

City, University of London

**BSc Computer Science
Final Year Project Report**

Academic Year: 2020-21

Active Notes

by



Systemized and automated web app that uses science-based strategies and studies to help students take effective notes

Project supervisor:



May 2021

Contents Page

CONTENTS PAGE	2
ABSTRACT	4
CHAPTER 1 – INTRODUCTION.....	4
1.1 PROJECT OVERVIEW	4
1.2 THE PROBLEM	4
1.3 PROJECT OBJECTIVES –	5
1.3.1 Primary Objective.....	5
1.3.2 Sub Objective.....	5
1.4 PROJECT BENEFICIARIES –	7
CHAPTER 2 – OUTPUT SUMMARY	7
2.1 DYNAMIC SINGLE PAGE APPLICATION	7
2.2 VUE – JS FRONT END	7
2.3 FLASK R.E.S.T API	8
2.4 USER MANUAL AND SETUP GUIDES	8
2.5 FIREBASE FIRE STORE DATABASE	9
2.5 SOFTWARE ANALYSIS DOCUMENTS.....	9
CHAPTER 3 – LITERATURE REVIEW	10
3.1 LEARNING & NOTE TAKING STRATEGIES.....	10
3.1.1 <i>Studies in Retention by Herbert F. Spitzer (The Journal of Educational Psychology 1939)</i>	10
3.2 SIMILAR SOFTWARE THAT EXISTS.....	12
3.3 GENERATING QUIZZES	12
CHAPTER 4 – METHODS	13
4.1 PROJECT METHODOLOGY & LIFE CYCLE.....	13
4.1.1 Structure.....	13
4.1.2 Design & Prototypes.....	14
4.1.3 Version Control.....	14
4.2 PROJECT REQUIREMENTS & ANALYSIS.....	14
4.2.1 System Requirements – Requirement gathering.....	14
4.2.1.1 Use Case Diagram.....	16
4.2.1.2 ERD Diagram.....	16
4.2.2 SYSTEM REQUIREMENTS.....	16
4.2.2.1 Class Diagram	16
4.3 IMPLEMENTATION	16
4.3.1 Tools and software used	16
4.3.2 Iteration 1 - Fundamental project skeleton.....	17
4.3.2.5 Iteration 1 - Views & Components	19
4.3.3 Iteration 2 – Project Functionality, Authentication & Cloud Database Integration	19
4.3.4 Iteration 3 – Advanced Functionalities.....	21
4.3.5 Iteration 4 – Additional Features	23
4.4 DEVIATIONS.....	24
4.5 TUTORIALS AND RE-USED CODE.....	25
4.6 TESTING	25
4.6.1 Functional Testing - Black Box Tests	25
4.6.1 Non-Functional Testing – User Acceptance Testing	26
CHAPTER 5 – RESULTS	26
5.1 NODE JS AND FRONT-END DEVELOPMENT	27
5.2 FRONT-END DEVELOPMENT DEPENDENCIES	27
5.3 VUE-JS FILE STRUCTURE	27

5.4 DESIGN & ANALYSIS	28
5.4.1 Use Case Diagram & Use case Specifications.....	28
5.4.2 Entity Relationship Diagram	28
5.4.3 Class Diagram	28
5.5 GRAPHICAL USER INTERFACE DESIGNS	28
5.5.1 Whiteboard and iPad GUI sketches.....	28
5.5.2 Wireframe (flow) Diagrams	29
5.6 IMPLEMENTATION	29
5.6.1 Iteration 1	29
5.6.2 Iteration 2	32
5.6.3 Iteration 3	37
5.6.4 Iteration 4 – Additional Features	43
CHAPTER 6 – CONCLUSION AND DISCUSSION	46
6.1 PROJECT OBJECTIVES	46
6.2 WHAT WAS LEARNT	46
6.3 PROJECT REFLECTION.....	47
6.4 FUTURE WORK.....	47
6.5 IMPROVEMENTS	47
REFERENCES:	49
APPENDIX	50
APPENDIX A – PROJECT DEFINITION DOCUMENT	50
APPENDIX B – PARTICIPANT INFORMATION SHEET	52
APPENDIX C – PARTICIPANT CONSENT FORM	53
APPENDIX D – INTERVIEW RECORDS.....	54
APPENDIX D.1 – SUBJECT A CONSENT FORM.....	57
APPENDIX D.2 – SUBJECT B CONSENT FORM.....	58
APPENDIX D.3 – SUBJECT B CONSENT FORM.....	59
APPENDIX D.3 – SUBJECT B CONSENT FORM.....	59
APPENDIX E – PROJECT TIME MANAGEMENT.....	60
Appendix E.1 – Gantt Diagram.....	60
Appendix E.2 – Milestone Document	61
APPENDIX F – SYSTEM ANALYSIS & REQUIREMENTS	61
Appendix F.1.1 - Use Case Diagram	61
Appendix F.2 - ERD Diagram For Firebase Database	67
Appendix F.3 - Noun/ Verb Analysis For Class Diagram	67
APPENDIX G – ROUTINE DESIGN REQUIREMENTS.....	70
Appendix G.1 - Whiteboard Sketches.....	71
Appendix G.2 - iPad Hand Drawn Wire - Diagrams.....	71
APPENDIX H – BLACK BOX TESTING.....	72
Appendix H.1 – Iteration 1 Testing.....	72
Appendix H.2 – Iteration 2 Testing.....	74
Appendix H.3 – Iteration 3 Testing.....	78
Appendix H.4 – Iteration 4 Testing.....	81
APPENDIX I – LINK TO ONE DRIVE FOLDER WITH PROJECT CODE	85

Abstract

This project examines several note-taking strategies university students use and investigates the optimum strategy to take effective notes. The project attempts to bring the theory from investigations in optimum note-taking strategies through a dynamic web application equipped to handle university students' needs. The web application was developed in four iterations with scientifically approved functionalities and features to help students enhance their study and revision sessions. The application is designed with a simple design to minimize distractions while abstracting sophisticated algorithms from the user. The project involves using Natural Language Processing, Neural Networks, Deep Learning and transformer models to assist with the learning and revision of students.

Keywords: Web Application, Natural Language Processing, Deep Learning, Effective Note – Taking, Transformer Models.

Chapter 1 – Introduction

1.1 Project overview

This project attempts to develop a web application designed specifically for university students to aid them with taking efficient notes. Note-taking is a researched topic that has gone through years of analysis and studies to understand the best techniques that yield the optimum results. This project examines the various theories and research involved with note-taking to develop a web application that puts those theories into practice.

1.2 The problem

From the author's experience and students around him, a struggle faced by many is trying to find a note-taking strategy that works for each individual. From books to countless online videos researching this topic, numerous options were suggested with very little or no emphasis on science-based evidence. A lack of time reading and analysing the extensive research performed on this topic will put many students at a disadvantage of not using scientifically recognised strategies to take notes.

With the many years of curiosity on this subject matter and the extensive research performed on this issue, the author found active recall the perfect strategy for students. This method is endorsed by top-performing students worldwide from world-leading universities and has been the go-to strategy by the author. The process starts with a student reading their material (from textbooks or lecture notes Etc.) then writing a set of questions based on what they have read. The questions are intended to be saved and consistently answered without referring to any learning materials. Students go back to their questions during the revision stage and try to recall the answers for each topic. Questions answered correctly are highlighted in green, and the questions incorrectly recalled are highlighted in red. Students focus their time during each revision period on the incorrectly recalled questions and repeat this step in every revision session until all questions have been correctly recalled.

From experience, the amount of time required to deploy this method traditionally on Microsoft Word or other note-taking platforms required significant amounts of effort and time to structure information correctly and keep track of the progress. The time lost and effort required to use this method traditionally defeats the foundational purpose of the strategy. From previous struggles and the issues discussed, the author decided to use this project as an opportunity to solve this problem by automating this process and creating an inclusive note-taking web application for university students using science-based evidence and research.

1.3 Project objectives –

1.3.1 Primary Objective

The main objective of this project is to design a single page interactive web application that automates a science-based note-taking strategy. The project also adds features to include all features that could further enhance learning and revision.

1.3.2 Sub Objective

1. Authentication and allow use by multiple users

The application is divided into multiple sub-objectives. In order to complete this project, the web app must be able to host multiple users and display only display the data associated with each unique user. Authentication and data storage was tested using the main database system (Firebase) for multiple accounts.

2. Structure and organize modules & Lectures

Many foundational functions were required to manage and organize students' modules and lectures. Some of these functions include:

- Adding a new module to the dashboard
- Editing or deleting an existing module
- Adding as many lectures as required
- Viewing each lecture with its unique content of questions and answer
- Adding new Questions and answers
- Editing or deleting existing questions and answers

(For the full functional requirements of the system see Chapter 4.2.1 System Requirements – Requirement gathering)

3. Track progress and automation

One of the critical requirements of the application is to visually show the recall status of each question so that users are aware of topics they need to focus on. In addition to the recall functionality, another fundamental requirement is the reduction of manual work from traditional methods of hiding and showing answers. The application automates this process and is tested by adding new questions and implementing the revision strategy.

4. Generating a quiz

A quiz is an additional advanced feature of the note-taking theories researched in this project to enhance students' learning abilities. The quiz is a chance to test their knowledge and broaden their understanding of specific topics. In adherence to the main objectives of automating traditional processes and increasing students' productivity, the quiz must be automatically generated without much effort and input from the user.

5. Minimize distraction

Careful attention and interviews with potential users were made when adding features to the web application. This is to avoid the mistakes of other notetaking platforms of overwhelming users with irrelevant features. The application is exclusive to university students and only includes features supported by scientific evidence or solve issues posed by other students.

1.4 Project beneficiaries –

This application is intended to be used by university students struggling to find an effective notetaking and revision strategy. Studies have shown that the process used in this application is effective for both factual and application-based subjects. For this reason, students from all academic backgrounds can benefit from this application.

Chapter 2 – Output Summary

2.1 Dynamic Single Page Application

Description	An application compressed in two files, the backend and the front-end, the two platforms communicate with each other using a FLASK R.E.S.T API
Usage	Author, Feedback reviewers and University students who want to use a platform to take notes using science-based strategies
Beneficiary	University Student
Link	One Drive link of a compressed file (see Appendix I)

2.2 Vue – JS Front End

Description	Vue -Js was used as the front-end platform for this project. This output consists of 4520 lines of code using HTML, CSS and JavaScript 10% of which are comments.
Usage	Used by the dynamic application to create the graphical user interfaces and connect to the database. The front-end was also used to make connections to the back-end to generate User interfaces from the generated outputs.
Beneficiary	Author
Link	One Drive link of a compressed file (see Appendix I)

2.3 FLASK R.E.S.T API

Description	FLASK is a python backend framework and was used in this project to handle the sophisticated deep learning, NLP and transformer model algorithms to generate the data for the quiz. Using FLASK, 278 lines of code was used to create a R.E.S.T API so the front-end can communicate with the backend to retrieve the relevant data. .
Usage	Used by the web application to generate a MCQ quiz for Users. The back end communicates with the front-end to transfer the relevant data to create a quiz.
Beneficiary	Author
Link	One Drive link of a compressed file (see Appendix I)

2.4 User Manual And setup guides

Description	The front-end and backend files require dependencies and environments to run the code. The user manual is a document that specifies how to setup the code.
Usage	To successfully use launch the system externally
Beneficiary	Author
Link	Document File (see Appendix I)

2.5 Firebase Fire store Database

Description	A NoSQL Database was successfully setup using Google's cloud Firebase platform. In addition to storing data, the platform was able to handle authentication for the system
Usage	Used to authenticate users and store all relevant data to the web application on the cloud
Beneficiary	Author

2.5 Software Analysis Documents

Description	Extensive analysis of the project was performed to develop the optimum solution to the problem addressed by the project.
Usage	Used to make key design choices as well as influence the data structure of the database for the application
Beneficiary	Author
Link	Document (see Appendix F)

Chapter 3 – Literature Review

3.1 Learning & Note Taking strategies

When we learn something we create new neurons or strengthen the connections between existing neurons in the brain. A typical neuron cell comprises the nucleus, the dendritic, the axon, and axon terminals. A neuron cell receives stimulation from other neurons through the dendritic and causes an electrical impulse to travel through the axon body. As it reaches the axon terminals, it releases

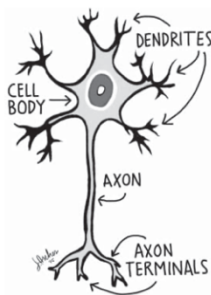


Figure 1.4 Typical Neuron

neurotransmitters which travel through a synapse and onto the dendritic of another neuron (Taylor, K. and Marienau, C., 2016). For many years, in an academic setting, the main form of learning was to take hand-written notes by transcribing the lecture & textbook material or summarising the key points. A study in 2012 explored various note-taking strategies to find out which method yields the best results. The study found that students who transcribe lecture material on a computer and then restudied their notes shortly after had the best performance results (Bui, D., Myerson, J. and Hale, S., 2013). The report suggests that the reasoning behind

this is based on the generation effect (Slamecka & Graf, 1978) and the translation hypothesis (Conway & Gathercole, 1990), both of which explain that information written is better stored in memory than information heard. The study further explains that a computer aids a student with the number of notes taken, which is a crucial factor in their test performance. An alternative form of note-taking was to take organised notes of essential topics in the material. In the case where participants were tested 24 hours after, students who used the transcription strategy with a computer performed better, however in the long term, students who took organised notes did significantly better. They attributed this to better levels of processing framework distinction between storage and retrieval of information (Craik & Lockhart, 1972) whereby organised notes require deeper processing of information which in turn results in better retention (Bui, D., Myerson, J. and Hale, S., 2013). These note-taking approaches come at the expense of students' times and often require significant attention spans to cope with the large amounts of notes required to reach healthy levels of efficacy.

3.1.1 Studies in Retention by Herbert F. Spitzer (The Journal of Educational Psychology 1939).

A study in 1939 (Spitzer, H., 1939) examined 3600 students to see the impact of using an active recall strategy on their exams. The study divided the students into eight groups and made two groups take a test after studying the material. The experiment then examined each group at

different intervals and found that the students who took a test immediately after their study session performed substantially better than the other groups.

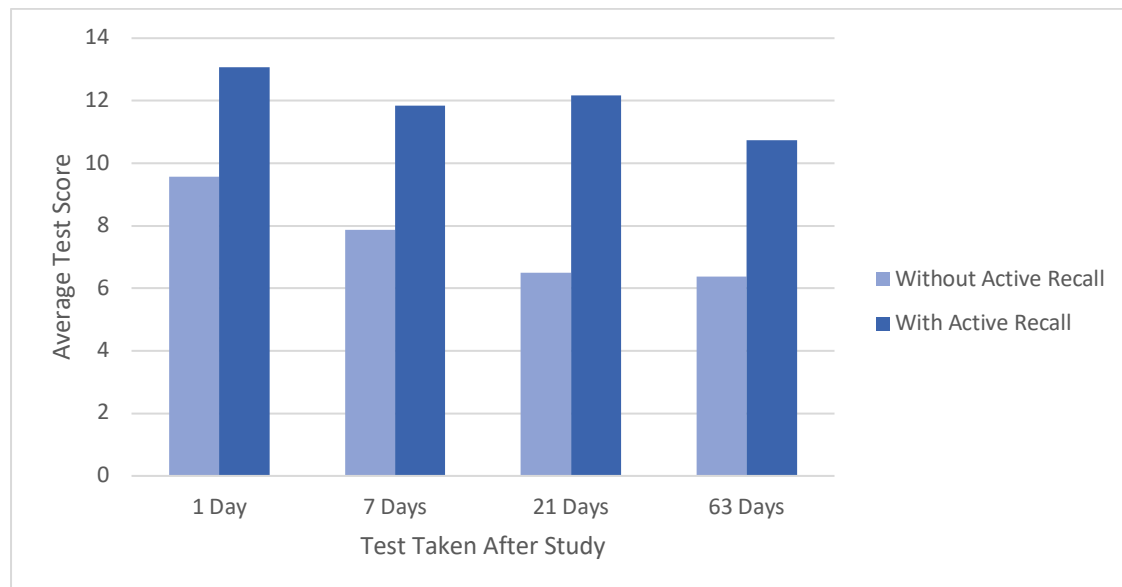


Figure 1 – Graph Depicting Active Recall Strategy

Each group took a test on the material they had revised at different intervals. The graph above compares the average test results of two groups on the same day a test was taken; those who used active recall and those who did not. This study also found that the group that used active recall and took another test one day after performed the best. The paper draws an extraordinarily insight stating that ‘more is forgotten in one day without recall than is forgotten in sixty-three days with the aid of recall’ (Spitzer, H., 1939)

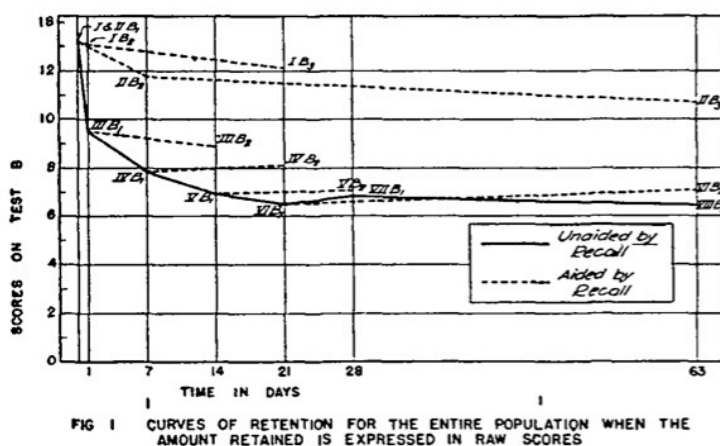


Figure 2 - Recall memory after time in days (Spitzer, H., 1939)

3.2 Similar Software that Exists

Notability and Remnotes are some of the most well-known and used applications in this field. With all the notetaking options they provide, they do not focus on the recall process for notetaking. If students try to implement this method using their platforms, they have to set up everything manually. This project proposes a web application for (university) students that automate this process without adding unnecessary options that may distract the user from their original goal, taking practical notes. From the author's research of similar applications, such platforms do not exist.

3.3 Generating Quizzes

One of the most challenging aspects of this project is finding a way to generate a multiple-choice question quiz from a topic chosen by the user. There are many fields in computer science related to this process as it requires algorithms from Natural Language Processing (NLP), Neural Networks and much more. NLP transforms textual data into numerical data through vectors or other processes and performs modifications to generate the desired outputs. The main element of making a quiz is using input data such as text and an answer to generate a question.

There are several ways to generate questions from text and answers; however, the primary approach that stood out was to use transformer models. Transformers were introduced by the paper "Attention is all you need" in 2017 (Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł. & Polosukhin, I. 2017). In summary, they are a sequence-to-sequence translation model that uses neural networks and an attention mechanism to map input text to an output text. The maths and algorithms of these transformers are beyond this project's complexity; without delving deep into the maths behind the architecture of a transformer model, below is a brief explanation of the architecture.

Transformers are split into encoding and decoding processes. On the left, figure x illustrates the encoding process that takes place where there are several attention mechanisms and neural network layers that are used to generate output encodings. On the right, the same process is used; however, this time, taking in the encoding outputs and using several decoding layers to generate an output sequence. The attention mechanism is a crucial process in the transformer model as its function is to swap each element in a sequence by a weighted average (Raffel, C.;

Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W. & Liu, P. J. 2019). This helps the encoder and decoder process capture and only work with the essential elements within a sequence.

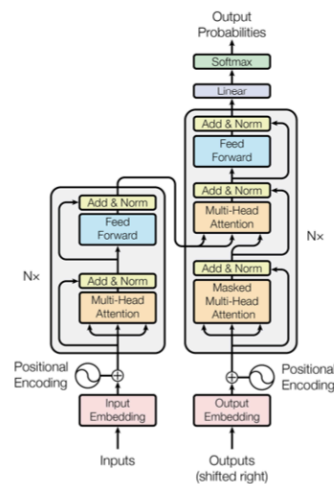


Figure 3 – Transformer model encoder and decoder architecture

*Source - (Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł. & Polosukhin, I. (2017)).

Chapter 4 – Methods

4.1 Project Methodology & Life Cycle

4.1.1 Structure

This project was developed using an iteration-based process to allow flexibility for any changes or feedback along the way. Therefore, it was appropriate to use an agile software development approach where each iteration is known as a sprint with its own set of pre-requirements. After each iteration, or when a considerable number of requirements have been completed, feedback from potential users and specialist consultants was used in consideration to alter or form the next set of requirements.

The first iteration was designing the basic structure and design of the web application. The second iteration included main functionalities and integrating the application to an online cloud database (including authentication). The third iteration was for programming the complex functionalities of the application and integrating it using a R.E.S.T API framework. The final iteration was for implementing an Arduino system that implements added beneficial functionalities to the application.

At first, a gnat chart (Appendix E.1) was used to understand this project's scope and outline the possible tasks required. However, it was much more efficient for an agile development approach to use a milestone action plan (Appendix E.2) on excel to track progress and deadlines of all activities related to the project. With the latter option, it was easier to alter any pre-requirements or add new tasks later on.

4.1.2 Design & Prototypes

The fundamental design approach for this project was to keep the user experience and interface of the application as simple as possible to the user while encompassing sophisticated and complex programming algorithms and architecture concealed behind the interface. Rough sketches of the GUI using a whiteboard and an iPad (Appendix G) were initially designed by the author to get an outlook on all the possible scenarios that could take shape. A wire-frame diagram was designed to understand how a user would interact with the application.

4.1.3 Version Control

Many files were made to organize and make the application work for this project. Although the author started using GitHub as a version control system to keep the code up to date, this approach became increasingly difficult as the number of files grew. The author decided it was better to back-up project files after each iteration using an SD card.

4.2 Project Requirements & Analysis

Understanding how the user interacts with the app is crucial to influence the most critical software decisions and tasks. In addition to user requirements, it was vital to understand the system requirements to get a sense of this project's fundamental functions.

4.2.1 System Requirements – Requirement gathering

Requirement	Brief Description
1	Iteration 1
1.1	Users must be able to view the dashboard where they can view all of their modules
1.2	Users must be able to add new modules consisting of a title and at least one lecture
1.3	Users must be able to edit module details
1.4	Users must be able to delete a completed or an unwanted module

1.5	Users must be able add as many lectures needed for each module
1.6	Users must be able to see all lectures associated with each module
2	Iteration 2
2.1	Users must be able to sign up with their names, email, and password
2.2	System must be able to validate password strength and prompt users to enter stronger passwords if necessary
2.3	System must be able to inform the user if an existing email is used by another account
2.4	System must be able to store users' credentials in a database alongside a unique Id for each user
2.5	Users must be able to login using a valid email address and password
2.6	System must be able to authenticate existing users
2.7	Users must be able to log out of the web application
2.8	Users must be able to access their dashboard upon logging in
2.9	System must be able to retrieve and display associated lecture and module data associated with each user
2.10	Users must be able to access each lecture note page when the click on a lecture
2.11	The system must be able to display all questions and answers associated with each lecture
2.12	Users must be able to view answers when the click on each question
2.13	Users must be able to hide the answer when it is no longer required
2.14	Users must be able to add new questions and answers
2.15	The system must be able to store new questions and answers in a database and link it to the current lecture, module, and user.
2.16	Users must be able to edit a question in case a mistake was entered in the question or answer
2.17	Users must be able to delete a question once it is no longer required
2.18	Users must be able to mark a question as correctly or incorrectly recalled
2.19	System must visually display the recall value for each question
2.20	The system must be able to update the database upon any changes in questions & answers
3	Iteration 3
3.1	The user must be able to click a button and take a MCQ quiz
3.2	The user must be able to add the topics of the quiz
3.3	The system must be able to generate a text, keyword answers, options and questions to generate a multiple-choice question quiz
3.4	The system must be able to display the quiz in a user-friendly interface
3.5	The user must be able to navigate through the quiz by going to the next and previous questions
3.6	The user must be able to close the quiz at any time
3.7	The system must be able to calculate the score of the answers and display it to the user
3.8	The user must be able to view their results

3.9	The system must be able to show the user the correct answers and give context for the correct answer
------------	--

4.2.1.1 Use Case Diagram

The application is only intended to be used by a single user at any given time, hence why it was appropriate to design a use case diagram (Using visual paradigm) to understand their interactions with the system.

4.2.1.2 ERD Diagram

The ERD Diagram helped form the database structure for this project. From the analysis of this data, it is clear that the information is hierarchal and is the best fit for a NoSQL Database.

4.2.2 System Requirements

4.2.2.1 Class Diagram

The application is data-centric, containing data as questions and answers, notes, and credential information from each user. A class diagram was implemented to understand how to model this data.

4.3 Implementation

4.3.1 Tools and software used

Tools & Software	Brief Description
Visual Studio Code	As there were many programming languages involved in developing this project, the visual studio code development environment provides exceptional support for writing code in different languages as well as integrating between them.
Visual paradigm	This software was used to develop analysis diagrams for the project.
Figma	This tool was used to design the mock-ups for the user interface.

Google Collab	Google collab provides memory and GPU & CPU power to perform intensive deep learning tasks on the cloud.
Google Fire store	This tool was used to handle authentication and the database for the project
Programming Languages	<ul style="list-style-type: none"> - Java script - HTML - CSS - Python
Frameworks	<ul style="list-style-type: none"> - Vue-Js - Flask (R.E.S.T API)
Libraries (python libraries for quiz generation)	<ul style="list-style-type: none"> - Pre-trained T5 model for Question generation - Auto-tokenizer - Sequence Matcher - Genism - NLTK - Wikipedia

4.3.2 Iteration 1 - Fundamental project skeleton

The first iteration involves designing the initial front-end design of the application. The style of this project is a single-page application, and therefore, software decisions were made in accordance to fulfil all app requirements efficiently and without adding unnecessary complications.

4.3.2.1 *Vue-Js Framework*

The web app consists of a back-end, a front-end, and a cloud database. Many software decisions were made with research to decide on the optimum software and platforms for each application stage.

This iteration required significant work on the front-end to design the skeleton and user interface of the application. There were several options to choose from; however, the three that stood out were React js, Angular JS, and Vue js. All options are JavaScript frameworks that

integrate JavaScript with HTML and CSS. The first two options packed far too many more bells and whistles for this project, and although they had good community support and have been around for many years, it was clear that it would have required more work to achieve similar results than using Vue.js. In addition to time costs, the primary choice behind settling with the Vue.js framework comes down to its invaluable documentation: concise, engaging, and supports all programming levels.

Since JavaScript was the primary programming language for the front-end, using ES6 as a programming framework was crucial to writing efficient code. This iteration also required asynchronous requests to a local database.

4.3.2.2 Vue Js Component and Views

To reduce clutter and organize the coding workspace, Vue.js structures its files as components and views. Views can be considered as pages, and components are different fragments of code displayed on views. Views are then programmed to organize the structure and flow of components.

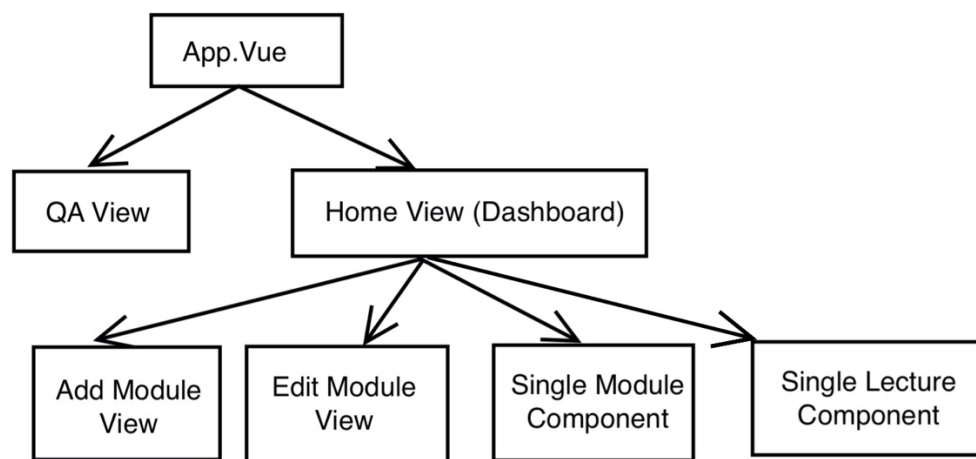


Figure 4 – Iteration 1 Vue File Structure:
Add Single Question Component

4.3.2.3 Re-usable Functions

A folder in VS code was used to store re-usable functions as JavaScript files to organize code better and reduce unnecessary complexity. For this iteration, 'get Modules' and 'get Lectures'

were used as re-usable functions to retrieve data from the local database. Keeping Http request files separate from the rest of the code makes it easier to focus on processing the data later.

4.3.2.4 Local Database

Two options were considered to visualize how dummy data will appear on the front-end. The first was creating a JavaScript file, storing all data as variables, and the second option was to use a JSON file and call the data using HTTP requests. Since the Vue framework is designed to process website data, it is straightforward to integrate and retrieve data from external sources. In addition to integration, the JSON format is mainly used to manage website data efficiently. For these reasons, the latter option was used as testing data.

4.3.2.5 Iteration 1 - Views & Components

Several views and components were made to make up the fundamental skeleton interface of the web application.

4.3.2.5.1 Home View (Dashboard)

The dashboard view is the home interface of the application. It comprises two additional views (Add Module View & Edit Module View) and two components (Single module component & Single Lecture Component). The main requirement for the dashboard is to display a list of all modules and associated lectures for each module. The 'get Module' re-usable function is used to perform HTTP requests to the local database.

4.3.2.5.3 Q&A View

The Q&A view uses the single question component to display all data related to the users' questions and answers. The view uses components rather than views to perform tasks of adding and editing questions. The decision for this was based on the fewer amount of data associated with a question than a module, for instance. It was, therefore, a better choice to edit and add a new question within the same view.

4.3.3 Iteration 2 – Project Functionality, Authentication & Cloud Database Integration

The completion of this iteration marks the minimum viable product (MVP) for this project. During this phase, the functionality of adding questions and answers for any given lecture was

implemented. In addition to functionality, authentication and a cloud fire store database were used to manage separate users and their data.

4.3.3.1 *Vue Js Router View*

Components and views form the fundamental building blocks for designing a single-page application in Vue Js. However, to control what the user views at any given moment, a router view assigns each view its custom link. For iteration one, a router view adds the necessary functionality by controlling what page is displayed to the user and associating each lecture with its unique Q&A view. This way, each lecture can display the same Q&A view design but only show data associated with the current lecture (for example, users' questions and answers).

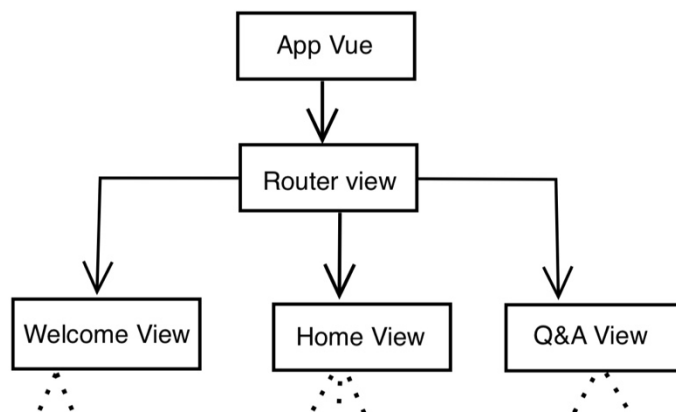


Figure 5 – Iteration 2 – Router View Display structure

4.3.3.2 *Fire Base Authentication*

Authentication is a crucial requirement for this project. In addition to security, it allows each user to access their own set of data. For this project, Google's firebase authentication system using emails and passwords completes this requirement. This feature comes with fundamental security measures such as password difficulty, length and warns users from existing account names. When a user signs up, firebase authentication provides them with a unique ID. In addition to security measures, the primary decision for selecting firebase as an authentication platform was its integration ability with the fire store database.

4.3.3.3 *Google Firebase Fire-store*

In addition to authentication, this project uses the firebase fire-store cloud database to store all user data for the application. Fire-store is a no SQL database management system whereby it does not primarily focus on the relationships between data and does not use SQL for data manipulation (Moniruzzaman, A B M & Hossain, Syed. 2013). The primary use case for this database was to store each user's data, including their modules, lectures, and data related to their questions. Using an API key provided by Firebase, The Vue Js front-end can perform requests to the database to access, update, and store relevant data for each user.

4.3.3.4 Iteration 2 - Views & Components

For this iteration, a new view and three additional components were implemented. The main app uses the main navigation bar component as it is always on display to the user. Its main functionality is to give the user an option to go to the dashboard and sign out. The welcome view uses firebase's authentication system to check whether a user is signed out and displays either login or sign-up component. Figure 1 below shows the structure of views and components in this iteration. Note, the views highlighted in grey are not part of this iteration.

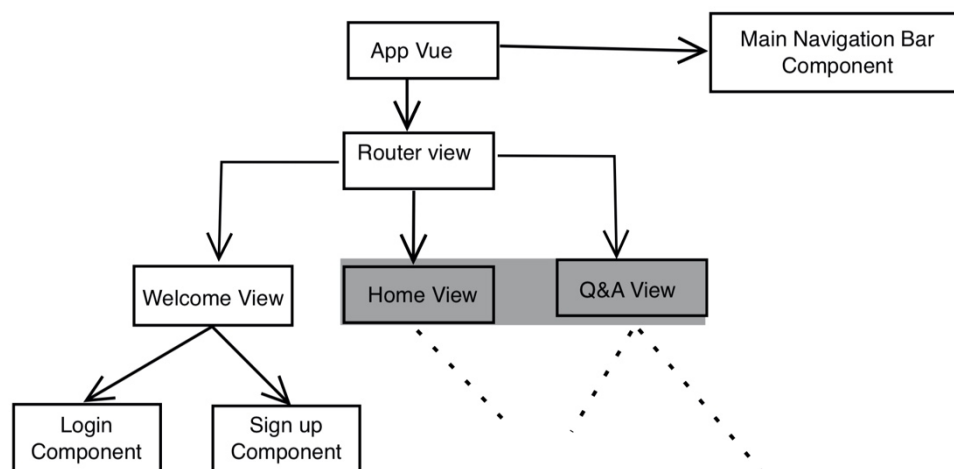


Figure 6 – Iteration 2 Views and Components

** Home View and Q&A View are not part of iteration 2*

4.3.4 Iteration 3 – Advanced Functionalities

This iteration is where the multiple-choice question quiz was generated and integrated into the application. It was by far the most challenging phase of developing the application. As this was a completely new field to the author and consisted of many combinations of different topics and moving parts, a substantial learning curve was required to complete this iteration.

4.3.4.1 NLP Model Research & Analysis

Generating questions and answers from any given text requires massive computer processes using NLP and deep learning strategies. Although the author of this project had previous experience with machine learning, working with transformers using deep learning was a new undergoing. For this reason, extensive research was required to understand how deep learning models work to achieve MCQ generation.

During the research, several transformer models were analysed and compared to determine which ones generate the best results. The study helped the author understand NLP techniques used to retrieve keywords from the text and generate keywords. Understanding how these processes worked was vital in making library decisions for the project.

4.3.4.2 Google Collab

Google collab is a Jupyter notebook python editor on the cloud that was initially used in this project to write and test code that generates a multiple-choice question (MCQ) quiz. There were several reasons for using this platform; The editor provides CPU and GPU power to run deep learning models and stores all required files, libraries, and dependencies on the cloud. Generating an MCQ quiz involves many functions and processes. An additional benefit of this platform was the ability to test small sections of code separately.

4.3.4.3 FLASK R.E.S.T API

This section involves setting up a back-end system to communicate with the front-end and retrieve data associated with the MCQ. Considering the quiz generation code was written in python, the two frameworks that stood out were Django and Flask. Django was a better choice for larger projects that required sophisticated queries to a database. For this project, Django packed too many bells and whistles that were unnecessary to complete the requirements of this iteration. FLASK was chosen as the application programming interface (API) for several reasons.

It was better suited for representational state transfer (R.E.S.T) API applications, can be efficiently set up without initializing unnecessary dependencies, and had invaluable community support.

4.3.4.4 Planning and creating functions

Countless hours were spent researching the approaches to take to generate an MCQ quiz. Upon research, the author settled on the following plan:

1. Generate text from a user's topic of choice
2. Extract Keywords (as answers) from the text
3. Using those keywords and the text, generate a question for each answer
4. For each keyword, generate similar words as options
5. Create a multiple-choice question, whereby each keyword has its own set of options and a question
6. Generate a quiz with multiple-choice questions for each keyword

Several functions were made to complete each step, and a final function was used to group everything and create an MCQ quiz with multiple questions by taking a topic as input. A complete breakdown of each function can be viewed in chapter 5, 5.4.3.3 MCQ Functions.

4.3.5 Iteration 4 – Additional Features

4.3.5.1 Bug Fixes

The following bugs were resolved during this iteration. Issues were faced as a result of some of the testing.

	Bug	Priority	Status
1	Update Log out button in nav bar & remove dashboard option	High	Complete
2	Delete module functionality	High	Complete
4	Add spinner to load quiz	Medium	Complete
5	Add pointer cursor when user hover on log in button	Medium	Complete
6	Add pointer cursor when user hover on sign up button	Medium	Complete

4.3.5.2 Additional Feature – Pomodoro Timer

The additional features in iteration four were only added if they had scientific evidence to suggest they can be valuable to students using the application. Using a Pomodoro timer to measure study and break intervals is an effective tool for students to focus on their study material. The additional feature is accessible from any page within the application and can be hidden from the view of users while studying.

4.3.5.3 Additional Feature – Note Taking Feature & Generate quiz from notes

Although this project is designed to encourage students to write more questions than notes, many students will have formed solid note-taking habits. A new component was added to the application in iteration four to cater for those students. The component was designed to be accessible only if users would like to use the feature to not obscure the main focus of getting users to write their questions. This feature opened the door to another beneficial option where users can create an MCQ quiz from their notes from one button.

Updates to the flask API were done whereby a new method was created that took the text in the form of students notes as input and generated an MCQ quiz. As for the front-end, an additional component was made to select if users want to create a quiz from their notes or topics of their choice.

4.3.5.4 Additional Feature – Tutorial Guide

A component was designed to give users a tutorial of the application when they first launch the dashboard. The tutorial is comprised of four pages with buttons to go next, back, or close the guide. Users have the option to select a 'Don't show again' check box, so they don't have to see the tutorial in the future. The primary decision for adding this component is so users are aware of spaced repetition, a study researched in chapter three, where a time frame is suggested for when they should review their notes.

4.4 Deviations

Based on the user feedback from subjects A-D, it was clear they each had their methods for prioritizing what lectures they wanted to revise. In addition to this, there was no scientific-based evidence or formula to prioritize learning material, and external factors also influenced the decision to prioritize what students revised. For example, subject A preferred revising the material they find challenging in the morning. So, the time of day (and what material is challenging to them) was a factor in their decision. For this reason, the lecture prioritization feature was removed to make time for additional features that add value to the user.

The second deviation was the removal of the Arduino system. During the development stage, money was spent on purchasing testing and all required equipment to build the Arduino. The purchase included: an Arduino UNO three starter kit, ESP6288 NodeMcu, and a touchscreen TFT External display. As this was a new field to the author, a significant amount of time was spent testing and learning how to use these resources, integrating everything, and connecting to the Firebase database. As this feature was not part of the central system and only an addition, it was removed due to the minimal time frame left to complete the project.

4.5 Tutorials and re-used code

With many new subject fields and frameworks, the use of online documentation and online tutorials was crucial to complete the requirements of this project. Even though the author has a strong foundation in programming fundamentals, developing a new web application that consumed a R.E.S.T API was novel. In the project's beginning stages, a great deal of time was spent researching how this process was done from online videos and articles. After the software decisions were made, most of the time spent during this project was spent learning these new frameworks and understanding how they all fit together.

For the Vue-Js front-end framework, many online videos and resources aided the author's learning, including videos from course sites such as Skillshare & Udemy. A similar learning style was applied to learning the back-end platform: FLASK and the required knowledge to complete the quiz generation section for this project. Where new programming frameworks were introduced, the developer documentation was used to aid learning. The documentation was heavily relied upon on two occasions: when learning Vue-Js and Google's Firebase database integration with Vue.

A pre-trained neural network model was used from Hugging Face's repository to generate questions during the quiz generation implementation. The fine-tuned hyperparameters tested by the developer of the model were re-used in this project to maximize efficiency.

4.6 Testing

4.6.1 Functional Testing - Black Box Tests

Black box tests were performed after each iteration, testing all functional elements and features

developed within that stage. Requirements were gathered from the milestone checkpoint excel before starting each iteration and test specifications were used to test each function and feature upon completion. After each iteration, features that were not critical to the usage of the application (such as styling issues) were moved into a 'bug' list as tickets to complete for iteration four so that the important parts and foundations of the program in other iterations were prioritized. (See appendix H for full tests)

4.6.1 Non-Functional Testing – User Acceptance Testing

Due to the current conditions of Covid-19, instead of a live user testing from users, four university students were given a walkthrough of the application and feedback was requested on the non-functional requirements. For full user test documents, see Appendix D.

Subject	Application Speed	Ease of Use	User Interface	Overall Satisfaction
A	90%	70%	95%	Very Satisfied, definitely see themselves using the application.
B	78%	90%	85%	Overall, Very Satisfied
C	99%	100%	85%	Really impressed with the application
D	90%	85%	95%	Exactly what they were looking for

Figure 7 – User Testing Report

Chapter 5 – Results

5.1 Node Js and front-end development

The front end of this project was set up and developed using Node Js to debug software-related issues and run the application on a local server. Vue CLI was used to set up the project and dependency files from the terminal.

5.2 Front-End Development Dependencies

The dependencies in figure 8 below were set up and used in the coding environment of Vue JS. A JSON-server was used in the first iteration to test the application UI using a local JSON file, modelling the data related to each user. The Vue-router is a fundamental dependency to this application; it adds routing functionality to the project – controlling what view (web page) a user is on at any time. In the second iteration, the firebase dependency was used to connect the project to Google's cloud database and add authentication functionalities. Axios was used in the third iteration to connect the FLASK API backend with the Vue-Js.

```
"dependencies": {  
  "axios": "^0.21.1",  
  "core-js": "^3.6.5",  
  "firebase": "^8.3.1",  
  "json-server": "^0.16.3",  
  "vue": "^3.0.0",  
  "vue-router": "^4.0.0-0"
```

Figure 8 – Front-End Dependencies

5.3 Vue-Js File Structure

Vue Js is a JavaScript framework used for developing front-end single-page applications. The file structure of a Vue – Js application consists of the main application, Views, Components, and a Router-View. Components are the basic building blocks of a Vue Js application. A component file stores code to carry out custom functionalities. A view is where all components are rendered and structured. The Router – View is the routing logic of how views are displayed to the users. The main Application file is made up of Views, Components, and the View Router. One Application file may consist of several View – routers, components, and views. During the planning stages of the application, the following structure was used as the skeleton for the application.

5.4 Design & Analysis

5.4.1 Use Case Diagram & Use case Specifications.

A use case diagram was made to understand how the user and external platforms interact with the system. The diagram helped the author examine how the database will interact with the application. All the use cases were made in priority order and use case specifications were made for the top five to explain how the fundamental features in the application work. (See appendix F.1 - for Use case diagram and appendix F for specifications)

5.4.2 Entity Relationship Diagram

An entity-relationship diagram enabled the author to visualize data relationships between each component in the application. From the diagram, it was clear that the relationships between entities were in a hierarchy shape. This insight was a significant factor in using a no SQL database as they are better equipped to handle this type of data structure. (See appendix F.2 for the entity relationship diagram)

5.4.3 Class Diagram

Before understanding what elements within the diagram will form the classes, a noun/ verb analysis on the project definition document was performed (See appendix F.3 for noun-verb analysis). The most relevant nouns and verbs were listed as ideas for classes and functions. The class diagram shows vital mechanisms of the application and (multiplicity) relationships between classes. In addition to functions, the class diagram was critical in analysing the attributes and type of data required for the database. (See appendix F.3.1 for class diagram)

5.5 Graphical User Interface Designs

5.5.1 Whiteboard and iPad GUI sketches

Several sketches of possible user interface designs were made using a whiteboard and an iPad, with the data analysis performed. As mentioned in the methods section, this application follows a simple design principle while utilizing sophisticated and advances algorithms under the hood. The idea behind this is to minimize distractions to students trying to revise and only show

relevant components and features to them. After several sketches, a UI design was chosen with the aid of interviews with two university students, subject A and B. (See appendix G.1 for Whiteboard and iPad sketches)

5.5.2 Wireframe (flow) Diagrams

After a decision was made on the UI design sketches, a wireframe diagram of the user interface was made to see how each UI page is connected. (see appendix G.2 for Wireframe diagrams)

5.6 Implementation

5.6.1 Iteration 1

5.6.1.1 Dashboard Views.

All view files associated with the dashboard were stored in the same folder to organize code. The folder consisted of three views: 'Home', 'Edit Module', and 'Add Module'. The 'Home' view is the application dashboard where a user can access all of their modules and subsequent lectures. The view uses the 'Single Module' Component to display modules in a list view and retrieves the data using the 'get module' re-usable function to perform HTTP 'FETCH' requests to the local JSON database.

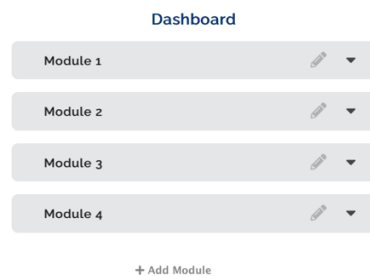


Figure 9 – Dashboard Screenshot

The Edit Module view gives users the option of changing the module title and removing or adding lectures to the module. Once the user confirms their changes, the view also uses the 'get module' re-usable function to perform a FETCH request to update the local database. Similarly, the Add Module view gives users an option to add a new module and update the database. Although these two functionalities could have been displayed within the Dashboard View itself

using components, a lecture may contain a relatively large number of lectures; hence it was decided to use views to maximize screen real estate.



Figure 10 – Edit Module Screenshot

5.6.1.2 Components.

During this stage, two components were built: a 'Single Lecture' Component and a 'Single Module' Component. The single module component is responsible for displaying modules along with their subsequent lectures in the dashboard view. Using CSS and JavaScript, each module is designed to appear on its own with edit and drop-down icons as options to indicate its functionalities. Lectures within a module are displayed to the user when a module or the drop-down icon is clicked. Alternatively, the user could click the module or drop-down icon again to hide the list of the lectures. The list is hidden when a user re-clicks on the module. CSS was used

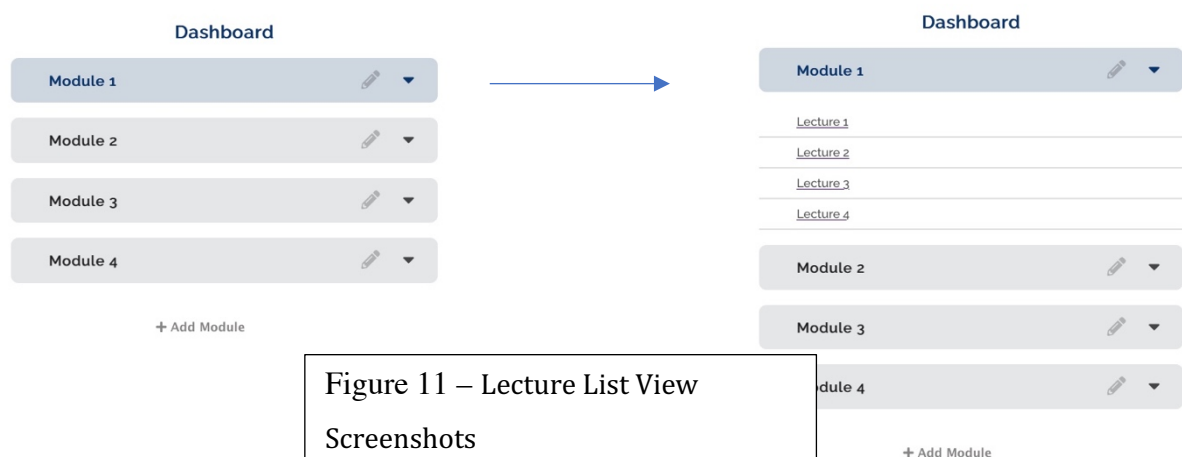


Figure 11 – Lecture List View Screenshots

to structure the component's colour and interactivity; when a user hovers over the modules or icons, a soft colour change is applied to the elements.

5.6.1.3 Json Database

The class and ERD diagrams were crucial in determining the relationships and structure of the data used in this application. For this iteration, testing data was made in a JSON file to model how the application will appear in the application. Each module contains data about several lectures. Each lecture holds information about numerous questions and answers users have for that lecture. Using JSON, the above data structure is represented as an array of module objects comprising of an additional array with data related to each lecture within that module. Another array of objects was used as values for question data within the lectures array. The following diagram is a visual structure of one module object; the entire JSON file contains an array of module objects.

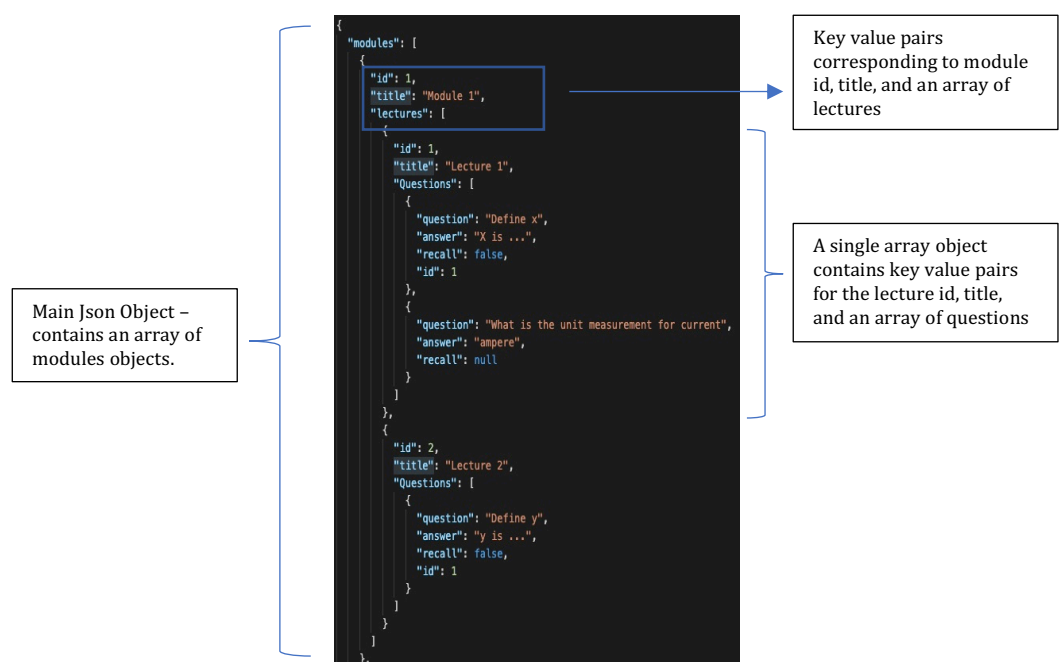


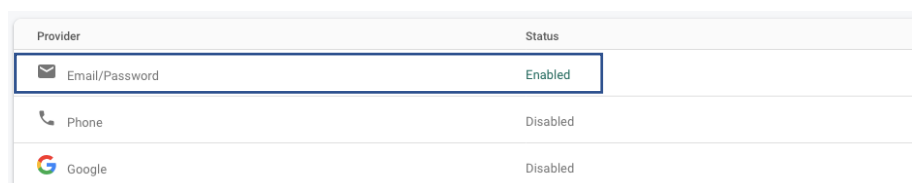
Figure 12 – Json Database for iteration 1

5.6.2 Iteration 2

Completing this iteration marks the creation of a minimum viable project (MVP) as it contains the bare basic functionality of the application. The second iteration was made to add functionality to iteration one, handle multiple users, provide authentication, and utilize the fire store cloud database

5.6.2.1 Firebase Authentication

The local Json database from iteration one was only designed to hold data for one user. Firebase was used as a platform to allow access to multiple users and provide authentication features. Firebase authentication has several options to login from email & password to using social media accounts. In adherence to data policies and reducing unnecessary data for this project, the email & password option was selected for this implementation. When a user creates a new account, the system automatically generates a unique id for them and provides options to specify each user with several parameters such as their names. During the sign-in stage, usernames are taken in as additional input and assigned to each user account.

A screenshot of the Firebase Authentication providers configuration interface. It shows a table with two columns: 'Provider' and 'Status'. The 'Email/Password' provider is highlighted with a blue border and is marked as 'Enabled'. The 'Phone' and 'Google' providers are marked as 'Disabled'.

Provider	Status
Email/Password	Enabled
Phone	Disabled
Google	Disabled

Figure 13 – Firebase authentication

5.6.2.2 Firebase Fire Store

Firebase is a no SQL database management system (DMBS) that stores data as collections and documents. The hierarchical structure of data influenced the decision behind selecting a no SQL database in this project. Upon research between SQL and no SQL databases, SQL databases perform complex queries and store relational data; however, they do not support hierarchal data storage, whereas no SQL databases are designed to support this format. The project database data was

structured to allow users, modules, lectures, and questions to be stored as collections. Each collection has its associated documents holding information using a key-value pair format.

active-notes	users	cLxES814RKNTx50zSXt6L8lILSw1	modules
+ Start collection	+ Add document	+ Start collection	+ Add document
modules	cLxES814RKNTx50zSXt6L8lILSw1	modules	4c8eFHFgq1NtIa4Cwexd >
users >	jFs1IibNdtYo6b9xapULVE0uPXr1		9EyG6mhkfCehkgUWe7kd
	sLzLhb2xZeTVew7XRy1umNZR0IR2		BPE19rRBktRUnEuuc6Qq
	wU83f9kLkNdfUtVEdGDLVT1K9MF2	+ Add field	E17uHb28103EsxmEZD8a

Figure 14 – sample data from fire store

The users' collections store documents about user ids, where each user has access to their module collection. The modules collection stores several documents with information for each module a user has. The document contains key-value pairs for the module id and title. A lecture collection includes documents with key-value pairs to store the lecture title and id. A Question collection contains documents with information about the question itself, an answer to the question, and a recall value.

5.6.2.3 Welcome View

A new welcome view was created to allow existing users to log in and new users to create an account. The view uses two components: log in and sign up and displays only one component at

a time. Users can switch between the two components by clicking on the login or sign-up button.

The image displays two wireframe screenshots of the ActiveNotes application's authentication interface. The left screenshot, titled 'ActiveNotes Log in', features the 'activenotes' logo, an 'Email' input field, a 'Password' input field, a blue 'Login' button, and a link for users without an account to 'Sign up'. The right screenshot, titled 'ActiveNotes Sign up', features the 'activenotes' logo, 'First Name', 'Email', and 'Password' input fields, a blue 'Sign up' button, and a link for existing users to 'Log in'.

Figure 15 – Login/ Sign up screen

5.6.2.4 Components

5.6.2.4.1 Login Component

The login component uses a re-usable function that sends asynchronous requests to authenticate users' credentials with firebase. An error message appears after a failed login attempt where there may have been issues with the credentials. If the credentials are correct, the login component uses the router-view to display the dashboard page. In a scenario where users are not registered, a sign-up button below the login form can close the login component and display the sign-up component.

5.6.2.4.2 Sign up Component

The sign-up component uses a re-usable function to create a new account and stores the information in the fire store database. The re-usable function updates the user's display name parameter provided by firebase to the username entered on sign-up. If a user enters an existing email account or uses a weak password, an error message is displayed. Upon successful sign-up, the component uses the router view to direct the user to the dashboard. A Login button under the sign-up form is displayed

and, when pressed, sends a signal message to the welcome view to hide the sign-up component and display the login component.

5.6.2.4.3 Main Navigation Bar Component

A navigation bar is displayed throughout the application and updates its options based on users' login status. When a user is logged in, a dashboard and an option to log out are displayed; however, when they are signed out, the dashboard button disappears, and the log out button updates to log out. This component is used in the main application file over the router view hence why all views within the router view display the navigation bar.

5.6.2.4.4 Single Question Component

HTML and CSS were used to structure how each question is displayed to the user in the QA view. A single question consists of the question itself, an answer, and a recall value. The question is displayed alongside options to edit and apply recall changes to the component. When a user double clicks on the question, the answer appears below it, and when they re-click the question, the answer disappears. The decision to make the user double click on a question was to avoid a user accidentally clicking on a question and seeing the answer.

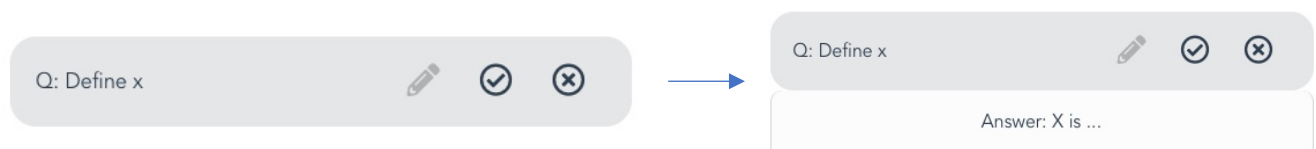


Figure 16 – View Answer Screenshot

Using Javascript, this recall functionality is added to this component. A tick and cross icons are displayed beside the question. Users click on the tick icon if they successfully recall the answer to a question and, conversely, click the cross icon if they did not. A traffic light system (green, amber, and red) was used to track the current status of the recall value. If the recall value is on green and the cross button is pressed, the value is updated to amber, and if they proceed to get the same question wrong, the recall value changes to red. The opposite is also true, whereby if the recall value is red and a user clicks on the tick button, the value changes to amber, and if the same question is recalled correctly, the value is updated to green. Using CSS and Vue-Js, a class

for each colour was made to display the current recall value as a colour on the left border of the question. When a new question is made, the recall value is automatically set to null, and no colour is displayed.

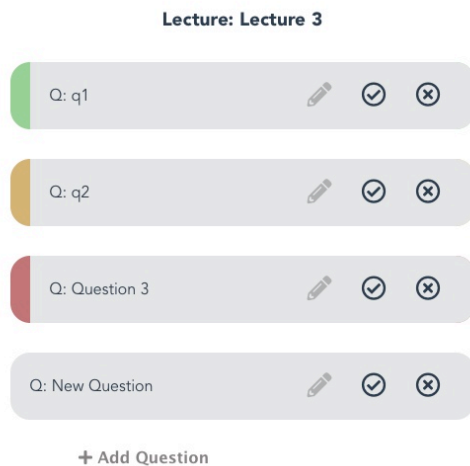


Figure 17 – Q&A view

5.6.2.4.5 Add Question Component

The add question button on the Q&A view activates the 'Add Question' component that appears under the list of existing questions. An input field to enter a question and answer is displayed to the user when this component is triggered. In contrast to adding a module (programmed using as a view), this functionality was designed as a component instead as the input required was less and could fit in the same display. When a user adds their question, the recall value is set to null, and an asynchronous function is used to add the question to the database. If a user decides to omit their decision to edit a question, the cancel button hides the component.

The image shows a form titled "Add Question & Answer". It has two text input fields. The first field is labeled "Write Your Question Here" and the second is labeled "Write Your Answer Here". Below the second field are two buttons: a blue button labeled "Add Question" and a grey button labeled "Cancel".

Figure 18 – Add Question & Answer screenshot

5.6.2.4.6 Edit Question Component

When a user clicks on the edit icon beside the question, the edit component for the question appears. The component has the same design and layout structure as the add question component; however, the existing question and answer values are placed in the input fields when this component appears. When a user submits their new question and answer, an asynchronous function searches for the database's initial question and updates it with the new values.

5.6.2.5 Router-View

For this iteration, the router view was used to display the welcome screen initially. Once a user has signed up or logged in, the router view displays the dashboard. When a user clicks on any lecture, the router view displays the Q&A view with the relevant data for that lecture. The router view takes in an id parameter (for each lecture) to specify the required data from the database and link each lecture to its unique Q&A view. The Edit module also requires an id parameter to specify the current module being edited.

```
{
  path: "/",
  name: "Home",
  component: Home,
  beforeEnter: Authentication,
},
{
  path: "/AddModule",
  name: "AddModule",
  component: AddModule,
  beforeEnter: Authentication,
},
{
  path: "/EditModule/:id",
  name: "EditModule",
  component: EditModule,
  props: true,
  beforeEnter: Authentication,
},
{
  path: "/Q&AView/:id/:lecture",
  name: "Q&AView",
  component: Q&AView,
  props: true,
  beforeEnter: Authentication,
},
{
  path: "/Welcome",
  name: "WelcomeView",
  component: WelcomeView,
}
```

The vue-router has an option to run a function before displaying a view. A function to check users' login status was designed to control access to the web app for users logged in only. The screenshot shows the application of this function by specifying the 'before Enter' parameter on all views where authentication was required (Home view, Add Module view, Edit module View and the Q&A View).

Figure 19 – Vue - Router

5.6.3 Iteration 3

This iteration adds an advanced feature to the project application using a wide range of fields in neural networks, natural language processing (NLP), and a R.E.S.T API. By the end of this

iteration, users can generate a multiple-choice question quiz from topics of their choice. The quiz is displayed using the front end and generated by an API back-end.

5.6.3.1 Model Selection Research

Based on the extensive research carried out, the best approach for this project was to use a neural network model known as the text to transfer transformer (T5) model to generate questions for the MCQ.

BERT was considered an alternative model for this project; however, after a critical evaluation of the two, T5 was selected. The (T5) model was released in 2019 and trained using the C4 corpus consisting of twice the text data (Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W. & Liu, P. J. (2019)). Although it is relatively new, the T5 model is considered a breakthrough in NLP neural networks as it uses both encoding layers and decoding layers from sequence to sequence (More about this model is discussed in chapter 3).

The T5 model used for this project is pre-trained with the SQUAD Dataset for question answering from the hugging face transformer repository. SQUAD is a dataset with tens of thousands of questions and answers from Wikipedia articles. The model is trained by taking in answers and context as input to generate new questions where answers and context are provided.

5.6.3.2 Quiz Generation on Collab.

The first stage of generating an MCQ quiz was getting context from a user's input. This process was simplified using the Python Wikipedia library to summarise an article from any topic. The following section will go through all the functions that turn the summarised text into a multiple-choice question quiz.

5.6.3.3 MCQ Functions

5.6.3.3.1 Get Keywords

This function extracts the essential keywords from any given text. Generating keywords from text involves complex maths algorithms using graphs and vectors to detect the relevancy score

of each word in the text. This project uses Genism's keyword library to handle the heavy calculations and return a list of the most important keywords.

An additional function, 'updated Array', was used to format these keywords to avoid duplicating similar words. The sequence matcher library comparing similarity scores of each word was used to achieve this functionality. It was noticed that the keyword library returned a list of keywords whereby similar keywords were side by side. Instead of comparing the similarity scores of a given keyword with all the words in the list, an extra function was made only to compare neighbouring keywords. This approach had the added benefit of reducing the time complexity of the code.

5.6.3.3.2 Get Options

There are several ways of generating wrong choices for a given question in the MCQ. This function uses the Python Sense to Vec library, whereby each keyword (the answer) is converted to a vector equivalent and used to generate twenty similar words. There could be words incorrectly generated or may have a different meaning than the answer. To solve this issue, the sequence matcher library was used to measure the ratio of each option with the keyword. Finally, seven options with the highest ratios were selected as options.

5.6.3.3.3 Get Sentence

To generate context for each keyword, this function looks through the entirety of the original text and returns all sentences containing the keyword. The NLTK tokenizer was used to split the original text into an array of sentences to achieve this functionality. A loop was made to search for the sentences where a keyword was present in the array.

5.6.3.3.4 Get Question

This function takes in a sentence where the keyword was present and uses the T5 model to encode the text, generate a question from the text and decode it back to a question where it is readable to the user. To get the best out of this model, the developer of the pre-trained model for question generation uses hyperparameters that yield the best results. For this reason, this function was developed using those parameters.

5.6.3.3.4 Get MCQ

To represent each question in the quiz, an MCQ question class was made with the following attributes: context, answer, question, and options. The get MCQ function is designed to put all the previous functions in this implementation together by populating the MCQ question class with the relevant data and returning a list of MCQ question objects. With this function, an MCQ quiz is generated from a topic.

5.6.3.6 Quiz Components

In total, four quiz components were made to handle quiz generation at this stage of the application. A generate button is found below users' questions in the Q&A view to trigger the quiz components. When a button is pressed, the background is blurred using CSS, and the quiz is displayed in the centre of the screen to minimize distractions.

5.6.3.6.1 Quiz Choices Component

A list of topics is required from the user to generate a quiz. This component gives users a friendly interface to enter a list of topics they would like to cover in the quiz. When the topics are submitted, the component sends the list to the main quiz component to perform an asynchronous HTTP GET request to the backend R.E.S.T API in FLASK. The function performs a request for each topic and combines the generated quizzes in one primary MCQ quiz.

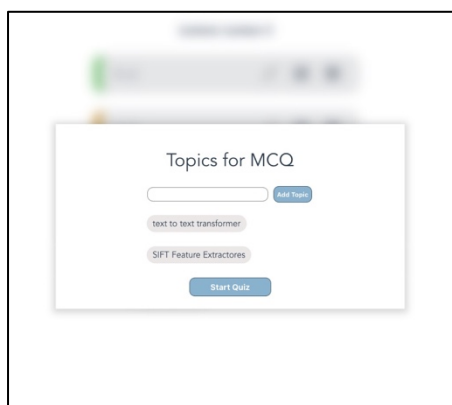


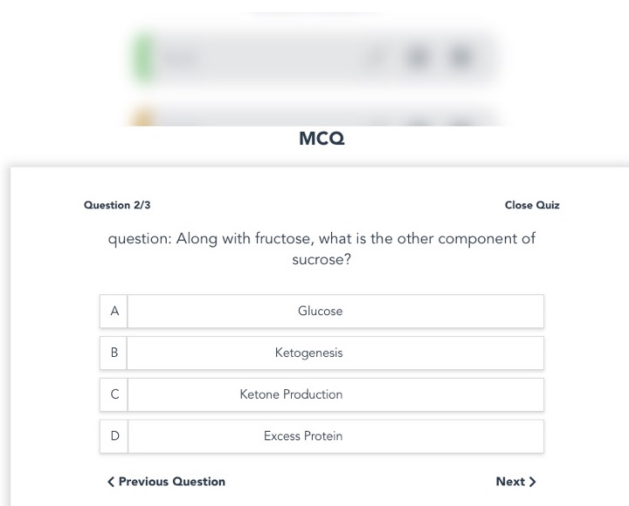
Figure 20 – Topics For MCQ Screenshot

5.6.3.6.2 (Main) Quiz Component

This component is triggered after a user submits their list of desired topics. The previous topic UI is hidden, and the main quiz component is displayed to the user. Its principal function is to display the data received from the FLASK API in a user-friendly interface. This component uses all the other quiz components and displays the most relevant component based on the users' choices.

When the data is received from the API, it is in the form of an object with four values: question, answer, options, and context. A JavaScript function is used to select three options and add the answer to the options array. Finally, the array is shuffled so that the answer is not in a fixed position for all questions.

The structure of the quiz is in the form of a question and four options. Using CSS and JavaScript, each option is associated with a letter in chronological order, and when an option is selected, the border of that option is highlighted in blue. Above the quiz, the current question, total number of questions and an option to close the quiz are displayed to the user. During the quiz, users can go back to a previous question and change their options. The next button is updated to complete and close the quiz in the final question. This button closes the main quiz component and opens the results component.



MCQ

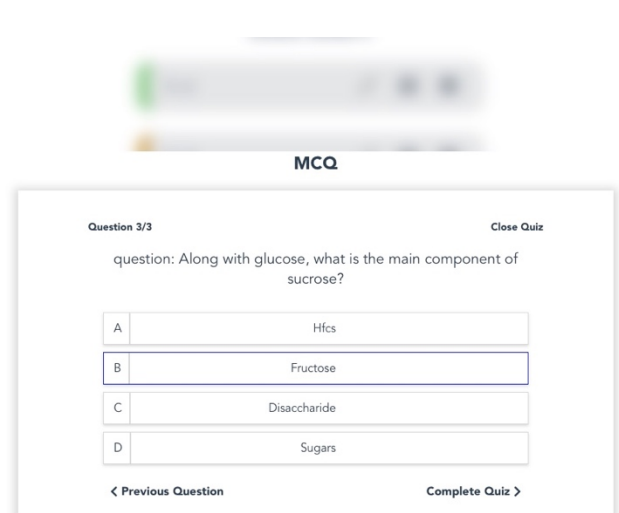
Question 2/3 Close Quiz

question: Along with fructose, what is the other component of sucrose?

A	Glucose
B	Ketogenesis
C	Ketone Production
D	Excess Protein

< Previous Question Next >

Figure 21 – Question form quiz



MCQ

Question 3/3 Close Quiz

question: Along with glucose, what is the main component of sucrose?

A	Hfcs
B	Fructose
C	Disaccharide
D	Sugars

< Previous Question Complete Quiz >

Figure 22 – Final Question in quiz with an option selected

5.6.3.6.3 Results Component

This component displays the result of the quiz in large and has the option to view results. Although it may look minimal, this component includes an important function whereby the previous quiz is replaced with a new quiz results component when the view results button is clicked.

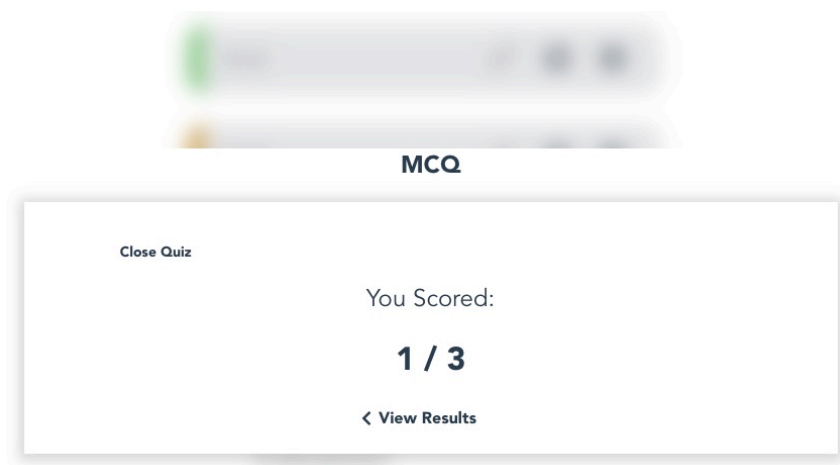


Figure 23 – Quiz results screenshot

5.6.3.6.4 Quiz Answers Component

This component gives the user an option to view the correct answers and learn from their mistakes using the same design structure as the main quiz component. In addition to the main quiz component, a context section is displayed under the options, and when users reach the final question, instead of 'complete quiz', the button is updated to 'close quiz'. Using CSS styling and JavaScript, visual UI is applied to the options' borders to display incorrect answers as red and correct answers as green.

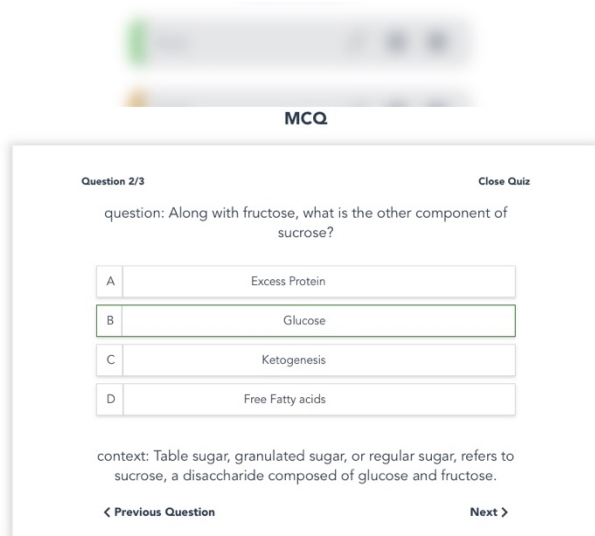


Figure 24 – Question with correct option selected

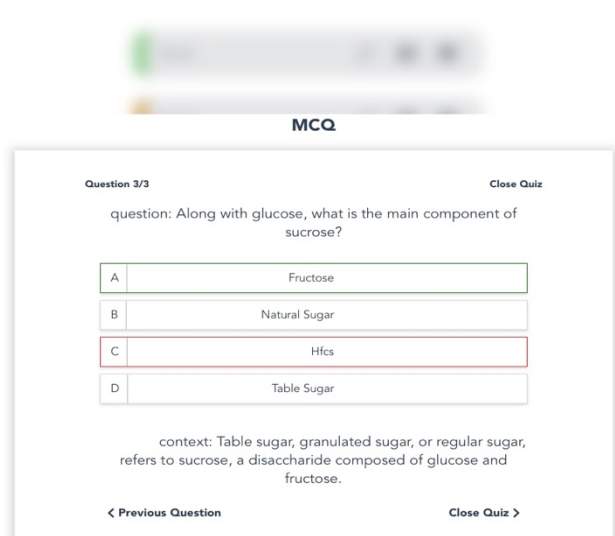


Figure 25 – Final Question with incorrect answer

5.6.4 Iteration 4 – Additional Features

5.6.4.1 Pomodoro Timer

The Pomodoro timer is a scientifically based strategy that helps students maximize the efficiency of each study or revision session. When users log in, they have the option to show the Pomodoro timer at any screen within the application. The timer consists of a focus mode (for studying) and a break mode. Users have the option to adjust the timer as per their choice. After a user has set the timer, they can hide the timer so they can focus on their work. When the timer is finished, users are alerted with a message to inform them.

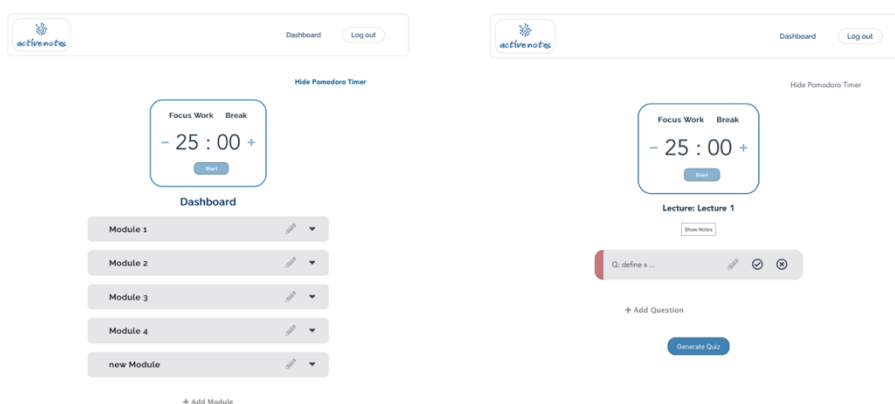


Figure 27 - Iteration 4 – Pomodoro timer in use in the dashboard and in the Q&A View

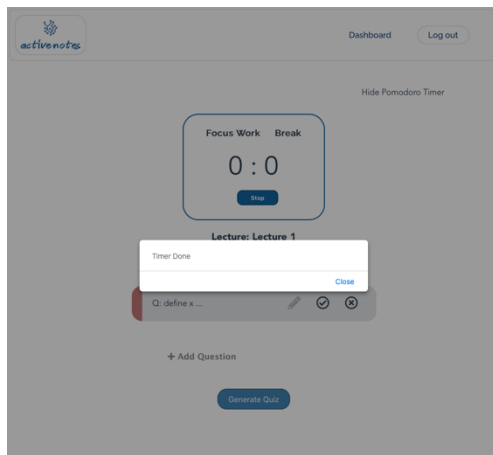


Figure 28 - Iteration 4 – Pomodoro timer alert screen

5.6.4.2 Note Taking Feature & Quiz Generation From Notes

With the click of a button, users can view the notes section to enter their notes for each lecture. The notes section is not designed as the central point of the application; however, many students would have formed strong habits using this method for studying. Considering this, the notes section is designed only to be used for those who need it using the aid of a show and hide button.

When a user activates the notes section, they can add as many topics as they want. Each topic contains an area where they can enter their notes. Users click on the Save button to update their notes to the database or delete to remove the note.

With this new feature added, the quiz component was updated, where users can now generate a multiple-choice question from their notes. By adding a new component, users can choose the type of quiz they would like to generate.

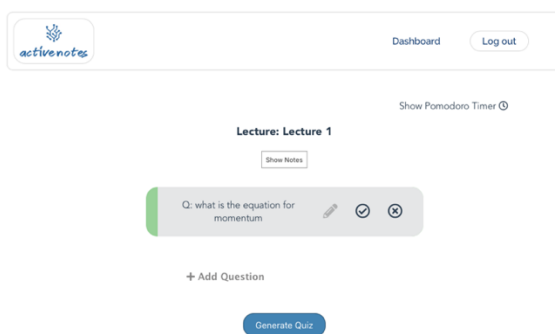


Figure 29 - Iteration 4 – Notes section hidden from the user

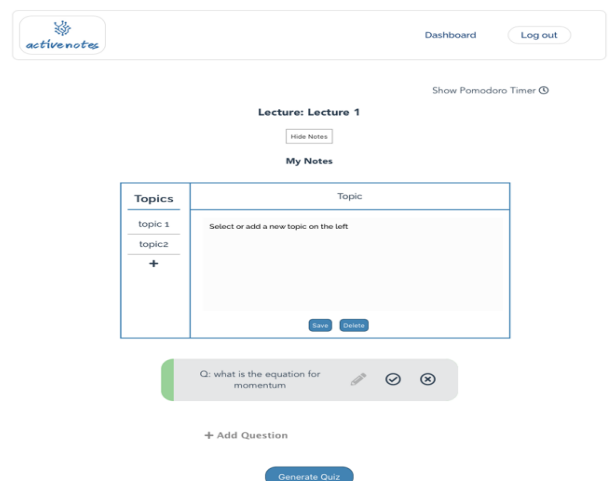


Figure 30 - Iteration 4 – Notes section Activated and in use

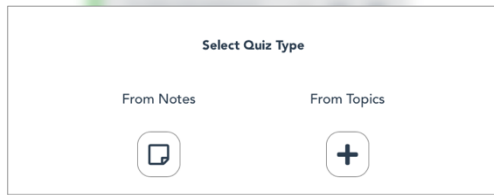


Figure 31 - Iteration 4 – Quiz Type Choice Selection

5.6.4.1 Tutorial Guide

When a new user creates an account and visits their dashboard for the first time, they are greeted with a tutorial guide explaining the features and how to use the application. The tutorial is made up of four pages with a simple navigation system. The added benefit of including this tutorial was explaining how to take advantage of spaced repetition, a scientifically approved time frame for effective study sessions. Users can select a 'Don't show again' tick box, and the application will update a value in the database so that users don't get the tutorial again. (See Appendix – for full tutorial screens)

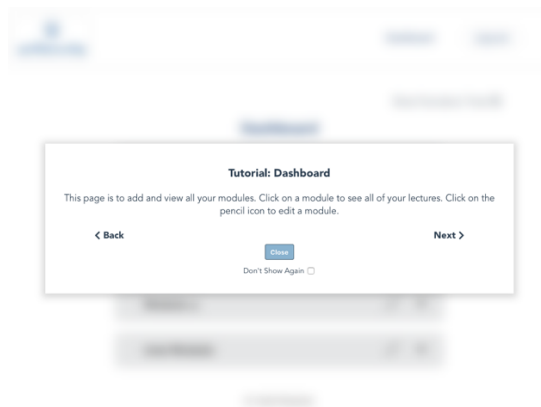


Figure 32 - Iteration 4 – First Page of tutorial guide

Chapter 6 – Conclusion and Discussion

6.1 Project Objectives

The final product of this project marks the development of a web application that satisfies the functional and non-functional requirements. The web application is a novel note-taking platform explicitly designed for university students to take effective notes using a scientifically based strategy. With the feedback received from university students who partook in the user testing phase, it was clear that this application is suitable for all academic fields and solves the crucial problem of taking effective notes for many students.

The front-end user interface and basic functionalities were developed using the Vue-JS framework utilizing HTML, CSS, and JavaScript to design elements displayed to the user. The backend of this application uses Python's FLASK framework to run the advanced neural network technology to generate an MCQ quiz. The front-end communicates with the back end through a R.E.S.T API. Google's cloud firebase platform is used to handle authentication and the database for this project.

6.2 What Was Learnt

After many years of developing fundamental knowledge and various fields in computer science, from Unified Model Language (UML) analysis to neural networks, this project was the first time the author was able to put everything together in an application that solves a real-world problem. With the many layers involved in developing this application, new fields were inevitably covered, and hence, a significant learning curve was required to complete this project successfully.

Although the author had some experience in HTML, CSS, and JavaScript, learning the Vue-Js framework for the front-end took several weeks of tutorials and practice to start working on the web application. Python is the strongest programming language known to the author; however, using transformer models to answer questions and NLP techniques was novel and required the most time to learn the theory behind these systems and how to use them in this project.

In previous years, a local database was used to manage applications developed by the author; however, Google's cloud database was used to manage authentication and the application's data for this project. Google firebase documentation was used to learn how the platform works and integrate it with the application. Initially, the author tried to deploy the application using

Heroku for the R.E.S.T API and Netlify for the front-end; however, the main issue was the charges that had to be made to store large files to handle the transformer model. To resolve this issue, the author tried several solutions, such as using threading; however, with several days of failed attempts, the decision was finalized to leave the program as it is. Although the project was not deployed, the author profited from learning a web application's deployment process.

6.3 Project Reflection

Due to current circumstances with Covid-19, it was anticipated that there might be some features that may not be completed in full before the due date. For this reason, The Arduino System was left for the final iteration. The equipment for the Arduino system was purchased, and many tutorials and courses were used to understand the system; however, with the timeframe left to complete the project, adding this feature in hindsight was over-ambitious. Upon reflection, it was evident that the author had underestimated the amount of time required to learn new frameworks and fundamental knowledge to complete this project successfully. As a result, the most amount of time spent in this project was trying to understand and researching how to use NLP techniques and transformer models to generate a quiz.

6.4 Future Work

The most sensible addition to this project would be a phone application to accompany the web application. As students carry their phones almost everywhere, a phone application will be a great way to answer and revise on the move. The Arduino system was part of the initial iteration plans of the project to serve as a distraction-free desk screen with questions from the database so that students are constantly reminded to answer questions they may be struggling with. In addition to this feature, the Arduino system could have been further developed to include all features a student may need, such as a Pomodoro timer, assignments reminders, motivational quotes, Etc.

As for future work to the web application itself, a subscription service could be added for more advanced features to cover any costs related to APIs, generating better sources of information and passive income on the side.

6.5 Improvements

T5 transformer model and transformer models, in general, are still novel in their discoveries (released in 2017 and 2019 respectively). Although the work done in quiz generation was remarkable in its results, this project only demonstrates the capabilities of these systems, yet there are still several improvements that could be made. First and foremost, it was noticed that on some occasions, the options generated for questions were very similar to each other to the point where multiple options could serve as valid answers to the question. Advanced word disambiguation techniques could be researched and used to solve this issue so that the options are not too similar or too different from the answer.

Moreover, as this application is designed for university students, using Python's Wikipedia library to generate summary texts for questions may not be the best source. Although a scholarly API was tested in the development of this program, the API is still in its infancy stages. Third-party sources developed it with very little community support. A better source of generating questions for the quiz could be used even if it was a paid API service so that university students could maximize their learning from the MCQ quizzes.

References:

[Taylor, K. and Marienau, C., 2016. *Facilitating learning with the adult brain in mind*. San Francisco: Jossey-Bass - A Wiley Brand.]

Bui, D., Myerson, J. and Hale, S., 2013. Note-taking with computers: Exploring alternative strategies for improved recall. *Journal of Educational Psychology*, 105(2), pp.299-309.

Spitzer, H., 1939. Studies in retention. *Journal of Educational Psychology*, 30(9), pp.641-656.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł. & Polosukhin, I. (2017), Attention is all you need, in 'Advances in Neural Information Processing Systems' , pp. 5998--6008 .

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W. & Liu, P. J. (2019), 'Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer' , cite arxiv:1910.10683Comment: Final version as published in JMLR .

Slamecka, N. J., & Graf, P. (1978). The generation effect: Delineation of a phenomenon. *Journal of Experimental Psychology: Human Learning and Memory*, 4(6), 592–604. <https://doi.org/10.1037/0278-7393.4.6.592>

Martin A. Conway & Susan E. Gathercole (1990) Writing and long-term memory: Evidence for a “translation” hypothesis, *The Quarterly Journal of Experimental Psychology Section A*, 42:3, 513-527, DOI: [10.1080/14640749008401235](https://doi.org/10.1080/14640749008401235)

Craik, F. I., & Lockhart, R. S. (1972). Levels of processing: A framework for memory research. *Journal of Verbal Learning & Verbal Behavior*, 11(6), 671–684. [https://doi.org/10.1016/S0022-5371\(72\)80001-X](https://doi.org/10.1016/S0022-5371(72)80001-X)

Moniruzzaman, A B M & Hossain, Syed. (2013). NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. *Int J Database Theor Appl*. 6.

Appendix

Appendix A – Project Definition Document

Project Definition Document - ActiveNotes

Systemized and automated web app that uses science-based strategies and studies to help students take effective notes

Proposed by

Supervisor:

Course: Computer Science BSc

Problem to be solved -

A software system (Dynamic web app) that uses strategic note taking and time management techniques to help students take more effective notes and manage their time during their studies. Note taking is a crucial aspect of a student's life and for many years, many students have had the issue of selecting an effective method to taking notes. There is a plethora of ways to take notes each being effective to a certain extent however, with some research, I noticed that many students from world leading universities (*1) recommended a specific strategy to their note taking process. The strategy is known as active recall and spaced repetition. It's based on reading material (Lectures or textbooks) then consistently asking yourself questions about the topic and later answering the questions without referring to the material. The strategy then requires students to go back to their questions after a set of time (e.g., a week, a month etc) and take note of the questions they get right or wrong and re-do them later on, focusing on the questions they got wrong. I've employed this strategy in my university studies and although it has proven to have helped and much more effective than my previous note taking methods, I noticed an opportunity to systemise and automate this process to make it seamless for students to take notes using this strategy.

Project Objectives -

My project shall use active recall and spaced repetition to allow students to take effective notes. The project interface will be a dynamic web app that students can log in to track their progress and add new work. Essentially the idea of the web app is to include all the modules of the current term. Each module should be able to be clicked and all the lectures for that module should be shown to the user. The student will then be able to go to the specific lecture they're studying and that will take them to the main interface of the web app. In this section there shall be a section that includes questions, answers, and a generate quiz section.

Project complexities -

Questions and answers – Although there will be an option for a user to enter their own questions and answers on the topic they're currently studying, the web app should use deep learning to generate its own questions or answers to questions a student provides. At first, I was thinking of using web scrapping techniques or an API to help with this problem however after some research I noticed that an alternative option would be to allow a student to upload their lecture materials to the website, which will then use deep learning algorithms to go through the material and generate its own questions and answers (or answers to questions a student provides).

Prioritizing recall questions and lectures – Once a student has all the questions and answers to the lectures, they're taking notes on, the next step of the software is to recognise which questions a student should recall next or what lecture a student should re-visit next. This is a very important process in student's studying life especially if a student is behind in their studies. After brainstorming

possible solutions to this problem, I noticed that an excellent solution to this would be to use some sort of priority algorithm to prioritize which lecture a student should re-visit next based on a number of factors for each lecture such as the number of incorrectly answered questions, unfinished lectures, deadline (for CW or exams and how much a student is behind after lecture release date), quiz score etc. With some research I came across a couple of priority algorithms such as the scheduling algorithm or the A* algorithm.

External display (Arduino View) - An Arduino display is an excellent external addition to the web app. Essentially it will be an Arduino system that uses a display to connect to notify and remind what lectures a student must do next and an option to view random questions from their account in a rapid-fire process. The Arduino system will have to connect to a database online to pull out any relevant information.

Database – A database will be essential for this project to store the questions, account information for each student/ user and to track the progress of their work. The database will also be used by the Arduino system to show all the relevant information to the user.

Similar Software -

I've looked into several similar web apps however they all seem to focus on note taking and having the user apply this method. Some examples of these software include Remnotes and Notability. The software my project proposes is to create the web app that automates active recall and spaced repetition for the user with software technologies mentioned above as opposed to the user manually using the technique.

Risk analysis –

The main technical features in this project are the generation of questions using deep learning and using a priority algorithm to give a priority number for each modules and lectures. There are also other technical features such as implementing a database and building and connecting to an external Arduino system. Failure of the main features is the main risk point for this project however in the worst-case scenario that this is the case, I have thought of (and still brainstorming more ideas for) other alternatives to reach a similar solution (e.g., instead of using deep learning to generate the questions, have the user input their own questions and answers).

Work Plan –

After a conversation with my consultant, I was advised the best approach to take for this project was to complete it in four iterations. I have listed what each iteration should consist of below. In the coming weeks and before I start working on an iteration, I will make its own grid plan (similar to the one for iteration #1).

Iteration #1 -

Build a basic version of the system with inputs and outputs

Iteration #1
Research

Start Times
Thu 29th Jan 2021

Extension



Project Write up

Initial research on problem domain – Active recall & spaced repetition

Research & Tutorials on what technologies I will be using.

Python backend (Research the front technologies to use)

Analysis & Design

Mon 1st Feb 2021

Initial GUI Designs

Basic Functionalities

Tue 2nd Feb 2021

Login page

User can enter all modules with corresponding lectures

User can enter questions and answers

Web app should be able to display modules and lectures in any order

Setup an online database

Iteration #2 –

Core-functions – priority algorithms and deep learning techniques for the questions and answers

Iteration #3 –

Any complications (advanced technologies) including the Arduino external display.

Iteration #4 –

Bells & Whistles/ cherry on the top.

Reference -

*1 - How to study for exams - Evidence-based revision tips. (2018). *YouTube*. Available at:

<https://www.youtube.com/watch?v=ukLnPblffxE>.

Appendix B – Participant Information Sheet

Participant Information Sheet

REC Reference Number

Date: __/__/2021

BSc Computer Science

Researcher: [REDACTED]

I would like to invite you take part in my project towards my study. It is important that you understand the research being done with any information or data you may provide. If there are any questions that you may have, please do feel free to ask. By participating in this study, you are helping [REDACTED] towards his 3rd year project – Active Notes, A systemised and automated note taking platform supported by scientific evidence.

This project entails me to develop a web app for university students to use towards their studies to take effective notes. The web app will use scientific based research and evidence to increase the efficiency of the users' note taking techniques. The project will last from 01/01/2021 until 05/05/2021.

You have been asked to participate in this project as you are in suitable position in your academic career to provide me with insightful information to help with the development of this

project. Please understand that your participation, if you chose to do so has no impact on your company, employment or promotion prospects. On Participation, data such as the full name, login details that you create, and feedback or ideas will be recorded. This data will be used to test the functionality of the web app and any feedback will be taken into consideration during development. In the occasion where the feedback of a participant has influenced a decision, the feedback data will be recorded in the project report otherwise all data will be destroyed and permanently removed once the development of the project has finished.

By no means do you HAVE to participate towards this project, any information/participation in this project on your behalf is voluntary. You can choose not to participate in this project at all. You are free to withdraw from this project at any stage without penalty. If you do decide to participate in this project you will be asked to sign a consent form. If you later decided, you would like to withdraw you are free to do so without giving a reason.

This study has been approved by City, University of London School of Mathematics, Computer Science & Engineering Research Ethics Committee.

If you have any problems, concerns or questions about this study, you should ask to speak to a member of the research team. If you remain unhappy and wish to complain formally, you can do this through City's complaints procedure. To complain about the study, you need to phone 020 7040 3040. You can then ask to speak to the Secretary to Senate Research Ethics Committee and inform them that the name of the project is: [Active Notes, scientific based note taking platform.] You can also write to the Secretary at:

Anna Ramberg
Research Integrity Manager
City, University of London, Northampton Square
London, EC1V 0HB
Email: Anna.Ramberg.1@city.ac.uk

Thank you for taking time to read this information sheet

[Redacted signature]

[Redacted text]

[Redacted text]

[Redacted text]

[Redacted box]

or
initial box

[Redacted box]	[Redacted box]
[Redacted box]	[Redacted box]
[Redacted box]	[Redacted box]