Recommendation System for Learning Materials Dissertation

Bishal Roy

BSc (Hons.) Computer Science
Honours Dissertation

Heriot-Watt University

School of Mathematical and Computer Sciences

Supervised by Dr Bental, Diana

March 2025

DECLARATION

I, Bishal Roy, confirm that this work submitted for assessment is my own and is ex- pressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: Bishal Roy

Date: 12/02/2025

ABSTRACT

This dissertation explores the development of an AI-powered recommendation system that is implemented into a note-taking application for students with the goal of improving learning efficiency by providing personalised educational content based on notes taken by the students/ The study tackles common problems that students have, such as information overload and ineffectiveness in finding relevant study materials, especially in self-directed and online learning settings.

In order to overcome these challenges, the project gathers important ideas and keywords from students' notes using Natural Language Processing (NLP) API provided by Google. Using cosine similarity and TF-IDF vectorization methods, a content-based recommendation approach is used to match extracted keywords with relevant learning materials from datasets Learning Resources Database and the Amazon Book Dataset.

The design of the system sets a high value on usability, effectiveness and accessibility. It offers a simple user experience through the front-end interface made with HTML, CSS, JavaScript, and Flask. Strong data management, password hashing, and efficient recommendation delivery are all provided by the back-end, which is driven by MySQL and Python Flask.

A recommendation precision rate of 56.3% was shown during user testing evaluation, showing the system's potential effectiveness while also flagging out areas that require more development, such as increasing the dataset to provide more relevant options and information. Most user comments were favourable, especially about the recommendations' applicability, usefulness, and ease of access to study resources. However, recommendations for additional improvements in recommendation diversity and user interface design were noted, as well as providing more than 3 recommendations at a time.

By providing the potential and benefits of implementing NLP-based recommendation systems into note-taking applications, this study contributes to AI-driven education technology and encourages a more individualised, and effective learning environment.

ACKNOWLEDGEMENTS

I thank this and that person for their help. My great thanks go out to my supervisor, Dr Bental, Diana, for all their help, encouragement, and support during this project. Their knowledge and guidance have greatly influenced my research and academic journey.

I am also deeply thankful to Heriot-Watt University for providing me with the opportunity to conduct this research. Their resources and support have enabled me to pursue my academic goals and contribute to the advancement of knowledge in this field.

Table of Contents

Contents

List of Figures

List of Tables

List of Code Listings

1 INTRODUCTION

Technology has evolved from an addition to a crucial aspect in modern education. Digital learning tools are essential for improving accessibility and educational quality in higher education, especially as personalised learning paths become more popular. Artificial Intelligence (AI) in education has shown great potential among the many technological developments, especially in providing real-time feedback and individualised learning experiences. AI-powered solutions can offer personalised recommendations, helping students' efficient use of a huge amount of educational material.

With the adoption of digital note-taking applications presents an opportunity to improve learning with AI-powered recommendations. Traditional study methods often overwhelm students with large online resources, making it hard to find relevant materials (Ross, 2023). Many faces information overload, ineffective study habits, and a lack of guidance in self-directed learning. AI-driven recommendation systems address this by analysing student notes and suggesting personalised resources (Er-Rafyg, 2023). This dissertation explores a note-taking app using Natural Language Processing (NLP) to extract key concepts and provide study recommendations, improving efficiency, and engagement, and self-directed learning.

1.1 Motivation

The ineffectiveness of traditional study methods, where students often find it difficult to sort and choose the most relevant learning resources, is what inspired this study. Overload of information and ineffective study sessions might result from the overwhelm of internet educational resources (Ross, 2023). The goal of this research is to increase learning effectiveness and engagement through the implementation of AI-powered recommendations into a note-taking application. Students will be able to focus on important subjects and spend less time looking for resources due to the recommended system, which will guarantee that they receive relevant and timely study material recommendations. Additionally, this technology can serve as an intelligent study assistant, directing students to helpful and personalised learning resources in digital and remote learning situations when there is little opportunity for direct teacher support.

1.2 Aim and Objectives

This dissertation aims to develop a recommendation system that improves student learning by providing personalised educational material suggestions based on their notes. Specifically, this project looks to:

- Develop an AI-driven note-taking application with NLP to extract key concepts from student notes and determine study focus areas.
- Design a personalised recommendation system that matches extracted concepts with datasets like the Amazon Book Dataset (Dubey, 2024) and Learning Resources Dataset (Patil, 2023).
- Evaluating learning efficiency by measuring how AI-powered recommendations reduce search time for relevant materials.
- Assessing student engagement and satisfaction through feedback on usability, accuracy, and impact on learning.

1.3 Contributions

This dissertation contributes to the field of AI-driven education technology by:

(1) Developing an AI-powered note-taking system that implements Natural Language Processing (NLP) for content-based recommendation generation.
(2) Introducing a personalized recommendation model that matches extracted keywords from student notes with relevant educational resources.
(3) Evaluating the impact of AI-based recommendations on learning efficiency and student engagement, providing insights into the effectiveness of AI-driven study aids.

1.4 Organisation

The dissertation is structured as follows:

- Chapter 2 – Background: Reviews AI in education, recommendation systems, and ethical challenges
- Chapter 3 – Requirements: Outlines system requirements, datasets, and technologies.
- Chapter 4 – Design: Details system architecture and design.
- Chapter 5 – Implementation: Covers implementation, including NLP integration and the developed recommendation model.
- Chapter 6 – Experiment: Presents testing, user feedback, and evaluation.
- Chapter 7 – Conclusion: Summarizes the findings, contributions and limitations of this research and suggests future improvements.

2 BACKGROUND

AI-driven recommendation systems in education are discussed in this background chapter, with a focus on their development, effects on learning, and potential for educational personalisation. To demonstrate how recommendation systems can be used to create personalised learning routes, it analyses multiple recommendation system types, their approaches, and e-learning difficulties.

2.1 Development of AI in Educational Technologies

Artificial Intelligence (AI)

From the paper (Ifenthaler, 2024), using complex algorithms for artificial intelligence (AI) significantly changed how data inputs are converted into outputs. By analysing and adjusting to their surroundings in order to accomplish particular goals, AI systems demonstrate intelligent behaviour. These systems are made to do more than simply carry out basic tasks; by using machine learning techniques to learn and adapt, they may independently approach complicated goals.

AI plays an important role in human-computer interaction by categorizing activities into three key areas:

- Human Activities: Tasks where humans excel over AI, such as leadership, creativity, and evaluation.
- Machine Activities: Tasks like prediction, repetitive processing, and process modification, where machines outperform humans.
- Human-Machine Alliances: Collaborative relationships where humans create, teach, and manage AI to enhance human capabilities, transforming computers from tools into partners in fields like education by enabling real-time data processing.

By highlighting the development towards AI as a collaborative partner in education, this new way of thinking improves the potential for better learning experiences. AI integration in educational environments is becoming increasingly unique demonstrating how these technologies can enhance human potential and introduce different approaches to productivity and customisation in learning.

Artificial intelligence in Education

In this paper (Ifenthaler, 2024), it discusses the use of data and algorithms in higher education has grown rapidly since the early 2010s, supporting activities including curriculum development, learning, teaching, assessments, and service improvement. AI has

been successfully implemented in several areas:

- Modelling Student Data: AI models improve earlier prevention techniques by predicting academic progress.
- Intelligent Tutoring and Adaptative Systems: These systems adjust to each learner's unique needs by offering personalised learning materials, support, and feedback.
- Automated Examination Systems: AI improves the evaluation process by automating the assessment of learning accomplishments.
- Educational Decision Support: AI technologies help teachers make educated decisions on their lesson plans and student engagement.

Despite these developments, there are still obstacles to the general use of AI in education. There are few studies that focus on the long-term effects of AI, and organisational and technological limitations prevent actual system-wide use from reaching its full potential. By lowering costs, improving student achievement, increasing access to education, and cutting down on study time, artificial intelligence (AI) has the potential to completely transform educational environments. There are several levels of classification for AI applications in education:

- Institutional Processes: In addition to supporting counselling for student's services, AI speeds up the admissions and application processes.
- Learning and Teaching Enhancement: AI predicts educational outcomes, facilitates social learning, automates assessments, and suggests learning steps.

As well as from this paper (Moturu, 2023), to represent a variety of learning outcomes, the changing educational environment places a significant value on application-oriented learning and ongoing assessment modifications. By providing the transition from standard lecture-based methods to more dynamic, technology-driven, and learner-centred approaches, artificial intelligence (AI) technologies are essential in transforming educational environments. The idea of education has expanded beyond traditional classroom settings to consist of continuous learning made possible by massive open online courses (MOOCs), which is representative of a larger movement towards open and widely available education.

Impact on Learning

As mentioned in this paper (Moturu, 2023), by emphasising individualised and engaging learning experiences that rival one-on-one instruction, artificial intelligence (AI) dramatically improves the educational landscape. AI plays a critical role in providing effective and personalised education, enabling students to get similar learning objectives in shorter amounts of time. Beyond fundamental field-level knowledge, AI in education has

grown to include aspects related to creative thinking, critical analysis, and teamwork. These changes provide chances to take advantage of educational frameworks that support pupil choice and personalisation.

AI has an impact on educational systems through analysing different learning behaviours, including exploring and step-based learning and enabling autonomous and peer-based learning environments. AI has been modified by many organisations and applications to fit their own legal and educational environments. Content distribution makes students a priority on learning processes in modern education, which encourages collaboration, independence, and internal motivation. An increasing amount of research examines how motivation and independent study improve satisfaction in AI-assisted environments, even though field-level learning still contributes to most research. Developing confidence is a primary objective of AI in education, which will probably enhance learning outcomes overall.

2.2 Recommendation Systems in E-Learning

Recommendation systems assist users to quickly explore huge amounts of information by analysing user behaviour and preferences to deliver relevant suggestions. By suggesting specific materials, improving engagement, and enhancing results, they personalise learning in the classroom. These systems use data to facilitate focused and efficient learning (Khanal, 2020).

Real-World Applications

Machines that provide recommendations for online learning are essential for managing the complicated structure of learning pathways and educational resources. By offering individualised learning paths and educational materials suited to the requirements of every student, they improve the educational process. To help students choose courses based on their past performance and preferences, for example, systems created for platforms such as Moodle allow a successful educational experience (Aberbach, 2022).

Online learning platforms have shown success with hybrid recommendation algorithms. To suggest the most appropriate educational resources, they combine data from multiple sources, such as characteristics of customers, behavioural information, and specific material preferences. These systems assist in managing the huge amount of online educational content as well as personalizing learning experiences (Er-Rafyg, 2023).

Datasets for Recommender Systems

Large datasets that enable customised predictions are essential for effective recommendation systems, as they allow researchers to test and refine algorithms, enhancing recommendation accuracy. Big data is needed to do data mining and machine learning, as it originates from online systems that track student progress, store work, and monitor resource usage. This data supports machine learning and data mining, facilitating research and algorithm optimization. Public datasets drive advancements in recommendation systems, with competitions like the Netflix Prize and ACM RecSys Challenges providing real-world data for testing. Web 2.0 technologies further enable access to extensive open-source repositories, allowing for diverse research and fine-tuning of algorithms (Reddy, 2020).

In this project, two datasets are used to personalise learning recommendations: Amazon Book Dataset (Dubey,2024) and Learning Resources Database (Patil,2023).

Benefits of Recommendation Systems in E-Learning

Some benefits in Aberbach, 2022 and Er-Rafyg, 2023 mentioned are:

- Improved Resource Accessibility**:** Personalised systems improve student engagement and efficiency by recommending materials that align with learning progress.
- Learning Content: AI-driven recommendations adapt educational resources to a learner's knowledge level, improving comprehension through tailored assessments.
- Research Assistance: These systems support academic research by suggesting relevant materials based on the researcher's field of study.
- Diverse Learning Strategies: Recommendation systems address learner attributes such as objectives, prior knowledge, and preferences to ensure resources meet student needs.

2.3 Types of Recommendations Approaches

Before discussing various types of recommendation systems, it's important to address a common challenge known as the cold start problem. This issue emerges when there is insufficient data from new users, resulting in inaccurate recommendations and potentially leading to user disengagement (Bobadilla, 2012). In the paper Khanal, 2020 and Moturu, 2023, it discusses the different types of recommendation systems:

Collaborative Filtering

This recommendation method that analyses the preferences and actions of similar users to generate suggestions. By analysing interactions between items and users in a hierarchical structure, these systems predict human preferences and recommend relevant materials. This approach employs algorithms that can either generate an ideal list of recommended items or predict user ratings. According to Er-Rafyg, 2023, collaborative filtering has some advantages, such as being based on community judgements, allowing it to support various item types, and not requiring domain knowledge, making it flexible across different types of items. However, it struggles with the cold-start issue, where new items with no prior reviews struggle to be recommended effectively, and recommendations from smaller or highly specialised communities may be biased and not representative of broader user preferences.

The image below, fig 1, from (Oubalahcen, 2024) shows that in the collaborative filtering framework in a recommendation system. It shows a rating table where users (u_1 to u_n) have given rating to various items (i_1 to i_n). The highlighted cell represents the rating of a specific item by the active user, which is not yet known and needs to be predicted. Using existing ratings, the system predicts this unknown rating (highlighted in red), which it then uses to recommend other items that this user might like. The process involves using the known preferences of similar users to predict and recommend items effectively.

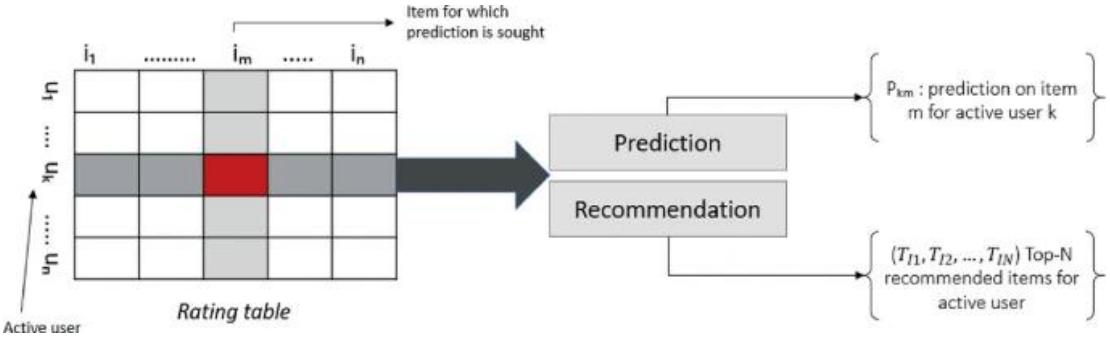


Fig. 1. Collaborative Filtering framework in a recommendation system of a rating table. (Oubalahcen, 2024)

Content-Based Filtering

This system focuses on item characteristics and user preferences to provide personalised recommendations. These recommendation systems analyse item content, extract features, and compare them to user profiles based on past preferences. By doing so,

recommendations closely align with user's previous choices, ensuring relevance and customisation (Er-Rafyg, 2023). One of the key advantages of this system is that it does not rely on data from other users, effectively avoiding cold-start issues, and it tailors recommendations to individual preferences and offers explanations based on item characteristics while suggesting new or less popular items. However, there are disadvantages, such as the need for manual feature input for items that cannot be analysed by machines, making the process time-consuming. Also, it is limited to explicitly defined item properties, restricting recommendation flexibility.

Hybrid

Hybrid recommendation systems combine the approaches of content-based and collaborative systems to overcome the drawbacks of each, including over-specialization and the cold-start issue. By integrating multiple recommendation methods, hybrid systems improve the diversity and accuracy of recommendations. According to Er-Rafyg, 2023, these systems offer advantages such as the ability to combine the strengths of different methods to improve prediction accuracy and overcome individual limitations, but it is a lot more complex to design and implement as it requires important planning and resources.

The image below, fig 2, from (Oubalahcen, 2024) shows that in the four hybrid recommender systems combining collaborative filtering (CF) and content-based filtering (CBF). Panel (a) shows CF and CBF working separately with merged results. Panel (b) has CD improved by CBF, Pabel (c) shows CBF improved by CF. Pabel (d) integrates both CF and CBF fully, combining their outputs for recommendations. By using both strategies, every combination aims for the greatest suggestion accuracy.

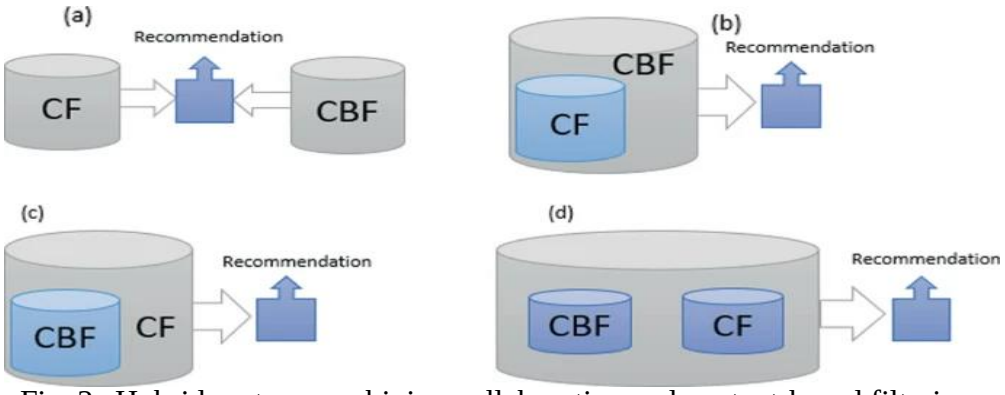


Fig. 2. Hybrid system combining collaborative and content-based filtering (Oubalahcen, 2024)

Demographic-Based Recommendations

These systems categorise users based on demographic information such as location, gender, and age. The recommendations are then personalised according to the preferences of users within the same demographic group. As noted by Er-Rafyg, 2023, one key advantage of this approach is that it does not require a user's evaluation history, making it easy to implement quickly. However, it requires demographic information, which raises privacy concerns, and it faces cold-start issues.

Knowledge-Based Recommendations

These systems use logical methods to connect user preferences with item specifications, making recommendations by matching item features to user demands. These systems are effective when exact user requirements are important. According to Er-Rafyg, 2023, a key advantage of this approach is that it emphasizes knowledge sources that collaborative and content-based filtering may not fully utilize while also avoiding the cold-start issue. However, it has disadvantages, including restrictions in getting the information, as knowledge engineers must create formal representations, and the high complexity in design and implementation, which makes it less favourable.

Memory-Based and Model-Based

Memory-based methods use past behaviours to suggest products or user based on similarity measurements. Model-based methods use item properties to determine similarities, allowing for real-time recommendations with lower computing demands.

This project implements a content-based recommendation system, which analyses the text content of student notes to recommend relevant learning materials. The system does not rely on collaborative filtering, indicating that it overlooks the preferences of other users. Instead, it uses TF-IDF vectorization to extract important terms from notes and compares them with a dataset of learning materials using cosine similarity. Additionally, no demographic information is used in the recommendation process, as it ensures that recommendations are entirely on the content of students' notes rather than user attributes. Also, this system follows a model-based approach, as it applies pre-trained vectorization techniques to create recommendations rather than relying on past interactions stored in memory.

Collaborative filtering was note used because it requires massive amounts user interaction data, which this system cannot provide because it prioritises individual study patterns over group preferences. A hybrid approach was also not implemented since integrating multiple recommendation techniques would add complexity without

significantly improving performance for text-based recommendations.

2.4 Challenges in Implementing Recommendation Systems

Some challenges described in Aberbach, 2022 and in Er-Rafyg, 2023 such as:

- Information Overload: Recommendation systems must offer relevant suggestions to help lessen cognitive load because the wide range of content in e-learning might overwhelm students.
- System Design and User Definition: It is difficult to define the "learning user" since successful systems need to take into consideration the objectives, backgrounds, and knowledge levels of students, needing lots of investigation and study.
- Personalisation vs. Generalisation: It's difficult to find a balance between personalised recommendations and general relevance. Systems must be adaptable to accommodate a wide range of user needs while also supporting individual learning objectives.
- Dynamic Learning Objectives and Background: Recommendation systems need to be able to instantly adjust to learners' evolving objectives, situations, and preferences, requiring for constant data collection and data analysis.
- Adapting to Dynamic Learning Objectives: Systems must be updated often to take consideration of students' evolving preferences and goals, leading to the need for ongoing data collecting and analysis.
- Integration and Implementation: Recommendation systems can be demanding on resources to integrate with current educational platforms, and both students and teachers may need to make important adjustments.

Despite their enormous potential, there are currently few actual research investigations evaluating recommendation systems' effects on learning effectiveness. To confirm that the systems are effective in improving learning outcomes and effectively managing educational content, more study is required (Aberbach, 2022).

2.5 Ethical and Social Considerations

Data Privacy and Security

Concerns about data privacy and personal information security have increased because of the use of AI in education. Platforms for educational technology have come under attention for maintaining compliance to digital privacy regulations, particularly those brought (Ifenthaler, 2024).

Significant privacy issues are associated with video chatting and other crucial online

learning tools because, due to the lack of effective security measures, participant data may be misused. To notify students, parents, and teachers about data collection and use, education technology businesses must keep up to national data privacy regulations and guarantee strong encryption for databases, applications, and cloud tools. Data must be protected via strict compliance requirements, encryption, and access restrictions to avoid misuse (Moturu, 2023).

Bias and Fairness

The ART principles—Accountability, Responsibility, Transparency—must be taken into consideration when designing and implementing AI systems in educational environments (Ifenthaler, 2024). This involves identifying the needs and values of stakeholders, making sure that these requirements and values are connected to system features, and assisting in decisions that promote long-lasting implementation.

2.6 Natural Language Processing

Keyword extraction is a core task in Natural Language Processing (NLP) that involves identifying key terms that best represent the meaning of a given text. This is especially useful in educational applications for understanding the core content of student notes and recommending relevant resources. In this project, keyword extraction is implemented using the Google Cloud Natural Language API, which is a powerful pre-trained machine learning service that enables developers to access advanced machine learning models trained by Google for analysing and understanding text data (Shiotsu, 2021).

Some benefits of NLP (Stryker, 2021):

- Automation of Tasks: NLP automates customer support, data entry, and document processing, reducing manual effort. Chatbots handle common queries, and machine translation provides simple communication.
- Data Analysis and Insights: NLP extracts information from customer reviews, social media, and news articles. Sentiment analysis helps companies make better decisions by detecting emotions and assessing customer views.

NLP involves many tasks that help machines process and interpret human text and speech, allowing computers to extract meaning, understand context and improve language-based applications. One of these tasks consist of is, (Stryker, 2021) Named Entity Recognition (NER), which recognizes words or phrases specific entities like people, locations, or organisations (e.g., “London” as a place, “Maria” as a person). Named Entity Recognition (NER) is an NLP technique, which is particularly relevant to this project and was used in this project, identifies and classifies key entities in text, such as names, locations, dates,

and monetary values. It was first developed to boost information extraction, but it has developed with machine learning and is now frequently used to improve text processing in chatbots, search engines, and financial analysis (IBM, 2023).

In this system, Google's NER functionality is used to perform keyword extraction, filtering entities such as organisation, events, locations, and more – all of which are categories supported by the API. These extracted entities are then converted into a keyword list, which is used as input to the TF-IDF vectorizer for generating learning material recommendations based on content similarity. This approach helps reduce noise and makes sure that only relevant keywords are used in the recommendation process.

2.7 TF-IDF Vectorization and Cosine Similarity

TF-IDF Vectorization

Term Frequency-Inverse Document Frequency (TF-IDF) is a widely used technique in document retrieval, and more, to measure the importance of words in a document relative to a collection of documents. Unlike simple word frequency counts, TF-IDF reduces the weight of commonly used word (e.g., "the", "is") while highlighting more informative and context-specific terms, which makes systems like recommendation systems distinguish relevant documents from irrelevant ones (Mondal, 2022). TF-IDF is calculated as:

$$W_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

Where $tf_{i,j}$ is the frequency of word i, in document j, N is the total number of documents, df_i number of documents containing word i and $W_{i,j}$ is the weighted TF-IDF value computed (Renuka, 2021).

In this project, TF-IDF is used to analyze student notes and extract important keywords that represent the main topics of interest. The system then compares these extracted keywords with a dataset of learning materials to generate personalized study recommendations.

When a student writes notes about "neural networks" and "backpropagation," for example, TF-IDF gives these phrases higher weights, making sure the system retrieves learning resources that are relevant to the student's specific field of study rather than general resources. By distinguishing between common and meaningful terms, TF-IDF enhances the accuracy of content matching in contrast to raw word counts, which would give all words the same weight (Wartena, 2010).

Cosine Similarity

Cosine similarity is a metric used to measure how similar two vectors are by calculating the cosine of the angle between them. This approach is particularly effective in high-dimensional spaces, such as text analysis, where data can be limited. It's widely applied in fields like information retrieval, text classification, clustering, and ranking. The cosine similarity between two vectors A and B is computed as:

$$\text{cosine similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

The resulting value ranges from -1 to 1, where 0 denotes orthogonality, -1 denotes opposite orientation, and 1 denotes identical orientation. Because of its effectiveness, cosine similarity is frequently employed in text mining, particularly when working with limited information. It is perfect for applications like text categorisation and information retrieval since it efficiently calculates the similarity between documents by considering the angle between their corresponding vector representations (Li, 2013).

2.8 Similar Systems

Amazon employs item-to-item collaborative filtering recommendation approach, identifying similarities between items based on purchase behaviour rather than direct user comparisons. Instead of matching users to similar customers, Amazon's system compares each purchased or rated item to similar items, combining them to create personalised recommendations. This method improves scalability and efficiency, allowing Amazon to handle millions of customers and products in real-time (Linden, 2003).

Amazon further integrates content-based filtering, analysing product characteristics such as titles, descriptions, and attributes to recommend similar items. Additionally, Natural Language Processing (NLP) techniques extract insights from customer reviews and product descriptions, allowing recommendations based on sentiment analysis and keyword recognition (Bouguezzi, 2024).

The efficiency of Amazon's item-item similarity computation is improved through cosine similarity measure and other optimised similarity metrics. Instead of comparing all possible item pairs, Amazon's approach iterates over customers who purchased the item and creates a similarity table dynamically (Linden, 2003).

Its main principle can be applied to the learning material recommendation system, as both systems focus on item similarity rather than user similarity. The integration of content-based filtering and NLP in Amazon's system aligns closely with the approach of text

analysis used in the learning material recommendation system, which plays an important role in matching study resources to student needs.

2.9 Summary

This chapter studied the role of AI-driven recommendation systems in education, focusing on their development, impact on learning, and ability to personalise education. It addressed the different types of recommendation systems, including collaborative, content-based, and hybrid approaches, evaluating their strengths and limitations. A content-based approach was chosen for this project due to its ability to provide personalised learning material recommendations.

Additionally, the chapter discussed key challenges in AI-Driven learning, such as information overload, personalisation vs generalisation, and ethical concerns like bias and data security. It introduced Natural Language Processing (NLP) and its role in keyword extraction, explaining the various techniques like Named Entity Recognition (NER), and more. The Google Cloud NLP API was highlighted for its ability to extract key topics from students' notes especially using the NER technique, forming the basis of the recommendation system.

The item-to-item collaborative filtering method used by Amazon, which finds similarities between products rather than individuals, was one of the recommendation systems that were covered in this chapter. This idea applies to recommendation systems for learning materials since both depend on item similarity to make useful suggestions. Amazon's use of NLP and content-based filtering closely resembles text analysis techniques employed in learning recommendations, hence improving the system's educational impact.

The TF-IDF Vectorization was presented as a method to convert text into numerical features for machine learning, allowing for better analysis of learning materials. Cosine similarity was also introduced to measure the relevance of learning resources to student notes by comparing text vector representations. The recommendation system is generated by these strategies combined, which ensure that study materials are efficient and scalable.

3 REQUIREMENTS

This section describes the requirements for creating and implementing an AI-driven note-taking software that facilitates learning by suggesting personalised educational materials. It describes key features, data requirements, and user interactions according to the needs of users and technology capabilities, making sure that each requirement is reasonable, and in line with the project's learning objectives.

3.1 Aim and Objectives Breakdown

This section breaks down the project's overall objectives and the suggested actions to complete the objectives are:

1. Create an AI-driven note-taking application: To effectively process and analyse text data, select and integrate an NLP API. Then, create features that allow the system to store and categorise user input in text-like formats.
2. Develop a personalised recommendation system: Integrate a filtering system that compares suggested material with data analysed from students' notes, ensuring that the system provides recommendations in response to user input and new information.
3. Evaluate the system’s effectiveness in improving learning efficiency: Collect feedback for time saved and improved study outcomes.
4. Evaluate the impact of the AI-driven recommendation system on student engagement: Get qualitive and quantitative data on user engagement and satisfaction.

3.2 Use Case Scenarios

Students are assisted by the application, which uses Natural Language Processing (NLP) to analyse their notes and recommends learning resources. The primary functions of collecting notes and creating personalised recommendations, which are crucial for improving learning effectiveness and offering relevant resources, are the focus of these use cases.

Use Case 1

Name:	Typing notes and getting notes analysed by the system
Actor:	Student
Description	A student uses the app to take study notes, which are processed via an NLP API to identify and categorize key concepts.
Preconditions	Student is logged into the application and has started a new note-taking session.
Main Flow	1. The student begins typing notes into the application. 2. The system uses the integrated NLP via API to analyse the text. 3. Key concepts are identified and highlighted in the text. 4. The system categorizes the notes based on the analysed concepts
Postconditions	Notes are saved with tags based on identified concepts.
Alternative Flows	If the system cannot categorize a concept, it tags the note for manual review or prompts the student for more information.

Use Case 2

Name	Receiving personalised learning material recommendations
Actor	Student
Description	After writing or saving a note, the student can request personalised learning material recommendations related to the content of their note.
Preconditions	The student has created and saved a note.
Main Flow	1. The student submits a note for recommendation. 2. The system analyses the note content using the Google NLP API to extract keywords. 3. The system compares these keywords against a dataset of learning materials using TF-IDF and cosine similarity. 4. A ranked list of 3 recommended learning materials is created and provided to the user to use.
Postconditions	The student can access and review the recommended materials.
Alternative Flows	If no relevant materials are found, then it suggests broader related topics or notifies the student about when suitable materials become available.

3.3 Requirements Analysis

Prioritizing requirements enables faster analysis and execution of key components, ensuring timely project completion. This research uses the MoSCoW approach: Must Have (essential requirements), Should Have (important but not critical), and Could Have (non-essential features with minimal impact).

In this section, the main requirements for this research are divided into four groups:

- Non-Functional: Aspects that ensure the system's usability, and performance.
- Functional: Required features and functionalities.
- Data: Required data from datasets for the recommendation system.
- Software: Specific software platforms, tools, and libraries required to build, and maintain the system.

ID	Requirement	Statement	Priority
NFR-1	Security	The system must securely store data like user passwords, by methods such as hashing, to ensure data confidentiality.	MUST
NFR-2	Compliance	The system must comply with the GDPR and ensure the privacy and security of user data.	MUST
NFR-3	Data Integrity	Ensure that data remains accurate, consistent, and in a usable state.	MUST
NFR-4	Usability	The application must be easy and enjoyable, use of the application is ensured by a simple user interface and ease engagement.	MUST
NFR-5	Maintainability	The code and system should be clean, well-documented, and flexible to simplify maintenance and updates.	SHOULD
NFR-6	Accessibility	The application should follow simple accessibility instructions, such using readable fonts.	SHOULD
NFR-7	Portability	The ability to operate across different platforms/systems.	COULD

Table 1. Non-Functional Requirements

ID	Requirement	Statement	Priority
FR-1	User Registration and Management	The application must allow the user to register an account by providing necessary personal and educational details and be able to modify and update their personal and educational information.	MUST
FR-2	Note-Taking	The application must allow users to create a new note within a dedicated interface and must be able to edit their notes as well as being able to save their notes for future use and editing.	MUST
FR-3	Recommendation System	The application must be able to generate learning material recommendations that come from the educational datasets, based on their analysed notes and the user's educational information such as what they study.	MUST
FR-4	Integration with External APIs	System must be integrated with external APIs for natural language processing (NLP) to analyse user's notes.	MUST
FR-5	Security Measures	The system must implement secure login validation processes from unauthorized access to ensure privacy and security.	MUST
FR-6	Search Functionality	The system should provide basic search functionality to locate their saved notes.	SHOULD
FR-8	Export and Import Options	The system could provide export notes to PDF and import documents from external sources.	COULD

Table 2. Functional Requirements

Data Requirements

The system needs access to labelled databases of learning materials to produce recommendations for the users' needs. The recommendation system, which matches with key concepts taken from the users' notes, depends on these databases. These datasets that were used for this project were:

- Amazon Book Dataset (Dubey,2024): This dataset comprises metadata for around 30,000 books organized into 30 distinct categories. It provides detailed information about each book, including the title, author(s), publication date, price, and book image. The recommendation system in this project primarily uses the author and title from the Amazon Book Dataset. They enable the algorithm to use the keywords taken from student notes to determine a book's contextual relevance.
- Learning Resources Database (Patil,2023): This dataset includes a wide variety of

educational materials such as journal articles, conference papers, videos, and online resources. It contains metadata fields like the title, author, record modified, URL link if available, and a description. The materials are categorized by subject areas, with a significant portion derived from reputable sources like the National Library of Medicine. The subjects covered include biomedical sciences, clinical medicine, public health, and other health-related topics, which are highly relevant for learners in these fields especially in the medical field. For the recommender system, the title, description, URL link, and subject area fields are used to generate matches with the key concepts identified in student notes. During evaluation, there were resources that were recommended based on their alignment with students' study topics as shown in appendix L.

To support the recommendation system, both datasets were merged into a single dataset. The following key features were extracted from each: the title of the learning material, the author, the URL (if available), description of the resource (if available), and the subject area (if available). These fields were selected to ensure each resource could be effectively represented and compared using the recommendation algorithm. The combined dataset formed the basis for generating learning material recommendations throughout the project.

Software Requirements

The software requirements for the AI-driven note-taking and recommendation system were selected for their efficiency in data handling, user interface development and supporting personalised learning.

- Backend Development: Python for its extensive libraries (Python Software Foundation, 2019).
- Frontend Development: HTML (MDN Web Docs, 2018), CSS (MDN Web Docs, 2019), JavaScript (MDN Contributors, 2023) for UI design.
- Web Framework: Flask for simplicity in back-end development (Palletsprojects.com, 2024).
- Database Management: MySQL for reliable relation data storage (MySQL, 2019), managed locally through XAMPP, which provides an easy-to-use platform for running Apache and MySQL services during development (Apache Friends, 2022).
- API Integration for Natural Language Processing (NLP): Google Cloud Natural Language for processing of user inputs by offering a collection of pre-trained models that analyse text and extract details (Google Cloud, 2018).
- Machine Learning and Data Processing Libraries: Pandas will be used for the management of data structures and the execution of data analysis (Pandas,2024). Also, Scikit-learn will help to provide methods for creating and assessing models (Scikit-learn, 2019).

- Version Control and Collaboration: Github will be used for code repository management and cooperation (GitHub,2024).
- Integrated Development Environment (IDE): Visual Studio Code was chosen because of its support for programming (Visual Studio Code, 2023).

3.4 Summary

The essential functional and non-functional needs for an AI-driven recommendation system in a student note-taking application are described in the requirements section. Using a MoSCoW analysis to separate key "must-have" features from optional improvements, key functionality includes user registration, secure note-taking, personalised learning material recommendations, and connection with NLP APIs. In addition to handling scalability, maintainability, and accessibility, the non-functional requirements ensure the application's usability, security, GDPR compliance, and data integrity.

4 Design

4.1 System Architecture

The note-taking application consists of multiple components working together to provide users with a simple experience in creating notes and receiving AI-powered learning material recommendations. The architecture follows a client-server model, where the front-end provides a simple and interactive user interface, and the back-end processes the requests and integrates with external API from Google, and a MySQL database that stores the user data and the recommendations as well.

The system comprises the following components:

- Front-end: A web-based interface where users can take notes and view AI-generated recommendations. Developed using HTML, CSS, and JavaScript, rendered through Flask templates, this web-based interface allows users to log in, takes notes, and view AI-generated recommendations in real time.
- Back-end: Built using Python and the Flask web framework. It handles user authentication, form submissions, communicates with the Google NLP API, processes data, and delivers personalised recommendations to the front-end.
- Database: Managed using MySQL, hosted locally with XAMPP, stores user account information (username, email, hashed password), educational preferences, notes, recommendation history, and feedback on relevance.
- External API: The system integrates the Google Cloud Natural Language API, which performs Natural Language Processing (NLP) tasks, including keyword extraction and entity recognition, to analyse the content of user notes.
- Recommendation System: Implements a content-based recommendation model using TF-IDF Vectorization and Cosine Similarity (with the scikit-learn library). It processes the extracted keywords and matches them with the combined dataset of learning materials. The top 3 most relevant recommendations are presented to the user.

The system follows a structured flow, ensuring a simple interaction between the user, backend, database, external API, and recommendation system. The process is divided into the following key stages:

- User Registration/Login: A new user registers an account or logs in using their credentials. The back-end verifies authentication and retrieves the user's existing notes (if any) from the database.
- Note-Taking and Saving: The user writes notes in the front-end interface and when

the user saves a note, the front-end sends the data to the back-end and the note is stored in the database for future retrieval.

- Text Analysis and Recommendation Generation: The user clicks on a button to receive recommendations, which makes the back-end to send the saved note to the Google NLP API for analysis and for the API to extract key concepts from the note. Based on the extracted topics, the back-end retrieves relevant learning materials from a predefined dataset.
- Displaying Recommendations: The recommendations generated are stored in the database and sent back to the front-end, where the user can see the suggested study materials related to their notes.
- User Interaction and Feedback: The user can interact with the recommendations by accessing them (if a link is given by the system), and rating each of the recommendations stating whether it is relevant or not.
- Logging Out: When the user logs out, their session is cleared, and when they next login, their previous notes and saved recommendations are reloaded from the database.

4.2 Front-end Design

The front-end of the note-taking application is designed to provide users with an intuitive and easy experience for creating notes, receiving personalised learning material recommendations, and managing their profiles. The interface is divided into many key pages, each serving a specific purpose in the user workflow (see appendix G for visual and mockup designs).

Login/Sign Up Page

Users can access the application using the Login/Sign Up Page. It has a straightforward simple style with two main choices:

- Login: Users can log in using their username and password.
- Sign Up: New users can register by providing their username, password, current subjects/modules, and more.

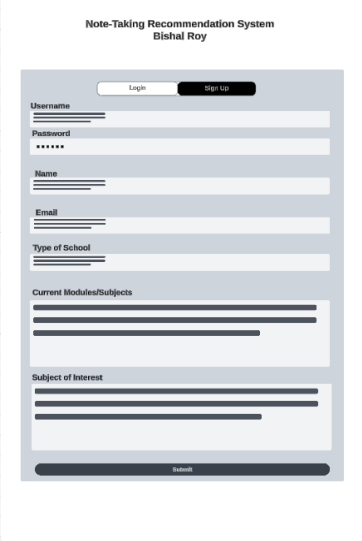


Fig. 3. Login/Sign Up mockup page

My Notes Page

The My Notes page is the central hub where users can view, search, and manage their notes and previous recommendations. It displays a list of all saved notes, along with their titles, creation dates, and a preview of the note content. Also, it displays a list of previous recommendations received and manages the recommendations by keeping the ones the user prefers and deleting the ones they do not want.

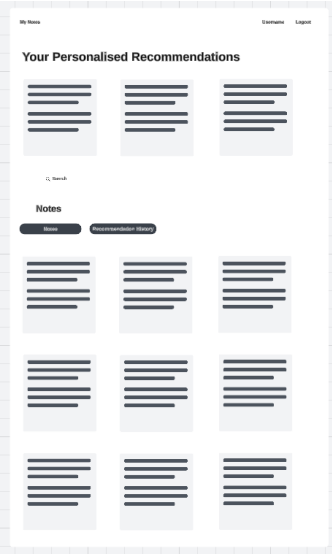


Fig. 4. My Notes mockup page

Create/Edit Note Page

The Create/Edit Note page allows users to write and save notes or edit existing notes. It provides a simple text input interface for note content. Underneath the text editor, there is a button that the user clicks on, and the recommendation system displays personalized learning material recommendations based on the content of the user's notes. Each recommendation includes details such as the resource name, type, author, and similarity score. Only the top 3 recommendations with the highest similarity score are listed, with each recommendation including buttons to mark them as "Relevant" or "Not Relevant".

Fig. 5. Create/Edit Note mockup page

User Settings Page

The User Settings page allows users to update their profile information, including their user, password, email address, subject of interest, current subjects/modules, and more. At the end of the form there is a button name "Save Changes" to apply the updates and update it in the database.

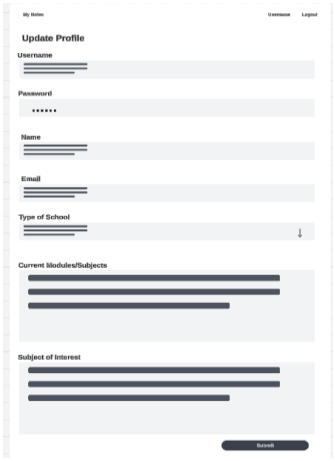


Fig. 6. User Settings mockup page

Logout

The Logout option is available on all pages, allowing users to securely log out their account. Upon logging out, the user's session is cleared, and they are redirected to the login page.

The following standards are followed in the front-end design:

- User-Centric Design: The interface is designed to be easy to navigate, ensuring a positive user experience
- Consistency: A consistent layout, colour scheme of red and black, are used across all pages to produce a single look and feel.
- Accessibility: The design is accessible to users with bright colours, and clear fonts.

While the current user interface was primarily designed with larger screens, mainly for desktop and laptop screens, it does not yet fully adapt to smaller devices like smartphones as its layout and component spacing may appear oversized or less user-friendly on mobile screens. This limitation has been noted, and as part of future work, the design will be refined using responsive design techniques such as flexible grids, scalable UI elements, and media queries to ensure a smoother and more accessible experience across all device types, including smartphones.

4.3 Back-end Design

The back-end of the application handles data storage, processing, and integration with the recommendation system. It is designed to ensure efficiency in processing user notes, analysing content, and generating learning material recommendations.

Data Storage and Management

User data, such as notes, recommendations, and more, are stored in a structured format to make preprocessing and retrieving simple. There are two primary components to storage design:

- User Data Storage: User-generated notes are stored in a database, along with metadata such as note IDs, user IDs, and timestamps. This data is essential for personalising recommendations based on individual notes, what they study, and their study of interest.
- Learning Materials Dataset: The main dataset used for the recommendation was created by merging two pre-processed datasets containing details about learning resources, such as titles and authors. The file named "Combined_learning_materials.csv" contains:
    - Title: The title of the resource.
    - Description: A brief overview of the resources (if available).
    - Author: Name of the author.
    - Subject: Subject area of the material.
    - URL: Link to the resource (if available).

Integration of the NLP API and Dataset

The back-end integrates the Google Natural Language Processing (NLP) API to extract meaningful keywords and entities from users' notes. These extracted entities are then matched against the dataset to identify relevant learning materials.

- NLP API Integration: When a user saves a note, the note content is sent to the NLP API for analysis. Then, the API extracts key entities, such as topics, that represent the core ideas of the note.
    - Input Note from a test result: "Learn the advanced techniques of HTML, CSS, and JavaScript, and learning how to improve my skills on it."
    - Extracted Entities: ['skills', 'HTML', 'JavaScript', 'CSS', 'techniques']
- Matching Keywords with Learning Materials: The extracted keywords are used to query the dataset which has been vectorized using TF-IDF. A similarity score is also

calculated using cosine similarity to identify the most relevant materials for the user.

- Recommendation Route: User notes → Keywords extraction with NLP API → Keyword matching with dataset → recommendations generated

4.4 Summary

The design of the note-taking application focuses on implementing a recommendation system that personalises study materials based on users' notes. The front-end was designed using the mockups (in the appendix G) to ensure a user-friendly experience, with key components such as a text editor and recommendation section. The back-end handles data storage and processing, using the Google NLP API to extract keywords from notes and match them with relevant learning materials from the dataset. The content-based recommendation system employs TF-IDF vectorization and cosine similarity to generate accurate suggestions. This approach ensures simple functionality and effective personalised learning support.

5 Implementation

5.1 Recommendation Model

The AI recommendation model is responsible for analysing the user notes and suggesting relevant study materials. This system makes use of Google's Natural Language Processing (NLP) API to extract meaningful information from user-input text and uses a content-based filtering approach to recommend learning resources. The development of the content-based recommendation system in this project was influenced by the step-by-step guide provided by Gaurav, 2023. This resource offered practical guidance and examples that helped me form the foundation for designing and implementing the recommendation algorithm used in the system. Full content-based recommendation system code with comments in appendix F and the example results of the recommendation system can be seen in appendix L.

The recommendation model is made to analyse users' notes, extract meaningful keywords, and match them against a database of learning materials. The system follows these steps:

- Extract key topics from user notes using Google's Natural Language Processing (NLP) API (full Google Cloud Integration NLP code in Appendix K).
- Vectorize learning materials from the dataset using TF-IDF (Term Frequency-Inverse Document Frequency), which converts the textual description of each material into numerical data that can be processed by the recommendation model, helping the system to focus on words that are more unique and meaningful.
- Calculate the similarity scores between extracted keywords and the existing learning material using cosine similarity, which calculates the angle between two vectors, where a smaller angle means higher similarity, to suggest the best matches.
- Generate ranked recommendations based on the highest similarity scores and provide the top 3 learning material recommendations.

Vectorizing Learning Materials

Once the system extracts keywords from users' notes, it compares them against a dataset of learning materials, which contains titles, descriptions, authors, and URLs of online resources. To prepare the data for comparison, the TF-IDF (Term Frequency-Inverse Document Frequency) technique is used, converting text descriptions of learning materials into numerical feature vectors. The TF-IDF vectorization is implemented as follows:

Here, the "Title" and "Description" fields are combined to create a single feature for each learning material. The TfidfVectorizer converts this text into a matrix of numerical values that represent the importance of different words.

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

filePath = 'combined_learning_materials.csv'
data = pd.read_csv(filePath)

data['Combined_Features'] = data['Title'].fillna('') + ' ' + data['Description'].fillna('')
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(data['Combined_Features'])
```

Code Listing 1. Python Code of recommendation system vectorizing learning materials part

Computing Similarity

To find the most relevant learning materials, the system calculates the cosine similarity between the extracted keywords and the TF-IDF vectors of all stored materials. It is implemented as follows:

```
from google_nlp_api import note_analyser
from sklearn.metrics.pairwise import cosine_similarity

def recommend_learning_materials(title, content):
    keywordsTitle = note_analyser(title)
    keywordsContent = note_analyser(content)
    keywordsCombined = ' '.join(keywordsTitle + keywordsContent)

    user_tfidf = tfidf.transform([keywordsCombined])
    userSimilarityScores = cosine_similarity(user_tfidf, tfidf_matrix)
    userSimilaritySeries = pd.Series(userSimilarityScores[0],
index=data['Title'])

    noDuplicatedSeries =
userSimilaritySeries[userSimilaritySeries.index.duplicated(keep='first') ==
False]
```

Code Listing 2. Python Code of recommendation system cosine similarity part

5.2 Front-end Implementation

The front-end of the note-taking application was implemented to provide a simple and user-friendly, allowing users to efficiently take notes and receive AI-powered learning recommendations. Design priorities ease of use, accessibility, and responsiveness while ensuring simple interaction with the back-end recommendation system. The technologies

used were HTML, CSS, JavaScript as the core structure, styling, and interactivity as well as Flask to dynamically render pages with user data.

User Authentication with Login/Signup page

The login and sign-up forms allow user to register, log in, and log out securely. Session handling ensures users remain logged in across sessions and navigating through the application. A sliding tab interface enables users to toggle between the login and sign-up forms using a smooth sliding effect, implemented with JavaScript. Additionally, user profile data is collected during sign-up, including information such as the user's current courses and subjects of interest. This data helps enhance the accuracy of recommendations.

Note-Taking Interface

The note-taking interface includes a custom-built rich text editor developed using HTML, JavaScript, and Font Awesome icon library. It allows users to format their notes with various options such as bold, italics, underline, strikethrough, font size adjustments, alignment tools, and lists. These formatting features are implemented using the built-in document.execCommand() method in JavaScript, enabling a browser-compatible editor without relying on external libraries. Notes are stored in the database and can be edited or reviewed later. This interface is also integrated with the recommendation system, clicking the "Provide Me Learning Recommendations Now!" button submits the current saved note to the back-end, which then returns three relevant study resources based on the analysed note.

```
<form action="{{ url_for('recommend') }}" method="post">
  <input type="hidden" name="noteid" value="{{ note.noteid if note else '' }}">
  <input type="hidden" name="titleNote" value="{{ note.title if note else '' }}">
  <input type="hidden" name="content" value="{{ note.content | safe if note else
'' }}">
  <button type="submit" class="clearRecommendations">Provide Me Learning
Recommendations Now!</button>
</form>
```

Code Listing 3. HTML Code of collecting user input to generate recommendations

Notes and Previous Recommendations Management

In this section, users can easily view, edit, and delete their saved notes. A search feature allows users to find specific notes or past recommendations quickly. Also, users can access their recommendation history, which displays previous suggestions for easy access and future

reference. Additionally, it displays three of their personalised recommendations based on their profile information on their current subject and subject of interest.

Profile Management

The profile management page enables users to update their information, including their username, email, school type, and interests. These updates are stored in the database, ensuring that personalised recommendations adapt over time to reflect any changes that the user has made to their preferences.

Challenges and Solutions during the front-end development:

- Dynamic UI updates: Users had to refresh the page for feedback and recommendations. Implemented JavaScript fetch() API for asynchronous updates, improving responsiveness.
- Managing large recommendation lists: Browsing became difficult as lists grew. Added a search function for easier navigation and relevant recommendations.

5.3 Back-end Implementation

The back-end of the note-taking application was developed using Flask and MySQL to handle user authentication, note management, and simple interaction with the front-end. The technologies that were used for the back-end were Flask to handle the routing, request processing and the API interactions, and along with Flask, the Flask-MySQLdb was used to provide communication between Flask and the MySQL database, and the Flask-Bcrypt was used to ensure secure password hashing. MySQL was used, which stores user accounts, notes, recommendations, and feedback. Also, Google NLP API was used to extract the keywords from the user's notes for recommendation generation. Additionally, Scikit-learn was used to create the recommendation system and for the precision evaluation of the recommendation system.

Key features that were implemented into this development were:

- User authentication: The system ensures secure login and registration with bcryot-hashed passwords and session management for simple authentication.
    - Bcrypt hashing encrypts passwords by adding salt (random value added to passwords before hashing to make it unique) and performing multiple hashing rounds (Dan, 2021).
- Note Management: Users can create, edit, delete, and retrieve notes stored in a MySQL database for durability and easy access.
- Recommendation system: Uses Google NLP API to extract keywords from notes and

applies content-based filtering to suggest relevant learning materials, which are saved for future reference.

Challenges that occurred during implementation that were later solved were:

- Secure authentication: Preventing unauthorized access was addressed by implementing bcrypt for password hashing and session validation for user authentication.
- Recommendation accuracy: Limited dataset size affected precision. Future work will expand datasets and NLP keyword extraction can be improved for more individualised suggestions.

Future Improvements for the back-end:

- Personalised learning paths: Improve recommendation filtering to better align with user interests, ensuring more relevant learning materials.
- Scalability improvements: Optimize database queries to maintain performance as user numbers and data volume grow.
- Advanced machine learning models: Implement deep learning techniques to improve recommendation accuracy and capture complex learning patterns.
- Expanding the learning material dataset: Increase resource variety (e.g., research papers, textbooks, interactive tools) to better meet a range of learning requirements.

5.4 Summary

The implementation of the note-taking application consists of a front-end and back-end for a simple user experience. The front-end, built with HTML, CSS, JavaScript, and Flask templates, includes features like a text editor, search option, user authentication, and a profile management system. The back-end uses Flask and MySQL for authentication, data storage, and the recommendation system. Passwords are securely managed with bcrypt, and Google NLP API extracts keywords from user notes. A content-based recommendation system using TF-IDF and cosine similarity matches notes with relevant materials, using user feedback helps to measure the precision from user evaluation by clicking on “Relevant” and “Not Relevant” buttons on each recommended learning material.

The recommendation model uses Google’s Natural Language Processing (NLP) API for keyword extraction and Scikit-learn for TF-IDF vectorization and cosine similarity. It ranks learning materials by relevance and provides the top 3 recommendations based on the user’s notes. Future improvements include personalisation, filtering options, expanded datasets, and more advanced AI models to improve recommendation accuracy and scalability.

6 TESTING AND EVALUATION

The evaluation approach outlines the methods, measurements, and tests used to evaluate functionality, performance, usability, and efficiency in improving student learning. It ensures that the application satisfies project objectives, such as evaluating the effectiveness of AI-driven note-taking, the accuracy of recommendations, time saved, and user satisfaction and engagement.

6.1 Approaches

The project adopted a combination of functional, usability, and performance testing to ensure the application met its technical and user-experience goals. This method helped in identifying key attributes, analysing user engagement, and determining the recommendation system's accuracy. The testing plan consisted of:

- Functional Testing: Core features such as note saving, retrieval, and recommendation generation were verified through manual test-cases.
- Usability Testing: The interface and overall ease of use were assessed with a focus on layout, accessibility, and user satisfaction. Feedback was gathered using Likert scale questions (Qualaroo Blog, 2022) and open-ended questions (university.sopact.com, 2024) to understand user experience.
- Recommendation System Testing: Users interacted with recommendations by marking them as “Relevant” or “Not Relevant”. Precision was used to calculate the proportion of relevant recommendations over total recommendations (Anon, 2024).

Evaluation Method

The evaluation method focused on measuring the system’s impact on learning efficiency and user satisfaction. A user study was conducted with students in different fields of study, who voluntarily participated in testing the note-taking and recommendation features. Participants completed a feedback survey consisting of:

- Likert-scale questions to rate the ease of use, interface simplicity, recommendation usefulness, and overall satisfaction.
- Open-ended questions to gather detailed information about the UI design, the quality of the recommendations, and areas that could use better.

Participants also rated how quickly they could find relevant study materials and how helpful the recommendations were in supporting their learning. This combination of quantitative and qualitative feedback offered an in-depth evaluation of how well it is able to assist students in a real-world academic setting. Participants were recruited on-campus at Heriot-Watt University, with emphasis on voluntary participation, privacy, and withdrawal options.

6.2 Experiment Setup

The experiment setup involved evaluating the functionalities of the note-taking application and the precision of the recommendation system. The tasks assigned to users were designed to test the application's core features, including note-taking, and recommendation generation.

Environment and Tools

The application was tested on a local server using the Python Flask servers for the back-end. The tools that were used for this application are:

- Python: For back-end logic and NLP integration.
- Google NLP API: For extracting keywords.
- "Combined_learning_materials.csv" dataset: For providing learning material recommendations, and pre-processed to include titles, descriptions, authors, subjects, and URLs, served as the source of learning materials for recommendations.

User Tasks

Amongst the 9 participants who took part in the experiment, the average time taken to complete it was 14 minutes and 14 seconds for completing the evaluation form (screenshots of the form can be found in appendix D), and overall, each session lasted approximately 30 minutes, depending on the depth of engagement with the note-taking process. Additionally, the students were free to choose their own topics based on their current study interest, to create a diverse set of recommendations of different fields, and on average, students create 2-3 notes per session (example notes inputs and recommendation outputs can be found in appendix L). Also, students were asked to complete their notes in a single session, while I was present during the testing as I read out the tasks that the participants had to complete. Participants were assigned the following tasks as part of the testing process (full testing plan can be found in the appendix M):

- Authentication Tests:
    - Register a new user with invalid details and verify the display of error messages.
    - Register a new user with valid details and confirm redirection to the login page.
    - Log in with both invalid and valid credentials to check error handling and redirection.
    - Log out to ensure the session ends correctly.
- Note-Taking Features:

    - Create multiple notes, save them, and confirm their appearance on the notes page.
    - Edit and save an existing note, verifying the updates.
    - Delete a note and confirm its removal from the notes page.
- Recommendation System and Feedback:
    - Open a saved note and click the “Recommend Me Learning Materials” button to generate recommendations and verify that the system provides three relevant learning materials.
    - Mark each recommended learning material as “Relevant” or “Not Relevant” and ensure the feedback is saved in the database.
- Search Functionality:
    - Search for previously recommended materials using their titles.
    - Test the system’s response when no matching results are found and clear the search input to display all recommendations.

Participants included 9 students from various backgrounds from college and university students studying different kinds of subjects such as Business and Computer Science, ensuring diverse feedback on usability and relevance. This setup ensured a thorough evaluation of the application’s functionality and the recommendation system's precision.

6.3 Evaluation Metrics

To assess the effectiveness of the recommendation system, two key evaluation metrics were used: precision and user satisfaction. These metrics provide both a quantitative and qualitative measure of the system’s performance, evaluating the relevance of the recommendations as well as how satisfied users were with the study materials provided.

To assess the effectiveness of the recommendation system, the following evaluation metrics were used:

Precision

Precision measures of how many of the recommended learning materials were relevant to the user. It is calculated as:

Precision = Number of relevant recommendations / Total recommendations made

The system collects user feedback on each recommendation, allowing users to mark materials as “Relevant” or “Not Relevant”. The precision score is then calculated on this

feedback. Full precision calculation python function code with comments in appendix J.

Why choose precision for content-based recommendation system?

In a content-based recommendation system, the goal is to provide highly relevant study materials based on the user's notes. Precision is the most appropriate metric because it measures the proportion of recommendation materials that are relevant, focusing on the accuracy of recommendations rather than their quantity (Jadon, 2024).

Reasons for using precision in a content-based recommendation system:

- Relevance over quantity: Precision directly evaluates the accuracy of the recommendations provided to the user. A high precision score means most recommended materials are useful, which aligns with the goal of a content-based system.
- User-Centric evaluation: Precision ensures the system matches the user's actual needs rather than just suggesting many potentially relevant items.
- Avoiding irrelevant recommendations: a high recall but low precision would mean that the system retrieves many learning materials, but many of them may be irrelevant, leading to poor user experience. Precision ensures that only highly relevant suggestions are prioritized.

User Satisfaction

To evaluate the user satisfaction of the note-taking application and its recommendation system, a questionnaire was designed and distributed to participants. The questionnaire included questions like Open-ended questions where the participants provided feedback on challenged, improvements, and experiences, and Likert-Scale questions where the participants rated their satisfaction on a 5-point scale (Strongly Disagree to Strongly Agree).

The questionnaire gathered feedback on usability, performance, recommendation quality, and overall user experience. The user satisfaction was measured across several factors:

- Ease of Use and Navigation: Participants rated the application's ease of navigation and suggested improvements. One of the questions that helped answer this was "How would you describe the ease of navigating through the application?".
- Accessibility: Participants evaluated the application's accessibility for users with disabilities. One of the questions that helped answer this was "How accessible do you find the application?".
- Performance and Reliability: Participants rated the application's consistency and

responsiveness. A couple of the questions that helped answer this was “The application consistently works as expected without crashing or losing data." And "The system maintains consistent performance without significant delays or lag." In the form of a Likert-Scale question.

- Efficiency: Participants assessed how effectively the application helped them access study materials and save time. A couple of the questions that helped answer this was "The application helps improve my efficiency in accessing study materials." And "Using the application helped me save time when looking for study materials." In the form of a Likert-Scale question.
- Recommendation Quality and Relevance: Participants evaluated the relevance and quality of the recommended learning materials. A couple of the questions that helped answer this was "I am satisfied with the learning materials recommended based on my notes." And The recommended materials are relevant to the topics covered in my notes." Using Likert-Scale questions.
- Overall Usability and Recommendation: Participants rated the overall usability and their willingness to recommend the application. A couple of the questions that helped answer this was “The application is easy to use and navigate." And "I would recommend this application to other students." Using Likert-Scale questions.

Overall, the user feedback indicates that the application is generally easy to use, reliable, and efficient in helping users access relevant study materials. With high performance and content relevance scores, most participants said the interface was easy to use and the suggestions were helpful. While some areas such as accessibility, UI design, and recommendation variety were highlighted for improvement, the overall user satisfaction suggests that the system is a promising tool for supporting student learning.

6.4 Results of the Experiment

This section presents the results of the experiment, focusing on user satisfaction and the precision of the recommendation system. The findings are based on questionnaire responses and the precision evaluation of the recommendation system.

User Satisfaction Results

The questionnaire was completed by 9 participants, and their feedback provided valuable insights into the usability, performance, and effectiveness of the application. Some of the key findings were:

- Ease of Use and Navigation: 6 out of 9 participants found the application easy to navigate, with comments such as “Navigation was simple and concise”. However, 4 out of 8 suggested improvements, such as reducing redundant actions and enhancing

the visibility of UI elements.

- Accessibility: 7 out of 9 participants recommended adding features like a light mode and voice control to assist users with disabilities.
- Performance and Reliability: 8 out of 9 participants agreed that the application consistently worked without crashing or losing data. However, 1 participant noted minor delays, but overall, the performance was rated highly.
- Efficiency: 8 out of the 9 participants reported that the application improved their efficiency in accessing study materials as well as 8 participants found that it helped them to save time compared to traditional search materials, while one of the participants were unsure whether it helped them.
- Recommendation Quality and Relevance: All 9 participants found the recommended materials relevant to their notes, and 8 were satisfied overall, describing them as "very relevant." While 6 of the 8 were happy with the variety, some suggested improvements like offering more than three suggestions and increasing precision, indicating room for enhancement.
- Overall Usability and Recommendation: 8 out of 9 participants found the app easy to use, and 6 said they would recommend it. Some suggested improvements like a cleaner UI/UX and a more visible colour scheme.

Precision of the Recommendation System

The precision score recorded is 0.563 (56.3%), meaning that 56.3% of the recommended materials were found relevant by users. The calculation was based on 40 relevant items out of 71 total recommendations. A precision of 56.3% indicates that slightly more than half of the recommended materials were relevant to the users' notes. While this shows a promising result, there is room for improvement, particularly in expanding the dataset and refining the recommendation algorithm to increase relevance.

6.5 Summary

The experiment evaluated the recommendation system's precision as well as the satisfaction of users. In general, participants thought the program was reliable, effective, and simple to use for accessing study materials. Enhancing the user interface, reducing pointless tasks, and increasing accessibility features were some recommendations for improvement. With a 56.3% precision score, the recommendation system made recommendations that were somewhat more relevant. Although this suggests encouraging results, drawbacks were noted, including a small dataset and the requirement for better recommendation filtering. The user experience and suggestion accuracy may be enhanced by future developments, such as dataset growth and algorithm improvement.

7 CONCLUSION

This dissertation proposed an AI-driven recommendation system implemented into a student note-taking application to improve personalised learning. The system makes use of Natural Language Processing (NLP) to extract key topics from student notes and generate content-based learning material recommendations. The goal was to address inefficiencies in traditional study methods by reducing information overload, increasing engagement, and improving learning efficiency.

7.1 Achievements

This project successfully designed, implemented, and evaluated an AI-driven recommendation system within a student note-taking application. The main achievements include:

- Development of an AI-powered note-taking system that implements Google's NLP API to extract key concepts from student notes and implement a content-based recommendation system to recommend relevant learning materials.
- Evaluation of system effectiveness through user testing, where the recommendation system achieved a precision score of 56.3%, indicating improvable accuracy.
- Identification of areas for improvement, including dataset expansion, better filtering techniques, and enhanced personalisation.

These achievements demonstrate the usefulness of AI-driven recommendation systems in learning environments and support further studies in individualised education.

7.2 Contributions

This research makes several key contributions to the field of AI-driven educational technology, showing how AI-driven solutions can improve self-directed learning by providing personalised study material recommendations.:

- Implemented a content-based recommendation system that personalises learning material recommendations based on students' notes and integrated Google's Natural Language Processing (NLP) API for automated keyword extraction and topic identification.
- Designed a content-based approach that uses TF-IDF vectorization and cosine similarity to match extracted keywords with relevant study materials. Also, I ensured recommendations were relevant by linking them to organized resources such as the Learning Resources Database and the Amazon Book Dataset.
- Conducted user testing to assess the precision of recommendations and the impact on learning efficiency and the recommendation precision score calculated by user

evaluation was 56.3%, showing improvable results. Additionally, survey results showed good user satisfaction regarding usability, engagement, and recommendation quality.

7.3 Limitations and Future Work

While this dissertation successfully developed and tested a content-based recommendation system, several limitations were identified:

- Limited dataset for recommendations: The recommendation system was trained on a predefined set of learning resources (Amazon Book dataset, Learning Resources Database). Expanding this dataset could improve recommendation diversity and accuracy.
- Content-based filtering limitations: The system does not consider user preferences over time, which means recommendations may not fully adapt to long-term changes in study habits.
- Absence of multiple learning resources: Although films, podcasts, and interactive content are becoming more and more popular in modern education, the system still only suggests text-based resources (books, articles) with some learning materials being links to web pages as well.
- Cold-start issue: When a user enters little information, the system may find it difficult to make useful recommendations because content-based filtering only uses note keywords.

Additionally, to enhance the system's capabilities, several improvements and extensions can be implemented for future work:

- Combining collaborative filtering with content-based filtering to create a hybrid recommendation model to improve recommendation diversity and adapt to evolving user preferences.
- To expand the recommendation dataset to include open educational resources, video tutorials, research papers, and more to offer a broader range of learning materials, as well as, to implement web scraping or API-based integration with digital libraries.
- Allow users to rate recommendations and provide specific comments to improve upcoming recommendations.
- Addressing user feedback on UI/UX design to make navigation and interaction more intuitive as well as providing accessibility enhancements, including dark mode, voice search, and more. As well as, adding filtering options that allow users to sort learning materials by subject or relevance, and allowing users to personalise their profiles further by saving content preferences, helping the system improve future suggestions.

REFERENCES

Apache Friends (2022). *XAMPP Installers and Downloads for Apache Friends*. [online] www.apachefriends.org. Available at: https://www.apachefriends.org/.

Aberbach, H., Jeghal, A., Sabri, A., Tairi, H. (2022). E-learning Recommendation Systems: A Literature Review. In: Motahhir, S., Bossoufi, B. (eds) Digital Technologies and Applications. ICDTA 2022. Lecture Notes in Networks and Systems, vol 454. Springer, Cham. Available at: https://doi.org/10.1007/978-3-031-01942-5_36

Anon. (2024) Understanding precision & recall: key metrics for evaluating model performance. Available at: https://graphite-note.com/understanding-precision-recall-key-metrics-for-evaluating-model-performance/ (Accessed: 17 Nov. 2024).

Arias, D. (2021) Hashing in action: understanding bcrypt. Available at: https://auth0.com/blog/hashing-in-action-understanding-bcrypt/ (Accessed: 17 Nov. 2024).

Bobadilla, J., Ortega, F., Hernando, A. and Bernal, J. (2012). A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems*, 26, pp.225–238. Available at: https://doi.org/10.1016/j.knosys.2011.07.021

Bouguezzi, S. (2024). *How Does the Amazon Recommendation System Work? | Baeldung on Computer Science*. [online] www.baeldung.com. Available at: https://www.baeldung.com/cs/amazon-recommendation-system (Accessed: 11 Mar. 2025).

Dan Arias (2021). *Hashing in Action: Understanding bcrypt*. [online] Auth0 - Blog. Available at: https://auth0.com/blog/hashing-in-action-understanding-bcrypt/ (Accessed: 17 November 2024).

Dubey, R. (2024) *Amazon book dataset: 30000+ books with 30+ category*. Kaggle [online]. Available at: https://www.kaggle.com/datasets/rajkumardubey10/amazon-book-dataset30000-books-with-30-category (Accessed: 17 November 2024).

Er-Rafyg, A., Idrissi, A. (2023). Review of Recommendation Systems and their Applications in E-learning. In: Idrissi, A. (eds) Modern Artificial Intelligence and Data Science. Studies in Computational Intelligence, vol 1102. Springer, Cham. Available at: https://doi.org/10.1007/978-3-031-33309-5_3

Font Awesome (2024) *Font Awesome: The Web's Most Popular Icon Set and Toolkit*. Available at: https://fontawesome.com/ (Accessed: 24 Mar. 2025).

Gaurav, P. (2023) Step by step content-based recommendation system. Available at: https://medium.com/@prateekgaurav/step-by-step-content-based-recommendation-system-823bbfd0541c (Accessed: 10 Feb. 2025).

GitHub (2024) GitHub.com help documentation. Available at: https://docs.github.com/en (Accessed: 17 Nov. 2024).

Google Cloud (2018) Natural language API basics: Cloud Natural Language API. Available at: https://cloud.google.com/natural-language/docs/basics (Accessed: 17 Nov. 2024).

IBM (2023) Named entity recognition. Available at: https://www.ibm.com/think/topics/named-entity-recognition (Accessed: 17 Feb. 2025).

Ifenthaler, D., Majumdar, R., Gorissen, P. *et al.* Artificial Intelligence in Education: Implications for Policymakers, Researchers, and Practitioners. *Tech Know Learn* **29**, 1693–1710 (2024). Available at: https://doi.org/10.1007/s10758-024-09747-0

Jadon, A. (2024) A comprehensive survey of evaluation techniques for recommendation systems. Available at: https://arxiv.org/pdf/2312.16015 (Accessed: 10 Feb. 2025).

Khanal, S.S., Prasad, P., Alsadoon, A. *et al.* A systematic review: machine learning based recommendation systems for e-learning. *Educ Inf Technol* **25**, 2635–2664 (2020). Available at: https://doi.org/10.1007/s10639-019-10063-9

Li, B., Han, L. (2013). Distance Weighted Cosine Similarity Measure for Text Classification. In: Yin, H., *et al.* Intelligent Data Engineering and Automated Learning – IDEAL 2013. IDEAL 2013. Lecture Notes in Computer Science, vol 8206. Springer, Berlin, Heidelberg. Available at: https://doi.org/10.1007/978-3-642-41278-3_74

Linden, G., Smith, B. and York, J. (2003). *Amazon Recommendations*. [online] Available at: https://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf (Accessed: 11 Mar. 2025).

Liu, T., Wu, Q., Chang, L. *et al.* A review of deep learning-based recommender system in e-learning environments. *Artif Intell Rev* **55**, 5953–5980 (2022). Available at: https://doi.org/10.1007/s10462-022-10135-2

MDN Web Docs (2018) HTML: HyperText Markup Language. Available at: https://developer.mozilla.org/en-US/docs/Web/HTML (Accessed: 17 Nov. 2024).

MDN Web Docs (2019) CSS: Cascading Style Sheets. Available at: https://developer.mozilla.org/en-US/docs/Web/CSS (Accessed: 17 Nov. 2024).

MDN Web Docs (2023) JavaScript. Available at: https://developer.mozilla.org/en-US/docs/Web/javascript (Accessed: 17 Nov. 2024).

MDN Web Docs (2024) *Document.execCommand() – Web APIs | MDN*. Mozilla Developer Network. Available at: https://developer.mozilla.org/en-US/docs/Web/API/Document/execCommand (Accessed: 24 Mar. 2025).

MDN Web Docs (n.d.) Fetch API – Web APIs. Available at: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API (Accessed: 17 Nov. 2024).

Mifsud, J. (2018) Usability metrics: a guide to quantify the usability of any system. Usability Geek. Available at: https://usabilitygeek.com/usability-metrics-a-guide-to-quantify-system-usability/ (Accessed: 17 Nov. 2024).

Mondal, S.K., Sahoo, J.P., Wang, J., Mondal, K., Rahman, M.M. (2022). Fake News Detection Exploiting TF-IDF Vectorization with Ensemble Learning Models. In: Sahoo, J.P., Tripathy, A.K., Mohanty, M., Li, KC., Nayak, A.K. (eds) Advances in Distributed Computing and Machine Learning. Lecture Notes in Networks and Systems, vol 302. Springer, Singapore. Available at: https://doi.org/10.1007/978-981-16-4807-6_25

Moturu, V.R., Nethi, S.D. (2023). Artificial Intelligence in Education. In: Chaurasia, M.A., Juang, CF. (eds) Emerging IT/ICT and AI Technologies Affecting Society. Lecture Notes in Networks and Systems, vol 478. Springer, Singapore. Available at: https://doi.org/10.1007/978-981-19-2940-3_16

MySQL (2019) MySQL: MySQL documentation. Available at: https://dev.mysql.com/doc/ (Accessed: 17 Nov. 2024).

Oubalahcen, H., El Ouadghiri, M.D. (2024). Social Recommender Systems in E-Learning Environments: A Literature Review. In: Mabrouki, J., Azrour, M. (eds) Advanced Systems for Environmental Monitoring, IoT and the application of Artificial Intelligence. Studies in Big Data, vol 143. Springer, Cham. Available at: https://doi.org/10.1007/978-3-031-50860-8_17

Palletsprojects.com (2024) Welcome to Flask—Flask documentation (3.0.x). Available at: https://flask.palletsprojects.com/en/stable/ (Accessed: 17 Nov. 2024).

Pandas (2024) pandas documentation — pandas 1.0.1 documentation. Available at: https://pandas.pydata.org/docs/ (Accessed: 17 Nov. 2024).

Patil, P. (2023) *Learning resources database*. Kaggle [online]. Available at: https://www.kaggle.com/datasets/prasad22/learning-resources-database (Accessed: 17 November 2024).

Python Software Foundation (2019) 3.7.3 documentation. Available at: https://docs.python.org/3/ (Accessed: 17 Nov. 2024).

Qualaroo Blog (2022) Likert scale surveys: why & how to create them (with examples). Available at: https://qualaroo.com/blog/likert-scale/ (Accessed: 17 Nov. 2024).

Reddy, B.D., Kumar, L.S.C., Nelatur, N. (2020). A Review on Datasets and Tools in the Research of Recommender Systems. In: Fiaidhi, J., Bhattacharyya, D., Rao, N. (eds) Smart Technologies in Data Science and Communication. Lecture Notes in Networks and Systems, vol 105. Springer, Singapore. Available at: https://doi.org/10.1007/978-981-15-2407-3_8

Renuka, S., Raj Kiran, G.S.S., Rohit, P. (2021). An Unsupervised Content-Based Article Recommendation System Using Natural Language Processing. In: Jeena Jacob, I., Kolandapalayam Shanmugam, S., Piramuthu, S., Falkowski-Gilski, P. (eds) Data Intelligence and Cognitive Informatics. Algorithms for Intelligent Systems. Springer, Singapore. Available at: https://doi.org/10.1007/978-981-15-8530-2_13

Ross, J. (2023) Students 'overwhelmed' by onslaught of course content. Times Higher Education. Available at: https://www.timeshighereducation.com/news/students-overwhelmed-onslaught-course-content (Accessed: 17 Nov. 2024).

Scikit-learn (2019) scikit-learn: machine learning in Python. Available at: https://scikit-learn.org/stable/ (Accessed: 17 Nov. 2024).

Stryker, C. and Holdsworth, J. (2021) Natural language processing. Available at: https://www.ibm.com/think/topics/natural-language-processing (Accessed: 17 Feb. 2025).

University.sopact.com (2024) Open-ended vs. closed-ended questions in survey. Available at: https://university.sopact.com/article/open-ended-vs-closed-ended-questions (Accessed: 17 Nov. 2024).

Visual Studio Code (2023) Documentation for Visual Studio Code. Available at: https://code.visualstudio.com/docs (Accessed: 17 Nov. 2024).

Wartena, C., Slakhorst, W. and Wibbels, M. (2010) 'Selecting keywords for content based recommendation', in *Proceedings of the 19th ACM Conference on Information and Knowledge Management*, Toronto, ON, Canada, 26–30 October 2010, pp. 1533–1536. Available at: https://dl.acm.org/doi/10.1145/1871437.1871665 (Accessed: 17 Feb. 2025).

Shiotsu, Y. (2021) Explore the Google Cloud Natural Language API for NLP work. Available at: https://www.upwork.com/resources/natural-language-processing-and-google-cloud-natural-language-api (Accessed: 17 Feb. 2025).

A APPENDIX: Professional, Legal, Ethical and Social Issues (PLES)

Professional Considerations

To make sure the system is reliable, maintainable, and user-friendly, the Maintainability requirement (NFR-5) provides a high priority on code quality, flexibility, and documentation. Prioritizing user-centered design and keeping to coding standards are important. User data is protected by security standards like password hashing (NFR-1). Furthermore, by putting user satisfaction first, a straightforward, user-friendly experience (NFR-4) shows professional standards. By following these guidelines, the project hopes to create a reliable, efficient program that satisfies user requirements.

Legal Considerations

For this project, legal compliance is important, especially in relation to data protection and user privacy. To ensure that user data is collected, handled, and stored legally, the application must comply with GDPR (NFR-2) as well as following the submitted Heriot-Watt ethical form. Along with a simple privacy policy and terms of use, this involves providing users with clear information on data usage, access, and deletion. To ensure that materials are used appropriately for educational purposes, copyright and licensing compliance are also crucial when integrating external learning resources (FR-3, FR-4). By following legal requirements, these procedures protect both developers and users.

Ethical Considerations

User privacy and recommendation accuracy are given top priority in the project's ethical considerations. Privacy is protected by informing users about the analysis of their data, especially notes (FR-3, FR-4). In line with ethical standards, user consent must be obtained before data analysis begins. The application also aims to remove disadvantages in recommendations by ensuring fair access to a number of educational resources.

Social Considerations

By providing individualized learning resources that satisfy students' academic needs, this application can have a beneficial impact on educational practices (FR-3). Additionally, personalised recommendations encourage lifelong learning and assist students in finding new materials.

B APPENDIX: Project Management

An organized method for finishing the AI-driven note-taking and recommendation system application during the academic year is outlined in the project management plan for this dissertation. It contains a schedule for Semesters 1 and 2, outlining important tasks, due dates, and deliverables.

The project is divided into Planning and Research, as well as followed by Development and Implementation and Testing and Evaluation in Semester 2. This phased approach ensures a solid structure is built before progressing to development, with flexibility in Semester 2 for adjustments.

In order to ensure that essential objectives are achieved, a risk analysis identifies possible obstacles and provides solutions for managing them. With room for more complex activities if time allows, the strategy ensures a safe core of essential aspects.

Project Plan

I have developed a gnat chart to show the project plan of semester 1 and semester 2 as shown in fig 3 and fig 4 (fig 4.1 and fig 4.2)

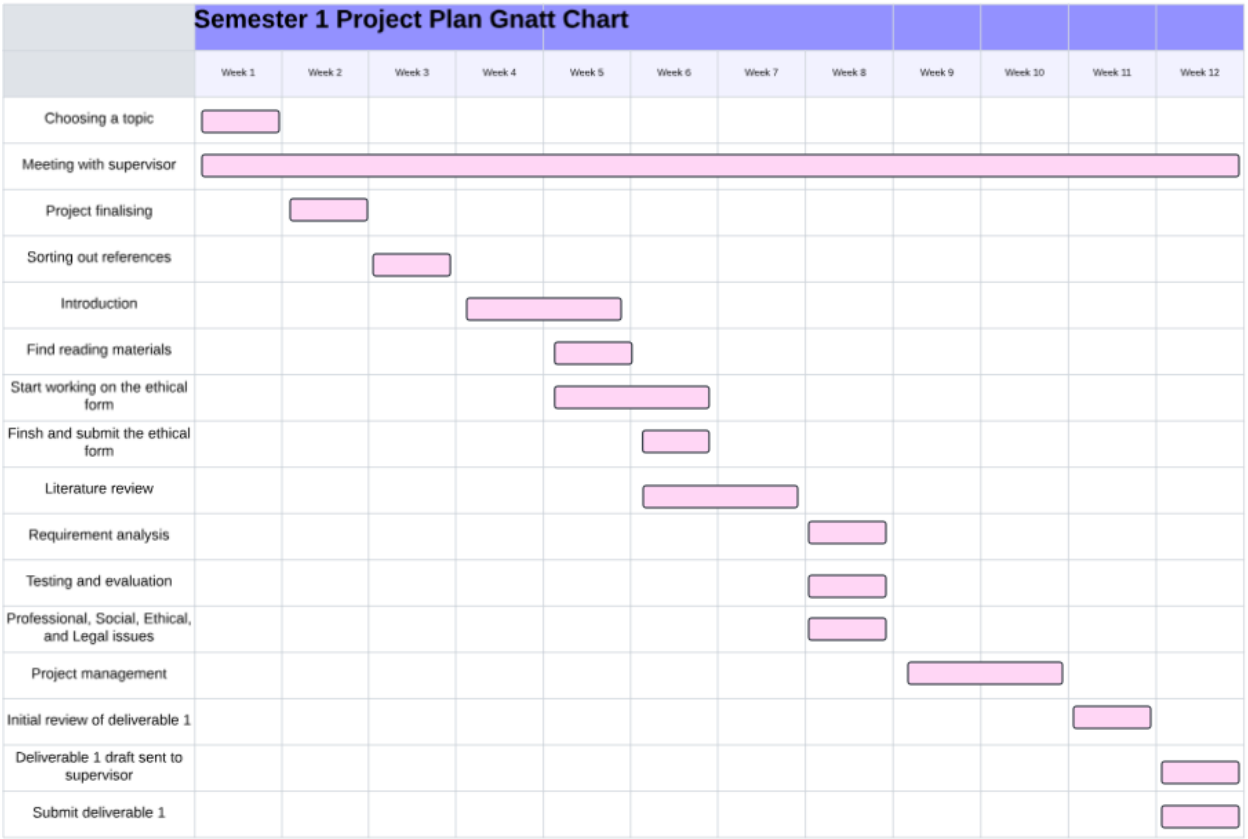


Fig. 7 Project plan of semester 1

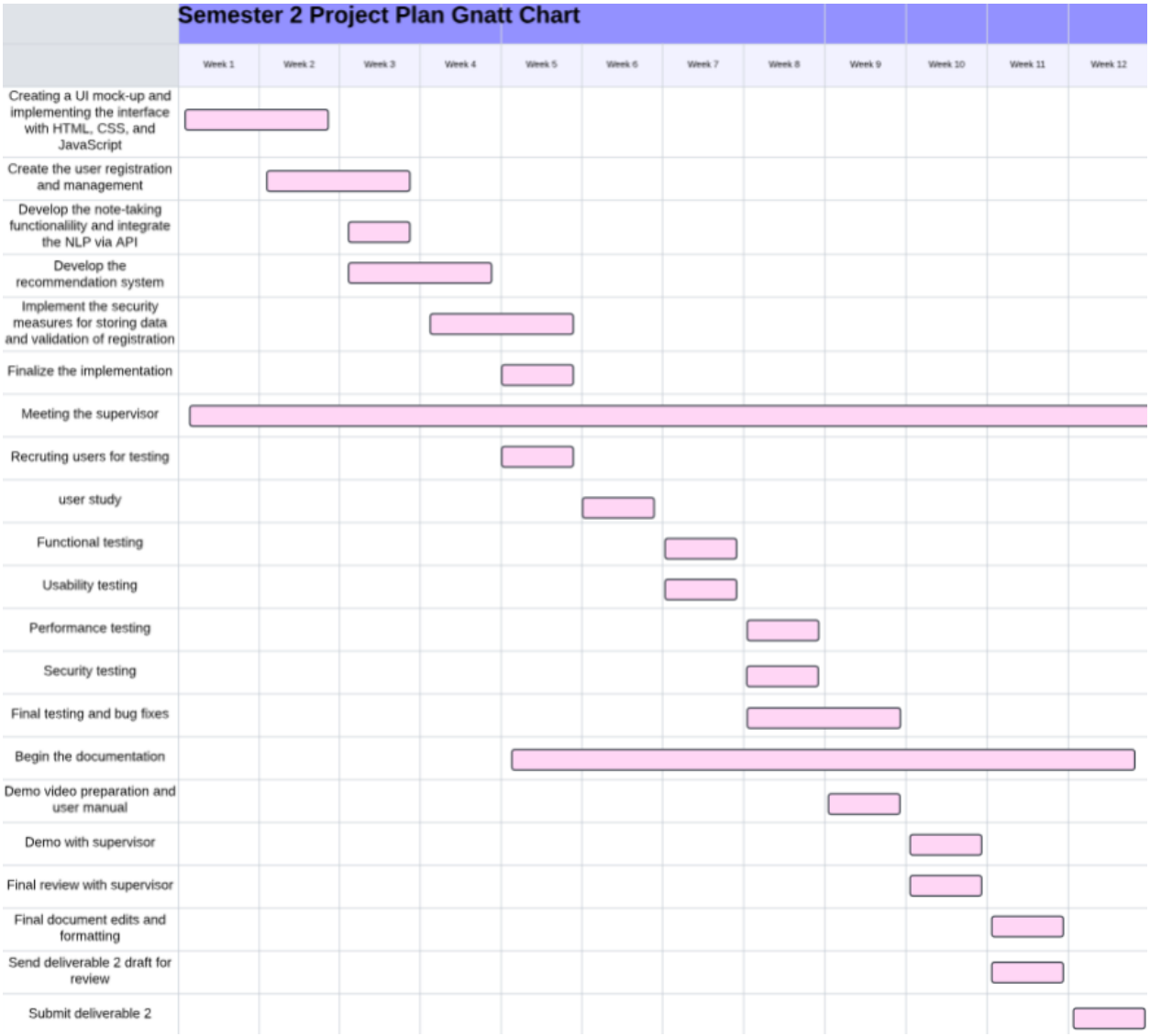


Fig. 8 project plan of semester 2

Risk Management

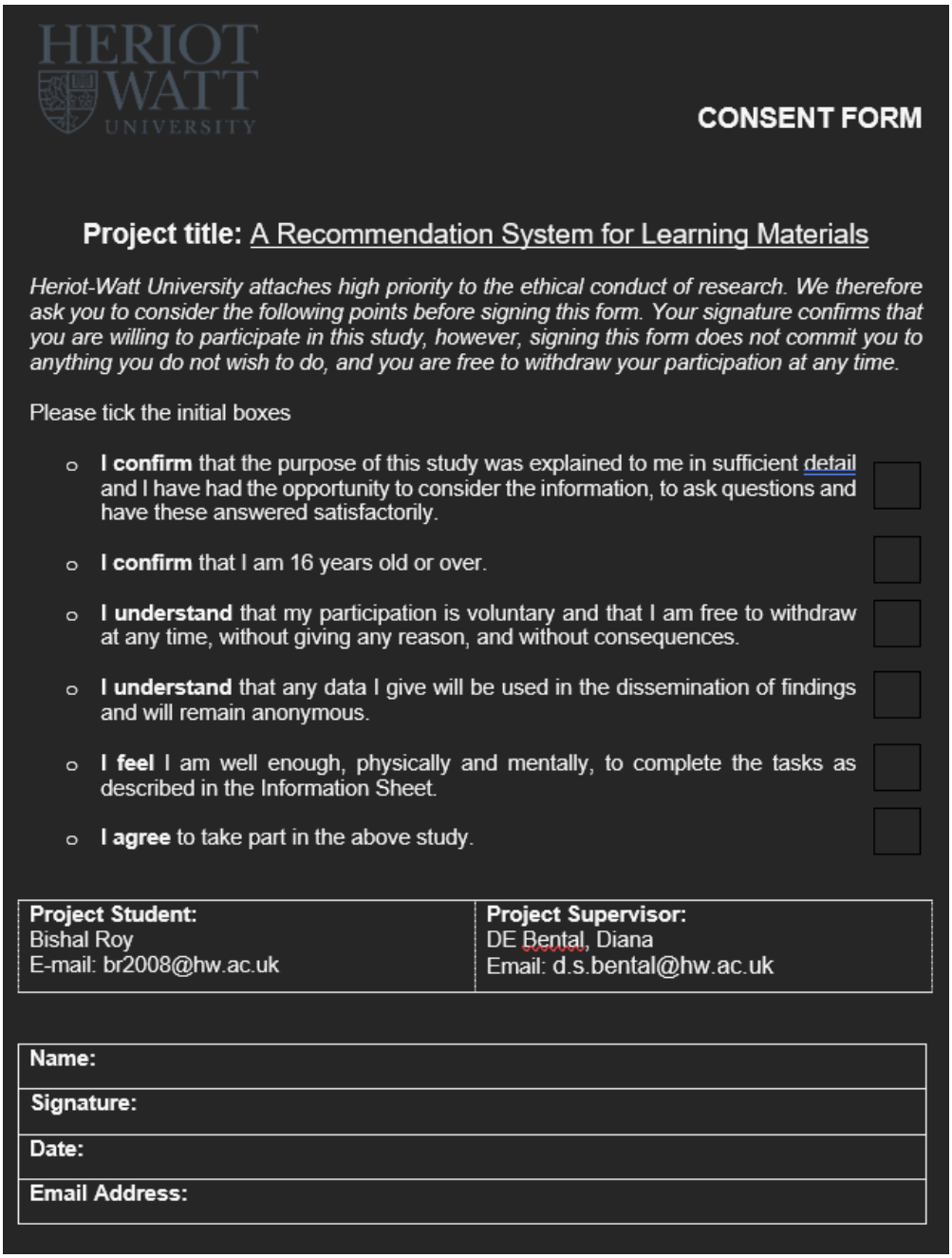
Managing risks is very important in this development to identify and manage risks early as possible. I have created a list of risks where we look at how likely each risk is to happen and how high the impact it could have as shown in table 3.

Risk	Description	Impact	Likelihood	Strategy
Technical Difficulties with NLP API	Issues in integrating or using the external NLP API for analysing notes, resulting in delays or limited functionality.	High	Medium	Choose a reliable NLP API with documentation and support; have a backup API as well.
Time Management Issues	Difficulty balancing project workload with other academic or personal commitments, potentially impacting project progress.	High	High	Create a detailed timetable with achievable milestones; hold regular check-ins with the supervisor to stay on track.
Unforeseen Technical Bugs	Bugs in code during development or testing phases can delay progress and compromise application quality.	Medium	High	Plan time for debugging, conduct continuous testing to identify and address issues early.
Data Privacy Compliance	Risk of non-compliance with data protection regulations (e.g., GDPR) if handling personal information improperly.	High	Medium	pseudonymize user data; review GDPR guidelines, consult supervisor for legal compliance guidance.
Incomplete or Poorly Detailed User Notes	Incomplete or poorly detailed notes from users could result in inaccurate recommendations, affecting the system's effectiveness.	Medium	High	provide guidelines to users on effective note-taking and conduct testing with sample notes to improve recommendation algorithms.
Lack of User Feedback for Testing	Limited access to actual users for usability and effectiveness testing, reducing the validity of evaluation outcomes.	Medium	Medium	Recruit volunteer participants early and consider simulated user inputs for initial testing.

Resource Limitations (Software/Hardware)	Insufficient access to necessary software tools or hardware resources, impacting development or testing capabilities.	Medium	Low	Check software availability early and discuss potential resources with supervisor.
Supervisor Unavailability	Limited availability of the supervisor for guidance, which could affect the timely resolution of issues or feedback on deliverables.	Medium	Low	Schedule regular meetings in advance; prepare specific questions for each meeting to maximize guidance
Security Vulnerabilities	Potential for data security issues.	High	Medium	Implement robust security protocols and conduct vulnerability testing
Testing Data Quality Issues	Poor quality or insufficient quantity of test data could impact the evaluation of the recommendation system's effectiveness.	Medium	Medium	Source good quality, relevant test data early and simulate realistic test cases to ensure valid evaluations.
Final Documentation Delays	Delays in completing final documentation due to extended development or testing periods, impacting submission timelines.	High	Medium	Start documentation early and documenting each phase.
Complexity of Recommendation Algorithm	The recommendation algorithm may be too complex to develop within the project timeline, especially if advanced machine learning models are required.	High	Medium	Start with a simpler algorithm and gradually add complexity if time allows it.
Poor Recommendation Accuracy	The recommendation system may not accurately match resources to the user's notes, leading to irrelevant or unsatisfactory recommendations.	High	High	To improve algorithms, use correctly labelled training data and carry out initial testing to evaluate accuracy and relevancy.

Table 3. Risk Management Table

C APPENDIX: The Consent Form



HERIOT WATT UNIVERSITY

CONSENT FORM

Project title: A Recommendation System for Learning Materials

Heriot-Watt University attaches high priority to the ethical conduct of research. We therefore ask you to consider the following points before signing this form. Your signature confirms that you are willing to participate in this study, however, signing this form does not commit you to anything you do not wish to do, and you are free to withdraw your participation at any time.

Please tick the initial boxes

- **I confirm** that the purpose of this study was explained to me in sufficient detail and I have had the opportunity to consider the information, to ask questions and have these answered satisfactorily. ☐
- **I confirm** that I am 16 years old or over. ☐
- **I understand** that my participation is voluntary and that I am free to withdraw at any time, without giving any reason, and without consequences. ☐
- **I understand** that any data I give will be used in the dissemination of findings and will remain anonymous. ☐
- **I feel** I am well enough, physically and mentally, to complete the tasks as described in the Information Sheet. ☐
- **I agree** to take part in the above study. ☐

Project Student:	**Project Supervisor:**
Bishal Roy E-mail: br2008@hw.ac.uk	DE Bental, Diana Email: d.s.bental@hw.ac.uk

Name:
Signature:
Date:
Email Address:

Fig. 9 Consent Form

D APPENDIX: The Survey

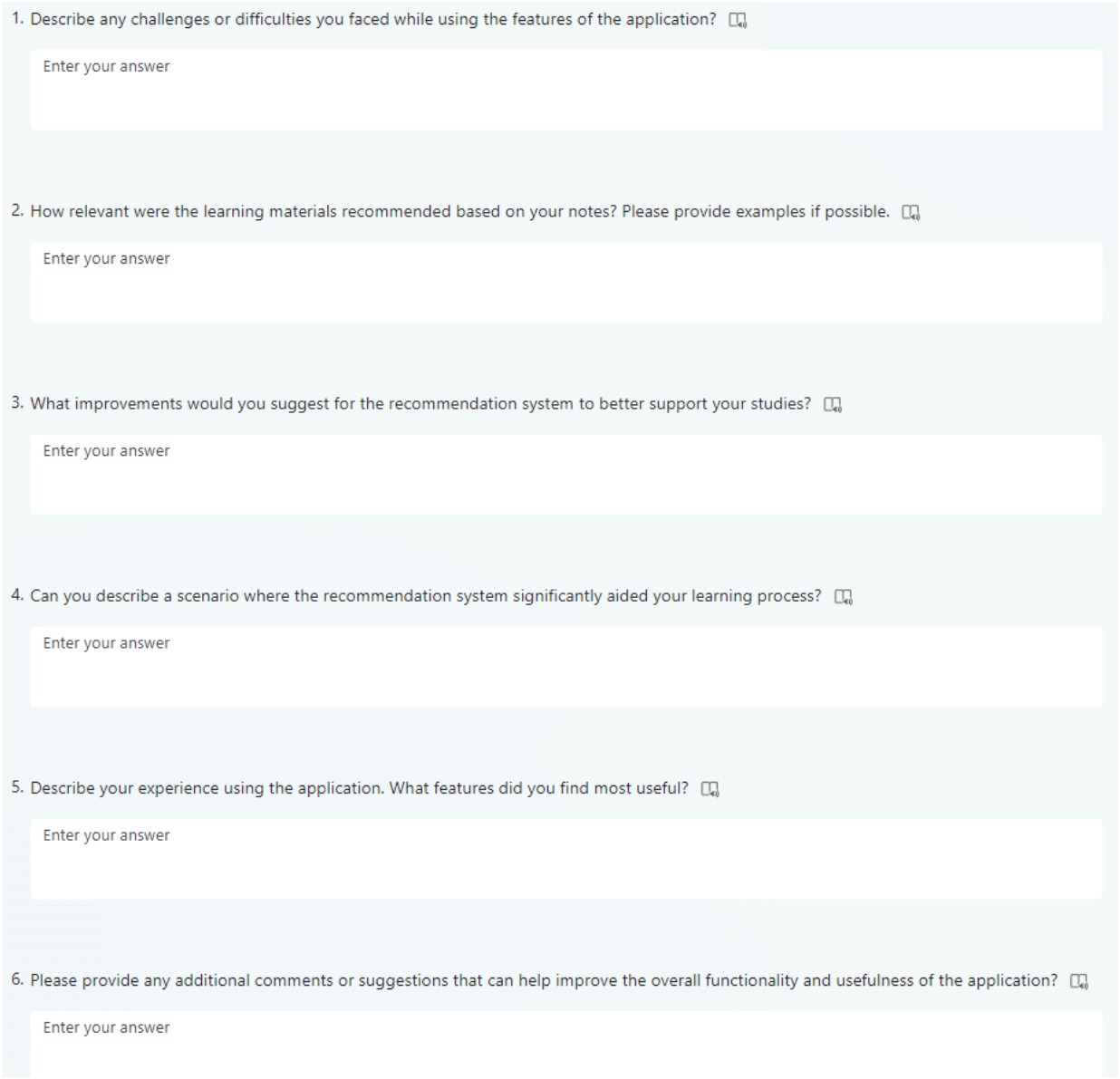


Fig. 10.1 Survey Form

7. How would you describe the ease of navigating through the application? Were there any areas where you felt improvements could be made for better usability?

Enter your answer

8. How accessible do you find the application? Are there any features that could be improved to assist users with disabilities?

Enter your answer

9. Performance and Reliability

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The application consistently works as expected without crashing or losing data.	○	○	○	○	○
The system maintains a consistent performance without significant delays or lag.	○	○	○	○	○

10. Efficiency

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The application helps improve my efficiency in accessing study materials.	○	○	○	○	○
Using the application helped me save time when looking for study materials.	○	○	○	○	○

Fig. 10.2 Survey Form

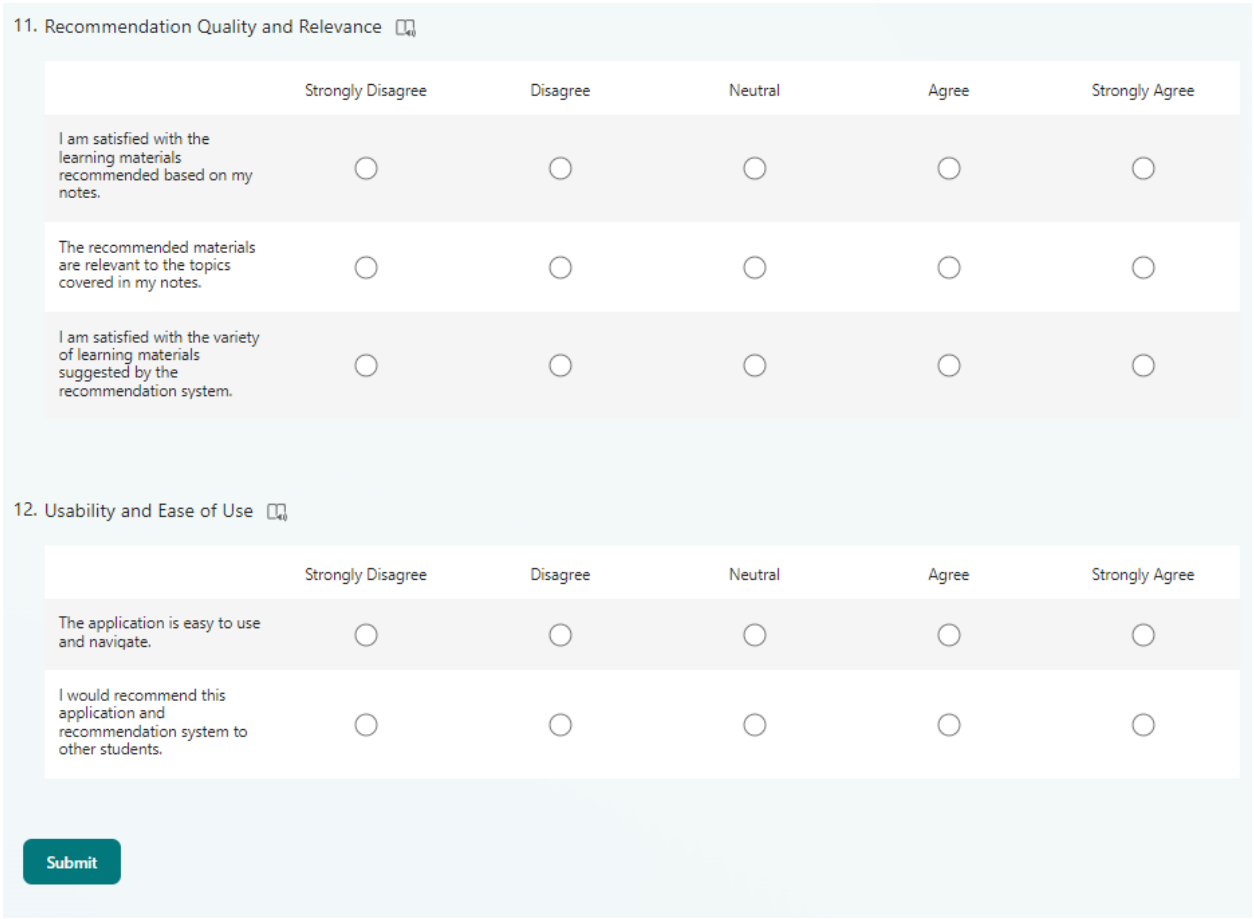
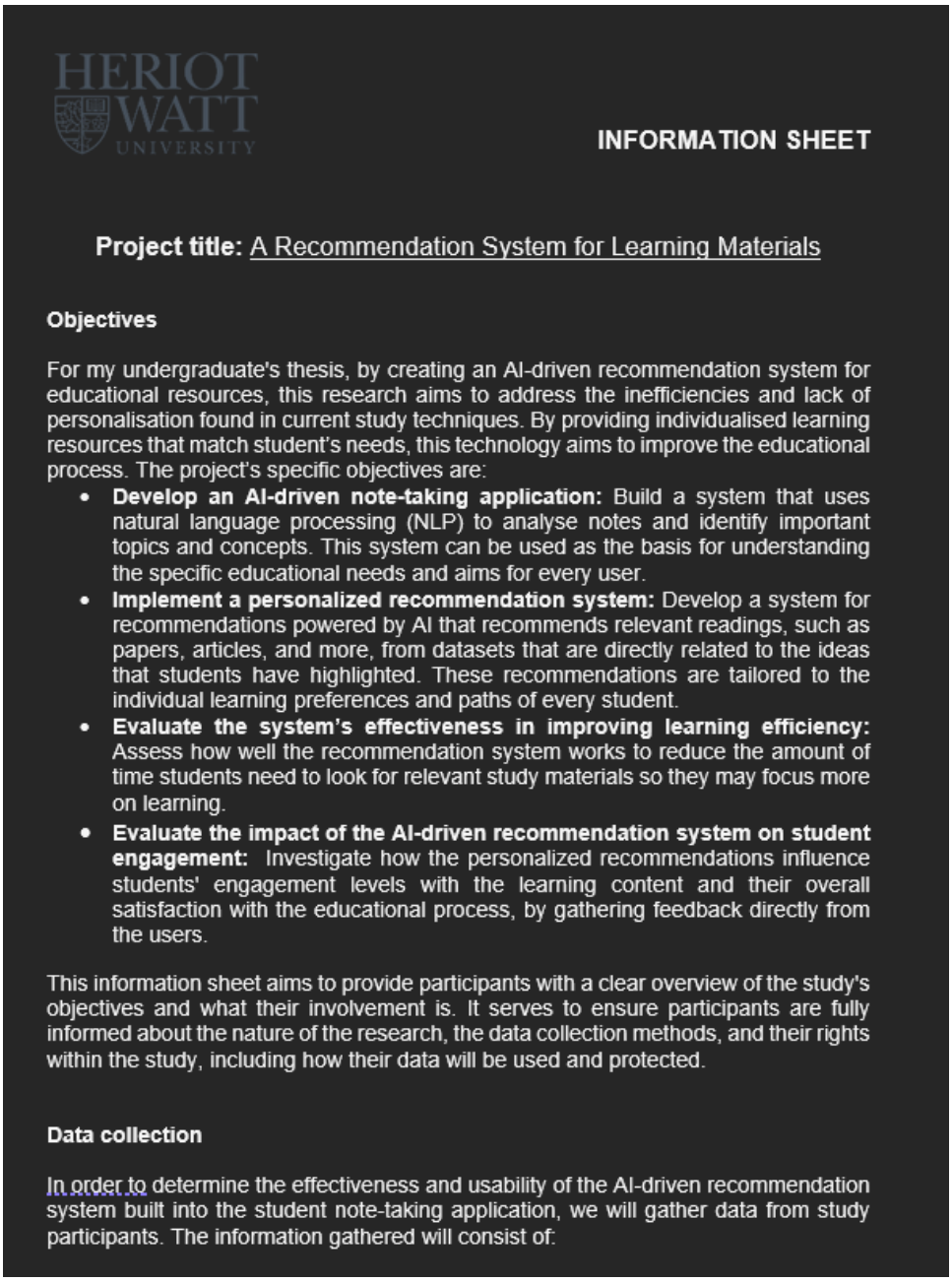


Fig. 10.3 Survey Form

E APPENDIX: The Information Sheet



HERIOT WATT UNIVERSITY

INFORMATION SHEET

Project title: A Recommendation System for Learning Materials

Objectives

For my undergraduate's thesis, by creating an AI-driven recommendation system for educational resources, this research aims to address the inefficiencies and lack of personalisation found in current study techniques. By providing individualised learning resources that match student's needs, this technology aims to improve the educational process. The project's specific objectives are:

- **Develop an AI-driven note-taking application:** Build a system that uses natural language processing (NLP) to analyse notes and identify important topics and concepts. This system can be used as the basis for understanding the specific educational needs and aims for every user.
- **Implement a personalized recommendation system:** Develop a system for recommendations powered by AI that recommends relevant readings, such as papers, articles, and more, from datasets that are directly related to the ideas that students have highlighted. These recommendations are tailored to the individual learning preferences and paths of every student.
- **Evaluate the system's effectiveness in improving learning efficiency:** Assess how well the recommendation system works to reduce the amount of time students need to look for relevant study materials so they may focus more on learning.
- **Evaluate the impact of the AI-driven recommendation system on student engagement:** Investigate how the personalized recommendations influence students' engagement levels with the learning content and their overall satisfaction with the educational process, by gathering feedback directly from the users.

This information sheet aims to provide participants with a clear overview of the study's objectives and what their involvement is. It serves to ensure participants are fully informed about the nature of the research, the data collection methods, and their rights within the study, including how their data will be used and protected.

Data collection

In order to determine the effectiveness and usability of the AI-driven recommendation system built into the student note-taking application, we will gather data from study participants. The information gathered will consist of:

Fig. 11.1 Information Sheet

- Survey Responses: Participants will answer questions about their experiences using the application, including feedback on the usability, relevance of recommendations, and overall satisfaction.
- Demographic Information: Basic demographic data such as age and study level (e.g., university) will be collected to understand how different groups engage with the system. Participants must be 16 years or older to participate.
- Application Usage Data: Data regarding how participants interact with the note-takin and recommendation features will be collected to analyse the system's effectiveness.

Collected data will be pseudo-anonymized. This means that any identifying information, such as names, will be replaced with pseudonyms or participants numbers to protect participants' identities. The key linking pseudonyms to participants will be stored securely and separately to reduce the risk of identification. Also, all data will be handled in compliance with data protection regulations like GDPR and participants will be informed about how they can access or remove their data at any time if they choose to withdraw from the study.

Experiment Protocol

Participants in this study will interact with a note-taking application that includes an AI-driven recommendation system for learning materials. After using the application, they will be asked to complete a survey, which is expected to take approximately 10-15 minutes to complete. This survey includes Likert scale questions and open-ended questions that assess the application's usability, the relevance of the recommended learning materials, the efficiency of accessing study resources, and the overall impact on engagement.

The procedure:

- **Account Registration (Approx. up to 5 minutes):** Participants will begin by creating an account on the application. During registration, They will be asked to provide basic information such as their name and field of study.
- **Note-taking Session (Approx. up to 20 minutes):** Participants will use the application to take notes based on a provided educational material. This could involve summarizing key concepts, typing out the main ideas, or other note-taking activities. The system will analyse these notes to identify relevant topics for recommending learning materials.
- **Receiving and Interacting with Recommended Learning Materials (Approx. up to 10 minutes):** Based on the notes taken, and the user's information, the application will generate a list of recommended learning materials tailored to the participant's subject of study and notes content. Participants are expected to review and engage with these recommended materials to assess their relevance and usefulness.
- **Completion of Survey (Approx. up to 15 minutes**): After interacting with the recommended materials, participants will complete a survey. This survey includes Likert scale and open-ended questions to gather feedback on the application's usability, the quality and relevance of the recommendations, and the impact on their study efficiency and engagement.

Fig. 11.2 Information Sheet

To ensure participants' safety:

- Participants are encouraged to use the application in a safe and comfortable environment, ideally while seated. Regular breaks are advised to prevent eye strain and maintain good posture.
- Personal information collected for registration will be pseudonymized to protect participants' identities. Participants are also informed of their right to withdraw at any time, with the guarantee that all information gathered will be removed at their request.

If you decide to participate:

- You will be required to read and sign a consent form, confirming your understanding of the study and your rights as a participant. Also, you will be required to read this information sheet carefully.
- Follow each of the tasks as outlined above.

Compensation

No compensation will be provided for participation in this study. Participation is entirely voluntary, and participants may withdraw at any time without consequences. This research relies on voluntary input to gather unbiased feedback for improving the AI-driven recommendation system.

Withdraw

Anytime, for any reason, and without facing any penalties, participants are free to leave this study. Withdrawing from the study will not have any impact on your grades, employment status, pay, or status as a student. Your participation is completely optional, and you can stop at any time without facing any consequences.

If you choose to participate, please make sure you read carefully the consent form and this information sheet and sign the consent form before starting the study. By signing the consent form, you confirm that you understand the purpose of the study, what your participation involves, and your rights as a participant, including your right to withdraw at any time.

Once you sign the consent form, you will be able to proceed with the study tasks. This includes using the application, interacting with the recommendation system, and completing the survey about your experience. Your signed consent form will be securely stored and treated with confidentiality. If at any point after signing the form you wish to withdraw, simply contact the researcher using the information provided, and your data will be removed from the study upon request.

For additional details or to request the deletion of your data, you can reach out to br2008@hw.ac.uk or you can contact the project supervisor via d.s.bental@hw.ac.uk

Fig.11.3 Information Sheet

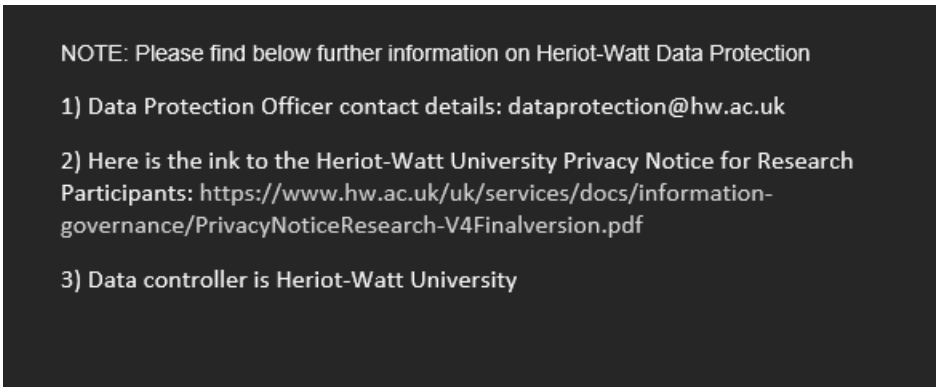


Fig. 11.4 Information Sheet

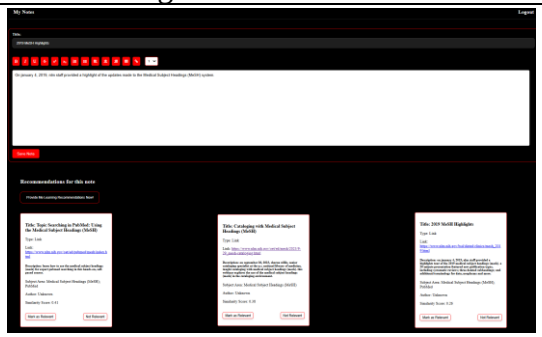
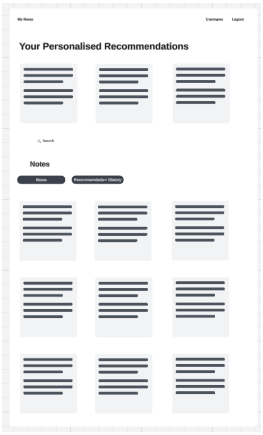
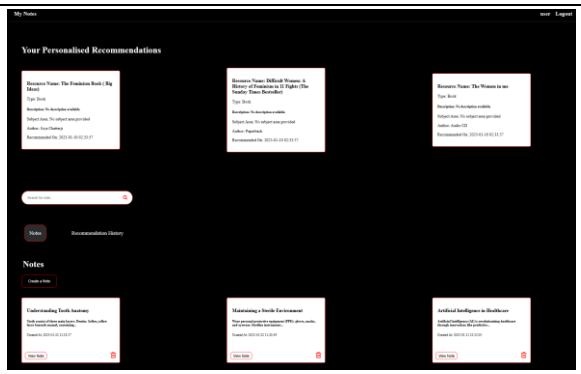
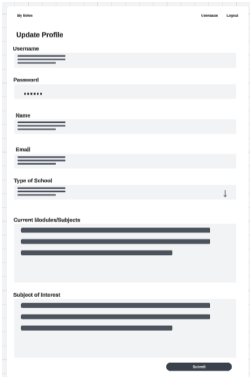
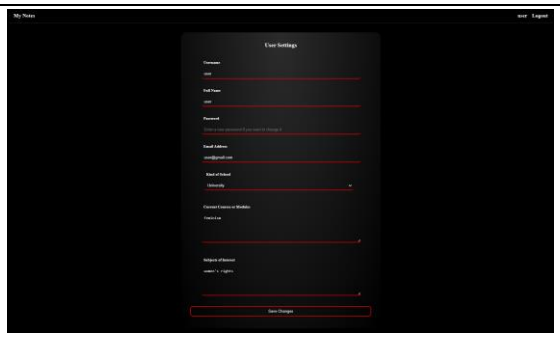
F APPENDIX: Content-Based Recommendation System Code (With Comments)

```
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer
from google_nlp_api import note_analyser # Custom module for keyword extraction

# Load the dataset containing learning materials
filePath = 'combined_learning_materials.csv'
data = pd.read_csv(filePath)

# Combine ‘Title’ and ‘Description’ into a single feature for text analysis
data['Combined_Features'] = data['Title'].fillna('') + '' +
data['Description'].fillna('')

# Initialise TF-IDF Vectorization to convert text into numerical vectors
while removing English stop words
tfidf = TfidfVectorizer(stop_words='english')
# Transform the combined features into a TF-IDF matrix
tfidf_matrix = tfidf.fit_transform(data['Combined_Features'])

# Compute the cosine similarity matrix for all learning materials
cosineSimilarity = cosine_similarity(tfidf_matrix)

def recommend_learning_materials(title, content):
    keywordsTitle = note_analyser(title) # Title of the note
    keywordsContent = note_analyser(content) # Content of the note
    # Combine keywords of keywords from title and content of note
    keywordsCombined = ' '.join(keywordsTitle + keywordsContent)

    # Transform user input keywords into a TF-IDF vector
    user_tfidf = tfidf.transform([keywordsCombined])

    # Compute cosine similarity between user input and learning materials
    userSimilarityScores = cosine_similarity(user_tfidf, tfidf_matrix)

    # Convert similarity scores into a Pandas Series indexed by material
```

```
titles
    userSimilaritySeries = pd.Series(userSimilarityScores[0],
index=data['Title'])

    # Remove duplicated material titles, keeping the first occurrence
    noDuplicatedSeries =
userSimilaritySeries[userSimilaritySeries.index.duplicated(keep='first') ==
False]

    # Select top 3 of most relevant learning materials
    top3 = noDuplicatedSeries.sort_values(ascending=False).head(3)

    # Recommendation list with details
    recommendations = []
    for title, score in top3.items():
        row = data.loc[data['Title'] == title].iloc[0] # Retrieve the first
matching row

        # Determine if the material is a book or an online resource (with URL)
        resourceType = 'Book' if pd.isna(row['URL']) else 'Link'
        # Create a dictionary with material details and add it to the
recommendation system
        recommendations.append({
            'title': title,
            'description': row['Description'] if not pd.isna(row['Description'])
else "No description available",
            'url': row['URL'] if not pd.isna(row['URL']) else "No link
available",
            'subject': row['Subject'] if not pd.isna(row['Subject']) else "No
subject area provided",
            'author': row['Author'] if not pd.isna(row['Author']) else "No
author",
            'similarity': score,
            'type': resourceType
        })
    return recommendations
```

Code Listing 4. Python full content-based recommendation system code

G APPENDIX: Mockups and Visual Design

	Mockups	Visual Design
Editing a note page with recommended learning materials.		
Notes page with all the created notes and personalised recommendations at the top of the page based on user's current subjects and subjects of interest.		
Page for the user to update their information.		

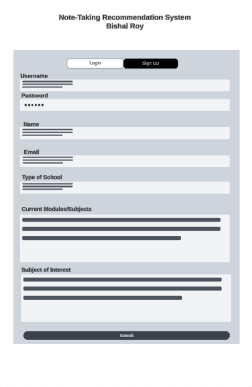
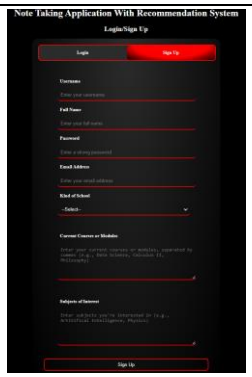
Page to sign up for the note-taking application.		

Table 4. Mockup vs Visual Design Table

H APPENDIX: Experiment Results

Describe any challenges or difficulties you faced while using the features of the application?	1) pressing ok when looking at relevancy of recommendations 2) no challenges 3) The UI of the application has some challenges such as you can't see if you have entered information into an area if you unselect from that text box. 4) There were no difficulties I faced when testing the web app 5) When adding a note, it popping up at the top was a bit annoying 6) UI/UX, it's not protected against SQL injections 7) having to the relevance of each individual Recommendations was redundant and tedious 8) Not enough relevant information 9) Some of text was a bit difficult to see
How relevant were the learning materials recommended based on your notes? Please provide examples if possible.	1) relatively relevant 2) quite relevant except for my second task 3) Very relevant, 2/3 of the recommended material was relevant every time I created a note 4) they were pretty relevant to my learning materials where when I added notes about the laws of thermodynamics it provided me with notes talking about thermodynamics 5) Very relevant, all related to the notes I put in 6) seemed to be quite relevant for the simple prompt it got 7) overall, the materials provided were relevant roughly 80% of the time 8) 4\10 9) Really relevant, provided results like a search engine, even ones without the same words
What improvements would you suggest for the recommendation system to better support your studies?	1) not having to press ok every time 2) larger dataset 3) give more models for precise recommendation 4) for it to become more precise 5) A find more options to get more 3 6) UI/UX, make it a bit cleaner (sometimes less is more), protect it from random tags within text box from breaking it 7) Make less redundant. 8) more sufficient data or recommendations

	9) Having all icons available at top menu bar
Can you describe a scenario where the recommendation system significantly aided your learning process?	1) when I need to find more sources fast 2) I cannot 3) it showed me a new avenue in the subject I was searching for 4) it helped me find materials faster 5) it made it quicker and easier to find relevant learning materials compared to googling it 6) well, I suppose if I give enough information for a specific area it seems like it would aid it 7) Made it easier to find appropriate materials 8) helped me find materials quicker 9) Provide recipe recommendations when writing notes about making a meal
Describe your experience using the application. What features did you find most useful?	- was pretty good the recommendations are useful - the recommendation part - the save noted feature - how easy it is to use and how it provides a similarity score on each recommended note. - It was easy to use, very clear and finding relevant information was easy - it worked mostly I guess that's good - The recommendation system - navigating the interface when creating a note 9) The relevant results that it returned
Please provide any additional comments or suggestions that can help improve the overall functionality and usefulness of the application?	1) good stuff bishal I'm proud 2) more recommendations 3) clearer, in the recommendations and how to access the resources 4) add a filter to sort out the recommendation materials 5) More options for relevant reading 6) improve UI and UX 7) Make less redundant 8) provide more than 3 suggestions 9) Just small refinements to make it more user friendly and give more examples of results
How would you	1) pretty easy

describe the ease of navigating through the application? Were there any areas where you felt improvements could be made for better usability?	2) easy to use but when you save a note it gives you a pop up 3) it was ok, maybe improve the flow of the applications 4) better structure of page 5) When submitting a note make it so it pops up and is easier to continue 6) Visibility not the best due to design choices for front end 7) Navigation was simple and concise, no improvement required 8) it was pretty good 9) spread across the screen so you can see all the options
How accessible do you find the application? Are there any features that could be improved to assist users with disabilities?	1) Having to press ok on the conformation, might restrict disabled people 2) It is good maybe for the colorblind or a light mode 3) Very accessible 4) Add light mode 5) N/A for me 6) Not very, at least there didn't seem to be any at first glance 7) Improved colour scheme 8) Add voice control 9) Different coloured writing or text format, for example, making it bolder

Table 5. Open Questions Answer Results Table

9. Performance and Reliability

9 Responses

ID ↑	Name	Responses	
		The application consistently works as expected without crashing or losing data.	The system maintains a consistent performance without significant delays or lag.
1	Bishal Roy	Strongly Agree	Strongly Agree
2	Bishal Roy	Agree	Agree
3	Bishal Roy	Strongly Agree	Neutral
4	Bishal Roy	Strongly Agree	Strongly Agree
5	Bishal Roy	Strongly Agree	Agree
6	Bishal Roy	Neutral	Neutral
7	Bishal Roy	Agree	Strongly Agree
8	Bishal Roy	Agree	Agree
9	Bishal Roy	Strongly Agree	Strongly Agree

Fig. 12.1 Likert Results Table

10. Efficiency

9 Responses

ID ↑	Name	Responses	
		The application helps improve my efficiency in accessing study materials.	Using the application helped me save time when looking for study materials.
1	Bishal Roy	Agree	Agree
2	Bishal Roy	Neutral	Agree
3	Bishal Roy	Agree	Agree
4	Bishal Roy	Strongly Agree	Strongly Agree
5	Bishal Roy	Strongly Agree	Strongly Agree
6	Bishal Roy	Agree	Neutral
7	Bishal Roy	Agree	Agree
8	Bishal Roy	Agree	Neutral
9	Bishal Roy	Strongly Agree	Strongly Agree

Fig. 12.2 Likert Results Table

11. Recommendation Quality and Relevance

9 Responses

ID ↑	Name	Responses		
		I am satisfied with the learning materials recommended based on my notes.	The recommended materials are relevant to the topics covered in my notes.	I am satisfied with the variety of learning materials suggested by the recommendation system.
1	Bishal Roy	Agree	Agree	Agree
2	Bishal Roy	Agree	Agree	Agree
3	Bishal Roy	Agree	Agree	Agree
4	Bishal Roy	Agree	Agree	Strongly Agree
5	Bishal Roy	Strongly Agree	Agree	Strongly Agree
6	Bishal Roy	Agree	Agree	Agree
7	Bishal Roy	Agree	Agree	Neutral
8	Bishal Roy	Neutral	Agree	Neutral
9	Bishal Roy	Strongly Agree	Strongly Agree	Strongly Agree

Fig. 12.3 Likert Results Table

12. Usability and Ease of Use

9 Responses

ID ↑	Name	Responses	
		The application is easy to use and navigate.	I would recommend this application and recommendation system to other students.
1	Bishal Roy	Agree	Agree
2	Bishal Roy	Agree	Neutral
3	Bishal Roy	Agree	Agree
4	Bishal Roy	Strongly Agree	Strongly Agree
5	Bishal Roy	Strongly Agree	Strongly Agree
6	Bishal Roy	Disagree	Neutral
7	Bishal Roy	Strongly Agree	Neutral
8	Bishal Roy	Agree	Agree
9	Bishal Roy	Agree	Agree

Fig. 12.4 Likert Results Table

I APPENDIX: Database Schema

Database schema for user notes:

```
CREATE TABLE `notes` (
  `noteid` int(11) NOT NULL,
  `userid` int(11) NOT NULL,
  `title` varchar(255) NOT NULL,
  `content` longtext NOT NULL,
  `createdAt` timestamp NOT NULL DEFAULT current_timestamp(),
  `updatedAt` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE
current_timestamp(),
  `is_archived` tinyint(1) DEFAULT 0
);
```

Code Listing 5. SQL Code to create notes database table

Database schema for previous recommendations:

```
CREATE TABLE `recommendations` (
  `recommendationid` int(11) NOT NULL,
  `noteid` int(11) DEFAULT NULL,
  `userid` int(11) NOT NULL,
  `resourceName` varchar(255) NOT NULL,
  `resourceURL` varchar(2083) DEFAULT NULL,
  `description` text DEFAULT NULL,
  `subjectArea` varchar(255) DEFAULT NULL,
);
```

Code Listing 6. SQL Code to create recommendations database table

Database schema for user's information:

```
CREATE TABLE `users` (
  `userid` int(11) NOT NULL,
  `username` varchar(50) NOT NULL,
  `name` varchar(100) NOT NULL,
  `password` varchar(255) NOT NULL,
  `email` varchar(100) NOT NULL,
  `kindOfSchool` enum('University','College','High School','Other') NOT NULL,
  `currentCM` text NOT NULL,
```

```
 `subOfInt` text NOT NULL,
`createdAt` timestamp NOT NULL DEFAULT current_timestamp());
```

Code Listing 7. SQL Code to create users database table

J APPENDIX: Python Flask Precision Calculation Code with Comments

```
# Define new route that only accepts GET requests
@app.route('/precision_evaluation', methods=['GET'])
def precision_evaluation():
    # Establish a connection to the MySQL database
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

    # Fetch all relevance feedback data from 'feedbackRelevance' table
    cursor.execute('SELECT isRelevant FROM feedbackRelevance')
    feedback = cursor.fetchall()

   # If not feedback data available, return an error message
    if note feedback:
        return jsonify({"error": "No feedback available to calculate
precision."})

    # Extract the 'isRelevant' values from the fetched feedback
    isRelevant = [relevance['isRelevant'] for relevance in feedback]

    totalItemsRecommend = len(isRelevant) # Total number of feedback entries
    relevantRecommend = sum(isRelevant) # Number of items marked as relevant
    # Calculate precision: (relevant recommendations) / (total recommendations)
    precision = relevantRecommend / totalItemsRecommend if totalItemsRecommend > 0
else 0

    return jsonify({
        "precision": precision,
        "relevant_items": relevantRecommend,
        "total_items_recommended": totalItemsRecommend)} # Return calculation
```

Code Listing 8. Python Flask precision calculation function with comments

K APPENDIX: NLP Integration Code with Comments

```
import os
from google.cloud import language_v1

# Set up Google NLP Cloud API credentials
keypath = "API KEY"
os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = keyPath
print("Environment variable set for credentials:",
os.environ["GOOGLE_APPLICATION_CREDENTIALS"])

def note_analyser(noteContent):
    # Initialse the Google Cloud NLP API client
    client = language_v1.LanguageServiceClient()

    # Create a document object for text analysis
    document = language_v1.Document(
        content=noteContent, # Input text for analysis
        type=language_v1.Document.Type.PLAIN_TEXT # tell us that the input is plan text
    )

    try:
        # Perform entity analysis to extract meaningful keywords form the note
        response = client.analyze_entities(document=document)

        keywords = [] # list to store extracted keywords
        for entity in response.entities:
            # Filter entities based on their type to ensure relevant recommendations
            if entity.type_ in [
                language_v1.Entity.Type.ORGANIZATION,  # e.g., Universities,
institutions
                language_v1.Entity.Type.LOCATION,     # e.g., educational regions
                language_v1.Entity.Type.EVENT,        # e.g., conferences, workshops
```

```
                language_v1.Entity.Type.WORK_OF_ART,  # e.g., books, publications
                language_v1.Entity.Type.CONSUMER_GOOD,  # e.g., learning tools
                language_v1.Entity.Type.OTHER         # Catch-all for other terms
            ]:
                keywords.append(entity.name) # Add extracted entity name to the list

        return list(set(keywords)) # Return unique keywords to avoid duplicates
    except Exception as e:
        print("Error during analysis:", e)
        return []
```

Code Listing 9. Python Google's NLP API Integration code with comments

L APPENDIX: Example note from user

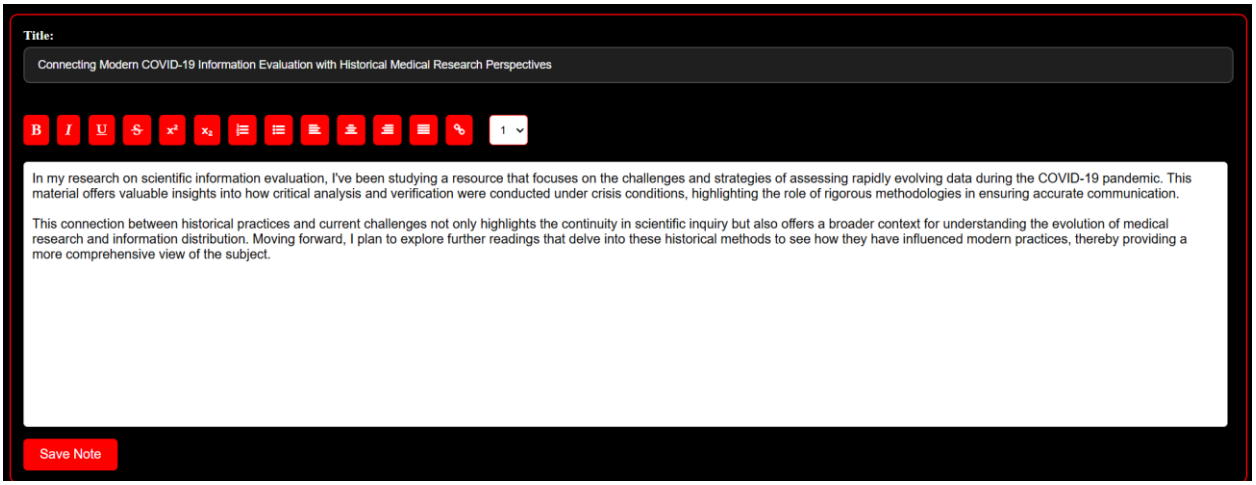


Fig. 13 Example note input from user

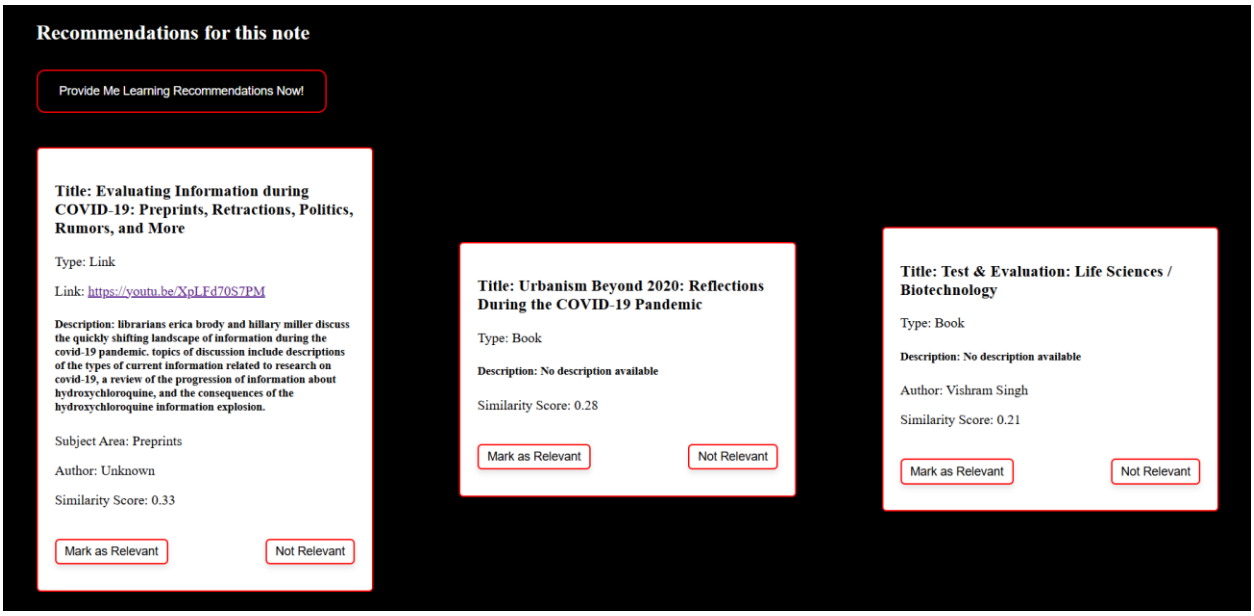


Fig. 14 Example note recommendation output from user

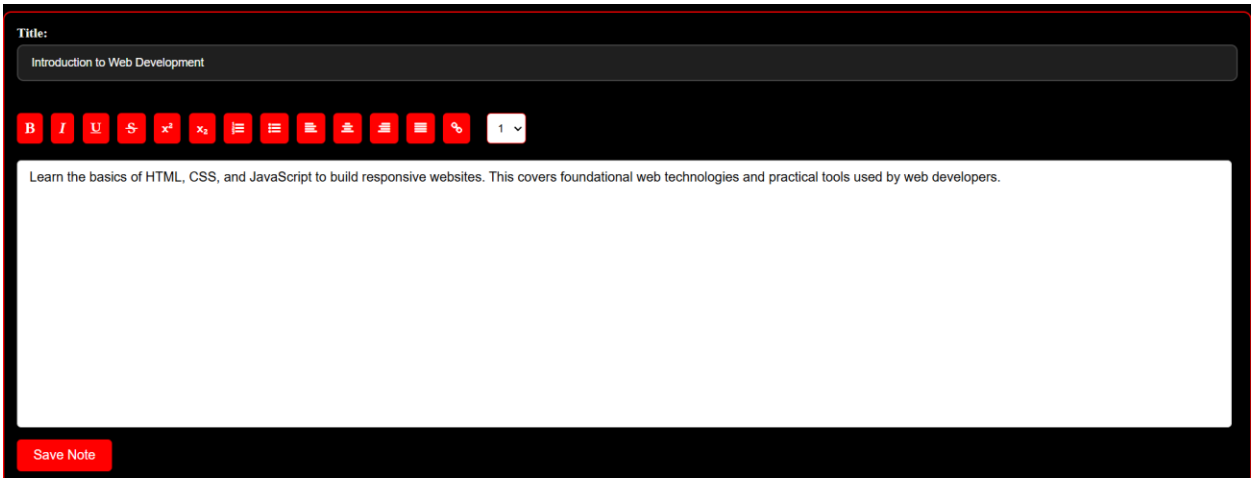


Fig. 15 Example note input from another user

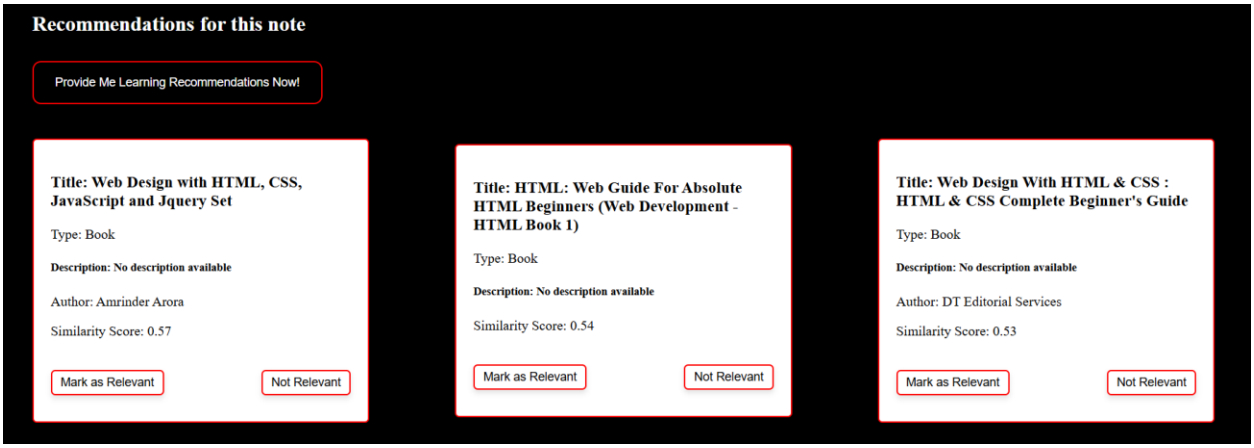


Fig. 16 Example note recommendation output from another user

M APPENDIX: Testing Plan

Tasks:

1) Register a new user with invalid details
    a. Expected: User should be able to see an error registration message above the login/registration form
2) Register a new user with valid details
    a. Expected: User should be directed to the login page and see a successful registration message above the login/registration form
3) Log in with invalid credentials
    a. Expected: error message should be displayed as incorrect username/password
4) Log in with valid credentials
    a. Expected: The user should be redirected to their notes page
5) Log out of the application
    a. Expected: The user should be redirected to the login page

6) Create 2 or 3 notes new note with a title and content then press the save button to save the note
    a. Expected: The note should be saved and displayed on their notes page
7) Edit an existing note and save the changes
    a. Expected: the notes should be updated and saved, and it should show its changes
8) Delete a note by clicking the delete button
    a. Expected: the note should be removed from their notes page and the database and everything related to that note as well

9) Visit one of your saved notes and then click on the recommend button to recommend you learning materials
    a. Expected: The recommendation system should recommend you 3 learning materials relevant to the note based on the extracted keywords from the note
10) Click on “Relevant” or “Not Relevant” on each of the recommended learning materials provided
    a. Expected: The database should update with the user input telling us which recommended material was relevant or note

11) Search for a previously recommended material using its title or description.
    a. Expected: Matching previous recommendations should be displayed as the user types in the search box.
12) Search with no matching results.
    a. Expected: A "No recommendations Found!" message should be displayed.
13) Clear the search input.
    a. Expected: All recommendations should be displayed again.