SFWR ENG 3A04 Project

Cabpool

Deliverable 2 - Software Architecture

Group 10

Amanda Boudreau	0344130
Jordan Floyd	1225123
Mario Machado	1211979
Rakesh Mistry	1221428
Ali Reda	1206754

November 6, 2014

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to identify and analyse the major components of the system. This document will describe use case scenarios of the system, decomposition of the system into classes and their categorization, the architecture of the system and responsibilities of the classes. This document was created for software engineers, developers and management. These users shall be able to read and understand how the Application will work on a higher level after reading this document.

1.2 System Description

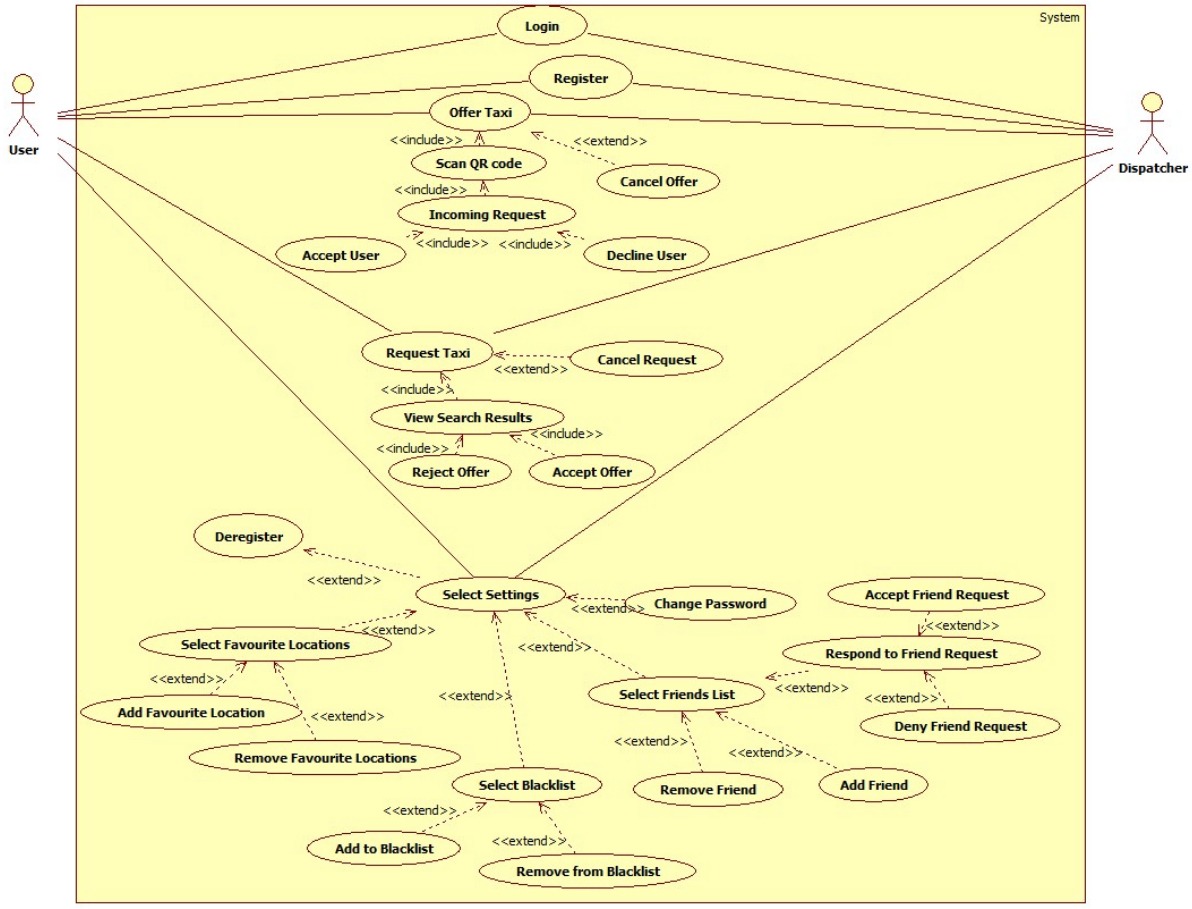
Cabpool will be an Android Application created for a Hamilton taxi company to be used in conjunction with their Dispatcher system with the aim of attracting customers by implementing a system that allows users to share taxis. The software will serve as a social networking service to connect users who want to travel to similar locations. The software will provide a platform for users to share taxis and reduce individual fare cost without having to share personal contact information. The software will allow the User to enter a taxi and scan the taxi QR code to offer the taxi and will also allow a User to request a taxi. When offering or requesting a taxi the User can accept or reject a taxi share match based on distance, rating and fare price.

Additionally, the User of the software shall be able to add users to their Friends List and share contact information with one another. Conversely, users will be able to add other users to a Blacklist to ensure they never share a cab with that User again. The User can also manage their own Favourite Locations List. If a User is travelling to a location on another user's Favourite Locations List, and wants to offer to share their taxi, the system can notify the user(s) outside the cab if they would like to join. The software will interface with Google Maps to enable users to view their route and potential route changes. The software shall not allow third party interception of personal information nor shall it allow release of personal information to unauthorized Users.

1.3 Overview

Section 2 of this document contains a use case diagram of our application with corresponding use case descriptions. Section 3 has an analysis class diagram. Section 4 discusses the architectural design of the application, showing the division of the system into subsystems. Section 5 of the document gives class responsibility collaboration cards, one is provided for each class identified.

2 Use Case Diagram



2.1 Login

A user enters their username and password in the login screen. The system verifies with the Dispatcher that the username/password pair exists in the user records database, then it displays the main menu. If the username/password pair does not exist an error message on the login screen will notify the user that they have input incorrect information and allow them to try again.

2.2 Register

A user enters their desired username and password, date of birth, sex, and phone number in the register screen. The system verifies with the Dispatcher that the username is appropriate and that the phone number is in the correct format. If everything is acceptable, the user's information will be stored in the user records database, the system then displays the main menu; otherwise an error message will appear on the registration screen notifying the user to correct indicated information.

2.3 Deregister

A user selects the option to delete their account in the settings screen. The system requests confirmation from the user before requesting the Dispatcher to remove the user account information from the user records database. If the user confirms the account deletion, user is returned to the main menu. If the user does not confirm the account deletion, the account will remain registered.

2.4 Scan QR Code

A user scans a QR code on the scan code screen. The system verifies that the QR code exists in the QR code database and then stores the users start location in the user records database, and displays the offer taxi screen. If the code is invalid the user is notified and allowed to rescan the QR code.

2.5 Offer Taxi

A user selects the option to offer taxi. They are taken to the Scan QR Code Screen. Once they have completed the Scan QR Code process, they may proceed. User enters their destination and any preferences as to who can accept their offer. The system will request the Dispatcher store the offer in the offers database.

2.6 Cancel Offer

A user cancels an offer they have made in the offer taxi screen. A user is only able to cancel an offer if they made the offer. The system removes the offer from the offers database. The system will then display to the user the main menu.

2.7 Incoming Request

A user is notified of an incoming request from the Dispatcher in the offer taxi screen once another user has accepted their offer. While the app is running, the system will notify the user of all incoming requests. The app can be minimized and/or the user does not need to be on the offer taxi screen to receive notification of the request to share the taxi.

2.8 Accept User

On the offer taxi screen, a user accepts a taxi share request another user has made. The system requests the Dispatcher retrieve the requesting user's location. The system then displays the requesting user location in the offer taxi screen. The system requires the Dispatcher to notify the requesting user that their request has been accepted.

2.9 Decline User

On the offer taxi screen, a user declines a taxi share request another user has made. The system will require the Dispatcher to notify the requesting user that their request has been declined. The system will require the Dispatcher to update the offer in the offers database so that it no longer displays to the declined user.

2.10 Request Taxi

A user enters their start location, start time, and arrival destination in the request taxi screen. The system will request for the Dispatcher to retrieve all relevant offers from the offers database and display them in the request taxi screen.

2.11 Cancel Request

A user cancels his/her request in the request taxi screen. The system will request the Dispatcher remove the request from the offers database. The main menu is displayed.

2.12 View Search Results

A user selects one of the offers in the request taxi screen. All of the information for the offer is displayed and the user is given the choice to accept or reject the offer.

2.13 Accept Offer

A user accepts an offer in the request taxi screen. The system notifies the user that the request is pending and requests the Dispatcher update the offers database to reflect the acceptance. The system will require the Dispatcher to notify the offering user that their offer has been accepted.

2.14 Reject Offer

A user rejects an offer in the request taxi screen. The system will request the Dispatcher update the offer database to reflect the offer rejection and remove this offer from the requesting user's list of offers.

2.15 Select Settings

A user selects a setting that they would like to modify in the settings screen.

2.16 Select Favourite Locations

A user views their favourite locations in the favourite locations screen.

2.17 Add Favourite Location

A user includes a favourite location to add to their list of favourite locations on the favourite locations screen. The system requests the Dispatcher add the location to the user's favourite locations list in the user records database.

2.18 Remove Favourite Location

A user selects a favourite location to remove from their list of favourite locations in the favourite locations screen. The system requests the Dispatcher remove the location from the user's favourite locations list in the user records database.

2.19 Select Blacklist

A user views their Blacklist in the Blacklist screen.

2.20 Add to Blacklist

A user selects a different user they do not want to taxi share with to add to their Blacklist on the Blacklist screen. The system requests the Dispatcher add the Blacklisted user to the user's Blacklist in the user records database.

2.21 Remove from Blacklist

A user selects a user on their Blacklist to remove from the Blacklist on the Blacklist screen. The system requests the Dispatcher remove the Blacklisted user from the user's Blacklist in the user records database.

2.22 Select Friends List

A user selects the Friends List button. The system will request the Dispatcher retrieve friend information from the user records database. User will view their friend status information in the Friend List on the Friend List screen.

2.23 Add Friend

A user selects a potential friend to add to their Friends List in the Friends List screen. The system verifies with the Dispatcher that the username entered is in the user records database. The system will request the Dispatcher sends the potential friend a notification asking them to accept the friend request. If the potential friend is not registered the user trying to add an unregistered user will be notified through an error message on the Friends List screen that no such user exists.

2.24 Remove Friend

A user selects a friend to remove from their Friends List in the Friends List screen. The system requests the Dispatcher remove the friend from the user's Friends List in the user records database.

2.25 View Friend Requests

A user views friend requests in the Friends List screen. The system requests the Dispatcher retrieve friend requests to which the user has not responded from the user records database.

2.26 Accept Friend Request

A user accepts a friend request in the Friends List screen. The system adds the accepted user to the accepting users Friends List in the user records database.

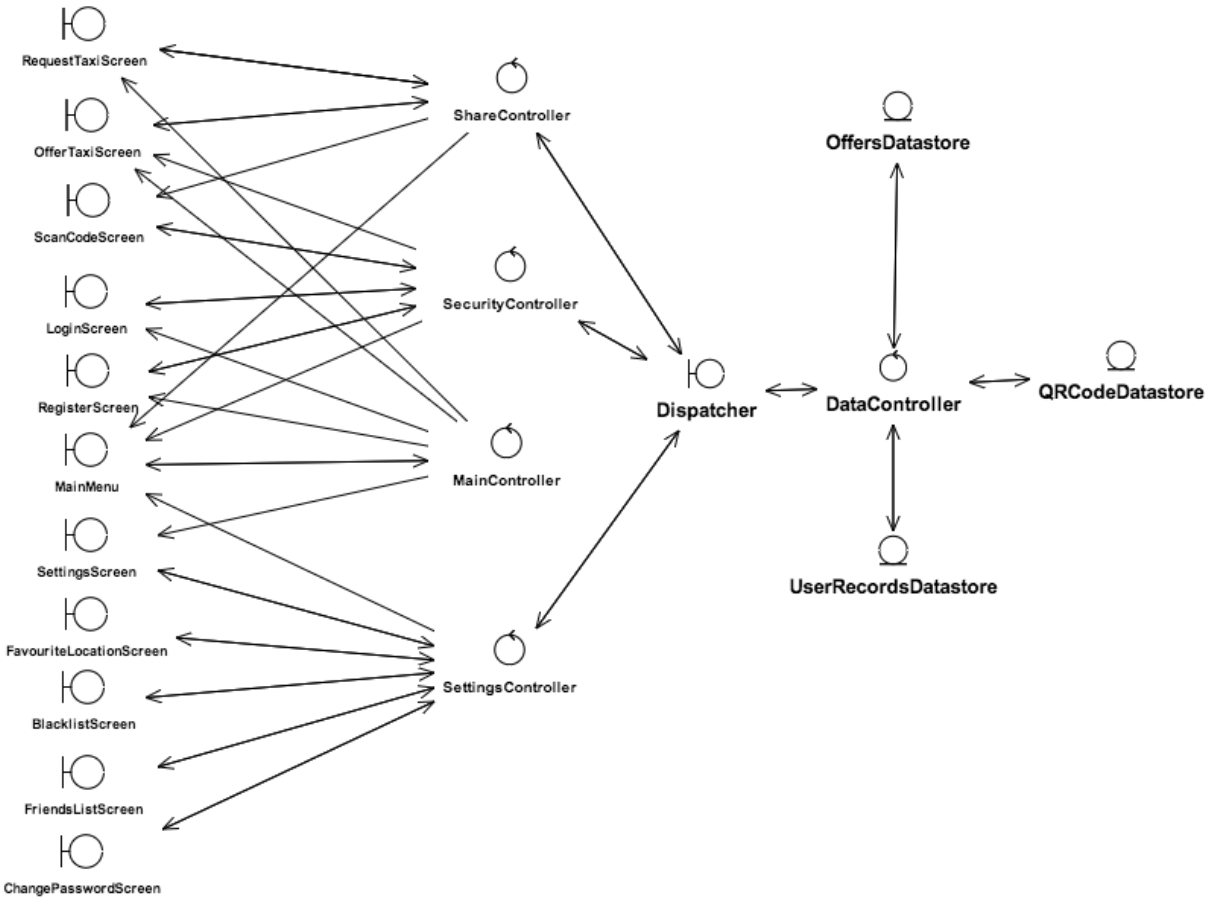
2.27 Decline Friend Request

A user declines a friend request in the Friends List screen. The system removes the request from the users list of friend requests in the user records database.

2.28 Change Password

A user enters their current password, a new password, and a confirmation of the new password on the change password screen. The system will verify with the Dispatcher that the current password is valid and that the new password and confirmation of the new password match. If the information is valid it will be entered through the Dispatcher into the user records database; otherwise it will display an error on the password screen notifying the user of the error.

3 Analysis Class Diagram



4 Architectural Design

4.1 System Architecture

Overall, the system exhibits a Client-Server based architecture. A Client-Server architecture allows for clean implementation of many-to-one relationships, as our system exhibits through the many users to one dispatcher. The many-to-one relationship is neccessary for the system because the many users need to be connected to a common data source in order for their mobile applications to reflect and impact the state of the system at any given time. This is important because ultimately the users are not just connecting to the system, but rather, they are connecting to other users through the system.

Upon closer inspection of the subsystems, it is apparent that the Client-side is designed using a Model View Controller (MVC) architecture. The user interacts with the different screens (the View) to elicit a response from the system (the Controller), which facilitates updates to the database (the Model) via the Dispatcher.

4.2 Subsystems

The system is a composition of five subsystems, the View, the ViewController, the Dispatcher, the DataController and the Datastore. The first subsystem is the View, which receives user input and displays system output. Input received by the View is sent to the ViewContoller subsystem, which is responsible for minor processing and passing the data to the Dispatcher subsystem. The Dispatcher subsystem is responsible for establishing a line of connection between the User and the Server in the Client-Server architecture. The Dispatcher receives data from the ViewController and sends it to the DataController which does the major processing of the system. The DataController subsystem receives data from the Dispatcher subsystem and takes data from the Datastore subsystem and the runs calculations to determine the output that will be sent back to the user through the Controller, and display in the View.

5 Class Responsibility Collaboration (CRC) Cards

Class Name: LoginScreen	
Responsibility:	**Collaborators:**
Display login screen	
Allow user to input user name and password	
Confirm user input valid credentials securely	SecurityController
If credentials are not valid, display error message for user	

Class Name: RegisterScreen	
Responsibility:	**Collaborators:**
Display register screen	
Allow user to input user name, password, date of birth, sex, and phone number	
Send user input to SecurityController for validation	SecurityController
If user registration information is invalid, display error message for user	

Class Name: MainMenu	
Responsibility:	**Collaborators:**
Display main menu	
Accept a choice from screen	
Send selection to MainController	MainController

Class Name: RequestTaxiScreen	
Responsibility:	**Collaborators:**
Display request taxi screen	
Accept taxi request information: start location, start time, arrival destination	
Send request taxi information to ShareController	ShareController
Display message that request was sent by user	
Respond to cancel request being selected by user	ShareController
Display message for request being cancelled by user	
Receive fare information	ShareController
Display fare information	
Allow user to rate the user with whom they shared the taxi upon arrival at destination	
Send user rating information to ShareController	ShareController
Allow user to add the user with whom they shared the taxi to their Friends List upon arrival at destination	
Send user friend request information to ShareController	ShareController
Allow user to add the user with whom they shared the taxi to their Blacklist upon arrival at destination	
Send user blacklist request information to ShareController	ShareController
Allow user to add the route taken to their Favourite Locations upon arrival at destination	
Send user favourite location request information to ShareController	ShareController

Class Name: OfferTaxiScreen	
Responsibility:	**Collaborators:**
Display offer taxi screen	
Accept offer information: start location, start time, arrival destination, preferred age range, preferred sex	
Send offer taxi information to ShareController	ShareController
Display message that offer was sent by user	
Receive offer cancellation requests from user	
Send offer cancellation request to ShareController	ShareController
Display message for offer being cancelled by user	
Receive incoming request information	ShareController
Display incoming requests	
Receive fare information	ShareController
Display fare information	
Allow user to rate the user with whom they shared the taxi upon arrival at destination	
Send user rating information to ShareController	ShareController
Allow user to add the user with whom they shared the taxi to their Friends List upon arrival at destination	
Send user friend request information to ShareController	ShareController
Allow user to add the user with whom they shared the taxi to their Blacklist upon arrival at destination	
Send user blacklist request information to ShareController	ShareController
Allow user to add the route taken to their Favourite Locations upon arrival at destination	
Send user favourite location request information to ShareController	ShareController
Send location data from the GPS	ShareController

Class Name: ScanCodeScreen	
Responsibility:	**Collaborators:**
Accept QR code information	
Send QR code information	SecurityController
Display an error message if QR code is not valid	

Class Name: SettingsScreen	
Responsibility:	**Collaborators:**
Display settings screen	
Accept user selection of menu items	
Send user selection to SettingsController	SettingsController

Class Name: FavouriteLocationScreen	
Responsibility:	**Collaborators:**
Display favourite locations screen	
Accept user addition to favourite location	
Accept user deletion of a favourite location	SettingsController
Send user favourite location input to SettingsController	SettingsController
Display message for successful favourite location adidtion or deletion	
Display an error message if addition to or deletion from favourite locations was unsuccessful	

Class Name: ChangePasswordScreen	
Responsibility:	**Collaborators:**
Display change password screen	
Accept password information: old password, new password and new password confirmation	
Send password information to SettingsController for validation	SettingsController
Display validation error message	
Display message for successful password change	

Class Name: BlacklistScreen	
Responsibility:	**Collaborators:**
Display blacklist screen	
Accept user input of a user to be added to blacklist	
Accept user input of a user to be deleted from blacklist	
Send blacklisted user information to SettingsController	SettingsController
Display blacklist modification error message	
Display message for successful blacklist modification	

Class Name: FriendsListScreen	
Responsibility:	**Collaborators:**
Display friends list screen	
Request for the settings controller to retrieve user's friends and incoming friend requests	SettingsController
Display the user's friends list	
Allow the user to input a user to add to their friends list	
Send username of potential friend to SettingsController	SettingsController
Allow the user to input a user to remove from their friendslist	
Send username of removed friend to SettingsController	SettingsController
Allow the user to choose to view their incoming friend requests	
Display user's incoming friend requests	
Allow the user to accept a friend request	
Send friend request acceptance to the SettingsController	SettingsController
Allow the user to decline a friend request	
Send friend request rejection to the SettingsController	SettingsController
Display message for successful friends list modification	
Display error message for unsuccessful friends list modification	

Class Name: ShareController	
Responsibility:	**Collaborators:**
Display request taxi screen	
Accept a choice from request taxi screen	RequestTaxiScreen
Send taxi request changes to Dispatcher	Dispatcher
Display updates on request taxi screen	RequestTaxiScreen
Display offer taxi screen	
Accept a choice from the offer taxi screen	OfferTaxiScreen
Send offer taxi changes to Dispatcher	Dispatcher
Display updates on offer taxi screen	OfferTaxiScreen
Display scan code screen	ScanCodeScreen
Send scanned code to Dispatcher	Dispatcher
Display main menu	
Accept a choice on the main menu	MainMenu
Accept location information from the user	OfferTaxiScreen, RequestTaxiScreen
Send user location information to the Dispatcher	Dispatcher
Accept rating information from the user	OfferTaxiScreen, RequestTaxiScreen
Send user rating information to the Dispatcher	Dispatcher
Accept friend request information from the user	OfferTaxiScreen, RequestTaxiScreen
Send user friend request information to the Dispatcher	Dispatcher
Accept blacklist addition from the user	OfferTaxiScreen, RequestTaxiScreen
Send user blacklist addition information to the Dispatcher	Dispatcher
Accept fare calculation from the Dispatcher	Dispatcher
Send fare calculation information to the user	OfferTaxiScreen, RequestTaxiScreen

Class Name: SecurityController	
Responsibility:	**Collaborators:**
Receive login information	LoginScreen
Send username and password to Dispatcher for login verification	Dispatcher
Send incorrect login errors	LoginScreen
Receive registration information	RegisterScreen
Send registration information to Dispatcher for validation	Dispatcher
Send registration errors	RegisterScreen
Receive Scanned QR Code information	ScanQRCodeScreen
Send Scanned QR Code information to Dispatcher for validation	Dispatcher
Display offer taxi screen	OfferTaxiScreen

Class Name: MainController	
Responsibility:	**Collaborators:**
Display main menu	MainMenu
Accept user input from main menu	MainMenu
Display settings screen	SettingsScreen
Display register screen	RegisterScreen
Display login screen	LoginScreen
Display scan code screen	ScanCodeScreen
Display offer taxi screen	OfferTaxiScreen
Display request taxi screen	RequestTaxiScreen

Class Name: SettingsController	
Responsibility:	**Collaborators:**
Display main menu	MainMenu
Accept a choice from main menu	MainMenu
Display settings screen	SettingScreen
Accept a choice from settings screen	SettingsScreen
Display favourite location screen	FavouriteLocationScreen
Accept user favourite locations input	FavouriteLocationScreen
Send changes to favourite locations to Dispatcher for processing and validation	Dispatcher
Display friends list screen	FriendsListScreen
Accept user friends list modification	
Send friends list modifications to Dispatcher	Dispatcher
Display blacklist screen	BlacklistScreen
Accept user blacklist modification	
Send blacklist modifications to Dispatcher	Dispatcher
Display change password screen	ChangePasswordScreen
Accept user password modification	
Send password modification information to Dispatcher for validation	Dispatcher
Send confirmation messages to user	FavouriteLocationsScreen, FriendsListScreen, BlackListScreen, ChangePasswordScreen
Send error messages to user	FavouriteLocationsScreen, FriendsListScreen, BlackListScreen, ChangePasswordScreen

Class Name: DataController	
Responsibility:	**Collaborators:**
Receive requests from Dispatcher	Dispatcher
Read data from OffersDatastore	OffersDatastore
Read data from QRCodeDatastore	QRCodeDatastore
Read data from UserRecordsDatastore	UserRecordsDatastore
Receive user location information from Dispatcher	Dispatcher
Store user location information in UserRecordsDatastore	UserRecordsDatastore
Check for matches between user favorite locations and offers or requests	
Perform fare calculations	
Validate login credentials	
Validate that received QR code exists in the datastore	
Validate registration credentials	
Add user account to or remove from UserRecordsDatastore	UserRecordsDatastore
Match offers and requests	
Add offers to the OffersDatastore	OffersDatastore
Remove offers from the OffersDatastore	OffersDatastore
Add friend request to the UserRecordsDatastore	UserRecordsDatastore
Remove friend request from the UserRecordsDatastore	UserRecordsDatastore
Change users passwords in the UserRecordsDatastore	UserRecordsDatastore
Add user to, or remove user from blacklist in the UserRecordsDatastore	UserRecordsDatastore
Add user to, or remove user from friends list in the UserRecordsDatastore	UserRecordsDatastore
Add route to, or remove route from favourite locations in the UserRecordsDatastore	UserRecordsDatastore
Store user request in the OffersDatastore	OffersDatastore
Send output information to Dispatcher: offer and request results, fare calculation, setting modification success status	Dispatcher

Class Name: Dispatcher	
Responsibility:	**Collaborators:**
Receive user input login credentials	SecurityController
Send login credentials to DataController for validation	DataController
Send login success or failure to SecurityController	SecurityController
Receive user input registration information	SecurityController
Send registration information to DataController for validation and processing	DataController
Send registration success or failure	SecurityController
Receive confirmed deregistration request	SettingsController
Send deregistration request to DataController for processing	DataController
Receive Scanned QR Code data	SecurityController
Send Scanned QR Code data to DataController for processing	DataController
Receive offer information	ShareController
Send offer information to DataController for processing	DataController
Receive user's location data	ShareController
Send location data for processing	DataController
Receive offer information from user	ShareController
Send offer information to DataController	DataController
Receive request information from user	ShareController
Send request information to DataController	DataController
Receive rating information from user	ShareController
Send rating information to DataController	DataController
Send request information to DataController for processing	DataController
Receive offer cancellation requests	ShareController
Send offer cancellation requests to DataController for processing	DataController
Receive Favourite Locations modifications from user	SettingsController
Send Favourite Locations modifications to DataController for processing	DataController
Receive Friends List modifications from user	SettingsController
Send Friends List modifications to DataController for processing	DataController
Receive Blacklist modifications from user	SettingsController
Send Blacklist modifications to DataController for processing	DataController
Receive password modifications from user	SettingsController
Send password modifications to DataController for processing	DataController

Class Name: OffersDatastore	
Responsibility:	**Collaborators:**
Store user offer information	DataController
Send user offer information to DataController	DataController
Store user request information	DataController
Send user request information to DataController	DataController

Class Name: QRCodeDatastore	
Responsibility:	**Collaborators:**
Send QR Code to DataController	DataController
Store QR Code information	

Class Name: UserRecordsDatastore	
Responsibility:	**Collaborators:**
Store user profile information	DataController

A Division of Labour

Due to the nature of this document, we worked collaboratively to complete each section. The sections built upon one another and, as such, we met regularly to complete the document as a team. Each group member made an equal contribution to the project. By signing below, we hereby acknowledge that the work was in fact divided equally.

Amanda Boudreau _____
Jordan Floyd _____
Mario Machado _____
Rakesh Mistry _____
Ali Reda _____