Student Hub - where Monday could feel like a Friday

Sagheer Ahmad

Mateusz Pawlowski

B.Sc.(Hons) in Software Development

May 10, 2021

Final Year Project

Advised by: Martin Hynes

Department of Computer Science and Applied Physics

Galway-Mayo Institute of Technology (GMIT)

Contents

List of Figures

About this project

Abstract In this project, we built a web application for students which will act as a student portal. We used SpringBoot, ReactJS, MongoDB and Google Firebase as the main frameworks for our web application, user authentication and data storage. We used AWS and DockerHub as our cloud platforms to deploy the Web Application and GitHub for controlling the source code and saving our work. The project is written mainly in java and the react framework and a few other languages such as Javascript and CSS.

Currently, with the times we are in, studying and keeping up with our work is very challenging. We cannot get access to course materials due to several reasons, such as little to no face to face contact with lecturers, and college work is being done all online. A lot of students miss the online lectures due to bad internet and because of this they may fall behind on their work. Other students do not even have the opportunity to visit their institute/university and are not able see any of their peers in person throughout the year. This is what sparked us and made us think that by making a student portal we can enhance and help these students with their college work and allow them to participate with each other.

This project is a research project as it will allow us to get a better understanding of the work that goes into making a high-end web application. It would also allow us to get a better comprehension of the different languages and frameworks that we will be using.

The overall objective of this project is to assist students with their online learning and to make online learning more enjoyable. It will also allow students to reach out to other students through the messaging part of the web application. First year students who have not been to college do not know who is in their class or what they look like. The web app will act as a social media platform which will allow the students to talk to each and get to know each other.

This project supports further expansion and insertion of new code and components such as a user profile page and a timetable to manage classes and modules. The proposed solution will be comprised of a high-end web application that allows the user to log in, talk to their fellow students and save notes in a notepad or as sticky notes. These actions will be done through CRUD functionalities of the app.

Authors The authors of this project are Mateusz Pawlowski and Sagheer Ahmad. We are college students studying at Galway Mayo Institute of Technology currently in our final year of Computing in Software Development.

Acknowledgements We would like to thank our supervisor Martin Hynes for all of his effort and supervision through our final year, as well as the GMIT team for their assistance over the last four years.

Chapter 1

Introduction

Initially our idea for our final year project was to create a web application but we did not know what direction to go in. We thought of different ideas such as a business website for buying and selling products or a blog website. While discussing our ideas together, we realized that due to the global pandemic happening around us, our college life will be affected. We will not be able to see each other and work on this project due to lockdown. We thought that if we are going to have this problem, then there will be other students who will be faced with the same issues. We decided that we would build a web app that will help us with this problem. This web app will allow students to get in contact with other students and share their notes and resources in a global notepad or talk to each other privately and share documents. This project's primary objective is to do research into designing web applications in order to assist students in their learning.

1.1 Developing our Design

At the beginning of this project it was challenging to select the language and the framework we wanted to work with. There were a lot of options that we could have gone with such as Python and Django or IonicJS. We decided not to use these languages and frameworks because we had previously used these, and we wanted to challenge ourselves with something new. While researching about different companies in our field, we discovered the most widely used frameworks that a lot of the companies use are SpringBoot and ReactJS. We thought this would be a great opportunity to learn new frameworks and enhance our skills further in Java.

This project uses the ReactJS framework to build the front end of the web

app, and it uses the SpringBoot framework for the back end of the application. The ReactJS framework uses Javascript as the primary language while the SpringBoot framework uses Java as the main language. We wanted to use MongoDB and Google firebase as the databases for the project because they are the most popular databases in the industry. We also wanted to run this project on the cloud so that it could be accessible from anywhere. We had numerous options when it came to cloud services, such as Google Cloud, Amazon Web Services and Microsoft Azure. We decided to use Amazon Web Services (AWS) and DockerHub for this part as they are industry standards.

The figures below indicate how both of these frameworks work. We will address these frameworks in depth further in the dissertation.

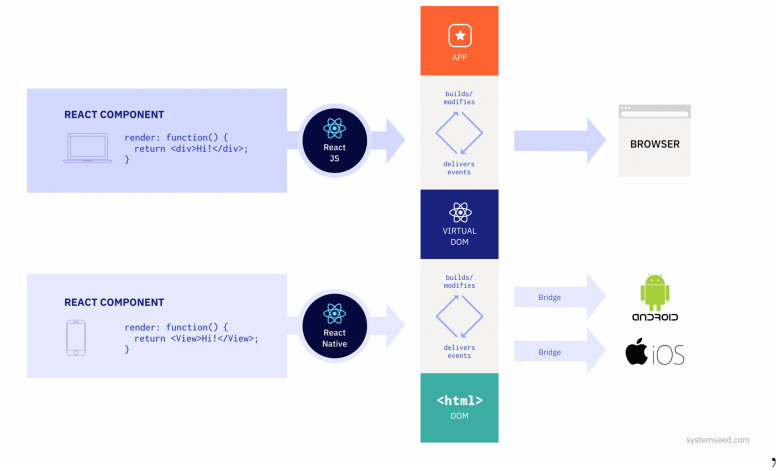

Figure 1.1: React Framework

1.2 Final Design

Student Hub, is a web application project that involves SpringBoot, ReactJS, MongoDB, FireBase and AWS cloud technology. The end product of this project will be a web application that will be running on AWS and can be accessed anywhere and on any device. The user will have access to several services to help them get through each semester of college and have a pleasant experience throughout. There will be a facility that will authenticate the user and give them access to these services. The users will be saved in MongoDB, running on the cloud. The user will also be able to contact us regarding any queries that they may have through our live contact page. Below is a

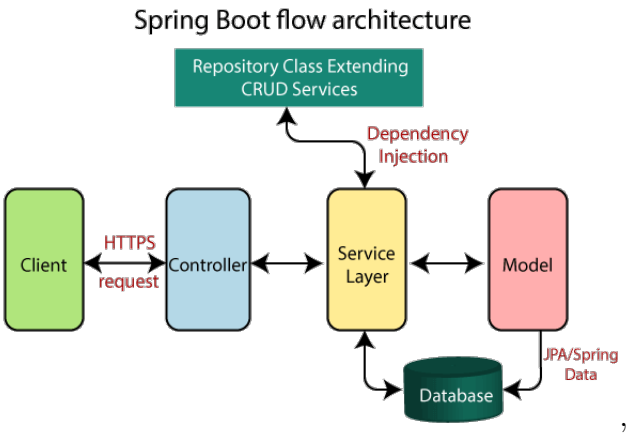

Figure 1.2: Spring Framework

schematic diagram that we came up with at the start of the project to give us an idea of the end product.

1.3 Objectives

In order to provide a high-end web application that is also high performing, multiple goals needed to be accomplished during the course of this project.

- The first objective was to create a SpringBoot and ReactJS application that could talk to each other and save details to the Mongo database. We also wanted to have a basic web application set up i.e. home page, navbar and footer.
- With the initial web app configured, we want to work on the common pages that usually appear on a website such as a contact page and an about us page. In order to achieve this we will be running a node server that will detect when the user wants to contact us and will send us an email.
- Subsequently, we will create pages for the user authentication. Once we have designed the register and login pages, we will connect them to the back end where the data will be saved in our database. The password will be hashed using the hash algorithm for extra security. We will implement a ReCAPTCHA on the login page to provide further security against bots and androids. The mongo database will be setup on the cloud so that it can be accessed from anywhere.

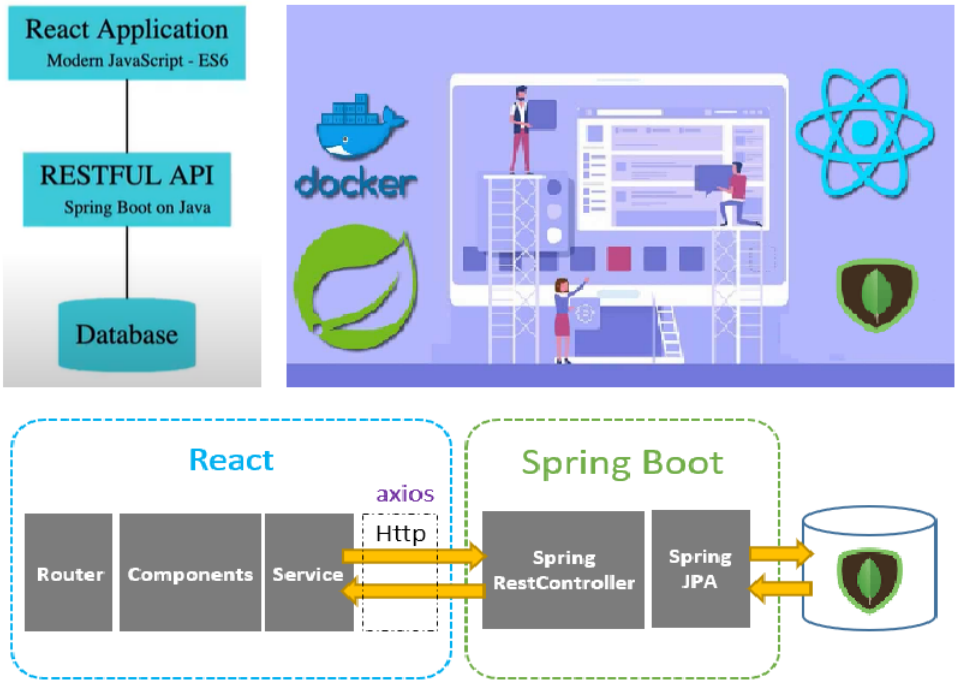

Figure 1.3: Schematic Diagram

- After the user has successfully logged in, they will be able to access a number of features such as sticky notes, notepad, messenger and a student forum page. All these features will be connected to the firebase database to allow for quick access. The messenger chat feature will be using an external API known as chat engine. We will edit this API and add our changes to the chat page.
- Lastly, the next step would be to deploy our project to the cloud. This will be completed through the use of Docker files and Docker-compose. The docker files will then be run on an Instance that we create in AWS.
- If we have extra time, we would also like to include a user profile page where the user can customise their profile and it can be viewed by other students.

1.4 Overview

As this is our final year project, we knew that we would have to challenge ourselves to work on a project that will allow for independent learning. We wanted to make sure that we learned something new through each stage of development. This dissertation will be laid into different chapters, these chapters will explain our thought process and the different aspects of the project. This section contains a small description of each of the chapters that we will be covering.

1.5 Methodology

In this chapter of the dissertation will be focusing on the steps that we took throughout the development stage to make sure we have a successful project. We will discuss the reasons for various technologies we used such as, SpringBoot, ReactJS and mention the problems that we encountered during the initial set up and the development stages of the project. We mainly used the agile methodology, and we will further provide diagrams where needed to show off the methodology used in our final project design.

1.6 Technology Review

This chapter will be looking at all the different technologies that we have encountered during our research. We will go into further details on how to set up various technologies, for example Docker, Robo3T, Postman, and their key functionalities and advantages.

1.7 System Design

This part of the dissertation will focus on the different components and how they were implemented. Snippets of our code and images will be provided when explaining components to give a better understanding of the project. Each component will be discussed in great detail and we will explain how each part of the project contributes to the final product.

1.8 System Evaluation

This section of our project will discuss how our project performs, how scalable it is and its robustness. We will also mention any limitations that we encountered such as, latency issues and cloud issues.

1.9 Conclusion

This chapter will be going over our goals and challenges that we set ourselves with. It will also cover if we accomplished these challenges and goals. The end result of the project will be discussed alongside all the difficulties that were experienced. These difficulties will then be discussed to show how we overcame our problems.

1.10 Project Links

Link to repository

- https://github.com/sagheerahmadGmit/FinalYearProject

Chapter 2

Methodology

As previously said, this section focuses on the various methodologies used to deter possible disasters. It was important to schedule and monitor the production stage ahead of time, so that any unanticipated issues or glitches could be addressed. This enables us to make smarter plans, resulting in a faster planning phase and less lost time. We can also write simpler, more readable code by planning ahead. If this project is not properly prepared, it will result in events such as missed deadlines or the need to cut back on the project, resulting in the goals not being achieved.

2.1 Planning Phase

We defined the project's scope during the initial planning process. We concluded that Agile would be the best methodology to use. We investigated other methodologies, such as waterfall, but after comparing them to Agile, we opted to stick with Agile. We were able to divide the project into multiple stages, thanks to the Agile approach. It allowed us to make incremental improvements to the project at each point. An Agile approach allows for greater flexibility in project planning and execution. We could also modify the project's various elements as it progressed. After discussing the different methodologies, we had to decide on the technologies to use. We researched into several technologies and there were a lot of good options to go with, but we concluded that Java, SpringBoot, ReactJS and MongoDB would be the main technology for developing this web application. In addition to the technologies, we had to choose an integrated development environment (IDE). We knew we had to use IntelliJ for the backend, we had previously used it in one of our modules and wanted to expand our skills and knowledge on it. We decided to use Visual Studio Code for the frontend. This way it would be

easier to navigate through the code and see where the backend and frontend connect to each other.

Figure 2.1: Agile Methodology

We developed a few diagrams during this process to help us better visualize the work that will be needed. The diagrams looked a lot like the schematic diagram described earlier. Since we couldn't meet in person to collaborate on the idea, we had to think about how we would approach everything. When doing research for this initiative, we came across a programming technique called mob programming. After doing more research, we decided to tackle the project with a mob programming approach. Mob programming is a method of software development in which a group or team of people work on the same project element at the same time. This allowed us to be constantly on track as we both have to be working together. We made the decision to call a couple days a week to work on the application.

2.2 Requirements Analysis

One of the most important facets of this project was defining the project's criteria, as it would help us to understand what we need to do and split each phase down into steps. These phases will then assist us in completing various activities based on priority and importance. We had to explore and research what is needed for a viable web app, as well as the components that are required. We needed our web app to be able to meet these standards as efficiently as possible. These requirements are as follows:

- To begin, we must have a functioning web application that operates correctly and enables the user to perform the function for which it was designed.
- Second, provide a completely functioning and stable register and login framework that saves user data in a database. To add to the encryption, the password must be hashed.
- Next, the user should then be able to log in and communicate with other students, as well as save notes and documents to their account. They should also have access to the student forum, where they can post questions and get answers.
- After that, we would like to have the project running on the cloud through docker hub and AWS.
- Finally, if we have enough time, we'd like to convert our web app into a mobile app so that users can access all of their information on the go.

2.3 Meetings

Since we were a two-person team, we had to organize and arrange our meetings to collaborate on the project on specific days and times. We had to schedule it around other classes and assignments. We agreed that we will work on the project every other day and make a few commits. Since we had other assignments to complete during the year, this was a little more complicated at times. We had talks with our supervisor as well. Every week, either on Monday or Thursday, we met with our supervisor. This was to provide a report on our development, get input on any prior work, and address potential future proposals. The minutes of the meeting were then emailed to our supervisor, and we hoped to complete the recommended conditions for the next meeting. We used Microsoft Teams as the main tool for communication and meetings.

2.4 Development

We divided our work into smaller tasks and kept track of all these tasks on Atlassian Jira. We registered any bugs or concerns on Jira so that if we have these problems in the future, we can report back to this documentation.

To ensure that we were on track with the project, we reviewed each feature separately and made regular GitHub commits.

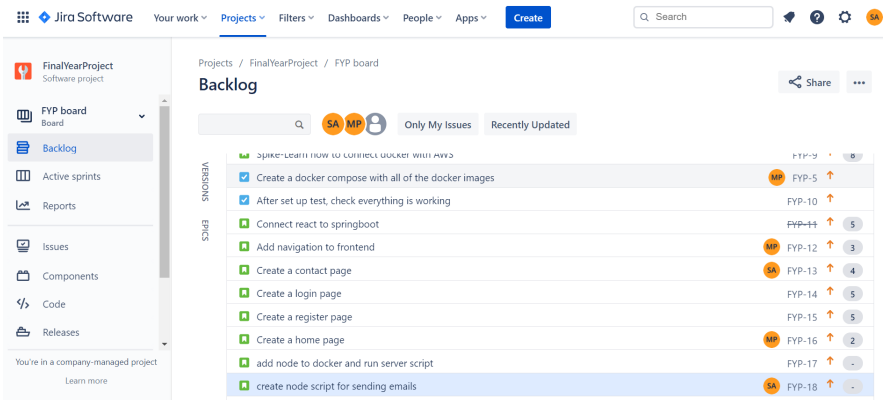

Figure 2.2: Atlassian Jira Software

This is an example of how we used Atlassian Jira. We gave each task a story point and that dictated the amount of time that we could devote to that specific task.

2.5 Testing and Validation

Testing was a critical component in this project, as well as software in general. Testing saved us a lot of time by preventing any future catastrophes in the code and project. It also reassures us that the software can fit in with other features and elements of the web application. To test the project, we used Robo3T and Postman, among other tools. We carried out black box testing on the project by allowing other students to register and access the website in order to assess the various functionality and components. Black-box testing is a form of software testing that looks at an application's capabilities without looking inside its internal mechanisms or workings. It has no understanding of the website's back end or the code that makes it work. This was extremely beneficial because we received direct feedback from students. We asked them if we could strengthen and change things, and we tried to make the necessary improvements. Allowing a client to log in with incorrect information and watching what happens or attempting to enter a website function without being logged in, are examples of black box testing.

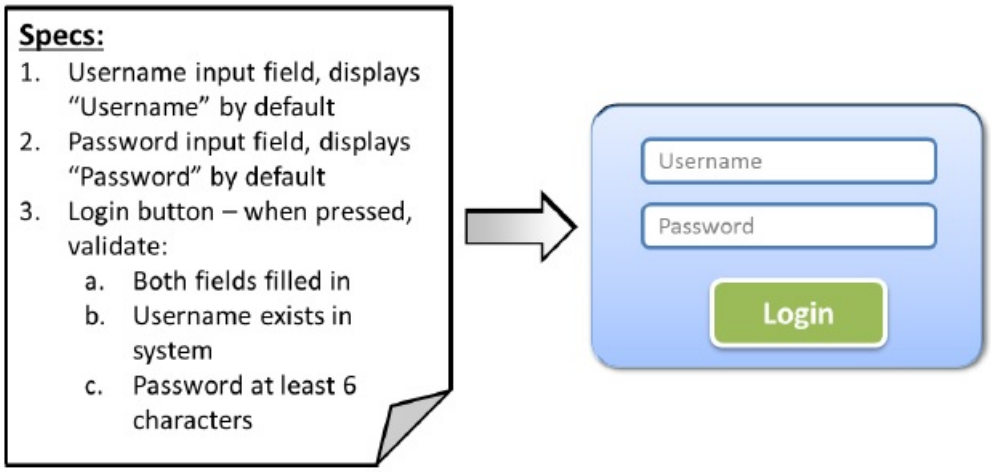

Figure 2.3: Blackbox Testing

2.6 Problems Encountered

We ran into a lot of issues and failures during the early stages of development, including docker not running, the user being unable to save their credentials while registering, and GitHub failing to push our code to the correct branch. We had to look at these mistakes and come up with suitable alternatives. We made a note of the solution after we had resolved these problems. This was so that if the same issue arose again, we would know how to correct the mistakes and issues.

2.7 Version Control Manager

We wanted to select a version control manager before we began working on the actual code component of the project. There were a couple of great options, including Bitbucket, an Atlassian version control repository hosting site, and AWS CodeCommit. We concluded that using GitHub, which is more well-known and widely used in the tech industry than any other version control manager, would be more useful to us. We had been using GitHub for our work and projects since our first year of college, so it made the decision easier. GitHub has a lot of features that we could take advantage of, the most important of which is the ability to work on various branches to finish different parts of the project. It also allowed us to keep track of our issues inside the project repository. If we make an error or accidentally insert the incorrect file, GitHub helps us to revert our commits and return to the original code.

We knew we wouldn't lose any of our code this way. Since GitHub helps us to see where in the code we make modifications and how the code is affected, we were able to quickly see where we had bugs and errors. We decided to commit to GitHub on a regular basis to encourage us to work on the project more often. We made small commits rather than huge commits or one final commit at the end. This is because it demonstrates our progress and thought process in the development stage. It further demonstrates the many issues we encountered and how we addressed them. In our view, GitHub was by far the best version control manager.

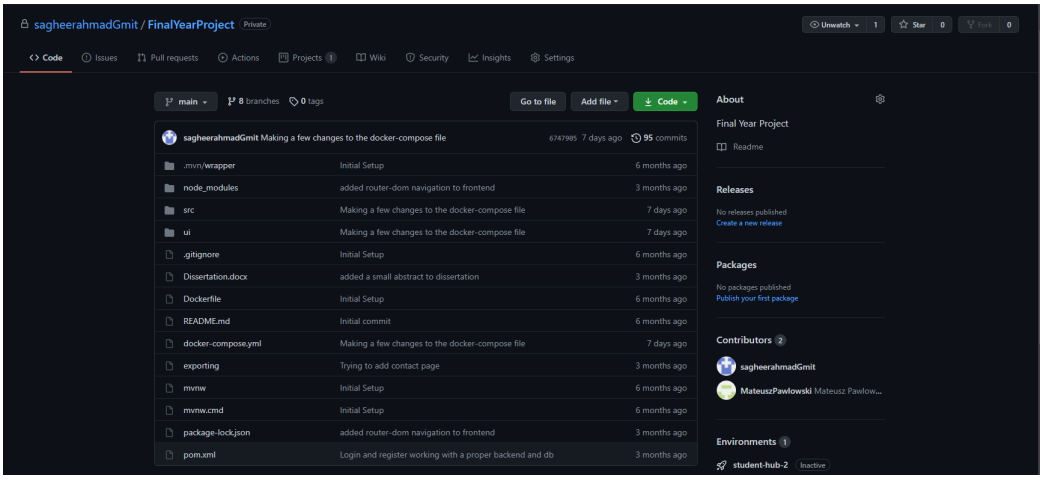

Figure 2.4: Github Version Controller

Chapter 3

Technology Review

This segment discusses the various technology that we used to construct our web application. We will use code samples and visual aids to help you visualize how the technologies work. The technologies we will be talking about are:

- MongoDB
- Firebase
- NodeJS
- ReactJS
- Spring Boot
- Postman
- Robo3T
- Atlassian Jira
- GitHub
- GitHub Desktop
- Docker
- Amazon Web Services (AWS)
- Visual Studio Code
- IntelliJ

3.1 Frameworks and Libraries

We used a few different libraries and frameworks while working on this project. These are Spring Boot, ReactJS and NodeJS.

3.1.1 Spring Boot

Figure 3.1: Spring Boot

Spring Boot is a java-based open source framework for developing microservices. It's also used to make stand-alone, industrial-grade spring applications. The majority of Spring Boot applications only require a few configurations to setup and run. Spring Boot has a variety of features. You don't need to deploy any WAR files because Spring Boot will embed Tomcat directly into the program. Tomcat is a Java application server that runs servlets and renders web pages using java server page coding. Spring Boot also comes with starter dependencies, which will make building applications easier. It will also dynamically configure any third-party libraries, stopping the program from crashing due to errors. Spring Boot also doesn't require any code generation or XML setup.[10]

Java

Spring Boot is written in Java. Java is a concurrency-oriented, object-oriented, and class-based programming language. Sun Microsystems first

published it in 1995. There are several technologies that we use on a daily basis that would not function without Java, ranging from mobile phones to game consoles to supercomputers, and more are being developed every day. Since then, Java has risen in popularity and is now one of, if not the most common, programming languages.[11]

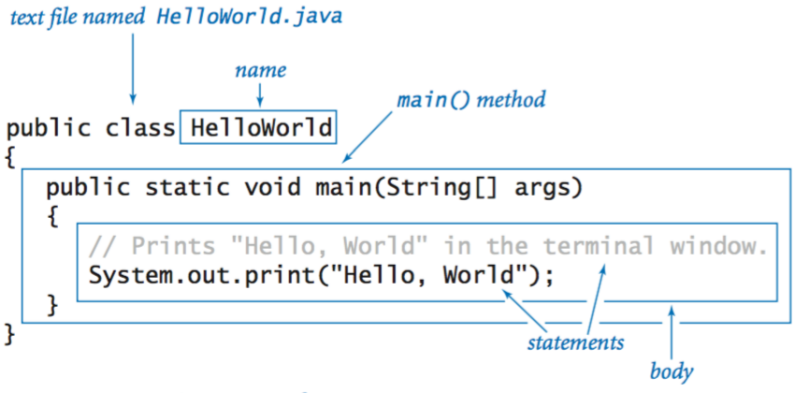

Figure 3.2: Java Hello World Program

For the back end of the project, we're using Java and Spring Boot. Both the front end and the mongo database are connected to our Spring Boot application. The Spring Boot program receives a register request from the front end and transfers it to the back end. An authentication controller is used by the Spring Boot application to process this incoming request. The authentication controllers would then submit this information to the user repository to see if the user's information is valid. If the user continues, the controller will respond with a message informing them that they have already signed up with these credentials. If the information is unique, the password is hashed, and the information is stored in the Mongo database. As a default, the user is given the role of user.
If the user sends a login request, the authentication controller will manage it once more. It will submit this request to the user repository to see whether the specifics match any past users; if they do, the user will be able to log in successfully; if they don't, the user will get a "incorrect details" response.

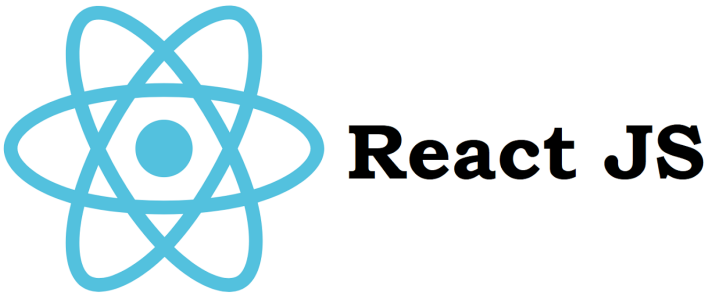

Figure 3.3: ReactJS

3.1.2 ReactJS

React is a JavaScript library for building user interfaces and UI modules that are open source and component oriented. Jordan Walke, a Facebook software developer was responsible for creating React. Facebook and a consortium of small developers and companies are now in charge of it. React is mainly used to develop single page websites and smartphone apps. It is arguably the most popular framework in the world for web application, development which means there are a plethora of tutorials and guides available. Netflix, Instagram, and Airbnb are only a few examples of major existing businesses that use React. The Model View Controller architecture is used by React, which ensures that the app's view layer is in charge of how it looks and feels.[12]

We used a variety of languages and styling sheets during our React development stage, including HTML, JavaScript, and CSS/SASS/SCSS.

HTML

HTML stands for hypertext mark-up language, and it is used to create documents that are meant to be viewed through a web browser. It can be aided by technologies like CSS for styling and JavaScript for complex functionality implementation. HTML includes all of the required elements for anybody to create tables, lists, add pictures/videos, simple bold or italic text, buttons, check boxes, and so on. It contains the fundamentals of any programming language, is extremely simple to learn, and is supported by all web browsers. HTML 5 is the most recent version, which was first released in 2014.[14]

```
1  <html>
2    <head>
3    </head>
4    <body>
5      <h1>Hello World<h1>
6    </body>
7  </html>
```

Figure 3.4: HTML

JavaScript

```
await SaveUser(details).then(response => {

    console.log(response);

    if(response.message === "User registered successfully!"){
        history.push( path: "/login")
        alert("You have successfully Registered!!")
        verify = true;
    }
    else{
        alert(response.message ? response.message : "The details entered are incorrect!!")
    }
});
```

Figure 3.5: JavaScript

JavaScript, or JS for short, is an interpreted programming language that adheres to the ECMAScript principles. It is the world's most commonly used programming language. Though HTML and CSS give a web page layout and style, JavaScript allows you to add complex interactive elements that are appealing to the user while being simple to manipulate and use. Importing JavaScript into your code greatly increases the user experience of the web page through translating a static web page into an interactive one. JavaScript is a scripting language that is both lightweight and complex. It supports the use of an object-oriented programming approach. One of the core languages used by the ReactJS library is Javascript. It is more widely used by ReactJS than HTML and CSS.[28]

CSS/ SASS/ SCSS

```css
#form-app {
    max-width: 600px;
    padding: 1rem 2rem 5rem;
    background-color: #fbfbfb;
    border-radius: 8px;
    box-shadow: 2px 2px 6px rgba(0, 0, 0, 0.3);
    margin: 100px auto;
    color: #384d4a;
    font-family: 'Nunito', sans-serif;
    font-size: 0.8rem;
}

h1 {
    margin-bottom: 3rem;
}

.form-container {
    margin: 0 2rem;
    text-align: left;
}
```

Figure 3.6: CSS

Within our project, we employ a variety of styling techniques; the following is a brief description of some of the styles we implement:

- CSS, or cascading style sheets, is a term for describing the presentation of text written in a mark-up language like HTML. CSS is not limited to HTML and can be used in any XML-based markup language. CSS is used to adjust the look of a page's colors, backgrounds, and fonts, among other things.[14]

- SASS (Syntactically Awesome Style Sheets) is a pre-processor scripting language with more capabilities than regular CSS. It offers a more elegant CSS syntax. SASS creates CSS after it has been compiled.[13]

- Sassy Cascading Style Sheets, or SCSS, is a newer CSS syntax and extension. Unlike other styling sheets, it supports nesting rules, inline imports, and variables. It also aids in keeping things more organized and speeds up the development of style sheets.[29]

3.1.3 NodeJS

Figure 3.7: NodeJS

NodeJS is a backend Javascript runtime server that was built on Google Chrome's V8 Javascript Engine. It was developed by Ryan Dahl in 2009. It is mainly used for event driven servers because of its single threaded nature. We used NodeJS to create and run a server that would allow us to create a fully functional contact page. The server uses external libraries such express and NodeMailer to send an email. It runs on port 5000. When the user fills in the detail on the contact page and presses send, it sends a request to this server, which in return sends an email to our Student Hub Gmail account[24].

3.2 Databases

The data tier refers to the processing of data that have been stored in the database. The user will be able to perform CRUD (create, read, update, delete) operations on the application. We have two databases setup for our project. Our account information is kept in MongoDB, while our website data is kept on Firebase.

Figure 3.8: MOngoDB

3.2.1 MongoDB

MongoDB is a document database, meaning it contains information in JSON like formats. According to MongoDB, "We believe this is the most natural way to think about data and is much more expressive and powerful than the traditional row/column model". MongoDB's goal is to store and recover data easily so that developers can switch and program quickly.[1] MongoDB is a NoSQL database, which means it isn't tabular and doesn't store data in the same way as relational tables do. MongoDB has several features, including horizontal scaling (distribution of data across multiple devices allowing for a better distribution system) and load balancing.[2] It can support modular schemas and is readily elastic when dealing with massive volumes of data and heavy user traffic. It allows data to be viewed instantly after being inserted into the database. MongoDB allowed us to change the structure of the database as the project progressed. Ad-hoc queries can be used by developers to obtain real-time metrics. Ad-hoc is a short-term command that is dependent on the value of a variable. With these features MongoDB is a very efficient software that helps us to view and analyse our data.[1]

As previously said, MongoDB stores all of its data in JSON format. JSON uses fewer data in general, lowering costs and reducing storage requirements. As a result, JSON parsing is faster than other formats such as CSV and

XML. No matter what programming language you use, JSON is easier to read and map to domain properties.[3] The mongo database was originally set up locally to validate the project, but after the initial implementation stage, we migrated the database to AWS cloud using a docker images.

Robo3T

Robo3T (originally known as Robomongo) helps users to communicate with MongoDB data using visual indications rather than a text-based interface. It has a graphical user interface that is like that of a desktop graphical user interface. It supports cross-platform applications, which means it can incorporate the mongo shell into its user interface for text-based or graphical interaction.[7]

While testing the project, we used Robo3T to keep track of our data. We had to make sure that the users were saved to the database and that the passwords of the users were hashed. We have used Robo3T to retrieve data and build new columns and tables. In the figure below we can see that there are several users saved in the user's section of the database

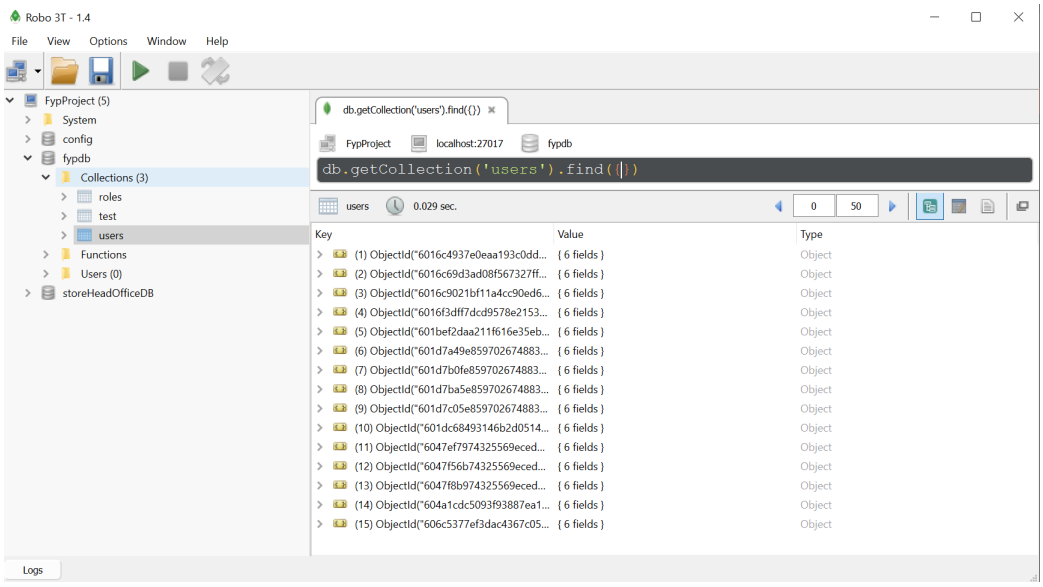

Figure 3.9: Robo3T

Figure 3.10: FireBase

3.2.2 Firebase

We chose firebase to handle all the data on our web app such as the student forum page or the student notepad element. Firebase is a database provided by Google to enable developers to create high-quality web apps easily. It has a variety of tools and services that aid the developers to grow their user base and create quality apps. Just like MongoDB, Firebase is also a NoSQL database and stores data in JSON format. Firebase has a lot of useful features that make developing more fun. The data is synced in real time and remains available even when the app goes offline. It can provide fast hosting for a web app and makes the whole hosting process much easier.[5]

Software development kits (SDK) are used by Firebase to provide methods for generating and handling data. These SDKs will submit information to Google's cloud database, which can then be shared with other web users.[16] All users share a single database instance and are immediately updated with the most recent information. If the user loses connection, the data will still be transmitted to the database. Below are details of the app that we created in firebase and the SDK to that app.

3.3 Integrated Development Environment (IDE)

We used two different IDEs to work on our project. We used Visual Studio Code as the main IDE for the front end and IntelliJ as the main IDE as the main IDE for the backend.[17]

```
// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
var firebaseConfig = {
  apiKey: "AIzaSyDEiiWJW8VdhtDIYl09Eh1xsftc9cuzG1E",
  authDomain: "studenthub3-92319.firebaseapp.com",
  projectId: "studenthub3-92319",
  storageBucket: "studenthub3-92319.appspot.com",
  messagingSenderId: "927205554956",
  appId: "1:927205554956:web:a2b2c1d64cfde1c3ada729",
  measurementId: "G-QML47SNVNK"
};
```

Figure 3.11: FireBase SDK

3.3.1 Visual Studio Code

Figure 3.12: Visual Studio Code

There are a lot of great IDEs that we could have chosen, such as Atom or NetBeans but we decided that we wanted to go ahead with Visual Studio Code as our development tool when building our front end. It is a Microsoft developed source editor that has the benefit of being compatible with the most popular operating systems such as Windows, Linux, and MacOS. Visual Studio Code is very popular in the outer world as it is very programmer friendly and compared to editors like Visual Studio it is small in size and

fast. Depending on your needs, the editor allows you to download several extensions. Many languages are supported by Visual Studio Code, including JavaScript, C, Dockerfile, and CSS. Visual Studio Code can also be used to NodeJS projects as well as ASP.NET It is highly customisable, with users being able to add their own shortcuts, it is very simple to navigate and has excellent code refactoring which was very useful for us especially with larger classes.[18]

We both have familiarity with Visual Studio Code, having used it for over two years on a variety of projects. We had tried a few extensions in VS Code and concluded that it was the better option. We also used Visual Studio Code's IntelliSense function, which allowed us to type Ctrl + Space to get code suggestions instead of typing it character by character.

3.3.2 IntelliJ

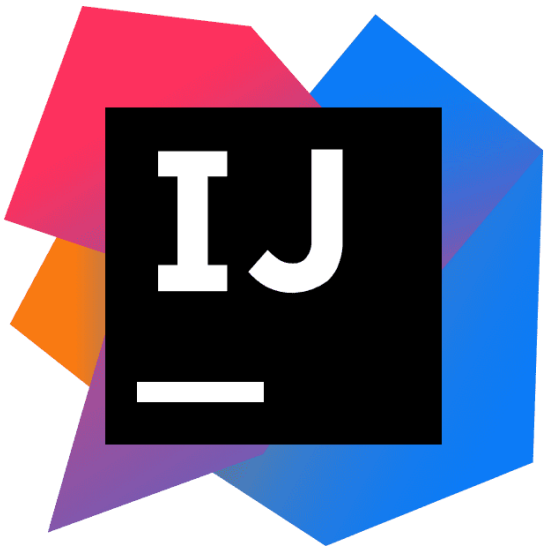

Figure 3.13: IntelliJ

As our backend development tool, we went ahead with IntelliJ. It is a Jet Brains-developed source editor that is compatible with most operating systems, much like Visual Studio Code. We did not have much experience with IntelliJ before, but it is a very simple and enjoyable editor to use. It has a fantastic smart completion feature that provides you with a list of the most important symbols for the current situation. IntelliJ has dedicated keyboard shortcuts for the most part, which helped us save time during the development stage. You are provided with a built-in terminal and can use command

line, bash, or PowerShell depending on the project. Another great feature of IntelliJ is its inline debugger. When debugging the code, IntelliJ IDEA displays variable values alongside their usages in the source code. Every time a variable's value varies, the IDE shows it in a different colour so you can see how the condition changes in the code. This came in very handy as we were checking unique features within our project.[19]

Errors and misunderstandings with the coding are likely to happen with large projects like this. With their fast fixes, IntelliJ provided us with a solution that was really successful. A lightbulb can appear if you are about to make an error and clicking on it will bring up a series of steps you may take to correct the mistake. When we were researching various IDEs, we discovered that IntelliJ has a separate Docker plugin, which was one of the main reasons we picked IntelliJ for backend. The plugin connects to a running Docker machine on your local network and handles its core services including images, containers, and most importantly Docker compose.

3.4 Cloud Hosting

We had to use two separate platforms to host our web application in the cloud. We were able to host our website and enable users to access it from anywhere, at any time, thanks to these technologies. These two technologies were Docker and AWS.

3.4.1 Docker

Figure 3.14: Docker

We have to run each part separately because our project uses several functionalities such as ReactJS, MongoDB, and Spring Boot. This became

a very tedious and time-consuming procedure as the project grew in size. We looked into this issue and discovered Docker. Docker is a popular open-source project written in Go and created by Dotcloud. It is essentially a container engine that creates containers on top of an operating system using Linux Kernel features such as namespaces and control groups.[20] This was the ideal solution to our dilemma because it saved us time from having to execute each function separately.

In order to use Docker, we had to first generate images for the backend and frontend, which were then uploaded and stored onto our Docker Hub account. In order to create the images, we had to run the following commands:

- docker build -t sagheergmit/frontendfyp:latest .
- docker build -f Dockerfile -t sagheergmit/final-year-project .

These commands generated images for the project's frontend and backend. We didn't need to build a mongo image because one is already available for public use by default. We then generated a new file on our local machine called docker-compose.yml that would use these images to execute the web application as a single file. This was ideal because we didn't have to run each project part individually.

Originally, we were running the project locally, but as it progressed, we began to use the cloud. We had to make a few adjustments to our code and remake the images which would be pushed up to Docker Hub. To run the application on cloud, we had to create a new docker-compose file on the aws instance. The images were pulled from our Docker Hub account using the command "docker-compose pull", this pulled all the required images and saved them to the instance. To run the images on the designated port we had to use the command "docker-compose up".[21]

We had never used Docker before this project, but after doing some research, we discovered that most businesses use it to create images and run them in the cloud. As soon, we will be applying for jobs in this industry, we decided it would be a great idea to have Docker covered. Docker has a number of benefits, the most important of which is that it speeds up the whole web hosting process, allowing you to devote more time to other activities. It has a quick and simple setup for cloud deployment, a secure and remote environment, it is very scalable, and has a rollback option if ever in need, among other features.

3.4.2 Amazon Web Services

Figure 3.15: Amazon Web Services

Amazon Web Services is a pay-as-you-go cloud computing network that allows customers to host web applications. Millions of people depend on it to fuel networks and applications. AWS enables developers to cut prices, improve agility, and innovate more quickly. Moving apps to the cloud becomes faster and more cost-effective as a result of this. This online portal for cloud computing provides a diverse set of tools and modules. Two of these platforms are Amazon Elastic Compute Cloud (EC2) and Elastic Beanstalk (EB). Customers may use these programs to build a virtual cluster of computers in the cloud that is always available to them.[22]

We looked at all of these services to see which will be the most appropriate for our project. We used EC2 to set up an instance on which we could run our web application. We had to first consider what kind of instance we wanted to create. Either a Windows or a Linux instance had to be chosen. We chose Linux over Windows because we believed the instance would run more smoothly on Linux. After selecting the instance type, we had to choose between the free and paying tiers. The free tier was our first option. The instance was set up and configured to run NodeJS and Docker Compose. To set these up, we needed to run a few commands[23]:

- sudo apt-get update

- sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
- curl -fsSL https://download.docker.com/linux/ubuntu/gpg — sudo apt-key add -
- sudo apt-get install docker-ce docker-ce-cli containerd.io
- apt-cache madison docker-ce
- sudo apt-get install docker-ce docker-ce-cli containerd.io
- sudo apt install docker.io
- sudo apt install docker-compose
- npm install nodejs

We were able to customize the instance using these commands. Then we attempted to run the docker compose file on the newly generated instance. The web application turned out to be far too large for the free tier instance. The instance was extremely slow, failing to even load the home page, much less process any requests.[24]

After that, we decided to start over and build a new instance. We chose the medium tier this time, which would cost us around 20c per hour. We chose a Linux instance once more and installed it with the same configurations as before. The instance was then used to build and run a Docker Compose file. We were able to navigate the website from the cloud after the Docker Compose file started running, without a hitch.

3.5 Version Control Manager

3.5.1 GitHub

GitHub is a collaborative code hosting site. It allows you and others to collaborate on projects from any place at any time. For Git commands, GitHub provides a graphical user interface (GUI). It is used as a version control manager, as previously mentioned. Java, Docker, Python are a handful of the languages that can be hosted on GitHub. Some of the features available to GitHub include, following other users, forking other people's repositories, subscribing to other's projects. Since it is widely used in the industry, GitHub is an excellent skill to acquire. Cloning other people's repositories is

Figure 3.16: GitHub

a fantastic feature of GitHub; it allows you to get another person's code and make improvements to it in your repository.[25]

These are some of the most commonly used GitHub commands when working on a project:

- Git add . - Adds all the files that are not already on the GitHub Repository.
- Git commit -m "First Commit" - Commit all the changes and give the commit a message.
- git push - Push all the changes to your repository on GitHub

GitHub Desktop

Along with all of this, GitHub has created GitHub Desktop, a software program that works in the same manner as their website. It's available as an interface which gives you easy access to GitHub repositories. We decided to focus on the desktop edition because it's easier to read and doesn't require the use of the command line for any inserting, committing, pushing, or pulling. It also enables users to clone any directory onto their machine and begin working on it right away. We may even inspect all of the modifications we made to the project before committing them using GitHub Desktop. If

Figure 3.17: GitHub Desktop

we feel that these modifications aren't what we want, we can roll back directly from GitHub Desktop, saving us the trouble of getting into the code and removing any changes we created. Many of these features are accessible via GitHub Desktop's user-friendly graphical user interface.[26]

3.6 Others

3.6.1 Jira Software

Figure 3.18: Jira Software

Atlassian's Jira is a project management platform that has a problem and error monitoring framework. It assists teams of all sorts in managing their

workload and staying on top of things. Jira can be used by developers to create specifications, handle test cases, and automate testing. It's ideal for agile growth. Jira has a number of quality assurance add-ons, including problem monitoring, Kanban forums, scrum boards, and progress reporting.[8]

From the beginning of our project, we used Jira to handle our specifications. We maintained meticulous records of everything we did, noting which challenges we encountered and how we resolved them. We divided the project into a number of smaller tasks. The task was then given a story point. The story point indicates how much time we can spend on each task; for example, if a task has a story point of 1, it should only take us a day to finish it, while a task with a story point of 3 should take us a week to complete. It also encouraged us to divide the work between ourselves so that we knew exactly what we needed to do, and it could help us keep a good workflow.

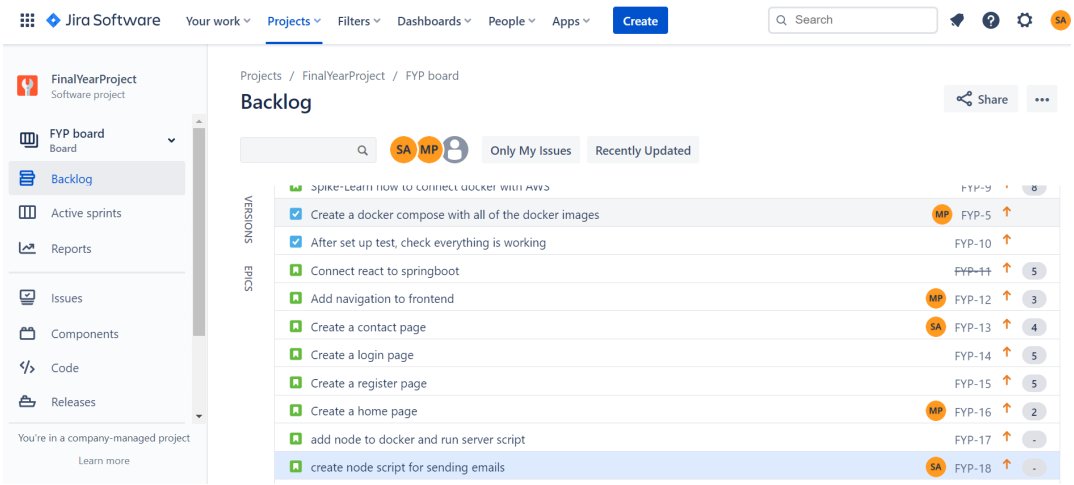

Figure 3.19: Jira Software

3.6.2 Postman

Figure 3.20: Postman

Postman is a collaboration platform for API development. We've been using Postman since the beginning of development. We initially used it to verify whether our mongo database was linked to our back end by sending Get and Post requests. The next step was to see how we could submit user information in JSON format to register a user and then use the same information to login. We gave Postman the link to our back end and sent a request to that link to see if we can get a response.[9]

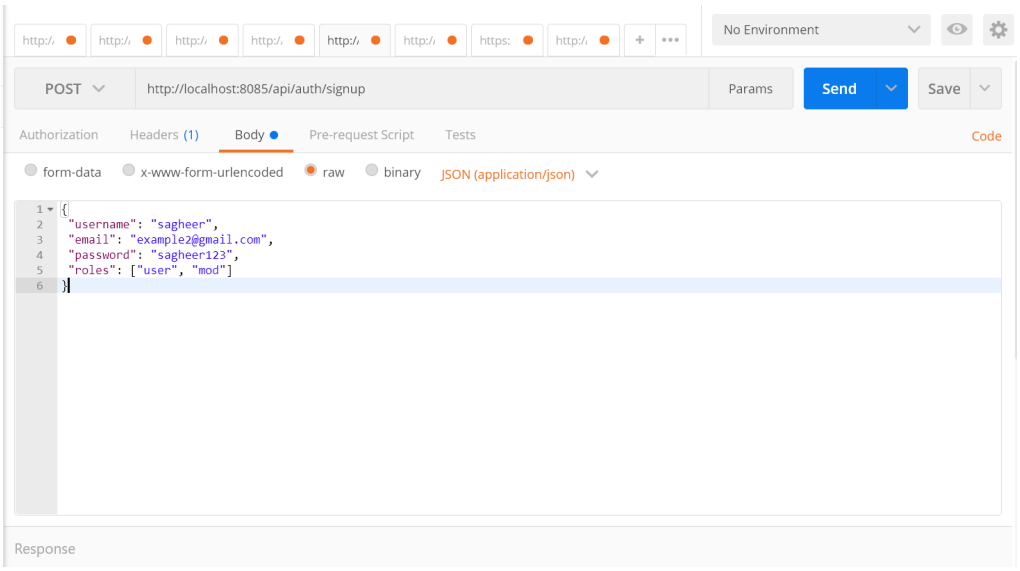

Figure 3.21: Postman Logs

Chapter 4

System Design

We'll go into the overall device architecture and configuration of our framework in detail in this chapter. We'll use visual aids and code fragments to help us imagine our reasoning and thought process. This chapter will be divided into four parts. Databases, backend, frontend, and cloud implementation will be included in this section. Our application's overall design is depicted in the diagram below.

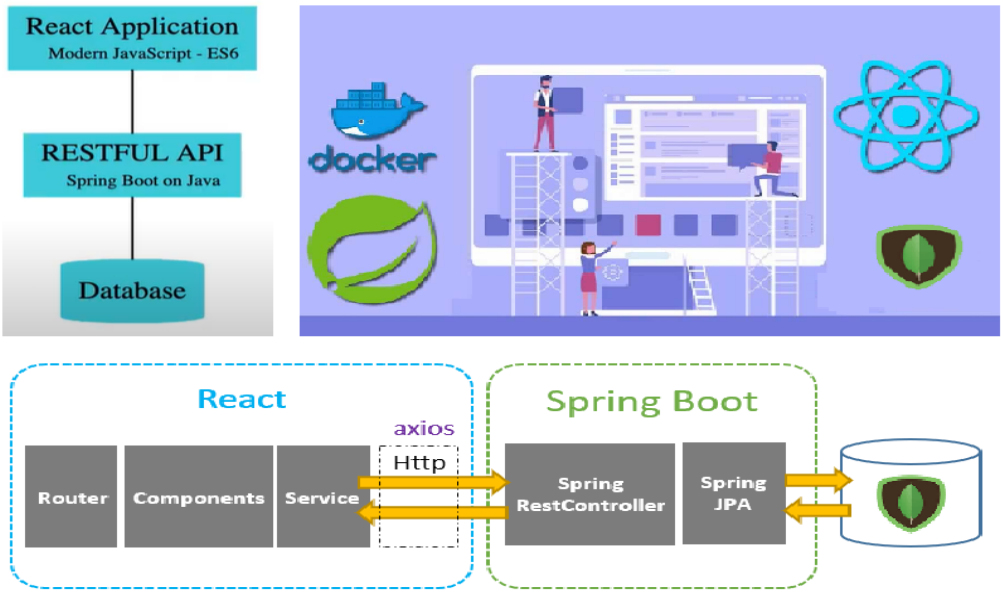

Figure 4.1: Project Schematic Diagram

4.1 Databases

This section refers to the overall data that has been stored in the databases. This data should be able to be created, read, updated, and deleted. In this project, we'll be using two databases, MongoDB and Firebase. One of the databases is used for storing users, while the other database is used to store data that comes from the various components of the web application.

4.1.1 MongoDB

The user information for signing in and registering was stored in MongoDB. Mongo is very adaptable, so we could set it up as we like. We first set up Mongo locally using the Mongo bash terminal. This was accomplished using Docker. We created a Mongo picture that we could run locally using the Docker Hub software. After downloading the app, we had to view the image and create a database inside it. To do so, we needed to run the commands below[27]:

- docker run -p 27017:27017 imageId
- docker exec -it mongo bash
- mongo
- use fypdb

These commands allowed us to populate the database with test data that the user can use when signing up and logging in. When the user tries to sign up or log onto the web application, the content is sent to the user repository class in the backend of the project. This class is connected to MongoDB and checks the user credentials to successfully register and log in the user.

When setting up Mongo, we ran into a few problems. We couldn't store or retrieve any data into or from the database because of the user repository. We spent a lot of time trying to figure out what was wrong, but we couldn't. While investigating the mistake, we discovered a program called Robo3T, which provided us with a proper GUI that enabled us to view the database in depth and see what was wrong. We discovered that the user roles were not specified in the database. To resolve this problem, we needed to run the following command:

- use fypdb

```java
package ie.gmit.sw.finalyearproject.repository;

import java.util.Optional;

import ie.gmit.sw.finalyearproject.model.User;
import org.springframework.data.mongodb.repository.MongoRepository;


public interface UserRepository extends MongoRepository<User, String> {
  Optional<User> findByUsername(String username);

  Boolean existsByUsername(String username);

  Boolean existsByEmail(String email);
}
```

Figure 4.2: Mongo User Repository

- db.roles.insertMany([name:"ROLE_USER" , name:"ROLE_MODERATOR" , name: "ROLE_ADMIN" ,])

The backend was successfully linked to the database after running this command. We made a note of the problem and solution so that we could correct it if the bugs resurfaced again when we set up the database for cloud deployment.

4.1.2 Firebase

We used Firebase to store data from the different components we included in our project. It is a Google-developed free database tool for developing mobile and web applications. We learned that there were a lot of interesting guides on firebase about how to set it up because it was our first time using it. The database was used to store user notes from the notepad section as well as posts from our forum page.

There was an alternative in Firebase to choose between the firestore database and the realtime database. Since the firestore database is Firebase's newest database for mobile app development, we wanted to use that in our project. It also has more powerful and quicker searches, as well as the ability to scale

further than a real-time database. If the database is set up, Firebase provides us with an API key and a few lines of code to incorporate into our project. After we imported everything, we ran into a problem where we could not initialize the database, which caused our web application to fail whilst it was running. On Stack Overflow, we discovered a workaround by a user named Valerii.[15] He suggested a different approach for the initialization and after reading that, we exported the database so it could be used elsewhere in our project.

After resolving this problem, all that remained was to run the project and verify that the database was operational. We tried posting to the website and using the notepad components, as well as checking to see if the database had been filled. The information had been safely saved to the archive.

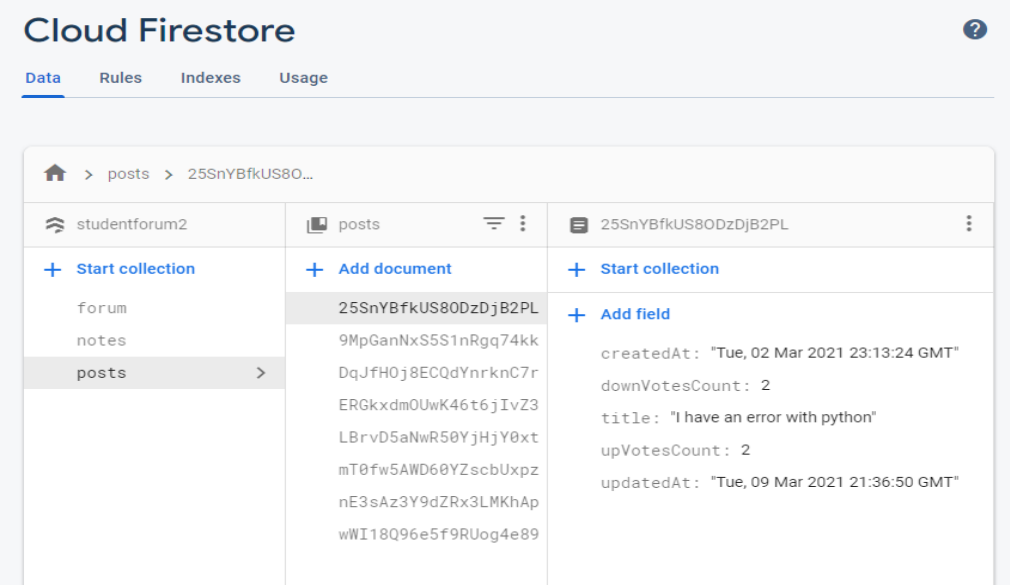

Figure 4.3: Firebase Cloud Firestore

4.2 Backend

In order for the user to save any details on to the database, we had to set up a backend. As mentioned previously we are using SpringBoot to create the backend element of the project. We have an authentication controller that checks for any incoming requests. If the controller receives a request from the frontend, it will decide to handle the request appropriately. The controller

has two main functions in the class to handle this request.

The first function is for allowing the user to register to our web application. This function requires four inputs, name, username, password, and an email address. The first step in registering a user is checking if the details the user has given, previously exist or not.

4.2.1 Register

```
@PostMapping("/signup")
public ResponseEntity<?> registerUser(@Valid @RequestBody SignupRequest signup)
  if (userRepository.existsByUsername(signUpRequest.getUsername())) {
     return ResponseEntity
       .badRequest()
       .body(new MessageResponse("Error: Username is already taken!"));
  }

   if (userRepository.existsByEmail(signUpRequest.getEmail())) {
     return ResponseEntity
       .badRequest()
       .body(new MessageResponse("Error: Email is already in use!"));
  }
```

The user credentials are sent to the user repository that is connected to Mongo. This class checks to see if either the username or email already exists. If either of the details exist, then the controller sends a message back to the user letting them know that either the username or email already exists.

```
  // Create new user's account
    User user = new User(signUpRequest.getUsername(),
            signUpRequest.getEmail(),
            encoder.encode(signUpRequest.getPassword()));

    Set<String> strRoles = signUpRequest.getRoles();
    Set<Role> roles = new HashSet<>();
```

If the user does not already exist, then a new user will be created with the provided credentials. The user's password is encoded using the standard hash algorithm to provide extra security. This is done so the password cannot be hacked or stolen. We used the standard hash algorithm as it cannot be reversed.

```
if (strRoles == null) {
     Role userRole = roleRepository.findByName(ERole.ROLE_USER)
       .orElseThrow(() -> new RuntimeException("Error: Role is not found."));
       roles.add(userRole);
    } else {
      strRoles.forEach(role -> {
        switch (role) {
          case "admin":
            Role adminRole = roleRepository.findByName(ERole.ROLE_ADMIN)
            .orElseThrow(() -> new RuntimeException("Error: Role is not found.")
            roles.add(adminRole);
            break;
    }
    user.setRoles(roles);
    userRepository.save(user);

    return ResponseEntity.ok(new MessageResponse("User registered!"));
```

The user is then given a role, as default the user is given the role of "user". Other roles include moderator and admin. The user is then saved onto the Mongo database.

4.2.2 Login

The second function in the controller class is the sign in function. This function is used to log the user into the web application. It is very similar to the sign-up function; in the sense it receives user credentials and checks to see if the user exists or not. The user is authenticated by comparing the details to the details saved in the user repository and Mongo. The controller generates a JWT token and sends that back to the user to confirm login. If the user does not receive a token, then the user receives a message saying, "incorrect details".

```
Authentication authentication = authenticationManager.authenticate(
 new UsernamePasswordAuthToken(request.getUser(), request.getPassword()));

 SecurityContextHolder.getContext().setAuthentication(authentication);
 String jwt = jwtUtils.generateJwtToken(authentication);

 UserDetailsImpl userDetails = (UserDetailsImpl) authentication.getPrincipal();
 List<String> roles = userDetails.getAuthorities().stream()
```

```
                .map(item -> item.getAuthority())
                .collect(Collectors.toList());

return new ResponseEntity<>(new JwtResponse(jwt,
                userDetails.getId(),
                userDetails.getUsername(),
                userDetails.getEmail(),
                roles), HttpStatus.ACCEPTED);
```

4.3 Frontend

We'll be discussing four related topics with our front end. We'll look at the structure of ReactJS and how it interacts with the backend and databases. We'll examine the application's various pages and see how they function, and then we'll dig a little further into how we used CSS to style our app.

4.3.1 ReactJS Structure

Firstly, lets take a look at the structure of our ReactJS app. This is in the src folder of the ReactJS application. This can be seen in the figure below.

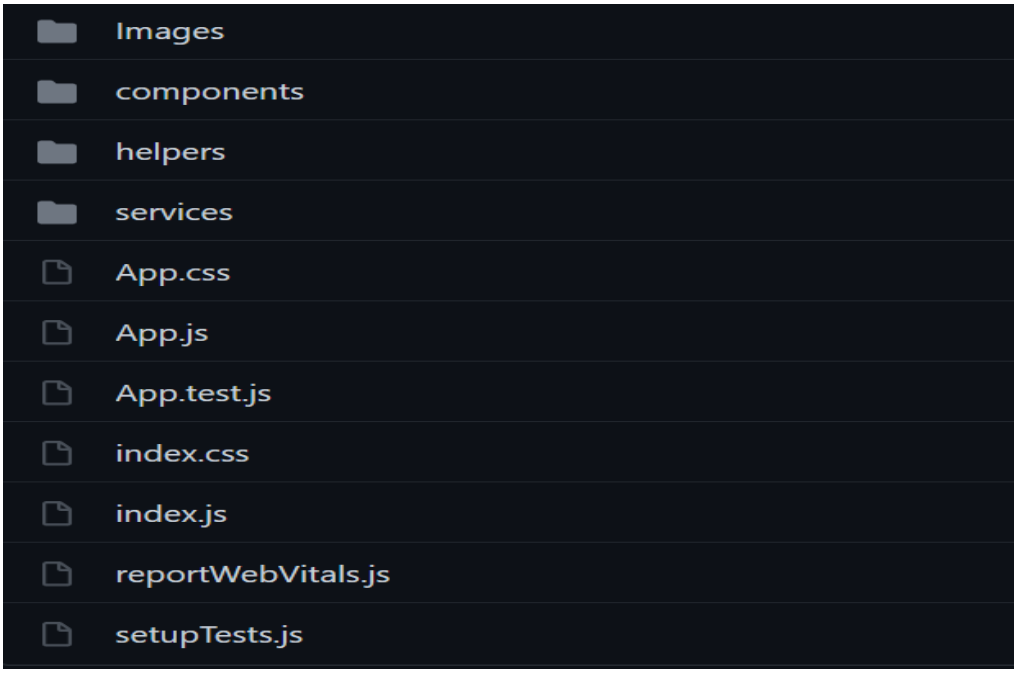

Figure 4.4: React App Structure

- images – this folder contains all the images that we used throughout our project.
- components – this folder contains all the different pages of the web application, from the contact page to the register page. We will discuss each of these pages in more details later in the dissertation.
- helpers – this folder contains code that is used for the notepad element of the code.
- Services – contains code that was initially used to test if the frontend was connected to the backend.
- App.js – this file contains all the routes that are available on the web application and is responsible for if a page should have a navbar and a footer.
- Index.js – this file calls the app.js class and loads in the pages so they can be viewed by the users.

4.3.2 Components

Each page in the web application has at least two different classes, one for the styling and another for the functionality. Our web application has the following pages:

- Home page
- About Us page
- Contact Us page
- Login page
- Register page
- Student Forum
- Student Notepad
- Sticky Notes
- Messenger application
- 404 page

4.3.3 Home Page

For our home page we wanted to take a minimalistic approach, but we also wanted the page to catch the user's attention. That is why we decided to put as little information on the home page as possible. We went for a video playing in the background, as it would be more appealing to the user. The home page also contains cards that outline the different components of the application. The web application is designed so that it can fit any screen size.

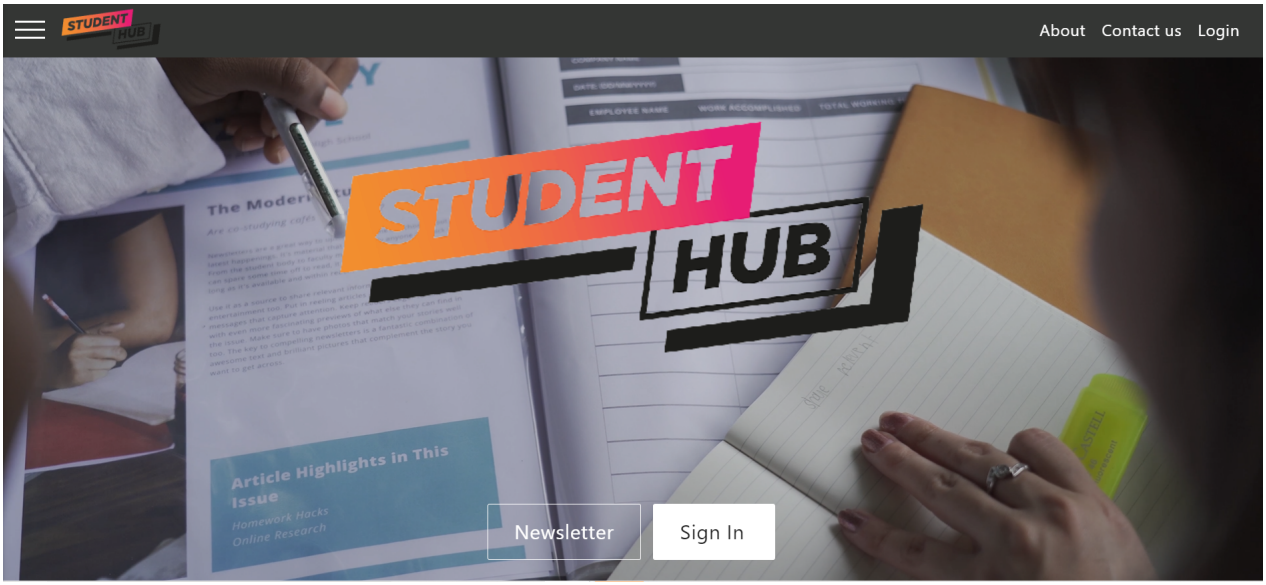

Figure 4.5: Student Hub Home Page

4.3.4 Navbar

We have a navbar in the web application that allows the user to navigate through the application. There are two types of navbars in our project. We have a standard navbar at the top of most pages and we have a side bar. The standard and side navbar contain only a few buttons at the start but after the user logs in the navbars update and contain all the pages of the application. Both of these navbars can be seen in the below figure. Once logged into the application, users can navigate freely. Alongside the pages, the user also has the option to logout from either navbar. The user will remain logged in even after they have closed their machines. This is so that the user does not need to sign in every time, which can become annoying.

The main aim of the sidebar was that when the user accesses the web application from their mobile phone, they will be able to easily navigate through the different components.

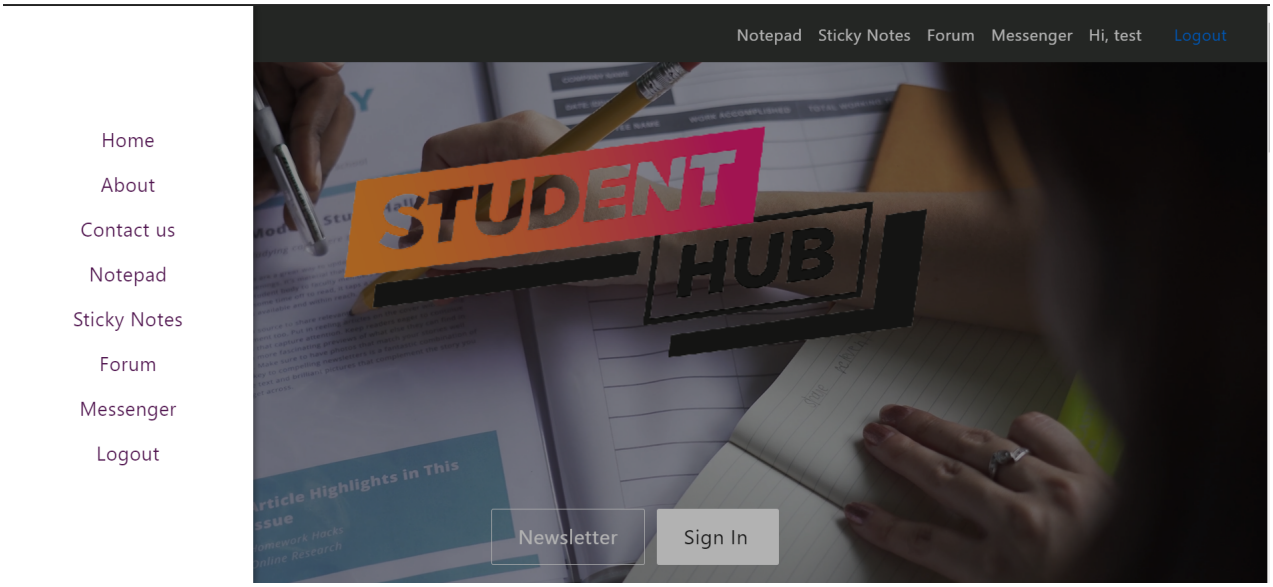

Figure 4.6: Student Hub Navbars

4.3.5 Footer

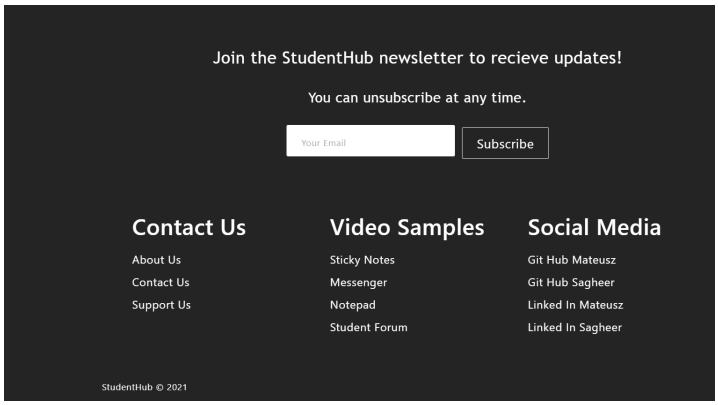

Figure 4.7: Student Hub Footer

The web application contains a footer that contains all the useful links to the website. These links range from our GitHub accounts to the different pages on the web application.

4.3.6 About Us Page

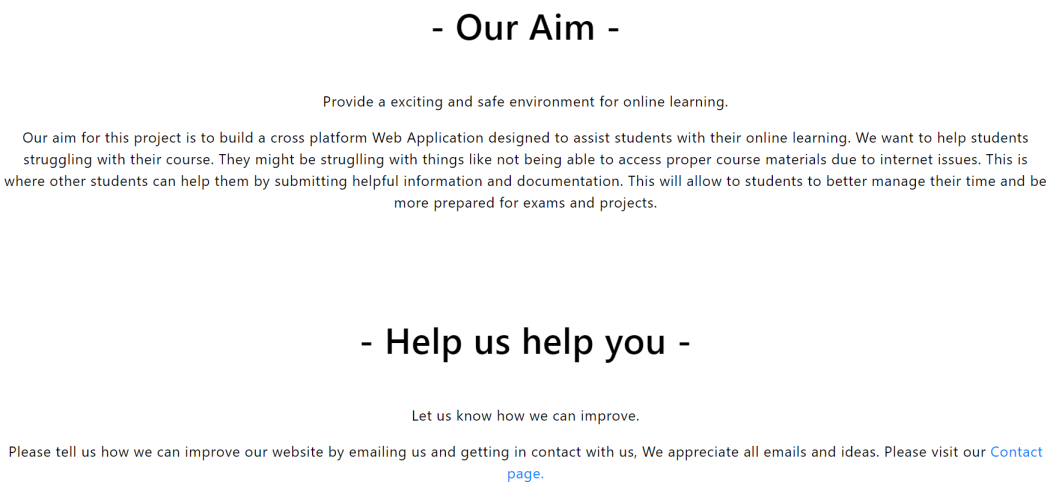

Figure 4.8: Student Hub About Us

We have a standard about us page, that gives the user more insight into our application and what we have planned for the future.

4.3.7 Contact Us Page

We set up a contact page because we wanted users to send us feedback on any issues they might be having or send us ideas on how to improve the website. We wanted user feedback to be one of the most important elements of the application. We set up a separate server for this part of the application. We created a NodeJS server, running on port 5000. The server was created to receive requests with some data that would be sent to our email address. When the user fills in the details for the contact page and presses the send button, a request is sent to the server. The server reads in these details and saves them into separate variables. The server then sends an email with those variables to our email address.

We used a dynamic object model while creating the contact page. This meant that the user had to first put in their name, after that the contact modal changes to allow the user to insert their email and upon inserting the email, the user can enter their message. If the user does not enter appropriate details, the modal will not change.

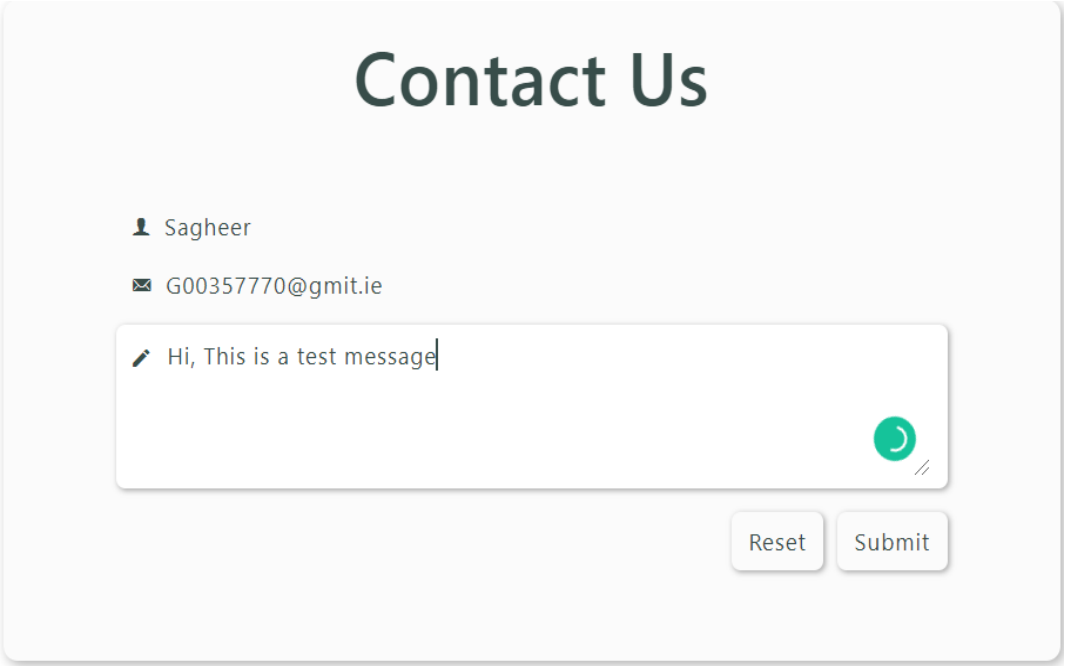

Figure 4.9: Student Hub Contact Us

4.3.8 Register Page

When the user enters the register page, they will be asked to enter in the required credentials. These credentials are their name, username, email and password. The user will be required to give a username longer than three characters and a password longer than six characters. This will allow the password to be more secure. When the user clicks the register button, a request will be sent to the backend to store the data. After registering, the user will be redirected to the login page to sign in.

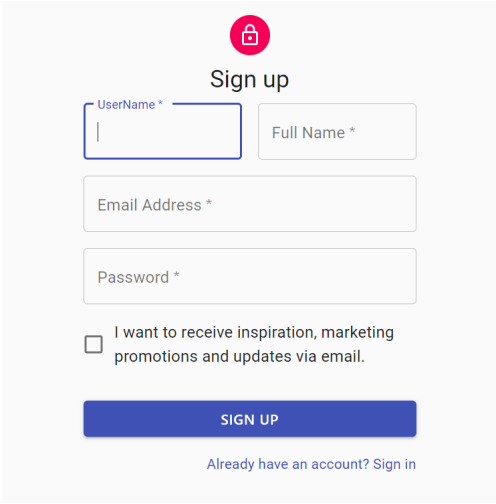

Figure 4.10: Student Hub Register Page

4.3.9 Login Page

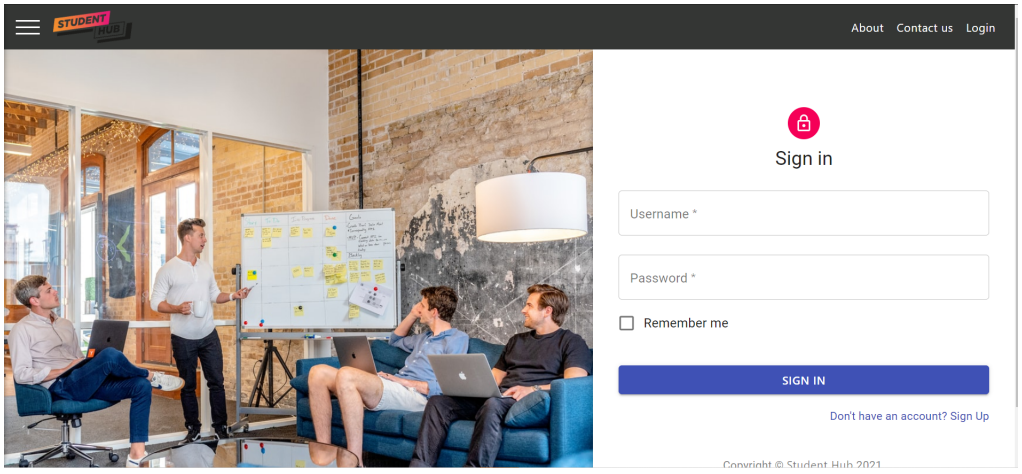

Figure 4.11: Student Hub Login Page

After the user has successfully registered onto the application, they will be required to login. If the user has an account, they can login using their credentials. They will be using their username and password to login. Once the user enters their details, a request is sent to the SpringBoot application to validate the user. Once the user is validated, the user will be taken to the home page. If the user is logged in, they will be able to access the different features of the web application. If the user is new, they will have to register in order to log in. The login page also requires the user to verify a Google

ReCAPTCHA, to ensure that a bot is not trying to access our web application.

We re also using an external API for generating random images in the background of the login page. There is a new image every time the user reloads the page.

4.3.10 Student Notepad

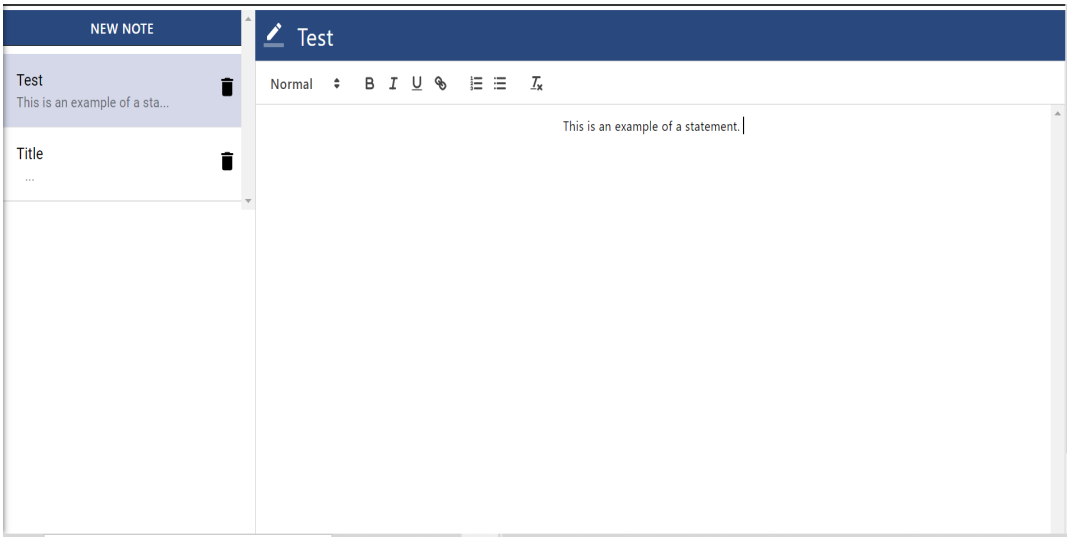

Figure 4.12: Student Hub Notepad

We wanted to create an application that would allow users to save big notes and paragraphs. We decided to create a notepad page on our web application. This component will allow users to write, edit and save these notes onto their account. These notes are then saved to firebase so that that the user can access these any time they like. The notepad has a couple features to allow the user to make changes to their writing. The user can change the title of the notepad at any given time. They can use different types of writing styles and fonts such as italic or bold. If the user does not require the document anymore, they can delete it.

4.3.11 Student Forum

Users also have access to a student forum, here students can discuss ideas and discuss different topics. Users can ask questions that they may be confused or lost on. They can even answer any questions that they may know

Figure 4.12: Student Hub Forum

the answers to. The forum is also saved to Firebase so that the notes stay on the web application even after the questions and topics have been discussed.

The forum also has a voting system, this is to see what the most popular topic or question is. Students can either up vote or down vote the post. This will determine the popularity of the post.

4.3.12 Sticky Notes

If the user wants to save notes but does not want to use the notepad feature, as the notes are small reminders. They can use the sticky notes of the feature. These sticky notes are meant to act as small notes or reminders for the user. They can be created and deleted instantly. They are saved to local storage and can persist in local storage even after logging out.

4.3.13 Messenger Page

We know how social media is so important for students nowadays, for talking to your friends and colleagues. We wanted to have a way for students to talk to each other, so we created a messenger feature. Students can talk to each other or even share documents such as a pdf and send images to each other. The user can have as many conversations as they like with other students and web application users. This was possible through an API known as Chat Engine. Chat Engine has a customizable UI that can be used to create

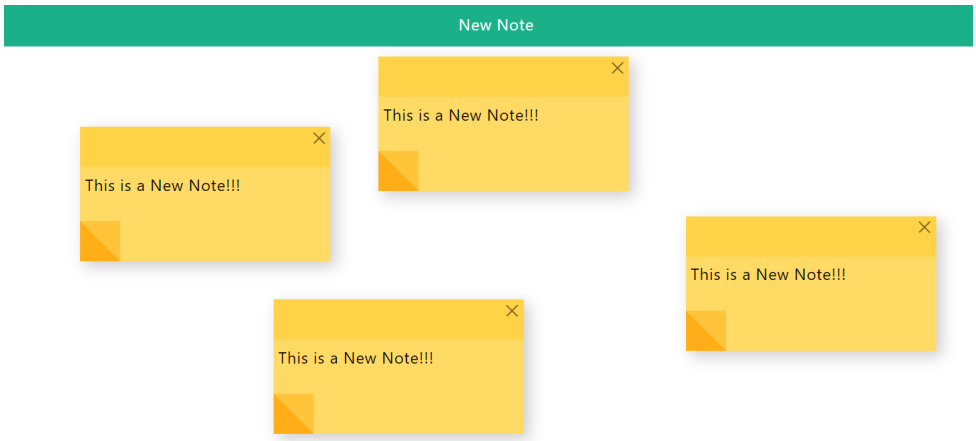

Figure 4.13: Student Hub Notes

a messenger page. We wanted to use an API because we had not used any before and wanted to expand our knowledge and skills on APIs.

4.4 Cloud Deployment

We decided to run our project on Amazon Web Services (AWS) so that it could be available from anywhere, on any device. If we did not host the project, then it would only run on a pc and that too after a long setup. In order to run the project on the cloud we had to create docker images for the frontend and backend. These images would then be pushed to our account on Docker Hub. We did not need to create an image for Mongo as Docker already provides an image for Mongo. As firebase uses an SDK, it does not require any images to run.

After creating and pushing the images, we had to create a file known as docker-compose.yml. This file is responsible for pulling these images from Docker Hub and running them on the cloud. But before we could run this file, we had to create an instance on AWS that would host our application. We went with Elastic Compute (EC2) as it works well with Docker. After creating the instance, we had to connect to it through a bash terminal. We decided to used Git Bash for this process. We ran the following line to connect to the instance:

- ssh -i ”studenthub2.pem” ubuntu@ec2-34-244-43-199.eu-west-1.compute

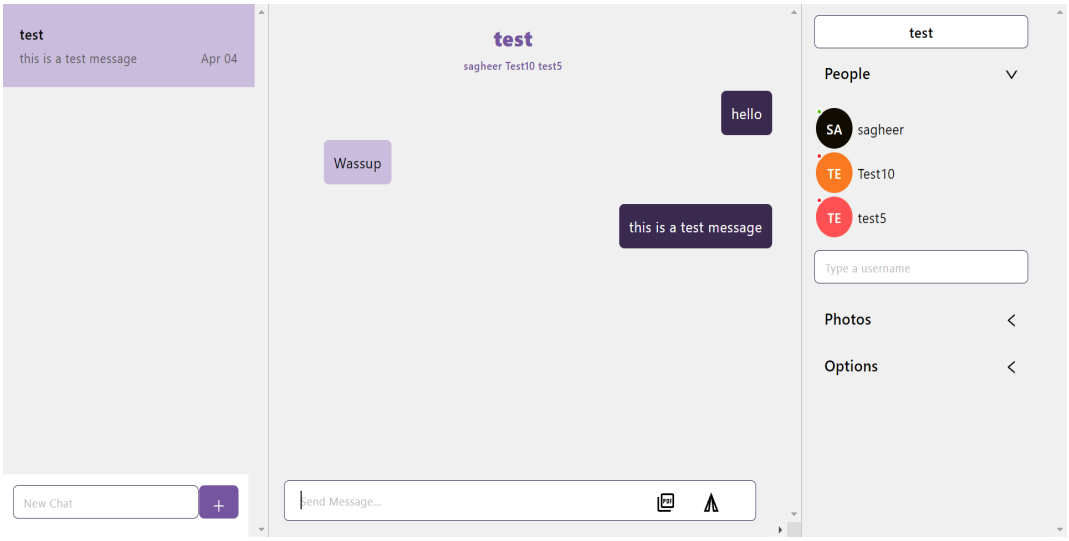

Figure 4.14: Student Hub Messenger

- .amazonaws.com

We recreated the docker-compose file on the instance and ran it to pull the images and start the web application. Once all these steps were completed, the application was running as intended.

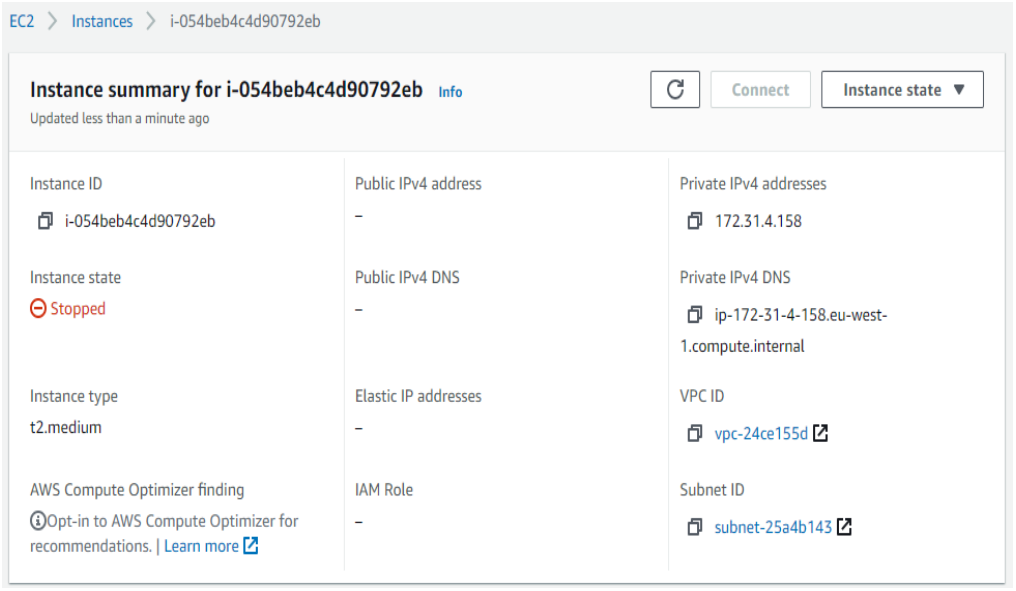

Figure 4.15: Amazon Web Services

Chapter 5

System Evaluation

The project's evaluation and testing will be discussed in Chapter 5. The testing process and steps taken during the development of specific parts of the project will be identified. The evaluation would also include any project flaws or limitations that we had to work with as a result of the original design. Finally, we will look for any enhancements or new functionality that may have been incorporated into the finished product.

5.1 Testing

We tested each part of the project during our entire development phase in order to conduct accurate testing. This ensured that all previously implemented components would work with the new ones. We used an agile approach during the project, which meant we focused on continuous testing while updates were made to the project. This was, in our view, a much better option than using the waterfall solution, given the project's size. Although this method took up a lot of our time, we felt it helped us to keep ahead of any problems that might become even bigger in the future.

To ensure that each component worked as expected and to reduce the number of bugs discovered in the code, we used black box testing. We were able to do this testing by asking our peers, colleagues, family, and supervisor to test the web application and flag any bugs/errors to us. We have questioned if they had any ideas for how to improve the web application. We incorporated some of the suggestions into the project. Such suggestions were, to add a ReCAPTCHA while signing into the website to prevent bots from logging in or add a footer at the bottom of the page with our social media links.

We opted to go for white box testing for our backend. We used a program called Postman to verify if our login and registration processes were working properly. We used Postman to send requests to the backend, to see if we could get some responses back. This was a time-consuming process because we had to submit new requests for each update in the project to ensure that nothing was broken. It did, however, allow us to verify that the backend was linked to the database. If the request failed to deliver, Postman provided us with accurate feedback. We could look at the code and figure out where the mistake was by using the feedback. This helped us to keep on board with everything and ensure that no major mistakes were made in the future.

When working with MongoDB, we double-checked that the data and information were properly stored in the database. We were able to test Mongo even more effectively and efficiently using Robot3T meaning there was no need to use the command line.

This type of testing, in our view, was very beneficial because it helped us to see if the web application was running as expected. It was difficult to obtain 100 percent testing coverage, as with any type of software testing. We concluded that this initiative resulted in a more robust method overall, and we were able to move on with our project after completing these tests.

5.2 Limitations

Appropriately, we found less limitations than anticipated with the technologies and programming paradigms we used for this project. Identifying any issues early in the project is critical for reducing the amount of time lost later on. This relates to our use of Agile and how it enabled us to keep ahead of any problems that could arise with our technology. These problems could be caused by a lack of time or even a lack of familiarity with the modern technologies.

When we talk about limitations, we do not have to speak of them as something negative. It provided us with very useful information about our project and how we developed it. It will allow us to improve on the project in the future and acquire expertise that we can apply in the upcoming years.

We encountered obstacles on our way when working on the project that blocked us to make progress. Initially, when implementing the ReCAPTCHA into our login page, it was giving us a problem as we were not able to prop-

erly implement it into our project. We could not get the ReCAPTCHA to work on localhost due to ReCAPTCHA only being able to run on a proper website. In order to fix this problem, we changed the ReCAPTCHA code to Google ReCAPTCHA, which could work on localhost. We had a problem with our contact page as well, where the contact page would not take in details from the user. We generated a class for each input box and imported these classes into a bigger class called contact.jsx class. By making smaller classes and calling them into a separate bigger class, we could interpret the code more clearly and fix any errors. This helped us to solve the problem. Then, we had to use the POST method to email all of the data to our Gmail account, which we had set up specifically for this project. We maintained track of the code we wrote so that we could reuse it in the future if necessary. When working on the newsletter, this was the case for the POST and GET approaches. This saved us a lot of time and allowed us to concentrate on other things.

The chat application, which allows users to communicate with each other from all around the world, is one of the project's key features. Initially, we relied solely on a ChatEngine REST API to perform all of our tasks. This was a limitation because implementing the API did not require much work. We wanted to challenge ourselves and build our own chat feed where people can talk to each other and only use the API to handle people and groups. It took a long time, but we eventually overcame the challenge and came up with a solution. We created our own classes that implemented the API and allowed us to use our own code. Another project limitation is that it can be rather difficult for someone who clones the repository, to run the project. It is a big project, and you will need to run several technologies at the same time to get it started locally. Since we were unable to introduce a profile page for the user to alter his information, one of our main limitations was time management.

We wanted to make the app available to consumers as a smartphone app. This was attempted using Android Studio and Ionic Capacitor. However, we ran into a slew of mistakes that prevented us from progressing. These platforms were unable to convert the application into an app because it was too big. We were unable to pursue any other options due to a lack of time.

5.3 Results vs Objectives

When we contrast the end result of the web application to our initial goals, which were stated in Chapter 1 of this dissertation, we concluded that we have met the majority of the criteria we set for ourselves. We haven't confirmed whether or not this application will benefit students, however we hope it will. Since it will be used by students, we also wanted the mobile interface to be as user-friendly as possible. The goal, we think, has been met. To achieve this, we looked into various student applications, especially LearnOnline, in order to gain a better understanding of a high-end website.

Chapter 6

Conclusion

This was one of the most difficult projects we've worked on in the last four years, but we enjoyed the task of developing this web application. We've learned a variety of skills, including teamwork, organization, and time management. We were also able to improve our skills as a result of the project, allowing us to work on larger projects like this in the future. Throughout the last two semesters, we have enjoyed studying new languages and technology, which was possible because of this project.

We, like most people, ran into issues that slowed us down, which was stressful at times. However, this did not deter us, rather, it motivated us to try even harder to solve the problems. Overall, the project runs efficiently now that we've seen our friends and peers inspect the finished product, and we believe we've accomplished the majority of the goals we set for ourselves.

6.1 Objectives

Our main goal when we started the project was to make a web application that would help students with their online learning and make it more pleasant. We're pleased to report that our web application has accomplished this.

The things we have accomplished are:

- Created a web application using SpringBoot and ReactJS
- Web application contains the necessary components for a website i.e. About us, Contact page, Footer and a Navbar.
- Users can email us

- Users can use the navbar and sidebar to navigate through the application
- Users can register and create a new account
- Users can safely log in into their account
- Users need to verify a ReCAPTCHA in order to log in
- Users can successfully log out of the application
- Logged in users can access the notepad components and perform CRUD operations on it.
- Logged in users can access the student forum and post queries and questions.
- Logged in users can access the messenger component and talk to their friends and other students on the web app.
- Logged in users can create sticky notes as small reminders of things to do
- Users are taken to an error page if they input the wrong URL
- The application runs successfully on AWS cloud.
- The application is mobile ready, as the web app is designed to fit any screen size

When we equate the activities completed to the necessary goals, we can see that we have accomplished almost all of them. We wanted to convert the web interface to a smartphone app but were unable to do so due to a lack of time. Overall, we are very pleased with the final product.

During this global pandemic, we think the app would be extremely beneficial to students. They will be able to relax and not have to think about missed classes due to poor internet. As they can ask other students for notes and resources through messaging each other privately.

6.2 What we learned

After completing the final project over the span of two semesters, we have gained a number of skills in different technologies and frameworks. From developing a SpringBoot and ReactJS application, to using technologies like Postman to test different features of the project. We also gained knowledge into cloud deployment and hosting. This project gave us an opportunity to work with technologies that we didn't even know existed such as Robo3T and Docker.

We had never written a paper as long as this before, but writing a dissertation was a new experience for both of us. We had a basic understanding of research papers thanks to one of our modules, Research Methods. We discovered that the dissertation allowed us to thoroughly clarify our project and thought process in a more comprehensive way than the project alone. We've been advised for the past few years that proper documentation is critical when it comes to writing software. This dissertation demonstrated the importance of documentation and explaining our code.

6.3 Future Development

We're fully committed to make this app the best it can be. We plan to add a few more elements to this app, such as a user profile page and a timetable page, to make it even more enjoyable.

Our next step would be to launch the app onto both Google Play Store and Apple App Store. This would allow users to have the application on demand. If the mobile app is successful on these platforms, we would like to launch it on other stores such as the Microsoft Store. We initially created the web app so that it can be mobile compatible, but we struggled to convert it onto a mobile app due to its size. The application was too large, and we could not find a way to convert it to a mobile app in time.

6.4 Business development

We'd like to contact GMIT (Galway Mayo Institute of Galway) and NUIG (National University of Galway) to see if they would like to use any of our features in their student portals, as this project is primarily aimed at students in Universities and Institutions. If they intend to go through with the plan,

we'll be able to help these institutes improve and update their student portal and receive feedback directly from students.

Chapter 7

Appendix

7.1 Project Source Code Link

https://github.com/sagheerahmadGmit/FinalYearProject

Bibliography

[1] *MongoDB Features*
https://www.mongodb.com/what-is-mongodb/features

[2] *What is NoSQL?*
https://www.mongodb.com/nosql-explained

[3] *Why you should be using JSON vs XML*
https://blog.cloud-elements.com/using-json-over-xml

[4] *Google Firebase*
https://firebase.google.com/

[5] *What is Firebase?*
https://www.educative.io/edpresso/what-is-firebase

[6] *Firebase Database*
https://firebase.google.com/docs/database

[7] *How To Connect To Your MongoDB Deployments Using Robo 3T GUI*
https://scalegrid.io/blog/how-to-connect-your-mongodb-deployments-to-robo-3t-gui-at-scalegrid/

[8] *What is Jira used for?*
https://www.atlassian.com/software/jira/guides/use-cases/what-is-jira-used-for

[9] *Postman*
https://www.postman.com/

[10] *Spring Boot*
https://spring.io/projects/spring-boot

[11] *What is Java technology?*
https://tinyurl.com/3nveduve

[12] *What is ReactJS: Introduction To React and Its Features*
https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs

[13] *Why Sass?*
https://alistapart.com/article/why-sass/

[14] *HTML CSS*
https://www.w3.org/standards/webdesign/htmlcss

[15] *How to use firebase auth and Cloud Firestore from different components as a single firebase App*
https://stackoverflow.com/questions/57234932/how-to-use-firebase-auth-and-cloud-firestore-from-different-components-as-a-sing

[16] *What is the Difference Between an API and an SDK?*
https://nordicapis.com/what-is-the-difference-between-an-api-and-an-sdk/

[17] *What Is an IDE?*
https://www.codecademy.com/articles/what-is-an-ide

[18] *Visual Studio Code*
https://code.visualstudio.com/

[19] *IntelliJ*
https://www.jetbrains.com/idea/

[20] *What is Docker?*
https://www.ibm.com/cloud/learn/docker

[21] *Overview of Docker Compose*
https://docs.docker.com/compose/

[22] *What is AWS?*
https://aws.amazon.com/what-is-aws/

[23] *Deploy-Docker-Container-on-AWS*
https://github.com/soumilshah1995/Deploy-Docker-Container-on-AWS

[24] *What exactly is Node.js?*
https://www.freecodecamp.org/news/what-exactly-is-node-js-ae36e97449f5/

[25] *What Is GitHub, and What Is It Used For?*
https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/

[26] *An Introduction to Version Control Using GitHub Desktop*
https://programminghistorian.org/en/lessons/retired/getting-started-with-github-desktop

[27] *How To Run MongoDB as a Docker Container*
https://www.bmc.com/blogs/mongodb-docker-container/

[28] *What is JavaScript?*
https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

[29] *What is the difference between CSS and SCSS ?*
https://www.geeksforgeeks.org/what-is-the-difference-between-css-and-scss/