Technical Manual

Interactive Map of IT Carlow

Name: Daniel Polak
Date: 30/04/2021
Supervisor: Hisain Elshaafi

Introduction

This document will contain all the code that was written during the project. In this document I will also explain how to run Expo and how to install all the dependencies required for the project to work.

Table of Contents

Installation

First you need to download and install Node.js. This can be done by going t this website https://nodejs.org/en/download/. Once that is done, install Expo CLI using npm by typing `npm install --global expo-cli` .

Now that Expo is installed navigate to a folder on your machine where you want your project to be and type that path into cmd e.g. cd C:\User\My Project.

In cmd, type expo init. This initialises expo in that directory.

Now, download the code from GitHub: https://github.com/DanielP1308/4thYearProject

After the code is downloaded, in the cmd, type npm install. This installs all dependencies in a file called package.json.

Next step is to run Expo CLI, in the cmd, while in the same directory as the project, type expo start.

The last step is to install Expo Go app on your mobile phone or emulator and run it by scanning QR code provided to you in previous step. You can also create a profile on Expo and tie it to your phone and therefore you don't need to use QR code to run it everytime.

1. Home.js

```
import 'react-native-gesture-handler';
import React from 'react';
import { StyleSheet, Text, View, Button, Image } from 'react-
native';
import { enableScreens } from 'react-native-screens';
import Header from '../Header.js';
import {scale, verticalScale} from 'react-native-size-matters';
enableScreens();
const size = {
  s: scale(600), //width
  s2: verticalScale(500), //Height
  top: scale(100),
};
const styles = StyleSheet.create({
  image: {
    height: size.s2 * 0.65,
    width: size.s * 0.6,
    resizeMode: 'contain',
  },
  container: {
    marginBottom: 16,
    flex: 1,
  },
  button: {
    color: '#f194ff',
    height: 20,
  },
  button: {
    backgroundColor: 'blue',
    color: 'white',
    fontSize: scale(35),
    width: '100%',
    height: scale(45),
    justifyContent: 'center',
    alignItems: 'center',
    position: 'relative'
```

```
  }
});

export default function HomeScreen({ navigation }) {
    return (
        <><Header title='Home' navigation={navigation} />
            <View style={{ flex: 1, justifyContent: 'center',
alignItems: 'center', height: 25 }}>
              <Image source={require('../Images/ITCarlowLogo.jpg')}
style={styles.image} />
            </View></>
    );
}
```

2. Login.js

```
import 'react-native-gesture-handler';
import React from 'react';
import { StyleSheet, Text, View, Button, TextInput, TouchableOpacity
} from 'react-native';
import { enableScreens } from 'react-native-screens';
import Header from '../Header.js';
import {scale, verticalScale} from 'react-native-size-matters';
import Map from './Map.js';

enableScreens();

export default function LoginScreen({ navigation }) {
  const map = () => {
    navigation.navigate(Map);
  };
  const size = {
    s: scale(600), //width
    s2: verticalScale(500), //Height
    top: scale(100),
  };
    return (
        <><Header title='Login/Sign Up' navigation={navigation} />
```

```
        <View style={{ flex: 1, alignItems: 'center', height: 25,
paddingTop: scale(120) }}>
            <Text>Email: </Text>
            <TextInput placeholder={"email@email.com"}
style={styles.search}></TextInput>
            <Text style={{paddingTop: scale(50)}}>Password: </Text>
            <TextInput placeholder={"***********"}
style={styles.search}></TextInput>
            <TouchableOpacity style={styles.button}>
              <Text>Login</Text>
            </TouchableOpacity>
        </View></>
    );
}
const styles = StyleSheet.create({
  search: {
      top: scale(32),
      height: scale(40),
      width: '90%',
      borderWidth: 1,
  },
  button: {
      top: scale(40),
      height: scale(40),
      width: '45%',
      fontSize: scale(25),
      justifyContent: 'center',
      alignItems: 'center',
      backgroundColor: 'gold',
  },
});
```

3. App.js

```
import 'react-native-gesture-handler';
import React, { Component } from 'react';
import { enableScreens } from 'react-native-screens';
import Navigation from './Drawer.js';
import { NavigationContainer } from '@react-navigation/native';
```

```
import firebase from 'firebase/app';

enableScreens();

var firebaseConfig = {
  apiKey: "AIzaSyCke_F5mdFgOFzJ3kGXWMNR9OrL_ziza-8",
  authDomain: "thyearproject-ddbca.firebaseapp.com",
  databaseURL: "https://thyearproject-ddbca-default-
rtdb.firebaseio.com",
  projectId: "thyearproject-ddbca",
  storageBucket: "thyearproject-ddbca.appspot.com",
  messagingSenderId: "876939868788",
  appId: "1:876939868788:web:e8530b3a7073ddd251ef21",
  measurementId: "G-MH6Z6RHL7M"
};
// Initialize Firebase
if (!firebase.apps.length) {
  firebase.initializeApp({});
}else {
  firebase.app(); // if already initialized, use that one
}
export default class App extends Component {
    render() {
        return (
          <NavigationContainer>
            <Navigation />
          </NavigationContainer>

        );
    };
}
```

4. Header.js

```
import 'react-native-gesture-handler';
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';
import { MaterialIcons } from '@expo/vector-icons';
import Search from './Screens/Search.js';
```

```
import { enableScreens } from 'react-native-screens';
import {scale, verticalScale} from 'react-native-size-matters';
enableScreens();

const size = {
    top: verticalScale(32),
    height: verticalScale(60),
    font: scale(20),
    icon: scale(28),
};

export default function Header({ title, navigation }) {

    const openMenu = () => {
        navigation.openDrawer();
    }

    const search = () => {
        navigation.navigate(Search);
    }

    return (
        <View style={styles.header}>
            <MaterialIcons name='menu' size={size.icon}
style={styles.icon} onPress={openMenu} />
            <MaterialIcons name='search' size={size.icon}
style={styles.icon1} onPress={search} />
            <MaterialIcons name='account-circle' size={size.icon}
style={styles.icon2} onPress={openMenu} />
            <View>
                <Text style={styles.headerText}>{title}</Text>
            </View>
        </View>
    );
}

const styles = StyleSheet.create({
    header: {
        top: size.top,
        width: '100%',
```

```
        height: size.height,
        flexDirection: 'row',
        alignItems: 'center',
        justifyContent: 'center',
        backgroundColor: 'blue',
        position: 'absolute',
    },
    headerText: {
        fontWeight: 'bold',
        fontSize: size.font,
        color: 'white',
        letterSpacing: 1,
    },
    icon: {
        position: 'absolute',
        left: 16,
        backgroundColor: 'white',
        borderRadius: 5
    },
    icon1: {
        position: 'absolute',
        right: 80,
        backgroundColor: 'white',
        borderRadius: 5
    },
    icon2: {
        position: 'absolute',
        right: 20,
        backgroundColor: 'white',
        borderRadius: 5
    },
});
```

5. Drawer.js

```
import 'react-native-gesture-handler';
import { createDrawerNavigator, DrawerItem, DrawerContentScrollView
} from '@react-navigation/drawer';
```

```
import { createMaterialBottomTabNavigator } from '@react-
navigation/material-bottom-tabs';
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import {View, StyleSheet} from 'react-native';
import { MaterialIcons } from '@expo/vector-icons';
import { enableScreens } from 'react-native-screens';
import {
    useTheme,
    Avatar,
    Title,
    Caption,
    Paragraph,
    Drawer,
    Text,
    TouchableRipple,
    Switch
} from 'react-native-paper';
import {scale, verticalScale} from 'react-native-size-matters';
import Settings from './Screens/Login.js';
import Home from './Screens/Home.js';
import Search from './Screens/Search.js';
import Map from './Screens/Map.js';

// drawer navigation options
const Drawers = createDrawerNavigator();

enableScreens();

const Bottom = createMaterialBottomTabNavigator();
const size = {
  s: scale(20),
}
function StackNav() {
  return (
      <Bottom.Navigator barStyle={{backgroundColor: 'blue'}}
activeColor="gold" inactiveColor="white">
        <Bottom.Screen name="Search" component={Search}
options={{tabBarIcon: ({ color }) => (<MaterialIcons name="search"
color={color} size={26} />),}}/>
```

```
        <Bottom.Screen name="Map" component={Map}
options={{tabBarIcon: ({ color }) => (<MaterialIcons name="map"
color={color} size={26} />),}}/>
      </Bottom.Navigator>
  );
}

function DrawerContent(props, {navigation}) {
  return (
    <View style={{flex: 1}}>
      <DrawerContentScrollView {...props}>
        <Drawer.Section style={styles.bottomDrawerSection}>
          <DrawerItem label="Home" icon={() => <MaterialIcons
name="home" size={size.s}/>} onPress={() =>
props.navigation.reset({routes: [{ name: 'Home' }]})}/>
        </Drawer.Section>
        <Drawer.Section style={styles.bottomDrawerSection}>
          <DrawerItem label="Map" icon={() => <MaterialIcons
name="map" size={size.s}/>} onPress={() =>
props.navigation.reset({routes: [{ name: 'Search' }]})}/>
        </Drawer.Section>
        <Drawer.Section style={styles.bottomDrawerSection}>
          <DrawerItem label="Profile" icon={() => <MaterialIcons
name="supervised-user-circle" size={size.s}/>} onPress={() =>
props.navigation.reset({routes: [{ name: 'Home' }]})}/>
        </Drawer.Section>
        <Drawer.Section style={styles.bottomDrawerSection}>
          <DrawerItem label="Settings" icon={() => <MaterialIcons
name="settings" size={size.s}/>} onPress={() =>
props.navigation.reset({routes: [{ name: 'Settings' }]})}/>
        </Drawer.Section>
        <Drawer.Section style={styles.bottomDrawerSection}>
          <DrawerItem label="Help" icon={() => <MaterialIcons
name="help" size={size.s}/>} onPress={() => alert('Link to help')}
/>
        </Drawer.Section>
        <Drawer.Section style={styles.bottomDrawerSection}>
          <DrawerItem label="Sign In" icon={() => <MaterialIcons
name="login" size={size.s}/>} onPress={() =>
props.navigation.reset({routes: [{ name: 'Settings' }]})}/>
```

```
        </Drawer.Section>
      </DrawerContentScrollView>
    </View>
  );
};

export default function MyDrawer() {
    return (
            <Drawers.Navigator initialRouteName="Home"
drawerStyle={{backgroundColor: '#FFD700', paddingTop: '20%'}}
drawerContent={props => <DrawerContent {...props} />}>
                <Drawers.Screen name="Home" component={Home} />
                <Drawers.Screen name="Search" component={StackNav}
/>
                <Drawers.Screen name="Profile" component={Home} />
                <Drawers.Screen name="Settings" component={Settings}
/>
                <Drawers.Screen name="Sign In" component={Settings}
/>
            </Drawers.Navigator>
    );
}

const styles = StyleSheet.create({
  bottomDrawerSection: {
    marginBottom: 15,
    borderTopColor: 'black',
  },
});
```

6. Search.js

```
import 'react-native-gesture-handler';
import React, { useState, Component } from 'react';
import { StyleSheet, Text, View, Button, TextInput,
TouchableOpacity, ScrollView, FlatList, List} from 'react-native';
```

```
import {ListItem, SearchBar} from 'react-native-elements';
import { enableScreens } from 'react-native-screens';
import { useNavigation } from '@react-navigation/native';
import Map from './Map.js'
import Header from '../Header.js';
import firebase from 'firebase/app';
import database from 'firebase/database';
import {scale, verticalScale} from 'react-native-size-matters';

var firebaseConfig = {
  apiKey: "AIzaSyCke_F5mdFgOFzJ3kGXWMNR9OrL_ziza-8",
  authDomain: "thyearproject-ddbca.firebaseapp.com",
  databaseURL: "https://thyearproject-ddbca-default-
rtdb.firebaseio.com",
  projectId: "thyearproject-ddbca",
  storageBucket: "thyearproject-ddbca.appspot.com",
  messagingSenderId: "876939868788",
  appId: "1:876939868788:web:e8530b3a7073ddd251ef21",
  measurementId: "G-MH6Z6RHL7M"
};
// Initialize Firebase
if (!firebase.apps.length) {
  firebase.initializeApp(firebaseConfig);
}else {
  firebase.app(); // if already initialized, use that one
}

enableScreens();

export default class Search extends Component {
  constructor(props){
    super(props);
    this.state={
      search : '',
      data: '',
      newData: '',
      backupData: '',
      startPoint: '',
      endPoint: ''
    }
```

```
  }

  componentDidMount() {
    firebase.database().ref('/Rooms/').once('value').then(snapshot
=> { this.setState({data: snapshot.val(), backupData:
snapshot.val()})});
  }
  Send_Search = () => {
    this.props.navigation.navigate('Map', {startPoint:
this.state.startPoint, endPoint: this.state.endPoint})
  }
  Search(search) {
    var keys = Object.keys(this.state.data);
    var tempData = this.state.data;
    search = search.toLowerCase();
    var searchData = [];
    var x = 0;

    keys.forEach((i, index) => {
      if(tempData[i].Name.toLowerCase().includes(search)) {
        searchData[x] = tempData[i];
        x += 1;
      }
    });
    if(search === '') {
      this.setState({backupData: this.state.data});
    }
    else {
      this.setState({backupData: searchData});
    }
    /*const filterData = tempData.filter((item) => {
      return item.Name.toLowerCase().includes(search);
    });*/
    //this.setState({backupData:
this.state.data.filter(i=>i.Name.includes(search))})
    console.log(search);
    //this.setState({backupData: filterData});
  }

  isSelected(item) {
```

```
    if(this.state.startPoint == '') {
      this.setState({startPoint: item});
    }
    else if(this.state.endPoint == '' && this.state.startPoint !=
'') {
      this.setState({endPoint: item});
    }
  }
  renderItemComponent = (item) => (
    <ListItem bottomDivider  onPress={() =>
this.isSelected(item.item)}>
      <ListItem.Content>
        <ListItem.Title>{item.item.Name}</ListItem.Title>
        <ListItem.Subtitle>Building:
{item.item.Building}</ListItem.Subtitle>
        <ListItem.Subtitle>Floor:
{item.item.Floor}</ListItem.Subtitle>
      </ListItem.Content>
    </ListItem>
  );

  renderHeader = () => {
    return (
      <SearchBar
        placeholder="Type Here..."
        lightTheme
        round
        onChangeText={(text) => this.Search(text)}
        autoCorrect={false}
      />
    );
  }
  keyExtractor = (item, index) => index.toString();
  render() {
    const dataArray = Object.values(this.state.backupData);
    return (
        <><Header title='Search' navigation={this.props.navigation}
/>
            <View style={{ flex: 1, height: 25, paddingTop: '27%'
}}>
```

```
                <FlatList data={dataArray} renderItem={(dataArray)
=> this.renderItemComponent(dataArray)}
keyExtractor={this.keyExtractor}
ListHeaderComponent={this.renderHeader} />
                <TouchableOpacity style={styles.navigate}
onPress={() => this.Send_Search()}>
                  <Text>Navigate</Text>
                </TouchableOpacity>
            </View></>
    );
  }
}

const styles = StyleSheet.create({
    search: {
        top: scale(32),
        height: scale(40),
        width: '90%',
        borderWidth: 1,
    },
    button: {
        top: scale(40),
        height: scale(40),
        width: '45%',
        fontSize: scale(25),
        justifyContent: 'center',
        alignItems: 'center',
        backgroundColor: 'gold',
    },
    select: {
      top: scale(40),
      height: scale(40),
      width: '100%',
      justifyContent: 'center',
      alignItems: 'center',
      backgroundColor: 'gold'
    },
    navigate: {
      height: scale(30),
      justifyContent: 'center',
```

```
      alignItems: 'center',
      fontSize: scale(20),
      backgroundColor: 'gold',
      color: 'white',
    }
});
```

7. Map.js

```
import 'react-native-gesture-handler';
import React, { Component } from 'react';
import { View, StyleSheet, Image, PixelRatio, Dimensions, Button,
ScrollView, TouchableOpacity } from 'react-native';
import { enableScreens } from 'react-native-screens';
import Svg, {Path, Rect} from 'react-native-svg';
import ImageZoom from 'react-native-image-pan-zoom';
import {scale, verticalScale} from 'react-native-size-matters';
import Header from '../Header.js';
import firebase from 'firebase/app';
enableScreens();

const size = {
  s: scale(600), //width
  s2: verticalScale(500), //Height
  top: scale(100),
};

const styles = StyleSheet.create({
  image: {
    height: size.s2 * 0.65,
    width: size.s * 0.6,
    resizeMode: 'contain',
  },
  container: {
    marginBottom: 16,
    flex: 1,
```

```
  },
  button: {
    color: '#f194ff',
    height: 20,
  },
  button: {
    backgroundColor: 'blue',
    fontSize: 25,
    width: '100%',
    height: 35,
    justifyContent: 'center',
    alignItems: 'center',
    position: 'absolute'
  }
});
export default class Map extends Component {
  constructor(){
    super();

    this.state={
      imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/CampusMap.jpg?alt=media&token=091f4b52-18d5-
4d60-acd5-3b429e26b338",
      path : "",
      toggle: false,
      graphs: '',
      index: 0,
      once: true,
    }
  }
  componentDidMount() {
    this.setState({startPoint: this.props.route?.params?.startPoint,
endPoint: this.props.route?.params?.endPoint});

firebase.database().ref('/Buildings/').once('value').then(snapshot
=> this.setState({data: snapshot.val()}));
    firebase.database().ref('/Rooms/').once('value').then(snapshot
=> this.setState({roomsData: snapshot.val()}));
```

```
    firebase.database().ref('/Paths/').once('value').then(snapshot
=> this.setState({pathsData: snapshot.val()}));
  }

  findIndexOfNodesInArray(shortest, alphabet, x_y_coord) {
    var nodes_used = [];
    var ins = 0;
    for (var i = 0; i < alphabet.length; i++) {
      for(var x = 0; x < shortest.path.length; x++) {
        if (alphabet[i] == shortest.path[x]) {
          if(shortest.path.length == 1) {
            nodes_used[ins] = i;
          }
          else {
            nodes_used[ins] = x;
          }
          ins += 1;
        }
      }
    }

    var nodes_to_be_used = [];
    for (var i = 0; i < x_y_coord.length; i++) {
      var index = nodes_used[i];
      for(var x = 0; x <= index; x++) {
        if(x == index) {
          nodes_to_be_used[i] = x_y_coord[index];
        }
      }
    }
    return nodes_to_be_used;
  }

  get_array(key, shortest, points) {
    var array = [[]];
    key.forEach((i, index) => {
      for(var x = 0; x < shortest.path.length; x++) {
        if(i == shortest.path[x]) {
          array[x] = Object.values(points[i]);
        }
```

```
      }
    });
    return array;
  }

  draw_path1(array) {
    var path = "M" + scale(array[0][0]) + "," + scale(array[0][1]);
    for(var i = 0; i < array.length; i++) {
      path += " L" + scale(array[i][0]) + "," + scale(array[i][1]);
    }
    return path;
  }

  getDataNeeded() {
    var start_point = this.state.startPoint;
    var end_point = this.state.endPoint;
    var pathData = this.state.pathsData;
    var tempData = this.state.data;
    var k = Object.keys(this.state.data);
    var pathKeys = Object.keys(this.state.pathsData);
    var getBuildingData = [];

    if(start_point.Building === end_point.Building &&
start_point.Floor != end_point.Floor) { // Searches data from
database if start location is the same as destination and floors do
not match and gets all the relevant data associated with it
      var m = 0;
      k.forEach((d, index) => {
        // Get graph and points of the building at start and end
        if(tempData[d].Name === start_point.Building) {
          getBuildingData[m] = tempData[d];
          m += 1;
        }
      });
    }
    else if(start_point.Building === end_point.Building &&
start_point.Floor === end_point.Floor) { // Searches data from
database if start location is the same as destination and floors
match and gets data for that particular floor of the building
      k.forEach((d, index) => {
```

```
        // Get graph and points of the building at start and end
        if(tempData[d].Name === start_point.Building &&
tempData[d].Floor === end_point.Floor) {
          getBuildingData = tempData[d];
        }
      });
    }

    if(start_point.Building != end_point.Building) { // This
searches database when start location and destination do not match
      pathKeys.forEach((s, index) => {
        if(start_point.Building === pathData[s].start &&
end_point.Building === pathData[s].end) {
          getBuildingData = pathData[s];
        }
        else if(start_point.Building === pathData[s].end &&
end_point.Building === pathData[s].start){
          getBuildingData = pathData[s];
          if(getBuildingData.between !== undefined) {
            getBuildingData.between.reverse();
          }
        }
      });
    }

    var arrayOfPathToTake = [];

    if(start_point.Building == getBuildingData.end &&
end_point.Building == getBuildingData.start) {
      if(getBuildingData.between === undefined) {
        arrayOfPathToTake[0] = getBuildingData.end;
        arrayOfPathToTake[1] = getBuildingData.start;
      }
      else {
        arrayOfPathToTake[0] = getBuildingData.end;
        var pp = 0;
        for(var i = 1; i < getBuildingData.between.length+1; i++) {
          arrayOfPathToTake[i] = getBuildingData.between[pp];
          pp += 1;
        }
```

```
        arrayOfPathToTake[getBuildingData.between.length + 1] =
getBuildingData.start;
      }
    }
    else if(start_point.Building == getBuildingData.start &&
end_point.Building == getBuildingData.end) {
      if(getBuildingData.between === undefined) {
        arrayOfPathToTake[0] = getBuildingData.start;
        arrayOfPathToTake[1] = getBuildingData.end;
      }
      else {
        arrayOfPathToTake[0] = getBuildingData.start;
        var pp = 0;
        for(var i = 1; i < getBuildingData.between.length+1; i++) {
          arrayOfPathToTake[i] = getBuildingData.between[pp];
          pp += 1;
        }
        arrayOfPathToTake[getBuildingData.between.length + 1] =
getBuildingData.end;
      }
    }
    var containBuilding = [];
    if(arrayOfPathToTake && arrayOfPathToTake.length) {
      var l = 1;
      var ll = 1;
      var end = arrayOfPathToTake.length -1;
      k.forEach((n, x) => {
        if(arrayOfPathToTake[0] == tempData[n].Name &&
start_point.Floor > 1) {
          containBuilding[0] = tempData[n];
          if(tempData[n].Floor == start_point.Floor - 1) {
            containBuilding[l] = tempData[n];
            l += 1;
          }
          if(tempData[n].Floor == start_point.Floor - 2) {
            containBuilding[l] = tempData[n];
            l += 1;
          }
        }
```

```
        if (arrayOfPathToTake[0] == tempData[n].Name &&
start_point.Floor == 1) {
          if(tempData[n].Floor == 1) {
            containBuilding[0] = tempData[n];
          }
        }

        for(var c = 1; c < arrayOfPathToTake.length-1; c++) {
          if(arrayOfPathToTake[c] == tempData[n].Name &&
tempData[n].Floor == 1) {
            containBuilding[l] = tempData[n];
            l += 1;
            ll += 1;
          }
          else if(arrayOfPathToTake[c] == tempData[n].Name &&
tempData[n].Floor == 0) {
            containBuilding[l] = tempData[n];
            l += 1;
          }
        }

        if(arrayOfPathToTake[end] == tempData[n].Name &&
end_point.Floor > 1) {
          if(tempData[n].Floor == 1) {
            containBuilding[l] = tempData[n];
            l += 1;
          }
          if(tempData[n].Floor == 2) {
            containBuilding[l] = tempData[n];
            l += 1;
          }
          if(tempData[n].Floor == 3) {
            containBuilding[l] = tempData[n];
            l += 1;
          }

        }
        else if (arrayOfPathToTake[end] == tempData[n].Name &&
end_point.Floor == 1) {
          if(tempData[n].Floor == 1) {
```

```
            containBuilding[l] = tempData[n];
            l += 1;
          }
        }
      });
      var temp_array = [];
      var ind = 0;
      for(var v = 0; v < arrayOfPathToTake.length; v++) {
        for(var d = 0; d < containBuilding.length; d++) {
          if(arrayOfPathToTake[v] == containBuilding[d].Name){
            temp_array[ind] = containBuilding[d];
            ind += 1;
          }
        }
      }
      containBuilding = temp_array;
    }
    if(!containBuilding.length) {
      containBuilding = getBuildingData;
    }
    return containBuilding;
  }
  // Get everything that is needed in between starting point and
ending point and then go through it!!!!!

  calcPath(startPoint, endPoint) {
    startPoint = this.state.startPoint;
    endPoint = this.state.endPoint;
    var tempData = this.state.data;
    var tempRoomsData = this.state.roomsData;
    var k = Object.keys(this.state.data);
    var r = Object.keys(this.state.roomsData);
    var points = "";
    let graph = "";
    var count = "";
    var indexOfC = [];
    var image = "";
    var getNextBuildingData = "";
    var getNextStartingPoint = "";
    var getName = "";
```

```
    var getStartingBuilding = '';
    var getDestinationBuilding = '';
    var getBuildingData = this.getDataNeeded();

    var start_node = "";
    var end_node = "";
    var c  = 0;
    //Find all instances of Campus
    for(var i = 0; i < getBuildingData.length; i++) {
      if(getBuildingData[i].Name == "Campus") {
        count += 1;
        indexOfC[c] = i;
        c += 1;
      }
    }
    // If instances of Campus are > 2 delete the excess
    if(count > 2) {
      var l = indexOfC[indexOfC.length-1];
      var newArrayOfData = [];
      var indes = 0;
      for(var i = 0; i < getBuildingData.length; i++) {
        if(i !== indexOfC[0]) {
          newArrayOfData[indes] = getBuildingData[i];
          indes += 1;
        }
        else if(i !== l) {
          newArrayOfData[indes] = getBuildingData[i];
          indes += 1;
        }
      }
      newArrayOfData.splice(indexOfC[0], 1);
      newArrayOfData.splice(indexOfC[indexOfC.length-2], 1);
      getBuildingData = newArrayOfData;
    }
    // Make it so that image changes on click of a button
    if(this.state.index < getBuildingData.length) {
      this.setState({imageURL:
getBuildingData[this.state.index].Image});
      this.setState({index: this.state.index + 1});
      points = getBuildingData[this.state.index].Points;
```

```
      graph = getBuildingData[this.state.index].Graph;
    }
    // Get starting point and ending point for Djikstras Algo if
start and destination are in the same building and floor
    if(startPoint.Building == endPoint.Building && startPoint.Floor
== endPoint.Floor) {
      this.setState({imageURL: getBuildingData.Image});
      points = getBuildingData.Points;
      graph = getBuildingData.Graph;
      start_node = startPoint.Point;
      end_node = endPoint.Point;
    }
    else { // Get starting point and ending point if start and
destination have different floors or buildings
      if(this.state.index === 0) { // This finds ending point at the
start of the navigation as starting point is defined by the user
        start_node = startPoint.Point;
        r.forEach((d, i) => {
          if(tempRoomsData[d].Building ==
getBuildingData[this.state.index].Name && tempRoomsData[d].Floor ==
getBuildingData[this.state.index].Floor) {
            if(tempRoomsData[d].Name ==
getBuildingData[this.state.index+1].Name){
              end_node = tempRoomsData[d].Point;
            }
            else if(getBuildingData[this.state.index+1].Floor >
getBuildingData[this.state.index].Floor) {
              if (tempRoomsData[d].Name === "Stairs") {
                end_node = tempRoomsData[d].Point;
              }
            }
          }
        });
      }
      else if(this.state.index === getBuildingData.length-1) { //
This finds start point in the last image as ending point is defined
by the user
        r.forEach((d, i) => {
```

```
          if(tempRoomsData[d].Building ==
getBuildingData[this.state.index].Name && tempRoomsData[d].Floor ==
getBuildingData[this.state.index].Floor) {
            if(tempRoomsData[d].Name ===
getBuildingData[this.state.index-1].Name){
              start_node = tempRoomsData[d].Point;
            }
            else if(getBuildingData[this.state.index+1].Floor <
getBuildingData[this.state.index].Floor) {
              if (tempRoomsData[d].Name === "Stairs") {
                end_node = tempRoomsData[d].Point;
              }
            }
          }
        });
        end_node = endPoint.Point;
      }
      else if(this.state.index < getBuildingData.length-1 &&
this.state.index > 0) { // This finds points based on your previous
building and your next building to find points that will then be
used to show path to get to your destination
        r.forEach((d, i) => {
          if(tempRoomsData[d].Building ==
getBuildingData[this.state.index].Name && tempRoomsData[d].Floor ==
getBuildingData[this.state.index].Floor) {
            if(tempRoomsData[d].Name ==
getBuildingData[this.state.index-1].Name){
              start_node = tempRoomsData[d].Point;
            }
            if(tempRoomsData[d].Name ==
getBuildingData[this.state.index+1].Name){
              end_node = tempRoomsData[d].Point;
            }
          }
        });
      }
    }

    console.log(indexOfC);
```

```
    // Part of Djikstras Algo to find the shortest path on the image
    let shortestDistanceNode = (distances, visited) => {
      // create a default value for shortest
      let shortest = null;

        // for each node in the distances object
      for (let node in distances) {
          // if no node has been assigned to shortest yet
          // or if the current node's distance is smaller than the
current shortest
        let currentIsShortest =
          shortest === null || distances[node] <
distances[shortest];

          // and if the current node is in the unvisited set
        if (currentIsShortest && !visited.includes(node)) {
                // update shortest to be the current node
          shortest = node;
        }
      }
      return shortest;
    };
    // Main part of Djistras Algo to find shortest path to a
destination
    let findShortestPath = (graph, startNode, endNode) => {

      // track distances from the start node using a hash object
        let distances = {};
      distances[endNode] = "Infinity";
      distances = Object.assign(distances, graph[startNode]);
     // track paths using a hash object
      let parents = { endNode: null };
      for (let child in graph[startNode]) {
       parents[child] = startNode;
      }

      // collect visited nodes
        let visited = [];
     // find the nearest node
        let node = shortestDistanceNode(distances, visited);
```

```
      // for that node:
      while (node) {
      // find its distance from the start node & its child nodes
       let distance = distances[node];
       let children = graph[node];

      // for each of those child nodes:
           for (let child in children) {

       // make sure each child node is not the start node
            if (String(child) === String(startNode)) {
              continue;
            } else {
              // save the distance from the start node to the child
node
              let newdistance = distance + children[child];
     // if there's no recorded distance from the start node to the
child node in the distances object
     // or if the recorded distance is shorter than the previously
stored distance from the start node to the child node
              if (!distances[child] || distances[child] >
newdistance) {
     // save the distance to the object
          distances[child] = newdistance;
     // record the path
          parents[child] = node;
         }
              }
           }
          // move the current node to the visited set
          visited.push(node);
     // move to the nearest neighbor node
          node = shortestDistanceNode(distances, visited);
         }

       // using the stored paths from start node to end node
       // record the shortest path
       let shortestPath = [endNode];
       let parent = parents[endNode];
```

```
      while (parent) {
       shortestPath.push(parent);
       parent = parents[parent];
      }
      shortestPath.reverse();

      //this is the shortest path
      let results = {
       distance: distances[endNode],
       path: shortestPath,
      };
      // return the shortest path & the end node's distance from the
start node
        return results;
    };
    // Gets keys from the object
    let key = Object.keys(points);
    let shortest = findShortestPath(graph, start_node, end_node);
    console.log(shortest);
    var array = [[]];
    // gets array tobe used for drawing the path
    array = this.get_array(key, shortest, points);
    // draws the path on the image
    var path = this.draw_path1(array);
    //path = "M" + scale(225) + "," + scale(270) + " L" + scale(430)
+ "," + scale(272);
    this.setState({path: path});
  }
Load_New_Image(id, floor) {
  if (id == 1) {
        if (floor == 1) {
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/gaaGround.jpg?alt=media&token=37d99067-733a-
4a09-bee3-c10901309e4f", path: ""})
    }
    else {
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
```

```
ddbca.appspot.com/o/gaaFirst.jpg?alt=media&token=1214be3c-a08a-4af9-
92a3-cb0df3602d2a", path: ""})
    }
  }
  else if (id == 2) {
    if (floor == 1) {
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/barrows.jpg?alt=media&token=c8b2f265-0144-4294-
91b6-26b5d328bf2e", path: ""})
    }
    else {
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/barrow2.jpg?alt=media&token=93cbd359-9d63-4c38-
88b9-fcc807b403d2", path: ""})
    }
  }
  else if (id == 3) {
    if (floor == 1) {
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/killeshinGround.jpg?alt=media&token=378c586a-
30ed-4d09-acef-9dd50b3f2d5a", path: ""})
    }
    else {
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/killeshinSecond.jpg?alt=media&token=3dd241be-
0b75-4769-9d8e-d995ae2e2fc4", path: ""})
    }
  }
  else if (id == 4) {
    if (floor == 1) {
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/haughtonGround.jpg?alt=media&token=9ab6438e-
f7c2-470e-bc84-a428791ca772", path: ""})
    }
    else {
```

```
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/haughtonFirst.JPG?alt=media&token=68ff9fce-ef6f-
4e2d-8230-6708ca1ca7ae", path: ""})
    }
  }
  else if (id == 5) {
    if (floor == 1) {
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/burrinGround.jpg?alt=media&token=7a361ff6-4ef9-
4258-b1ac-4de1333e6a6b", path: ""})
    }
    else {
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/burrinFirst.jpg?alt=media&token=7f5a1af2-d776-
48c8-a6ab-ecdd8be454ec", path: ""})
    }
  }
  else if (id == 6) {
    if (floor == 1) {
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/lrcGround.jpg?alt=media&token=f5f67ff7-e994-
4060-8482-56327ea68ae8", path: ""})
    }
    else if (floor == 2) {
      this.setState({imageURL:
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/lrcFirst.jpg?alt=media&token=edc6ea50-56c5-4277-
9b53-3d86a5c96b07", path: ""})
    }
    else {
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/lrcSecond.jpg?alt=media&token=73222843-18b9-
4940-ac97-7239d292cf94", path: ""})
    }
  }
```

```
  else if (id == 7) {
    if (floor == 1) {
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/slaneyGround.jpg?alt=media&token=32a3dcd6-74ad-
4ee9-9c68-2a36951c6b7e", path: ""})
    }
    else {
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/slaneyFirst.jpg?alt=media&token=22a49f7a-28d7-
460f-8c6a-bb3ab68420ac", path: ""})
    }
  }
  else if (id == 8) {
    if (floor == 1) {
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/noreGround.jpg?alt=media&token=48dab923-0b1e-
4a80-a82a-78a3826ba4c7", path: ""})
    }
    else {
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/noreFirst.jpg?alt=media&token=3278c579-817f-
4d65-bdfc-5fd91be3bb78", path: ""})
    }
  }
  else if (id == 9) {
    if (floor == 1) {
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/darganGround.jpg?alt=media&token=02562358-92c5-
4974-9f99-a5dd3b97a093", path: ""})
    }
    else if (floor == 2) {
      this.setState({imageURL:
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/darganFirst.jpg?alt=media&token=dfff8994-5add-
4b36-914d-956f3e63ad78", path: ""})
```

```
    }
    else {
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/darganSecond.jpg?alt=media&token=aff4856a-e08e-
407a-ab75-068029ee55d3", path: ""})
    }
  }
  else if (id == 10) {
    if (floor == 1) {
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/engineeringGround.jpg?alt=media&token=2cd0cd13-
611a-40be-8a9e-e0e6809a2ebe", path: ""})
    }
    else {
      this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/engineeringFirst.jpg?alt=media&token=2187c3c3-
e3b1-410f-ad45-70736c799859", path: ""})
    }
  }
  else if (id == 0) {
    this.setState({imageURL :
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/CampusMap.jpg?alt=media&token=091f4b52-18d5-
4d60-acd5-3b429e26b338", path: ""})
  }
}
mainMapSVG(image) {
  let gaa_building = "M325 150 L325 170 L360 170 L360 150 Z" ;
  let barrow_centre = "M430 110 L430 145 H430 480 V145 110 Z";
  let kileshin_centre = "M445 80 L445 100 H445 480 V100 80 Z";
  const sizeSVG = {
    rectX1: verticalScale(125),
    rectX2: verticalScale(226),
    rectX3: verticalScale(230),
    rectX4: verticalScale(290),
    rectX5: verticalScale(288),
    rectX6: verticalScale(290),
```

```
    rectX7: verticalScale(323),
    rectX8: verticalScale(338),
    rectX9: verticalScale(365),
    rectX10: verticalScale(435),
    //Y
    rectY1: verticalScale(150),
    rectY2: verticalScale(110),
    rectY3: verticalScale(83),
    rectY4: verticalScale(55),
    rectY5: verticalScale(83),
    rectY6: verticalScale(133),
    rectY7: verticalScale(88),
    rectY8: verticalScale(90),
    rectY9: verticalScale(60),
    rectY10: verticalScale(235),
    // Height
    h1: verticalScale(15),
    h2: verticalScale(35),
    h3: verticalScale(20),
    h4: verticalScale(20),
    h5: verticalScale(35),
    h6: verticalScale(47),
    h7: verticalScale(60),
    h8: verticalScale(60),
    h9: verticalScale(20),
    h10: verticalScale(30),
    // Width
    w1: scale(30),
    w2: scale(50),
    w3: scale(40),
    w4: scale(50),
    w5: scale(33),
    w6: scale(33),
    w7: scale(14),
    w8: scale(65),
    w9: scale(35),
    w10: scale(25),
  };
  let studentService_stairs = "M600 205 L560 205 V205 200 H200 200
";
```

```
  // M315,195 M315,210 M315,165 M215,210
  let paths = "M" + scale(215) + "," + scale(210) + "L" + scale(390)
+ "," + scale(205)
  if (image ==
"https://firebasestorage.googleapis.com/v0/b/thyearproject-
ddbca.appspot.com/o/CampusMap.jpg?alt=media&token=091f4b52-18d5-
4d60-acd5-3b429e26b338") {
    return(
      <Svg>
        <Rect x={sizeSVG.rectX1} y={sizeSVG.rectY1}
height={sizeSVG.h1} width={sizeSVG.w1} stroke="#ffffff00"
strokeWidth="1" onPress={() => this.Load_New_Image(1, 1)} id="GAA"/>
        <Rect x={sizeSVG.rectX2} y={sizeSVG.rectY2}
height={sizeSVG.h2} width={sizeSVG.w2} stroke="#ffffff00"
strokeWidth="1" onPress={() => this.Load_New_Image(2, 1)}
id="BarrowCentre"/>
        <Rect x={sizeSVG.rectX3} y={sizeSVG.rectY3}
height={sizeSVG.h3} width={sizeSVG.w3} stroke="#ffffff00"
strokeWidth="1" onPress={() => this.Load_New_Image(3, 1)}
id="Kileshin"/>
        <Rect x={sizeSVG.rectX4} y={sizeSVG.rectY4}
height={sizeSVG.h4} width={sizeSVG.w4} stroke="#ffffff00"
strokeWidth="1" onPress={() => this.Load_New_Image(4, 1)}
id="Haughton"/>
        <Rect x={sizeSVG.rectX5} y={sizeSVG.rectY5}
height={sizeSVG.h5} width={sizeSVG.w5} stroke="#ffffff00"
strokeWidth="1" onPress={() => this.Load_New_Image(5, 1)}
id="Burrin"/>
        <Rect x={sizeSVG.rectX6} y={sizeSVG.rectY6}
height={sizeSVG.h6} width={sizeSVG.w6} stroke="#ffffff00"
strokeWidth="1" onPress={() => this.Load_New_Image(6, 1)} id="LRC"/>
        <Rect x={sizeSVG.rectX7} y={sizeSVG.rectY7}
height={sizeSVG.h7} width={sizeSVG.w7} stroke="#ffffff00"
strokeWidth="1" onPress={() => this.Load_New_Image(7, 1)}
id="Slaney"/>
        <Rect x={sizeSVG.rectX8} y={sizeSVG.rectY8}
height={sizeSVG.h8} width={sizeSVG.w8} stroke="#ffffff00"
strokeWidth="1" onPress={() => this.Load_New_Image(8, 1)}
id="Nore"/>
```

```
        <Rect x={sizeSVG.rectX9} y={sizeSVG.rectY9}
height={sizeSVG.h9} width={sizeSVG.w9} stroke="#ffffff00"
strokeWidth="1" onPress={() => this.Load_New_Image(9, 1)}
id="Dargan"/>
        <Rect x={sizeSVG.rectX10} y={sizeSVG.rectY10}
height={sizeSVG.h10} width={sizeSVG.w10} stroke="#ffffff00"
strokeWidth="1" onPress={() => this.Load_New_Image(10, 1)}
id="Engineering"/>
        <Path d={this.state.path} fill="none" stroke="red" stroke-
width="10" />
      </Svg>
    );
  }
  else {
    return (
      <Svg>
        <Path d={this.state.path} fill="none" stroke="red" stroke-
width="10" />
      </Svg>
    );
  }
}

Change_State_Toggle_True = () => {
  this.setState({toggle: true})
}

Change_State_Toggle_False = () => {
  this.setState({toggle: false})
}
buttons(image) {
  if(this.state.startPoint != null && this.state.endPoint != null) {
    return (
      <ScrollView style={styles.container}>
        <Button title="Campus Map" onPress={() =>
this.Load_New_Image(0)}/>
        <Button title="Navigate" onPress={(param) =>
this.calcPath(1, 2)}/>
      </ScrollView>
    );
```

```
  }
  else {
    return (
      <ScrollView style={styles.container}>
        <Button title="Campus Map" onPress={() =>
this.Load_New_Image(0)}/>
      </ScrollView>
    );
  }
  /*else if (image == barrow_map_second && this.state.toggle) {
      return (
        <ScrollView style={styles.container}>
          <Button title="Campus Map" onPress={() =>
this.Load_New_Image(0)}/>
          <Button title={buttonsBarrow2[0]} onPress={(param) =>
this.calcPath(1, 2)}/>
          <Button title={buttonsBarrow2[1]} onPress={(param) =>
this.pathSet(this.state.imageURL, 1)}/>
          <Button title={buttonsBarrow2[2]} onPress={(param) =>
this.pathSet(this.state.imageURL, 2)}/>
          <Button title={buttonsBarrow2[3]} onPress={(param) =>
this.pathSet(this.state.imageURL, 3)}/>
          <Button title={buttonsBarrow2[4]} onPress={(param) =>
this.pathSet(this.state.imageURL, 4)}/>
          <Button title={buttonsBarrow2[5]} onPress={(param) =>
this.pathSet(this.state.imageURL, 5)}/>
          <Button title={buttonsBarrow2[6]} onPress={(param) =>
this.pathSet(this.state.imageURL, 6)}/>
          <Button title={buttonsBarrow2[7]} onPress={(param) =>
this.pathSet(this.state.imageURL, 7)}/>
          <Button title="Ground Floor" onPress={() =>
this.Load_New_Image(2)}/>
          <Button title="Hide"
onPress={this.Change_State_Toggle_False}/>
        </ScrollView>
      );
  }
  else if (image == mapImage && this.state.toggle) {
    var output = [];
    var i = 0;
```

```
    for (i; i < 9; i++) {
      var temp = (<Button title={buttonsMain[i]} key={i}
onPress={(param) => this.calcPath(1 , 1)}/>);
      output[i] = temp;
    }
      return (
        <ScrollView style={styles.container}>
          <Button title="Campus Map" onPress={() =>
this.Load_New_Image(0)}/>
          {output}
          <Button title="Hide"
onPress={this.Change_State_Toggle_False}/>
        </ScrollView>
      );
  }
  else if (image == gaa_ground && this.state.toggle) {
    var output = [];
    var i = 0;
    for (i; i < 4; i++) {
      var temp = (<Button title={buttonsGAA[i]} key={i}
onPress={(param) => this.pathSet(this.state.imageURL, i)}/>);
      output[i] = temp;
    }
      return (
        <ScrollView style={styles.container}>
          <Button title="Campus Map" onPress={() =>
this.Load_New_Image(0)}/>
          {output}
          <Button title="Hide"
onPress={this.Change_State_Toggle_False}/>
        </ScrollView>
      );
  }
  else if (image == dargan_ground && this.state.toggle) {
    var output = [];
    var i = 0;
    for (i; i < 4; i++) {
      var temp = (<Button title={buttonsDargan[i]} key={i}
onPress={(param) => this.pathSet(this.state.imageURL, i)}/>);
      output[i] = temp;
```

```
    }
      return (
        <ScrollView style={styles.container}>
          <Button title="Campus Map" onPress={() =>
this.Load_New_Image(0)}/>
          {output}
          <Button title="Hide"
onPress={this.Change_State_Toggle_False}/>
        </ScrollView>
      );
  }
  else if (image == haughton_ground && this.state.toggle) {
    var output = [];
    var i = 0;
    for (i; i < 4; i++) {
      var temp = (<Button title={buttonsHaughton[i]} key={i}
onPress={(param) => this.pathSet(this.state.imageURL, i)}/>);
      output[i] = temp;
    }
      return (
        <ScrollView style={styles.container}>
          <Button title="Campus Map" onPress={() =>
this.Load_New_Image(0)}/>
          {output}
          <Button title="Hide"
onPress={this.Change_State_Toggle_False}/>
        </ScrollView>
      );
  }
  else if (image == slaney_ground && this.state.toggle) {
    var output = [];
    var i = 0;
    for (i; i < 4; i++) {
      var temp = (<Button title={buttonsSlaney[i]} key={i}
onPress={(param) => this.pathSet(this.state.imageURL, i)}/>);
      output[i] = temp;
    }
      return (
        <ScrollView style={styles.container}>
```

```
          <Button title="Campus Map" onPress={() =>
this.Load_New_Image(0)}/>
          {output}
          <Button title="Hide"
onPress={this.Change_State_Toggle_False}/>
        </ScrollView>
      );
  }
  if (this.state.toggle == false) {
    return(
      <TouchableOpacity  style={styles.button}
onPress={this.Change_State_Toggle_True}>
        <Text style={{color: 'white'}}>Toggle</Text>
      </TouchableOpacity>
    );
  }*/
}
  render() {
      return (
          <><Header title="Map" navigation={this.props.navigation}
/>
              <View style={{paddingTop: size.top, flex: 1}}>
                <ImageZoom
cropWidth={Dimensions.get('window').width}
cropHeight={Dimensions.get('window').height * 0.6}
imageWidth={size.s} imageHeight={size.s2}>
                  <TouchableOpacity onPress={(evt) =>
this.handlePress(evt) }>
                  <View id="map" style={{flex: 1, alignContent:
'center', alignItems: 'center'}}>
                    <Image source={{uri: this.state.imageURL}}
style={styles.image} />
                  </View>
                  </TouchableOpacity>
                  <View style={{flex: 0}}>
                    {this.mainMapSVG(this.state.imageURL)}
                    {}
                  </View>
                </ImageZoom>
            </View>
```

```
            <View style={{flex: 0.5, paddingTop: '0%'}}>
              {this.buttons(this.state.imageURL)}
              {}
            </View>
          </>
    );
  }
}
```