Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

ATU Medical Website - Booking Appointments Made Easy

By
Faustas Tamulis

for
Joseph Corr

April 24, 2023

Minor Dissertation

Department of Computer Science & Applied Physics,
School of Science & Computing,
Atlantic Technological University (ATU), Galway.

B.Sc. (Hons) in Software Development

Contents

List of Figures

Chapter 1

Introduction

My initial plan for my capstone project was to build a web application, but I wasn't sure where to take it from there. I considered other concepts, including a blog website or perhaps a business website for purchasing and selling goods. One week when I was coming up with ideas, I became quite ill and realised I needed to see a doctor. When I called my doctor's office, they took forever to schedule an appointment and kept putting me on wait. So, as I was brainstorming ideas, I made the decision to create a web application that facilitates scheduling appointments. Instead of having to call and wait a long time for an appointment, this website enables users to register, login, and schedule appointments anytime they want. Because we have technology at our fingertips, initiatives like this should be taken to simplify our daily lives. The main goal of this project is to do research on creating online applications to help patients in their daily life.

1.1 Developing the Design

It was difficult to choose the language and framework I wanted to utilise to accomplish this concept at the beginning of the project. There were many options available, including NextJS, Python, and Java. Because I had used these languages and frameworks before and wanted to challenge myself with something new and different, I made the decision not to utilise them. ReactJS is the most popular framework that many organisations utilise, I discovered when doing research on various businesses. I believed that working on this project with this framework would provide me with a fantastic opportunity to learn more about it and advance my web development abilities. The front end of the web application for this project is built using the ReactJS framework, and the back end is built using NodeJS. Both the NodeJS framework and the ReactJS framework employ Javascript as their core language. I had to conduct extensive studies and practise javascript because I am not the greatest at it. Because MongoDB is one of the most well-liked databases in the business, I chose to use it as the database for this project. Regarding cloud services, I also had a variety of possibilities. A few of the more well-known ones include Google Cloud, Amazon Web Services, and Microsoft Azure. Since this is the industry standard, I choose to use DockerHub and Amazon Web Services (AWS) for this section.

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

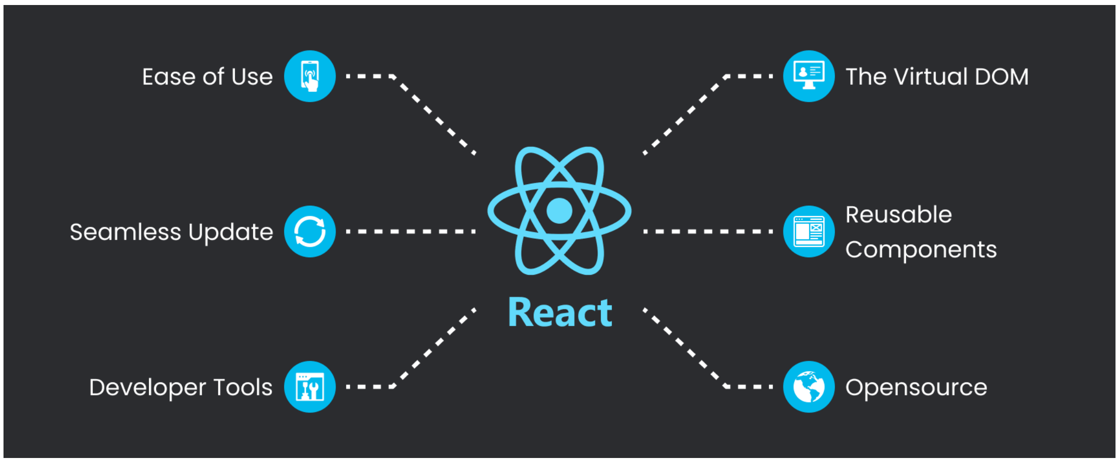


Figure 1.1: React Framework

1.2 Final Design

Medical ATU, is a web application project that involves NodeJS, ExpressJS, ReactJS, MongoDB 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. The user will have access to several services to help get through the process of making appointments 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 users will also be able to contact me directly regarding any queries that they might have through the live contact page.

1.3 Objectives

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

- The first objective was to create a NodeJS and ReactJS application that could talk to each other and save details to the Mongo database. I also wanted to have a basic conventional web application which had to be set up for example the home page, navbar, footer, contact page, registration page and login page.
- With the initial web pages setup and configured on the web app, I want to work with the most common pages of the web application such as the contact page and an about us page. In order to achieve this I will have to connect the contact page with a client side technology called EmailJS which works in react to allow the user to send an email directly to me.
- Subsequently, I will create pages for the user authentication. Once I have designed the register and login pages, I will connect them to the back end where the data will be saved into the database. The password will be hashed using a hashing algorithm provided by NodeJS called BcryptJS which will give extra security. I 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.

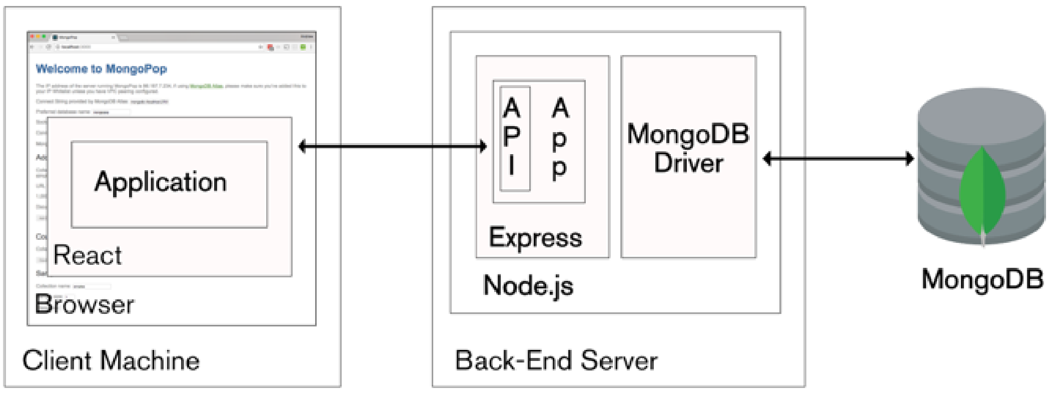


Figure 1.2: Schematic Diagram

- After the user has successfully logged in, they will be able to access a number of features such as applying to be a doctor, booking an appointment, managing the doctor profile, contacting for help and viewing their appointments. I will do this so that the front end will communicate with the back end to get all the data needed to make this happen.

- Lastly, the next step will be to deploy the project on to the cloud. This will be completed through the use of a Docker file and Docker-compose. The docker files will then be run on an Instance that I will create in AWS.

1.4 Overview

As this the final year project, I knew that I would have to challenge myself to work on a project that will allow for independent learning and meet the learning outcomes of the module. I wanted to make sure I learned something new through each stage of the development cycle. This dissertation will be laid into a number of different chapters, these chapters will go more in detail and explain the thought process and the different aspects of the project. This section contains a small description of each of the chapters that I will be covering.

1.5 Methodology

In this chapter of the dissertation I will be focusing on the steps that I took throughout the development stage to make sure I had a successful project. I will discuss the reasons for various technologies such as NodeJS and ReactJS and why I decided to use them and will talk about the various problems I encountered during the initial set up and development stages of the project. I mainly used agile methodology and I will provide a digram where needed to show off the methodlogy used in the final year project.

1.6 Technology Review

This chapter will be looking at all the different technologies that I encountered during the research. I will go into further detail on how set up various technologies for example Docker and Postman and their key features, 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 the code and images will be provided when explaining the components to give a better understanding of the project. Each component will be discussed in great detail and I will explain how each part of the project contributes to the final product.

1.8 System Evaluation

This section of the project will discuss how the project performs, how scalable it is and its robustness. I will also mention any limitations that I encountered such as, latency issues, cloud issues and other errors.

1.9 Conclusion

This chapter will be going over the goals and challenges that I set myself with. It will also cover if I accomplished these challenges and goals. The end result of the project will be discussed alongside all the difficulties that I have experienced in the development cycle.

1.10 Project Links

Finally the link to the Github Repository:

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

Chapter 2

Methodology

As previously said, this section focuses on the various methodologies used to deter the 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 of time. I can also write simpler, more readable code by planning ahead. If this project is not properly prepared ahead of time, 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

I defined the project's scope during the initial planning process. I concluded that Agile would be the best methodology to use. I investigated other methodologies, such as Waterfall, but after comparing them to Agile, I opted to stick with Agile. I was able to divide the project into multiple stages, thanks to the Agile approach. It allowed to make incremental improvements to the project at each point. An Agile approach allows for greater flexibility in project planning and execution. It can also be used to modify the project's various elements as its progressed. After discussing the different methodologies, I had to decide on the technologies to use. I researched into several technologies and there was a handful of good technologies to use. The options were endless but after making the decision I decided to use MongoDB, Express, ReactJS and NodeJS which were the main technologies in developing this web application. In addition to the technologies I had to use to chose which integrated development environment (IDE) to use. I decided to use Visual Studio Code for the frontend and backend. Visual Studio Code has a number of extensions which help with programming with the technologies I decided to use. I've never used these extensions before so it was a learning experience to use them for the first time and do some research on them. These extensions aided me in developing the project.

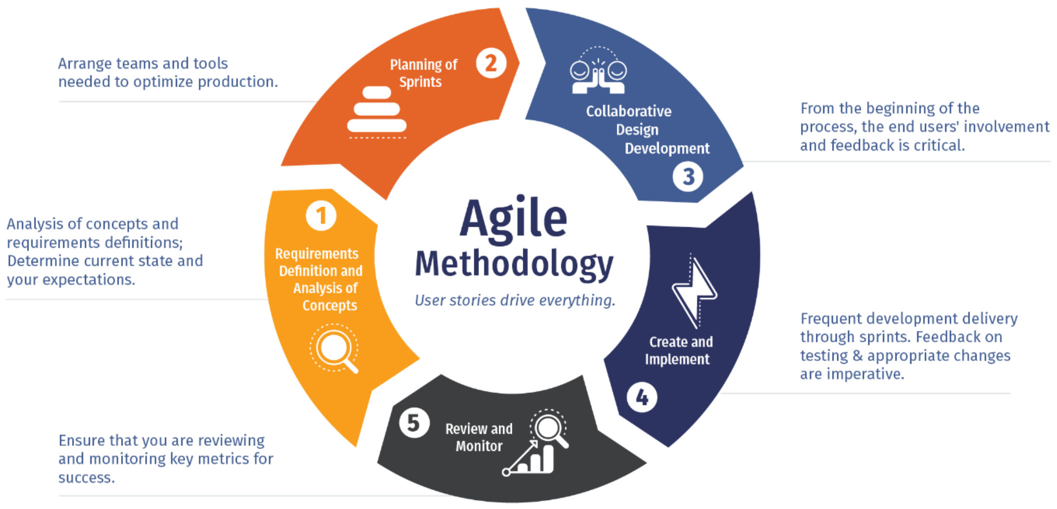

Agile Methodology
User stories drive everything.
1 Requirements Definition and Analysis of Concepts
Analysis of concepts and requirements definitions; Determine current state and your expectations.
2 Planning of Sprints
Arrange teams and tools needed to optimize production.
3 Collaborative Design Development
From the beginning of the process, the end users' involvement and feedback is critical.
4 Create and Implement
Frequent development delivery through sprints. Feedback on testing & appropriate changes are imperative.
5 Review and Monitor
Ensure that you are reviewing and monitoring key metrics for success.

Figure 2.1: Agile Methodology

I tried using a few diagrams during this process to help me better visualize the work that will needed to be done. The diagrams looked a lot like the schematic diagram described earlier. Since this was a single person project I decided to split up the project into a to do list, this helped me develop the project because as I got things done I could tick them off so I made sure to complete the tasks and not forget any details of the project. Since this is the final year I had a lot of work to do in other assignments and modules so planning and time management was a key factor to achieve a successful project.

2.2 Requirements Analysis

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

- To begin, I must have a functioning web application that operates correctly and enables the user to perform the function for which it was designed.
- Secondly, I must develop a web pages for the home page, about us page and contact page which lets the user browse and see what this web application is about and contact me personally using his gmail.
- Thirdly, provide a completely function and stable register and login framework that saves the user data in a database. To provide the necessary encryption for the password it had to be hashed so the user data is protected.
- Next, the user should then be able to log in and book appointments, doctors can register to become a doctor for the website which is then approved by the admin, users can access the appointments. The ability to log out safely and securely.
- After that, I would like to have the project running through the docker hub and AWS cloud service.

2.3 Meetings

Since this was a single person project, I had to organize and arrange the tasks to be complete in a timely manner while participating in other classes and assignments. I had decided to work on the project on the days where I didn't have any classes or assignments. This was very complicated at times because there was loads of assignments to do in final year. Me and the supervisor attempted to contact each other but I ended up never having enough time to meet up and should of planned my tasks better. In the future I will try to use my time better and contact my supervisor more often as at times I was stuck in the initial development cycle where I wasn't sure what to do for the final year project.

2.4 Development

I decided to divide the tasks into smaller tasks and cross them off when I completed them. I did some research into a way of making this method a lot easier and came across Atlassian Jira. This is a useful tool that can be used for projects that I have used before but I found since this was a single person project, it didn't work well with I was trying to achieve but when it comes to group projects, Jira is a very useful tool because it allows you create a project and set tasks for people to complete in the project and as the tasks get complete you can mark them off in the software. I made sure to make as many Github commits as possible so I didn't lose any progress when developing the web application.

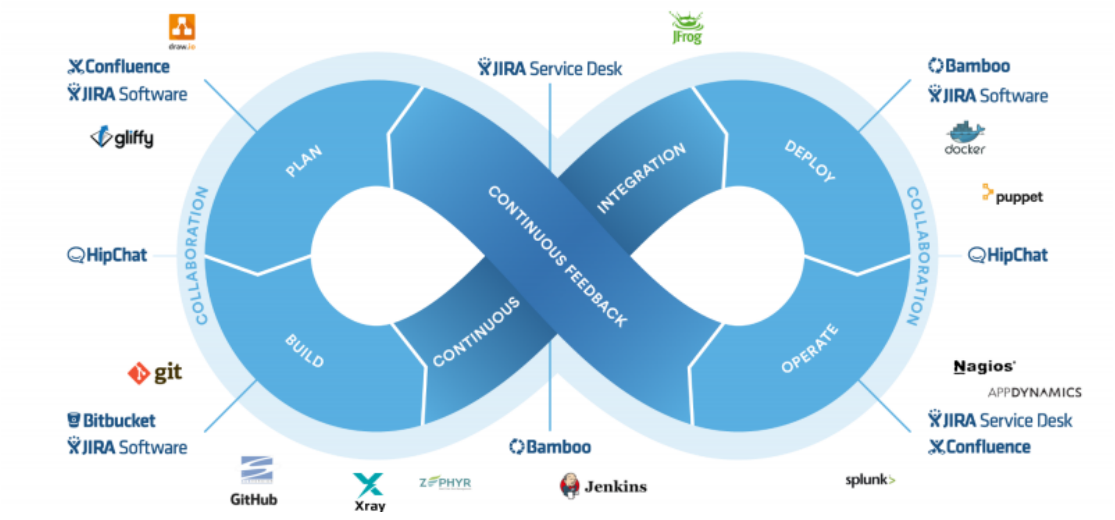


Figure 2.2: Atlassian Jira

2.5 Testing and Validation

Testing was a critical component in this project, as well as software in general. Testing saved 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, I used Postman and React Dev Tools, among with other tools. I carried out black box testing on the project by allowing other people to register and access the website in order to assess the various functionality and components. It allowed me to test if there was any faults in the application and listen to their feedback to see if there was any other ways it could be improved. Black-Box testing is a form of software testing that looks at an application's capabilities without looking inside the internal workings or mechanisms of the application. It has no understanding of the website's back end or front end and has no idea how the code works. This was extremely beneficial because it allowed me to receive direct feedback from different users and students from college. It also allowed me the opportunity to ask them if there is anything I could do to improve and change things they thought they didn't like. This feedback was very valuable and recorded and I proceeded to make the appropriate and necessary improvements to the web application. 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.

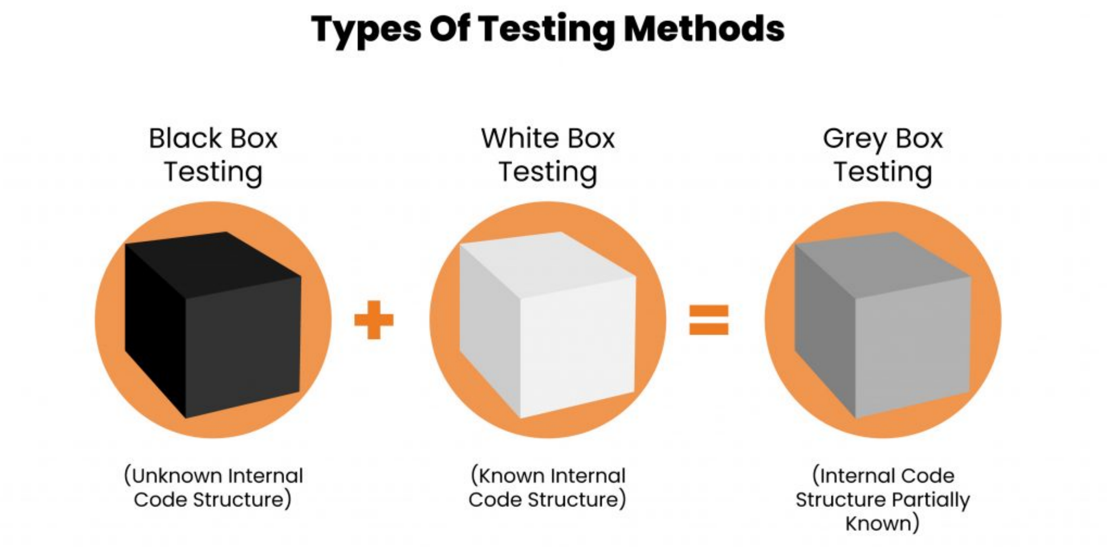

Types Of Testing Methods
Black Box Testing
White Box Testing
Grey Box Testing
(Unknown Internal Code Structure)
(Known Internal Code Structure)
(Internal Code Structure Partially Known)

Figure 2.3: Blackbox Testing

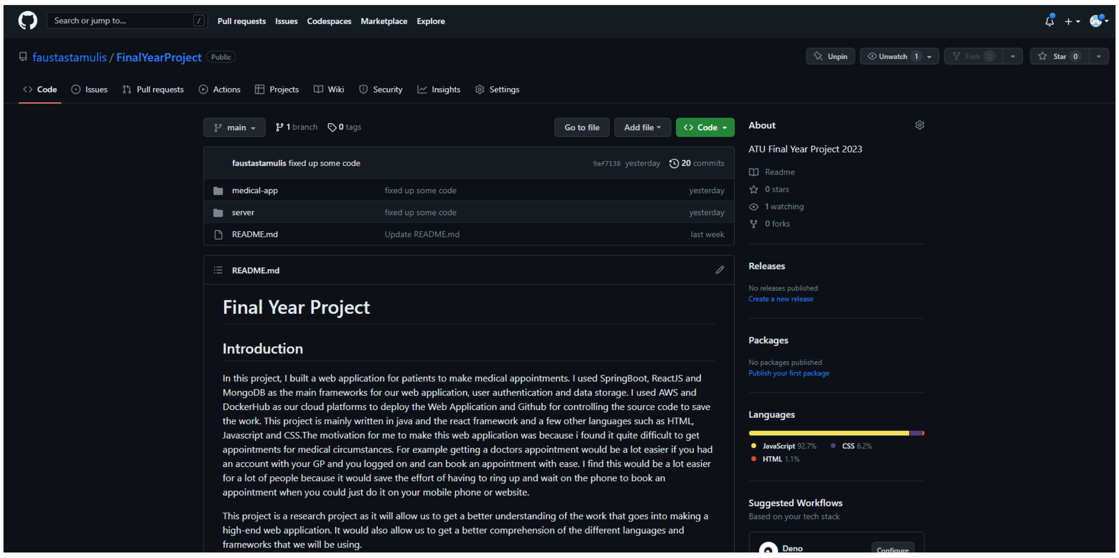
2.6 Problems Encountered

I ran into a lot of issues during the stages of development. Firstly, time management was a big issue of mine trying to find the time to get the project done was very tough, the data and time not working in the appointments section also setting up the Docker, Github failing to push code into the repository. I had to look at these problems and try and come up with solutions. Many of the problems I faced I looked at them as an opportunity to learn from them. I made notes of these problems and in the future I will know hot to overcome these problems.

2.7 Version Control Manager

I wanted to select a version control manager before I began working on the actual code of the project. There was many great options to choose from including

Bitbucket, Atlassian has its own control repository site, AWS has its own repository hosting site also. I concluded that using Github, which is the more well-known and widely used in the industry than any other version of these control managers. I had been using Github for my assignments and projects since the first year of college, so it made my decision easier. I stuck to the control manager which I was most comfortable using. Github has a lot features that I could take advantage of, the most important of which is the abilty to work on various branches to finish different parts of the project. Github makes the assembly of the project very presentable in an interview also. I tried making the repository as user friendly as possible and made sure to keep nice and tidy so I could reuse it in a job interview. It also allowed me to keep track of our issues in the repository. If I made an error it was also very easy to go back to the original code without the error. Github has the feature to revert to a previous commit which meant it was very easy to return to the original code and also meant that it was very important to commit frequently so the repository doesn't fall to far behind which could mean loss of progress. This also insured that none of the code could get lost or worst case scenario corrupted. Since Github is so widely used there's extensions for Github in most of the software we use today so it made working on the project quite easy and made committing to the repository very easy. I decided to make a lot of small commits but when I was stuck for time, I made a few big commits. It also shows the progress of the development of web application. In my view, Github was by far the best choice for the version control manager.



faustastamulis fixed up some code 9af7138 yesterday 20 commits

medical-app	fixed up some code	yesterday
server	fixed up some code	yesterday
README.md	Update README.md	last week

README.md

Final Year Project

Introduction

In this project, I built a web application for patients to make medical appointments. I used SpringBoot, ReactJS and MongoDB as the main frameworks for our web application, user authentication and data storage. I used AWS and DockerHub as our cloud platforms to deploy the Web Application and Github for controlling the source code to save the work. This project is mainly written in java and the react framework and a few other languages such as HTML, Javascript and CSS.The motivation for me to make this web application was because i found it quite difficult to get appointments for medical circumstances. For example getting a doctors appointment would be a lot easier if you had an account with your GP and you logged on and can book an appointment with ease. I find this would be a lot easier for a lot of people because it would save the effort of having to ring up and wait on the phone to book an appointment when you could just do it on your mobile phone or website.

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.

About

ATU Final Year Project 2023

Readme
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

● JavaScript 92.7% ● CSS 6.2% ● HTML 1.1%

Suggested Workflows
Based on your tech stack

Deno Configure

Figure 2.4: Github Version Controller

Chapter 3

Technology Review

This segment discusses the various technology that I used to construct the web application. I will use code snippets and visual aids to help visualize how these technologies work in the hope that you get a better understanding why I used these technologies. The technologies I will be talking about are:

- MongoDB
- NodeJS
- ReactJS
- ExpressJS
- EmailJS
- Postman
- Github
- Github Desktop
- Docker
- Robo3T
- Amazon Web Services(AWS)
- Visual Studio Code

3.1 Frameworks and Libraries

I used a few different libaries and frameworks while working on this project. These are ReactJS and NodeJS.

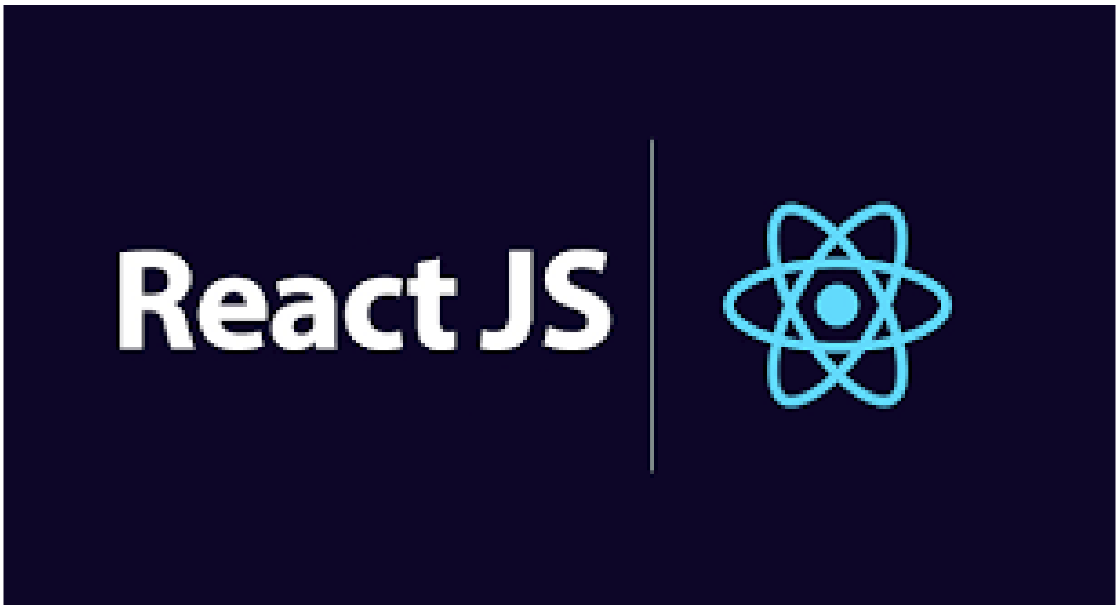


Figure 3.1: ReactJS

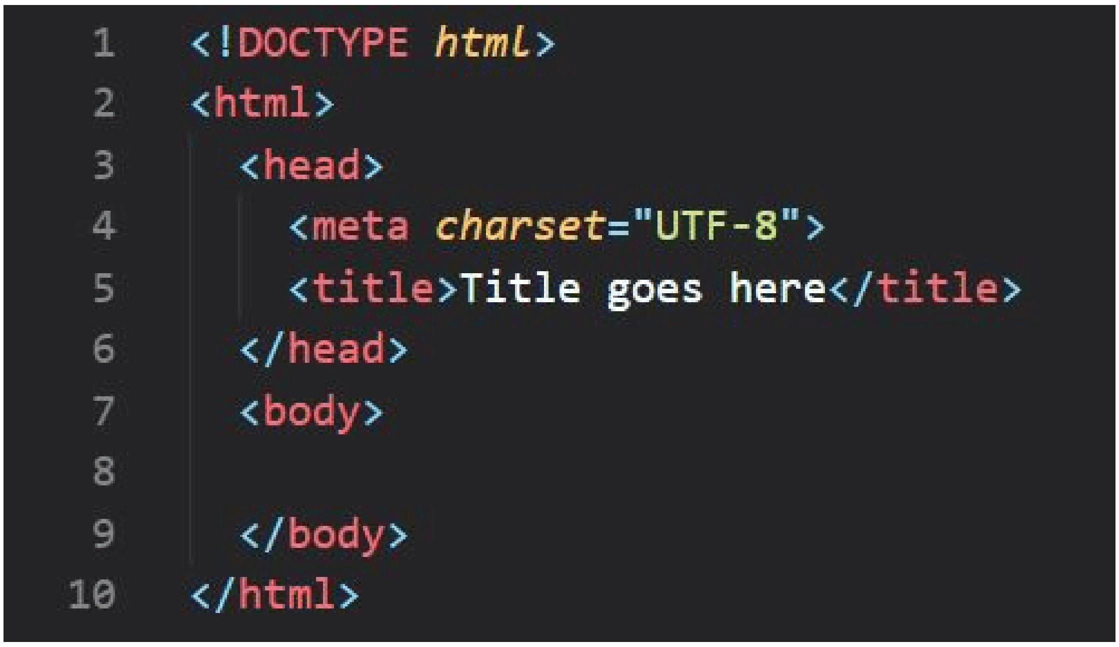
3.1.1 ReactJS

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.[1]

I 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.[2]



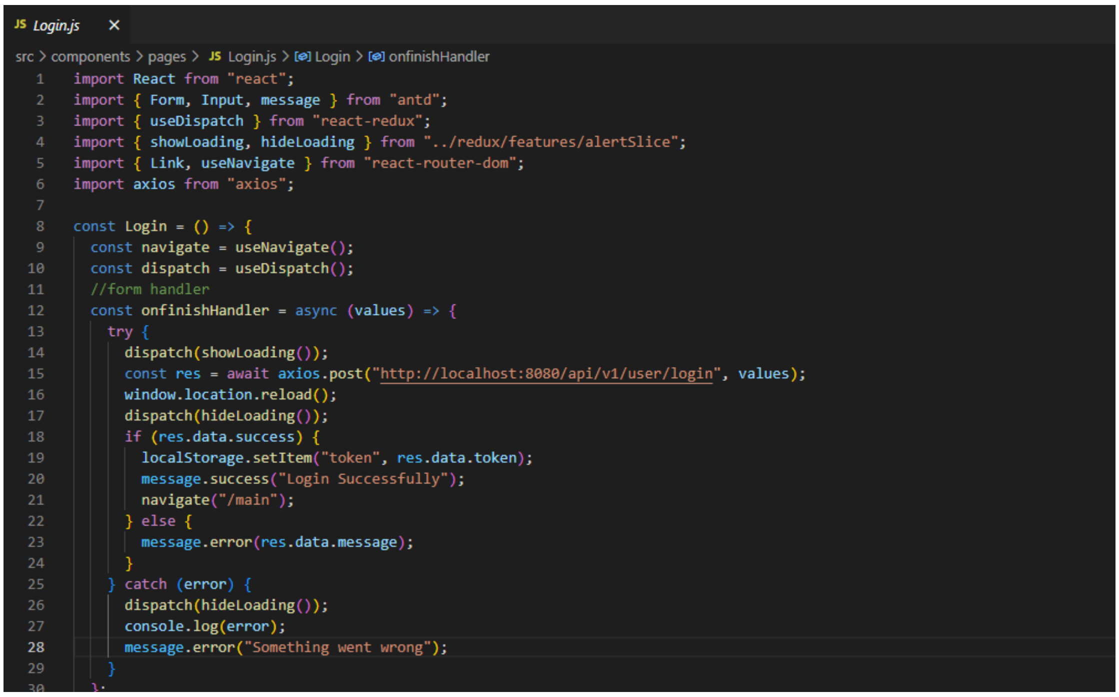
```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Title goes here</title>
  </head>
  <body>

  </body>
</html>
```

Figure 3.2: HTML

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.[3]


Login.js
src > components > pages > Login.js > Login > onfinishHandler

```
import React from "react";
import { Form, Input, message } from "antd";
import { useDispatch } from "react-redux";
import { showLoading, hideLoading } from "../redux/features/alertSlice";
import { Link, useNavigate } from "react-router-dom";
import axios from "axios";

const Login = () => {
  const navigate = useNavigate();
  const dispatch = useDispatch();
  //form handler
  const onfinishHandler = async (values) => {
    try {
      dispatch(showLoading());
      const res = await axios.post("http://localhost:8080/api/v1/user/login", values);
      window.location.reload();
      dispatch(hideLoading());
      if (res.data.success) {
        localStorage.setItem("token", res.data.token);
        message.success("Login Successfully");
        navigate("/main");
      } else {
        message.error(res.data.message);
      }
    } catch (error) {
      dispatch(hideLoading());
      console.log(error);
      message.error("Something went wrong");
    }
  };
```

Figure 3.3: JavaScript

CSS/ SASS/ SCSS

Withing the project, I employ a variety of styling techniques; the following is a brefi description of some of the styles I 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.[2]
- 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.[4]

- 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.[5]

3.1.2 EmailJS

Developers can send emails using client-side code from their own applications using EmailJS, a cloud-based email delivery service. Without having to set up and operate their own email servers, developers can easily write and send emails using HTML templates and a straightforward API thanks to EmailJS.

A robust collection of capabilities offered by EmailJS make it simple for developers to incorporate email functionality into their projects. These capabilities include the capacity to send both plain text and HTML emails, support for several email providers (including Gmail, Yahoo, and Outlook), and the capability to monitor email openings and clicks.

EmailJS offers a variety of additional services and features in addition to its basic email delivery capability to assist developers in creating and managing their email campaigns. A visual email template editor, email statistics, and reporting are a few of these. Others include integrations with well-liked third-party programmes like Slack and Zapier.[6]

3.1.3 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.[7]

3.2 Databases

The data tier refers to the processing of data that have been stored in the database. The user will be able perform CRUD (create, read, update, delete) operations on

Figure 3.4: NodeJS

the application. The main database I have setup for the project, The account information is stored in MongoDB and website data is also stored there.

3.2.1 MongoDB

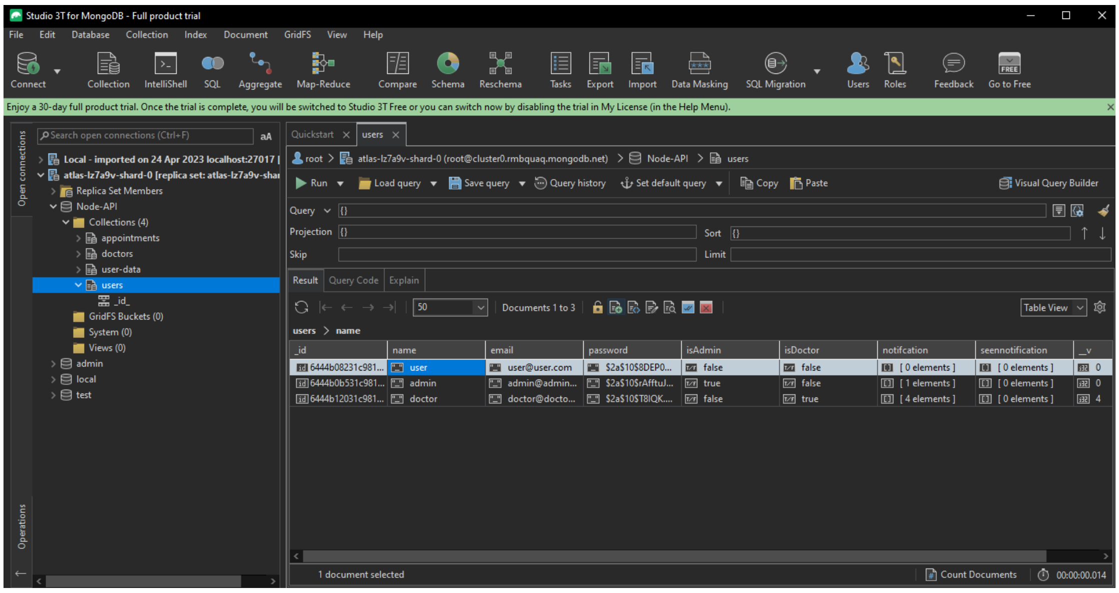
MongoDB is a document database, meaning it contains information in JSON like formats. According to MongoDB, “We believe this is the most natural way to think about data and is much more expressive and powerful than the traditional row/column model”. MongoDB’s goal is to store and recover data easily so that developers can switch and program quickly.[8] 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.[9] 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

Figure 3.5: MongoDB

very efficient software that helps us to view and analyse our data.[8]
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.[10] The mongo database was originally set up locally to validate the project, but after the initial implementation stage, I 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. While testing the project, I used Robo3T to keep track of the data coming into the database. I had to make sure that the user data was being saved to the database and that their passwords being hashed so the user stayed protected. I 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.[11]


Studio 3T for MongoDB - Full product trial
File Edit Database Collection Index Document GridFS View Help
Connect Collection IntelliShell SQL Aggregate Map-Reduce Compare Schema Reschema Tasks Export Import Data Masking SQL Migration Users Roles Feedback Go to Free
Enjoy a 30-day full product trial. Once the trial is complete, you will be switched to Studio 3T Free or you can switch now by disabling the trial in My License (in the Help Menu).
Local - imported on 24 Apr 2023 localhost:27017
atlas-lz7a9v-shard-0 [replica set: atlas-lz7a9v-shar
Replica Set Members
Node-API
Collections (4)
appointments
doctors
user-data
users
_id_
GridFS Buckets (0)
System (0)
Views (0)
admin
local
test
Quickstart users
root > atlas-lz7a9v-shard-0 (root@cluster0.rmbquaq.mongodb.net) > Node-API > users
Run Load query Save query Query history Set default query Copy Paste Visual Query Builder
Result Query Code Explain
Documents 1 to 3
Table View
1 document selected
Count Documents
00:00:00.014

Query {}
Projection {} Sort {}
Skip Limit

users > name

_id	name	email	password	isAdmin	isDoctor	notifcation	seennotification	__v
6444b08231c981...	user	user@user.com	$2a$10$8DEP0...	false	false	[0 elements]	[0 elements]	0
6444b0b531c981...	admin	admin@admin...	$2a$10$rAfftuJ...	true	false	[1 elements]	[0 elements]	0
6444b12031c981...	doctor	doctor@docto...	$2a$10$T8IQK....	false	true	[4 elements]	[0 elements]	4

Figure 3.6: Robo3T

3.3 Integrated Development Environment (IDE)

The main IDE used to work on this project was Visual Studio Code. This handled the backend and frontend of the web app. [12]

3.3.1 Visual Studio Code

There are a lot of great IDEs that I 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.[13]

Figure 3.7: Visual Studio Code

I have familiarity with Visual Studio Code, having used it for over two years on a variety of projects. I have tried a few extensions in VS Code and concluded that it was the better option. I 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.

3.4 Cloud Hosting

I had to use two separate platforms to host our web application in the cloud. I was able to host the 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

I had to run each part each part separately because the project uses several functionalities such as ReactJS, MongoDB and NodeJS. This became very tedious and time-consuming procedure as the project grew in size. I looked into this issue and discovered Docker. Docker is a popular opensource 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.[14] This was the ideal solution to my dilemma because it saved me

Figure 3.8: Docker

time from having to execute each function separately.

In order to use Docker, I 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 faustastamulis/medical-app:katest
- docker build -f Dockerfile -t faustastamulis/FinalYearProject

These commands generated images for the project's frontend and backend. I 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 I didn't have to run each project part individually.

Originally, I was running the project locally, but as it progressed, I began to use the cloud. I had to make a few adjustments to the code and remake the images which would be pushed up to Docker Hub. To run the application on cloud, I had to create a new docker-compose file on the AWS instance. The images were pulled from the 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”.[15]

I had never used Docker before this project, but after doing some research, I discovered that most businesses use it to create images and run them in the cloud and soon, I will be applying for jobs in this industry, I 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

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.[16]

I looked at all of these services to see which will be the most appropriate for the project. I decided to use EC2 to set up an instance on which I could run our web application. I 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, I 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. [17]

- sudo apt-get update
- sudo apt-get install apt-transport-https ca-certificates curl gnupgagent software-properties-common
- curl -fsSL https://download.docker.com/linux/ubuntu/gpg — sudo aptkey add -
- sudo apt-get install docker-ce docker-ce-cli containerd.io

Figure 3.9: Amazon Web Services

- 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

I was able to customize the instance using these commands. Then I 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.[7]

After that, I decided to start over and build a new instance. I chose the medium tier this time, which would cost me around 20c per hour. I 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. I was 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 a fantastic feature of GitHub; it allows you to get another person's code and make improvements to it in your repository.

Figure 3.10: Github

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. I 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. I can also even inspect all of the modifications I have made to the project before committing them using GitHub Desktop. If I feel that these modifications aren't what I want, I can roll back directly from GitHub Desktop, saving me the trouble of getting into the code and removing any changes that I have created. Many of these features are accessible via GitHub Desktop's user-friendly graphical user interface.[18]

3.6 Others

3.6.1 Postman

Postman is a collaboration platform for API development. I've been using Postman since the beginning of development. I initially used it to verify whether the Mongo database was linked to the back end by sending Get and Post requests. The next step was to see how I could submit user information in JSON format to register a user and then use the same information to login. I gave Postman the link to our back end and sent a request to that link to see if we can get a response.[19]

3.6.2 Microsoft To Do

Microsoft To Do is a cloud based to-do list and task management programme that aids in organising and setting priorities for your chores. You may create tasks and subtasks, set reminders and due dates, group tasks into lists, and annotate tasks with Microsoft To Do. If you're working on a group project, you can also provide responsibilities to particular individuals. You can access your tasks from anywhere because the app is available on a variety of platforms, including Windows, iOS, Android, and the web. You can manage your tasks from within other Microsoft products like Outlook, Microsoft Teams, and OneNote thanks to Microsoft To Do's integration with them. To assist you in staying on top of your duties, it also provides intelligent ideas, such as reminders for approaching deadlines.

From the beginning of the project, I used to-do to handle the necessary tasks needed to be completed. I maintained meticulous records of everything I did, noting which challenges we encountered and how I resolved them. I divided the project into a number of smaller tasks. I also allocated how much time I can

Figure 3.11: Postman

spend on each task; for example, Some of these tasks took longer than usual so I decided to give my self certain time limitations and targets to hit. This piece of software motivated me on getting tasks done because I found checking off the tasks quite rewarding and satisfying. This also helped maintain a good workflow for the project.

Figure 3.12: Microsoft To Do

Chapter 4

System Design

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

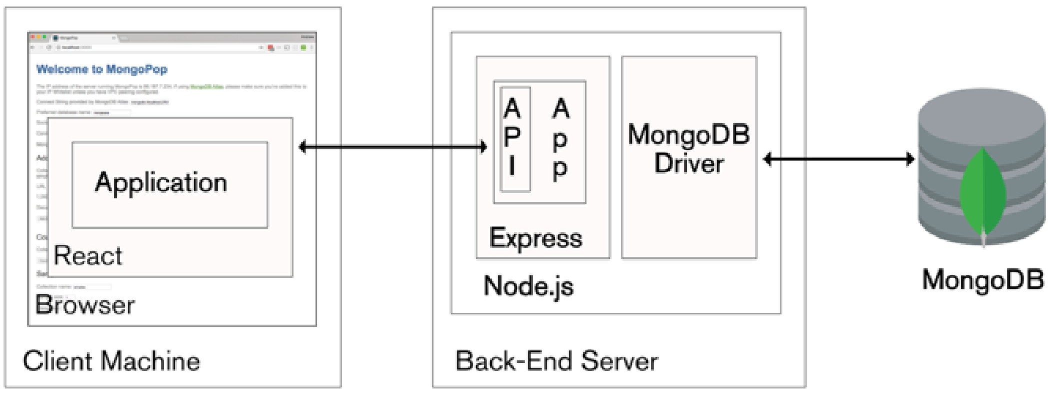


Figure 4.1: Project Schematic Diagram

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, I'll be using one database, MongoDB. The database is used for storing users and is also 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[20] below:

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

These commands allowed me 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 server API in the backend of the project. This server is connected to MongoDB using mongoose and checks the user credentials to successfully register and log in the user.

When setting up Mongo, I ran into a few problems. I couldn't store or retrieve any data into or from the database because of the user schema not being set up. I spent a lot of time trying to figure out what was wrong, but I couldn't. While investigating the mistake, I discovered a program called Robo3T, which provided me with a proper GUI that enabled me to view the database in depth and see what was wrong. I discovered that the user data wasnt being posted to the database. To resolve this problem, I tweaked a some of the server API.

Another major fault I ran into was CORS when I was trying to process the requests. Since the website is running as a HTTP file and i was converting the data to JSON the CORS policy blocks this and I had to overcome this with the following commands and add the following code into the backend:

- npm install cors
- var cors = require('cors')

- app.use(cors())

The backend was successfully linked to the database after making these changes. I also made a note of the problem and solution so that I could correct it if the bugs resurfaced again when I set up the database for cloud deployment.

4.2 Backend

In order for the user to save any details on to the database, I had to set up a backend. As mentioned previously I was using NodeJS and ExpressJS 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 backend has a number of folders which allocate different tasks to be done for example I have the controllers which have a number of features like getting data processing data. Then I have routes so there are multiple routes to use in the web application for example if the user is an admin they would go through the admin route which would give them admin privileges, there's the user route which is a public route and there's the doctor route which doctors can use as a private route. I have the models folder which sets up the necessary field of data in the database. Then we have the config folder which connects the server to the Mongo Database. Then finally the main server.js file which does the requests and sends them out the necessary controllers.

4.3 Frontend

I'll be discussing four related topics with our front end. I'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.

- Data - this folder contains the data for when the user is logged in and the sidebar.
- Hero - is the beginning screen page you see in the home page, I was going to incorporate a nice graphic but there was no good graphics.

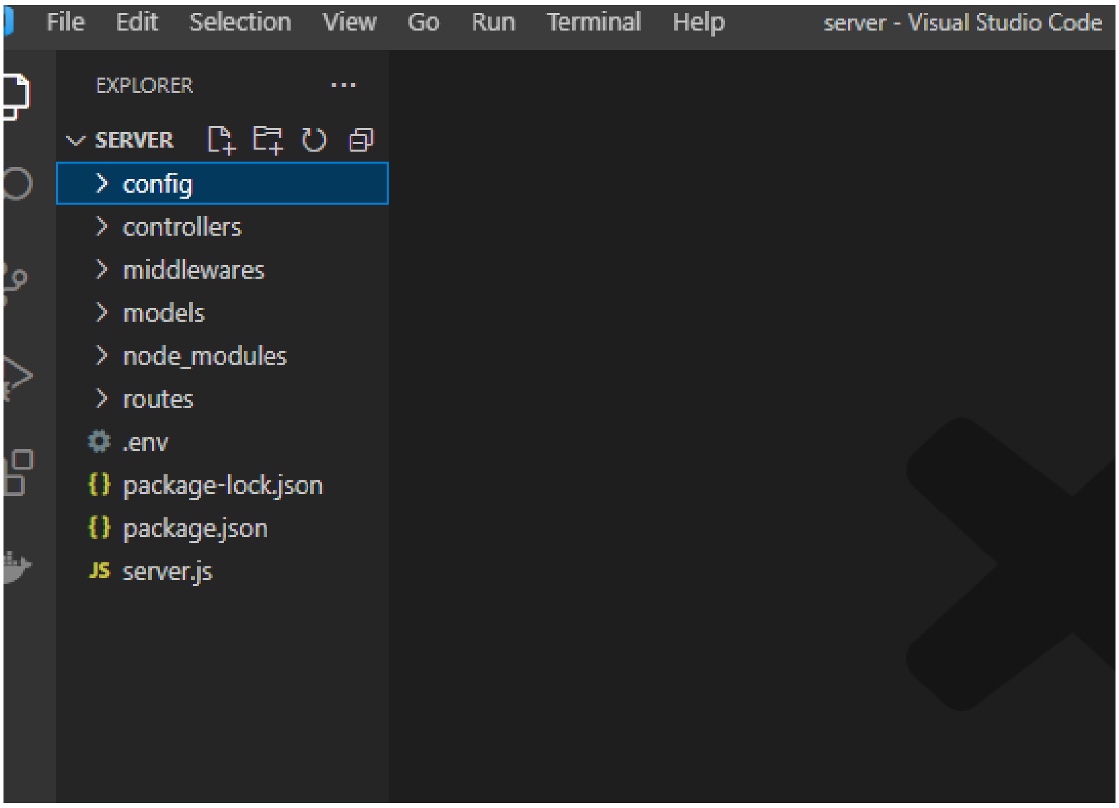
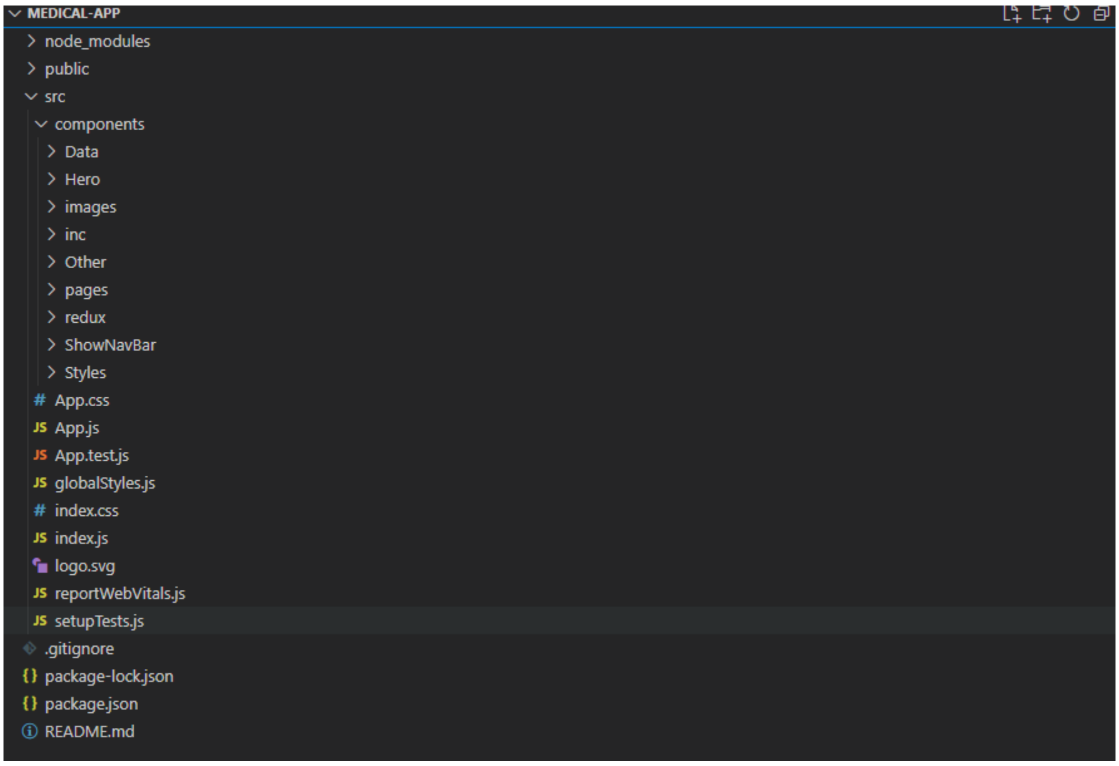


Figure 4.2: Backend Project File Setup

- Images - this folder contains all the images that we used throughout our project.
- Inc - is the Navbar
- Other - is the routes and doctors list the user sees when they log in.
- Pages - which contains all the different pages in web application.
- Redux - is the alert notifications
- ShowNavBar - is the file that contains when to show navbar or not.
- Styles - is the css files for the project which give the website a nice graphic element.
- 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.



```
MEDICAL-APP
  node_modules
  public
  src
    components
      Data
      Hero
      images
      inc
      Other
      pages
      redux
      ShowNavBar
      Styles
    App.css
    App.js
    App.test.js
    globalStyles.js
    index.css
    index.js
    logo.svg
    reportWebVitals.js
    setupTests.js
  .gitignore
  package-lock.json
  package.json
  README.md
```

Figure 4.3: React App Structure

- 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

- Appointment page
- Notification page
- Apply-doctor page
- Booking page

4.3.3 Home Page

For our home page I wanted to take a minimalist approach, but I also wanted the page to catch the user's attention. That is why I decided to put as little information on the home page as possible. I went for a simple 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.

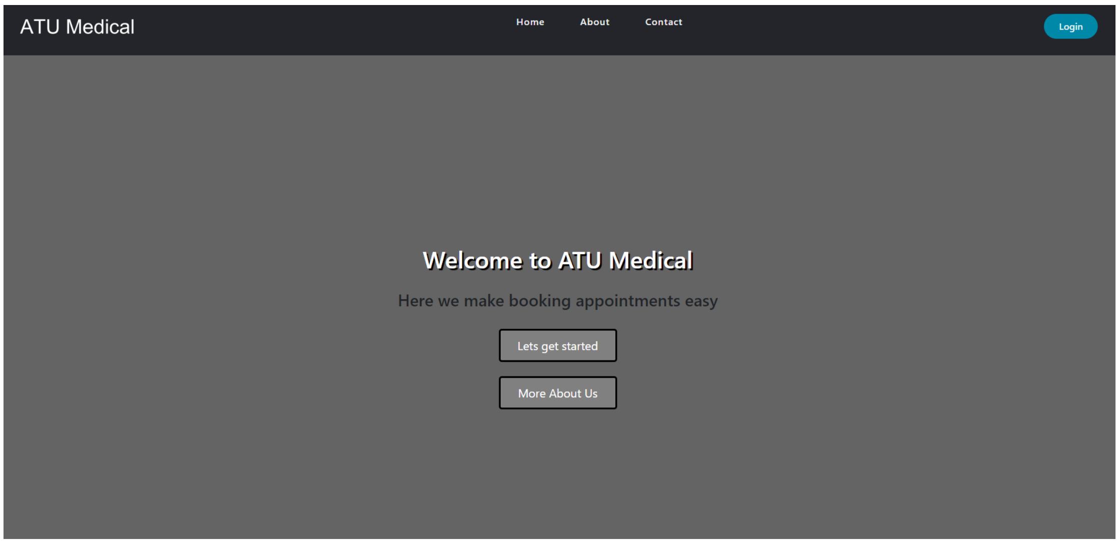


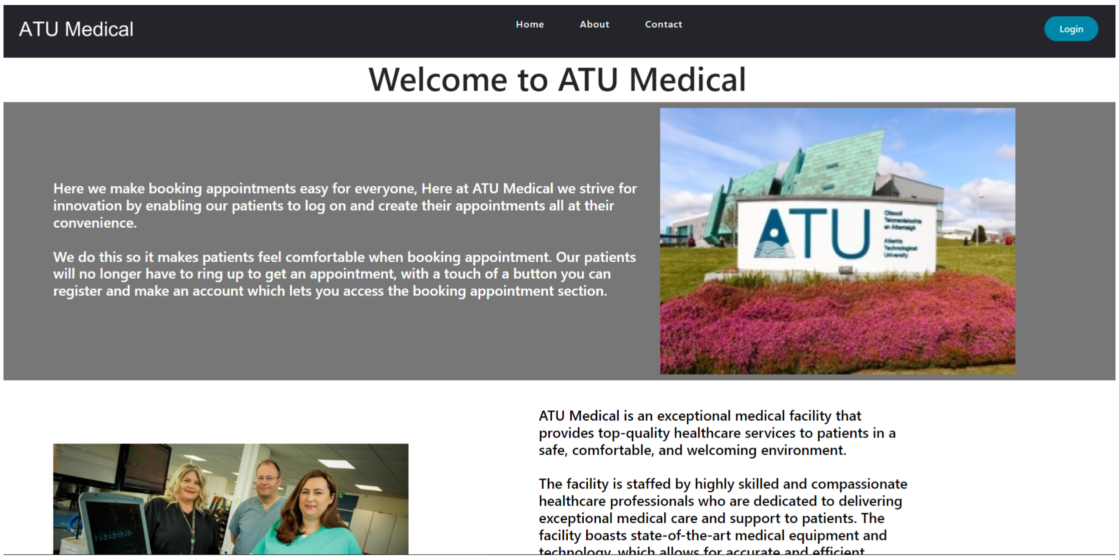
Figure 4.4: Home Page

4.3.4 Navbar

I have a navbar in the web application that allows the user to navigate through the application. There are two types of navbars in my project. I have a standard navbar at the top of most pages and I 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. 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 be signed out after they have closed their machines. 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.

4.3.5 About us page

I have a standard about us page, that gives the user more insight into the application and what I have planned for the future. This can also be edited so it would any industrial medical company.



Welcome to ATU Medical

Here we make booking appointments easy for everyone, Here at ATU Medical we strive for innovation by enabling our patients to log on and create their appointments all at their convenience.

We do this so it makes patients feel comfortable when booking appointment. Our patients will no longer have to ring up to get an appointment, with a touch of a button you can register and make an account which lets you access the booking appointment section.

ATU Medical is an exceptional medical facility that provides top-quality healthcare services to patients in a safe, comfortable, and welcoming environment.

The facility is staffed by highly skilled and compassionate healthcare professionals who are dedicated to delivering exceptional medical care and support to patients. The facility boasts state-of-the-art medical equipment and technology, which allows for accurate and efficient

Figure 4.5: About us page

4.3.6 Contact us page

I set up a contact page because I wanted users to send us feedback on any issues they might be having or send us ideas on how to improve the website. I wanted user feedback to be one of the most important elements of the application. I set up a separate page for this part of the application. The contact page was connected to the EmailJS database which is an application allows development through react without using the server side. The docs clearly gave me an explanation on how incorporate this into the contact page and it was very easy to set up. Now the user can contact me directly with any issues, questions or feedback they would like to give me.

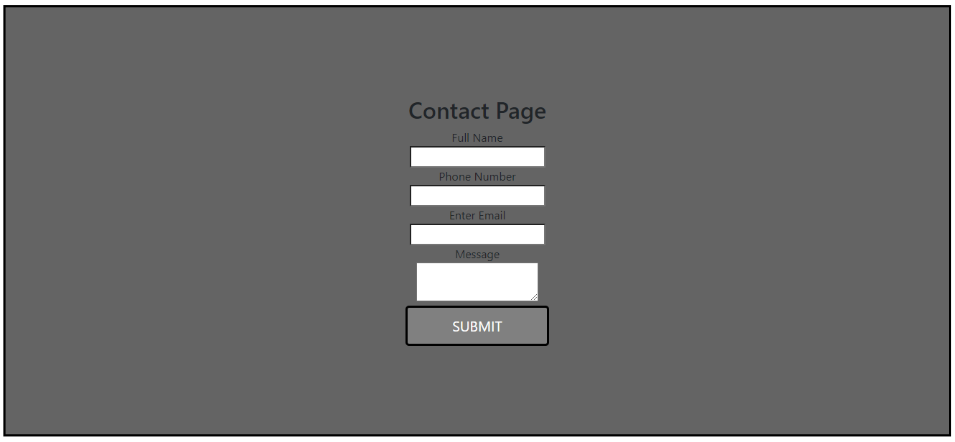


Figure 4.6: Contact us page

4.3.7 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 password must be hashed using bcryptjs which is a hasing algorithm. 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.

4.3.8 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 Server 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.

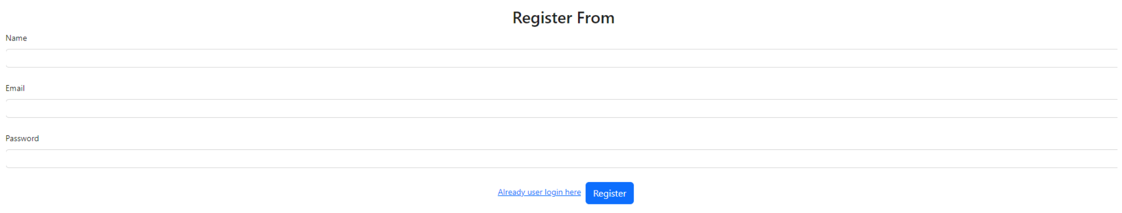


Figure 4.7: Register Page

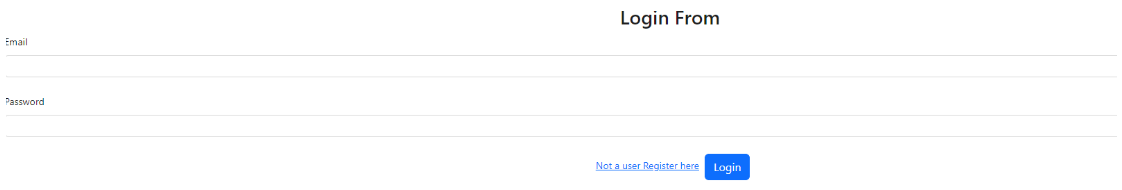


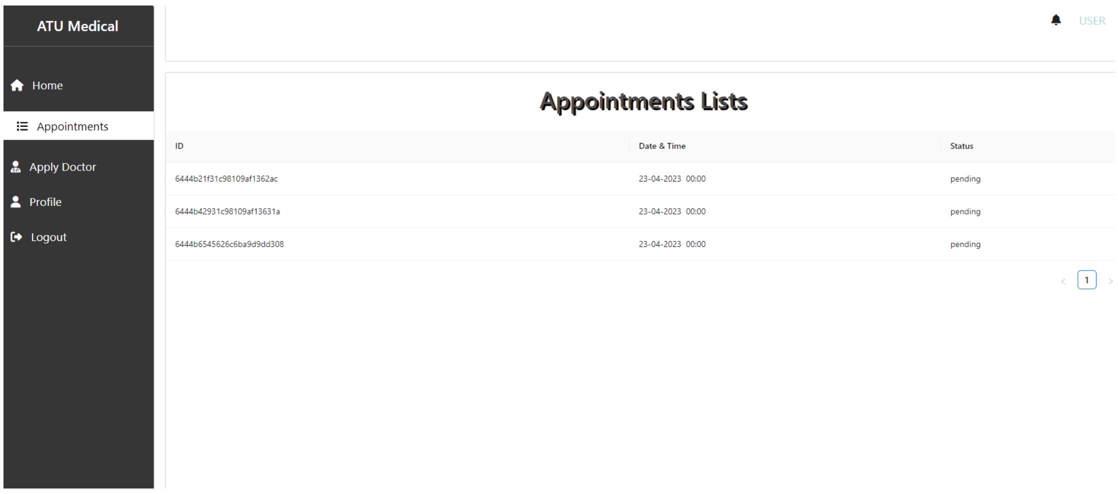
Figure 4.8: Login Page

4.3.9 Appointment Page

Once the user is logged on they can view their appointments by clicking one of the sidebar appointments tab. This sends a get request from the server side and pulls the data out of MongoDB to output for the use to see the appointments. It has a specific id, date and time and status. Status can change when the doctor logs and either approves or denies the appointment.

4.3.10 Notification Page

Notification Page when a user makes a request whether it be the admin, doctor or patient it sends a notification to right person. For example, if the patient books an appointment the doctor gets a notification that the patient had booked an appointment the doctor can then approve or deny the appointment.



Appointments Lists

ID	Date & Time	Status
6444b21f31c98109af1362ac	23-04-2023 00:00	pending
6444b42931c98109af13631a	23-04-2023 00:00	pending
6444b6545626c6ba9d9dd308	23-04-2023 00:00	pending

Figure 4.9: Appointment Page

4.3.11 Apply-Doctor Page

Users can apply to become doctors which then have to be accepted by the admin user, this would aid a lot of GP's and medical institutes because there is a shortage of staff right now so that's why the waiting timings are very long. The user has a form to fill out including their personal details such as Name,Phone Number, Address, Email and Website. It also asks for the professional details such as specialization, experience and fees per consultation and timings.

4.3.12 Booking Page

The booking page is when a user logs in, they can click on the doctor they would like to book an appointment with and schedule an appointment it comes up with the necessary fields to fill like the name, phone number, email, time and date. It allows the user to check the availability and prompts whether the appointment is available. Then the user can click the Book Now button which sends a request to server which is handled by the controller to send over to the doctor. The doctor then can proceed and to either accept or decline the appoinment.

4.4 Cloud Deployment

I decided to run our project on Amazon Web Services (AWS) so that it could be available from anywhere, on any device. If I 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

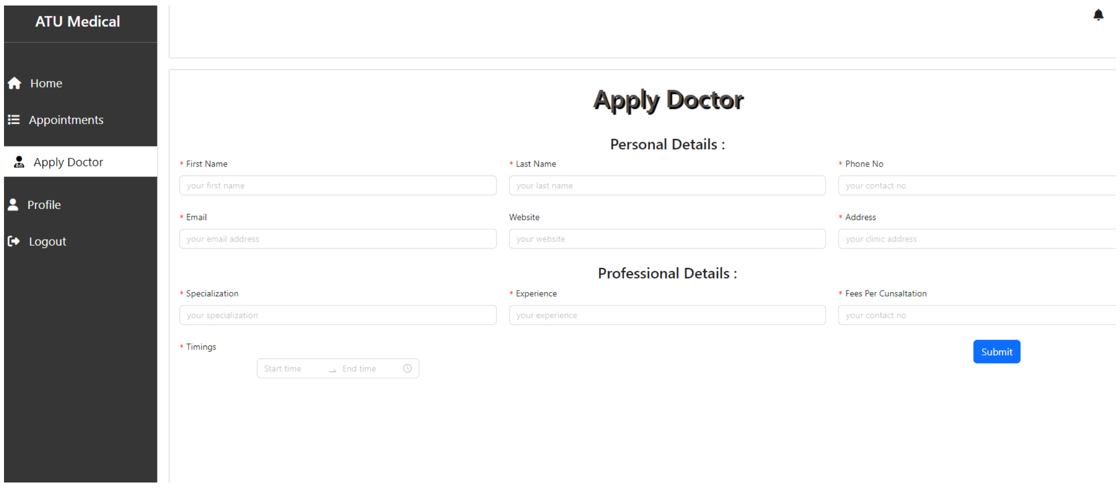
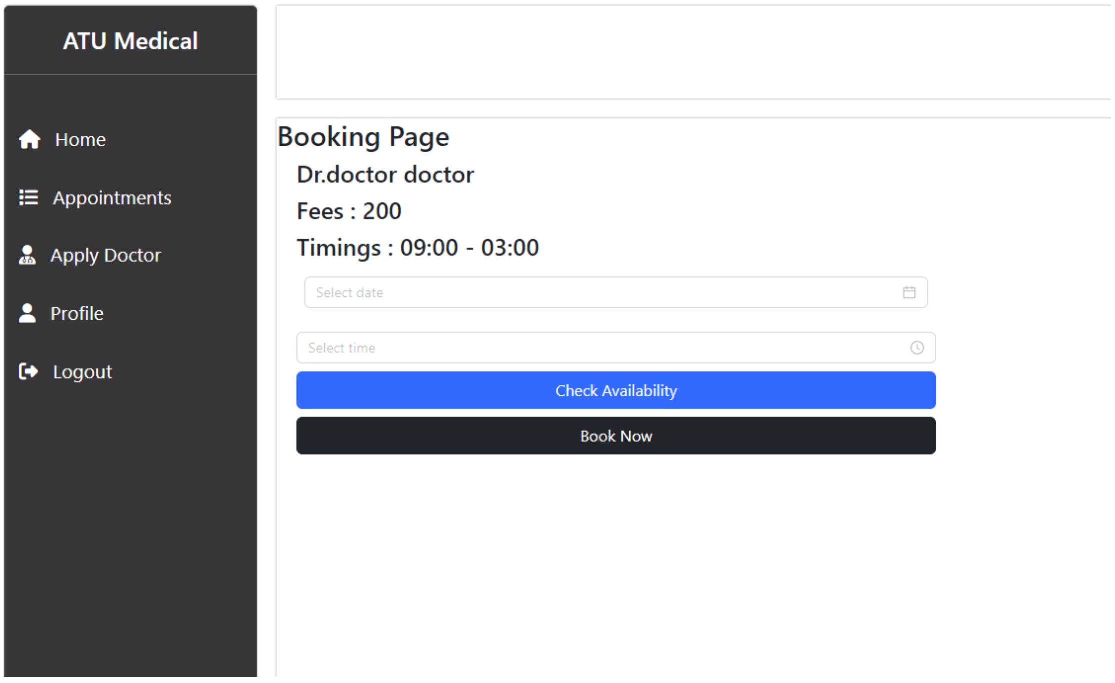


Figure 4.10: Apply Doctor Page

cloud I 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.



Booking Page

Dr.doctor doctor

Fees : 200

Timings : 09:00 - 03:00

Figure 4.11: Booking Page

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, I will look for any enhancements or new functionality that may have been incorporated into the finished product.

5.1 Testing

I 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. I used an agile approach during the project, which meant I focused on continuous testing while updates were made to the project. This was, in my view, a much better option than using the waterfall solution, given the project's size. Although this method took up a lot of my time, I felt it helped me 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, I used black box testing. I was able to do this testing by asking my peers, colleagues and family to test the web application and flag any bugs/errors to me. I 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.

I opted to go for white box testing for the backend. I used a program called

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

When working with MongoDB, I double-checked that the data and information were properly stored in the database. I was 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 my view, was very beneficial because it helped me 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. I concluded that this initiative resulted in a more robust method overall, and I was able to move on with our project after completing these tests.

5.2 Limitations

Appropriately, I found less limitations than anticipated with the technologies and programming paradigms I 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 my use of Agile and how it enabled me 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, I do not have to speak of them as something negative. It provided me with very useful information about the project and how we developed it. It will allow me to improve on the project in the future and acquire expertise that I can apply in the upcoming years.

I encountered obstacles on my way when working on the project that stopped me to make progress. Initially, when implementing the ReCAPTCHA into the login page, it was giving me a problem as I was not able to prop-erly implement it into our project. I 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, I changed the ReCAPTCHA code to Google ReCAPTCHA, which could work on localhost. We maintained track of the code we wrote so that we could reuse it in the future if necessary. When working on the appointments page the date and time wasn't properly recording it would always just set to 00:00. The notifications icon is supposed to send out notifications also but there was a bug where the frontend wasn't picking up the notification array to database. The contact page was a bit tricky to set-up but the docs for the technology I used called EmailJS made it quite easy providing me with some sample code for a react project.

The booking appointment application, which allowed users to book appointments with the doctor from the medical facility, is one of the project's key features. I decided to rely solely on a REST API to perform all of the necessary tasks. I wanted to challenge myself and build my own API where users could log on and make appointments with their doctors. It took a long time, but I eventually overcame the challenge and came up with a solution. I created multiple controllers and classes that implemented the API in the backend which then I had to connect to the frontend so it would send those requests. This allowed me to reuse my 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. One of the main limitations was time management.

I wanted to make the app available to consumers as a smartphone app. This was attempted using Android Studio and Ionic Capacitor. However, I ran into a slew of mistakes that prevented me 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 I contrast the end result of the web application to the initial goals, which were stated in Chapter 1 of this dissertation, I concluded that I have met the majority of the criteria that I have set for myself. I haven't confirmed whether or not this application will benefit patients, however I hope it will. Since it will be used by patients, I also wanted the interface to be simple and easy to navigate as majority of patients are quite old. The goal, I think, has been met. To achieve this, I looked into various medical applications, especially HSE, in order to gain a better understanding of a high-end website.

Chapter 6

Conclusion

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

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

6.1 Objectives

The main goal when I started the project was to make a web application that would help patients with their online experience and make it more pleasant. I am pleased to report that our web application has accomplished this.

The things I have accomplished are:

- Created a web application using NodeJS and ReactJS
- Web application contains the necessary components for a website i.e. About us, Contact page, Home page and a Navbar.
- Users can email me.
- 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 appointments component and perform CRUD operations on it.
- Logged in users can see all the available doctors and pricing.
- Logged in users can apply to become a doctor.
- The application runs successfully on AWS cloud.
- The application is mobile ready, as the web app is designed to fit any screen size

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

I think with the current shortage of nurses and long wait times for appointments because of the global pandemic. This web application would be very useful to organise appointments and patients into a website. Users can apply to become a doctor/nurse which would aid the shortage of nurses currently in the health care sector.

6.2 What I learned

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

I had never written a paper as long as this before, but writing a dissertation was a new experience for me and my peers. I had a basic understanding of research papers thanks to one of our modules, Research Methods. I discovered that the dissertation allowed me to thoroughly clarify my project and thought process in a more comprehensive way than the project alone. I'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 the code.

6.3 Future Development

I'm fully committed to make this app the best it can be. I plan to add a few more elements to this app, such as a messenger, to make it even more enjoyable.

The 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. I initially created the web app so that it can be mobile compatible, but I struggled to convert it onto a mobile app due to its size. The application was too large, and I could not find a way to convert it to a mobile app in time.

6.4 Business development

I would like to contact HSE (Health Service Executive) and different GP's (General Practitioner's) to see if they would like to use any of our features in their operations, as this project is primarily aimed at patients in Medical Institutes. If they intend to go through with the plan, we'll be able to help these institutes improve and update their website and receive feedback directly from patients.

Chapter 7

Appendix

7.1 Project Source Code Link

https://github.com/faustastamulis/FinalYearProject

Bibliography

[1] What is reactjs: Introduction to react and its features
https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs.

[2] Html / css
https://www.w3.org/standards/webdesign/htmlcss.

[3] What is javascript?
https://developer.mozilla.org/en-us/docs/learn/javascript/firststeps/whatisjavascript.

[4] Why sass?
https://alistapart.com/article/why-sass/.

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

[6] What is emailjs?
https://blog.openreplay.com/sending-emails-from-react-with-emailjs/#:~:text=emailjs.

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

[8] Mongodb features
https://www.mongodb.com/what-is-mongodb/features.

[9] What is nosql?
https://www.mongodb.com/nosql-explained.

[10] Why you should be using json vs xml
https://cloud-elements.com/.

[11] 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-

[12] What is an ide?
`https://www.codecademy.com/articles/what-is-an-ide`.

[13] Visual studio code
`https://code.visualstudio.com/`.

[14] What is docker?
`https://www.ibm.com/cloud/learn/docker`.

[15] Overview of docker compose
`https://docs.docker.com/compose/`.

[16] What is aws?
`https://aws.amazon.com/what-is-aws/`.

[17] Deploy-docker-container-on-aws
`https://github.com/soumilshah1995/deploy-docker-container-on-aws`.

[18] An introduction to version control using github desktop
`https://programminghistorian.org/en/lessons/retired/getting-started-with-github-desktop`.

[19] Postman
`https://www.postman.com/`.

[20] How to run mongodb as a docker container
`https://www.bmc.com/blogs/mongodb-docker-container/`.