

UNIVERSITY COLLEGE LONDON

DEPARTMENT OF COMPUTER SCIENCE

Clinic scheduling assistant

Project Supervisor
Yuzuko Nakamura

Author
Guangpeng Zhan (19072393)

Project Client
Emma Matthews

*Project
Representative*
Gemma Molyneux

1 September 2020



Abstract

The Great Ormond Street Children's charity set up a new team to support the development of new technologies in 2018. The client is GOSH DRIVE, a new unit dedicated to explore new technologies to improve healthcare service at GOSH as well as NHS. In GOSH Drive, a small clinic team provides regular physical test to children. However, scheduling new patients and ensuring current patients are seen on time is a very time consuming task, which currently is done by administrator. A program that could arrange appointment automatically could save clinician's time on this administrative work allowing them to spend more time on clinical work. The goal of this project is to create a responsive web application that could generate clinical schedule automatically based on inputted parameters.

The technologies in this project include MongoDB, Nodejs, Express as essential back-end languages and JQuery is used to add interactive feature on the front end. It is the first time for the author to use MongoDB, JQuery, which has been proved to be a great challenge. In order to have a full command of these programming languages, effective communication with supervisor and great learning skills help a lot.

The website meet all "must have" and "should have" requirements in MosCow requirement table. It can generate timetable automatically based on many parameters. Admin can edit or delete all appointments and refresh the timetable. Clinicians can view all the appointment they have on their main page and add note to one single appointment record. The application help admins arrange appointment quickly and spend more time on patients.

Contents

1	Introduction	3
1.1	Client	3
1.2	Problem statement	3
1.3	Project Goal	3
1.4	Project Management	4
1.5	Overview	4
2	Related Research	7
2.1	Related Application	7
2.2	Related Technologies	8
3	Requirement Analysis	13
3.1	requirement	13
3.1.1	Persona	13
3.1.2	User Cases	17
4	Design and Implementation	19
4.1	System Architecture	19
4.2	Implementation of important Features and Key modules	20
4.3	Database Representation	26
5	Testing	27
5.1	Responsive Page Testing	27
5.2	User Acceptance Testing	28
5.3	Conclusion	29
6	Appendix A	32
6.1	Use Case Specification	32
6.2	User Manual	40
6.3	Source Code Examples	45

Chapter 1

Introduction

1.1 Client

Digital Research, Informatics and Virtual Environment (DRIVE) is a part of Great Ormond Street Hospital, focusing on using cutting-edge technology and advanced data analysis to improve healthcare service for GOSH patients as well as wider NHS.

1.2 Problem statement

As a part of Great Ormond Street Children's charity, GOSH is a unit dedicated to enhance healthcare service through digital innovation. A clinic team in GOSH provides regular and psychological tests to children. Scheduling new patients and ensuring current patients to be seen on time is a very time consuming task. A program that can schedule appointments would help save time and help clinicians focus on clinic work rather than administrative work.

Each clinician works part-time and every child has three face-to-face appointments and one school observation. More appointments will be arranged if needed. After the patient registered, the first appointment should be arranged within 18 weeks. After the first appointment, clinicians should see the patient once a month. Due to the limitation of the working place, the clinic team will only work on Monday, Tuesday and Thursday. When the system generates timetable, clinicians' maximum working day, bank holiday etc. should also be taken into consideration. To make the schedule more flexible, admins can add, edit and delete any appointment record through the website. Clinicians can see each appointment they have, and add note to the appointment if needed.

1.3 Project Goal

After communicating with the client, a clear MosCow requirement table has been established to identify the order of priority. The requirement table become more and more clear and precise after further meetings and email contacts. The aim of

the project is to save clinician's valuable time on administrative work and spend more time on meaningful clinic work.

1.4 Project Management

The whole project can be divided into four parts: Design, front-end coding, back-end coding, Testing Phase and Report Writing.

As it is indicated by 1.1, during design phase, the first task was to identify the client's need and MosCow table was utilised to prioritizes requirement. Then, interactive prototype was made to visualize the application to help the client better understand how the website work. Finally, Grantt Chart was made to ensure the project finished on time.

For the front-end coding parts, HTML, CSS are used to make static web page. Bootstrap played an important role in website decoration, allowing the author to make beautiful web design with ease. JavaScript and JQuery were used to add interactive elements, As for the back-end, Nodejs, Express had been utilised to the server side application. MongoDB was used as database system.

1.5 Overview

The report has been divided into six parts: Introduction, Research, Requirement Analysis, Design Implementation, Testing and Conclusion.

chapter1: Introduction

This chapter outlines the client information and a brief overview of project goal and problem.

chapter2: Research

This chapters introduces background information, related researches of previous academic experiment. Furthermore, it will also indicates the selected technologies gives reasons why they are selected.

chapter3: Requirement Analysis

Chapter3 provides detailed problem statement, MosCow requirement list, use case as well as requirement analysis. The aim of this chapter is to identify the exact requirement of the client.

chapter4: Design Implementation

This chapter will introduce detail of the website application, including application architecture, the mechanism of the website as well as database storage. UML will be utilised to visualize the final design of the application.

chapter5: Testing

The chapter5 describes the selected testing strategy such as unit test, and the reason why the author chose this strategy. Moreover, A list of example of tests will be shown to display how the test were conducted.

chapter6: Conclusion

This chapter will conclude the final result of the project and analyze the requirement list to evaluate the result critically. It will also present some final thoughts of the author.

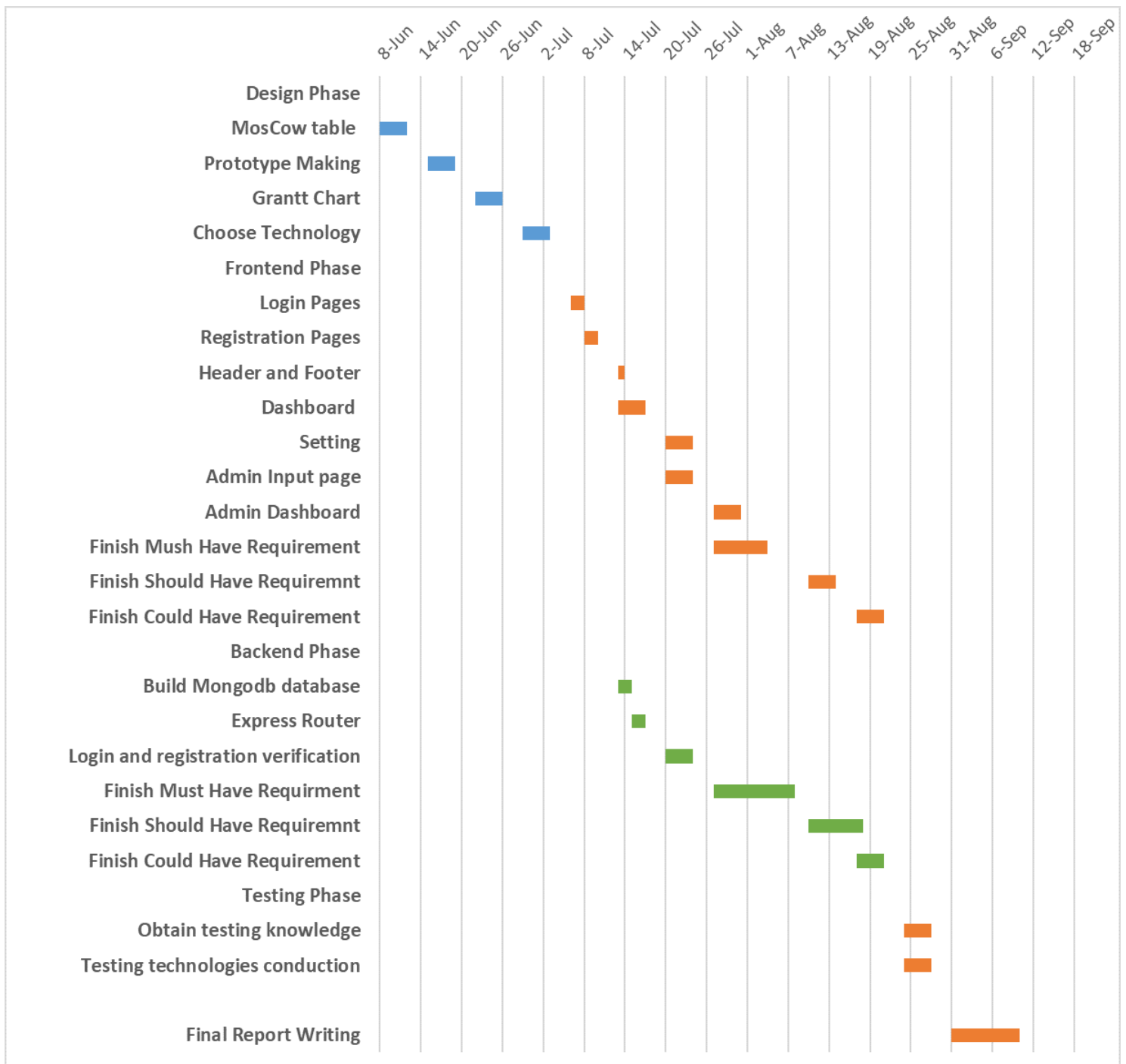


Figure 1.1: Grantt Chart

Chapter 2

Related Research

2.1 Related Application

FreeBusy

FreeBusy Scheduling assistant for google calendar help users find the time when their team members are available, and know whether they are inside or outside the working place, even though teammates are using different calendar system. The main features are shown below:

1. Co-organize meeting with teammates based on availability of each team member
2. Seamless group meeting across different time zone
3. Integrate data from different calendars and web conference service, such as Zoom, Skype for B etc.

Meetingbird

Meetingbird's clean and precise interface makes its easier for user to choose appropriate meeting time. Based on user's preference, it can customize user meeting time, location and time zones.

1. User friendly interface, Clean and professional
2. User can customize meeting time, location and time zones based on preferences
3. Integrate data from different calendars and web conference service, such as Zoom, Skype for B etc.

Woven

Woven has a smart calendar where user can arrange all their appointments. The biggest feature of this application is that it can analyze how user use their time and help them focus on what matters most.

1. summarize how users user their time and prioritize the events and activities that matter most to you, helping them make a better plan.

2. Divide calendar into work and personal life, separating events user must attend such as work meeting, and listing the events listed for information such as a museum opening time.
3. Integrate data from different calendars and web conference service, such as Zoom, Skype for B etc.

These three applications share some features in common: availability to multiple conference server and calendar; user-friendly interface and generating time table based on team member's available time. The review of related application provides some valuable suggestions to the project.

2.2 Related Technologies

Frontend

HTML

HTML5 is referred as HyperText Markup Language for web, which used to structure and define content on the website. The author has used this language in previous project, having a good command of it. HTML5 allows incorporation of styling languages such as CSS and JavaScript, helping developers make the website have a polished look with ease (Nixon 2014).



Figure 2.1: HTML5

CSS

CSS stands for cascading Style Sheets, used to describe how HTML elements are presented on the website. CSS helps developers decorate the website page with ease since can control website layout in multiple pages at once. The author is very familiar with it due to previous project experience.

The primary teaching material consulted to build the application is W3School.com. This helps me obtain basic understanding of how to design website in multiple pages. However, only using CSS is very time consuming, so another styling language will be introduced below.



Figure 2.2: CSS

Bootstrap

Bootstrap is the most popular front-end framework for developing responsive, mobile-first web applications. It includes HTML, CSS and JavaScript and help develops the web with ease via responsive grid system, useful JavaScript plugins etc. The author has a good command of it due to experience in previous projects.



Figure 2.3: Bootstrap

JavaScript

JavaScript, often abbreviated as JS, is a text-based language used both in server and client sides. Unlike CSS and HTML5 giving content and layout to the website, JS provides interactive elements to the web pages(W3school 2017).

David Flanagan's *JavaScript: The Definitive Guide* had been consulted to tackle most of the issues the author met in the project. It contains comprehensive knowledge about JS, but only reading this book is never enough, since it lacks some practical examples. Searching examples from W3school.com and other books is also very important(Flanagan 2006).



Figure 2.4: JavaScript

JQuery

JQuery is a JavaScript library, which simplifies JavaScript programming. It contains most of widely-used javascript functions as ready-to-use methods. Moreover, JQuery plays an important role in creating action in the page, developing Ajax applications etc.(Benedetti and Cranley 2011). In this project, the JQuery will be used in calendar picker, which will help the admin to pick appointment date

Ryan Benedetti's *Head First jQuery: A Brain-Friendly Guide* plays an important roles when the author learnt JQuery for the first time. Moreover, basic concepts and examples of JQuery has been shown in W3school page, which helps me a lot.



Figure 2.5: JQuery

Backend

Nodejs

As for Nodejs, the author had limited experience to it, which had been proven to be a great challenge. The biggest advantage of nodejs is memory efficient, because it uses asynchronous programming. This means that when it opens a file, it is ready to handle next request. When the file has been open, the server will display the file to the user. Moreover, nodejs has a great varieties of packages, which help developers make web page easily(Powers 2012).

To be able to grasp important skills in nodejs, Shelley Powers's *Leaning Node: Moving Server Side* has been consulted. However, in order to learn nodejs quickly, tutorials on W3school, Youtube and stack overflow.com has been learned.



Figure 2.6: Nodejs

Express

Express is a web application framework for nodejs. It also provides easy-to-use routing system to support transfer between pages based on request made. After establishing the express framework, redirection to different pages can easily be achieved by changing the URL end point (Dickey 2014).

The express module is quite easy to learn. Following several tutorials in W3school and Youtube helps the author have a good command of this technique in a short time.



Figure 2.7: ExpressJS

EJS

EJS, Embedded JavaScript template, is a simple templating language . It uses javascript syntax and gets data from database through nodejs. The template tags is very easy to learn, the author has some experience in it.



Figure 2.8: EJS

PHP

PHP stands for Hypertext Pre-processor, used to create static or dynamic web pages (Nixon 2014). It can interact with many databases, such as MySQL. PHP can be embedded in HTML and is supported by web servers.



Figure 2.9: PHP

MySQL

MySQL is one of the most popular relational database management systems in the world. It was initially released 25 years ago by Oracle Corporation. As a relational database, it requires database to have a structural schema.

However, one of the biggest disadvantages of MySQL is that it may perform poorly when too many operations occur at the same time. Web applications with high concurrency levels are better off finding other database management systems.



Figure 2.10: MySQL

MongoDB

MongoDB is a non-relational database, which means that it does not need a strict schema. Unlike MySQL, the database is consisted of columns of certain data type. In MongoDB, users can have different data types in a separate document, so MongoDB is very flexible to users.

Thanks to MongoDB's document-oriented feature, the speed of MongoDB is about 100 times faster than relational databases. It also has high scalability. When the data volume is very large, users can distribute the database to several sites.

However, the biggest disadvantage of MongoDB is that it does not support joins like MySQL. Even though users can use join functionality by connecting tables together, the performance may be affected negatively.

The author has limited experience in MongoDB, but was willing to have a try.



Figure 2.11: MongoDB

Mongoose

Mongoose is an object modelling tool for MongoDB. It models data by defining data schema. Compared with Mongodb, it provides some convenience in creation of data and data validation.

Technologies Selection

As for the front-end programming languages, HTML, CSS, Javascript, JQuery and Bootstrap are selected. HTML was used to provide content to the website. CSS served as a style Sheet language, describing how HTML elements were presented on the website. JavaScript was chosen because it could make the website interactive with users. Some input box in the website needs to check whether it was empty when it is submitted, so JavaScript plays an important role in it. In order to fulfill some requirement such as calendar picker, JQuery was used to add animation to it. Responsive web page is one of "Could Have" requirement in MosCow requirement list. Therefore, Bootstrap must be chosen. Moreover, JQuery plays an important part in some Bootstrap package such as bootstrap date picker.

As for the back-end programming languages, a stack of nodeJS, expressJS, EJS, MongoDB are chosen. Even though the author has some knowledge in PHP, but compared with PHP, nodeJS runs asynchronously. This means that it can handle request and response faster than synchronously programming language. The clinician schedule needs to deal with many requests in certain time period, so nodeJS is a better choice. Moreover, since nodeJS uses the same syntax as JavaScript, it is more convenient to learn. ExpressJS's routing system is easy to learn in a short time and it just needs to download a module from nodeJS, so it is also easy to use. The routing system can redirect web page and set access permissions easily by using middle ware.

As for the reason why MongoDB is chosen. It is admitted that the author has a relatively good command of MySQL. However, compare with MySQL, MongoDB has some advantages. MongoDB has large scalability, which means that it can meet the consumption of data growth. As the clinician team grows, many clinician may check their timetable at the same time. Since mongoDB does not support Joins, some functionalities may be used. Moreover, mongoDB is more and more popular in the internet companies today, having a good command of it helps my future career.

In conclusion, PHP is not chosen because nodeJS is more suitable to this project. NodeJs is asynchronous, so it can handle a large scalability of request. It also has many 3rd party package (node package manager), so it can achieve many requirement.

Chapter 3

Requirement Analysis

3.1 requirement

A well-defined requirement list is essential to the development of clinic scheduling website. Before the first meeting, related applications were consulted to prepare potential need of the client. After the first meetings, the scope of the project, as well as functionalities was well understood. The features and functionaries were further elaborated after a series of email contact with the client.

3.1.1 Persona

Personas are visual representations of the application's users. These fictional generalizations allow the author to understand users better by knowing their job description, daily routine, role as well as interest, motivation goals and so on. It closes the gap between software developer and user's perceptions (Shahri et al. 2016). Therefore, personas serve as a constant reminder to the author what the client really need.

The following personas showing two clinicians' daily work, showing why they need a clinician scheduler during the work and what kind of application can help them best. These two clinicians share the same frustration point: scheduling appointments by themselves a very time-consuming task. This means that they have to spend much more time on administrative work rather than really take time to care about patient. From these two personas, the author can know what drive them to need a clinician scheduling system.

Persona1

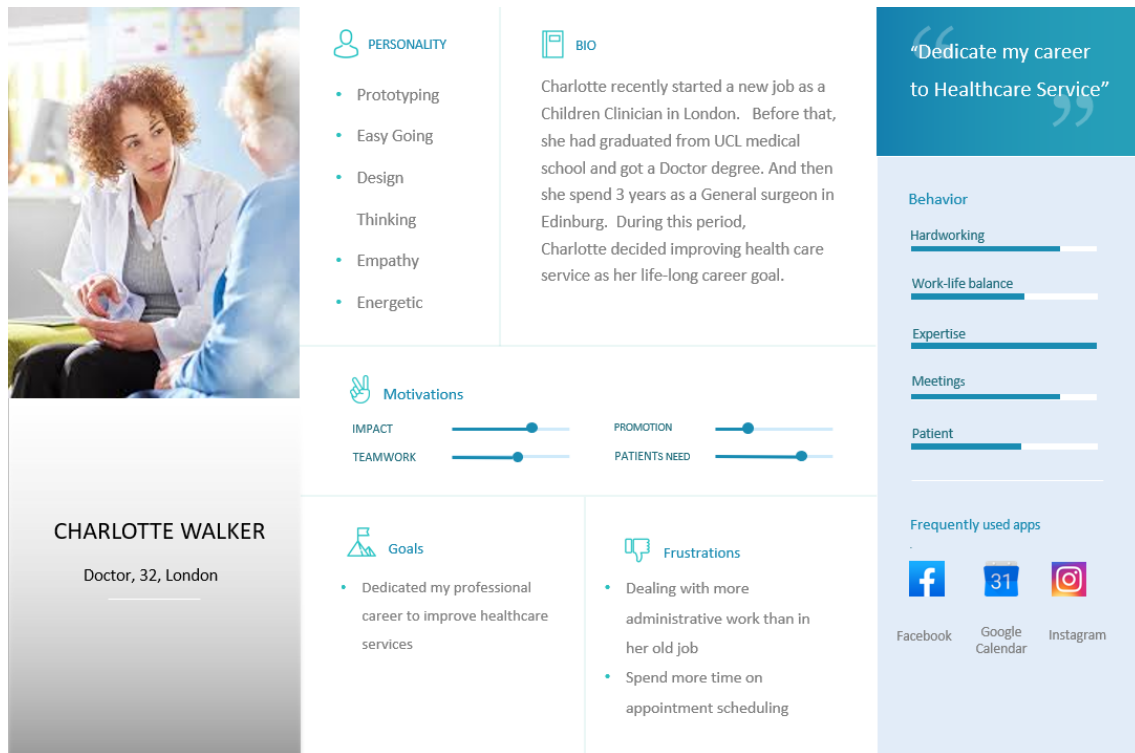


Figure 3.1: Persona1

Persona2

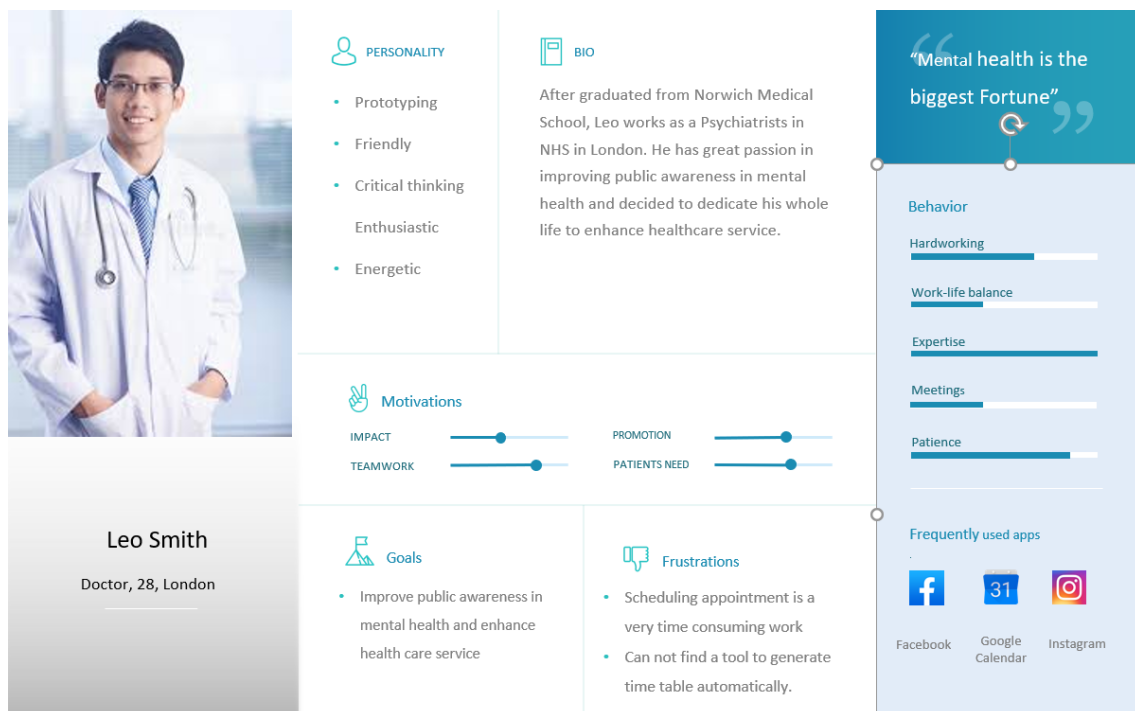


Figure 3.2: Persona2

MosCow method

The MosCow method is a technique used to prioritize tasks. It is widely used in many areas, including business, software development and project management. This method divides requirements into two parts: functional and non-functional requirements. In each part, there are Must have, Should have, Could have, Won't have requirements (Waters 2009).

1. 'Must have' refers to the core requirements that are essential to the success of the web application.
2. 'Should have' refers to the important features which add necessary functions to the application but not as necessary as "Must have" requirements.
3. 'Could have' category includes requirements that useful but not so important. The project can operate well without them. For instance, considering Bank Holidays into schedule generating system is placed into this category, since the number of Bank Holidays can be quite small and admin can adjust the appointment manually if there is a conflict in these days.
4. "Won't have" refers to the features that are too difficult to make for the author's programming ability. Without these requirements, the application can also work well.

The final version of Moscow table for this project has been shown below after a series of consulting and meeting with the client.

Functional Requirements		
ID	Functional Requirements	Priority
R.1	Users (Clinicians and Admins) can register account via emails	Must
R.2	Users (Clinicians and Admins) can login and logout account	Must
R.3	User can retrieve password via verification email if needed	Must
R.4	User can change their password in the profile setting page	Must
R.5	Clinicians can view appointments they have on the main page	Must
R.6	Clinicians can edit note on every appointment record they have	Must
R.7	Clinicians can edit their own appointment records note on appointment edit page	Must
R.8	Clinician can edit his/her own profile information, such as username, photo, maximum working day per month	Must
R.9	Admin can view all the appointment record of the whole clinic team	Must
R.10	Admin can edit and cancel any appointments of the clinic team	Must

Functional Requirements		
R.11	Admin can input patients information such as six-digital identifier, referral date and additional note	Must
R.12	Admin can edit and delete all the patients' record by clicking edit and delete button	Must
R.13	The application allows admin to refresh new appointment by clicking refresh button	Must
R.14	The application can generate time table based on inputted parameters such as referral date	Must
R.15	The application ensure that first appointment occurs within 18 weeks after registration date	Must
R.16	The application ensure that the 2nd, 3rd, 4th appointment occurs once a month	Must
R.17	The application timetables generator ensure that appointments are placed on Monday, Tuesday and Wednesday as the request from the client	Must
R.18	Admin can edit information sector in every appointment Record	Should
R.19	The appointment generator takes clinician's maximum working days number per month into consideration	Should
R.20	Responsive page element adaptive to different type of screen i.e. mobile friendly	Could
R.21	The appointment generator takes bank holidays into consideration	Could
R.22	The application syncs calendar with other calendars, such as Outlook calendar, Google Calendar	Won't
R.23	The application allows user share their own schedule with others	Won't

Non Functional Requirements		
ID	Non-Functional Requirements	Priority
Timeliness		
R.1	The application will show schedule and any changes of appointment without unreasonable delay	Must
Availability		
R.2	The application is web-based, so a users can access it as long as internet is available.	Must
R.3	The application could be mobile friendly, it has resonpsive page that adaptive a variety type of devices	Could
Security		
R.4	The application protects user information via password and email verification	Must
R.5	The application protects users' account information via password encryption	Must
Standard Compliance		
R.6	The application should be suitable to work with Google Chrome	Should

Non Functional Requirements		
R.7	The application should check validation of text of input box(data integrity)	Should

3.1.2 User Cases

Use case diagram provides a high-level visual representation of actors(users) and how they interact with the application(Armour and Miller 2000). After obtain a well-defined requirement list, use case can help us prioritise the functionalities, so the application can carry out important features firstly. The use case diagram has been shown below:

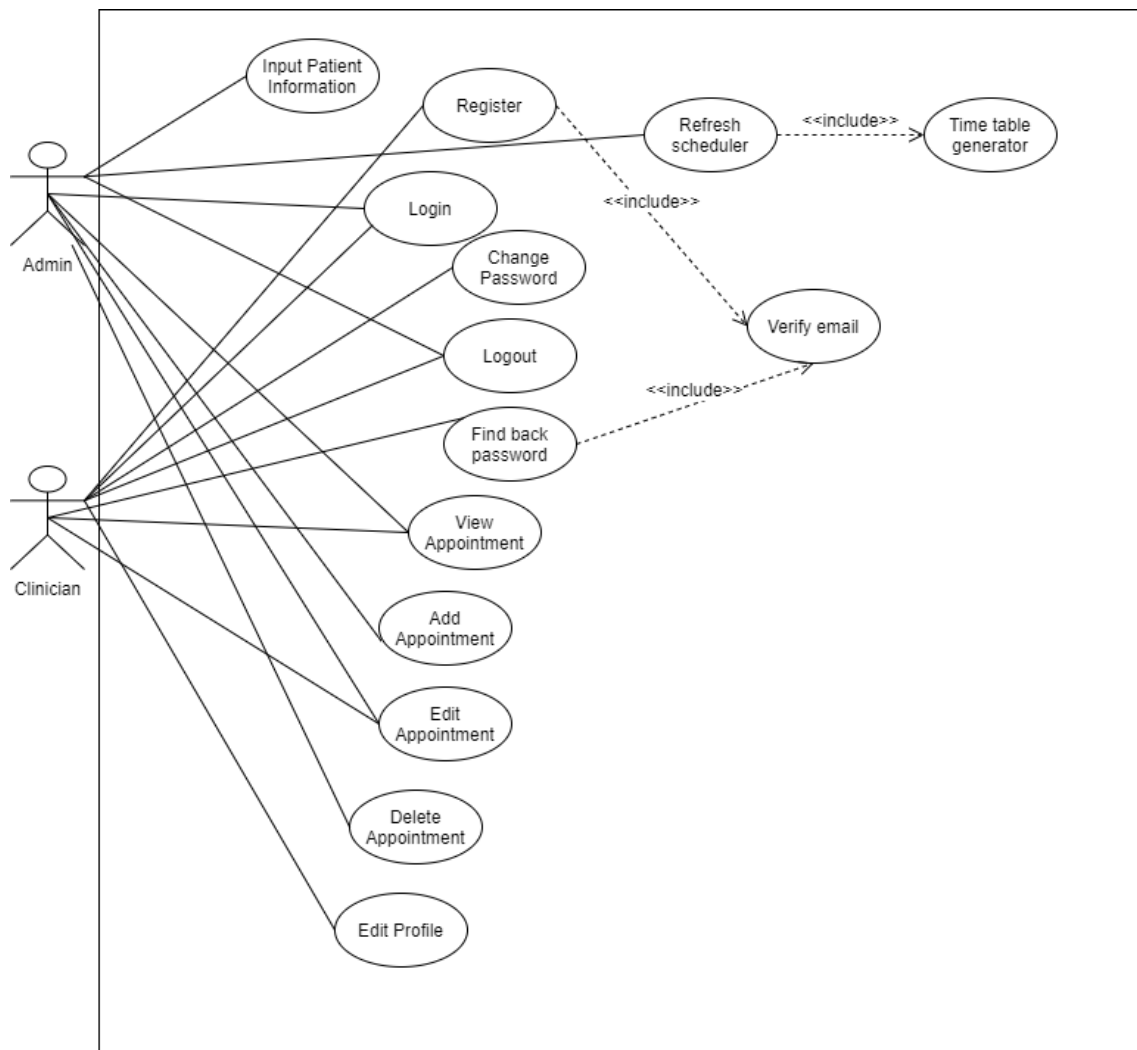


Figure 3.3: Use Case

A list of use case title has been shown below, which is followed by use case specification:

Use case ID	Use Case Description
Admin	

Use case ID	Use Case Description
UC101	Login
UC102	Logout
UC103	Input patient Information
UC104	View Appointment
UC105	Edit Appointment
UC106	Cancel Appointment
UC107	Add Appointment
UC108	Refresh Appointment Table
Clinician	
UC201	Change Password
UC202	Find back password
UC203	View appointment
UC204	Edit Appointment note
UC205	Edit profile information

The use case specification has been attached to the appendix.

Chapter 4

Design and Implementation

4.1 System Architecture

Figure 4.1 outlines an overview of the system architecture, which is divided into three parts: frontend, backend and database. Users directly interact with user interface which is backed up by HTML, CSS, JS and so on. The main function of front end is to provide content and help users to trigger functions in back end through graphical information, such as button.

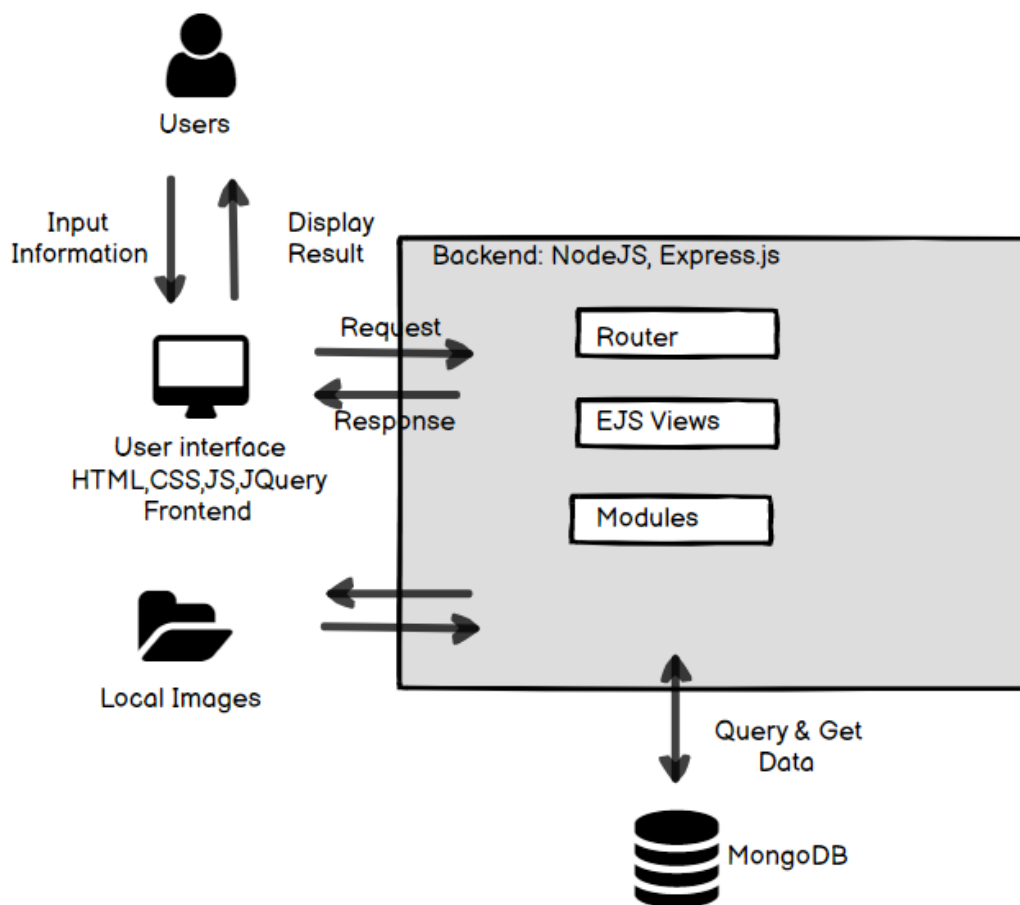


Figure 4.1: System Architecture

The backend has Nodejs, Express.js and MongoDB as back-end technologies. According to the Figure 4.1, the backend servers as data access layer. It deals with the request from the frontend, accesses the data in the database and generates feedback to the frontend through "Response". The following table shows the functions of each component in the application.

Component	Function
User interface	This component includes a great varieties of devices, such as mobile, PC
HTML, CSS, JavaScript, JQuery	HTML provides content to the website; CSS design the style and layout; JQuery and Javascript add interactive elements to the website and verify whether the data inputted by the users is in appropriate format.
EJS	EJS generates HTML markup with plain javascript; NodeJS can also pass data to the front end via EJS
Router	Router can handle http request and carry out page redirection by changing the req.url
Modules	Modules serves as a package of some customized function. When some function needs to be carried out, the function in the package can be called easily.
Nodejs	Nodejs is an open-source environment that allows JavaScript to be executed outside the web browser.
ExpressJS	ExpressJS is a web application framework of nodejs that simplify the creation of web application and server
Local Images Browsers	It stores profiles images of clinicians, but the address of those images are stored in database.
MongoDB	MongoDB database stores all the data of the application and carry out table joins via some functionalities.

4.2 Implementation of important Features and Key modules

ExpressJS

ExpressJs plays an important role in routing system. It provides "app" object in response to HTTP method. For instance, app.get() and app.post() handle corresponding GET and POST request. After it gets a request that has specific route and method, it will specify the callback function to meet the request. Moreover, it also provides various types of middle ware function, such as application-level application, error-handling application etc.

```

var express = require('express');
var app = new express();
var router = express.Router();
app.use(session({
  secret: 'keyboard cat',
  resave: false,
  saveUninitialized: true,
  cookie: {
    maxAge: 1000*60*30
  },
  rolling: true
})))

```

Figure 4.2: Express middleware for session

Body-parser

Body-parser is a package of nodeJs which is available from node package manager(NPM). It extracts incoming request bodies which are available under req.body. Body-parser provides the request formats including JSON and urlencoded.

Multiparty

Multiparty parses http requests and extracts information of request through multiparty.form() method. Compared with Body-parser, Multiparty can also be used to upload files. In this project, multiparty is used to upload images when clinicians upload profile images with the content type of 'multipart/form-data'.

Nodemailer

As for the nodemailer, Nodemailer is a node module which make sending email from computer easier. In this project, Nodemailer is used to email confirmation when user register via email and when user forget password, a link will be also sent to user's email for verification.

Forget Password

When a user forget their password, the application allows them to change their password through email verification. After they click "forget password" button on the login page, the application will redirect to forget password page, where users need to input their unique email. Then the corresponding userId will be found in the database based on the inputted email and then will attached to the end of an URL link which will be sent to the email.

After the user receives the link and clicks on it, they will be redirected to 'change password page' where they need input their new password. In this process, Node-mailer package has been used.

```

router.post('/doforgetpassword', function(req,res){

    var form = new multiparty.Form();
    form.uploadDir='upload';    // the path where you store the images
    form.parse(req, function(err, fields, files) {
        var email = fields.email[0];

        clinicianmodel.find({email:email},function(err,result){

            if (result.length<1){

                res.send("<script>alert('The email not exists, please input again'); location.href='/forgetpassword'</script>")
            }else{

                var id = result[0]._id;
                var id_string = String(id);
                console.log(typeof id_string);

                let code ="http://localhost:8010/findpassword?id="+id_string;
                console.log(code);
                // codeObj[mail]=code;
                // console.log(codeObj);

                Mail.send(email,code,(state)=>{
                    if(state===1){
                        res.send("<script>alert('Verificaion Email has been send successfully,please check your email'); location.href="
                    }else{
                        res.send("<script>alert('Fail to send email'); location.href='/forgetpassword'</script>")
                    }
                })
            }
        })
    })
}

```

Figure 4.3: Function: Find back password

Add and Edit appointment

Admin can add appointment on the "admin input page". After they input appointment information, appointment information will be stored to variables in the figure4.4.

```

var patient_identifier = fields.patient_identifier[0];
var clinician1_name = fields.clinician1_name[0];
var clinician2_name = fields.clinician2_name[0];
var date_string = fields.date[0].split(" ");
var date1 = isodate(date_string[0]+"T"+date_string[1]+":00.00Z");
var note = fields.note[0];

```

Figure 4.4: Add Appointment1

Then the system will fetch clinician ids from the database based on inputted clinician names. And add clinician ids to the appointment tables in the database.

```

DB.insert('appointment',{
  patient_identifier:patient_identifier,
  clinician1_id:new DB.ObjectID(clinician1_id),
  clinician2_id:new DB.ObjectID(clinician2_id),

  date:date1,
  note:note
},function(err){
  if (!err){
    res.redirect('/admin/adminmain')
  }
})

```

Figure 4.5: Add Appointment2

Appointment Generator

Appointment generator is the most difficult part in this project, since there are many restricted conditions for appointment arrangement. Basically, in order to arrange a practical appointment, two elements must be considered: 1, whether the time is appropriate; 2, whether the clinician is available at given time. Figure4.6 shows all conditions needed to be consider when the application generate appointment automatically.


- 
- 1,The first clinician should happen within 18 weeks after patient registers
 - 2, After the first appointment, the patients should be seen once a month
 - 3, The clinic usually only works on Monday, Tuesday and Wednesday.
 - 4, The clinic team does not work on Bank Holidays.
 - 5, Each clinician works part-time on selected weekday
 - 6, Each clinician has maximum number of working days per month;

Figure 4.6: Appointment Generating conditions

As Figure 4.6 shows, condition 1 to 4 is limitation about time. For instance, appointments should happen within 18 months after registration, appointments should not happen on bank holidays, Monday to Wednesday is the clinic "working day". Therefore, generating an appropriate four appointment time is the top priority for the generator.

After four appointment dates are created, it is time to find two available clinicians for the generated time. There are many restrictions for the clinicians: 1, clinicians only work part time, so their appointments should be arranged on their working days. 2, Each clinicians has a limited working day number per month, which they can set when they register. 3, They also need to be free on potential appointment date.

Figure 4.7 shows how appointments of certain patients be generated. Firstly it generated four basic appointment times, whose gap are one month. For each appointment time, it will check whether it is from Monday to Wednesday and whether it's no bank holiday. If yes, it will be added to Available-date list. If not, the appointment time will be added one day. This loop will stop until all four appointment times are generated.

After the 4 appointment times are generated, the system generate a list of available clinician for each appointment time. Then chosen two clinicians randomly from the list and then input the two clinicians' ids and corresponding appointment date into database.

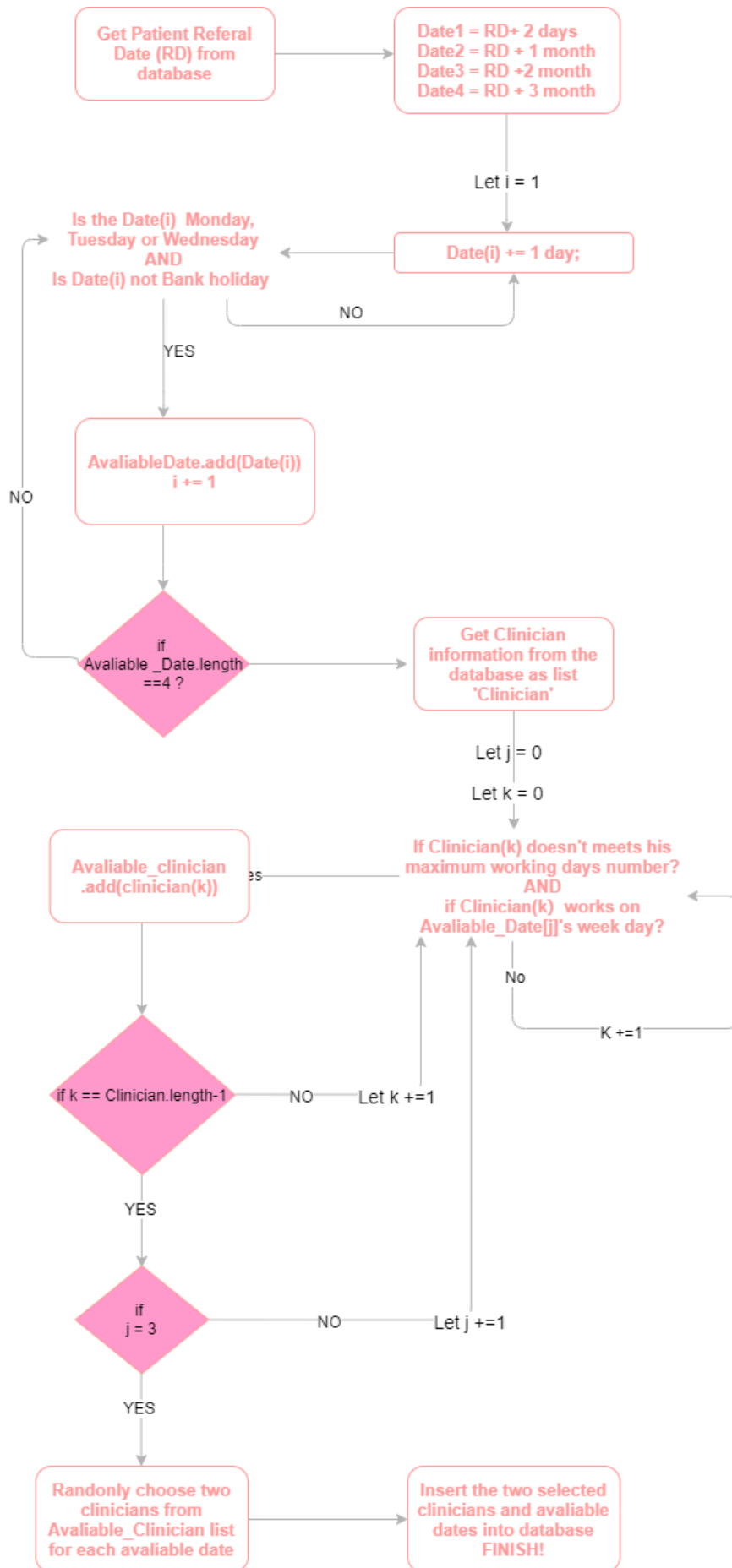


Figure 4.7: Appointment Generator mechanism

4.3 Database Representation

ER diagram

The database is well-structured and represents efficiently. It is first normal form, since it only has single(atomic) valued attributes/columns. It is also second normal form, as it has no partial dependency. Every column of the tables are fully dependent on corresponding primary key. Since the structure do not have any transitive dependency, so it can also called third normal form.

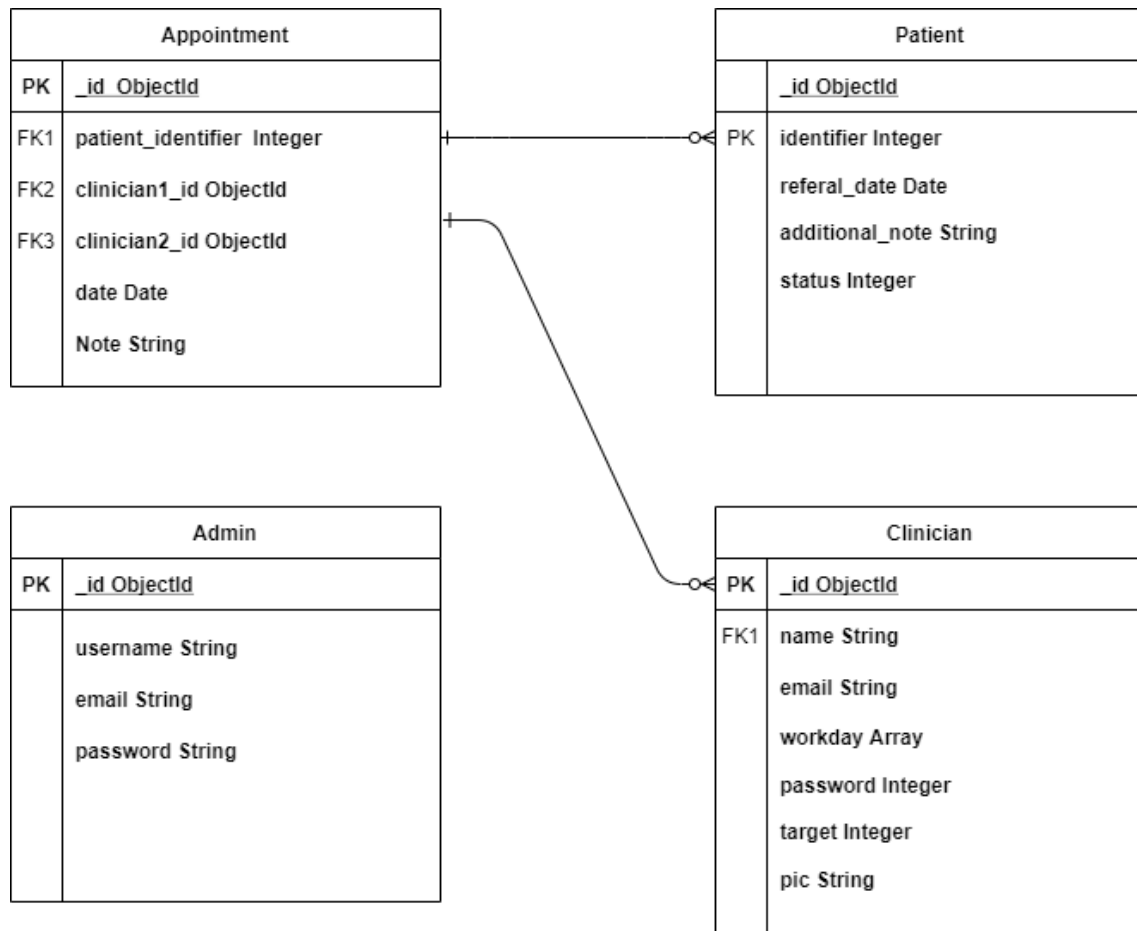


Figure 4.8: Database Representation

Chapter 5

Testing

5.1 Responsive Page Testing

As the client states, the responsive page element is necessary, since group member in the clinic team may use different devices. In order to achieve this feature, Google chrome is used to visualize whether this application is adaptive to different device.

In order to conduct more rigorous responsive test, five types of typical screen will be tested. Google Chrome developer tool and responsive design.com will be used.

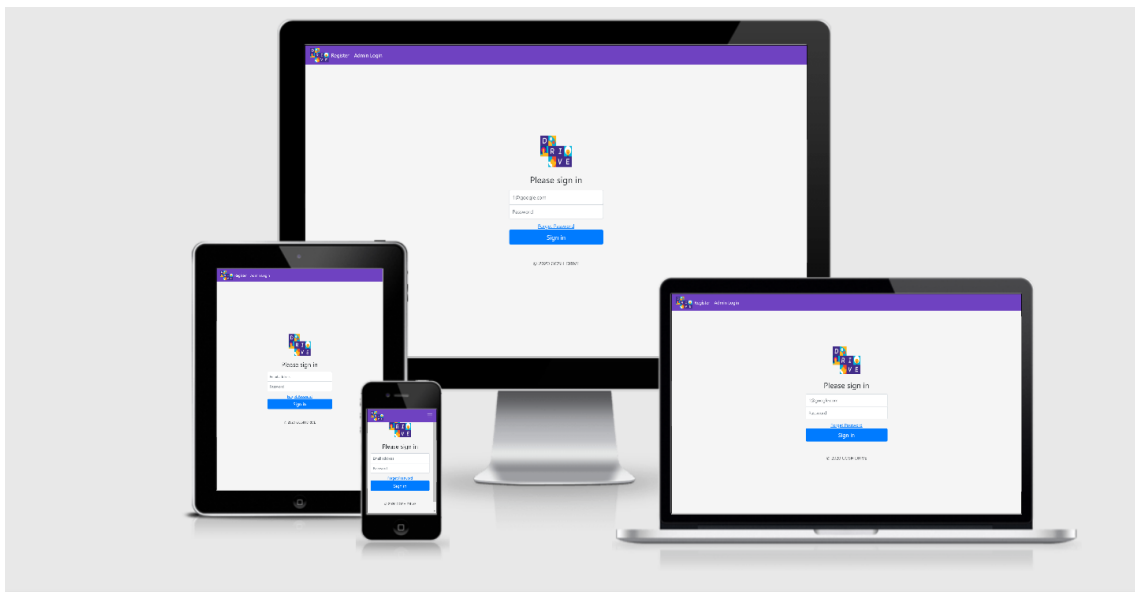


Figure 5.1: Responsive Test

As Table 5.1 shows, there are four types of devices in the responsive test: Smartphone, tablet, laptop and monitor. Smartphone such as iPhone XS, Samsung Galaxy Note has the width range 320-600px; Tablet such as iPad has the width range 650-1300px. Laptop is much more bigger and it has width range 1300-1700px; Monitor usually has width bigger than 1700px;

	Smartphone	Tablet	Laptop	Monitor
Clinician				

Login	YES	YES	YES	YES
Register	YES	YES	YES	YES
Dashboard	YES	YES	YES	YES
Profile Page	YES	YES	YES	YES
Profile edit Page	YES	YES	YES	YES
Appointment Note Edit Page	YES	YES	YES	YES
Admin				
Login	YES	YES	YES	YES
Dashboard	YES	YES	YES	YES
Patient Input	YES	YES	YES	YES
Clinician page	YES	YES	YES	YES
Appointment Edit	YES	YES	YES	YES

??

Table 5.1: Responsive test result

5.2 User Acceptance Testing

User Acceptance Test (UAT) is used to establish users' confidence to the product they will use in the future. It is the last phase of software testing. During UAT, the actual use will use the application in the real word scenario (Pandit and Tahiliani 2015). However, if there is some error during the test, the client may not start the next-stage development.

User Acceptance test was conducted on 10th September, when the clinic team representative used the application and gave the author some advises. For instance, the client suggested that the appointment table should be sorted by the appointment date. This is a very good point for improving user experience.

It is admitted that there is a bug in the appointment edit page. When the admin tries to add a non-existing clinician to the appointment, the application may alert some error on the page. But after the meeting, the bug has been fixed.

As for the result, representatives tested the all the functions of the application. The application is now available for the use of the whole clinic team members.

5.3 Conclusion

From many aspects, the project is a success. The core aim of the project is to help clinician save time when they arrange appointment. The application's timetable generator achieve this goal. According to Table 5.2, all the 'Must have' requirements are met. However, due to the time constrains, two optional requirements is not implemented.

ID	Functional Requirement	Priority	Status
R.1	User(Clinicians and Admins) can register account via email	Must	Finished
R.2	User(Clinicians and Admins) can login and logout account	Must	Finished
R.3	User can retrieve password via email verification	Must	Finished
R.4	User can cahnge their password in the setting page	Must	Finished
R.5	Clinicians can view appointments they have on the main page	Must	Finished
R.6	Clinicians can edit note on every appointment record they have	Must	Finished
R.7	Clinicians can edit his/her own profile information such as username, photo, maximum working day per month	Must	Finished
R.8	Admin can view all the appointment record of the whole clinic team	Must	Finished
R.9	Admin can edit and cancel any apooointment of the clinic team	Must	Finished
R.10	Admin can input sex-digital information, such as six-digital identifier, referral date and additional note	Must	Finished
R.11	Admin can edit and delete all patients' record by clicking edit and delete button	Must	Finished

R.12	The application allows admin to refresh new appointment by clicking refresh button	Must	Finished
R.13	The application can generate timetable based on inputted parameters such as referral date	Must	Finished
R.14	The application ensures that the first appointment generated occurs within 18 weeks after registration date	Must	Finished
R.15	The application ensure that the 2nd, 3rd, 4th appointment occurs once a month	Must	Finished
R.16	The application timetabel generator ensures that the appointments are placed on Monday, Tuesday, Wednesday	Must	Finished
R.17	Admin can view all clincian's information	Must	Finished
R.18	Admin can edit 'note' sector in every appointment sector	Should	Finished
R.19	The appointment generator takes clinicians's maximum working day number per month into consideration	Should	Finished
R.20	Responsive page element is needed i.e. mobile friendly	Could	Finished
R.21	The appointment generator takes bank holidays into consideration	Could	Finished
R.22	The application syncs calendar with other calendars, such as Outlook calendar	Won't	Unfinished
R.24	The application allows user share their own schedule with others	Won't	Unfinished

Table 5.2: Project Result

Result Analysis

The result of the project is quite successfully. Initially, a practical timetable was made and the project followed the timetable step by step.

In the designing phase, the interface design is easy-to-use, which help user schedule appointment with ease. In order to make precise and neat interface, the prototype was sent to the client who gave the author some practical suggestions. Most of the page was made with Bootstrap template, but the author changes the template with CSS.

In the implementation phase, the MosCow Requirement table is also adaptive. In the period meeting, the client also gave us some useful requirements. For instance, the client wants to take clinician's maximum working day per month into consideration, so this suggestion is adopted and implemented.

Nearly all the key requirement of the client has been met. One of the key requirement is to generate appointments for certain patient. Initially, the appointment generator is not perfect enough, it can just generate appointment based on patient's registration date. However, after several updates, it can take all the elements required by the client, such as Bank Holiday, clinician's working weekday, client's maximum working day and so on.

Future Work

Although the application has met all the key requirement, there is still some requirements that fails to meet. The application fails to share clinicians' appointment date with their colleagues, which may cause inconvenience in the work place.

Another area needs to be improve is that the application not provides connection of the google calendar, which means that user can not add the appointment date to the google calendar. User may have to spend some time to add their appointment events on the google calendar manually.

Moreover, some user habits has been ignored by the author. For instance, when the admin check the appointment, they would like to sort the appointments by date. In this way, admin can view the appointment table in a more logical way. However, the application does not provide this function. These functions are important to the application. However, due to the time constraint, many of them has not been implemented.

Chapter 6

Appendix A

6.1 Use Case Specification

UC101- Registration	
Brief Description	User can register an account for the application via email
Actors(s)	User without account
Preconditions	User has an valid email address
Main Flow	<p>1, User clicks register button on navigation bar</p> <p>2, User will be redirected to the register page</p> <p>3, User input user information into the form</p> <p>4, User clicks submit button</p> <p>if no error.</p> <p>5, the application shows "register successfully", user will be redirected to login page</p> <p>Else,</p> <p>6, the error infomation will be shown</p>
Postconditions	Database updated;
Alternative Flows	None

UC102- Login	
Brief Description	The admin logs their user account
Actors(s)	User(Admin and Clinician)
Preconditions	User is not logged in User has an account
Main Flow	1, The participant enter email and password 2, The participant submit the form 3, if the email and password are matched, the participant will be redirect to admin dashboard 4, Else the application will inform error to the user
Postconditions	The user has logged in
Alternative Flows	None

UC103- Logout	
Brief Description	The admin logs out their user account
Actors(s)	Logged User(Admin/Clinician)
Preconditions	User is logged in
Main Flow	1, The user click logout button on the navigation bar 2, The user will be redirect to login page
Postconditions	The user has logged out
Alternative Flows	None

UC104- Input Patient Information	
Brief Description	The admin inputs patient information, including: patient identifier, referral date and notes
Actors(s)	Admin

Preconditions	User is logged in
Main Flow	1, Click input page button on navigation bar to redirect to input page; 2, Input patient information to the boxes 3, Click submit button 4, If no error, inputted information will be shown on table 5, Else, the error will be informed
Postconditions	Database has been updated; The new patient info will be shown on the table
Alternative Flows	None

UC105- View appointment records	
Brief Description	The admin can see all the appointment record on the mainpage
Actors(s)	Admin
Preconditions	User is logged in
Main Flow	1, After looged in, the admin can view appointment table generated by the application automatically
Postconditions	None
Alternative Flows	None

UC106- Edit appointment records	
Brief Description	The admin can edit all the appointment record on the mainpage
Actors(s)	Admin
Preconditions	User is logged in

Main Flow	<p>1, Admin clicks the 'edit' button behind the appointment record he wants to change</p> <p>2, Admin will be redirected to edit page, where they can input updated info</p> <p>3, Click 'Submit' button</p> <p>4, The admin will be redirected to the dashboard, where the updated record will be shown</p>
Postconditions	Database updated; The appointment record on the dashboard will also be updated.
Alternative Flows	None

UC107- Cancel appointment records	
Brief Description	<p>The admin can cancel appointment records on the mainpage</p>
Actors(s)	Admin
Preconditions	User is logged in
Main Flow	<p>1, admin finds out the appointment he wants to delete on the dashboard</p> <p>1, admin click on the 'delete' button behind the appointment record</p>
Postconditions	Database updated; The appointment record on the dashboard will also be updated.
Alternative Flows	None

UC108- Add appointment records	
Brief Description	The admin can add appointment records on the mainpage
Actors(s)	Admin
Preconditions	User is logged in
Main Flow	<ol style="list-style-type: none"> 1, admin input new appointment information into the form on the mainpage 2, admin clicks on the submitted button 3, if no error, the appointment table will be updated; 4, Else, the error will be shown
Postconditions	Database updated; The appoinment record on the dashboard will also be updated.
Alternative Flows	None

UC109- Refresh appointment record table	
Brief Description	The admin can refresh appointment record table, so new appointment info about newly inputted patient will be shown
Actors(s)	Admin
Preconditions	User is logged in
Main Flow	<ol style="list-style-type: none"> 1, After login in, the admin be redirected to user info page. 2, user clicks on the 'refresh' button 3, if no new patiet exists, new appointment information about new patient will be added to the record table; 4, Else, no new appointment info will be shown

Postconditions	Database updated; New appointment info about new patient will be shown;
Alternative Flows	None

UC201- Change password	
Brief Description	User can change password on the profile page
Actors(s)	User clinician
Preconditions	User is logged.
Main Flow	<p>1, User inputs the old password;</p> <p>2, User inputs the new password and confirm new password;</p> <p>3, User clicks submit button</p> <p>if error, 4, the error will be shown,user needs to input again</p> <p>Else, information box will show: change password successfully.</p>
Postconditions	Database updated;
Alternative Flows	None

UC202- Find back password	
Brief Description	User can find back password via email if he forgets the password
Actors(s)	User (Clinician)
Preconditions	The user has account and remember the email

Main Flow	<p>1, User click 'forget password' link</p> <p>2, User will be redirected to the 'email verification' page;</p> <p>3, User need to input email, and click 'send' to send verification button'</p> <p>If the email is not valid,</p> <p>4, related error information will be shown</p> <p>Else,</p> <p>5, User need to login in their email and click the link sent by the application</p> <p>6, User will be redirected to the 'change password' page</p> <p>7, User needs to input new password and click submit</p> <p>8, the application will show 'change password successfully'</p>
Postconditions	Database updated;
Alternative Flows	None

UC203- View appointment	
Brief Description	User can see the appointment information on the clinician's dashboard
Actors(s)	User (Clinician)
Preconditions	The user has been logged
Main Flow	<p>1, After login in, user will be redirected to the dashboard;</p> <p>2, User can see the appointment table of his/her own, which is shown by application automatically</p>
Postconditions	None
Alternative Flows	None

UC204- Edit Appointment Note	
Brief Description	User can edit the additional note for his appointment record
Actors(s)	User (Clinician)
Preconditions	The user has been logged
Main Flow	<ol style="list-style-type: none"> 1, User finds out the record needed be edited. 2,Click on the 'Edit Note' button; 3, User will be redirected to the 'note edit' page, and input the note the user wants to edit 4, click on the 'submit' button
Postconditions	<ol style="list-style-type: none"> 1,Database updated 2, User will be redirected to the dashboard; 3, the new note will be shown on the record
Alternative Flows	None

UC205-Edit Profile Information	
Brief Description	User can edit the personal informaion on the profile setting pages
Actors(s)	User (Clinician)
Preconditions	The user has been logged
Main Flow	<ol style="list-style-type: none"> 1, Clicking on the 'Profile button on the navigation bar 2, User will be redirecte to the profile setting page 3, Click on the button behind the information the user wants to change; 4, User will be redirected to the profile edit page; 5, Inputs or edits the information to the edit box; 6, Click on the 'submit' button

Postconditions	1, Database updated 2, User will be redirected to the profile page; 3, The new personal information will be shown
Alternative Flows	None

6.2 User Manual

Account Credentials

Clinician Account information:

- User Email: ucabgz5@ucl.ac.uk
- Password: 123

Admin Account information:

- User Email: admin1@google.com
- Password: 123

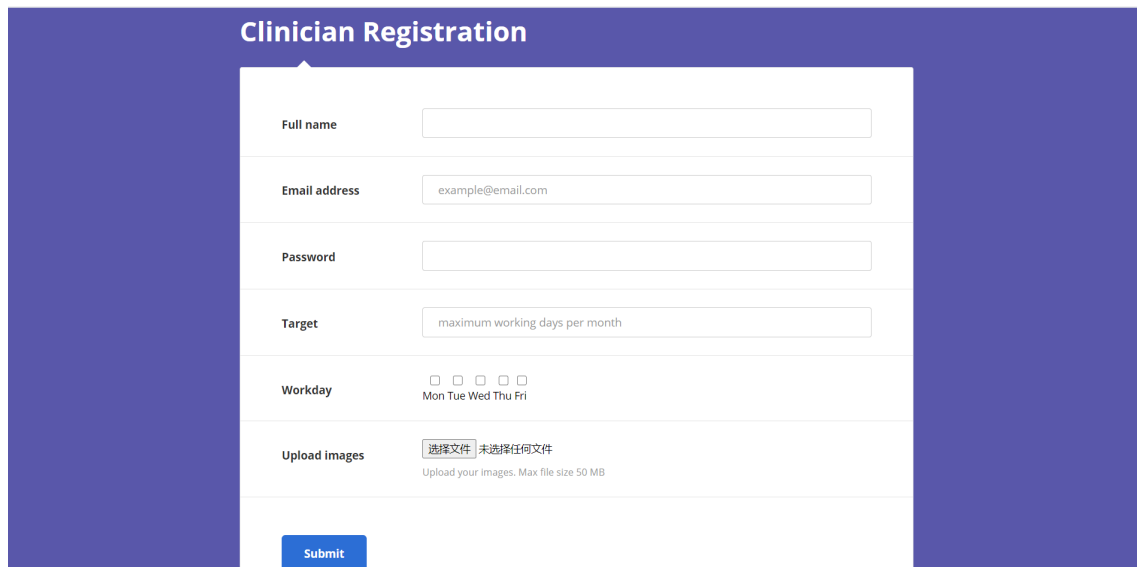
Youtube link: <https://youtu.be/5BZAFA5q9yo>

Website page demo

Firstly, User can register an account in the register page by clicking the register link on the login in page.

The screenshot shows a web application interface for 'GOSH DRIVE'. At the top, there is a purple navigation bar containing the application logo and two links: 'Register' and 'Admin Login'. The main body of the page is light gray. In the center, there is a login section titled 'Please sign in'. This section includes two text input fields labeled 'Email address' and 'Password'. Below these fields is a blue link labeled 'Forgot Password'. At the bottom of the login section is a prominent blue button labeled 'Sign in'. The footer of the page, centered, displays the copyright notice '© 2020 GOSH DRIVE'.

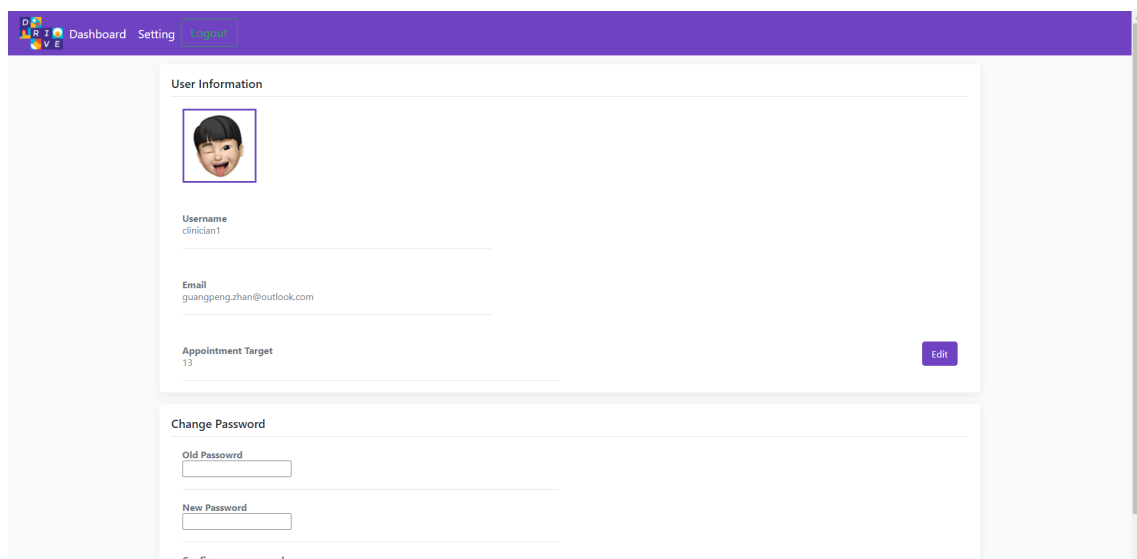
Figure 6.1: Clinician Login



The image shows a 'Clinician Registration' form on a purple background. The form is a white box with the following fields: 'Full name' (text input), 'Email address' (text input with 'example@email.com' pre-filled), 'Password' (text input), 'Target' (text input with 'maximum working days per month' pre-filled), 'Workday' (checkboxes for Mon, Tue, Wed, Thu, Fri), and 'Upload images' (a file selection button labeled '选择文件' and '未选择任何文件', with a note 'Upload your images. Max file size 50 MB'). A blue 'Submit' button is at the bottom.

Figure 6.2: Clinician Login

After login, the user will be redirected to the profile page.



The image shows a 'Clinician Profile' page. At the top is a purple navigation bar with links for 'Dashboard', 'Setting', and 'Logout'. The main content area has a 'User Information' section with a profile picture placeholder, 'Username' (clinician1), 'Email' (guangpeng.zhan@outlook.com), and 'Appointment Target' (13) with an 'Edit' button. Below this is a 'Change Password' section with fields for 'Old Password', 'New Password', and 'Confirm new password'.

Figure 6.3: Clinician Profile page

By clicking the 'Dashboard' link on the navigation bar, user will be redirected to the dashboard.

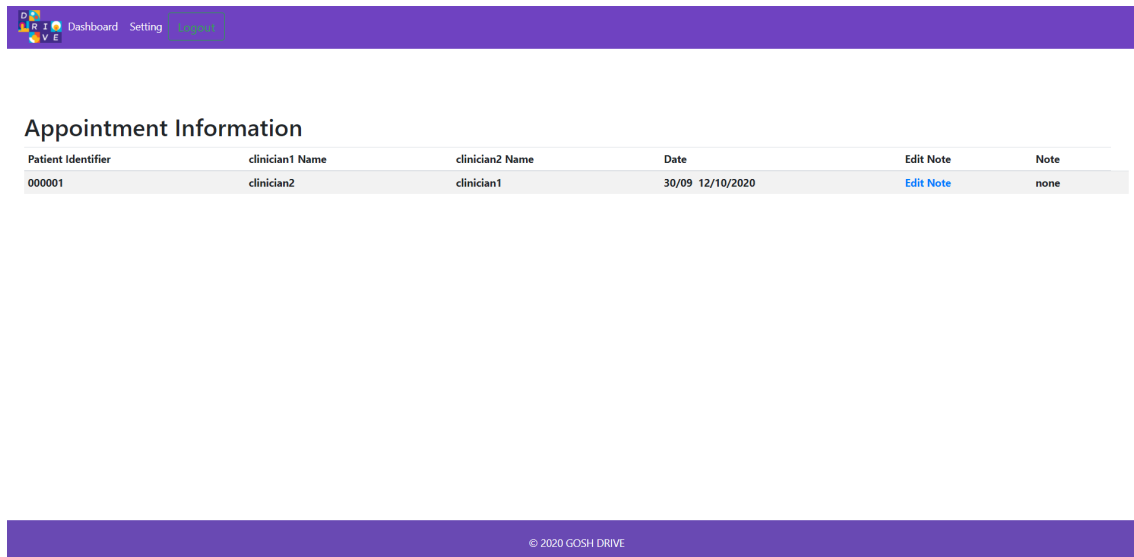


Figure 6.4: Clinician Dashboard

As for the admin page, User can login via admin login page:

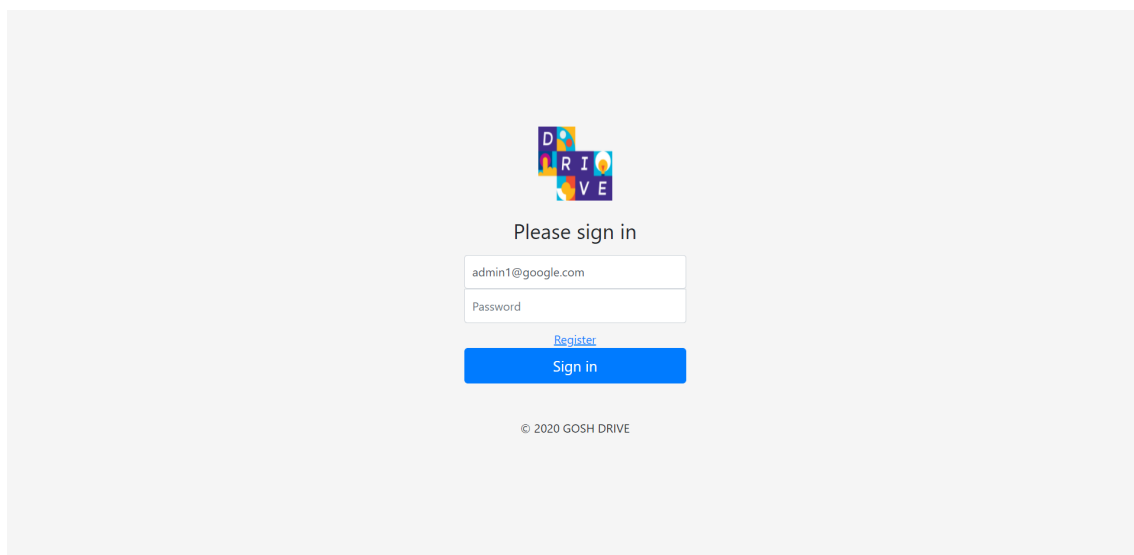


Figure 6.5: Admin Login Page

After login, admin will be redirected to the admin dashboard, where all the clinic team's appointments will be shown:

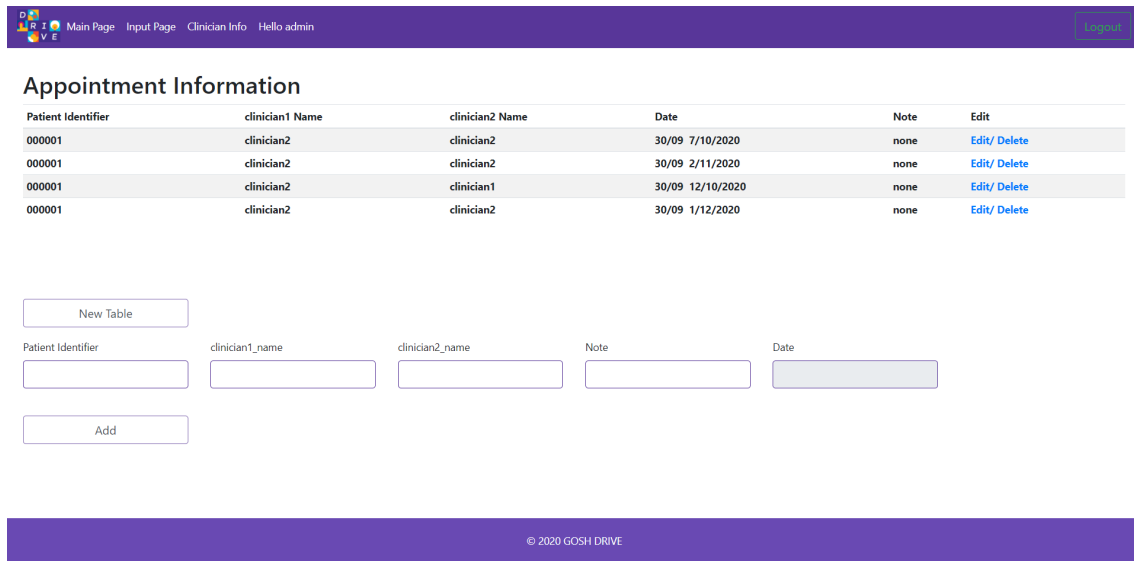


Figure 6.6: Admin Dashboard

By clicking the 'edit' button, user will be redirected to the appointment edit page of corresponding appointment:

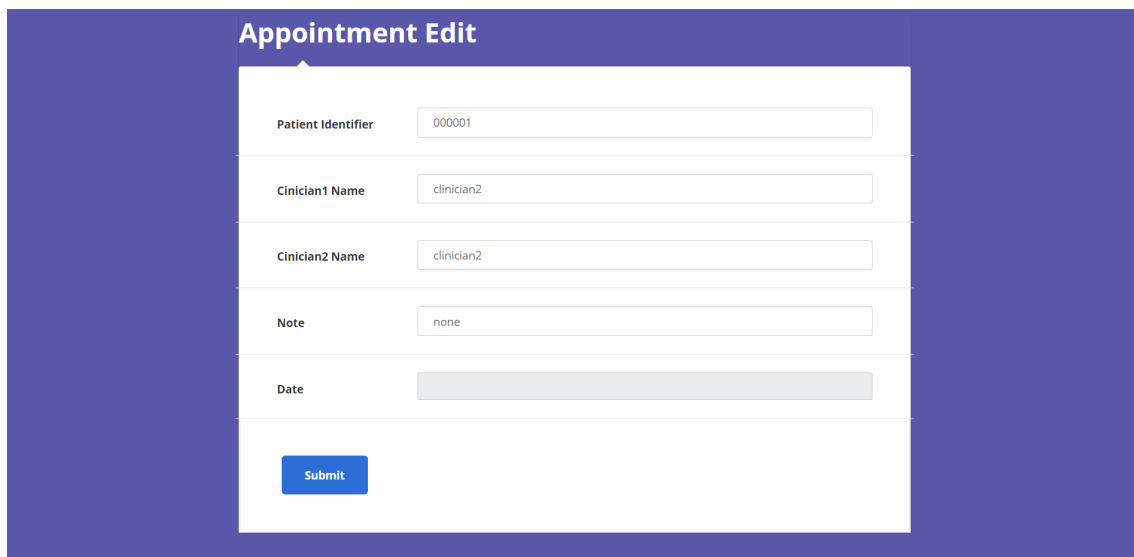


Figure 6.7: Admin Appointment Edit page

By clicking the Input page link on the navigation bar of the dashboard page, the admin will be redirected to the Patient information input page:

Digital Identifier	Referral Date	Additional Information	Delete
000001	9/9/2020	none	Edit/ delete

Digital Identifier:

Referral Date:

Additional Note:

© 2020 GOSH DRIVE

Figure 6.8: Patient Information Input page

By clicking the 'Edit' buttons behind patient records, admin can edit the corresponding patient information:

Patient Info Edit

Identifier:

Referral Date:

Note:

Figure 6.9: Patient Information Edit page

By clicking on the "Clinician Info" button on the navigation bar, the admin will be redirected to the clinician information page, where all the clinician information will be shown:

Children Information			
Name	Email	Workday	Target
clinician1	guangpeng.zhan@outlook.com	1,3,4	13
clinician2	ucabgz5@ucl.ac.uk	1,2,3	12

Figure 6.10: Patient Information Edit page

6.3 Source Code Examples

A small part of the source code will be shown below. Others will be submitted as zip file.
admin.js

```
1  var express = require('express');
2
3  var router = express.Router();// modularization,mounting,router, handle
4  var url = require('url')
5
6  // Back end router deal with all back end router
7
8  var login = require('./admin/login.js');
9  var admininput = require('./admin/admininput.js');
10 var register = require('./admin/register.js');
11 var adminmain = require('./admin/adminmain.js');
12 var clinician_info = require('./admin/clinician_info.js')
13 console.log('line11_admin.js');
14
15
16 //Customized middleware,check the status of login
17 router.use(function(req,res,next){
18   var pathname = url.parse(req.url).pathname
19
20   if (pathname=='/login' || pathname=='/login/dologin' || pathname=='/register' || pathname=='/register/doregister' || pathname=='/register/getCode'
21     ||pathname=='/doforgetpassword' ||pathname=='/findpassword' || pathname=='/dofindpassword' || pathname=='/admin/login' ||pathname=='/admin/login'
22
23     next();
24   }else{
25
26     if(req.session.userinfo && req.session.userinfo.username != ''){ // check whether has logged
27
28       console.log('line30'+req.session.userinfo.username);
29
30       next();
31     }else{
32       console.log('line37 admin.js'+ req.url);
33       res.redirect('/admin/login');
34     }
35   }
36 }
37 })
```

Figure 6.11: Admin.js(1)

adminmain.js. Some functions are abbreviated.

```

1 var express= require('express');
2 var router = express.Router();
3 var DB = require('../modules/db.js');
4 var Appointment_get = require('../modules/appointment_generate_new.js')
5 var multiparty = require("multiparty");
6 var isodate = require("isodate");
7 const { Db } = require('mongodb');
8 var mongoose =require('mongoose');
9
10
11 var clinicianmodel = require('../modules/clinician.js')
12 var appointmentmodel = require('../modules/appointment.js')
13
14
15
16 router.get('/',function(req,res){
17
18     var userinfo = req.session.userinfo;
19
20     var list = [];
21
22
23
24
25     appointmentmodel.aggregate([
26
27         {
28             $lookup:
29             {
30                 from:"clinician",
31                 localField:"clinician_id",
32                 foreignField:"_id",
33                 as:"clinician1"
34             }
35
36
37     ],{

```

Figure 6.12: adminmain.js(1)

```

65         res.render('admin/admin_mainpage',{
66             list:docs,
67             userinfo:userinfo
68         });
69         //console.log(JSON.stringify(docs));
70     })
71
72
73
74
75 })
76
77
78 > router.get('/generatetable', function(req,res){ ...
104 })
105
106
107 > router.get('/appointmentedit', function(req,res){ ...
161 })
162
163
164 > router.post('/appointmentdoedit',function(req,res){ ...
239 })
240
241 > router.get('/delete',function(req,res){ ...
254 })
255
256
257 > router.post('/add',function(req,res){ ...
322 })
323
324
325
326 module.exports = router;

```

Figure 6.13: adminmain.js(3)

Bibliography

- Armour, Frank and Granville Miller (2000). *Advanced use case modeling: software systems*. Pearson Education.
- Benedetti, Ryan and Ronan Cranley (2011). *Head First JQuery: A Brain-Friendly Guide*. " O'Reilly Media, Inc."
- Dickey, Jeff (2014). *Write modern web apps with the MEAN stack: Mongo, Express, AngularJS, and Node. js*. Pearson Education.
- Flanagan, David (2006). *JavaScript: the definitive guide*.
- Nixon, R (2014). *Learning PHP, MySQL, JavaScript, CSS & HTML5: A Step-by-Step Guide to Creating Dynamic Websites*. 3rd O'Reilly Media.
- Pandit, Pallavi and Swati Tahiliani (2015). "AgileUAT: A framework for user acceptance testing based on user stories and acceptance criteria". In: *International Journal of Computer Applications* 120.10.
- Powers, Shelley (2012). *Learning Node*. " O'Reilly Media, Inc."
- Shahri, Alimohammad et al. (2016). "Engineering software-based motivation: a persona-based approach". In: *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*. IEEE, pp. 1–12.
- W3school (2017). *w3school. com HTML*.
- Waters, Kelly (2009). "Prioritization using moscow". In: *Agile Planning* 12, p. 31.