Concept Development

The application I have developed is a public transport application, named as SGTravel. This application helps to provide information regarding buses and trains in Singapore, such as bus stops, available time and bus and train number. It also provides user's current location and public transport arrival time. There are many other similar public transport applications such as SG BusLeh, SingBus and SimplyGo. However, most of the applications only provide bus information instead of both bus and train information. Also, some of the applications' UI are unclear to the users, for example, the color of the UI is too blend. Thus, I wanted to create a better designed looking public transport application.

Wireframing

Flow chart:

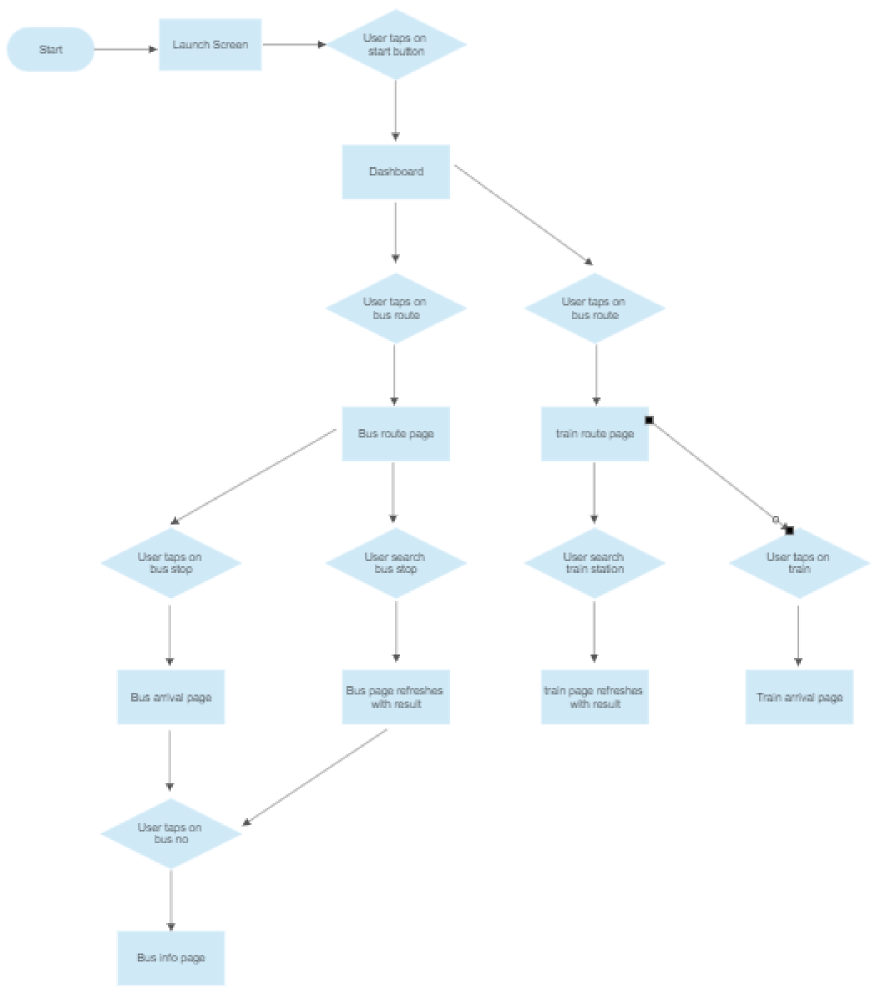
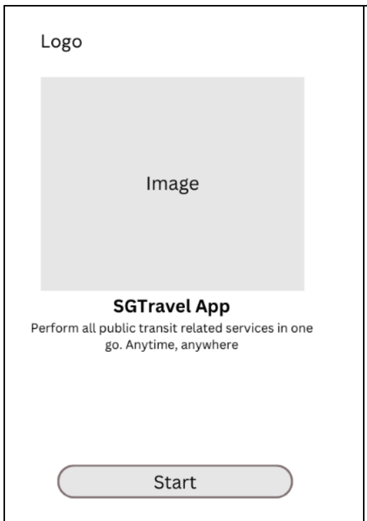
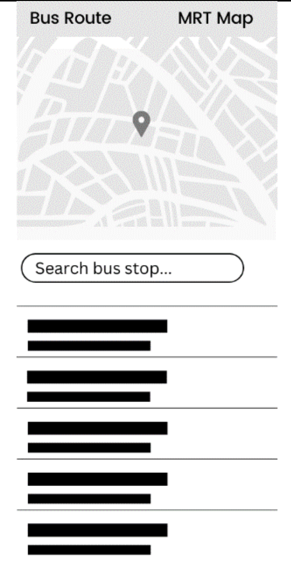
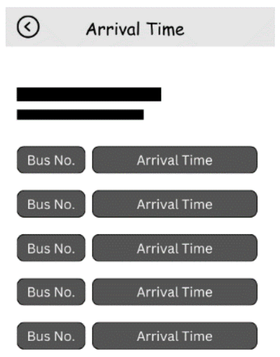
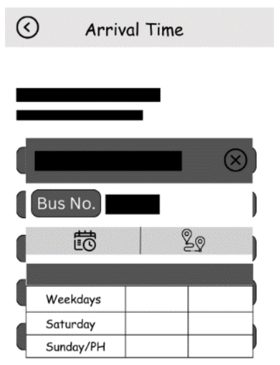


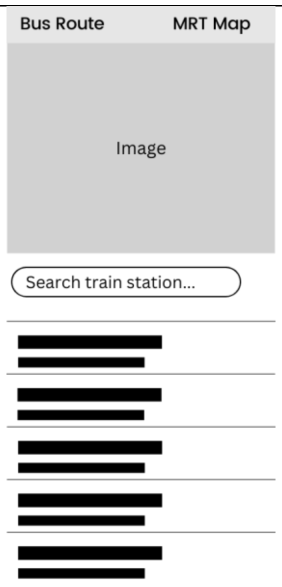
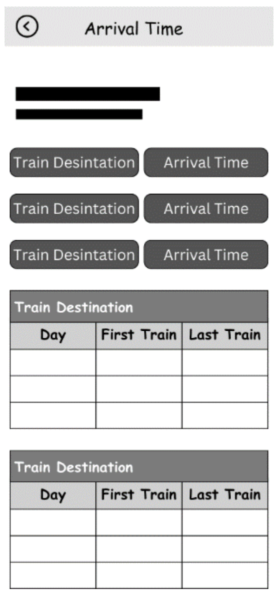
Figure 1: SGTravel_Flowchart

Figure 1 shows the flowchart of the application, describing the steps of the whole application process. The user will be first directed to the launch screen, followed by the dashboard screen after tapping the start button. In the dashboard screen, the user can either tap on the bus route page or the train route page. Both pages have the same features such as search through the data list and display the arrival time page based on what the user has selected from the list. The only different is that there is a bus info page for the bus services to show the route and working hours of each bus service.

Wireframes:

 Logo Image SGTravel App Perform all public transit related services in one go. Anytime, anywhere Start	Launch page. Description: The first page user will see after launching the application. The page shows a short description about the application. User must click on start button to go to the dashboard page.
	Dashboard/Bus route page. Description: After the user has tapped the start button at the launch page, he/she will be directed to the dashboard, also known as bus route page. The page displays the user's current location on the map and the list of bus stops, with names and bus stop codes. User can search through the list by inputting the key value on the search bar. Also, user can tap on the bus stop from the list to go to bus arrival page.

 Arrival Time Bus No. Arrival Time Bus No. Arrival Time Bus No. Arrival Time Bus No. Arrival Time Bus No. Arrival Time	Bus Arrival page. Description: This page displays each bus's estimated arrive time at the bus stop based on what the user has selected. User can also tap on the bus number to check the bus service working hours and routes.
 Arrival Time Bus No. Weekdays Saturday Sunday/PH	Bus info page. Description: After the user has tapped the bus number from the bus arrival page, a pop up will appeared and displayed the bus service working hours and route.

	Train route page. Description: This page displays the train map and list of train stations, with stations names and stations codes. User can search through the list by inputting key value in the search bar. Also, user can tap on the train station to go to train arrival page.
 Arrival Time Train Desintation Arrival Time Train Desintation Arrival Time Train Desintation Arrival Time Train Destination Day First Train Last Train Train Destination Day First Train Last Train	Train arrival page. Description: This page displays each train's estimated arrive time at the train station based on what the user has selected. It also displays the train route and working hours.

User Feedback

Feedback 1:	The layouts look very neat and easy to understand. Not much content is squeezed into one single page, thus easy to read.
Feedback 2:	Just by looking at the wireframes, I can tell what the application is about and the purposes of it. The layouts look user friendly, do not include too many steps to go the features. The contents are also less wordy, only show the important information, which is good and easy to read.
Feedback 3:	Wireframes are simple and neat, easy to navigate.

Prototyping

During prototyping, I have chosen less damaging to eyes colors for the UI layout as the application is used by all users regardless of ages. This would help the users to read the text easily.

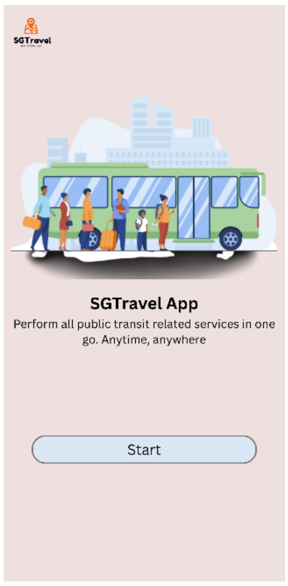

SGTravel
SGTravel App
Perform all public transit related services in one go. Anytime, anywhere
Start

Figure 2: LaunchScreen

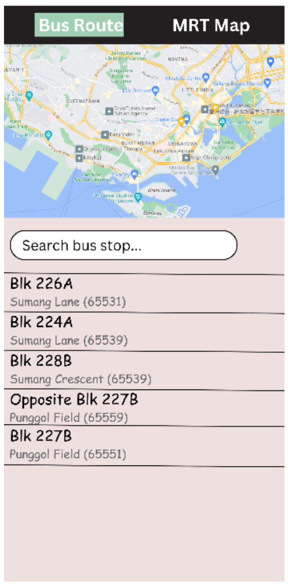


Figure 3: Dashboard/Bus route

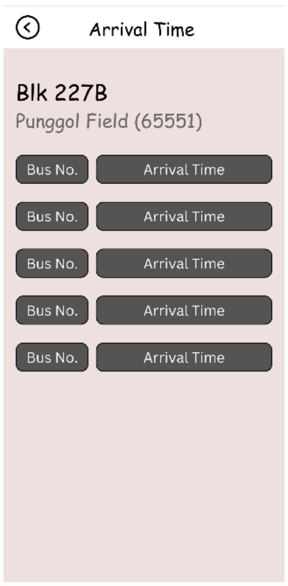


Figure 4: Bus arrival

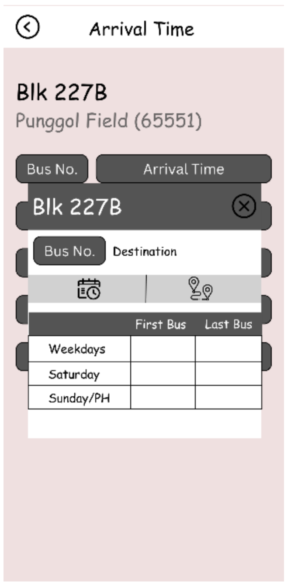


Figure 5: Bus info pop up

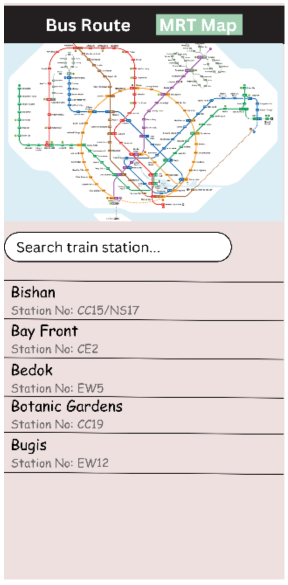


Figure 6: Train route

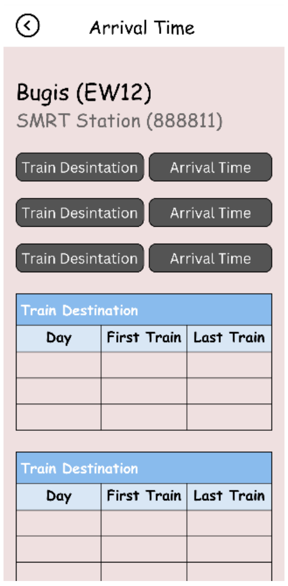


Figure 7: Train arrival page

Development

Launch page:

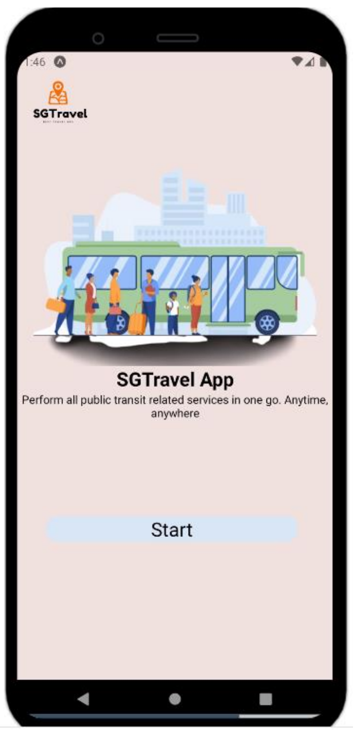

1:46
SGTravel
SGTravel App
Perform all public transit related services in one go. Anytime, anywhere
Start

Figure 8: Launch Screen (Android)

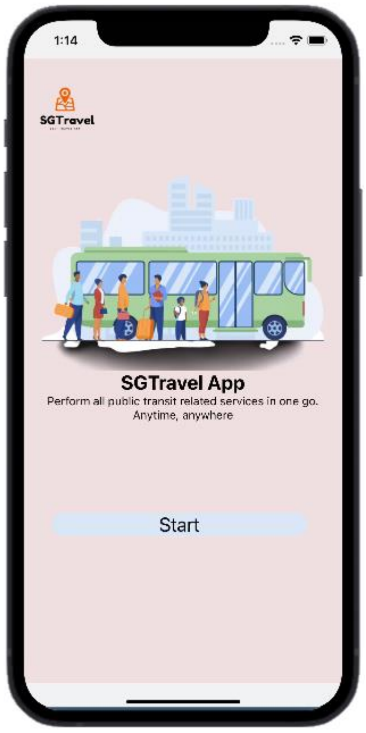
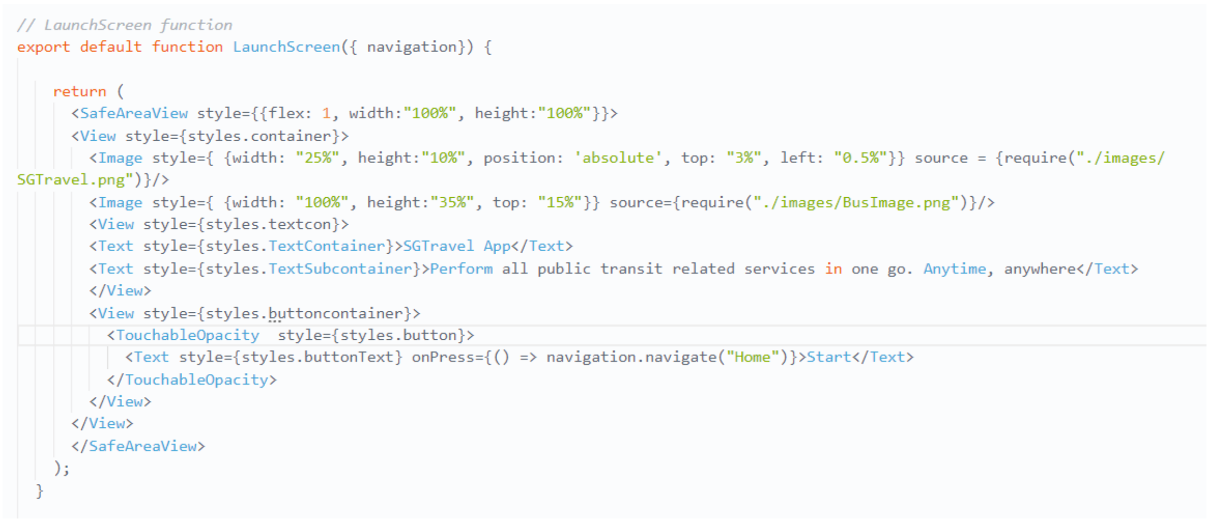

1:14
SGTravel
SGTravel App
Perform all public transit related services in one go. Anytime, anywhere
Start

Figure 9: Launch Screen (ios)

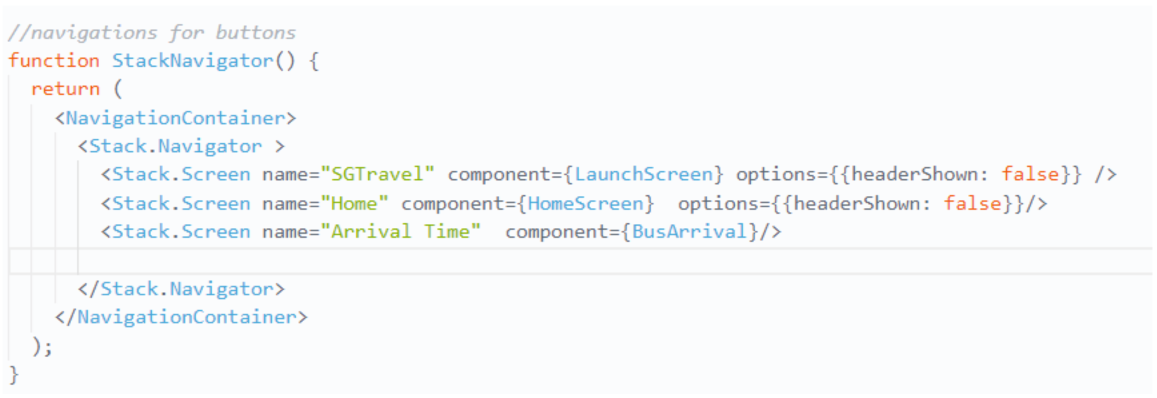
In the development, I have created LaunchScreen function in the index.js to display the UI layout using view, text, touchableopacity as shown below.



```
// LaunchScreen function
export default function LaunchScreen({ navigation}) {

    return (
      <SafeAreaView style={{flex: 1, width:"100%", height:"100%"}}>
      <View style={styles.container}>
        <Image style={ {width: "25%", height:"10%", position: 'absolute', top: "3%", left: "0.5%"}} source = {require("./images/
SGTravel.png")}/>
        <Image style={ {width: "100%", height:"35%", top: "15%"}} source={require("./images/BusImage.png")}/>
        <View style={styles.textcon}>
        <Text style={styles.TextContainer}>SGTravel App</Text>
        <Text style={styles.TextSubcontainer}>Perform all public transit related services in one go. Anytime, anywhere</Text>
        </View>
        <View style={styles.buttoncontainer}>
          <TouchableOpacity  style={styles.button}>
            <Text style={styles.buttonText} onPress={() => navigation.navigate("Home")}>Start</Text>
          </TouchableOpacity>
        </View>
      </View>
      </SafeAreaView>
    );
  }
```

Figure 10: LaunchScreen function

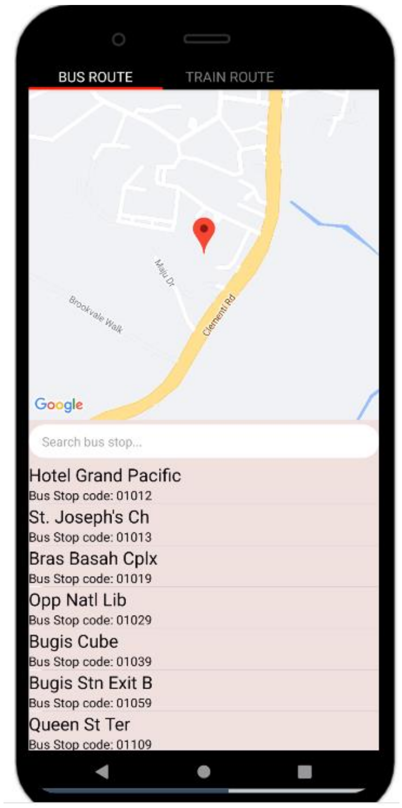
As for the navigation to other pages, I have created StackNavigator function in App.js and used Stack navigator to allow the user to tap on the button and navigate to other page. Screenshot of the code is shown as below.



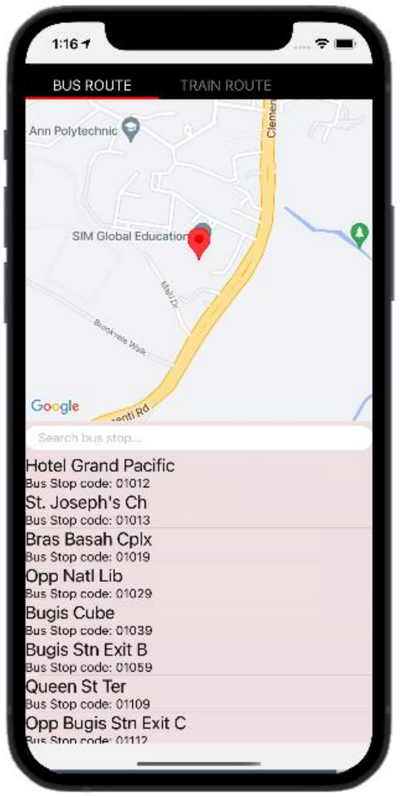
```
//navigations for buttons
function StackNavigator() {
  return (
    <NavigationContainer>
      <Stack.Navigator >
        <Stack.Screen name="SGTravel" component={LaunchScreen} options={{headerShown: false}} />
        <Stack.Screen name="Home" component={HomeScreen}  options={{headerShown: false}}/>
        <Stack.Screen name="Arrival Time"  component={BusArrival}/>

      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

Figure 11: StackNavigator function

Dashboard/Bus route page:



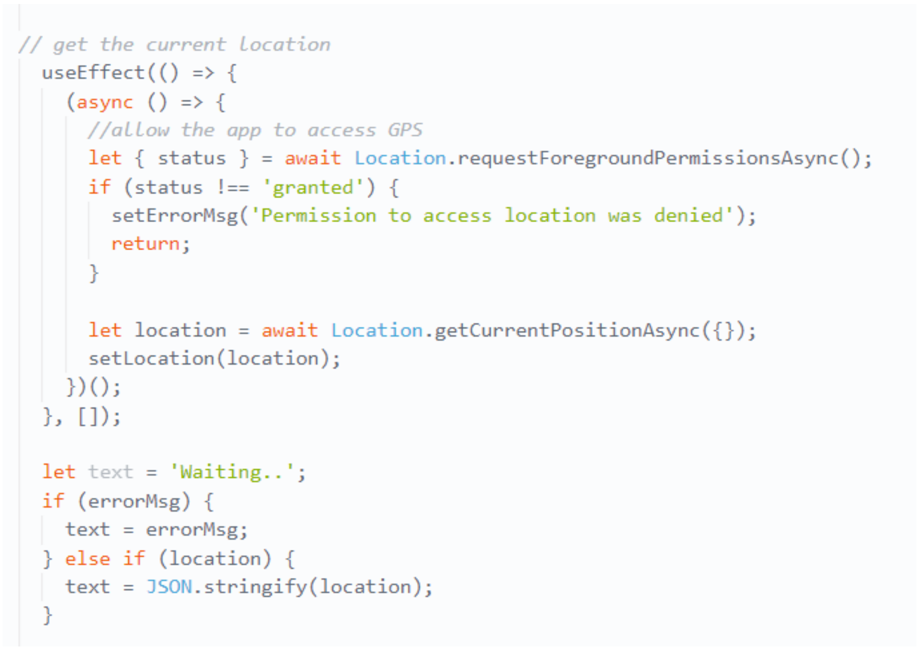
Hotel Grand Pacific
Bus Stop code: 01012
St. Joseph's Ch
Bus Stop code: 01013
Bras Basah Cplx
Bus Stop code: 01019
Opp Natl Lib
Bus Stop code: 01029
Bugis Cube
Bus Stop code: 01039
Bugis Stn Exit B
Bus Stop code: 01059
Queen St Ter
Bus Stop code: 01109



Hotel Grand Pacific
Bus Stop code: 01012
St. Joseph's Ch
Bus Stop code: 01013
Bras Basah Cplx
Bus Stop code: 01019
Opp Natl Lib
Bus Stop code: 01029
Bugis Cube
Bus Stop code: 01039
Bugis Stn Exit B
Bus Stop code: 01059
Queen St Ter
Bus Stop code: 01109
Opp Bugis Stn Exit C
Bus Stop code: 01112

Figure 12: Dashboard/Bus route (android) *Figure 13: Dashboard/Bus route (ios)*

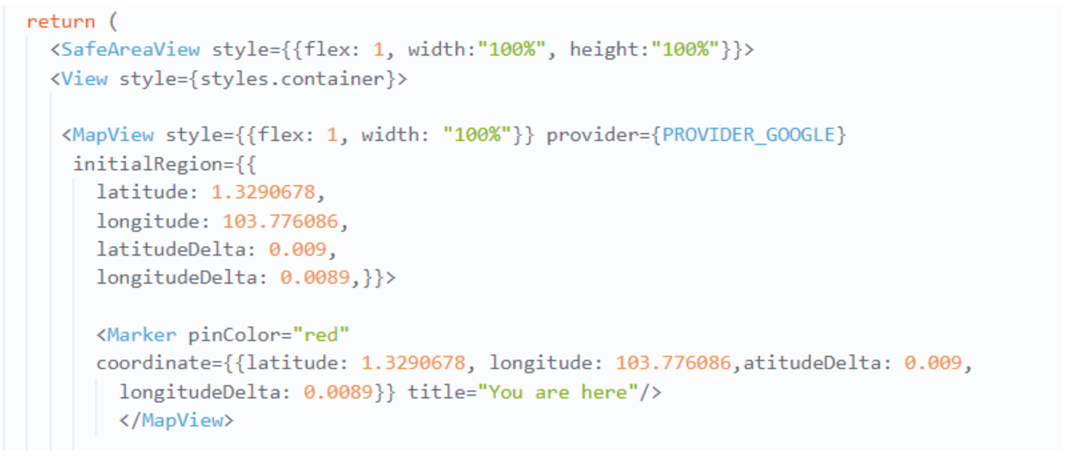
In the busInfo.js, I have implemented code to get the user's current location by using Location from expo-location as shown below.



```
// get the current location
  useEffect(() => {
    (async () => {
      //allow the app to access GPS
      let { status } = await Location.requestForegroundPermissionsAsync();
      if (status !== 'granted') {
        setErrorMsg('Permission to access location was denied');
        return;
      }

      let location = await Location.getCurrentPositionAsync({});
      setLocation(location);
    })();
  }, []);

  let text = 'Waiting..';
  if (errorMsg) {
    text = errorMsg;
  } else if (location) {
    text = JSON.stringify(location);
  }
```

Figure 14: get current location code

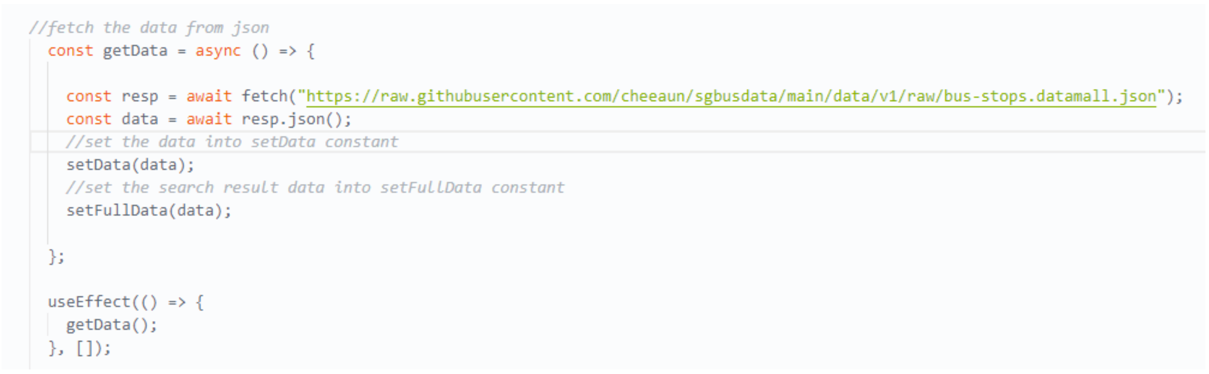
I am able to get current location data, however, as the bus stops data is from Singapore, the application might not be able to display the data based on current location if the user is outside of Singapore. Thus, I did not implement the current location data into the google map. Instead, I have used a fix location to display onto the google map as shown below.



```
return (
  <SafeAreaView style={{flex: 1, width:"100%", height:"100%"}}>
  <View style={styles.container}>

   <MapView style={{flex: 1, width: "100%"}} provider={PROVIDER_GOOGLE}
    initialRegion={{
      latitude: 1.3290678,
      longitude: 103.776086,
      latitudeDelta: 0.009,
      longitudeDelta: 0.0089,}}>

      <Marker pinColor="red"
      coordinate={{latitude: 1.3290678, longitude: 103.776086,atitudeDelta: 0.009,
        longitudeDelta: 0.0089}} title="You are here"/>
        </MapView>
```

Figure 15: display google map

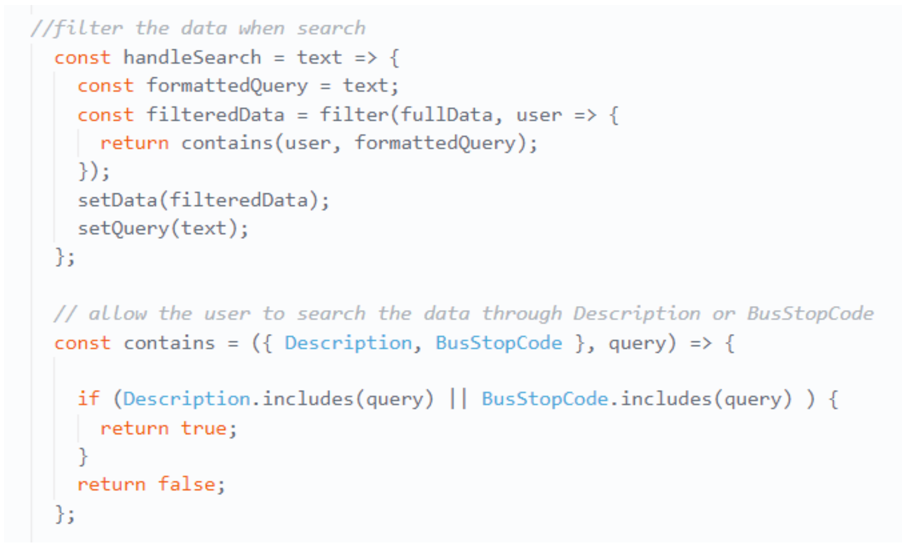
During the development, I have faced a problem when fetching the bus stops data. Supposedly, I wanted to fetch the data from the LTA API, however, there is a JSON parser error which prevented me from fetching. To solve the issue, I have to fetch the data from JSON in github as shown below. The JSON from github has the same data from LTA API.



```
//fetch the data from json
  const getData = async () => {

    const resp = await fetch("https://raw.githubusercontent.com/cheeaun/sgbusdata/main/data/v1/raw/bus-stops.datamall.json");
    const data = await resp.json();
    //set the data into setData constant
    setData(data);
    //set the search result data into setFullData constant
    setFullData(data);

  };

  useEffect(() => {
    getData();
  }, []);
```

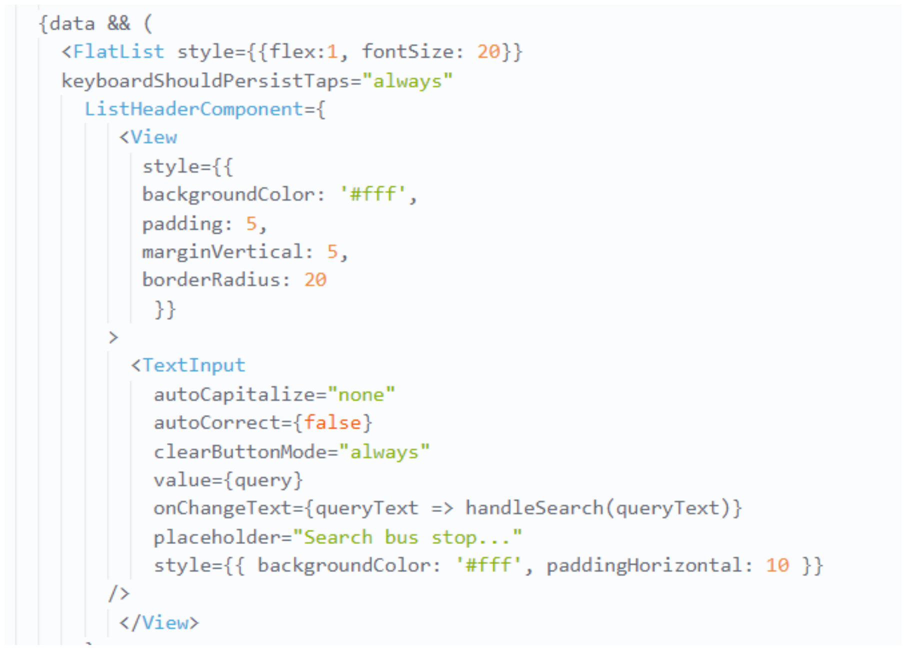
Figure 16: fetch bus stop data

I also have created handleSearch and contains constants to handle the search feature. handleSearch is used to filter the search result data from the flatlist based on key value user has inputted in the search bar. The filtered data will then be stored into setData constants and displayed it in the same flatlist. The outcome of it is the flatlist would refresh every time the value is keyed into the search bar. Contains is used to allow the user to search through based on specific data. In this case, the specific data would be bus stop name and bus stop code. Screenshot of the code is shown below.



```
//filter the data when search
  const handleSearch = text => {
    const formattedQuery = text;
    const filteredData = filter(fullData, user => {
      return contains(user, formattedQuery);
    });
    setData(filteredData);
    setQuery(text);
  };

  // allow the user to search the data through Description or BusStopCode
  const contains = ({ Description, BusStopCode }, query) => {

    if (Description.includes(query) || BusStopCode.includes(query) ) {
      return true;
    }
    return false;
  };
```

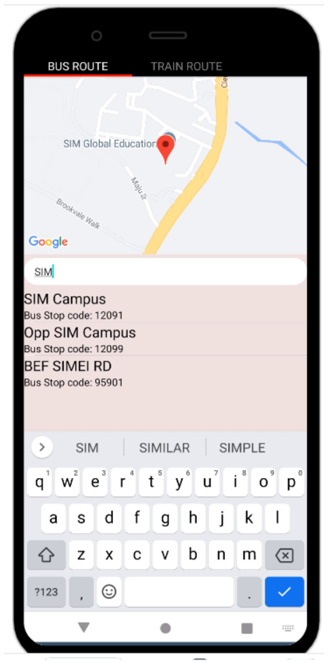
Figure 17: code for searching bus stop



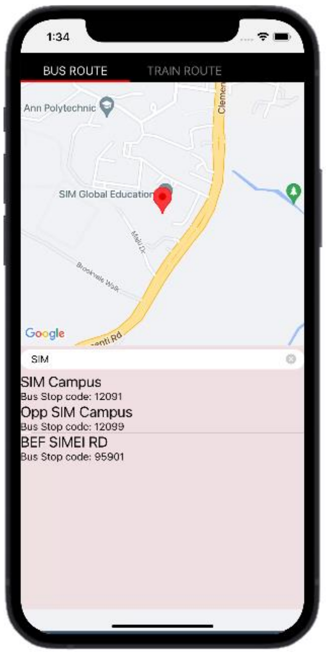
```
{data && (
  <FlatList style={{flex:1, fontSize: 20}}
  keyboardShouldPersistTaps="always"
    ListHeaderComponent={
       <View
         style={{
         backgroundColor: '#fff',
         padding: 5,
         marginVertical: 5,
         borderRadius: 20
          }}
      >
        <TextInput
          autoCapitalize="none"
          autoCorrect={false}
          clearButtonMode="always"
          value={query}
          onChangeText={queryText => handleSearch(queryText)}
          placeholder="Search bus stop..."
          style={{ backgroundColor: '#fff', paddingHorizontal: 10 }}
       />
       </View>
```

Figure 18: data and search result using Flatlist

Outcome of the search:



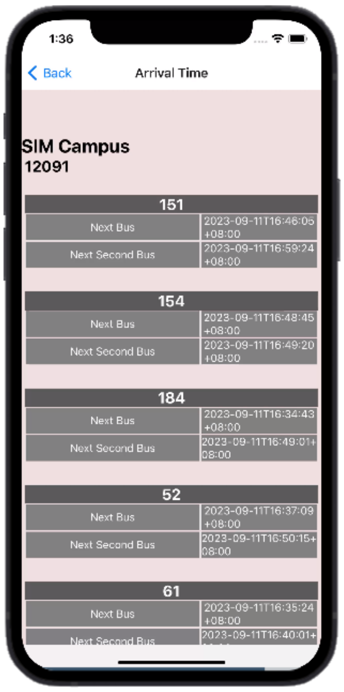
SIM Campus
Bus Stop code: 12091
Opp SIM Campus
Bus Stop code: 12099
BEF SIMEI RD
Bus Stop code: 95901



SIM Campus
Bus Stop code: 12091
Opp SIM Campus
Bus Stop code: 12099
BEF SIMEI RD
Bus Stop code: 95901

Figure 19: bus stop search result (android) *Figure 20: bus stop search result (ios)*

Bus arrival page:



SIM Campus
12091

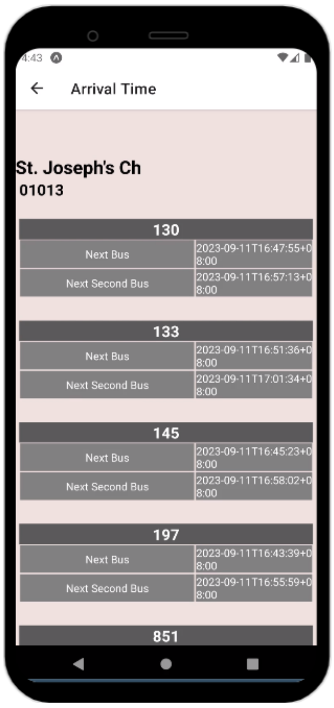
151	
Next Bus	2023-09-11T16:46:05 +08:00
Next Second Bus	2023-09-11T16:59:24 +08:00

154	
Next Bus	2023-09-11T16:48:45 +08:00
Next Second Bus	2023-09-11T16:49:20 +08:00

184	
Next Bus	2023-09-11T16:34:43 +08:00
Next Second Bus	2023-09-11T16:49:01+08:00

52	
Next Bus	2023-09-11T16:37:09 +08:00
Next Second Bus	2023-09-11T16:50:15+08:00

61	
Next Bus	2023-09-11T16:35:24 +08:00
Next Second Bus	2023-09-11T16:40:01+



St. Joseph's Ch
01013

130	
Next Bus	2023-09-11T16:47:55+08:00
Next Second Bus	2023-09-11T16:57:13+08:00

133	
Next Bus	2023-09-11T16:51:36+08:00
Next Second Bus	2023-09-11T17:01:34+08:00

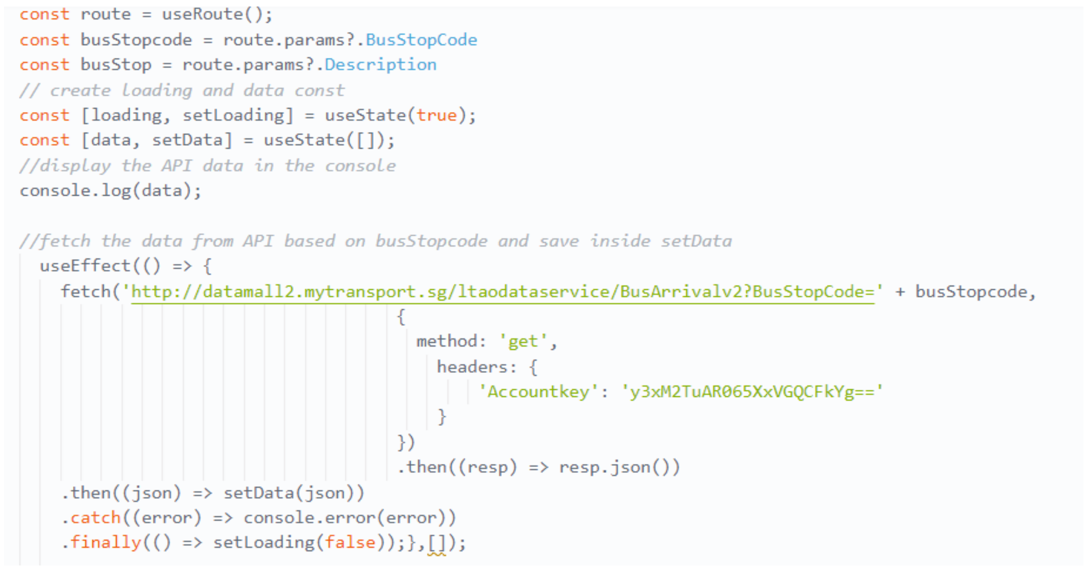
145	
Next Bus	2023-09-11T16:45:23+08:00
Next Second Bus	2023-09-11T16:58:02+08:00

197	
Next Bus	2023-09-11T16:43:39+08:00
Next Second Bus	2023-09-11T16:55:59+08:00

851

Figure 21: Bus arrival page (ios) *Figure 22: Bus arrival page (android)*

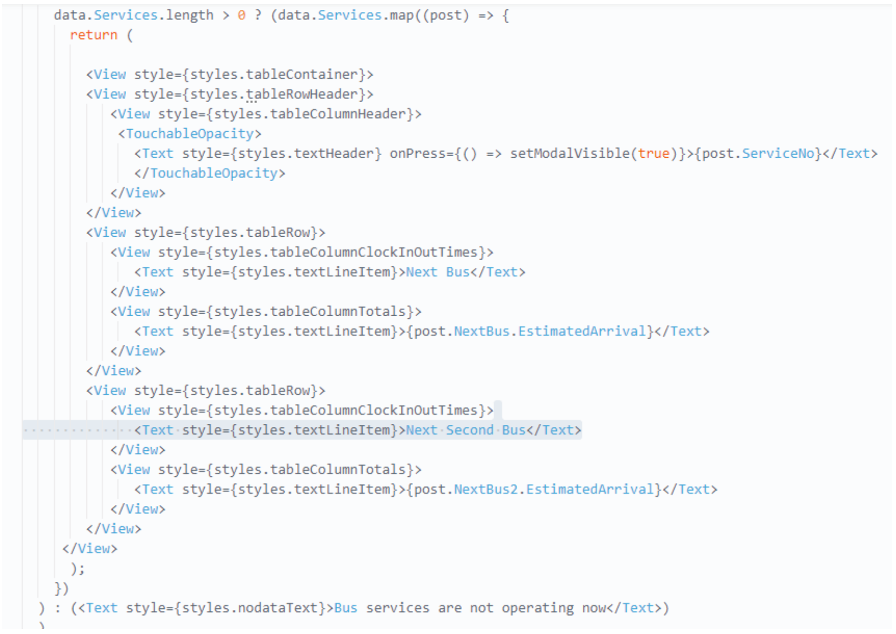
During the development, I have fetched the bus services arrival time from LTA API based on the bus stop code. I have used the bus stops data which I have displayed in the bus route page for this bus arrival page by using useRoute(). Screenshot of the code is shown below.



```
const route = useRoute();
const busStopcode = route.params?.BusStopCode
const busStop = route.params?.Description
// create loading and data const
const [loading, setLoading] = useState(true);
const [data, setData] = useState([]);
//display the API data in the console
console.log(data);

//fetch the data from API based on busStopcode and save inside setData
  useEffect(() => {
    fetch('http://datamall2.mytransport.sg/ltaodataservice/BusArrivalv2?BusStopCode=' + busStopcode,
                                       {
                                         method: 'get',
                                           headers: {
                                               'Accountkey': 'y3xM2TuAR065XxVGQCFkYg=='
                                           }
                                       })
                                       .then((resp) => resp.json())
    .then((json) => setData(json))
    .catch((error) => console.error(error))
    .finally(() => setLoading(false));},[]);
```

Figure 23: fetch arrival API

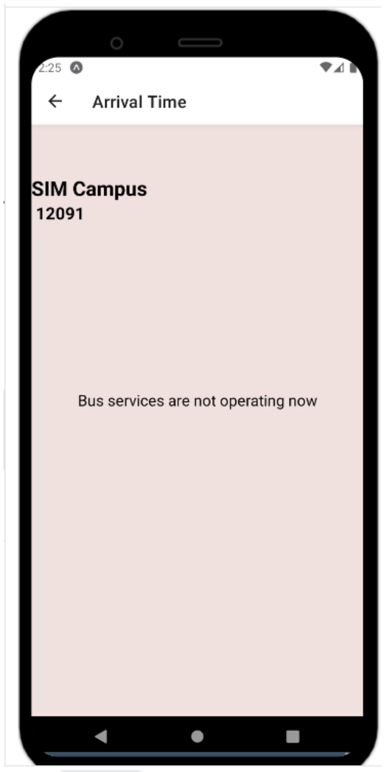
I displayed the bus service numbers and the arrival time in the table as shown below. Since the data is from a real time API, I also implemented a message when there is no data available as the bus services are not operating. I also have changed the layout of displaying the arrival time. In the prototype, I was supposed to display the arrival time on the same line as the bus service no. However, due to the API data returns me the arrival time with the date and time which is very long, I have changed the layout into table list. Screenshot of the code is shown below.



```
    data.Services.length > 0 ? (data.Services.map((post) => {
      return (

        <View style={styles.tableContainer}>
        <View style={styles.tableRowHeader}>
          <View style={styles.tableColumnHeader}>
           <TouchableOpacity>
            <Text style={styles.textHeader} onPress={() => setModalVisible(true)}>{post.ServiceNo}</Text>
            </TouchableOpacity>
          </View>
        </View>
        <View style={styles.tableRow}>
          <View style={styles.tableColumnClockInOutTimes}>
            <Text style={styles.textLineItem}>Next Bus</Text>
          </View>
          <View style={styles.tableColumnTotals}>
            <Text style={styles.textLineItem}>{post.NextBus.EstimatedArrival}</Text>
          </View>
        </View>
        <View style={styles.tableRow}>
          <View style={styles.tableColumnClockInOutTimes}>
            <Text style={styles.textLineItem}>Next Second Bus</Text>
          </View>
          <View style={styles.tableColumnTotals}>
            <Text style={styles.textLineItem}>{post.NextBus2.EstimatedArrival}</Text>
          </View>
        </View>
      </View>
       );
    })
  ) : (<Text style={styles.nodataText}>Bus services are not operating now</Text>)
  )
```

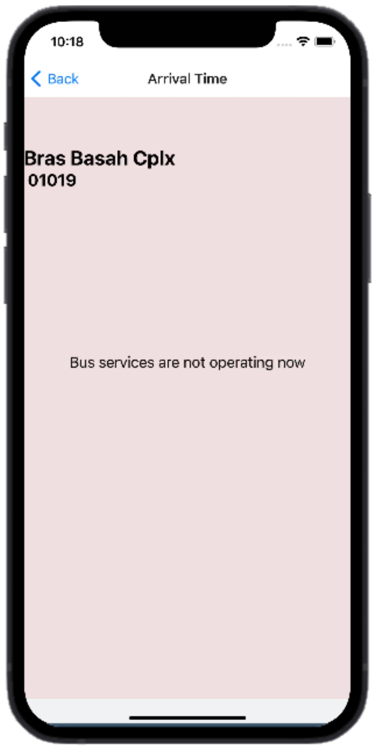
Figure 24: display API data

Outcome of no data available:



SIM Campus
12091

Bus services are not operating now



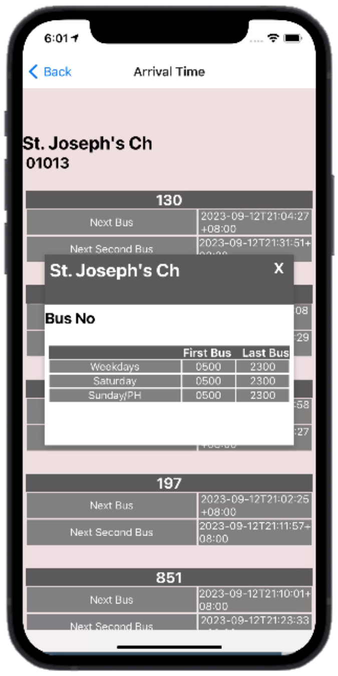
Bras Basah Cplx
01019

Bus services are not operating now

Figure 25: no arrival time (android)

Figure 26: no arrival time(ios)

Bus info pop up:



St. Joseph's Ch
01013

130	
Next Bus	2023-09-12T21:04:27+08:00
Next Second Bus	2023-09-12T21:31:51+

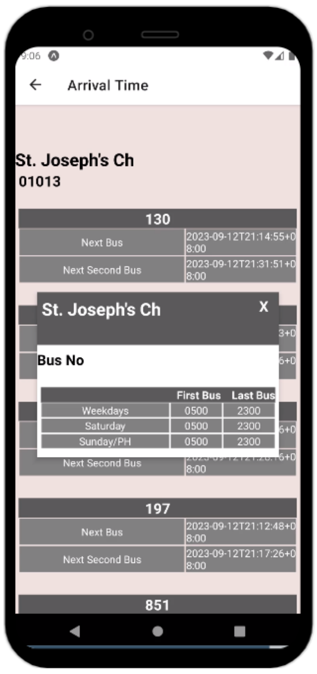
St. Joseph's Ch X

Bus No

	First Bus	Last Bus
Weekdays	0500	2300
Saturday	0500	2300
Sunday/PH	0500	2300

197	
Next Bus	2023-09-12T21:02:25+08:00
Next Second Bus	2023-09-12T21:11:57+08:00

851	
Next Bus	2023-09-12T21:10:01+08:00
Next Second Bus	2023-09-12T21:23:33



St. Joseph's Ch
01013

130	
Next Bus	2023-09-12T21:14:55+08:00
Next Second Bus	2023-09-12T21:31:51+08:00

St. Joseph's Ch X

Bus No

	First Bus	Last Bus
Weekdays	0500	2300
Saturday	0500	2300
Sunday/PH	0500	2300

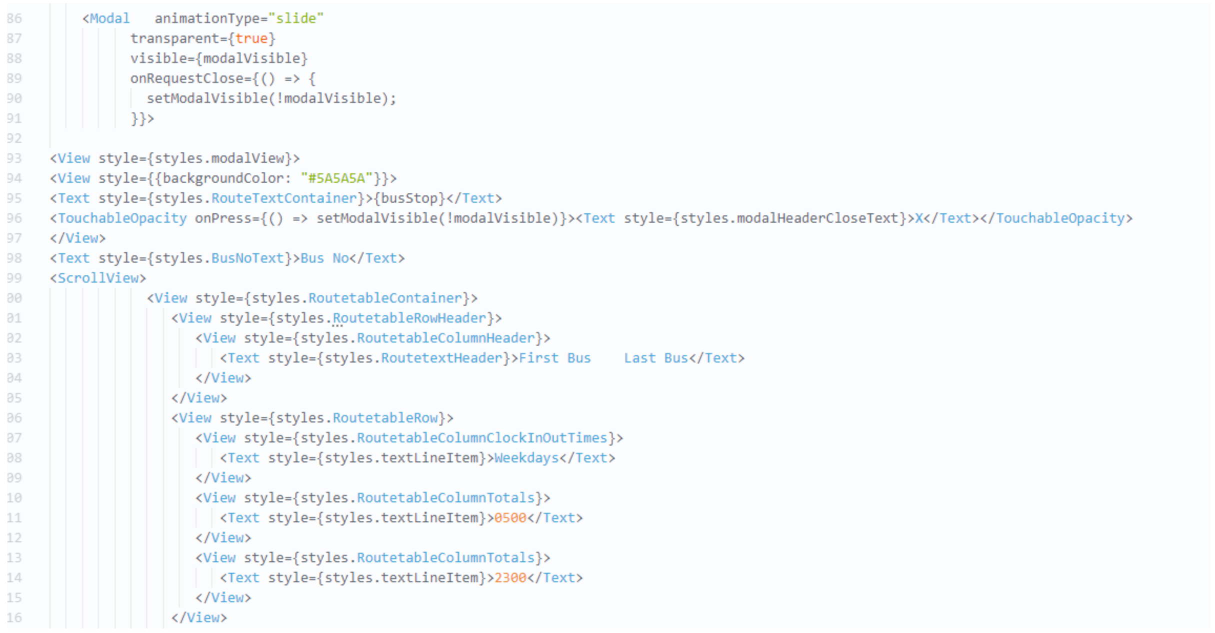
197	
Next Bus	2023-09-12T21:12:48+08:00
Next Second Bus	2023-09-12T21:17:26+08:00

851

Figure 27: bus info pop up(ios)

Figure 28: bus info pop up (android)

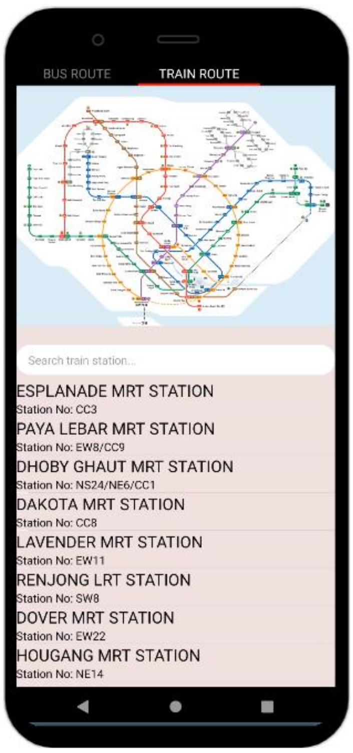
During the development, I was unable to find the bus information data online and the API data I got from LTA API is not the data I wanted. Thus, I decided to add in the bus service available timing manually and remove the bus route information. In the BusArrival.js, I have used modal component to create a pop up box and scrollview component to display the bus information data as shown below.



```
    <Modal   animationType="slide"
          transparent={true}
          visible={modalVisible}
          onRequestClose={() => {
            setModalVisible(!modalVisible);
          }}>

<View style={styles.modalView}>
<View style={{backgroundColor: "#5A5A5A"}}>
<Text style={styles.RouteTextContainer}>{busStop}</Text>
<TouchableOpacity onPress={() => setModalVisible(!modalVisible)}><Text style={styles.modalHeaderCloseText}>X</Text></TouchableOpacity>
</View>
<Text style={styles.BusNoText}>Bus No</Text>
<ScrollView>
            <View style={styles.RoutetableContainer}>
              <View style={styles.RoutetableRowHeader}>
                <View style={styles.RoutetableColumnHeader}>
                  <Text style={styles.RoutetextHeader}>First Bus    Last Bus</Text>
                </View>
              </View>
              <View style={styles.RoutetableRow}>
                <View style={styles.RoutetableColumnClockInOutTimes}>
                  <Text style={styles.textLineItem}>Weekdays</Text>
                </View>
                <View style={styles.RoutetableColumnTotals}>
                  <Text style={styles.textLineItem}>0500</Text>
                </View>
                <View style={styles.RoutetableColumnTotals}>
                  <Text style={styles.textLineItem}>2300</Text>
                </View>
              </View>
```

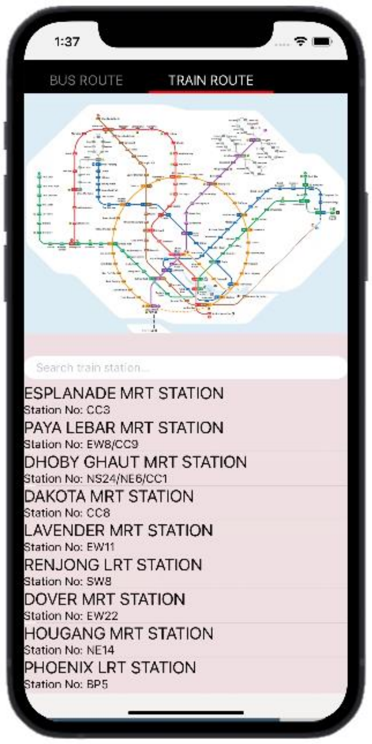
Figure 29: code for pop up box – bus info

Train route page:



ESPLANADE MRT STATION
Station No: CC3
PAYA LEBAR MRT STATION
Station No: EW8/CC9
DHOBY GHAUT MRT STATION
Station No: NS24/NE6/CC1
DAKOTA MRT STATION
Station No: CC8
LAVENDER MRT STATION
Station No: EW11
RENJONG LRT STATION
Station No: SW8
DOVER MRT STATION
Station No: EW22
HOUGANG MRT STATION
Station No: NE14

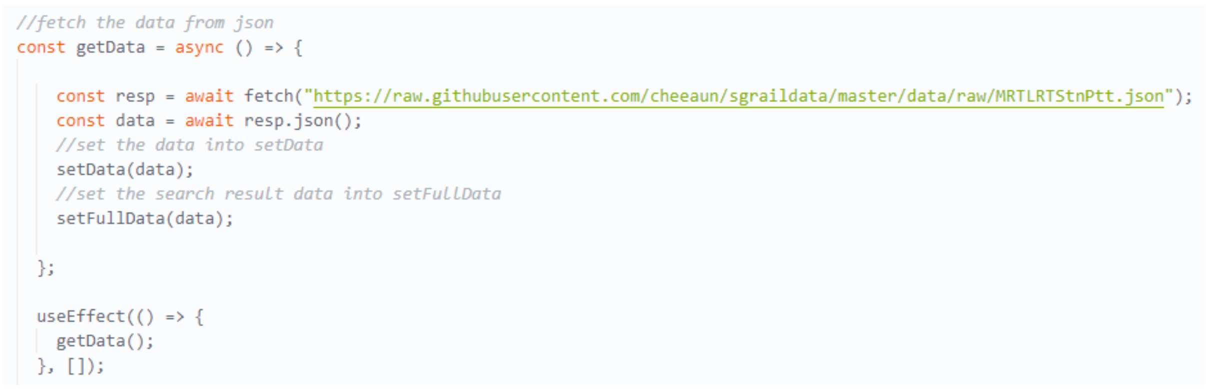
Figure 30: Train route page (android)



ESPLANADE MRT STATION
Station No: CC3
PAYA LEBAR MRT STATION
Station No: EW8/CC9
DHOBY GHAUT MRT STATION
Station No: NS24/NE6/CC1
DAKOTA MRT STATION
Station No: CC8
LAVENDER MRT STATION
Station No: EW11
RENJONG LRT STATION
Station No: SW8
DOVER MRT STATION
Station No: EW22
HOUGANG MRT STATION
Station No: NE14
PHOENIX LRT STATION
Station No: BP5

Figure 31: Train route page (ios)

Similar to what I have done for bus route page, I have fetched the train data from the github JSON as it is extremely difficult to find an API for trains since Singapore government does not provide one. The code for fetching the data is very similar to the bus route page, figure 15.

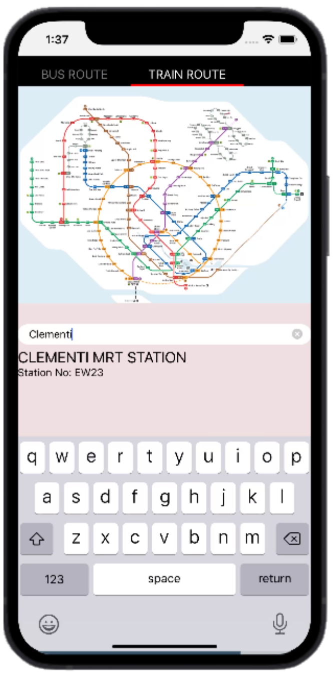


```
//fetch the data from json
const getData = async () => {

    const resp = await fetch("https://raw.githubusercontent.com/cheeaun/sgraildata/master/data/raw/MRTLRTStnPtt.json");
    const data = await resp.json();
    //set the data into setData
    setData(data);
    //set the search result data into setFullData
    setFullData(data);

  };

  useEffect(() => {
    getData();
  }, []);
```

Figure 32: fetch train stations data

I also have implemented the search feature for the user to search through the train stations list by train station names and station code. The code is very similar to the search feature in bus route page. The only different is that I have changed the values in the contains constant to STN_NAME and STN_NO to allow the user to search through based on those data.

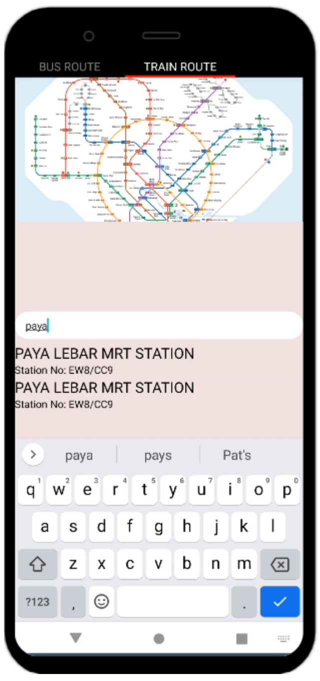
Outcome of the train search result:



Clementi

CLEMENTI MRT STATION
Station No: EW23

Figure 33: train station search result (ios)



paya

PAYA LEBAR MRT STATION
Station No: EW8/CC9
PAYA LEBAR MRT STATION
Station No: EW8/CC9

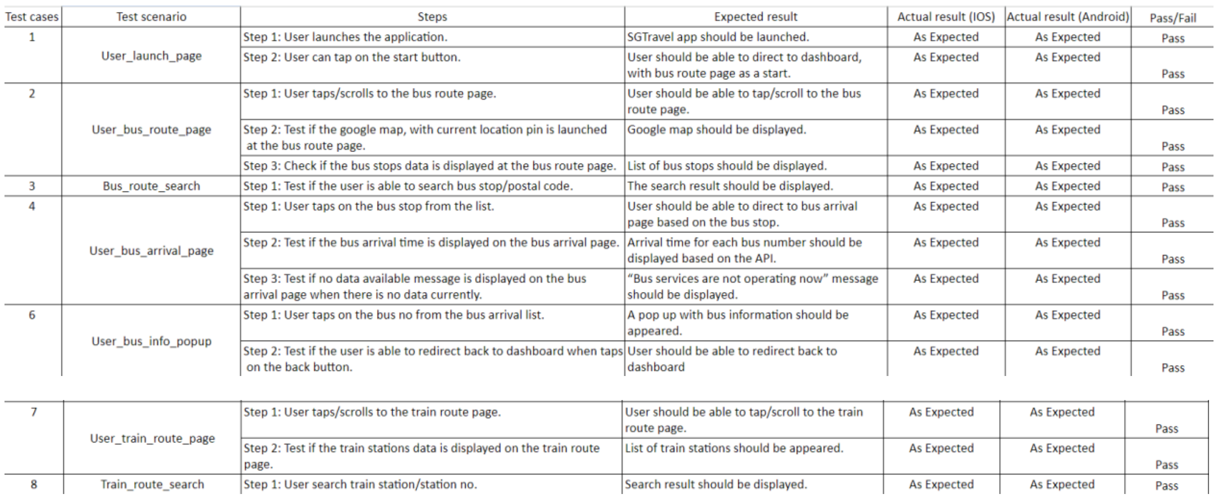
Figure 34: train station search result (android)

Train arrival page:

Due to unable to find the real time API data for the train arrival time and data for train route as mentioned in the train route page section, I have decided not to implement this page.

Unit Testing

Once the development is completed, I have asked a few people to test out the whole application on both ios and android by using expo snack. This is to ensure that the application is launched successfully without any errors, buttons are all workable and data is displayed correctly. Below shows the test script for the testings.



Test cases	Test scenario	Steps	Expected result	Actual result (IOS)	Actual result (Android)	Pass/Fail
1	User_launch_page	Step 1: User launches the application.	SGTravel app should be launched.	As Expected	As Expected	Pass
		Step 2: User can tap on the start button.	User should be able to direct to dashboard, with bus route page as a start.	As Expected	As Expected	Pass
2	User_bus_route_page	Step 1: User taps/scrolls to the bus route page.	User should be able to tap/scroll to the bus route page.	As Expected	As Expected	Pass
		Step 2: Test if the google map, with current location pin is launched at the bus route page.	Google map should be displayed.	As Expected	As Expected	Pass
		Step 3: Check if the bus stops data is displayed at the bus route page.	List of bus stops should be displayed.	As Expected	As Expected	Pass
3	Bus_route_search	Step 1: Test if the user is able to search bus stop/postal code.	The search result should be displayed.	As Expected	As Expected	Pass
4	User_bus_arrival_page	Step 1: User taps on the bus stop from the list.	User should be able to direct to bus arrival page based on the bus stop.	As Expected	As Expected	Pass
		Step 2: Test if the bus arrival time is displayed on the bus arrival page.	Arrival time for each bus number should be displayed based on the API.	As Expected	As Expected	Pass
		Step 3: Test if no data available message is displayed on the bus arrival page when there is no data currently.	"Bus services are not operating now" message should be displayed.	As Expected	As Expected	Pass
6	User_bus_info_popup	Step 1: User taps on the bus no from the bus arrival list.	A pop up with bus information should be appeared.	As Expected	As Expected	Pass
		Step 2: Test if the user is able to redirect back to dashboard when taps on the back button.	User should be able to redirect back to dashboard	As Expected	As Expected	Pass
7	User_train_route_page	Step 1: User taps/scrolls to the train route page.	User should be able to tap/scroll to the train route page.	As Expected	As Expected	Pass
		Step 2: Test if the train stations data is displayed on the train route page.	List of train stations should be appeared.	As Expected	As Expected	Pass
8	Train_route_search	Step 1: User search train station/station no.	Search result should be displayed.	As Expected	As Expected	Pass

Figure 35: test script

Evaluation

Problem faced:

During the development of this application, I have faced several issues as show in the followings:

1. JSON parser error when trying to fetch bus stops API data.
2. Bus arrival time data is too long, hard to display it based on the prototype layout.
3. Unable to find JSON data or API data for bus information.
4. Unable to find JSON data or API data for train arrival time and information.

As mentioned under the development section, I have made some changes to solve these issues as shown in the followings:

1. Instead of using API data, I have used JSON data from github which contains the exact same data from the API.

2. I have changed the layout of the arrival time list and used table list instead so that the full data is able to display.
3. Since there is no data available online, I decided to add in the bus services available time manually and decided not to implement bus route information since the API data is not what I expected to be.
4. Due to the time constraint and no data available online, I decided not to implement the page.

Improvement to make:

If I have more time and stronger react native coding knowledge, I will want to improve more on this application such as adding more features to it, for example, favourites, card services etc. I would also want to research more on what other ways to find train information and arrival time data in Singapore. As for the bus arrival time data, I would find a way to display specific string from the data instead of changing the layout and display all. For example, the API data returns “2023-04-23T21:34:00:00+08:00”, I would display the data as “21:34:00”.