NYIT

Department of Computer Science

CSCI-380-W01

Introduction to Software Engineering

Spring 2018

Parking Reservation Application

Professor Maherukh Akhtar

Rajendra Bhagroo (Leader)
516-439-7066 | Rbhagroo@nyit.edu | 1066410

Memembers

Christopher Guevarra
516-330-2081 | Cguevarr@nyit.edu | 1070954

Chengjun Dong
646-508-2328 | cdong01@nyit.edu | 1078296

Myeongkeun Kim
347-256-1433 | mkim28@nyit.edu | 1077180

Table of Contents

Abstract

Application Title: Parking Reservation Application

We approached the task of implementing our parking reservation software by using specific technologies to develop an application targeted to execute on the android mobile platform. To accomplish our goal, we used various technologies such as Javascript, Node.js, React-Native, Java, Android-Studio and Google Firebase. Node.js allows us to write server-side Javascript code using a Javascript library written by Facebook known as React-Native. This Javascript library allows us to develop the user interface for our application, with Google Firebase handling user authentication and login as well as our NoSQL database named Realtime-Database. We used Android-Studio and Java to integrate our application with the android platform so we can use our application on real devices and to integrate API's to work with our application such as Google Maps. Our application is targeted towards users aged between 18 and 40 who commute frequently in high traffic and commercial areas such as large cities or famous destinations, and have access to a smartphone running the Android operating system.

Keywords: Android, Multiple-Reservations, Firebase, React-Native, NoSQL, Realtime

Introduction

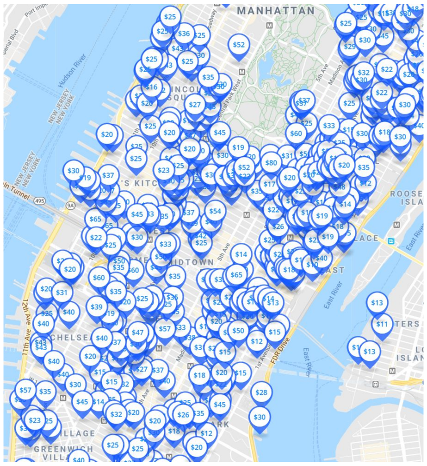
1.1) Purpose

The purpose of our application is to provide the owner of any parking-system establishment a tool that offers their customers a convenient and reliable method to reserve a maximum of three parking spaces on their own mobile device, as opposed to manually confirming reservation details on the spot, or by calling an employee.

1.2) Existing Applications / Models

1.2.1) Spot Hero

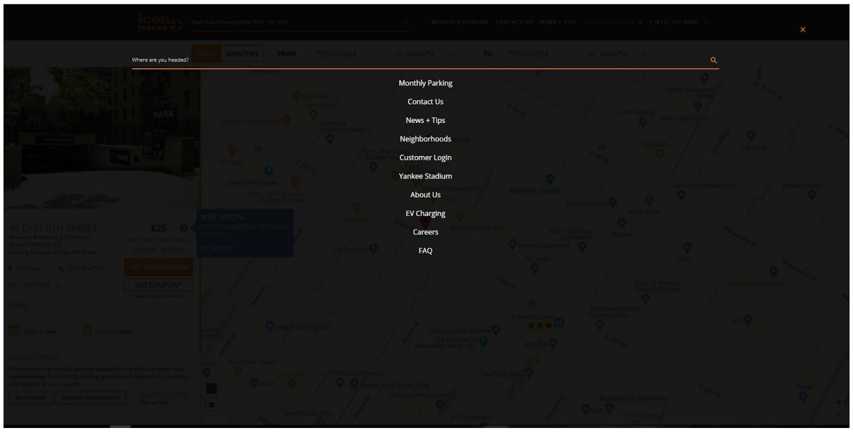
- Explanation:
  - Spot Hero is the most prominent parking reservation application on the market today. This service allows you to reserve parking spots exclusively within parking garages and valet parking areas. A user can set his or her reservation details and view information about the parking garages such as hours of operation and other information such as wheelchair accessibility.
- Disadvantages:
  - User Interface becomes extremely cluttered because it loads every single parking lot location on your map view simultaneously

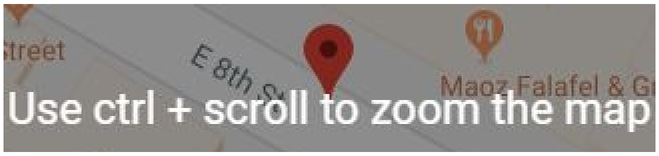


  - This service only targets parking garages and valet parking, disregarding other parking establishments such as hotels, airports and private companies
  - User may be unaware of the advantages from the map view because it takes an excessive amount of time to render all of the parking locations with their respective prices onto the map.

1.2.2) Icon Parking Systems

- Explanation:
  - Icon Parking Systems allows their users to exclusively reserve parking locations within New York. Unlike Spot Hero, this service allows their users to reserve parking spots based on events such as concerts or plays. Daily or monthly parking passes are sold through this platform per location.
- Disadvantages:
  - Parking reservations are restricted to the area of New York
  - Mobile first approach makes it difficult to use their service on a desktop, examples include obtrusive navigation panel and inability to scroll on the map via mouse scrollwheel without use of keyboard.





Use ctrl + scroll to zoom the map

  - User must have a specific address to enter to view map and choose parking location. There is no other way to access the map.

1.2.3) Other Existing Applications

- There are various existing applications and models we did not mention such as ParkWhiz and Parking Panda. We chose not to study these solutions in depth for the sake of brevity. These solutions also contain similar features and disadvantages as the applications listed above.

1.3) Proposed Idea

In consideration of the existing software solutions, we chose to approach the task of building a parking reservation system by targeting the android mobile platform and providing our users the unique functionality of reserving upto three parking spots simultaneously. We accomplished our goal by leveraging the power of a NoSQL database through Google Firebase and their NoSQL Realtime-Database. NoSQL allows us to implement multiple reserved spots as a JSON array within the database. The exchange of information through JSON works perfectly with our technology stack because we chose to use Javascript and the React-Native Javascript library to implement our application. JSON stands for Javascript Object Notation and the conversion from Javascript objects to JSON is simplistic, allowing us to send and retrieve data smoothly between our application and our database.

1.3.1) Software Engineering Model

To develop our application we chose to utilize the Spiral Model. We chose this model over other models such as waterfall, prototype and incrementals models because of its inclusion of risk analysis and team structure. Technical risks such as the storage of user passwords and authentication as well as cost of production was considered in the analysis stage of software development. The cost of production was not assessed on real money, instead we assessed the cost of production in correlation to the cost of time it will take to implement each feature based on the number of developers we have. We logically segmented our team based on features such as information retrieval and implementation of our NoSQL database, user interface, and overall semantic structure of our system and code base. That includes anything from the logical segmentation and slicing of our code base to formatting the code for consistency with our developer code guidelines.

1.4) User Access Levels

Our application distinguishes user access levels based on authentication. The two groups of users represented within our application are authenticated users and anonymous users. Our application functions in an establishment and consumer fashion, therefore establishments implement our software with the expectation that not every user will sign up for the service. Instead of forcing users to sign up for our service, we distinguish between authenticated users and anonymous users with one slight difference. Authenticated users are able to view their current reservations in a proprietary view, allowing them to modify their current reservations by extending or deleting reservations in various establishments. Anonymous users do not have this functionality, however they are able to use our application for initial reservations.

1.5) Technologies

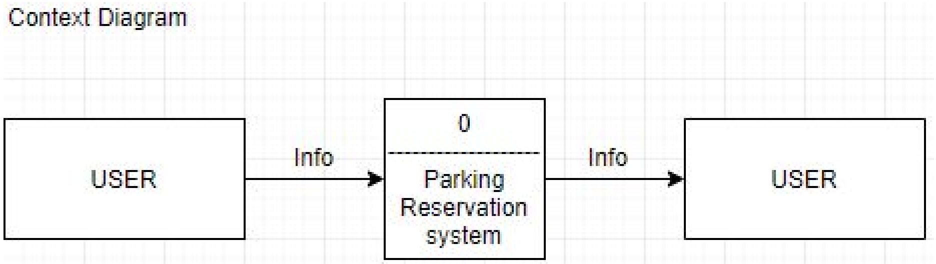
- Languages
    - Java SE 8
    - Javascript ECMAScript 6
        - Node.js [Enables Javascript On Back End] [8.10.0]
        - React-Native [Front End UI Development] [0.5.4]
- Tools
    - Android Studio [3.0.1]
        - Integrated Development Environment
    - Atom [1.24.1]
        - Lightweight Text Editor
    - Google Firebase [Back End]
        - User Registration & User Authentication
        - Firebase Realtime Database [NoSQL]
    - Git & Github [Concurrent Developer Tracking]

1.5.1) Implementation Of Technologies

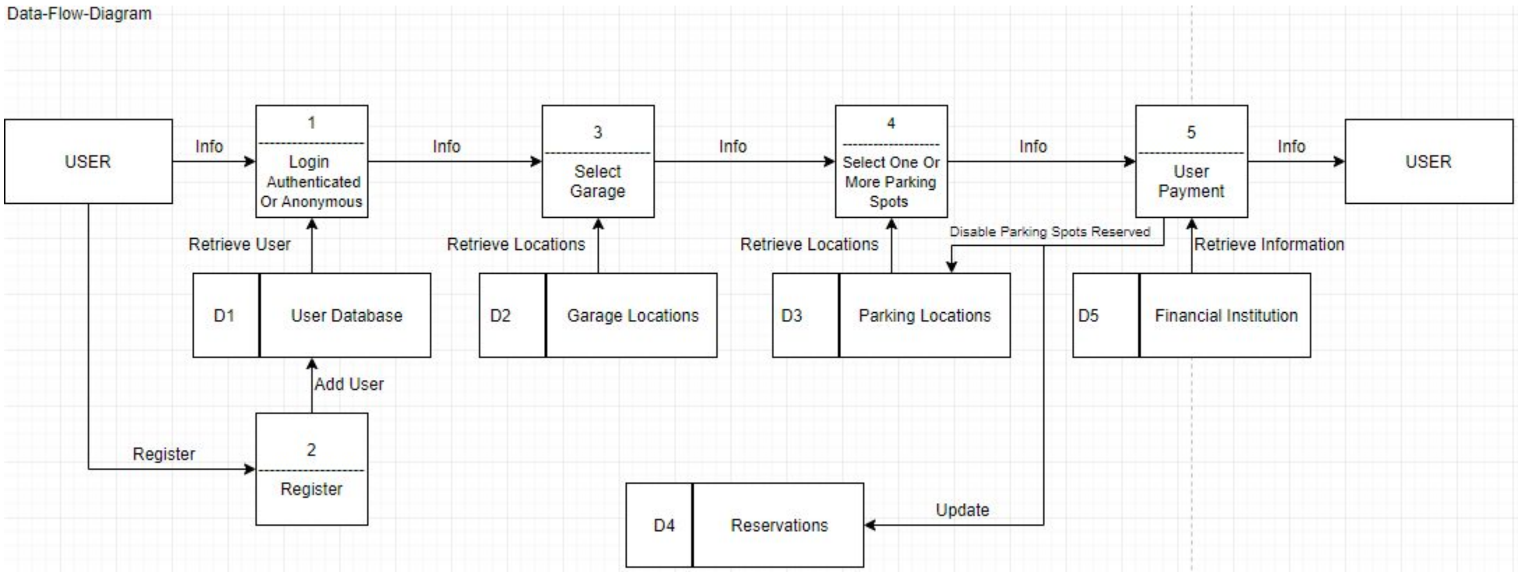
For an overview on the specific uses of each technology, please refer to the abstract section on page 2 within this documentation.

2.0) Analysis

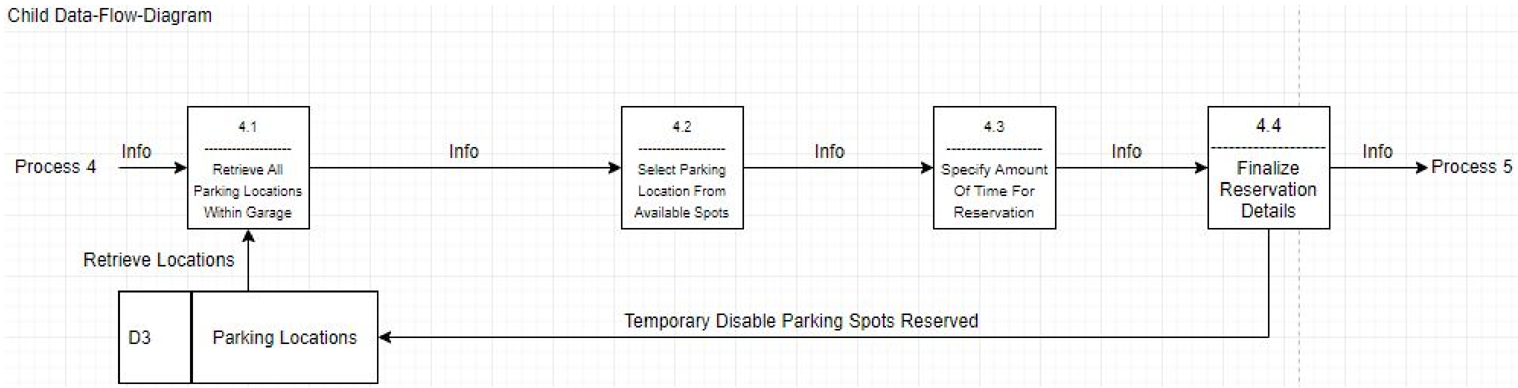
2.0.1) Context Diagram



2.0.2) Data Flow Diagram



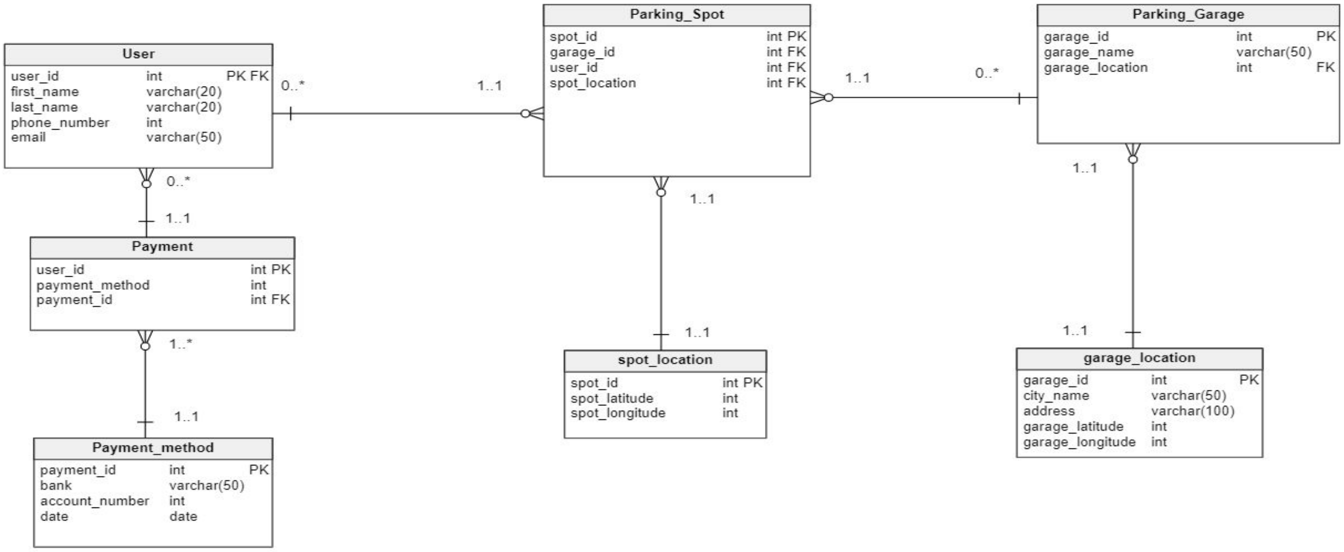
2.0.3) Child Data Flow Diagram


Child Data-Flow-Diagram
Process 4
Info
4.1
Retrieve All Parking Locations Within Garage
Info
4.2
Select Parking Location From Available Spots
Info
4.3
Specify Amount Of Time For Reservation
Info
4.4
Finalize Reservation Details
Info
Process 5
Retrieve Locations
D3
Parking Locations
Temporary Disable Parking Spots Reserved

3.0) Data Modeling

This project was designed using Google Firebase's NoSQL Realtime-Database. We decided to leverage the power of a NoSQL database because it works well with our technology stack. Previously this was explained within section 1.3 when we proposed our idea. NoSQL allows us to implement multiple reserved spots as a JSON array within the database. The exchange of information through JSON works perfectly with our technology stack because we chose to use Javascript and the React-Native Javascript library to implement our application. JSON stands for Javascript Object Notation and the conversion from Javascript objects to JSON is simplistic, allowing us to send and retrieve reservation data smoothly between our application and our database.

3.0.1) Table Schema



User		
user_id	int	PK FK
first_name	varchar(20)	
last_name	varchar(20)	
phone_number	int	
email	varchar(50)	

Parking_Spot	
spot_id	int PK
garage_id	int FK
user_id	int FK
spot_location	int FK

Parking_Garage		
garage_id	int	PK
garage_name	varchar(50)	
garage_location	int	FK

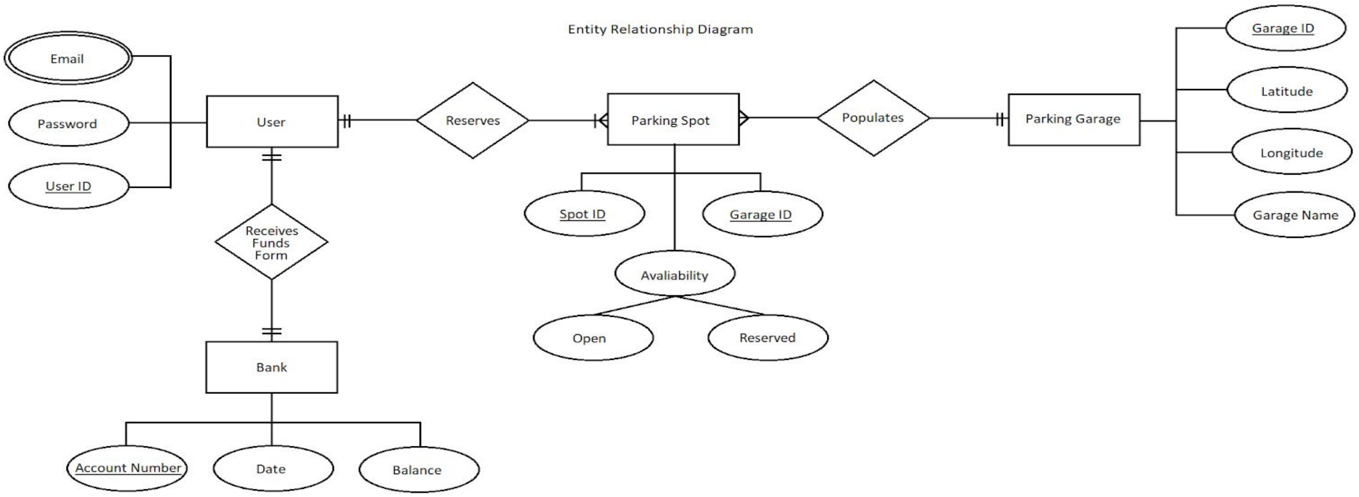
Payment	
user_id	int PK
payment_method	int
payment_id	int FK

spot_location	
spot_id	int PK
spot_latitude	int
spot_longitude	int

garage_location		
garage_id	int	PK
city_name	varchar(50)	
address	varchar(100)	
garage_latitude	int	
garage_longitude	int	

Payment_method		
payment_id	int	PK
bank	varchar(50)	
account_number	int	
date	date	

3.0.2) Entity Relationship Diagram



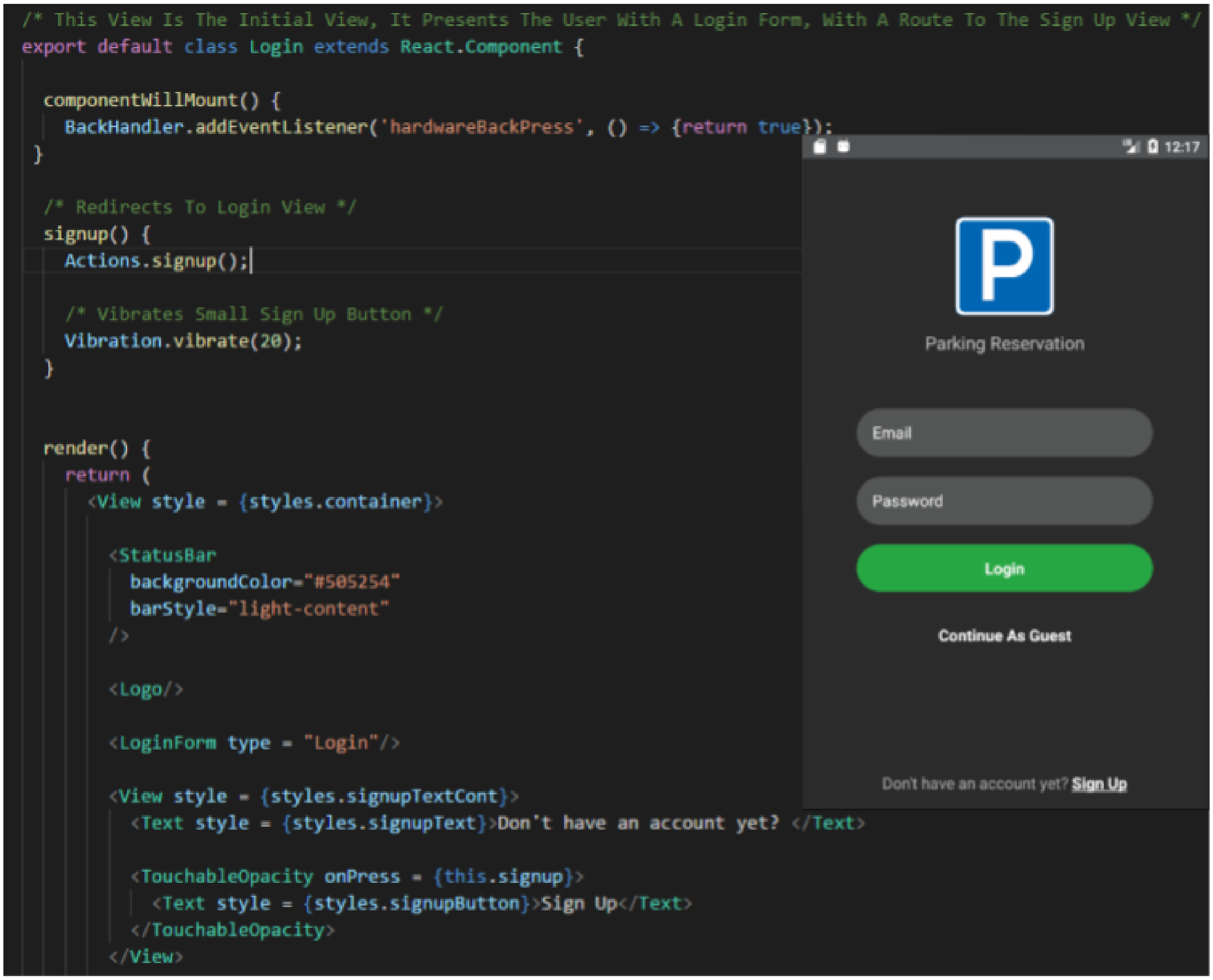
4.0) Implementation

This project was designed using technologies such as…

- Javascript
- Node.js
- React-Native
- Java
- Android-Studio
- Google Firebase

We had chosen **React Native** over other mobile development languages because it allowed us to avoid the steep learning curve needed to develop Android & IOS applications. To develop in an Android environment alone would have taken months of time to grasp the concepts needed to create a market ready application. Luckily React Native is in a sense a flavor of **Javascript**, so if one knows Javascript they could easily work with React Native. **Node.js** is a JavaScript runtime open source server environment that enabled us to develop and test our application as it was being developed. Additionally we used **Android Studio** which is the official IDE for the Android operating system, to further assist us in the development of the application. Finally we used Google's **Firebase** which allowed us to set up database without needing knowledge of a server side programming language. Firebase allowed us to store user data on its real-time database as well as syncing up data seamlessly.

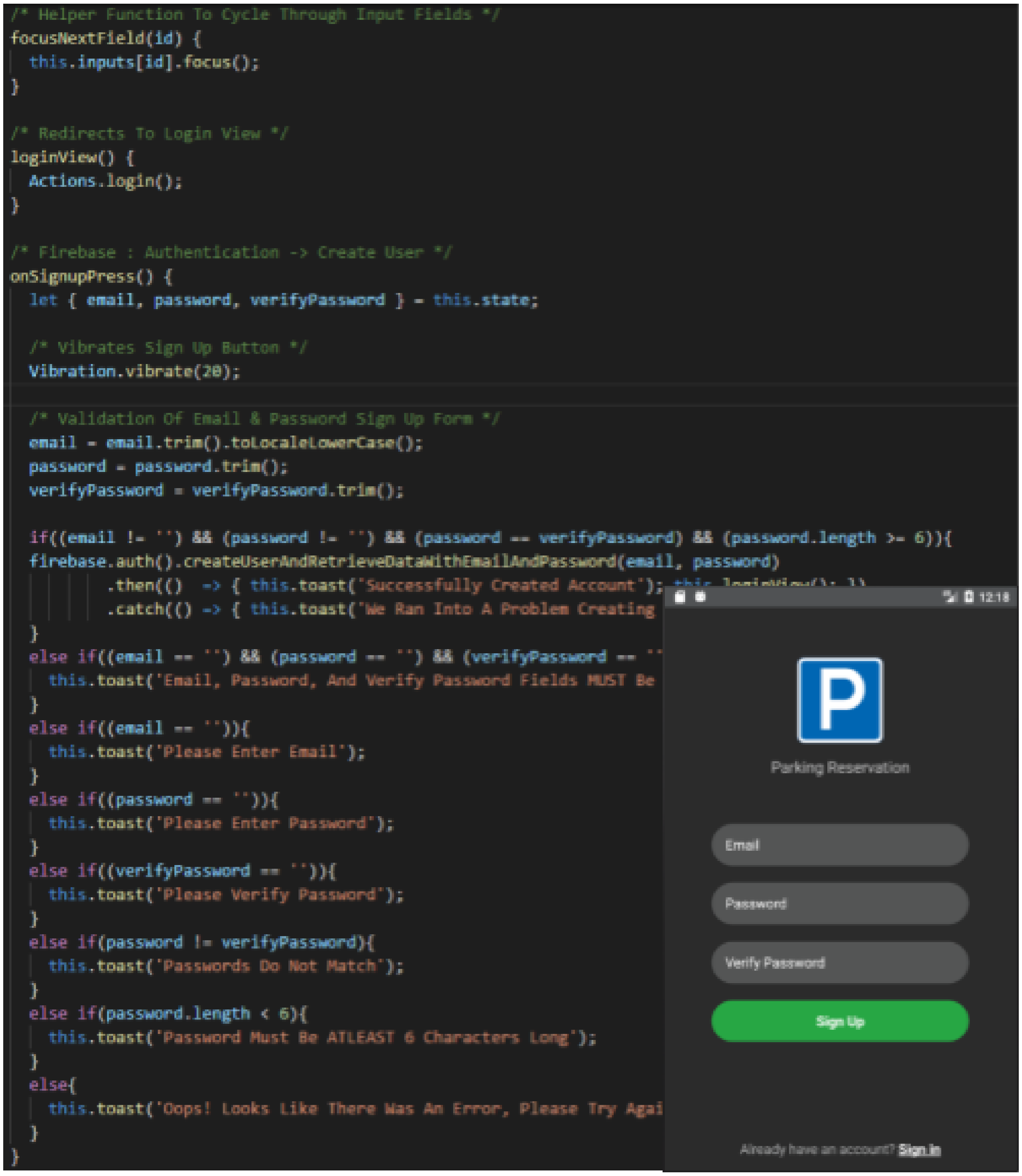
4.0.1) Login Page



```
/* This View Is The Initial View, It Presents The User With A Login Form, With A Route To The Sign Up View */
export default class Login extends React.Component {

  componentWillMount() {
    BackHandler.addEventListener('hardwareBackPress', () => {return true});
  }

  /* Redirects To Login View */
  signup() {
    Actions.signup();

    /* Vibrates Small Sign Up Button */
    Vibration.vibrate(20);
  }


  render() {
    return (
      <View style = {styles.container}>

        <StatusBar
          backgroundColor="#505254"
          barStyle="light-content"
        />

        <Logo/>

        <LoginForm type = "Login"/>

        <View style = {styles.signupTextCont}>
          <Text style = {styles.signupText}>Don't have an account yet? </Text>

          <TouchableOpacity onPress = {this.signup}>
            <Text style = {styles.signupButton}>Sign Up</Text>
          </TouchableOpacity>
        </View>
```

This is login page when users launch the application. Users can either sign in if registered, sign up or continue as guest as they prefer. Email and Password has to be matched as the data in Firebase. After hit the login button, it will direct to the google map.

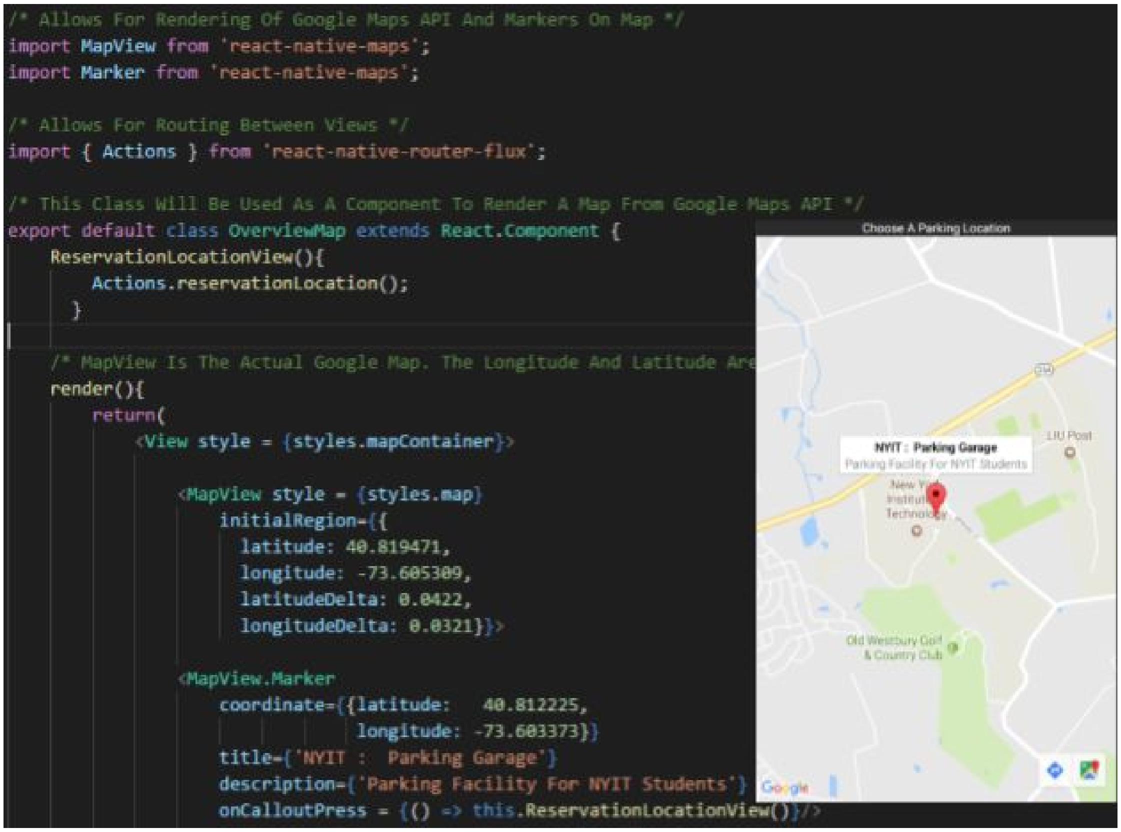
4.0.3) Sign up Page



```
/* Helper Function To Cycle Through Input Fields */
focusNextField(id) {
  this.inputs[id].focus();
}

/* Redirects To Login View */
loginView() {
  Actions.login();
}

/* Firebase : Authentication -> Create User */
onSignupPress() {
  let { email, password, verifyPassword } = this.state;

  /* Vibrates Sign Up Button */
  Vibration.vibrate(20);

  /* Validation Of Email & Password Sign Up Form */
  email = email.trim().toLocaleLowerCase();
  password = password.trim();
  verifyPassword = verifyPassword.trim();

  if((email != '') && (password != '') && (password == verifyPassword) && (password.length >= 6)){
  firebase.auth().createUserAndRetrieveDataWithEmailAndPassword(email, password)
          .then(()  => { this.toast('Successfully Created Account'); this.loginView(); })
          .catch(() => { this.toast('We Ran Into A Problem Creating
  }
  else if((email == '') && (password == '') && (verifyPassword == ''
    this.toast('Email, Password, And Verify Password Fields MUST Be
  }
  else if((email == '')){
    this.toast('Please Enter Email');
  }
  else if((password == '')){
    this.toast('Please Enter Password');
  }
  else if((verifyPassword == '')){
    this.toast('Please Verify Password');
  }
  else if(password != verifyPassword){
    this.toast('Passwords Do Not Match');
  }
  else if(password.length < 6){
    this.toast('Password Must Be ATLEAST 6 Characters Long');
  }
  else{
    this.toast('Oops! Looks Like There Was An Error, Please Try Agai
  }
}
```

This is *Sign Up* page where users have to enter a valid email which includes @, and matches passwords, else users will not sign up. If users sucessily create their account, informations will be updated and stored in Firebase, make authentications each time when user trying to sign in. After they hit sign up, it will redirect to the sign in page.

4.0.4) Google Map

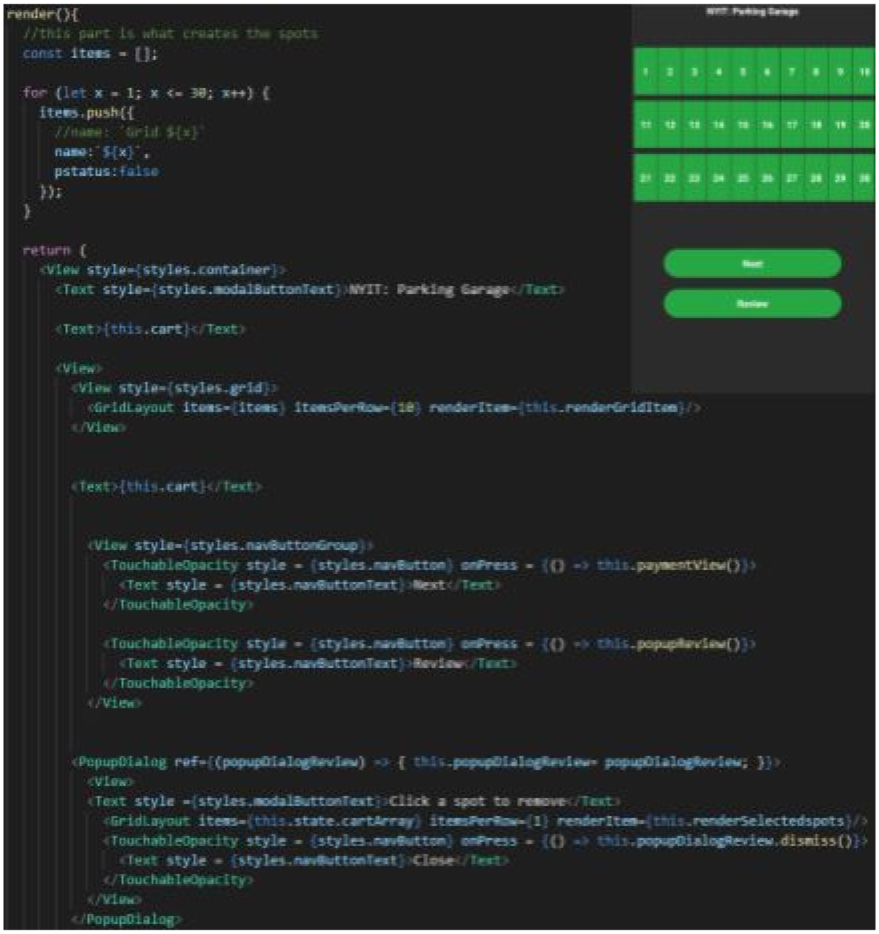


```
/* Allows For Rendering Of Google Maps API And Markers On Map */
import MapView from 'react-native-maps';
import Marker from 'react-native-maps';

/* Allows For Routing Between Views */
import { Actions } from 'react-native-router-flux';

/* This Class Will Be Used As A Component To Render A Map From Google Maps API */
export default class OverviewMap extends React.Component {
    ReservationLocationView(){
        Actions.reservationLocation();
      }

    /* MapView Is The Actual Google Map. The Longitude And Latitude Are
    render(){
        return(
            <View style = {styles.mapContainer}>

                <MapView style = {styles.map}
                    initialRegion={{
                      latitude: 40.819471,
                      longitude: -73.605309,
                      latitudeDelta: 0.0422,
                      longitudeDelta: 0.0321}}>

                <MapView.Marker
                    coordinate={{latitude:   40.812225,
                                 longitude: -73.603373}}
                    title={'NYIT :  Parking Garage'}
                    description={'Parking Facility For NYIT Students'}
                    onCalloutPress = {() => this.ReservationLocationView()}/>
```

Choose A Parking Location
NYIT : Parking Garage
Parking Facility For NYIT Students
LIU Post
New York Institute of Technology
Old Westbury Golf & Country Club
Google

This is the page of the Google Map. For now it located at NYIT Old Westbury Campus, however, companies can add more locations if it is necessary. Users have to click "*NYIT: Parking Garage*" in order to get in *Reservation Page*.

4.0.5) Reservation page



```
render(){
  //this part is what creates the spots
  const items = [];

  for (let x = 1; x <= 30; x++) {
    items.push({
      //name: `Grid ${x}`
      name:`${x}`,
      pstatus:false
    });
  }

  return (
    <View style={styles.container}>
      <Text style={styles.modalButtonText}>NYIT: Parking Garage</Text>

      <Text>{this.cart}</Text>

      <View>
        <View style={styles.grid}>
          <GridLayout items={items} itemsPerRow={10} renderItem={this.renderGridItem}/>
        </View>


        <Text>{this.cart}</Text>


          <View style={styles.navButtonGroup}>
            <TouchableOpacity style = {styles.navButton} onPress = {() => this.paymentView()}>
              <Text style = {styles.navButtonText}>Next</Text>
            </TouchableOpacity>

            <TouchableOpacity style = {styles.navButton} onPress = {() => this.popupReview()}>
              <Text style = {styles.navButtonText}>Review</Text>
            </TouchableOpacity>
          </View>


        <PopupDialog ref={(popupDialogReview) => { this.popupDialogReview= popupDialogReview; }}>
          <View>
          <Text style ={styles.modalButtonText}>Click a spot to remove</Text>
            <GridLayout items={this.state.cartArray} itemsPerRow={1} renderItem={this.renderSelectedspots}/>
            <TouchableOpacity style = {styles.navButton} onPress = {() => this.popupDialogReview.dismiss()}>
              <Text style = {styles.navButtonText}>Close</Text>
            </TouchableOpacity>
          </View>
        </PopupDialog>
```

This is reservation page, users can select up to THREE parking spots. After users select a spot, it require users to select the range of time that they wish to park. After they hit next, it will redirect to *Payment Page*. If they click review, a popup will display their current selected spots, which they can cancel or not before proceeding to the *Payment Page*

5.0) Conclusion

5.1) Testing

The type of testing done was regression testing. Regression testing is the repeated testing after modification of the program. When we add and modify each others codes, we keep run the application to see the code works fine or makes any errors. Our scope of project application is not really big, so we could repeat testing the application everytime we modified the code. If we find any error while testing, we fixed it and tested it again to go next step of development.

5.2) Team Members Contribution

Leader: Rajendra Bhagroo

- Login View & Login Form Component
- Registration View & Registration Form Component
- Integration Of Google Maps API With Android Studio
- Map View & Map Overview Component
- Integration Of Google Firebase With Android Studio
- Implementation Of Gradle Files Used By Android Studio
- Implementation of Anonymous Users, Authenticated Users & Verification / Validation Of Authenticated Users
- Code-Consistency In Cooperation With Developer Code Guidelines
- Overall Styling Of Elements Across Project
- Entity Relationship Diagram (1)
- Constructed Final Data Flow Diagram
- Constructed Final Child Data Flow Diagram
- Written Project Proposal
- Inclusion Of Haptic Feedback For A Better User Experience On Mobile Devices

Member 1: Christopher Guevarra

- Reservation Location View
- Data Flow Diagram
- Child Data Flow Diagram
- Parking Space Data Format
- Debugging

Member 2: Chengjun Dong

- Table Schema Diagram

- Payment View
- Project Proposal

Member 3: Myeongkeun Kim

- Main Menu View
- Registration Review View
- Block Hardware Back
- Entity Relationship Diagram (2)

5.3) Learning Outcomes

Leader: Rajendra Bhagroo

- I used this project as a challenge to utilise technologies that I am not familiar with in order to gauge learning capabilities in a team environment. I enjoyed creating a mobile application and learning about new technologies such as react-native and integrating the Google Maps API. Also, this is an important project where I got to work in a team of developers and learn how to share and incorporate ideas and feedback into the project as we were in the code implementation phase. I also learned how to properly budget time to expect the unexpected and finish the project while continuously meeting deadlines.

Member 1: Christopher Guevarra

- What I took away from this project was basically experience learning a new language on the fly, as well as how to work in a team setting. I know in the actual work field there will be times when I'm left with no choice but to learn a the bare necessities of a language to meet project requirements and deadlines. After completing this project I'm assured if this situation ever comes up again I'll be able to face it easily, but personally would still prefer to know all the nitty gritty parts of a language before using it for market ready applications. Additionally I've learned that communication is key in a team setting.

Member 2: Chengjun Dong

- Before I started this project, I have never heard of React-native and Android studio. I have known what is github and node before, but I have never used it. In addition, my laptops does not run the whole app because node.js and java SE had conflict, have tried many times to reinstall from the beginning but did not work eventually. However, I have managed to create Payment View page with Expo. I also learned that communications between teammates and greater team leading skill is really important for the team project. Finally, even though I have learned javascript for html, there is difference with react-native, and I am getting hand of it.

Member 3: Myeongkeun Kim

- From this Parking Reservation Application development project, I learned a lot of new things such as React Native, Android Studio, Node Js, Github, Firebase, and more about Java language. I never heard about these programs before and did not use them. I learned how to run the emulator in Android Studio and connect React Native and Node Js together. The React Native helps and allows to use JavaScript to develop the application. So I could learned more about JavaScript's codes that I don't know before.

6.0) References

6.0.1) Development:

https://facebook.github.io/react-native/

https://developers.google.com/maps/documentation/

https://www.npmjs.com/package/react-native-layout-grid

https://github.com/react-community/react-native-maps

https://github.com/aksonov/react-native-router-flux

https://firebase.google.com/

https://firebase.google.com/docs/guides/?authuser=0

6.0.2) Research:

https://spothero.com/

https://iconparkingsystems.com/

https://www.parkwhiz.com/

https://www.parkingpanda.com/

6.0.3) Logo:

https://upload.wikimedia.org/wikipedia/commons/thumb/c/cb/Japanese_Road_sign_(Parking_lot_A,_Parking_permitted).svg/450px-Japanese_Road_sign_(Parking_lot_A,_Parking_permitted).svg.png

https://commons.wikimedia.org/wiki/Main_Page

https://commons.wikimedia.org/wiki/Commons:Licensing