

**Lifelong Learning**

**Learning Beyond The Classroom**

**Submitted by: Ho Bing Hui**

**Matriculation Number: U1721376E**

**Supervisor: Dr Shao Xuguang**

**Examiner: Assoc Prof Tang Xiaohong**

School of Electrical & Electronic Engineering

A final year project report presented to the Nanyang Technological University

in partial fulfilment of the requirements of the degree of

Bachelor of Engineering

2021

# **Table of Contents**

[**Table of Contents** ii](#_Toc68711025)

[**Abstract** v](#_Toc68711026)

[**Acknowledgements** vi](#_Toc68711027)

[**Acronyms** vii](#_Toc68711028)

[**List of Figures** viii](#_Toc68711029)

[**Chapter 1 Introduction** 1](#_Toc68711030)

[1.1 Background and Motivation 1](#_Toc68711031)

[1.2 Objectives 2](#_Toc68711032)

[1.3 Report Organization 4](#_Toc68711033)

[**Chapter 2 Literature Review** 5](#_Toc68711034)

[2.1 Web Application 5](#_Toc68711035)

[2.1.1 Front-End Development 6](#_Toc68711036)

[2.1.2 Back-End Development 6](#_Toc68711037)

[2.1.3 Database 7](#_Toc68711038)

[2.2 Email API Service Provider 8](#_Toc68711039)

[2.3 Cloud Hosting Providers 9](#_Toc68711040)

[**Chapter 3 Common Requirements** 10](#_Toc68711041)

[3.1 Non-Functional Requirements 11](#_Toc68711042)

[3.1.1 Web Responsiveness 11](#_Toc68711043)

[3.1.2 Usability 12](#_Toc68711044)

[3.2 Front-end Design and Implementations 13](#_Toc68711045)

[3.2.1 Overall Sitemap Design 13](#_Toc68711046)

[3.2.2 Logic Design 14](#_Toc68711047)

[3.2.3 Layout Design 15](#_Toc68711048)

[3.2.4 Code Structure 16](#_Toc68711049)

[3.2.5 Login/Registration View 18](#_Toc68711050)

[18](#_Toc68711051)

[3.2.6 Statistic View 21](#_Toc68711052)

[3.2.7 Achievements View 23](#_Toc68711053)

[3.2.7 Profile View 23](#_Toc68711054)

[3.3 Database & Back-end Implementations 24](#_Toc68711055)

[3.3.1 Back-end Design 24](#_Toc68711056)

[3.3.2 Database Design and Considerations 26](#_Toc68711057)

[3.3.3 User Collection 27](#_Toc68711058)

[3.3.4 Statistic and Achievement Collection 28](#_Toc68711059)

[3.3.4 Cron Job 30](#_Toc68711060)

[**Chapter 4 Design and Implementation for Leaners** 30](#_Toc68711061)

[4.1 Learner’s Requirements 30](#_Toc68711062)

[4.2 Front-end Implementations 31](#_Toc68711063)

[4.2.1 Browse and Register for Courses 31](#_Toc68711064)

[4.2.2 Discussion Boards 33](#_Toc68711065)

[4.2.3 Course Classes 36](#_Toc68711066)

[4.3 Database & Back-end Implementations 37](#_Toc68711067)

[4.3.1 Registration Collection 37](#_Toc68711068)

[4.3.2 Discussion Board collections 38](#_Toc68711069)

[4.3.3 Class Board Collection 39](#_Toc68711070)

[4.3.4 Quiz Results Collection 40](#_Toc68711071)

[**Chapter 5 Design and Implementation for Instructors** 41](#_Toc68711072)

[5.1 Instructor’s Requirements 41](#_Toc68711073)

[5.2 Front-end Implementations 41](#_Toc68711074)

[5.2.1 Course Creation and Edit 41](#_Toc68711075)

[5.2.2 Survey/Quiz module 44](#_Toc68711076)

[5.2.3 Discussion Board 48](#_Toc68711077)

[5.2.4 Class Board 48](#_Toc68711078)

[5.3 Database & Backend Implementations 49](#_Toc68711079)

[5.3.1 Course Collection 49](#_Toc68711080)

[5.3.2 Survey Collection 50](#_Toc68711081)

[5.3.3 Quiz Collection 51](#_Toc68711082)

[**Chapter 6 Design and Implementation for Administrators** 53](#_Toc68711083)

[6.1 Administrator’s’ Requirements 53](#_Toc68711084)

[6.2 Front-end Implementations 53](#_Toc68711085)

[6.2.1 Home Page 53](#_Toc68711086)

[6.2.2 Data Table List 54](#_Toc68711087)

[6.2.3 Instructors applications 56](#_Toc68711088)

[6.2.4 Report Review 57](#_Toc68711089)

[6.3 Database & Back-end Implementations 58](#_Toc68711090)

[6.3.1 CRUD Operations 58](#_Toc68711091)

[**Chapter 7 Application Testing** 60](#_Toc68711092)

[**Chapter 8 Conclusion and Future Work** 68](#_Toc68711093)

[8.1 Conclusion 68](#_Toc68711094)

[8.2 Future Work 68](#_Toc68711095)

[**References** 69](#_Toc68711096)

[**Appendix** 70](#_Toc68711097)

[71](#_Toc68711098)

# **Abstract**

There is an increasing popularity in lifelong learning. Many institutes are providing courses to help enhance the skills of the workforce. The Singapore government is also putting in more attention to lifelong learning by providing incentives for the people to take these courses.

Courses provided by institutes are usually conducted physically. Although online learning is becoming popular, it will not replace physical classes in the near future. This project aims to supplement physical classes by providing a platform where users continue to receive help even after the class has ended. It also aims to encourage continuous learning.

Through this project application, users will be able to register for lifelong learning courses and have access to a discussion board where they can communicate with other users and discuss questions or topics related to the course.

# **Acknowledgements**

I would like to express my gratitude to my supervisor, Dr Shao Xuguang, for her continuous support for the past one year. She provides valuable feedback throughout the whole process from idea generation to product implementation.

# **Acronyms**

SPA Single Page Application

DOM Document Object Model

UI User Interface

JSX JavaScript XML

HTML Hypertext Markup Language

CSS Cascading Style Sheet

PHP PHP: Hypertext Preprocessor

SQL Structured Query Language

API Application Programming Interface

AWS Amazon Web Services

GCP Google Cloud Platform

DB Database

JSON JavaScript Object Notation

CRUD Create, Read, Update, Delete

HTTP Hypertext Transfer Protocol

URL Uniform Resource Locator

# **List of Figures**

[Figure 1 MERN stack architectural diagram 5](#_Toc67605510)

[Figure 2 Database trends of popular NoSQL databases 8](#_Toc67605511)

[Figure 3 Venn Diagrams for roles requirements 10](#_Toc67605512)

[Figure 4 Media query example 12](#_Toc67605513)

[Figure 5 Fluid typography formula 12](#_Toc67605514)

[Figure 6 Main Application Sitemap 13](#_Toc67605515)

[Figure 7 Admin Application Sitemap 14](#_Toc67605516)

[Figure 8 Registration module flowchart 14](#_Toc67605517)

[Figure 9 Login and navigating flowchart 15](#_Toc67605518)

[Figure 10 Application template Page 16](file:///C:\Users\Binghui\Desktop\fyp\FYP%20Final%20Report%20Draft%20v2.docx#_Toc67605519)

[Figure 11 App and file structure for a Vue application 17](#_Toc67605520)

[Figure 12 Signup page with email validation 18](#_Toc67605521)

[Figure 13 Selection of interest 19](#_Toc67605522)

[Figure 14 Avatar creation 19](#_Toc67605523)

[Figure 15 Creation of password 20](#_Toc67605524)

[Figure 16 Login page 20](#_Toc67605525)

[Figure 17 Confirmation code page to change password 21](#_Toc67605526)

[Figure 18 Statistic View 21](#_Toc67605527)

[Figure 19 Line graph of number of learners registered. 22](#_Toc67605528)

[Figure 20 Line graph for courses registered 22](#_Toc67605529)

[Figure 21 Achievement page 23](#_Toc67605530)

[Figure 22 Sample image of Profile view 24](#_Toc67605531)

[Figure 23 Code block that initialize the backend environment 24](#_Toc67605532)

[Figure 24 Console log of successful initialization 25](#_Toc67605533)

[Figure 25 Backend API endpoint 25](#_Toc67605534)

[Figure 26 Code block to call backend endpoint 26](#_Toc67605535)

[Figure 27 SQL vs NoSQL terminology 26](#_Toc67605536)

[Figure 28 Data Schema for user collection 27](#_Toc67605537)

[Figure 29 Code to determine login 28](#_Toc67605538)

[Figure 30 Statistic collection schema 29](#_Toc67605539)

[Figure 31 Achievement collection data schema 29](#_Toc67605540)

[Figure 32 Code that runs the cron job 30](#_Toc67605541)

[Figure 33 Browse course view + search function 32](#_Toc67605542)

[Figure 34 Code block of course card component 32](#_Toc67605543)

[Figure 35 Course details view 33](#_Toc67605544)

[Figure 36 Discussion boards available to the user 34](#_Toc67605545)

[Figure 37 Discussion View for a course 34](#_Toc67605546)

[Figure 38 Creating a new thread to ask question 35](#_Toc67605547)

[Figure 39 Thread messages view 35](#_Toc67605548)

[Figure 40 Course Class view 36](#_Toc67605549)

[Figure 41 Learners can access and complete quiz 37](#_Toc67605550)

[Figure 42 Data schema for registration collection 38](#_Toc67605551)

[Figure 43 Data schema for threads collection 38](#_Toc67605552)

[Figure 44 Data schema for messages collection 39](#_Toc67605553)

[Figure 45 Data schema for class collection 39](#_Toc67605554)

[Figure 46 Data schema for quizResults collection 40](#_Toc67605555)

[Figure 47 Sample view for course creation 42](#_Toc67605556)

[Figure 48 Course details page 42](#_Toc67605557)

[Figure 49 Course Edit view 43](#_Toc67605558)

[Figure 50 Option to port files and start new batch 43](#_Toc67605559)

[Figure 51 Code sample for survey builder 44](#_Toc67605560)

[Figure 52 Survey builder page 45](#_Toc67605561)

[Figure 53 Quiz builder page 45](#_Toc67605562)

[Figure 54 Sample code to add question 46](#_Toc67605563)

[Figure 55 Survey Result statistics 46](#_Toc67605564)

[Figure 56 Quiz result statistics 47](#_Toc67605565)

[Figure 57 Quiz marker 47](#_Toc67605566)

[Figure 58 Instructor’s discussion board view 48](#_Toc67605567)

[Figure 59 Instructor's class board view 48](#_Toc67605568)

[Figure 60 Student list per class 49](#_Toc67605569)

[Figure 61 Data schema for courses collection 50](#_Toc67605570)

[Figure 62 Data schema for survey collection 50](#_Toc67605571)

[Figure 63 Code to Retrieving survey results 51](#_Toc67605572)

[Figure 64 Code to aggregate and format the data 51](#_Toc67605573)

[Figure 65 Code to retrieve quiz data to mark. 52](#_Toc67605574)

[Figure 66 Code to retrieve quiz results 52](#_Toc67605575)

[Figure 67 Code to aggregate quiz results 52](#_Toc67605576)

[Figure 68 Home page of admin portal 54](#_Toc67605577)

[Figure 69 List of learners 54](#_Toc67605578)

[Figure 70 List of instructors 55](#_Toc67605579)

[Figure 71 Admin creating a new user 55](#_Toc67605580)

[Figure 72 List of courses 56](#_Toc67605581)

[Figure 73 List of applications 56](#_Toc67605582)

[Figure 74 Learner's application information 57](#_Toc67605583)

[Figure 75 Admin can review report on discussion messages 57](#_Toc67605584)

[Figure 76 Message deleted by admin 58](#_Toc67605585)

[Figure 77 Code block to insert a new user document 58](#_Toc67605586)

[Figure 78 Code block to send an email to user 59](#_Toc67605587)

[Figure 79 Update the related documents when changing role to instructor 59](#_Toc67605588)

[Figure 80 Update discussion message if report was approved 60](#_Toc67605589)

# **Chapter 1 Introduction**

## Background and Motivation

The way we live, work and learn are dependent on technologies. And as we know, technologies are always changing and improving. In order to not be left behind, we have to continuously update our knowledge and skills. This makes learning a continuous process and no longer fixed to a certain period of our lives. Learning new skills or improving existing skills are very important in ensuring our employability. This is especially so during period of economic shifts or unexpected circumstances such as the recent Covid-19 pandemic [1].

The importance of life-long learning can be seen in these times where employment rate is low, and the level of skills will determine whether you will get or keep your jobs. Singapore government have also been promoting life-long learning and supporting it by providing the citizens with skillsfuture credits which can be used to take up classes from myskillsfuture portal [2].

There are many avenues to learn new skills, be it through online learning platform such as Udemy, Coursera, Skillshare or physical lessons through institutes such as SMU Academy or NTU MiniMaster. A certificate will usually be provided to show that you have completed the course, and this can be used to help show your competency is the skills. However, online learning requires great self-discipline to complete the course and also to continue using what was learn so that we will not forget the skills [3]. Therefore, I find that there is a need to encourage not only life-long learning but continuous learning.

## Objectives

The project aims to design and develop a web application to supplement physical classes of lifelong learning by encouraging and motivating learners to continue learning and improving their skills. This is done by providing a community for users to discuss and learn from one another through a discussion platform where you can answer and ask a question related to the topic of the course.

To encourage users to stay and remain active on the platform, a small amount of gamification elements is added to provide the interest and drive to continue learning and using the platform.

The project aims to design and develop a fully functional web application that will enable users to do the following.

There are a few key checkpoints that this project will be working towards.

1. Functionalities of a standard web applications. For example, logging and registration of users.
2. Functionalities of a standard learning platform. For example, browsing and registering for courses.
3. Allow instructors to create/manage courses and manage course classes (See point 5).
4. Discussion section for users/instructors to ask and answer related questions
5. Class section to support physical classes through allowing Q&A, quiz assessment and disseminating course materials.
6. Gamification elements which allow users to gain points with most actions taken on the application.
7. An admin management portal to manage users or issues.

There will be two portals of the application. The main portal that users can access and an admin portal that is used for admin management. Below will briefly show what each user is able to do on the application.

**Learners** will be the main users of our platform.

1. They will be able to view and register for courses that are provided on the platform.
2. Discussion board for the individual course will be accessible when learners register for a course.
3. They will also be able to see their statistic and achievement on the platform.
4. Class section will be available and accessible within a week to the start of a registered course.

**Instructor** will be the second main user of our platform. Instructor will have the same features as learners including the rights to create and manage their personal courses.

1. They will be able to view and edit their own courses
2. They will be able to access Discussion board of their own course or their registered courses
3. They will be able to access the class section of their course or registered course.

**Admin** will be managing the data of the platform. Main data that an admin can manage is user and course data.

1. Able to view all data on the platform. Such as list of users, list of courses and reports list.
2. Able to create or delete users.
3. Able to approve or ignore reports made by users.

## 1.3 Report Organization

The report consists of eight chapters. The first chapters of the project give an overview of the project. It includes the motivation, objectives, scopes and report organization.

The second chapter will be the literature reviews. It discusses the current style of web development and the different tools that are mainly used by web developers. Here, I also talked about the cloud services I used to store data and also the mailing services to send out emails to the users automatically.

The third chapter we will look at the common requirements of the application and the different roles. We will also look at how the frontend and backend is developed for these requirements.

The fourth chapter we will look at the learner’s requirements and how the frontend and backend was developed.

The fifth chapter we will look at the instructor’s requirements and how the frontend and backend was developed.

The sixth chapter we will look at the administrator’s requirements and how the frontend and backend was developed.

The seventh chapter we will look application testing. What are the test cases for each functional requirements that was listed previously, their expected result and the actual result.

Lastly, the final chapter will be on the conclusion of the project, highlighting major achievements and also future improvements of the projects.

# **Chapter 2 Literature Review**

## 2.1 Web Application

A web application is a program that is stored on a server the delivered over the internet through a web browser. Building a web application consist of three parts, the front-end, back-end and database. The front-end of the website is the part where the user interacts with. It usually focuses on user experience and responsiveness. The back-end is part of the application that the user does not see. It enables the application to run, store and organize data. It also manages the interaction between the front-end and the back-end to send and receive information to be displayed to the user.

A popular JavaScript technological stack is the MERN stack. It is a traditional 3-tier architectural design using JavaScript. There are different variations to the technological stack by changing out the frontend framework [4].

M – MongoDB as a NoSQL database  
 E – Express.js as a backend framework   
 R – React as a frontend framework  
 N – Node.js as a web server

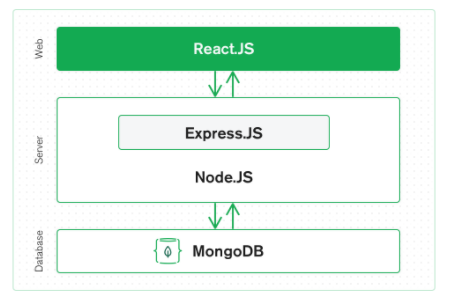


Figure MERN stack architectural diagram

### 2.1.1 Front-End Development

Single Page Applications (SPA) have been gaining popularity when designing a web application. SPAs dynamically reload the current page instead of loading the entire new page from the server. This minimizes user interruption and improves user experience. This also helps to maintain consistency for the application as codes are reused instead of rewriting it on another page. Development time can also be greatly reduced [5]. There are three main frameworks that help to build a SPA, namely React, Angular, and Vue.

React is a frontend library developed by Facebook. It is a tool for building UI components. React utilized a virtual DOM for faster DOM manipulation. It uses JSX syntax, a JavaScript extension to make it easier to add HTML in React. This also means that to use React, we have to take time to pick up JSX syntax.

Angular is a frontend framework for building SPA. It is developed by Google. Angular uses HTML and TypeScript. It uses a mechanism called change detection to detect changes and update the DOM accordingly [6].

Vue is a frontend framework developed by an ex-Google employee, Evan You. Vue also uses a virtual DOM to manipulate changes. Vue uses HTML-based templates which can be easier to pick up [7].

For this project because it is easier to pick up and I am familiar with it. Even though it has lower popularity compared to the other two frameworks, it is growing steadily.

### 2.1.2 Back-End Development

Backend is the server-side of the web application. It focuses on how the web application will work. It mainly holds the business logic of the application It handles the communication between the front-end interaction made by the user with the database and also manages user connection to the web application [8].

There is a variety of backend frameworks for different languages. For example, Laravel for PHP, Django for Python or Express for JavaScript. For this project, I will be using Express. Express is a fast, unopinionated, minimalist web framework based on Node.js. As Express is primarily written using JavaScript, the development process will be more seamless with both the front-end and the back-end sharing the same programming language.

### 2.1.3 Database

There are two main types of databases, relational and non-relational databases. A relational database is also known as a SQL database that stores data in a rigid and structured way. It was developed by IBM in the 1970s and consists of uses two or more tables with columns and rows. Each row represents a data entry, and each column represents an attribute. It also uses a primary key and foreign keys to establish relations between the tables [9]. However, SQL databases are usually more expensive to query due to a complex number of tables and relations also, it scales up vertically as more data are added, meaning more resources are required [10].

Non-relational database also known as NoSQL database stores data in a document style as compared to a table in relational databases. Each document is a single data entry, and it can store non-structured data. As there is no need for each entry to follow the same schema, it is more flexible but requires extra processing effort and storage compared to a more organized SQL database [9].

For this project, I will be using MongoDB which is one of the more popular NoSQL databases. MongoDB provides more flexibility, scales horizontally and supports a large volume of data and traffic.

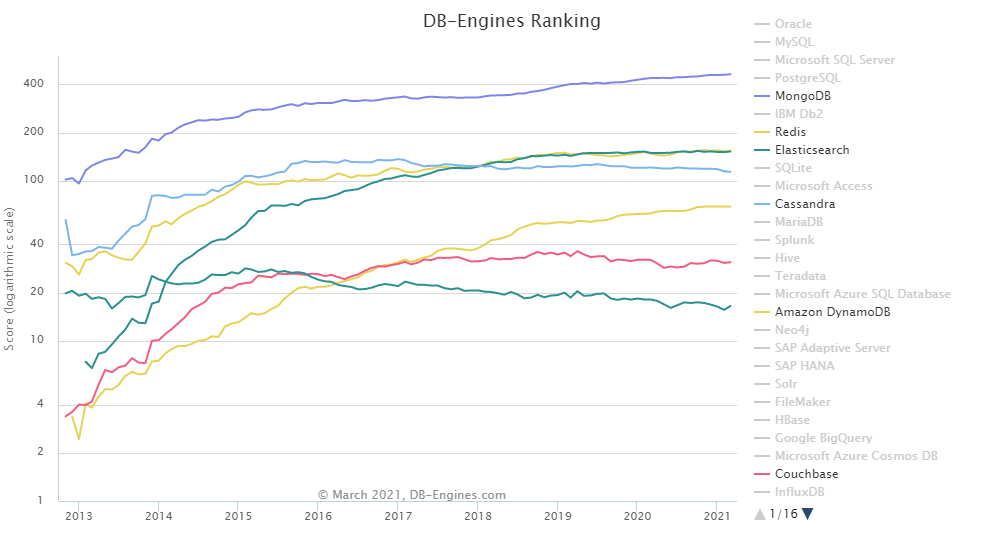


Figure Database trends of popular NoSQL databases

## 2.2 Email API Service Provider

Email is a standard form of communication between the business and its clients. However, manually sending emails can be inefficient and error-prone. There is a need to use Email API to help automate email sending, such as generating and sending transactional emails [11].

Some common use cases that require email are

1. When a user forgot his password and the server sent a confirmation code to the registered email
2. When a user completed a payment or a reminder for the payment.
3. When a user successfully registered for an account.

Most of these situations that requires an email have a fixed template and this can be easily automated with the help of an email API. There are two email API providers that I know and considered, Mailgun and SendGrid.

SendGrid is a market leader in email delivery with over 80,000 paying customers and sends over 70 billion emails every month. They have Airbnb and Uber as their clients. SendGrid has a more complex email service as it provides different ways for a developer to use its product. SendGrid also has a dedicated product for marketing emails and offers all the features a marketeer requires [12].

Mailgun provides its services to over 225,000 businesses and has Microsoft and Lyft as their clients. They market themselves as a service for developers. It is much easier to use their services as they provide a good documentation on their services and also multiples libraries to enable developers to integrate their services with different languages [12].

For this project, I have chosen SendGrid as my email API service provider mainly because it provides free 100 emails per month permanently. This is beneficial during my development and testing phase as I do not need to worry about the cost.

## 2.3 Cloud Hosting Providers

Web applications are required to be hosted on the internet so that users can access them around the world. This can be done through cloud computing, which is the delivery of services such as servers, databases, storage, and networking over the internet. Cloud computing is the current industry standard of hosting applications. Benefits such as reduce expenses, security, scalability, performance and reliability greatly out scales the traditional way of hosting applications [13]. The three main cloud service providers by market share[14] are, Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform (GCP) with AWS being the market leader. They all provided the necessary services that support cloud computing such as compute, databases, storage and networking services.

AWS being the market leader in cloud computing, its technical expertise and tools are vast and unmatched. It focuses on a public cloud instead of a hybrid or private cloud which can interoperate with physical data centers.

Azure, the second market leader is aimed at enterprises that have little expertise in cloud computing. It focuses on a hybrid cloud that will interoperate with enterprise data centers.

GCP which enters the market slightly later is also an upcoming competition to the established AWS or Azure. It incorporates its industry-leading tools in machine learning, artificial intelligence and data analytics into its platform.

# **Chapter 3 Common Requirements**

When designing an application, we will consider the requirements of our application. There are two types of requirements, functional and non-functional. Functional requirements are the services that the application should offer. It is the main core of the application as it determines what the user can do on the applications [15]. Non-functional requirements are the attributes of the application. Responsiveness, usability security and others fall under this category [16].

There are 3 main users on this application. They are the learners, instructors and admin. Each of these roles will determine what functions they have access to on the application. Therefore, we can divide the requirements into different roles.

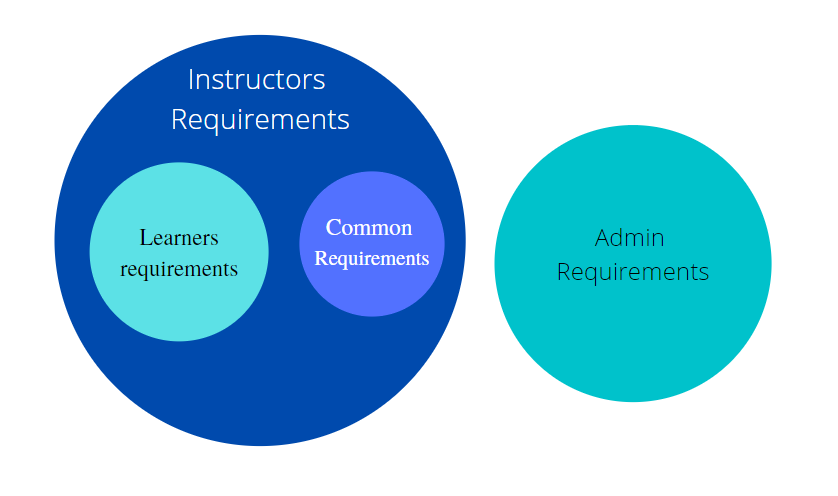


Figure Venn Diagrams for roles requirements

Common requirements are general requirements that is offered by the application. These includes non-functional requirements and the list of requirements as seen from the table below.

|  |  |
| --- | --- |
| No. | Description |
| FR1 | Users should be able to sign up with their email |
| FR2 | Users should be able to log in to their accounts |
| FR3 | Users should be able to log out of the application |
| FR4 | Users should be able to view their statistics |
| FR5 | Users should be able to view their achievements |
| FR6 | Users should be able to view their and others profile page |
| FR7 | Class status should be check daily at midnight |

Table 1 List of common requirements

## 3.1 Non-Functional Requirements

### 3.1.1 Web Responsiveness

Web responsiveness is an important aspect of web development as it makes the application flexible across different devices and resolutions. This provides the maximum experience for the user as the content scales automatically without needing any inputs from the users [17].

In most cases, when we are developing a web application, we will use a CSS library. For example, Bootstrap or Material UI. These libraries come with in-built responsiveness. However, this is limited to the things from the libraries.

Media queries are a feature of CSS that helps web application to scale according to different screen size. This is a fundamental part of responsive design as developers are able to customize the different breakpoints and layout of the application [18].

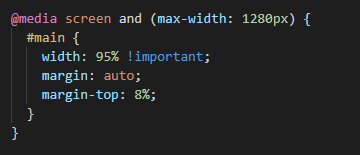


Figure Media query example

Looking at the image above, the above query activates when the user screen size is 1280 pixels wide or less. From this, we can declare different CSS styling at different screen sizes to scale the objects being displayed on the web page.

Typography is also one aspect to make the web application responsive. The font size of the typography should also scale according to the device screen size. We could use media queries, but it will not be efficient. We could use a CSS function ‘calc()’ to calculate the font size base on the screen size. The math formula was created by Mike Riethmuler [19].

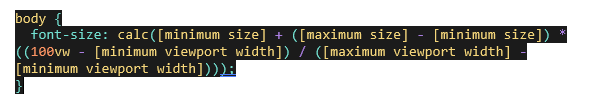


Figure Fluid typography formula

Setting the maximum and minimum font-size, we are able to use the formula to scale the font-size dynamically between the maximum and minimum screen size.

### 3.1.2 Usability

Web usability is the ease of use of the website or application. One key aspect of usability is web responsiveness. If the application can be used on different devices, it becomes easier for users to access it. This will increase the usability of the application. As discussed previously, web responsiveness was taken into considerations when building this project.

Another aspect of usability is clarity. Web design should be simple and intuitive. It should be easy and logical for the user to navigate through the website. This can be done through the type of layouts used.

The color scheme also affects the usability of the website. Having too many colors can be distracting but too few colors can make the website look dull. A simple design can focus on one color and varies the hues and shade.

## 3.2 Front-end Design and Implementations

### 3.2.1 Overall Sitemap Design

Before starting to build the application, we need to plan how our application will be navigated. A sitemap is a development road map that helps developers have a clear overview of the application. It includes the goals of the application, the different features or pages and provides a clear direction for the development.

The sitemap shown below outlines the structure of my application. It shows how each user will navigate through my application and the different features that are included.

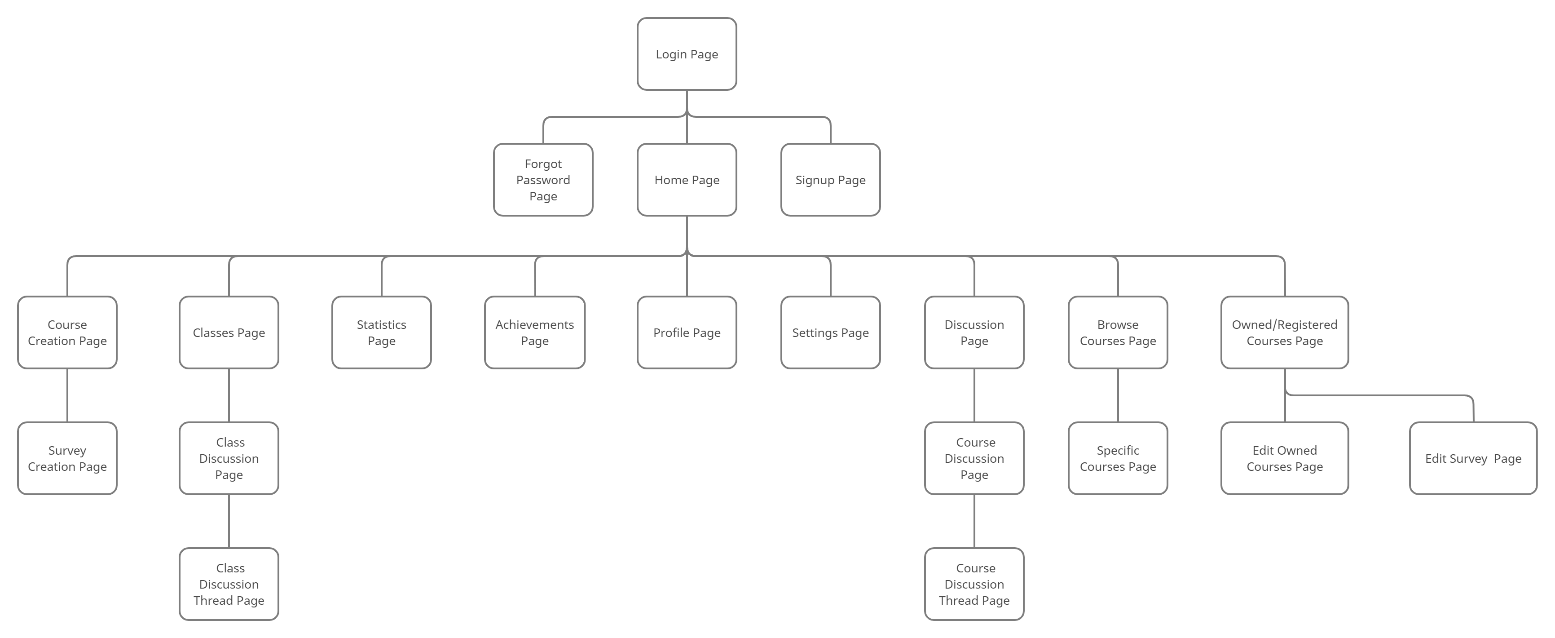


Figure Main Application Sitemap

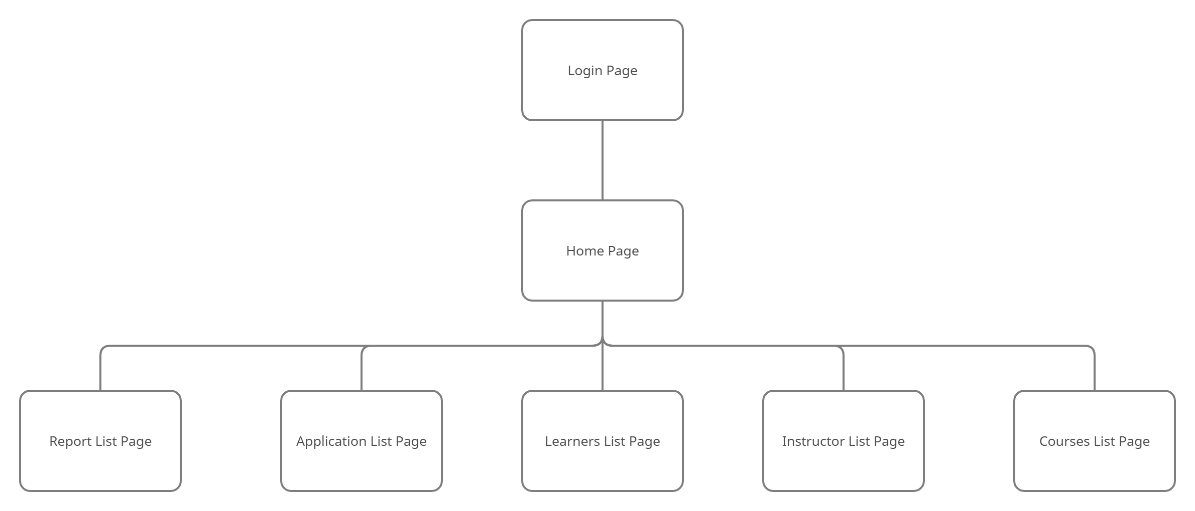


Figure Admin Application Sitemap

### 3.2.2 Logic Design

After planning the sitemap, we will need to plan the programming logic of our application. We will have a login and registration module to collection user information that will be used by the application.

There are also two types of roles, learners and instructor. The application will serve the pages accordingly to the roles of the users. We will look at the logic flow diagram for the login module and the logic flow for determining the pages to serve.

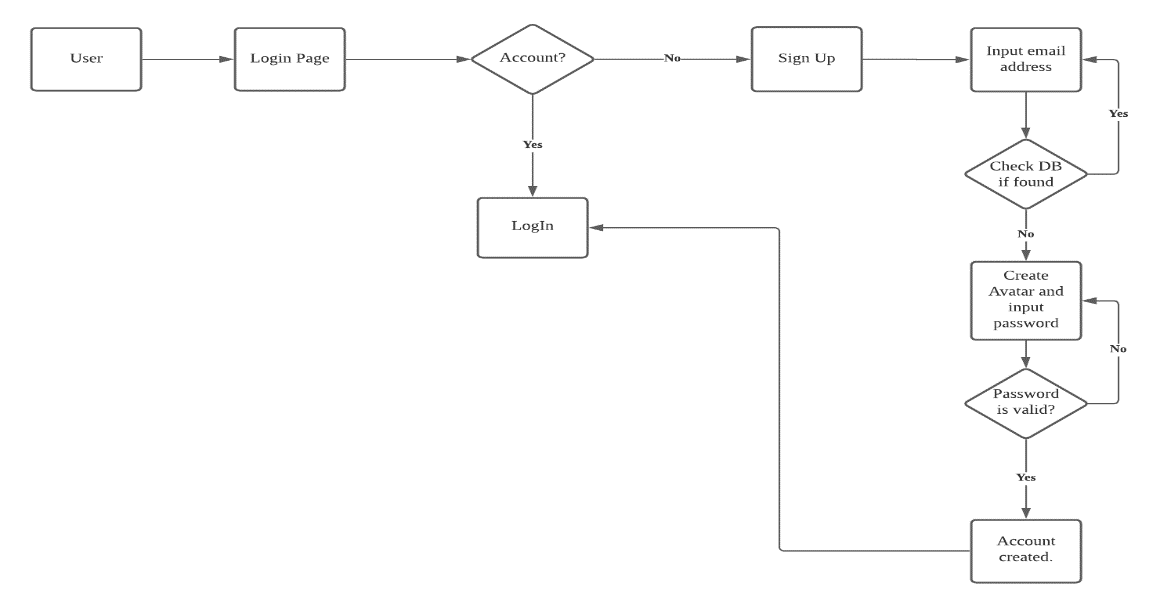


Figure Registration module flowchart



Figure Login and navigating flowchart

Based on figure 8, you can see that when a user logs in, the application will check for their role. From there, it will decide what content to be served. This process is then continued whenever the user navigates through the application. This is because SPA does not change the whole HTML file but instead only changes part of it. This is done by manipulating the virtual DOM by the Vue framework.

### 3.2.3 Layout Design

Layout design is important to ensure that the application is consistent in its looks. It gives the developers and clients a clearer overview of how the application will look like. Since this is a SPA, I used the single page layout where the navigation elements remain on the page. All the dynamic contents are served according to the user inputs. This let the web application to look clean, neat and sharp as it is not cluttered with too many objects to look at. This makes navigating the web application more intuitive.

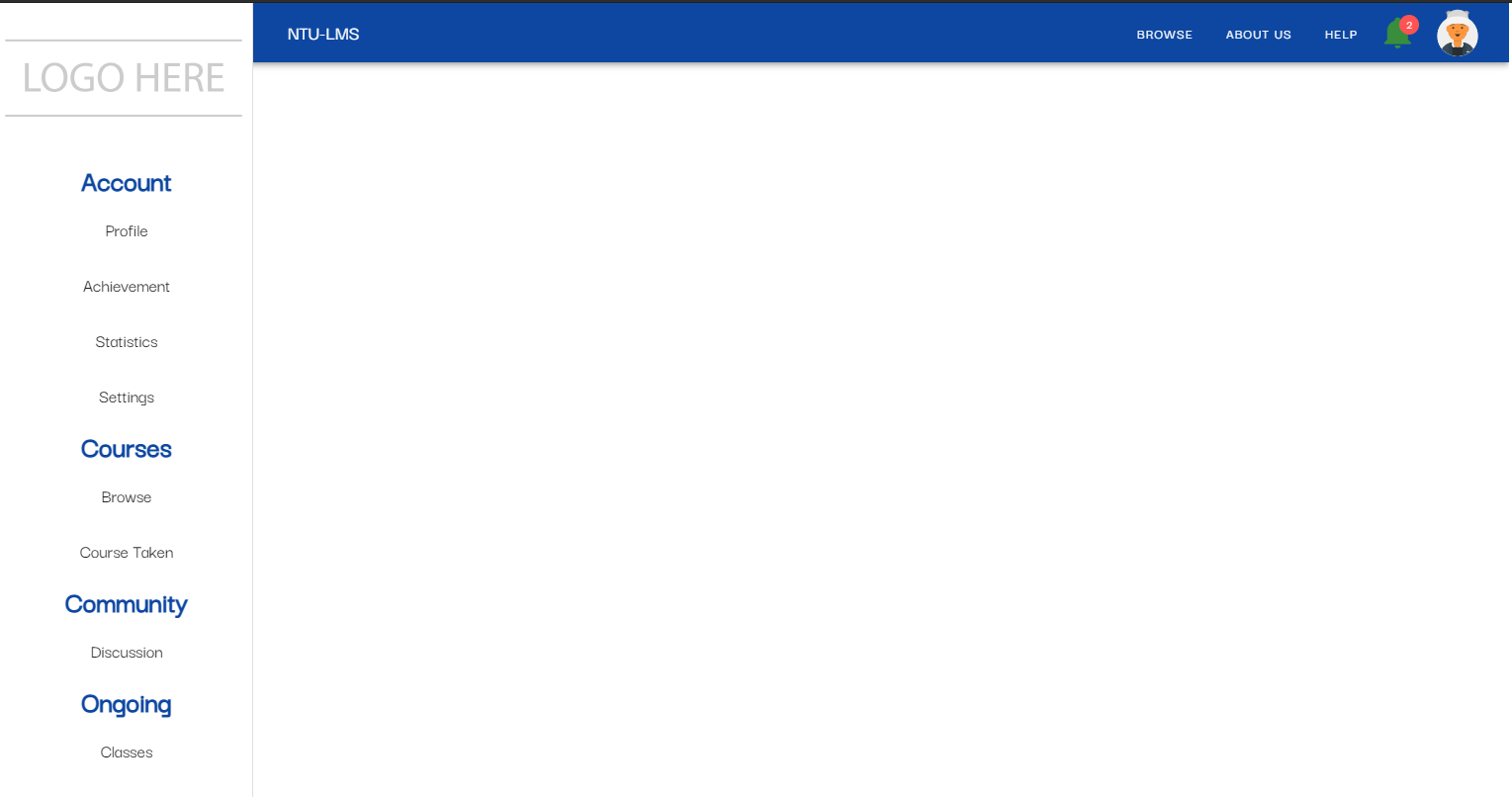


Figure Application template Page

From the image above, you can see a sample of how the application layout looks like. The sections that are highlighted by red color are the constant elements and the section highlighted by blue is dynamic.

There is a navigation drawer on the left side of the page. Users will mainly use this to navigate through the application. At the top, there is also another bar which is a secondary navigation bar. There are some overlaps between them but the main function for the top bar is the notification bell, profile logo and application logo.

### 3.2.4 Code Structure

Implementing the page views is one of the main effort for my project. I have to create the different views that the users will interact with from scratch. Vue framework is easy to use and learn as it follows basic HTML and CSS syntax.

In Vue, the constant contents are stored under layouts folder and dynamic contents are stored under the pages folder. A Vue file contains three sections. The template section, the script section and the style section. The syntax of these follows HTML and CSS.

To reduce repeated codes, we can also use Vue components which are reusable instance to create repeating objects or elements. This help to keep the code clean and manageable.

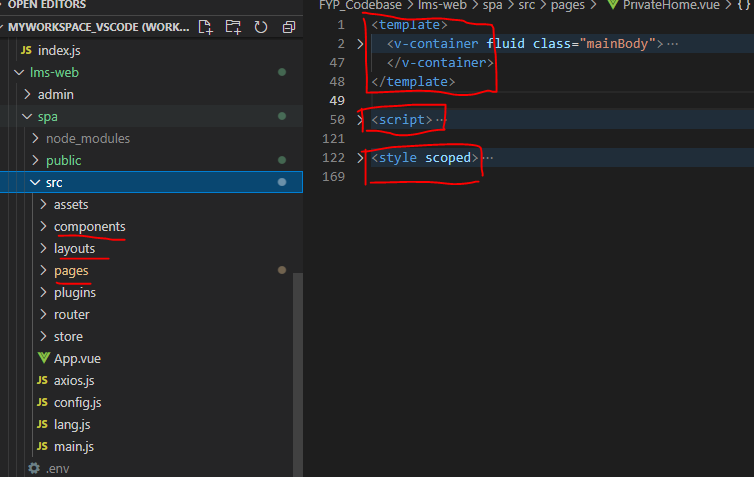


Figure App and file structure for a Vue application

### 3.2.5 Login/Registration View

Login and registration page is an important function of a web application. It enables you to track your users’ data and also manages access to your application.

Here, we enable the user to create an account using their email. Email used should be unique and it will be checked with the database to ensure that no accounts have been registered. Standard email validation is in place to check for valid email syntax. Once confirm, the user will proceed to select their interest, create an avatar and create a password for their account. Password creation also has standard validation.

### 

Figure Signup page with email validation

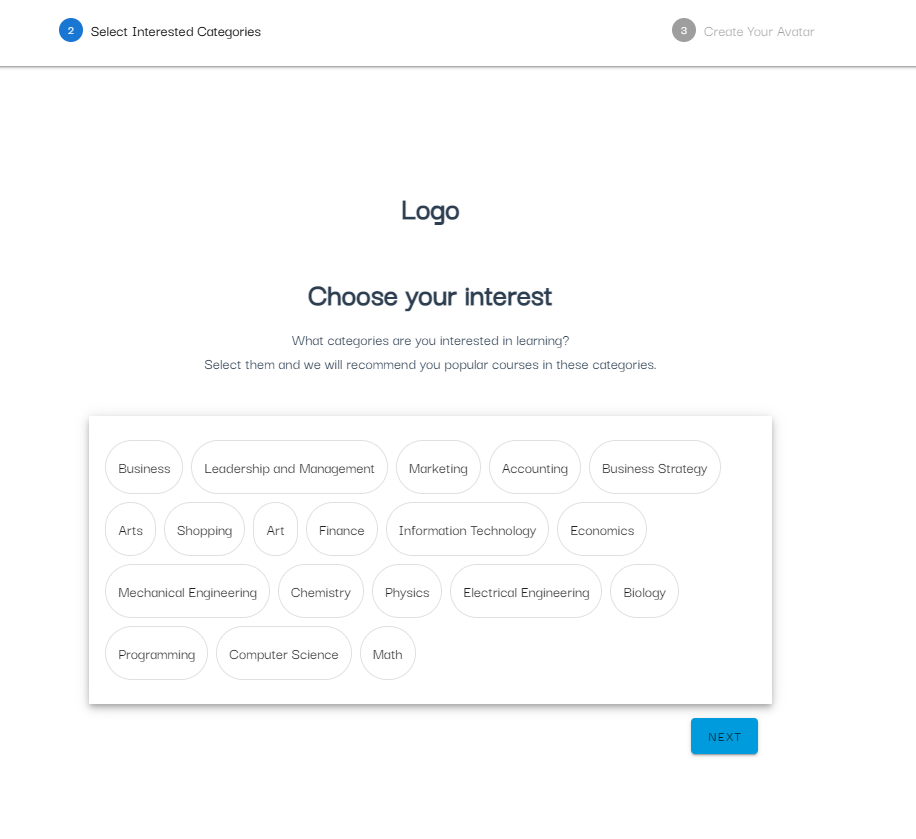


Figure Selection of interest

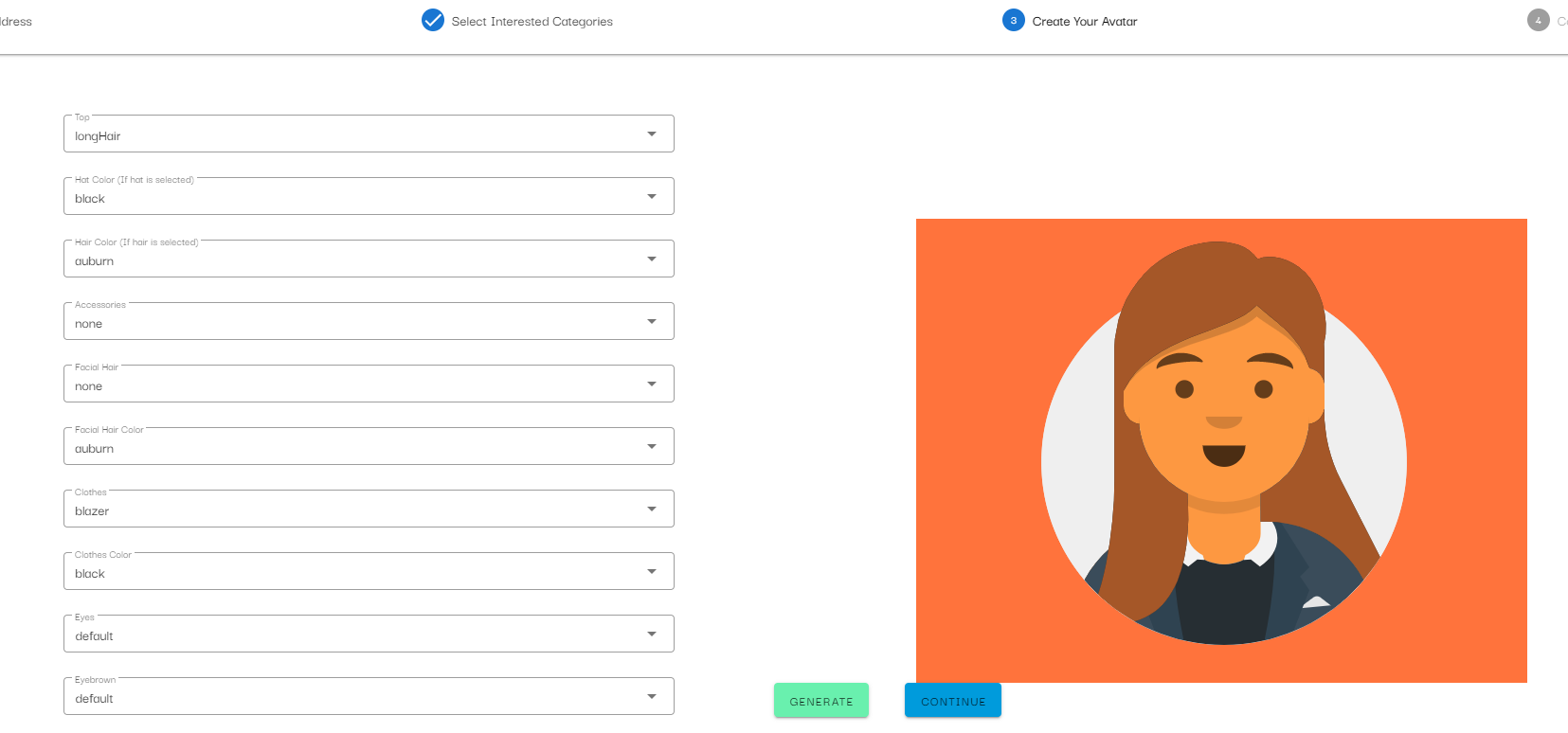


Figure Avatar creation

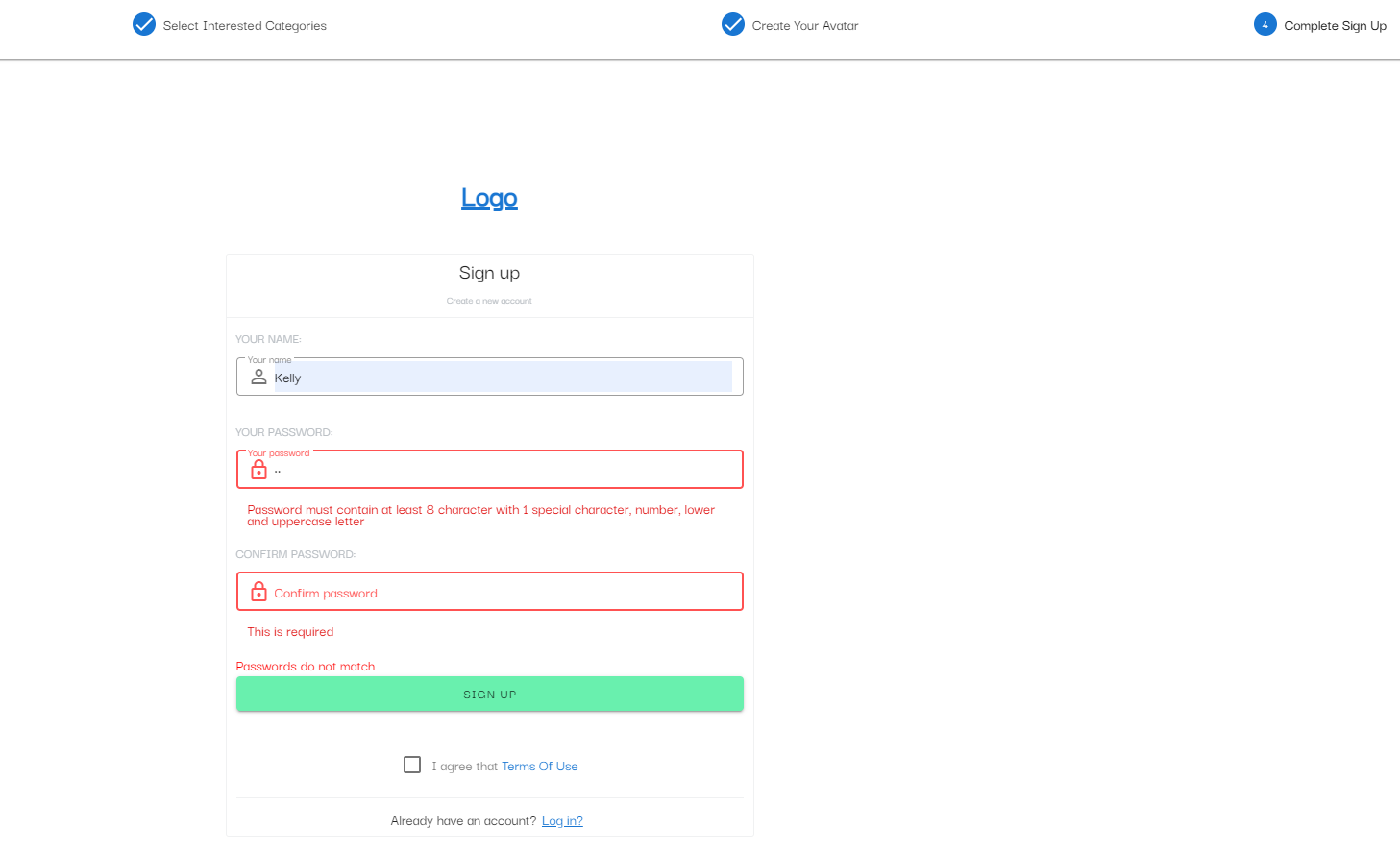


Figure Creation of password

Login page is a form which will gather the user login credentials. When a user log in, data will be sent to the backend to verify with the database. Only a valid user will be able to log in to the application.



Figure Login page

Users can also reset their password through the application if they have forgotten their password. Going through the process, user will be required to key in a confirmation code that will be sent to their registered email. Once the confirmation code is confirmed, user can then proceed to change their password.

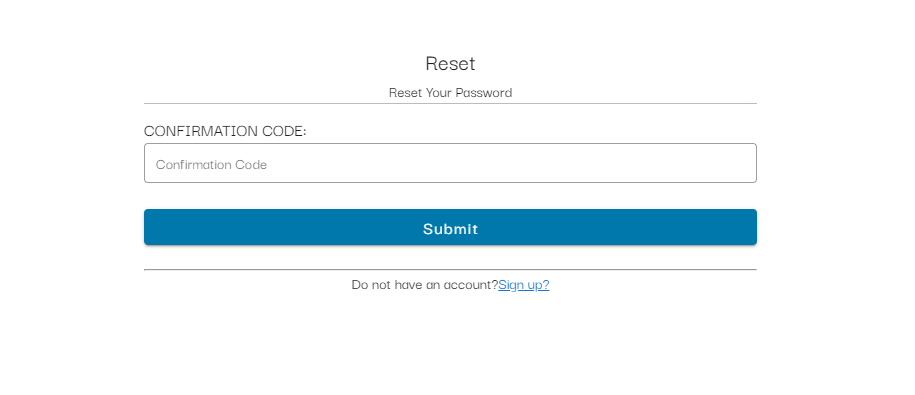


Figure Confirmation code page to change password

### 3.2.6 Statistic View

The statistic view is a page where users can view their statistics. Certain stats that are pre-defined will be calculated and added to the user’s stats database when the users complete different actions on the application. For example, instructors can see how many students they have, how many users sign up for their course or how many reviews they have.

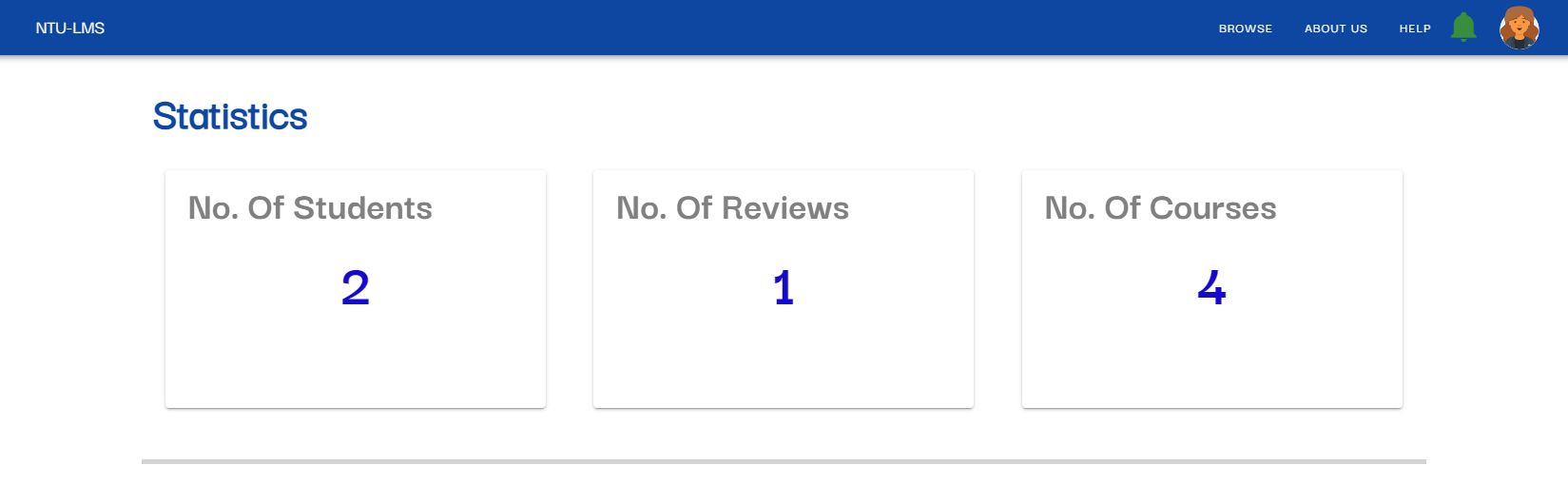


Figure Statistic View

A Line graph is also used to display data over a period of time. I displayed a number of registered courses over a period of one year. This can allow learners to see how many courses they have registered on a monthly basis over the year or allow instructors to understand how many learners registered for their courses. Users can also filter by year or courses depending on what the graph is showing.

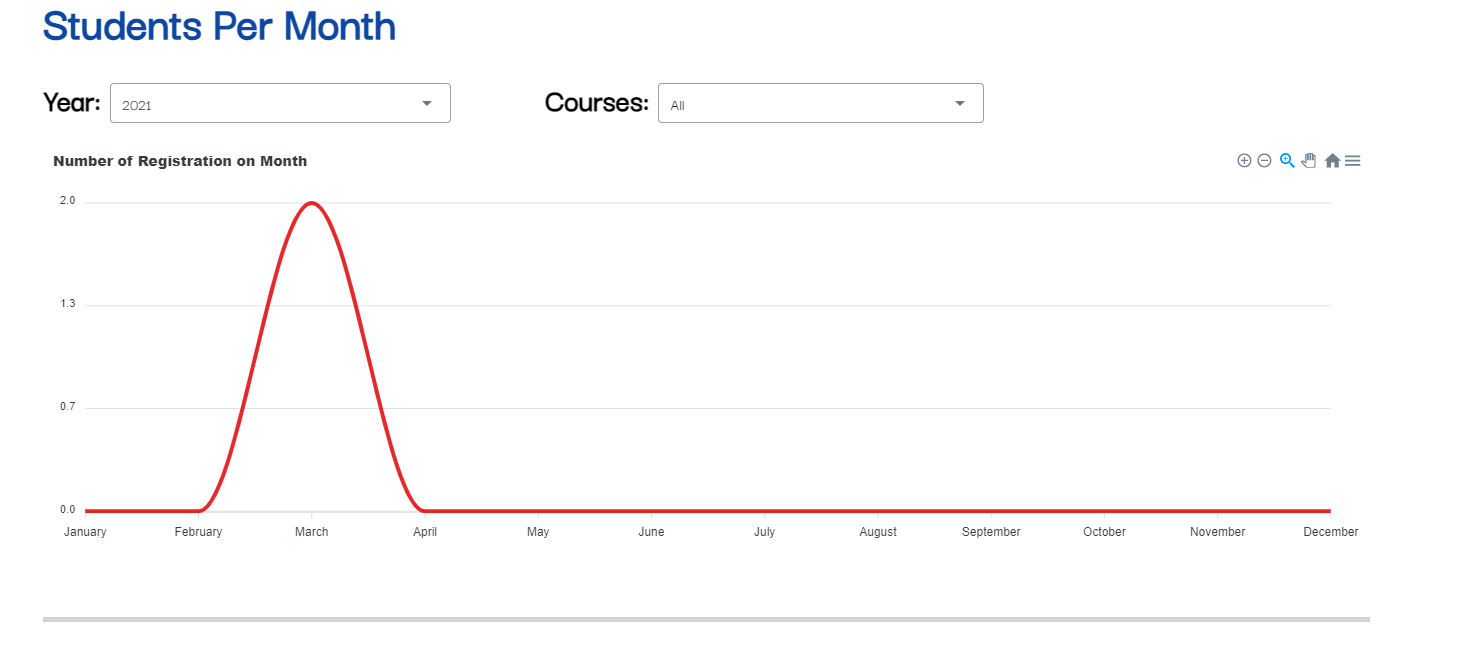


Figure Line graph of number of learners registered.



Figure Line graph for courses registered

### 3.2.7 Achievements View

One of my idea to encourage continuous learning is to include gamification elements. Achievements and levels are the foundation of a game. Hence, I also bring such elements to my project. There are a few achievements that users can aim to obtained as they use the application. The statistics database will be used to keep track of the user’s achievements. Once the user obtained the achievements, it can be seen in the achievement view. The stats and level of the achievement will be displayed along with it. Achievements that are not obtained will be greyed out.

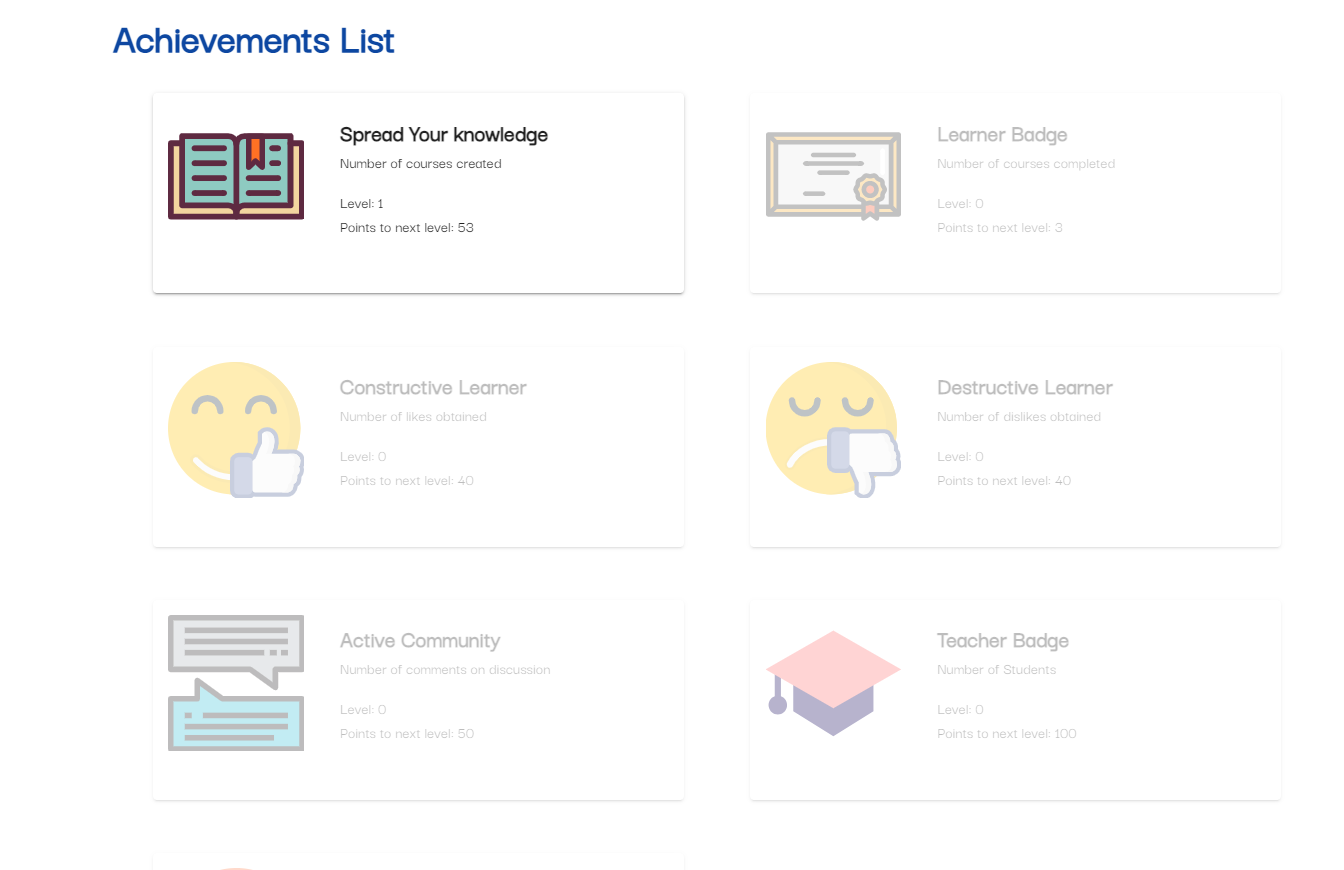


Figure Achievement page

### 3.2.7 Profile View

Users can also view their own profile which will display an overview of their statistics, courses registered and achievements. They will only be able to see the first 4 of their achievement or courses so as not to clutter the page. Users can go to their relevant pages to view more. Users can also see other user’s profile page by clicking on their icons.

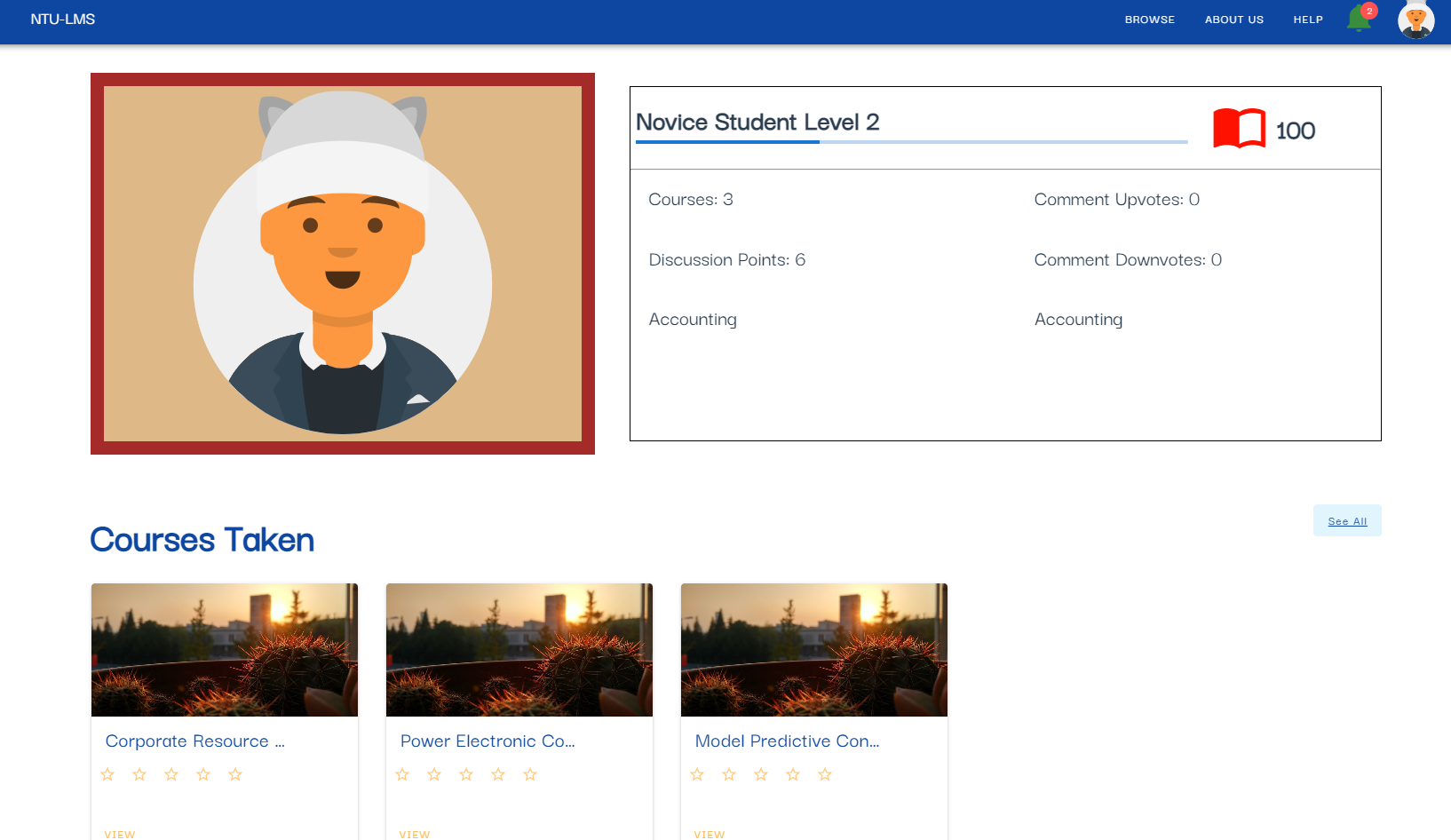


Figure Sample image of Profile view

## 3.3 Database & Back-end Implementations

### 3.3.1 Back-end Design

The backend of the application is used to handle a request that is made by the users from the front-end. It communicates with the database and serves the necessary data when required. For this project, the backend structure is implemented using Express which is a Nodejs server framework.

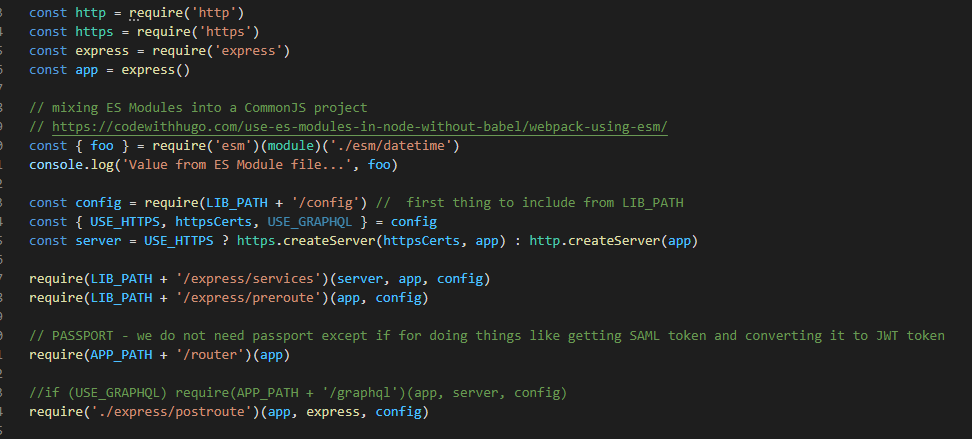


Figure Code block that initialize the backend environment

From figure 23, it will initialize the backend environment, setup connection to the database and other necessary services. Once the connection is successfully establish, I output a log statement to the console so that I know that the connection is successful. This can be seen from figure 24.

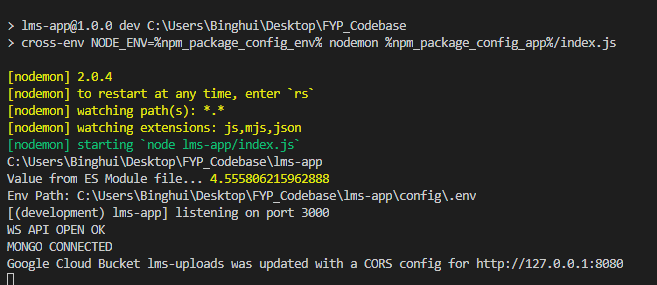


Figure Console log of successful initialization

Express is able to handle different HTTP requests at different URL route paths and pass them through the necessary middleware such as CORS validation and content authorization within the request pipeline [20].

HTTP endpoint allows the front end to send data to the backend through a specific route path. From there, specific actions will be executed. For example, if I want to get a specific course information, I create a route path called “/api/me/course”. The frontend will call this endpoint and pass the necessary data to it and the backend will execute the action of retrieving from the database and return the information to the frontend.

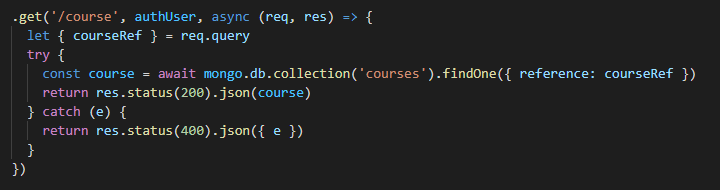


Figure Backend API endpoint



Figure Code block to call backend endpoint

### 3.3.2 Database Design and Considerations

Database used is MongoDB. Mongo is a non-relational database and uses documents to store the data. Each document is stored in a collection which is the equivalent to a table in SQL. SQL databases have a fixed schema for each table and the data within have to follow that schema that was defined. However, in NoSQL databases, the schema is not fixed, and a collection can store documents of different schema.

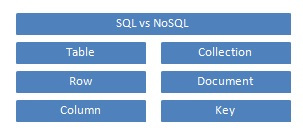


Figure SQL vs NoSQL terminology

When designing the database for a non-relational database like MongoDB, we decide on the collection required by looking at what kind of data we need to store and what we need to query. Some key considerations when designing mongo database collection [21]:

* *Design your schema according to user requirements.*
* *Combine objects into one document if you will use them together. Otherwise separate them (but make sure there should not be need of joins).*
* *Duplicate the data (but limited) because disk space is cheap as compared to compute time.*
* *Do joins while write, not on read.*
* *Optimize your schema for most frequent use cases.*
* *Do complex aggregation in the schema.*

### 3.3.3 User Collection

This is a user database. It stores information related to the user. When a user registers for an account on the application, the information collected will be stored in this collection. Additional information that is stored here will also be created within the document.

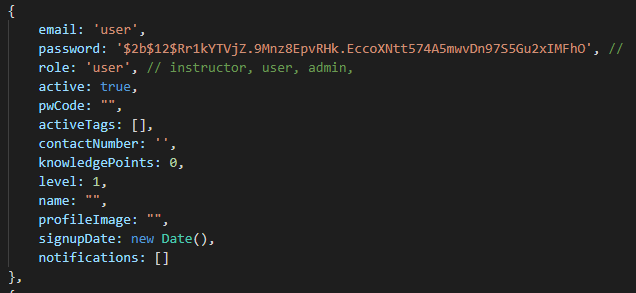


Figure Data Schema for user collection

From figure 28, we can see that the data stored in the user collection are related to the user. Notifications are stored together with the user as the notifications are only for a specific user hence, we store them together.

Once an account has been registered, the user can login to our platform. The backend system will cross check the login credentials with the user collection to find a match. If a match is found, access will be given and rejected if not.



Figure Code to determine login

### 3.3.4 Statistic and Achievement Collection

When an account is created, a statistic and achievement will also be automatically created and stored in the database. Here, there are two different ways to design the statistic and achievement database. We could use either embedding or referencing. As both statistic and achievement documents are related to one user, we could actually embed both their data within the user document.

In my case, I have decided to use the referencing method and create a collection for each of them. This allows me to organize the data separately and have efficient queries.



Figure Statistic collection schema

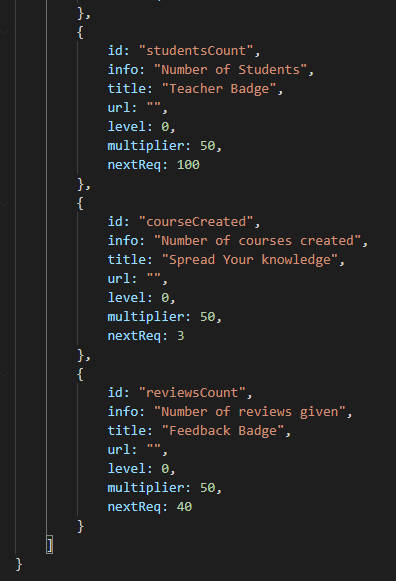
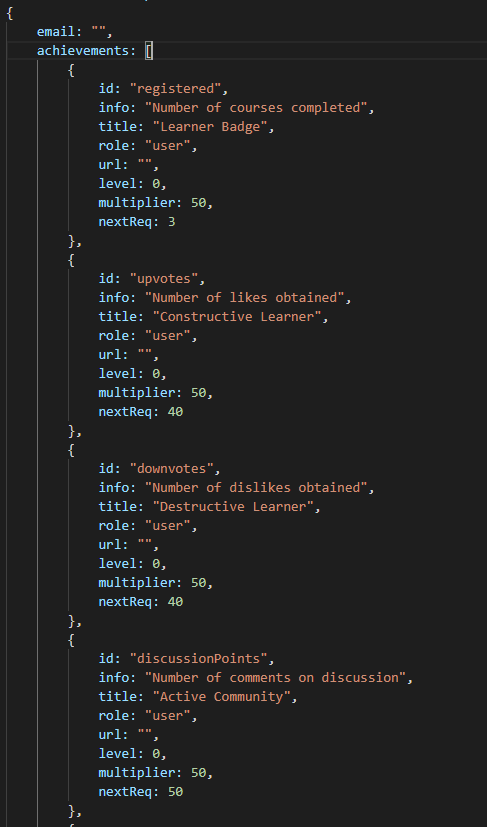


Figure Achievement collection data schema

Data stored in statistic achievement collection is used to display information in their respective views. Statistic view and achievement view which can be seen above.

### 3.3.4 Cron Job

Cron job is a time-based job scheduler. It enables a block of codes or functions to run periodically at a fixed time or interval. In this project, I use cron job to run a function at 12am every day. The function will run through all the active registration and do a check on whether the course has ended or not. Once it is found that the course has ended, it will update the user data with two notifications.



Figure Code that runs the cron job

# **Chapter 4 Design and Implementation for Leaners**

## 4.1 Learner’s Requirements

Learners are one of our two main users on the application. They will be using our platform to enhance their skills and participate in continuous learning. Our application should consider the functionality that a learner will be able to do on our platform to achieve continuous learning. Below is a table that lists the different functional requirements that a learner should be able to do.

|  |  |
| --- | --- |
| No. | Description |
| FR8 | They should be able to browse for available courses |
| FR9 | They should be able to register for courses |
| FR10 | They should be able to view discussion board of registered courses |
| FR11 | They should be able to post a thread or message in the discussion board |
| FR12 | They should be able to view the class of their registered course |
| FR13 | They should be able to post a thread or message in class discussion |
| FR14 | They should be able to access quiz assignments |
| FR15 | They should be able to view a list of courses registered |

## 4.2 Front-end Implementations

### 4.2.1 Browse and Register for Courses

As a lifelong learning platform to support physical classes, it should allow learners to browse and register for courses online. Through our platform, learners will be able to look at the available courses and their detailed information. They can also register for the courses they wish to take.

On the browse course view, it will list out the available courses in rows of three. Learners can filter the display according to the category or sort them in ascending or descending order by the date the course was added to the platform.

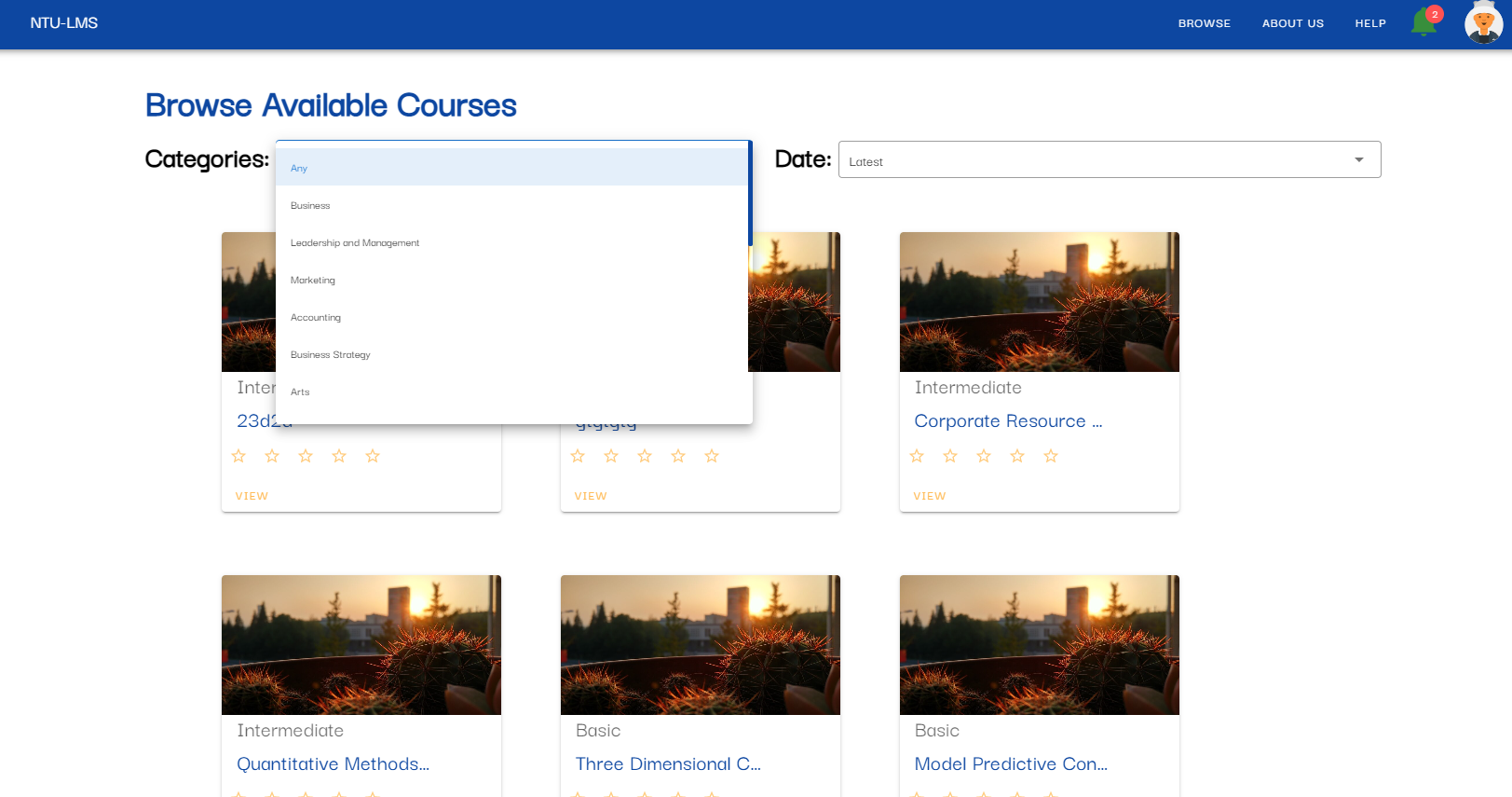


Figure Browse course view + search function

As you can see in figure 31, each course card is similar in design. This is done by creating a reusable component that will output the HTML base on the inputs. This helps to minimize duplicated code, error and speed up development. Below is an image of the code for the course card. As seen in figure 32, the course details on the card are dynamic as it depends on the input data.

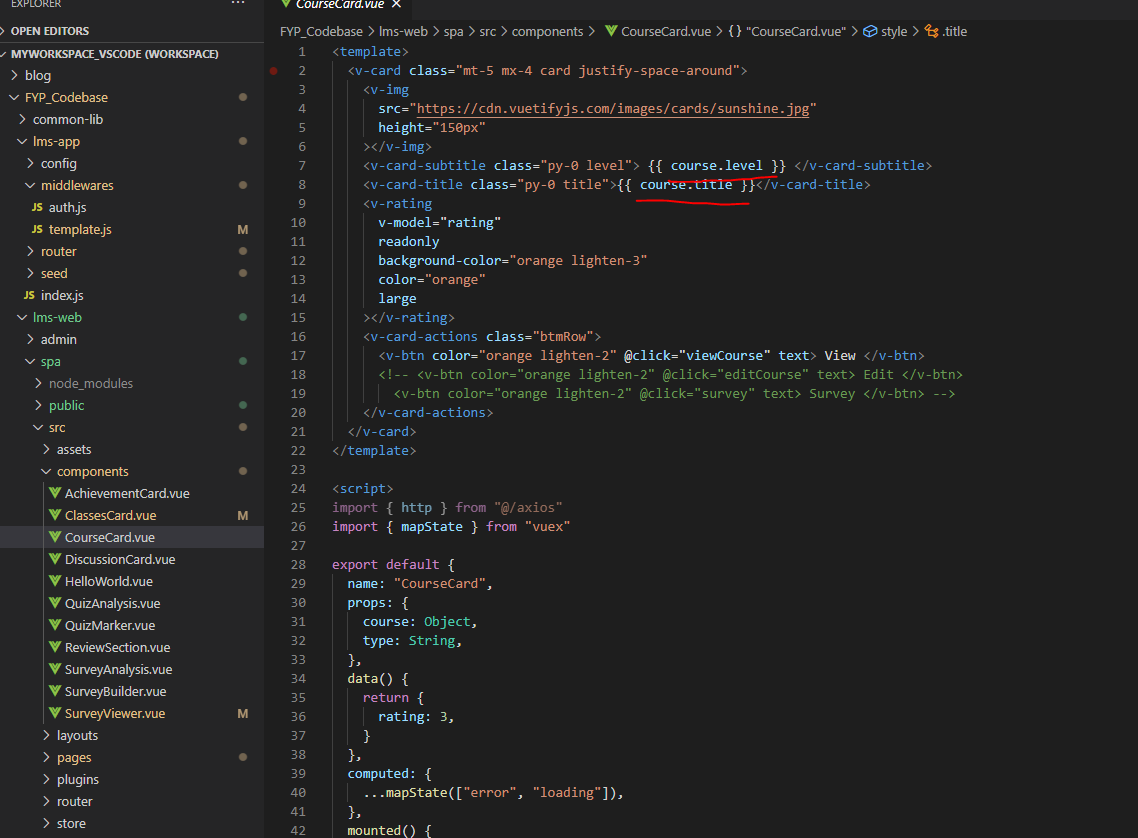


Figure Code block of course card component

Learners can view the course details by clicking on the “view” button. This action will route them to the course details page where they can see the course. As seen from figure 33, there is a register button on the right side. Learners can register for the course by clicking on it. A confirmation screen will pop up which the learners have to agree to officially register for the course. Backend logic is in place to prevent the user from registering the course multiple times. This will be further discussed in section 4.2.

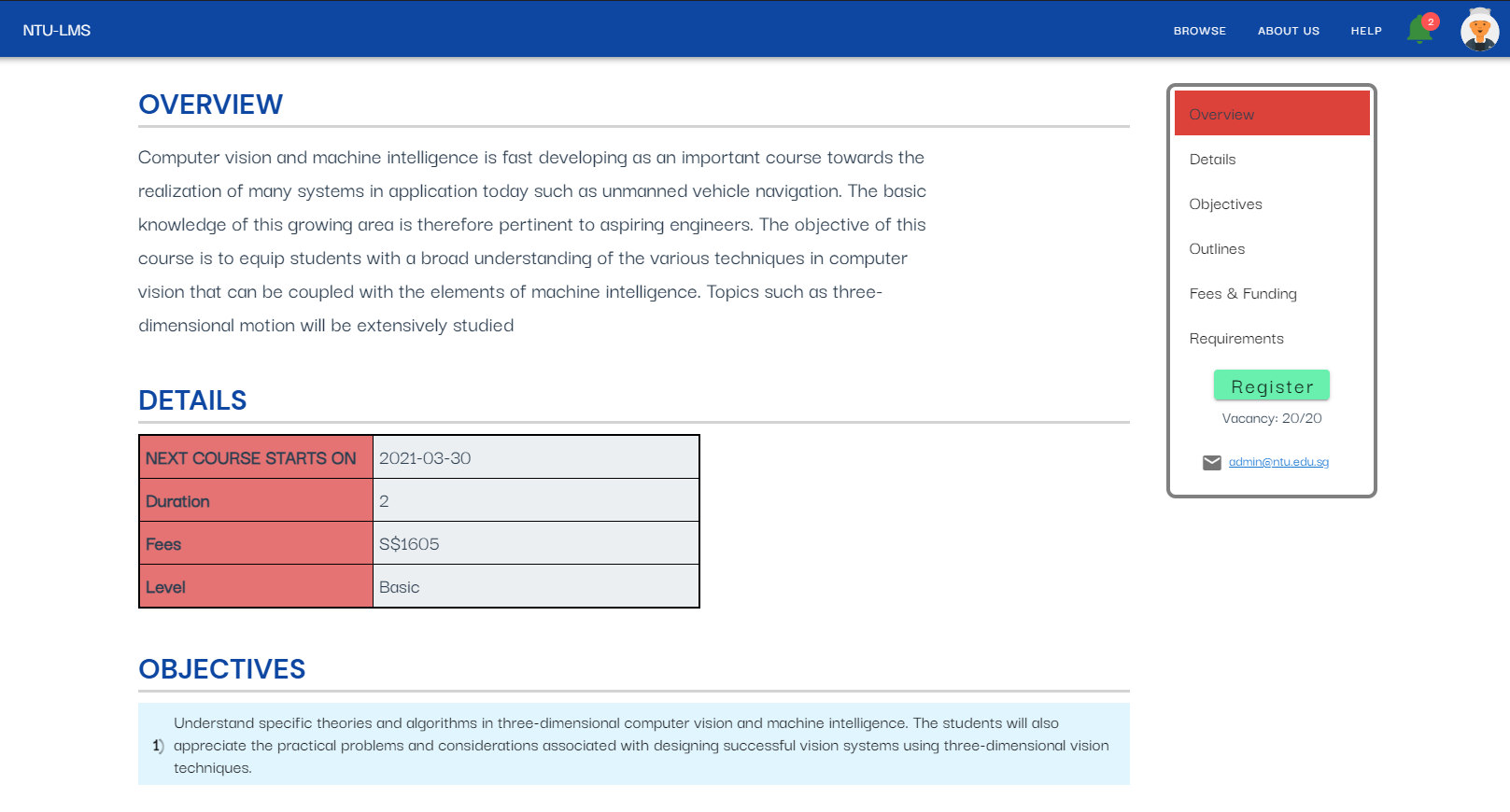


Figure Course details view

### 4.2.2 Discussion Boards

Discussion Board is one key area that will encourage learners to keep learning and stay active on the platform. Discussion is very common in any platform. Here, when learners interact on the discussion board, they will gain points that will enable them to unlock achievements. These points can also help to level up their virtual character on the platform.

There will be a discussion board for each course available on the platform. Learners can gain access by registering for the course. There are three parts to the discussion board.

First, learners can select the discussion board of a course that they have access to.

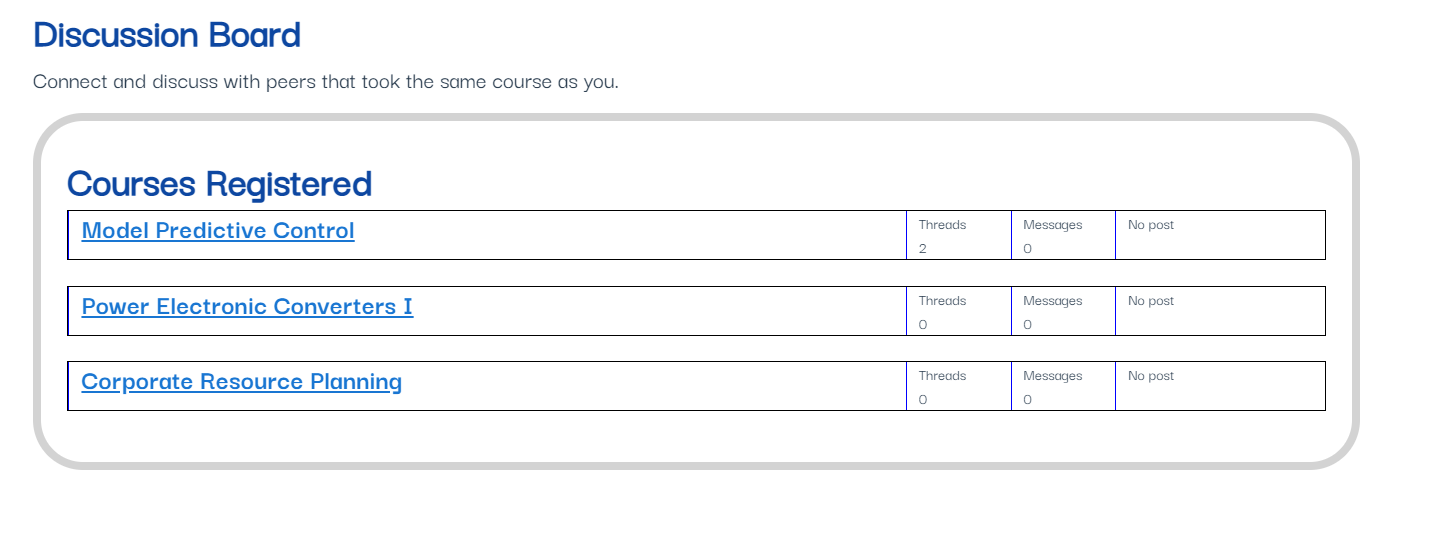


Figure Discussion boards available to the user

Second, learners can then view the different threads related to the course discussion board. Users can post create a new thread under the discussion section by clicking on the button as seen in figure 35 with a red underline. A pop up will appear for learners to input their question as seen in figure 36.

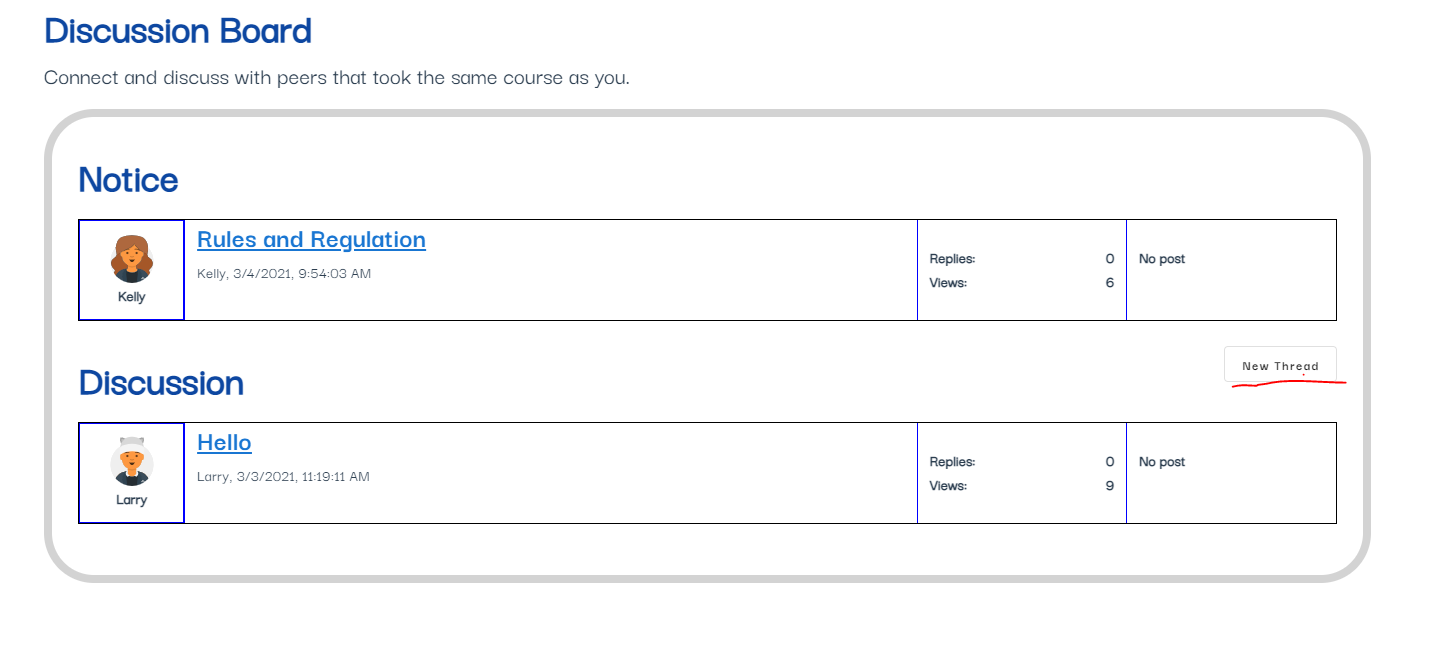


Figure Discussion View for a course



Figure Creating a new thread to ask question

Lastly, learners can click on a thread to view the related messages. They can also reply to the thread by clicking on the button as seen in figure 37 with a red underline. A pop up will appear similar to figure 36.

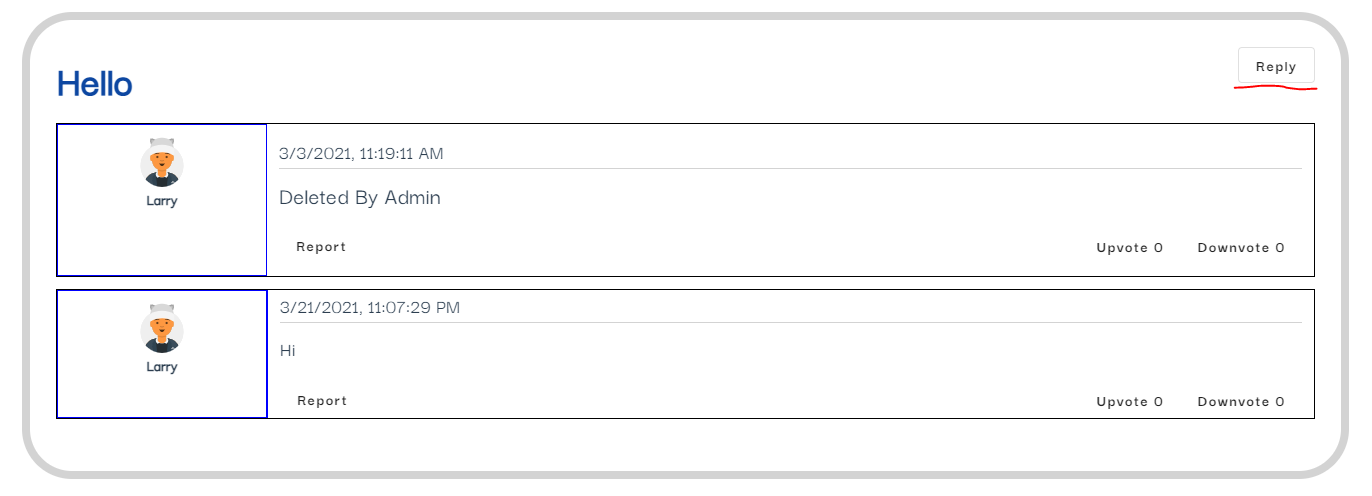


Figure Thread messages view

### 4.2.3 Course Classes

Learners will have access to the class board when they register for the course. The class section has a similar function to the discussion board. The class board is only available over the period of the course duration.

As the platform aims to support physical classes, I hope to increase the communications between the instructors and the learners by allowing them to continue discussion even after class hours. Learners will be able to ask questions or give feedback to the instructors throughout the duration of the course. This can be more efficient and helpful than only asking during the physical class.

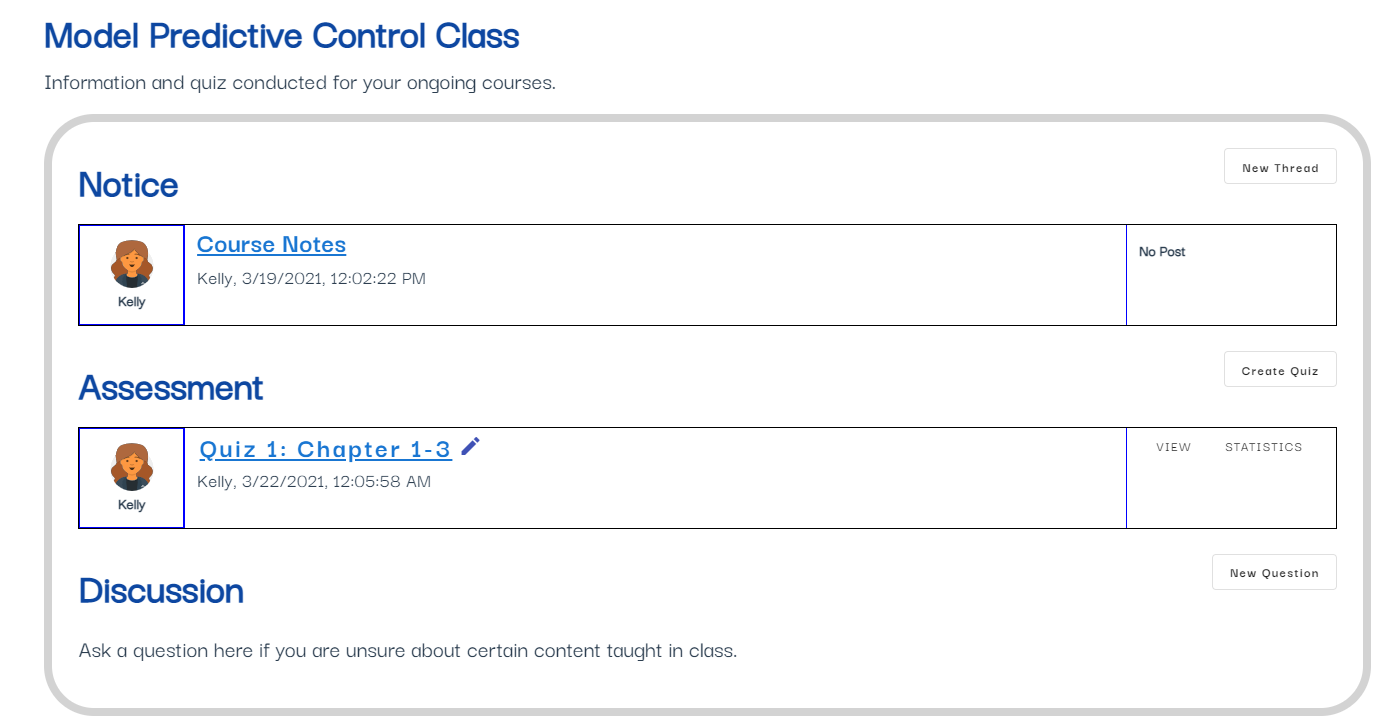


Figure Course Class view

The class board also support the conduct of simple quiz assignments. Mainly in the form of multiple-choice questions. This will be further elaborated in chapter 5. Learners can access these quiz through the class board.

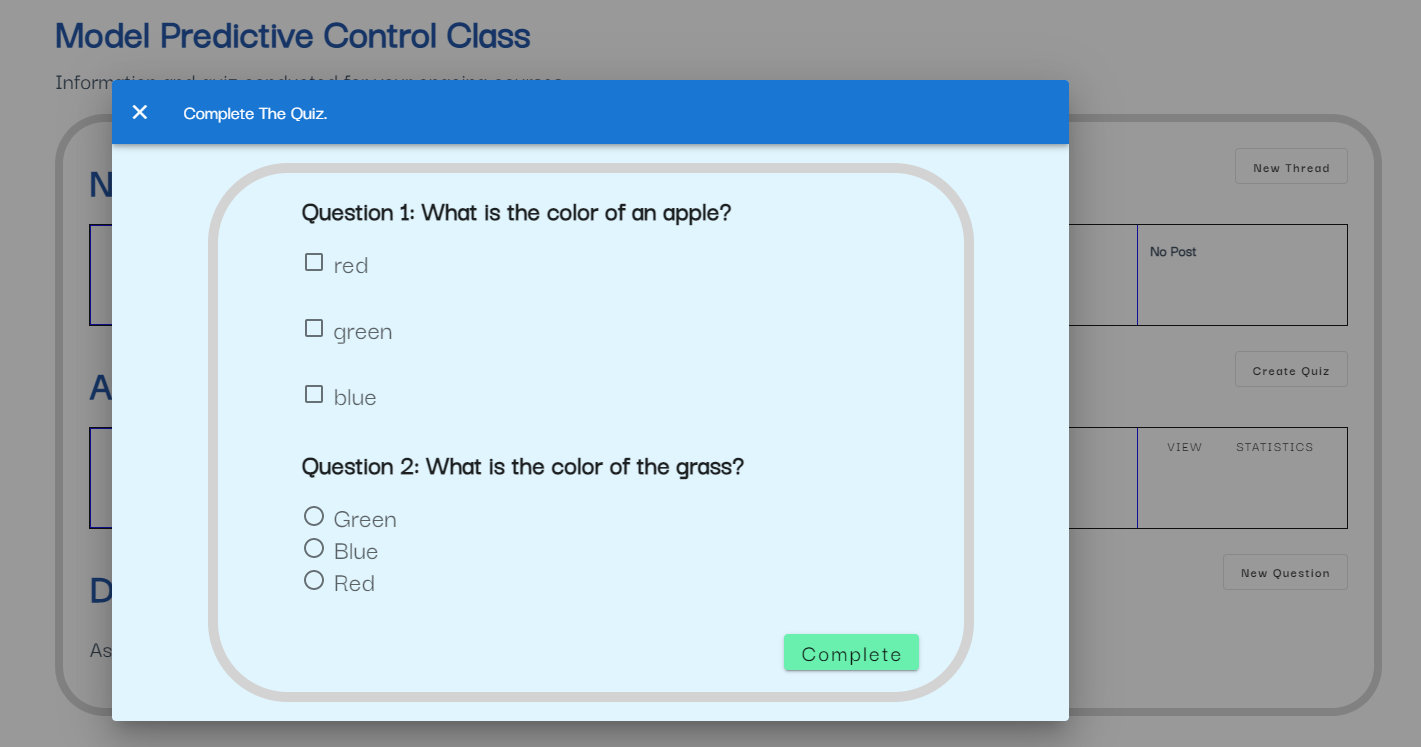


Figure Learners can access and complete quiz

## 4.3 Database & Back-end Implementations

### 4.3.1 Registration Collection

Information has to be stored when a learner registers for a course. The system needs to know who registered for a course, what course they registered and the period of the course. This information can be obtained from the user document and course document.

Each registration points to a user and a course. It is not ideal to embed these information in any existing collections. Therefore, I create a new registration collection to store this information. I used a key value pair to reference the course document that the learner registered for. This can be seen in figure 40 with a blue underline.

Each registration has a start and end date. The application checks for the time when the course period is over to send out a notification to the learners. In order to prevent the notification to be sent multiple times, we have a key value pair called “endAlert”. This will be set to true once the notification is sent the first time. This can be seen in figure 40 with a red underline.

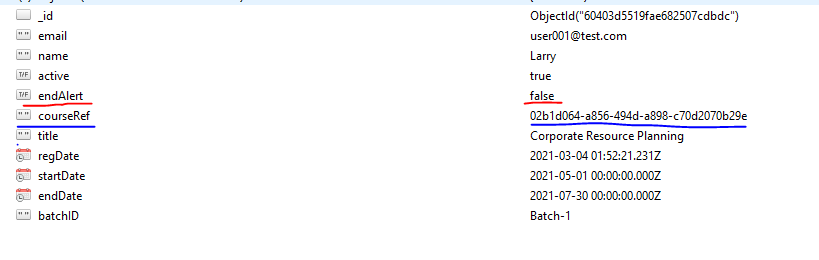


Figure Data schema for registration collection

### 4.3.2 Discussion Board collections

Learners can create a thread or reply to existing threads. There are two different informations that needs to be stored in the database. The thread information and the replies information. I created a thread collection to store data regarding the thread information. As the discussion board is based on individual courses, I have included a course reference in the data schema.

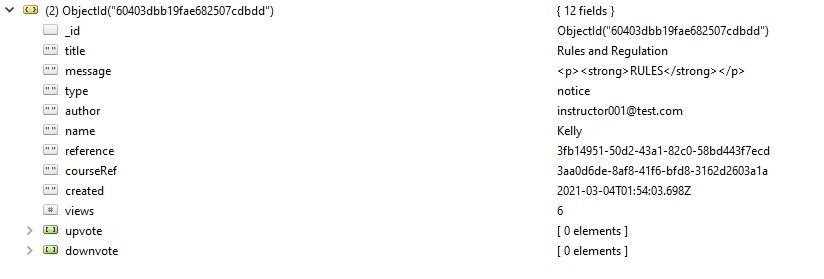


Figure Data schema for threads collection

Replies information can be embedded inside the thread data schema but it is not ideal. As the discussion board can be access permanently once registered, there is a possibility that the number of replies to a thread can become very large. MongoDB documents have a max size of 16 megabytes hence if the replies information was to be embedded inside, it might exceed the max document size in the long run. Therefore, I created a messages collection to store the replies information.

Since, the replies information is now stored in a different collection, I need to create a thread reference in the message data schema so as to query the data correctly.

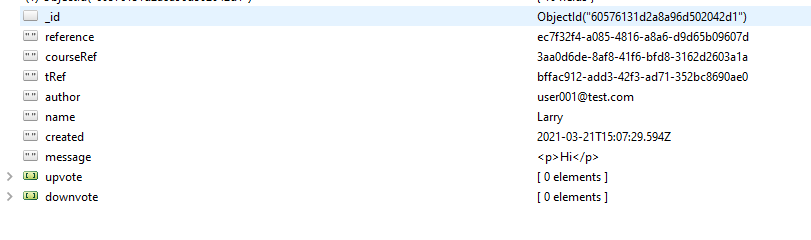


Figure Data schema for messages collection

### 4.3.3 Class Board Collection

Information regarding the class will be stored in a “classes” collection. Each class will have a function similar to a discussion board. Learners can create threads or reply to existing threads. As the class board is only available over the duration of the course, there is less likely that the document size will exceed the maximum allowed size if the thread and replies information is embedded inside the class document.

Hence, I created a class collection to store all the necessary information including the threads and messages.



Figure Data schema for class collection

From figure 43, we can see the notice, quiz and feedback array which is used to store the threads that will be created in the class board.

### 4.3.4 Quiz Results Collection

Learners are able to access and complete the quiz assignment through the class board. Their inputs will be saved to the database. I created a “quizResults” collection to store their answer. As the quiz results is referenced to a specific quiz and course, I will need to save the key information in the data schema.

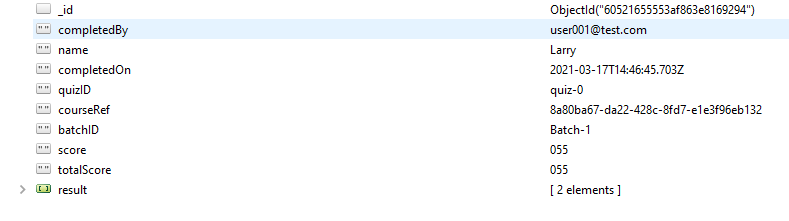


Figure Data schema for quizResults collection

# **Chapter 5 Design and Implementation for Instructors**

## 5.1 Instructor’s Requirements

Instructors are also learners, but they have access to additional functions on the application. They have the functional requirements explained in chapter 4 and additional requirements that will be explained below.

|  |  |
| --- | --- |
| No. | Description |
| FR16 | They should be able to create and edit courses |
| FR17 | They should be able to create and edit course survey |
| FR18 | They should be able to view a list of courses they created |
| FR19 | They should be able to view discussion board of their owned courses |
| FR20 | They should be able to view the class of their owned course |
| FR21 | They should be able to create quiz assignment for the class they teach |
| FR22 | They should be able to mark the quiz assignment for the class they teach |
| FR23 | They should be able to view the students list for the classes they teach |

## 5.2 Front-end Implementations

### 5.2.1 Course Creation and Edit

Instructors will be able to create their courses through our platform. The course creation page is created using a form to collect the necessary information required. Validations are used for certain inputs that are deemed required or requires a certain pattern. Instructors can also edit the courses they created. This process is the same as creating a course, only the input fields are filled with the course details.

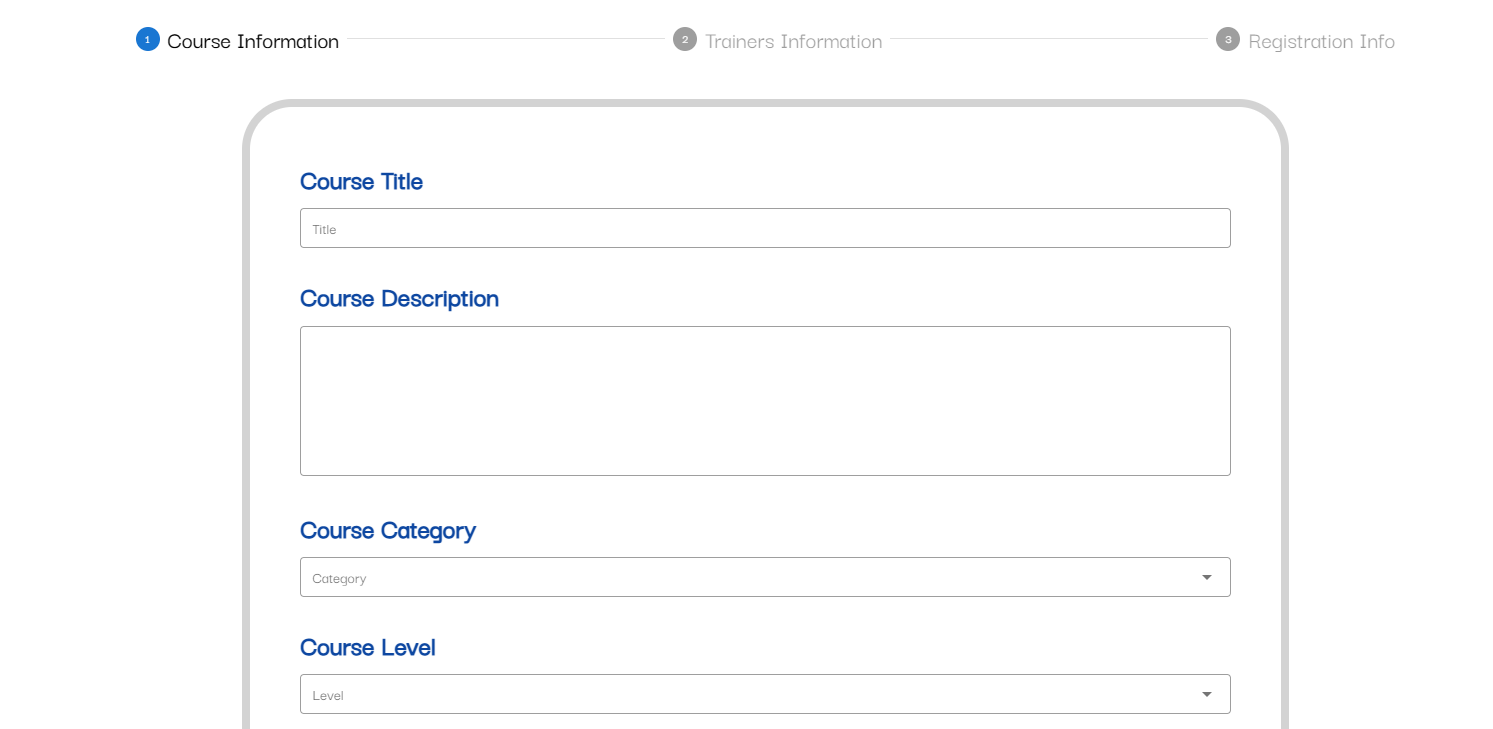


Figure Sample view for course creation

The information collected will be used to create a course details page. Users are able to browse for this course and view the course details and register if they like.

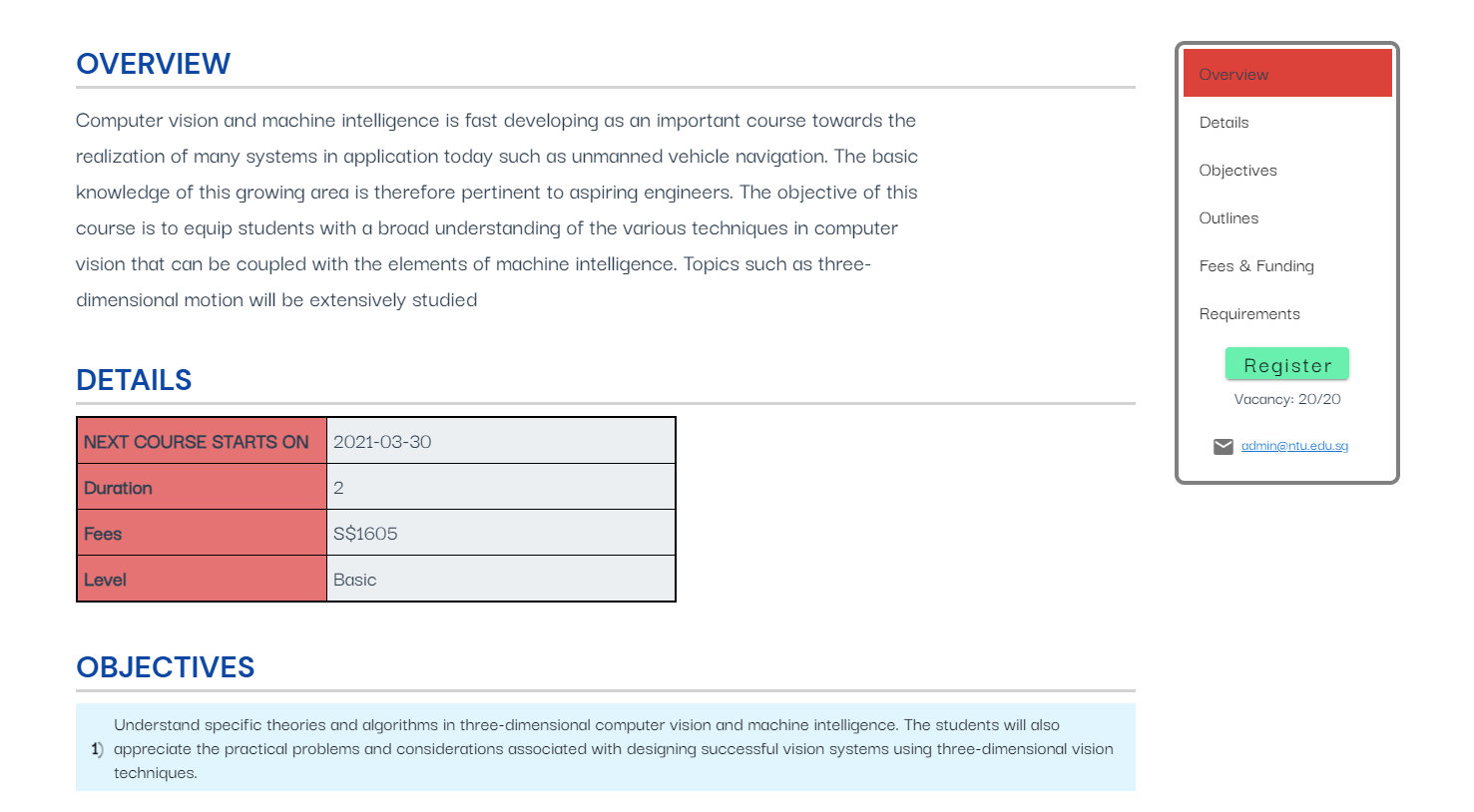


Figure Course details page

Instructors are able to edit the courses they have created. They could change information on the course. The course edit view is similar to the creation except that the fields are filled up with their existing data.

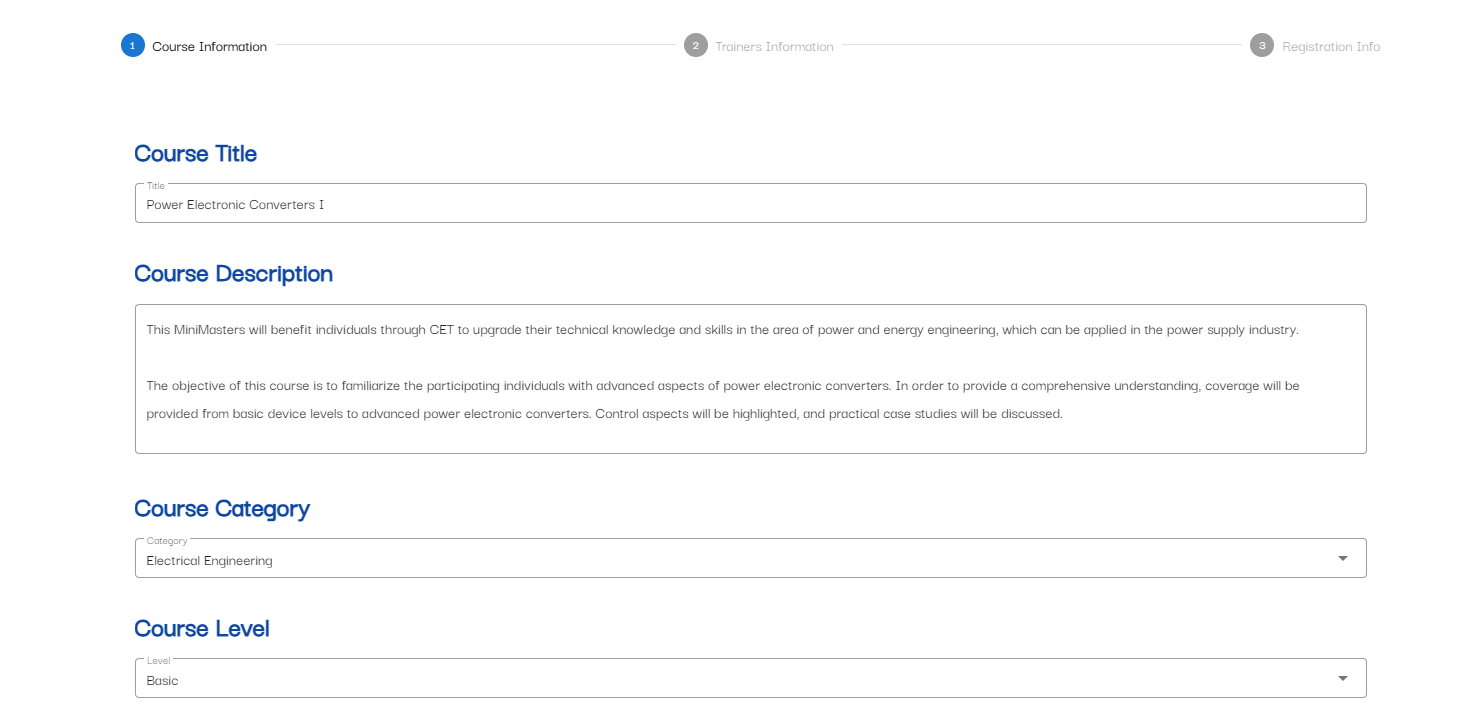


Figure Course Edit view

Instructors can start a new batch for the course by changing the registration period and course start date. They have to select the checkbox to stat that they wish to start a new batch. Additionally, instructors can choose to port the files that they have posted in the class board for the previous batch. This can be seen from figure 48.

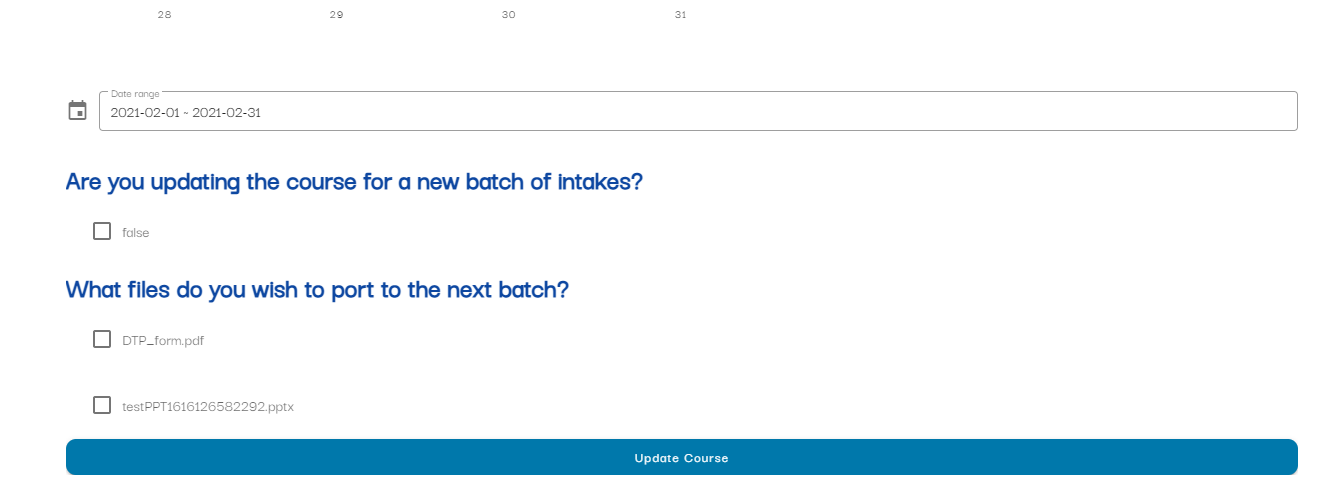


Figure Option to port files and start new batch

### 5.2.2 Survey/Quiz module

Instructors can also create a survey or quiz for their course. Survey and quiz module are similar in their function therefore I created a Vue component so that it can be called at different pages that requires it.

There are four different types of inputs to choose from: text, radio, checkbox and rating. Instructors can choose the type of question from the toolbox. Once clicked, a template box will be created and displayed on the page. Each of these template boxes will collect the information required for the question. For example, title, options or even points for quiz module.

From the images below, each of the blocks of codes highlighted by the red box will generate a template box (blue box) of the different question types.



Figure Code sample for survey builder

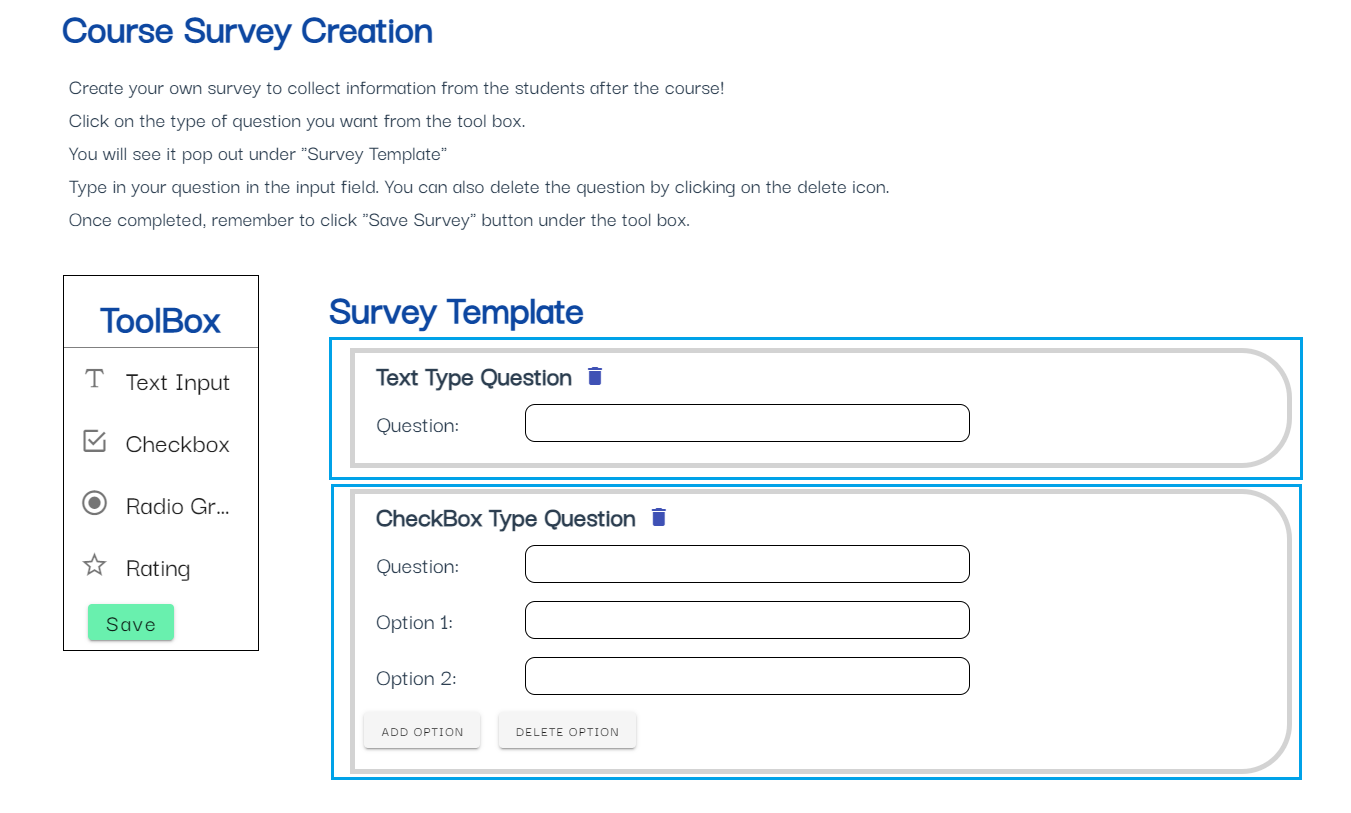


Figure Survey builder page

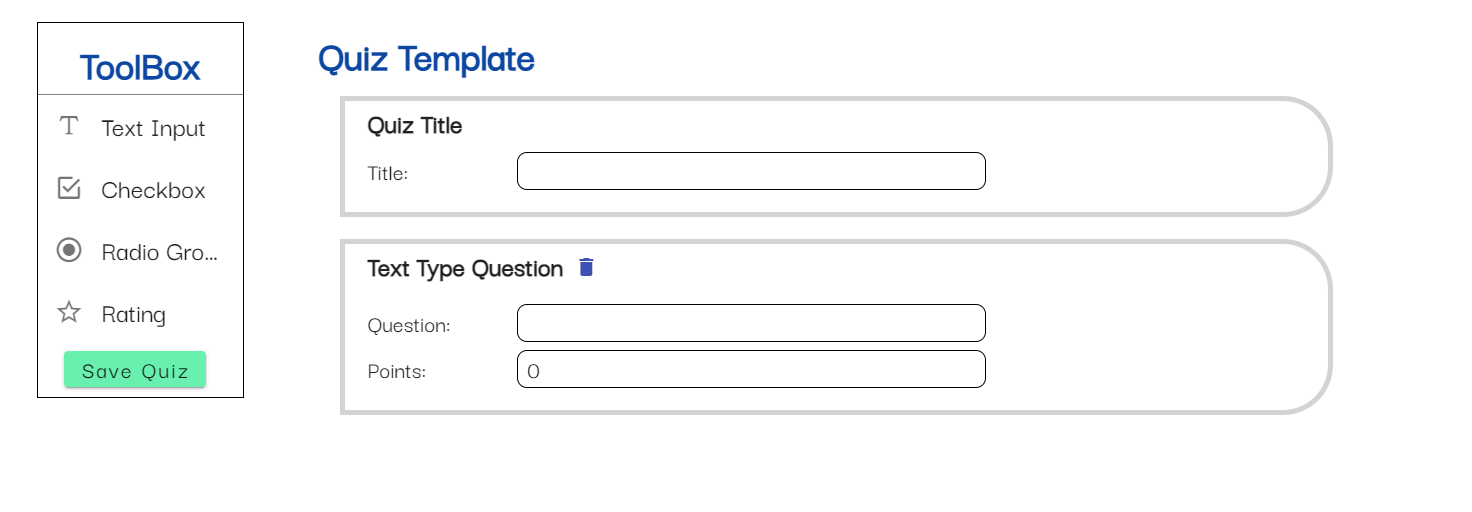


Figure Quiz builder page

From figure 23, you can see that the template box is created by looping through a questions array (green box).

From figure 26, whenever the instructors click on the toolbox to add a question, an object (red box) that store the information is push into a question array (orange box). By doing this, we can also allow the instructors to delete the question by removing it from the array.

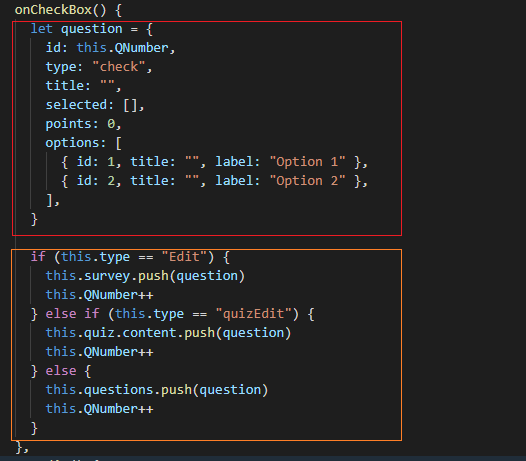


Figure Sample code to add question

Instructors are also able to view the results of the survey or quiz. For both the quiz and survey, we will be displaying the results per question. In addition, quiz results will show an average percentage for the quiz and the number of learners that take it. Each induvial quiz will also display the average percentage and score.

A table will be used to display text type question as there will be many different types of answers therefore, using a bar or pie chart is not suitable. Bar charts will be used for displaying radio and checkbox type questions as there is a fixed number of available answers.

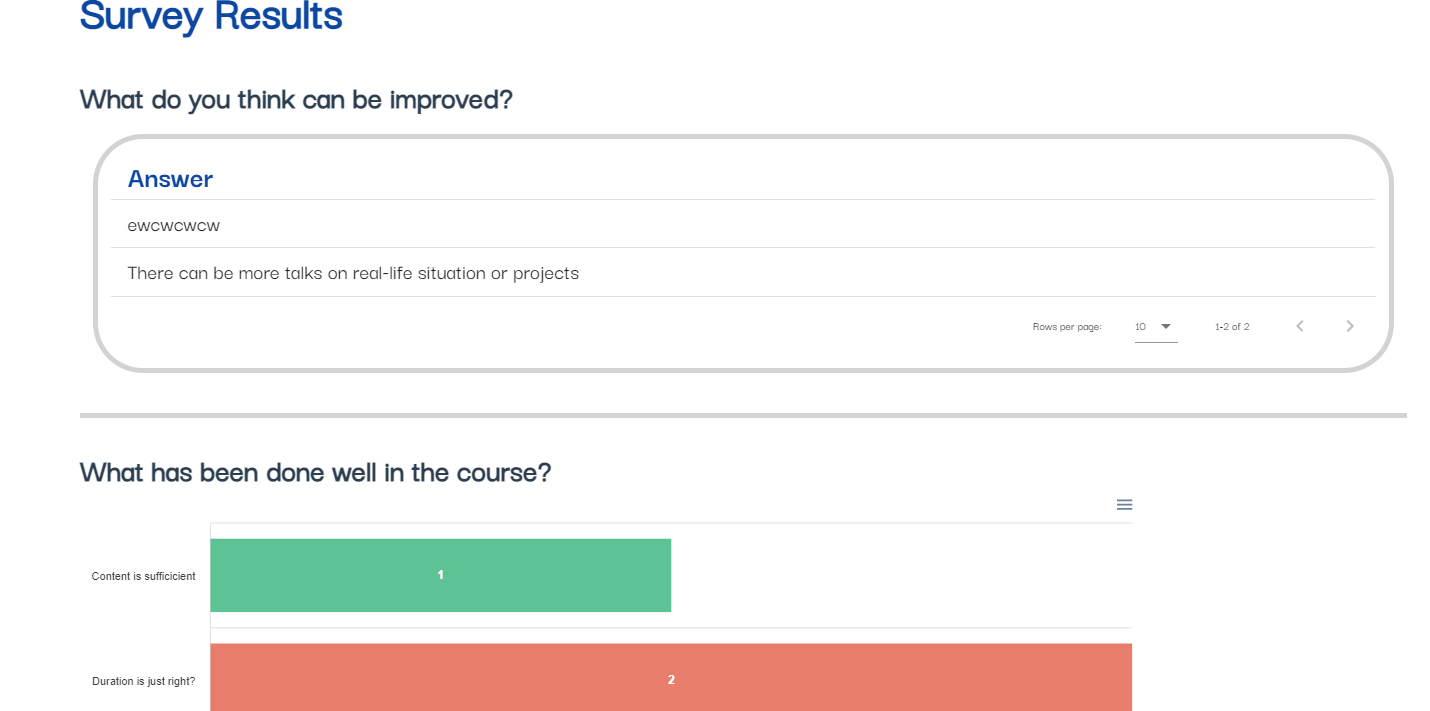


Figure Survey Result statistics

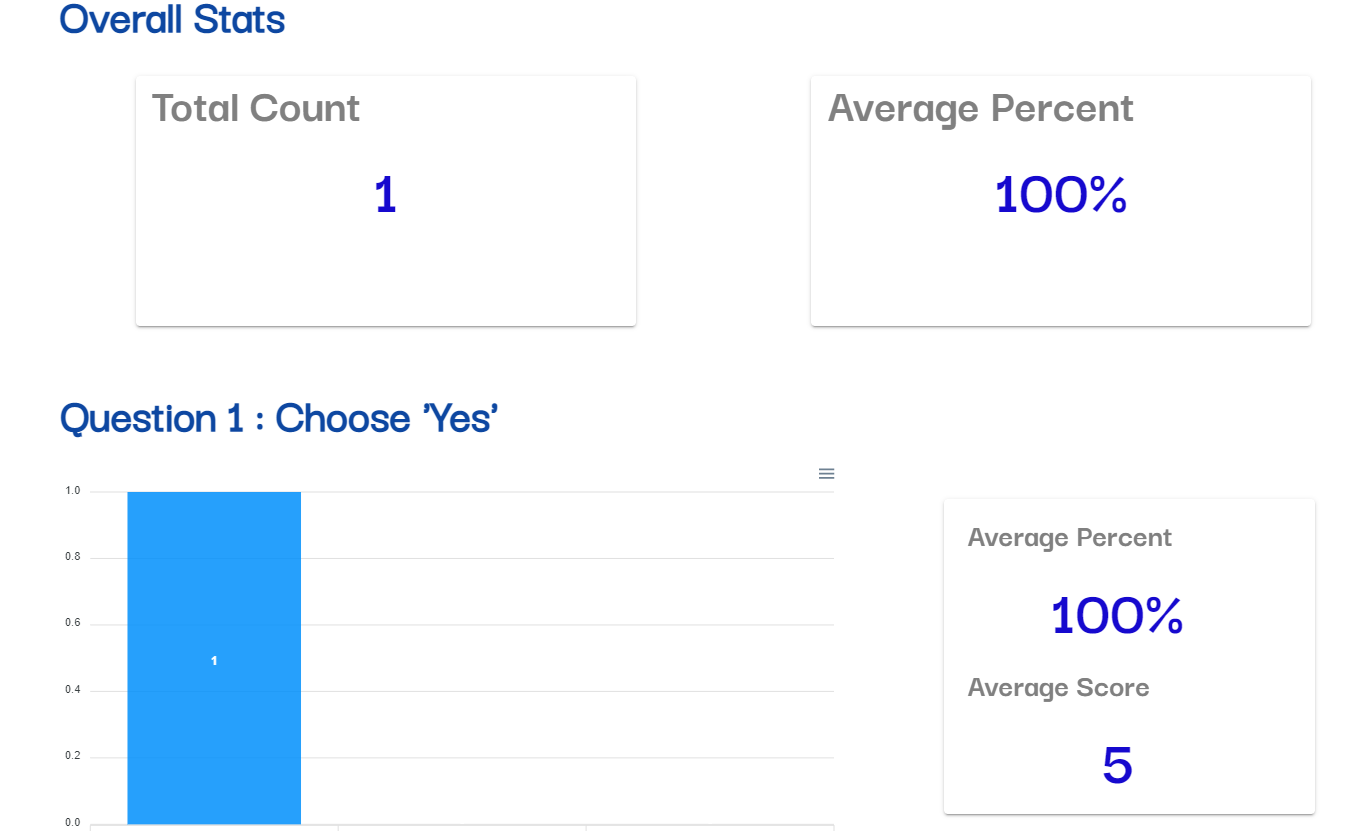


Figure Quiz result statistics

Instructors are able to view and mark the quiz. The results will be calculated and display as seen in figure 28 above. To do this, I created a component to view the quiz with an additional column that allows the instructors to select the marks given to the answer. There are 3 options that the instructor can choose, full mark, half mark and zero marks.

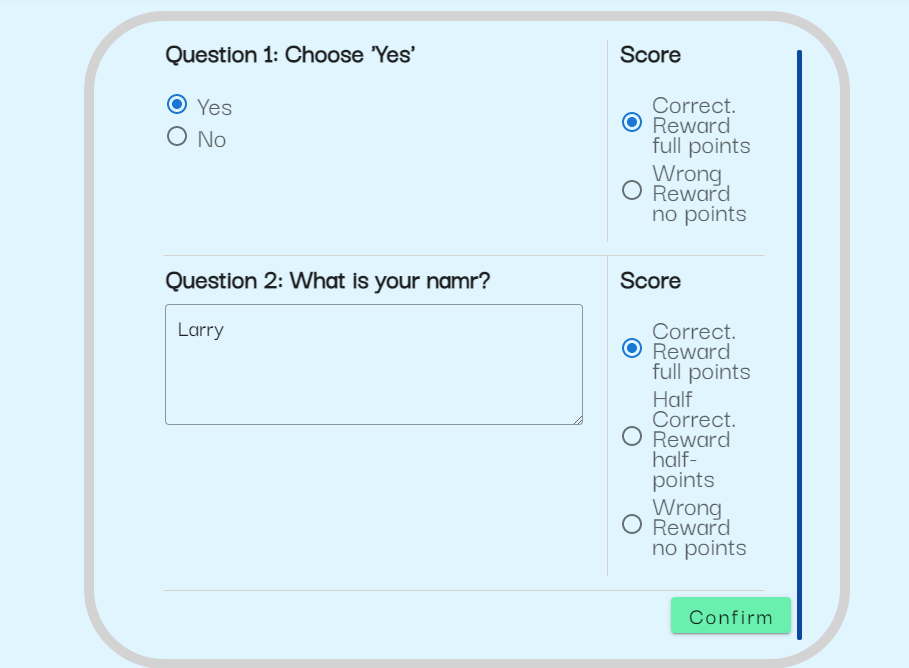


Figure Quiz marker

### 5.2.3 Discussion Board

The discussion board is similar to the learners but because they are instructors, they have an addition section to view the discussion board of their own courses.

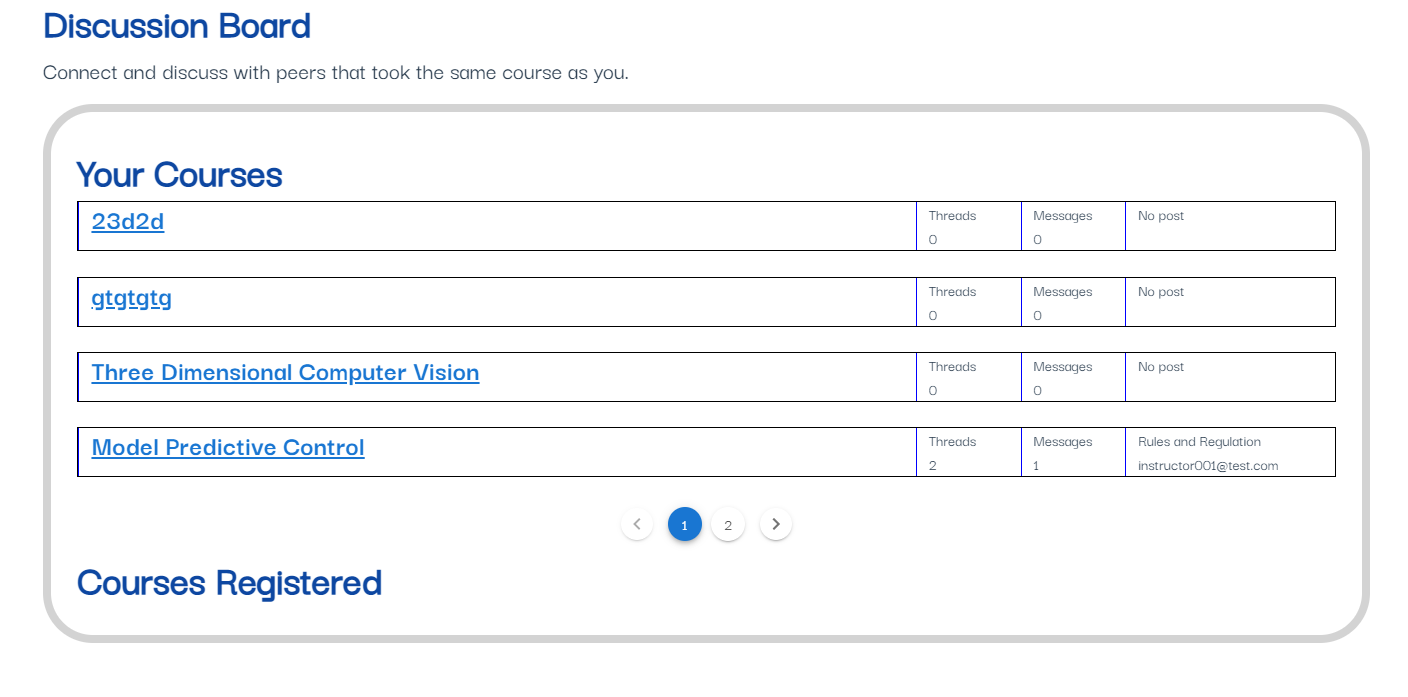


Figure Instructor’s discussion board view

### 5.2.4 Class Board

The class board is also the same as that of the learners. Instructors have an additional section to view the classes of their own courses. In addition, instructors are able to create a thread in the notice section and also create a quiz assignment. This can be done by clicking the button on the right side of each section.



Figure Instructor's class board view

The instructors are also able to view the list of students that registered for their class. They could filter the list by course or by batches. The list of students will be displayed using a table. Instructors are able to remove the students before the course start date.

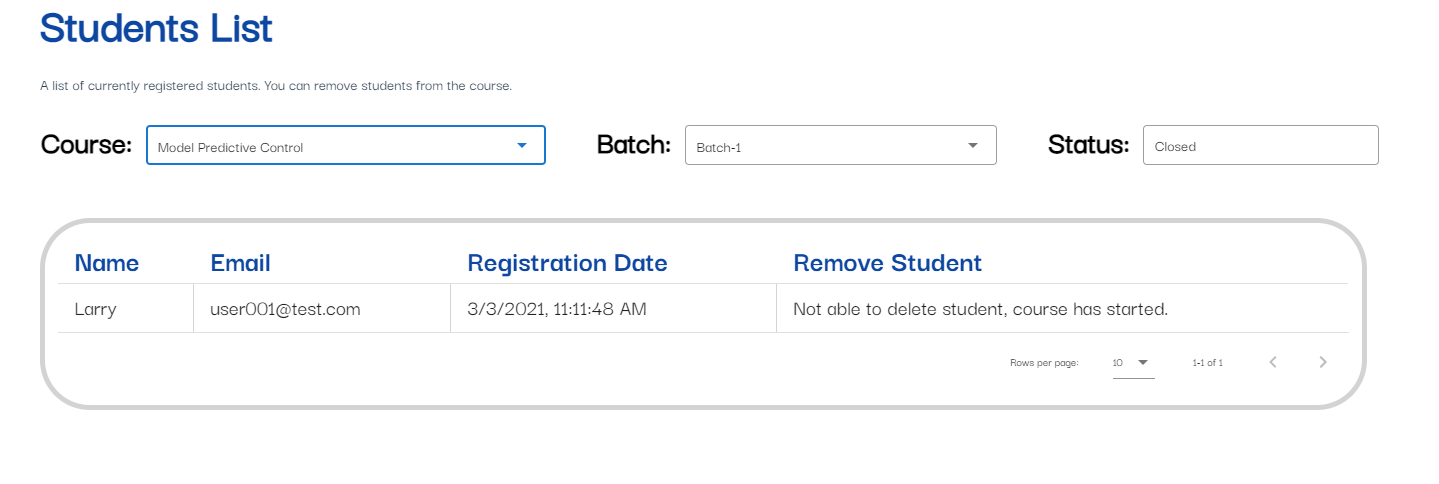


Figure Student list per class

## 5.3 Database & Backend Implementations

### 5.3.1 Course Collection

Instructors are able to create courses. I will need to store this information in the database. Courses are created by instructors and it will not be a good idea to store them together with the user collection. Therefore, I created a “courses” collection to store all the course documents.

The course document will include all the details that the instructor input through our frontend interface. It will also include a reference key to the instructor which will be the email. Once a course is created, it will also automatically generate and store the related documents for discussion board and class board in their respective collections.

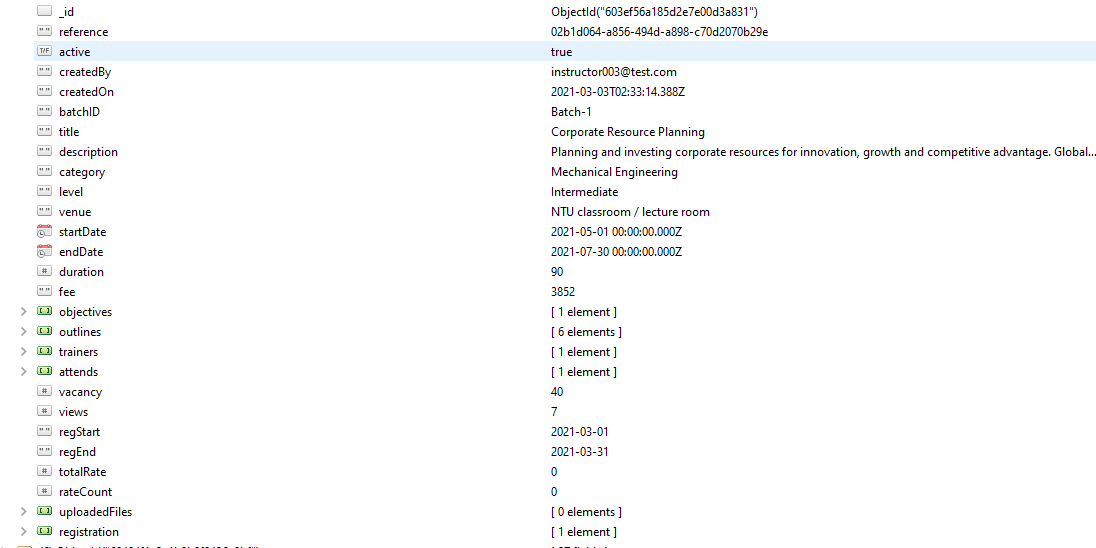


Figure Data schema for courses collection

### 5.3.2 Survey Collection

Instructors will be able to create a survey to collect feedback for the course they teach. They will be prompt to create a survey after they create a course through the platform. There will be only one survey for each course.

Survey information can be embedded within the course document but to have an efficient query, I do not want to retrieve the survey data every time I retrieve the course data. Hence, I created a separate collection to store the survey data.

A survey data will reference a course and instructor. The primary key of the course and instructor will be included in the data schema of the survey document.

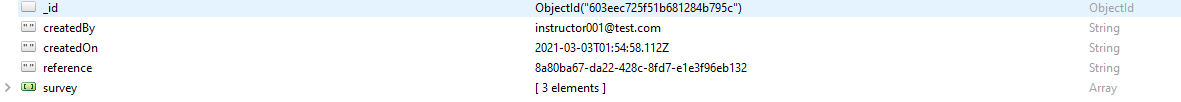


Figure Data schema for survey collection

Instructors can also view the statistic of the survey. In order to display the survey statistics, I have to retrieve the relevant survey results from the surveyResults collection. The data will be further aggregate into a single JSON object with the form that the chart library take.

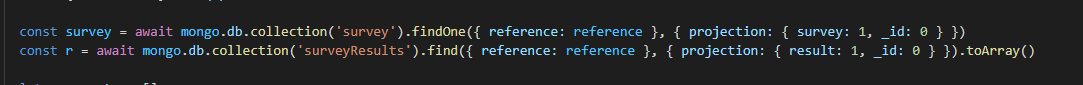


Figure Code to Retrieving survey results

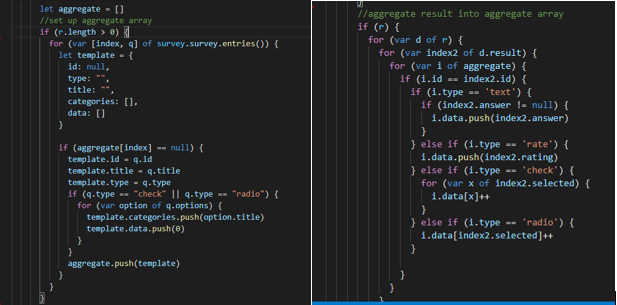


Figure Code to aggregate and format the data

### 5.3.3 Quiz Collection

Instructors can create quiz assignments. The data of the quiz created will be stored within the class document as explained in chapter 4.2.3. Instructors can mark the quiz for each submission. Selecting to view the quiz of a student, the backend will retrieve the data from the quizResults collection and display it to the instructor. The Instructor can then mark the quiz and update the quiz data.

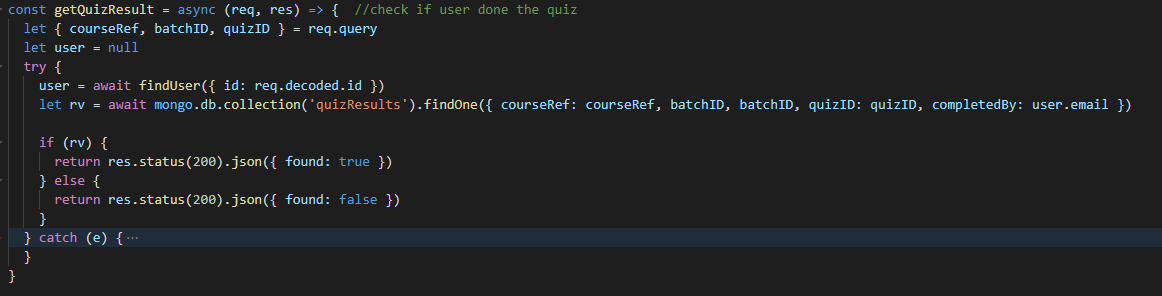


Figure Code to retrieve quiz data to mark.

Instructors can also view the statistic of the quiz assignment. The process is similar to view the survey statistics. Retrieval of data and aggregating the data into a JSON format that the graph library takes.

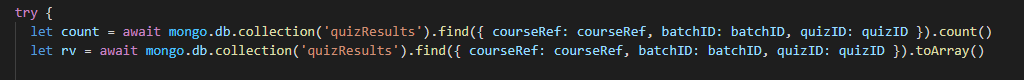


Figure Code to retrieve quiz results

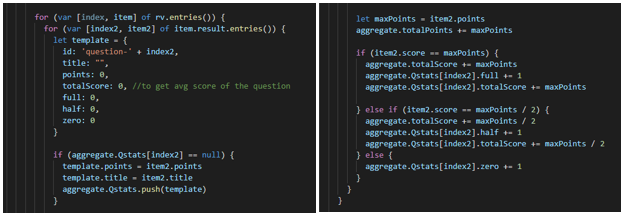


Figure Code to aggregate quiz results

# **Chapter 6 Design and Implementation for Administrators**

## 6.1 Administrator’s’ Requirements

The administration portal is built separately from the main web application. The administer role is to manage the user base and handles report feedback from the users. The administrator will not have access to the main application and its related function. However, the login function is the same as that of the main application that is explained in chapter 3.3.3.

|  |  |
| --- | --- |
| No. | Description |
| FR24 | Admin should be able to log in/out to the admin portal |
| FR25 | Admin should be able to see a list of learners |
| FR26 | Admin should be able to see a list of instructors |
| FR27 | Admin should be able to approve applications |
| FR28 | Admin should be able to edit and create users |
| FR29 | Admin should be able to handle discussion message report |

## 6.2 Front-end Implementations

### 6.2.1 Home Page

The administration portal manages the user base of the application. It should be able to see an overview of the information on the application. Therefore, I created a statistic card to display the number of users, courses and registrations.

The layout considerations are the same as the main application. There is a navigation bar on the side that allows the admin to navigate through the admin portal.

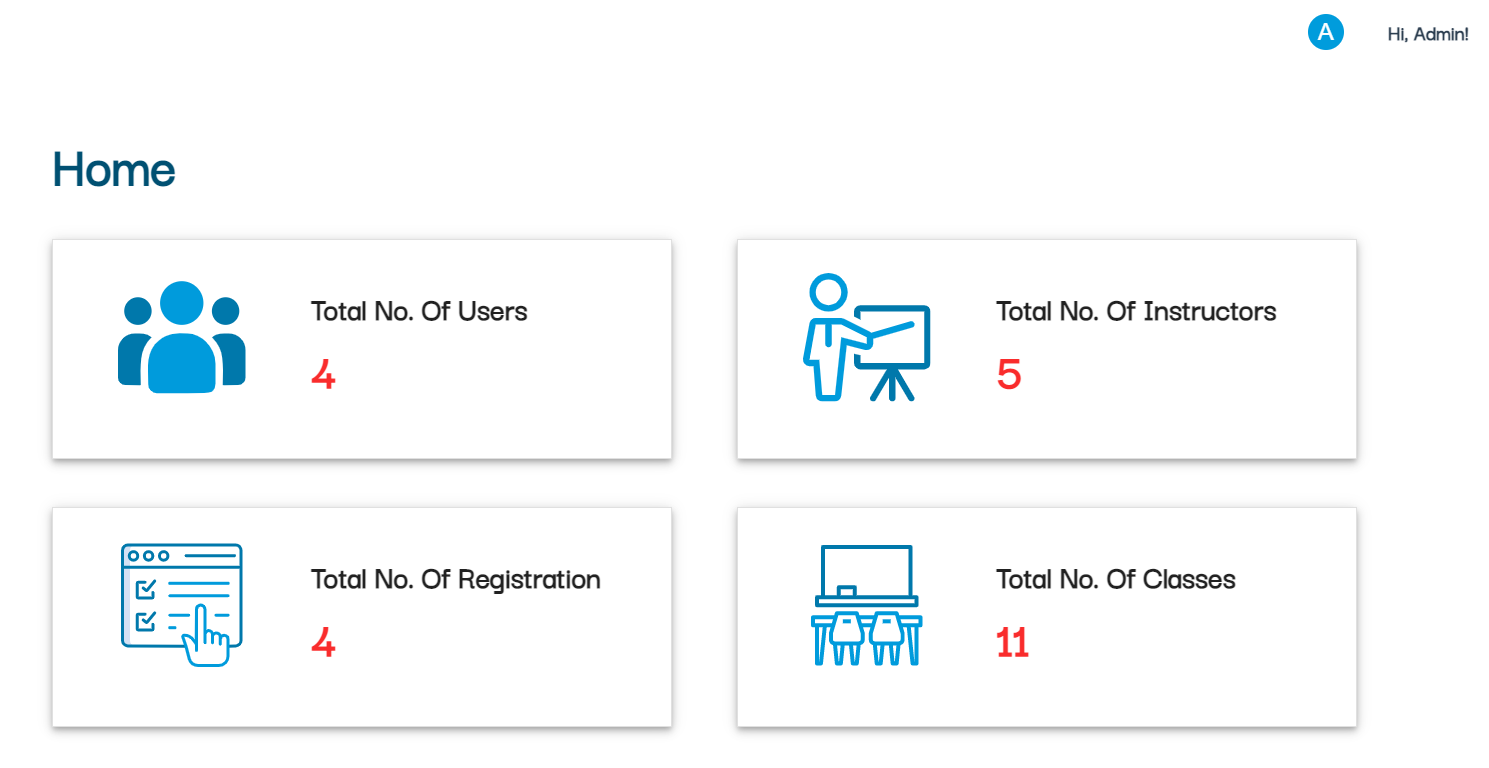


Figure Home page of admin portal

### 6.2.2 Data Table List

Admin can view the list of learners and instructors on the platform. The information is displayed in the form of a data table. Admin can then manage the users by editing their information or deleting the users’ accounts. When an account is deleted, data will not be removed from the database instead, it will be set as inactive. A search function is implemented to allow the admin to search the list efficiently.

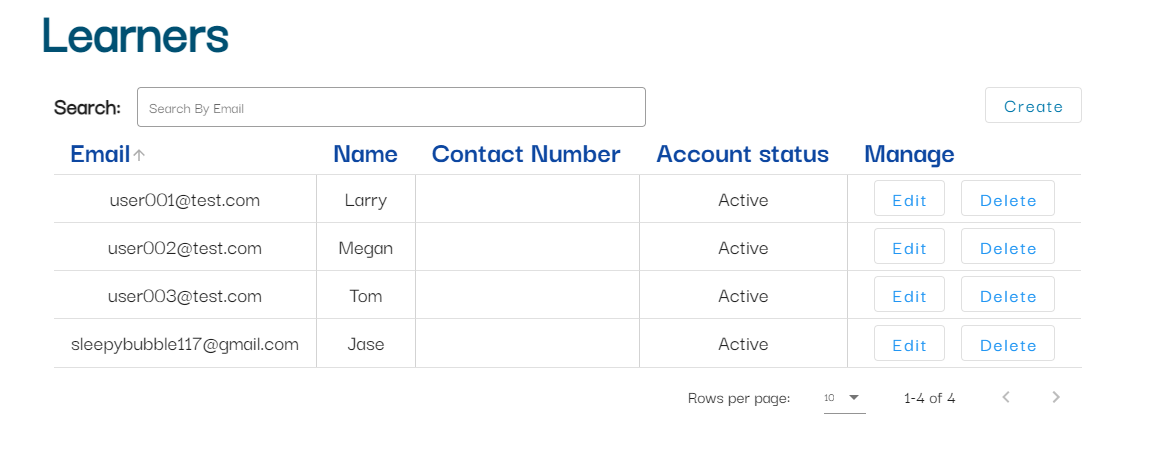


Figure List of learners

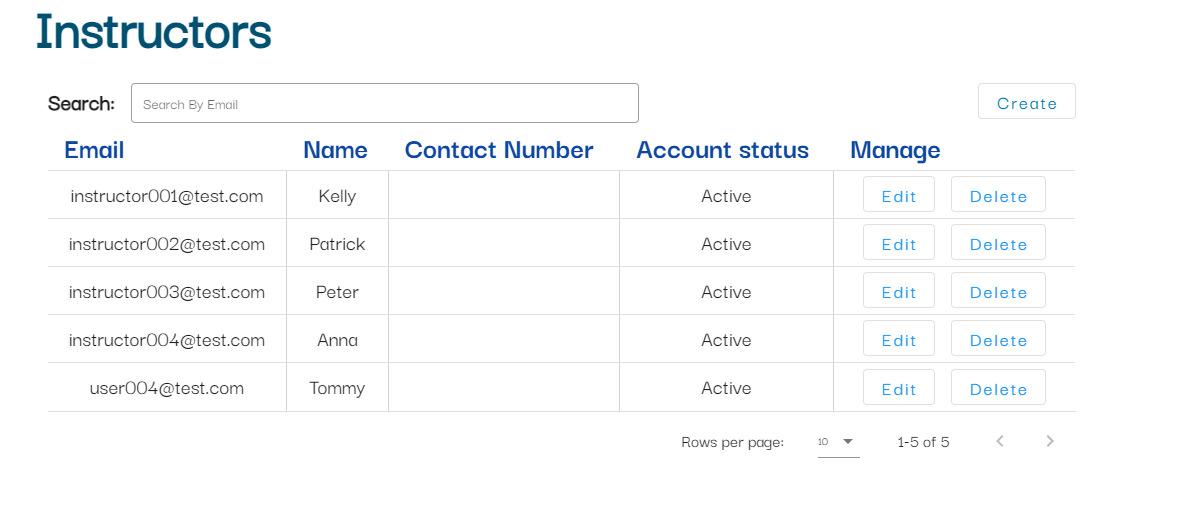


Figure List of instructors

Admin can also create an account for the user by clicking on the “create” button. Information such as user email and name have to be filled in. Password has to be auto-generated by clicking on the button. Once the account is successfully created, an email will be sent to the user with their password.

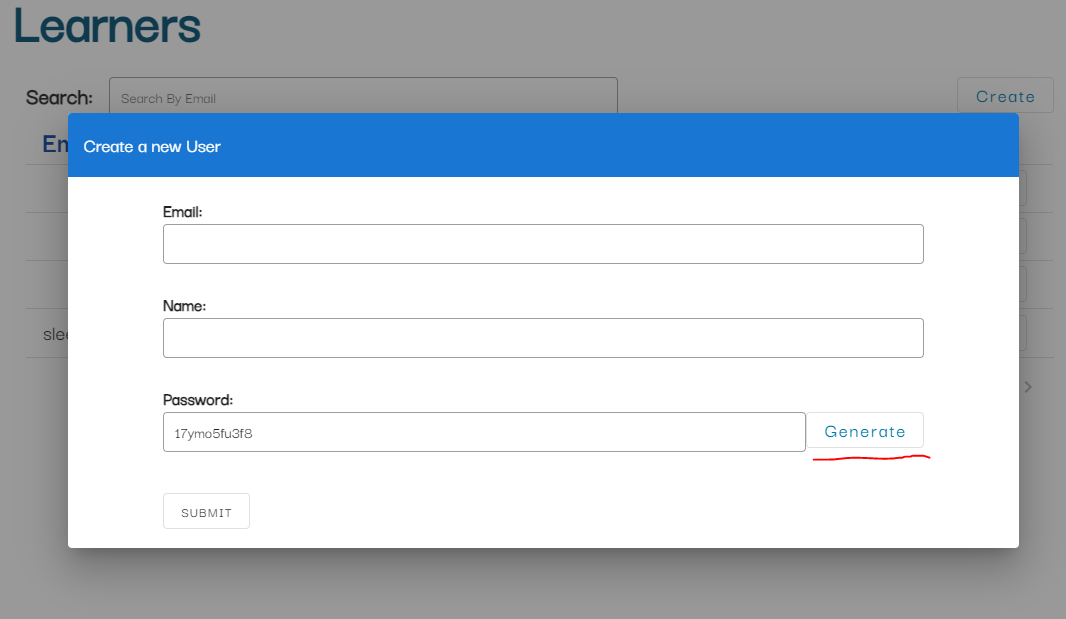


Figure Admin creating a new user

Admin can also view a list of the courses available on the platform. They can also delete the courses, but they are not able to edit the information on the courses. A search function is also available to search the list.

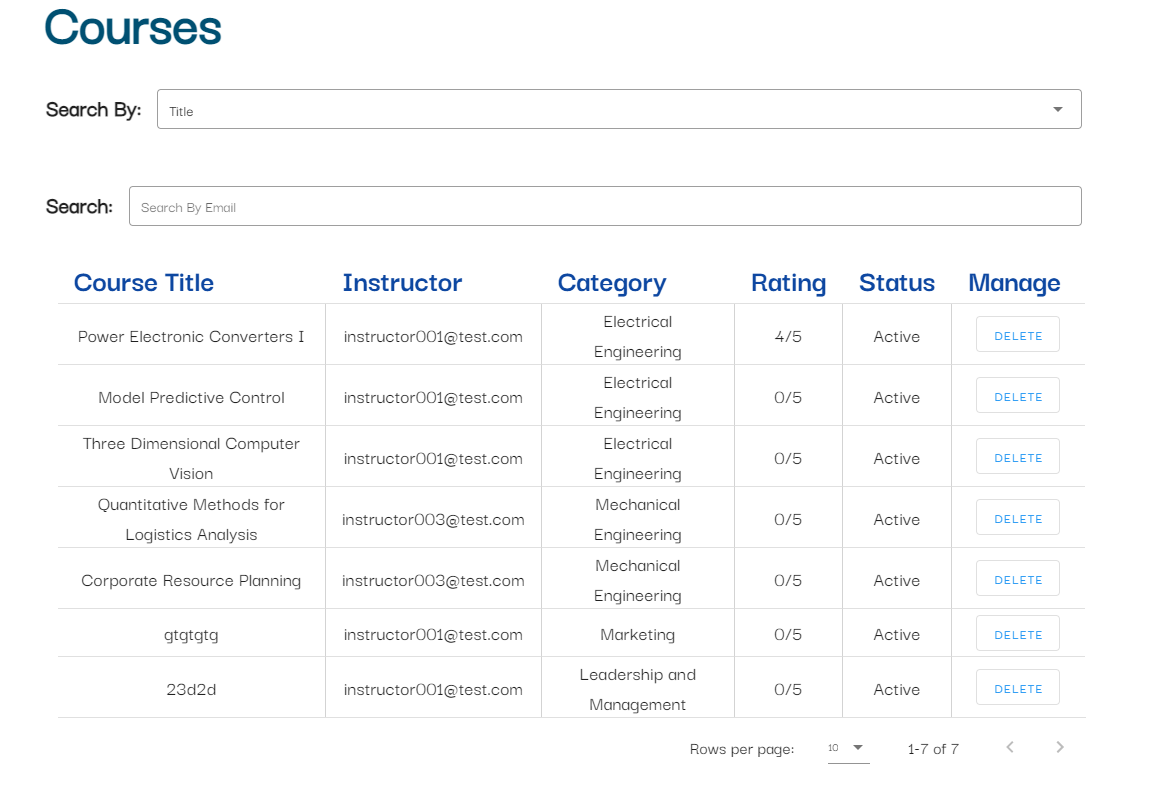


Figure List of courses

### 6.2.3 Instructors applications

Admin will the one who will approve the learner’s application to become an instructor on the platform. Admin will be able to view a list of applications and download the documents that was uploaded together with the application.

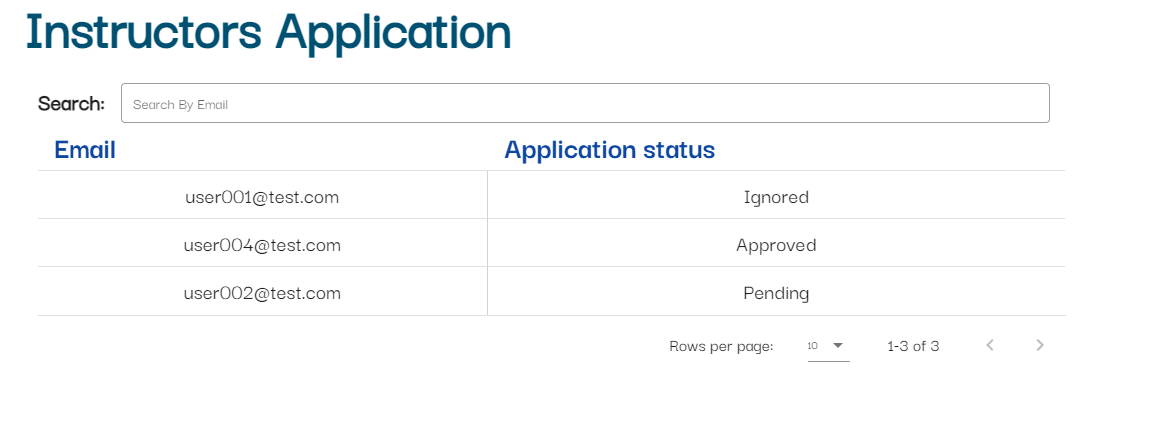


Figure List of applications

Admin can click on the application to view the information and download the document. There are two choices the admin can make, approve or ignore. From figure 71, you can see the result of the decision made by the admin.

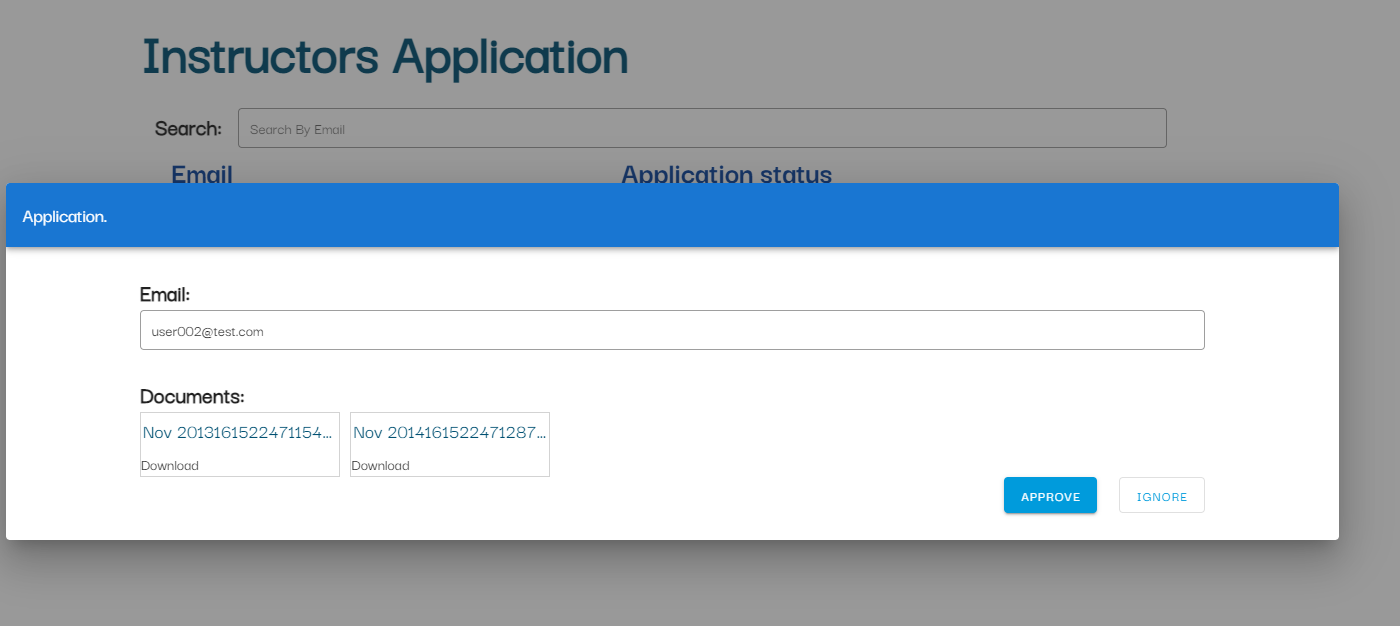


Figure Learner's application information

### 6.2.4 Report Review

The admin can review the report on the discussion board message. Decision to approve or ignore can be made. Once approved, the message will be deleted and replace with a message “Message deleted by admin”.

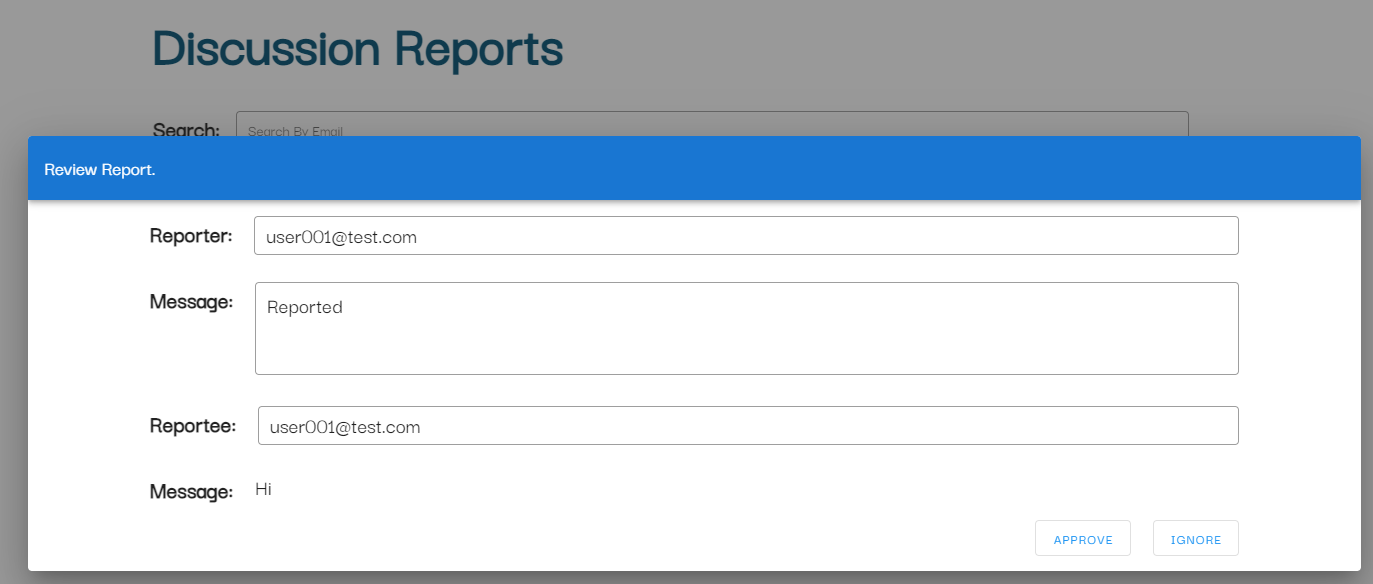


Figure Admin can review report on discussion messages

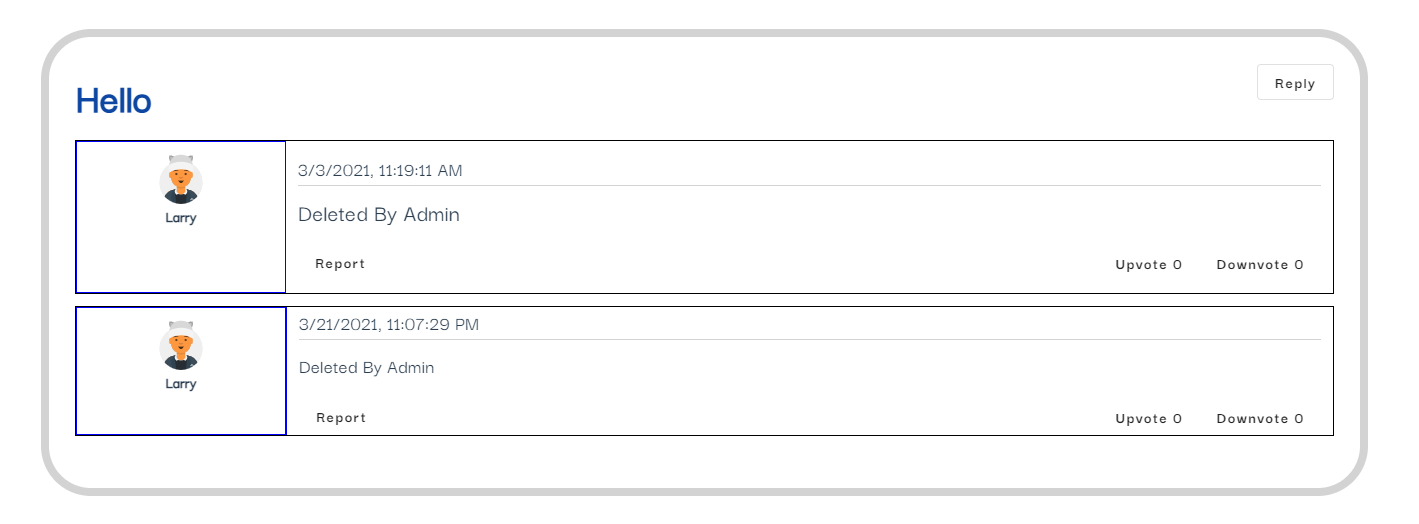


Figure Message deleted by admin

## 6.3 Database & Back-end Implementations

### 6.3.1 CRUD Operations

The administrator role mainly does crud operations on the database. The admin can create or edit users by interacting with the database. The admin can create an account for the user by inserting a new document into the user collection.



Figure Code block to insert a new user document

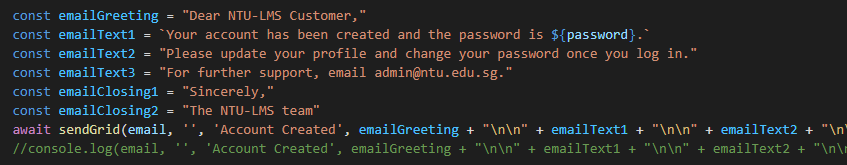


Figure Code block to send an email to user

Admin can also approve user applications to become an instructor. This action will also update the database. If a learner’s application is approved, it will update the related documents to set the learner’s role to be an instructor.

This is the same for approving or ignoring the reports on the discussion board comments. It will update the document with the required changes.



Figure Update the related documents when changing role to instructor

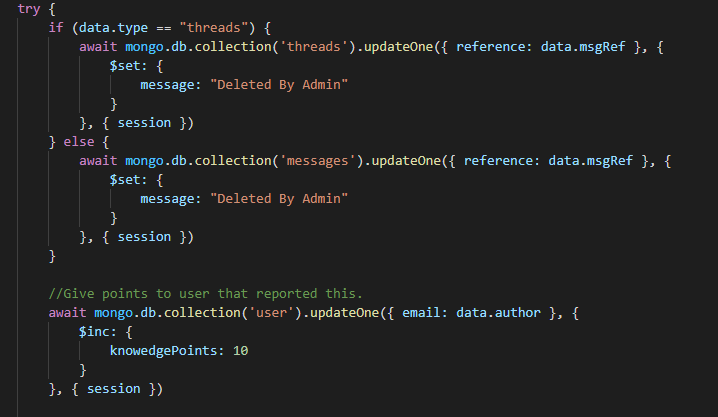


Figure Update discussion message if report was approved

# **Chapter 7 Application Testing**

Application testing is the process of going through the application and ensuring that things are working as intended. For this project, I did a web application testing that focuses on functional and performance testing. I went through the functional requirements that was explained in the previous chapter to ensure that the function is performing as stated.

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Test Case** | **Expected Result** | **Result** |
| FR1 | Registering an account with an email: test123@test.com | There should have a new document in the database |  |
| FR2 | Log in with email: test123@test.com | Should be redirected to the home page. |  |
| FR3 | Log out of the application. | Redirected to the login page. |  |
| FR4 | Go to statistic page after logging in | Statistic should be displayed according to the user |  |
| FR5 | Go to achievement page after logging in | Achievement should be displayed and greyed out if it is not achieved |  |
| FR6 | Go to profile page to see own profile. | Personal profile and basic information should be seen |  |
| FR6 | Go to discussion thread and click on another user profile | Redirect to another user profile page. |  |
| FR7 | Create dummy data with courses that have ended | Users will receive notification on the application. |  |
| FR8 | Go to the browse page and view courses | Able to see courses and filter by category |  |
| FR9 | User register for course. | Successfully registered and able to view under “Course taken” page |  |
| FR10 | Once registered, they should have access to discussion board | Course discussion board appear in the discussion page. |  |
| FR11 | Create a new thread call “Test New Thread” | New thread created will appear in the discussion page |  |
| FR12 | Navigate to class page in learner’s account | Able to see the newly registered course. |  |
| FR13 | In learner account, create a new thread in class discussion. | Able to see the thread created |  |
| FR14 | In learner account, able to view quiz assignment | The quiz assignment will pop up in a new dialog. |  |
| FR15 | Navigate to course taken page. | Able to view a list of courses registered |  |
| FR16 | Create a new course | Able to see the change reflected. |  |
| FR16 | Edit a course title. | Able to see the new change reflected |  |
| FR17 | Create a survey | Survey is successfully created and can be edited |  |
| FR17 | Edit a survey | Able to see the new change reflected |  |
| FR18 | Go to course created page | Able to see a list of course created |  |
| FR19 | Go to discussion board page | Discussion board will reflect own courses section |  |
| FR20 | Go to discussion board page | Class board will reflect own courses section |  |
| FR21 | Create a quiz assignment called “quiz\_test” in class board | Newly create quiz assignment is displayed in class board |  |
| FR22 | View a quiz submission | A pop up should appear to allow marking of the quiz |  |
| FR23 | Go to the students list in page | Able to view the list of students by filtering courses and batch |  |
| FR24 | Log in to admin portal | Able to log in and redirect to home page |  |
| FR25 | Go to accounts tab and select learners page | Able to see a list of learners |  |
| FR26 | Go to accounts tab and select instructors page | Able to see a list of instructors |  |
| FR27 | Select an instructor application and approve | Status displayed should change to “approved” |  |
| FR28 | Create a user | Newly created user should appear be reflected in the list |  |
| FR28 | Edit a user | Changes should be reflected in the database |  |
| FR29 | Approve a discussion message report | Message should be deleted and reflected in the discussion board |  |

# **Chapter 8 Conclusion and Future Work**

## 8.1 Conclusion

The project is built with a popular technological stack that includes MongoDB, Express, NodeJS and Vue. This technological stack ensures that the application is clean and maintainable. It will also be easier for another developer to provide support, maintenance or improvements to the application.

The application serves as a bridge between physical and virtual learning. Even though online learning platforms are becoming more popular, physical classes will not be easily replaced. As highlighted in the future work, the application can also slowly incorporate online learning onto the platform while keeping physical lessons available.

## 8.2 Future Work

The goal of the project was to supplement physical lessons and encourages continuous learning through the platform. It aims to create a community on the platform that will continue to gather and share and discuss related knowledge. Motivates the user to keep learning their simple gamification elements such as levels and achievements. There are still several areas that can take the application one step further by future developers.

With more time and resources put in, we could further improve on a certain aspects of the application or work towards providing a more rounded application. Some of the improvements that can be made includes:

* Working on a more engaging user interface and design
* Implementing options to conduct virtual lesson through our platform
* Implementing e-classes that is based on recorded lessons
* Providing e-certificate that is recognize in the industry

These additional features will be able add more functionalities and usability to our application. With more features and the possibility of obtaining recognizable certificate, it will attract more users to our platform.

# **References**

[1] https://www.investopedia.com/ask/answers/050115/what-difference-between-frictional-unemployment-and-structural-unemployment.asp

[2] https://www.straitstimes.com/singapore/singapore-budget-2020-1000-skillsfuture-credit-top-up-this-year-for-mid-career-workers

[3] https://e-student.org/disadvantages-of-e-learning/#e-learning-requires-strong-self-motivation-and-time-management-skills

[4] https://www.mongodb.com/mern-stack

[5] https://www.bloomreach.com/en/blog/2018/07/what-is-a-single-page-application.html#single-page-applications-advantages

[6] https://angular.io/guide/architecture

[7] https://vuejs.org/v2/guide/comparison.html#Preact-and-Other-React-Like-Libraries

[8] https://learntocodewith.me/posts/backend-development/

[9] https://www.upwork.com/resources/sql-vs-nosql-databases-whats-the-difference

[10] https://www.mongodb.com/nosql-explained/nosql-vs-sql

[11] https://www.nylas.com/blog/how-you-can-use-email-apis/

[12] https://www.piesync.com/blog/mailgun-vs-sendgrid/

[13] https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/#uses

[14] https://www.channele2e.com/channel-partners/csps/cloud-market-share-2020-amazon-aws-microsoft-azure-google-ibm/

[15] https://www.guru99.com/functional-requirement-specification-example.html#6

[16] https://www.guru99.com/non-functional-requirement-type-example.html

[17] https://digitalnext.co.uk/dn-hub/blog/importance-of-responsive-web-design/#:~:text=Providing%20an%20optimal%20experience%20for,of%20your%20brand%20and%20business.

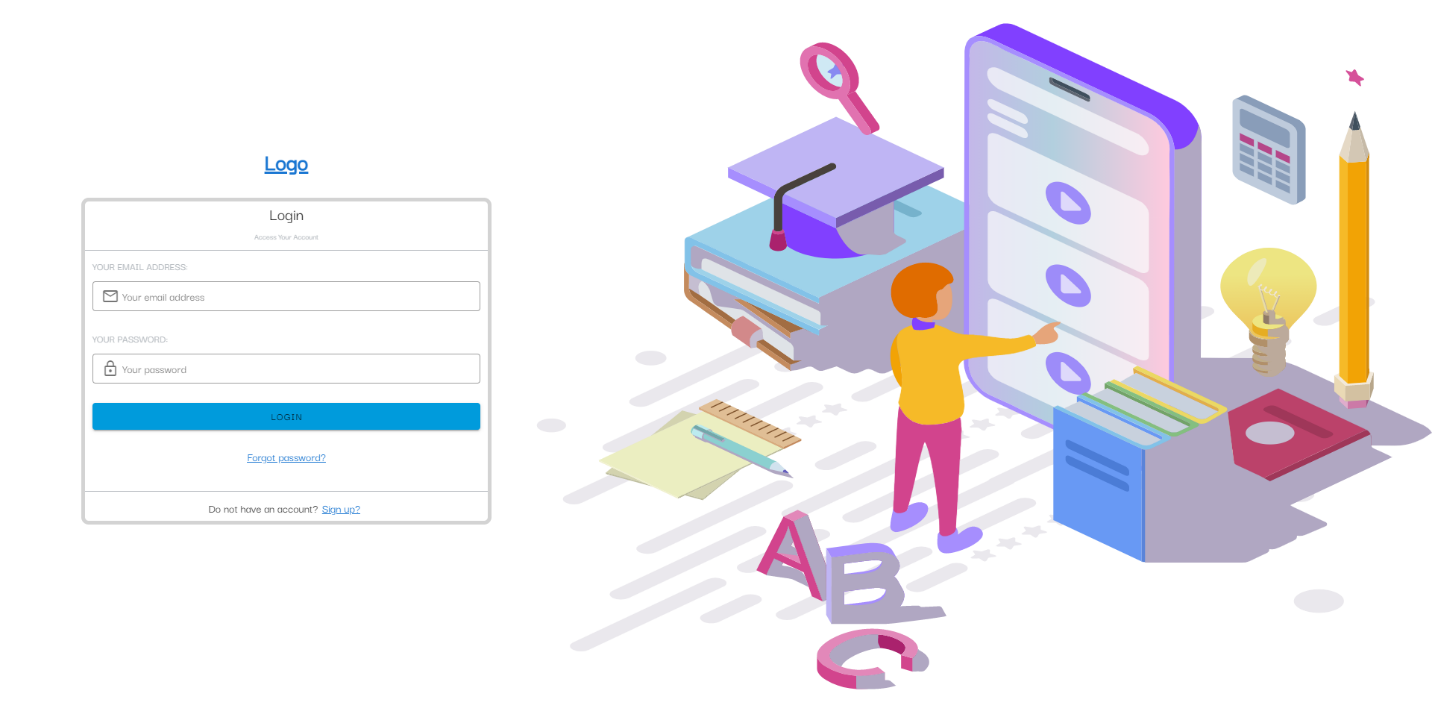
[18] https://techterms.com/definition/media\_queries

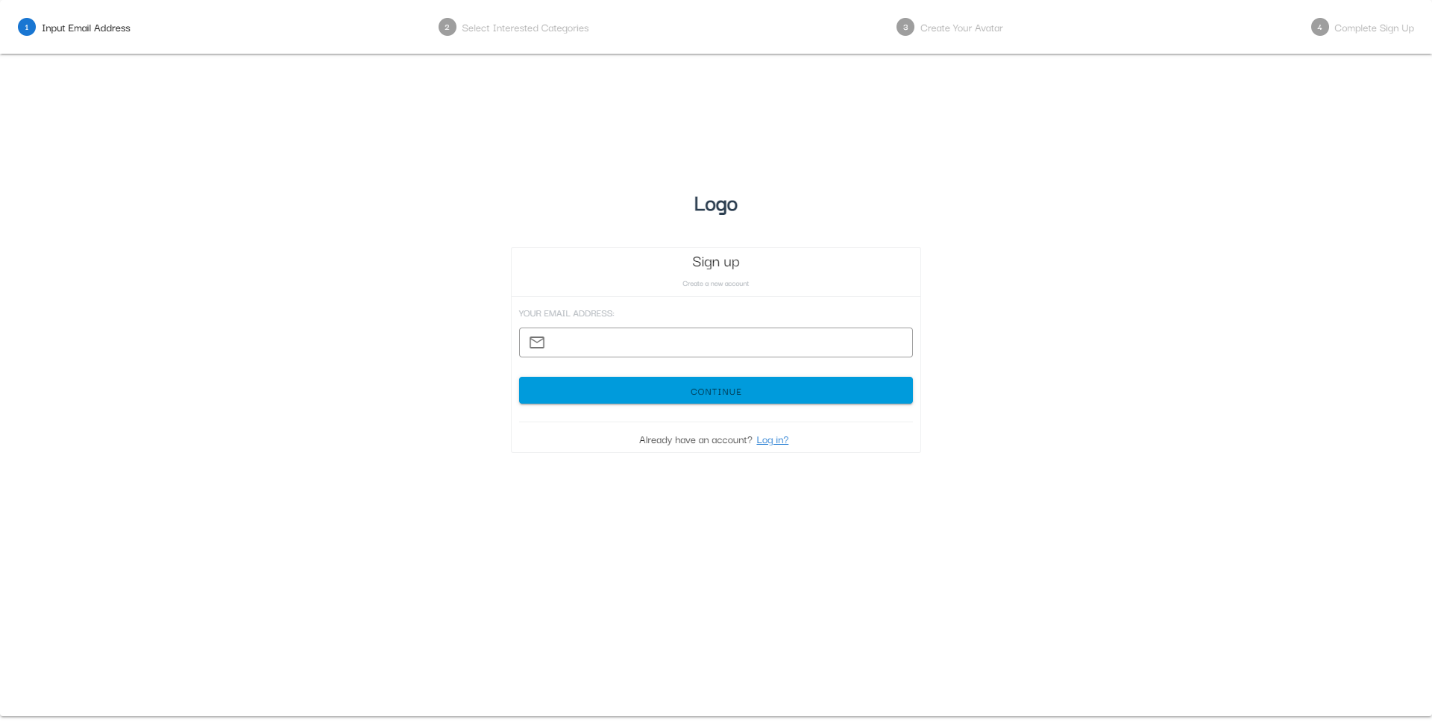
[19] https://css-tricks.com/snippets/css/fluid-typography/

[20] https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express\_Nodejs/Introduction

[21] https://medium.com/@shadikkhan/part-2-mongodb-step-by-step-designing-schema-2f24f86cd304

# **Appendix**





# 

