**Software Requirements Specification**

**for**

Park AnyWhere

**Version 1.0 approved**

**Prepared by <author>**

**<organization>**

**<date created>**

**Table of Contents**

[**1. Introduction 1**](#_heading=h.1fob9te)

[1.1 Purpose 1](#_heading=h.3znysh7)

[1.2 Document Conventions 1](#_heading=)

[1.3 Intended Audience and Reading Suggestions 1](#_heading=)

[1.4 Product Scope 2](#_heading=)

[**2. Overall Description 2**](#_heading=)

[2.1 Product Perspective 2](#_heading=h.2s8eyo1)

[2.2 Product Functions 2](#_heading=)

[2.3 User Classes and Characteristics 2](#_heading=)

[2.4 Operating Environment 2](#_heading=)

[2.5 Design and Implementation Constraints 2](#_heading=)

[2.6 User Documentation 2](#_heading=)

[2.7 Assumptions and Dependencies 3](#_heading=)

[**3. External Interface Requirements 3**](#_heading=)

[3.1 User Interfaces 3](#_heading=h.2jxsxqh)

[3.2 Hardware Interfaces 4](#_heading=)

[3.3 Software Interfaces 4](#_heading=)

[3.4 Communications Interfaces 4](#_heading=)

[**4. System Features 4**](#_heading=)

[4.1 Create Account 4](#_heading=)

[4.2 Log In 5](#_heading=h.7pl9c1shzpfk)

[4.3 View Profile 5](#_heading=h.br1g3lz9o2xd)

[4.4 Search Location for Car Parks 5](#_heading=h.hmveta8nohdn)

[4.5 Calculate Route 6](#_heading=h.mid4iy9hiq8i)

[4.6 Send Feedback 6](#_heading=h.bg5yh3c5kjvv)

[4.7 System Feature X 6](#_heading=)

[**5. Other Non-Functional Requirements 8**](#_heading=h.3whwml4)

[5.1 Performance Requirements 8](#_heading=h.2bn6wsx)

[5.2 Safety Requirements 8](#_heading=)

[5.3 Security Requirements 8](#_heading=)

[5.4 Software Quality Attributes 8](#_heading=)

[5.5 Business Rules 9](#_heading=)

[**6. Other Requirements 9**](#_heading=)

**Revision History**

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

ParkAnyWhere v1 is a carpark application that aids users in locating suitable car parks near their destination. The application conveniently provides information on the car park for users, such as its rates, lot availability and more. The app allows users to give ratings and feedback on car parks, and gives a comprehensive experience by also noting down the user’s history.

This SRS will describe the application and its system.

## Document Conventions

This document is generally written in font size 11, Times New Roman.

## Intended Audience and Reading Suggestions

This application is a prototype for a car park searching system and it is restricted within Singapore. This document is useful for the app development team, users as well as testers.

## Product Scope

Car parking is an everyday activity for those who drive. This software system being developed automates searching for a suitable rate, at the preferences of the drivers. It allows drivers to share feedback on car parks, and records the driver’s parking history.

# Overall Description

## Product Perspective

ParkAnywhere uses information from the database provided by the Housing Development Board (HDB) as well as data from Google Maps. Firebase was also used for user authentication, as well as the storing of databases such as Customer information and Feedback.

## Product Functions

*Product will be able to*

2.2.1 Display and filter nearby car parks for user to choose

2.2.2 Display route from current location to car park

2.2.3 Allow users to provide feedback for car parks as well as leave a rating

2.2.4 Display user profile information such as parking history and total spent

## User Classes and Characteristics

ParkAnywhere expects general users as a user class. General Users would be frequent users, and only basic technical expertise is expected to be required. Users must also be in access to WiFi networks as a requirement.

## Operating Environment

ParkAnywhere is based on React Native and is coded in a way where ParkAnywhere will operate on any android phone or android simulator that has access to a WiFi connection.

## Design and Implementation Constraints

ParkAnywhere deploys data from various databases and thus would have to be updated if there are major changes to how these databases can be fetched. Future developer working on this app will have to use their own user authentication, as the Firebase user base is not shared publicly.

## User Documentation

Since the functions in ParkAnywhere are currently simple and straight forward, no user manuals are prepared but if the situation arises, an in-app user manual would be implemented in the MainPage.

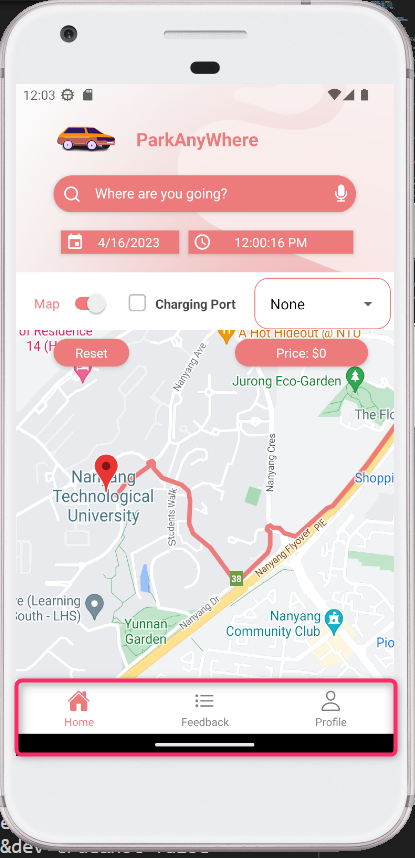
## Assumptions and Dependencies

ParkAnywhere will continue to have the functions operational as long as HDB continues to have their carpark data be publicly accessable. This is the same for the Google Maps data. Other than that, there is also assumptions that future android updates would continue to support the current implementation of ParkAnywhere.

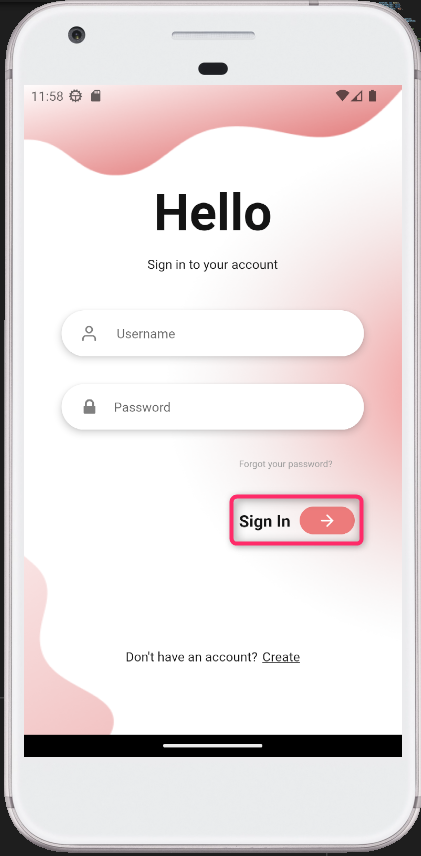
# External Interface Requirements

## User Interfaces

* + 1. Clear tools to assist the user with navigating between screens in the app.
       1. Navigation bar at the bottom of the screen for easier navigation between screens.



* + - 1. Buttons clearly placed and designed with intuitive intention for navigation purposes.



## Hardware Interfaces

Not applicable to this project.

## Software Interfaces

* + 1. For ParkAnywhere to operate, it depends on the following external software components
       1. Firebase is used as the ParkAnywhere’s database to store user data, parking history, and authentication when logging into the app.
       2. Google Maps API is used to show the map view in ParkAnywhere’s main page and for calculating the best route from current location to user’s chosen destination.
       3. Carpark data used in ParkAnywhere is collected from data.gov.sg and HDB website.
    2. The app is developed and run on React Native and requires an Android Device or emulator to run.

## Communications Interfaces

3.4.1 For ParkAnywhere to operate, it depends on the following communication components

3.4.1.1 Since Firebase Authentication is used as the means for user sign up, users are required to have an operational email address as well as access to it

3.4.1.2 ParkAnywhere must also be able to access the internet in order to draw data from FireBase and Google Maps API.

3.4.1.2.1 Firebase uses a cloud messaging communication method

3.4.1.2.2 Google Maps uses an Application Programming Interface (API) communication method.

# System Features

*<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>*

## Create Account

* + 1. **Description and Priority**
       1. Allows the user to create an account to use the rest of the app. High priority.
    2. **Stimulus/Response Sequences**
       1. The user will need to tap on “Don’t have an account?”, provide a username, email and password. App will either successfully create an account or return an error.
    3. **Functional Requirements**
       1. User must be able to create an account
          1. Account must have a unique ID.
          2. Account must have an unique email address used to log in.

User attempting to register with an existing email address must be denied creation.

* + - * 1. Account must have a username used to identify other users in the app.

## Log In

* + 1. **Description and Priority**

Allows the user to log into the app with a created account. High priority.

* + 1. **Stimulus/Response Sequences**

The user will need to key in a valid email and password and tap on “Sign in”. The app will either bring the user to the map screen after logging in or there will be an error displayed due to invalid email or password.

* + 1. **Functional Requirements**
       1. User must be able to login with a created account.
          1. User must be denied access if either email or password is incorrect.

## View Profile

* + 1. **Description and Priority**

Allows the user to track various statistics like expenditure, bookings and reviews. Medium priority.

* + 1. **Stimulus/Response Sequences**

The user can scroll the booking history or change to the Feedback or Map screen.

* + 1. **Functional Requirements**
       1. User must be able to see account’s username and display icon.
       2. User must be able to view account’s parking history.
          1. Each car park in the booking history must display its name.
          2. Each car park in the booking history must display expenditure and time spent.
       3. User must be able to view account’s total expenditure.
       4. User must be able to view total number of bookings made.
       5. User must be able to view account’s number of times feedback was given.

## Search Location for Car Parks

* + 1. **Description and Priority**

Allows the user to find a car park that suits their needs near their destination. Medium priority.

* + 1. **Stimulus/Response Sequences**

The user can toggle between showing appropriate car parks on the map or in a list. The user can then indicate preference for an indoor or outdoor car park, or a car park with electric vehicle charging stations. The user can scroll the map to find a car park manually, or use the search bar to find car parks around a destination. The app will display suitable car parks by name, along with details such as parking rate.

* + 1. **Functional Requirements**
       1. User must be able to search around a given location for nearby car parks.
          1. User can toggle between a map view or a list of suitable car parks.
          2. User can toggle preferences to filter for specific features in a car park.

Users can choose to filter for an indoors or outdoors car park.

Users can choose to filter for a car park with electric vehicle charging stations.

* + - * 1. If no nearby car parks can be found matching, an error will be displayed.

## Calculate Route

* + 1. **Description and Priority**

The user can select a car park and the app will map out a route for the user to follow to the destination. Medium priority.

* + 1. **Stimulus/Response Sequences**

On launch, the user allows the app access to location services and turns on Location on their mobile device. The user selects a car park and the app displays additional details on the car park. The user can tap again and lock the car park in as a destination. The app will display a route navigating roads to the destination.

* + 1. **Functional Requirements**
       1. User can calculate a route from current location to a selected car park.
          1. If user does not allow app access to location, route is calculated from user’s last known location

## Send Feedback

* + 1. **Description and Priority**

The user can select a previously booked car park and leave a star rating and a comment. Medium priority.

* + 1. **Stimulus/Response Sequences**

The user selects a car park from the drop-down menu of previously booked car parks. The user then taps or drags along the star rating to leave a rating from 0 to 5. The user then leaves a comment for the selected car park in the text box. After tapping submit, the feedback form is cleared and the feedback is uploaded to the database.

* + 1. **Functional Requirements**
       1. User must be able to send feedback for any car park in booking history.
          1. User must be able to input a rating from 0 to 5 and a comment for the selected car park.

If any of the input fields are empty, feedback will not be uploaded and will be voided.

* + - * 1. If user has no car park in booking history, “No parking history found.” will be displayed.

# Other Non-Functional Requirements

## Performance Requirements

* + 1. ParkAnywhere requires a stable Internet Connection in order for app to run.
    2. ParkAnywhere requires users’ permission to access their real-time location while using the app.
    3. ParkAnywhere must be able to reflect real-time changes in location within 2 seconds in order to reflect accurate information on the map.

## Safety Requirements

* + 1. Users must exercise discretion and attention if driving while using ParkAnywhere.

* + 1. ParkAnywhere will not be held responsible for any road-related incidents occured due to user’s divided attention when using our app while driving.

## Security Requirements

* + 1. The database of user passwords must not be accessible by any personnel, external hackers or other users.

* + - 1. ParkAnywhere uses Firebase Authentication to verify user login details.

* + - 1. Firebase Authentication uses an internally modified version of scrypt to hash account passwords

## Software Quality Attributes

* + 1. Usability - Help messages must be displayed in the local language according to the user’s locale
    2. Reliability - Upon system restart, ParkAnywhere’s functionality must be restored within 5 minutes.
    3. Correctness - ParkAnywhere must accurately load up details for the specific user within 15 seconds of logging into the app.
    4. Portability - ParkAnywhere must be able to run on different Operating Systems (Android OS, Apple iOS) on multiple versions.
    5. Reliability - ParkAnywhere must load up the correct details for the specific user 95% of the time.

## Business Rules

There are no restrictions to accessibility or features within the app.

# Other Requirements

* 1. Database Requirements
     1. Database needs to be able to handle multiple hundreds of queries per day to meet the demand from customers based on the app.
     2. Database needs to provide simple data validation to ensure customer inputs are valid before being added to the database.
  2. Government Statistics Website Requirements
     1. The statistics provided by Singapore Government regarding carparks needs to be accurate and routinely updated, since ParkAnywhere gets its data from there.

**Appendix A: Glossary**

*<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>*

**Appendix B: Analysis Models**

*<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams*.>

**Appendix C: To Be Determined List**

*<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>*

Source: http://www.frontiernet.net/~kwiegers/process\_assets/srs\_template.doc