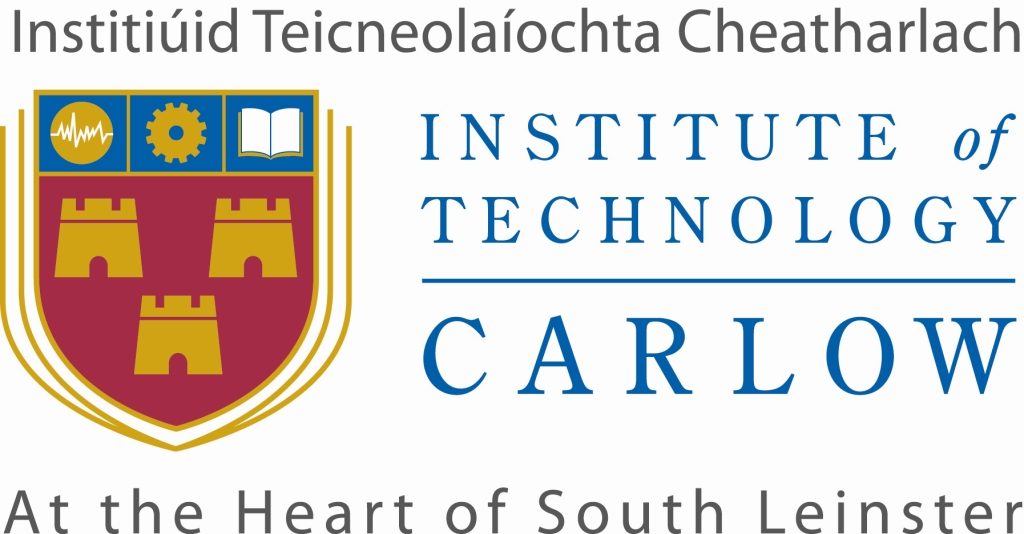
CAIRDE

**Car Sharing App**

Design Document



18th April 2018

BSc (Hons) Software Development

**Name:** Dylan Lawlor

**Year:** 4th year

**Student ID:** C00197013

**Supervisor:** Lei Shi

# Abstract

The purpose of this document is to provide the design specification for the project, including system architecture, database design, use cases and UI design prototypes.

Contents

[Abstract 1](#_Toc511737747)

[System Architecture 3](#_Toc511737748)

[Database Design 4](#_Toc511737749)

[Table design 5](#_Toc511737750)

[Users 5](#_Toc511737751)

[UserProfiles 5](#_Toc511737752)

[Journeys 5](#_Toc511737753)

[Use Cases 7](#_Toc511737754)

[Use Case Diagram 7](#_Toc511737755)

[Detailed Use Cases 7](#_Toc511737756)

[Login Use Case 7](#_Toc511737757)

[Register Use Case 8](#_Toc511737758)

[Set up Profile Use Case 9](#_Toc511737759)

[Create a Journey Use Case 9](#_Toc511737760)

[Search for Journey Use Case 9](#_Toc511737761)

[View Journeys Use Case 10](#_Toc511737762)

[Accept Applicant Use Case 10](#_Toc511737763)

[Design 11](#_Toc511737764)

[UI Mock-ups 11](#_Toc511737765)

[Login Page 11](#_Toc511737766)

[Registration Page 12](#_Toc511737767)

[Set up Profile Page 12](#_Toc511737768)

[Main Menu 13](#_Toc511737769)

[Create Journey 13](#_Toc511737770)

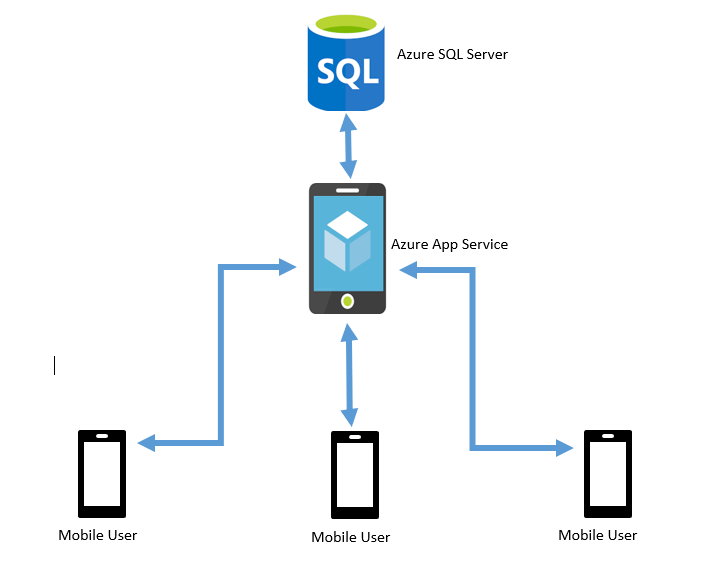
[Search Journey 14](#_Toc511737771)

[View Journey 14](#_Toc511737772)

[UI Flow Diagram 15](#_Toc511737773)

# System Architecture

The system layout for this application is as follows:

* Multiple Mobile Users
* Online cloud hosted App Service on Azure
* Azure hosted SQL database

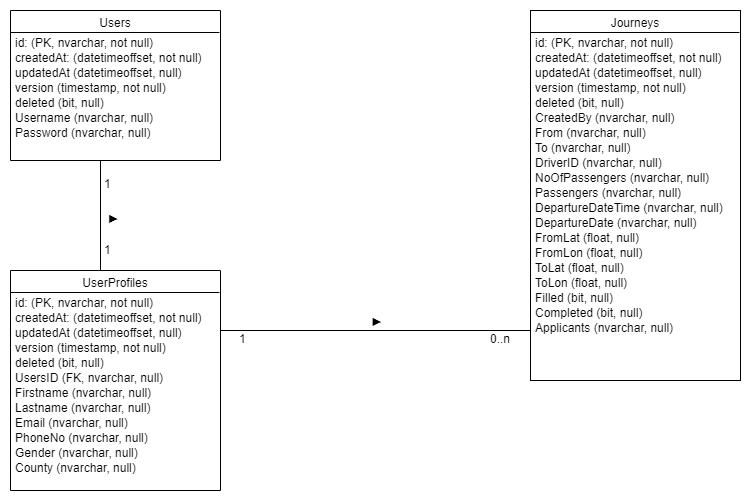
This system design gives the advantage of separating access to the database from the mobile devices, putting a layer of security between the outside world and the users hosted data. Only the App Service can communicate with the database and can only do so with the pre-compiled functionality of the application, which will protect from any malicious attempts to gain access to the data. It also allows the app service to handle traffic from users so at no point the database can be overwhelmed.

# 

# Database Design

Given that it will be the Azure App service communicating with the database, this leads to an unusual database design, since the App service allows for a flexible approach to table creation and insertion, most similar to MongoDB, in that a Table name is specified, but the contents of that table need not be. The contents are derived from the object being posted into the database from the application, through the app service (See system architecture above) and changes to the object or missing values do not necessarily corrupt the table. Also the App service provides a unique ID field for every entry, which can be used to update that recoed at a later point, or as a foreign key in another table

With the above in mind here is the database design for this application:



This design looks very lightweight with only three tables but contains all the fields necessary to power the application. The aim is to connect users, not gather mountains of data. The Journeys table will contain the bulk of the information required by the app. Most fields are strings (nvarchar) due to the method of posting c# classes to the database which does not always contain a matching variable type.

## Table design

In this section the 3 tables function will be outlined, as well as a brief description of each field.

### Users

This table is used to validate a users login, and allow a user to register. The id field is used as a foreign key in other tables to identify this user account.

* id: unique id generated by web app
* createdAt: date record created
* updatedAt: date record is updated (change of password etc)
* version:
* deleted: Boolean for if the record is flagged for deletion
* Username: Users chosen username, unique to the system, used to login
* Password: Users chosen password, in encrypted form. Password is encrypted by the app prior to being posted to the database. The password is never in readable form during any network transaction.

### UserProfiles

This table contains personal information about the user required by the app. A user account is associated to one user profile through the UserID field.

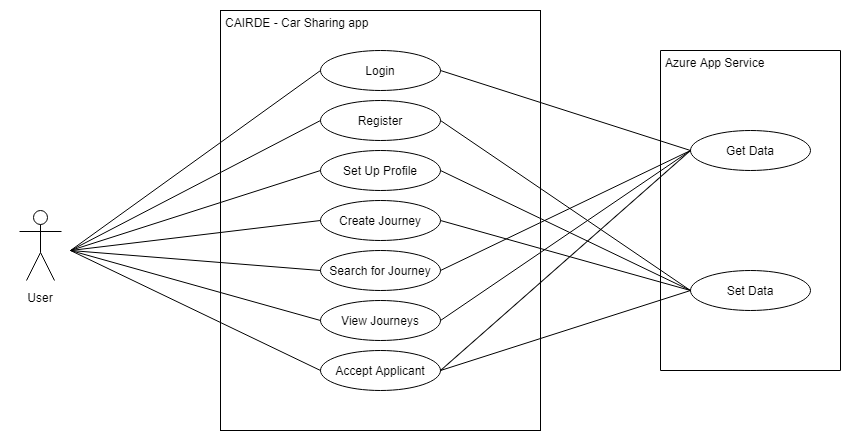
* id: unique id generated by web app
* createdAt: date record created
* updatedAt: date record is updated (change of password etc)
* version:
* deleted: Boolean for if the record is flagged for deletion
* UsersID: id from Users table to associate this profile with that account
* Firstname: Users first name
* Lastname: Users last name
* Email: Users email address used to receive notifications
* PhoneNo: Users mobile phone number, to be used for optional messaging beween users on same journey
* Gender: Users gender, optional
* County: User base county, for location usage

### Journeys

* id: unique id generated by web app
* createdAt: date record created
* updatedAt: date record is updated (change of password etc)
* version:
* deleted: Boolean for if the record is flagged for deletion
* CreatedBy: User ID of the user who created the Journey
* From: Address of the departure location
* To: Address of the destination
* DriverID: User ID of the user who will be driving the car on this journey
* NoOfPassengers: The max number of seats available on this journey (1-4)
* Passengers: A comma-delimited list containing the user ID’s of the users who have been accepted as passengers for this journey
* DepartureDateTime: The time of the day this journey will begin
* DepartureDate: The date this journey will take place on
* FromLat: Latitude of the departure location
* FromLon: Longitude of the departure location
* ToLat: Latitude of the destination
* ToLon: Longitude of the destination
* Filled: Boolean to indicated if this journey is at max passenger limit
* Completed: Boolean to indicate this journey is completed
* Applicants: A comma-delimited list containing the user ID’s of the users who have applied to be passengers for this journey

# Use Cases

## Use Case Diagram



## Detailed Use Cases

This section will look at the brief use cases from the functional specification document and turn them into detailed use cases. First a brief description of terms:

* User – A user of the Mobile app CAIRDE
* Azure – The cloud backend, Azure App Services and SQL Database
* App – The CAIRDE Mobile App

### Login Use Case

**Actors**: User, Azure

**Brief description**: This use case begins when the user starts the app. The user is presented with a form prompting for two inputs, username and password. The user enters their details and clicks a button to login. The app communicates with Azure to determine the details provided are correct, and when successful the user is logged in.

**Main success scenario:**

1. The user starts up the app
2. The app displays the login screen
3. The user enters their username and password
4. The user clicks login
5. The app validates the inputted details with Azure
6. The user is logged in

**Alternatives**

4A. Details entered are of incorrect

1. App displays an error message
2. Step 2 begins again

### Register Use Case

**Actors:** User, Azure

**Brief description:** This use case begins when the user wants to register with the app. The user is presented with the register screen and is asked to choose a username and password, and confirm the password. The user submits the details and the app sends the data to Azure which stores the information. The user is now registered.

**Main success scenario:**

1. The user starts up the app
2. The app displays login page
3. The user chooses to register by clicking register
4. The app displays the register screen
5. The user fills in the required details and clicks Register to submit their details
6. The app verifies that both passwords match, and that the username is available
7. The app submits the data to Azure which stores the information.
8. The user is registered.

**Alternatives**

6A. The username is already taken

1. The app displays a message to the user that the username is not available
2. Step 4 begins

6B. The passwords entered do not match

1. The app displays a message to the user stating the passwords don’t match
2. Step 4 begins

### 

### Set up Profile Use Case

**Actors:** User, Azure

**Brief description:**  This use case begins the first time the user logs into the App. The user is presented with a screen requiring several details about the user to be filled in. The user fills in the details and clicks Set up Profile. The app sends the data to Azure where it is stored. The user is show the home screen.

**Main success scenario:**

1. The user successfully logs into the app for the first time
2. The app checks Azure to see if the user ID has a profile set up
3. Since there is no profile set up the user is shown the set up profile screen
4. The User fills in their details and clicks submit.
5. The app sends the details to Azure where they are stored
6. The app shows the home menu.

### 

### Create a Journey Use Case

**Actors:** User, Azure

**Brief description:** This use case begins when the user selects the Create Journey option from the home menu. The user is shown a screen where they can set a from and to location, as well as other details of the journey, and then submit the journey, which will be stored on Azure.

**Main success scenario:**

1. The user selects create a journey from the home screen
2. The app prompts the user for details regarding the lift – location, destination, depart date, depart time, how many spaces are available
3. The sends the data to Azure
4. Azure stores the information
5. The user is returned to the home screen

### Search for Journey Use Case

**Actors:** User, Azure

**Brief description:** This use case begins when the user clicks the search journey button on the home screen. The user is taken to a screen where they can enter in details about the journey they are looking for: location to and from, date and time and range within which to search. The user submits the search and the app requests matching journeys from Azure. The user is presented with a list of matching journeys.

**Main success scenario:**

1. The user selects search journey from the home screen
2. The app displays a screen where the user can fill in details about the journey
3. The app sends this information to Azure to find matching journeys
4. Azure returns a list of Journeys to the app that match the specifications
5. The app displays a list of matching journeys to the user

**Alternatives:**

4A. No matches are found

1. The app displays a message to the user that there were no such journeys found
2. Step 2 Begins

### View Journeys Use Case

**Actors:** User, Azure

**Brief description:** This use case begins when the user selects one of the 4 options to view journeys (Driver upcoming, Driver old, Passenger Upcoming, Passenger Old). The App sends the request along with the users ID to Azure, and Azure returns a list of journeys that match the query.

**Main success scenario:**

1. The user selects one of the view journey buttons on the home screen
2. The app queries Azure for matching journeys for this user
3. Azure returns a list of matching journeys which the app displays

### 

### Accept Applicant Use Case

**Actors:** Azure, User

**Brief description:** This use case begins when a user receives a notification from the app that another user has applied for a lift on one of this users journeys. The user logs into his app and checks the journey. The user clicks Check Applicants and a list of applicants is displayed. The user clicks on an applicants name and selects accept applicant from the popup. The user is removed from applicants and added as a passenger to the journey, and a notification is sent to the passenger. This change is stored on Azure.

**Main success scenario:**

1. The user checks their upcoming driver journeys
2. A journey shows up as having new applicants
3. The user views this journey, and clicks check applicants
4. The user selects the applicants name and clicks accept applicant
5. The applicant is added to the journey as a passenger
6. The passenger receives a notification informing them of their acceptance

# Design

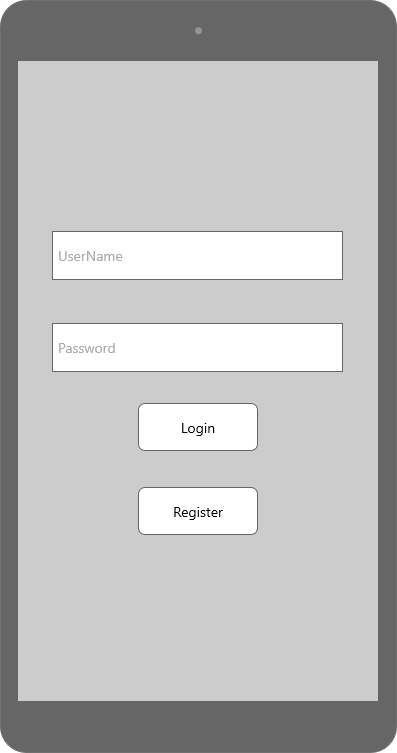
The aim for this app is to keep the UI simple and clean to make it as easy and intuitive for the user to follow the layout. Bright colours will be used only to highlight certain features, with the majority of the colour palette following a dark theme.

## UI Mock-ups

The following are several mock-ups for various parts of the app. They are not intended to be indicative of the final version, but more of a guideline for the design.

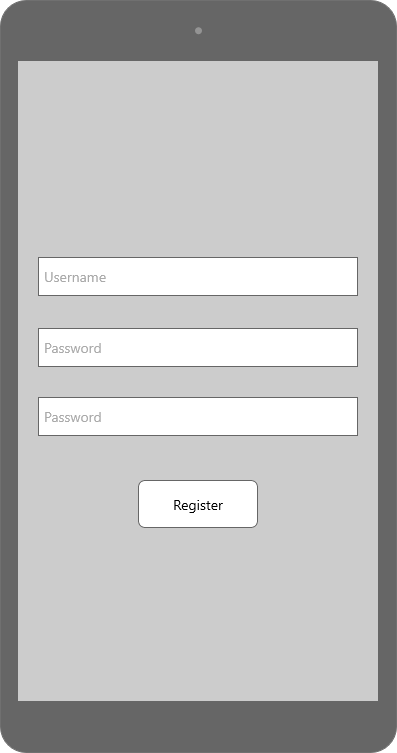
### Login Page

The login page is also the landing page for the app.



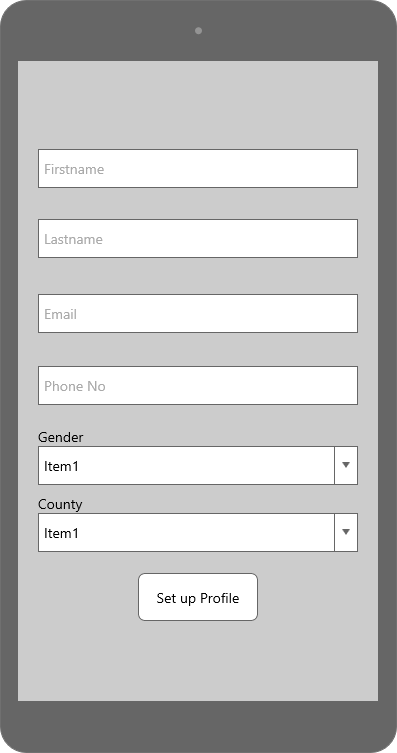
### Registration Page

This is the page where users will register for the app. It requires little information so that the process is quick.



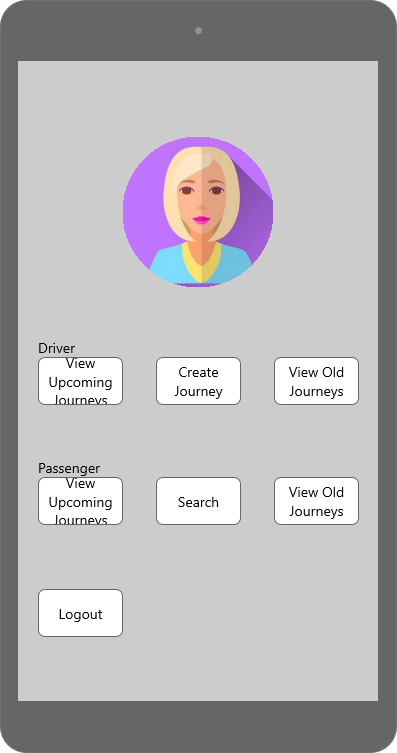
### Set up Profile Page

This page is accessed the first time a user log into the app, allowing the user to set up their account.



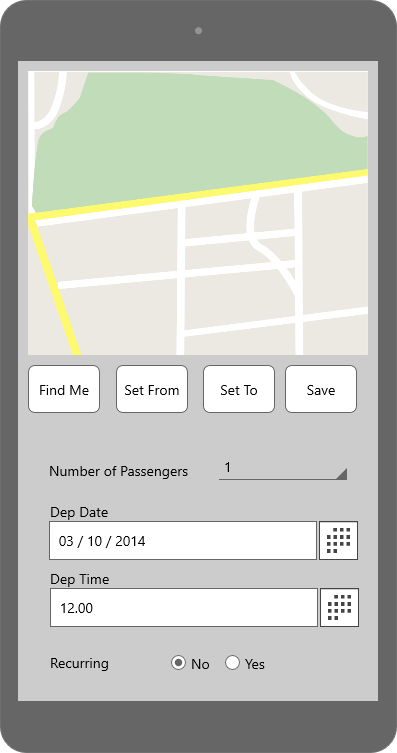
### Main Menu

This is the main page of the app. It will be the first page a user is shown after they set up profile, and every time they successfully log-in after that. This page contains the controls to reach the rest of the app, with all options just one button press away.



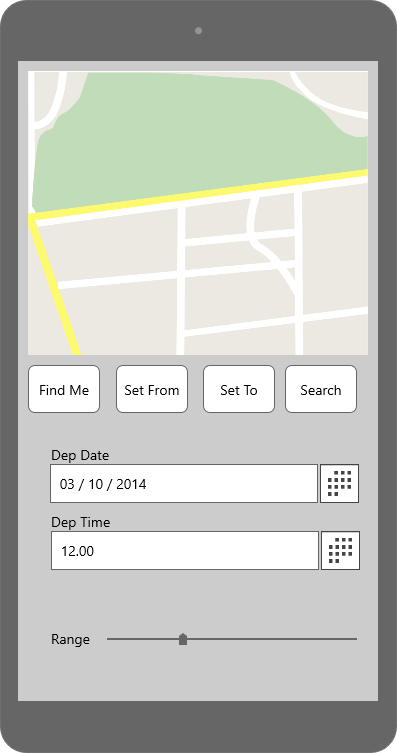
### Create Journey

This page allows a user to create a journey as a Driver, making that journey available to other passengers who wish to go a similar route.



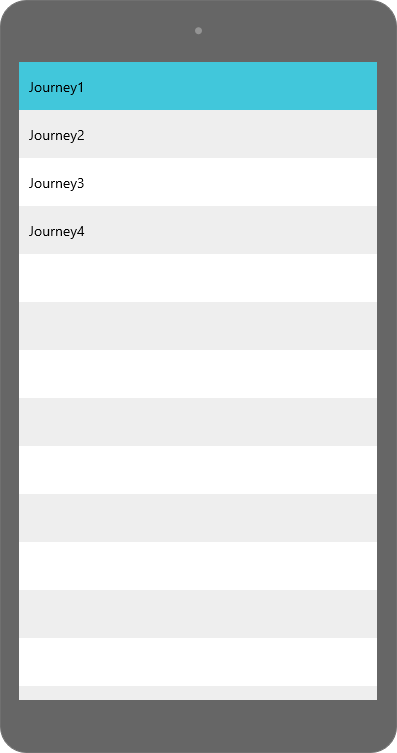
### Search Journey

This page is similar to the create journey page, but allows a user to search for a journey as a passenger, using the range slider to set the area for the search to accept.



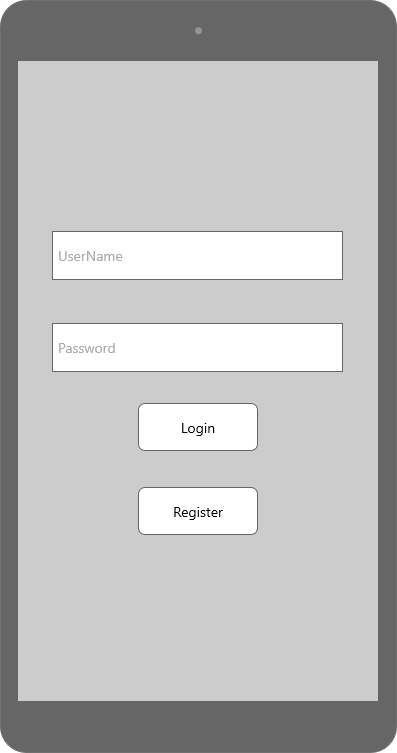
### View Journey

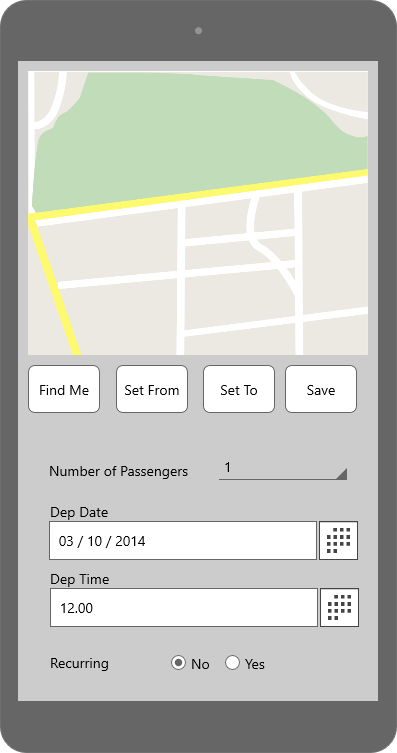
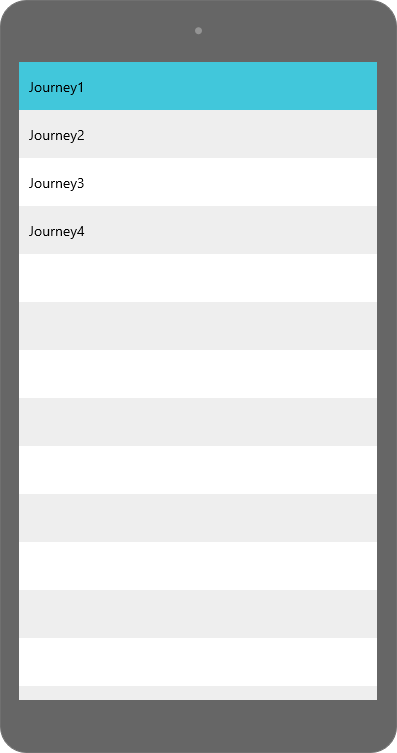
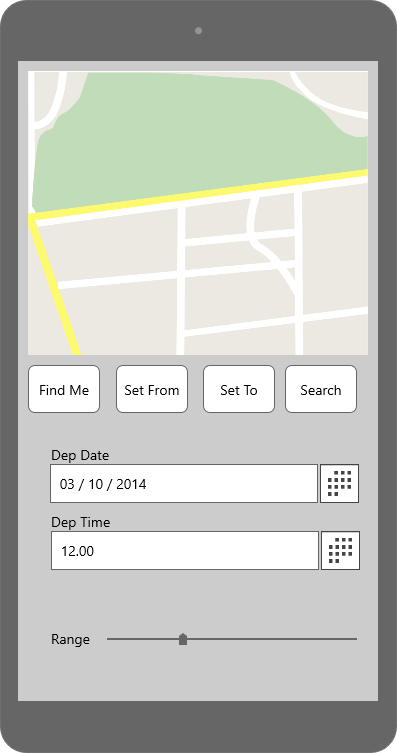
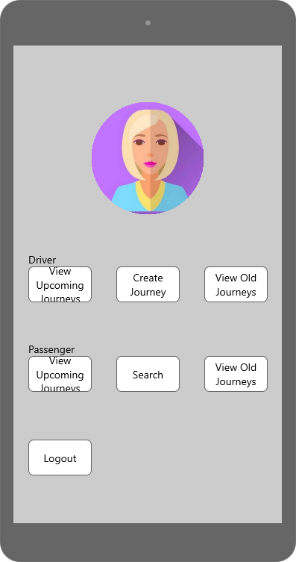
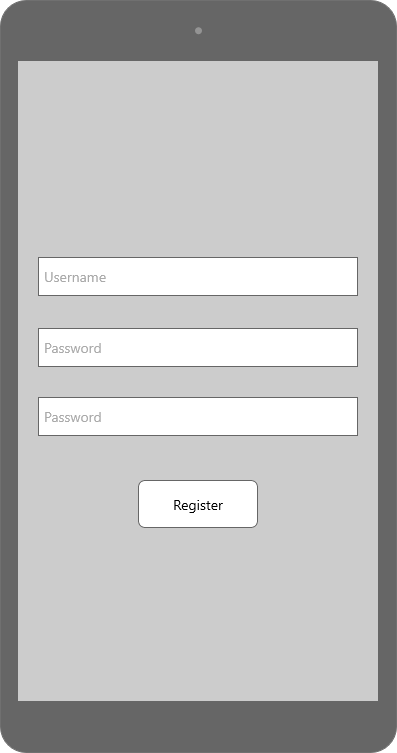
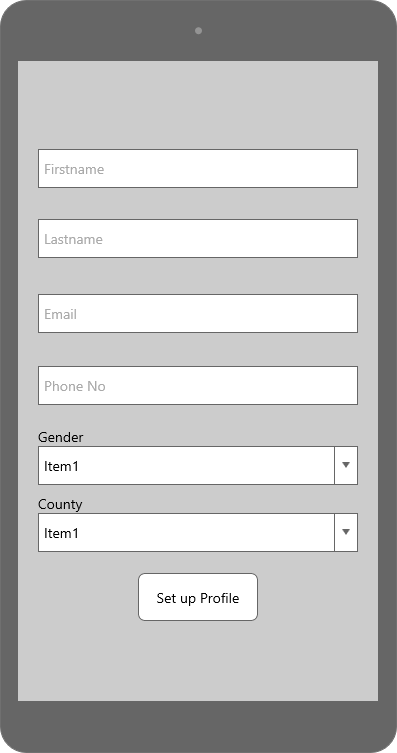
This is the page that is displayed when the user selects one of the View Journey buttons. It will display a list of the journeys that match the control, ie Upcoming Journeys in which the user is a driver etc.

.

# UI Flow Diagram

Below is a picture of the potential flow of the main activities of the app.





First Time