access to raw text, using TF-IDF features reduced via TruncatedSVD, to isolate the effect of representation quality from that of the clustering method itself.

Experiment 4 (WikiSection) compared two domains — en_disease and en_city — at section level, using an identical processing pipeline for both, to test how taxonomic separability affects performance independently of the pipeline configuration.

Experiment 5 (20 Newsgroups, sentence level) directly tested the project's central hypothesis: whether individual sentences carry sufficient semantic signal for reliable topic assignment. A stratified subset of 2,000 documents was cleaned and segmented into 29,299 sentences for this run.

Taken together, the experiments established that the embedding-and-clustering pipeline performs strongly on whole-document and section-level units, and identified — as the project's core

negative result — that individual sentences are generally too small a unit for reliable topic assignment. These findings directly informed the recommendation, carried into the WENDY methodology, that theme discovery should operate on text units of approximately 50–150 words (paragraphs, sections or sliding windows). The full results and their synthesis are presented in Section 5.2.3.

4.5 O4 Validation Framework Implementation

The educational ontology was implemented in **Protégé** using the Web Ontology Language (OWL). Its purpose is to provide the formal semantic structure against which module content and AI-generated knowledge graphs can be interpreted and validated. Whereas the extraction pipeline generates candidate instances from teaching materials, the ontology defines the expected educational entities, their hierarchy, the relationships permitted between them and the associated data properties.

The uploaded ontology identifies itself as **University Module Ontology, version 4.0**. It represents the general structure of a university module delivered across a twelve-week semester. Although its namespace retains the Mathematics for Computing module code, the ontology has been progressively generalised so that its principal classes and relationships can be reused for other modules. The Mathematics for Computing module remains included as a populated case study through which the model can be inspected and tested.

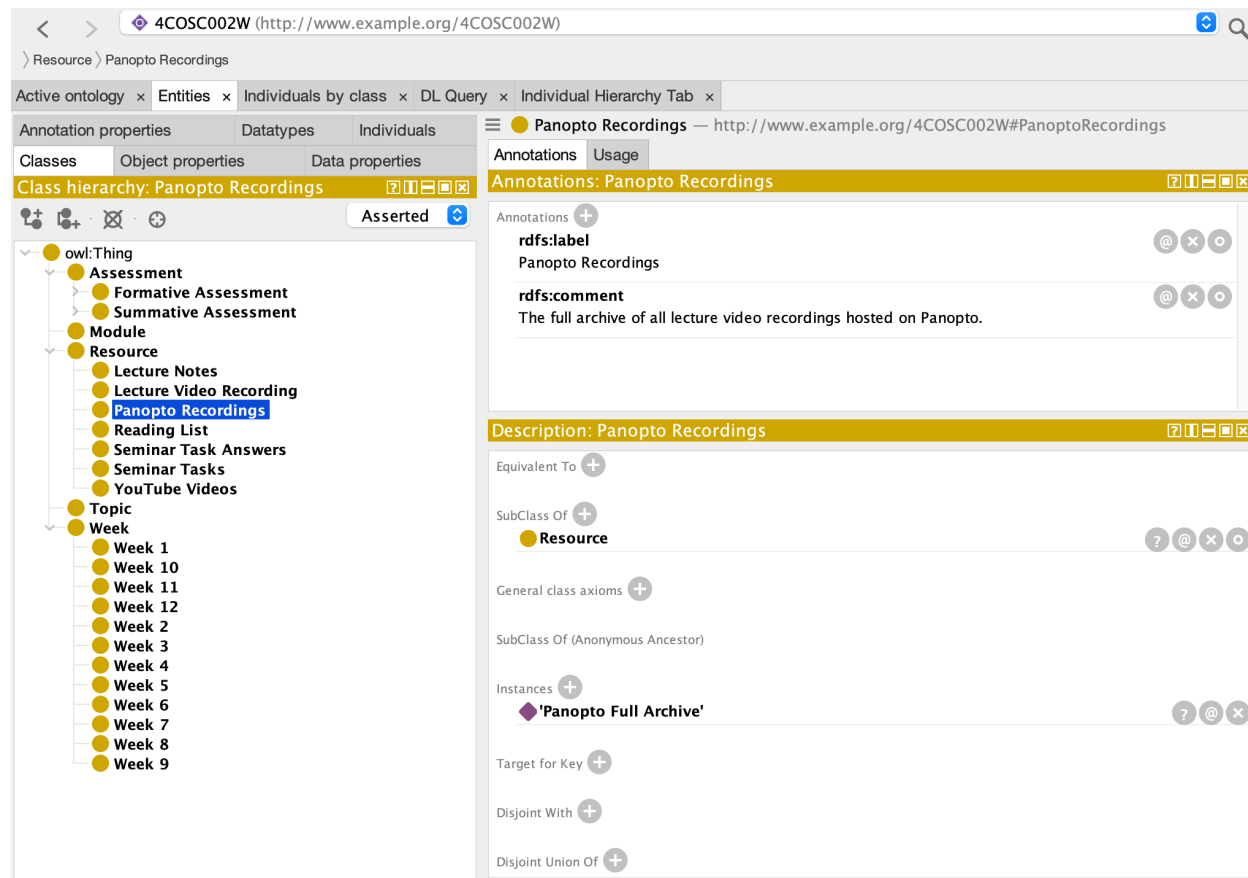


Figure 4.5.1. University Module Ontology displayed in Protégé, showing the principal module, week, topic, resource and assessment classes.

4.5.1 Class Structure

The ontology contains five principal domain classes:

- **Module**
- **Week**
- **Topic**
- **Resource**
- **Assessment**

These classes represent the main organisational components of a taught module. The **Module** class identifies the overall unit of study; **Week** represents its temporal structure; **Topic** represents the subject content; **Resource** covers materials made available to learners; and **Assessment** represents formative and summative evaluation activities.

The assessment hierarchy distinguishes between **FormativeAssessment** and **SummativeAssessment**. **MockTest** is modelled as a subclass of formative assessment, while the summative branch contains **Coursework**, **Exam**, **InClassTest**, **LabPractical** and **Portfolio**. This hierarchy allows relationships and constraints defined for the general **Assessment** class to apply to all specialised assessment types.

The **Resource** hierarchy includes:

- **LectureNotes**
- **LectureVideoRecording**
- **PanoptoRecordings**
- **ReadingList**
- **SeminarTasks**
- **SeminarTaskAnswers**
- **YouTubeVideos**

The **Week** class is specialised through subclasses from **Week1** to **Week12**. This provides an explicit semester structure while retaining the broader classification of each weekly unit as a **Week**.

4.5.2 Object Properties

Object properties define the relationships between the ontology classes. The principal structural properties are:

Object property	Domain	Range	Purpose
hasWeek	Module	Week	Connects a module to its teaching weeks
isWeekOf	Week	Module	Inverse of hasWeek
hasTopic	Week	Topic	Associates a teaching week with its topics

hasResource	Week	Resource	Associates a teaching week with learning resources
hasAssessment	Module or Week	Assessment	Connects assessments to the module or relevant week
testsKnowledgeOf	Assessment	Topic	Identifies the topic evaluated by an assessment
isTestedBy	Topic	Assessment	Inverse of testsKnowledgeOf
preparesFor	Topic	Assessment	Indicates that study of a topic supports preparation for an assessment
preparedBy	Assessment	Topic	Inverse of preparesFor
reviewsTopic	Topic	Topic	Connects a revision topic to another topic that it reviews

The [reviewsTopic](#) relationship is declared transitive. This allows review connections to be propagated through a chain of topics where appropriate. Several earlier properties remain in the ontology but are marked as deprecated, preserving the development history without treating them as part of the current active model.

The explicit domains and ranges are particularly important for validation. For example, [testsKnowledgeOf](#) can connect an assessment only to a topic. If an extracted graph attempts to use this property between incompatible entity types, the assertion can be identified as non-conforming.

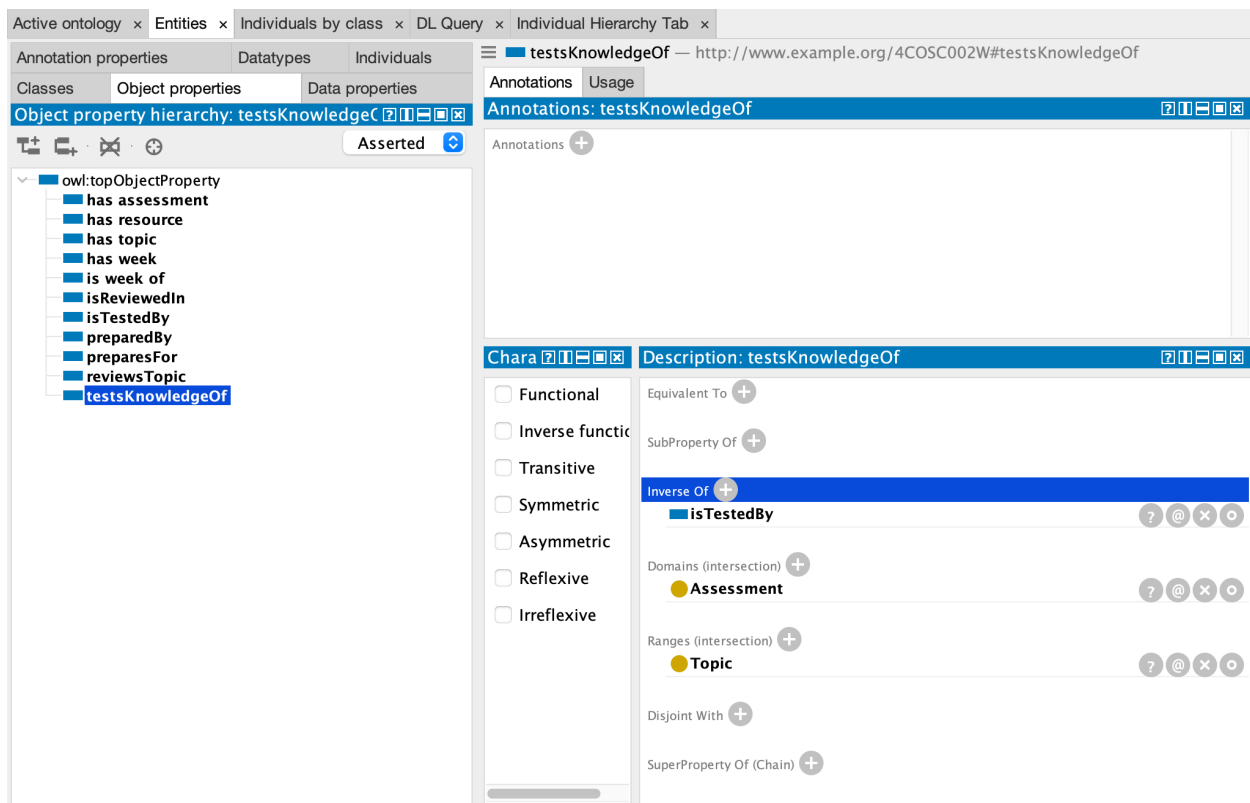


Figure 4.5.2. Protégé Object properties tab showing active properties and inverse-property relationships. Definition of the `testsKnowledgeOf` object property, with `Assessment` specified as its domain and `Topic` as its range.

4.5.3 Data Properties

The ontology defines six data properties for representing literal module information:

Data property	Domain	Data type	Purpose
<code>moduleCode</code>	Module	string	Stores the module code
<code>academicYear</code>	Module	string	Stores the academic year
<code>weekNumber</code>	Week	integer	Identifies the teaching-week number
<code>weightingPercentage</code>	Assessment	decimal or numeric value	Records assessment weighting
<code>assessmentMonth</code>	Assessment	string	Records the scheduled assessment month
<code>isSummative</code>	Assessment	boolean	Distinguishes summative from non-summative assessments

These properties separate literal metadata from semantic relationships. For example, an assessment weighting is represented as a value attached to an assessment, whereas the topic tested by that assessment is represented through an object property.

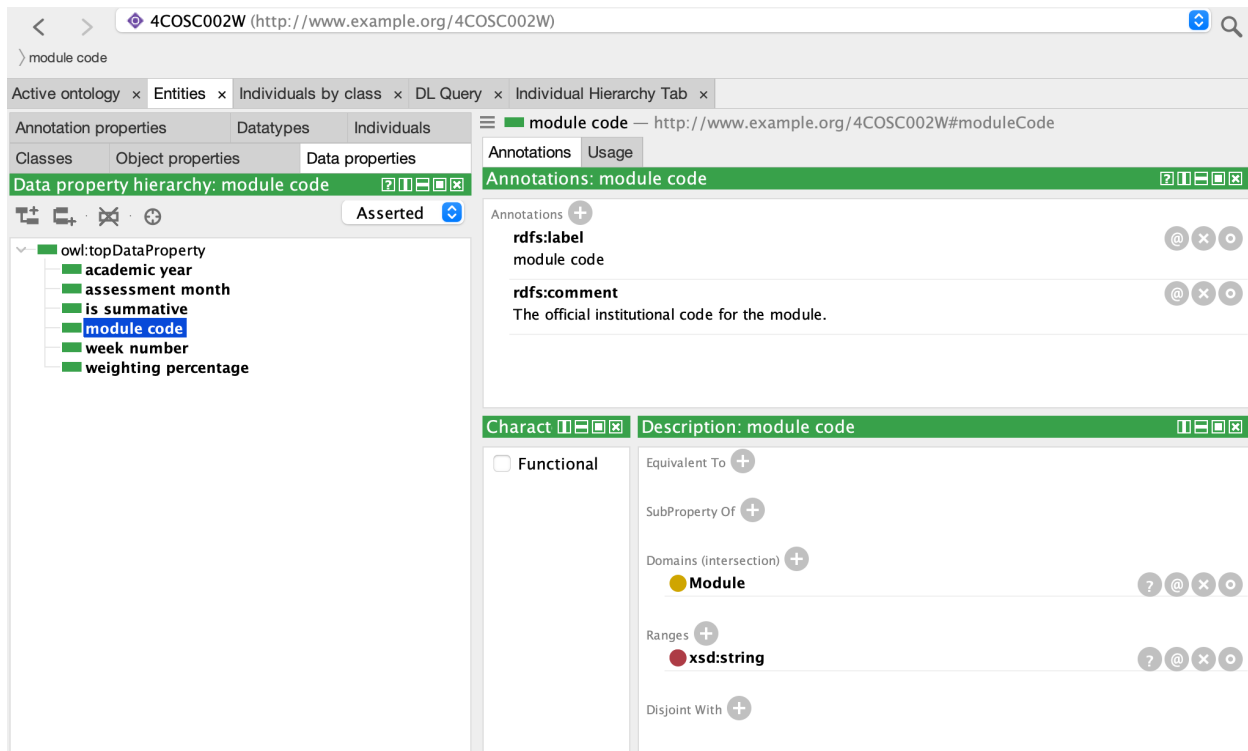


Figure 4.5.3. Data properties used to represent module, week and assessment metadata.

4.5.4 Populated Mathematics for Computing Case Study

The ontology contains a populated representation of the Level 4 Mathematics for Computing module. The module Individual **MathematicsForComputing** is assigned the code **4COSC002W.1** and the academic-year value **2024**. It is connected to twelve week individuals through **hasWeek** and to the final examination through **hasAssessment**.

The weekly individuals are linked to topics corresponding to the module structure. Examples include:

- ◆ Week 1: module introduction, number theory and sequences;
- ◆ Week 2: intervals and set theory;
- ◆ Week 3: functions and relations;
- ◆ Week 4: logic;
- ◆ Week 5: graph theory;
- ◆ Week 7: matrices, Part 1;
- ◆ Week 8: matrices, Part 2;
- ◆ Week 9: probability;
- ◆ Week 10: statistics fundamentals;
- ◆ Week 11: statistical applications;
- ◆ Week 12: examination revision.

Week 6 is retained within the twelve-week structure but does not have a disciplinary topic assigned in the current ontology, reflecting its role in the teaching schedule rather than inventing

subject content.

The populated ontology also contains assessment individuals for the final examination, in-class test and mock tests in Weeks 7 and 11. Learning-resource individuals include weekly lecture notes, lecture recordings, seminar tasks and seminar answers, together with module-level resources such as the reading list, Panopto archive and YouTube materials.

The screenshot displays an ontology editor interface with the following components:

- Top Bar:** Active ontology x Entities x Individuals by class x DL Query x Individual Hierarchy Tab x
- Left Panel (Individuals):**
 - Annotation properties
 - Datatypes
 - Individuals
 - Classes
 - Object properties
 - Data properties
 - Individuals: Mathematics for Computing
 - Exam Revision
 - Final Examination
 - Functions
 - Graph Theory
 - In-Class Test (ICT)
 - Intervals
 - Logics
 - Mathematics for Computing**
 - Matrices Part 1
 - Matrices Part 2
 - Mock Test (Week 11)
 - Mock Test (Week 7)
 - Module Introduction
 - Number Theory
 - Panopto Full Archive
 - Probability
 - Reading List
 - Relations
 - Seminar Task Answers
 - Sequences
 - Set Theory
 - Statistical Application
 - Statistics Basics
 - Week 1
 - Week 1 Lecture Notes
 - Week 1 Lecture Video Recording
 - Week 1 Seminar Task Answers
 - Week 1 Seminar Tasks
 - Week 10
- Right Panel (Usage: Mathematics for Computing):**
 - Show: ☒ this ☒ different
 - Found 35 uses of 'Mathematics for Computing'
 - Mathematics for Computing
 - Mathematics for Computing 'has week' 'Week 10'
 - Mathematics for Computing 'module code' '4COSC002W.1'
 - Mathematics for Computing 'rdfs:label' 'Mathematics for Computing'
 - Mathematics for Computing 'has week' 'Week 12'
 - Mathematics for Computing 'has week' 'Week 11'
 - Mathematics for Computing 'has week' 'Week 3'
 - Mathematics for Computing 'has week' 'Week 2'
- Bottom Panel (Description: Mathematics for Computing):**
 - Types: Module
 - Same Individual As: +
 - Different Individuals: +
 - Property assertions: Mathematics for Computing
 - 'has week' 'Week 9'
 - 'has week' 'Week 8'
 - 'has week' 'Week 7'
 - 'has week' 'Week 6'
 - 'has week' 'Week 5'
 - 'has week' 'Week 4'
 - 'has assessment' 'Final Examination'
 - Data property assertions:
 - 'module code'
 - 'academic year' '2024'
 - Negative object property assertions: +
 - Negative data property assertions: +

Figure 4.5.4. Mathematics for Computing module individual and its asserted module metadata, teaching weeks and assessment relationships.

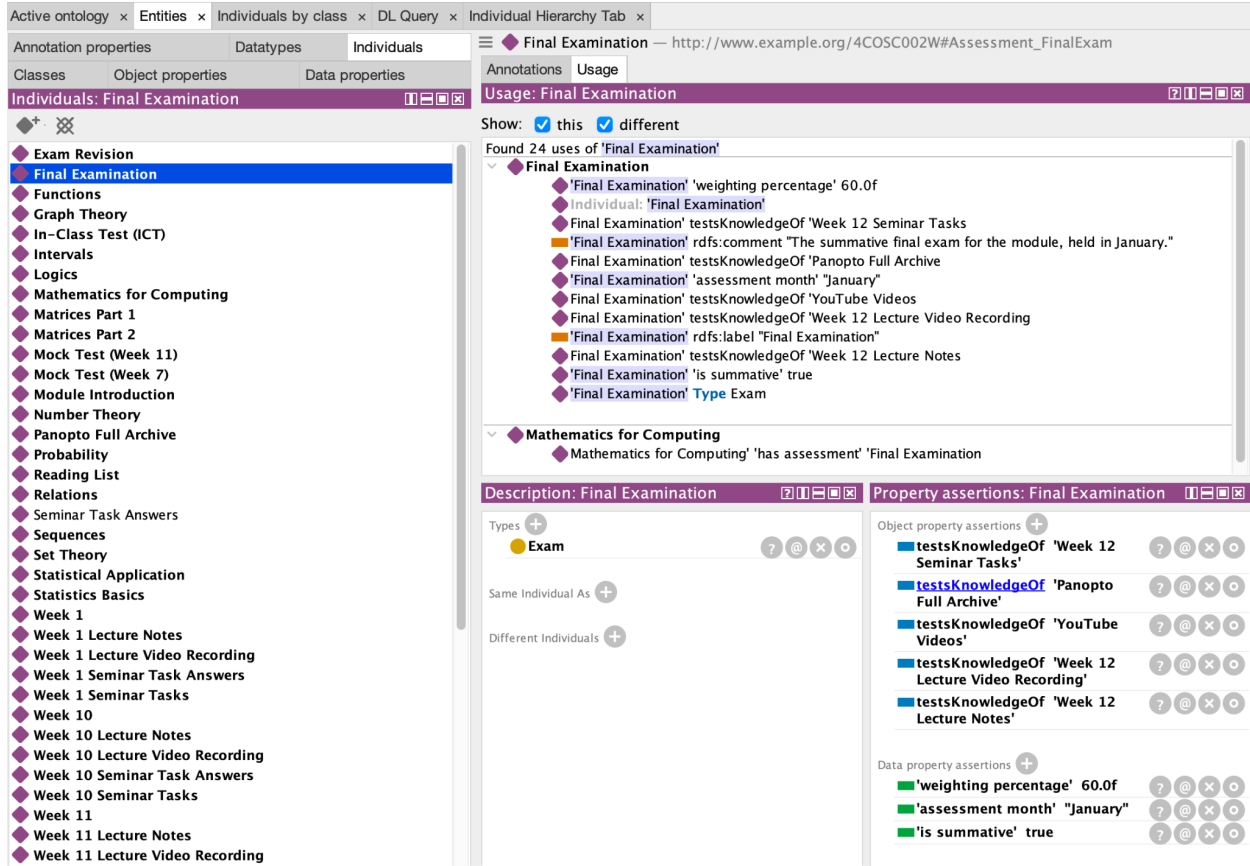


Figure 4.5.5. Example assessment individual and its semantic connections within the module ontology.

4.5.5 Reasoning and Consistency Checking

The ontology was checked in Protégé using the **HermiT reasoner**. Reasoning was applied to verify the consistency of the class hierarchy, domain and range declarations, inverse properties and populated assertions. No unsatisfiable classes or logical inconsistencies were reported during the final check.

Temporary proof-of-concept assertions were used during development to test newly introduced relationships. For example, an assessment-to-topic assertion was created to confirm the behaviour of `testsKnowledgeOf`. Provisional assertions were removed where the corresponding curriculum data had not yet been formally confirmed, preventing demonstration data from being retained as authoritative module information.

4.5.6 Role in the Implemented Pipeline

The OWL file is consumed by the knowledge-graph pipeline as the formal reference representation of the module structure. It supplies known curriculum entities, including topics, weeks and assessments, and provides the semantic model against which extracted graph elements can be compared. In the implemented workflow, the ontology supports:

- ◆ restriction of LLM outputs to recognised educational categories;
- ◆ checking of generated entity and relationship types;
- ◆ verification of domain and range compatibility;

- ◆ comparison of extracted topics with the expert-curated module structure;
- ◆ and production of validation findings before graph content is treated as suitable for academic review.

The ontology is therefore not simply a visual classification produced in Protégé. It constitutes an executable knowledge-representation artefact that links curriculum modelling, automated extraction and validation. Its implementation demonstrates how an academically informed module structure can be formalised for reuse within an AI-assisted concept-map generation pipeline.

4.6 Reproducibility and Deployment

The prototype was developed using reproducible Python environments and version-controlled through a shared project repository. Configuration files define model selection, extraction schemas and database connections, enabling experiments to be repeated across different datasets and LLM configurations. The complete source code for both O2 pipeline prototypes, together with their dependency specifications, ontology and SHACL validation artefacts and test data, is archived in Appendix D. While the current implementation operates as a research prototype, the architecture was designed to support future deployment within the SMARTTEST platform and integration with other learning platforms. By combining LLM-based extraction, graph storage, theme discovery, retrieval-augmented conversational support and ontology-driven validation, the implementation demonstrates the technical feasibility of automating educational knowledge-graph creation while maintaining academic oversight.

5. Evaluation and Testing

This chapter reports how the WENDY prototypes were evaluated and what the testing showed. The evaluation combines quantitative benchmarking with qualitative inspection: comparative benchmarking of candidate conversational models against defined response criteria (O1); ontology-conformance and graph-quality measurement of the knowledge-graph generation pipelines across multiple LLM configurations (O2); clustering-quality metrics over public benchmark corpora (O3); and formal reasoning and constraint-based validation of the educational ontology and generated graphs (O4). All benchmark runs were recorded in a shared testing workbook, archived with the row-level outputs in Appendix C.

5.1 Evaluation Methodology

5.1.1 Benchmarking Framework and Test Scenarios

Benchmarking across the work packages was coordinated through a shared framework defined in the project's LLM benchmark workbook (Appendix C). Five test scenarios were defined, one or more per work package, so that each pipeline was exercised on a realistic task with declared

input material and success criteria (Table 5.1.1). For every run, the workbook records the model and provider, run number, processing time, input and output tokens, API cost, output validity, an output-quality rating on a 1–5 scale, the execution environment, the tester and date, and — for the graph-generation runs — Topic F1 and graph size. A usage budget of £400 was allocated for API-based testing; the recorded spend across all runs was £0.31, reflecting the project's deliberate emphasis on low-cost and locally hosted configurations.

Scenario	Work package	Task	Input material	Success criteria
S1	O2	Generate a concept map (entities and relationships) from a single lecture	Mathematics for Computing — one lecture (~2,000 words)	Valid JSON knowledge graph; key concepts captured
S2	O2	Generate a concept map from a full module	Mathematics for Computing — full module (~15,000 words)	Valid graph; coherent cross-topic links
S3	O3	Extract overarching themes / topic clusters	Big Data module — lecture set (~10,000 words)	Sensible themes; prerequisite ordering
S4	O4	Ontology-based validation of a generated graph	O2 output graph + module ontology	Inconsistencies flagged; pass/fail per constraint
S5	O1	Answer a student guidance query over a stored graph	Existing stored graph (1 query)	Correct, grounded answer; acceptable latency

Table 5.1.1. Benchmark test scenarios defined in the shared testing workbook (Appendix C).

5.1.2 O1: Conversational Response Benchmarking

The O1 evaluation compares four candidate conversational models — Llama 3 (locally hosted), gemini-3.6-flash, gpt-5-mini and claude-sonnet-5 — answering the same set of fifteen student-facing queries over the same JSON-encoded Mathematics for Computing knowledge base (Section 4.2). The query set spans the assistant's intended use cases: module-content overview, module administration, assessment preparation, concept explanation (number theory, set theory), practice-question generation, revision guidance, and cross-topic connections between mathematical concepts and computing. Responses were assessed against the criteria defined in Section 3.4.3 — correctness, completeness and communication — with hallucination and faithfulness to the supplied module material examined explicitly. Response times were logged programmatically for the locally executed Llama 3 runs; the complete row-level outputs (query, model, response and timing) are archived in Appendix C (O1-TestingResults-1.csv and O1-TestingResults-2.txt).

5.1.3 O2: Ontology Conformance and Graph Quality

The O2.1 pipeline evaluation framework operates at two complementary levels, summarised in Table 5.1.2. The first level measures ontology conformance: alignment between the extracted graph and the ground truth encoded in the O4 team's OWL ontology (ontology_v3_updated.owl),

which formally specifies valid topic names, week identifiers, and week-to-topic mappings for the 4COSC002W module. Five metrics are computed: Topic Precision (valid extracted topics / all extracted topics), Topic Recall (valid extracted topics / all ontology topics), Topic F1 (harmonic mean of the two), Week Coverage (correctly named weeks / all ontology weeks), and Week-Topic Alignment (correct week-topic pairs / all ontology week-topic pairs). Metric computation is fully automated: a validation step parses the OWL file using Python's `xml.etree.ElementTree`, queries Neo4j AuraDB via Cypher, and compares both sets, producing a structured conformance report at the end of each run.

The second level covers graph quality metrics that assess internal pipeline consistency independently of the ontology, also listed in Table 5.1.2. These include Concept Anchor Rate (proportion of CONCEPT nodes linked to a WEEK via INTRODUCED_IN), Definition Coverage (proportion of concepts with at least one HAS_DEFINITION relationship), Relationship Diversity (number of distinct schema relationship types present out of nine defined), Duplicate Concept count, and Noise Rate (proportion of concept names exceeding 50 characters, flagging verbatim slide text in place of genuine concepts). To isolate the effect of model capability, all pipeline components were held constant across runs — text extraction, preprocessing filters, extraction schema, prompt wording, and post-processing steps — with only the LLM substituted. The Llama 3 8B local model run, covering eight lectures, serves as the quantitative baseline.

Metric	Definition	Level
Topic Precision	Valid extracted topics / all extracted topics	Ontology conformance
Topic Recall	Valid extracted topics / all ontology topics	Ontology conformance
Topic F1	Harmonic mean of Topic Precision and Topic Recall	Ontology conformance
Week Coverage	Correctly named weeks extracted / all ontology weeks	Ontology conformance
Week-Topic Alignment	Correct week-topic pairs / all ontology week-topic pairs	Ontology conformance
Concept Anchor Rate	CONCEPT nodes linked to a WEEK via INTRODUCED_IN / total CONCEPT nodes	Graph quality
Definition Coverage	CONCEPT nodes with at least one HAS_DEFINITION / total CONCEPT nodes	Graph quality
Relationship Diversity	Distinct schema relationship types in graph / 9 defined types	Graph quality
Duplicate Concepts	Count of CONCEPT nodes with identical names	Graph quality
Noise Rate	CONCEPT nodes with name length > 50 characters / total CONCEPT nodes	Graph quality

Table 5.1.2. O2.1 pipeline evaluation metrics reference.

The O2.2 pipeline is evaluated through its per-run construction statistics — chunks validated and saved, chunks rejected by the SHACL gate, empty or noisy chunks skipped, database write failures and node/relationship write attempts — together with manual inspection of the resulting graph structures. Because SHACL validation is applied before storage, the pass rate of the validation gate itself serves as a quality signal for the extraction and filtering stages.

5.1.4 O3: Clustering Quality Metrics

The O3 experiments were assessed with several complementary metrics. Adjusted Rand Index (ARI) and Normalised Mutual Information (NMI) measure the agreement between generated clusters and existing labels while accounting for chance agreement, with ARI providing the stricter evaluation. The Silhouette Score evaluates the compactness and separation of clusters based only on the embedding space, without reference to external labels. Accuracy was calculated by assigning each cluster to its most frequent label using majority voting, and Coverage records the proportion of items assigned to any cluster, since density-based methods may classify items as noise. Accuracy calculated on partial coverage is therefore not directly comparable with accuracy from methods that assign all items — a distinction that matters when interpreting the results in Section 5.2.3. Precision, recall, F1-score and confusion matrices were reported per category where ground-truth labels permitted.

5.1.5 O4: Reasoning and Constraint-Based Validation

The O4 validation framework was evaluated in two ways. First, the ontology itself was verified in Protégé using the HermiT reasoner, checking the consistency of the class hierarchy, domain and range declarations, inverse properties and the populated Mathematics for Computing assertions. Second, the framework's operational effectiveness was assessed through its integration with the two O2 pipelines: the conformance metrics of Table 5.1.2 measure how closely generated graphs align with the ontology ground truth (O2.1), while the SHACL shapes measure whether candidate graph fragments satisfy the formal structural constraints before storage (O2.2).

5.2 Results by Objective

5.2.1 O1: Conversational Assistant Benchmarking Results

Across the fifteen-query test set, the locally hosted Llama 3 model returned responses in 1.15–5.68 seconds (mean 2.33 s, median 1.54 s, 34.9 s in total); as the other three models ran through cloud APIs, their latencies are not directly comparable and timing conclusions are restricted to the local configuration. The row-level responses for every query are archived in Appendix C.

Llama 3 was the quickest in response time but suffered from hallucinations and misinterpretation of the input, and its capabilities are limited compared with the other candidates. Its "straight to the point" approach offers some advantages — it does not overload the user with information — but the lack of contextual detail and its preference for broader responses limit its usefulness. Its summary of module content, while accurate in outline (set theory, number theory, functions, matrices, vectors, graph theory, probabilistic and statistical reasoning), lacked the detail offered by the other models, and in one case it inferred a student's performance from the student's name alone — a clear hallucination where acknowledging insufficient information would have been appropriate.

gpt-5-mini produced the most comprehensive responses: well structured, detailed, and consistently adapted to the student's profile, achieving the personalised learning-path use case, with proactive follow-up suggestions. In its strongest examples it connected topic explanations directly to the module learning outcomes and explained why the student should prioritise them. The trade-offs are the longest generation times of the group and a large output-token footprint, and occasional over-claiming — in one query it cited specific lecture and slide references while simultaneously admitting the material contained no explicit entry, a risky behaviour for a source-grounded assistant.

gemini-3.6-flash communicated excellently and formatted mathematical content well, but was sometimes overly conservative and missed information, scoring lower on completeness. When asked for practice questions it declined because none were present in the module material — where the other models generated questions while flagging that the source material contained none. Arguably this restraint is a desirable trait for academic rigour: it demonstrates stronger adherence to source restrictions than the other models and consistent faithfulness to the supplied material.

claude-sonnet-5 showed capability similar to gpt-5-mini and prioritised factual grounding, but was often more conservative, occasionally refusing tasks that could have been completed safely (such as generating general practice questions) through strict interpretation of the query. In the number-theory explanation task it did not connect the topic to the module content, instead referring to the student profile in a way that did not answer the query. One run failed with a response-parsing error in the comparator script (an extended-thinking content block preceding the text block); this is an integration issue rather than a model capability failure.

Conclusions. According to the criteria, gpt-5-mini performed best for correctness, completeness and communication, consistently producing accurate responses. Gemini consistently demonstrated faithfulness to the source material, providing concise, accurate responses while avoiding unsupported claims, and emerged as one of the strongest models for source-grounded education; with further prompt and input-data engineering it is a strong candidate where ethical guardrails are the priority. Claude was often more trustworthy than GPT in its grounding but more conservative in refusals. Llama showed the greatest tendency to introduce unsupported information, making it the least reliable of the four, though cost-effective and fast when run locally. An ideal assistant would combine GPT-level completeness with Gemini-level faithfulness, and would be capable of generating practice questions grounded in facts from the knowledge base.

5.2.2 O2: Knowledge Graph Generation Results

O2.1 Benchmark Progression

The benchmark workbook (Appendix C) records the full model progression for the O2.1 pipeline across seven test runs. Two small local models failed the single-lecture scenario S1: phi3:mini-4k produced schema collapse on all inputs (no valid output), and phi3:14b produced marginally better but still unusable output at an impractical ~45 minutes per lecture. Llama 3 8B achieved

the first valid extraction (94 nodes and 103 relationships from a single lecture), and on the full-module scenario S2 initially collapsed on Lectures 7 and 10 before a revised prompt produced the eight-lecture baseline run. The two API models then processed the full ten-lecture set without failure. Total recorded API spend across all runs was £0.31 of the £400 budget.

Run	Model	Scenario	Time	Cost	Valid output	Quality (1–5)	Topic F1	Nodes / Relationships
T1	phi3:mini-4k (local)	S1	8 min	£0	No — schema collapse	1	—	0 / 0
T2	phi3:14b (local)	S1	45 min	£0	Partial	1	—	10 / 6
T3	Llama 3 8B (local)	S1	6 min	£0	Yes	2	—	94 / 103
T4	Llama 3 8B (local)	S2	30 min	£0	No — collapse on L7, L10	2	—	—
T5	Llama 3 8B (local, revised prompt)	S2 (8 lectures)	31 min	£0	Yes	3	32%	414 / 445
T6	gpt-4o-mini (API)	S2 (10 lectures)	5 min	~£0.03	Yes	4	64.0%	551 / 578
T7	gpt-4o (API)	S2 (10 lectures)	4 min 35 s	~£0.28	Yes	4	60.9%	698 / 746

Table 5.2.1. O2.1 benchmark run log, summarised from the testing workbook in Appendix C.

O2.1 Cross-Model Evaluation

Table 5.2.2 presents the cross-model evaluation results for the three successful full-pipeline configurations: Llama 3 8B running locally via Ollama over eight lectures, and OpenAI's gpt-4o-mini and gpt-4o each running via the API over ten lectures. The Llama run covers eight weeks because Lectures 7 and 10 caused schema collapse and had to be excluded, while both API models processed the full set of ten lectures without failure. All other pipeline components remained identical across the three runs to ensure the model was the only variable under comparison.

Metric	Llama 3 8B (8 weeks)	gpt-4o-mini (10 weeks)	gpt-4o (10 weeks)
Total nodes	414	551	698
Total relationships	445	578	746
CONCEPT nodes	176	229	210
EXAMPLE nodes	3	45	182
FORMULA nodes	3	18	44
Topic Precision	40%	80%	87.5%
Topic Recall	26.7%	53.3%	46.7%
Topic F1	32%	64.0%	60.9%
Week-topic alignment	30%	42.9%	50.0%
Week coverage	66.7%	83.3%	83.3%
Definition coverage	96%	77.3%	75.7%
Schema collapse	Lectures 7, 10	None	None
Runtime	~30 min	~5 min	~4 min 35 s
Estimated cost per run	£0 (local)	~£0.02	~£0.28

Table 5.2.2. O2.1 pipeline cross-model evaluation results (Llama 3 8B, gpt-4o-mini, gpt-4o).

Graph size increases progressively from Llama 3 8B (414 nodes, 445 relationships) to gpt-4o-mini (551, 578) and gpt-4o (698, 746). The most notable differences are in EXAMPLE and FORMULA extraction: Llama 3 8B produced only 3 of each, while gpt-4o-mini produced 45 EXAMPLE and 18 FORMULA nodes, and gpt-4o produced 182 and 44 respectively — a clear progression in the models' ability to extract structured content beyond basic concept definitions. On ontology conformance, gpt-4o-mini produced the strongest Topic F1 at 64.0%, against 32% for Llama 3 8B and 60.9% for gpt-4o. Notably, gpt-4o achieved higher Topic Precision (87.5% versus 80%) but lower Topic Recall (46.7% versus 53.3%): it extracted fewer topics overall, but those it extracted were more likely to be valid — more selective, but not more complete. Week-topic alignment improved across models from 30% to 42.9% to 50.0%, reflecting progressively better anchoring of topics to the correct weeks.

Runtime and cost present a clear argument in favour of gpt-4o-mini: Llama 3 8B required approximately 30 minutes per run at no API cost, gpt-4o-mini reduced runtime to around 5 minutes at ~£0.02 per run, and gpt-4o was marginally faster but roughly fourteen times more expensive while delivering lower Topic F1. Taking conformance, runtime and cost together, gpt-4o-mini represents the strongest configuration for the O2.1 pipeline at this stage of development — consistent with the conclusion in Section 4.3.3 that O2.1 (in its gpt-4o-mini configuration) is the preferable pipeline for further research. Topic names are constrained at extraction time by a fixed allowlist drawn from the ontology; however, prompt-level constraints alone proved insufficient to guarantee exact string matching, as the model continues to paraphrase, over-specify or omit topics — the primary driver of the Topic F1 gap across all runs.

O2.2 Construction and Validation Results

The O2.2 pipeline's final documented run processed two Mathematics for Computing lecture decks (Weeks 1 and 2). Table 5.2.3 summarises the construction statistics. The SHACL gate operated as designed: 89.8% of chunks passed validation and were stored, while the three rejected chunks were all cases in which filtering had collapsed a Topic and a Concept to the same name (for example Modular Arithmetic, Golden Ratio and Subset) — precisely the malformed structures the validate-before-write design is intended to intercept. The strongest outputs were hierarchical topic structures for number systems, sequences, set operations, set algebra and intervals, with notation retained as Concept properties rather than proliferating into separate low-value nodes. The natural-language question-answering script correctly translated schema-conformant questions (topics per week, concepts per topic, week counts) into valid Cypher and returned grounded answers, with the generated query and database context visible for inspection.

Metric	Result
Lecture slide decks processed	2 (Weeks 1 and 2)
Cleaned documents	49
Text chunks created	49 (23 Week 1; 26 Week 2)
Validated and saved chunks	44 (89.8%)
Chunks rejected by SHACL gate	3 (6.1%)
Empty/noisy chunks skipped	2 (4.1%)
Neo4j write failures	0
Node write attempts	228
Relationship write attempts	187

Table 5.2.3. O2.2 final construction run statistics (two lecture decks, Llama 3 via Ollama).

5.2.3 O3: Theme Discovery Experimental Results

Five experiments were conducted. The first four applied the clustering pipeline to complete documents with increasing levels of difficulty, establishing a baseline with BERTopic; the fifth applied the same approach at sentence level and produced the project's core negative result. Table 5.2.4 summarises the headline outcomes.

Experiment	Dataset and unit	Key results
1. Sanity check	BBC News, 2,225 whole documents	K-Means ARI 0.90; BERTopic accuracy 95% (94% after outlier reassignment); HDBSCAN without UMAP rejected 53% as noise
2. Multi-label validation	Reuters-21578, top 10 categories, 9,034 documents	BERTopic 91% accuracy on assigned documents (33.5% outliers); UMAP+HDBSCAN ARI 0.44; K-Means 82% but four minority categories at F1 = 0; raw HDBSCAN collapsed to two clusters (ARI 0.0009)
3. Negative control	RCV1-v2, TF-IDF vectors only (no raw text), SVD to 200 dims	All methods degraded; Silhouette–ARI paradox: UMAP+HDBSCAN Silhouette 0.5756 with ARI 0.0491 — clusters were meaningful sub-topics not represented by coarse labels
4. Section-level units	WikiSection en_disease (25,033 sections) and en_city (120,140 sections)	Same pipeline, different taxonomy: K-Means ARI 0.0733 (disease) vs 0.2895 (city); BERTopic 0.69 accuracy on en_city excluding outliers; taxonomy separability, not the method, drove the fourfold difference
5. Sentence level (core result)	20 Newsgroups, 2,000 documents segmented into 29,299 sentences	41.1% of sentences rejected as outliers; 471 topics from 20 categories; 52.4% accuracy on assigned sentences; document-level majority voting 47.5%; accuracy rose with sentence length from 28.1% (shortest) to 72.5% (>50 words)

Table 5.2.4. O3 experiment summary across the five evaluation corpora.

The experiments establish three conclusions. First, the embedding-and-clustering stack performs strongly on whole-document and section-level units — up to 95% accuracy and ARI 0.90 on clean corpora — so weaker results elsewhere are attributable to dataset characteristics and unit size rather than the representation. Second, applying UMAP before density-based clustering is essential: without it, HDBSCAN rejected over half of even a well-separated corpus. Third,

individual sentences are generally too small a unit for reliable topic assignment: the high outlier rate, severe over-fragmentation and the consistent monotonic relationship between sentence length and accuracy show that errors are systematic rather than random, which is why document-level aggregation failed to recover the lost information (47.5% versus 44.4% at sentence level). Performance improved continuously with available context, motivating the recommendation — carried into the WENDY methodology — that theme discovery should operate on units of approximately 50–150 words (paragraphs, sections or sliding windows), a structure matched by WikiSection's sections (average 153 words). The dataset profiling additionally concluded that no available public corpus combines multi-label ground truth, sub-document structure and educational content, and that a purpose-built educational corpus of 200–500 paragraph-annotated documents would provide a stronger evaluation framework — a concrete recommendation for future work.

5.2.4 O4: Validation Framework Results

Reasoning over the University Module Ontology (version 4.0) with HermiT reported no unsatisfiable classes and no logical inconsistencies in the final check, verifying the class hierarchy, domain and range declarations, inverse properties and the populated Mathematics for Computing case study. Operationally, the framework demonstrated its purpose through both O2 pipelines. In O2.1, the ontology-derived conformance metrics quantified extraction quality and model differences (Table 5.2.2), and the automated conformance report identified specific failure modes such as paraphrased topic names and misaligned week-topic pairs. In O2.2, the SHACL shapes rejected 6.1% of candidate chunks before storage, in each case correctly identifying a structurally malformed fragment (Table 5.2.3). Together these results show that the hybrid AI–human validation design functions as intended at its automated first stage: formal artefacts (OWL and SHACL) detect structural and conformance faults automatically, narrowing the space of issues that require academic review.

5.3 Case Study

The Level 4 Mathematics for Computing module (4COSC002W) served as the principal case study across the work packages: its lecture materials were the extraction source for both O2 pipelines, its academically reviewed curriculum structure was formalised in the O4 ontology, and its JSON-encoded knowledge base grounded the O1 conversational benchmarking. This gave the evaluation a single consistent reference point — an existing, expert-curated module against which generated structures and assistant responses could be interpreted. The O3 experiments deliberately used public benchmark corpora (BBC News, Reuters-21578, RCV1-v2, WikiSection, 20 Newsgroups) rather than module materials, because clustering evaluation requires ground-truth labels at a scale no single module provides. A second scenario using a Big Data module lecture set was defined in the benchmark framework (Table 5.1.1, S3) as the intended bridge from public corpora back to Westminster teaching material, and extending the case-study base to further modules and disciplines — the test of the architecture's subject independence — is identified as future work in Chapter 8.

5.4 Summary of Evaluation Findings

The evaluation supports four headline findings. For the conversational assistant (O1), gpt-5-mini scored highest against the response criteria, while Gemini's stricter source-grounding makes it a strong candidate where faithfulness outweighs completeness; locally hosted Llama 3 was fast and free but least reliable. For knowledge-graph generation (O2), gpt-4o-mini in the O2.1 pipeline is the strongest configuration on the combined evidence of ontology conformance (Topic F1 64.0%), runtime and cost, and the O2.2 run demonstrated that formal validation before storage operates effectively at zero API cost — reinforcing the Chapter 4 conclusion that O2.1 is the preferable pipeline for further research, with O2.2's validation and querying techniques recommended for incorporation. For theme discovery (O3), the method is sound on adequately sized text units, and the sentence-level configuration is empirically ruled out in favour of 50–150-word units. For validation (O4), the ontology is formally consistent and both of its operational validation modes — post-hoc conformance scoring and pre-storage SHACL gating — detected real faults in generated graphs, demonstrating the feasibility of the project's hybrid AI–human validation approach. Constraints and open issues arising from these results are discussed in Chapter 8.

6. Team Work and Collaboration

6.1 Overview

This chapter gives a brief account of how the project team — introduced by work package in Chapter 1 — actually worked together over the course of the project, as a complement to the technical account given in Chapters 3-5. It focuses on the meeting rhythm that structured collaboration, on how decisions crossing work-package boundaries were made, and closes with a short reflection on team working as part of the Students-as-Researchers experience.

6.2 Meeting Rhythm and Governance

Collaboration was organised on two levels. Each work-package team held brief weekly meetings with its academic leads, at a fixed time, to review progress on the technical work described in Chapters 3-5, agree next steps and flag anything needing input from outside the team. Separately, the whole project met as a group roughly every three to five weeks in an all-groups meeting chaired by the Principal Investigator, in which each work package reported in turn, cross-cutting issues were raised and resolved, and the project's overall direction and deadlines were confirmed. In practice, because O1, O2 and O4 depend directly on one another's outputs, several of the weekly meetings were run jointly across those three work packages rather than as single-team sessions. All meetings, at both levels, were held on Microsoft Teams, whose built-in transcription and Copilot recap features were used to keep an accurate, shareable record of decisions for team members who could not attend a given session.

This rhythm — frequent, small-group contact with academic leads alongside periodic whole-team alignment — gave the project fast local decision-making within each work package while still keeping the four objectives coordinated. The schedule itself followed four broad phases, running in parallel across the four work packages before converging into shared integration and write-up phases (Table 6.2.1). Development was deliberately wound down in the final weeks so the team could concentrate on consolidating evidence and drafting the report ahead of the 31 July deadline.

Phase	Period	Focus
Design	Feb – Mar	Requirements, method design and ontology/rules design within each work package
Build and experiment	Mar – May	Prototype development and experimentation (O1 build; O2 extraction; O3 clustering; O4 checks)
Test and evaluate	May	Testing, tuning and evaluation within each work package
Integration and report writing	Late May – Jul	Cross-work-package integration and system demonstration, overlapping with drafting, then review and finalisation of the report

Table 6.2.1. Project phases, reconstructed from the weekly Gantt schedule.

6.3 Data and Code Management

Outside of meetings, two shared platforms underpinned day-to-day collaboration. Project documents — the shared report file and its supporting appendices, work-package folders, meeting transcripts, the Gantt schedule, the Work Package Allocation Table and timesheets — were kept in a single shared Google Drive project folder, organised by work package and document type, so that any team member could find and contribute to the current version of a document rather than working from emailed copies. All project code was kept in one shared GitHub repository rather than separate per-team repositories: the O2 pipelines were pushed there first, and the other work packages, including O1, were asked to add their own code to the same repository so that the codebase, and an informal record of who had contributed what, stayed in one place. Database credentials and API keys were deliberately kept out of version control, in a private environment file shared separately from the repository. Alongside its role as a code store, the GitHub repository also served as a lightweight point of reference during meetings — for example, the shared benchmark-testing results were linked from it — giving the technical work packages a second, asynchronous channel of coordination alongside the Teams meetings.

6.4 Working Across Work Packages

Because O1's assistant, O2's knowledge-graph pipelines and O4's validation framework form a single conceptual pipeline, several decisions had to be agreed jointly rather than made by one team alone. The most significant were: settling the scope of O1's output — reframed during discussion from a simple "chatbot" to a system-integrated "AI assistant" with access to the graph and database layers, which fixed its design brief; agreeing a dual-database architecture, with MongoDB retained for user accounts and conversational history and Neo4j adopted for graph-structured content, connected through JSON as a lightweight interchange format; and pooling shared infrastructure, including a shared code repository and a shared LLM API budget for benchmark testing across work packages. Support was also redirected across teams where useful — for example, a student's request for literature on chatbot architecture was answered by directing her to a colleague's existing reading list from his teaching role, rather than each team searching independently.

The team also had to manage a work package whose central experiment did not succeed: sentence-level topic modelling in O3 was found not to carry enough semantic context to identify coherent themes. This was reported to the wider team as a negative but legitimate research result rather than treated as a shortfall, and the team's response was to redirect O3's remaining effort towards documenting what had been tried, what had not worked, and why — the same framing this report gives the result in Chapters 5 and 8 — while also reducing the workload placed on the O3 research assistant, who was moving on to his own dissertation.

The report itself was produced through a similar collaborative process: each work package drafted into its own working document, a shared master document with an agreed structure was used to consolidate contributions, and the two PhD Research Officers took on an editorial role identifying

gaps and inconsistencies across the combined text ahead of a final academic review in the last week before submission. The consistency issues this restructuring exercise has needed to resolve across earlier chapters are a natural consequence of that distributed drafting process.

6.5 Pedagogical Reflection

Beyond its technical outputs, WENDY functioned as a structured research-training experience for the students involved, consistent with the aims of the Students-as-Researchers scheme. The project's team-working model gave students practice in several research skills that are not exercised by individual coursework: negotiating a technical interface with another team whose priorities differ from one's own, as in the O1/O2/O4 architecture discussions; taking part in evidence-based decision-making under time and budget constraints, as in the choice between LLM providers and the database architecture; and, in the case of O3, experiencing directly how a negative or null result is handled with integrity in a research setting — documented honestly rather than suppressed or reframed as success. The layered supervision model, in which PhD Research Officers provided day-to-day research mentoring within a work package while academic Work Package Leads and the Principal Investigator provided technical direction and cross-project coordination, also gave the postgraduate researchers themselves supervisory and editorial experience, particularly in the final consolidation of the report.

The collaborative drafting process was, in this respect, as pedagogically significant as the technical work: producing a single coherent report from four independently working teams required students to write for readers outside their own work package, to accept editorial revision of their material, and to reconcile their own findings with the project's shared conclusions — skills that mirror collaborative academic authorship more closely than most undergraduate or postgraduate project work does.

7. Ethics, Cybersecurity, Sustainability and Responsible AI

WENDY is designed for a setting in which errors have real consequences: an assistant that misinforms a student, a concept map that misrepresents a curriculum, or a data store that exposes learner information would each undermine the educational purpose the project serves. This chapter sets out the project's position and practice on four connected themes: the responsible-AI stance embedded in the system design (7.1), the ethical considerations specific to AI in higher education (7.2), data protection and privacy (7.3), cybersecurity practice and requirements (7.4), and the project's sustainability profile and contribution to the UN Sustainable Development Goals (7.5).

7.1 Responsible AI Stance: AI Supports, Not Replaces, Academic Judgement

The project's governing principle is that AI supports rather than replaces academic judgement. This is not a policy statement appended to the system; it is embedded in the architecture at three points. First, generation is grounded: the conversational assistant retrieves validated content from the knowledge graph and inserts it into the model prompt before responding, so that answers derive from approved module material rather than from the model's pre-trained knowledge alone (Section 4.2). Second, generated structures are validated before use: both O2 pipelines treat their output as candidate concept maps, checked against the O4 ontology — post-hoc conformance scoring in O2.1 and pre-storage SHACL gating in O2.2 — before any content is considered suitable for review (Sections 4.3 and 5.2.4). Third, a human remains in the loop: the hybrid AI–human validation framework (Section 3.3) positions automated checking as the first stage only, with academic review and approval required before generated structures reach students. Full academic-approval interfaces remain future work, but the workflow is designed so that no AI-generated educational content is presented to learners as authoritative without expert confirmation.

The evaluation results justify this caution empirically. The O1 benchmarking recorded a locally hosted model inferring a student's academic performance from the student's name alone, and an API model citing lecture and slide references that the source material did not establish (Section 5.2.1). These observed failure modes are precisely the behaviours the grounding-and-validation design exists to contain, and they inform the conclusion that model faithfulness to source material — demonstrated most consistently by Gemini in the benchmark — is a first-order selection criterion for an educational assistant, alongside accuracy and completeness.

7.2 Ethical Considerations for AI in Higher Education

Instead of guesswork, the conversational agent should be clear about its limits, in the sense that it can rely only on module content. To ensure academic integrity — a key concern of using conversational chatbots in higher education (Ganguly et al., 2025) — the chatbot should signpost or summarise module content and should avoid generating explanations or descriptions that could include inaccuracies or bias. The following ethical concerns, summarised by the aforementioned author, frame the project's ethical analysis:

- Academic misconduct capabilities;
- Inaccurate content generation;
- Privacy and data security;
- Algorithmic bias;
- Transparency;
- Risk of educational inequalities through differential access to AI-enhanced learning tools;
- Collection of sensitive student data;
- Concerns of overreliance on technology impacting metacognitive capabilities.

The project's design responds to each concern in concrete terms. Academic misconduct and inaccurate content are addressed through source-grounding and the restriction of the assistant to validated module material: the benchmark showed that a well-constrained model will decline to produce content (such as practice questions) that the approved material cannot support, and the project regards this restraint as a desirable property for academic rigour rather than a defect (Section 5.2.1). Algorithmic bias is mitigated structurally, because the knowledge the assistant draws on is an expert-curated curriculum formalised in the ontology rather than open web content, and any residual model bias is bounded by the validation layer. Transparency is served by design decisions made in both pipelines: the O2.2 question-answering prototype deliberately displays the generated Cypher query and the raw database records alongside every answer, making the evidence for each response inspectable, and the O2.1 pipeline exposes its intermediate outputs rather than concealing the process behind a single automated interface (Chapter 4). Educational inequality and overreliance are governance matters as much as technical ones: WENDY is conceived as a module-integrated tool available to every enrolled student rather than a premium add-on, and its purpose is to help students navigate and connect validated curriculum content — supporting metacognition through concept maps and learning pathways — rather than to complete assessed work on their behalf.

7.3 Data Protection, Privacy and Consent

The project processed no live student data. All experimentation used teaching materials (lecture slides and module documents for Mathematics for Computing) that are institutional content, public benchmark corpora for the O3 experiments, and synthetic test profiles for the O1 benchmarking — the student profile used in the benchmark queries is fictitious test data, not a real learner record. The user-data layer envisaged in the broader architecture (MongoDB storage of profiles, conversation histories and personalisation data) was deliberately not implemented during the project window (Section 4.1), which means the prototypes as delivered hold no personal data at

all.

This boundary is also a statement of intent for future phases. A deployed WENDY would necessarily process personal data — conversation histories, performance signals, personalised learning paths — and would therefore require, before any live pilot: a lawful basis and informed consent for each category of learner data; data minimisation, so that personalisation stores only what the pedagogical function requires; clear retention limits for conversational data; and institutional data-protection review under UK GDPR ahead of integration with SMARTTEST and university authentication. Designing the personalisation layer around these requirements from the outset — rather than retrofitting them — is recorded as a condition for the platform-integration work outlined in Chapter 8.

7.4 Cybersecurity

The project applied standard secure-development practice throughout. Service credentials and database connection information — the Neo4j AuraDB URI, usernames, passwords and API keys — were stored in environment variables and .env files excluded from version control, and were never embedded in source code or committed to the shared repository (Section 4.1); the benchmark workbook likewise records an explicit instruction that API keys must never be pasted into shared files. Cloud graph storage used Neo4j AuraDB's authenticated, encrypted connections, and the O2.1 pipeline verifies authentication independently before any processing run. The O2.2 pipeline demonstrates a complementary security posture: because both its generative and embedding models run locally through Ollama, teaching materials never leave institutional infrastructure during extraction, which removes the third-party exposure that API-based processing entails — a relevant consideration for any future module whose materials are commercially or personally sensitive.

Development of the natural-language querying prototypes also surfaced security requirements specific to LLM-generated database access, carried into the future-work programme (Section 8.4.3): any model-generated Cypher should execute under a read-only database account; non-read operations (CREATE, DELETE, SET, MERGE, REMOVE, DROP) and multi-statement queries should be rejected by parsing and validating queries against an approved schema before execution, since sanitisation alone does not prove a query is safe. These requirements follow the general security principle of least privilege applied to an LLM agent: the model is treated as an untrusted query author whose output must be constrained by the permissions of the account it runs under, not by trust in the model itself. For a deployed system, the same posture extends to institutional authentication, role-based access separating student, academic and administrative functions, and audit logging of AI-generated content and its approval history — all identified as requirements for the SMARTTEST integration phase.

7.5 Sustainability and Contribution to the Sustainable Development Goals

The project's sustainability practice is visible in its model-selection strategy. Rather than defaulting to the largest available models, each work package deliberately evaluated lightweight and locally hosted configurations first: the O2.1 benchmark concluded that gpt-4o-mini — the smallest API model tested — is the preferable configuration on combined conformance, runtime and cost, and the O2.2 pipeline demonstrated a complete construction-and-querying workflow on a local 8-billion-parameter model at zero API cost. Across the entire testing programme the project spent £0.31 of its £400 API budget (Section 5.1.1), with token counting built into the pipeline precisely so that computational consumption is measured rather than assumed. Choosing the smallest model that meets the quality requirement is both a cost decision and an environmental one: it reduces the energy footprint of every future extraction run and keeps the marginal cost of re-processing a module low enough that the platform can scale across an institution without significant recurring expenditure. The design is also sustainable in the software sense: modular pipelines, reproducible environments, formally versioned knowledge artefacts (the ontology) and archived code (Appendix D) allow future cohorts to maintain and extend the work rather than rebuild it.

The project contributes to the UN Sustainable Development Goals on two fronts. Its contribution to SDG 4 (Quality Education) is direct: WENDY exists to improve the quality, navigability and personalisation of higher education content, while its human-in-the-loop design protects the integrity of what is taught. Its contribution to SDG 8 (Decent Work and Economic Growth) operates through digital capability in two distinct ways: the platform aims to strengthen the analytical and digital skills of the students who use it, and the Students-as-Researchers model through which WENDY was built gave the student research team sustained professional experience of applied AI engineering, formal knowledge representation, benchmarking practice and collaborative research — capabilities that transfer directly into the graduate workforce.

8. Conclusion, Limitations and Future Work

8.1 Conclusions: Contributions Against the Aims and Objectives

WENDY set out to investigate how artificial intelligence could support the scalable creation and use of educational knowledge structures while retaining academic control over their accuracy and pedagogical value, in pursuit of two complementary aims: to automate and support academics in any discipline in developing concept maps and topic outlines (A1), and to guide students in learning and navigating these educational knowledge structures (A2). The project delivered against both, not as a finished platform but as a validated, reusable methodology — component workflows, prototypes, formal artefacts and evaluation evidence — ready for future integration into SMARTEST or another graph-capable learning environment.

Against O1 (conversational assistance), the project produced a structured framework for an AI-powered assistant serving both academics and students: defined use cases, conversational flows, prompt-design principles and interface concepts, together with an executable benchmarking method. The four-model benchmark over a validated knowledge base established an evidence-based model shortlist — gpt-5-mini strongest against the response criteria, Gemini strongest on source-grounding — and demonstrated empirically why grounding and validation are non-negotiable in an educational assistant.

Against O2 (concept and relationship extraction), the project delivered a working knowledge-graph construction capability twice over: two independently developed pipelines that transform lecture materials into machine-readable educational graphs in Neo4j. The O2.1 pipeline achieved a Topic F1 of 64.0% against the formal ontology at roughly £0.02 and five minutes per full-module run, and the cross-model benchmark isolated model capability as the deciding variable; the O2.2 pipeline proved that formal validation can operate before storage and that the entire workflow can run at zero API cost on local hardware. The comparative analysis (Section 4.3.3) converts this duplication into a strength: an evidenced recommendation that O2.1 is the preferable pipeline for further research, with the O2.2 techniques identified for incorporation.

Against O3 (theme discovery), the project delivered a defined methodology for identifying overlapping themes across educational collections, evaluated through five systematic experiments across six corpora. The work established where the embedding-and-soft-clustering approach performs strongly (whole documents and sections, up to 95% accuracy and ARI 0.90), empirically ruled out sentence-level units through the project's core negative result, and converted that finding into a precise design recommendation — analysis units of approximately 50–150 words — together with a specification for the purpose-built educational corpus that rigorous

evaluation requires.

Against O4 (hybrid validation), the project delivered a hybrid validation framework combining ontology-based modelling, formal constraints, SHACL validation, semantic filtering and academic review. The University Module Ontology (version 4.0) is formally consistent under Hermit reasoning, populated with the Mathematics for Computing case study and progressively generalised for reuse; both of its operational modes were shown to detect real faults in generated graphs — post-hoc conformance scoring in O2.1 and pre-storage SHACL gating in O2.2.

The central methodological principle — that AI-generated outputs are candidate educational structures requiring computational validation and academic approval before use — was not merely asserted but stress-tested: the prototypes generated structurally valid graphs that nonetheless contained weak names, sparse learning-path links and paraphrased topics, demonstrating precisely why the human-in-the-loop design is necessary. Finally, the project fulfilled its Students-as-Researchers purpose: a student team designed, built, benchmarked and critically evaluated a multi-component AI system, producing archived, reproducible artefacts (Appendices B–D) on which future cohorts can build.

8.2 Commentary on the Project Deliverables

Section 1.6 committed the project to five deliverables. Each was produced; their form and maturity are commented on individually below and summarised in Table 8.2.1.

D1 — Conversational Assistant Framework. Delivered in full as a design-and-evaluation framework. The use cases, conversational flows, prompt-design principles and interface concepts promised in Section 1.6 are presented in Section 3.4, and the benchmarking method — assessing correctness, completeness and communication over module materials, extracted ontology data, evidence files and simulated student profiles — was not only defined but executed across four candidate models (Sections 4.2, 5.2.1), yielding an evidence-based model shortlist. The deliverable is a framework rather than software by design: the assistant itself remains to be implemented on the platform side, and the framework specifies exactly what it must do and how its quality will be judged.

D2 — Educational Knowledge-Graph Construction Pipeline. Delivered beyond the stated scope. Section 1.6 promised one prototype workflow covering ingestion, preprocessing, schema-constrained extraction, semantic filtering, structured output and graph construction; the project produced two executable pipelines realising that workflow through deliberately different designs (Sections 4.3.1, 4.3.2), plus a cross-model benchmark and a comparative analysis identifying the preferred configuration (gpt-4o-mini within O2.1). The deliverable's defining condition — that the resulting graph is a draft representation subject to validation and academic review — is enforced technically in both pipelines rather than stated as policy.

D3 — Natural-Language Knowledge-Graph Access. Delivered as two complementary prototype routes: the O2.1 PropertyGraphIndex query engine, which grounds answers in graph

traversal plus source text, and the O2.2 natural-language-to-Cypher script, which makes the generated query and retrieved records fully inspectable (Sections 4.3.1, 4.3.2). The requirements identification promised in Section 1.6 — schema guidance, query generation, response reliability and safe execution — was completed in concrete engineering terms and now forms the hardening programme in Section 8.4.3.

D4 — Theme-Discovery Methodology. Delivered and matured by evidence. The sentence-embedding and soft-clustering basis promised in Section 1.6 was implemented and evaluated in five experiments (Section 5.2.3); the results empirically revised the methodology's unit of analysis from sentences to 50–150-word segments, which is a strengthening of the deliverable rather than a departure from it — the methodology now carries quantified evidence of where it works, where it fails and why. Its remaining step, evaluation on educational rather than public news corpora, is specified precisely (Section 8.4.5).

D5 — Hybrid Validation Framework. Delivered and operational at its automated first stage. The framework combines the consistent, populated ontology, SHACL constraints, semantic filtering and conformance metrics, and was exercised in earnest by both O2 pipelines — detecting malformed fragments before storage and quantifying curriculum conformance after construction (Section 5.2.4). As Section 1.6 anticipated, the framework formalises the boundary on which the whole project rests: automated checks establish formal validity, while pedagogical quality remains the academic reviewer's judgement.

Deliverable	Delivered as	Evidence	Maturity
D1 Conversational assistant framework	Use cases, flows, GUI concepts, prompt principles and an executed 4-model benchmark	Sections 3.4, 4.2, 5.2.1; Appendix C	Framework complete; assistant implementation is platform-side future work
D2 KG construction pipeline	Two executable pipelines (O2.1, O2.2) with comparative analysis and preferred configuration	Sections 4.3, 5.2.2; Appendix D	Research prototypes; O2.1 recommended for continuation
D3 Natural-language graph access	Two query prototypes (PropertyGraphIndex engine; NL-to-Cypher script) with safe-execution requirements	Sections 4.3.1, 4.3.2; Appendix D	Prototypes; hardening programme defined (7.4.3)
D4 Theme-discovery methodology	Embedding + soft-clustering methodology validated in five experiments; unit size revised on evidence	Sections 3.5, 4.4, 5.2.3	Methodology defined; educational-corpus evaluation pending
D5 Hybrid validation framework	Consistent ontology (v4.0), SHACL shapes, conformance metrics; both modes operational	Sections 3.2–3.3, 4.5, 5.2.4; Appendix B	Operational first stage; concept-level extension is future work

Table 8.2.1. Status of the project deliverables stated in Section 1.6.

8.3 Limitations

The most consequential limitation is that the prototypes were not deployed into SMARTTEST within this project. Integration of all results into the platform — as a working proof of concept and final product — remains the goal, but it was decided that this development would not start during the project, as it was uncertain how much time and resources it would require. That uncertainty stems from the SMARTTEST platform's current architecture, which is not adapted to handle new features easily or to maintain them easily once added; future work will therefore involve improving these aspects alongside the integration itself, likely with an update to the SMARTTEST API. In practical terms, there is currently no supported route through which an externally generated knowledge graph can be inserted as a SMARTTEST learning structure, no mechanism for embedding the assistant in the platform interface, and no connection between platform accounts and the personalisation layer the assistant design assumes. This is why everything that presupposes platform integration remained unrealised: frontend access to the graphs and assistant, institutional authentication and role management, the live personalisation layer (deliberately unbuilt, Section 4.1), the academic approval workflow, and server-side ingestion. Not opening this front was a deliberate scoping decision on a production system serving real students — but it is the single constraint that most directly explains why the project's outputs are research prototypes rather than deployed features. The prototypes were nonetheless architected for eventual integration: Neo4j's node-and-link representation is structurally compatible with SMARTTEST's model, the ontology has been generalised beyond the case-study module, and the components can be attached individually as integration points become available.

The remaining limitations are briefly stated, as each is answered by a specific line of future work in Section 8.4. The evidence base is narrow: one module served as the principal case study, so subject-independence remains untested, and lighter-weight models were deliberately used within the £400 budget (of which £0.31 was spent). In O2, the learning-path relationships that motivate the work — PREREQUISITE_OF above all — produced zero instances in every run of both pipelines, topic recall plateaued at 53.3% because models paraphrase rather than reproduce exact ontology strings, and both pipelines retain module-specific heuristics; the O2.2 evaluation additionally covered only two lecture decks with a single model and no annotated ground truth, and its question-answering script is not yet securely constrained to read-only execution. The O1 benchmark relied on human judgement against defined criteria, single-turn interactions and the quality of the upstream JSON extraction. The O3 evaluation lacked a suitable educational corpus — the decisive gap identified by the dataset analysis — and its metrics under-credit soft, overlapping assignments. Across the validation framework, formal checks verify structure, not pedagogy: an accepted name or relationship may still be educationally weak, which is exactly why academic review remains the final stage of the hybrid design.

8.4 Future Work

8.4.1 SMARTTEST Integration Roadmap

Future work should begin on the platform side: adapting the SMARTTEST architecture so that new features can be added and maintained more easily, and defining an updated SMARTTEST API through which externally generated knowledge graphs can be inserted as learning structures. On that foundation, the integration sequence follows the project's own architecture: an academic approval interface within the platform's authoring environment, so that validated candidate graphs can be reviewed, amended and approved by module leaders; institutional authentication and role management, separating student, academic and administrative functions; the personalisation layer (user profiles, conversation histories, learning paths), designed around the UK GDPR requirements set out in the ethics chapter from the outset; and a server-side ingestion pipeline whose processing time and progress reporting are engineered for the post-upload workflow identified in Section 4.2. Audit logging of AI-generated content and its approval history should be part of the integration design rather than an afterthought.

8.4.2 Knowledge-Graph Generation (O2)

In line with the conclusion of Section 4.3.3, future O2 development should continue from the O2.1 pipeline while incorporating the techniques proven in O2.2. For the O2.1 line, several concrete improvements present themselves from this work. First, forced topic injection: rather than relying entirely on the LLM to generate TOPIC nodes matching the ontology allowlist, the pipeline could pre-inject all valid ontology topics for each week into the graph before extraction runs, guaranteeing their presence regardless of LLM behaviour. Second, ASSESSED_BY extraction currently produces nothing because assessment information does not appear in lecture slides; processing module descriptor documents and assessment briefs as additional inputs would unlock this relationship type. Third, extending the pipeline to a second module would test the claim that the architecture is subject-independent, which is central to the WENDY platform's scope. Fourth, fine-tuning a smaller model on validated graph examples produced by the O4 validation layer would eventually allow local inference to reach API-level schema compliance at zero cost per run, making the pipeline scalable without ongoing API expenditure.

From the O2.2 construction pipeline, the following improvements carry forward:

- Build the full candidate graph in memory and perform global deduplication before database insertion.
- Run SHACL and semantic checks again over the final merged graph.
- Infer cross-slide and cross-week relationships in a second pass.
- Develop a dedicated prerequisite-inference stage using validated graph context.
- Replace mathematics-specific topic corrections with ontology-guided classification.
- Improve table and slide-layout extraction so parallel columns remain separate.
- Reduce prompt-example leakage through structured outputs or minimal demonstrations.

- Store multiple provenance records when one concept appears in several files or weeks.
- Print final unique node, relationship and property statistics directly from Neo4j.
- Create an academic ground-truth graph and measure concept, topic and relationship accuracy.
- Test the pipeline on additional modules and disciplines.

The prerequisite-inference stage deserves particular priority: with PREREQUISITE_OF at zero across all runs of both pipelines, deriving learning-path relationships — whether through a more capable model with explicit multi-step reasoning or a post-processing inference pass over the validated graph — is the single development that would most increase the educational value of the generated structures.

8.4.3 Natural-Language Graph Access and Query Safety

The question-answering route demonstrated by both pipelines should be hardened into a safely deployable component:

- Use a database account with read-only permissions.
- Reject non-read Cypher and multiple statements before execution — sanitisation alone does not prove a query is safe.
- Generate the prompt schema automatically from the live validated graph, keeping the query layer synchronised with what the construction pipeline actually produces.
- Expose description, formula, example and application properties safely, so definition and example questions can be answered.
- Add name normalisation for common user wording (e.g. "week one" versus "Week 1") without broad fuzzy database scans.
- Create a benchmark containing questions, expected Cypher, expected database records and expected final answers.
- Return the supporting graph records or source file with each answer.
- Add a controlled fallback when the graph has insufficient evidence.
- Support follow-up questions and integrate the prototype into a web interface.

8.4.4 Conversational Assistant (O1)

The benchmarking identified the assistant's next steps directly. Prompt engineering was not fully explored within the project window; systematically investigating prompt variants against the response criteria would benefit the final implementation, including tighter control of response length. Conversation memory would substantially improve the user interaction, and the implementation should define how the assistant classifies and stores student responses to satisfy the personalised-learning objectives — within the data-protection conditions set out in the ethics chapter. Model selection should follow the benchmark evidence and the deployment priority: gpt-5-mini where completeness and tutoring quality lead, Gemini where faithfulness to source material is paramount, with the ideal configuration combining GPT-level completeness with Gemini-level grounding. A grounded practice-question capability — generating questions strictly from facts in the knowledge base rather than refusing or hallucinating — is a well-defined target that would

serve both student use cases and academic quality assurance.

8.4.5 Theme Discovery (O3)

The experimental programme defined the next iteration precisely. The unit of analysis should move from sentences to paragraphs or sliding windows of approximately 50–150 words, evaluating window sizes of three, five and eight sentences with overlap; UMAP reduction before density-based clustering should be standard; per-category F1 and coverage should be reported alongside aggregate metrics; and soft assignments should be evaluated with multi-label metrics rather than being collapsed into single labels. Cases where low agreement reflects taxonomy limitations rather than model failure should be separated, as the WikiSection contrast demonstrated. WikiSection should be prioritised among public corpora for its combination of raw text, hierarchical labels and naturally sized sections, and a like-for-like comparison against BERTopic at the same paragraph-level representation remains to be run. Most importantly for WENDY, the evaluation should return to educational content: executing the Big Data module scenario (S3) defined in the benchmark framework, and constructing the purpose-built corpus recommended by the dataset analysis — approximately 200–500 educational documents organised into paragraphs and annotated with overlapping topics — which would provide the evaluation framework that no existing public dataset offers.

8.4.6 Validation Framework (O4)

The ontology should be extended downward from the course-structure layer to concept-level definitions, enabling validation of individual extracted concepts rather than only topics, weeks and assessments — the extension both O2 teams identified as the key enabler for deeper automated checking. Generalisation should be completed by moving the ontology to a module-independent namespace and populating further modules, beginning with the second case-study discipline. Operationally, ontology-guided classification should replace the module-specific correction heuristics in the pipelines, and validation should be extended from per-chunk checking to revalidation of complete merged graphs. As the framework matures, the boundary it draws — formal artefacts verify structure, academics judge pedagogy — should be preserved as its defining design principle.

8.4.7 Programme Development and Dissemination

Beyond the technical roadmap, three programme-level directions follow. First, widening the subject base: applying the methodology to additional Westminster modules and disciplines is both a research need (the subject-independence claim) and the natural route to broader academic engagement with SMARTEST. Second, continuity of the student-research model: the project's scale and its archived, reproducible artefacts make it well suited to continuation through credit-bearing formats or vertically integrated projects (VIPs), in which successive cohorts extend the pipelines, corpus and evaluation base. Third, dissemination: the project's methodology and findings — particularly the two-pipeline comparison, the sentence-level negative result and the hybrid validation design — merit dissemination through a research paper, workshop presentation

and exhibition of the prototypes, subject to the staff leads' publication plans.

8.5 Closing Remark

WENDY's contribution is a controlled, human-centred methodology for transforming existing teaching materials into structured, navigable, and validated representations of knowledge. The project demonstrates that the transformation can be substantially automated; that its quality can be measured, compared and formally checked; and that the automation can be governed so that academic judgement remains decisive. Every deliverable stated at the outset was produced, and the remaining constraints are known, bounded, and paired with a concrete plan. What remains is engineering and scale (the platform integration, additional disciplines, and richer relationship extraction set out above) on the foundations that this project has built, evidenced, and archived for those who continue the work.

References

- Abu-Rasheed, H., Al Amin, R.J., Weber, C., Wiese, V., Obermaisser, R. and Fathi, M. (2025) 'LLM-Assisted Knowledge Graph Completion for Curriculum and Domain Modelling in Personalized Higher Education Recommendations', *Proceedings of EDUCON 2025*, pp. 1–5. Available at: <https://doi.org/10.1109/EDUCON62633.2025.11016377> (Accessed: 31 July 2026).
- Adamopoulou, E. and Moussiades, L. (2020) 'Chatbots: History, Technology, and Applications', *Machine Learning with Applications*, 2, 100006. Available at: <https://doi.org/10.1016/j.mlwa.2020.100006> (Accessed: 31 July 2026).
- Aleedy, M., Atwell, E. and Meshoul, S. (2025) 'A Multi-Agent Chatbot Architecture for AI-Driven Language Learning', *Applied Sciences*, 15(19), 10634. Available at: <https://doi.org/10.3390/app151910634> (Accessed: 31 July 2026).
- Anastasiou, D., Wirngo, C.N. and Bagos, P. (2024) 'The effectiveness of concept maps on students' achievement in science: A meta-analysis', *Educational Psychology Review*, 36, article 39.
- Attigeri, G., Agrawal, A. and Kolekar, S.V. (2024) 'Advanced NLP Models for Technical University Information Chatbots: Development and Comparative Analysis', *IEEE Access*, 12, pp. 29633–29647. Available at: <https://ieeexplore.ieee.org/stampPDF/getPDF.jsp?arnumber=10440622> (Accessed: 31 July 2026).
- Batsaikhan, B. and Correia, A. (2024) 'The effects of Generative Artificial Intelligence on Intelligent Tutoring Systems in Higher Education: A Systematic Review'. Available at: <https://doi.org/10.21428/8c225f6e.33570bb1> (Accessed: 31 July 2026).
- Bench-Capon, T.J.M. (2001) 'The role of ontologies in the verification and validation of knowledge-based systems', *International Journal of Intelligent Systems*, 16(3), pp. 377–390. Available at: <https://cgi.csc.liv.ac.uk/~tbc/publications/ontvv.pdf> (Accessed: 31 July 2026).
- Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. et al. (2020) 'Language Models are Few-Shot Learners', *Advances in Neural Information Processing Systems*, 33, pp. 1877–1901. Available at: <https://arxiv.org/abs/2005.14165> (Accessed: 31 July 2026).
- Carlander-Reuterfelt, D., Carrera, Á., Iglesias, C.A., Araque, Ó., Sánchez Rada, J.F. and Muñoz, S. (2020) 'JAICOB: A Data Science Chatbot', *IEEE Access*, 8, pp. 180672–180680. Available at: <https://ieeexplore.ieee.org/document/9200315> (Accessed: 31 July 2026).
- Chu, K., Yi-Pei, C. and Nakayama, H. (2024) 'A Better LLM Evaluator for Text Generation: The Impact of Prompt Output Sequencing and Optimization'. Available at:

- <https://doi.org/10.48550/arxiv.2406.09972> (Accessed: 31 July 2026).
- Cui, J. and Yu, S. (2019) 'Fostering deeper learning in a flipped classroom: Effects of knowledge graphs versus concept maps', *British Journal of Educational Technology*, 50(5), pp. 2308–2328. Available at: <https://doi.org/10.1111/bjet.12841> (Accessed: 31 July 2026).
- Dagdelen, J., Dunn, A., Lee, J., Rosen, A.S., Han, Y., Srinivasan, V., Rong, Z., Hippalgaonkar, K., Jain, A. and Persson, K.A. (2024) 'Structured information extraction from scientific text with large language models', *Nature Communications*, 15, article 1418. Available at: <https://doi.org/10.1038/s41467-024-45563-x> (Accessed: 31 July 2026).
- de Ries, K.E., Schaap, H., van Loon, A.-M.M.J.A.P., Kral, M.M.H. and Meijer, P.C. (2022) 'A literature review of open-ended concept maps as a research instrument to study knowledge and learning', *Quality & Quantity*, 56, pp. 73–107. Available at: <https://doi.org/10.1007/s11135-021-01113-x> (Accessed: 31 July 2026).
- Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A. et al. (2024) 'From Local to Global: A Graph RAG Approach to Query-Focused Summarization'. arXiv preprint, arXiv:2404.16130. Available at: <https://arxiv.org/abs/2404.16130> (Accessed: 31 July 2026).
- Espinoza, A., Del-Moral, E., Martínez-Martínez, A. and Alí, N. (2021) 'A validation and verification driven ontology: an iterative process', *Applied Ontology*, 16(3), pp. 297–337. Available at: <https://doi.org/10.3233/AO-210251> (Accessed: 31 July 2026).
- Fettach, Y., Ghogho, M. and Benatallah, B. (2022) 'Knowledge graphs in education and employability: A survey on applications and techniques', *IEEE Access*, 10, pp. 80174–80183. Available at: <https://doi.org/10.1109/ACCESS.2022.3194063> (Accessed: 31 July 2026).
- Galletti, M., Blin, I. and Ilkou, E. (2025) 'Automated Concept Map Extraction from Text', in *Proceedings of the 5th Conference on Language, Data and Knowledge*, pp. 87–99. Available at: <https://aclanthology.org/2025.ldk-1.10/> (Accessed: 31 July 2026).
- Ganguly, A., Mehjabin, N., Malik, A. and Johri, A. (2025) 'Conversational AI agents in education: an umbrella review of current utilization, challenges, and future directions for ethical and responsible use', *AI and Ethics*. Available at: <https://doi.org/10.1007/s43681-025-00916-0> (Accessed: 31 July 2026).
- Gonda, D.E. and Chu, B. (2019) 'Chatbot as a learning resource? Creating conversational bots as a supplement for teaching assistant training course', in *IEEE TALE 2019*, pp. 1–5. Available at: <https://doi.org/10.1109/TALE48000.2019.9225974> (Accessed: 14 July 2026).
- Grootendorst, M. (2022) *BERTopic: Neural Topic Modeling with a Class-Based TF-IDF Procedure*. arXiv preprint, arXiv:2203.05794. Available at: <https://doi.org/10.48550/arXiv.2203.05794> (Accessed: 14 July 2026).
- Han, H., Wang, Y., Shomer, H., Guo, K., Ding, J. et al. (2025) *Retrieval-Augmented Generation with Graphs (GraphRAG)*. arXiv preprint, arXiv:2501.00309. Available at:

- <https://arxiv.org/abs/2501.00309> (Accessed: 14 July 2026).
- Hong, L. and Davison, B.D. (2010) 'Empirical Study of Topic Modeling in Twitter', in *Proceedings of the First Workshop on Social Media Analytics (SOMA '10)*. Washington, DC: ACM, pp. 80–88. Available at: <https://doi.org/10.1145/1964858.1964870> (Accessed: 31 July 2026).
- Jana, S., Jana, M., De Santis, A. and Minerva, T. (2026) 'An AI-Powered Chatbot for Enhancing Learner-Centered Support in Moodle-Based Learning Environments', in *EdMedia 2026*, pp. 383–388.
- Jeong, E., Nagasaki, M., Ueno, K. and Miyano, S. (2011) 'Ontology-based instance data validation for high-quality curated biological pathways', *BMC Bioinformatics*, 12(Suppl 1), S8. Available at: <https://doi.org/10.1186/1471-2105-12-S1-S8> (Accessed: 14 July 2026).
- Ji, S., Pan, S., Cambria, E., Marttinen, P. and Yu, P.S. (2022) 'A survey on knowledge graphs: Representation, acquisition, and applications', *IEEE Transactions on Neural Networks and Learning Systems*, 33(2), pp. 494–514. Available at: <https://doi.org/10.1109/TNNLS.2021.3070843> (Accessed: 14 July 2026).
- Jia, J. (2003) *The Study of the Application of a Keywords-Based Chatbot System on the Teaching of Foreign Languages*. arXiv preprint. Available at: <https://doi.org/10.48550/arXiv.cs/0310018> (Accessed: 14 July 2026).
- Kolb, A.Y. and Kolb, D.A. (2009) 'Experiential learning theory: A dynamic, holistic approach to management learning, education and development', in Armstrong, S.J. and Fukami, C.V. (eds.) *The SAGE Handbook of Management Learning, Education and Development*. London: SAGE, pp. 42–68.
- Kuhail, M.A., Alturki, N., Alramlawi, S. and Alhejori, K. (2023) 'Interacting with educational chatbots: A systematic review', *Education and Information Technologies*, 28(1), pp. 973–1018. Available at: <https://doi.org/10.1007/s10639-022-11177-3> (Accessed: 14 July 2026).
- Kuo, Y. and Chen, Y. (2023) 'The impact of chatbots using concept maps on correction outcomes: A case study of programming courses', *Education and Information Technologies*, 28, pp. 7899–7925. Available at: <https://doi.org/10.1007/s10639-022-11506-6> (Accessed: 14 July 2026).
- Labadze, L., Grigolia, M. and Machaidze, L. (2023) 'Role of AI Chatbots in Education: Systematic Literature Review', *International Journal of Educational Technology in Higher Education*, 20(56). Available at: <https://doi.org/10.1186/s41239-023-00426-1> (Accessed: 14 July 2026).
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V. et al. (2020) 'Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks', *Advances in Neural Information Processing Systems*, 33, pp. 9459–9474.
- Lin, C., Huang, A.Y.Q. and Lu, O.H.T. (2023) 'Artificial intelligence in intelligent tutoring systems toward sustainable education: a systematic review', *International Journal of Educational*

- Technology in Higher Education*, 10, 41. Available at: <https://doi.org/10.1186/s40561-023-00260-y> (Accessed: 31 July 2026).
- Liu, Q., Wang, Z. and Yang, Q. (2025) 'From barriers to bridges: A TPACK-XL framework for knowledge graph integration in higher education', in *Proceedings of the ACM Conference 2025*, pp. 767–773.
- Ma, W., Ma, W., Hu, Y. and Bi, X. (2025) 'The who, why, and how of AI-based chatbots for learning and teaching in higher education: A systematic review', *Education and Information Technologies*, 30, pp. 7781–7805.
- McInnes, L., Healy, J. and Astels, S. (2017) 'hdbscan: Hierarchical density based clustering', *Journal of Open Source Software*, 2(11), article 205. Available at: <https://doi.org/10.21105/joss.00205> (Accessed: 31 July 2026).
- McInnes, L., Healy, J. and Melville, J. (2018) *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. arXiv preprint, arXiv:1802.03426. Available at: <https://arxiv.org/abs/1802.03426> (Accessed: 31 July 2026).
- Nageshwaran, V. and Nageshwaran, S. (2025) 'Leveraging LLM-Powered Chatbot for Learning and Optimization in Databases and Information Systems Education', in *ICSADL 2025*, pp. 268–273.
- Nazir, S., Motla, Y.H., Abbas, T., Khatoon, A., Jabeen, J., Iqra, M. and Bakhat, K. (2014) 'A process improvement in requirement verification and validation using ontology', in *2014 IEEE Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE)*. IEEE, pp. 1–8. Available at: <https://doi.org/10.1109/APWCSE.2014.7053837> (Accessed: 31 July 2026).
- Nesbit, J.C. and Adesope, O.O. (2006) 'Learning with concept and knowledge maps: A meta-analysis', *Review of Educational Research*, 76(3), pp. 413–448.
- Neupane, S., Hossain, E., Keith, J., Tripathi, H., Ghiasi, F. et al. (2024) 'From Questions to Insightful Answers: Building an Informed Chatbot for University Resources', *Frontiers in Education Conference (FIE 2024)*.
- Novak, J.D. and Cañas, A.J. (2007) 'Theoretical origins of concept maps, how to construct them, and uses in education', *Reflecting Education*, 3(1), pp. 29–42.
- Novak, J.D. and Cañas, A.J. (2008) *The Theory Underlying Concept Maps and How to Construct and Use Them*. Technical Report IHMC CmapTools 2006-01, revised 2008. Pensacola, FL: Florida Institute for Human and Machine Cognition.
- Noy, N.F. and McGuinness, D.L. (2001) *Ontology Development 101: A Guide to Creating Your First Ontology*. Stanford, CA: Stanford Knowledge Systems Laboratory.
- Nuruzzaman, M. and Hussain, O.K. (2018) 'A Survey on Chatbot Implementation in Customer

- Service Industry through Deep Neural Networks', in *IEEE International Conference on e-Business Engineering*, pp. 54–61.
- OpenAI (2023) *GPT-4 Technical Report*. Available at: <https://arxiv.org/abs/2303.08774> (Accessed: 31 July 2026).
- Patchava, R.S. and Jonnalagadda, S.K. (2020) 'Intelligent Response Retrieval for Semantically Similar Querying Using a Chatbot', in *IC/CCS 2020*, pp. 502–508.
- Perin, W., Cury, D., Aguiar, C.Z. and Menezes, C.S. (2023) 'From Text to Maps: Automated Concept Map Generation Using Fine-Tuned Large Language Models', in *Proceedings of the Brazilian Congress on Informatics in Education (CBIE 2023)*.
- Qiang, J., Qian, Z., Li, Y., Yuan, Y. and Wu, X. (2022) 'Short Text Topic Modeling Techniques, Applications, and Performance: A Survey', *IEEE Transactions on Knowledge and Data Engineering*, 34(3), pp. 1427–1445. Available at: <https://doi.org/10.1109/TKDE.2020.2992485> (Accessed: 31 July 2026).
- Qu, K., Li, K.C., Wong, B.T.M., Wu, M.M.F. and Liu, M. (2024) 'A survey of knowledge graph approaches and applications in education', *Electronics*, 13(13), article 2537.
- Ranoliya, B.R., Raghuwanshi, N. and Singh, S. (2017) 'Chatbot for University Related FAQs', in *ICACCI 2017*, pp. 1525–1530.
- Reimers, N. and Gurevych, I. (2019) 'Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks', in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong: Association for Computational Linguistics, pp. 3982–3992. Available at: <https://doi.org/10.18653/v1/D19-1410> (Accessed: 31 July 2026).
- Singh, J., Joseph, M.H. and Jabbar, K.B.A. (2019) 'Rule-based chatbot for student enquiries', *Journal of Physics: Conference Series*, 1228(1), 012060.
- Stoyanova, T. and Anguelov, K. (2025) 'A comparative analysis of different platforms for creating educational chatbots', in *COMSCI 2025*, pp. 1–6.
- Tang, X., Ni, J., Meng, Y., Chen, Q. and Zhang, L. (2026) 'Knowledge graphs as cognitive scaffolding for sustainable engineering education', *Sustainability*, 18(2), 736.
- Traag, V.A., Waltman, L. and Van Eck, N.J. (2019) 'From Louvain to Leiden: guaranteeing well-connected communities', *Scientific Reports*, 9(1), 5233. Available at: <https://doi.org/10.1038/s41598-019-41695-z> (Accessed: 31 July 2026).
- Tsaneva, S., Dessì, D., Osborne, F. and Sabou, M. (2025) 'Knowledge graph validation by integrating LLMs and human-in-the-loop', *Information Processing & Management*, 62, 104145.

- Wang, K., Ramos, J. and Lawrence, R. (2023) *ChatEd: A Chatbot Leveraging ChatGPT for an Enhanced Learning Experience in Higher Education*. arXiv preprint.
- Wang, K.S. (2025) *Transforming Student Support with AI: A Retrieval-Based Generation Framework for Personalized Support and Faculty Customization*. Master's thesis. University of British Columbia.
- Wang, X.-M., Wang, J.-L., Xu, S.-Y. and Xu, S.-J. (2025) 'Concept mapping in STEM education: A meta-analysis of its impact on students' achievement (2004–2023)', *International Journal of STEM Education*, 12, article 30.
- Wijaya, R.A., Kusumaningtyas, E.M. and Barakbah, A. (2019) 'Knowledge-Based Chatbot with Context Recognition', in *International Electronics Symposium*, pp. 432–438.
- Wollny, S., Schneider, J., Di Mitri, D., Weidlich, J., Rittberger, M. and Drachsler, H. (2021) 'Are We There Yet? A Systematic Literature Review on Chatbots in Education', *Frontiers in Artificial Intelligence*, 4, 654924.
- Wu, Y., Li, Z., Wu, W. and Zhou, M. (2018) 'Response selection with topic clues for retrieval-based chatbots', *Neurocomputing*, 316, pp. 251–261.
- Yang, J., Duan, C., Zeng, Z., Liu, Y., Bai, J. and Zhang, M. (2024) 'Research on Knowledge Graph Recommendation Method for Online Education', in *IEEE ICET 2024*, pp. 59–64.
- Yang, T., Ren, B., Gu, C., He, T., Ma, B. and Konomi, S. (2025) 'Leveraging LLMs for automated extraction and structuring of educational concepts and relationships', *Machine Learning and Knowledge Extraction*, 7(3), 103.
- Yin, J. and Wang, J. (2014) 'A Dirichlet Multinomial Mixture Model-based Approach for Short Text Clustering', in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. New York: ACM, pp. 233–242. Available at: <https://doi.org/10.1145/2623330.2623715> (Accessed: 31 July 2026).
- Yusuf, H., Money, A. and Daylamani-Zad, D. (2025) 'Pedagogical AI conversational agents in higher education: A conceptual framework and survey of the state of the art', *Educational Technology Research and Development*, 73, pp. 815–874.
- Zhai, X. (2025) 'Generative large language models for knowledge representation: A systematic review of concept map generation', *arXiv preprint*, arXiv:2509.14554.
- Zhang, T., Patterson, L., Webb, B., Jin, Z. and Beiting-Parrish, M. (2026) 'Evaluating generative AI chatbots for large-scale assessment data: comparing LLM-as-a-judge and human ratings', *Large-scale Assessments in Education*, 14(1), article 19. Available at: <https://doi.org/10.1186/s40536-026-00287-w> (Accessed: 31 July 2026).
- Zhou, L., Gao, J., Li, D. and Shum, H.-Y. (2020) 'The Design and Implementation of Xiaolce, an Empathetic Social Chatbot', *Computational Linguistics*, 46(1), pp. 53–93.

Software and Technical Documentation

The sources below are software and platform documentation cited for implementation detail (Chapter 4) rather than academic literature, and are conventionally listed separately from the reference list above.

LangChain (2025) *LangChain Documentation*. Available at: <https://python.langchain.com/docs/> (Accessed: 23 July 2026).

LlamaIndex (2025a) *Using a Property Graph Index*. Available at: https://docs.llamaindex.ai/en/stable/module_guides/indexing/lpg_index_guide/ (Accessed: 23 July 2026).

LlamaIndex (2025b) *Schema-Guided Property Graph Extraction*. Available at: https://docs.llamaindex.ai/en/stable/examples/property_graph/property_graph_advanced/ (Accessed: 23 July 2026).

LlamaIndex (2025c) *Property Graph Index with Neo4j*. Available at: https://docs.llamaindex.ai/en/stable/examples/property_graph/property_graph_neo4j/ (Accessed: 23 July 2026).

LlamaIndex (2025d) *Using a Property Graph Store*. Available at: https://developers.llamaindex.ai/python/examples/property_graph/graph_store/ (Accessed: 23 July 2026).

LlamaIndex (2025e) *Local LLM Starter Guide*. Available at: https://docs.llamaindex.ai/en/stable/getting_started/starter_example_local/ (Accessed: 23 July 2026).

Microsoft (2026a) *Explore AI Capabilities in Copilot Studio*. Microsoft Learn. Available at: <https://learn.microsoft.com/en-us/microsoft-copilot-studio/guidance/ai-capabilities> (Accessed: 31 July 2026).

Microsoft (2026b) *Microsoft Azure AI Documentation – Retrieval-Augmented Generation (RAG)*. Available at: <https://learn.microsoft.com/azure/ai-foundry/> (Accessed: 31 July 2026).

Microsoft Research (2024) *GraphRAG: New Tool for Complex Data Discovery, Now on GitHub*. Available at: <https://www.microsoft.com/en-us/research/blog/graphrag-new-tool-for-complex-data-discovery-now-on-github/> (Accessed: 31 July 2026).

Microsoft Research (2025) *GraphRAG*. GitHub Repository. Available at: <https://github.com/microsoft/graphrag/> (Accessed: 31 July 2026).

MongoDB (2025) *Importing JSON Data into MongoDB*. Available at: <https://www.mongodb.com/resources/languages/json-to-mongodb/> (Accessed: 31 July 2026).

Neo4j (2024a) *Customizing Property Graph Index in LlamaIndex*. Available at: <https://neo4j.com/blog/developer/property-graph-index-llamaindex/> (Accessed: 31 July 2026).

Neo4j (2024b) *Integrating Microsoft GraphRAG into Neo4j*. Available at: <https://neo4j.com/blog/developer/microsoft-graphrag-neo4j/> (Accessed: 31 July 2026).

Neo4j (2025a) *Connecting LlamaIndex to Neo4j using Neo4jPropertyGraphStore*. Available at: <https://neo4j.com/labs/genai-ecosystem/llamaindex/> (Accessed: 31 July 2026).

Neo4j (2025b) *Neo4j Python Driver Manual*. Available at: <https://neo4j.com/docs/python-manual/current/> (Accessed: 31 July 2026).

Neo4j (2025c) *Neo4j AuraDB Documentation*. Available at: <https://neo4j.com/docs/aura/aurads/connecting/> (Accessed: 31 July 2026).

Neo4j (2025d) *Neo4j APOC Import Documentation*. Available at: <https://neo4j.com/docs/apoc/current/import/> (Accessed: 31 July 2026).

Neo4j (2025e) *APOC JSON Import Procedure Reference*. Available at: <https://neo4j.com/docs/apoc/current/overview/apoc.import/apoc.import.json/> (Accessed: 31 July 2026).

Neo4j Labs (2025a) *LLM Knowledge Graph Builder*. GitHub Repository. Available at: <https://github.com/neo4j-labs/llm-graph-builder> (Accessed: 31 July 2026).

Neo4j Labs (2025b) *LLM Knowledge Graph Builder: From Zero to GraphRAG in Five Minutes*. Available at: <https://neo4j.com/blog/developer/graphrag-llm-knowledge-graph-builder/> (Accessed: 31 July 2026).

Ollama (2025) *Ollama Documentation*. Available at: <https://ollama.com> (Accessed: 31 July 2026).

pdfplumber (2025) *pdfplumber Documentation*

Supporting Materials and Appendices

This report is supported by a set of technical artefacts too large or too detailed to include in full in the main text: diagrams, the project ontology, raw testing and benchmarking data, the two O2 prototype codebases, and the records underlying Chapter 6's account of team working. These are held in a shared folder (APPENDICES) organised into five sub-folders, Appendix A to Appendix E, summarised in Table 1 and described in turn below. Where a chapter of the main report draws directly on a given appendix, that chapter is identified so the two can be read side by side.

Appendix	Contents	Format	Used in
A — Diagrams	System architecture, SMARTTEST-with-web-service and SMARTTEST use-case diagrams	PDF (3 files)	Chapters 3–4
B — Ontology	WENDY project ontology (top-level version)	OWL (1 file)	Chapter 4 (O4), Chapter 7
C — Testing	O1 model-response testing results; O2 cross-model LLM benchmark table	CSV, TXT, XLSX	Chapter 5
D — Project Code	O2.1 (Alexandru) and O2.2 (Akil) prototype codebases	Jupyter notebook; Python scripts, with data, images and a Git repository	Chapter 4
E — Teamwork	Gantt schedule, work-package allocation table, meeting transcripts, one screenshot	XLSX, GSHEET, DOCX, GDOC, PNG	Chapter 6