Journal of Mathematics and Computing Science, 11 (1): 88-101, 2025
Copyright © UiTM Press
eISSN: 0128-0767

STUDYMATE: A Centralised Study Note Web Application Enhanced by LLM for Technical Majors

Chow Ching Huey[1]* and Khor Jia Yun [2]

[1,2]School of Computing, INTI International College Penang, 1-Z, Lebuh Bukit Jambul, Bukit Jambul, 11900 Bayan Lepas, Pulau Pinang, Malaysia

Authors' email: chinghuey07chow@gmail.com[1]* and karen.khor@newinti.edu.my[2]

*Corresponding author

Received 27 March 2025; Received in revised 21 April 2025; Accepted 16 June 2025
Available online 27 June 2025
DOI: https://doi.org/10.24191/jmcs.v11i1.8100

Abstract: Cognitive overload presents a considerable obstacle for self-directed learners, who are required to manage and assimilate extensive amounts of study materials from many sources. StudyMate, a centralised web-based program, aims to lessen this difficulty by optimising the self-study process. The system incorporates Large Language Models (LLMs) to analyse submitted documents and offer intelligent study support, including summarisation, key point extraction, and flashcard generation, thereby alleviating the cognitive burden of digesting extensive information. StudyMate offers features including organised note management, an intuitive user interface, annotation capabilities, and a comprehensive text editor to enhance organisation, usability, and user engagement. This study concentrates on students in technical disciplines (IT, Computer Science, and Electrical and Electronics Engineering), as StudyMate is designed for self-directed learners, and the creator is well-acquainted with their specific academic requirements and obstacles. The system, developed under the Waterfall model, underwent evaluation via usability testing. LLM-based elements attained an average rating of 4.40 out of 5 for clarity and relevance in study assistance, whereas organisational and note-taking capabilities garnered a satisfaction score of 4.11 out of 5. The findings underscore StudyMate's capacity to alleviate cognitive stress and enhance the learning experience for autonomous learners, especially in technical fields

Keywords: Cognitive overload, Large Language Models, Note organisation, Self-directed learners, Technical majors

1 Introduction

In the modern digital era, self-directed learning has become essential for individuals aiming to advance their studies and careers. The internet offers unprecedented opportunities for learners to explore diverse topics and acquire new skills. However, the fragmented nature and overwhelming volume of study materials often lead to cognitive overload. According to Cognitive Load Theory, this occurs when the complexity and quantity of information – combined with external distractions like switching platforms – exceed the brain's working memory capacity, impairing comprehension and productivity [1].

This problem is especially evident among students in technical fields such as Electrical and Electronics Engineering, Computer Science, and Computing. According to author-collected data from INTI International College Penang [2], 69.7% of students in these disciplines frequently experience being overwhelmed by the quantity of study information, 66.7% reported challenges in comprehension, and 78.8% saw a decrease in motivation attributed to the stress resulting from cognitive overload. This local observation aligns with broader findings – Meyer [3] reported that, in a representative German sample, 22.5% of respondents identified information overload as one of the most frequent sources of stress. Factors such as traversing various platforms, alternating between interfaces, and querying multiple sources (including documents, images, videos, and texts) exacerbate this issue by introducing distractions, elevating cognitive demands, and exhausting the working memory essential for effective learning.

This paper presents StudyMate, a centralised web-based tool aimed at mitigating cognitive overload by optimising access to and management of study resources. The system is designed in accordance with Cognitive Load Theory, focusing on two core objectives: (1) leveraging Large Language Models (LLMs) to analyse uploaded documents and provide intelligent study support, such as summarisation, key concept extraction, and flashcard creation, and (2) offering a structured platform for efficient study material organisation with features like annotation tools and folder management. Tailored for technical students at INTI International College Penang, StudyMate aims to reduce cognitive demands, enhance organisational efficiency, and contribute to the development of intelligent tools for self-directed learning.

The shift toward cloud-based smart technologies and AI-supported platforms has become increasingly prominent in international educational research. Recent studies highlight how such tools revolutionise open learning environments by improving learner autonomy, reducing cognitive barriers, and enabling personalised engagement with materials [4], [5]. StudyMate contributes to this global trajectory by applying transformer-based models to provide intelligent study assistance, offering a scalable, intelligent study companion for technical students.

StudyMate is entirely conceived and created by the authors. All aspects of architectural design, development tasks, and feature implementation – including UI/UX design, integration of AI models, and system testing – were performed from the ground up by the development team. This approach ensures that StudyMate delivers a novel, context-specific learning solution tailored specifically for technical students in higher education.

To support the system’s AI functionalities, particularly summarisation and flashcard generation, a review of deep learning architectures was included in the Literature Review. This section highlights the evolution and advantages of transformer-based models, which underpin modern Large Language Models. Understanding these architectures was essential for selecting a reliable, state-of-the-art foundation for StudyMate’s intelligent features. A comparison between RNN and transformer-based models is essential to understand trade-offs in context length, computational efficiency, and latency. Such analysis supports the selection of GPT-3.5 Turbo for StudyMate by balancing performance, resource demands, and context window for effective AI-driven study assistance.

2 Literature Review

A Cognitive Load Theory in Self-Directed Learning

John Sweller introduced Cognitive Load Theory (CLT) in the late 1980s. This paradigm investigates the constraints of working memory during the learning process. It divides cognitive load into three categories – intrinsic, extraneous, and germane – that influence the amount of information that can be processed simultaneously, as shown in Table 1 [1].

Table 1: Cognitive Load Types and Definitions

Intrinsic Load	The inherent complexity of the material itself.
Extraneous Load	The manner in which information is presented to learners.
Germane Load	The mental effort required to process new information and integrate it into existing knowledge structures.

Due to the complexity of the subject matter, students frequently encounter a high intrinsic burden in self-directed learning, particularly in technical disciplines. This challenge is further exacerbated by the extraneous burden that results from disorganised or fragmented study materials. The objective of effective instructional design is to minimise extraneous load and optimise germane load in order to facilitate learning [6].

This is especially true in self-directed learning environments, where the ability to effectively manage cognitive burden is essential for improving learning outcomes. Knowledge retention and engagement are significantly enhanced when instructional designs are in accordance with learners' cognitive capabilities, according to research. A study on microlearning modules by Lopez [7] showed that learners' retention rates and motivation increased when extraneous cognitive burden was reduced and germane load was increased. Additionally, it is imperative to comprehend the relationship between self-efficacy and cognitive burden. Zhang's study [8] discovered that students' self-efficacy can be diminished by high cognitive burden, resulting in a decrease in motivation and persistence in learning tasks. On the other hand, students' confidence and motivation are increased and learning results are enhanced when cognitive load is properly managed. Learning assists can be developed to assist self-directed learners in managing cognitive load, thereby improving learning outcomes, by comprehending and implementing the principles of CLT.

B Deep Learning Architectures for NLP: RNN vs. Transformer

Deep learning has revolutionised natural language processing by employing models that encapsulate the intricacies of sequential data. Early architectures, such as Recurrent Neural Networks (RNNs), processed text sequentially but encountered difficulties, including vanishing gradients, which impeded the capturing of long-term dependencies and constrained parallel processing during training [9].

Vaswani et al. [10] introduced transformers to address these challenges by utilising self-attention mechanisms to facilitate the parallel processing of entire sequences. This is crucial for StudyMate, which reduces extensive study materials – such as PDFs and slides – to concise outputs, thereby expediting training and enhancing contextual comprehension.

StudyMate uses a transformer-based Large Language Model (LLM) because it performs well in language interpretation and long-range dependency modelling. State-of-the-art models (such as GPT models) excel at summarisation, key point extraction, and creating study materials [11]. They efficiently minimise intrinsic load (by simplifying content) and extraneous load (by filtering irrelevant information), boosting the learning experience for self-directed students in technical disciplines.

While transformers represent a significant advancement in sequential NLP processing, RNNs were the pioneers. StudyMate uses a cutting-edge transformer approach to handle complicated instructional content efficiently, offering learners well-organised, contextually rich study support.

C Selection of the Transformer Model for StudyMate

A comprehensive comparative analysis of three prominent Large Language Models (LLMs): GPT-4o, GPT-3.5 Turbo, and Mixtral 8x22B, is conducted to ascertain the most suitable transformer model for StudyMate shown in Table 2. The analysis concentrated on critical metrics, including context window, speed, latency, benchmark scores (MMLU and Chatbot Arena), pricing, and supplementary features such as fine-tunability and open-source availability [12] [13] [14].

Table 2: Comparative Analysis of LLMs

Model	GPT-4o	GPT-3.5 Turbo	Mixtral 8x22B
Provider	OpenAI	OpenAI	Mistral AI
Context Window	128k	16k	64k
Speed	69	57	62
Latency	0.51	0.37	0.26
Benchmark (MMLU)	88.70	70.00	77.80
Benchmark (Chatbot Arena)	1287	1107	1146
Open-Source	No	No	Yes
Access via API	Yes	Yes	Yes
Price Input (USD per 1 million tokens)	5.00	0.50	2.00

Price Output (USD per 1 million tokens)	15.00	1.50	6.00
Fine-Tunability & Prompt Engineering	Yes	Yes	Yes

These critical metrics are summarised in the preceding table. Despite the fact that GPT-4o boasts the highest benchmark scores and an enormous 128k context window, its high cost and latency make it unsuitable for a student project with restricted resources. In contrast, Mixtral 8x22B is open-source and cost-effective; however, it may have limited community support and fall short in benchmark performance.

The assessment indicates that GPT-3.5 Turbo achieves the optimal equilibrium: it provides strong performance with a 16k context window, notable speed and latency, and – crucially – a much reduced cost. The API-based access, robust community support, and potential for quick engineering render GPT-3.5 Turbo the most feasible choice for StudyMate. This approach satisfies the technical specifications for effectively processing and summarising extensive study materials while adhering to the project’s financial limitations, hence ensuring scalability and sustainability for our intended audience.

D Critical Analysis on Relevant Existing Applications

Three relevant current systems [15] [16] [17] were analysed to evaluate their features and functionalities, hence informing the design of the proposed web application. A comparison was conducted on the pertinent aspects required for the proposed system, as illustrated in Table 3.

Table 3: Comparison of proposed web application with current relevant applications

	StudySmarter	**PDFgear**	**Microsoft OneNote**	**Proposed System**
Primary Function	Study and revision tool	PDF management	Note-taking	Centralised study and revision tool
Note Organisation	Study sets only	None	Notebooks, Sections	Modules with note and flashcard sections
Document & Code File Support	Documents only	PDF documents only	Attachment of documents only	Documents and code files
LLM Features	Flashcard generation	Contextual Q&A	None	Summarisation, Contextual Q&A, Flashcards generation, code explanation, flashcard answer evaluation
Flashcards	Yes	No	No	Yes
Gamification	No	No	No	Subtle (Flashcards progress checking)

The proposed system aims to provide a comprehensive self-study note-taking platform developed entirely from scratch, with no adaptation of any existing system. The design was informed by observing common strengths and user-friendly features across popular self-study and note-taking tools. Building on these observations, the system introduces original enhancements and tailored functionalities not commonly found in existing platforms, addressing specific needs of self-directed learners. It offers an extensive folder structure and hierarchy to enhance the organisation of notes, documents, and resources. The system offers a rich text editor in conjunction with uploaded documents for efficient note-taking, allowing users to access documents while composing notes. A prominent feature that employs LLM is the provision of button commands for actions on submitted documents (such as summarisation and flashcard creation) to enhance the self-study experience. Flashcards can also be made manually. Subtle gamification aspects will be integrated into the flashcards function, enabling progress tracking, performance evaluations, and an overall score summary at the conclusion of each session. Features are crafted to facilitate an engaging study experience that assesses comprehension and progress over time.

Although StudyMate incorporates features similar to existing tools, its core innovation lies in the way it integrates Large Language Models directly into the study workflow – enabling users to generate flashcards, summaries, and contextual queries from documents and code within a single interface. Unlike conventional note-taking or revision tools, StudyMate is grounded in Cognitive Load Theory and explicitly designed to reduce extraneous cognitive effort for students in technical disciplines. This pedagogically-informed approach, coupled with targeted LLM functionalities and subtle gamification, represents a novel contribution that supports structured, self-directed learning beyond basic feature aggregation.

3 Method

The web application development was decided upon using the waterfall model because of its methodical, sequential approach that stresses early objective definition and well-defined phases. Beginning with broad planning, research, and requirement analysis before moving to design, implementation, testing, and maintenance, this model guarantees consistency and clarity throughout the development life. Every phase contains particular deliverables and objectives, so guaranteeing a methodical and orderly procedure [18]. The linear progression of the waterfall model, as depicted in the image [19], offers a definitive framework, rendering it suitable for projects with steady and comprehensively known requirements from the beginning [20].

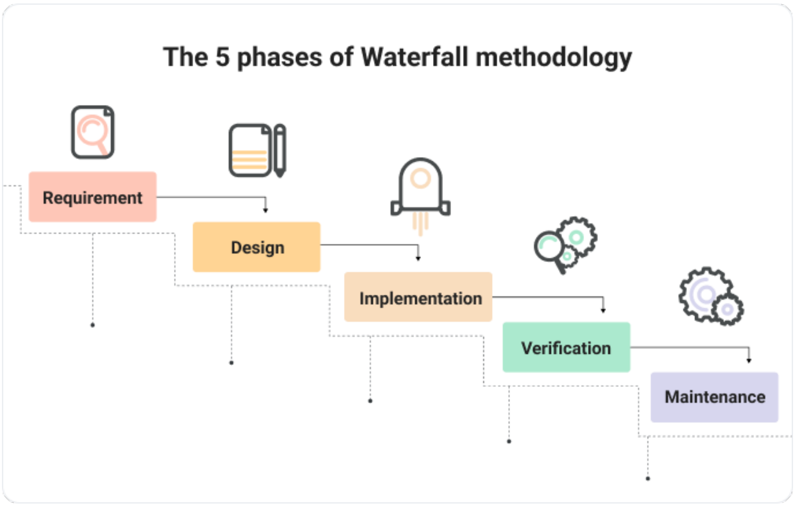


Figure 1 The Waterfall Model

A Requirements Gathering and Analysis

In order to properly create StudyMate, a thorough survey of INTI International College Penang undergraduate students enrolled in technical degrees was carried out. Demographics, study material utilisation, access platforms, cognitive load, and feature preferences were the main topics of the survey. The results aided in identifying important system design specifications. Table 4 provides a clear summary of the key findings.

Table 4: StudyMate Design Requirements Based on User Survey

Aspect	Key Findings	Design Implications
Respondent Demographics	- 27% enrolled in IT degree. - 33% enrolled in Computer Science degree. - 40% enrolled in Electrical and Electronics Engineering degree.	The system must address the distinct study needs of IT and Computer Science students as well as Electrical and Electronics Engineering student.
Study Material Usage	- 97% relies on slides - 93.9% use online websites/articles.	Emphasise digital resource management; design features that support various digital formats (e.g.,

	- 90.9% finds PDF documents helpful. - Typed notes used by only 27%. - Traditional textbooks are not used.	slides, PDFs, online content) to streamline content retrieval and review.
Access Platforms	- 93.9% accesses study materials via laptops/desktops. - 12.1% uses mobile devices - 12.1% uses tablets.	Prioritise a web-based interface optimised for larger screens, while ensuring responsiveness for mobile and tablet use.

Table 5: StudyMate Design Requirements Based on User Survey *(cont…)*

Aspect	Key Findings	Design Implications
Cognitive Load	- 69.7% feels overwhelmed by the volume of information. - 66.7% reports decreased comprehension. - 66.7% experiences difficulty retaining information. - 66.7% note decreased motivation or feelings of disengagement.	Integrate intelligent summarisation, flashcard generation, and organisational tools to reduce cognitive overload and facilitate focused study sessions.
Feature Preferences	- 100% of respondents expressed interest in a system that combines intelligent document scanning with customizable note organisation.	Develop a centralised web application that not only leverages AI to provide intelligent study assistance document scanning and summarisation but also provides flexible organissation tools – such as folder structures, tagging, and annotation capabilities – to cater to varied user preferences.

The survey found that undergraduate computer science, electrical and electronics engineering, and information technology students use laptops to study and rely heavily on digital resources. Processing several vast amounts of information is cognitively challenging. In response, StudyMate is a centralised web software with customisable organising and AI-driven document scanning and summary. This method meets the need for an easy-to-use, effective, and adaptive study aid.

B Design

i. System Architecture

A use case diagram and pertinent activity diagrams are used to explain the system architecture of StudyMate. Key interactions between students and system elements, such as document upload, intelligent summarisation, note organising, and flashcard creation, are depicted in the use case diagram (see Figure 2). The step-by-step workflow for processes is described in depth by complementary activity diagrams. For instance, Figure 3's Document Note Activity diagram illustrates note editing, document interaction, and AI-assisted content creation. In addition to making user interactions more understandable, our dual-diagram method confirms that every functional need is satisfied.

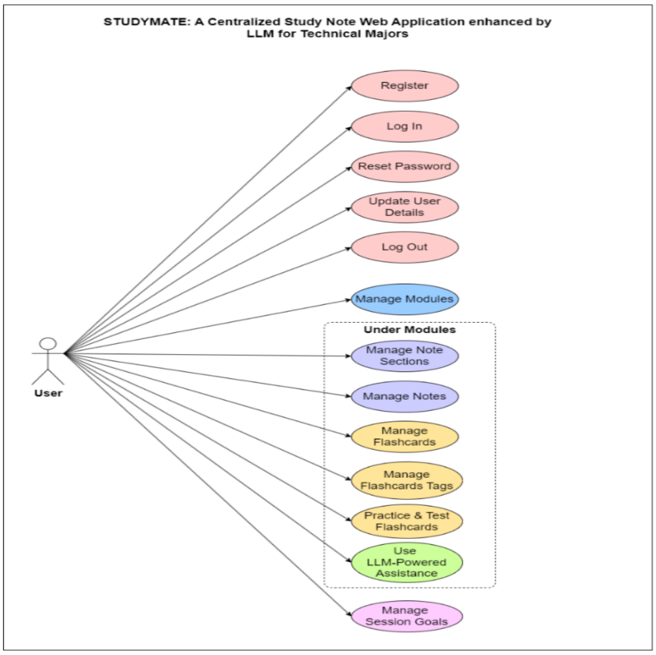


Figure 2 Use Case Diagram of StudyMate

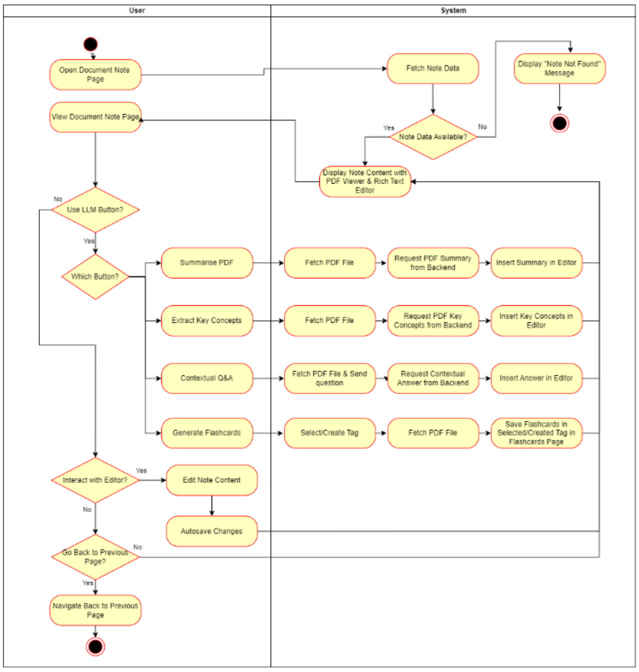

User
System
Open Document Note Page
Fetch Note Data
Display "Note Not Found" Message
View Document Note Page
Yes
Note Data Available?
No
Display Note Content with PDF Viewer & Rich Text Editor
No
Use LLM Button?
Yes
Which Button?
Summarise PDF
Fetch PDF File
Request PDF Summary from Backend
Insert Summary in Editor
Extract Key Concepts
Fetch PDF File
Request PDF Key Concepts from Backend
Insert Key Concepts in Editor
Contextual Q&A
Fetch PDF File & Send question
Request Contextual Answer from Backend
Insert Answer in Editor
Generate Flashcards
Select/Create Tag
Fetch PDF File
Save Flashcards in Selected/Created Tag in Flashcards Page
Interact with Editor?
Yes
Edit Note Content
No
Autosave Changes
Go Back to Previous Page?
No
Yes
Navigate Back to Previous Page

Figure 3 Document Note Activity Diagram

ii. User Interface Design

To lessen cognitive stress and guarantee an easy-to-use, distraction-free study experience, StudyMate uses a simple, minimalistic design. The interface has a condensed menu for easy access to important features and straightforward navigation, with the system logo serving as a home button that takes users to the main Modules page. Colourful module cards and flashcard tags complement a serene, monochromatic blue colour scheme, allowing for visual organising without being overpowering. Three different note page formats are supported by the design: Code Note, Document Note, and Plain Note. For simple note-taking, Plain Note provides a rich text editor; Document Note combines a PDF viewer, rich text editor, and a number of LLM-powered buttons for tasks like document summarisation, concept extraction, contextual Q&A, and flashcard creation; and Code Note offers a dedicated button to explain the complete code in addition to a code reader and rich text editor. The design is responsive on mobile and tablet devices even though it is geared for desktop viewing. Furthermore, customisable, session-based task tracking is made possible by the Session Goals feature, which does not interfere with the study flow. The survey's findings, which highlight the necessity of an easy-to-use layout and efficient digital resource management, are immediately reflected in these design choices. In this design stage, Figma is used, as it enables quick UI iteration [21].

C Development

Industry-standard tools were used in the StudyMate implementation to create a reliable, scalable, and intuitive solution.

Development was conducted in Visual Studio Code for its versatility and debugging tools [22]. GitHub streamlined version control and collaboration through pull requests and automated workflows [23]. The project adopted the FERN stack (Firebase, Express, React, Node.js) for seamless front-end and back-end integration [24]. Firebase, functioning as a NoSQL backend, adeptly administers various study resources (e.g., PDFs, slides, notes) with real-time synchronisation, guaranteeing autosave and data integrity [25]. LibreOffice [26] guarantees format consistency by converting DOCX/PPT to PDF for document processing, whereas PyMuPDF [27] extracts text and images. The GPT-3.5 Turbo API is the source of power for AI-driven features, including flashcard generation, contextual Q&A, key concept extraction, and summarisation, which are specifically designed to alleviate cognitive fatigue for technical students. The Mantine React component package [28] offers a versatile user interface, whereas TipTap [29] facilitates sophisticated note-taking with rich text capabilities, media integration, and LaTeX compatibility. React PDF Viewer augments document interactivity, while React Syntax Highlighter boosts code readability. Together, these technologies create a scalable and maintainable system, meeting StudyMate's functional and usability goals.

D Testing and Evaluation

In order to guarantee error-free performance, compliance with user requirements, and a positive user experience, testing is indispensable. Functional testing for StudyMate encompassed unit and integration tests to verify the functionality of components and their interactions. Twenty evaluators evaluated user-friendliness through satisfaction questionnaires during non-functional usability testing. The procedure is reverted to implementation for adjustments if errors are detected. Targeted refinements, such as interface or prompt tweaks, are prioritised to address gaps without necessitating a complete redesign if usability results fall below benchmarks (e.g., satisfaction levels). This method guarantees that user expectations and functional objectives are in accordance.

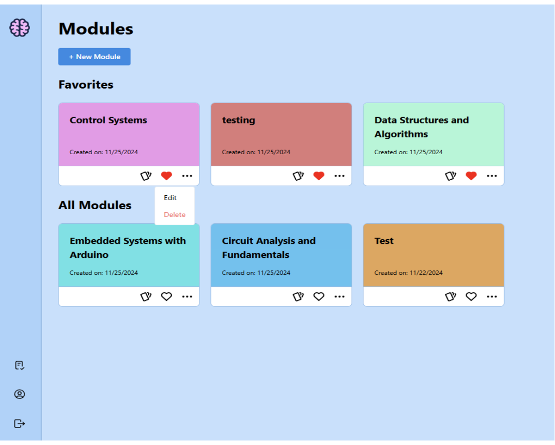
E Maintenance

The maintenance phase commences subsequent to the deployment of the software product to the consumers. New issues may be discovered, and users' change requests may arise, as no system is impeccable. Maintenance encompasses the incorporation, revision, or removal of essential components in response to user feedback, the resolution of defects and errors, and the modification of the product. Documentation will be generated for any usability benchmarks that were not entirely met during testing, enabling additional refinement during the maintenance phase. The methodology guarantees that the system is consistently enhanced in response to user feedback and experience.

4 Results

A System Overview

The StudyMate system was effectively executed as a comprehensive web application that incorporates intelligent document management, note organisation, and AI-enhanced study capabilities. Essential features comprise three unique note pages: Plain Note (a rich text editor for uncomplicated notetaking), Document Note (integrating a PDF viewer with AI capabilities for summarisation, key concept extraction, contextual Q&A, and flashcard creation), and Code Note (intended for code viewing and annotation, with an option for automatic explanation generation). The Modules page functions as the primary centre for arranging study resources, and the Flashcards Test page facilitates interactive self-evaluation. Figures 4-9 provide representative screenshots of the Modules, Modules Overview, Document Note, Code Note, Module Flashcards and Flashcards Test pages, respectively, illustrating the system's clean, minimalist design and fluid movement across components.



Control Systems
Created on: 11/25/2024

testing
Created on: 11/25/2024

Data Structures and Algorithms
Created on: 11/25/2024

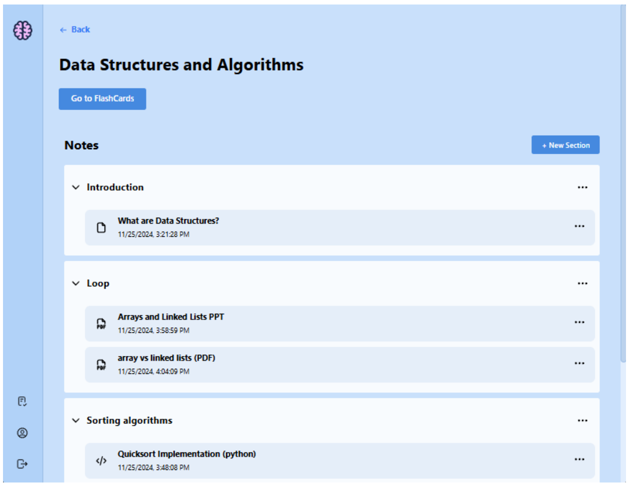
Embedded Systems with Arduino
Created on: 11/25/2024

Circuit Analysis and Fundamentals
Created on: 11/25/2024

Test
Created on: 11/22/2024

Figure 4: Modules Page (Home Page)

The Modules page functions as the primary repository for coordinating study materials. Users can build, favourite, and manage various modules, each representing a distinct theme or topic. Modules are presented using colour-coded cards for convenient identification.



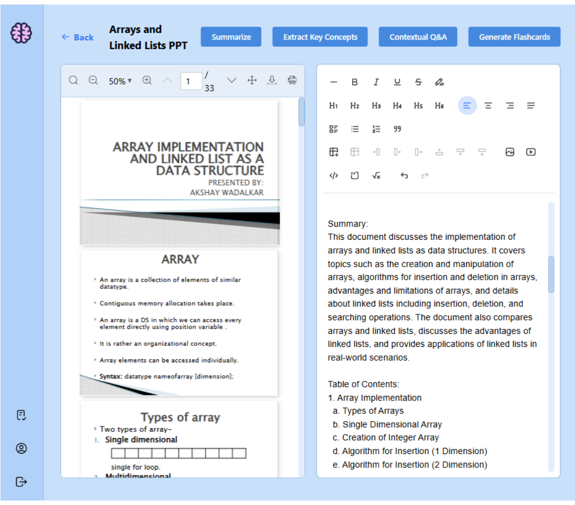
Introduction
What are Data Structures?
11/25/2024, 3:21:28 PM

Loop
Arrays and Linked Lists PPT
11/25/2024, 3:58:59 PM
array vs linked lists (PDF)
11/25/2024, 4:04:09 PM

Sorting algorithms
Quicksort Implementation (python)
11/25/2024, 3:48:08 PM

Figure 5: Module Overview Page

The Module Overview page offers an organised representation of a chosen module, including grouped notes in expandable sections. Users can traverse many subjects/topics and get learning resources effectively.



ARRAY IMPLEMENTATION AND LINKED LIST AS A DATA STRUCTURE
PRESENTED BY:
AKSHAY WADALKAR

ARRAY

- An array is a collection of elements of similar datatype.
- Contiguous memory allocation takes place.
- An array is a DS in which we can access every element directly using position variable .
- It is rather an organizational concept.
- Array elements can be accessed individually.
- **Syntax:** datatype nameofarray [dimension];

Types of array

- Two types of array–
1. **Single dimensional**

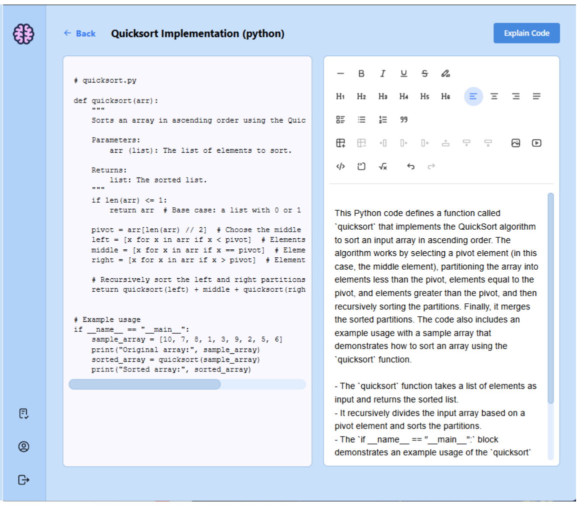
single for loop.

2. **Multidimensional**

Summary:
This document discusses the implementation of arrays and linked lists as data structures. It covers topics such as the creation and manipulation of arrays, algorithms for insertion and deletion in arrays, advantages and limitations of arrays, and details about linked lists including insertion, deletion, and searching operations. The document also compares arrays and linked lists, discusses the advantages of linked lists, and provides applications of linked lists in real-world scenarios.

Table of Contents:
1. Array Implementation
  a. Types of Arrays
  b. Single Dimensional Array
  c. Creation of Integer Array
  d. Algorithm for Insertion (1 Dimension)
  e. Algorithm for Insertion (2 Dimension)

Figure 6: Document Note Page



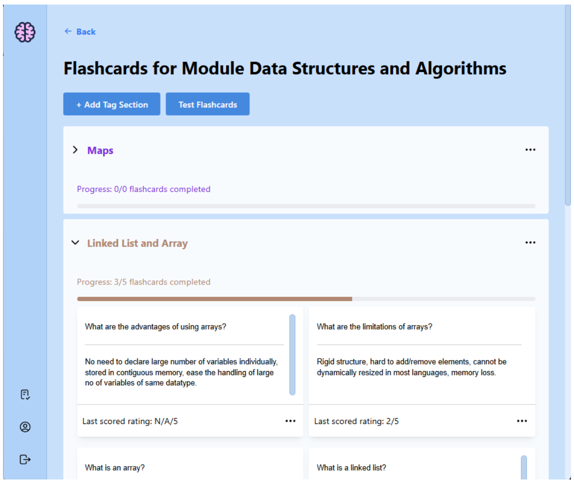
```
# quicksort.py

def quicksort(arr):
    """
    Sorts an array in ascending order using the Quic

    Parameters:
        arr (list): The list of elements to sort.

    Returns:
        list: The sorted list.
    """
    if len(arr) <= 1:
        return arr  # Base case: a list with 0 or 1

    pivot = arr[len(arr) // 2]  # Choose the middle
    left = [x for x in arr if x < pivot]  # Elements
    middle = [x for x in arr if x == pivot]  # Eleme
    right = [x for x in arr if x > pivot]  # Element

    # Recursively sort the left and right partitions
    return quicksort(left) + middle + quicksort(righ


# Example usage
if __name__ == "__main__":
    sample_array = [10, 7, 8, 1, 3, 9, 2, 5, 6]
    print("Original array:", sample_array)
    sorted_array = quicksort(sample_array)
    print("Sorted array:", sorted_array)
```

This Python code defines a function called `quicksort` that implements the QuickSort algorithm to sort an input array in ascending order. The algorithm works by selecting a pivot element (in this case, the middle element), partitioning the array into elements less than the pivot, elements equal to the pivot, and elements greater than the pivot, and then recursively sorting the partitions. Finally, it merges the sorted partitions. The code also includes an example usage with a sample array that demonstrates how to sort an array using the `quicksort` function.

- The `quicksort` function takes a list of elements as input and returns the sorted list.
- It recursively divides the input array based on a pivot element and sorts the partitions.
- The `if __name__ == "__main__":` block demonstrates an example usage of the `quicksort`

Figure 7: Code Note Page

The Document Note page feature integrates a PDF viewer with LLM-powered study tools. Users can summarize documents, extract key concepts, generate flashcards, and ask contextual questions to enhance understanding. Meanwhile, designed for programming-related study materials, the Code Note page allows users to view code. An LLM-powered explanation tool helps users understand code snippets with generated explanations. These features are specifically designed to address common barriers to effective self-study – such as the cognitive overload caused by the abundance of online resources, and the time-consuming task of reading lengthy documents just to locate relevant information. By offering summarisation, concept extraction, and contextual Q&A, StudyMate aims to reduce such friction in the study process. Importantly, these tools are not intended to provide direct answers or replace critical thinking, but rather to scaffold the student's engagement with the material. The objective of StudyMate is to support the process of studying, not to act as an authority on content accuracy or correctness. Users are encouraged to critically evaluate the information provided, as the responsibility for verifying content ultimately lies with them.



Progress: 0/0 flashcards completed

Progress: 3/5 flashcards completed

What are the advantages of using arrays?

No need to declare large number of variables individually, stored in contiguous memory, ease the handling of large no of variables of same datatype.

Last scored rating: N/A/5

What are the limitations of arrays?

Rigid structure, hard to add/remove elements, cannot be dynamically resized in most languages, memory loss.

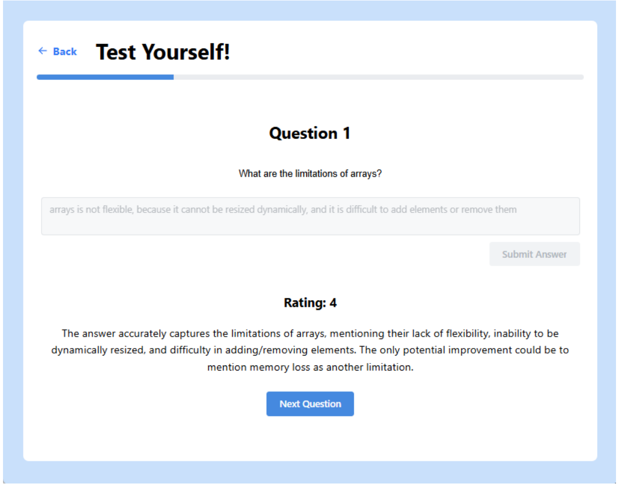
Last scored rating: 2/5

What is an array?

What is a linked list?

Figure 8:" Module Flashcards Page

The Module Flashcards page uses interactive flashcards to help with self-directed learning. Within a module, users can make, arrange, and review flashcards according to various subjects while keeping track of their progress.



Test Yourself!

Question 1

What are the limitations of arrays?

arrays is not flexible, because it cannot be resized dynamically, and it is difficult to add elements or remove them

Rating: 4

The answer accurately captures the limitations of arrays, mentioning their lack of flexibility, inability to be dynamically resized, and difficulty in adding/removing elements. The only potential improvement could be to mention memory loss as another limitation.

Figure 9: Module Flashcards Test Page

The Module Flashcards Test page features an interactive quiz-like interface that enables users to test their knowledge by answering flashcard questions. Responses are evaluated using a large language model (LLM), which takes into account both the question and the correct answer to assess the user's response in context. Unlike conventional systems that mark answers as wrong even for slight variations, the LLM is capable of understanding the intent and relevance of the response. This allows for a more accurate and fair evaluation. The rating is also generated by the LLM, making it unbiased – unlike platforms such as StudySmarter, where users must rate themselves, potentially leading to inconsistent or subjective feedback.

B Functional Testing Results

To make sure every component functions both independently and in unison with other features, a thorough functional testing method was implemented, combining unit and integration tests. File uploads, module and note management, user registration, login, and AI-driven features were all addressed in the test cases. The main test cases, their anticipated results, and pass status are compiled in Table 1 below, which attests to the system's compliance with the design specifications.

Table 6: StudyMate Functional Testing Summary

Test Case	Expected Outcome	Result
User Registration	Account successfully created	Pass
Login and Password Reset	Valid credentials enable login; reset works as expected	Pass
Module, Note Item & Flashcard Creation and Management	Modules, note items and flashcards are added, edited, and deleted successfully	Pass
Note Editing (Plain, Document, Code)	Notes autosave and update in real time, and features of the rich text editor	Pass
File Upload and Conversion	Files are validated, converted (if needed), and stored in Firebase	Pass
Summarization, Key Concept Extraction, Contextual Q&A, Code Explanation	Accurate and context-specific responses	Pass
Flashcard generation and user's answer evaluation	Flashcards are generated from uploaded documents, and can be practiced with feedback from LLM	Pass

C Non-Functional Testing Results

Usability testing was performed with 20 volunteers to assess the user experience of StudyMate on a 5-point Likert scale (1 = Strongly Disagree to 5 = Strongly Agree). The testing technique concentrated on two primary objectives: evaluating the effectiveness of the AI-assisted (LLM) capabilities and reviewing the fundamental system features (non-LLM). A benchmark score of 4 was established for both areas. The average assessment for the LLM component, particularly regarding the clarity and relevance of the generated responses, was 4.40, above the norm. The non-LLM functions were assessed by five questions that examined various components of the system. The individual average ratings for these enquiries were as follows: satisfaction with the organisation of study materials and support for self-study (4.25), ease of navigation and usability (4.10), satisfaction with the rich text editor for notetaking (4.50), effectiveness in organising diverse types of study materials (3.55), and helpfulness in enhancing studying efficiency (4.15). The five ratings yielded an overall non-LLM average of 4.11, which satisfies the standard. The results suggest that although most features are positively received, there exists potential for enhancement in the organisational functionalities.

Table 7: Usability Feedback on LLM-Integrated Features

Features	Average Rating	Comments
Clarity & Relevance of AI-generated responses.	4.40	AI assistance is highly effective in supporting study.

Table 8: Usability Feedback on Non-LLM Features

Feature	Average Rating	Comments
Organising study materials and supporting self-study.	4.25	Good feedback, though search/filter options could improve this further.
Ease of navigation and use.	4.10	Intuitive layout and clear navigation reported.
Rich text editor satisfaction (formatting, media embedding).	4.50	Advanced editing features are highly appreciated.
Effectiveness in organising various study materials.	3.55	Basic structure works; enhancements like filtering would help.
Helpfulness in improving studying efficiency.	4.15	Core features effectively support efficient self-study.
Average from 5 questions.	4.11	Overall positive; organisational features need enhancement.

5 Conclusion

StudyMate effectively fulfils its goals by offering a centralised, intuitive platform for autonomous learners in technical disciplines, including IT, electrical engineering, and computer science. The platform improves study efficiency and minimises cognitive load through the integration of LLM-based study aid with effective organising features. Usability testing confirmed its efficacy, with LLM features achieving an average rating of 4.40 for relevance and clarity, and core elements garnered an overall satisfaction score of 4.11. Prospective improvements may substantially boost StudyMate's capabilities. Implementing sophisticated search and filtering capabilities would enhance content browsing, while a tailored dashboard with task monitoring could assist students in managing deadlines more efficiently.

Augmented gamification, such as incorporating badges, levels, and rewards, may improve engagement in flashcard-based learning. Improving LLM capabilities by optimised prompt engineering and fine-tuning may enhance response precision. Furthermore, extending text extraction capabilities to encompass LaTeX equations, diagrams, and photos would facilitate the effective processing of increasingly intricate study materials. Enhancing user personalisation in flashcard creation could further optimise the study experience. Through the implementation of these enhancements, StudyMate can evolve into a more robust and sophisticated self-study instrument.

While StudyMate leverages Large Language Models (LLMs) to enhance learning through summarisation and flashcard generation, the system assumes the accuracy of the original study materials uploaded by users. Evaluating the factual correctness or credibility of source content is beyond the scope of this project, as it would require a separate framework for content verification. As such, users are responsible for the quality of their input materials, and the AI assistance provided is intended to support – not replace – critical thinking. Future research could explore mechanisms for detecting misinformation or content inconsistencies in educational inputs, which represents a valuable but distinct area of inquiry.

This study was conducted with a limited sample of 20 students from a single institution and focused on technical disciplines. While the findings offer promising insights into user satisfaction and the potential of AI-enhanced learning tools, the small and localised sample size limits the generalisability of the results. Broader evaluations across diverse academic backgrounds, institutions, and cultures would be needed to assess how well StudyMate adapts to different learning contexts. Additionally, the current evaluation emphasises subjective satisfaction; future research could include behavioural metrics such as time on task, learning outcomes, or retention rates to better measure educational impact.

Acknowledgement

I am grateful to my supervisor, Ms Khor Jia Yun, for her advice, support, and constructive criticism during this assignment. I would also want to thank the School of Computing for their help and academic basis. My peers' discussions and cooperation improved this project, so thank you. I am also grateful to my family and friends for their tolerance, encouragement, and support. Finally, I thank everyone who helped test and assess the project. Their constructive critique has really improved StudyMate.

Conflict of Interest Statement

The authors agree that this research was conducted in the absence of any self-benefits, commercial or financial conflicts and declare the absence of conflicting interests with the funders.

References

[1] O. Chen, "A Cognitive Load Theory Approach to Defining and Measuring Task Complexity Through Element," *A Cognitive Load Theory Approach to Defining and Measuring Task Complexity Through Element*, vol. 1, no. 1, p. 18, 2023.

[2] C. H. Chow and J. Y. Khor, "Internal survey on cognitive load and study habits among technical students," INTI International College Penang, Penang, 2024.

[3] S. Meyer *et al.*, "Entspann dich, Deutschland! TK-Stressstudie 2021," TK-Stressstudie 2021, pp. -, 2021.

[4] S. P. *et al.*, "Revolutionizing education: using computer simulation and cloud-based smart technology to facilitate successful open learning," Digital Library NAES of Ukraine (National Academy of Educational Sciences of Ukraine), 2023.

[5] S. P. *et al.*, "Unlocking the power of synergy: the joint force of cloud technologies and augmented reality in education," 2023.

[6] R. Churches, "Using cognitive load theory as a touchstone for curriculum reform: recommendations for policymakers," EDT, 29 October 2024. [Online]. Available: https://www.edt.org/research-and-insights/using-cognitive-load-theory-as-a-touchstone-for-curriculum-reform-recommendations-for-policymakers/?utm_source=chatgpt.com. [Accessed: Jan. 21, 2025].

[7] S. Lopez, "The Impact of Cognitive Load Theory on the Effectiveness of Microlearning Modules," *EJEDU*, vol. 5, no. 2, pp. 29–35, 2024.

[8] H. Zhang, "Cognitive load as a mediator in self-efficacy and English learning motivation among vocational college students," *PLoS ONE*, vol. 19, no. 11, p. e0314088, 2024.

[9] A. O. Nicolas Zucchet, "Recurrent neural networks: vanishing and exploding gradients are not the end of the story," arXiv, 2024.

[10] A. Vaswani *et al.*, "Attention Is All You Need," arXiv.org, 2017.

[11] M. S. L. Basyal, "Text Summarization Using Large Language Models: A Comparative Study of MPT-7b-instruct, Falcon-7b-instruct, and OpenAI Chat-GPT Models," arXiv, 2023.

[12] M. S. M. P. C. Mardini, "Updated June 2024: a Comparative Analysis of Leading Large Language Models," 2024. [Online]. Available: https://mindsdb.com/blog/navigating-the-llm-landscape-a-comparative-analysis-of-leading-large-language-models#heading-openai. [Accessed: Jun. 12, 2024].

[13] MistralAI, "Mixtral 8x22B," 2024. [Online]. Available: https://mistral.ai/news/mixtral-8x22b/. [Accessed: Jun. 11, 2024].

[14] OpenAI, "Models Documentation," OpenAI. [Online]. Available: https://platform.openai.com/docs/models. [Accessed: May 12, 2024].

[15] StudySmarter, "StudySmarter | The learning app for university & school," 2023. [Online]. Available: https://www.studysmarter.co.uk. [Accessed: May 14, 2024].

[16] Microsoft Support, "Introducing OneNote - Microsoft Support," Microsoft. [Online]. Available: https://support.microsoft.com/en-au/office/introducing-onenote-38be036d-5b5a-49ad-83be-292fe53ad7b3. [Accessed: Jun. 9, 2024].

[17] PDFgear, "Edit and View PDF Files | PDFgear Software," PDFgear. [Online]. Available: https://www.pdfgear.com/pdf-editor-reader/. [Accessed: May 4, 2024]. (Missing year for "n.d.").

[18] ProjectManager, "Waterfall Methodology," 2022. [Online]. Available: https://www.projectmanager.com/guides/waterfall-methodology. [Accessed: May 3, 2024].

[19] J. U. and Ksenia, "Waterfall Methodology: All You Need To know," Bordio, 2023. [Online]. Available: https://bordio.com/blog/waterfall-methodology/. [Accessed: May 3, 2024].

[20] B. Lutkevich, "personal computer (PC)," TechTarget, 2023. [Online]. Available: https://www.techtarget.com/whatis/definition/personal-computer-PC. [Accessed: Jun. 2, 2024].

[21] S. Nytka, "Figma Tech Stack: 10 Reasons Why Designer Need To Use," 14 July 2023.

[22] MICROSOFT, "Visual Studio Code," Microsoft, 2016. [Online]. Available: https://code.visualstudio.com. [Accessed: Jun. 15, 2024].

[23] R. Ashraf, "GitHub Explained: Features, Benefits & Uses - Learn Now!," VPSServer.com, 2023. [Online]. Available: https://www.vpsserver.com/what-is-github/. [Accessed: May 27, 2024].

[24] S. More *et al.*, "Research and Analysis on FERN Stack in E-Commerce Web Application Development," *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*, vol. 2, no. 3, pp. 416-418, 2022.

[25] Google, "Firebase," 2019. [Online]. Available: https://firebase.google.com. (Missing access date).

[26] The Document Foundation, "LibreOffice – Free and Open Source Office Suite," LibreOffice. [Online]. Available: https://www.libreoffice.org. [Accessed: Jun. 21, 2024].

[27] PyMuPDF, "PyMuPDF 1.25.4 Documentation," Read the Docs. [Online]. Available: https://pymupdf.readthedocs.io/en/latest/. [Accessed: Jul. 3, 2024].

[28] Mantine, "Mantine: A fully featured React components library," Mantine. [Online]. Available: https://mantine.dev. [Accessed: Jun. 14, 2024].

[29] TipTap, "Tiptap – The headless editor framework for web artisans," TipTap. [Online]. Available: https://tiptap.dev. [Accessed: Jun. 17, 2024].