THE UNIVERSITY OF ZAMBIA

SCHOOL OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF COMPUTING AND INFORMATICS

FINAL YEAR PROJECT REPORT

ACADEMIC YEAR 2026

INTELLIGENT FILE MANAGEMENT SYSTEM

An Individual Subsystem of the Intelligent Digital Library with Personalized Study Assistance

A report submitted to the Department of Computing and Informatics, The University of Zambia, in partial fulfilment of the requirements of the degree of Bachelor of Science in Computer Science (Software Engineering)

By

Name: Arthur F. Chipeta
Computer Number: 2022068424

Supervisor:
Dr Lighton Phiri

October 23, 2026

Declaration

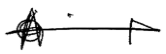
I declare that this final year project report is my own work and has been prepared in accordance with the academic requirements of the University of Zambia. All sources of information, ideas, data, software, images, and other material that are not my own have been appropriately acknowledged and referenced.

Where this project was undertaken as part of a group, I have clearly identified my individual contributions and have not presented the work of other group members as my own.

Student Name: Arthur Franklin Chipeta

Computer Number: 2022068424

Signature:



Date: October 23, 2026

Approval

This final year project report has been submitted for examination with the approval of the project supervisor.

Supervisor Name: Dr Lighton Phiri

Signature: _____

Date: October 23, 2026

Departmental/Examiner Approval (if applicable): _____

Abstract

University students accumulate academic resources across downloads, learning platforms, email, messaging applications, and websites, making consistent organisation and later retrieval difficult. This project designed, implemented, and evaluated an Intelligent File Management System that provides a secure web-based personal library for supported academic files and webpages. The system combines a browser interface, a FastAPI service, SQLAlchemy persistence, and configurable local or private cloud object storage. Available text is extracted from supported document formats and used by a transparent hybrid classifier that considers course codes, filenames, headings, controlled terms, and TF-IDF cosine similarity. The system stores the proposed category, confidence information, ranked alternatives, review state, and topic labels with page or slide evidence where available. Resources are organised in a Course-to-Material-Type hierarchy, while manual category corrections, topic edits, renaming, and folder movements remain authoritative. Evaluation of the implemented subsystem produced 23 successful automated tests. Ten attempted cross-user operations returned 404, demonstrating owner isolation for the covered resource paths. Five controlled category fixtures and four topic-and-location assertions also passed. Local microbenchmarks recorded median core times of approximately 191.1 ms for classification and 10.9 ms for topic extraction on short fixtures. These results support the functional correctness and technical feasibility of the tested prototype. They do not establish general classifier accuracy, participant usability, deployed scalability, or production security because an independent labelled collection, a completed user study, and end-to-end deployment measurements were not available. The project therefore provides an explainable and user-correctable foundation for content-aware academic resource organisation, with independent evaluation and production hardening identified as the main next steps.

Keywords: academic resource management; intelligent file management; explainable classification; topic extraction; personal information management

Acknowledgements

I am grateful to my project supervisor for the guidance, review, and academic direction provided throughout the project. I also acknowledge the Department of Computing and Informatics and the University of Zambia for the academic framework and resources that supported this work.

I appreciate my group members for their collaboration on the wider project while each member developed and documented an individual contribution. Their coordination helped maintain a shared project direction without obscuring responsibility for the Intelligent File Management System presented in this report.

Finally, I thank my colleagues, friends, and family for their encouragement and practical support during the design, implementation, testing, and documentation of the project.

Table of Contents

1. Introduction 11
   1.1. Study Background 11
   1.2. Problem Statement 11
   1.3. Study Objectives 12
      1.3.1. Broad Objective 12
      1.3.2. Specific Objectives 12
      1.3.3. Research Questions 13
   1.4. Study Rationale 13
   1.5. Scope and Limitations 13
      1.5.1. Specific Scope 14
      1.5.2. Out of Scope 14
   1.5.3. Limitations 15
   1.6. Ethical Considerations 15
   1.7. Report Organisation 16
2. Literature Review 17
   2.1. Introduction 17
   2.2. Key Concepts and Theories 17
      2.2.1. Personal Information Management 17
      2.2.2. Student Academic Resource Management 17
      2.2.3. Folders Tags and Retrieval 17
      2.2.4. Content Based Classification 18
      2.2.5. Explainability and Human Correction 18
      2.2.6. Security Privacy and Ownership 18
   2.3. Existing Systems and Related Work 18
      2.3.1. General Cloud File Services 18
      2.3.2. Reference and Research Managers 18
      2.3.3. Content Aware Research Systems 19
   2.4. Comparative Analysis 19
   2.5. Research and Knowledge Gap 19
   2.6. Conceptual Framework 20
   2.7. Chapter Summary 20
3. Methodology 21
   3.1. Introduction 21
   3.2. Research and Development Approach 21
   3.3. Requirements and Data Collection 22
      3.3.1. Sources of Requirements 22
      3.3.2. Requirement Prioritisation 22
   3.4. System Architecture and Research Workflow 22
      3.4.1. Three Tier Web Architecture 22
      3.4.2. Resource Processing Workflow 22
      3.4.3. Maintenance and Backfill Workflow 23
   3.5. Tools and Technologies 23
   3.6. Ethical and Legal Considerations 24
   3.7. Evaluation Method 24
      3.7.1. Functional and Security Evaluation 24
      3.7.2. Classification and Topic Evaluation 24
      3.7.3. Performance and Quality Evaluation 25
   3.8. Chapter Summary 25
4. System Analysis and Design 26
   4.1. Introduction 26

4.2. Stakeholder and User Analysis 26
4.2.1. Primary User Profile 26
4.2.2. Access and Trust Boundaries 27
4.3. Functional Requirements 27
4.3.1. Business Rules 28
4.4. Non-Functional Requirements 29
4.5. Use Cases / User Stories 30
4.5.1. Detailed Use Case UC-02: Upload and Organise a File 31
4.5.2. Detailed Use Case UC-05: Review and Correct a Category 32
4.6. System / Solution Architecture 32
4.6.1. Resource Ingestion and Processing 34
4.6.2. Retrieval and Maintenance 36
4.6.3. Deployment Configuration 36
4.7. Database / Data Design 37
4.7.1. Relationship and Integrity Rules 38
4.7.2. Data Ownership Queries 39
4.8. Interface / Interaction Design 39
4.8.1. Feedback and Error Handling 39
4.8.2. Consistency and Accessibility 40
4.9. Algorithms / Models / Technical Design 43
4.9.1. File Validation, Naming and Storage 43
4.9.2. Format-Specific Text Extraction 43
4.9.3. Hybrid Course Classification 43
4.9.4. Topic Extraction 44
4.9.5. Folder Resolution and Manual-Choice Preservation 44
4.9.6. Security and Failure-Control Logic 44
4.10. Chapter Summary 44
5. Implementation and Development 46
5.1. Introduction 46
5.2. Development Environment 46
5.3. System Components and Modules 46
5.4. Key Implementation Details 47
5.4.1. Authentication and Ownership Enforcement 47
5.4.2. Resource Ingestion and Private Storage 48
5.4.3. Format Specific Text Extraction 48
5.4.4. Explainable Hybrid Classification 48
5.4.5. Topic Labels with Location Evidence 48
5.4.6. Hierarchical Folder Organisation 49
5.4.7. Browser Interface 49
5.5. Integration 51
5.6. Challenges and Solutions 51
5.7. Individual Contribution 52
5.8. Chapter Summary 52
6. Testing, Results and Evaluation 53
6.1. Introduction 53
6.2. Testing Strategy 53
6.3. Test Cases and Results 54
6.4. Experimental Results / Performance Results 55
6.4.1. Automated Suite Result 55
6.4.2. Classification and Topic Fixture Results 55
6.4.3. Core Algorithm Timing 56
6.4.4. Ownership and Security Results 56
6.5. User Evaluation 56

6.6. Analysis of Results 56
6.7. Objective-by-Objective Evaluation 57
6.8. Individual Results 57
6.9. Chapter Summary 58
7. Discussion 59
7.1. Introduction 59
7.2. Interpretation of Findings 59
7.2.1. Organisation of Academic Resources 59
7.2.2. Classification Topic Evidence and Human Review 59
7.2.3. Privacy and Ownership 59
7.2.4. Performance and User Experience 60
7.2.5. Relationship to the Objectives and Research Questions 60
7.3. Comparison with Existing Work 60
7.3.1. General Cloud Storage and Reference Managers 60
7.3.2. Content Aware Organisation 60
7.3.3. Position of the Implemented Subsystem 61
7.4. Contributions of the Project 61
7.4.1. Practical Contribution 61
7.4.2. Technical Contribution 61
7.4.3. Evaluation Contribution 61
7.5. Limitations and Implications 62
7.5.1. Limits of the Evaluation Evidence 62
7.5.2. Limits of the Implemented Scope 62
7.5.3. Implications for Use and Further Evaluation 62
7.6. Chapter Summary 63
8. Conclusion and Recommendations 64
8.1. Introduction 64
8.2. Summary of the Project 64
8.3. Conclusions 64
8.3.1. Achievement of the Project Aim 64
8.3.2. Value of Explainable and Correctable Automation 64
8.3.3. Privacy and Consistency 64
8.3.4. Limits of the Final Conclusion 65
8.4. Achievement of Objectives 65
8.5. Recommendations 65
8.6. Future Work 66
9. References 68
APPENDIX A: INDIVIDUAL CONTRIBUTION STATEMENT FOR GROUP PROJECTS 69
APPENDIX B: USER MANUAL / INSTALLATION GUIDE 75
APPENDIX C: ADDITIONAL SYSTEM DESIGNS / DIAGRAMS 84
APPENDIX D: ADDITIONAL TEST CASES AND RESULTS 88
APPENDIX E: DATA COLLECTION INSTRUMENTS 92
APPENDIX F: ADDITIONAL CODE / CONFIGURATION 94
APPENDIX G: OTHER SUPPORTING MATERIAL 97

List of Tables

Table 2.1 Comparative analysis of related systems 19
Table 2.2 Conceptual framework for intelligent resource management 20
Table 3.1 Requirement evidence and resulting subsystem decisions 22
Table 3.2 Tools and technologies used in the subsystem 23
Table 3.3 Evaluation criteria 25
Table 4.1: Stakeholders, Needs and Design Implications 26
Table 4.2: Functional Requirements 27
Table 4.3: Non-Functional Requirements 29
Table 4.4: Use Case Summary 30
Table 4.5: Architectural Components and Responsibilities 34
Table 4.6: Core Data Entities 37
Table 4.7: Interface Areas and Interaction Decisions 39
Table 5.1 Implemented subsystem modules 46
Table 5.2 Implementation challenges and adopted solutions 51
Table 6.1: Test Strategy and Evidence Sources 53
Table 6.2: Representative Functional and Security Test Cases 54
Table 6.3: Automated Results by Functional Area 55
Table 6.4: Local Core-Service Microbenchmark 56
Table 6.5: Evaluation of Project Objectives 57
Table 8.1: Final Assessment of Project Objectives 65
Table A.1: Project and Group Information 69
Table A.2: Individual Role and Responsibilities 70
Table A.3: Individual Deliverables 71
Table A.4: Individual Implementation and Technical Work 72
Table A.5: Individual Testing and Evaluation Contribution 73
Table A.6: Estimated Contribution 74
Table B.1: System Overview 75
Table B.2: Supported Resource Types 76
Table B.3: Installation Requirements 76
Table B.4: Create and Edit the Environment File 77
Table B.5: Start the Application 78
Table B.6: Storage and Deployment Configuration 82
Table B.7: Troubleshooting 82
Table B.8: Quick Reference 84
Table C.1: Diagram Index 84
Table D.1: Reproduced Test Run 88
Table D.2: Complete Automated Test Inventory 88
Table D.3: Representative Assertions and Observed Results 90
Table D.4: Cross-User Ownership Checks 90
Table D.5: Controlled Classification and Topic Fixtures 91
Table D.6: Local Core-Service Timing 91
Table E.1: Importance of Intelligent File-Management Features 93
Table F.1: Implemented API Inventory 94
Table F.2: Principal Implementation Modules 95
Table G.1: Requirements-to-Evidence Traceability 97
Table G.2: Reproducibility Checklist 98
Table G.3: Evidence and Claim Boundary 98
Table G.4: Supporting Evidence Index 98

List of Figures

Figure 3.1: Scrum-Based Iterative Development Methodology 21
Figure 4.1: Use Case Diagram for the Intelligent File Management System 30
Figure 4.2: High-Level System Architecture 33
Figure 4.3: UML Component Diagram of the Intelligent File Management System 33
Figure 4.4: Resource Upload and Organisation Activity Diagram 35
Figure 4.5: Resource Processing Sequence Diagram 36
Figure 4.6: Deployment Diagram of the Intelligent File Management System 37
Figure 4.7: Entity-Relationship Diagram 37
Figure 4.8: Automatic Folder Hierarchy Diagram 38
Figure 4.9: Classification and Human-Review Flowchart 40
Figure 4.10: Registration and Login Wireframe 41
Figure 4.11: Resource Library and Dashboard Wireframe 41
Figure 4.12: File and Webpage Upload Wireframe 42
Figure 4.13: Resource Details and Category Review Wireframe 42
Figure 4.14: Folder Browser Wireframe 43
Figure 5.1: Authentication and Authorisation Sequence Diagram 47
Figure 5.2: User Account Registration Interface 49
Figure 5.3: Academic Resource Upload Interface 50
Figure 5.4: Resource Library with Automatic Classification and Topic Labels 50
Figure 5.5: Hierarchical Course Folder Management Interface 51
Figure C.1: Reproduction of Figure 4.4 - Resource Upload and Organisation Activity Diagram 85
Figure C.2: Reproduction of Figure 4.5 - Resource Processing Sequence Diagram 86
Figure C.3: Reproduction of Figure 4.7 - Entity-Relationship Diagram 86
Figure C.4: Full-size reproduction of Figure 4.9 - Classification and Human-Review Flowchart 87

List of Acronyms and Abbreviations

Abbreviation	Meaning
ACM	Association for Computing Machinery
API	Application Programming Interface
ASGI	Asynchronous Server Gateway Interface
CFCOS	Content-Based File Classification and Organization System
CSS	Cascading Style Sheets
DBMS	Database Management System
DOCX	Microsoft Word Open XML document format
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
JPEG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
JWT	JSON Web Token
LLM	Large Language Model
MB	Megabyte
MP4	MPEG-4 multimedia container
OCR	Optical Character Recognition
OSI	Open Systems Interconnection
PDF	Portable Document Format
PNG	Portable Network Graphics
PPTX	Microsoft PowerPoint Open XML presentation format
REST	Representational State Transfer
SQL	Structured Query Language
SUS	System Usability Scale
TCP/IP	Transmission Control Protocol / Internet Protocol
TF-IDF	Term Frequency-Inverse Document Frequency
UI	User Interface
URL	Uniform Resource Locator
UUID	Universally Unique Identifier

1. Introduction

1.1. Study Background

Students gather academic material through learning platforms, email, messaging applications, websites and downloads. Notes, slides, PDFs, images and videos for one course often end up in different locations. As collections grow, students must decide what to retain, how to describe and organise it, and how to retrieve it later.

Personal information management involves acquiring, organising, maintaining, retrieving and using information [1]. Despite mature storage tools, file organisation and retrieval remain difficult [2], particularly when related information is fragmented across formats or applications and users must remember where it was stored [3].

Students often accumulate resources without a stable filing scheme. Inconsistent filenames, course codes buried inside documents, and topics appearing on individual pages complicate retrieval. Studies describe reactive organisation, accumulation problems and information overload [4]. Students may possess a resource yet waste time searching for it because its course, type and topic are not represented consistently.

Folders provide familiar locations; search and descriptive tags offer alternatives. Content-based classification can suggest a category from document evidence [5], but sparse text, misleading filenames and overlapping subjects create uncertainty. A useful system must expose that uncertainty and permit correction rather than silently reorganising files.

The Intelligent File Management System (IFMS) provides an authenticated, private academic library for files and webpages. It extracts available text, proposes explainable computer-science course categories and topics, and organises resources by course and material type. It stores confidence, ranked alternatives and review state, while preserving manual category corrections and folder moves. Its implementation uses a FastAPI backend, PostgreSQL/SQLAlchemy, an HTML/CSS/JavaScript interface, and configurable local or private cloud object storage.

1.2. Problem Statement

Resources for one course are frequently scattered across downloads, learning platforms, email, messaging applications, bookmarks and directories. Manual filing during other work leads to obscure filenames, inconsistent folders, duplicates and difficult retrieval.

General file systems provide storage, folders and search but do not consistently interpret mixed resources in the student's academic context. A coded PDF, a topic-labelled presentation and a related webpage may remain separate; filename search cannot reveal topics present only within

content, while one folder cannot represent every topic. Retrieval then depends on remembered names, sources or locations.

Unexplained automatic filing also risks hiding resources or overriding deliberate choices. IFMS therefore addresses the need for secure, explainable organisation of mixed files and webpages by course, material type and topic, with reviewable predictions, durable user corrections and owner-only access to sensitive academic material.

1.3. Study Objectives

1.3.1. Broad Objective

The aim of the project is to design, implement, and evaluate a secure web-based intelligent file management system that organises a student's academic files and webpages through explainable content classification, topic extraction, and hierarchical folders while preserving the student's control over automated decisions.

1.3.2. Specific Objectives

The specific objectives of the project are to:

1. Analyse the academic resource-management problem and define functional, security, privacy, and usability requirements for a personal student library.

2. Design a three-tier web architecture and relational data model for authenticated users, user-owned resources, classification results, topic labels, and hierarchical folders.

3. Implement secure ingestion, private storage, retrieval, and deletion for supported academic files and webpage resources, with validation and ownership enforcement.

4. Develop an explainable hybrid classification method and topic-extraction process that use filenames, course codes, headings, controlled terms, and document text to produce categories, confidence information, alternatives, and location evidence.

5. Implement automatic course and material-type folder organisation together with browsing, search, filtering, category review, topic editing, and persistent manual corrections or folder moves.

6. Evaluate the completed subsystem through automated functional and security tests, labelled-resource classification measures, and appropriate usability and performance checks.

1.3.3. Research Questions

The project is guided by the following questions:

RQ1. What difficulties arise when university students store and retrieve academic resources distributed across different file types and digital sources?

RQ2. How can the available filename, structural, and textual evidence in a resource be combined to assign an explainable academic course category and useful topic labels?

RQ3. How can automatic folder organisation reduce repetitive filing while preserving a student's manual category decisions and folder movements?

RQ4. What authentication, ownership, storage, and validation controls are required to keep each student's resources private and consistent?

RQ5. To what extent does the implemented system satisfy its functional requirements and expected classification, usability, performance, and security qualities?

1.4. Study Rationale

Students benefit from fewer filing decisions: IFMS proposes course categories and topics and creates predictable folders for documents, media and web resources. Browsing by course/type, searching and filtering offer multiple retrieval paths even when the original filename or location is forgotten.

Unlike rigid automation, IFMS exposes confidence, alternatives and review status. Students can revise categories or topics, rename folders and move resources; these choices survive later processing. Automation assists without replacing owner authority.

Technically, the system combines secure persistence, format-specific extraction, transparent weighted classification and hierarchical folders without external generative inference. Decisions can be traced to course codes, names, headings, repeated terms and similarity, while separated services support testing and incremental improvement.

For universities, the project demonstrates content-aware academic organisation on modest infrastructure. Owner-scoped records and private storage support responsible handling; validated resource IDs and metadata can enable future group-system integration without attributing other members' functions to this author.

1.5. Scope and Limitations

The author's subsystem covers authenticated submission, private storage, text processing, organisation, retrieval, correction and deletion. Its scope follows the implemented repository and excludes study-assistance features assigned to other group members.

1.5.1. Specific Scope

- Account registration, login, session restoration, sign-out, password hashing, JSON Web Token validation, and owner-scoped access to protected resources.
- Upload and persistent metadata for PDF, DOCX, PPTX, JPEG, PNG, and MP4 files, subject to configured type and size validation, together with capture of webpage addresses and visible page content.
- Private binary-object storage through a local provider during development or a configured private Supabase-compatible bucket for deployment.
- Format-specific text extraction from PDF pages, PowerPoint slides, Word document paragraphs, and webpage content, with page or slide locations retained where the source format permits.
- Hybrid classification against a controlled undergraduate computer-science taxonomy using course codes, filenames, headings, term evidence, and TF-IDF cosine similarity.
- Storage and presentation of the predicted category, confidence estimate, review flag, and three leading category alternatives, including user-initiated reclassification and manual correction.
- Automatic and manual topic labels with confidence values and page or slide references where available.
- Automatic Course to Material Type folder placement, nested owner-created folders, browsing, breadcrumbs, renaming, and manual movement that is preserved during later processing.
- A responsive browser interface for library viewing, search, category and resource-type filters, resource details, topic editing, folder navigation, deletion, webpage capture, and authorised PDF viewing.
- Database maintenance and backfill operations, environment-based deployment configuration, and automated tests for the main functional and ownership rules.

1.5.2. Out of Scope

- Personalised learning-resource recommendations, study schedules, reminders, progress tracking, and analytics implemented or planned by other members of the group project.
- Public file sharing, real-time multi-user collaboration, lecturer-managed class repositories, and synchronisation with external learning-management systems or personal cloud drives.
- Native Android or iOS applications and offline synchronisation; the current interface is browser based.
- Optical character recognition for scanned images, speech-to-text transcription for audio or video, and semantic interpretation of image and video content.
- A universal classifier for all academic disciplines or languages; the current taxonomy and seed evidence focus on undergraduate computer-science subjects.
- Replacement of general cloud storage, a bibliographic reference manager, or a full institutional records-management platform.

1.5.3. Limitations

The classifier uses a finite computer-science taxonomy, course codes and seed profiles; unfamiliar subjects may fall into General or require review. Broader coverage would need curated categories and representative examples without sacrificing interpretability.

Classification and topics depend on extractable text. PDFs and slides can supply page/slide evidence, but Word parsing does not supply rendered page numbers. Scanned images require unavailable OCR; images and videos are stored by type without interpreting visual or spoken content.

The local rule-and-TF-IDF approach is inspectable and avoids external model dependence, but may miss paraphrases and unfamiliar vocabulary. Its confidence values reflect evidence strength, not calibrated statistical probabilities; calibration needs a larger labelled corpus.

Webpage extraction requires accessible, useful HTML; login walls, anti-bot controls and client-side rendering may cause incomplete results or retrieval errors. The default 20 MB upload limit also restricts large media unless reconfigured.

Evaluation is constrained by limited labelled resources and participant evidence. Automated tests show API, ownership and workflow behaviour but cannot establish long-term usefulness. Any accuracy, usability, security or performance claim must remain within the tested dataset, users, sizes and environment.

The browser-based library excludes native mobile apps, offline sync, collaboration and learning-platform integration so that secure, explainable, owner-controlled organisation remains the project focus.

1.6. Ethical Considerations

Owner-scoped authentication, private storage and deletion protect user files and metadata. Evaluation uses automated tests and controlled fixtures, not completed participant usability data; Sections 3.6 and 6.5 and Appendix E describe safeguards and proposed consent procedures.

1.7. Report Organisation

The remainder of the report is organised as follows:

Chapter 2 reviews personal information management, student resource practices, classification, folders, tags, explainability, privacy, and related systems; it identifies the research gap and conceptual framework.

Chapter 3 describes the design-science and incremental development approach, requirements, tools, ethics, and methods used to evaluate functionality, classification, usability, performance, and security.

Chapter 4 presents stakeholders, requirements, use cases, architecture, data design, interfaces, algorithms, and controls for manual review and uncertainty.

Chapter 5 documents the implementation, including authentication, storage, extraction, classification, topics, folders, browser integration, and the author's contribution.

Chapter 6 describes functional, security, classification, usability, and performance evaluation procedures and results.

Chapter 7 interprets the findings against the objectives, research questions, and related work, acknowledging the evidence limits.

Chapter 8 evaluates achievement of the objectives, draws conclusions, and recommends future subsystem improvements.

2. Literature Review

2.1. Introduction

This chapter reviews personal information management, student practices, folders and tags, content classification, explainability, security and related systems before identifying the project gap and conceptual framework.

2.2. Key Concepts and Theories

2.2.1. Personal Information Management

Personal information management includes acquiring, storing, organising, retrieving and using information for individual goals [1]. File management also requires meaningful labels and locations. A review of over 200 studies by Dinneen and Julien [2] found that digital filing remains widespread, difficult and insufficiently supported.

Information is fragmented across tools: Boardman and Sasse [6] observed separate file, email and bookmark collections; Bergman et al. [3] showed the cognitive burden of dividing one project across format-based hierarchies. Academic slides, notes, webpages, images and videos face the same problem.

2.2.2. Student Academic Resource Management

Student studies confirm the problem. Robinson and Johnson [7] found a gap between actual and intended organisation practices; Ameen's survey of 221 postgraduate students [8] documented reliance on downloads, documents and URLs. Sillence et al. [4] linked reactive digital accumulation with overload and reduced productivity.

These findings justify reducing repetitive filing without removing control. Students associate resources with courses, topics and tasks rather than just extensions; folders should therefore work alongside content-derived metadata and editable organisation.

2.2.3. Folders Tags and Retrieval

Folders retain contextual locations; Barreau and Nardi [9] observed users' preference for location-based retrieval. Tags allow several labels per item without copying it. In an academic library, hierarchy answers where an item belongs, while topics, search and filters reveal its content.

Folders, tags and search complement one another: automatic course/type placement reduces initial effort, topics support discovery, and manual edits accommodate user preferences. Later processing must not undo a manual move.

2.2.4. Content Based Classification

Text categorisation maps documents to predefined classes [5]. Without a sufficiently labelled corpus, a small academic application can use a controlled course taxonomy, transparent rules and text similarity instead of claiming a fully trained statistical model.

TF-IDF weighs terms by their document frequency and their rarity across profiles [10]; cosine similarity compares document/category vectors. IFMS combines this measure with direct evidence from course codes, filenames, early headings and repeated domain phrases, forming a rule-and-similarity hybrid.

2.2.5. Explainability and Human Correction

IFMS makes predicted categories reviewable by recording alternatives, confidence and a review flag. Users retain authority to correct them, and manual choices are not silently overwritten. Visible evidence also enables investigation of classification errors.

Evidence-based explanations reveal whether a label arose from filenames, headings, vocabulary or similarity. Son and Kim [11] use LLMs for richer summaries, tags and rationales, but with greater model and processing demands. IFMS instead favours local, predictable, inspectable classification.

2.2.6. Security Privacy and Ownership

Personal notes, licensed texts and assessment files require owner-scoped authentication, authorisation and private storage. Password hashing, expiring tokens, file-type/size checks and generated object names protect accounts and prevent untrusted paths from becoming storage paths.

IFMS extracts only text needed for organisation, ties records to their owners, and supports deletion and correction. It does not send the contents to an external generative model.

2.3. Existing Systems and Related Work

2.3.1. General Cloud File Services

Google Drive and OneDrive provide mature sharing, synchronisation, folders and search [12]. Gemini can suggest organisation for eligible Drive content [13], but these services do not default to IFMS's controlled course taxonomy, page-referenced topics or locally inspectable classification scores.

2.3.2. Reference and Research Managers

Zotero supports references, hierarchical collections, tags, saved searches and some full-text indexing [14], [15]. Its bibliographic focus differs from IFMS's automatic course/type placement for mixed lectures, documents, images, videos and webpages.

2.3.3. Content Aware Research Systems

Content-aware systems organise resources using derived meaning rather than names or locations alone. CFCOS uses LLMs for summaries, tags and restructuring [11]. IFMS instead offers a bounded computer-science taxonomy, weighted local evidence, visible confidence and alternatives, and authoritative manual correction.

2.4. Comparative Analysis

Table 2.1 compares documented capabilities relevant to academic resource management, rather than ranking the products as complete platforms.

Table 2.1 Comparative analysis of related systems

System	Primary strength	Content aware organisation	Gap relative to this project
Google Drive	Cloud storage, sharing, folders, search, and optional Gemini suggestions	Available through Gemini suggestions for eligible users and files	Not centred on a fixed computer-science course taxonomy or page-referenced topic evidence
Microsoft OneDrive	Cloud storage, synchronisation, sharing, folders, and search	Search and Microsoft 365 intelligence support discovery	General-purpose organisation; no project-specific explainable course classifier
Zotero	References, collections, tags, saved searches, and selected full-text indexing	Metadata, imported subject tags, and indexed attachment text	Designed around bibliographic items rather than mixed academic course resources
CFCOS	LLM summaries, semantic tags, and content-based restructuring	Yes, using large language models	Higher model dependency and a general semantic scope rather than a lightweight academic taxonomy
Implemented subsystem	User-owned academic library with explainable classification and folders	Yes, using filenames, course codes, headings, terms, TF-IDF similarity, and topics	Current taxonomy is computer-science specific and non-text media receive limited semantic analysis

2.5. Research and Knowledge Gap

Prior work documents difficult file retrieval, fragmented student collections, and the benefits of folders and content-based access. Cloud suites emphasise scale, reference managers emphasise scholarly items, and LLM systems emphasise semantic interpretation. The gap addressed here is a lightweight, course-aware resource manager with inspectable classification evidence.

IFMS combines authenticated personal libraries; PDF, DOCX, PPTX, image, video and webpage ingestion; supported text extraction; explainable course classification; page/slide-referenced topics; and editable Course-to-Material-Type folders. It is not a replacement for general cloud storage or bibliographic tools.

2.6. Conceptual Framework

The conceptual framework extends the keep–organise–maintain–refind cycle with interpretation and correction. After validation and private storage, text evidence produces categories, topics and folder locations; users retrieve through browsing/search and may correct metadata, with manual decisions protected.

Table 2.2 Conceptual framework for intelligent resource management

Stage	System activity	User control
Keep	Validate and store a file or webpage with owner and metadata	Choose the resource and delete it when no longer required
Interpret	Extract text, headings, topics, and category evidence	Inspect confidence, alternatives, and topic references
Organise	Place the resource under Course and Material Type folders	Create or rename folders and move the resource manually
Refind	List, filter, search, browse, and serve authorised content	Select the preferred retrieval route and open the resource
Correct	Persist manual categories and topics as explicit metadata	Override or remove incorrect automated labels

2.7. Chapter Summary

The literature supports combining folders, tags, search and classification to reduce student filing effort. IFMS's proposed contribution is explainable course categorisation with page-referenced topics, owner control and automatic folders, motivating the methodology in Chapter 3.

3. Methodology

3.1. Introduction

This chapter presents the design-science and iterative development method, requirements, tooling, ethical safeguards and evaluation approach for the author's IFMS subsystem.

3.2. Research and Development Approach

An applied design-science approach was selected to build and evaluate a software artefact for fragmented student files [16]. The artefact is an authenticated personal library with content-derived organisation.

Development proceeded in Scrum-inspired increments [17]: setup, authentication, browser sessions, persistent resources, classification and topics, folders, and deployment preparation. Backlog items included features, persistence, validation, security, tests and documentation; Git recorded completed changes.

Each increment moved from a requirement through a coherent implementation, API/browser checks and automated tests to integration. Iteration exposed the need to preserve later manual corrections and folder moves without deferring privacy or persistence safeguards.

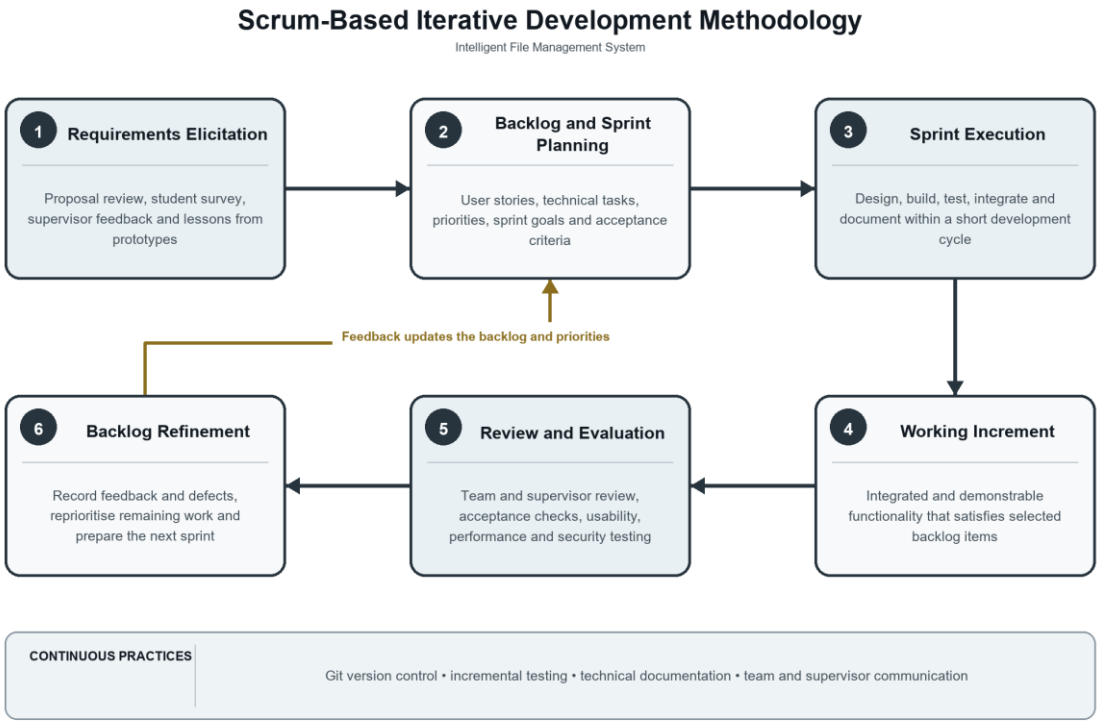


Figure 3.1: Scrum-Based Iterative Development Methodology

3.3. Requirements and Data Collection

3.3.1. Sources of Requirements

Requirements came from project objectives, supervisor guidance, reporting criteria, prototype observations and defects found in implementation/testing. A questionnaire on students' filing, retrieval, intelligent features and privacy was prepared; no numerical survey results are claimed because completed responses were unavailable.

Repository evidence separated completed IFMS features from other group members' recommendations, schedules, reminders and progress analytics.

3.3.2. Requirement Prioritisation

MoSCoW prioritisation classified authenticated upload, private metadata, retrieval and deletion as Must-have; classification, topics, course folders, manual correction and browser usability as Should-have; and advanced semantics/media processing as Could-have. Other group features were excluded.

Table 3.1 Requirement evidence and resulting subsystem decisions

Requirement source	Observed need	Resulting decision
Project objectives	Reduce manual organisation of academic files	Classify resources and create course and material-type folders automatically
Security analysis	Keep each student's resources private	Require authentication and owner-scoped database queries for every resource and folder operation
Prototype use	Automatic labels can be ambiguous	Return confidence, alternatives, a review flag, and manual correction controls
Mixed resource formats	Students keep PDFs, Word files, slides, links, images, and videos	Use format detection and extraction adapters while retaining unsupported media as organised resources
User autonomy	A personal filing choice may differ from the predicted category	Preserve manual category corrections and manual folder moves

3.4. System Architecture and Research Workflow

3.4.1. Three Tier Web Architecture

IFMS has three tiers: an HTML/CSS/JavaScript browser for account and resource workflows; a FastAPI application providing REST endpoints, validation, owner checks and processing; and PostgreSQL plus private object storage. SQLAlchemy abstracts persistence, while a storage service supports local files or private Supabase objects.

3.4.2. Resource Processing Workflow

An authenticated file/URL submission is checked for token validity, name, type, size and nonempty content. Generated owner-specific object paths prevent collisions and untrusted filenames from controlling storage locations.

PDF/DOCX/PPTX adapters extract text and page/slide evidence where supported; the webpage adapter keeps title, domain, URL and visible content after removing navigation and scripts. Classification combines course codes, names, headings, repeated terms and TF-IDF similarity, returning a category, confidence, review state and three alternatives. Topic matching saves labels, confidence and source locations.

The folder service creates or finds the course and material-type folders, unless a manual move has precedence. All read, edit, download, move and delete operations repeat owner checks before returning results.

3.4.3. Maintenance and Backfill Workflow

Startup maintenance creates missing tables and backfills topics, automatic categories and folder assignments where needed. Manual categories and moves are skipped, keeping user decisions intact through schema or taxonomy changes.

3.5. Tools and Technologies

Table 3.2 Tools and technologies used in the subsystem

Area	Technology	Purpose
Frontend	HTML5, CSS3, JavaScript	Browser interface, authentication flow, resource library, viewer, filters, classification review, and folders
Backend	Python, FastAPI, Uvicorn, Pydantic	REST API, request validation, dependency injection, response models, and application hosting
Persistence	PostgreSQL, SQLAlchemy, psycopg2	Users, resources, predictions, topics, folders, and assignments
Object storage	Local private storage or Supabase Storage	Environment-selectable storage of uploaded binary objects
Document processing	pdfplumber, python-docx, python-pptx, Requests, Beautiful Soup	Text extraction from PDF, Word, slides, and webpages
Intelligent processing	Controlled taxonomy, weighted rules, TF-IDF, cosine similarity	Explainable course categories, confidence, alternatives, and topic evidence
Security	Argon2 password hashing and JSON Web Tokens	Credential protection and expiring authenticated sessions
Testing	Pytest, FastAPI TestClient, SQLite test database	Repeatable tests for API, ownership, classification, topics, and folders

Area	Technology	Purpose
Versioning and deployment	Git, GitHub, Render configuration, environment variables	Traceable development and deployable configuration without committed secrets

3.6. Ethical and Legal Considerations

Documents may contain private or copyrighted information. Every resource/folder query uses the authenticated owner ID; content is delivered through authorised endpoints, not public upload paths. Passwords are hashed, tokens expire, and deployment secrets remain in environment configuration.

Users must hold rights to uploaded material. The personal library does not publish files and extracts text only for organisation. Users can delete items and correct topics, categories or folders so automation remains reversible.

Uncertain predictions trigger review flags and alternatives to prevent hidden misfiling. Local processing also avoids transmitting file content to external language models.

3.7. Evaluation Method

3.7.1. Functional and Security Evaluation

Functional evaluation uses automated API tests and manual browser scenarios. The repository contains seventeen automated test cases covering registration, duplicate and weak-password rejection, login and token validation, protected upload routes, resource persistence and deletion, cross-user isolation, topic extraction, manual topic editing, category separation and correction, folder placement, folder renaming, preserved manual moves, and prevention of cross-user folder operations. The test design uses an isolated SQLite database and a temporary upload directory so that cases do not depend on production data.

Manual scenarios verify the complete browser interaction: create an account, sign in, upload a supported resource, inspect its category evidence and topics, open the stored content, browse the generated folders, correct metadata, move the resource, sign out, and confirm that protected functions are no longer available. Negative cases include invalid credentials, unsupported types, empty or oversized uploads, invalid PDF signatures, expired or malformed tokens, and attempts to access another account's identifiers.

3.7.2. Classification and Topic Evaluation

The classification method should be evaluated with a labelled collection of representative computer-science resources that is separate from any examples used to tune the profiles. Suitable measures are accuracy, macro-averaged precision, recall, F1-score, top-three accuracy, and the proportion of items flagged for review. A confusion matrix can show categories with overlapping vocabulary. Topic evaluation should use a manually checked sample to determine

whether labels are relevant and whether recorded page or slide references point to supporting content.

Because the subsystem returns confidence and alternatives, evaluation should also examine calibration and review usefulness rather than category accuracy alone. A review flag is useful when it captures uncertain or incorrect cases without requiring the user to inspect every confident result. Manual corrections should remain stable after application restart and reclassification of other resources.

3.7.3. Performance and Quality Evaluation

Performance evaluation should record upload-processing time by format and file size, API response time for common library and folder operations, and behaviour under concurrent requests. Quality evaluation follows relevant ISO/IEC 25010:2023 characteristics [18], particularly functional suitability, performance efficiency, interaction capability, reliability, security, maintainability, and flexibility. The assessment should report the test environment and distinguish measured results from planned checks.

Table 3.3 Evaluation criteria

Quality area	Evidence	Success criterion
Functional suitability	Automated tests and browser scenarios	Required workflows produce the expected records, files, labels, and folders
Security	Unauthorised and cross-user requests	Protected endpoints reject missing, invalid, or non-owner access
Classification	Labelled academic resource set	Report category metrics, top-three results, review rate, and confusion matrix
Topic quality	Manually reviewed documents	Relevant labels are returned with correct page or slide evidence where available
Performance	Timed uploads and API requests	Processing remains practical for the configured 20 MB upload limit
Maintainability	Module and test inspection	Extraction, storage, classification, and folder rules remain separated and testable

3.8. Chapter Summary

Design-science iterations produced authenticated resource workflows across browser, FastAPI, PostgreSQL and private storage. Extraction feeds classification, topics and folders, with manual overrides preserved. Automated/manual checks and proposed independent evaluation are detailed next.

4. System Analysis and Design

4.1. Introduction

This chapter translates the IFMS problem into implementable requirements, interactions, architecture, data structures and processing rules. Users add resources; the system privately stores and analyses them, suggests courses/topics and creates navigable folders.

The author's implemented design covers authentication, ingestion, storage, extraction, classification, topics, folders, browsing, search, correction and deletion using FastAPI, PostgreSQL, HTML/CSS/JavaScript and local/cloud storage. Group study-assistance features are excluded.

Human control is central: the system records prediction confidence and alternatives, flags uncertain labels, and preserves manual category and folder choices. This principle drives the architecture, database, interfaces and algorithms below.

4.2. Stakeholder and User Analysis

The registered student is the sole day-to-day content owner. Other stakeholders influence assessment, deployment and maintenance but have no inherent access to private libraries.

Table 4.1: Stakeholders, Needs and Design Implications

Stakeholder	Main needs	Design implications
Student resource owner	Quick capture of files and web resources; accurate organisation; simple retrieval; privacy; ability to correct mistakes.	Owner-scoped records and storage, low-friction upload, search and filters, visible classification evidence, editable categories and topics, and preservation of manual choices.
System administrator or deployer	Predictable configuration, secure operation, recoverable failures, and manageable storage.	Environment-based database and storage settings, local and Supabase-compatible back ends, controlled upload limits, startup maintenance, and consistent database/storage cleanup.
Project supervisor and examiner	A traceable design that addresses the stated problem and can be evaluated against clear requirements.	Numbered functional and non-functional requirements, explicit scope boundaries, inspectable algorithm rules, and acceptance criteria tied to implemented behaviour.
Future maintainer	Readable separation of concerns and the ability to extend formats, taxonomies, or storage without rewriting the application.	Layered services, SQLAlchemy models, central configuration, stable folder keys, format-specific extractors, and a classifier taxonomy separated from request handling.

4.2.1. Primary User Profile

The primary user manages slides, notes, assessments, articles, images, videos and webpages, often with inconsistent names or sparse text. The interface must offer automated assistance without requiring machine-learning knowledge and retain direct manual control.

Students need to capture resources without first selecting folders; retrieve by course, type, topic, name or search; inspect and correct classifications; and keep private files inaccessible to other accounts.

4.2.2. Access and Trust Boundaries

The application treats the browser, uploaded content, filenames, web addresses, and authentication tokens as untrusted inputs. A request becomes authorised only after the API validates a signed token and resolves the current user. Database queries for resources, folders, predictions, and topics are then constrained by ownership. Stored object names use generated identifiers rather than user-supplied paths, while local storage operations verify that resolved paths remain under the configured storage root. These boundaries prevent a valid user from obtaining another user's records through guessed identifiers or crafted filenames.

4.3. Functional Requirements

Functional requirements specify the observable services the system must provide. The word ‘shall’ is used to identify a required behaviour. Each requirement is assigned an identifier so that it can later be linked to implementation evidence and testing.

Table 4.2: Functional Requirements

ID	Requirement	Acceptance evidence
FR-01	The system shall register a user with a name, unique email address, and protected password, and shall authenticate returning users.	A valid account can sign in; duplicate email and invalid credentials are rejected without exposing stored password data.
FR-02	The system shall issue and validate a session token and expose only records owned by the authenticated user.	Unauthenticated requests are rejected, and resource or folder identifiers owned by another account do not return that account's content.
FR-03	The system shall accept supported PDF, DOCX, PPTX, image, and MP4 files up to the configured size limit of 20 MB by default.	A supported file within the limit is accepted; unsupported extensions and oversized files receive a clear validation response.
FR-04	The system shall allow a user to save a webpage address as a resource and obtain usable text when the page can be retrieved and parsed.	A valid address creates a webpage resource; unavailable or unsafe input produces a controlled error rather than a partial record.
FR-05	The system shall store uploaded content using a private, generated object name and shall persist resource metadata in the database.	The database record contains owner, display name, type, size and storage reference, while the physical object cannot be addressed by a user-supplied path.

ID	Requirement	Acceptance evidence
FR-06	The system shall extract text from PDF, DOCX, PPTX, and supported webpage resources for downstream analysis.	Extracted text is stored when available, with page or slide boundaries retained where the source format permits.
FR-07	The system shall predict a course category using available filename, course-code, heading, content, and textual-similarity evidence.	A processed resource receives a predicted category, confidence, ranked alternatives, source indicator, and review status.
FR-08	The system shall identify academic topics from extracted text and retain confidence and source-location information when available.	Generated topics are associated with the correct resource and duplicate topic names for that resource are not created.
FR-09	The system shall automatically place a resource in a course folder and an appropriate material-type subfolder.	A new automatic resource assignment resolves to one folder path such as Course/Lecture Slides, Course/Images, or Course Materials.
FR-10	The system shall list the current user's resources and support search and filtering by relevant metadata.	A user can narrow the library by a filename or text query and by available category, type, topic, or folder controls.
FR-11	The system shall provide an in-application view for supported content and a secure download or open action for stored files.	Selecting a resource opens its details or supported viewer, while the content endpoint verifies ownership before returning data.
FR-12	The system shall show classification confidence, review status, alternatives, and evidence and shall allow the user to correct a category.	A manual correction updates the displayed category and marks the prediction source as manual.
FR-13	The system shall allow the user to add, edit, and remove resource topics.	Topic changes are saved only against a resource owned by the current user and appear on subsequent retrieval.
FR-14	The system shall allow the user to create, rename, browse, and navigate folders and to move resources manually.	Folder names are unique among siblings for one owner, and a manual move changes the single active folder assignment.
FR-15	The system shall preserve deliberate manual category and folder decisions during automatic reprocessing and startup maintenance.	Backfill or automatic organisation does not replace a prediction or assignment whose source is manual unless an explicit forced operation is requested.
FR-16	The system shall delete a resource and its dependent metadata and shall remove the stored object where applicable.	After deletion, the resource, prediction, topics, assignment, and storage object are no longer accessible; storage failure is handled consistently.
FR-17	The system shall backfill missing classification, topic, and folder information for older resources when the application starts.	Existing incomplete resources gain missing automatic metadata without creating duplicates or overwriting protected manual choices.

4.3.1. Business Rules

The requirements are governed by several business rules. An email address identifies one account. A resource has one owner and, at most, one current folder assignment. A folder belongs to one owner and may have one parent, allowing a hierarchy while preventing duplicate sibling

names. Each resource may have many topics but only one current category-prediction record. A manual category correction or folder move has higher authority than later automatic processing. Resource types are derived from validated input rather than accepted blindly from a client request.

Automatic processing may produce an uncertain result. Uncertainty is represented explicitly rather than treated as failure: the system stores a confidence value, ranked alternatives, and a review-required flag. A General category is permitted when the available evidence is too weak to support a specialised category. This rule prevents the application from presenting unsupported precision to the user.

4.4. Non-Functional Requirements

Non-functional requirements define the quality constraints within which the functional behaviour must operate. They are stated as design targets or verifiable safeguards rather than as unsupported claims about production-scale performance.

Table 4.3: Non-Functional Requirements

ID	Quality attribute	Requirement and verification criterion
NFR-01	Security	Passwords shall be represented with Argon2 hashes; bearer tokens shall be signature- and expiry-checked; every protected resource and folder operation shall verify ownership.
NFR-02	Privacy	Files and extracted academic text shall be private to their owner. Object paths shall use generated identifiers and shall not expose another user's content through predictable URLs.
NFR-03	Input safety	The API shall validate supported types, filename semantics, URL schemes, and the configurable upload-size limit before committing a resource.
NFR-04	Reliability	Database and object-storage changes shall be coordinated so that a failed database write does not leave an accepted orphan upload, and delete failures are reported consistently.
NFR-05	Data integrity	Foreign keys, uniqueness rules, cascading removal, and one-resource/one-assignment constraints shall prevent contradictory or detached metadata.
NFR-06	Usability	Common tasks—add, find, inspect, correct, move, and delete—shall use clear labels, visible feedback, and recoverable validation messages without requiring knowledge of the classification algorithm.
NFR-07	Responsiveness	The interface shall remain usable on common laptop and mobile viewport widths, and long filenames or metadata shall wrap or truncate without hiding essential actions.
NFR-08	Performance	Normal library listing, filtering, and metadata updates for an individual student's collection should return interactively; expensive text analysis shall avoid repeated work by persisting results.
NFR-09	Maintainability	Authentication, extraction, classification, topics, folders, storage, and API routing shall remain separated into modules with central configuration and typed database models.

ID	Quality attribute	Requirement and verification criterion
NFR-10	Portability	The application shall support local development storage and a Supabase-compatible object store through configuration without changing resource-domain logic.
NFR-11	Extensibility	New file extractors, taxonomy categories, and storage implementations should be introducible through bounded service changes rather than changes across every endpoint.
NFR-12	Auditability	Automatic and manual origins shall be retained for categories, topics, and folder assignments so that the system can explain whether a value was inferred or selected by the user.

4.5. Use Cases / User Stories

The use cases describe interactions between the authenticated student and the system. They are deliberately limited to managing resources. The administrator configures the deployment but does not participate in the student's resource-management flows. The principal actor-system interactions are summarised in Figure 4.1.

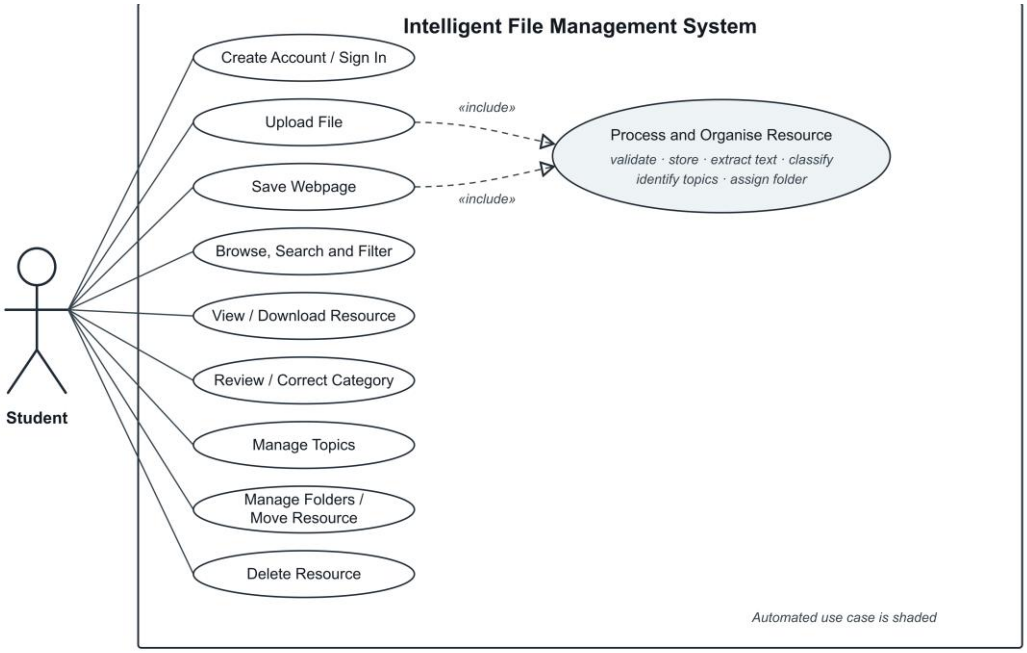


Figure 4.1: Use Case Diagram for the Intelligent File Management System

Table 4.4: Use Case Summary

ID	Use case / user story	Expected outcome
UC-01	As a student, I want to register and sign in so that my resource library remains private.	An authenticated session is established and only the student's own data is returned.
UC-02	As a student, I want to upload a file without first choosing a folder so that organisation is automatic.	The file is validated, stored, analysed, categorised, tagged with topics, and assigned to a course/material folder.

ID	Use case / user story	Expected outcome
UC-03	As a student, I want to save a useful webpage so that online material appears beside uploaded resources.	The address and extracted page text are stored as a webpage resource and processed by the same classification pipeline.
UC-04	As a student, I want to search and filter my library so that I can retrieve a resource quickly.	The resource list is narrowed using query text and selected metadata without exposing other accounts.
UC-05	As a student, I want to inspect and correct a category so that an uncertain prediction does not damage my organisation.	Evidence, confidence, alternatives, and review status are visible; the saved manual category is protected.
UC-06	As a student, I want to manage topics so that resource descriptions reflect the concepts important to me.	Automatic topics can be supplemented, renamed, or removed within the owned resource.
UC-07	As a student, I want to browse and manage folders so that I can adapt the automatic hierarchy.	Folders can be created or renamed and a resource can be manually moved to one selected folder.
UC-08	As a student, I want to open or download a stored resource so that I can use it from its organised record.	The system validates ownership and returns the supported view or file content.
UC-09	As a student, I want to delete a resource so that obsolete material and its metadata are removed.	The record, dependent metadata, assignment, and stored object are removed from the student's library.

4.5.1. Detailed Use Case UC-02: Upload and Organise a File

Primary actor: Authenticated student.

Preconditions: The user has a valid session; the selected file is available to the browser; the API, database, and configured storage provider are reachable.

Trigger: The user selects a file and confirms the upload action.

Main success flow:

1. The client submits the file with the current bearer token.

2. The API validates the session, filename, supported type, and size limit.

3. The storage service creates an owner-specific generated object name and stores the file privately.

4. The extraction service obtains text and location information supported by the file format.

5. The classifier combines available metadata and text signals to produce a category, confidence, alternatives, and review status.

6. The topic service identifies controlled or plausible topic labels and their evidence locations.

7. The folder service resolves the course folder and a material-type subfolder and creates missing system folders.

8. The application commits the resource, prediction, topics, and folder assignment and returns the organised resource to the client.

Alternative flows: An unsupported or oversized file is rejected before storage. If extraction yields little text, the classifier uses the remaining signals and may select General with review required. If a database operation fails after a new object is stored, the service attempts to remove the object so that an untracked upload is not left behind.

Postconditions: On success, one owner-scoped resource is retrievable with its processing metadata and current folder assignment. On failure, the user receives a controlled error and the system avoids presenting a partially accepted resource.

4.5.2. Detailed Use Case UC-05: Review and Correct a Category

Primary actor: Authenticated resource owner.

Preconditions: The resource exists, belongs to the current user, and has a category prediction record or can be processed to create one.

Main success flow: The user opens the category details, reads the predicted category, confidence, review status, ranked alternatives, and available evidence, selects a more appropriate category, and saves the change. The service validates ownership, updates the resource category and prediction, records the source as manual, and reorganises the automatic folder only where doing so does not override a protected manual folder move.

Postconditions: The corrected category is shown in subsequent lists and details. Automatic startup processing does not replace the manual decision. The retained source label makes the human intervention auditable.

4.6. System / Solution Architecture

The solution uses a layered web architecture. The presentation layer runs in the browser and communicates with the application through HTTP requests. The application layer is implemented with FastAPI routes and focused domain services. The persistence layer combines a relational PostgreSQL database for searchable metadata with object storage for uploaded binary content. Separating metadata from file objects avoids storing large binaries in relational tables while preserving transactions and constraints for the information used by search, classification, and organisation. The resulting component boundaries and communication paths are shown in Figure 4.2.

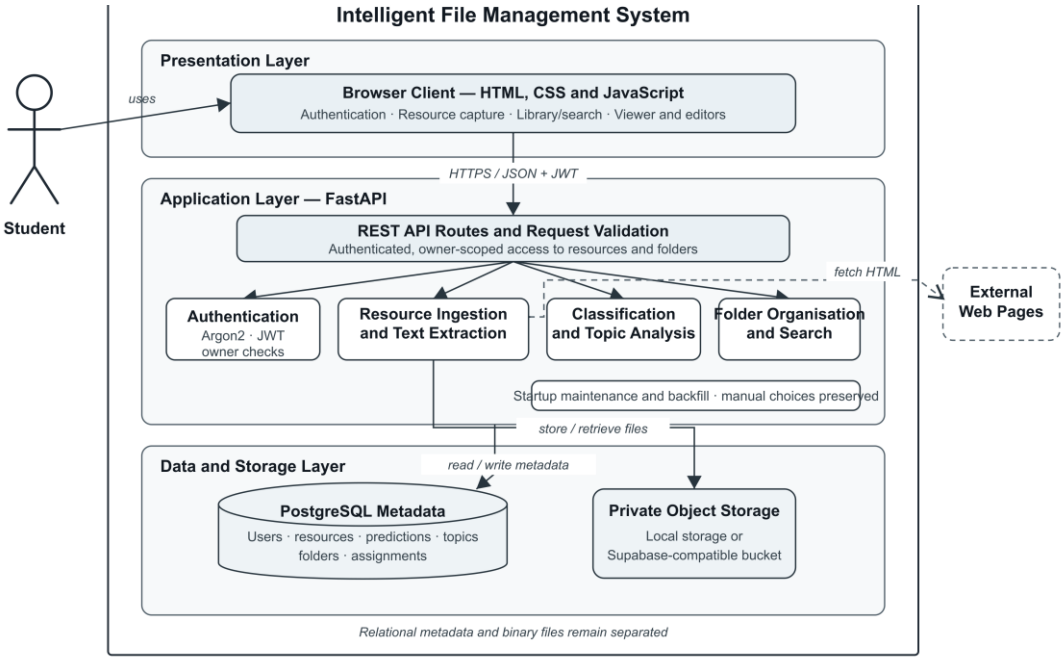


Figure 4.2: High-Level System Architecture

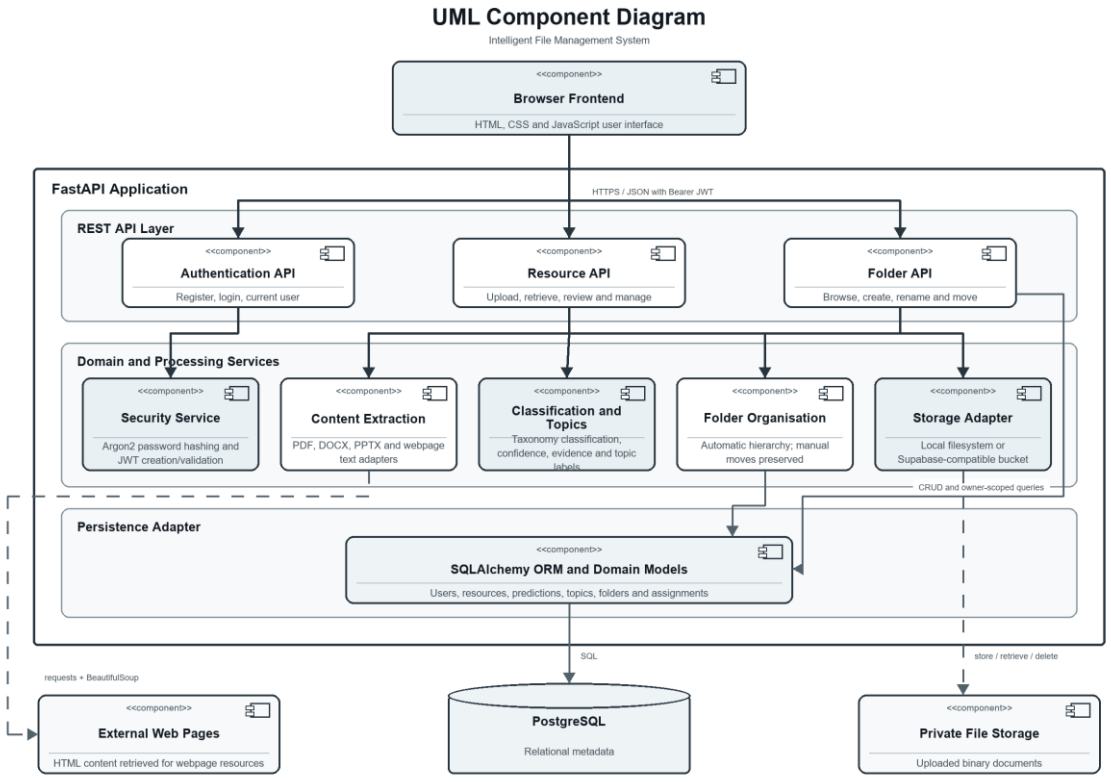


Figure 4.3: UML Component Diagram of the Intelligent File Management System

Table 4.5: Architectural Components and Responsibilities

Layer / component	Responsibilities	Key design boundary
Browser client	Authentication forms, upload and webpage capture, library lists, filters, resource details, PDF viewing, category review, topic editing, and folder navigation.	Uses API responses and tokens; it is not trusted to decide ownership or final validation.
FastAPI route layer	Receives requests, validates schemas and authentication dependencies, maps domain outcomes to HTTP responses, and serialises resource data.	Keeps transport concerns separate from extraction, classification, storage, and folder rules.
Authentication service	Hashes and verifies passwords, issues signed JWTs, resolves the current user, and supplies the owner identity to protected operations.	Plain-text passwords are never persisted; token claims are verified before owner-scoped queries.
Resource orchestration	Coordinates upload or webpage capture, metadata creation, extraction, classification, topic generation, folder assignment, and failure cleanup.	Provides one workflow over several services while avoiding algorithm logic in routes.
Extraction service	Selects a format-specific extractor and returns text with page, slide, or logical-location information when available.	Binary parsing errors are contained and do not grant trust to embedded content.
Classification service	Scores taxonomy categories, calculates confidence and margin, stores alternatives, and marks uncertain predictions for review.	Produces advice and evidence; manual user decisions remain authoritative.
Topic service	Matches controlled aliases and plausible headings, aggregates occurrences and source locations, and supports manual topic changes.	Topic names are unique per resource and retain automatic or manual source.
Folder service	Creates owner-specific course and material folders, derives a destination from category and type, and preserves manual assignments.	Stable system keys separate internal identity from a user-visible folder name.
SQLAlchemy/PostgreSQL	Stores users, resources, predictions, topics, folders, and assignments with relational constraints.	Transactions protect metadata consistency and ownership relationships.
Storage provider	Stores, retrieves, and deletes private file objects using either a local root or a Supabase-compatible bucket.	Generated owner-scoped keys and safe path resolution prevent arbitrary filesystem access.

4.6.1. Resource Ingestion and Processing

A file ingestion request passes through authentication and validation before it reaches storage. Once the object is safely stored, the orchestrator creates the resource metadata and invokes text extraction. Classification and topic extraction use the same normalised text so that binary parsing is not repeated. The folder destination is derived from the accepted category and resource type. Database changes are committed as a coherent unit; if persistence fails after the

object has been created, cleanup is attempted through the same storage abstraction. Figure 4.4 presents this upload and automatic-organisation activity flow.

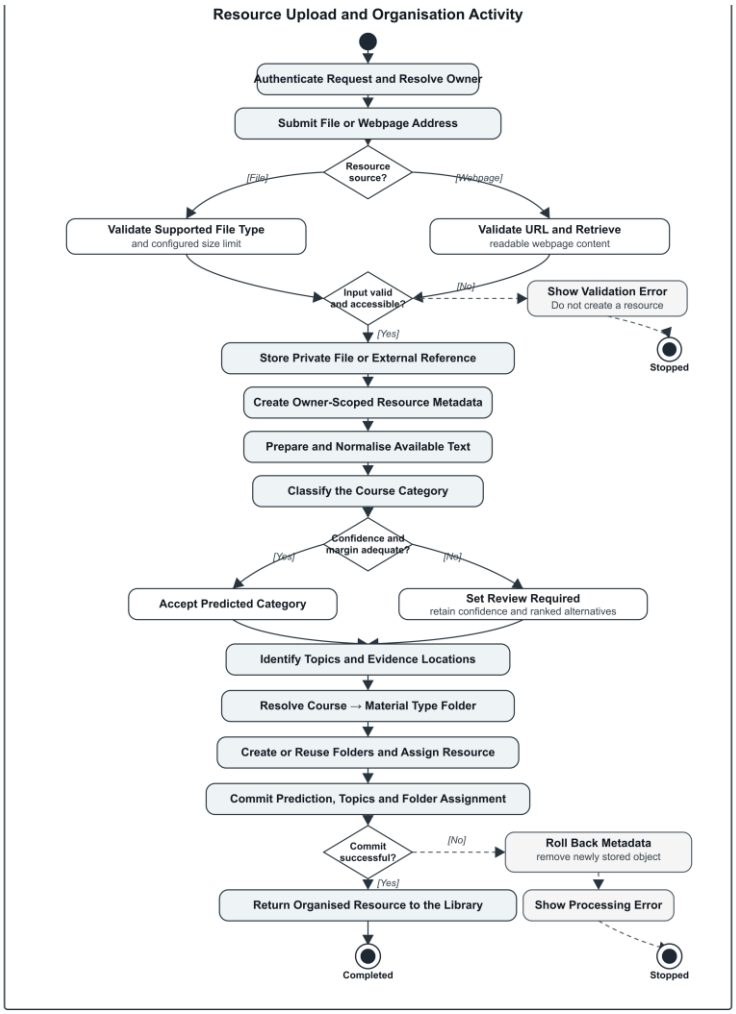


Figure 4.4: Resource Upload and Organisation Activity Diagram

Web resources follow a related path but store an external address instead of an uploaded binary. The service retrieves and parses readable page content, records the resource type as a webpage link, and passes the resulting text into classification and topic extraction. Retrieval failures are reported as controlled application errors. The stored address remains part of the resource metadata so that the user can return to the original source. The service interactions shared by the processing workflow are detailed in Figure 4.5.

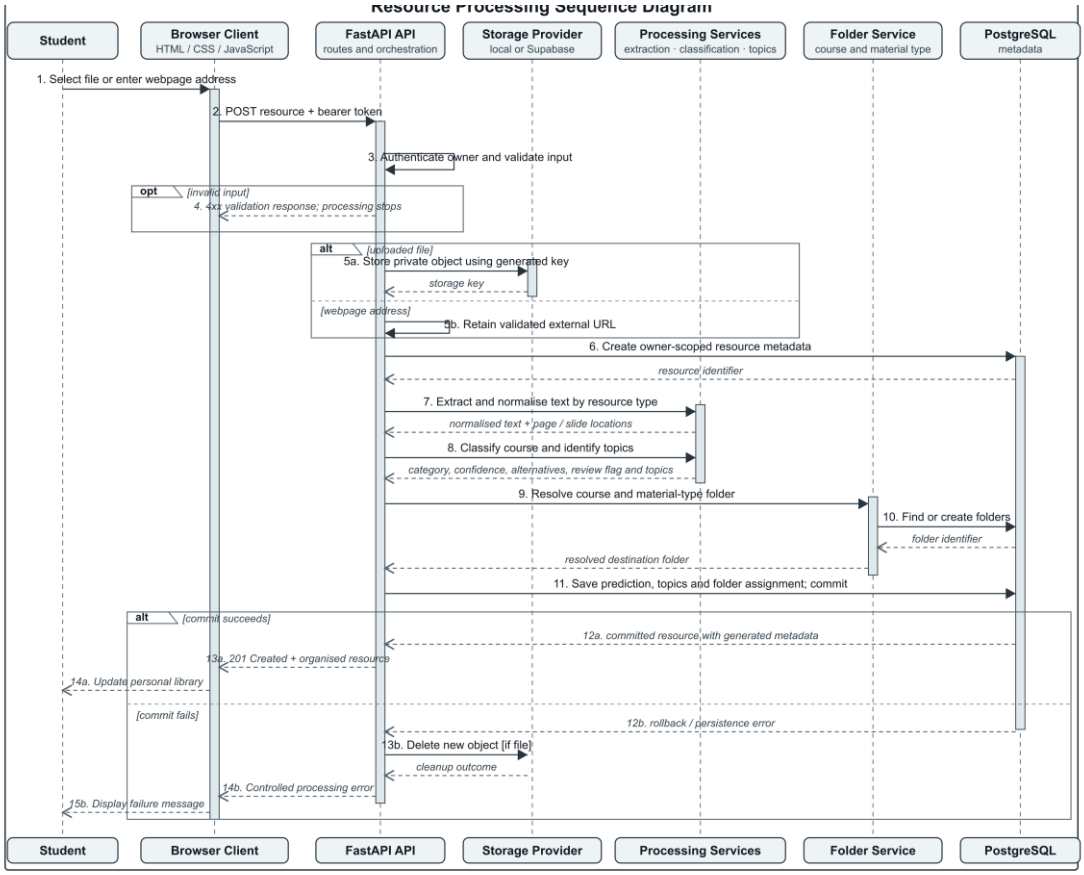


Figure 4.5: Resource Processing Sequence Diagram

4.6.2. Retrieval and Maintenance

Owner-scoped queries apply search and filters to personal resources; detail and content paths repeat authorisation. Startup maintenance backfills missing categories, topics and folders without creating duplicates or overriding manual decisions.

4.6.3. Deployment Configuration

Configuration defines database, token expiry, client origin, upload limit and storage credentials. The same storage interface supports local objects or a private Supabase-compatible bucket without changing route and processing logic.

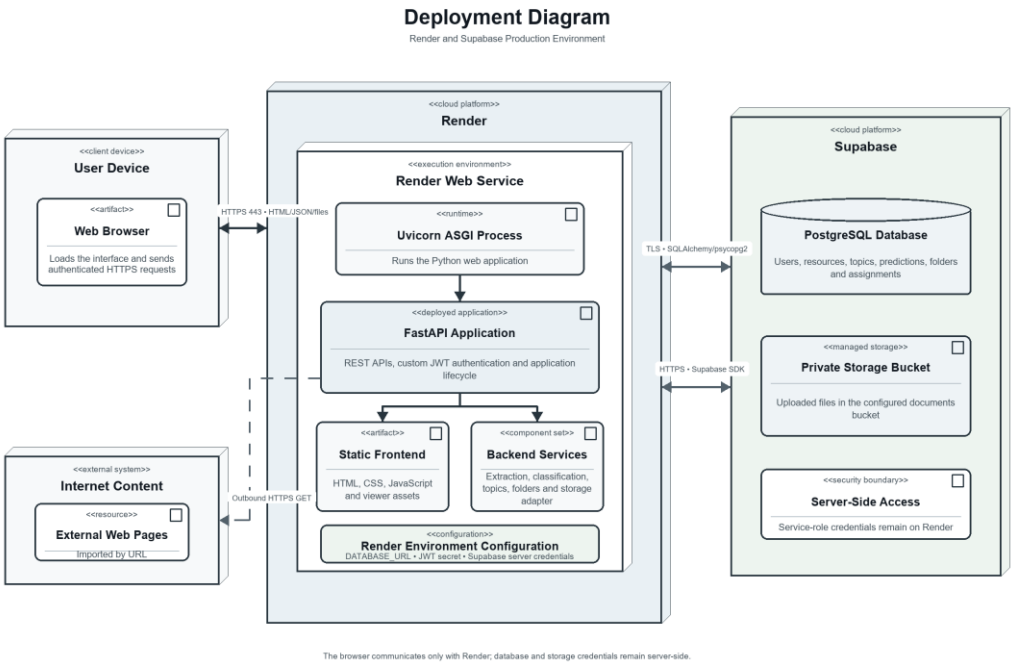


Figure 4.6: Deployment Diagram of the Intelligent File Management System

4.7. Database / Data Design

The data model separates resources from analysis and organisation, with user ownership propagated through dependent records. Keys and cascades keep topics, predictions and assignments consistent. Figure 4.7 shows the entities and relationships.

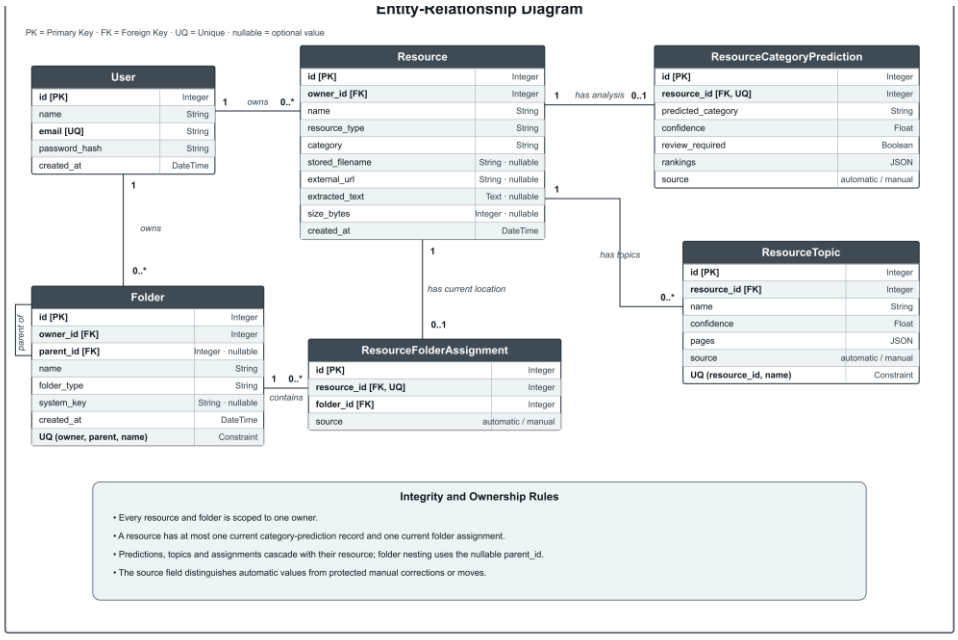


Figure 4.7: Entity-Relationship Diagram

Table 4.6: Core Data Entities

Entity	Purpose and principal attributes	Relationships and constraints
User	Stores name, unique email, password hash, and creation time.	One user owns many resources and folders. Email is unique; password plaintext is not stored.
Resource	Stores owner, display name, resource type, category, generated stored filename or external URL, extracted text, byte size, and creation time.	Belongs to one user; has topics, at most one prediction, and at most one current folder assignment.
ResourceCategoryPrediction	Stores predicted category, confidence, review-required status, ranked alternatives, and automatic or manual source.	One-to-one with Resource through a unique resource foreign key; removed when the resource is deleted.
ResourceTopic	Stores topic name, confidence, source locations, and automatic or manual source.	Many topics belong to one Resource; resource/topic-name combinations are unique.
Folder	Stores owner, optional parent, display name, folder type, stable system key, and creation time.	Self-referencing hierarchy; sibling names are unique for an owner; owned folders cannot be shared implicitly.
ResourceFolderAssignment	Stores the resource's current folder and whether the assignment is automatic or manual.	Resource foreign key is unique, enforcing one current folder; assignment is removed with the resource.

4.7.1. Relationship and Integrity Rules

Users own many resources and folders; a resource has many topics but at most one current prediction and one folder assignment. Self-referencing folders form the parent–child hierarchy. Uniqueness rules prevent duplicate predictions, topics and automatically created sibling folders. Relational cascades delete dependent metadata, while the application separately deletes stored binary objects. Figure 4.8 depicts the hierarchy.

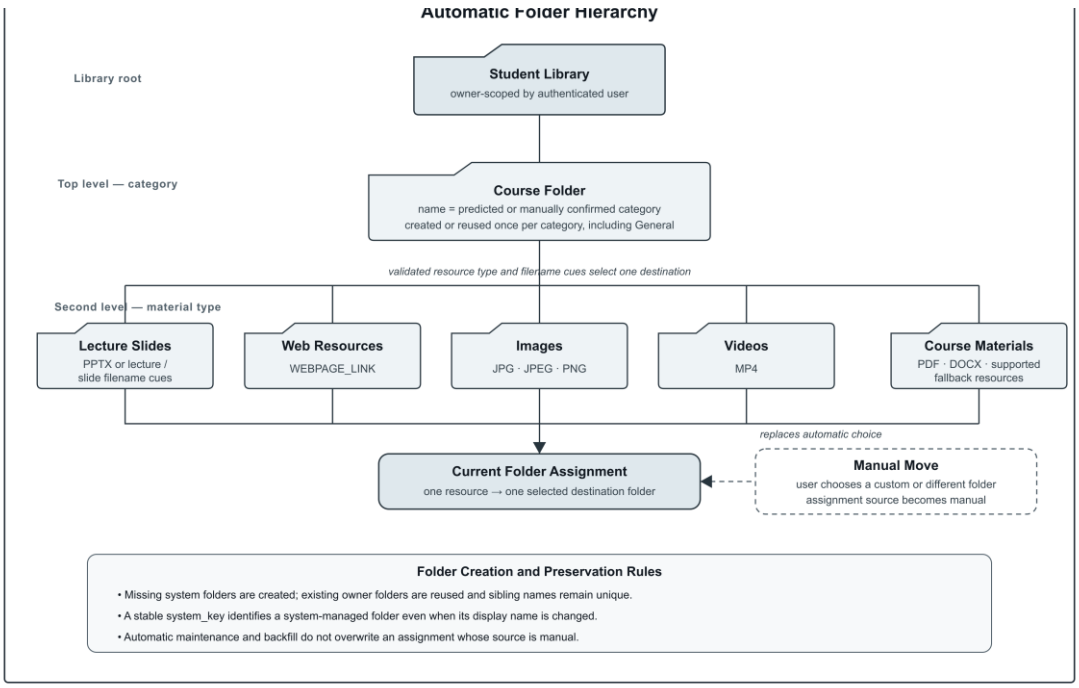


Figure 4.8: Automatic Folder Hierarchy Diagram

4.7.2. Data Ownership Queries

Resource and folder identifiers never establish access alone: each API lookup verifies the authenticated owner, including parent resources and target folders, preventing insecure direct-object-reference access.

4.8. Interface / Interaction Design

The browser presents authentication, personal resources, retrieval, correction, movement and deletion as user tasks rather than service boundaries. It translates scores into reviewable confidence and alternatives (Figure 4.9); interface states appear in Figures 4.10–4.12.

Table 4.7: Interface Areas and Interaction Decisions

Interface area	Primary actions and information	Interaction decision
Registration and sign-in	Create account, enter credentials, receive validation, establish session.	Forms use explicit labels and do not reveal whether a password or stored hash matched internally.
Library dashboard	View owned resources, search, apply category/type/topic/folder filters, open a resource, or begin capture.	High-frequency retrieval controls remain close to the resource list; empty and error states explain the next action.
File and webpage capture	Choose a local file or enter a webpage address, submit, and receive success or validation feedback.	Supported formats and the size limit are stated before submission; failures do not appear as successful resources.
Resource detail and viewer	See name, type, category, topics, folder, dates, extracted evidence, and open or download content.	Metadata and actions are grouped around one resource; supported PDFs can be viewed without leaving the application.

Interface area	Primary actions and information	Interaction decision
Category review	Inspect confidence, review flag, ranked alternatives, evidence, correct a category, or request reclassification.	Uncertainty is visible and editable; the user is not forced to accept an automatic prediction.
Topic editor	Add, rename, or remove topic labels and review generated locations.	Manual topic changes are direct and owner-scoped; automatic confidence does not prevent correction.
Folder browser	Navigate hierarchy, create or rename a folder, inspect contents, and move a resource.	Automatic folders provide an immediate structure while manual moves remain protected from later background organisation.

4.8.1. Feedback and Error Handling

The client distinguishes invalid credentials/files, session expiry, missing content and service errors without exposing credentials or traces. Destructive actions require confirmation; views refresh after successful API responses.

4.8.2. Consistency and Accessibility

Consistent terminology, accessible keyboard controls, readable contrast, visible focus, responsive layout and review status that is not conveyed solely by colour support usability. Long names must remain inspectable.

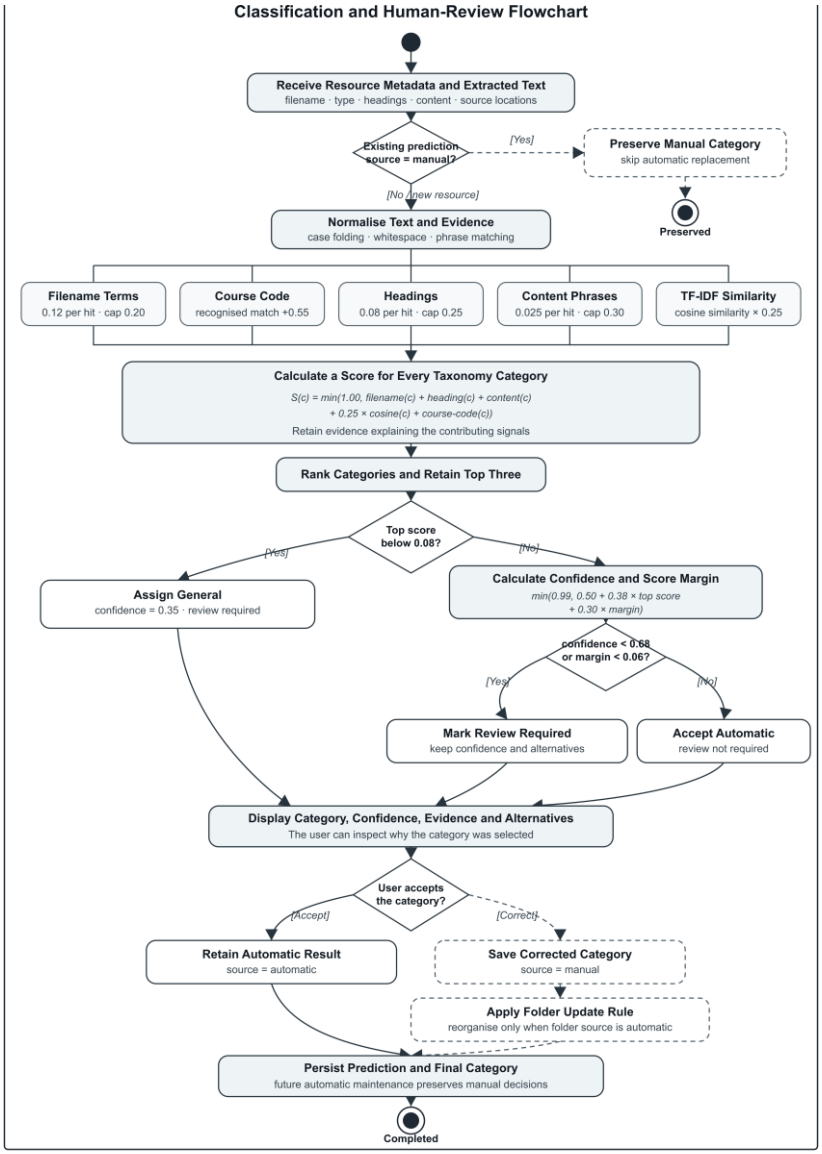


Figure 4.9: Classification and Human-Review Flowchart

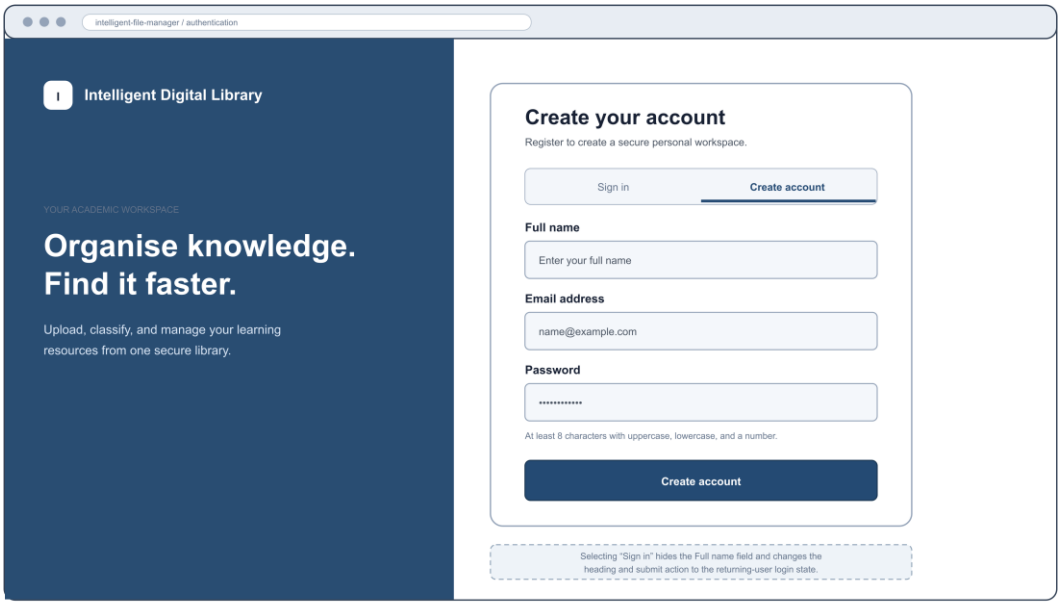


Figure 4.10: Registration and Login Wireframe

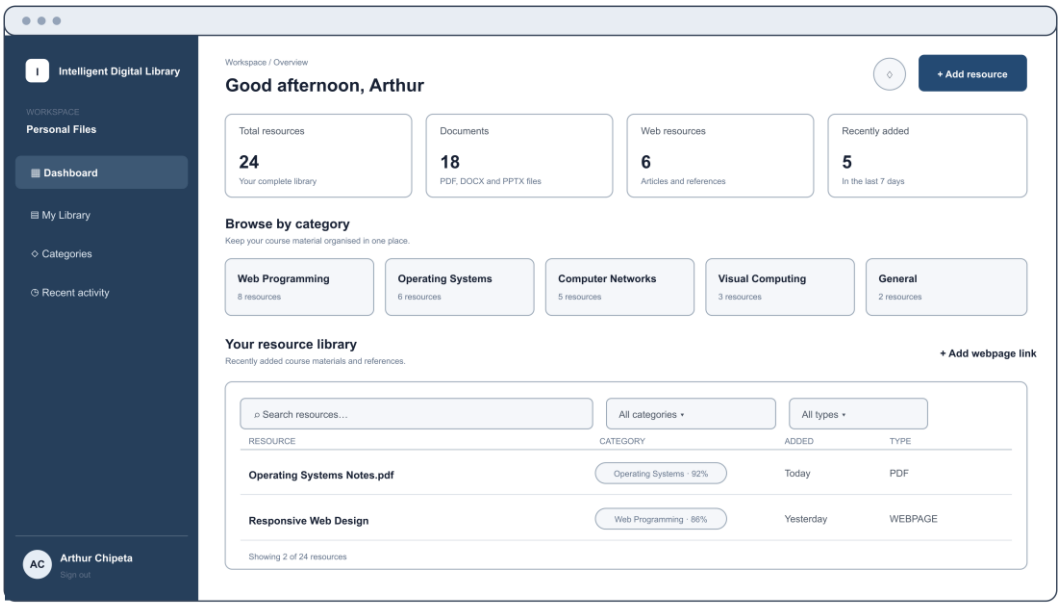


Figure 4.11: Resource Library and Dashboard Wireframe

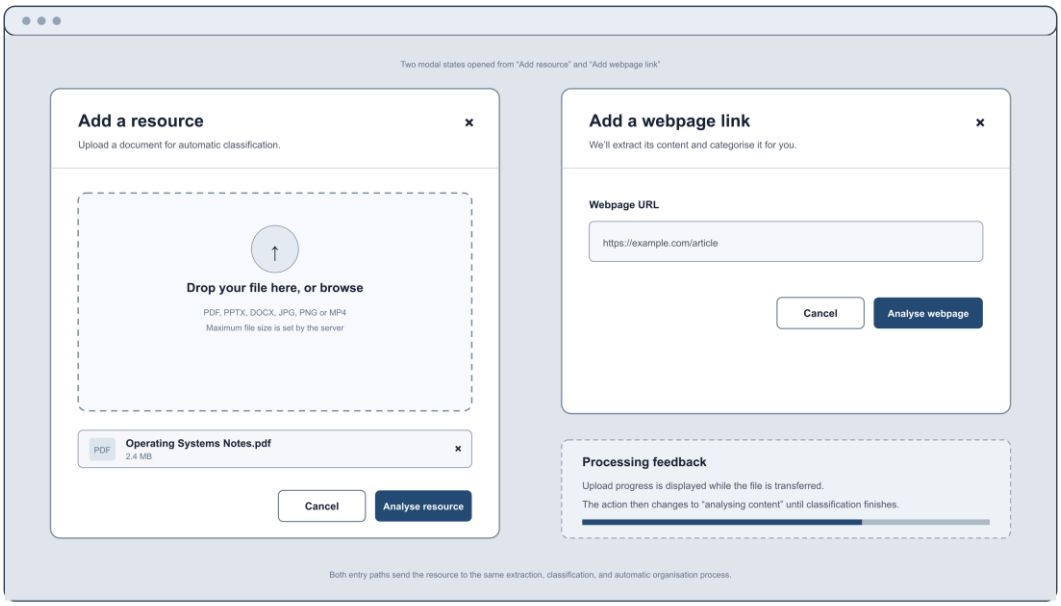


Figure 4.12: File and Webpage Upload Wireframe

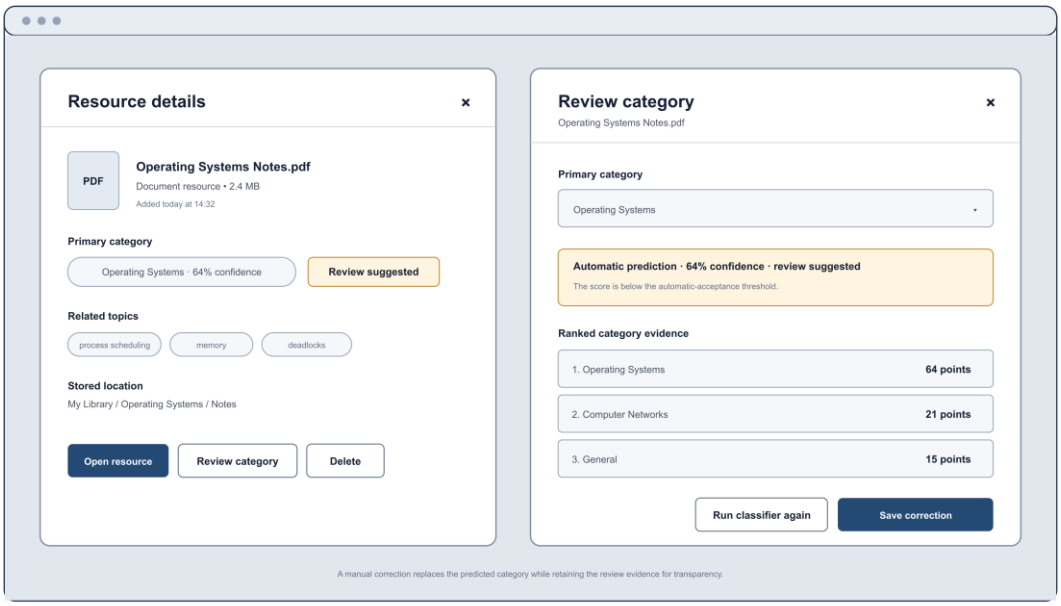


Figure 4.13: Resource Details and Category Review Wireframe

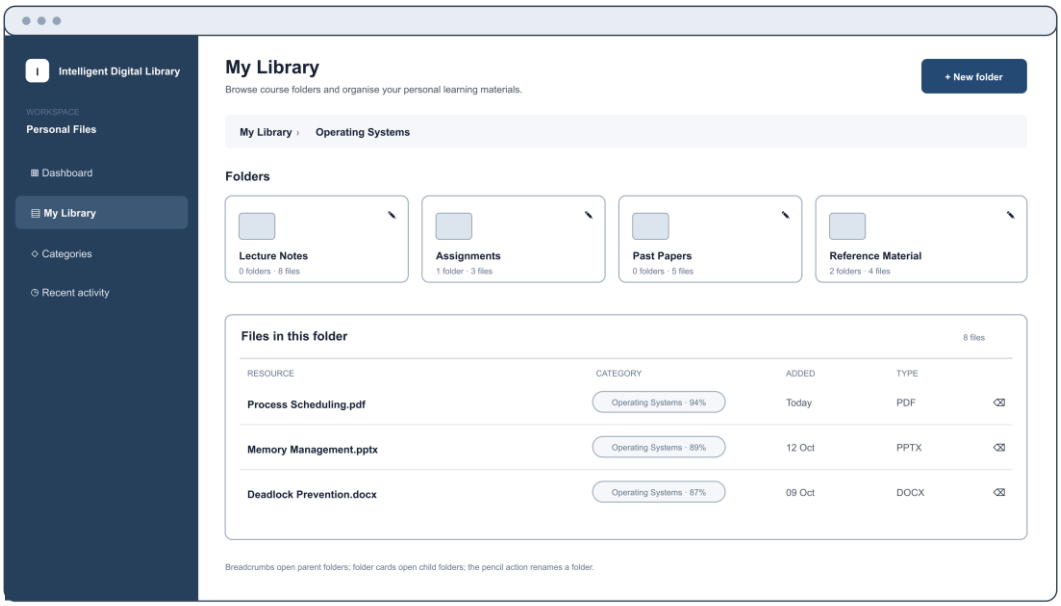


Figure 4.14: Folder Browser Wireframe

4.9. Algorithms / Models / Technical Design

Deterministic services and lightweight text analysis keep categorisation traceable and reduce reliance on remote models; sparse or unreadable documents must fail safely.

4.9.1. File Validation, Naming and Storage

Uploads validate the displayed filename, mapped file type and byte limit but generate an owner-scoped UUID key for storage. Local paths are constrained to the configured root; cloud objects use private bucket keys.

1. Authenticate the request and resolve the current owner.
2. Validate the file name, extension, mapped resource type, and total size.

3. Generate an owner-scoped UUID object key and store the binary through the selected provider.
4. Create metadata and run extraction, classification, topic, and folder services.
5. Commit the database transaction; on failure, roll back metadata and attempt object cleanup.

4.9.2. Format-Specific Text Extraction

Extraction dispatches PDF by page, presentations by slide, Word into parser-available text units, and webpages into readable HTML text. Images/videos remain managed resources without semantic extraction. Results carry source locations when possible.

4.9.3. Hybrid Course Classification

Classification scores each controlled course category using capped filename, heading and phrase evidence plus TF-IDF similarity; recognised course codes add a stronger signal.

$$S(c) = filename(c) + heading(c) + content(c) + 0.25 \times cosine(c) + course\text{-}code(c)$$

In the implemented weighting, filename evidence is capped at 0.20, heading evidence at 0.25, and content-phrase evidence at 0.30. TF-IDF similarity contributes up to 0.25, while a recognised course code adds 0.55. The combined category score is capped at 1.00. The highest-scoring category becomes the proposed result, but the decision retains the top three ranked categories for review.

$$confidence = \min(0.99,\ 0.50 + 0.38 \times top\ score + 0.30 \times score\ margin)$$

The score margin is the difference between the best and second-best category. A result is flagged for review when confidence is below 0.68 or the margin is below 0.06. If the best raw score is below 0.08, the classifier selects General with a confidence of 0.35 and requires review. These thresholds prevent weak evidence from being presented as a confident specialist category. A manual correction changes the prediction source to manual so that maintenance processing does not silently reverse it. These thresholds and the protected manual-correction path correspond to the review flow in Figure 4.9.

4.9.4. Topic Extraction

Topic extraction normalises controlled aliases, aggregates occurrences and page/slide locations, and raises confidence with repeated evidence. Plausible headings receive lower confidence; results are ranked and capped to avoid noisy labels.

$$topic\ confidence = \min(0.99,\ 0.58 + 0.06 \times occurrences + 0.03 \times distinct\ locations)$$

Automatic and manual topics share the same resource relationship but retain their source. The resource/topic uniqueness rule prevents repeated aliases from creating duplicates. Users can add or correct a label when the controlled vocabulary does not capture the intended concept.

4.9.5. Folder Resolution and Manual-Choice Preservation

Folders use the accepted course as the top level and the resource type to select Lecture Slides, Web Resources, Images, Videos or Course Materials. Owner-scoped creation and stable system keys preserve renamed automatic folders (Figure 4.8).

Only automatic assignments may be updated by subsequent automatic organisation; manual moves and category corrections take precedence unless an explicitly forced operation is applied.

4.9.6. Security and Failure-Control Logic

Argon2 hashes, expiring HS256 JWTs and owner checks protect account operations. Rollback and attempted binary cleanup handle failed database stages. Clients cannot grant access through identifiers alone.

4.10. Chapter Summary

Design analysis defined the student-owner model, seventeen functional and twelve quality requirements, use cases and a browser–FastAPI–PostgreSQL/private-storage architecture. Its data model links resources to predictions, topics and one folder assignment under owner and integrity constraints.

Format adapters, weighted classification, topics, folders, review thresholds and preserved manual choices form a traceable, testable design isolated from unrelated group features.

5. Implementation and Development

5.1. Introduction

This chapter documents IFMS technologies, modules, implemented workflows, integration, challenges and individual contribution, based on the authored subsystem repository.

5.2. Development Environment

FastAPI/Pydantic implement APIs and validation; Uvicorn serves ASGI; SQLAlchemy with psycopg2 connects to PostgreSQL. Automated tests substitute isolated in-memory SQLite.

An HTML5/CSS3/JavaScript frontend is served as static files from FastAPI, without a separate build pipeline. It stores a session token for protected same-origin requests and works locally or on the configured host.

Local storage uses a configured upload root; hosted storage can use a private Supabase bucket. Environment settings select PostgreSQL, JWT, upload limit (default 20 MB) and storage credentials. Git/GitHub track changes, with Render/Supabase deployment configuration.

5.3. System Components and Modules

Table 5.1 Implemented subsystem modules

Module	Main responsibility	Key implementation location
Authentication	Registration, login, password verification, JWT issue and current-user resolution	api/auth.py, services/security.py, models/auth.py, models/user.py
Resource API	Upload, webpage capture, list, retrieve, serve, delete, recategorise, and edit topics	api/upload.py
Persistence	Database engine, sessions, users, resources, predictions, topics, folders, and assignments	database.py and models
Extraction	Read text from PDF pages, Word documents, PowerPoint slides, and webpages	services/pdf_service.py, docx_service.py, pptx_service.py, webpage_service.py
Classification	Score a controlled taxonomy and return evidence, confidence, alternatives, and review status	services/classification_service.py
Topics	Match controlled aliases and plausible headings with confidence and page references	services/topic_service.py
Folders	Create hierarchical folders, assign resources, browse, rename, and preserve manual moves	api/folders.py, services/folder_service.py, models/folder.py
Storage	Upload, download, and delete local or Supabase objects through one interface	services/storage_service.py

Module	Main responsibility	Key implementation location
Browser interface	Authentication screens, dashboard, filters, viewer, metadata editing, and folder navigation	frontend/index.html, script.js, style.css

5.4. Key Implementation Details

5.4.1. Authentication and Ownership Enforcement

Registration normalises names/email and requires an eight-character password with upper/lowercase letters and digits. Argon2 hashes passwords; login issues an HS256 JWT containing user ID, issue time and expiry (30 minutes by default). Hashes are never returned. Protected routes validate the bearer token and load the current account. Resource/folder queries require matching owner IDs, returning the same not-found response for absent and cross-user identifiers.

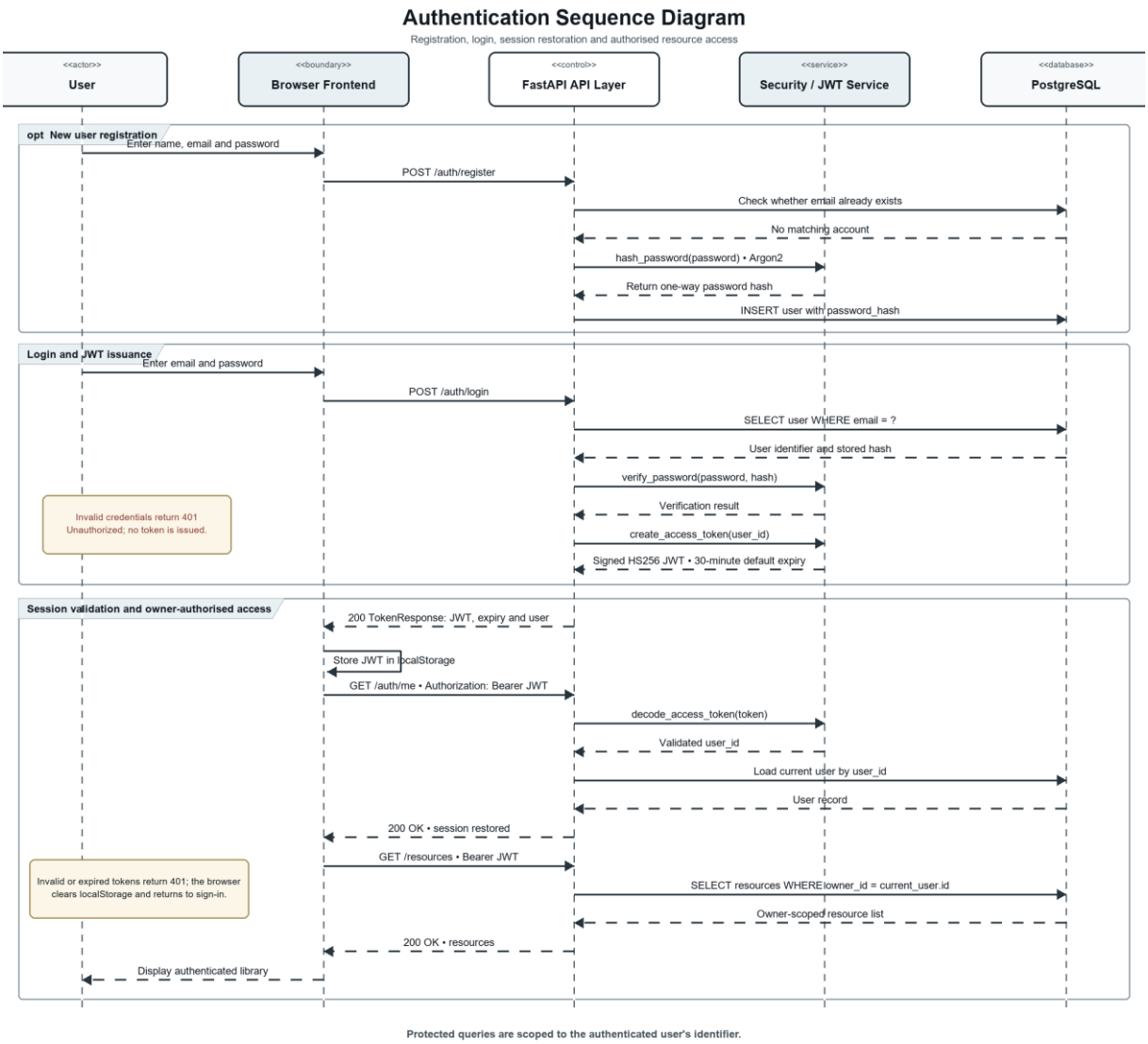


Figure 5.1: Authentication and Authorisation Sequence Diagram

5.4.2. Resource Ingestion and Private Storage

Uploads accept PDF, DOCX, PPTX, JPEG, PNG and MP4, reject empty/oversized bodies and check PDF signatures. Original names remain metadata; generated owner-scoped UUID paths prevent collisions and unsafe filenames.

A common storage service uploads, downloads and deletes from a confined local root or private Supabase bucket. On later database failure, the transaction rolls back and the binary is removed where possible.

Webpage capture fetches a URL with timeout, removes non-content HTML and records title, domain, submitted address and visible text. It has no binary object but shares classification, topics, folders and owner rules with uploaded files.

5.4.3. Format Specific Text Extraction

pdfplumber extracts PDF pages; PowerPoint parsing retains slide numbers; python-docx provides paragraphs but not rendered page boundaries. All adapters return a shared page/text representation.

Images and videos are organised by resource type without content interpretation. For reprocessing, previously stored extracted text can be reused if binary retrieval is unavailable.

5.4.4. Explainable Hybrid Classification

The classifier has over twenty-five computer-science profiles and selected course-code mappings. It normalises filenames, reads the first three nonempty lines per page/slide as headings, and analyses extracted text.

Each category receives capped filename, heading, term and TF-IDF evidence, with a strong recognised course-code weight. Deterministic ranking selects a category or General when evidence is weak; confidence uses the leading score and runner-up margin, which may trigger review.

The current prediction retains category, confidence, review status, top-three alternatives and source. A user-selected replacement is saved as manual with confidence 1.0; startup reclassification skips it.

5.4.5. Topic Labels with Location Evidence

Topic extraction matches controlled aliases spanning APIs, databases, operating systems, routing, security, data structures, machine learning and graphics. Repetition and source pages affect confidence; plausible headings contribute additional candidates. Up to ten labels retain page/slide references.

Users can add or remove topics, optionally recording valid page numbers. Uniqueness prevents duplicate names, and the PDF viewer links topic chips to the first supporting page.

5.4.6. Hierarchical Folder Organisation

Automatic filing creates owner-scoped course folders and material subfolders: Lecture Slides, Course Materials, Web Resources, Images or Videos. Stable system keys prevent renamed automatic folders from being recreated.

Each resource has one automatic or manual assignment. Reclassification can update automatic locations but not manual moves. Folder APIs enforce ownership and expose counts, breadcrumbs, browsing and edits.

5.4.7. Browser Interface

The browser supports registration, sessions, file/URL submission, search, filters, resource editing/deletion, category review, topics and folders. Bearer-protected APIs supply confidence, review flags and topic chips; the PDF viewer uses an authorised content endpoint.

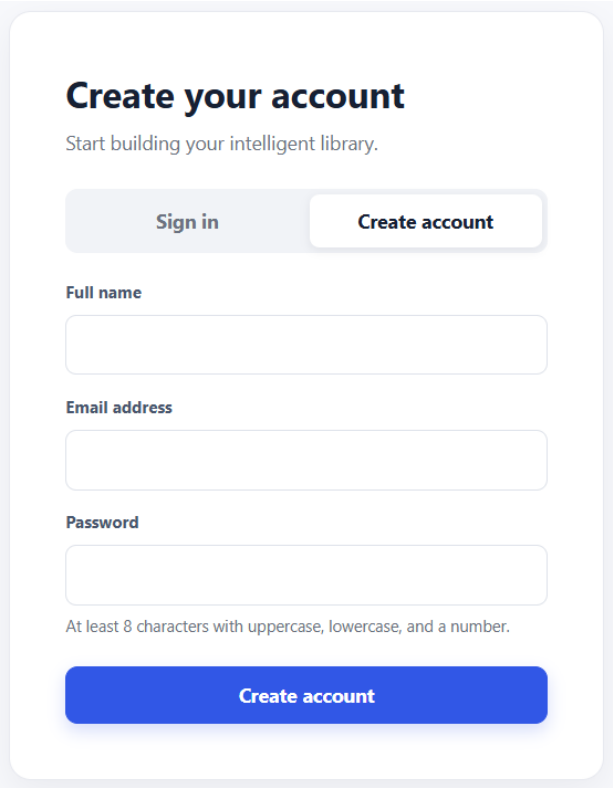


Figure 5.2: User Account Registration Interface

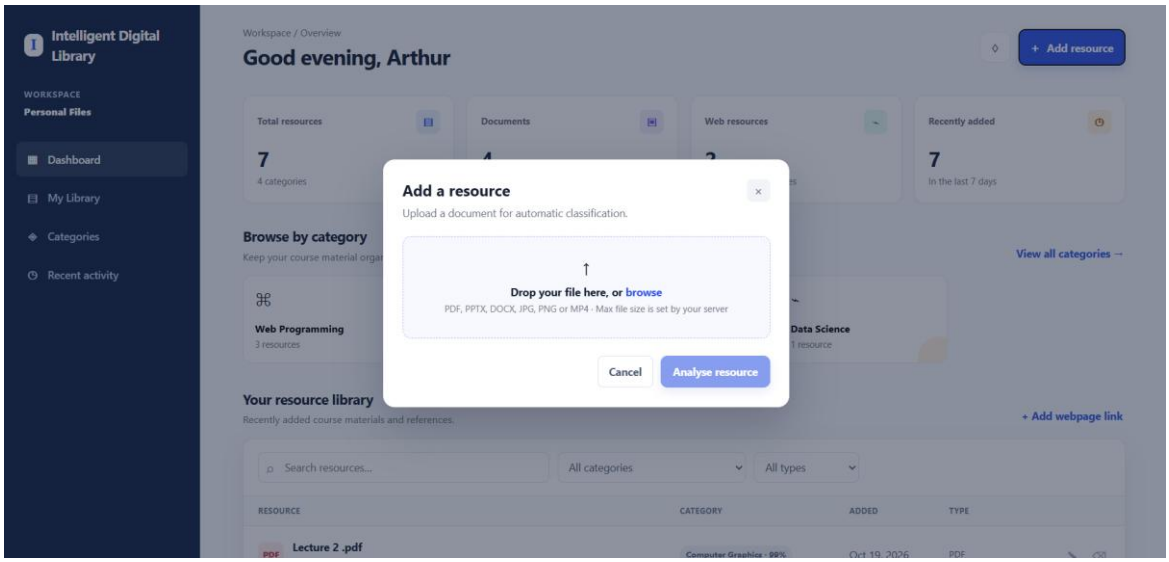


Figure 5.3: Academic Resource Upload Interface

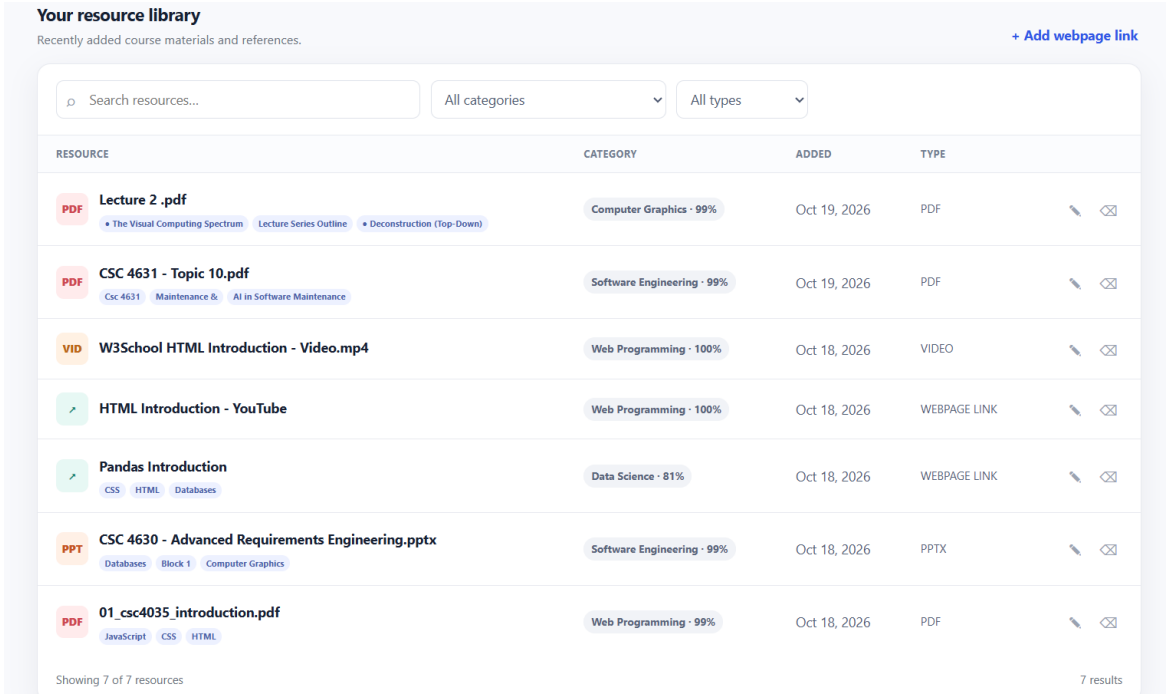


Figure 5.4: Resource Library with Automatic Classification and Topic Labels

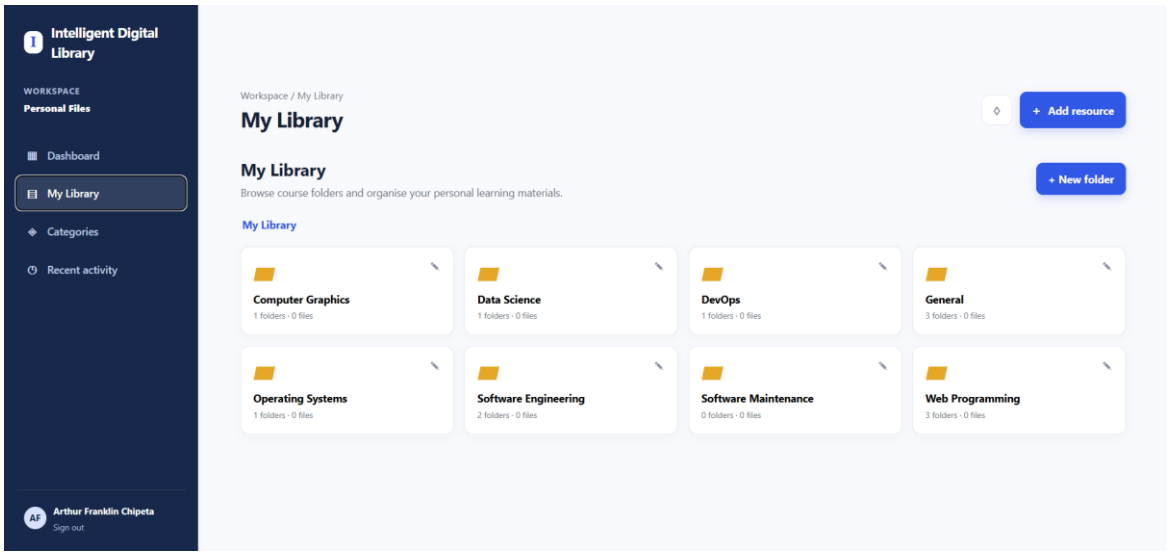


Figure 5.5: Hierarchical Course Folder Management Interface

5.5. Integration

FastAPI hosts the client and REST API on one origin. Pydantic responses and SQLAlchemy relationships connect resources with predictions, topics and assignments. Error-handled upload coordinates storage, extraction, classification, folders and database persistence.

Application startup initialises tables and fills missing automatic metadata without replacing manual categories or moves. Environment settings switch the same codebase between local and hosted database/storage.

Authenticated JSON/multipart requests link the browser and backend. Session restoration loads resources/categories; successful API updates refresh views from persisted state.

5.6. Challenges and Solutions

Table 5.2 Implementation challenges and adopted solutions

Challenge	Risk	Implemented solution
Ambiguous academic vocabulary	Related subjects may receive the same label	Combine filename, course code, headings, phrase frequency, and TF-IDF similarity; return alternatives and a review flag
Heterogeneous resource formats	One parser cannot preserve evidence across all documents	Use separate PDF, DOCX, PPTX, and webpage adapters with a common page-text structure
Conflicting filenames	One user's upload may overwrite another file	Generate a UUID object name under an owner-specific path while retaining the original display name
Automation overriding the user	Correct manual organisation may be repeatedly undone	Record manual prediction and folder sources and skip them during automatic backfill
Local and hosted storage	Deployment cannot depend on the developer's filesystem	Provide one storage interface with local and Supabase implementations selected by configuration

Challenge	Risk	Implemented solution
Older database records	New topic, prediction, or folder features may leave incomplete rows	Run bounded startup backfills using stored objects or existing extracted text
Cross-user access	A guessed numeric identifier may expose a private resource	Require JWT authentication and query every resource and folder by both identifier and owner

5.7. Individual Contribution

My IFMS contribution covers requirements, architecture, frontend/backend, persistence, private storage, security, extraction, classification, topics, folders, testing and deployment documentation, as recorded in repository history and Table 5.1.

Deliverables include the authenticated library, private file/URL lifecycle, local/Supabase storage, text extraction, explainable classification, editable categories/topics, course folders and responsive interface, plus the reported functional and owner tests.

Recommendations, schedules, reminders, progress tracking and related dashboards belong to other group members. IFMS supplies authenticated identifiers, content and metadata for possible integration but claims none of those implementations.

5.8. Chapter Summary

The implemented FastAPI/SQLAlchemy/PostgreSQL system combines private ingestion, text extraction, explainable categories/topics and owner-controlled course folders through a browser and configurable storage. Modular tests document the individual subsystem boundary.

6. Testing, Results and Evaluation

6.1. Introduction

Evaluation covers only this author’s authentication, private resource lifecycle, extraction, classification, topics, folders, user corrections and ownership. Other group study-assistance functions are excluded.

Evidence consists of automated functional/ownership tests, controlled classifier/topic fixtures and local service timings reproduced on 23 October 2026. Taxonomy-derived seeds are not an independent accuracy set, and no participant usability dataset was available.

6.2. Testing Strategy

Risk-based tests prioritised credentials, owner isolation, persistence, validation and manual-choice precedence. API tests used in-memory SQLite and temporary local storage; webpage tests used controlled responses.

The automated suite was executed with Python 3.14.5, pytest 8.4.2, FastAPI's TestClient, SQLAlchemy's StaticPool, and the local-storage configuration. Each integration test recreated the database schema before execution. The run completed 23 tests in 13.42 seconds. Pytest emitted one warning because its optional cache directory could not be written; this did not change collection, execution, assertions, or the final status.

Table 6.1: Test Strategy and Evidence Sources

Evaluation area	Method and evidence	Acceptance basis
Authentication and validation	API tests for registration, duplicate email, password rules, login, current-user retrieval, invalid credentials and tokens.	Expected HTTP status, normalised identity data, and no password-hash disclosure.
Resource lifecycle	Integration tests for protected upload, persistence, list, inline file access, viewer metadata, download, rename and deletion.	Expected response content, metadata persistence, validation result, and deletion state.
Ownership and privacy	Two-account tests attempt to read, download, rename, delete, relabel, browse or move another owner's data.	Every cross-user operation returns 404 without exposing the resource or folder.
Intelligent organisation	Controlled documents exercise category separation, course-code mapping, topic/page extraction, manual correction and Course-to-Material-Type folders.	Expected labels, page references, confidence/review state, and persistence of manual choices.
Webpage capture	Mocked YouTube oEmbed and Open Graph responses isolate title-selection behaviour.	Descriptive title is preferred to a generic browser title.

Evaluation area	Method and evidence	Acceptance basis
Core processing time	One hundred measured calls after ten warm-ups for classification and topic extraction.	Median and 95th-percentile local service time reported with scope limitations.
Scientific classification and usability	Independent labelled corpus and participant study required by the methodology.	Not scored because neither completed evidence set exists in the repository.

6.3. Test Cases and Results

Table 6.2 lists representative passing assertions; it does not establish coverage of untested browsers, loads, file types or datasets.

Table 6.2: Representative Functional and Security Test Cases

ID	Test action	Expected result	Observed result
TC-01	Register with an upper-case email and a valid password.	201; email normalised; password hash absent from response.	Expected identity returned. Pass.
TC-02	Repeat an existing email and submit a weak password.	409 for duplicate; 422 for weak password.	Both validation paths matched. Pass.
TC-03	Login, request /auth/me, then use bad credentials, no token and an invalid token.	Valid token succeeds; invalid authentication returns 401.	All expected states observed. Pass.
TC-04	Upload without authentication and upload an unsupported text file after login.	401 when anonymous; 415 for unsupported media.	Both requests rejected correctly. Pass.
TC-05	Create an image resource, list it, open inline content/viewer, download and delete it.	Metadata and bytes persist; dispositions differ; deleted item returns 404.	Complete lifecycle matched. Pass.
TC-06	Use a second account to read, view, download, rename or delete another user's resource.	All operations concealed with 404.	All five resource-access attempts plus rename were denied. Pass.
TC-07	Capture a webpage, prefer its extracted title, then rename it.	Descriptive title stored and manual name persisted.	Both title and rename assertions matched. Pass.
TC-08	Extract topics from three numbered pages.	OSI Model on page 1; TCP/IP and Routing on page 2; Subnetting on page 3.	All four labels, pages and confidence bounds matched. Pass.
TC-09	Add and remove a manual topic containing duplicate and invalid page values.	Manual source/confidence retained; pages normalised to [2, 4]; deletion clears topic.	Normalisation and lifecycle matched. Pass.
TC-10	Classify Data Mining and Operating Systems fixtures plus three UNZA course-code fixtures.	Five named fixture categories selected; confident cases not flagged.	Data Mining, Operating Systems, Cybersecurity, Database Systems and Computer Networks matched. Pass.

ID	Test action	Expected result	Observed result
TC-11	Correct a category manually, then try an unknown category and a cross-user correction.	Manual choice persists at confidence 1.0; invalid input is 422; other user receives 404.	All category-review rules matched. Pass.
TC-12	Upload Data Mining lecture slides and inspect generated folders.	Course folder contains Lecture Slides folder and resource/topic metadata.	Expected hierarchy and content observed. Pass.
TC-13	Rename a course folder, create Exam Revision, move a resource, and repeat from another account.	Manual move persists; other account cannot browse or move the item.	Owner move succeeded; both cross-user folder operations returned 404. Pass.
TC-14	Extract titles from mocked YouTube and Open Graph pages.	oEmbed/Open Graph titles take precedence over generic HTML title.	Both service tests passed.

6.4. Experimental Results / Performance Results

6.4.1. Automated Suite Result

All 23 collected tests passed: 21 integration cases for authentication/resources and two webpage-title service tests. The slowest, at 0.82 s, exercised several lifecycle operations, not one API response.

Table 6.3: Automated Results by Functional Area

Functional area	Tests	Result	Principal behaviours covered
Authentication and account validation	5	5 passed	Registration, duplicate email, password rule, login/current user, invalid credentials and tokens.
Resource upload, lifecycle and naming	6	6 passed	Protected upload, persistence, serving, download, deletion, file/webpage rename and resource ownership.
Topic handling	3	3 passed	Automatic page references, manual add/remove, and cross-user protection.
Classification and category review	4	4 passed	Subject separation, course-code mapping, manual correction, input and owner validation.
Folder organisation	3	3 passed	Automatic hierarchy, rename, manual movement and owner isolation.
Webpage extraction service	2	2 passed	YouTube oEmbed and Open Graph title preference.
Total	23	23 passed	No assertion failure; one non-functional pytest cache warning.

6.4.2. Classification and Topic Fixture Results

Five category fixtures predicted Data Mining, Operating Systems, Cybersecurity, Database Systems and Computer Networks; topics OSI Model, TCP/IP, Routing and Subnetting were located on expected pages. Named security and Data Mining cases met the review threshold.

These are controlled checks, not scientific accuracy estimates: the seed CSV is generated from the classifier taxonomy. Independent human-labelled, unseen resources are required for accuracy, macro precision/recall/F1, top-three accuracy, review rates and a confusion matrix.

6.4.3. Core Algorithm Timing

Following ten warm-ups, each service processed a short fixture 100 times with a high-resolution timer. Table 6.4 reports only core analysis, excluding transfer, parsing, persistence, storage, rendering and network overhead.

Table 6.4: Local Core-Service Microbenchmark

Operation	Measured calls	Median	95th percentile	Interpretation
Hybrid classification	100	191.0884 ms	253.9642 ms	Sub-second core scoring for the small fixture; not end-to-end upload latency.
Topic extraction	100	10.8519 ms	22.7001 ms	Low local processing time for three short page records; not representative of large documents.

6.4.4. Ownership and Security Results

Ten cross-user metadata/content/viewer/download/delete/rename/topic/category/folder/move operations all returned 404. Tests also confirmed 401 for invalid or missing credentials, 409 for duplicate email, 422 for weak password/unknown category and 415 for unsupported media.

6.5. User Evaluation

No completed usability task logs, consented observations, questionnaires or SUS data were available. Numerical satisfaction scores are therefore not reported.

Service tests show the operations exist, not that students understand them or save time. A future study should recruit representative users, observe upload/retrieval/correction/movement tasks, record errors/completion and administer SUS.

6.6. Analysis of Results

Tests showed consistent owner checks across records, content, topics, categories and folders. A Data Mining lecture was classified and placed in its expected course/material folder with topics intact.

A manual category retained its source, and renamed folders and resources moved to Exam Revision preserved their chosen locations, confirming the intended precedence of user edits.

Three limits remain: controlled fixtures do not estimate general classification accuracy; local short-text timings do not predict deployed performance; and no participant usability data were collected. Thus, broad effectiveness claims remain premature.

6.7. Objective-by-Objective Evaluation

Table 6.5: Evaluation of Project Objectives

Objective	Evidence from implementation and testing	Assessment
1. Analyse the problem and define requirements.	Chapters 1-4 define the academic file-management problem, scope, user needs, functional requirements, privacy and usability constraints.	Achieved for the individual subsystem.
2. Design the architecture and data model.	Layered architecture, owner-scoped relational entities, storage abstraction, classification evidence, topics and folder hierarchy were designed and implemented.	Achieved.
3. Implement secure ingestion, storage, retrieval and deletion.	Lifecycle, validation and authentication tests passed; cross-user attempts were concealed; file bytes and metadata persisted and were deleted as expected.	Achieved for tested formats and paths.
4. Develop explainable classification and topic extraction.	Five named category fixtures and four topic/page assertions passed; confidence, alternatives, review state and manual provenance are persisted.	Functionally achieved; independent accuracy evaluation pending.
5. Implement organisation, retrieval and correction.	Automatic course/material folders, browsing, rename, manual move, category correction and topic editing tests passed.	Achieved for tested workflows.
6. Evaluate functional, classification, usability, performance and security qualities.	23/23 automated tests passed; owner isolation and local core timing were measured. No independent labelled corpus or completed participant study is available.	Partially achieved; scientific classifier and formal usability evaluation remain outstanding.

6.8. Individual Results

The results attributable to the author are the completed and tested Intelligent File Management System functions. No result in this section is assigned to recommendation, scheduling, reminder, progress-tracking, or other group-member functionality.

Authentication and privacy: registration, login, current-user recovery, password validation, token rejection and owner-scoped access were implemented and verified.

Resource lifecycle: supported files and webpages can be represented as private resources, listed, viewed or downloaded where applicable, renamed and deleted, with validation at the service boundary.

Intelligent processing: the system combines course codes, names, headings and text similarity to propose a category with confidence, alternatives and review state; topics retain source and page or slide evidence where available.

Human review: manual categories and topics are stored with explicit provenance, and invalid or unauthorised changes are rejected.

Organisation and retrieval: resources are placed into an automatic Course-to-Material-Type hierarchy, while folder renaming, nested folders, browsing and persistent manual movement preserve user control.

Verification: the current repository contains 23 passing automated tests covering the major API, service, ownership and organisation paths, together with a documented local timing result for the two core analysis services.

6.9. Chapter Summary

The 23 tests passed in 13.42 s, covering authentication, resource lifecycle, webpage titles, five classifier fixtures, topic/page evidence, category review, folders, manual moves and owner isolation. Core median times were 191.0884 ms (classification) and 10.8519 ms (topics) on short fixtures.

The evidence supports tested workflow consistency and owner control, not general accuracy or user-rated usability. Objectives 1–5 were achieved within the tested scope; the evaluation objective remains partial pending independent labels and student studies.

7. Discussion

7.1. Introduction

This chapter interprets IFMS test findings for authenticated libraries, storage, classification, topics, folders and manual correction. Other group members' study-assistance functions are outside scope.

All 23 automated tests passed across resource lifecycle, organisation, corrections and owner isolation. This supports tested behaviour, not accuracy on unseen resources, participant usability or deployed scalability; independent labelled data and user studies are still absent.

7.2. Interpretation of Findings

7.2.1. Organisation of Academic Resources

Tests confirmed authenticated resources could be stored with owner metadata, classified, placed in Course-to-Material-Type folders and kept in user-selected locations after renaming or movement. A Data Mining fixture reached its expected folder with topics intact, addressing the fragmentation described in Chapter 2.

IFMS connects keeping, organising and refinding [1] and responds to reactive student accumulation [4]. Automatic placement reduces initial filing, while familiar folders, search, filters and topic labels offer complementary retrieval paths.

7.2.2. Classification Topic Evidence and Human Review

Five fixtures predicted Data Mining, Operating Systems, Cybersecurity, Database Systems and Computer Networks; a topic fixture located OSI Model, TCP/IP, Routing and Subnetting on expected pages. They confirm controlled course-code, filename, heading, term and TF-IDF evidence paths, not general accuracy.

A manual category persisted with manual provenance and confidence 1.0; folder moves likewise remained authoritative. Alternatives, confidence and review flags expose uncertainty while protecting user correction from later processing. Five structured fixtures cannot establish population-level classifier accuracy.

7.2.3. Privacy and Ownership

Ten cross-user attempts against resource metadata, inline content, viewer data, downloads, deletion, renaming, topics, category changes, folders and moves all returned 404. Consistent owner checks across these paths protect private academic material within the tested scope.

Authentication and validation tests also rejected missing/invalid tokens, duplicate emails, weak passwords, unknown categories and unsupported media. These are not penetration tests; deployment storage policies, expiry, malicious uploads, denial-of-service protection and dependency security require further assessment.

7.2.4. Performance and User Experience

Local median times were 191.0884 ms for hybrid classification and 10.8519 ms for topic extraction on short fixtures. Classification evaluates more category and similarity evidence. Neither figure includes transfer, parsing, persistence, storage, browser rendering or network delays, so neither represents end-to-end latency.

Automated tests verify that upload, viewing, edits, moves and deletion work, not whether students understand or prefer them. Without completed task observations or SUS responses, usability of review flags, alternatives and folder controls remains unmeasured.

7.2.5. Relationship to the Objectives and Research Questions

Objectives 1–5 have implementation support: requirements, secure lifecycle, controlled classification/topics, folders and correction. RQ1 rests mainly on literature; RQ2–4 have tested design answers; RQ5 remains partial pending independent classification and participant usability evaluation.

7.3. Comparison with Existing Work

7.3.1. General Cloud Storage and Reference Managers

Google Drive and OneDrive offer greater scale, collaboration and integration, with optional Gemini organisation for eligible Drive content [13]. IFMS differs in its course taxonomy, saved prediction evidence, page/slide topics and course/type folders, but no comparative usability or accuracy advantage was measured.

Zotero offers scholarly collections, tags, search, citation metadata and attachment indexing [14], [15]. IFMS instead manages mixed academic files and webpages even without bibliographic records; the tools address complementary purposes.

7.3.2. Content Aware Organisation

Like CFCOS [11], IFMS uses content for organisation. CFCOS generates semantic summaries and tags with LLMs; IFMS relies on transparent local rules, TF-IDF, a bounded taxonomy and explicit correction.

Local weighted evidence is inspectable and needs no external generative request, supporting constrained deployments and privacy. Its coverage is narrower for unfamiliar vocabulary or sparse text; the controlled fixtures validate configured paths, not broad semantic equivalence.

7.3.3. Position of the Implemented Subsystem

The project occupies a specific position between ordinary folders, reference managers, and general semantic organisers. It retains folders because location remains useful, adds topics because one resource may contain several concepts, and uses course-aware classification to reduce repetitive filing. The integration of these mechanisms is the relevant comparison point. Chapter 6 shows that the tested parts exchange and preserve the required metadata across the workflow. The remaining question is whether this combination improves real student retrieval and organisation over time, which requires comparative user tasks and longitudinal use rather than more implementation fixtures alone.

7.4. Contributions of the Project

7.4.1. Practical Contribution

The practical contribution is a working browser-based personal library for heterogeneous academic resources. A student can store supported files and webpages, obtain an initial course and material organisation, inspect topics and classification state, retrieve or download permitted content, and correct the stored organisation. This contribution addresses the gap identified in Chapter 2 at a prototype scale: the system is centred on mixed university course material rather than on generic storage or bibliographic records.

7.4.2. Technical Contribution

The technical contribution is the integration of several modest, inspectable mechanisms into one owner-scoped pipeline. Format-specific extraction produces text and page or slide records; the hybrid classifier combines direct academic evidence with local similarity; topic extraction preserves locations; relational records store confidence, alternatives, source, and review state; and folder rules create stable course and material-type locations. Manual category and folder choices are protected from automatic replacement. The tests provide evidence that these components maintain consistent state across the main resource lifecycle.

7.4.3. Evaluation Contribution

The project also contributes a reproducible baseline for continued evaluation. The current suite defines expected behaviour for authentication, validation, private resource access, topic editing, category correction, and folder organisation. The discussion separates those verified behaviours from unmeasured claims. It also identifies the next evidence needed: an independent human-labelled collection for category metrics, representative large-file and deployed performance tests, and task-based student evaluation. This distinction prevents fixture success from being presented as general model effectiveness.

7.5. Limitations and Implications

7.5.1. Limits of the Evaluation Evidence

The automated suite covers important success and rejection paths, but it remains a developer-controlled test environment using an in-memory database, temporary local storage, and small fixtures. The pass rate applies only to those assertions. It does not measure concurrent users, long-running sessions, recovery from partial storage failure, large document collections, browser compatibility, accessibility, or deployment reliability. The local timing figures likewise describe core functions on one computer and should not be used as service-level performance targets.

The lack of an independent labelled corpus has a direct implication for the classifier. Taxonomy-derived seed rows cannot provide a fair test because they repeat the vocabulary used to construct the category profiles. Until unseen resources are labelled by people independently of the algorithm, the report cannot establish accuracy, macro precision, recall, F1, top-three accuracy, review-flag rate, or a meaningful confusion matrix. The absence of participant records similarly means that task completion, error rate, satisfaction, and SUS cannot be discussed as findings.

7.5.2. Limits of the Implemented Scope

The taxonomy is limited to undergraduate computer-science subjects and known course-code evidence. Material from another discipline or a specialised topic may be placed in General or flagged for review. Classification and topics also depend on extractable text. Scanned pages, photographs of notes, image-only documents, and video speech are not interpreted because optical character recognition and transcription are outside the implementation. Word documents do not provide reliable rendered page boundaries through the selected parser, so their topic locations are less precise than those from PDF pages or presentation slides.

Webpage capture depends on the content returned by the remote site and may be incomplete when a page requires authentication, relies on client-side rendering, or blocks automated requests. Large media files are also constrained by the configured upload limit, which defaults to 20 MB. These limits imply that the system is best understood as a focused academic resource organiser for supported, text-bearing material rather than a universal content-understanding service.

7.5.3. Implications for Use and Further Evaluation

The current evidence supports cautious pilot use in which predictions remain suggestions and review controls stay visible. A deployment should preserve owner checks at every endpoint, use private object storage, monitor failed extraction and low-confidence categories, and avoid

treating confidence as a calibrated probability. Manual corrections should remain authoritative. With appropriate consent and privacy controls, accumulated corrections could later help construct an independent evaluation set or inform improved category profiles.

Further evaluation should use resources that were not involved in taxonomy design and should report results by category as well as overall. Student participants should complete realistic tasks such as adding a mixed set of resources, locating a document by topic, correcting a category, and moving a file into a personal folder. End-to-end timings should separate upload, extraction, classification, persistence, and retrieval. These studies would show whether the technically coherent workflow also improves organisation and refinding in practice.

7.6. Chapter Summary

The findings show that the Intelligent File Management System implements the intended owner-controlled workflow for the tested cases. Resource lifecycle, category and topic handling, automatic folders, manual correction, and cross-user protection all behaved as specified. The project therefore addresses the identified gap at prototype level by combining academic classification, page- or slide-linked topics, familiar folders, and private user ownership in one web application.

The discussion also limits that conclusion. The classifier fixtures demonstrate implemented logic but not general accuracy, the local timings do not represent end-to-end deployment performance, and no participant study establishes usability or academic benefit. The project contributes a functional and inspectable foundation together with a clear evaluation baseline. Independent labelled data, student usability testing, broader content extraction, and deployed performance and security assessment are required before stronger conclusions can be made.

8. Conclusion and Recommendations

8.1. Introduction

This chapter summarises the individual IFMS implementation, assessed objectives, evidence-based conclusions, recommendations and future work; other group functions are excluded.

8.2. Summary of the Project

IFMS addresses scattered academic files and links that are difficult to find by course, type or topic. It builds a private web library with explainable classification, topic extraction and editable hierarchical folders, reducing dependence on manually chosen filenames and locations.

The HTML/CSS/JavaScript and FastAPI application stores owner-scoped files/webpages via SQLAlchemy and private local/cloud storage. Supported documents yield course categories, confidence, alternatives and page/slide topics; users browse course/type folders, edit metadata, move resources and delete them. Manual choices override automatic reprocessing.

All 23 automated tests passed, covering resource lifecycle and edits; ten cross-user operations returned 404. Five classifier and four topic/page assertions passed. Short-fixture medians were 191.0884 ms for classification and 10.8519 ms for topic extraction.

Independent labelled accuracy data and participant usability results remain unavailable. Microbenchmarks exclude transfer, parsing, database, storage, browser and network overhead; findings therefore establish tested functionality, not broad accuracy, usability, scaling or production security.

8.3. Conclusions

8.3.1. Achievement of the Project Aim

The tested prototype achieved the project aim: an integrated private academic library combining ingestion, content-derived categories and topics, automatic folders and lasting manual control.

8.3.2. Value of Explainable and Correctable Automation

Rule-based course/name/heading/term and TF-IDF evidence yields inspectable suggestions; confidence, alternatives and review flags signal uncertainty. Manual categories and moves rightly override uncertain automation.

8.3.3. Privacy and Consistency

Tests consistently enforced owner-only access across view, download, edit, topic, category, folder and delete paths. This is functional security evidence, not a production penetration assessment.

8.3.4. Limits of the Final Conclusion

The prototype shows technical feasibility, but not general accuracy, student efficiency or scalability. Its computer-science taxonomy and extractable-text limits require independent datasets, participant tasks and deployed evaluation.

8.4. Achievement of Objectives

Table 8.1 records the final assessment of each objective using the implementation and evaluation evidence presented in the report.

Table 8.1: Final Assessment of Project Objectives

Objective	Evidence and conclusion	Status
1. Analyse the academic resource-management problem and define requirements.	The report identified fragmented student collections, retrieval difficulties, privacy needs, and requirements for automatic assistance with retained user control.	Achieved for the individual subsystem.
2. Design the architecture and relational data model.	The implemented layered architecture, owner-scoped entities, storage abstraction, classification evidence, topics, and hierarchical folders follow the documented design.	Achieved.
3. Implement secure ingestion, storage, retrieval, and deletion.	Authentication, validation, resource lifecycle, deletion, and cross-user access tests passed for the covered paths and formats.	Achieved for the tested scope.
4. Develop explainable classification and topic extraction.	Five category fixtures and four topic/page assertions passed; confidence, alternatives, review state, and manual provenance are retained.	Functionally achieved; general accuracy not yet established.
5. Implement organisation, retrieval, and correction.	Automatic course/material folders, browsing, renaming, manual category correction, topic editing, nested folders, and persistent movement were implemented and tested.	Achieved for the tested workflows.
6. Evaluate functional, classification, usability, performance, and security qualities.	Twenty-three automated tests passed and local core timing was measured. Independent classifier metrics, formal usability results, deployed performance, and full security assessment remain outstanding.	Partially achieved.

8.5. Recommendations

The following recommendations concern use and evaluation of the current subsystem rather than unimplemented group-project features.

1. **Use a controlled pilot:** Introduce the system to a small group of computer-science students before wider adoption. Pilot use should focus on supported text-bearing resources and should record extraction failures, review cases, and manual corrections.

2. **Keep human review visible:** Present categories as suggestions, show confidence and alternatives, and keep low-confidence or closely scored results in a review state. Manual categories, topics, names, and folder movements should remain authoritative.

3. **Preserve private ownership controls:** Use private object storage, validate tokens and upload limits, scope every query and operation to the authenticated owner, and retain the non-disclosing 404 response for unauthorised object access.

4. **Monitor processing and storage:** Record controlled operational logs for extraction errors, unsupported content, classification review rates, storage failures, and deletion cleanup without exposing document content or sensitive user data.

5. **Complete independent evaluation before strong claims:** Do not report classifier accuracy or usability scores from taxonomy-generated examples or informal impressions. Use unseen human-labelled resources and consented participant tasks before claiming effectiveness beyond the tested implementation.

8.6. Future Work

1. **Independent classifier evaluation:** Build a human-labelled dataset of previously unseen academic resources and report per-category precision, recall and F1, macro averages, top-three accuracy, review-flag rate, and a confusion matrix. The dataset should be separate from the terms used to construct category profiles.

2. **Formal usability study:** Recruit representative students to complete upload, retrieval, category correction, topic review, and folder-movement tasks. Measure completion, errors and time, collect qualitative observations, and administer the System Usability Scale after the tasks.

3. **Broader content extraction:** Add optical character recognition for scanned pages and images and speech-to-text processing for audio or video where consent, copyright, cost, and storage policies permit. Preserve page or time references so extracted topics remain traceable.

4. **Configurable and broader classification:** Allow an institution or user to manage course codes, category profiles, and material types without changing application code. Compare the current transparent hybrid method with local embeddings or another semantic method while retaining evidence, review thresholds, privacy controls, and manual override.

5. **Production readiness:** Evaluate end-to-end upload and retrieval time across file sizes, concurrent users, database and object-storage configurations, and failure

conditions. Add dependency scanning, malicious-file handling, rate limiting, backup and recovery tests, accessibility checks, and a formal security review before production deployment.

Introduce extensions incrementally while retaining privacy, explicit uncertainty and authoritative user corrections.

9. References

[1] W. Jones, "Personal information management," *Annual Review of Information Science and Technology*, vol. 41, no. 1, pp. 453–504, 2007. doi: 10.1002/aris.2007.1440410117.

[2] J. D. Dinneen and C.-A. Julien, "The ubiquitous digital file: A review of file management research," *Journal of the Association for Information Science and Technology*, vol. 71, no. 1, pp. E1–E32, 2020. doi: 10.1002/asi.24222.

[3] O. Bergman, R. Beyth-Marom, and R. Nachmias, "The project fragmentation problem in personal information management," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2006, pp. 271–274. doi: 10.1145/1124772.1124813.

[4] E. Sillence, J. A. Dawson, K. McKellar, and N. Neave, "How do students use digital technology to manage their university-based data: Strategies, accumulation difficulties and feelings of overload?" *Behaviour & Information Technology*, vol. 42, no. 14, pp. 2442–2451, 2023. doi: 10.1080/0144929X.2022.2126948.

[5] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1–47, 2002. doi: 10.1145/505282.505283.

[6] R. Boardman and M. A. Sasse, "Stuff goes into the computer and doesn't come out: A cross-tool study of personal information management," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2004, pp. 583–590. doi: 10.1145/985692.985766.

[7] S. Robinson and F. Johnson, "The process and affective environment of students' personal information management," *Enhancing Learning in the Social Sciences*, vol. 4, no. 2, pp. 1–13, 2012. doi: 10.11120/elss.2012.04020005.

[8] K. Ameen, "Personal information management practices and behaviors of social sciences' students," *Pakistan Journal of Information Management and Libraries*, vol. 18, no. 1, pp. 12–24, 2016. doi: 10.47657/2016181952.

[9] D. Barreau and B. A. Nardi, "Finding and reminding: File organization from the desktop," *ACM SIGCHI Bulletin*, vol. 27, no. 3, pp. 39–43, 1995. doi: 10.1145/221296.221307.

[10] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988. doi: 10.1016/0306-4573(88)90021-0.

[11] W. Son and H. Kim, "Content-based file classification and organization system using LLMs," *Electronics*, vol. 15, no. 7, Art. no. 1524, 2026. doi: 10.3390/electronics15071524.

[12] Microsoft, "Organize documents and sites in Microsoft 365," *Microsoft Support*, 2026. [Online]. Available: https://support.microsoft.com/en-us/onedrive/sharepoint/organize-documents-and-sites-in-microsoft-365. [Accessed: Oct. 7, 2026].

[13] Google, "Let Gemini organize your files," *Google Drive Help*, 2026. [Online]. Available: https://support.google.com/drive/answer/16671865. [Accessed: Oct. 7, 2026].

[14] Zotero, "Collections and tags," 2026. [Online]. Available: https://www.zotero.org/support/collections_and_tags. [Accessed: Oct. 7, 2026].

[15] Zotero, "Searching," 2026. [Online]. Available: https://www.zotero.org/support/searching. [Accessed: Oct. 7, 2026].

[16] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004. doi: 10.2307/25148625.

[17] K. Schwaber and J. Sutherland, *The Scrum Guide: The definitive guide to Scrum*, 2020. [Online]. Available: https://scrumguides.org/scrum-guide.html.

[18] International Organization for Standardization, *ISO/IEC 25010:2023: Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Product quality model*, 2023. [Online]. Available: https://www.iso.org/standard/78176.html.

APPENDIX A: INDIVIDUAL CONTRIBUTION STATEMENT FOR GROUP PROJECTS

Purpose of this Statement

Because this was a group project, this statement distinguishes my individual contribution from shared project work and other members' responsibilities.

My contribution is the Intelligent File Management System: its requirements, design, implementation, testing, evaluation, documentation, and deployment preparation, as evidenced in this report.

I do not claim group members' recommendation, scheduling, reminder, or progress-tracking features. My subsystem may supply authenticated users, resource identifiers, content, and metadata for their separate integrations.

A.1 Project and Group Information

Table A.1: Project and Group Information

Item	Details
Overall Project Title	Intelligent Digital Library with Personalized Study Assistance
Individual Subsystem	Intelligent File Management System
Student Name	Arthur Franklin Chipeta
Computer Number	2022068424
Supervisor	Dr Lighton Phiri
Group Size	3
Group Members	Arthur F. Chipeta (2022068424); Janet M. Tambwisha (2022057805); Lunda Simuwelu (2021399125)

A.2 Overall Group Project

The group project addresses academic resources and study activities fragmented across downloads, learning platforms, email, messaging applications, and websites through a shared web-based support platform.

My subsystem provides authenticated ingestion of files and webpages, private storage, text extraction, explainable course categories and topics, Course-to-Material-Type folders, and preserved user corrections. Other group members develop recommendations, scheduling, reminders, progress tracking, and study support; their implementations and results are outside my scope.

A.3 Individual Role and Responsibilities

My primary role was to analyse, design, implement, and evaluate the Intelligent File Management System as an independently identifiable subsystem within the group project.

Table A.2: Individual Role and Responsibilities

No.	Area of Responsibility	Description
1	Problem and requirements analysis	Defined the subsystem problem, scope, functional requirements, privacy needs, owner controls, and evaluation criteria.
2	Architecture and data design	Designed the layered web architecture and owner-scoped relational entities for users, resources, predictions, topics, folders, and assignments.
3	Backend, security, and storage	Implemented authentication, validation, resource lifecycle endpoints, private storage abstraction, and owner-filtered access.
4	Content processing	Implemented text extraction for PDF, DOCX, PPTX, and webpage content using a common page-text representation.
5	Intelligent organisation	Developed explainable hybrid course classification, topic extraction, review state, alternatives, and hierarchical folder rules.
6	Browser interface	Implemented registration, login, library, upload, filtering, viewer, resource editing, category review, topic editing, and folder navigation.
7	Testing and evaluation	Developed and executed functional, ownership, classifier, topic, folder, and local performance checks; analysed their limits.
8	Documentation and deployment preparation	Documented the subsystem and prepared environment-based configuration for local and hosted operation.

A.4 Detailed Individual Contribution

A.4.1 Requirements Analysis and Background Research

I studied student resource-management practices, folders and tags, classification, explainability, privacy, and related work. This informed functional, security, privacy, usability, and evaluation requirements and distinguished my scope from study-assistance features.

A.4.2 System and Database Design

I designed the browser, FastAPI service, persistence, storage, and processing layers; the resource lifecycle from upload through extraction, classification, topics, folders, retrieval, correction, and deletion; and owner-scoped relationships for users, resources, predictions, topics, and folders. I specified use cases, interfaces, review, deletion consistency, and preservation of manual choices.

A.4.3 Backend, Persistence, and Security

I implemented registration, login, hashed passwords, JWT issue/validation, owner-scoped resource and folder APIs, SQLAlchemy persistence, and input validation. A common private-storage interface supports local files and a Supabase-compatible bucket, with generated owner-specific paths and cleanup after failed database transactions.

A.4.4 Extraction, Classification, and Topics

I built PDF, PowerPoint, Word, and webpage extraction, and a hybrid classifier using course codes, filenames, headings, controlled terms, and TF-IDF cosine similarity. It records confidence, alternatives, review status, and provenance. Topics include confidence and page/slide evidence where available, with manual editing.

A.4.5 Folder Organisation and User Control

I implemented automatic Course-to-Material-Type folders, nested user folders, breadcrumbs, browsing, renaming, moves, and assignment provenance. Reprocessing can alter automatic placement but never override manual category decisions or folder moves.

A.4.6 Browser Interface and Integration

I implemented the HTML, CSS, and JavaScript interface for accounts, sessions, uploads, library browsing, filters, search, resource details, PDF viewing, edits, deletion, category review, topics, and folders. Authenticated JSON and multipart requests connect it to FastAPI and refresh saved state after changes.

A.4.7 Testing, Documentation, and Deployment Preparation

I created pytest and FastAPI TestClient tests with isolated SQLite, temporary storage, and controlled webpage responses. I documented the project and prepared environment-based database, JWT, upload, and storage configuration for local and hosted deployment without exposing secrets.

A.5 Individual Deliverables

Table A.3: Individual Deliverables

No.	Deliverable	My Contribution
1	Authenticated personal library	Implemented account registration, login, session validation, sign-out behaviour, and owner-scoped access.
2	File and webpage resource lifecycle	Implemented submission, validation, metadata persistence, listing, retrieval, download, renaming, and deletion.
3	Private storage layer	Implemented local and Supabase-compatible object storage with owner-specific paths and cleanup on failure.
4	Text-extraction services	Implemented PDF, DOCX, PPTX, and webpage adapters with page or slide locations where available.
5	Explainable course classifier	Implemented weighted evidence, TF-IDF similarity, confidence, alternatives, review state, and manual correction.
6	Topic service	Implemented controlled and heading-derived topics, confidence, location evidence, addition, and removal.
7	Hierarchical folder system	Implemented automatic course/material folders, nested folders, browsing, breadcrumbs, renaming, and persistent manual movement.
8	Browser user interface	Implemented the authentication, dashboard, library, upload, viewer, review, topic, and folder interactions.
9	Test and evaluation baseline	Produced the automated suite, ownership cases, controlled classifier/topic fixtures, and local core microbenchmarks.

No.	Deliverable	My Contribution
10	Technical report and deployment configuration	Documented the individual subsystem and prepared environment-driven deployment settings.

A.6 Contribution to the Main System or Research Output

My IFMS subsystem creates authenticated, owner-scoped resources, stores file/webpage content and metadata, derives course categories and topics, and exposes organised material through its API and browser. Its validated data can support other group subsystems.

Shared work covered project direction and interface expectations, not implementation of other members' recommendations, schedules, reminders, progress tracking, or study support. Their consumption of IFMS records does not make those features my work.

A.7 Individual Implementation and Technical Work

The following table identifies the principal technical components implemented for my subsystem and their repository locations.

Table A.4: Individual Implementation and Technical Work

Component	Implementation Location	Technical Responsibility
Authentication	backend/app/api/auth.py; services/security.py; models/auth.py; models/user.py	Registration, login, hashing, tokens, current-user resolution, and credential validation.
Resource API	backend/app/api/upload.py	File/webpage submission, list, retrieve, serve, rename, delete, recategorise, and topic operations.
Persistence	backend/app/database.py; backend/app/models	Database configuration and entities for users, resources, predictions, topics, folders, and assignments.
Extraction	backend/app/services/pdf_service.py; docx_service.py; pptx_service.py; webpage_service.py	Format-specific text extraction and a common page-text structure.
Classification	backend/app/services/classification_service.py; backend/data taxonomy resources	Course-code, phrase, heading, controlled-term, and TF-IDF scoring with confidence and alternatives.
Topics	backend/app/services/topic_service.py	Controlled aliases, plausible headings, confidence, and page/slide evidence.
Folders	backend/app/api/folders.py; services/folder_service.py; models/folder.py	Automatic and manual hierarchy, assignment, browsing, breadcrumbs, renaming, and ownership.
Storage	backend/app/services/storage_service.py	Local and Supabase-compatible upload, download, delete, path validation, and cleanup.
Application integration	backend/app/main.py	API/static-client composition, application lifespan, database setup, and conservative backfills.
Browser interface	frontend/index.html; script.js; style.css; viewer.html; viewer.js	Authentication views, dashboard, library, submission, filters, viewer, editing, and folders.

Component	Implementation Location	Technical Responsibility
Automated tests	backend/tests/test_auth.py; test_webpage_service.py	Functional, rejection, ownership, classification, topic, folder, and webpage service checks.

A.8 Individual Testing and Evaluation Contribution

I designed, implemented, executed, and interpreted the automated tests for the subsystem. The final repository run on 23 October 2026 passed all 23 automated tests. The suite covers authentication, invalid input, resource lifecycle operations, owner isolation, webpage processing, classification evidence, topic editing, automatic and manual folder behaviour, and preservation of user corrections.

Ten attempted cross-user operations returned 404, confirming the non-disclosing owner boundary for the covered content, metadata, category, topic, and folder paths. Five named category fixtures passed for Data Mining, Operating Systems, Cybersecurity, Database Systems, and Computer Networks. Four topic-and-location assertions passed for OSI Model on page 1, TCP/IP on page 2, Routing on page 2, and Subnetting on page 3.

Table A.5: Individual Testing and Evaluation Contribution

Measured Core Operation	Median	95th Percentile	Measurement Scope
Classification	191.0884 ms	253.9642 ms	Ten warm-ups and 100 measured calls on a short controlled fixture.
Topic extraction	10.8519 ms	22.7001 ms	Ten warm-ups and 100 measured calls on a short controlled fixture.

The seed fixtures are not an independent labelled corpus, and no completed user study is available. Microbenchmarks exclude network, parsing, persistence, object storage, rendering, and concurrency. Thus, results support tested workflows and owner isolation, not general classifier accuracy, usability, scalability, penetration resistance, or production security.

A.9 Collaboration with Other Group Members

Group collaboration established the project direction, subsystem boundaries, and interface compatibility. Integration uses authenticated user context, resource identifiers, content, categories, topics, and folder metadata.

Other members developed or planned recommendations, scheduling, reminders, progress tracking, and study support. Their code, tests, results, and design decisions are not claimed here; the intelligent library and its evaluation are my contribution.

A.10 Contribution Summary

1. My work: problem analysis and research; architecture and data design; authentication, ownership, resource lifecycle and private storage; extraction, classification, topics and

folders; browser interactions; automated tests and timing; evaluation, documentation, and deployment preparation.

2. Collaborative work: overall project direction, subsystem boundaries, shared interface requirements, and plans for consuming authenticated resource data.

3. Other members' work: recommendations, study scheduling, reminders, progress tracking, and study-assistance dashboards outside IFMS.

A.11 Estimated Contribution

The percentages below should be completed after comparison with the agreed division of group work. They indicate the individual and collaborative share of each activity and do not need to total 100 percent across rows.

Table A.6: Estimated Contribution

Project Activity	Individual Contribution	Collaborative Contribution
Problem Analysis	70 %	30 %
Literature Review	60 %	40 %
Requirements Analysis	70 %	30 %
System / Research Design	75 %	25 %
Implementation / Development	85 %	15 %
Testing and Evaluation	80 %	20 %
Documentation	75 %	25 %
Other: Deployment and Configuration	90 %	10 %

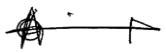
A.12 Statement of Authorship and Contribution

I confirm that the contribution described in this statement accurately represents the work that I personally undertook as part of the group project. I have identified collaborative work and have not knowingly claimed the work of another group member as my own.

Student Name: Arthur Franklin Chipeta

Computer Number: 2022068424

Signature:



Date: October 23, 2026

A.13 Supervisor Verification

I confirm that, to the best of my knowledge, the contribution described above is consistent with the student's participation in the group project.

Supervisor Name: Dr Lighton Phiri

Signature: _____

Date: October 23, 2026

APPENDIX B: USER MANUAL / INSTALLATION GUIDE

This appendix explains how to install, configure, start, and use the Intelligent File Management System. It is limited to the implemented personal resource library and does not cover personalised recommendations, study schedules, reminders, progress tracking, or other group-member features.

B.1 Purpose and Scope

The guide is intended for a developer, assessor, system administrator, or end user who needs to operate the subsystem from the repository. Installation instructions describe a local PostgreSQL and local-file-storage setup first, followed by the optional Supabase storage configuration used for hosted deployment. The user manual follows the labels and behaviour of the current browser interface.

Scope note: The system automatically suggests categories, topics, and folders, but the user remains responsible for reviewing important classifications and retaining an independent backup of original academic material.

B.2 System Overview

The application is a single FastAPI service that exposes authenticated APIs and serves the browser interface from the same address. PostgreSQL stores users, resource metadata, classifications, topics, folders, and assignments. File bytes are stored privately either on the local machine or in a configured Supabase Storage bucket.

Table B.1: System Overview

Area	Implemented Function
Account access	Registration, sign-in, session restoration, and sign-out.
Resource submission	Authenticated upload of supported files and capture of accessible webpage text.
Intelligent processing	Text extraction, explainable course classification, topic labels, confidence, alternatives, and review state.
Organisation	Automatic Course-to-Material-Type folders, user-created nested folders, browsing, breadcrumbs, and folder renaming.
Retrieval and control	Search, filtering, viewing, download, resource renaming, category correction, topic editing, and deletion.
Ownership boundary	Resources, topics, folders, and stored objects are resolved within the authenticated owner's account.

B.2.1 Supported Resource Types

Table B.2: Supported Resource Types

Submission	Extensions / Input	Processing Behaviour
Documents	PDF, DOCX, PPTX	Stored privately; text is extracted for classification and topics. PDF pages and PowerPoint slides retain location references.
Images	JPG, JPEG, PNG	Stored and organised as images. The current version does not perform optical character recognition.
Video	MP4	Stored and organised as video. The current version does not perform speech-to-text transcription.
Webpage	HTTP or HTTPS URL	Visible server-accessible text is fetched, stored as metadata, classified, and linked to the original page.

Upload limit: The default maximum file size is 20 MiB (20,971,520 bytes). An administrator may change MAX_UPLOAD_BYTES. A webpage may fail when it requires authentication, blocks automated requests, or renders essential content only after client-side scripts run.

B.3 Installation Requirements

Table B.3: Installation Requirements

Requirement	Recommended / Required Setting
Operating system	Windows 10/11, Linux, or macOS with command-line access.
Python	Python 3.14, matching the repository .python-version file; include pip and venv support.
Database	A running PostgreSQL instance and permission to create or use one database.
Browser	A current version of Edge, Chrome, Firefox, or Safari with local storage enabled.
Network	Required when capturing webpages, installing packages, or using Supabase storage.
Project source	A local copy of the intelligent-file-management-system repository.

B.4 Local Installation and Configuration

B.4.1 Prepare PostgreSQL

1. Start PostgreSQL. Start the PostgreSQL service or open the server through pgAdmin.

2. Create a database. Create the database named in POSTGRES_DB. The example configuration uses intelligent_file_manager.

```
CREATE DATABASE intelligent_file_manager;
```

3. Record the connection details. Confirm the host, port, database name, user, and password. The standard PostgreSQL port is 5432, while the repository example uses 5433; use the port configured on the target machine.

B.4.2 Create the Python Environment

From PowerShell in the project root, run:

```
python -m venv .venv
.\.venv\Scripts\Activate.ps1
python -m pip install --upgrade pip
python -m pip install -r backend\requirements.txt
```

On Linux or macOS, use:

```
python3.14 -m venv .venv
source .venv/bin/activate
python -m pip install --upgrade pip
python -m pip install -r backend/requirements.txt
```

B.4.3 Create and Edit the Environment File

Create a working environment file from the supplied example. Do not replace an existing .env file unless its settings have been backed up.

```
Copy-Item .env.example .env # PowerShell
cp .env.example .env # Linux or macOS
```

Table B.4: Create and Edit the Environment File

Variable	Purpose and Local Setting
POSTGRES_HOST	Database host, normally localhost for local installation.
POSTGRES_PORT	Port used by the installed PostgreSQL server.
POSTGRES_DB	Database created for the application.
POSTGRES_USER	PostgreSQL login with access to the database.
POSTGRES_PASSWORD	Password for the PostgreSQL login.
DATABASE_URL	Optional complete hosted connection string. When present, it replaces the five POSTGRES settings.
JWT_SECRET_KEY	Long, unique, unpredictable secret used to sign access tokens. Replace the example value.
ACCESS_TOKEN_EXPIRE_MINUTES	Token lifetime in minutes; the example and default are 30.
MAX_UPLOAD_BYTES	Maximum file size in bytes; the example and default are 20971520.
STORAGE_BACKEND	Use local for local installation or supabase for Supabase Storage.
UPLOAD_ROOT	Optional absolute local directory for private uploads; otherwise the project uploads folder is used.

Secret handling: Never commit the populated .env file, JWT secret, database password, or Supabase service-role key. The service-role key is server-side only and must not be placed in frontend JavaScript.

B.5 Starting and Verifying the System

B.5.1 Start the Application

With the virtual environment active, run the following commands from the project root:

```
Set-Location backend
python -m uvicorn app.main:app --reload
```

On Linux or macOS, change the first command to cd backend. At startup, the application creates any missing database tables, reprocesses eligible automatic classifications and topics, backfills folder assignments, and then serves both the API and frontend.

Table B.5: Start the Application

Address	Expected Result
http://127.0.0.1:8000	Registration or sign-in screen for the Intelligent Digital Library.
http://127.0.0.1:8000/health	JSON response showing status as ok.
http://127.0.0.1:8000/docs	Interactive FastAPI API documentation for technical verification.

B.5.2 Run the Automated Tests

Stop the development server or open a second terminal in the project root, then run:

```
$env:PYTHONPATH = "backend"
.\.venv\Scripts\python.exe -m pytest backend/tests
```

The equivalent Linux or macOS command is:

```
PYTHONPATH=backend .venv/bin/python -m pytest backend/tests
```

Verification: A successful installation should start without a database connection error, return the health response, display the sign-in page, and complete the automated test suite. Test execution uses isolated test configuration and does not verify a hosted deployment.

B.6 User Manual

B.6.1 Create an Account and Sign In

1. Open the application. Navigate to the application address supplied by the administrator. For a local installation, use http://127.0.0.1:8000.

2. Choose Create account. Enter a full name, email address, and password. The password must contain at least eight characters, including uppercase, lowercase, and numeric characters.

3. Submit the form. After successful registration, the application signs in with the same credentials and opens the dashboard.

4. Return later. Choose Sign in, enter the registered email address and password, and submit. An expired or invalid session returns the user to the sign-in screen.

B.6.2 Use the Dashboard and Navigation

The sidebar provides Dashboard, My Library, Categories, and Recent activity. The dashboard summarises total resources, documents, web resources, recently added items, subject categories, and the latest additions. A notification indicator lists resources whose automatic category requires review.

B.6.3 Upload a File

1. Open the uploader. Select Add resource.

2. Choose a file. Drag a supported file into the drop zone or browse to select it. Confirm the displayed name, type, and size.

3. Start processing. Select Analyse resource and keep the window open while upload and analysis progress are shown.

4. Review the result. After completion, the resource appears in the library with its type, proposed category, topics where available, and automatic folder placement.

If upload fails: Check the extension, file size, file contents, session status, database connection, and storage configuration. A file with a .pdf extension must begin with a valid PDF signature.

B.6.4 Add a Webpage

1. Open the link form. Select Add webpage link from the resource-library area.

2. Enter the address. Provide a complete HTTP or HTTPS URL and select Analyse webpage.

3. Confirm the saved resource. The system stores the resolved title, original address, extracted text, category, topics, and folder assignment. Opening the resource provides a link back to the original webpage.

B.6.5 Find and Open Resources

• Use Search resources to match a resource name, category, type, or topic label.

• Use All categories or All types to narrow the displayed library. Filters can be combined with search text.

• Select a resource row to open its details. PDF, image, and video content can be previewed when the browser supports the format; extracted PDF pages, Word text, and PowerPoint slide text are displayed in the resource viewer.

• Select Download for a stored file. Webpage resources instead provide a link to the original page.

B.6.6 Review or Correct a Category

1. Open category review. Select the category shown beside the resource or choose a review notification.

2. Inspect the evidence. Review whether the result is automatic or manually corrected, the confidence value, the review flag, and the strongest alternative categories.

3. Choose an action. Select Reclassify to run the current classifier again, or choose the correct primary category and save it.

4. Confirm organisation. A saved category correction is recorded as manual, clears the review flag, and updates automatic organisation. Automatic reclassification may replace

an automatic result but is not intended to silently override a manual correction during startup backfills.

B.6.7 Review and Edit Topics

1. Open the resource viewer. Select the required resource from the library.

2. Use Related topics. Select a topic to jump to its first matching page or slide when location evidence is available.

3. Add a label. Enter a concise topic name in the topic form and submit it. Manual topics are saved with full confidence.

4. Remove a label. Use the remove control beside an incorrect or unnecessary topic. Removal affects only the current owner's resource.

B.6.8 Browse and Maintain Folders

Select My Library to browse the automatic Course-to-Material-Type hierarchy. Typical material folders include Lecture Slides, Course Materials, Web Resources, Images, and Videos. Breadcrumbs show the current path and allow navigation to a parent folder.

1. Open a folder. Select its folder card. The view displays child folders and resources assigned to the current folder.

2. Create a folder. Select New folder, enter a unique name for the current location, and select Save folder.

3. Rename a folder. Use the rename control on the folder card, enter the revised name, and save. Renaming an automatic course folder does not cause the original name to be recreated on the next upload.

Current interface limit: The backend supports owner-authorised resource movement between folders, but the current browser interface does not provide a move command. Folder creation, browsing, and renaming are available to end users.

B.6.9 Rename or Delete a Resource

• To rename a resource, select the pencil control in the library or Rename in the viewer, enter the desired name, and save. A stored file keeps its original extension when the new name omits it.

• To delete a resource, select the delete control, review the confirmation message, and choose Delete resource. Deletion removes the database record and its stored object. This action cannot be undone.

B.6.10 Sign Out

Select Sign out beneath the account name in the sidebar. The browser removes the stored access token and returns to the sign-in screen. Sign out before leaving a shared computer.

B.7 Storage and Deployment Configuration

Table B.6: Storage and Deployment Configuration

Mode	Required Configuration	Operational Notes
Local storage	STORAGE_BACKEND=local; optional UPLOAD_ROOT	Default mode. Files are stored beneath a private server directory. Back up both the upload directory and PostgreSQL database.
Supabase storage	STORAGE_BACKEND=supabase; SUPABASE_URL; SUPABASE_SERVICE_ROLE_KEY; SUPABASE_STORAGE_BUCKET	Create the bucket before startup and keep it private. The default bucket name is documents. The service-role key must remain server-side.
Hosted database	DATABASE_URL	Use the provider's PostgreSQL connection string. It takes precedence over the individual POSTGRES variables.

A hosted service must run the FastAPI application from the backend directory with an equivalent command to python -m uvicorn app.main:app --host 0.0.0.0 --port PORT, where PORT is supplied by the platform. Configure every secret in the hosting provider's environment settings rather than in source control. Use HTTPS at the public endpoint and verify /health after deployment.

Backup rule: A usable backup requires both the PostgreSQL database and the associated private object storage. Backing up only one of them can leave resource metadata without files or files without their owner and classification records.

B.8 Troubleshooting

Table B.7: Troubleshooting

Symptom	Likely Cause	Corrective Action
Application fails during startup	PostgreSQL is unavailable or environment values are missing.	Start PostgreSQL; confirm host, port, database, user, password, or DATABASE_URL; then restart Uvicorn.
Password or account form is rejected	The values do not meet validation rules or the email is already registered.	Use a valid email; use a 2-100 character name; use an 8-128 character password with uppercase, lowercase, and a number.
Session returns to sign-in	The access token expired, was invalid, or was removed.	Sign in again. Confirm JWT_SECRET_KEY is stable across server restarts and review ACCESS_TOKEN_EXPIRE_MINUTES.
Upload returns unsupported type	The extension is not in the supported list or a PDF is invalid.	Use PDF, DOCX, PPTX, JPG/JPEG, PNG, or MP4 and confirm that the file opens normally.
Upload exceeds limit	File is larger than MAX_UPLOAD_BYTES.	Reduce the file size or have the administrator raise the limit after assessing storage and server capacity.

Symptom	Likely Cause	Corrective Action
Webpage cannot be added	The site requires login, blocks automated access, redirects unexpectedly, or depends on client-side rendering.	Confirm the URL in a browser and use another accessible source or upload a saved supported document.
Stored file cannot be opened	Local upload path or Supabase credentials/bucket are wrong, or the object is missing.	Verify STORAGE_BACKEND, UPLOAD_ROOT, Supabase values, bucket name, and object availability.
Category looks incorrect	Evidence is weak, ambiguous, or unavailable for that format.	Open Review category, inspect alternatives, rerun classification, or save a manual correction.
No topics are shown	The source has little extractable text or is an image/video.	Add a manual topic; for scanned material, provide a text-based version because OCR is not implemented.
Folder appears empty	The user is at the wrong hierarchy level or no resource is assigned there.	Use breadcrumbs to return to My Library and reopen the course and material-type folders.

B.9 Security, Privacy, and Operational Guidance

• Use a long random JWT secret in every non-development installation and protect all environment secrets from source control, screenshots, logs, and client-side code.

• Grant the PostgreSQL and storage credentials only the access needed by this application. Keep Supabase buckets private and never expose the service-role key to a browser.

• Serve hosted installations through HTTPS. Restrict network access to database services and monitor failed authentication, storage, and database errors.

• The system stores academic content and account identifiers. Upload only material the user is authorised to store, and follow institutional copyright, privacy, and retention requirements.

• Classification and topic labels are decision-support outputs, not authoritative academic judgments. Review low-confidence or important results before relying on them.

• Deletion is irreversible through the interface. Maintain tested backups and define a retention and recovery procedure before production use.

• Local development defaults are not a production security baseline. Production deployment still requires platform hardening, updates, monitoring, rate limiting, recovery testing, and an independent security assessment.

B.10 Quick Reference

Table B.8: Quick Reference

Task	Action
Start locally	Activate .venv, change to backend, and run python -m uvicorn app.main:app --reload.

Task	Action
Open application	Browse to http://127.0.0.1:8000.
Check service	Browse to /health and confirm the ok response.
Run tests	Set PYTHONPATH to backend and run python -m pytest backend/tests from the project root.
Add a file	Add resource > choose file > Analyse resource.
Add a webpage	Add webpage link > enter URL > Analyse webpage.
Correct category	Select resource category > inspect evidence > choose category > save.
Edit topics	Open resource > Related topics > add or remove a topic.
Browse folders	My Library > course folder > material-type folder.
Rename or delete	Use the pencil or delete control beside the resource.
End session	Select Sign out.

Before presenting or assessing the system, confirm that the database and storage services are available, the health endpoint responds, a test account can sign in, one supported document can be uploaded and retrieved, a category can be reviewed, a folder can be browsed, and the user can sign out.

APPENDIX C: ADDITIONAL SYSTEM DESIGNS / DIAGRAMS

This appendix reproduces four detailed Chapter 4 design diagrams for closer inspection. They supplement the main report and do not add separate implementation or evaluation claims.

C.1 Diagram Index

Table C.1: Diagram Index

Appendix Figure	Chapter 4 Source	Design
Figure C.1	Figure 4.4	Resource Upload and Organisation Activity Diagram
Figure C.2	Figure 4.5	Resource Processing Sequence Diagram
Figure C.3	Figure 4.7	Entity-Relationship Diagram
Figure C.4	Figure 4.9	Classification and Human-Review Flowchart

C.2 Resource Upload and Organisation Activity Diagram

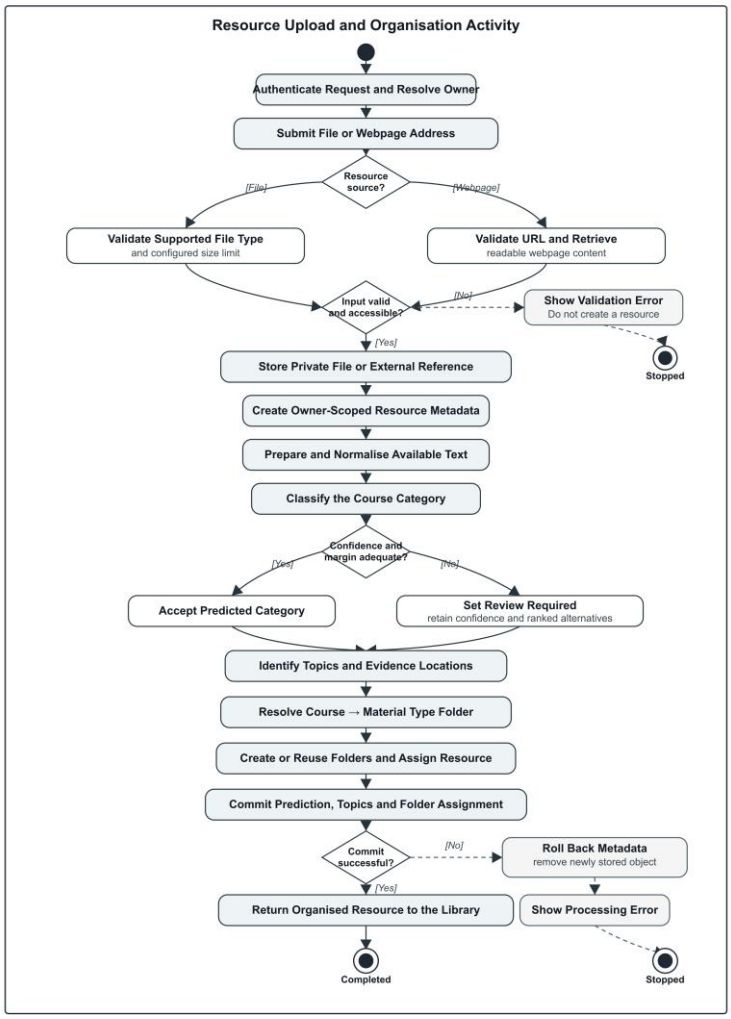


Figure C.1: Reproduction of Figure 4.4 - Resource Upload and Organisation Activity Diagram

C.3 Resource Processing Sequence Diagram

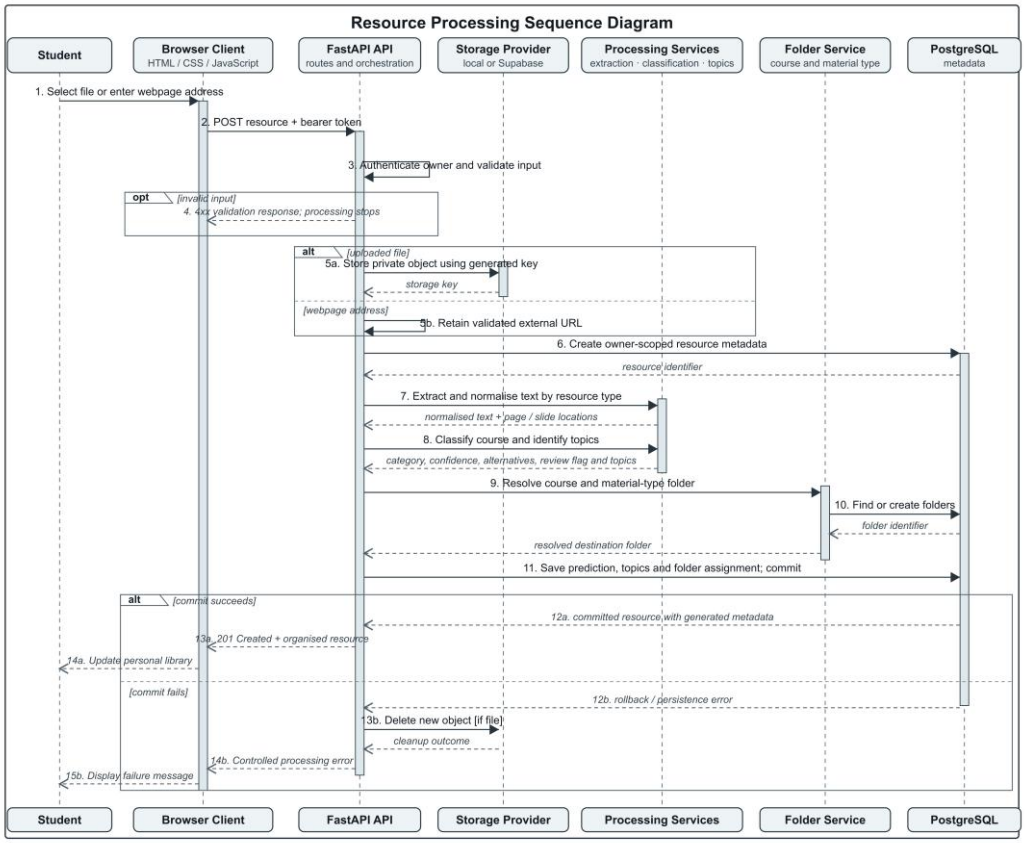


Figure C.2: Reproduction of Figure 4.5 - Resource Processing Sequence Diagram

C.4 Entity-Relationship Diagram

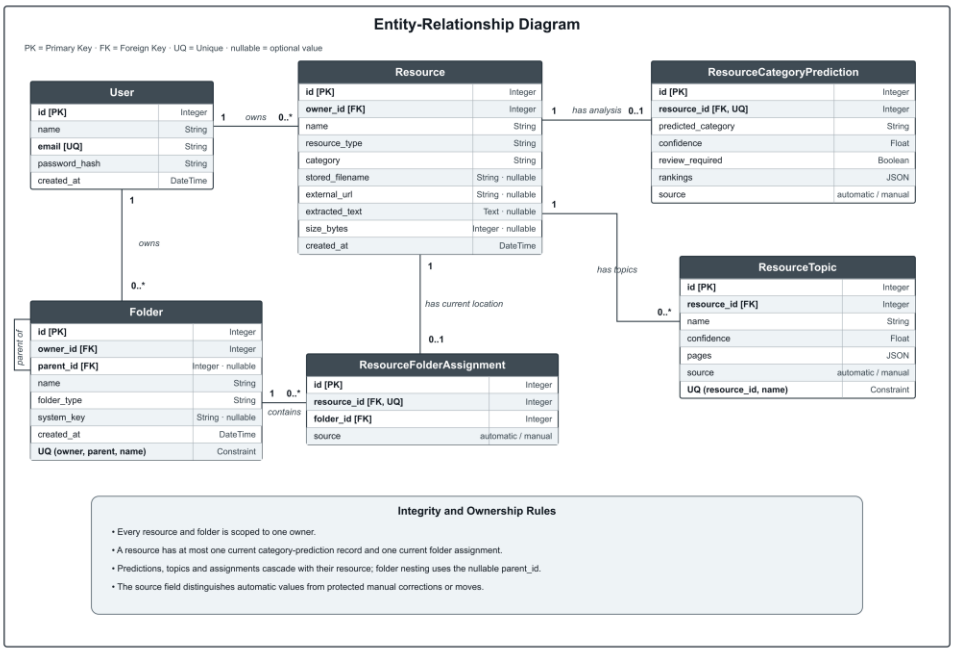


Figure C.3: Reproduction of Figure 4.7 - Entity-Relationship Diagram

C.5 Classification and Human-Review Flowchart

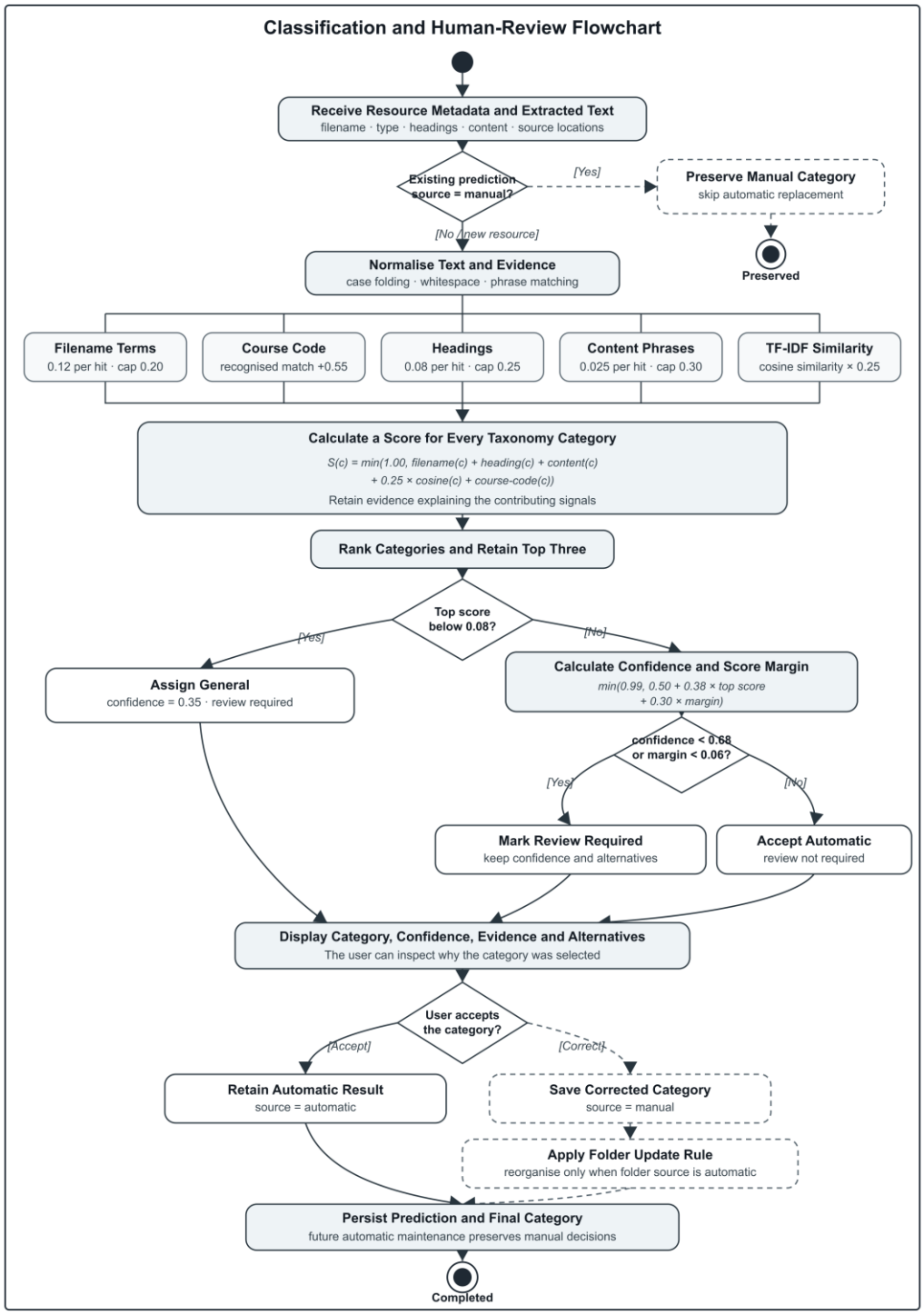


Figure C.4: Full-size reproduction of Figure 4.9 - Classification and Human-Review Flowchart

APPENDIX D: ADDITIONAL TEST CASES AND RESULTS

This appendix expands the executable test evidence summarised in Chapter 6. It records the repository test inventory, principal assertions, ownership checks, controlled classification and topic fixtures, and local core-service timings for the Intelligent File Management System only.

D.1 Reproduced Test Run

Table D.1: Reproduced Test Run

Item	Recorded Value
Date reproduced	23 October 2026
Python	3.14.5
Test runner	pytest 8.4.2
Integration client	FastAPI TestClient
Database and storage	Isolated in-memory SQLite database, SQLAlchemy StaticPool, and temporary local object storage
Collected tests	23
Result	23 passed; 0 failed; 0 skipped
Elapsed suite time	13.42 seconds
Non-functional warning	One optional pytest cache-directory warning; assertions and status were unaffected

Repository command:

```
$env:PYTHONPATH = "backend"
.\.venv\Scripts\python.exe -m pytest backend/tests
```

D.2 Complete Automated Test Inventory

Table D.2: Complete Automated Test Inventory

ID	Executable Test	Principal Behaviour	Result
AT-01	test_registers_user_without_exposing_password_hash	Registration, email normalisation, and password-hash non-disclosure	Passed
AT-02	test_rejects_duplicate_email	Duplicate-account conflict handling	Passed
AT-03	test_rejects_weak_password	Password policy validation	Passed
AT-04	test_login_and_current_user_flow	Login token and current-user retrieval	Passed
AT-05	test_rejects_bad_credentials_and_invalid_tokens	Bad credentials, missing token, and invalid token rejection	Passed

ID	Executable Test	Principal Behaviour	Result
AT-06	test_upload_routes_require_authentication	Authentication requirement and unsupported-media rejection	Passed
AT-07	test_resource_is_persisted_listed_served_and_deleted	Create, persist, list, serve, view, download, and delete lifecycle	Passed
AT-08	test_users_cannot_access_each_others_resources	Cross-user resource concealment	Passed
AT-09	test_resources_can_be_renamed_and_empty_names_are_rejected	Resource rename and empty-name validation	Passed
AT-10	test_webpage_title_is_used_and_can_be_renamed	Webpage title persistence and rename	Passed
AT-11	test_users_cannot_rename_each_others_resources	Cross-user rename protection	Passed
AT-12	test_topic_extraction_returns_confidence_and_page_references	Automatic topics, confidence, and page evidence	Passed
AT-13	test_owner_can_add_and_remove_a_manual_topic	Manual topic lifecycle and page normalisation	Passed
AT-14	test_other_user_cannot_modify_resource_topics	Cross-user topic protection	Passed
AT-15	test_hybrid_classifier_separates_data_mining_from_operating_systems	Hybrid classifier subject separation	Passed
AT-16	test_unza_taxonomy_improves_course_code_classification	Course-code-supported classification	Passed
AT-17	test_owner_can_correct_category_and_manual_choice_is_persisted	Manual category correction and provenance	Passed
AT-18	test_unknown_category_and_other_user_corrections_are_rejected	Category validation and owner protection	Passed
AT-19	test_uploads_are_organized_into_course_and_material_folders	Automatic course/material hierarchy	Passed
AT-20	test_folders_can_be_renamed_and_manual_resource_moves_are_preserved	Folder rename and persistent manual move	Passed
AT-21	test_users_cannot_browse_or_move_files_into_another_users_folders	Cross-user folder browse and move protection	Passed
AT-22	test_youtube_uses_oembed_title_instead_of_generic_page_title	YouTube oEmbed title preference	Passed
AT-23	test_webpage_prefers_open_graph_title	Open Graph title preference	Passed

D.3 Representative Assertions and Observed Results

Table D.3: Representative Assertions and Observed Results

ID	Action	Expected Result	Observed
TC-01	Register with uppercase email and valid password	201; email normalised; password hash absent	Matched; pass
TC-02	Repeat email and submit weak password	409 for duplicate; 422 for weak password	Matched; pass
TC-03	Login, request /auth/me, then use bad or invalid credentials	Valid token succeeds; invalid authentication returns 401	Matched; pass
TC-04	Upload anonymously and submit unsupported text file after login	401 when anonymous; 415 for unsupported media	Matched; pass
TC-05	Create image resource, list, serve, view, download, and delete	Metadata/bytes persist; deletion followed by 404	Matched; pass
TC-06	Attempt resource operations from another account	Protected operations concealed with 404	Matched; pass
TC-07	Capture and rename a webpage	Extracted title stored; manual name persists	Matched; pass
TC-08	Extract topics from three numbered pages	Four expected topic/page pairs and confidence bounds	Matched; pass
TC-09	Add/remove manual topic with duplicate or invalid pages	Manual source retained; pages normalised to 2 and 4	Matched; pass
TC-10	Run five controlled category fixtures	Five named categories selected	Matched; pass
TC-11	Correct category; try unknown and cross-user correction	Manual result persists; 422 and 404 rejection paths	Matched; pass
TC-12	Upload Data Mining lecture slides	Course/Lecture Slides hierarchy and metadata	Matched; pass
TC-13	Rename folder, create Exam Revision, move resource, repeat cross-user	Owner move persists; cross-user paths return 404	Matched; pass
TC-14	Extract titles from mocked YouTube and Open Graph pages	Descriptive metadata title preferred	Matched; pass

D.4 Cross-User Ownership Checks

Table D.4: Cross-User Ownership Checks

No.	Attempt	Observed Status
1	Read resource metadata	404
2	Open inline stored file	404
3	Open resource viewer metadata	404
4	Download stored resource	404
5	Delete resource	404
6	Rename resource	404
7	Modify resource topics	404
8	Correct resource category	404
9	Browse another user's folder	404

No.	Attempt	Observed Status
10	Move a resource into another user's folder	404

All ten non-owner operations returned 404. This confirms the non-disclosing ownership boundary for the covered paths; it is not a substitute for penetration testing or a complete production security assessment.

D.5 Controlled Classification and Topic Fixtures

Table D.5: Controlled Classification and Topic Fixtures

Fixture	Expected	Observed
Category 1	Data Mining	Matched; confident case not flagged
Category 2	Operating Systems	Matched
Category 3	Cybersecurity	Matched; course-code evidence
Category 4	Database Systems	Matched; course-code evidence
Category 5	Computer Networks	Matched; course-code evidence
Topic/page 1	OSI Model - page 1	Matched
Topic/page 2	TCP/IP - page 2	Matched
Topic/page 3	Routing - page 2	Matched
Topic/page 4	Subnetting - page 3	Matched

D.6 Local Core-Service Timing

After ten warm-up calls, each service processed the same short controlled fixture 100 times using a high-resolution process timer. The measurements exclude transfer, parsing, persistence, object storage, browser rendering, network delay, and concurrency.

Table D.6: Local Core-Service Timing

Operation	Calls	Median	95th Percentile	Interpretation
Hybrid classification	100	191.0884 ms	253.9642 ms	Core scoring on a short fixture; not upload latency
Topic extraction	100	10.8519 ms	22.7001 ms	Three short page records; not a large-document result

D.7 Evidence Limits

- The taxonomy-derived seed rows are not an independent labelled dataset and cannot support scientific classifier accuracy claims.
- No completed participant task logs, consented observations, questionnaires, or System Usability Scale dataset are available.
- Local core-service timings do not predict large-file, concurrent, networked, or hosted end-to-end performance.
- Passing automated assertions supports the tested behaviours only; it does not establish production security, availability, scalability, or long-term academic benefit.

APPENDIX E: DATA COLLECTION INSTRUMENTS

This appendix contains instruments prepared for requirements elicitation of the Intelligent File Management System. It excludes recommendation, scheduling, reminder, and progress-tracking questions assigned to other group components.

E.1 Requirements-Gathering Questionnaire

E.1.1 Participant Information and Consent

Purpose: to understand how university students store, find, classify, and correct digital academic resources. Estimated time: 6-8 minutes. Responses were collected as anonymous, voluntary, reported only in aggregate, and collected only after the supervisor approved.

1. I am at least 18 years old, have read the information above, and voluntarily agree to participate.

( ) Yes ( ) No - end questionnaire

E.1.2 Student Context and Current Practice

2. What is your current level of study?

( ) First year ( ) Second year ( ) Third year ( ) Fourth year ( ) Fifth year or above ( ) Postgraduate ( ) Other

3. What is your school, faculty, or broad programme area? Do not provide a computer number.

Response:

_____

4. How often do you use digital academic resources?

( ) Several times a day ( ) About once a day ( ) Several times a week ( ) About once a week ( ) Less often

5. Which resource formats do you use most often? Select all that apply.

( ) PDF ( ) PowerPoint ( ) Word ( ) Webpages ( ) Images/scanned notes ( ) Video ( ) Other

6. Where do you keep or access most digital academic resources? Select all that apply.

( ) Computer folders ( ) Phone/tablet folders ( ) Cloud storage ( ) Email/messaging ( ) Learning platform ( ) Bookmarks ( ) Removable storage ( ) Other

7. How difficult is it to find a specific resource when needed?

( ) Very easy ( ) Easy ( ) Neither ( ) Difficult ( ) Very difficult

8. When a resource is difficult to find, how long do you usually search?

( ) Under 1 minute ( ) 1-5 minutes ( ) 6-10 minutes ( ) Over 10 minutes ( ) I usually stop

9. Which problems do you experience? Select all that apply.

( ) Scattered resources ( ) Unclear filenames ( ) Duplicates ( ) Cannot remember location ( ) Not organised by course/topic ( ) Poor search results ( ) Hard to identify document contents ( ) No major problem ( ) Other

10. How do you normally organise resources? Select all that apply.

( ) Course/module ( ) Topic ( ) Resource type ( ) Semester/year ( ) Assessment/exam ( ) Search only ( ) No consistent method ( ) Other

11. Describe one recent difficulty when finding or organising a digital academic resource.

Response:

_____

_____

____

E.1.3 Importance of Intelligent File-Management Features

Rate each feature: 1 = Not important, 2 = Slightly important, 3 = Moderately important, 4 = Very important, 5 = Essential.

Table E.1: Importance of Intelligent File-Management Features

Feature	1	2	3	4	5
Secure personal account and private library	( )	( )	( )	( )	( )
Upload documents, images, video and webpage links	( )	( )	( )	( )	( )
Automatic organisation by course and resource type	( )	( )	( )	( )	( )
Automatic subject or course classification	( )	( )	( )	( )	( )
Topic labels with page or slide references	( )	( )	( )	( )	( )
Search and filter by name, category, type or topic	( )	( )	( )	( )	( )
Create and rename folders	( )	( )	( )	( )	( )
View or download a resource	( )	( )	( )	( )	( )
See confidence and alternative categories	( )	( )	( )	( )	( )
Correct an incorrect category or topic label	( )	( )	( )	( )	( )

12. Which three features above should have the highest priority for a first usable version?

1. _____ 2. _____ 3. _____

13. Which privacy controls would you expect? Select all that apply.

( ) Explanation of data stored ( ) Private owner-only resources ( ) View stored metadata ( ) Delete resources ( ) Defined retention period ( ) Other

14. If the system worked reliably, how likely would you be to use it during a semester?

( ) Very unlikely ( ) Unlikely ( ) Unsure ( ) Likely ( ) Very likely

15. What single improvement should the system make to your present resource-management process?

Response:

_____

_____

____

APPENDIX F: ADDITIONAL CODE / CONFIGURATION

This appendix records selected configuration and interface material needed to reproduce the Intelligent File Management System. The authoritative implementation remains in the repository. Values below are sanitized examples; the populated .env file and all secrets are intentionally excluded.

F.1 Sanitized Environment Configuration

```
POSTGRES_HOST=localhost
POSTGRES_PORT=5433
POSTGRES_DB=intelligent_file_manager
POSTGRES_USER=postgres
POSTGRES_PASSWORD=replace-with-database-password
#                           DATABASE_URL=postgresql://user:password@host:5432/database
JWT_SECRET_KEY=replace-with-a-long-random-secret
ACCESS_TOKEN_EXPIRE_MINUTES=30
MAX_UPLOAD_BYTES=20971520
STORAGE_BACKEND=local
#                                    UPLOAD_ROOT=C:/absolute/path/to/private/uploads
#                                       SUPABASE_URL=https://project-ref.supabase.co
#                            SUPABASE_SERVICE_ROLE_KEY=replace-with-server-side-secret
SUPABASE_STORAGE_BUCKET=documents
```

DATABASE_URL takes precedence over the individual PostgreSQL settings. Supabase values are required only when STORAGE_BACKEND is set to supabase. The service-role key must remain server-side and must never be embedded in frontend code.

F.2 Local Start and Test Commands

Create the environment and install dependencies from the project root:

```
python                          -m                      venv                      .venv
.\.venv\Scripts\Activate.ps1
python -m pip install -r backend\requirements.txt
```

Start the application:

```
Set-Location                                                                  backend
python -m uvicorn app.main:app --reload
```

Run the automated tests from the project root:

```
$env:PYTHONPATH                          =                          "backend"
.\.venv\Scripts\python.exe -m pytest backend/tests
```

F.3 Implemented API Inventory

Table F.1: Implemented API Inventory

Method	Path	Responsibility
POST	/auth/register	Create a validated account
POST	/auth/login	Issue a short-lived bearer token
GET	/auth/me	Return the authenticated account
GET	/categories	List controlled Computer Science categories
GET	/resources	List the owner's resources
POST	/resources	Upload and process a supported file

Method	Path	Responsibility
POST	/resources/webpage	Capture and process a webpage
GET	/resources/{id}	Return owner-scoped resource metadata
GET	/resources/{id}/viewer	Return viewer metadata and extracted pages/slides
GET	/resources/{id}/file	Serve a stored file inline
GET	/resources/{id}/download	Download a stored file
PATCH	/resources/{id}/name	Rename a resource
PATCH	/resources/{id}/category	Save a manual category correction
POST	/resources/{id}/reclassify	Run the current classifier again
POST	/resources/{id}/topics	Add a manual topic
DELETE	/resources/{id}/topics/{topic_id}	Remove a topic
DELETE	/resources/{id}	Delete metadata and stored object
GET	/folders/contents	Browse owner-scoped folders and resources
POST	/folders	Create a nested folder
PATCH	/folders/{id}	Rename a folder
PATCH	/folders/resources/{resource_id}/move	Move a resource and record manual provenance
GET	/health	Return application health status

F.4 Principal Implementation Modules

Table F.2: Principal Implementation Modules

Repository Location	Responsibility
backend/app/main.py	Application composition, lifespan tasks, API routers, health route, and static frontend
backend/app/database.py	Environment-based PostgreSQL connection and SQLAlchemy session management
backend/app/api/auth.py	Registration, login, current-user resolution, and owner identity
backend/app/api/upload.py	Resource lifecycle, category review, topics, file serving, and webpage capture
backend/app/api/folders.py	Folder browse, create, rename, and manual resource move
backend/app/services/security.py	Argon2 password hashing and JWT issue/validation
backend/app/services/storage_service.py	Local and Supabase private object storage abstraction
backend/app/services/classification_service.py	Controlled taxonomy, evidence scoring, TF-IDF similarity, confidence, and alternatives
backend/app/services/topic_service.py	Controlled and heading-derived topics with confidence and locations
backend/app/services/folder_service.py	Automatic Course-to-Material-Type organisation and preservation rules
backend/app/services/*_service.py	PDF, DOCX, PPTX, and webpage extraction adapters
frontend/index.html and script.js	Authentication, dashboard, library, submission, filtering, review, and folders

Repository Location	Responsibility
frontend/viewer.html and viewer.js	Resource viewing, download, rename, and topic editing
backend/tests	Functional, ownership, classifier, topic, folder, and webpage service tests

F.5 Storage Selection Logic

```
if STORAGE_BACKEND == 'local':
    resolve object path beneath UPLOAD_ROOT
    reject path traversal
    write, read, or delete the private file
elif STORAGE_BACKEND == 'supabase':
    require SUPABASE_URL and SUPABASE_SERVICE_ROLE_KEY
    use SUPABASE_STORAGE_BUCKET (default: documents)
    upload, download, or remove the private object
else:
    raise unsupported-storage error
```

F.6 Processing and Organisation Logic

```
authenticate owner
validate type, size, and content
extract page/slide text when supported
score course-code, filename, heading, term, and TF-IDF evidence
store category, confidence, alternatives, review flag, and source
extract controlled and heading-derived topics
store object using an owner-specific generated path
place resource in Course -> Material Type
preserve manual category and folder decisions during later backfills
```

APPENDIX G: OTHER SUPPORTING MATERIAL

This appendix provides a compact traceability and reproducibility record for the Intelligent File Management System. It links requirements to implementation and evidence without extending the report's claims beyond the tested individual subsystem.

G.1 Requirements-to-Evidence Traceability

Table G.1: Requirements-to-Evidence Traceability

ID	Requirement	Implementation	Evidence	Assessment
FR-01	Register, sign in, and restore user identity	api/auth.py; services/security.py	AT-01 to AT-05	Implemented and tested
FR-02	Upload supported files	api/upload.py; storage_service.py	AT-06, AT-07	Implemented and tested
FR-03	Capture webpage resources	api/upload.py; webpage_service.py	AT-10, AT-22, AT-23	Implemented and tested
FR-04	Extract text from supported documents	pdf/docx/pptx services	AT-07, AT-12	Implemented for text-bearing formats
FR-05	Classify by Computer Science category	classification_service.py	AT-15 to AT-18	Implemented; independent accuracy pending
FR-06	Generate and edit topics	topic_service.py; api/upload.py	AT-12 to AT-14	Implemented and tested
FR-07	Automatically organise folders	folder_service.py; api/folders.py	AT-19 to AT-21	Implemented and tested
FR-08	Search, filter, browse, and retrieve	frontend/script.js; viewer.js	AT-07, AT-19	Implemented; participant usability pending
FR-09	Rename resources and folders	api/upload.py; api/folders.py	AT-09 to AT-11, AT-20	Implemented and tested
FR-10	Review/correct automatic decisions	category/topic endpoints	AT-13, AT-17, AT-18	Implemented and tested
FR-11	Download and delete stored files	api/upload.py; storage_service.py	AT-07, AT-08	Implemented and tested
NFR-01	Owner isolation	owner-scoped queries and paths	Ten 404 ownership checks	Covered for tested operations
NFR-02	Explainable automation	rankings, confidence, review flag, provenance	AT-12, AT-15 to AT-18	Implemented for controlled fixtures
NFR-03	Configurable deployment	database.py; storage_service.py; .env.example	Configuration inspection	Local/Supabase options documented
NFR-04	Usability	Browser interface	Appendix E instrument	Formal participant evaluation pending

G.2 Repository Structure Map

```
intelligent-file-management-system/
  backend/app/api/          authentication, resources, folders
  backend/app/models/       users, resources, topics, folders
  backend/app/services/     extraction, classification, storage
  backend/app/database.py   SQLAlchemy engine and sessions
  backend/app/main.py       FastAPI application
  backend/data/             controlled taxonomy resources
  backend/tests/            executable pytest suite
  frontend/ ; output/diagrams/     browser/viewer; design assets
  uploads/ ; .env.example ; README.md   private files; config; commands
```

G.3 Reproducibility Checklist

Table G.2: Reproducibility Checklist

No.	Verification Item	Expected Evidence
1	Use Python 3.14 and install backend/requirements.txt	Documented in Appendix B/F
2	Create PostgreSQL database and configure connection	Required before startup
3	Set a unique JWT secret	Required outside disposable development
4	Choose local or Supabase private storage	Do not expose service credentials
5	Start FastAPI from the backend directory	Health endpoint should return ok
6	Run the 23-test repository suite	Expected result: 23 passes for recorded revision
7	Create test account and upload a supported file	Verify list, viewer/download, category, topics, and folder
8	Test manual category/topic/folder decisions	Verify corrections persist
9	Test sign-out and owner separation	Use separate accounts and non-sensitive samples
10	Back up database and private object storage together	Required for recoverable operation

G.4 Evidence and Claim Boundary

Table G.3: Evidence and Claim Boundary

Included Individual Work	Excluded or Pending
Authentication, owner-scoped resources, private storage, extraction, classification, topics, folders, browser resource workflows, automated tests, documentation, and deployment configuration	Personalised recommendations, study schedules, reminders, progress tracking, and other group-member functions
23 passing automated tests and ten covered cross-user 404 results	Penetration testing, production security certification, deployed load/scalability testing, and availability measurement
Five controlled category fixtures and four topic/location assertions	General classifier accuracy, precision, recall, F1, confusion matrix, or performance on an independent labelled corpus
Local core-service timing on short controlled fixtures	End-to-end upload latency, large-document performance, network delay, browser rendering, object-store latency, or concurrency
Prepared requirements and usability instruments	Participant-rated usability or satisfaction because no completed consented dataset is available

G.5 Supporting Evidence Index

Table G.4: Supporting Evidence Index

Evidence	Location
System designs	Appendix C and output/diagrams
Extended test inventory and results	Appendix D and backend/tests
Requirements and planned usability instruments	Appendix E
Sanitized configuration and interface inventory	Appendix F, .env.example, README.md
Individual contribution boundary	Appendix A
Installation and user operation	Appendix B
Evaluation interpretation and limits	Chapter 6 and Chapter 7