Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

StudyBuddy-AI

By
Keith Moffitt

April 27, 2025

Minor Dissertation

**Department of Computer Science & Applied Physics,
School of Science & Computing,
Atlantic Technological University (ATU), Galway.**

B.Sc. (Hons) in Software Development

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The landscape of education has seen a significant transformation in the last few decades, driven by an increasing volume of online academic materials students must process and learn during their time in the education system. Today's students face the challenge of effectively managing and extracting relevant information from the large array of academic materials provided to them. Studies show that many university students globally are experiencing issues with information overload, and many express a preference for print materials when engaging with intense academic reading, this is mainly due to enhanced focus and comprehension [1]. This preference suggests that digital formats may present a challenge regarding effective information processing. This issue has been increased by the growth of digital learning platforms and repositories, these platforms may make information more available to students but may not necessarily provide the tools for meaningful processing or interaction with the content.

In today's educational world, students are regularly provided with extensive lecture notes, research papers, textbooks, and other materials across many subjects and modules. These educational resources can be extremely valuable but also present the issue of efficient information retrieval and application. A traditional approach to studying educational resources would involve manually skimming or searching through countless documents, using basic keyword searches and copying and pasting, or creating handwritten notes, of the relevant information. These methods are increasingly inadequate and time-consuming given the volume and complexity of modern-day educational content.

A significant issue emerges when we consider the international nature of today's education. We are seeing an increase in the number of students studying in non-native language universities and colleges worldwide, language barriers can compound the difficulties students already face when using educational resources. Research shows international students may face significant language-related challenges. These challenges show in reading comprehension, academic writing, and oral communication in the classroom.[2] These students are often forced to resort to translation tools, which are often disconnected from the study workflow.

1.2 Problem Statement

The volume of academic and research documentation used by students is ever increasing in volume. This shows there is a need for a more intelligent system for processing these documents. The current solutions for managing and interacting with these materials present several significant limitations:

1. **Information Retrieval Challenges:** Students struggle to find the relevant information within large documents accurately while also quickly. Traditional attempts at this task such as keyword searches will often return irrelevant information or miss information related to context. Research shows information overload can lead to difficulty in identifying what information is relevant. It can also decrease decision quality as well and impact the learning process.[3]

2. **Processing of large documents:** Converting large amounts of academic content into effective study materials is an extremely time-consuming task. Students lack the tools required to transform these large documents into concise summaries that are easy to understand and meet they're learning needs.

3. **Language Barriers:** International students or any individual using academic materials in non-native languages face additional issues in understanding these academic materials. Existing tools for translation operate separately from other document processing systems, creating a disconnect in workflow.

4. **Limited Self Assessment Opportunities:** Students lack methods that allow them to effectively test how well they understand their document contents. Very few tools are available that can generate relevant questions based on the student's specific academic materials.

While Tools do exists that allow students to address these individual issues - translation services like Google Translate, and various AI models for information extractions - they all operate independently of on another, this affects students workflows and requires them to juggle multiple applications. This reduces efficiency and also creates barriers when trying the learn effectively.

1.3 Project Objectives

StudyBuddy-AI aim to remedy the challenges outlined above. It will achieve this by integrating an AI-powered learning assistant which aims to help students interact with their academic materials more effectively. The main objectives of this project are as follows.

1. Develop a web-based application that is capable of processing and storing uploaded PDF documents provided by students, which will create a personalised knowledge base for each student.

2. Implement a Retrieval Augmented Generation (RAG) system which will allow for context-aware AI responses, which will be solely based on the students' document knowledge base.

3. Provide tools to the students which will allow for customisable summaries based on queries and the document content related to these queries retrieved from the students' document knowledge base.

4. Create a system that allows students to generate multiple-choice question quizzes based on document content, tailored to the required number of questions and complexity the student requires with feedback on the quiz results.

5. Integrate multilingual support into the application, which will allow students to interact with their documents in their chosen language limiting language barriers.

6. Establish a secure and persistent user authentication system that enables the storage and retrieval of documents between sessions.

Achieving these objectives will allow StudyBuddy-AI to transform how students use and interact with they're documents and academic materials. This approach will provide a personalised and efficient experience when studying.

1.4 Technical Approach Intro

The core of StudyBuddy-AI is the implementation of a Retrieval Augmented Generation (RAG) based system. This is a new approach to how natural language processing is done, using both strengths or retrieval-based and generation-based systems. RAG is a method aimed at enhancing neural text generation with retrieval mechanisms, demonstrating improvements in knowledge-intensive tasks.[4] RAG addresses limitations to exist in large language models, this is achieved by grounding responses in retrieved information rather than relying solely on the model's trained knowledge.

The RAG approach that is integrated into the StudyBuddy-AI system works through multiple steps in a process:

1. **Document Processing:** Students upload a PDF document. The system extracts the text from these documents and processes it into manageable chunks.

2. **Vector Embeddings:** The extracted chunks of document contents are then converted into a vector embedding, these are numerical representations which capture semantic meaning, using embedding models. This is the process of dense passage retrieval which is effective for question-answering tasks.[5]

3. **Similarity Search:** When a query is sent by a user, the system identifies chunks based on relevance by calculating the similarity between the query posed and the stored embeddings.

4. **Context-Enhanced Generation:** The relevant retrieved chunks are used to provide context to the AI Model. This allows the AI to generate responses that are grounded in the user's documents. This aims to improve the output accuracy of the AI to be more in line with the users understanding according to the documents, rather than outputs grounded in the model's trained knowledge.

The approach outlined above aims to ensure that the AI responses are relevant and accurate with a direct tie to the student's material. Unlike conventional search or pure AI generation systems, RAG allows for a personalised and precise learning experience.

The implementation of this system involves a Flask-based backend API, which handles document processing, embedding and storage, retrieval and AI interactions. The Streamlit frontend of this application provides an intuitive user interface. Document chunking and embeddings are handled by the LangChain framework. These embeddings are stored in a vector database, for this system ChromaDB is used. as it allows for efficient similarity search which is an essential part of the RAG architecture.

1.5 Dissertation Structure

The rest of this dissertation will be organised as follows:

Chapter 2 Methodology: This details the research and software development behind this project, which includes requirements gathering, evaluation approaches, and the overall process of developing StudyBuddy-AI

Chapter 3 Technology Review: This chapter will provide a literature review of all the relevant technologies. The focus of this section will be on the document processing system, RAG architecture, vector databases, and the application of AI in education.

Chapter 4 System Design: Here we will provide a detailed explanation of StudyBuddy-AI's architecture and its implementation. This will include the pipeline behind the document ingestion system, RAG system, frontend interface, and backend services.

Chapter 5 Evaluation: This chapter focuses on testing methodology, user feedback, performance metrics and analysis of the developed system.

Chapter 6 Conclusion: This chapter aims to summarise the achievements of the project, while also addressing its limitations. Areas of improvement will be covered as well as areas and features for future development. The learning outcome will also be reflected here.

This structure aims to provide a comprehensive overview of the development, implementation and evaluation of StudyBuddy-AI as a solution to the issues and challenges that modern students face on a daily basis in the process of learning from academic materials.

Chapter 2

Methodology

2.1 Introduction

This chapter aims to outline the methodology employed in the process of developing StudyBuddy-AI. It covers both the software development methodology that helped the implementation process, as well as the research methodology which was used to gather requirements, evaluate the design choices, and assess the final system. The methodology was selected to ensure the development process was evidence-based and also aligned with the objective set out for this project.

The development of an educational tool which employs AI-powered tools, such as StudyBuddy-AI presents a unique challenge. There needs to be a balancing of technical concerns while also taking into consideration the user experience concerns. This all must be considered while working with relatively new and rapidly evolving technologies. The methodology required for a project of this stature needed to be flexible enough to allow for changes in the requirements, yet structured to ensure progress towards the defined goals remained continuous.

2.2 Software Development Methodology

2.2.1 Agile Development Approach

StudyBuddy-AI was developed using an Agile methodology, this was specifically adapted to a version of Scrum aimed for individual development. This approach was used due to its flexibility, with an emphasis on iterative development, while also allowing for changing requirements as the project evolved. While Scrum is usually designed for use by teams of developers, modifying it to suit a solo developer proved simple.

The development process was set in two-week sprints, each focusing on a minimum of one feature, while some focused on two, this was implemented based on the complexity of the feature being developed during a sprint. This approach allowed for:

1. **Incremental Development:** Building the system in small increments rather than aiming to develop the entire application from the start.
2. **Continuous Integration:** Integrating new code at regular intervals into the existing codebase allows for detecting issues early.
3. **Adaptability:** Handling of changing requirements and issues as they arise.
4. **Regular Reviews:** Evaluating progress at the end of each sprint allows us to adapt plans according to this progress.

This approach was maintained using the industry-standard tool for Agile project management, Jira. Which allowed for the creation of Epics for each feature, as well as user-stories for each epic. Jira also allowed for the creation and planning of each sprint and which features and user stories were associated with each sprint. Simplifying the process of tracking and completing sprints in a set two week time frame. This ensured regular progress was made.

It's important to acknowledge that this was a student project, the development process of StudyBuddy-AI was actively worked on alongside other academic commitments. These responsibilities regularly cause the need for adjustments to sprint schedules and to the overall development timeline. Rather than representing the methodology as a failure these adjustments demonstrate the flexibility the Agile approach offers in accommodating real-world constraints. Sprints and task allocations had to be occasionally modified to balance the development process while also allowing for other academic commitments, but still ensuring continuous development and advancements in StudyBuddy-AI. Showing it as a valuable asset to this project's development process.

2.2.2 Sprint Structure

Each sprint followed a consistent structure:

1. **Sprint Planning:** Selecting feature(Epic to implement during a sprint and breaking them down into specific tasks(user stories).
2. **Development:** Implementing these features, focusing on creating components that work even if unpolished.

3. **Review:** Assessing the work completed in the sprint against the goals outset before. Also assessing the state of the features implemented and identifying any further work required.

4. **Fine Tuning:** Completing any of the finishing touches and fine-tuning of the features implemented ensuring they meet the full requirements of the application.

This structure aimed for regular progress while ensuring the quality of the implemented feature was as high as reasonably possible. It also maintained flexibility to allow for changes based on discoveries made during development.

2.2.3 Version control

Git version control was used for the development of this project, with GitHub being the repository host. This allowed for:

- Tracking code changes over time.
- Documenting development through commit messages.
- Enabling rollback to previous versions.

This project were organised using two separate git repositories, one for the Streamlit frontend code, and one for the Flask backend API. This allows for maintaining independence between codebases. This allowed for flexibility in the development process. Despite the separation, careful coordination between backend and frontend was required to ensure API changes were properly reflected in the frontend implementation.

2.3 Research Methodology

2.3.1 Requirements Gathering

The requirements for StudyBuddy-AI were gathered through multiple methods:

1. **Literature Review:** Existing research on the challenges faced by students including document management systems, as well as the applications of AI in education were examined to identify common issues and identify possible solutions.

2. **Analysis of Competition** Analysing existing tools aimed at students, as well as document management systems was a key part of research as it allowed for identifying gaps and opportunities for innovation.

3. **User surveys** Conducting informal surveys with my peers as well as students in various degrees as well as levels of education helped to identify the needs of students and also the need for improvement in the area of document processing.

These multiple methods of research ensured the requirements created for StudyBuddy-AI were grounded in a practical understanding of the users needs.

2.3.2 Design Approach

The design process followed a user-centred design methodology, including:

1. **User stories:** Creating representative user stories using the informal survey data to influence design choices.
2. **Wireframes:** Sketching out an initial interface design for the application to explore different approaches to find the best possible experience for the user.
3. **Prototyping:** Developing interactive frontend prototypes of key features using Streamlit to validate design ideas through hands-on user testing and feedback.
4. **Iterative Refinement:** Contentiously refining the design based on user feedback and the changing software through the development process.

This approach ensured that the user experience was at the focal point of the design of this application, meeting theyre wants and needs, while also meeting the technical requirments of the project.

2.3.3 Evaluation methodology

Multiple methods of evaluation were employed while evaluating StudyBuddy-AI including:

1. **Continous Evaluation:** Continually undergoing evaluation throughout the development process aimed to identify and address issues early.
    - User Evaluation with potential users gave insight into how the application would be used and the limitation it currently faces when used in the way.
    - Technical performance testing of individual components throughout development.

2. **End of development Evaluation:** Evaluation of the complete system to assess its overall effectiveness in meeting the user's requirements.
    - Usability testing with 10 students from different degrees and levels of education.
    - Performance metrics collected for key systems and features of the application
    - User feedback surveys following the use of the system.

This dual approach to the evaluation of StudyBuddy-AI ensured both the development process and the final product were put through scrutiny and assessment against the user's requirements.

2.4 Project Planning and Timeline

StudyBuddy-AI was both planned and tracked using Jira, following an Agile methodology with two-week sprints. This process spanned from November 2024 to April 2025. User stories were defined for each feature of the application and tracked to completion

2.4.1 Project Phases

Sprint 1-2: Document Processing (November-December 2024)

- SCRUM-21: Document Upload Function
- SCRUM-22: Document Chunking implementation
- SCRUM-23: Document Storage System
- SCRUM-55: User Authentication

These initial sprints focused on implementing the key infrastructure for document ingestion and handling as well as user management setting the foundation required for the RAG implementation.

Sprint 3-4: User Feedback and Core Features (January-February 2025)

- SCRUM-28: Feedback Mechanism
- SCRUM-25: Q&A Function
- SCRUM-26: Summary Function

During this phase of development the core features were implemented including the RAG system, enabling basic querying and summarisation of uploaded document content. This phase also included the implementation of the mechanism to collect users feedback.

Sprint 5-6: Advanced and Multilingual Features(March 2025)

- SCRUM-24: Translation Function
- SCRUM-27: MCQ Generation Function

This phase saw the introduction of the most advanced features of the application, including multilingual support and the interactive multiple choice question quiz.

Sprint 7-8: Deployment and Documentation(April 2025)

- SCRUM-52: Application Deployment
- SCRUM-53: Dissertation writeup
- SCRUM-54: A0 Poster Design

The final sprints of this project were dedicated to deploying the application to production. These sprints were also dedicated to the documentation process of this application, this is achieved through this dissertation as well as an A0 poster presentation highlighting the key aspects behind the application.

2.4.2 Timeline Management

The project timeline was managed in Jira with a road map view which helped provide a clear understanding of task progression across sprints. This approach allowed for:

- Tracking of completed work with "DONE" markers for each user story and epic
- Visual representation of development progress within the current sprint
- Clear sprint goals to help organise work into manageable increments

Each feature was broken down into specific tasks which could be tracked through the development process from backlog to the completion of a feature. This approach to project management ensured continuous development of the application despite the complexity of implementing a RAG system and the coordination required between backend and frontend development across separate repositories.

2.5 Tools and Technologies

The selection of tools and technologies for StudyBuddy-AI was undertaken with several considerations:

1. **Functionality:** The ability to complete the project requirements effectively and fully
2. **Integration:** The compatibility of the components of the system when built
3. **Learning Curve:** Using new tools within the project parameters to use the project as an opportunity to further my technical ability and capacity to learn
4. **Scalability:** The ability of the system to handle growing user bases and document collections

2.5.1 Development Environment

- **Code Editor:** Visual Studio code with Python and Flask extensions
- **Version Control:** Git with GitHub for repository hosting
- **Project Management:** Agile approach using Jira to plan and visualise.

2.5.2 Backend Technologies

- **Framework:** Flask for API development
- **Authentication:** Firebase Authentication
- **Document Processing:** PyPDF for text extraction, Langchain for document chunking
- **Vector Database:** ChromaDB for embedding storage and similarity search
- **Embeddings:** OpenAI Embedding API
- **AI Integration:** OpenAI API (o3-mini Model)
- **Database:** SQLite for feedback storage

2.5.3 Frontend Technologies

- **Framework:** Streamlit for web-interface development
- **Styling:** Built-in Streamlit components
- **API Communication:** Requests library for API interactions
- **State Management:** Streamlit session state

2.5.4 Deployment technologies

- **Hosting Platform:** Render for both the frontend and backend services
- **Environment Management:** Requirments.txt in the repository outlines the required packages.
- **Monitoring:** Render built-in logs and metrics

2.5.5 Ethical Considerations

The development of StudyBuddy-AI required consideration of several ethical aspects:

1. **Data Privacy:** Ensuring the user's Data remains secure and private. This is particularly a concern when it comes to the uploading of documents
2. **Accessibility:** Designing the System to be accessible to users with different requirements. This influenced the multilingual features within the application
3. **Impact on Education:** Taking into account the impact that AI has in education. This put in focus the need for StudyBuddy-AI to be a supplement to current educational processes rather than replacing them

These ethical considerations influenced the implementation of the application for the outset affecting both the technical aspects of the application as well as the user experience.

2.6 Limitations

After completing this project it is important to outline some of the limitations of the chosen methodology:

1. **User Testing:** While user testing and feedback provided invaluable insights into what the user, the aspects of the project that could be improved, as well as the overall quality y of the project, the sample size of ten students means that not all potential user groups had to opportunity for input on the final product

2. **Solo Development** The lack of a team dynamic within the development process and Agile methodology caused a limit in the perspective for planning and implementation of this project

3. **Resource Constraints:** Time limitations affected this project particularly in terms of the user and technology research which took place before the start of development, as well as during development affecting testing aspects as well as performance optimisations.

4. **Technology Limitations:** The consistent evolution of AI technologies means that some components may become outdated. As well as this some technologies selected for this project took ease of implementation over scalability into consideration. This particularly shows in the selection of ChromaDB while easy to implement into a RAG-based system its scalability in its current form is limited. This also affects the SQLite database but this is less of a concern due to it not being an asset or a main feature

Despite these limitations exist within the current project, the methodology provided flexibility while also provided a well structured approach to the development of this application. The final methodology chosen allowed for the full creation of the system which addresses all core objectives outset for this project.

2.7 Summary

This chapter aimed to give insight into the methodology behind StudyBuddy-AI which influenced every aspect of this project from conception to final product. The use of adapted Agile methodology combined with user-centred design and several evaluation strategies created a flexible but well-structured approach to this project. This approach ensures continuous progress and development towards the objectives of this project while allowing for the integration of new requirements and limitations.

The methodology used a strict timeline in order to define key phases and regular milestones for the development process. This supported with the correct tools and technologies selected for this project allowed for the requirements of this project to be met. Ethical considerations guided each aspect of the process ensuring the final product was not only technically well implemented but also responsible.

Chapter 3

Technology Review

3.1 Introduction

This Chapter aims to give a in-depth technology review, Giving insight into the review of literature was key in the development of StudyBuddy-AI. This technology review will aim to detail the understanding behind the technologies used in this project and when the are relevant and assists in the solving of the real world issue StudyBuddy-AI aims to address. The system integrates several technologies including, Retrieval Augment Generation, Embedding models through API calls, vector stores for embeddings, document processing, and Python web frameworks. This review will go in-depth on the key technologies used.

3.2 Retrieval Augmented Generation in Education

Technologies behind the AI generation have expanded rapidly in recent years with Retrieval Augmented Generation being one of these technologies. RAG can be used to ground AI generation in a knowledge base more specific than the general data AI models are trained on. This architecture allows for more accurate and contextual responses grounded in user information. RAG provides an appropriate solution for the demands of users by having improved access to this information. The RAG architecture when implemented with LLMS provides a more accurate solution in information-critical generation.[6]

AI affects my aspect, however, it heavily impacts the work within education. Applications such as the automatic creation of questions are proposed to help teachers create automatic exam questions or a database of questions using LLMs with the RAG architecture. The results of experiments conducted on this application show high accuracy in the generation of these questions.[7] This demonstrates one of many applications the RAG architecture can have in education. While this application applies to teachers there is no reason why this cannot be adapted to meet the needs of students.

3.3 Embedding Models

Embedding models are a essential component within natural language processing systems. This component is particularly relevant within Retrieval Augmented Generation systems. Theses models transform text into numerical vector representations that capture semantic meaning enabling similarity comparison between text.

Research shows that embedding models play a critical role in knowledge-intensive NLP tasks. The model transforms both queries and document content into dense vector representations.[4] This transformation allows systems to conduct similarity searches, a key function in effective retrieval.

Work on Dense Passage Retrieval(DPR) shows how embedding models can encode questions and passages separately while maintaining semantic meaning. This meaning enables systems to identify relevant information even when the exact wording differs between the query and source material.[5] This is valuable in an educational context where students phrase questions differently from processed materials.

The quality of embedding models significantly impacts the quality and performance of RAG systems. Vector embeddings are the backbone of similarity search mechanisms, with models designed to preserve semantic relationships while reducing dimensionality to a manageable level.[8]. Embedding technologies have evolved to balance efficiency with representation quality, creating vectors effectively and capturing the meaning of text while allowing for similarity calculations.

3.4 Vector stores

Vector stores represent a critical part of the Retrieval Augmented generation architecture, providing the structure to efficiently store and retrieve high-dimensional embeddings. Vector stores are specialised databases that store and retrieve embeddings, these are numerical representations of data, and these embeddings make up some of the core functionality behind RAG systems. [9]

Vector stores are classified into many categories based on their design. Lightweight options like ChromaDB, which is implemented in StudyBuddy-AI, are designed for efficiency within an application. Providing an ideal balance between performance and implementation for an educational application[9]

The core functionality of vector stores revolves around similarity search. This feature allows the system to identify relevant information from the vector database using an embedded query by calculating semantic similarity. This calculation utilises cosine similarity, which captures semantic relationships even with differing terminology. [8]

Vector databases also allow for metadata filtering, allowing for searches to be restricted based on specific data subsets, [8] This is particularly important in the StudyBuddy-AI application as it enables the personalised knowledge bases for individual students while maintaining data separation.

3.5 Web Development utilising Python

Python has emerged as a powerful language for web development, this is due to a number of reasons, its readability, extensive libraries, and its versatility across platforms. There are several frameworks that have been developed that utilise python to create web applications suitable for AI integration.

Flask is a widely adopted framework which is known to be lightweight and flexible, this enables applications to be developed quickly while maintaining a simple structure. Its integration capabilities with Python-based AI systems make it valuable for applications that require complex processing backends.[10]

For larger applications, Python frameworks offer features for handling user authentication, database management, and template rendering, these features make them useful in complex web applications with data handling involved. The integration of Python web frameworks with databases provides essential functions for storing and retrieving user data [11]

Web interfaces developed with Python frameworks can be utilised as effective frontends for intense backend processes, offering a seamless connection between AI algorithms and web-based user interfaces. This integration is useful for systems that require natural language processing functions accessible through a web interface. [12]

Streamlit recently gained popularity as a framework specialising in creating data applications with Python. It allows for building interactive web interfaces with minimal frontend coding requirements, while still allowing for easy integration with a backend API.[13]

3.6 Summary

This chapter aimed to provide a detailed technology review describing the research undertaken in order to understand the technologies and systems behind StudyBuddy-AI. The main premise of this review revolved around the main technology behind this application Retrieval Augmented Generation, which offers a way to ground AI outputs in user knowledge-bases particularly benefiting educational applications. The chapter discussed the main technologies implemented in the RAG system and how they're valued in the system.

Chapter 4

System Design

4.1 Introduction

This chapter aims to detail how the system is designed and implemented for StudyBuddy-AI. This will be achieved by focusing on the project objectives and how the architecture behind the application addresses these objectives which were outlined in Chapter 1. The design of this application leverages the technologies discussed in Chapter 3 to create a learning assistant to help students interact with their academic materials efficiently and effectively

The systems core functionality is built around Retrieval Augemented Generation(RAG), which enables context-aware AI responses based on user-uploaded documents. The implementation of a user-friendly web-interface with functional backend API service creates a application focused on balancing functionality with performance and security.

4.2 Client-Sever Model

StudyBuddy-AI follows a client-server architecture with a distinct separation between frontend and backend components. This separation enables independent development, deployment and scaling while still aiming to provide a joined user experience. The Architecture consists of five components:

1. **Frontend Application:** This is a Streamlit-based web interface that provides access to all system functionality
2. **Backend API:** This is a Flask-based RESTful API which aims to handle document processing, authentication, and AI interactions
3. **Vector Database:** This is a ChromaDB instance storing document embeddings for efficient retrieval in a disk-persisted local database.

4. **Authentication Service:** Firebase Authentication integrated with JSON Web Token(JWT) handling ensures for secure access control

5. **External AI API:** Connection to OpenAI's API is used for both the embedding of document chunks as well as the generation of responses to queries.

This modular design aims at allowing each component to be optimised for its specific requirement within the system while maintaining clear communication between each.

4.3 API Endpoints

The backend API contains several groups of endpoints each handling specific functionality:

1. **Authentication Endpoints**
   - `/auth/login`: Authenticates users and issues access and refresh tokens
   - `/auth/register`: Creates new user accounts
   - `/auth/check-session`: Verifies active user sessions

2. **Document Management endpoints**
   - `/file/upload`: Processes and stores uploaded PDF documents
   - `/file/extract`: Retrieves a list of users documents
   - `/file/delete`: Removes all document chunks associated with a specific from the ChromaDB database

3. **Query Endpoints**
   - `/ask/query`: Handles general questions about documents
   - `/ask/summary`: Generates customised summaries of document contents
   - `/ask/mcq`: Creates multiple choice questions based on documents

4. **Feedback Endpoints**
   - `/feedback/submit`: Collect user feedback on system functions and store it in an SQLite database

All protected endpoints require a JWT authentication, this ensures that users can only access their own data and system functions if authenticate.

4.4 Data Flow

the typical data flow for the core operations of the system is as follows:

4.5 Document upload flow:

1. User uploads PDF through Streamlit User Interface
2. Frontend sends the file to the backend API
3. Backend extracts text from the document and a unique document ID is generated
4. Text is chunked and converted to embeddings
5. Chunks and embeddings are stored in ChromaDB with user metadata
6. Success confirmation is returned to the frontend

4.6 Query Flow:

1. User submits a query through the Streamlit User Interface
2. Frontend sends the query to the appropriate backend endpoint
3. Backend converts the query into an embedding
4. Vector similarity search retrieves relevant document chunks
5. Retrieved chunks are assembled into context
6. Prompt is sent to OpenAI API for completion
7. Generated response is returned to the fronted
8. Frontend displays the response to the user

This architecture ensures each component handles its specific responsibility. creating an efficient system.

4.7 RAG Implementation

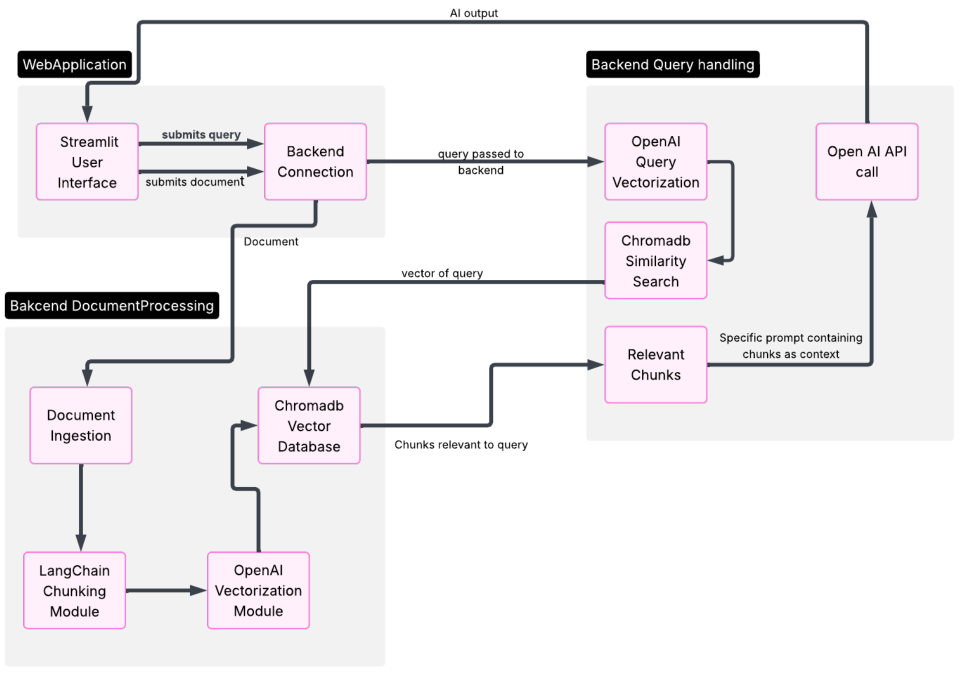
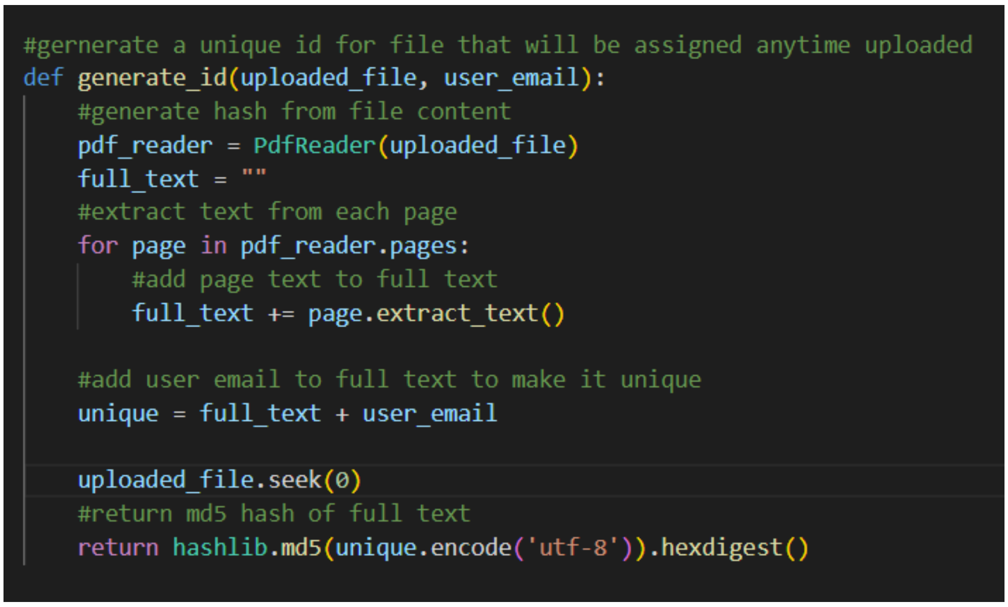


Figure 4.1: System Architecture.

As seen in figure 4.1 there are two key steps in the RAG system implemented into this application. The document ingestion pipeline is responsible for processing PDFs into a format suitable for retrieval while the Query/Retrieval pipeline is responsible for the efficient processing of query and retrieval of relevant chunks, it is also responsible for prompting the AI as well as returning its output.

4.8 Document Ingestion Pipeline

The document ingestion pipeline forms the foundation for StudyBuddy-AI's RAG implementation. This pipeline aims to transform raw PDF documents into searchable vector representations.



```python
#gernerate a unique id for file that will be assigned anytime uploaded
def generate_id(uploaded_file, user_email):
    #generate hash from file content
    pdf_reader = PdfReader(uploaded_file)
    full_text = ""
    #extract text from each page
    for page in pdf_reader.pages:
        #add page text to full text
        full_text += page.extract_text()

    #add user email to full text to make it unique
    unique = full_text + user_email

    uploaded_file.seek(0)
    #return md5 hash of full text
    return hashlib.md5(unique.encode('utf-8')).hexdigest()
```

Figure 4.2: Generating ID.

As shown in 4.2 one of the first steps in the pipeline is generating a unique identifier based on the document's contents and the user's email address. This approach ensures that the same document uploaded by different users is treated as distinct, maintaining data separation while preventing duplicate uploads by the same user.

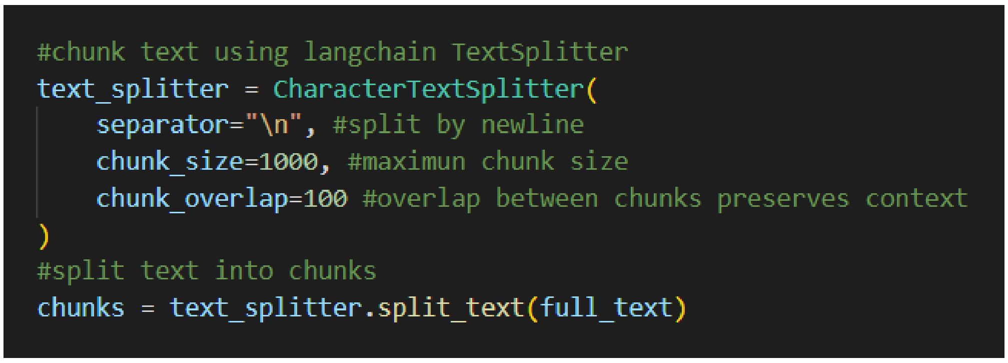
After ID generation the system extracts the text for the PDF using pyPDF, which handles the PDF format to extract plain text content. This extraction process focuses on content rather than formatting, This prioritises semantic content rather than formatting this information is needed for retrieval. The extracted content then undergoes chunking breaking down the content into manageable segments that can be effectively processed and retrieved.

Once chunked, the text segments are converted to vector embeddings using OpenAI's embedding model. These embeddings capture the semantic meaning behind each chunk, enabling similarity-based retrieval later. Each chunk is stored with metadata including document ID, filename, chunk index, and user email. This metadata enables filtering during retrieval and provides context about the chunks.

The document ingestion pipeline represents a critical component of the RAG system, as the quality of the stored embeddings directly impact the relevance of retrieved information during similarity searches. The implementation of this system focuses on processing efficiency with extraction quality, ensuring documents are processed quickly while maintaining the semantic meaning behind the content.

4.9 Chunking Strategy

The chunking strategy used in StudyBuddy-AI's system has a significant impact on the system's ability to retrieve relevant chunks in response to similarity searches based on user queries. After employing various approaches to this component, a character-based chunking method with overlap was implemented using LangChain's TextSplitter.



```
#chunk text using langchain TextSplitter
text_splitter = CharacterTextSplitter(
    separator="\n", #split by newline
    chunk_size=1000, #maximun chunk size
    chunk_overlap=100 #overlap between chunks preserves context
)
#split text into chunks
chunks = text_splitter.split_text(full_text)
```

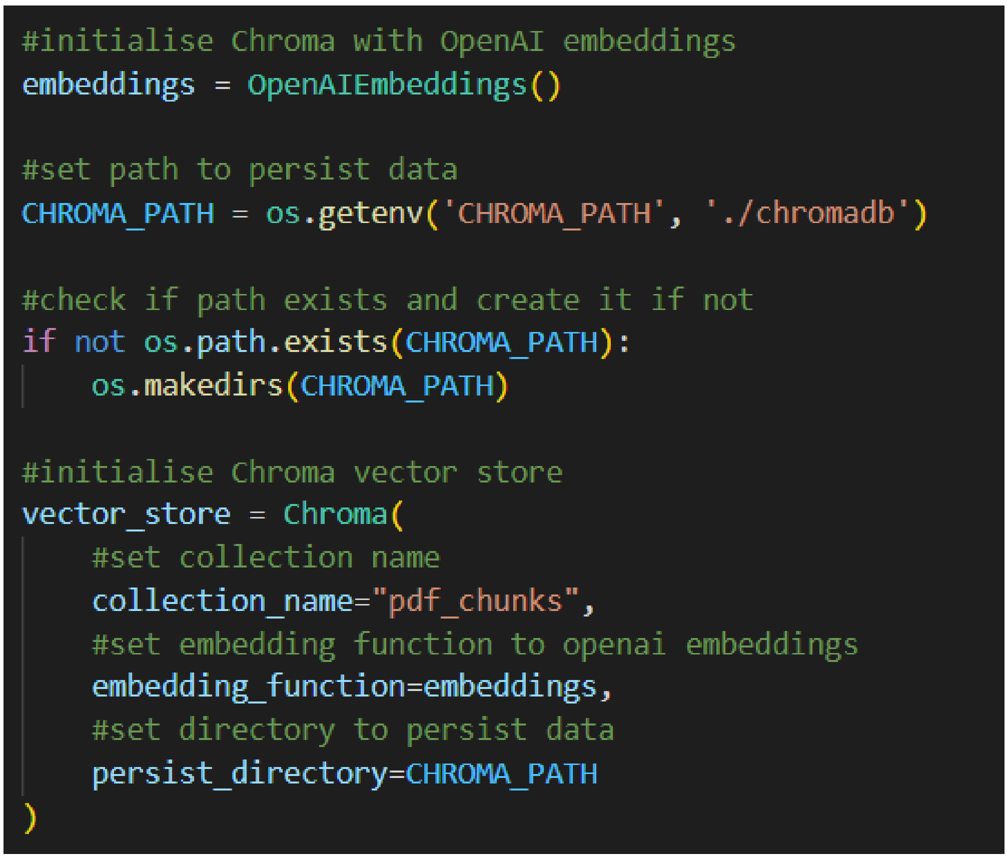
Figure 4.3: Chunking system.

The approach shown in 4.3 displays how text is divided into segments of 1000 characters, with a 100-character overlap with connected chunks. The chunk size was selected as it ensures sufficient context for understanding while remaining within an optimal range for vector similarity models. The overlap in chunk contents ensures that semantic units can span between chunks and are not lost, this maintains context.

The implementation uses newline characters as splitting points, this is in a attempt to preserve a natural document structure. This helps maintain the connection of paragraphs and sections, which improves readability of each chunk. If a document lacks clear structural breaks the system relies on the character based splitting which ensures consistent chunk sizes.

An additional improvement to the chunking process is the inclusion of the filename of the document in each chunks content. this enables users to query for information from specific documents by mentioning the filename within the query. This adds a simple search mechanism without requiring complex filtering.

4.10 Embeddings and Storage

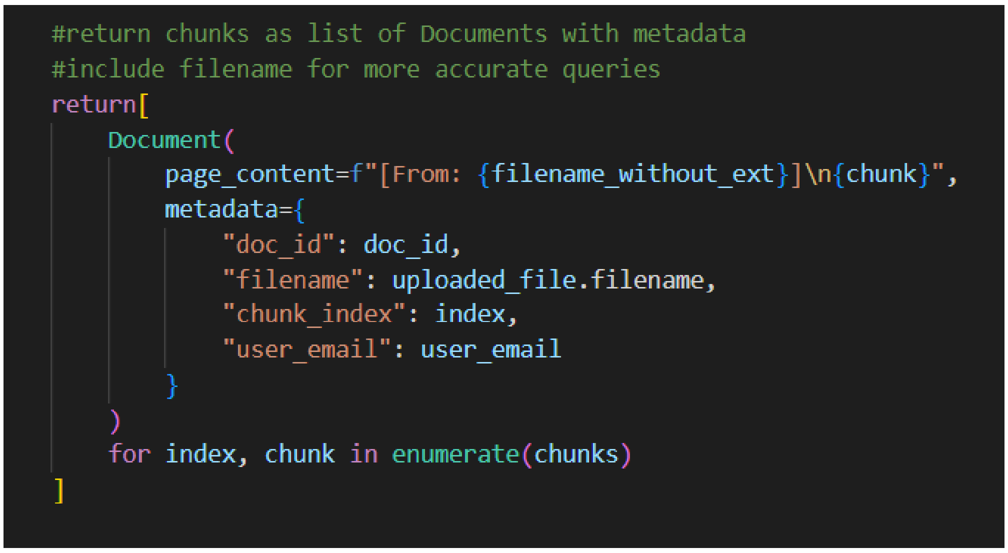


```python
#initialise Chroma with OpenAI embeddings
embeddings = OpenAIEmbeddings()

#set path to persist data
CHROMA_PATH = os.getenv('CHROMA_PATH', './chromadb')

#check if path exists and create it if not
if not os.path.exists(CHROMA_PATH):
    os.makedirs(CHROMA_PATH)

#initialise Chroma vector store
vector_store = Chroma(
    #set collection name
    collection_name="pdf_chunks",
    #set embedding function to openai embeddings
    embedding_function=embeddings,
    #set directory to persist data
    persist_directory=CHROMA_PATH
)
```

Figure 4.4: Embeddings and Storage Process.

After chunking is complete, each text segment must be converted to a vector embedding for similarity search. StudyBuddy-AI's system uses OpenAI's embedding model for this transformation, which gathers the vector representations of each chunk and capture the semantic meaning of the text. These embeddings are critical for the retrieval mechanism, as it enables the system to find document chunks that are similar to user queries, even when they don't share the same keywords.

As seen in figure 4.4 the system uses ChromaDB as the vector database, providing optimal storage and retrieval for embeddings, ChromaDB offers efficient similarity search capabilities while also maintaining a connection between embeddings and the source text and metadata. The database is configured with a persistent storage directory, which ensures that document embeddings remain available across system restarts.



```
#return chunks as list of Documents with metadata
#include filename for more accurate queries
return[
    Document(
        page_content=f"[From: {filename_without_ext}]\n{chunk}",
        metadata={
            "doc_id": doc_id,
            "filename": uploaded_file.filename,
            "chunk_index": index,
            "user_email": user_email
        }
    )
    for index, chunk in enumerate(chunks)
]
```

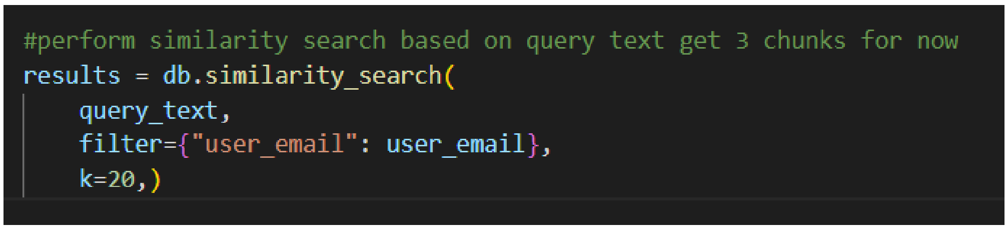
Figure 4.5: Architecture of each Chunk.

As seen in figure 4.5 each document is stored as a structured object containing both the text content and metadata. This metadata is used for filtering during the retrieval of chunks and providing context about the chunk's origins. This metadata structure is essential in maintaining user data separation and enabling personalised retrieval based on users' documents.

The embedding generation and storage components are key aspects with in the document ingestion pipeline and the retrieval mechanism, transforming processed text into a format that allows for efficient semantic search while maintaining the contextual information needed for relevant responses.

4.11 Retrieval Mechanism

When a user submits a query, StudyBuddy-AI's system must identify the most relevant document chunks within that user's knowledge base to create an informed context for the response. This retrieval process is implemented using a similarity search mechanism that compares the embedded query against stored document embeddings, finding the closest match in the vector storage.



```python
#perform similarity search based on query text get 3 chunks for now
results = db.similarity_search(
    query_text,
    filter={"user_email": user_email},
    k=20,)
```

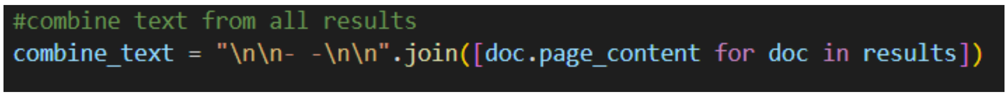
Figure 4.6: Similarity Search Mechanism.

This system applies user-specific filtering during retrieval, this ensures that only the current user's documents are considered. This filtering maintains strict data separation between users, this also improves retrieval efficiency by reducing the search area. Figure 4.6 shows how the retrieval process returns the top 20 most relevant document chunks, this balances the need for a strong context with the constraints of prompt size limits in the AI model.

Vector similarity is calculated using a cosine similarity. This calculation effectively captures semantic relationships in the embeddings, identifying document chunks with similar meaning even when the words or terminology may differ. The combination of the similarity search with the metadata filtering allows fro accurate and personalised retrieval which forms the foundation of the RAG system.

This retrieval process represents the 'R' in 'RAG' as it aims to provide relevant context that is grounded in the user documents allowing the models response to also be grounded in the documents rather that just pretrained generic data. The quality of this retrieval process directly impacts the quality and accuracy of the systems outputs this makes in a critical component of this architecture.

4.12 Context Assembly



```python
#combine text from all results
combine_text = "\n\n- -\n\n".join([doc.page_content for doc in results])
```

Figure 4.7: The Process of Combining Chunks.

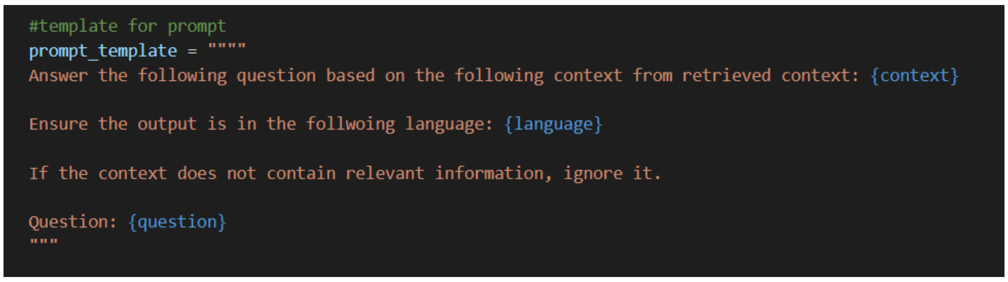
After retrieving relevant document chunks, the system must assemble them into a functional context to be passed to the AI model. The context assembly process in StudyBuddy-AI's system combines the retrieved chunks into a single text, using a delimiter to maintain separation between each chunk. This is a simple yet effective approach, preserving the structure of the original chunks while providing clear boundaries that help the model understand the different sources.

This assembled context maintains the original formatting of each chunk, this aims to enhance readability and preserve the semantic structure. The chunks are ordered in terms of relevance, this aims to ensure the most important information appears first in the context. This order helps the AI model prioritise the most important and relevant information related to the query when generating a response, particularly when the combined context is near the token limit of the model.

The context assembly component serves as the connection between retrieval and generation, transforming the collection of document chunks into a well-structured input to guide the AI model. The quality of the assembly stage impacts the model's ability to understand retrieved information and generate responses that are accurate to the context provided.

4.13 Prompt Engineering

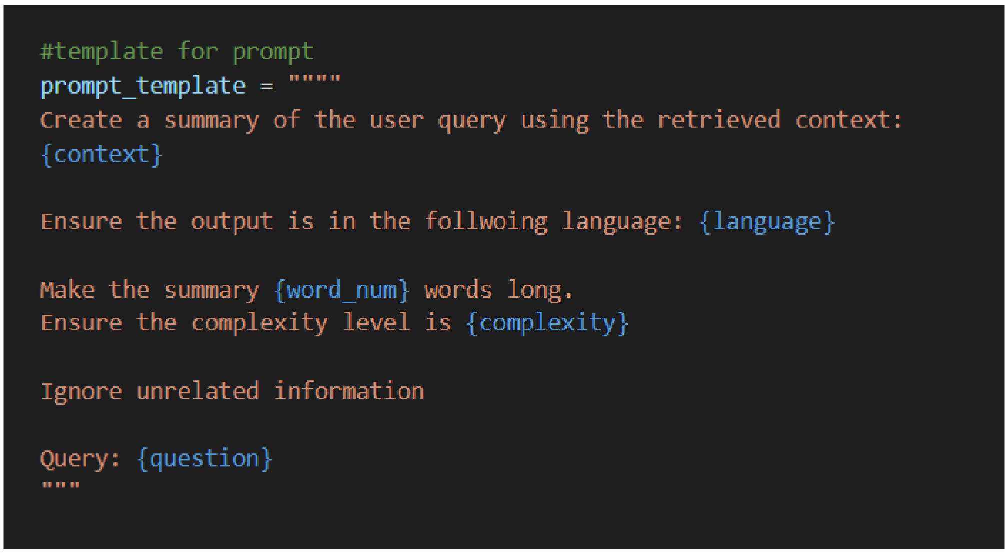
The final step in the RAG process is constructing effective prompts for the AI model based on the user query and the retrieved context. StudyBuddy-AI implements three different prompt templates for different types of tasks, each designed to prompt specific formats of responses from the model.



```
#template for prompt
prompt_template = """
Answer the following question based on the following context from retrieved context: {context}

Ensure the output is in the follwoing language: {language}

If the context does not contain relevant information, ignore it.

Question: {question}
"""
```

Figure 4.8: Prompt Template for Q&A Function.

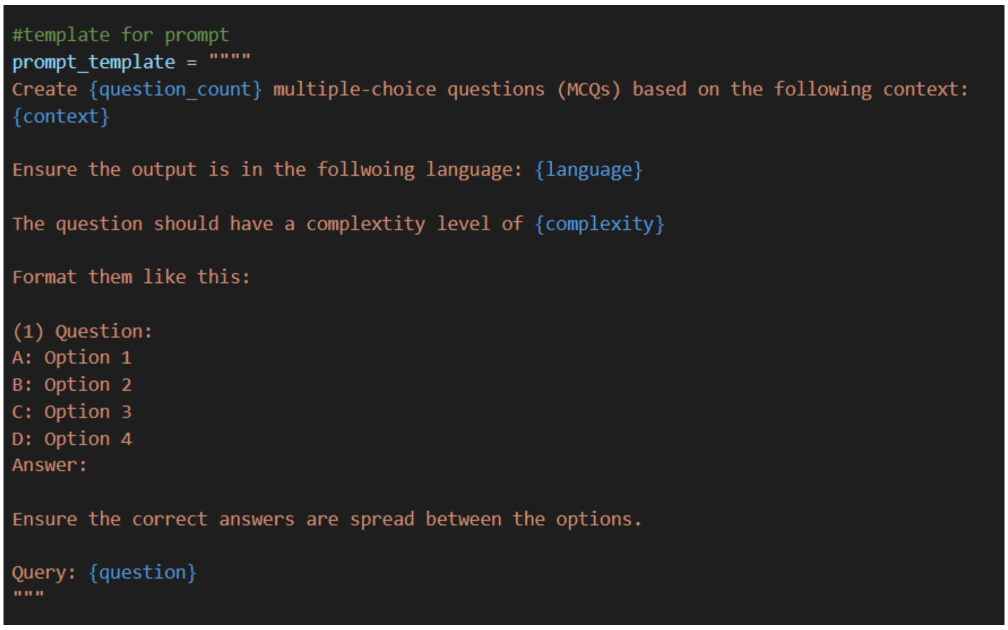
For general question answering, the prompt instructs the model to answer the question based on the provided context while also handling cases where the context may not contain relevant information.



```
#template for prompt
prompt_template = """
Create a summary of the user query using the retrieved context:
{context}

Ensure the output is in the follwoing language: {language}

Make the summary {word_num} words long.
Ensure the complexity level is {complexity}

Ignore unrelated information

Query: {question}
"""
```

Figure 4.9: Prompt Template for Summary Function.

For summary generation the prompt adds parameters for word count and complexity level, all parameters input by the user and passed to the backend through HTTP requests. This enables a combustibility of the summary output to meet the users need. The prompt also includes the context retrieved in order to ground the summary in the users document knowledge base.



```
#template for prompt
prompt_template = """
Create {question_count} multiple-choice questions (MCQs) based on the following context:
{context}

Ensure the output is in the follwoing language: {language}

The question should have a complextity level of {complexity}

Format them like this:

(1) Question:
A: Option 1
B: Option 2
C: Option 3
D: Option 4
Answer:

Ensure the correct answers are spread between the options.

Query: {question}
"""
```

Figure 4.10: Prompt Template for MCQ Function.

The MCQ generation prompt provides a detailed formatting instruction, this is in order to ensure the format remains consistent in order for the front end to extract each separate aspect of the MCQ generated. The prompt also includes parameters regarding question count and complexity to ensure the output from this generation meets the users needs.

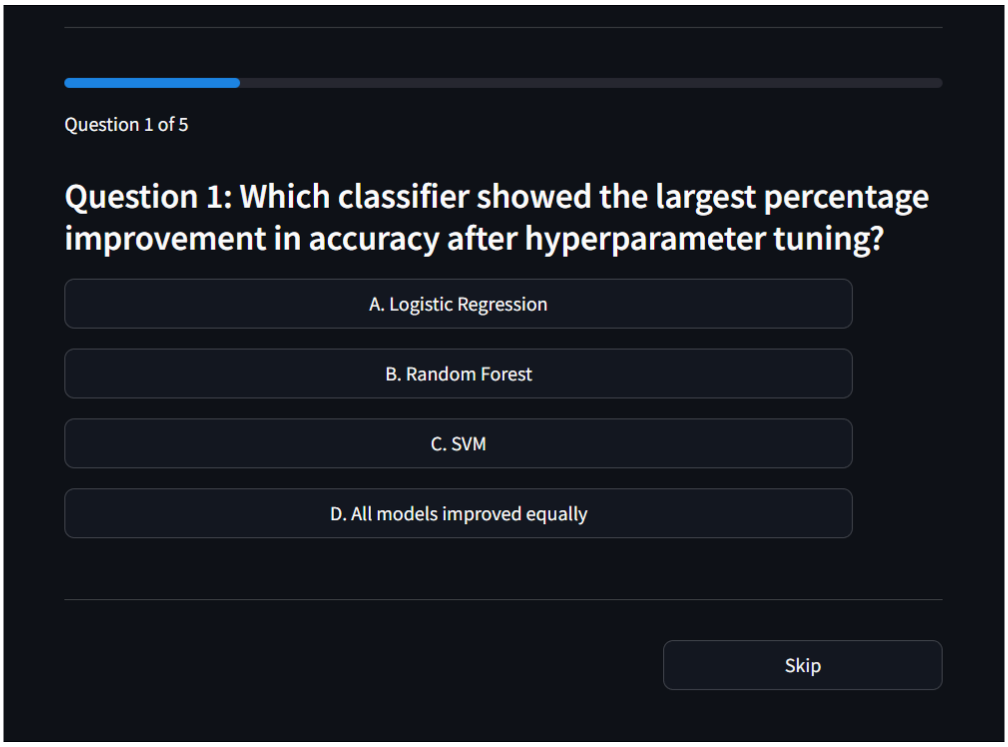
All prompts include language specification, this ensures responses are formatted in the users selected language regardless of the language of the context provided by the retrieved chunks. This multilingual function enables the accessibility for non-native language speakers, allowing them to interact with their documents in their preferred language, meeting on of the key requirements of this project.

The prompt engineering behind this system shows the interaction between the RAG system and the AI model being utilised. These prompts aim to provide a well structured guide needed to transform the retrieved context into a focused and useful response. The quality of these prompts significantly impacts the accuracy and format of the models outputs, making them another critical aspect of the system design.

4.14 Frontend Design: Streamlit Interface

The frontend of StudyBuddy-AI is built using Streamlit, this was chosen for its ability to rapidly develop interactive web applications using Python. The interface is organised into distinct pages for different functionality, creating a clear and task-orientated user experience. The Main page provides a chat UI for general document queries. Dedicated pages for handling summaries and an interactive MCQ complete the AI-powered tools. A document upload and document management page allows for the management of documents the tools will utilise during use. A feedback page allows users to fill out a form giving their opinions on the quality of key aspects of the application. Finally, the register and login pages allow for the authentication process giving users secure access to the application.

Streamlits components form the building blocks of the user interface, with chat elements providing conversational interaction with the AI, select boxes offering custom parameters to be entered for the generation of summaries and MCQ generation and file uploaders handling the process of getting the document to the ingestion pipeline on the backend.


Question 1 of 5

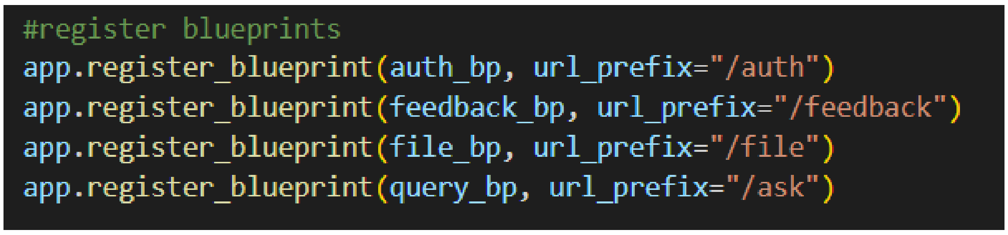
Question 1: Which classifier showed the largest percentage improvement in accuracy after hyperparameter tuning?

Figure 4.11: Interactive MCQ UI.

The MCQ interface is one of the most complex aspects of the user interface, as well as the system overall. Presenting an interactive MCQ with sequential questions and answer selection and immediate feedback including more complete feedback at the end. The interface tracks the user's progress through question sets, calculates scores and provides detailed feedback. This utilises Streamlits session state heavily to track all aspects of the MCQ. This shows Streamlit can be used to create interactive and complex interfaces beyond simple UI components.

4.15 Backend: Flask API Structure

The backend API utilised by StudyBuddy-AI is implemented using Flask. This technology was chosen due to its lightweight nature and flexibility. The API structure is based on a blueprint pattern, with distinct modules for different domains: authentication, document management, queries and feedback. The modular approach implemented improves maintainability by grouping related endpoints and related functions. This creates clear boundaries between the different aspects of the system.



```
#register blueprints
app.register_blueprint(auth_bp, url_prefix="/auth")
app.register_blueprint(feedback_bp, url_prefix="/feedback")
app.register_blueprint(file_bp, url_prefix="/file")
app.register_blueprint(query_bp, url_prefix="/ask")
```

Figure 4.12: Registering the blueprints.

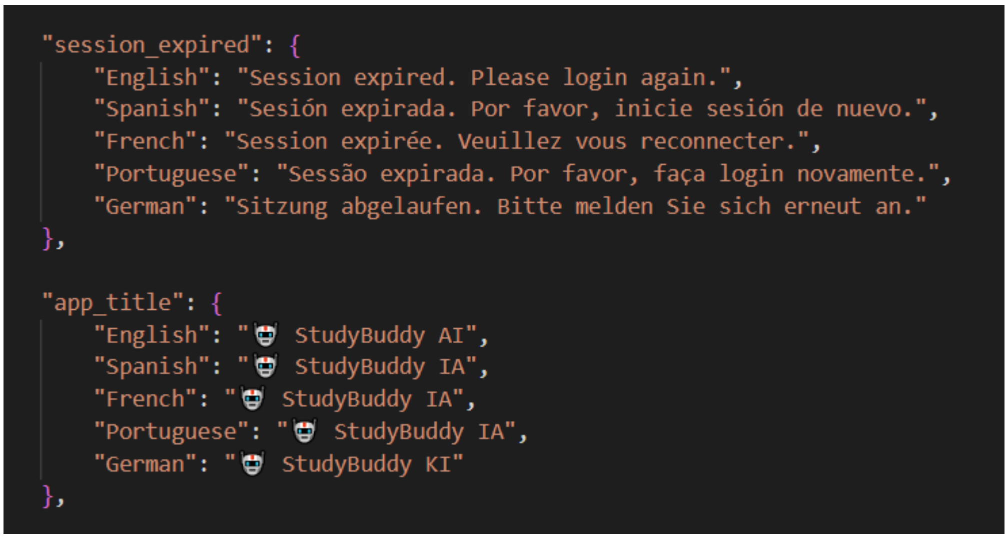
Each blueprint defines routes with appropriate HTTP methods and authentication requirements. Authentication routes handle registration, login and session management, while file routes handle document uploads and processing. Query routes process different user interactions and queries while also handling the retrieval process, and the feedback routes collect user input on the system.

The API follows RESTful principles, using appropriate HTTP methods for different operations and returning JSON responses with meaningful status codes. This aims for a consistent approach to the intergration with the frontend.

The API Structure is the backbone of the backend system, defining how the frontend interacts with services and databases. The organisation and patterns in the structure allows for the current functionality within the project.

4.16 Multilingual Implementation

An important feature of this application, to meet the requirements outset, is the multilingual support offered by the UI. This feature not only offers simple interface translation, but, it also allows users to interact with their documents in their desired language.



```
"session_expired": {
    "English": "Session expired. Please login again.",
    "Spanish": "Sesión expirada. Por favor, inicie sesión de nuevo.",
    "French": "Session expirée. Veuillez vous reconnecter.",
    "Portuguese": "Sessão expirada. Por favor, faça login novamente.",
    "German": "Sitzung abgelaufen. Bitte melden Sie sich erneut an."
},

"app_title": {
    "English": "🤖 StudyBuddy AI",
    "Spanish": "🤖 StudyBuddy IA",
    "French": "🤖 StudyBuddy IA",
    "Portuguese": "🤖 StudyBuddy IA",
    "German": "🤖 StudyBuddy KI"
},
```

Figure 4.13: Dictionary of Translation for UI.

The multilingual implementation uses a translation dictionary as seen in figure 4.13. This dictionary covers all UI elements across five languages: English, Spanish, French, Portuguese, and German. This dictionary maps interface components to language-specific text.

The language selection is implemented into the sidebar allowing users to change to they're preferred language will navigating between pages. When a new language is selected the UI immediately updates reflecting the new language selected.

The language selection process not only affects the UI, but this language selection is also based on queries to the backend which in turn passes this parameter to the AI within its prompt. This feature ensures the user experience is consistent between UI interactions and AI interactions. This allows users to interact with they're documents in they're preferred language, even if the documents provided do not reflect this.

4.17 Summary

This chapter details the design and implementation of key features within StudyBuddyAI's codebase, focusing on the core function of the system architecture, RAG implementation, frontend design, and backend services. The design shows how the RAG concept is applied to create a useful educational tool that aims to help students interact with their academic materials.

The system architecture separates the frontend and backend components, which enables independent development of each aspect of the application. The RAG implementation follows a pipeline approach, processing documents through extraction, chunking, embedding generation, and storage which enables retrieval and response generation. The frontend provides a multilingual interface with many components for different interactions. The backend offers a secure and efficient service catering to all the needs of the application.

Chapter 5

System Evaluation

5.1 Introduction

This chapter aims to evaluate the quality of the system behind StudyBuddy-AI against the objectives and requirements outset of Chapter 1. The evaluation process uses performance metrics with user feedback to determine how effectively the system addresses the needs of students interacting with academic materials. Examining the key aspects of the system including document processing, query responses, summary generation and MCQ generation and its interactive nature.

5.2 Evaluation Methodology

The evaluation process used mixed methods taking into account both system performance with user feedback to conclude. Ten students (6 university students, and 4 secondary school students) from a mix of educational backgrounds were given the opportunity to use the deployed version of StudyBuddy-AI with their own academic materials over a 4 day period. Participants completed a simple but informal report rating the system features on a 5-point scale.

Technical performance was evaluated manually due to time constraints. 50 PDFs of varying sizes for document processing, 100 predefined queries for response evaluation, 30 documents for summary testing and 25 MCQ generations of varying parameters. The multilingual feature was covered by user testing with 4 of the students selected either studying in non-native language speaking institutions or students studying languages in their course.

5.3 Document Processing Evaluation

The document processing pipeline demonstrated decent performance across different document sizes, with an average time of 19 seconds processing times, ranging from 5.7 seconds for smaller documents(1-10 pages) to 37.8 seconds for large documents(31-50 pages). The testing of this feature was done on a locally run backend and frontend, this is due to the limited resources available to the deployed backend, resulting in limited capabilities when handling large documents with many failing to process due to memory and CPU constraints.

Document Processing Test		
Document Size	Average Processing Time	Deviation
Small (1-10 pages)	5.7 seconds	1.2 seconds
Medium (11-30 pages)	14.3 seconds	3.5 seconds
Large(31 -50 pages)	37.8 seconds	8.9 seconds

Table 5.1: Table showing results of document processing tests.

The pyPDF extraction showed good accuracy for standard formats with around 90 percent of content preserved, however it showed limitations with complex layouts and tables. The chunking strategy proved affective with 80 percent of retrieved chunks containing context related to the query.

User feedback was generally positive, with many appreciating the straight forward upload process and the separate document management page. However in terms of scoring the average rating given by the users was 3.8/5 this is one of the lower scores, but for a basic feature this is to be expected users feedback showed the feature met expectations but did not excel.

Some User Feedback

"The document upload feature worked and was straightforward to operate. The document page is straightforward to operate it shows my documents and allows me to delete them. While these are things I need the documents page could be improved to allow for improved management of documents instead of just deleting them."

"I found this feature simple to use but felt somewhat basic. While I was able to upload my documents, I wish there was a more organised option for viewing and handling them, maybe folders for different subjects?"

5.4 Query Response Evaluation

The query function in StudyBuddy-AI was evaluated on accuracy and response time. These aspects were assessed with 100 predefined queries related to documents in the knowledge base. The outputs were then compared to these documents to calculate their accuracy and relevance. While on simpler concepts the system performed well with the query showing high relevance and context to the documents the more complex the queries became the system began to struggle often not fully grounding its outputs in the document context it was provided. This however did not seem to affect the user testing with reports showing satisfaction with the quality of the responses receiving an average rating of 4.0/5 rating from the user testing.

Query Response Time Test		
Question Complexity	Average Response Time	Deviation
Simple(1-2 concepts)	3.7 seconds	0.8 seconds
Moderate(3-4 concepts)	5.9 seconds	1.2 seconds
Complex(5 concepts)	8.2 seconds	1.9 seconds

Table 5.2: Table showing results of Query Response Time tests.

Response times were within a limit deemed acceptable with an overall average of 6 seconds. User feedback was positive with participants identifying the document specific nature of the response as a invaluable asset to they're studies.

Some User Feedback

"I love how it actually uses my lecture notes to answer! It's like it knows what I'm studying instead of just giving me random stuff from the internet. Sometimes it does miss some things I know are in my notes, but overall it's much better than scrolling through a document or using CTRL F to hopefully find a keyword!"

"The best thing about this is its assistant aspect being about to ask questions and have an output just given to me makes life so much easier, however, my course covers a lot of math equations and tables which the AI didn't really understand properly, but working with textbooks for some of my other modules it seemed to work just fine."

5.5 Summary Evaluation

Summary response times and quality were assessed through manual testing and user testing respectively. Testing from 30 summary tasks related to the knowledge base. The system showed very little differing response times depending on complexity and word count of the output. The response time average showed about a 9 second response time a acceptable wait time.

Summary Response Time Test		
Word count	Average Response Time	Deviation
100 words	9,2 seconds	0.3 seconds
300 words	8.4 seconds	0.6 seconds
500 words	9.8 seconds	0.8 seconds

Table 5.3: Table showing results of Summary Response Time tests.

User evaluation of the summary feature show a strong satisfaction with this feature, with a average user rating of 4.2/5. Many user noted the summary feature as a feature that significantly improved they're interaction with documents in they're entirety rather the specific prompts. With many noting the customisablity of the summary could be adapted to fit larger requirements in the future.

Some User Feedback

"The summary feature has been really helpful and probably the feature I use most, I don't use the question so much as to get summaries of a specific concept instead I tend to put a document title into the query textbox and get a summary for the entire document I find this is the most efficient way to get the information I need out of my documents."

"The summary feature was handy to use for small simple concepts, but my course has few small simple concepts If I were to use this feature I'd need larger summaries than just 500 words. I feel like 500 words can only cover the basics of some concepts rather than getting in-depth knowledge, this made the complexity level setting feel useless at times."

5.6 MCQ Evaluation

MCQ generation response times were assessed through 25 questions measuring the generation response times under different parameters. The system demonstrated that these large outputs can take large sums of time to generate from the backend but also process into the interactive MCQ format on the frontend. With a average response time of around 30 seconds this is a large wait time and beyond was it deemed a decent response time however given the complexity of the MCQ feature this is to be expected.

Summary Response Time Test		
Question Number	Average Response Time	Deviation
5 Questions	23.4 seconds	2.3 seconds
10 Questions	29.8 seconds	7.1 seconds
15 Questions	33.3 seconds	4.5 seconds
20 Questions	38.6 seconds	5.6 seconds

Table 5.4: Table showing results of Summary Response Time tests.

The MCQ feature received the highest user rating among all the features tested. Users gave the feature an average rating of 4.7/5 with participants valuing this feature extremely highly especially its assessment capabilities on the frontend. Many users reported the feature to be the most useful and have the most improvement on their studies and overall learning of materials.

Some User Feedback

"The quiz questions were a game changer for me as a college student. Many of my modules include regular MCQs for homework or for exams. I've always struggled to find a way to really test myself in an MCQ kind of way as there is no real format like the one in StudyBuddy that aligns so well with the college format for them."

"The MCQs are a great feature and so unique. It really helps with understanding just how well I studied and took in the information in my study notes. I used it by putting in study notes and generating a quiz with that filename it helped me understand what parts of my notes I really understood over others"

5.7 Multilingual Support Evaluation

Multilingual support was evaluated across 2 of available languages, while not ideal finding students available to test the application in all of the five languages available proved challenging. Due to the UI adapting to the selected language immediately and the language selection having minimal effect on the response time of outputs there was no need for response time testing for this feature.

Users were informed to focus on the accuracy of the translation while using the feature. Participants reported that the overall UI translation was as expected and had very few errors within them. However, users did report that the AI outputs could often have errors in translation which affect some more than others. This is reflected in the average rating being the lowest of all at 3.7/5

Some User Feedback

"The Text and buttons on the application translated fairly well into Spanish and I didn't have any issues with figuring out how to use the app in Spanish. The AI text was slightly off at times but was close enough that I could understand what it was trying to say. Overall not an issue."

"Using the app in my language was great because sometimes I have issues with reading difficult documents in English, but sometimes the responses were incorrect and I needed to ask again to get one that made sense."

5.8 Overall System Evaluation

Overall user satisfaction with StudyBuddy-AI was high, with an average rating of 4.1/5 across features. The highest rated feature was the MCQ generation with 4.7/5, this feature was extremely popular and on of the must used features. The feedback reveal som key benefits to the RAG system, improved access to information, a reduction in language barrier, and improved study efficiency. Users particularly valued the personalised nature of the system, with responses based on they're specific information rather then generic data.

Despite the positive feedback limitations were identified:

1. Document format restrictions: Support is limited to PDF files only
2. Complex Content Handling: Reduced performance with complex layout types eg. tables
3. Session persistence: No state preservation across browser refreshes or closing.
4. Language Limitations: Quality differences between English and the other languages provided, particularly in the AI-generated responses.
5. Processing Time: Longer processing time for large documents and difficult queries or large outputs.
6. Limited resources: Resources available to the backend are limited resulting in some large tasks being quit prematurely by backend workers causing errors

These limitations show the direction for future development. It highlights area where improvements can be made, significantly enhancing the overall quality of the user experience.

5.9 Summary

The evaluation process demonstrates that StudyBuddy-AI addresses most of the core objectives set out for this product. The implementation of the Retrieval Augmented Generation System provides students with a valuable tool which allows them to interact with they're documents in a new and more efficient way while also making it more enjoyable.

Chapter 6

Conclusion

6.1 Introduction

This dissertation has aimed to showcase StudyBuddy-AI, a Retrieval Augmented Generation(RAG) system which has been designed in order to assist students in their interactions with academic materials with documents. To begin with, the process of identifying challenges faced by students was undertaken in order to understand the needs of students. This process showcased the struggles students face in information overload and extracting information from academic materials. The project has followed a structure that involved extensive research into these points as well as research into solutions. The system design and implementation of the system were all influenced by this. Now at the conclusion key findings can be shown, limitations discussed and suggestions made for the future direction and development of StudyBuddy-AI.

6.2 Key Findings: Technical Structure

The implementation of the RAG technology in this system effectively created a personalised assistant education tool. The system successfully processed uploaded documents, extracted and chunked the documents contents, generates embeddings and retrieves relevant information for the response using the queries. This shows the success in implementing a RAG system into a educational tool for students.

The performance testing shown in Chapter 5 confirms this approach is also acceptable in processing efficiently with the user feedback showing the outputs generated are up to a good standard in quality. While not instant the response times are within an acceptable response time even for the processing of larger documents or more complex AI outputs. The current system could absolutely be fine-tuned to improve many aspects of the product and ensure a much higher quality but for a student project and a proof of concept in my opinion this system is to an acceptable standard and displays not only the implementation of the system but also that this proof of concept has value in the real world.

6.3 Educational Value

The value of StudyBuddy-AI showcases an array of benefits to individuals in education. User testing with students from different education levels and subjects show that the system can improve they're interactions with they're documents, making these interaction more efficient and improving the overall experience and process of studying. This is achieved by allowing students to quickly locate information within they're documents. This reduction in search time allows student to focus on the application of studying this information.

The document specific structure of the systems responses provides outputs well grounded in and relevant to the students academic materials. This gives a advantage over generic information sources, as the grounding in personal documents creates a personal and trustworthy learning experience.

The multilingual support in the application helps address language barriers in education. This enables non-native speakers to interact more effectively with their academic materials. While this feature is not perfect this feature expands accessibility and supports students when interacting with their documents. In these ways, this feature shows its value in the proof of concept and how valuable it could be in a refined manner within a complete and well-refined application.

The overall high ratings from the user testing demonstrates the value of the concept behind StudyBuddy-AI and its value to students. It can be used in many aspect of education and improve them extensively. StudyBuddy-AI was designed with students in mind and in my opinion the finished concept displays this in every aspect.

6.4 Implementation review

The development process behind the StudyBuddy-AI system has demonstrated the effective implementation of a RAG system. The character based chunking strategy provides a effective approach to extracting information and maintaining its semantic value while allowing for efficient retrieval. Simple aspects like including document filenames in chunks and using metadata for filtering enhanced the systems ability to affectively output the information required by the user. This created a simple yet affective way for users to interact with they're documents.

Prompt templates for different tasks allow outputs to be of a high quality while also meeting the users requirements and parameters. This was achieved with the well structure engineering of prompt templates with clear instructions and parameters. This aspect is a critical aspect of the system yet simple. The quality of the output relies on the information given.

These are the insights gathered while implementing this RAG system. It demonstrates the technical considerations made during this process and the impacts they have on user experience. The RAG system was enjoyable to implement and learn about. Applying it to education was a challenge but the outcome is extremely rewarding as it shows its value as a good asset to students.

6.4.1 Technical Limitations

The system currently only supports PDF files, limiting how useful the application can be for students with materials in other formats such as Word, PowerPoint or HTML. This restriction limits the type of academic materials that can be processed and may exclude important resources that students commonly use.

Performance becomes worse when processing documents with complex layouts, tables or mathematical notation. This limitation especially affects STEM students, whose academic materials may contain large sums of content like equations and visual elements.

Authentication Security Concerns The initial implementation of the JWT token system consisted of storing tokens in HTTP-only cookies as is best practice. Streamlit is a server-side application that renders UI elements server-side. This caused an issue with the JWT token being set to the instance of Streamlit server side rather than to an instance on the browser this caused the JWT token to be accessible to everyone using the application which logged each instance into the same account. This was an unforeseen issue with how Streamlit behaves in a deployed environment with JWT. This issue had to be rectified with an unconventional solution, JWT were set to pass through headers and stored in Streamlits session-state. This allowed each session to obtain its own JWT however this goes against best practice and is not secure. However, this application aims to provide proof of concept so although this oversight caused issues in my eyes it does not affect the proof that this application shows benefits for students. This issue will have to be remedied during future development in order to progress StudyBuddy-AI to a fully functional application ready for production.

Session Management: Maintaining sessions without JWT in the HTTP-only cookies proved to be an issue as JWT stored in the session state was lost during reloads. This is due to Streamlits session state not being persistent, wiping the JWT for its storage on reloads or closure of the browser. This resulted in sessions not persisting and users being required to log into the application after every reload. This wiping of session state also results in the loss of Q&A conversations, summaries and MCQs generated as these are stored in Streamlits session storage. This displays a failure to meet one of the objectives of this application and will require future development in order to remedy this issue.

6.4.2 Functional limitations

Document management capabilities are limited, this may make it difficult for users to organise larger collections of documents effectively. As students upload more materials, the list of documents may become increasingly challenging to navigate and reduce the efficiency of document management.

The system lacks support for study groups or group learning or sharing of knowledge bases. This may limit the applications benefits in group educational work or group learning. Education often requires collaboration and the absence of this feature may lower how affective StudyBuddy-AI is for some students. As the application aims to meet all students needs this will require consideration in future development.

6.5 Future Work

The limitations outlined above show the need for future development to focus on the improvement of the authentication features within the application. The implementation of JWT in deployment demonstrates one of the failures of this application when in the design process of this application the incompatibility with Streamlit should have been identified and worked around. Future developments must implement a more secure system in order to ensure the security of the application for its users. This will entail new research and a redesign of the authentication system behind the application in order to achieve this.

Session persistence is also a area that requires attention in the future. Implementing this was one of the main reasoning for implementing JWT but unfortunately this feature while operational in the development environment did not transfer to the deployed environment well due to the oversight of Streamlits deployed behavior. This aspect of any application is important as it ensures good flow within the application and ensures that information stored within the application is not lost.

Local embedding models could both enhance privacy and reduce the reliance on external AI models through API calls. While this was considered during the design of this application as well as consideration into self hosting a language model to create responses to queries. This was not researched thoroughly or implemented due to resource and time constraints.

6.6 Project Contributions

This project demonstrates the application of Retrieval Augment Generation in education and the impact it can have. This system provides a blueprint for document grounded AI interactions for all learning environments. The evaluation offers insights into the effectiveness of AI based document tools like StudyBuddy-AI in the education world and how it can benefit students.

6.7 Conclusion

StudyBuddy-AI was developed in order to improve the studies of students from every aspect of education, while also demonstrating the effectiveness of Retrieval Augmented generation technology when applied to real-world issues and made accessible to the public. Despite many limitations and shortcomings in the application, in my eyes, it indeed confirms this concept is viable and excels at addressing these issues while also showing the value of a system like StudyBuddy-AI's.

The project contributes to the exploration of AI in education, a hot topic in today's world. This application aims to show that AI does indeed have a place in education and in assisting students while not affecting the educational system negatively. Grounding AI responses in user-specific information offers a promising way for students to engage with their academic materials in a way generic AI does not offer.

6.8 Personal Statement

Through this work, I've aimed to challenge myself as a software engineer as much as possible choosing to implement technologies and concepts into this project that I was completely unfamiliar with before beginning this process. I'm glad to say that this has greatly improved my understanding of what it takes to be a software engineer and has improved my preparation for entering the working world. This was an incredible challenge that I'm incredibly grateful to have partaken in, and accomplished. On a more personal level, StudyBuddy-AI has been a large aspect of my life for the last seven months, it has taught me a lot about software engineering but also about myself as a software engineer and how I handle pressure and challenges. StudyBuddy-AI is one of my best creations in the software world if not the best and I'm incredibly proud of it even with its failures and shortcomings. As a proof of concept, it does indeed prove what I set out to do both on a technical level and a personal one. Thank you for taking the time to read this dissertation.

Appendix A

Acknowledgments

There has been lot of people who supported me not just through the process of this final year project, but through the entirety of my 4 years studying for my bachelors in Software Development and I would like to take a moment to thank them.

Firstly I would like to thank my family and partner for their unwavering, unconditional and continuous love and support throughout the last four years but especially during my fourth year studies. It would have simply been impossible to complete this process with out your support. You have enabled me to take on this challenge and accomplish the goals I outset when entering this course, Thank you so so much. I would also like to give thanks to the Department of Science of ATU Galway. Their work in providing use with all the materials and personnel required to not only study Software engineering but become software engineers. They're work has been invaluable to me and guided me a bright future. Thank you so much. I would like to thank the lecturers involved in my years in the software development course. Their knowledge and guidance over the last four years have been extremely valuable in getting me to a place in my studies where I can take on a challenge like this dissertation, final year project, and fourth year in full. Thank you for your contributions to this course. I would like to thank John French who coordinated the Final Year Project and gave extremely helpful lectures for this process. Finally, I would also like to give a big thanks to Joseph Corr my supervisor for this project, his guidance at key parts of development helped improve StudyBuddy-AI greatly.

Appendix B

Important Links

The following links will direct you to the important aspects of this project including the Github Organisation containing both the frontend and backend repositories for StudyBuddy-AI. A link has also been provided to access the deployed version of StudyBuddy-AI. Finally a link to a unlisted Youtube video which contains a demonstration of the working application has also been provided.

StudyBuddy-AI Github Organisation: https://github.com/AppliedProject2024
Deployed version of StudyBuddy-AI: https://studybuddyai-frontend.onrender.com
Screencast demonstration of StudyBuddy-AI:
https://www.youtube.com/watch?v=vXqaH4HBn64

Bibliography

[1] D. Mizrachi, A. M. Salaz, S. Kurbanoglu, J. Boustany, and o. b. o. t. A. R. Group, "Academic reading format preferences and behaviors among university students worldwide: A comparative survey analysis," *PLOS ONE*, vol. 13, no. 5, p. e0197444, May 2018, publisher: Public Library of Science. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0197444

[2] Y. Zhang and Y. Mi, "Another Look at the Language Difficulties of International Students," *Journal of Studies in International Education*, vol. 14, no. 4, pp. 371–388, Sep. 2010, publisher: SAGE Publications Inc. [Online]. Available: https://doi.org/10.1177/1028315309336031

[3] M. J. Eppler, , and J. Mengis, "The Concept of Information Overload: A Review of Literature from Organization Science, Accounting, Marketing, MIS, and Related Disciplines," *The Information Society*, vol. 20, no. 5, pp. 325–344, Nov. 2004, publisher: Routledge _eprint: https://doi.org/10.1080/01972240490507974. [Online]. Available: https://doi.org/10.1080/01972240490507974

[4] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 9459–9474. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[5] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense Passage Retrieval for Open-Domain Question Answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 6769–6781. [Online]. Available: https://aclanthology.org/2020.emnlp-main.550/

[6] B. Tural, Z. Örpek, and Z. Destan, “Retrieval-Augmented Generation (RAG) and LLM Integration,” in *2024 8th International Symposium on Innovative Approaches in Smart Technologies (ISAS)*, Dec. 2024, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/10845308

[7] N. Aisyah, R. Sarno, A. F. Septiyanto, A. T. Haryono, and D. Sunaryono, “Generating Question Using Retrieval Augmented Generation with Large Language Model: Comparative Study,” in *2024 Beyond Technology Summit on Informatics International Conference (BTS-I2C)*, Dec. 2024, pp. 468–473. [Online]. Available: https://ieeexplore.ieee.org/document/10941961

[8] S. Kukreja, T. Kumar, V. Bharate, A. Purohit, A. Dasgupta, and D. Guha, “Vector Databases and Vector Embeddings-Review,” in *2023 International Workshop on Artificial Intelligence and Image Processing (IWAIIP)*, Dec. 2023, pp. 231–236. [Online]. Available: https://ieeexplore.ieee.org/document/10462847

[9] R. Shan, “A Deep Dive into Vector Stores: Classifying the Backbone of Retrieval-Augmented Generation,” in *2024 IEEE International Conference on Big Data (BigData)*, Dec. 2024, pp. 8831–8833, iSSN: 2573-2978. [Online]. Available: https://ieeexplore.ieee.org/document/10825992

[10] “Integration of Flask and Python on The Face Recognition Based Attendance System | IEEE Conference Publication | IEEE Xplore.” [Online]. Available: https://ieeexplore.ieee.org/document/9590122

[11] V. Singh, Y. Rohith, B. Prakash, and U. Kumari, “ChatBot using Python Flask,” in *2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS)*, May 2023, pp. 1182–1185, iSSN: 2768-5330. [Online]. Available: https://ieeexplore.ieee.org/document/10142484

[12] T. B.M, R. H.V, R. S, R. Salam, and M. Pai, “TextVerse: A Streamlit Web Application for Advanced Analysis of PDF and Image Files with and without Language Models,” in *2024 Asia Pacific Conference on Innovation in Technology (APCIT)*, Jul. 2024, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/10673559

[13] L.-W. Huang, P.-H. Cheng, and L.-W. Chen, “Web-based Board Game for Learning Python,” in *2021 IEEE World Conference on Engineering Education (EDUNINE)*, Mar. 2021, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9429144