****

**CZ2006: Software Engineering**

**Project Title: SickGoWhere**

**Submission Date:**

11 April 2021

**Lab Group:**

BCG2, Team 1

**Names of group members:**

|  |  |
| --- | --- |
| **Name:** | **Matriculation Number:** |
| Ernest Ang Cheng Han (L) | U1921210H |
| Goh Tse Yinn, Sheryl (D) | U1922180C |
| Ernest Tan Yan Heng | U1920436K |
| David Tay Ang Peng | U1910603L |
| Heng Jiu Xiao | U1922496D |
| Kim HeonJung | U1920224C |

# 

[**1 Product Overview**](#_daeum6s0ionr) **2**

[1.1 Purpose](#_pywgb2v6d9ex) 2

[1.2 Target Audience](#_iwzg21bmezhu) 2

[1.3 Initial UI Mockup](#_se394ih0xbms) 2

[**2 Functional Requirements**](#_k9l7ni4jvlt) **6**

[2.1 Use Case Diagram](#_pgq1l7l8215v) 9

[2.2 Use Case Descriptions](#_mqlhc4d3dmzi) 10

[2.3 Class Diagram](#_xo3brw6oqys7) 25

[2.4 Sequence Diagrams](#_ubeache1a8c5) 26

[2.5 Dialog Map](#_luxdkgqysvu6) 30

[**3 Non Functional Requirements**](#_8wd1h2r8qy4) **30**

[3.1 Usability Requirements](#_5pkqemvwg4gb) 30

[3.2 Reliability Requirements](#_qqhabqqokth0) 30

[3.3 Performance Requirements](#_l22oqznf5a2t) 31

[**4 Interface Requirements**](#_65y2t6e543w0) **32**

[4.1 User Interfaces](#_qsnd7jg9p7cc) 32

[4.2 Hardware Interface](#_84ov7rmlartu) 32

[4.3 Software Interfaces](#_rjdzem75m56t) 32

[4.4 Communication Interfaces](#_iu4qafxzdzb6) 32

[**5 Architecture Design**](#_b0q1o9jxjq9t) **33**

[5.1 System Architecture Diagram](#_blccd7yijglr) 33

[5.2 Design Patterns](#_3r811k7i7ugw) 33

[**6 Data Dictionary**](#_ksbc1nu4tw3s) **34**

[**7 Testing**](#_gite9gmpky46) **36**

[7.1 Black Box Testing Overview](#_tndhdystc7c5) 36

[7.2 Black Box Testing Framework - Jest](#_zai2x6zd48yh) 39

[7.3 White Box Testing](#_vtv9ye3bswoi) 40

[7.4 White Box Testing Framework - Mocha](#_gjufmhom0shd) 42

[7.5 Automated Testing with Continuous Integration (CI) Pipelines](#_7fzep7vqsnuo) 42

[**8 Appendix**](#_k9gnnchjpabt) **42**

[**To Do list**](#_yf5wfmpgzumu) **42**

# **1 Product Overview**

## 1.1 Purpose

With an increasingly volatile health climate because of COVID-19, we wanted to create a simple and intuitive app which could enhance the accessibility of clinics throughout Singapore.

Our application mainly serves two purposes. They are as follows:

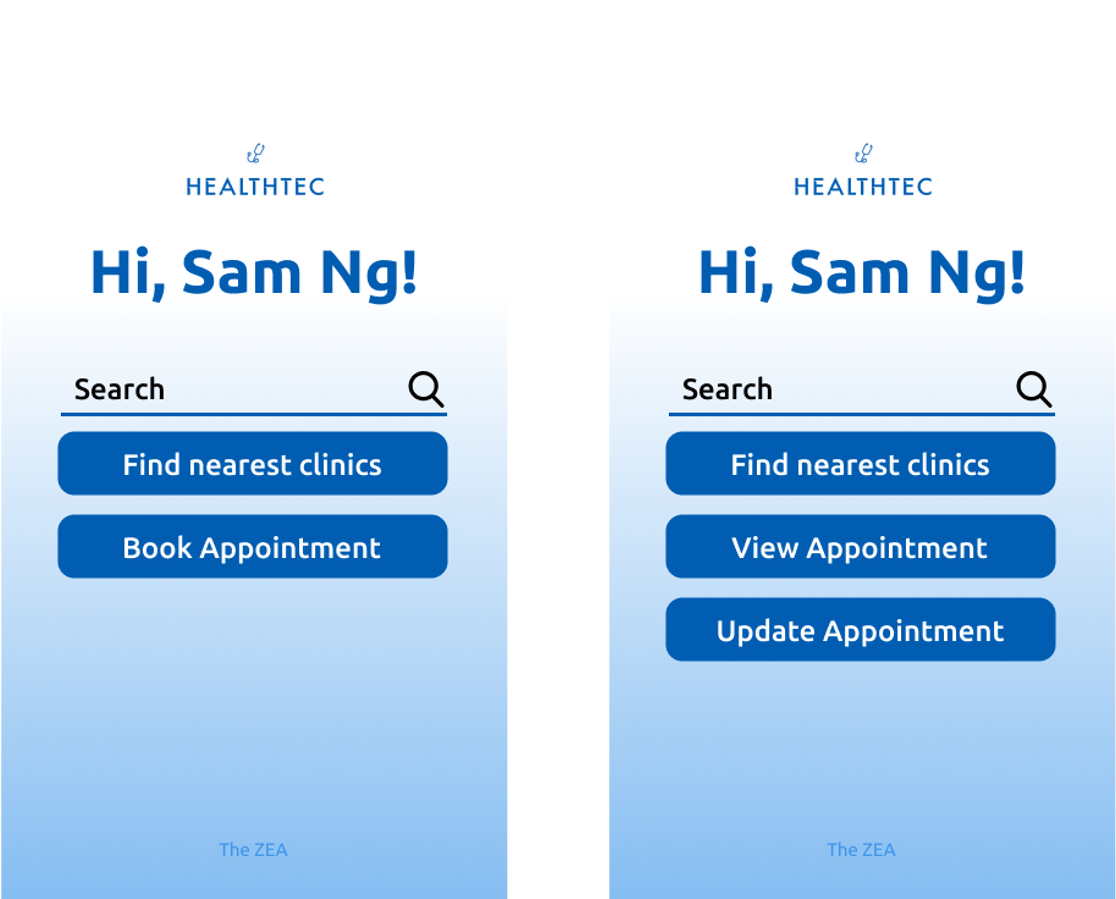
1. To provide users with information (including travelling times, directions) to the nearest 5 clinics based on the geolocation of an individual.
2. To allow users to create and manage (View, Edit, Delete) appointments at the clinic of their choice.

## 1.2 Target Audience

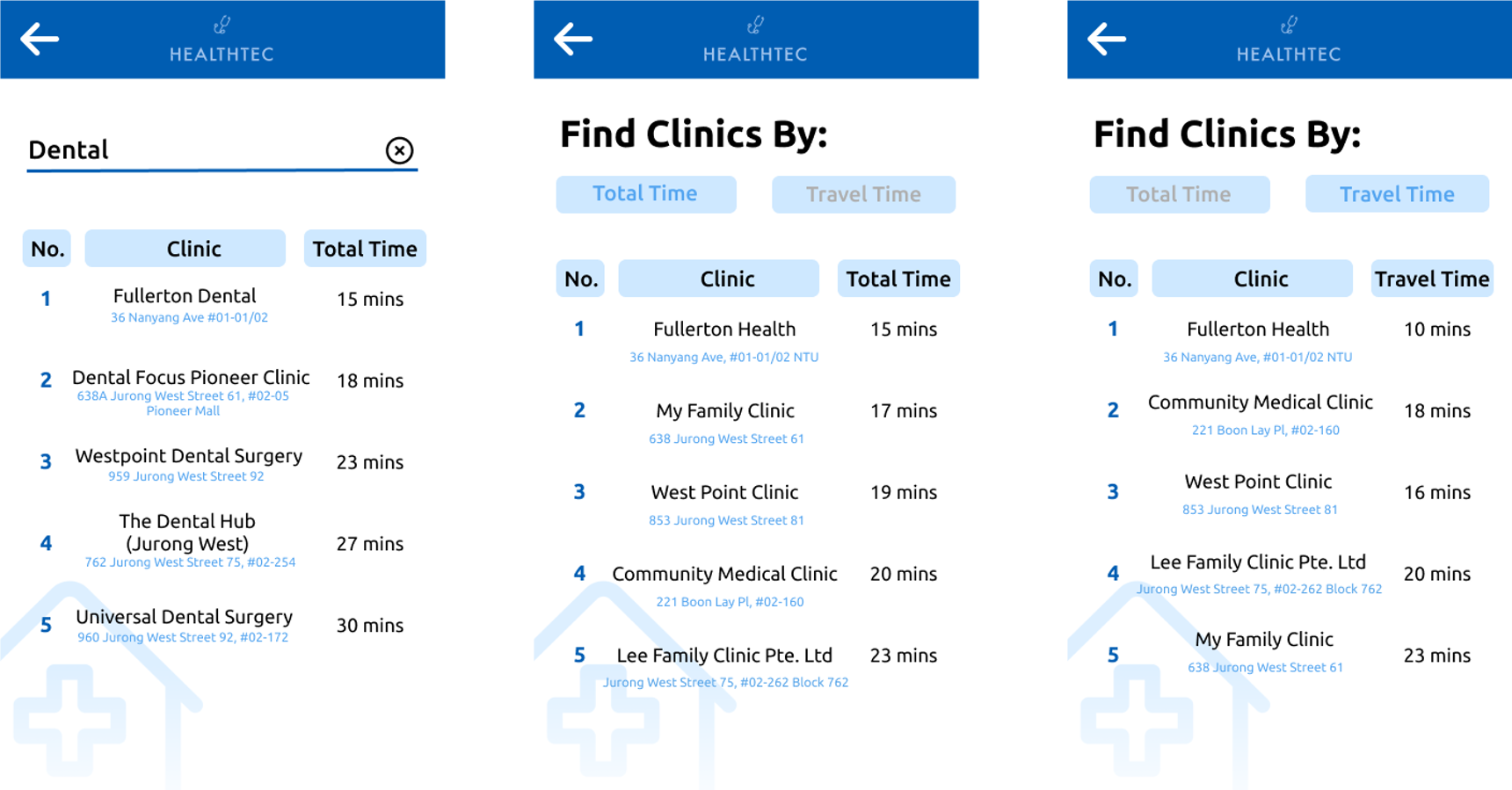
Our application targets anyone who wants to visit a doctor, and do not have a regular family doctor. In Singapore, more than 50% of people do not have a regular family doctor that they visit whenever they require a consultation. This application is designed to allow for greater accessibility to healthcare for everyone in Singapore.

## 1.3 Initial UI Mockup

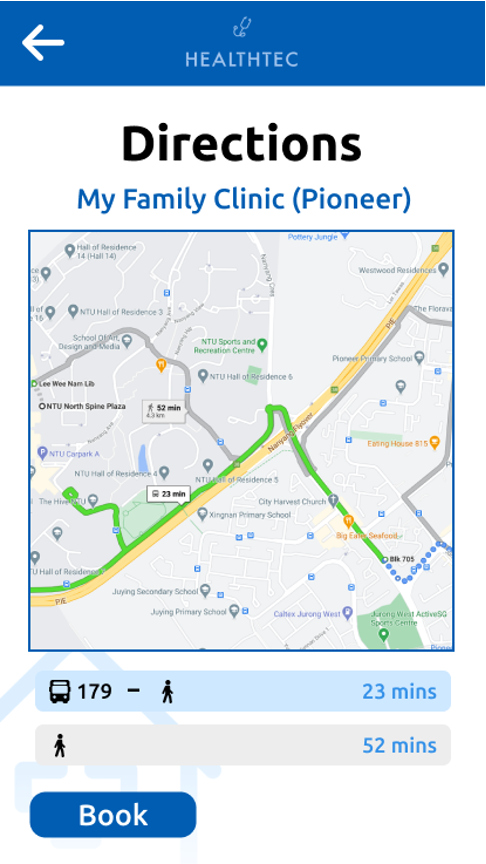


*Figure 1. Log in & Sign Up page*

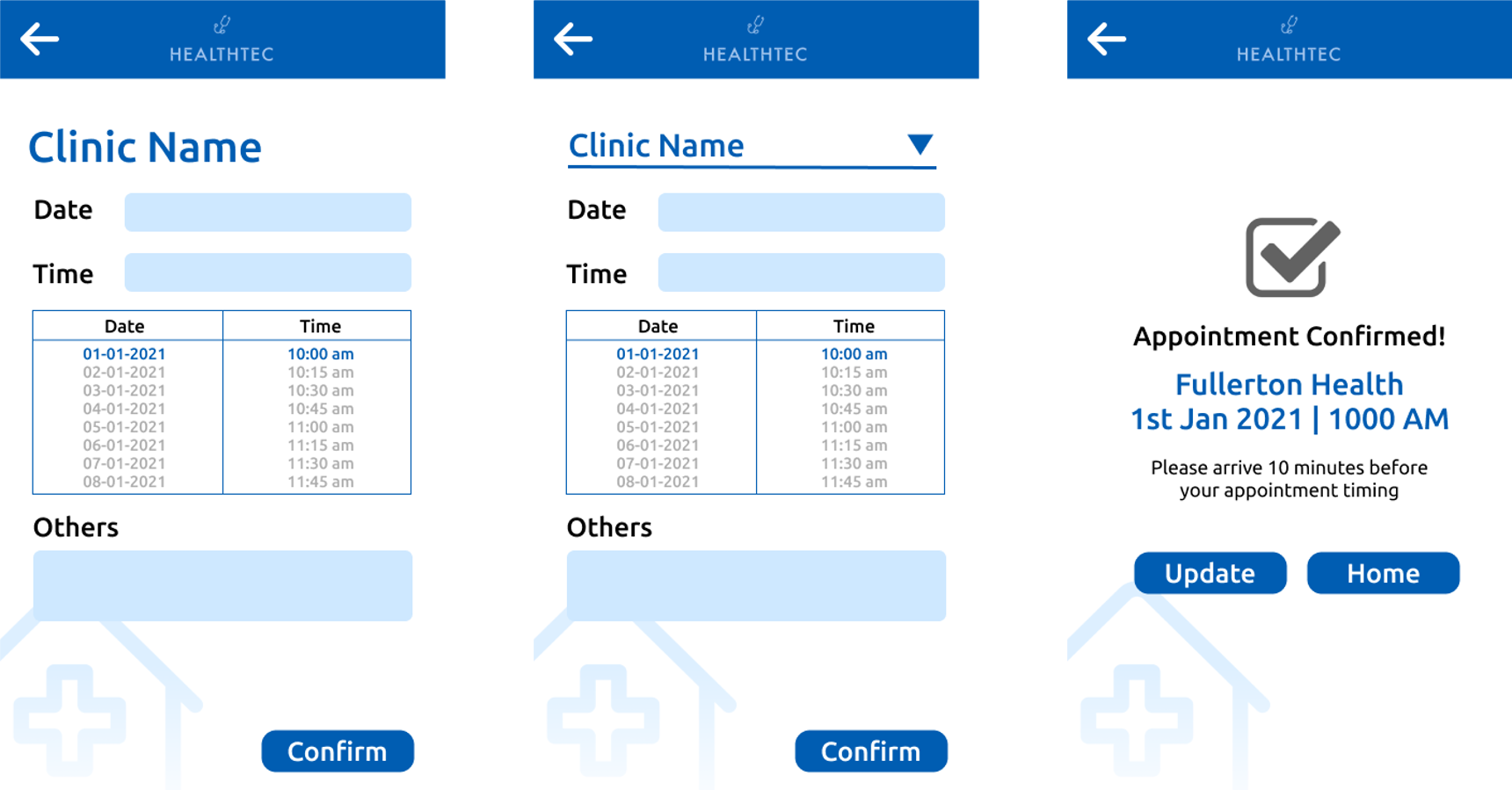
*Figure 2. Main Page - ⅰ) If the user doesn’t have any appointment ⅱ) If the user has an appointment booked before*



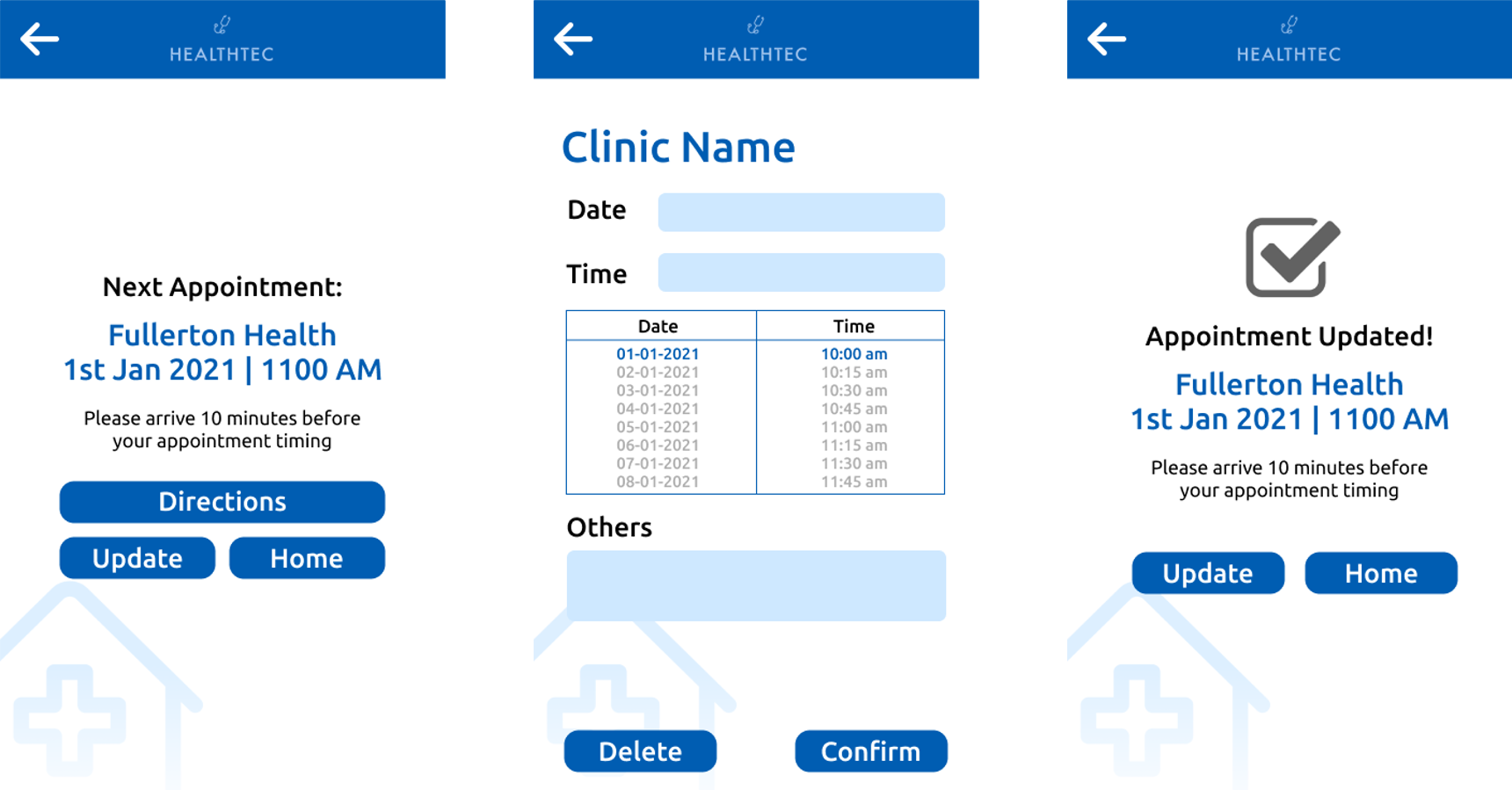
*Figure 3. Search by Clinic’s name, Find nearest clinics by travelling time & travel time page*

**

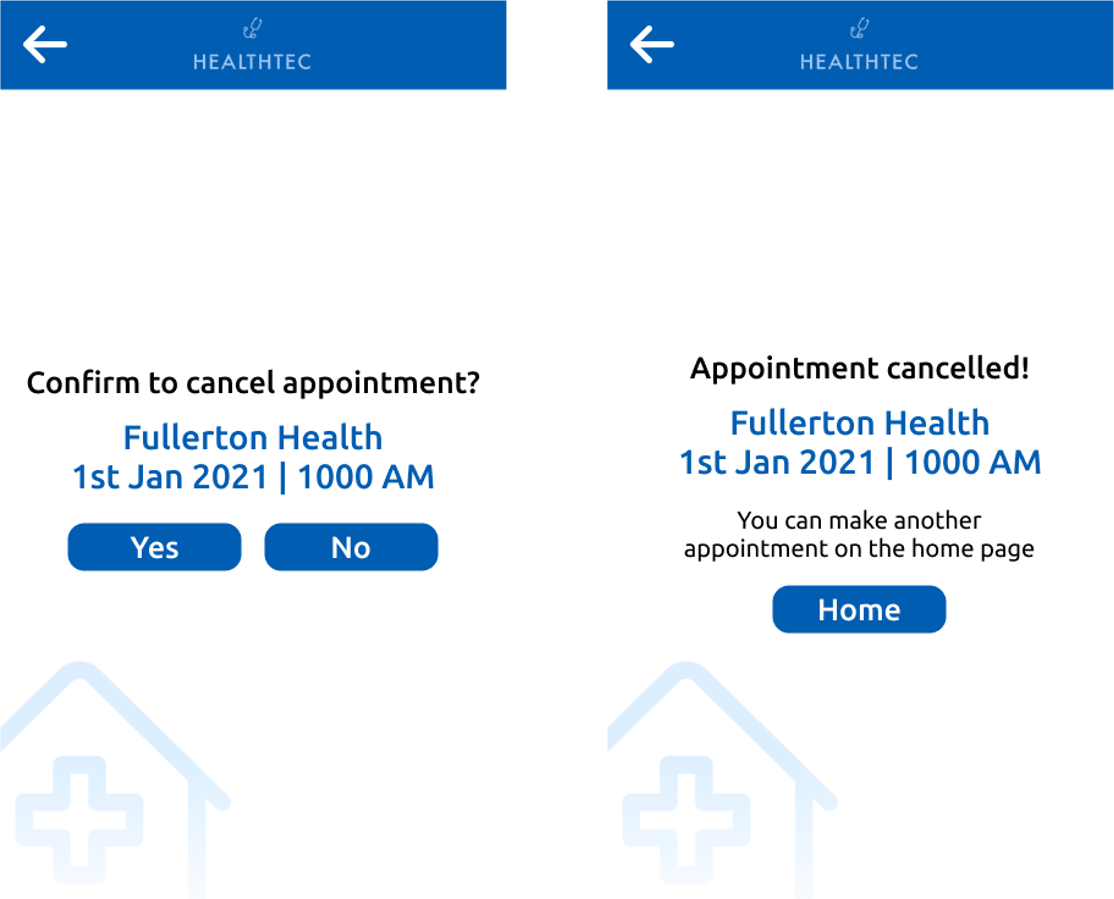
*Figure 4. Direction page*

**

*Figure 5. Create an appointment, Appointment Confirmation page*

**

*Figure 6. View Appointment & Update Appointment & Update Confirmation page*

**

*Figure 7. Delete Appointment & Delete Appointment Confirmation page*

# **2 Functional Requirements**

1. User registration

1.1 User must be able to input information for account registration

1.1.1 User must be able to enter name

1.1.2 User must be able to enter email

1.1.3 User must be able to enter password

1.1.4 User must be able to enter password confirmation

1.2 System must verify whether email already exists in system

1.3 System must check for password validity

1.3.1 System must check that password and password confirmation

is identical

1.3.2 System must check that password is at least 8 characters long

1.3.3 System must check that password is at most 15 characters

Long

1.4 System must be able to encrypt password

1.5 System must store registered information

1.6 System must inform the user that the registration is successful

1.7 System must inform the user that the registration is unsuccessful

1.8 System must be able to ensure all field are entered

2. User Login

2.1 User must be able to input information for logging in into system

2.1.1 User must be able to enter email

2.1.2 User must be able to enter password

2.2 System must verify user input login credentials

2.2.1 System must check that email exists in system

2.2.2 System must retrieve the password tagged to the email from system

2.2.3 System must be able to compare input credentials with retrieved

email and password

2.3 System must inform user on whether login is successful

2.3.1 System must be able to inform the user that the login is successful if

input credentials match with system

2.3.2 System must inform the user that login has failed if input credentials

does not match with system

3. Search for clinic by name

3.1 User must be able to enter clinic name in the search bar on the main page

3.2 System must be able to retrieve list of clinics from Google Maps API

3.3 System must be able to display list of clinics that matches the search criteria

3.4 System must be able to show the travelling time for each clinic on the list

3.4.1 System must be able to retrieve the user’s current location using the

Geolocation API

3.4.2 System must be able to retrieve the shortest travelling time required

To travel from user’s location to clinics using Google Maps API

4. Appointment System

4.1 User must be able to schedule an appointment if they do not already

have an appointment

4.1.1 User must be able to choose appointment date

4.1.2 User must be able to choose appointment time

4.2 User must be able to view details of their scheduled appointment

4.2.1 User must be able to view date of appointment

4.2.2 User must be able to view time of appointment

4.2.3 User must be able to view location of appointment

4.3 User must be able to edit the details of their scheduled appointment

4.3.1 User must be able to change date

4.3.2 User must be able to change time

4.4 User must be able to delete their scheduled appointment

4.5 System must be able to allow users to schedule appointment

4.5.1 System must be able to show the available time slots only

4.5.2 System must allow user to choose appointment date

4.5.3 System must allow user to choose appointment time

4.6 System must be able to allow users to edit their scheduled appointment

4.6.1 System must be able to show the available time slots only

4.6.2 System must allow user to choose appointment date

4.6.3 System must allow user to choose appointment time

4.7 System must be able to store all details of scheduled appointments

4.8 System must be able to show details of scheduled appointments

4.8.1 System must be able to show date

4.8.2 System must be able to show time

4.8.3 System must be able to show location

5. Find nearest clinics

5.1 User must be able to select how the top 5 clinics are ranked

5.2 System must be able to compute travel time for the clinics

5.2.1 System must be able to retrieve list of clinics using the API

5.2.3 System must be able to retrieve the user’s current location using the

API

5.2.4 System must be able to retrieve the shortest travelling time required

to travel from user’s location to clinics using API

5.3 System must display the travelling time to the top 5 clinics

5.3.1 System must filter out the top 5 clinics with least travel time

5.3.2 System must display the names of the clinic

5.3.3 System must display the travel time to the respective clinics

5.4 User must be able to click on the clinics to access the shortest route to the

selected clinic

6. Show the shortest route to the selected clinic

6.1 System must detect the user’s current position using API

6.2 System must show the shortest route to the selected clinic on the map

6.3 System must show the various routes to the selected clinic based on type of transport

6.3.1 System must be able to show various types of transportation mode

below the map

6.3.2 System must show the estimated time of each route

6.4 User must be able to click on the booking button to create an appointment for the selected clinic

## 

## 

## 

## 

## 

## 

## 2.1 Use Case Diagram

## 

## 

## 2.2 Use Case Descriptions

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | U1 | | |
| Use Case Name: | Register for new account | | |
| Created By: | Heng Jiu Xiao | Last Updated By: | Heng Jiu Xiao |
| Date Created: | 28/01/2021 | Date Last Updated: | 01/04/2021 |

|  |  |
| --- | --- |
| Actor: | User |
| Description: | Allow users to register for an account |
| Preconditions: | 1. User’s device must be connected to the Internet through Wi-Fi / Mobile Data |
| Postconditions: | 1. User’s account is created  2. User’s personal and login information are stored in the database  3. Existing users are denied repeated registration |
| Priority: | High |
| Frequency of Use: | Once in entire lifetime |
| Flow of Events: | 1. User selects “Register for new account” option  2. User will enter their email  3. User will enter password  4. User will enter confirm password  5. User will enter First Name  6. User will enter Last Name  7. User will press on the “Create” button  8. System will check against database to verify if email already exists  9. System will save login information into database  10. System will create a verification object with the User’s Id and verification code  11. System will send an email with the verification code to the user’s email  12. User will enter the verification code  13.User will press the “verify” button  14. User will be redirected back to Verify page |
| Alternative Flows: | AF-S9: If the email already exists in database   1. User would be prompted that account already exists with a warning message 2. User will be redirected back to the account creation page |
| Exceptions: | - |
| Includes: | 1. Verify email in system |
| Extends: | - |
| Special Requirements: | - |
| Assumptions: | 1. User inputs **accurate** information during registration process in steps 2 to 6 of Flow of Events  2. User will enter proper and identical passwords  3. User will enter a valid email address |
| Notes and Issues: | - |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | U2 | | |
| Use Case Name: | Verify email in system | | |
| Created By: | Heng Jiu Xiao | Last Updated By: | Heng Jiu Xiao |
| Date Created: | 28/01/2021 | Date Last Updated: | 01/04/2021 |

|  |  |
| --- | --- |
| Actor: | Authentication System |
| Description: | Check whether the email already exists in the database |
| Preconditions: | 1. System must be connected to the authentication system  2. System must be connected to Internet through Wi-Fi |
| Postconditions: | 1. Authentication system updates the verification status of the user |
| Priority: | High |
| Frequency of Use: | Depends on number of account creation |
| Flow of Events: | 1. User will enter 6 character code sent to email address  2. Authentication system will verify if the email is in the database and that the entered 6 character code corresponds to the unverified email  3. Authentication system will update the verification status of the account |
| Alternative Flows: | - |
| Exceptions: | - |
| Includes: | - |
| Extends: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | U3 | | |
| Use Case Name: | Login into system | | |
| Created By: | Heng Jiu Xiao | Last Updated By: | Heng Jiu Xiao |
| Date Created: | 28/01/2021 | Date Last Updated: | 01/04/2021 |

|  |  |
| --- | --- |
| Actor: | User |
| Description: | Login into system |
| Preconditions: | 1. User’s device must be connected to the Internet through Wi-Fi / Mobile Data  2. User is logged into the application |
| Postconditions: | 1. User is brought forward to home page |
| Priority: | High |
| Frequency of Use: | 1-2 times a month |
| Flow of Events: | 1. User will input email address  2. User will input password  3. User selects the “Login” button  4. System will verify if email exists in the database  5. System will verify if password is correct for the tagged phone number  6. System will display “Login successful”  7. User will press the “Home” button  8. User will be brought to home page |
| Alternative Flows: | AF-S6: Email address does not exist in database   1. System will display the following warning message “Email doesn’t exist!” 2. User will click the “Dismiss” button 3. System returns user to login page   AF-S6: Password does not tally with database’s password   1. System will display the following warning message “Phone number doesn’t exist!” 2. User will click the “Dismiss” button 3. System returns user to login page   AF-S6: Account is not verified   1. System will display account is not verified 2. System will redirect to the verification page |
| Exceptions: | EX-S6: If email field is left blank   1. System displays “Please enter valid email” 2. User will click the “Dismiss” button 3. User is returned to login page   EX-S6: If password is left blank,   1. System displays “Please enter valid password” 2. User will click the “Dismiss” button 3. User is returned to login page |
| Includes: | 1. Verify login credentials |
| Extends: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | U4 | | |
| Use Case Name: | Verify login credentials | | |
| Created By: | Heng Jiu Xiao | Last Updated By: | Heng Jiu Xiao |
| Date Created: | 28/01/2021 | Date Last Updated: | 01/04/2021 |

|  |  |
| --- | --- |
| Actor: | Authentication System |
| Description: | Check whether the entered password corresponds to the tagged password for a email address in the database |
| Preconditions: | 1. System must be connected to the authentication system  2. System must be connected to Internet through Wi-Fi |
| Postconditions: | 1. Authentication system returns a binary value to indicate if password corresponds to tagged email address |
| Priority: | High |
| Frequency of Use: | Depends on number of login attempts |
| Flow of Events: | 1. System will send an email address and a password to authentication system  2. Authentication system will verify if the password entered corresponds to the tagged password in the database  3. Authentication system sends a binary value to the system to indicate whether the login information tallies |
| Alternative Flows: | - |
| Exceptions: | - |
| Includes: | - |
| Extends: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | U5 | | |
| Use Case Name: | Create appointment under user | | |
| Created By: | Ernest Tan | Last Updated By: | Ernest Tan |
| Date Created: | 27/01/2021 | Date Last Updated: | 01/04/2021 |

|  |  |
| --- | --- |
| Actor: | User |
| Description: | To create an appointment at clinic of choice |
| Preconditions: | 1. User is connected to the Internet via Wi-Fi / Mobile Data  2. User does not have an existing appointment in the system |
| Postconditions: | 1. Database is updated with the new booking  2. User will not be able to create a new appointment |
| Priority: | High |
| Frequency of Use: | 1-3 times per month |
| Flow of Events: | 1. User selects the “Book” option on the Directions page  2. System will display the appointment page  3. User must select the date of appointment  4. System will return available time slots based on location and date input  5. User selects one time slot from the available time slots  6. System displays appointment details consisting of date, location, and time  7. User selects the “Confirm” button  8. System displays “Appointment confirmed” page  9. User is sent to the “View Appointment” page  11. System will send the details of the appointment back to the database |
| Alternative Flows: | AF-S3: User inputs a date with no available time slots   1. System displays message “There are no more available time slots! Please change your time or date!” 2. Return to Step 3 |
| Exceptions: | - |
| Includes: | - |
| Extends: | 1. Display appointment under user  2. Update appointment under user  3. Delete appointment under user  4. Display shortest route to selected clinic from current location |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | U6 | | |
| Use Case Name: | Display appointment under user | | |
| Created By: | Ernest Tan | Last Updated By: | Ernest Tan |
| Date Created: | 27/01/2021 | Date Last Updated: | 01/04/2021 |

|  |  |
| --- | --- |
| Actor: | User |
| Description: | To view the details of the current appointment |
| Preconditions: | 1. User is connected to the Internet via Wi-Fi / Mobile Data  2. User must have a pre-existing appointment within the app  3. User must be logged in to the app |
| Postconditions: | 1. User is able to view the location, date, time and comments of their existing appointment |
| Priority: | Medium |
| Frequency of Use: | 3-5 times per month |
| Flow of Events: | 1. User selects the “View Appointment” button  2. System will check the user’s phone number for a pre-existing appointment in the database  3. System returns details of the user’s appointment, including location, date and time |
| Alternative Flows: | AF-S3: System cannot find a pre-existing appointment tagged to user   1. System will return an error message “There are no prior appointments!” 2. User will be returned to the application homepage |
| Exceptions: | - |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | U7 | | |
| Use Case Name: | Update appointment under user | | |
| Created By: | Ernest Tan | Last Updated By: | Ernest Tan |
| Date Created: | 27/01/2021 | Date Last Updated: | 01/04/2021 |

|  |  |
| --- | --- |
| Actor: | User |
| Description: | To change the date or time of the current booking |
| Preconditions: | 1. User is connected to the Internet via Wi-Fi / Mobile Data  2. User must have a pre-existing appointment within the app  3. User must be logged in to the app |
| Postconditions: | 1. User’s booking details will be successfully updated and saved in the database |
| Priority: | Medium |
| Frequency of Use: | 1-3 times per month |
| Flow of Events: | 1. User selects the “Update Appointment” button on homepage  2. System will check the user’s phone number for a pre-existing appointment in the database  3. System will display the appointment page  4. User must select the new date of appointment  5. System will return available time slots based on location and date input  6. User selects one time slot from the available time slots  7. System displays appointment details consisting of date, location, and time  8. User selects the “Confirm” button  9. System displays “Appointment updated”  10. User clicks on the “Dismiss” button  11. User is sent to the “View Appointment” page  12. System will send the details of the appointment back to the database |
| Alternative Flows: | AF-S1: User selects “Edit” button on “View Appointment” page   1. Return to step 2   AF-S3: System cannot find a pre-existing appointment tagged to user   1. System will return an error message “There are no prior appointments!” 2. User will be returned to the homepage   AF-S4: User inputs a date with no available time slots   1. System returns an error message stating that there are no more available time slots, and that user must change either the location or date of booking 2. Return to Step 3 |
| Exceptions: | - |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | U8 | | |
| Use Case Name: | Delete appointment under user | | |
| Created By: | Ernest Tan | Last Updated By: | Ernest Tan |
| Date Created: | 27/01/2021 | Date Last Updated: | 01/04/2021 |

|  |  |
| --- | --- |
| Actor: | User |
| Description: | To delete the current appointment |
| Preconditions: | 1. User must be logged in to the application  2. User is connected to the Internet via Wi-Fi / Mobile Data  3. User must have pre-existing appointment |
| Postconditions: | 1. User’s appointment is successfully deleted from the system’s database  2. User will be able to create a new appointment |
| Priority: | Medium |
| Frequency of Use: | 1-3 times per month |
| Flow of Events: | 1. User selects the “Delete” button  2. System will check the user’s phone number for a pre-existing appointment in the database  3. System will display the message, “Do you want to delete your current appointment?”  4. User selects “Confirm” button  5. System will delete the user’s appointment from the system’s database  6. System displays “Appointment deleted”  7. User clicks on the “Dismiss” button  8. User is sent to the Home page |
| Alternative Flows: | AF-S3: User does not have a pre-existing appointment in the database   1. System will display the message “You do not have an appointment!” 2. User will be returned to the application homepage |
| Exceptions: | - |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | U9 | | |
| Use Case Name: | Display 5 nearest clinics according to current location | | |
| Created By: | David Tay | Last Updated By: | David Tay |
| Date Created: | 27/01/2021 | Date Last Updated: | 01/04/2021 |

|  |  |
| --- | --- |
| Actor: | User |
| Description: | To display to the user the 5 nearest clinics by distance |
| Preconditions: | 1. User is connected to the Internet via Wi-Fi / Mobile Data  2. User’s device must have location services turned on on their device  3. User must be logged into application |
| Postconditions: | 1. User will be shown the 5 nearest clinics with the respective distances from the current location |
| Priority: | High |
| Frequency of Use: | 1-3 times a month |
| Flow of Events: | 1. User selects the “Find nearest clinics” option on the application homepage  2. System will compute the distances of the 5 nearest clinics based on the user’s current location  3. System will display a list of 5 nearest clinics based on distance, sorted in descending order |
| Alternative Flows: | - |
| Exceptions: | - |
| Includes: | 1. Consolidate information |
| Extends: | 1. Display shortest route to selected clinic from current location |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | U10 | | |
| Use Case Name: | Display clinic by name | | |
| Created By: | David Tay | Last Updated By: | David Tay |
| Date Created: | 28/01/2021 | Date Last Updated: | 01/04/2021 |

|  |  |
| --- | --- |
| Actor: | User |
| Description: | Returns a list of clinics that names matches the user’s search condition |
| Preconditions: | 1. User is connected to the Internet via Wi-Fi / Mobile Data  2. User’s device must have location services turned on on their device  3. User must be logged into application |
| Postconditions: | 1. User will be given a list of clinics that names matches the search condition |
| Priority: | High |
| Frequency of Use: | 1-3 times per month |
| Flow of Events: | 1. User will input a search phrase in the search bar  2. User will select button to find clinics from input  3. System will retrieve the travelling time from the user’s location to each clinic using the Google Maps API  4. System will display a list of clinics that fulfill the search requirement and their respective travelling time |
| Alternative Flows: | - |
| Exceptions: | - |
| Includes: | 1. Consolidate information |
| Extends: | 1. Display shortest route to selected clinic from current location |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | U11 | | |
| Use Case Name: | Display shortest route to selected clinic from current location | | |
| Created By: | David Tay | Last Updated By: | David Tay |
| Date Created: | 28/01/2021 | Date Last Updated: | 01/04/2021 |

|  |  |
| --- | --- |
| Actor: | User |
| Description: | Displays the shortest route to the selected clinic from current location |
| Preconditions: | 1. User is connected to the Internet via Wi-Fi / Mobile Data  2. User’s device must have location services turned on on their device  3. User must be logged into application |
| Postconditions: | 1. User will be shown the shortest route to the selected clinic from current location |
| Priority: | High |
| Frequency of Use: | 1-3 times a month |
| Flow of Events: | 1. User will select a clinic from the list of clinics provided  2. System will display a map with the selected clinic and current location marked with a pin and a line displaying the shortest route |
| Alternative Flows: | - |
| Exceptions: | - |
| Includes: | - |
| Extends: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | U12 | | |
| Use Case Name: | Consolidate Information | | |
| Created By: | David Tay | Last Updated By: | David Tay |
| Date Created: | 28/01/2021 | Date Last Updated: | 01/04/2021 |

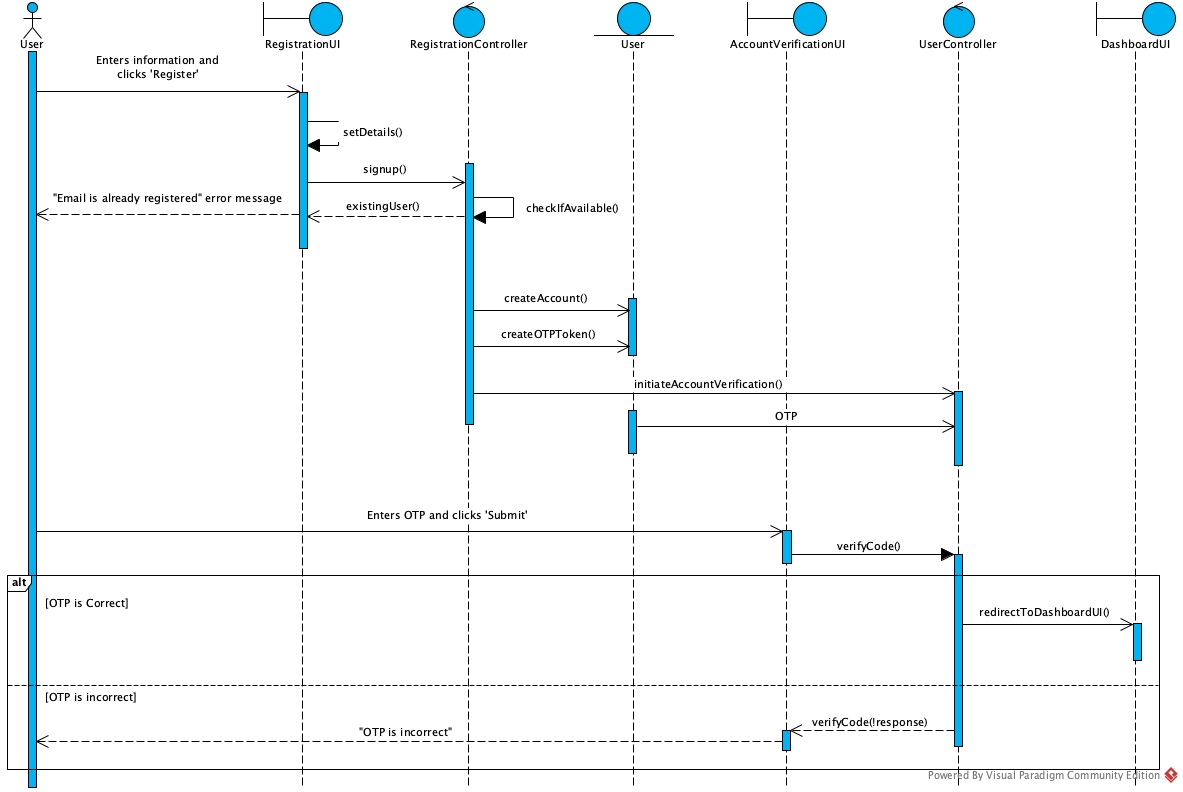
|  |  |
| --- | --- |
| Actor: | Google Maps API |
| Description: | Retrieve list of clinics, location and distance from Google Maps API |
| Preconditions: | 1. User’s device is connected to the Internet via Wi-Fi / Mobile Data  2. User’s device must have location services turned on on their device |
| Postconditions: | 1. Application will receive list of clinics and their details from Google Maps API  2. Application will receive user’s current location from Google Maps API |
| Priority: | High |
| Frequency of Use: | Dependent on number of application users |
| Flow of Events: | 1. System will retrieve list of clinics and their details from Google Maps API  2. System will receive user’s current location from Google Maps API  3. System will store all information within the database for further use |
| Alternative Flows: | - |
| Exceptions: | - |
| Includes: | - |
| Extends: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

## 2.3 Class Diagram

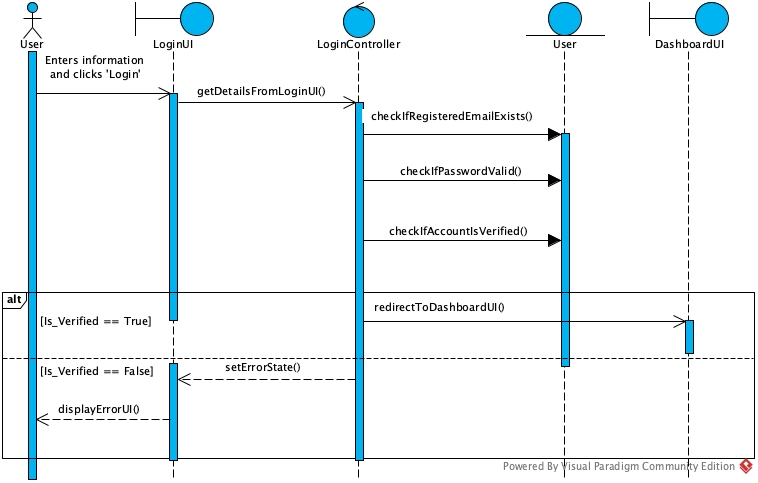
## 

## 2.4 Sequence Diagrams

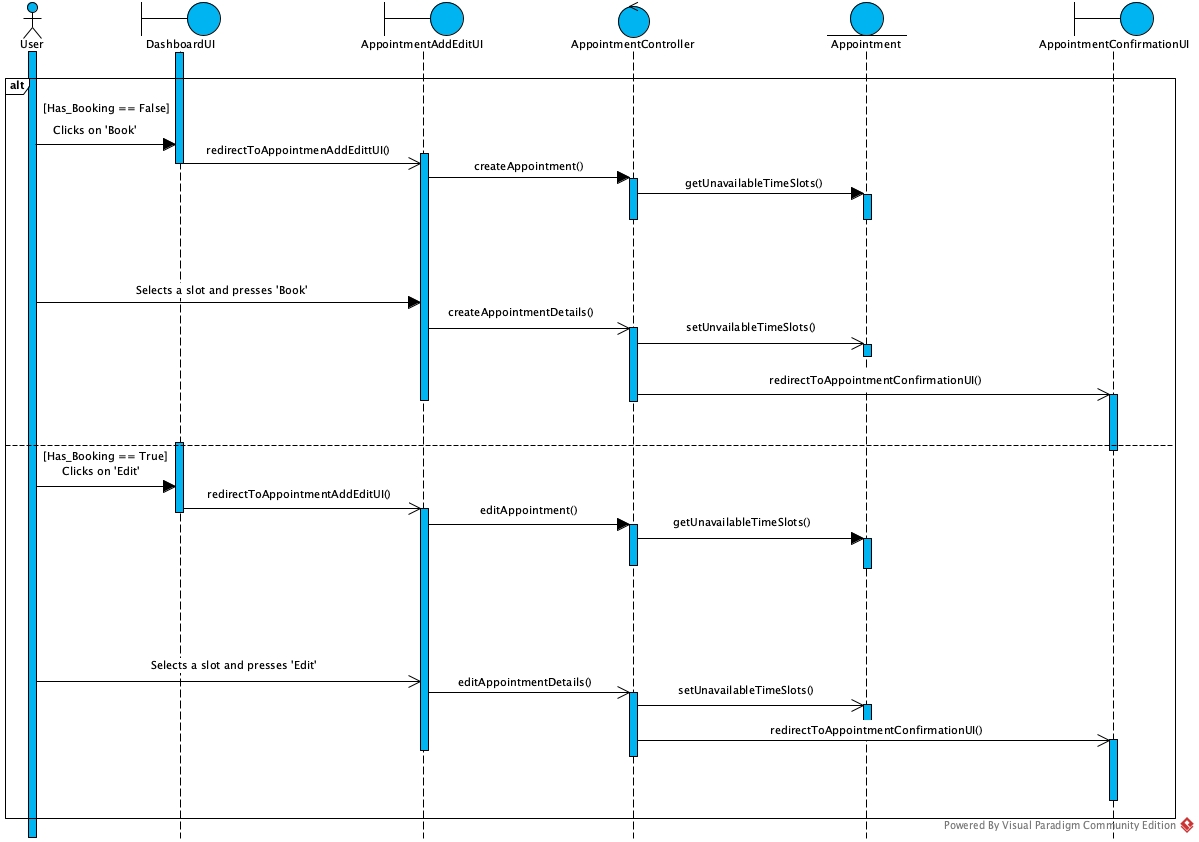
2.4.1 User Registration Sequence Diagram



2.4.2 User Login Sequence Diagram



2.4.3 Create/Edit Appointment Sequence Diagram



# 

# 

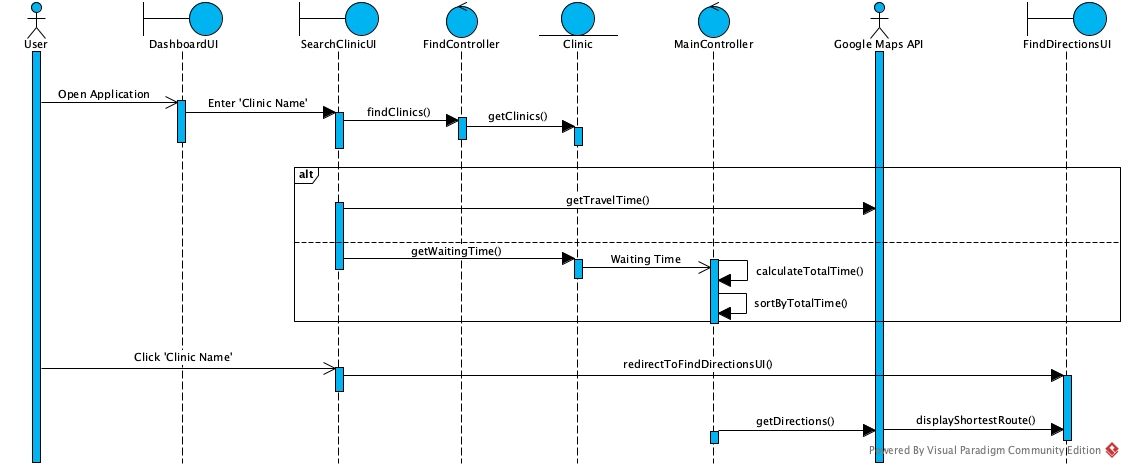
# 

# 

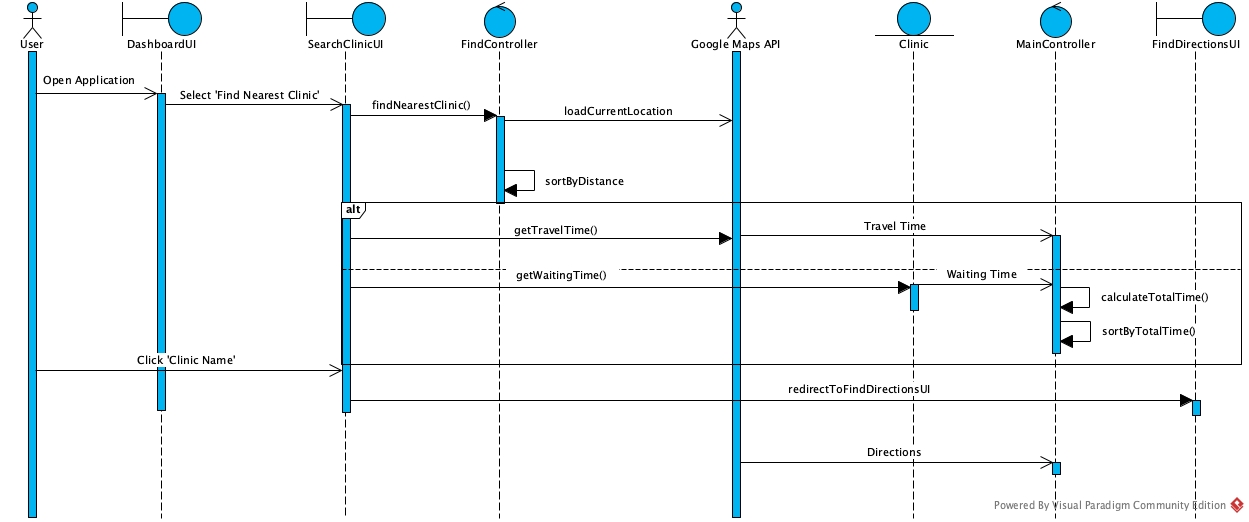
# 

# 

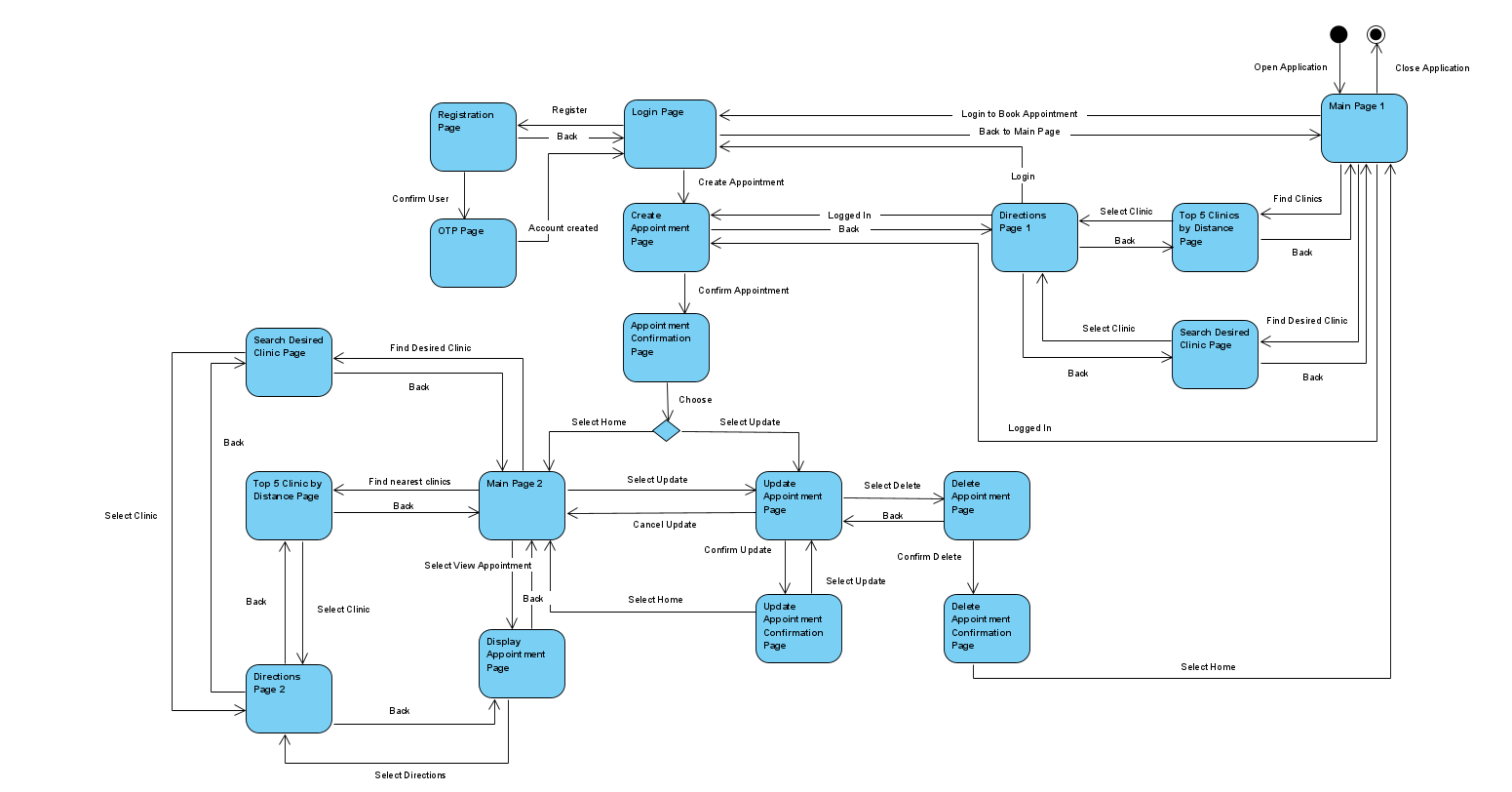
2.4.4 Search by Clinic name Sequence Diagram



2.4.5 Find Nearest Clinic Sequence Diagram



## 2.5 Dialog Map



**3 Non Functional Requirements**

## 3.1 Usability Requirements

3.1.1 The system must strive to be universally usable

3.1.1.1 The system must support different languages

3.1.2 The system must try to reduce memory load

3.1.2.1 The UI must be kept simple while allowing users to find desired clinic

3.1.2.2 The users require minimal instructions to use the system

3.1.3 The system must strive for consistency

3.1.3.1 A consistent series of actions to perform similar types of tasks such as

booking a consultation through the three different access points

3.1.3.2 A consistent design of UI must be achieved (Colors, Buttons, Fonts etc.)

## 3.2 Reliability Requirements

3.2.1 The system must not crash when user opens the application

3.2.2 The system must maintain information consistency

3.2.3 The system must be able to restart to full functionality within 10 minutes of crashing

3.2.4 The system must be able to work at any time of the day

3.2.5 The system must not lose any information

3.2.5.1 The system must not lose appointment information

3.2.5.2 The system must not lose user information

## 3.3 Performance Requirements

3.3.1 The system should take no more than 3 seconds to load after every action by the user

3.3.2 The system must offer informative feedback

3.3.2.1 System must display an error message that allows the users to know what went wrong when an error occurs

3.3.3 The system must allow easy reversal of the user’s actions

3.4 Supportability Requirements

3.4.1 The system must have ease of maintainability

3.4.1.1 Object-Oriented Design to ensure that debugging or maintenance is affordable and efficient

3.4.1.2 Test-cases must be comprehensive but succinct

3.4.2 The system must work across different platforms

3.4.2.1 Built on hybrid app development so that it can work across different mobile operating systems

3.4.3 The system must be portable

3.4.3.1 The app can be migrated from legacy system to a new platform in

accordance to technology changes/updates

3.5 Security Requirements

3.5.1 The system must ensure password is secure

3.5.1.1 To mask the input password characters during login

3.5.1.2 To mask password inputs during password creation at registration

3.5.1.3 To ensure password meets stringent criteria (i.e minimum length 8, one

special character etc.)

3.5.2 The system must ensure accounts’ information is inaccessible by unauthorised

personnel

3.5.2.1 The account information is only accessible by the system administrators

3.5.2.2 The account information is encrypted in database

**4 Architecture Design**

## 4.1 System Architecture Diagram

3-layered architecture:

1. Presentation Layer: user interface and communication layer of the application, which the end user uses to interact with the application. Its main purpose is to display and collect information to and from the user
2. Logic Layer: contains all the functional business logic, which controls the application’s functionality by performing detailed processing
3. Persistence Layer: also known as the database layer, this layer stores and manages all of the information processed by the application

Main Advantage

Easily extend, modularize, and configure the layers of the application accordingly, without impacting the other layers.

Functional process logic, data access, computer data storage and user interface are developed and maintained as separate and independent modules.

Other Advantages

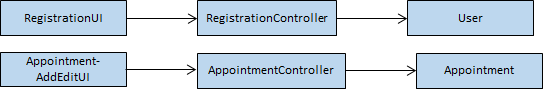
Layering the system allows us to focus more on our areas of expertise (Front-end designed by the more creative, back-end by those who are more familiar with APIs and routing). It also gives developers the ability to scale up and out (persistence layer/ back end can be deployed to a variety of databases, scale up by adding multiple web servers). Furthermore, it enhances reliability and gives more independence of underlying servers or services. Ease of code maintenance, as changes made in one layer do not affect the other layers

4.2 Design Patterns

1. Factory Pattern (Creational)

Application contains a set of classes that may or may not be instantiated until runtime. These include the User class, Appointment class etc.

Our solution is to define an interface for creating different users or appointments, without knowing beforehand what sort of user information is entered or the details of the appointment.



The creation of Users and Appointments are encapsulated. Also, it is easy to replace, edit, add new subclasses, and change object creation logic, by just simply changing the parameters in the User/Appointment data.

1. Façade Design Pattern

Application hides the complexities of the system and provides an interface to the users.



In React Native, we utilised stack navigator to provide a unified interface to a set of interfaces. Through the façade design pattern, the code is easier to understand, and reduce class dependency.

1. SOLID Principle

Classes abide by the Single Responsibility Principle. All classes are tied to one entity in the system. All classes have a single responsibility.

**5 Data Dictionary**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| User | User refers to a person who is using the application to search for clinics and schedule appointments |
| System | System refers to the mobile application |
| Appointment | Appointment refers to a medical consultation at selected clinic |
| Search | Search is a feature that allows users to find their desired clinic based on certain filters |
| Travel Time | Travel Time refers to the amount of time taken for the user to travel from their current location to the selected clinic |
| GPS | Global Positioning System which is used to pinpoint the user’s current location |
| Geolocation | Geolocation refers to the exact geographic location of the user, found using GPS |

# 

# 

# 

# 

# 

# 

# **6 Testing**

## 6.1 Black Box Testing Overview

**Login**

|  |  |  |
| --- | --- | --- |
| Scenario | Expected Result | Actual Result |
| User has an existing account and login with correct credentials | Successful login and system redirects user to DashBoardUI | Successful login and system redirects user to DashBoardUI |
| User has an existing account but login with incorrect credentials | Unsuccessful login and system prompts users to try again | Unsuccessful login and system prompts users to try again |
| User did not fill in all required fields | Unsuccessful login and system prompts user to fill in all required fields | Unsuccessful login and system prompts user to fill in all required fields |

|  |  |  |  |
| --- | --- | --- | --- |
| Email | Password | Expected Result | Actual Result |
| sheryl\_gtyinn@hotmail.com | (test) | Successful login | Successful login |
| sheryl\_gtyinn@hotmail.com | test | Wrong Username/ Password! | Wrong Username/ Password! |
| wronguser@hotmail.com | (test) | Wrong Username/ Password! | Wrong Username/ Password! |
| sheryl\_gtyinn@hotmail.com | Empty | Enter all fields please! | Enter all fields please! |
| Empty | (test) | Enter all fields please! | Enter all fields please! |

**Registration**

|  |  |  |
| --- | --- | --- |
| Scenario | Expected Result | Actual Result |
| User does not have existing account and enters all required fields correctly | Successful creation of account and system redirects user to DashBoardUI | Successful creation of account and system redirects user to DashBoardUI |
| User has an existing account and tries to re-register | Unsuccessful creation of account and system shows an error message that email is already registered | Unsuccessful creation of account and system shows an error message that email is already registered |
| User enters passwords that do not match | Unsuccessful creation and system shows an error message to user that passwords do not match | Unsuccessful creation and system shows an error message to user that passwords do not match |
| User did not fill in all required fields | Unsuccessful creation and system prompts user to fill in all required fields | Unsuccessful creation and system prompts user to fill in all required fields |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| First Name | Last Name | Email | Password | Confirm Password | Expected Result | Actual Result |
| testFirst | testLast | test@gmail.com | (test) | (test) | Success! | Success! |
| testFirst | testLast | user@gmail.com | (test) | (test) | Email already registered! | Email already registered! |
| testFirst | testLast | test@gmail.com | (test) | test | Passwords do not match | Passwords do not match |
| Empty | testLast | test@gmail.com | (test) | (test) | Enter all fields please! | Enter all fields please! |
| testFirst | Empty | test@gmail.com | (test) | (test) | Enter all fields please! | Enter all fields please! |
| testFirst | testLast | Empty | (test) | (test) | Enter all fields please! | Enter all fields please! |
| testFirst | testLast | test@gmail.com | Empty | (test) | Enter all fields please! | Enter all fields please! |
| testFirst | testLast | test@gmail.com | (test) | Empty | Enter all fields please! | Enter all fields please! |

**Create Appointment**

|  |  |  |
| --- | --- | --- |
| Scenario | Expected Result | Actual Result |
| User enters all required fields | Successful creation of appointment and system redirects user to confirmAppointmentUI | Successful creation of appointment and system redirects user to confirmAppointmentUI |
| User did not enter all required fields | Unsuccessful creation of appointment and system prompts user to fill in all required fields | Unsuccessful creation of appointment and system prompts user to fill in all required fields |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Clinic Name | Date | Time | Other Comments | Expected Result | Actual Result |
| Chen Family Clinic | Apr 1 2021 | 12:00 | test | Success! | Success! |
| Chen Family Clinic | Apr 1 2021 | Empty | test | Enter All Fields! | Enter All Fields! |

**Edit Appointment**

|  |  |  |
| --- | --- | --- |
| Scenario | Expected Result | Actual Result |
| User enters all required fields | Successful update of appointment and system redirects user to confirmAppointmentUI | Successful update of appointment and system redirects user to confirmAppointmentUI |
| User did not enter all required fields | Unsuccessful update of appointment and system prompts user to fill in all required fields | Unsuccessful update of appointment and system prompts user to fill in all required fields |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Clinic Name | Date | Time | Other Comments | Expected Result | Actual Result |
| Chen Family Clinic | Apr 2 2021 | 12:00 | test | Success! | Success! |
| Chen Family Clinic | Apr 2 2021 | Empty | test | Enter All Fields! | Enter All Fields! |

## 6.2 Black Box Testing Framework - Jest

Since black box testing revolves around testing the functionalities of the applications without the knowledge of its internal structure, we wrote test cases using Jest, a frontend testing framework for React Native. Jest is the perfect tool to conduct black box testing given that frontend testing involves testing from the perspective of our end-users who most likely have no idea of the inner business logic of the application. We also used a plugin, Jest-stare, which generates a html report which organizes the details of all the test cases and its statuses everytime Jest is run.

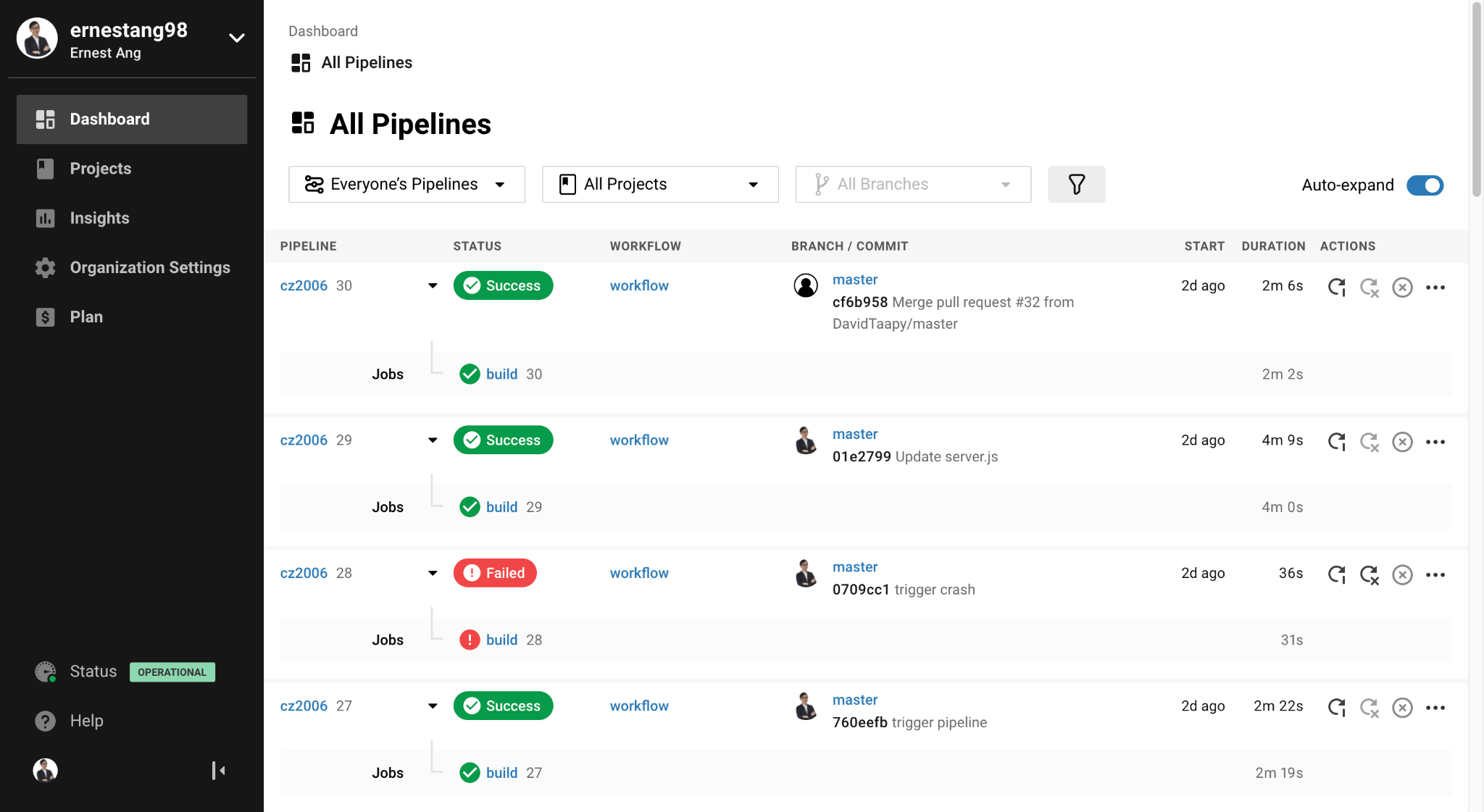
## 6.3 White Box Testing

## 6.4 White Box Testing Framework - Mocha

Since white box testing revolves around testing the functionalities of the applications with the knowledge of its internal structure, we wrote test cases using Mocha, a backend testing framework which uses Chai, a Test Driven Development (TDD) and Behavior Driven Development (BDD) assertion library for Node.js. With Mocha, we are able to test our business logic layer thoroughly which is in charge of managing communications between the presentation layer which users interact with and the database layer or the APIs. Similar to our report generator plugin used with Jest, we leveraged on Mocha Awesome, a report generator plugin for Mocha which organizes all of the test cases and its statuses everytime Mocha is run.

## 6.5 Automated Testing with Continuous Integration (CI) Pipelines

Typically, after softwares are deployed into production on servers or hosting platforms such as Amazon Web Services or Heroku, developers that wish to integrate new features or perhaps commit new changes to fix any bugs surfaced by users would typically run the above mentioned test cases before integrating their code in order to have assurance that their changes will not affect any of the previous functionalities or cause crashes when these updates are pushed into production. CI pipelines can be constructed to aid developers to automatically run written test cases everytime commits are made. Our group used CircleCI to set up our CI pipeline and the building steps of our pipeline are configured using YAML files.



# **7 Appendix**

7.1 Software Requirement Specification

7.2 Youtube Video Link For Demo

<https://youtu.be/07jsQJjUuI8>

Select 1080p for better quality