By: Kevin Dawson

Supervisor: Joseph Kehoe



Design Document

Abstract

The purpose of this document is to provide the design specification for the project, this includes the system architecture, database design, detailed use cases and a potential UI design.

Table of Contents

[1. System Architecture 4](#_Toc479103200)

[2. Database Design 5](#_Toc479103201)

[2.1 Table design 5](#_Toc479103202)

[2.1.1 User 6](#_Toc479103203)

[2.1.1.1 SQL command 6](#_Toc479103204)

[2.1.2 UserRating 6](#_Toc479103205)

[2.1.2.1 SQL command 6](#_Toc479103206)

[2.1.3 Driver 7](#_Toc479103207)

[2.1.3.1 SQL command 7](#_Toc479103208)

[2.1.4 Passenger 8](#_Toc479103209)

[2.1.4.1 SQL command 8](#_Toc479103210)

[2.1.5 Experience 8](#_Toc479103211)

[2.1.5.1 SQL command 8](#_Toc479103212)

[2.1.6 CarDetails 9](#_Toc479103213)

[2.1.6.1 SQL command 9](#_Toc479103214)

[2.1.7 Lift 9](#_Toc479103215)

[2.1.7.1 SQL command 9](#_Toc479103216)

[2.1.8 CompletedLifts 10](#_Toc479103217)

[2.1.8.1 SQL command 10](#_Toc479103218)

[2.1.9 Request 11](#_Toc479103219)

[2.1.9.1 SQL Command 11](#_Toc479103220)

[2.1.10 CarGroup 12](#_Toc479103221)

[2.1.10.1 SQL Command 12](#_Toc479103222)

[3. Use Cases 13](#_Toc479103223)

[3.1 Use Case Diagram 13](#_Toc479103224)

[3.2 Detailed Use Cases 13](#_Toc479103225)

[3.2.1. Register Use Case 14](#_Toc479103226)

[3.2.1.1. System Sequence Diagram 14](#_Toc479103227)

[3.2.2. Login Use Case 15](#_Toc479103228)

[3.2.2.1. System Sequence Diagram 16](#_Toc479103229)

[3.2.3. Delete Account Use Case 16](#_Toc479103230)

[3.2.3.1. System Sequence Diagram 17](#_Toc479103231)

[3.2.4. Offer Lift Use Case 17](#_Toc479103232)

[3.2.4.1. System Sequence Diagram 18](#_Toc479103233)

[3.2.5. Accept/Deny Lift Request Use Case 18](#_Toc479103234)

[3.2.5.1. System Sequence Diagram 19](#_Toc479103235)

[3.2.6. Search for Lift Use Case 19](#_Toc479103236)

[3.2.6.1. System Sequence Diagram 20](#_Toc479103237)

[3.2.7. Request Lift Use Case 20](#_Toc479103238)

[3.2.7.1. System Sequence Diagram 21](#_Toc479103239)

[3.2.8. Completed Lift Use Case 21](#_Toc479103240)

[3.2.8.1. System Sequence Diagram 22](#_Toc479103241)

[3.2.9. Rate User Use Case 22](#_Toc479103242)

[3.2.9.1. System Sequence Diagram 23](#_Toc479103243)

[3.2.10. Get Experience Use Case 23](#_Toc479103244)

[3.2.10.1. System Sequence Diagram 24](#_Toc479103245)

[3.2.11. View Leader Board Use Case 24](#_Toc479103246)

[3.2.11.1. System Sequence Diagram 25](#_Toc479103247)

[4. Design 25](#_Toc479103248)

[4.1 UI Screenshots 25](#_Toc479103249)

[4.1.1. Landing Page 26](#_Toc479103250)

[4.1.2. Registration Page 26](#_Toc479103251)

[4.1.3. Car Details Page 27](#_Toc479103252)

[4.1.4. Main Page 27](#_Toc479103253)

[4.1.5. Available Lifts Page 28](#_Toc479103254)

[4.1.6. Request Lift Page 28](#_Toc479103255)

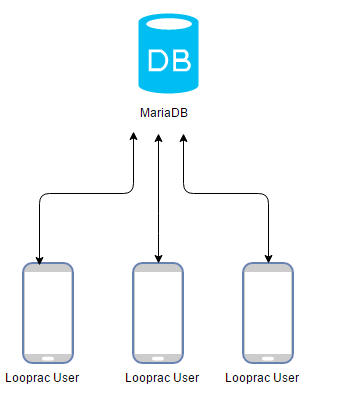
[4.1.7. Driver Details Page 29](#_Toc479103256)

[4.1.8. Offer Lift Page 29](#_Toc479103257)

[4.2. UI Flow 30](#_Toc479103258)

# System Architecture

MySQL



Looprac API

Figure 1

# Database Design

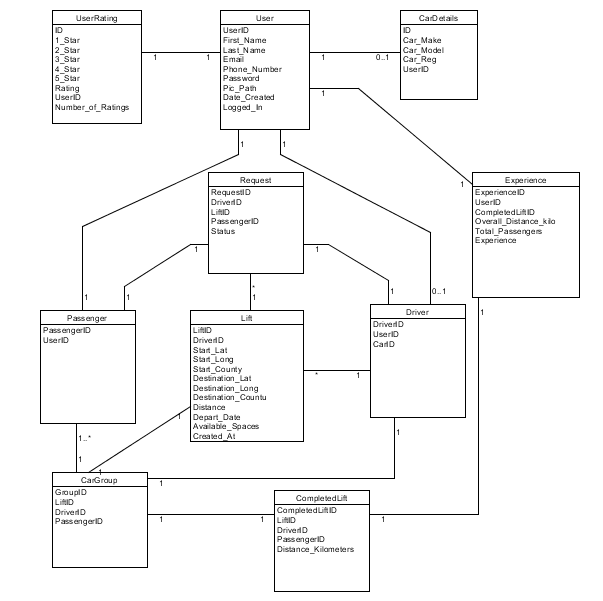


Figure 2

The database chosen for Looprac is MySQL. Below are the tables that will be created in the database along with the SQL commands to add data to them.

## 2.1 Table design

The table insert examples shown below are different to that which are used in the API for that reason a lot of records are automatically inserted into other tables while being processed. Also within the API the insert values are parameterised for added security. So the shown insert commands below just show basic examples of how the data would be entered into each table with sample values.

### 2.1.1 User



Figure 3

#### 2.1.1.1 SQL command

**Table creation:**

CREATE TABLE `User`

( `UserID` int(11) NOT NULL AUTO\_INCREMENT,

`First\_Name` varchar(40) NOT NULL,

`Last\_Name` varchar(40) NOT NULL,

`Email` varchar(100) DEFAULT NULL,

`Phone\_Number` varchar(20) DEFAULT NULL,

`Password` varchar(65) DEFAULT NULL, `Pic\_Path` varchar(100) DEFAULT NULL,

`Date\_Created` date DEFAULT NULL,

`Logged\_In` tinyint(1) DEFAULT '0',

PRIMARY KEY (`UserID`));

**Adding data to the table**

INSERT into User(First\_Name, Last\_Name, Email, Phone\_Number, Password, Pic\_Path,)

VALUES

(‘John’, ‘Sample’, ‘John@hotmail.com’, ‘0851234567’, ‘85f5e10431f69bc2a14046a13aabaefc660103b6de7a84f75c4b96181d03f0b5’, ‘johnemail.com.jpg‘);

The password is hashed in the API before it is entered into the database. The ‘Date\_Created’ field is a timestamp entered by the API when the record is created.

### 2.1.2 UserRating



Figure 4

#### 2.1.2.1 SQL command

**Table creation:**

CREATE TABLE `UserRating` (

`ID` int(11) NOT NULL AUTO\_INCREMENT,

`1\_Star` int(5) DEFAULT '0',

`2\_Star` int(5) DEFAULT '0',

`3\_Star` int(5) DEFAULT '0',

`4\_Star` int(5) DEFAULT '0',

`5\_Star` int(5) DEFAULT '0',

`Rating` decimal(3,2) DEFAULT '0.00',

`UserID` int(11) DEFAULT NULL,

`Number\_of\_Ratings` int(6) DEFAULT '0',

PRIMARY KEY (`ID`), KEY `UserID` (`UserID`),

CONSTRAINT `UserRating\_ibfk\_1` FOREIGN KEY (`UserID`) REFERENCES `User` (`UserID`));

Users are automatically added into this table by the API when they register with the application, from the User table.

**Inserting data:**

INSERT INTO UserRating(UserID)

VALUES(100)

### 2.1.3 Driver



Figure 5

#### 2.1.3.1 SQL command

**Table creation:**

CREATE TABLE `Driver` (

`DriverID` int(11) NOT NULL AUTO\_INCREMENT,

`UserID` int(11) DEFAULT NULL,

`CarID` int(5) DEFAULT NULL,

`LiftID` int(11) DEFAULT NULL,

PRIMARY KEY (`DriverID`),

KEY `UserID` (`UserID`),

KEY `LiftID` (`LiftID`),

CONSTRAINT `Driver\_ibfk\_1` FOREIGN KEY (`UserID`) REFERENCES `User` (`UserID`),

CONSTRAINT `Driver\_ibfk\_2` FOREIGN KEY (`LiftID`) REFERENCES `Lift` (`LiftID`))

**Adding data to the table**

Users are added to this table automatically the first time they offer a lift in the app.

INSERT into Driver(UserID, CarID)

VALUES

(100, 1);

Where car ID is gotten from CarDetails table.

### 2.1.4 Passenger



Figure 6

#### 2.1.4.1 SQL command

**Table creation:**

CREATE TABLE `Passenger` (

`PassengerID` int(11) NOT NULL AUTO\_INCREMENT,

`UserID` int(11) DEFAULT NULL,

PRIMARY KEY (`PassengerID`),

KEY `UserID` (`UserID`));

**Adding data to the table**

Users are automatically inserted into this table when they register with the application, from the User table.

INSERT into Passenger(UserID)

VALUES

(100);

### 2.1.5 Experience



Figure 7

#### 2.1.5.1 SQL command

**Table creation:**

CREATE TABLE `Experience` (

`ExperienceID` int(11) NOT NULL AUTO\_INCREMENT,

`UserID` int(11) NOT NULL,

`Overall\_Distance\_kilo` decimal(5,1) DEFAULT '0.0',

`Overall\_Passengers` int(6) DEFAULT '0',

`Experience` decimal(8,2) DEFAULT '0.00',

PRIMARY KEY (`ExperienceID`),

KEY `UserID` (`UserID`));

**Adding data to the table**

Users are automatically inserted into this table when they register with the application, from the User table.

INSERT into Experience(UserID)

VALUES

(100);

### 2.1.6 CarDetails



Figure 8

#### 2.1.6.1 SQL command

**Table creation:**

CREATE TABLE `CarDetails` (

`ID` int(11) NOT NULL AUTO\_INCREMENT,

`Car\_Make` varchar(20) NOT NULL,

`Car\_Model` varchar(20) NOT NULL,

`Car\_Reg` varchar(20) NOT NULL,

`UserID` int(11) DEFAULT NULL,

PRIMARY KEY (`ID`),

KEY `UserID` (`UserID`),

CONSTRAINT `CarDetails\_ibfk\_1` FOREIGN KEY (`UserID`) REFERENCES `User` (`UserID`))

**Adding data to the table**

INSERT into CarDetails(UserID,Car\_Make, Car\_Model, Car\_Colour, Car\_Reg)

VALUES

(100, ‘Hyundai, ‘Coupe, ’01-MH-12345’);

### 2.1.7 Lift

Figure 9

#### 2.1.7.1 SQL command

**Table creation:**

CREATE TABLE `Lift` (

`LiftID` int(11) NOT NULL AUTO\_INCREMENT,

`DriverID` int(5) DEFAULT NULL,

`Start\_Lat` double(17,15) DEFAULT NULL,

`Start\_Long` double(17,15) DEFAULT NULL,

`Start\_County` varchar(50) DEFAULT NULL,

`Destination\_Lat` double(17,15) DEFAULT NULL,

`Destination\_Long` double(17,15) DEFAULT NULL,

`Destination\_County` varchar(50) DEFAULT NULL,

`Distance` decimal(6,1) DEFAULT '0.0',

`Depart\_Date` datetime DEFAULT NULL,

`Available\_Spaces` int(11) DEFAULT NULL,

`Created\_At` datetime DEFAULT NULL,

PRIMARY KEY (`LiftID`),

KEY `DriverID` (`DriverID`))

**Adding data to the table**

INSERT into Lift(DriverID, Start\_Lat, Start\_Long, Start\_County, Destination\_Lat, Destination\_Long, Destination\_County, Distance, Depart\_Date, Available\_Spaces)

VALUES

(1, 52.82483, -6.9202759, ‘Carlow’, 51.88552293, -8.513025278, ‘Cork’, 193.0, 01/04/2017 15:20, 4);

Distance is calculated using Google Distance Matrix API. Created\_At is a timestamp made by the API when it is registered.

### 2.1.8 CompletedLifts

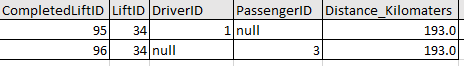


Figure 10

#### 2.1.8.1 SQL command

**Table creation:**

CREATE TABLE `CompletedLifts` (

`CompletedLiftID` int(11) NOT NULL AUTO\_INCREMENT,

`LiftID` int(11) DEFAULT NULL,

`DriverID` int(11) DEFAULT NULL,

`PassengerID` int(11) DEFAULT NULL,

`Distance\_Kilometers` decimal(5,1) DEFAULT NULL,

PRIMARY KEY (`CompletedLiftID`),

KEY `LiftID` (`LiftID`),

KEY `DriverID` (`DriverID`),

KEY `PassengerID` (`PassengerID`),

CONSTRAINT `CompletedLifts\_ibfk\_1` FOREIGN KEY (`LiftID`) REFERENCES `Lift` (`LiftID`), CONSTRAINT `CompletedLifts\_ibfk\_2` FOREIGN KEY (`DriverID`) REFERENCES `Driver` (`DriverID`), CONSTRAINT `CompletedLifts\_ibfk\_3` FOREIGN KEY (`PassengerID`) REFERENCES `Passenger` (`PassengerID`));

**Adding data to the table**

A record is created here automatically by getting fields from CarGroup table.

**For driver:**

INSERT into CompletedLifts(LiftID, DriverID, Distance\_Kilometers)

VALUES

(34, 1, 193.0);

**For passengers:**

INSERT into CompletedLifts(LiftID, PassengerID, Distance\_Kilometers)

VALUES

(34, 3, 193.0);

### 2.1.9 Request

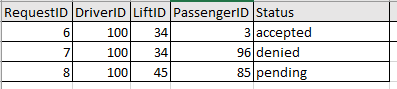


Figure 11

#### 2.1.9.1 SQL Command

**Table creation:**

CREATE TABLE `Request` (

`RequestID` int(11) NOT NULL AUTO\_INCREMENT,

`DriverID` int(11) NOT NULL,

`LiftID` int(11) NOT NULL,

`PassengerID` int(11) NOT NULL,

`Status` varchar(8) DEFAULT 'Pending',

PRIMARY KEY (`RequestID`),

KEY `fk\_Request\_Driver` (`DriverID`),

KEY `fk\_Request\_Lift` (`LiftID`),

KEY `fk\_Request\_Passenger` (`PassengerID`),

CONSTRAINT `fk\_Request\_Driver` FOREIGN KEY (`DriverID`) REFERENCES `Driver` (`DriverID`), CONSTRAINT `fk\_Request\_Lift` FOREIGN KEY (`LiftID`) REFERENCES `Lift` (`LiftID`),

CONSTRAINT `fk\_Request\_Passenger` FOREIGN KEY (`PassengerID`) REFERENCES `Passenger` (`PassengerID`));

**Adding data to the table**

INSERT INTO Request(DriverID, LiftID, PassengerID)

VALUES

(100, 34, 3)

### CarGroup



Figure 12

#### 2.1.10.1 SQL Command

**Table creation:**

CREATE TABLE `CarGroup` (

`GroupID` int(11) NOT NULL AUTO\_INCREMENT,

`LiftID` int(11) NOT NULL,

`DriverID` int(11) NOT NULL,

`PassengerID` int(11) NOT NULL,

PRIMARY KEY (`GroupID`),

KEY `fk\_CarGroup\_Lift` (`LiftID`), KEY `fk\_CarGroup\_Driver` (`DriverID`),

KEY `fk\_CarGroup\_Passenger` (`PassengerID`));

**Adding data to the table:**

This record is automatically inserted when a driver accepts a passenger’s request to join a lift.

INSERT INTO CarGroup

(LiftID, DriverID, PassengerID)

VALUES

(34, 100, 3)

# Use Cases

## 3.1 Use Case Diagram

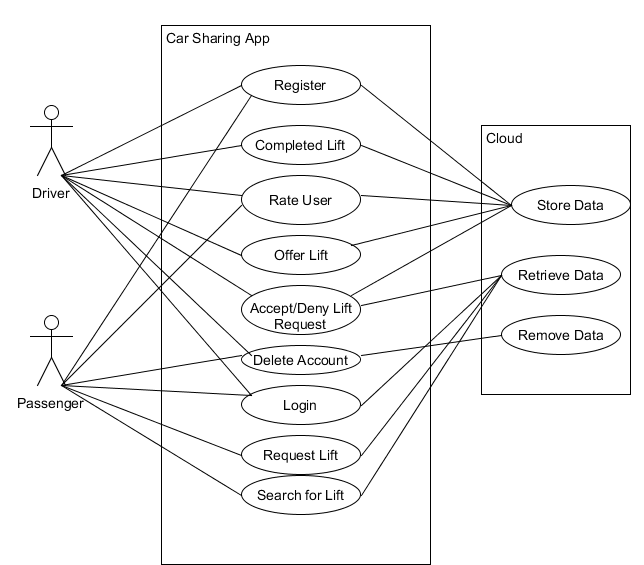


Figure 13

## 3.2 Detailed Use Cases

This section will look at the brief use cases from the functional specification document and turn them into detailed use cases.

### 3.2.1. Register Use Case

**Actors**: user, cloud

**Brief description**: This use case begins when a user wishes to register with the app. The user enters their details (name, email, phone number)). The app validates the entered information and stores it to the cloud. The use case ends when the user’s details are registered with the app.

**Main success scenario:**

1. The user starts up the app
2. The app offers them choice to login or register
3. The user selects register
4. The app displays register page with detail inputs
5. The user enters details – name, email address, phone number, password and picture
6. The app validates the inputted details
7. The app sends the data to be stored in the cloud
8. The app presents user with acknowledgment that they are registered

**Alternatives**

6A. Details entered are of incorrect format

1. App alerts user that inputted details were of incorrect format
2. The app displays register page with detail inputs

#### 3.2.1.1. System Sequence Diagram

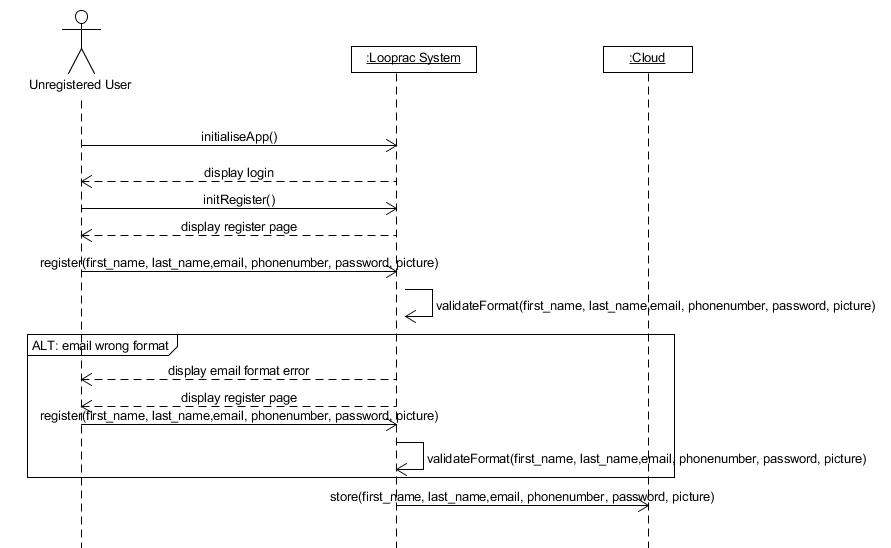


Figure 14

### 3.2.2. Login Use Case

**Actors:** user, cloud

**Brief description:** This use case begins when a user wants to login to the app. The user enters their email address and password. The app validates the information and queries the cloud. This use case ends when the information is validated and grants the user access to the app.

**Main success scenario:**

1. The user starts up the app
2. The app displays login page
3. The user submits their email address and password
4. The app validates the email format is correct
5. The app queries the cloud with the entered email address
6. It verifies that the right password was entered
7. The app grants the user access to the app

**Alternatives**

5A. The email format entered is incorrect

1. The app alerts the user that the email was incorrect
2. The app prompts the user to recheck their email

7A. The password entered was incorrect

1. The app alerts the user that the password was incorrect
2. The app prompts the user to enter password

#### 3.2.2.1. System Sequence Diagram

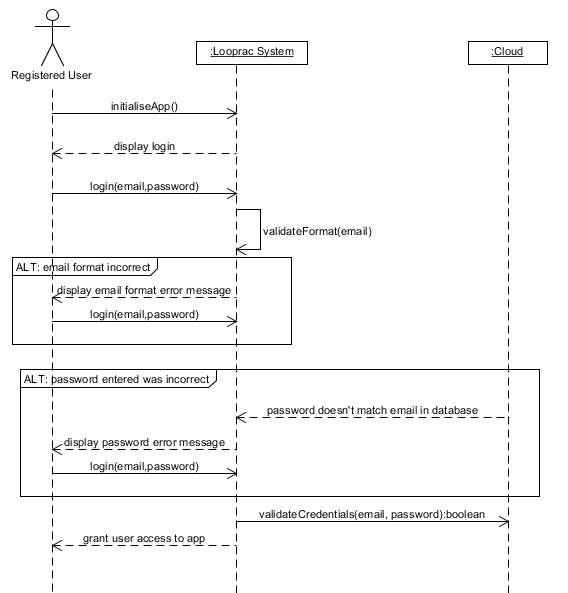


Figure 15

### 3.2.3. Delete Account Use Case

**Actors:** User, cloud

**Brief description:**  This use case begins when a user wishes to delete their account. The user will select ‘delete account’ option from their profile. The app will verify that they want to continue. The user selects ‘yes’. The app updates the cloud by removing the user’s data from the cloud. This use case ends when the app provides the user with an acknowledgement that the task is complete.

**Main success scenario:**

1. The user selects delete account option
2. The app alerts the user of the choice they are making and asks them if they are sure
3. The user selects yes
4. The app queries the cloud with the user’s unique ID
5. The cloud updates itself by removing the account associated with the unique ID
6. The app displays an acknowledgement to the user that their account was deleted

#### 3.2.3.1. System Sequence Diagram

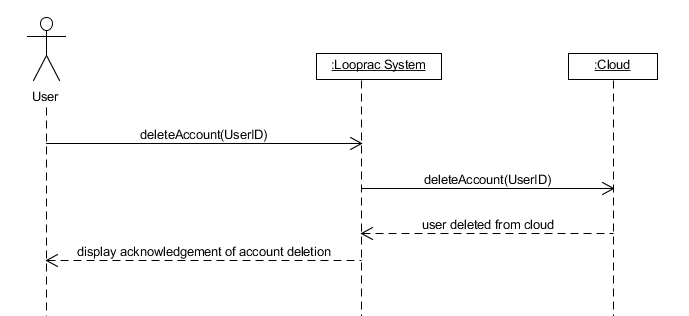


Figure 16

### 3.2.4. Offer Lift Use Case

**Actors:** driver, cloud

**Brief description:** This use case begins when a driver wishes to offer a lift in their vehicle. The app will ask the driver for their location, destination, the depart time and how many spaces are available in the car. The app validates this information and sends it to the cloud. This use case ends when the details are validated and an acknowledgment is shown to the driver.

**Main success scenario:**

1. The driver selects to offer a lift
2. The app prompts the driver for details regarding the lift – location, destination, depart date, depart time, how many spaces are available
3. The app validates that the entered information is valid format
4. The app sends the information to the cloud
5. The app presents the driver with acknowledgement that the lift offer was accepted

**Alternatives:**

3A. Driver enters date/time that has passed

1. App alerts driver that the entered date/time is not valid
2. App prompts driver to re-enter a valid date

#### 3.2.4.1. System Sequence Diagram

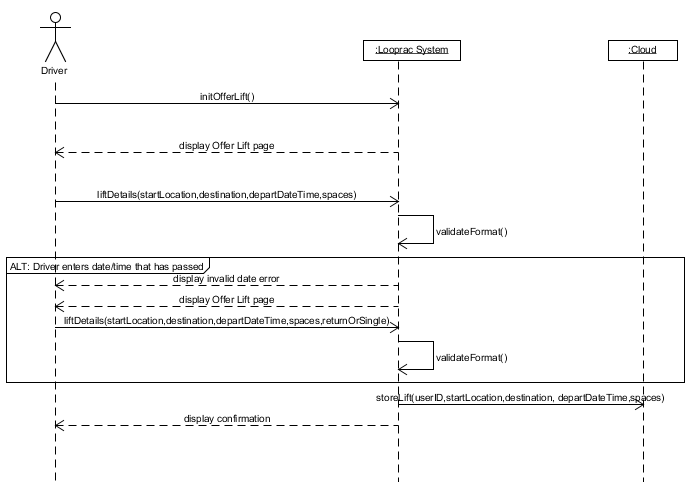


Figure 17

### 3.2.5. Accept/Deny Lift Request Use Case

**Actors:** driver, passenger, cloud

**Brief description:** this use case begins when a driver gets a notification that a lift request is waiting for their approval or denial. The driver will select the lift request. The app will display details to which lift that the request is for that the driver has previously setup. The app will display the potential passengers name and rating to the driver. The driver can accept the request or deny the request. This use case ends when the driver accepts or denies the request.

**Main success scenario:**

1. The app notifies the driver that a passenger has requested a space in their vehicle
2. The driver selects the request
3. The app displays information about the request – what lift offer it is connected to (location, destination, time) and the requested passengers name and rating
4. The driver selects to accept the request
5. The app displays an acknowledgement to the driver of their choice
6. The app creates a group for the lift record by adding the passenger, driver and updating spaces available for the lift.
7. The app sends a notification to the passenger that their request was approved with details about the lift and the car including make, model, colour and registration

**Alternatives:**

4A. The driver denies the request

1. The app displays an acknowledgement to the driver of their choice
2. The app notifies the passenger that their request was denied

#### 3.2.5.1. System Sequence Diagram

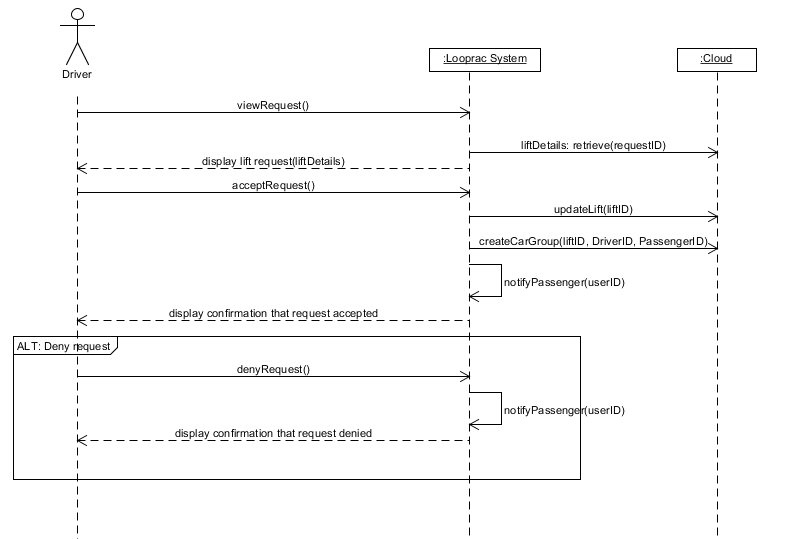


Figure 18

### 3.2.6. Search for Lift Use Case

**Actors:** passenger, cloud

**Brief description:** this use case begins when a passenger wants to search for available lifts. The passenger selects the option ‘available lifts’. The app queries the cloud and retrieves and populates the list with all available lifts, which the passenger has not already requested to join. The app displays a short description of the lift in the list. This use case ends when the app displays the available lifts to the passenger.

**Main success scenario:**

1. The passenger selects the available lifts option
2. The app queries the cloud and updates the list with the available lifts
3. The app displays the list of lifts with the location, destination and time added

#### 3.2.6.1. System Sequence Diagram

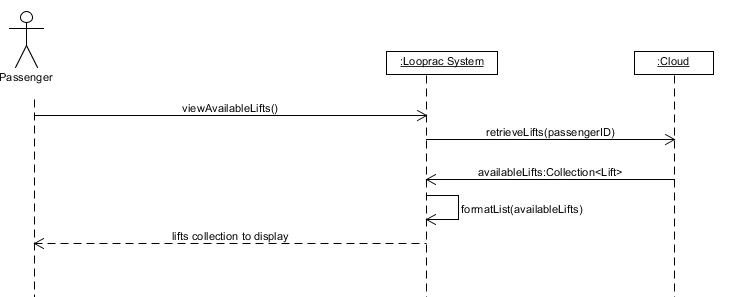


Figure 19

### 3.2.7. Request Lift Use Case

**Actors:** passenger, cloud

**Brief description:** this use case begins when a passenger has chosen a lift from search for lift use case and they wish to request a space in the vehicle. The passenger will select the lift from the list. The app will query the cloud and display details of the lift and the driver. The passenger will select the request lift option. This use case ends when the app gives the passenger acknowledgement that the request has been sent.

**Main success scenario:**

1. The passenger selects the lift from the list of available lifts
2. The app queries the cloud for information on the lift and driver
3. The app displays to the passenger – location of lift and destination on a map, the time the lift will be leaving at, spaces available in the vehicle and the drivers rating
4. The passenger selects the request lift option
5. The app provides the passenger with acknowledgment that the request has been sent

#### 3.2.7.1. System Sequence Diagram

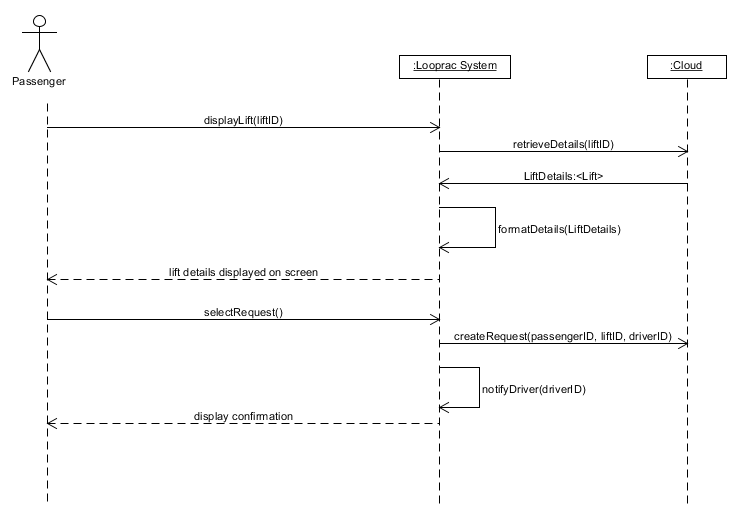


Figure 20

### Completed Lift Use Case

**Actors:** driver, passenger, cloud

**Brief description:** this use case begins when the driver gets within five hundred meters of destination. The app will allow the driver to select the completed option. The app will store the completed journey to the cloud with the lift details. This use case ends when the app informs the driver and passenger with a message showing them they’re new statistics.

**Main success scenario:**

1. The app enables a ‘complete’ option
2. The user selects the ‘complete’ option when they are within five hundred meters of the destination
3. The app stores the completed journey to the cloud along with distance the driver travelled
4. The app stores the completed journey to the cloud with the distance for the passengers
5. The app retrieves from the cloud the total distance that the user has from completed journeys
6. The app displays a message to the driver and passengers informing them that the lift is complete.

**Alternatives**

2A. if the driver selects complete lift when the distance is greater than five hundred meters

1. App informs driver that they are too far from destination to complete the lift.
2. App updates the drivers location on the map

#### System Sequence Diagram

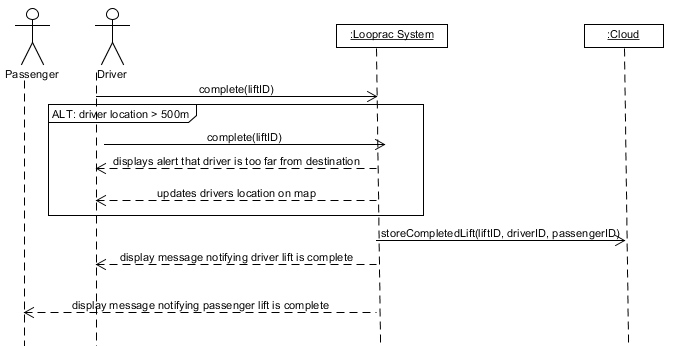


Figure 21

### Rate User Use Case

**Actors:** user, cloud

**Brief description:** this use case begins when the user is prompted to rate the driver/passenger(s) when the lift has been completed. The app will display a five-star system and allow the user to select between one star to five stars. The user submits their rating. The app updates the driver and passenger’s rating in the cloud. This use case ends when the app displays an acknowledgement that the rating was submitted.

**Main success scenario:**

1. The app prompts the user to rate other users in the vehicle
2. The app retrieves other users sharing the lift
3. The app displays a screen with five stars and informs the user to select a rating out of five for each user
4. The user selects a rating and selects submit
5. The app updates the other user’s ratings in the cloud
6. The app displays an acknowledgement to the user that the ratings were successfully submitted.

#### System Sequence Diagram

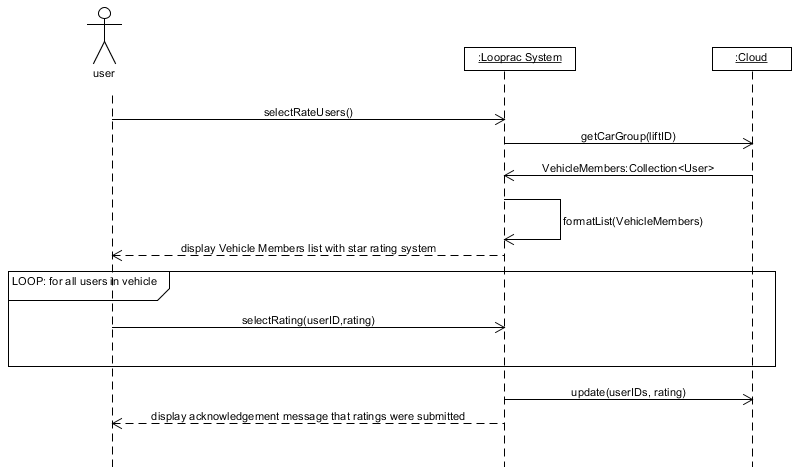


Figure 22

### Get Experience Use Case

**Actors:** User, cloud

**Brief Description:** this use case begins when the user has completed the user rating use case. The user experience is updated in the cloud with data from the lift they just completed. The cloud retrieves their new experience and overall distance completed. This use case ends when the application displays the results to the user.

**Main Success Scenario**

1. The application updates the users experience and distance in the cloud
2. The app retrieves the new experience and distance from the cloud
3. The app displays the users experience and distance from the lift and their overall experience and distance from shared lifts

**Alternatives**

**1A.** If the user is the driver

1. The app updates the users experience, distance and overall passengers
2. The app retrieves the new experience, overall distance and overall passengers
3. The app displays the users experience, distance and number of passengers from the lift and their new overall experience, overall distance and overall passengers

#### System Sequence Diagram

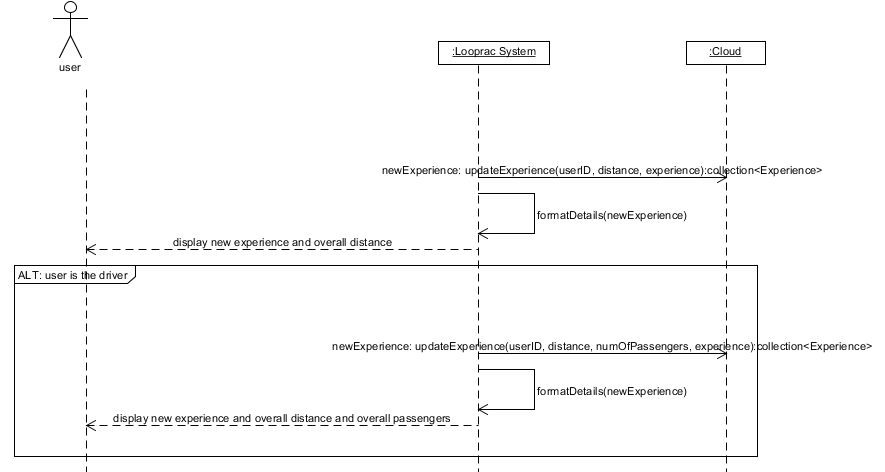


Figure 23

### View Leader Board Use Case

**Actors:** User, cloud

**Brief Description:** this use case begins when the wants to see where they rank against other users of the application in terms of experience. The user selects ‘leaderboard’ option. The application retrieves a collection of all users of the application and their experience. The application displays the collection in a table and highlights the users position within the table and displays their rank at the top. This use case ends when the application displays the table.

**Main Success Scenario**

1. The user select ‘Leaderboard’
2. The application retrieves all users of the application and their experience
3. The application displays all the users in a table
4. The application highlights the users name and ranking within the table and displays their rank at the top of the page

#### System Sequence Diagram

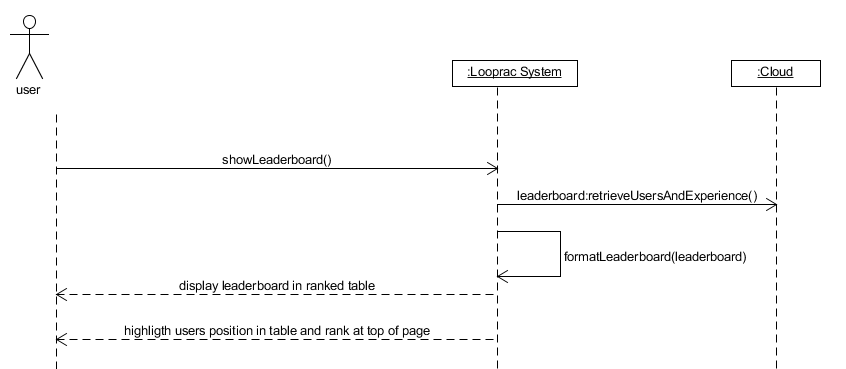


Figure 24

# Design

The design of the app will be a simplistic interface aiming at a positive user experience. This involves an easy and fluid flow of navigation through the app. Below are some potential UI mock ups for certain pages. These few screenshots are to give an idea of the UI design and colour palette ideas.

Below is the primary colour palette chosen for the UI of Looprac.

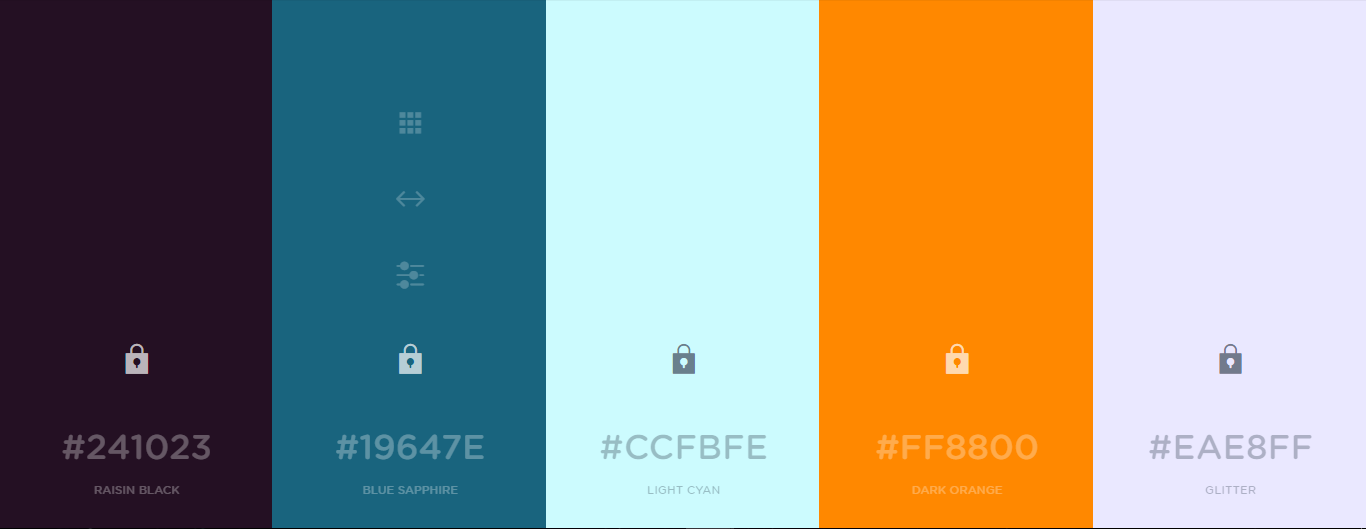


Figure 25

## UI Screenshots

Below are mock ups for what a potential look that the app may incorporate. These mock ups may be accurate in some cases and only guidelines in others.

### 4.1.1. Landing Page

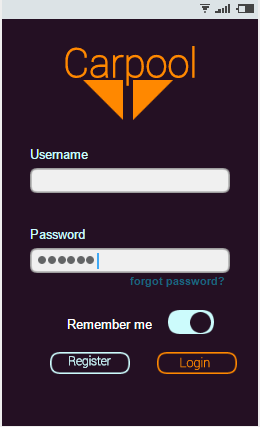
The landing page is the very first page that the users will see. Below are mock ups.

Figure 26

### 4.1.2. Registration Page

This is the page where users will register for the app. It requires little information so that the process is quick. There is an optional link to register a car or this can be done at another time within the app.

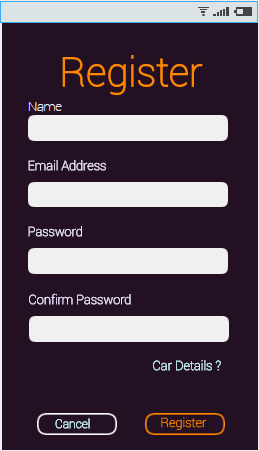


Figure 27

### 4.1.3. Car Details Page

This page can be accessed when the user first registers with Looprac or can be found in the options if they wish to complete it another time or update the details.

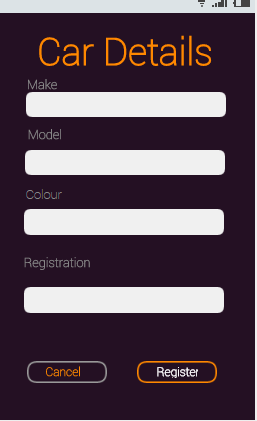


Figure 28

### 4.1.4. Main Page

This is the main page of the app. It will be the page that the users will be brought to when they log in for the first time, and thereafter. Potentially, the map will be populated with available lifts in their area. Alternatively, they can select to find lift which will bring them to a list of available lifts. They can also offer a lift from this page.

Figure 29

### 4.1.5. Available Lifts Page

This page presents the user with a list of available. It displays available lifts start point, destination and time it was added. The user can also select different ways to sort the list.

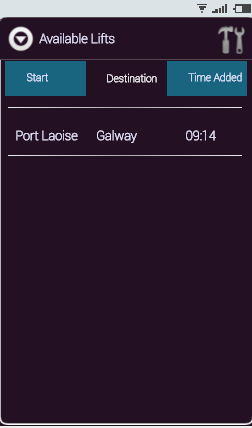


Figure 30

### 4.1.6. Request Lift Page

This page presents the user with details about the lift that they are interested in. It displays the route at the top which they can expand by selecting it. It also provides them with details about the lift and the drivers name.

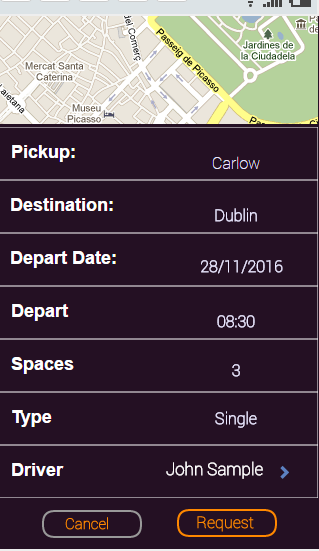


Figure 31

### 4.1.7. Driver Details Page

This page presents the user with some details about the driver that is offering the lift. It includes their name, rating out of five and any badges that they may have earned.

.

Figure 32

### 4.1.8. Offer Lift Page

This page is presented to the user when they want to become a driver and offer a lift. They are presented with a page that asks for details regarding the lift which includes start location, destination, depart date, depart time, whether it is single or return and number of spaces they are offering in the car.

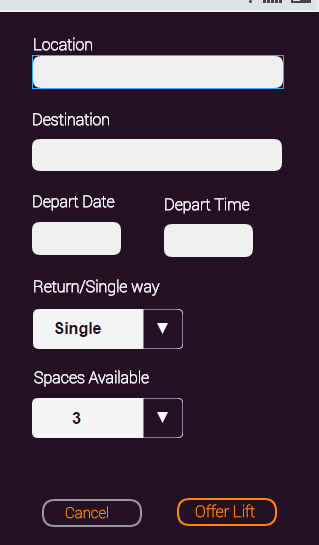


Figure 33

## UI Flow

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

Figure 34