
COMP9900 Project Report



Team: Zava

Lab: W13B

z5302785 Zhetai Jiang z5302785@ad.unsw.edu.au Scrum Master
z5384025 Zhenwei Wu z5384025@ad.unsw.edu.au Backend Developer
z5404627 Haodong Ke z5404627@ad.unsw.edu.au Backend API Layer
z5430604 Zhou Sha z5430604@ad.unsw.edu.au Frontend Developer
z5305832 Yalin Li z5305832@ad.unsw.edu.au Frontend Developer

Submission date: 17/11/2023

Content

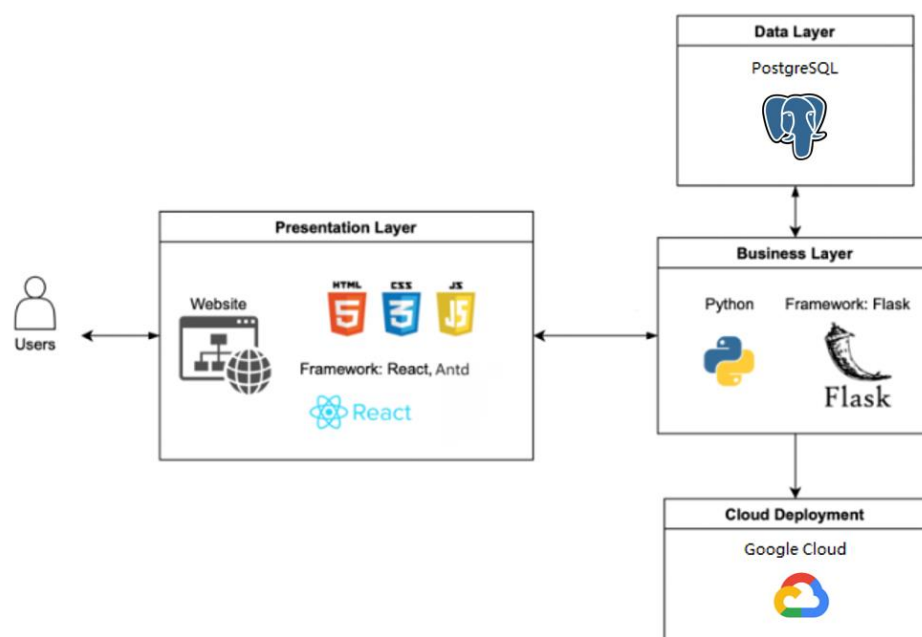
Overview	3
System Architecture	3
Page Structure	4
Page Flow Diagram	4
Database Design	5
Project Objectives	5
Functionality Details	8
Login page	8
Register Page	8
Appointments Page	9
Record Page	10
Statistic Page	11
Settings Page	12
Create Page	12
Edit Page	14
Technologies Descriptions	15
Third-party functionalities	15
Frontend	15
Backend	15
Challenges and Difficulties	16
Frontend	16
Backend	17
Installation and Prerequisites	18
User Documentation	19
Download and deploy the project	19
Login Page	20
Register Page	21
Appointments Page	21
Appointments Details Page	22
Add Appointments Page	23
Create new patient Page	24
Edit Appointment Page	24
Record Page	24
Statistic Page	25

Settings Page	26
References	28

Overview

When it is necessary to access the appointment schedule of general practitioners, it can only be done at the practicing institution or remote internship server, if any. There is a need for a more universal solution that allows general practitioners to easily view and manage their appointment schedules through a web browser, whether on desktop or mobile devices. The proposed project aims to fill this gap by developing a web-based application, making it easy to access the appointment scheduling medical system (www.bpsoftware.net) in accordance with widely applied best practices. In our project, doctors can register and log in to the system, query and support changing their daily schedule, and add patient appointments. In addition, they can also change it at any time according to the patient's situation.

System Architecture



Presentation layer

The presentation layer is what a system user sees or interacts with. Users will directly access our website and use the function that we provide on the interface. This layer can render website pages by using frontend frameworks and tools to provide a user-friendly interface. We used React as the framework of our website, which is one of the most popular frameworks in the industry. ...

Business layer

The business layer contains our business logic. This layer can accept user requests from the upper layer (presentation layer) through API, query from database in data layer, and finally process and return result data in correct format to upper layer.

Data layer

The data layer is an important component, storing all the data needed for the project and interacting with the business layer. It is responsible for data storage and management.

Cloud

The cloud platform offers many functionalities such as computation, storage, and more. In our project, we only use the storage feature. Users can access and manage this data over the internet, providing great convenience and efficiency.

Deployment

Page Structure

Login page: Login for an existing user's account.

Register page: Register a new account for a new user.

Appointments page: Show all the information of every appointment of the logged-in user.

Create page: Create a new appointment or a new medical record for a new patient.

Record page: Show specific patient history and future appointments.

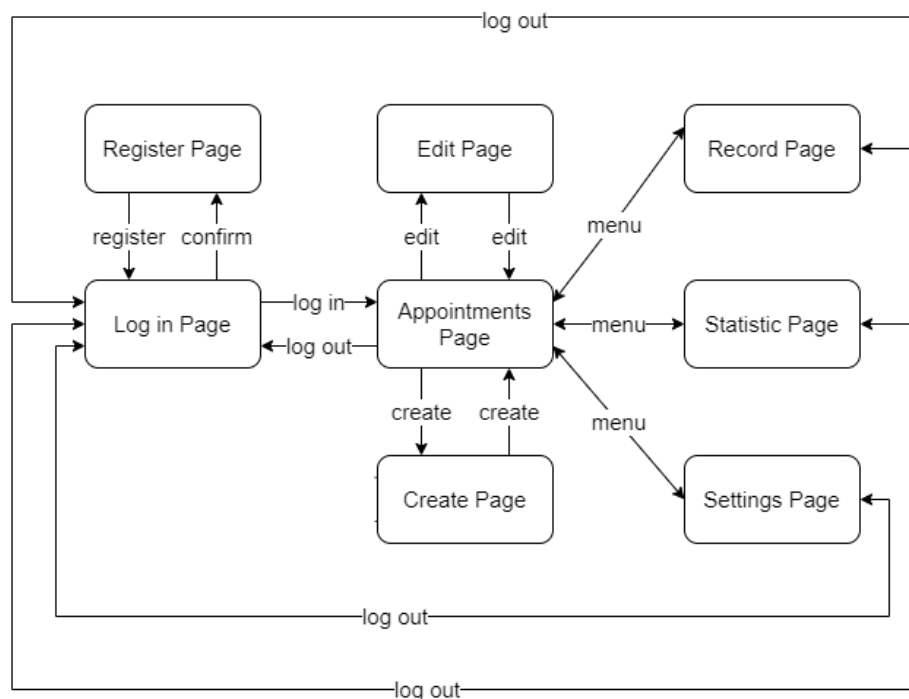
Statistic page: Show the appointment book statistics.

Edit page: Edit an existing appointment.

Settings page: Configure personal settings for every user.

Page Flow Diagram

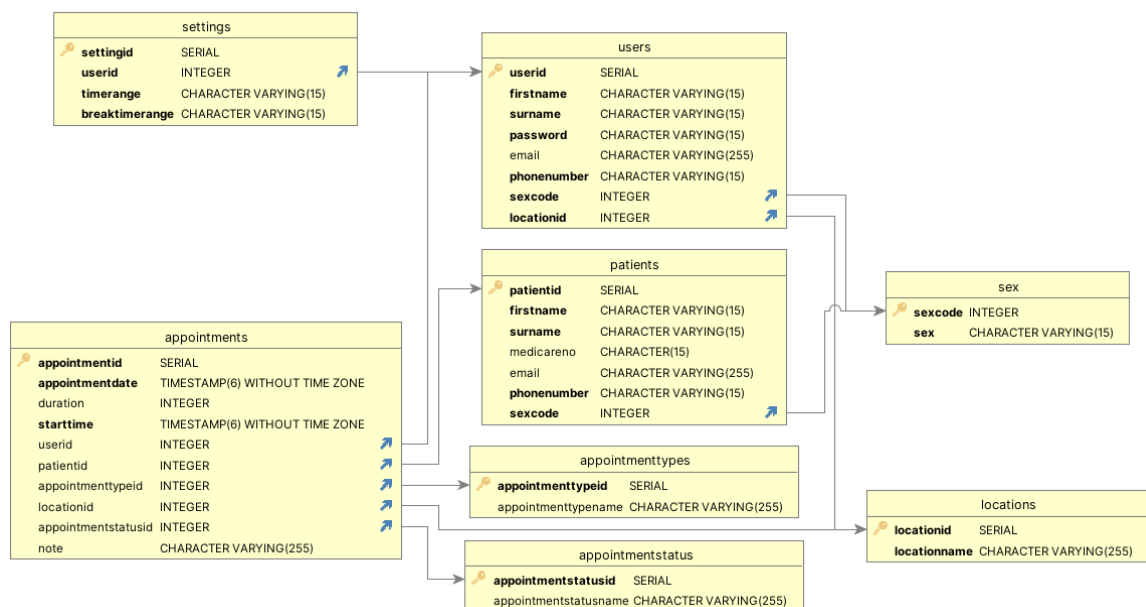
There are eight pages in our web-based application. The picture below shows their relationship.



Database Design

About our database, there is something that need to mention. At first, client want us to use the BP Sample Database he provided, but the drawback of that database is that we only have read permission. So following our lecture's advice and under client's permission, we use postgresSQL to create this dummy database. Although our database is build based on client's one, but can't simply switch to BP sample database because the name of tables and columns are different.

There are 8 tables in our database. The 'users' table stores information of users, i.e. doctors or admin, including password, which can be used to login. The 'patients' table stores information of patients. The 'appointmenttypes' table stores different appointment types, like 'Phone Consultation'. The 'appointmentstatus' table stores different appointment status, like 'With Doctor'. The 'locations' tables store location of the clinic. The content of these three tables (appointmenttypes, appointmentstatus and locations) are copied from BP Sample Database. The 'sex' table stores two records, male and female. The 'settings' table stores user settings. The 'appointments' table is the most important table, stores every appointment doctor created.



Project Objectives

We have 3 stages to finish our projects.

Stage1:

We focus on basic functions and frame. We should complete the initial design of the front-end interface and apply some of the APIs (application programming interfaces). Meanwhile, the backend should be capable of performing SQL data retrieval and fetching data from Best Practice.

PO1: As a doctor, I want to type my username, location and password, so that I can log in to the website about appointments.

PO 2: As a doctor, I want to type my username, location and password twice, so that I can register on the web for appointments.

PO 3: As a doctor, I want to log in my account to use subsequent functions.

PO 4: I would like to check the schedule for a certain day to arrange work accordingly.

PO 5: As a doctor, sometimes patients need to change their appointment time temporarily, and I

need to receive their information to challenge myself to adjust my schedule.

PO 6: As a doctor, I understand that some symptoms are quite special, and I would like to search for appointment information for these people.

PO 7: As a doctor, sometimes I may be busy and need to share my patients with other doctors to adjust the time.

PO 8: As a doctor, I would like to check my available reservation times because the government has restrictions on this.

Stage2:

We implement all the features requested by the client and have a smoothly functioning front-end as well as back-end.

PO9: As a doctor, I need to check appointments for specific situations.

PO10: As a doctor, I need to find my available time slot to schedule the appointment.

PO11: As a doctor, I can see that patients without medical insurance can adjust their treatment policies appropriately.

PO12: As a doctor, I need to know which patients have emergency symptoms.

PO13: As a doctor, I need to be able to log out my account at certain critical moments.

PO14: As a doctor, I want to know as much as possible about the patient's future arrangements at the hospital.

PO15: As a doctor, to better understand patients, I would like to know their previous records.

PO16: As a doctor, I want to check my daily schedule at any time.

PO17: As a doctor, I need to adjust my schedule myself.

PO18: As a doctor, sometimes I hope to make temporary adjustments to arrangements, such as deleting certain appointments.

PO19: As a doctor, I want to edit existing appointments to adjust the time or other information.

PO20: As a doctor, I want to be able to add notes and attachments to appointments to record some important information.

PO21: As a doctor, I understand that some patients are special, and I would like to add special appointments.

PO22: As a doctor, I wish I could add a new appointment at any time.

PO23: As a doctor, I need to check the patient's past medical history related to their condition to make a better judgment.

Stage3:

We finished all assessments in the Moodle successfully and improved and optimized it.

PO24: As a doctor, I wish the operation interface could be more good-looking.

PO25: As a doctor, I wish I could create new cases for new patients.

PO26: As a doctor, I wish I could see the statistic of appointments' sum during one time range.

PO27: As a doctor, I wish I could see the statistics of the sum of different status appointments during one time range.

PO28: As a doctor, I wish I could set the personalization's configuration of the page.

PO29: As an administrator, I wish I could check every doctor's appointment.

PO30: As an administrator, I wish I could set a break time for relaxing.

Functionality Details

Login page

Related to: PO1, PO3 and PO4

The Login page is for users to log in their account by selecting the username, typing the password and choosing a location. It could be visited through the URL address “http://localhost:3000/#”.

The button “Log in” and clickable text “Register now” with blue color show on this page. Meanwhile, the password is not visible due to security reasons.

Pressing clickable text “Register now” will navigate to the Register page.

All possible conditions for pressing button “Log in” are listed in the table below:

Condition	Result
The username, the location and the password match.	Login successfully. A notification “Login Successful! Hello XX YY” (‘XX’ means the first name of the patient and ‘YY’ means the last name of the patient), will be shown in the top-right corner. And the route navigates from the login page to the appointments page.
The username and the password do not match.	Login failed. A notification “User And Location Does Not Match!” will be shown in the top-right corner. And the website will stay on the Login page.
The password is wrong.	Login failed. A notification “Wrong Password!” will be shown in the top-right corner. And the website will stay on the Login page.
The username, the location or the password is empty.	Login failed. A notification “Please input all information!” will be shown in the top-right corner. And the website will stay on the Login page.

The frontend file of the Login page shows below:

Name	Address
Login	/code_1/front-end/src/login/index.jsx

The API for the Login page shows below:

API name	URL	Description
login	http://127.0.0.1:5000/login	Login an existing account
getAllUsers	http://127.0.0.1:5000/GetAllUsers	Get all users’ name
getAllLocation	http://127.0.0.1:5000/GetAllLocatio	Get all location

Register Page

Related to: PO2

The Register page is for users to register a new account by inputting the first name, surname, password, confirm password, email and telephone as well as choosing a location and a gender. It

could be visited through the URL address “http://localhost:3000/#/register”.

The buttons “Confirm” and “Cancel” show on this page. Meanwhile, the password and confirm password is not visible due to security reasons.

All possible conditions for pressing button “Confirm” are listed in the table below:

Condition	Result
All information is correct, and the user's name does not exist in the database.	Register successfully. A notification “Registration successful!” will be shown in the top-right corner. And the route navigates from the Register page to the Login page.
The format of the telephone number is wrong	Register failed. A notification “The format of the telephone is incorrect!” will be shown in the top-right corner. And the website will stay on the Register page.
The format of the email is wrong.	Register failed. A notification “The format of the email is incorrect!” will be shown in the top-right corner. And the website will stay on the Register page.
One of the pieces of information is empty.	Register failed. A notification “Please input all information!” will be shown in the top-right corner. And the website will stay on the Register page.
All information is correct, but the user's name exists in the database.	Register failed. A notification “User Already Exists!” will be shown in the top-right corner. And the website will stay on the Register page.
The password does not match the confirm password.	Register failed. A notification “Passwords Do Not Match!” will be shown in the top-right corner. And the website will stay on the Register page.
Other special problems happen.	Register failed. A notification “Insert Error, Wrong Input” will be shown in the top-right corner. And the website will stay on the Register page.

The frontend file of the Register page shows below:

Name	Address
Register	/code_1/front-end/src/register/index.jsx

The API for the Register page shows below:

API name	URL	Description
register	http://127.0.0.1:5000/register	Register a new account
getAllLocation	http://127.0.0.1:5000/GetAllLocation	Get all location

Appointments Page

Related to: PO4, PO5, PO7, PO8, PO9, PO10, PO11, PO13, PO16, PO17, PO18, PO24 and PO29

The Appointments page shows the information of every appointment. The user can choose the date and then can see the information of three days' appointments from 6:00 AM to 18:00 PM. Different types of appointment's status own different color. Meanwhile, if the patient does not have medical insurance or has not had any in-person visits within the past year, the patient's name and the appointment type will be highlighted respectively. It could be visited through the URL address “http://localhost:3000/#/appointments”.

The selector for date, the menu for navigation and the appointments' table are shown on this page. Meanwhile, the expected workload duration is shown on this page, too.

There are five buttons on the menu, which are called "APPOINTMENTS", "RECORD", "STATISTIC", "SETTINGS" and "LOG OUT" respectively. Clicking on the first four buttons can navigate to the respective pages based on the buttons' names. And the last one is for logging out. This menu also exists in the Record page, the Statistics page and the Settings page and they are the same.

If the user logs in with an administrator account, he/she can select a user from all users' list.

All possible operations of the appointments' table are listed in the table below:

Operation	Result
Click blank time range	The route navigates from the Appointments page to the Create page.
Click one of the appointments	A modal for choosing to edit or delete the appointment is shown on this page. The website will stay on the Register page.

The frontend file of the Appointments page shows below:

Name	Address
Appointments	/code_1/front-end/src/appointments/index.jsx

The API for the Appointments page shows below:

API name	URL	Description
showPanel	http://127.0.0.1:5000/ShowPanel	Get all information of all appointments
getAllLocation	http://127.0.0.1:5000/GetAllLocatio	Get all location
delete	http://127.0.0.1:5000/DeleteAppointmen	Delete an appointment
getAllUsers	http://127.0.0.1:5000/GetAllUsers	Get all information of all users
getSettings	http://127.0.0.1:5000/GetSettings	Get all information of all settings

Record Page

Related to: PO6, PO11, PO12, PO13, PO14, PO15 and PO23

The Record page shows the information of specific patient history and future appointments. The user can choose the patient having appointments with the user. Meanwhile, if the patient does not have medical insurance or has not had any in-person visits within the past year, the patient's name and the appointment type will be highlighted respectively. It could be visited through the URL address "http://localhost:3000/#/record".

The selector for patients, the menu for navigation and the appointments' table are shown on this page. When one of the appointments in the appointments' table is clicked, all information about this appointment will be shown in a modal.

If the user logs in with an administrator account, he/she can select a user from all users' list.

The frontend file of the Record page shows below:

Name	Address
Record	/code_1/front-end/src/appointments/index.jsx

The API for the Record page shows below:

API name	URL	Description
showPatientList	http://127.0.0.1:5000/ShowPatientList	Get all information of all patients having appointments with the user
showPatientRecord	http://127.0.0.1:5000/ShowPatientRecord	Get all information of a patients' record of appointments.
getAllUsers	http://127.0.0.1:5000/GetAllUsers	Get all information of all users

Statistic Page

Related to: PO8, PO 13, PO17, PO26 and PO27

The Statistic page shows bar charts of statistics about appointments. The user can choose the time range for two kinds of bar charts. One of the bar charts shows the sum of different types of appointments' status among one period. Another one shows the sum of the appointments among one period. It could be visited through the URL address "http://localhost:3000/#/statistic".

The picker for the time range, the button "OK" and the menu for navigation are shown on this page. After the button "OK" is clicked, a bar chart for statistics will be shown on this page.

If the user logs in with an administrator account, he/she can choose any number of users from all users' list.

The frontend file of the Statistic page shows below:

Name	Address
Statistic	/code_1/front-end/src/statistic/index.jsx

The API for the Statistic page shows below:

API name	URL	Description
getSpecRangeStatusStatistics	http://127.0.0.1:5000/GetSpecRangeStatusStatistics	Get the sum of different types of appointments' status among one period
getSpecRangeAppNumStatistics	http://127.0.0.1:5000/GetSpecRangeAppNumStatistics	Get the sum of the appointments among one period
getAllUsers	http://127.0.0.1:5000/GetAllUsers	Get all information of all users

Settings Page

Related to: PO13, PO28 and PO30

The Settings page shows some settings. The user can set the time range of appointments shown on the Appointments page.

It could be visited through the URL address “http://localhost:3000/#/settings”.

The picker for the time range, the button “Default time”, the button “OK” and the menu for navigation are shown on this page. After the button “OK” is clicked, this setting will be stored in the database. After the button “Default time” is clicked, the time range will be set to 6:00AM and 9:00 PM.

If the user logs in with an administrator account, he/she can additionally set the time range of break time for every user. During the period of the break time, the user cannot edit, add or delete appointments.

The frontend file of the Settings page shows below:

Name	Address
Statistic	/code_1/front-end/src/settings /index.jsx

The API for the Settings page shows below:

API name	URL	Description
editSettings	http://127.0.0.1:5000/EditSettings	Confirm the edition of settings
getSettings	http://127.0.0.1:5000/GetSettings	Get the information of user’s settings

Create Page

Related to: PO20, PO21, PO22 and PO25

The Create page is for users to create a new appointment or a new medical record for a new patient by completing all necessary information. It could be visited through the URL address “http://localhost:3000/#/create”.

The buttons “Create”, “Clear”, “Cancel”, “Check patient existence” and “Create new patient” are shown on this page. After the button “Clear” is clicked, the value of every component will be set to null. After the button “Cancel” is clicked, the website will go back to the Appointments page. After the button “Check patient existence” is clicked, the website will check whether the medical record of this patient exists or not.

All possible conditions for pressing button “Create” are listed in the table below:

Condition	Result
All information is correct.	Create a new appointment successfully. A notification “New Appointment Created!” will be shown in the top-right corner. And the route navigates from the Create page to the Appointments page.
One of the pieces of information is empty.	Create a new appointment failed. A notification “Please input all information” will be shown in the top-right corner. And the website will stay on the Create page.

The medical record of the patient does not exist.	Create a new appointment failed. A notification “Patient Does Not Exist, Please Create One!” will be shown in the top-right corner. And the website will stay on the Create page. Patient Does Not Exist, Please Create One!
--	--

After the button “Create new patient” is clicked, a new modal will be shown on this page. And two more buttons “Create new patient” and “Cancel” are shown on this modal. After the button “Cancel” is clicked on the modal, the modal will be closed.

All possible conditions for pressing button “Create new patient” on the modal are listed in the table below:

Condition	Result
All information is correct. This patient does not have a medical record in the database.	Create a new medical record for a new patient successfully. A notification “New Patient XX YY Added!” (“XX” is the patient’s first name and “YY” is the patient’s surname) will be shown in the top-right corner. The modal will be closed
All information is correct. This patient has a medical record in the database.	Create a new medical record for a new patient failed. A notification “Patient Already Exist!” will be shown in the top-right corner.
The format of the email is incorrect.	Create a new medical record for a new patient failed. A notification “The format of the email is incorrect!” will be shown in the top-right corner.
The format of the phone number is incorrect.	Create a new medical record for a new patient failed. A notification “The format of the phone number is incorrect!” will be shown in the top-right corner.
The format of the medicare number is incorrect	Create a new medical record for a new patient failed. A notification “The format of the medicare number is incorrect!” will be shown in the top-right corner.
One of the pieces of information is empty.	Create a new medical record for a new patient failed. A notification “Please input all information!” will be shown in the top-right corner.

The frontend file of the Create page shows below:

Name	Address
Create	/code_1/front-end/src/create /index.jsx

The API for the Create page shows below:

API name	URL	Description
judgePatient	http://127.0.0.1:5000/JudgePatient	Judge whether one patient has a medical record or not.
createAppointment	http://127.0.0.1:5000/CreateAppointment	Create a new appointment
createPatient	http://127.0.0.1:5000/CreatePatient	Create a new medical record for a new patient
getAllAppointmentTypes	http://127.0.0.1:5000/GetAllAppointmentTypes	Get the information of all

		appointments' types
getAllLocation	http://127.0.0.1:5000/GetAllLocation	Get the information of all locations

Edit Page

Related to: PO5 and PO19

The Edit page is for users to edit an appointment by editing some information. It could be visited through the URL address "http://localhost:3000/#/edit".

The buttons "Edit", "Clear" and "Cancel" are shown on this page. After the button "Clear" is clicked, the value of every component will be set to null. After the button "Cancel" is clicked, the website will go back to the Appointments page.

All possible conditions for pressing button "Edit" are listed in the table below:

Condition	Result
All information is correct.	Edit an appointment successfully. A notification "Appointment Updated!" will be shown in the top-right corner. And the route navigates from the Edit page to the Appointments page.
One of the pieces of information is empty.	Edit an appointment failed. A notification "Please input all information" will be shown in the top-right corner. And the website will stay on the Edit page.

The frontend file of the Edit page shows below:

Name	Address
Edit	/code_1/front-end/src/edit /index.jsx

The API for the Edit page shows below:

API name	URL	Description
editAppointment	http://127.0.0.1:5000/EditAppointment	Edit a new appointment
getAppointment	http://127.0.0.1:5000/GetAppointment	Get the information of an appointment
getAllAppointmentTypes	http://127.0.0.1:5000/GetAllAppointmentTypes	Get the information of all appointments' types
getAllLocation	http://127.0.0.1:5000/GetAllLocation	Get the information of all locations

Technologies Descriptions

Third-party functionalities

Frontend

1. **ReactJS**

ReactJS is a JavaScript library developed by Facebook for building user interfaces. It focuses on creating single-page applications (SPAs) and enables developers to build interactive user interfaces through a component-based approach. The core philosophy of ReactJS is component-based development, where the user interface is divided into independent, reusable components. Each component has its own state and properties, allowing for easier maintenance, scalability, and testing of code.[1]

2. **Ant Design**

Ant Design (AntD) is a React-based UI library developed by Alibaba. It provides a wide range of customizable and well-designed components for building elegant and responsive web applications. Ant Design follows a design system that focuses on a clean, consistent, and user-friendly interface, offering a set of components, icons, and styles that adhere to a unified design language.[2]

3. **Recharts**

Recharts is a composable charting library built with React components. It provides a set of customizable and interactive charts, allowing developers to create visually appealing data visualizations in web applications. Recharts leverages the power of React to create reusable chart components, making it relatively easy to integrate charts into React-based projects. It supports various chart types such as line charts, bar charts, pie charts, and more, offering a rich set of features for data presentation.[3]

4. **Dayjs**

Dayjs is a blazing-fast, ultra-light (2kb) alternative to the moment.js library that helps you parse, validate, manipulate, and display dates and times in pure JavaScript.[4]

Backend

1. **Flask**

Flask is an API of Python. It is a lightweight web application framework. It is designed to be easy to use and scalable, making it very suitable for rapid development of small to medium-sized web applications.

2. **psycopg2**

Psycopg2 is a popular PostgreSQL database adapter used for the Python. It is an open-source project widely used to connect Python applications to PostgreSQL databases.

3. SQL

SQL is a standard programming language for managing and operating relational database systems. It is the foundation of database management and is widely used for accessing, updating, and managing data.

4. PostgreSQL

PostgreSQL is one of the most compatible database systems with SQL standards, supporting a wide range of SQL data types and operations. It can be integrated with multiple programming languages, such as Python used in our project. PostgreSQL provides powerful indexing technology and query optimizer for high-performance applications, suitable for processing large datasets and high concurrency requests, therefore we use it in projects.

5. Google Cloud

Google Cloud is a series of cloud computing services provided by Google, including various hosted services for computing, storage, data processing, analysis, machine learning, and more, all running on Google's internal infrastructure. We have created a PostgreSQL database on Google Cloud to store project data, such as the appointments we created. Storing data in a cloud database is convenient for all developers to maintain and query together.

Challenges and Difficulties

Frontend

1. Route navigation with parameters

In the early period, we didn't know how to carry parameters during route navigation, so we used local storage to store the necessary parameters for transmission. However, using this method to carry parameters has a significant issue, which requires configuring in the App.jsx file each time to determine which pages can receive parameters and which pages can transmit parameters.

Solution:

Directly modifying the route address became a more convenient way to achieve parameter transmission.

2. Drawing charts

The native chart drawing tool in JavaScript is not user-friendly and cannot swiftly produce the bar graphs that we require. Additionally, the style of the generated bar graphs isn't appealing, and making modifications is cumbersome.

Solution:

We utilized a library called recharts, which allowed us to successfully create the needed graphs.

3. Format problem

While using the Antd library, we frequently encountered challenges in configuring the styles of certain components. It took a considerable amount of time to figure out the correct way to set the styles for these components. For instance, within the table component, it was difficult to adjust the height of each row and assign different click events to individual cells within the table.

Solution:

Researching information and gradually understanding how to configure the styles for each component was crucial. Simultaneously, for components that proved excessively challenging to change the style, our approach was to rewrite their CSS styles in the index.css file.

Backend

1. Connection with BP Sample Database

As mentioned above, at first, the database we use is BP Sample Database which is provided by our client. But only two of our team members have successfully downloaded the BP and took a long time trying to connect to BP Sample Database.

Solution:

We asked for help on weekly meetings with our client, one member from another group provided assistance to us and helped us resolve this issue. We are very grateful to her.

2. Only have read permission

While using BP Sample Database, we only get read permission of the database. So it's hard to test some features we have implemented.

Solution:

By the lecture's suggestion and with the client's approval, we mimicked the BP database and created a dummy database locally using PostgreSQL.

3. The inconvenience of local databases

Because our database was locally created, it was inconvenient for subsequent testing by both the lecturer and the client.

Solution:

We created a PostgreSQL database on Google Cloud that can be accessed remotely from any IP address. Our lecture and client do not need to modify the backend configuration to connect to this database.

Installation and Prerequisites

Download and Initialization

1. Clone the code on your local environment:
`git clone https://github.com/uns-w-cse-comp3900-9900-23T3/capstone-project-9900w13bzava.git`
2. Install python3.7 in your local environment
3. Install Node.js in your local environment

Run backend server

1. Enter a python virtual environment.
2. Open a terminal under /code_1/back-end/ folder.
3. Install all required packages: `pip3 install -r requirement.txt`
4. Run: `Python3 app.py`

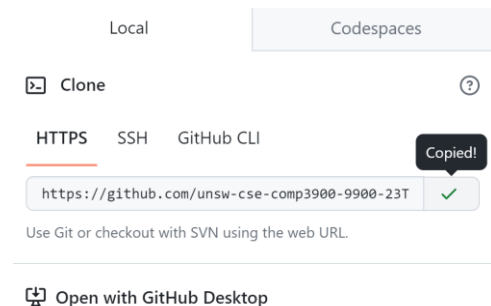
Run frontend websites

1. Open a terminal under /code_1/frontend/ folder
2. Run: `npm install`
3. Run: `npm start`. The website will open automatically.
4. Run: `npm run build`. This command will establish a new file called “bulid”. And you can open the index.html in this new file rather than Run: `npm start`.

User Documentation

Download and deploy the project

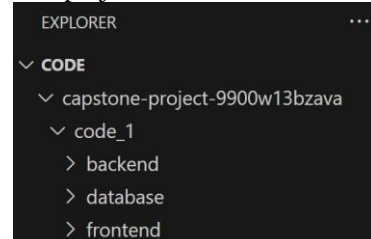
First, copy the link of the project from Github. The URL: <https://github.com/uns-w-cse-comp3900-9900-23T3/capstone-project-9900w13bzava.git>



Then open a new terminal in VSCode and clone the project.

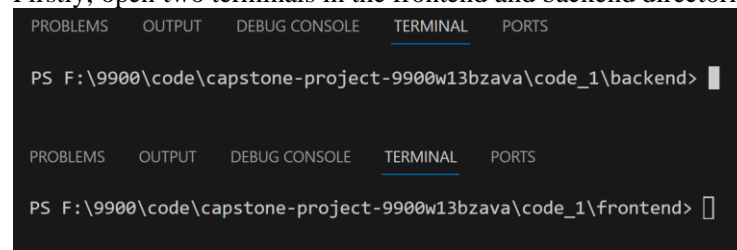
```
PS F:\9900\code> git clone https://github.com/uns-w-cse-comp3900-9900-23T3/capstone-project-9900w13bzava.git
Cloning into 'capstone-project-9900w13bzava'...
remote: Enumerating objects: 1338, done.
remote: Counting objects: 100% (81/81), done.
remote: Compressing objects: 100% (65/65), done.
remote: Total 1338 (delta 36), reused 48 (delta 16), pack-reused 1257
Receiving objects: 100% (1338/1338), 526.36 KiB | 12.24 MiB/s, done.
Resolving deltas: 100% (775/775), done.
PS F:\9900\code>
```

The project has been downloaded.



Starting the Application

Firstly, open two terminals in the frontend and backend directories respectively.



For backend, firstly, we need to install all the Python libraries required for the project, which we have summarized in the requirements.txt file.

Type `pip install -r requirements.txt` or `python -m pip install -r requirements.txt`

```
PS F:\9900\code\capstone-project-9900w13bzava\code_1\backend> python -m pip install -r requirements.txt
12 zipp==3.17.0
```

Then type `python backend.py` to start the backend.

```
Press CTRL+C to quit
```

Before starting with frontend, Node.js should be downloaded to the computer first. Go to <https://nodejs.org/en> and download the 20.9.0 LTS (The recommended one) version.

Node.js® is an open-source, cross-platform JavaScript runtime environment.

Download Node.js®

20.9.0 LTS

Recommended For Most Users

21.2.0 Current

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#) [Other Downloads](#) | [Changelog](#) | [API Docs](#)

For information about supported releases, see the [release schedule](#).

And install the Node.js follow the steps.

Then switch to the frontend terminal. First type `npm install` to installing project dependencies and creating the `node_modules` directory.

```
PS F:\9900\code\capstone-project-9900w13bzava\code_1\frontend> npm install
```

After the installation process ends, type `npm start` to start the frontend.

```
Compiled successfully!

You can now view frontend in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.0.150:3000

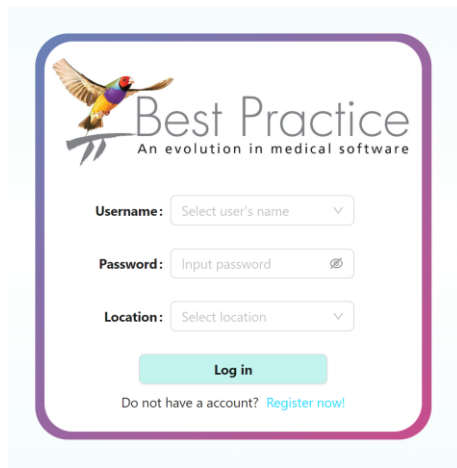
Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Now that the application has been opened, it will automatically redirect to the webpage. You can also use the URL to open it.

Login Page

In the login page, you can see the Register button. When you click it, you can switch to the register page. Input the information and create a new user here.



Best Practice
An evolution in medical software

Username:

Password:

Location:

[Log in](#)

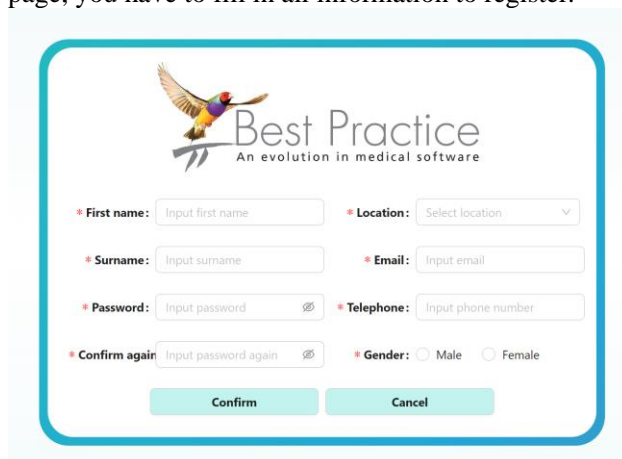
Do not have an account? [Register now!](#)

There are six users in our database. Their name and the passwords show in the table below.

Name	Password
administrator	password
Haodong Ke	password2
Sha Zhou	password3
Yalin Li	12345678
Zhenwei Wu	password1
Zhetai Jiang	password4

Register Page

If you want to create a new account, on 'Login Page', click 'Register now', it will return to following page, you have to fill in all information to register.



Best Practice
An evolution in medical software

* First name: * Location:

* Surname: * Email:

* Password: * Telephone:

* Confirm again: * Gender: ☐ Male ☐ Female

[Confirm](#) [Cancel](#)

Appointments Page

As shown in the following picture, this page shows the appointments of the user who login. Different color means different status. The 'Workload' shows the work load of the chosen date. Click on the specific appointment, the detail of the appointment will pop out. User can also add, edit and delete appointments on this page.

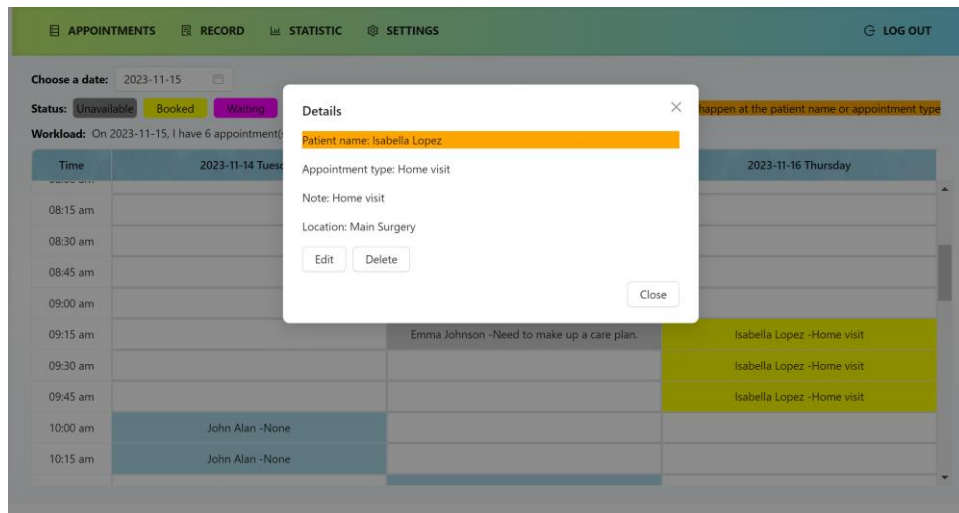
<div> <div>APPOINTMENTS</div> <div>RECORD</div> <div>STATISTIC</div> <div>SETTINGS</div> </div> <div>LOG OUT</div>			
<div>Choose a date: 2023-11-15</div> <div> <div>Status: Unavailable</div> <div>Booked</div> <div>Waiting</div> <div>Urgent</div> <div>With doctor</div> <div>At billing</div> <div>Completed</div> </div> <div>Highlight: happen at the patient name or appointment type</div> <div>Workload: On 2023-11-15, I have 6 appointment(s) in total. My expected workload duration is 3 hour(s) 0 minute(s).</div>			
Time	2023-11-14 Tuesday	2023-11-15 Wednesday	2023-11-16 Thursday
08:45 am			
09:00 am		Emma Johnson -Need to make up a care plan.	
09:15 am		Emma Johnson -Need to make up a care plan.	Isabella Lopez -Home visit
09:30 am			Isabella Lopez -Home visit
09:45 am			Isabella Lopez -Home visit
10:00 am	John Alan -None		
10:15 am	John Alan -None		
10:30 am		James Brown -Need a phone call.	
10:45 am		James Brown -Need a phone call.	
11:00 am	Jordan Habib -None		

If login with administrator account, there will be an extra ‘Choose a doctor’ dropdown on ‘Appointments Page’, admin can choose any doctor to see his/her appointments.
Note that admin account should not create any appointments, otherwise may lead to undefined behavior.

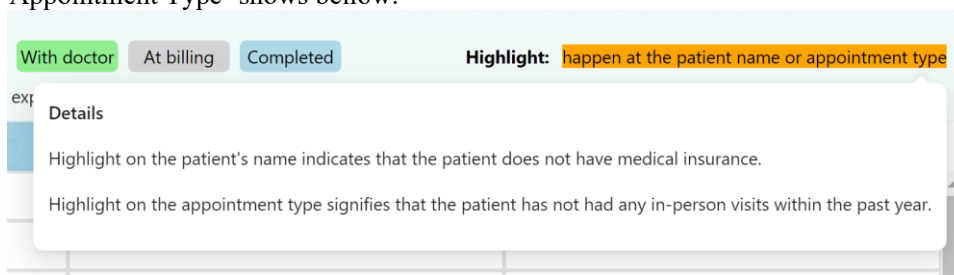
<div> <div>APPOINTMENTS</div> <div>RECORD</div> <div>STATISTIC</div> <div>SETTINGS</div> </div> <div>LOG OUT</div>			
<div>Choose a date: 2023-11-16</div> <div>Choose a doctor: Select doctor</div> <div> <div>Status: Unavailable</div> <div>Booked</div> <div>Waiting</div> <div>Urgent</div> <div>With doctor</div> <div>At billing</div> <div>Completed</div> </div> <div>Highlight: happen at the patient name or appointment type</div> <div>Workload: On 2023-11-16, I have 0 appointment(s) in total. My expected workload duration is 0 hour(s) 0 minute(s).</div>			
Time	2023-11-15 Wednesday	2023-11-16 Thursday	2023-11-17 Friday
07:45 am			
08:00 am			
08:15 am			
08:30 am			
08:45 am			
09:00 am			
09:15 am			
09:30 am			
09:45 am			
10:00 am			

Appointments Details Page

If the user want to see the detail of the appointment: On the ‘Appointments Page’, click on the appointment, a window will pop out, like following picture, user can edit or delete this appointment on this Page.



There are one thing need to be noted is that, the meaning of the high light on the 'Patient name' or 'Appointment Type' shows bellow:



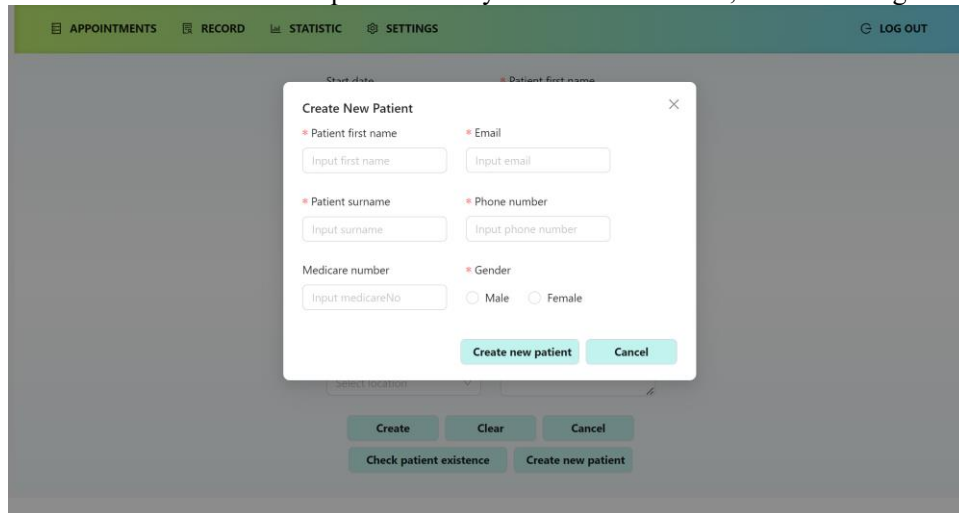
Add Appointments Page

On the 'Appointments Page', click any time slot you want, following page will show to user, this is where user create their appointments. The parts marked with a red star are required fields.

It should be noted that, an appointment can only be created for a patient if the patient's information exists in the database. So, after fill in the 'Patient first name' and 'Patient surname', should click 'Check patient existence' button to check whether this patient exist in the database, if not, click 'Create new patient' to create a new patient record, if exists, then just create a new appointment.

Create new patient Page

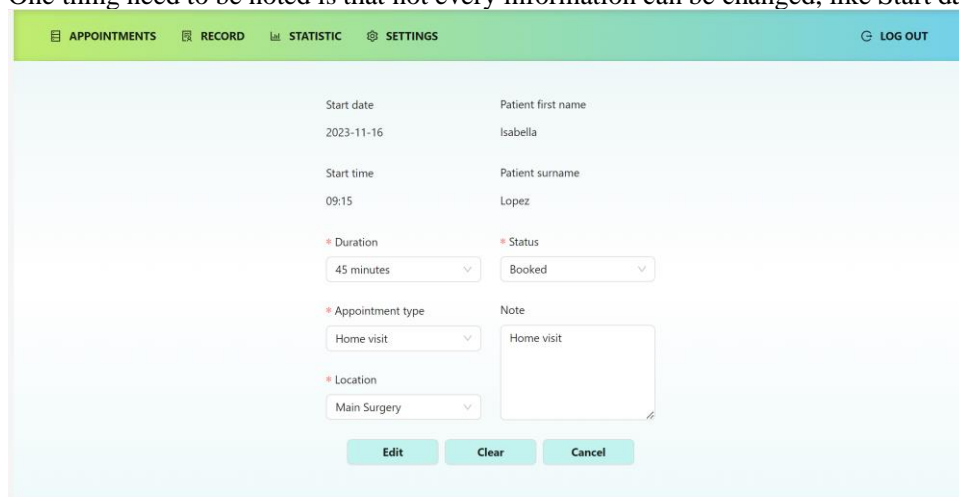
On Add appointments Page, if patient record don't exist on the database, then need to create a new patient record, after click Create new patient, a window will pop out, like following picture. After fill in information of patient, click 'Create new patient' button, then these information will be stored in the database. If the patient already exist in the database, can't create again.



The screenshot shows a web application interface with a top navigation bar containing 'APPOINTMENTS', 'RECORD', 'STATISTIC', 'SETTINGS', and 'LOG OUT'. A modal window titled 'Create New Patient' is open, featuring the following fields: 'Patient first name' (with 'Input first name' placeholder), 'Email' (with 'Input email' placeholder), 'Patient surname' (with 'Input surname' placeholder), 'Phone number' (with 'Input phone number' placeholder), 'Medicare number' (with 'Input medicareNo' placeholder), and 'Gender' (with radio buttons for 'Male' and 'Female'). At the bottom of the modal are 'Create new patient' and 'Cancel' buttons. Below the modal, on the main page, are buttons for 'Create', 'Clear', 'Cancel', 'Check patient existence', and 'Create new patient'.

Edit Appointment Page

On 'Appointments Details Page', click 'Edit' button, will turn to following page then user can edit that appointment. After changes, click 'Edit' button, those changes will be stored in the database. One thing need to be noted is that not every information can be changed, like Start date, Start time.



The screenshot shows the 'Edit Appointment' form within the same web application. The top navigation bar is identical. The form contains the following fields: 'Start date' (2023-11-16), 'Start time' (09:15), 'Duration' (45 minutes), 'Appointment type' (Home visit), 'Location' (Main Surgery), 'Patient first name' (Isabella), 'Patient surname' (Lopez), 'Status' (Booked), and a 'Note' field with the text 'Home visit'. At the bottom are 'Edit', 'Clear', and 'Cancel' buttons.

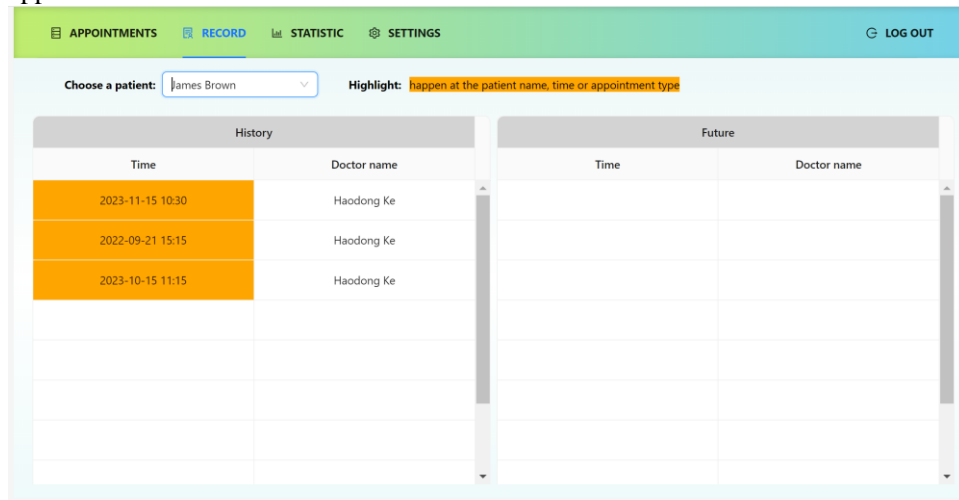
Delete Appointment

On the 'Appointments Details Page', click 'Delete' button will delete that appointment record from database.

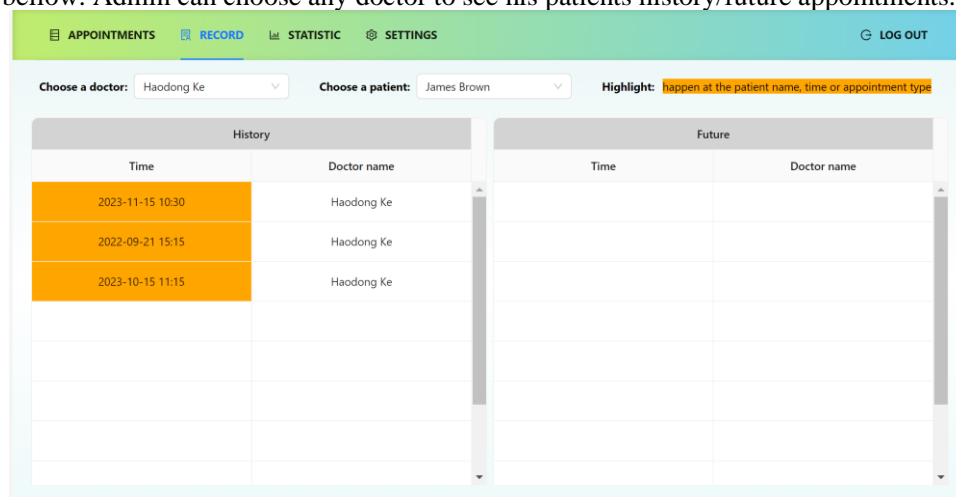
Record Page

This page shows all record of this patient for this user. User can use 'Choose a patient' dropdown

to choose a patients. Note that user can only see the record of his patient, i.e if the patient has previously made an appointment with that doctor, then the patient will appear in the 'choose' dropdown menu; otherwise, they will not appear. Click on the record shown bellow, the detail of the appointment will pop out. 'History' shows the appointment history of this patient, 'Future' shows the patient's future appointments.



If login with administrator account, there will be an extra 'Choose a doctor' dropdown, as shown bellow. Admin can choose any doctor to see his patients history/future appointments.



Statistic Page

'Statistic Page' is a statistic of the user's appointments, which can reflect the user's work status. In the following picture, the left side shows the total number of appointments with different statuses for this user during this period; the right side shows the total number of appointments on different dates for this user during the same period. Users can choose different time periods.



If login with administrator account, the 'Statistic Page' will like bellow. The admin can choose any doctors, to see their total statistics. In bellow picture, I choose 'Sha Zhou', 'Yalin Li' and 'Zhenwei Wu' these three doctors to see their overall statistics in a time period.



Settings Page

Here users can make some settings. By setting the time range, users can adjust the display range of the 'Appointment Page'. For example, if the time range is set from 06:00 to 18:00, then only appointments within this time range can be seen on the 'Appointment Page'.

APPOINTMENTS RECORD STATISTIC **SETTINGS** LOG OUT

Time range: 06:00 → 18:00 Default setting

OK

If login with administrator account, there is an extra setting 'Break time'. This setting specified mid break time. For example, in the following picture, we set it from 12:00 to 14:00, so that all users can't create new appointments (on 'Appointments Page') in this time range. This setting will fit to every other normal users (i.e doctors), and also this setting can only be seen when admin account is logged in.

APPOINTMENTS RECORD STATISTIC **SETTINGS** LOG OUT

Time range: 07:00 → 18:00 Default setting

Break time: 12:00 → 14:00 Default setting

OK

References

- [1] Rawat, P., & Mahajan, A. N. (2020). ReactJS: A modern web development framework. *International Journal of Innovative Science and Research Technology*, 5(11), 698-702.
- [2] Alibaba Group. (n.d.). Ant Design. <https://ant.design/>
- [3] Recharts Contributors. (n.d.). Recharts. <https://recharts.org/>
- [4] JQueryScript. (n. d.). Lightweight Date & Time Manipulation Library – dayjs .<https://www.cssscript.com/date-time-manipulation-library-dayjs/>