VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Md Walid Wise

ONLINE BOOKING AND CONFIRMATION WEB APPLICATION

School of Technology

2022

ACKNOWLEDGMENTS

I would like to thank my supervising teacher Dr. Smail Menani for supporting me throughout my thesis process. I am very grateful for that. I have also received great support from Mohammad Arafat Mollik. He is one of my senior friends. He supported me mentally whenever I needed it.

Furthermore, I want to express my heartfelt gratitude to my family and friends for believing in me and supporting me through my successes and disappointments.

Md Walid Wise

06.05.2022

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Md Walid Wise
Title	Online Booking and Confirmation Web
Year	2022
Language	English
Pages	24
Name of Supervisor	Smail Menani

The aim of this thesis was to make a simple web application to book an appointment or make a reservation. This thesis also explains how the reservation system and Google authentication work, and how to get a confirmation email after every change.

To ensure the simplicity of the software, it was necessary to implement REACT and Redux in the software project. Redux allows managing an application to the state in a single place and keeps changes in the application more predictably. Reason about changes occurring in the app gets easier with Redux.

The result of this thesis is a simplified and generalized program structure and user interface by implementing React, and Redux, and adding a calendar, and dashboard to the software. This creates a simple user interface where users can easily book an appointment, saving time.

Keywords Reservation, simplicity, authentication, and confirmation

Contents

LIST OF FIGURES

LIST OF CODE SNIPPETS

1 INTRODUCTION

If a potential buyer or user finds a product or program too difficult to use, they may abandon it right away. If customers give up at this point, they may miss out on all beneficial products or application features, thus making sure the product is simple to use is critical. When it comes to providing customer service, exceptional simplicity of use can drastically lower the number of resources required. There will be less need for customer assistance if a client can easily use the product, service, or application, saving time and money as a business. /5/

If we think about, for example, a hospital where people will book an appointment online for corona vaccination, there will be a few key features to look at. First, the user needs to have a login page where he/she can sign up or log in. Then after logging in, the main page will be accessible. There will be a calendar on the dashboard from where the user can check his/her previous appointments, cancel one appointment if needed, and book a new appointment. After booking an appointment the user and the hospital both will get a confirmation email. In the confirmation email, there will be details about the name, time, and place of the appointment. The user can then again cancel the appointment if needed. After canceling one appointment again the user and the hospital will get another confirmation email about the cancellation. Whenever there will be changes in the calendar for reservation or cancellation the busy slots for the appointments will be updated. The hospital will be able to add available hours and unavailable hours in the application. After booking an appointment, there will be a logout button for the user to log out.

The concept of this thesis was to make a web application to book appointments, which is as simple as possible also easy to use, and which will in return save users' time and money. For these qualities, a calendar and dashboards were added to the software. Also, React and Redux were used to implement the coding.

2 THE REQUIREMENTS

2.1 Requirements of the Application

When it comes to what customers desire from a product or service, ease of use is always at the top of the list. Whether it is something they expect to use in their personal life, such as a mobile phone, or something they intend to use at work, such as office equipment or business software, simplicity is critical when striving to make the product or service stand out to potential clients. This implies that doing it correctly the first time is crucial, and simplicity of use is a major feature.

With this web application, a user can book an appointment within a few clicks. First, the user must authenticate themself which will be done by google authentication or manual information. Then the user will be directed to the dashboard where he/she can check the calendar and book a new appointment. To book the appointment the user must input the region, hospital, date, and time which will be done with a few clicks one by one. After confirming the date and time the user will give the patient information and click the FINISH button. Then a dashboard will appear showing all the information about the patient and hospital. The user must confirm it by clicking the Confirm button. After selecting the confirm button there will be a pop-up showing the appointment has been created successfully. Also, as output, the user and the doctor of that hospital both will receive a confirmation email for the appointment.

The main aim of this project is to make a web application by which users can book an appointment very easily and fast, which in return will save people's time and effort.

2.2 The Use Case

With this web application, users can book an appointment for vaccination in a hospital. They can monitor their profile, change reservation time and details, and get notified.

2.3 Application Description

2.3.1 Logging in

When the user opens the application, they is required to log in. There are two ways to log in.

1. The user will manually put their email address and password to sign in.

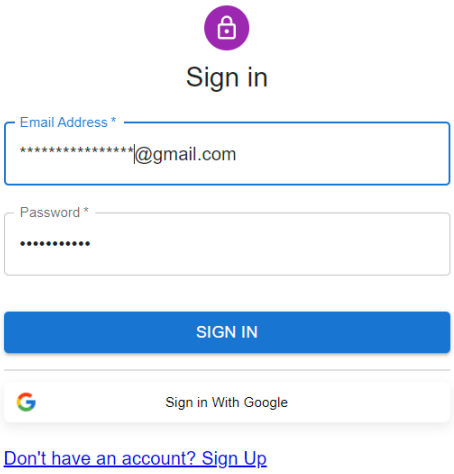

Figure 1. Login Via Email address and password

If the user does not have an account, they can sign up with another clickable link below, which will lead to a form to fill up First name, last name, email address, and password to sign up.

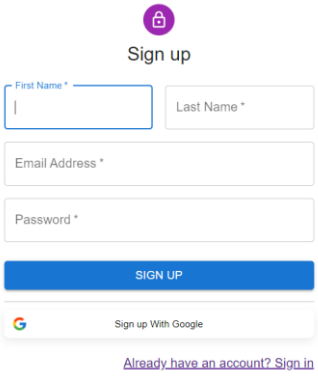

Figure 2. Sign up Via Name, Email address, and password

2. Using Google authentication. If the user has a Google account, they can easily log in with their google account by clicking the "Sign up With Google" button.

2.3.2Appointment Scheduling System Dashboard

After logging in the user will be directed to the appointment schedular dashboard shown in Figure 5, where they can check previously booked appointments.

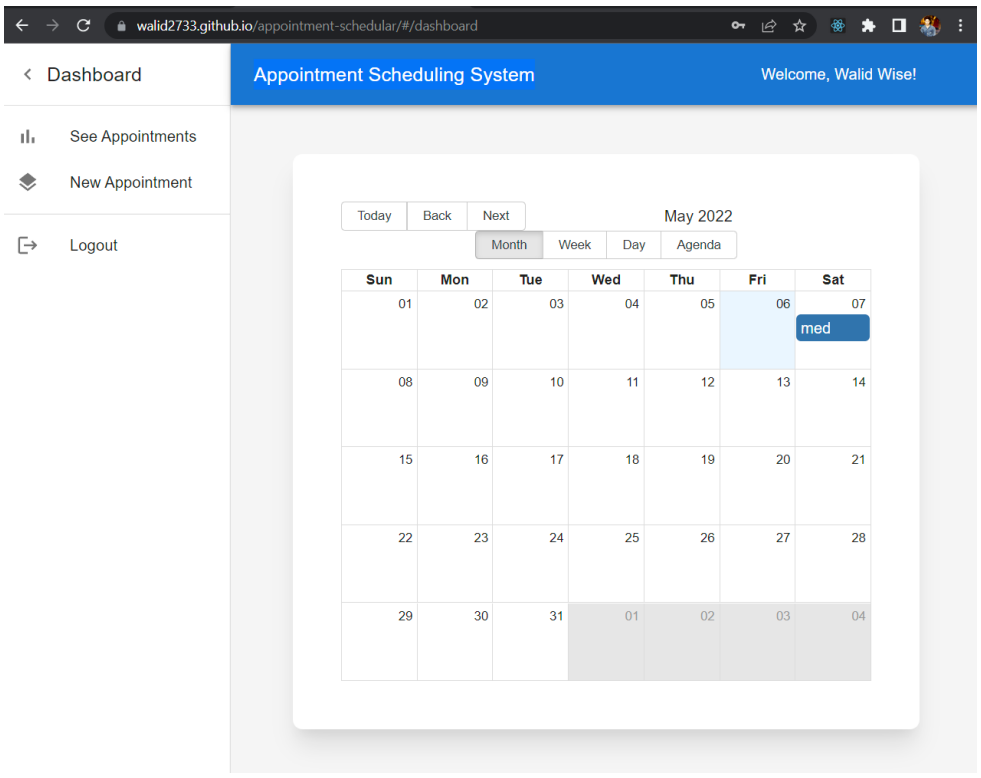

Figure 3. The Appointment Scheduling System Dashboard

2.3.3 Making a New Reservation

By clicking on “New Appointment” on the dashboard the user can make a new reservation. After clicking the button, the user can choose the region where they want to make the appointment.

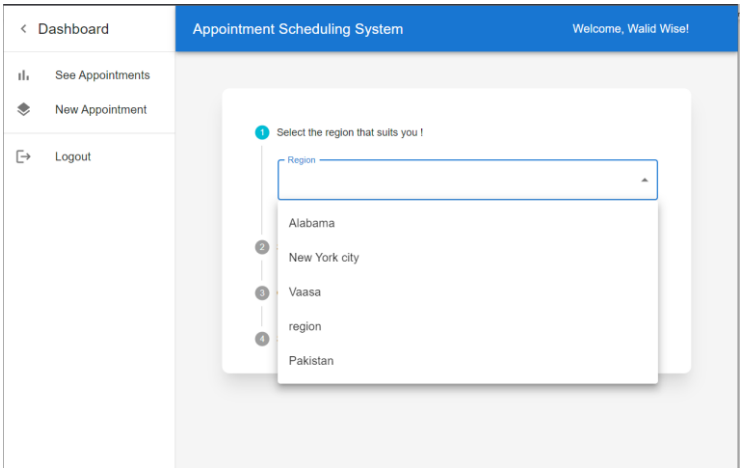

Figure 4. Selecting the region

After selecting the region, they can check the available hospitals in that region and select one hospital.

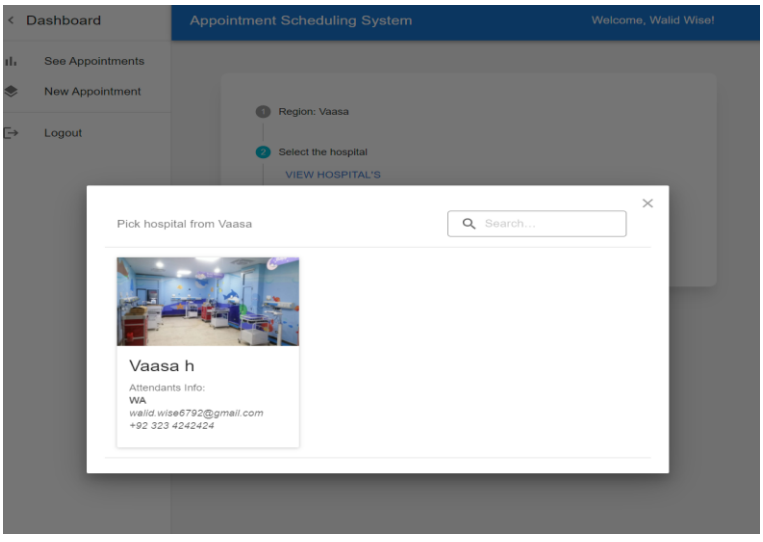

Figure 5. Selecting the hospital

After selecting the hospital, a calendar will appear where the user can check all the available hours and select a suitable time for the reservation.

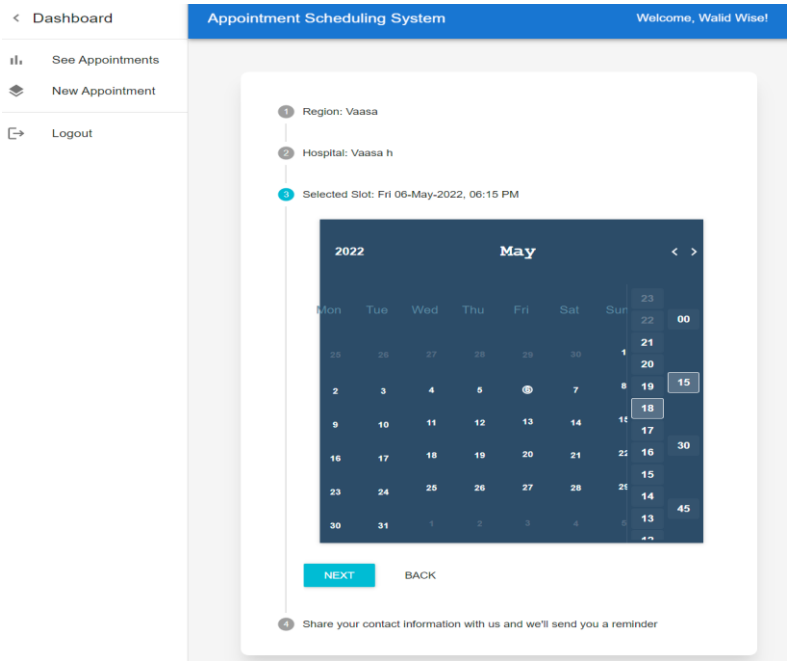

Figure 6. Calendar to choose a suitable time for reservation

After selecting the time, the user will need to fill up the patient's name, email address, contact number, reason, and optional reason. The email address provided in this section will get a confirmation email.

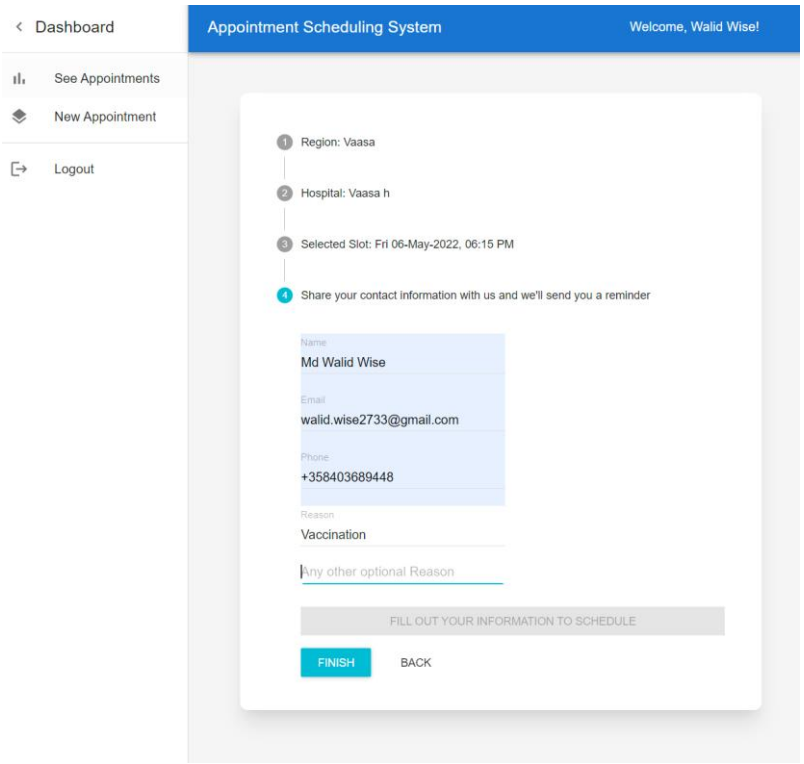

Figure 7. The Information form of the patient

After filling up the details the user can press the “FINISH” button and by pressing it a pop-up window will pop up shown in Figure 10, where all the appointment details will be available for confirmation. Pressing the "Confirm" button will schedule the appointment.

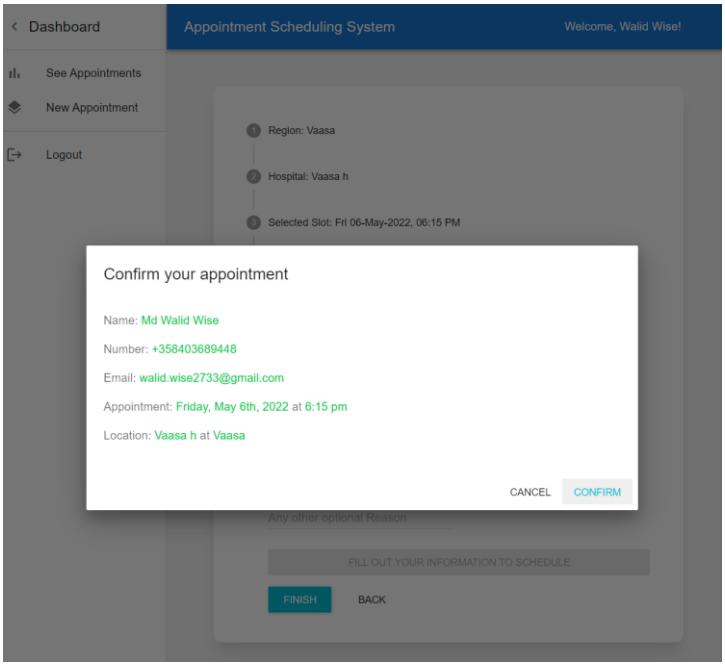

Figure 8. Confirmation details Pop-up window

After pressing “CONFIRM” there will be a pop-up on the top right corner shown in Figure 11, confirming the appointment has been scheduled.

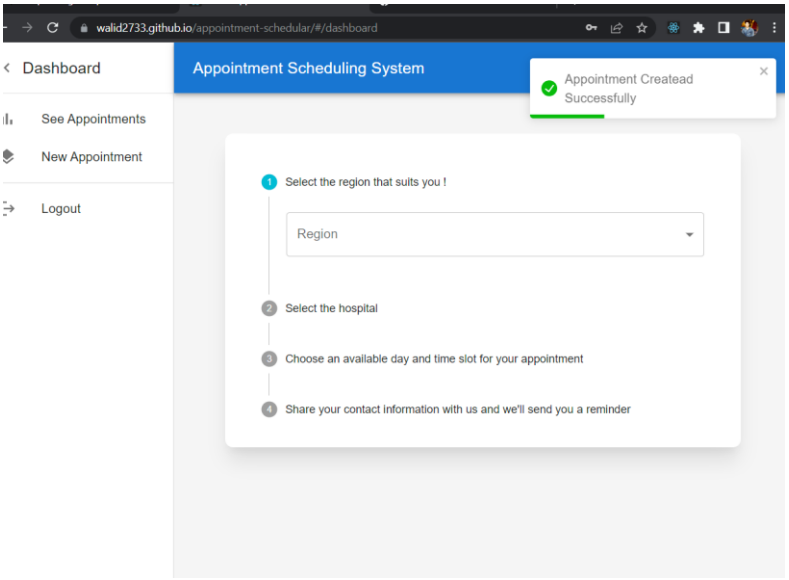

Figure 9. Appointment Scheduled confirmation

After the confirmation, the user and the hospital both will get a confirmation email shown in Figure 12, with the details of the appointment.

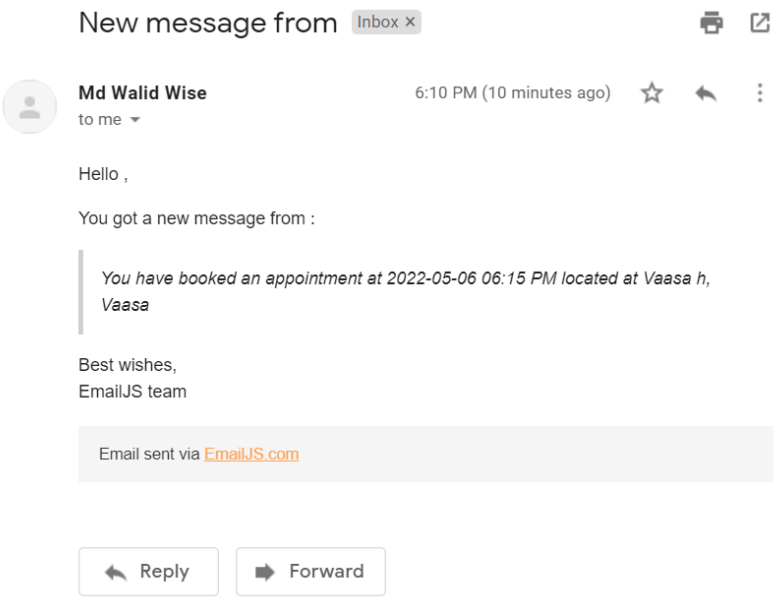

Figure 10. Email confirmation with details

Now by clicking “See Appointments” on the dashboard the user can check the updated calendar with the newly added reservation shown in Figure 13.

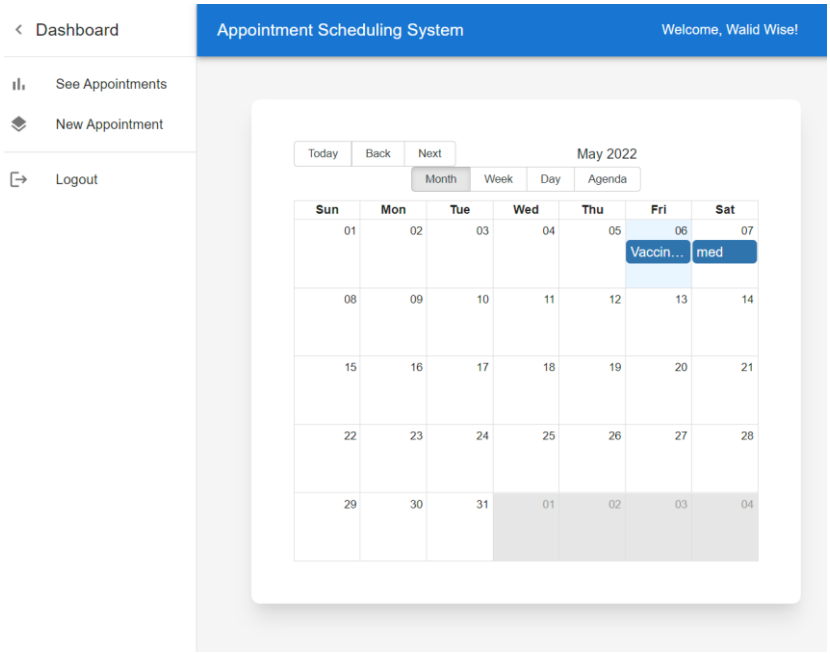

Figure 11. New reservation check

By clicking on one reservation the user can check the details of the reservation and can delete the reservation by clicking the “DELETE” button on the pop-up window.

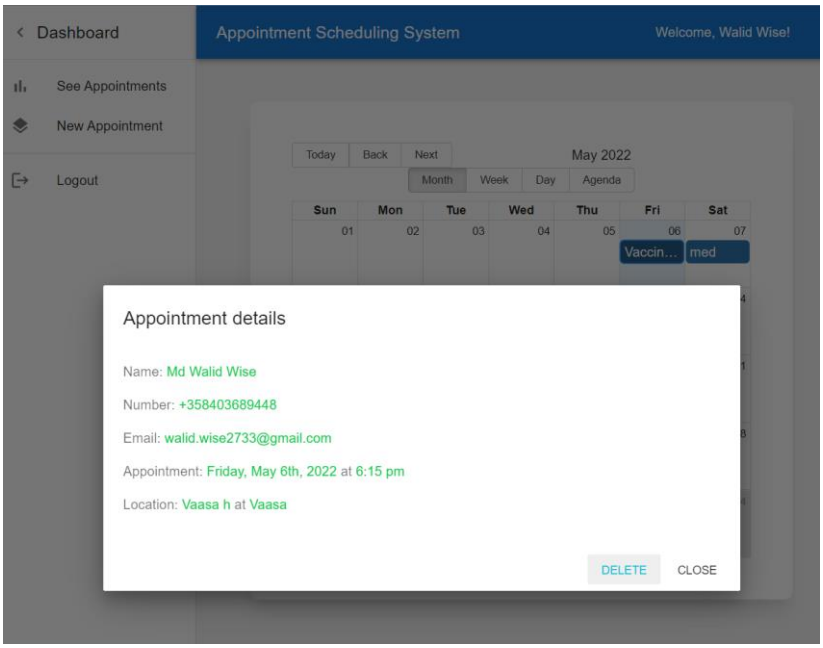

Figure 12. The Pop-up window to delete one reservation

If the user deletes one appointment again both the user and the hospital will get

a confirmation email. The calendar will be updated as well.

Figure 13, shows how the web application will work.

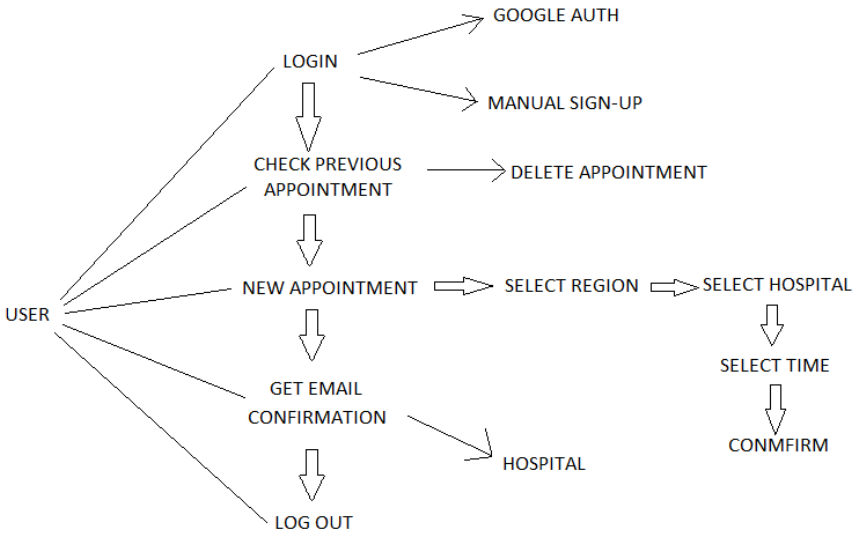

Figure 13. The functionality diagram of the web application.

3 THE PRELIMINARY DESIGN

3.1 The Methodology to Build the Application

3.1.1 React OAuth 2.0 for Google Authentication

Google Sign-In handles the OAuth 2.0 flow and token lifetime, making a connection with Google APIs easier. A user can always cancel access to an application at any moment. /2/

3.1.2 Firebase for Real-time Database and Authentication

For the backend of the web application, Firebase will be used to add and store user data and hospital information. The Firebase Realtime Database is a cloud-hosted NoSQL database that allows data to be saved and synchronized in real-time across users. /4/

Firebase will also be used to authenticate the user and sign in. Firebase Authentication makes it simple for developers to create secure authentication systems while also improving user sign-in and onboarding. This feature provides a comprehensive identification solution, including email and password accounts, phone authentication, and Google.

3.1.3. EmailJS for Email Confirmation and Big Calendar

By connecting EmailJS to one of the email providers supported and generating an email template using JavaScript we can send an email to both user and the hospital. /1/

Big Calendar is a full-featured calendar component that will allow us to manage all events and dates for the appointment. Figure 14 shows the Big Calendar user interface. where it is easy to monitor the schedule and make changes.

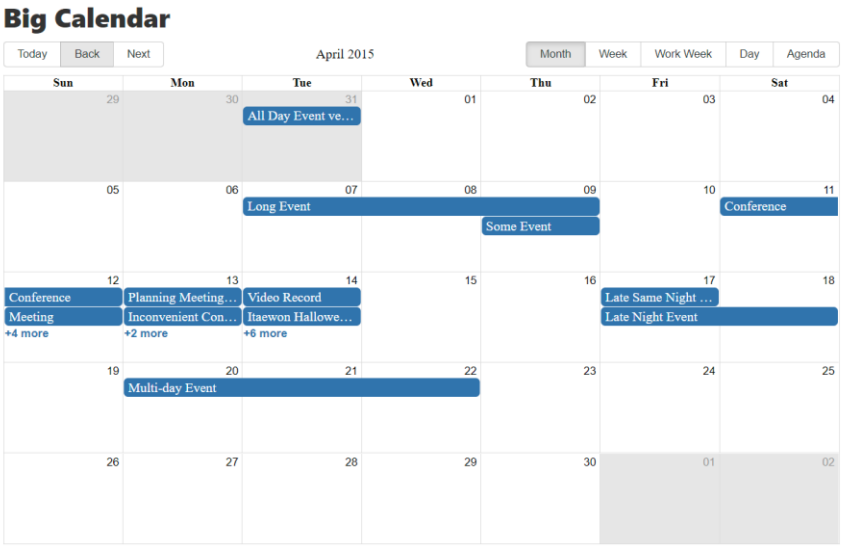

Figure 14. UI of Big Calendar.

3.2 Data Structures

3.2.1 Used Tools

To make simple and interactive user interfaces React will be used. After creating the basic views for each state of the application React will update and render only the necessary components when any data changes. /3/

For managing the application state and functionality Redux will be used. It will help to create applications that act consistently, run on a variety of platforms (client, server, and native), and are simple to test. Furthermore, it offers an excellent developer experience, such as live code editing mixed with a time-traveling debugger.

For storing the data and integrating the data inside of react, Firebase will be, provided by Google Inc. Through this the real-time data will be stored in the Firebase fire store. And after that, the data is queried throughout the React application and stored in React state (Redux)

3.3 Implementation Steps

The first step was to install node.js. After downloading and installing it from the Node official website, the following command can check that the updated version is installed.

node -v; This will show the installed version of node.js on the computer.

npm -v; This will show the installed version of npm on your computer.

The next step was to create an account on the following and setup.

- Google Firebase (For handling Auth & Database).
- EmailJs (A free service used to send emails)

Once the above setup is finished, the following command was run at the root directory of the project to install the dependencies of this project

npm install

After installing all the dependencies, npm start was run which started the system. Index.js the main page of the website. The main content is in *the src/* folder in which all the folders and subfolders exist. This file contains the building block of the project which is holding the file App.js through *<App/>* and renders everything in the root div of the *public/index.html* file.

```
ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById('root')
)
```

Code-Snippet 1. The index.js file

App.js is the file that holds the tree structure of the entire application that is connected to index.js mentioned above. It holds the routing of the project, and each route holds a separate page of the app.

For example, whenever someone hits the */dashboard* route in the web browser

<Dashboard/> component will render and so on.

```
<Routes>
  <Route
    path="/dashboard"
    element={user ? <Dashboard /> : <Navigate to="/" replace />}
  />

  <Route
    path="/register"
    element={user ? <Navigate to="/dashboard" replace /> : <SignUp />}
  />
  <Route
    path="/"
    element={user ? <Navigate to="/dashboard" replace /> : <Login />}
  />
</Routes>
```

Code-Snippet 2. App.Js file for the tree structure

The pages folder contains all the folders of relevant pages for example,

Auth pages (Login, Signup) and Dashboard Pages (Creating Appointments, View appointments)

Each subfolder contains the XHTML code of the relevant page (XHTML is the merge of HTML and JavaScript).

The Redux folder contains the state of the whole application and handles all the incoming connections requests. There are two subfolders, one for user authentication and one for data handlings such as appointments and hospitals. which are all connected to the store which Redux provides through the following code

```
export const store = createStore(pReducer,
composeWithDevTools(applyMiddleware(...mid
dleWares))
```

Code-Snippet 3. The redux/store.Js file for the connection request

Inside each subfolder, there are three main files.

Th Actions. Js file emits all the actions that are being fired by the components. For example, when someone hits to trigger the actionName function, this will emit an action which will further be notified to reducers.

```
export const actionName = () => ({
  type: 'ACTION_TYPE',
})
```

Code-Snippet 4. The users.action.js file for emitting action

When some components trigger an action, the action type listens to by two files reducers and sagas which then further proceed to the API calling of any other step.

The Reducers.js file holds the state of the entire application in an object which changes the object value as per actions, which is connected through store.js

```
export const reducer = (state = initialState,
action) => {
      switch (action.type) {
             case 'ACTION_TYPE':
                    return
do_something_in_state
      }
}
```

Code-Snippet 5. The users.action.js file for changing objects state

Here, the reducer is the function that is listening to action type through naive switch condition which checks on every action type and performs actions on it, initialState could be a piece of object.

sSaga.js also listens to actions and handles the external API's calling and emits actions as per the response of the API which will be further stored in reducers.js. For example, through the following code

```
export default function* rootSaga() {
  yield all([
```

```
        call(some_function),
        call(another_function)])
}
```

Code-Snippet 6. For emitting API's action root.saga.js file

Comparatively, both Reducers and Sagas listen to action types, the core difference between both is that the Reducer is handling the state of the application through action types, whereas the Saga is dealing with external services such as APIs as per listening to action and then it emits actions for the reducers. It should be noted that Saga is the middleware

Th Firebase folder contains a file firebase.js with all the functions that are being handled with database and backend, (e.g., CRUD operation on appointments, creating users, handling users, login, register) and is initializing by the following line.

```
const firebase = initializeApp(firebaseConfig)
```

Code-Snippet 7. The firebase function firebase.config.js

4 CONCLUSIONS

As per the Functionality Testing, all the test cases have been checked and verified successfully including regression testing, auth testing, session management testing, database testing, and performance testing. For the usability testing, each scenario had to be inspected separately to make the human-computer interaction smooth and user friendly. This was challenging to achieve since this is the most important part but he application has been optimized with the following features:

- Ease of learning
- Quick navigation
- Subjective user satisfaction
- General appearance

To summarize, the web application is designed as simple as possible to ensure ease of use and time-saving. Users can now easily book an appointment with the application and cancel a previous appointment if needed. In this respect, the implementation was successful.

REFERENCES

/1/ Anjula Shanaka. 2021. EmailJS: Send emails directly from your client-side JavaScript code. Accessed 22.04.2022. https://medium.com/weekly-webtips/emailjs-send-emails-directly-from-your-client-side-javascript-code-d85a6b23ff82

/2/ Integrating Google Sign-In into your web app. Accessed 21.04.2022. https://developers.google.com/identity/sign-in/web/sign-in

/3/ React. A JavaScript library for building user interfaces. Accessed 16.04.2022. https://reactjs.org/.

/4/ Rosengrance, Linda. N.d. Google Firebase. Accessed 22.04.2022. https://www.techtarget.com/searchmobilecomputing/definition/Google-Firebase.

/5/ Reckon. 2017. Products, services, and software: Why is the ease of use so important? Accessed 20.04.2022. https://www.reckon.com/reckon-blog/products-services-software-ease-use-important/.