A Project Report

on

PARKIT
Online Parking Slot Booking System

Submitted in partial fulfillment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

by

Yasaswini Annem	**(16WH1A1206)**
Sunvitha Kandem	**(16WH1A1227)**
Soundarya Kasarla	**(16WH1A1237)**
Swetha Potti	**(16WH1A1252)**

Under the esteemed guidance of

Mr. Ch. Anil Kumar
Assistant Professor

Department of Information Technology
BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN
(Accredited by NBA & NAAC '**A**' Grade, Approved by AICTE, and
Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

April 2020

DECLARATION

We hereby declare that the work presented in this project entitled **"Online Parking Slot Booking System"** submitted towards completion of the major project in IV year of B.Tech IT at "BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN", Hyderabad is an authentic record of our original work carried out under the esteem guidance of Ch. Anil Kumar, Assistant Professor, IT department.

Yasaswini Annem

(16WH1A1206)

Sunvitha Kandem

(16WH1A1227)

Soundarya Kasarla

(16WH1A1237)

Swetha Potti

(16WH1A1252)

BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN

(Accredited by NBA & NAAC 'A' Grade, Approved by AICTE, and Affiliated to JNTUH, Hyderabad)

Bachupally, Hyderabad – 500090

Department of Information Technology

Certificate

This is to certify that the Project report on " **Online Parking Slot Booking System**" is bonafide work carried out by **Yasaswini Annem (16WH1A1206), Sunvitha Kandem (16WH1A1227), Soundarya Kasarla (16WH1A1237), Swetha Potti (16WH1A1252)** in the partial fulfillment for the award of B.Tech degree in **Information Technology, BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Internal Guide
Mr. CH. Anil Kumar
Assistant Professor
Department of IT

Head of the Department
Dr. Aruna Rao S L
Professor and HOD
Department of IT

External Examiner

Acknowledgements

We would like to express our sincere thanks to **Dr. K. V. N. Sunitha, Principal**, BVRIT Hyderabad, for providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr**. **Aruna Rao S L, Head, Dept. of IT**, BVRIT HYDERABAD for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Ch. Anil Kumar, Assistant Professor, Department of IT,** BVRIT HYDERABAD for his constant guidance, encouragement and moral support throughout the project.

Finally, We would also like to thank our Project Coordinator, all the faculty and staff of IT Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

Yasaswini Annem
(16WH1A1206)

Sunvitha Kandem
(16WH1A1227)

Soundarya Kasarla
(16WH1A1237)

Swetha Potti
(16WH1A1252)

ABSTRACT

The Online Parking Slot Booking System ia s system that enables customers/drivers to reserve a parking space. It also allows the customers/drivers to view the parking status at BVRIT HYDERABAD College of Engineering for Women. It was developed because of the congestion and collision of the vehicle, the system was developed. Therefore the project aimed at solving such problems by designing a android based system that will enable the customers/drivers to make a reservation of available parking space. The requirements for the developed system were collected using observation and interviewing the customer and staff members. The data was analyzed so as to come up with the functional, non- functional and system requirements. These requirements were later used to design the system by creating data flow diagrams and entity relationship diagrams. The designed system was implemented using different development tools which include JavaScript and reactJS is used to connect the user interface to the database.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

Term/Abbreviation	Definition
OPSBS	Online Parking Slot Booking System
QR Code	Quick Response Code
JS	Java Script
SQL	Structured Query Language
IDE	Integrated Development Environment

1. INTRODUCTION

The parking place is very important all over the world especially in the cities of the countries. Every day thousands of car drivers spend a lot of the time to find where to park. The result of this situation is theft in urban areas, increasing traffic congestion and frustration of drivers. In order to solve this problem, the implementation of Online Parking Slot Booking System in this city for managing parking places is mandatory. It will allow the drivers to Reserve a parking place on the Platform of BVRIT HYDERABAD College anytime, anywhere.

1.1 OBJECTIVE

To enable drivers to locate and reserve a parking place online through accessing it on web platform. To establish possible solutions to improve on the current Vehicle Parking Reservation system. To design and implement Online Vehicle Parking Reservation system. To make a good research and gather all necessary information that helped in designing

1.2 PROBLEM IN EXISTING SYSTEM

The Drivers especially those who may need get the parking spaces may find it impossible to access it since there could be other vehicles blocking the way and yet they have to hurry to book for parking spaces. This needs self-contact to reserve for parking and it’s also time consuming.

1.3 SOLUTION

Proposed system assists the driver in finding a vacant place in a short time. The most essential features of the OPSBS is to detect and present available parking spaces, provide payment facilities and various kinds of parking space. After finding the preferable space the drivers will follow the procedures for booking if there is a vacant space. This system will probably reduce time, cost and effort for finding a vacant place. They believe both car park operators and drivers will benefit from the system as parking spaces are easily acquired and parking space wastage is reduced.

1.4 FEATURES

- Booking slot
- View available parking slots
- Recent Bookings Display
- Displaying details along with QR Code
- Cancellation of slot

2. LITERATURE SURVEY

2.1 APPROACH:

2.1.1 ONLINE PARKING SLOT BOOKING SYSTEM:

Online Parking Slot Booking System is an Android Application used to reserve a parking slot. Drivers are no longer disturbed to park their vehicle since the system generate the parking lot number on OPSBS platform.

2.1.2 THE WAY DRIVERS LOCATE AND RESERVE PARKING SLOTS:

The Drivers especially those who may need get the parking spaces may find it impossible to access it since there could be other vehicles blocking the way and yet they have to hurry to book for parking spaces. This needs self-contact to reserve for parking and it’s also time consuming, to design online vehicle parking reservation system will provide better efficiency in locating parking space and paying for it.

online vehicle parking reservation will be also much faster, easier on both sides (means clients and company). The time taken to serve a client is significantly reduced.On company side this system will help their records (such as clients’ details) to keep it in a secure way.

2.2 INFORMATION GATHERING:

Proposed system assists the driver in finding a vacant place in a short time. The most essential features of the OPSBS is to detect and present available parking spaces, provide payment facilities and various kinds of parking space. After finding the preferable space the drivers will follow the procedures for booking if there is a vacant space. This system will probably reduce time, cost and effort for finding a vacant place. They believe both car park operators and drivers will benefit from the system as parking spaces are easily acquired and parking space wastage is reduced.

3. REQUIREMENT SPECIFICATION

3.1.1 SOFTWARE REQUIREMENTS

The software requirements specify the use of all required software products like data management system. The required software product specifies the numbers and version. Each interface specifies the purpose of the interfacing software as related to this software product.

➢ Operating system	:	Windows XP/7/10
➢ Development kit	:	Android Studio
➢ Technologies used	:	Java and React JS

3.1.2 HARDWARE REQUIREMENTS:

The hardware requirement specifies each interface of the software elements and the hardware elements of the system. These hardware requirements include configuration characteristics.

➢ Processor	:	1 gigahertz (GHz) or faster.
➢ RAM	:	1 gigabyte (GB) (32-bit) or 2 GB (64-bit)
➢ Free hard disk space	:	16 GB.

3.2 COMPONENTS:

A central server and mobile device system hardware QR code scanner is three main component. In the following, we communicate with the system between them, in order to discuss the design of each component and implementation details. In the following hardware system, the three main components, QR code scanner, is to be held in the central server and the mobile device, along with your surround configuration dialog, detailed design and implementation of each component during the discussion. The system includes hardware components, eg. Android smart phone. The user will have an Android smart phone app and parking Scanning QR code is a parking lot and administrator. Both phones must have an Internet connection. The smart phone is connected to the central server of the various working with SQL.

4. DESIGN OF THE SYSTEM

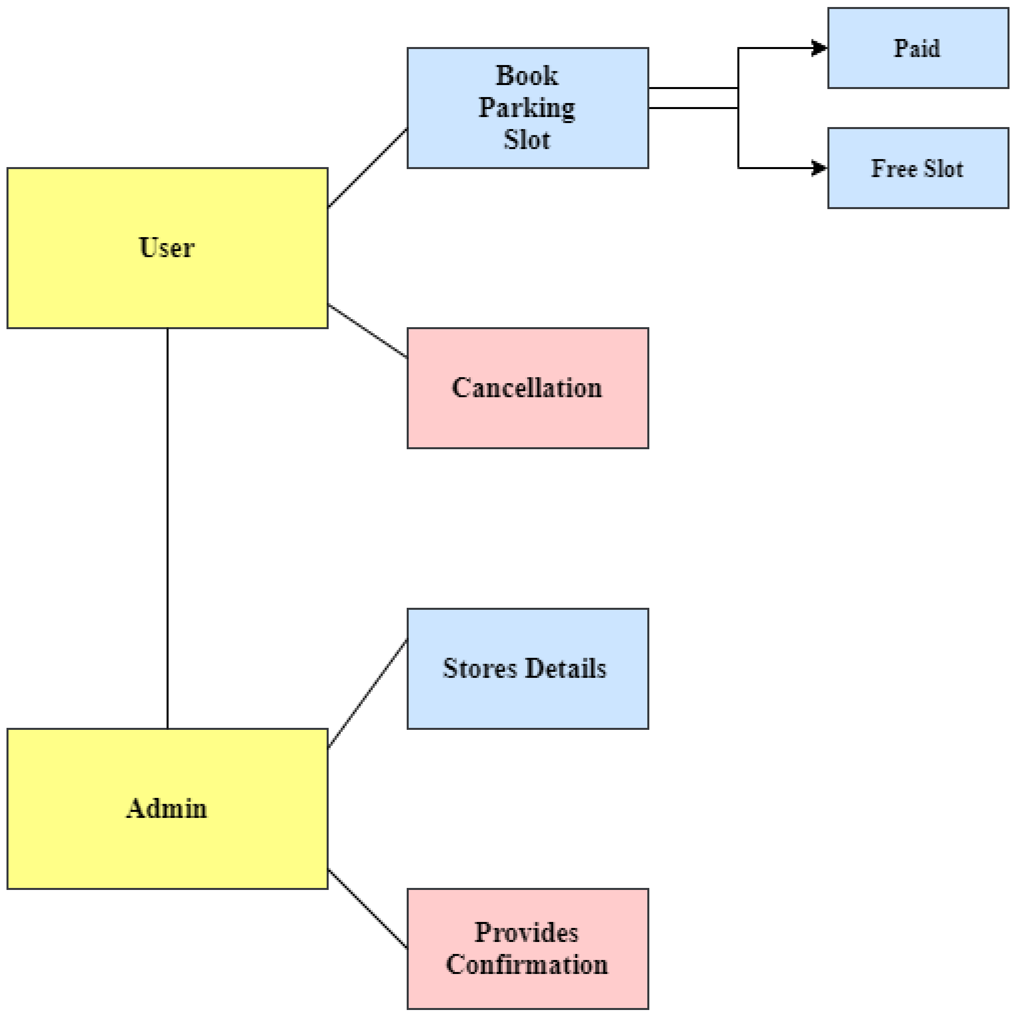


Fig 1. Architecture of PARKIT

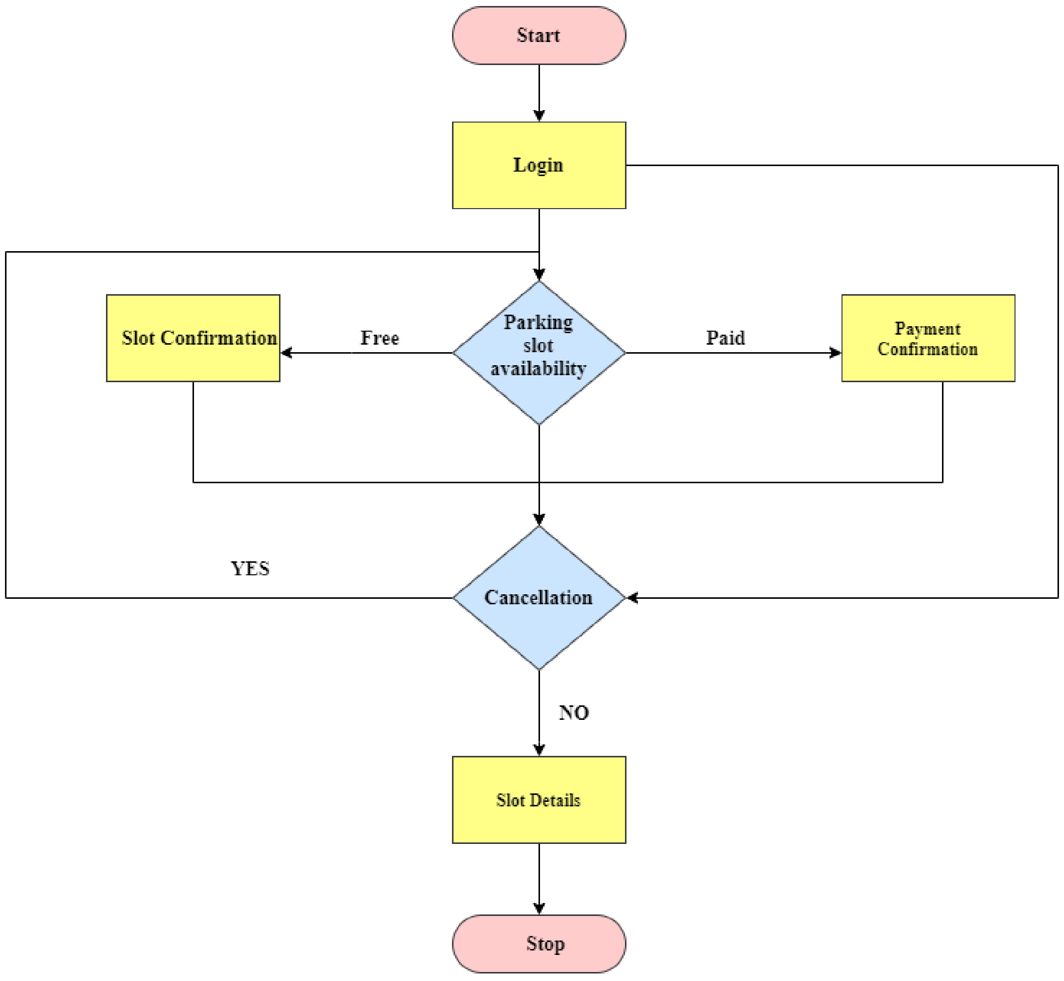


Fig 2. Method described through flowchart

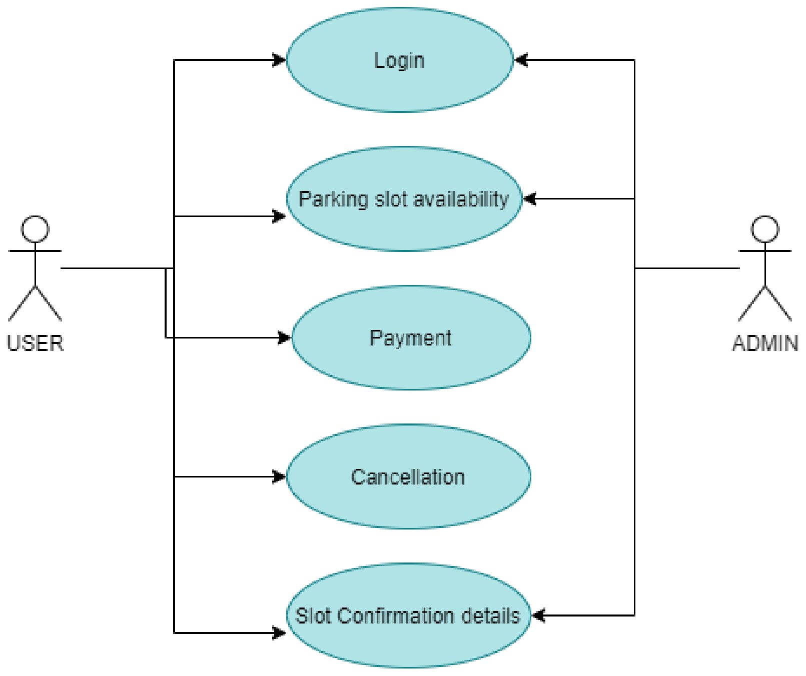


Fig 3. Usecase diagram for OPSBS

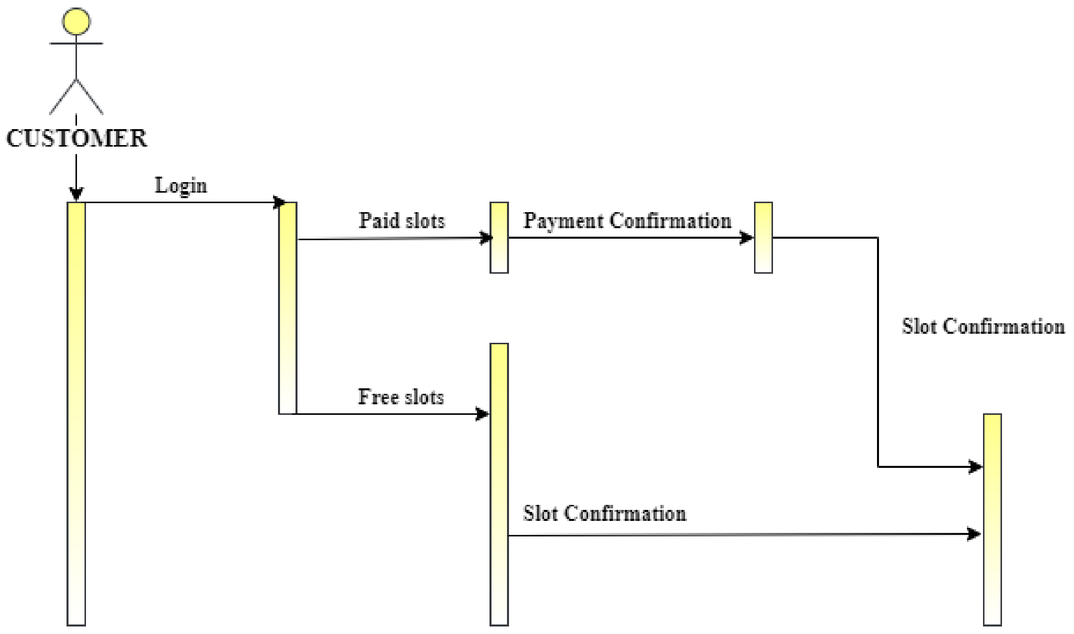


Fig 4. Sequence diagram for OPSBS

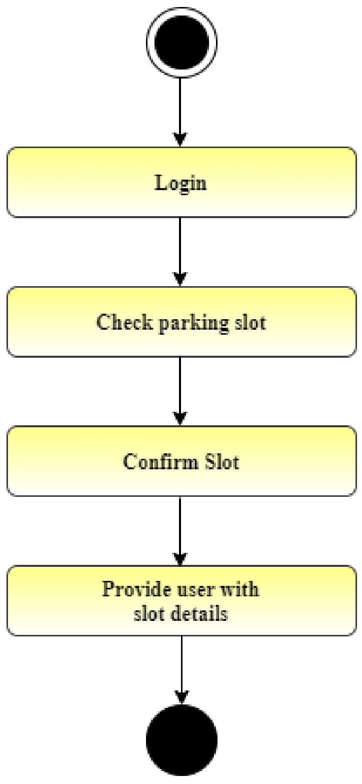
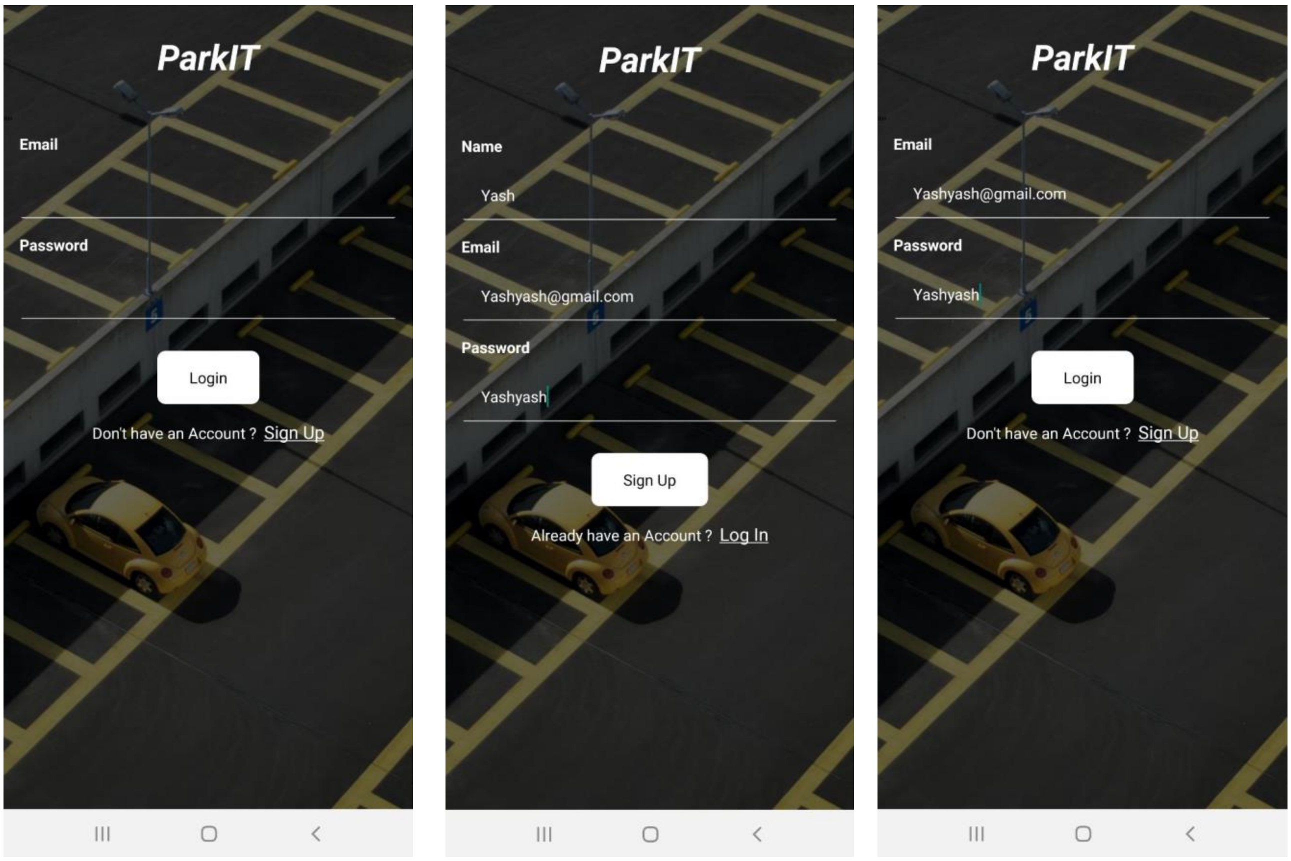


Fig 5. Activity diagram for OPSBS

5. MODULES

5.1 USER LOGIN AND REGISTRATION:

This module of the application deals with the user interface/user experience. This module provides the user with the flexibility of registering, logging in, booking and making the payment.


ParkIT
Email
Password
Login
Don't have an Account ? Sign Up
ParkIT
Sign Up
Already have an Account ? Log In
ParkIT
Login
Don't have an Account ? Sign Up

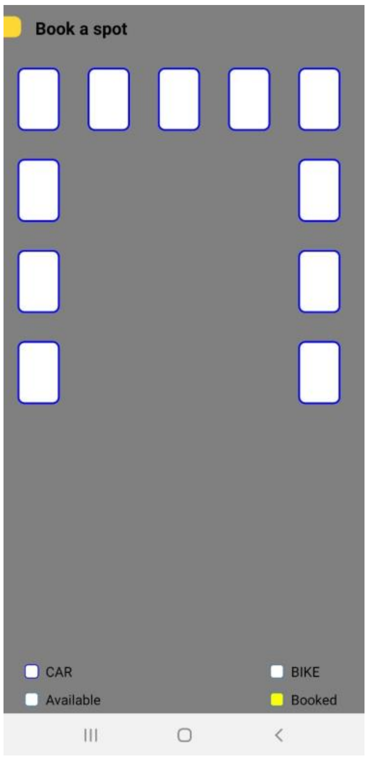
Name: Yash
Email: Yashyash@gmail.com
Password: Yashyash

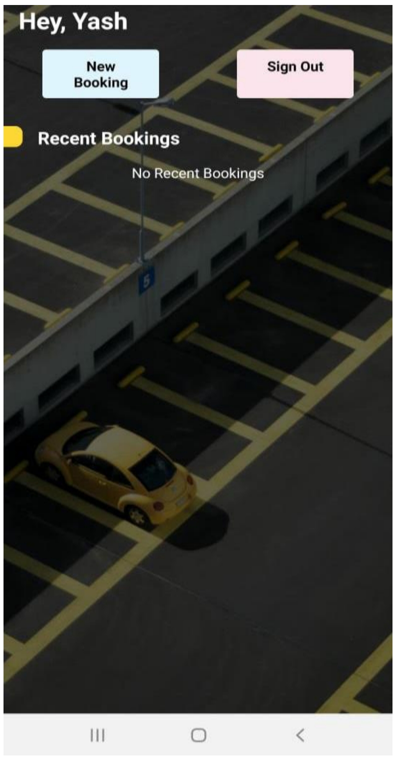
Email: Yashyash@gmail.com
Password: Yashyash

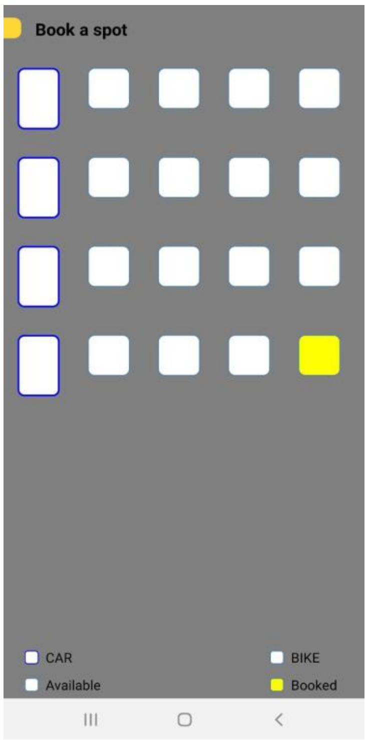
Fig 6. Login and Registration output screens

5.2 SEARCH PARKING AVAILABILITY:

User login the application where he can view various parking slots in his destination location. User selects his desired parking slot that is nearest to his destination. After selecting a slot the user needs to check for the availability of that respective slot.







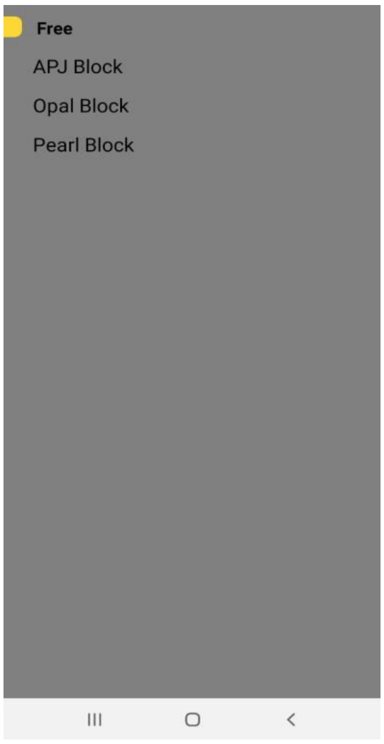
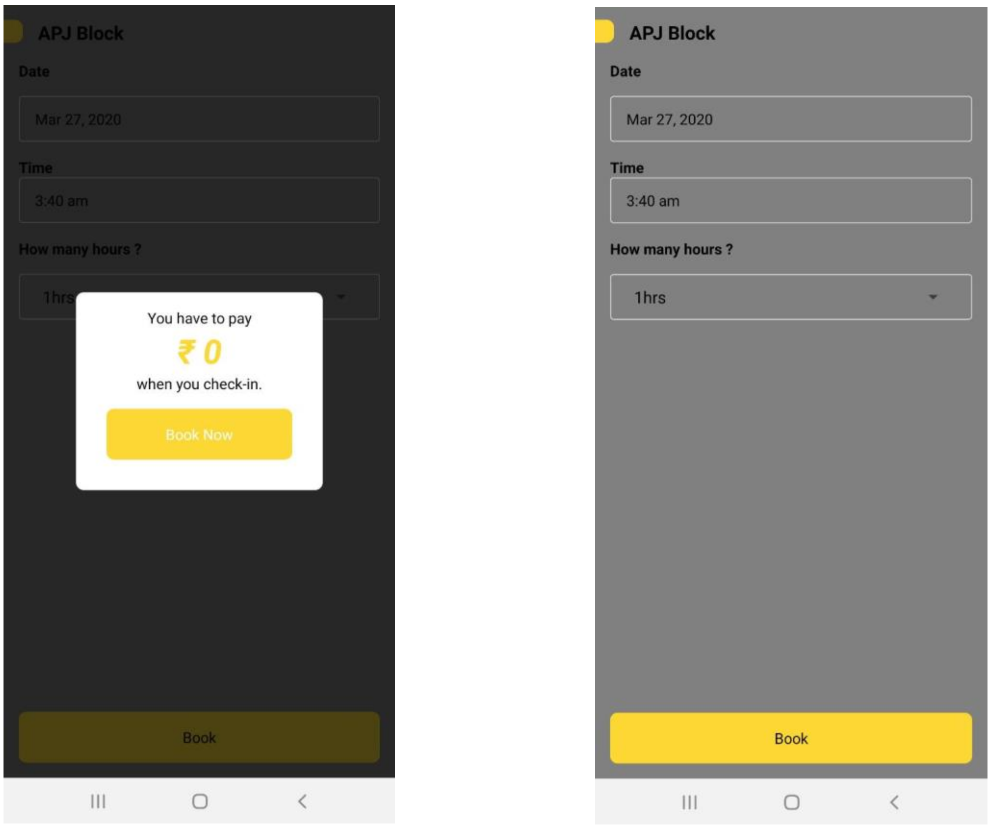


Fig 7. Output screens of searching parking availability

The user can check the status of the slots with the help of blue and yellow color indications. Where blue color indicates that the respective slot is empty and the yellow color indicates that the respective slot is already allocated to some other user.

5.3 PARKING BOOKING ONLINE FOR DATE AND TIME:

The administrator can add different locations where parking slots are available. The user can select any location which is nearest to his destination. The administrator can also delete the locations if he wishes. The administrator can view different locations where parking slots are available and can also check the status of different parking slots.



APJ Block

Date: Mar 27, 2020

Time: 3:40 am

How many hours ? 1hrs

You have to pay

₹ 0

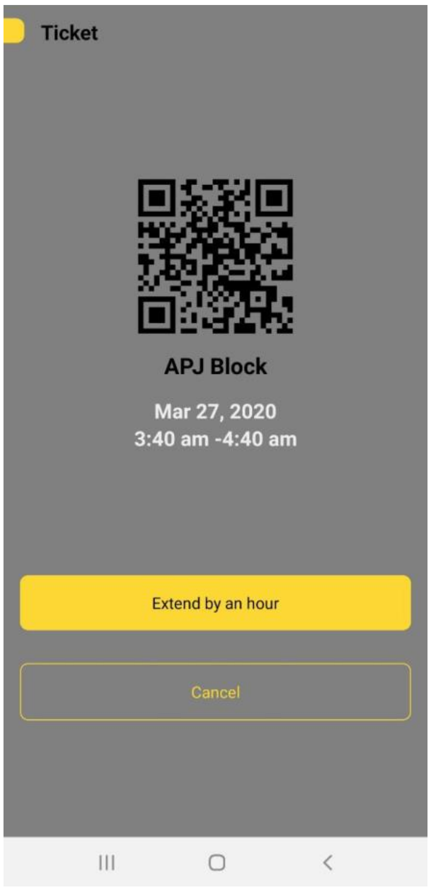
when you check-in.

Fig 8. Output screen for online parking booking

The administrator can view all the users who are using the application and can also check the booking details such as the time and date at which the user requires a slot, number of hours a user is using the allocated slot, at which location he requires a slot etc.

5.4 CONFIRMATION ON SUCCESSFUL BOOKING:

After completing the booking users are provided about the booking users are provided about the booking details which have a QR Code is sent to be scanned during check in and check out at the parking space.



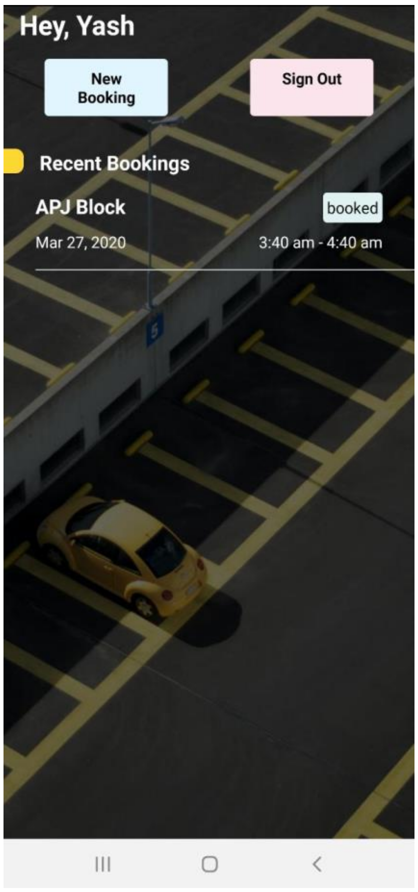


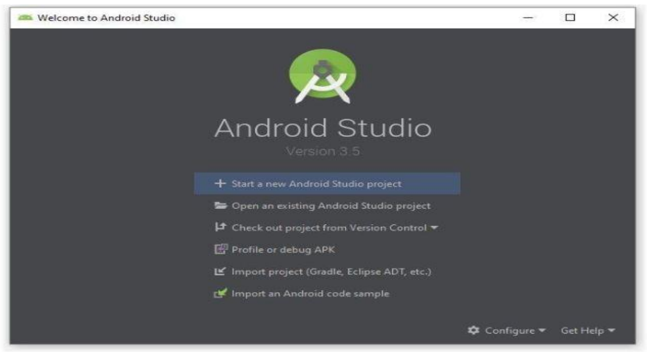
Fig 9. Output screen for successful booking

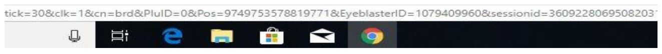
6. IMPLEMENTATION

6.1 ANDROID STUDIO

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA.

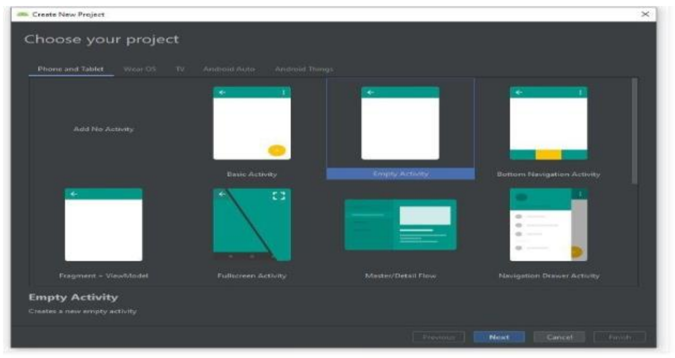
Step 1:


Welcome to Android Studio
Android Studio
Version 3.5
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Configure
Get Help


tick=30&clk=1&cn=brd&PluID=0&Pos=9749753578819771&EyeblasterID=1079409960&sessionid=3609228069508203

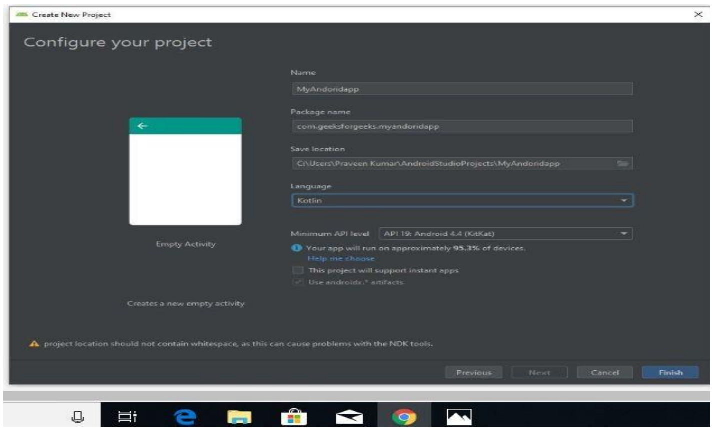
In Welcome to Android Studio screen, select Start a new Android Studio Project.

Step 2:



Select empty activity and click on the next button.

Step 3:



Create New Project

Configure your project

Name: MyAndoridapp

Package name: com.geeksforgeeks.myandoridapp

Save location: C:\Users\Praveen Kumar\AndroidStudioProjects\MyAndoridapp

Language: Kotlin

Minimum API level: API 19: Android 4.4 (KitKat)

Your app will run on approximately **95.3%** of devices.

Help me choose

This project will support instant apps

Use androidx.* artifacts

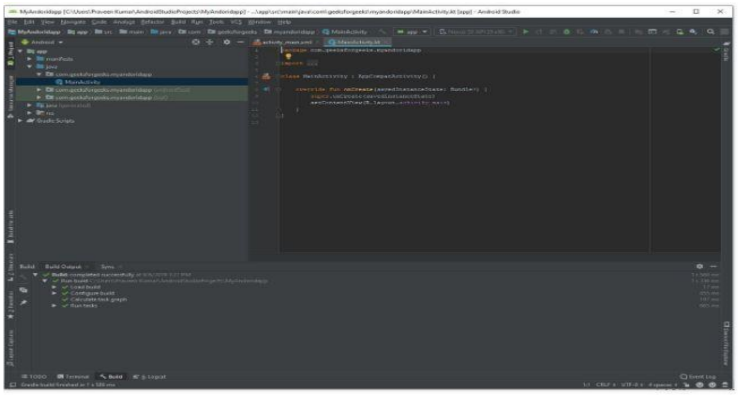
Empty Activity

Creates a new empty activity

project location should not contain whitespace, as this can cause problems with the NDK tools.

Write name of application and click on the finish button to launch the project.

Step 4:



MyAndoridapp [C:\Users\Praveen Kumar\AndroidStudioProjects\MyAndoridapp] - ...\app\src\main\java\com\geeksforgeeks\myandoridapp\MainActivity.kt [app] - Android Studio

```
package com.geeksforgeeks.myandoridapp

import ...

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

Newly created project opens with different number of files

6.2 CODE:

MainActivity.java:

```
Package
Com.parking;
import com.facebook.react.ReactActivity;
public class MainActivity extends ReactActivity {
  @Override
  protected String getMainComponentName() {
    return "parking";
  }
}
```

MainApplication.java:

```
Package
Com.parking;
import android.app.Application;
import android.content.Context;
import com.facebook.react.PackageList;
import com.facebook.react.ReactApplication;
import com.facebook.react.ReactNativeHost;
import com.facebook.react.ReactPackage;
import com.facebook.soloader.SoLoader;
import java.lang.reflect.InvocationTargetException;
import java.util.List;
public class MainApplication extends Application implements ReactApplication {
  private final ReactNativeHost mReactNativeHost =
      new ReactNativeHost(this) {
        @Override
        public boolean getUseDeveloperSupport() {
          return BuildConfig.DEBUG;
        }
        @Override
        protected List<ReactPackage> getPackages() {
          @SuppressWarnings("UnnecessaryLocalVariable")
          List<ReactPackage> packages = new PackageList(this).getPackages();
          // Packages that cannot be autolinked yet can be added manually here, for example:
          // packages.add(new MyReactNativePackage());
          return packages;
        }
        @Override
        protected String getJSMainModuleName() {
          return "index";
        }
      };
  @Override
  public ReactNativeHost getReactNativeHost() {
    return mReactNativeHost;
  }
  @Override
  public void onCreate() {
    super.onCreate();
    SoLoader.init(this, /* native exopackage */ false);
    initializeFlipper(this); // Remove this line if you don't want Flipper enabled
  }
  /**
   * Loads Flipper in React Native templates.
   *
   * @param context
   */
  private static void initializeFlipper(Context context) {
    if (BuildConfig.DEBUG) {
      try {
        /*
         We use reflection here to pick up the class that initializes Flipper,
        since Flipper library is not available in release mode
        */
        Class<?> aClass = Class.forName("com.facebook.flipper.ReactNativeFlipper");
        aClass.getMethod("initializeFlipper", Context.class).invoke(null, context);
      } catch (ClassNotFoundException e) {
        e.printStackTrace();
      } catch (NoSuchMethodException e) {
        e.printStackTrace();
      } catch (IllegalAccessException e) {
        e.printStackTrace();
      } catch (InvocationTargetException e) {
        e.printStackTrace();
      }
    }
  }
}
```

Home.js:

```
import React, {useState, useEffect} from 'react';
import {
  View,
  Text,
  StatusBar,
  SafeAreaView,
  ScrollView,
  StyleSheet,
  FlatList,
  TouchableNativeFeedback,
  TouchableOpacity,
} from 'react-native';
import Heading from '../components/heading';
import firestore from '@react-native-firebase/firestore';
import AsyncStorage from '@react-native-community/async-storage';
import {StackActions} from '@react-navigation/native';
import moment from 'moment';

function HomeScreen(props) {
  const {container, button, heading} = styles;
  const [transactions, setTransactions] = useState([]);

  const getUserData = async () => {
    //Get the collection of bookings and check if there are any bookings done by the current user
    const documentSnapshot = await firestore()
      .collection('bookings')
      .onSnapshot(function(querySnap) {
        const newt = [];
        querySnap.forEach(doc => {
          if (doc.data()['bookedBy'] == props.route.params.email) {
            newt.push({id: doc.id, ...doc.data()});
          }

        });
        setTransactions(newt);
      });

    // setUsers(documentSnapshot[0].data());

    // setUsers(documentSnapshot.docs);
  };

  useEffect(() => {
    getUserData();

  }, []);


  return (
    <SafeAreaView style={container}>
      <StatusBar backgroundColor={'#fff'} />
      <Text style={heading}>Hey, {props.route.params.name}</Text>
      <View
        style={{
          flexDirection: 'row',
          justifyContent: 'space-around',
          marginVertical: 16,
        }}>
        <TouchableNativeFeedback
          onPress={() => {
            props.navigation.navigate('parkingSpots', {
              email: props.route.params.email,
            });
          }}>
          <Text style={[button, {backgroundColor: '#E1F5FE'}]}>
            New Booking
          </Text>
        </TouchableNativeFeedback>
        <TouchableNativeFeedback
          onPress={async () => {
            await AsyncStorage.clear();
            props.navigation.navigate('login');
          }}>
          <Text style={[button, {backgroundColor: '#FCE4EC'}]}>Sign Out</Text>
        </TouchableNativeFeedback>
      </View>
      <Heading>Recent Bookings</Heading>
      <FlatList
        data={transactions}
        ListEmptyComponent={
          <Text style={{textAlign: 'center'}}>No Recent Bookings</Text>
        }
        renderItem={({item, index}) => {
          return (
            <View key={index}>
              <TouchableOpacity
                style={{marginHorizontal: 32}}
                onPress={() =>
                  props.navigation.navigate('ticket', {
                    ticket: item,
                  })
                }>
                <View
                  style={{
                    flexDirection: 'row',
                    justifyContent: 'space-between',
                    marginBottom: 8,
                  }}>
                  <Text style={{fontSize: 18, fontWeight: 'bold'}}>
                    {item.spot}
                  </Text>
                  <Text
                    style={{
                      padding: 4,
                      borderRadius: 4,
                      backgroundColor:
                        item.status == 'booked' ? '#E0F2F1' : '#FFEBEE',
                    }}>
                    {item.status}
                    </Text>
                  </View>
                  <View
                    style={{
                      flexDirection: 'row',
                      justifyContent: 'space-between',
                    }}>
                    <Text style={{fontSize: 14}}>
                      {moment(item.on).format('MMM DD, YYYY')}
                    </Text>
                    <Text style={{fontSize: 14}}>
                      {moment(item.from).format('h:mm a')} -{' '}
                      {moment(item.from)
                        .add(Number(item.for), 'hours')
                        .format('h:mm a')}
                    </Text>
                    {/* <Text>{item.duration}</Text> */}
                  </View>
                </TouchableOpacity>
                <View
                  style={{
                    height: 1,
                    backgroundColor: '#ddd',
                    width: '100%',
                    marginHorizontal: 32,
                    marginVertical: 16,
                  }}
                />
              </View>
            );
          }}
        />
      </SafeAreaView>
     );
}

const styles = StyleSheet.create({
  container: {
    backgroundColor: '#fff',
    flex: 1,
  },
  button: {
    paddingHorizontal: 16,
    paddingVertical: 8,
    backgroundColor: 'red',
    borderRadius: 4,
    width: '30%',
    textAlign: 'center',
    fontWeight: 'bold',
  },
  heading: {
    fontWeight: 'bold',
    marginHorizontal: 16,
    fontSize: 24,
  },
});

export default HomeScreen;
```

Booking.js:

```
import React, {useState, useEffect} from 'react';
import {
  View,
  Text,
  StyleSheet,
  SafeAreaView,
  ScrollView,
  TouchableOpacity,
  ActivityIndicator,
} from 'react-native';
import Heading from '../components/heading';
import firestore from '@react-native-firebase/firestore';
import {StackActions} from '@react-navigation/native';

const Booking = props => {
  const {container, spotName} = styles;
  const [spots, setSpots] = useState([]);
  const [loading, setLoading] = useState(false);

  const data = [...Array(20).keys()].map((y, index) => ({
    bookedBy: '',
    for: '',
    from: '',
    bookedAt: '',
    on: '',
    id: props.route.params.spotName + index,
  }));

  const getUserData = async () => {
    // await firestore()
    //   .collection('spots')
    //   .doc(props.route.params.spotName)
    //   .update({
    //     spots: data,
    //   });

    setLoading(true);
    const documentSnapshot = await firestore()
```

```
      .collection('spots')
      .doc(props.route.params.spotName)
      .get();

    setLoading(false);
    setSpots(documentSnapshot.data().spots);
    // setSpots(JSON.parse(documentSnapshot.data().spots));
    // setUsers(documentSnapshot[0].data());
    // documentSnapshot.forEach(doc => {
    //   setSpots([...spots, ...doc.data()]);
    // });
    // setUsers(documentSnapshot.docs);
  };

  useEffect(() => {
    getUserData();
  }, []);

  return (
```

ParkingSpots.js:

```
import React from 'react';
import {
  View,
  Text,
  StyleSheet,
  SafeAreaView,
  ScrollView,
  TouchableOpacity,
} from 'react-native';
import Heading from '../components/heading';

const ParkingSpots = props => {
  const {container, spotName} = styles;
  return (
    <SafeAreaView style={container}>
      <ScrollView style={{flex: 1}}>
        <Heading>Free</Heading>
        <TouchableOpacity
          onPress={() =>
            props.navigation.navigate('booking', {
              spotName: 'apj',
              spot: 'APJ Block',
              email: props.route.params.email,
              price: 0,
            })
          }>
          <Text style={spotName}>APJ Block</Text>
        </TouchableOpacity>
        <Heading>Paid</Heading>
        <TouchableOpacity
          onPress={() =>
            props.navigation.navigate('booking', {
              spotName: 'opal',
              spot: 'Opal Block',
              email: props.route.params.email,
              price: 20,
            })
          }>
          <Text style={spotName}>Opal Block</Text>
        </TouchableOpacity>
        <TouchableOpacity
          onPress={() =>
            props.navigation.navigate('booking', {
              spotName: 'pearl',
              spot: 'Pearl Block',
              email: props.route.params.email,
              price: 20,
            })
          }>
          <Text style={spotName}>Pearl Block</Text>
        </TouchableOpacity>
        {/* <TouchableOpacity
          onPress={() =>
            props.navigation.navigate('booking', {
              spotName: 'apj',
              spot: 'APJ Block',
              email: props.route.params.email,
            })
          }>
          <Text style={spotName}>APJ Block</Text>
        </TouchableOpacity> */}
      </ScrollView>
    </SafeAreaView>
  );
};
```

SignUp.js:

```
import React, {useState, useEffect} from 'react';
import {
  SafeAreaView,
  View,
  Text,
  TextInput,
  StyleSheet,
  TouchableOpacity,
  ActivityIndicator,
} from 'react-native';
import firestore from '@react-native-firebase/firestore';
import AsyncStorage from '@react-native-community/async-storage';
import Snackbar from 'react-native-snackbar';

const Signup = props => {
  const [isLoading, setIsLoading] = useState(false);
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [name, setName] = useState('');
  const [error, seterror] = useState('');
  const {container, borderContainer} = styles;
const onSIgninPressed = async () => {
    setIsLoading(true);

    // when submit is pressed add the email, first name and password to the collection
    const documentSnapshot = await firestore()
      .collection('users')
      .add({
        name,
        email,
        password,
      });

    // After adding to the collection navigate to login screen for the user to login
    if (documentSnapshot) {
      Snackbar.show({
        text: 'Account created successfully. Log In',
        duration: Snackbar.LENGTH_SHORT,
      });
      props.navigation.navigate('login');
    } else {
      seterror('No Account exists with the email');
    }
    // setPasswoinrd('No registered email');
    setIsLoading(false);
  };
 Return;
```

7. TESTING

In this section, all the functionalities of the application are tested under various scenarios. This application is mainly tested using phone with Android Studio

7.1 LOGIN VALIDATION

The login page is the first page the user sees when they open the application. The user is asked to provide the username and password. They should provide correct credentials. If the correct credentials are not provided, a user cannot access the application.

7.2 BOOKING FROM AN ACCOUNT BEFORE THE EXPIRATION TIME

The user can select a slot in their desired parking lot. After this selection, the user is asked to scan the generated Before this expiration. They also cannot book another slot before the expiration time.

7.3 IS THE BOOKING REFLECTING ON THE MAIN SCREEN?

The main screen of the application has all the parking lots on campus and corresponding to it is the number of slots available at that moment. After the selection of the desired slot from a parking lot, the main screen shows a decrease in the available slots

7.4 CANCELING A BOOKING

When the user cancels their booking, the parking slot is de-allocated. Later shows an increase in parking slots, after the cancellation of the previously selected slot booking.

8. CONCLUSION AND FUTURE SCOPE

8.1 CONCLUSION

Online Parking Slot Booking System improves the existing system since we are in computerized world. With this new system is mandatory, it enables the user of the system to reserve a parking lot online and this reduces the wasting of time of the clients looking for where to park, increase the safety of the property since the parking lot is numbering.

8.2 FUTURE SCOPE

The following functions will help improve the application:

- The parking history collected in this application, can be used to make statistics and can notify to the user, that their usual parking lot has free slots at that moment.
- The application can provide an option to certain professors or employees to select their own reserved parking.
- Provide-ability for user to link their account with islander account. By doing so they can access their sail account using this platform and buy their parking pass for the semester.
- A map with directions can be provided to the user from the entrance to the parking slot.

REFRENCES:

[1] https://www.ijraset.com/fileserve.php?FID=6482

[2] Andre Barroso, Jonathan Benson, Tina Murphy, UtzRoedig, Cormac Sreenan, John Barton, Stephen Bellis, Brendan Flynn, and Kieran Delaney, “The DSYS25 SensorPlatform,” t Ireland,2004

[3] B. Yan Zhong, S. Li Min, Z. Hong Song, Y. Ting Xin, and L. Zheng Jun, “A Parking Management System Based on Wireless Sensor Network,” tech. rep., Institute of Software Graduate School of Chinese Academy of Sciences, Beijing, November 2006

[4] Itziar Marin, Eduardo Arceredillo, AitzolZuloaga, and Jagoba Arias, “Wireless Sensor Networks: A Survey on Ultra-Low Power-Aware Design,” tech. rep., World Academy of Science, Engineering and Technology, August 2005.

[6] Vanessa W.S Tang, Yuan Zheng, and JiannongCao,“An Intelligent Car Park Management System based on Wireless Sensor Networks,” tech. rep., Internet and Mobile Computing Lab, Department of Computing, The Hong Kong Polytechnic University, P.R.China, August 2006.

[7] ZigBee/IEEE 802.15.4 Summary, “SinemColeriErgen,”tech. rep., EECS Berkely, September 2004.ech. rep., University College, Cork, Irelandand National Microelectronic Research.