Introduction

1.1 Purpose

This document is intended to visually describe the system architecture of our proposed application. Its intended audience are software architects and the development team.

1.2 System Description

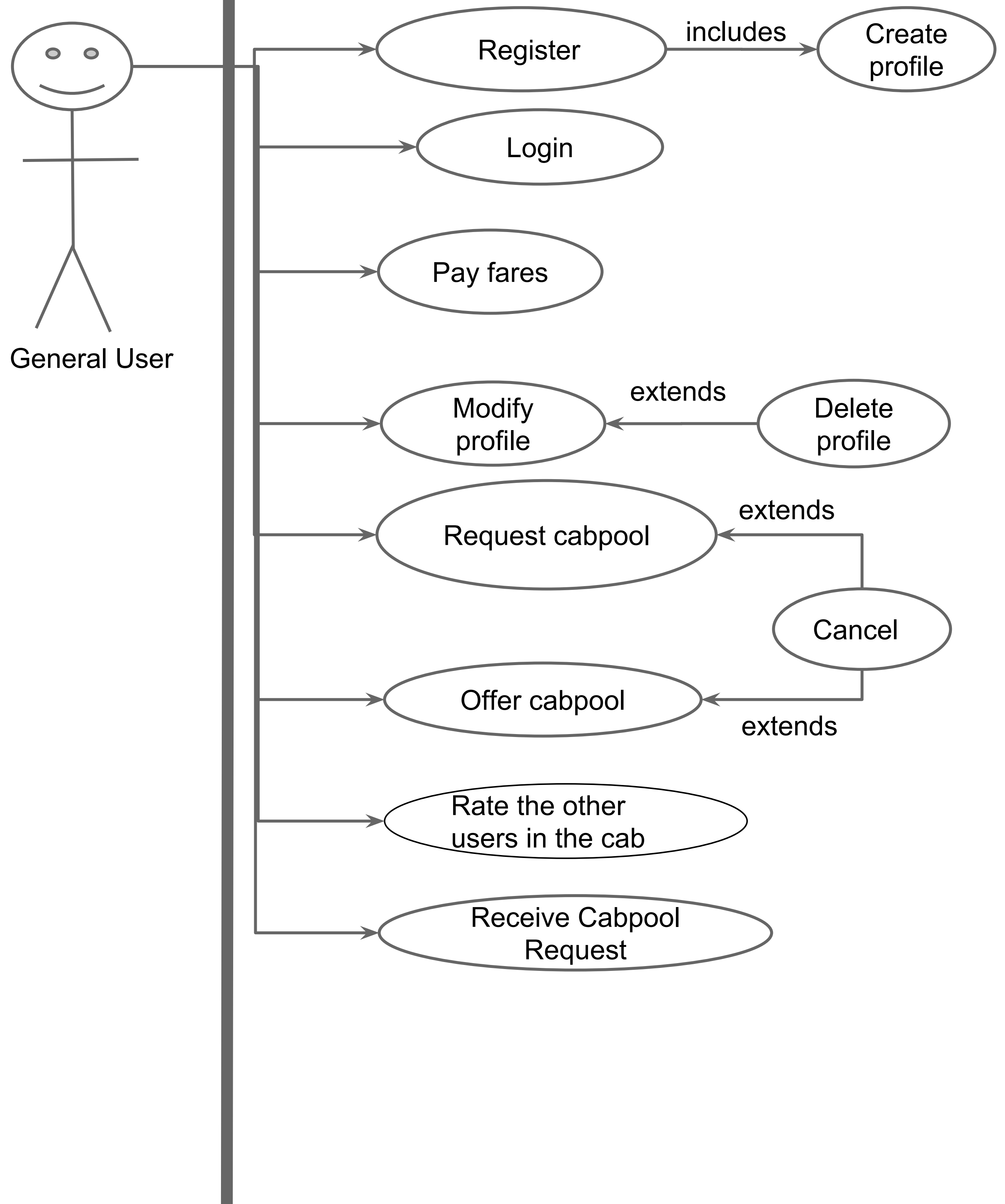
The system is designed to securely allow users to search for cabs with passengers willing to cabpool to a user's destination, or include a stop along the way to a further destination, and to allow users to offer a cab for cabpooling. The system will connect commuters and cabpoolers to lower individual fares while still providing a profit for the cab company.

1.3 Overview

Provided in this document is a description of the general architectural design which our system conforms to as well as justification for the use of said system architecture. It provides the system's main functionality within a use-case diagram, upon which an analysis class diagram is built. From this stems the system's architecture and class responsibility collaboration cards, outlining the classes with which each class collaborates as well as its responsibilities.

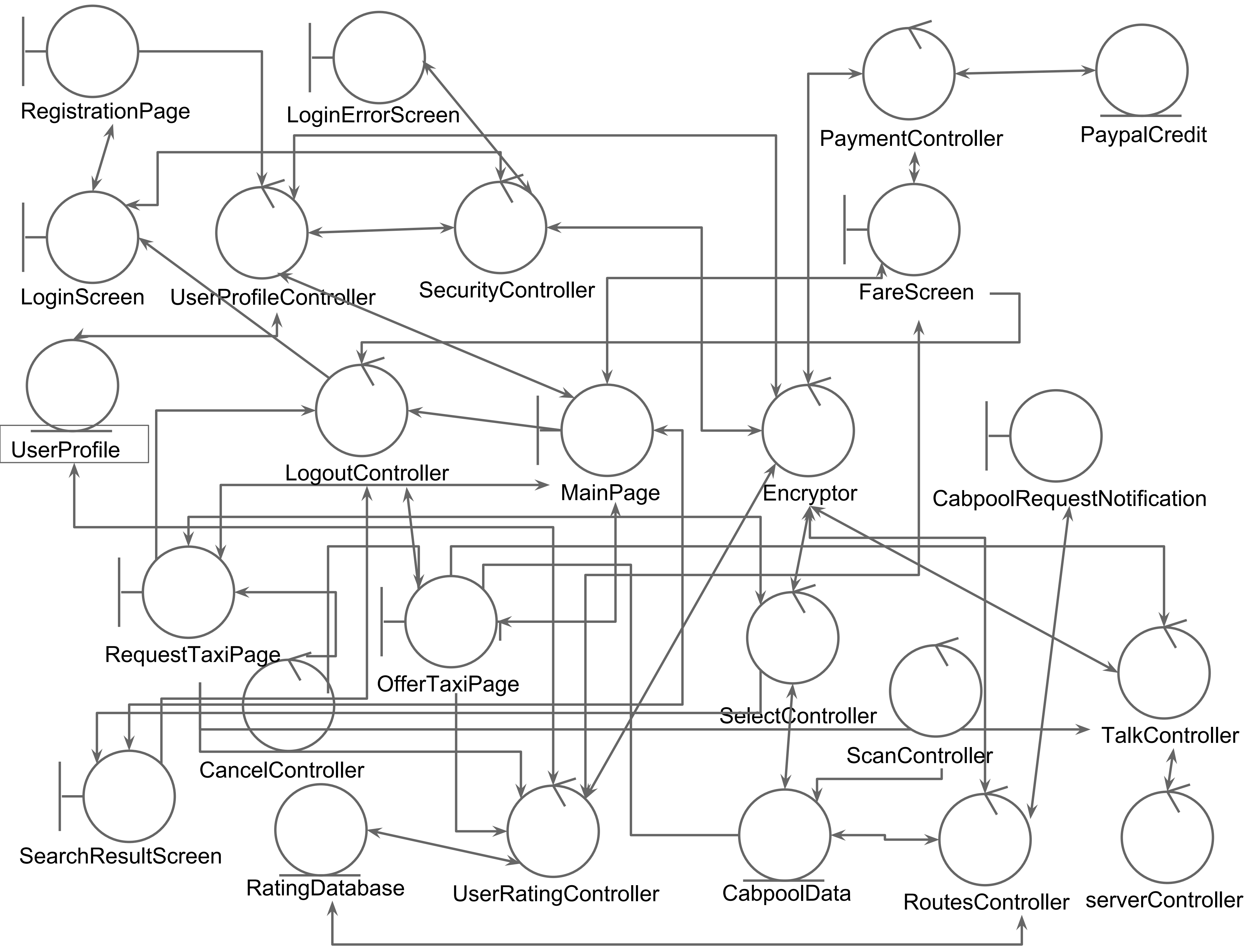
Revisions

Version	Date	Sections changed/added	Summary of changes
Version 0	whenever we handed this one in	Document created	N/A
Version 1	fri nov 14	talk data removed talk controller edited that other controller on slide 19 added	need for talkData no longer relevant so it was deleted upon looking into security features and communication, ServerController was added
version 1.1	tues nov 18	scanController	need to add a new controller for scanning the QR code
ver 1.2	fri nov 21st	use case descriptions added CRC collaborators	missing use case diagrams added missing collaborators added


General User
Register
includes
Create profile
Login
Pay fares
Modify profile
extends
Delete profile
Request cabpool
extends
Cancel
Offer cabpool
extends
Rate the other users in the cab
Receive Cabpool Request

Use Case Descriptions

1. Register
2. Create profile
   -
3. Login
   -
4. Pay fares
   -
5. Modify Profile
   -
6. Delete Profile
   -
7. Request Cabpool
   -
8. Cancel
   -
9. Offer Cabpool
   -
10. Rate other users in the cab
    -
11. Receive cabpool request
    -


RegistrationPage
LoginErrorScreen
PaymentController
PaypalCredit
LoginScreen
UserProfileController
SecurityController
FareScreen
UserProfile
LogoutController
MainPage
Encryptor
CabpoolRequestNotification
RequestTaxiPage
OfferTaxiPage
SelectController
TalkController
ScanController
CancelController
SearchResultScreen
RatingDatabase
UserRatingController
CabpoolData
RoutesController
serverController

System Architecture

4.1 System Architecture

The architecture which drives our application is Model-View-Controller (MVC) architecture, because the analysis diagram exhibits three distinct levels of interconnections between boundary classes, controller classes, and entity classes. The MVC architecture is specifically used in applications where user interfaces are prone to data changes. In our application each controller accepts, processes and passes data and can be regarded as the combination of Model and Controller. Each boundary class displays the actual view that users interact with. Whenever the data is changed through the controllers, the view will be changed accordingly. View modules and controller modules are separate so that it allows for division of labor. All the data generated through communications is encrypted and stored into databases.

Sub System Architecture

Login

The login subsystem uses SecurtityController and UserProfileController to check whether the input password and username in the system and password is correct, as the input password is encrypted. Once the user information is verified, UserProfileController will go to MainPage in order to make the users begin their session. If the user information is not authenticated, the SecurityController will shows the LoginErrorPage to the users.

Registration

The registration subsystem asks the user to create an account and edit their profile in order to log in the system, the UserProfileController allows the users to edit their profile, and some of the information users edit would be encrypted by Encryptor. Finally, all the user information will be stored at UserProfile. Once the registration is done the user will be automatically logged in with no necessity to run through the standard login process.

Payment

The payment subsystem contains PaypalController, PaypalCredit and FareScreen. It allows the user to pay the calculated fees by using PayPal as a payment method. It does this by first having the user in the FareScreen boundary class, where the user can view information. When the user wants to make a payment the information is transfered to the PaypalController where the information is process and any saved information is encrypted and sent to the PaypalCredit for storage.

Offer

The Offer subsystem is used by the user to offer one cabpool. It contains the OfferTaxiPage as interface. And this subsystem uses RoutesController to set the specific location and destination for the user in order to offer the cabpool, and these information will be stored in the CabpoolData. After matching the CabpoolData information and request from the other users. The offeror may receive the CabpoolNotification from the other users to share one taxi, all information are encrypted with Encryptor.

Request

The Request subsystem is used to request cabpool which provides convenient routes for the users. The users uses RequestTaxiPage to choose the taxi, the SelectController will proceed to selections,the SelectController will match the requests with data from CabpoolData and give the best choices for the users. Then the SelectController shows user SearchResultScreen, which tell the users which cabshare is available and suitable, all information are encrypted with Encryptor.

Talk

This subsystem is used to communicate between users. The subsystem uses TalkController to control communications between users, and it uses Encryptor to encrypt the communication as well. All the information of communications are stored and updated in the Talkdata database.

Rating

The Rating subsystem allows the users to rate the users who share one taxi each other. this component uses UserRatingController to rate the other users as the rating information is encrypted with the Encryptor. And finally, the rating information will be stored and updated in the Ratingdatabase.

Core system

The core system contains all of the portions of the system which are consistently used by other subsystems. It contains the Encryptor which prevents any information in the system being stored to be as is or not protected by some level of cryptography.   The MainPage is a central viewing hub where all other pages can be reached and all pages can return to. The final portion of the Core system is the LogoutController which allows you from any page exempting registration and login to jump to the login page effectively signing out of your account. All of the classes of the core system are extensively used by other subsystems for the reasons already stated above.

Class name: RegistrationPage	
Responsibilites:	Collaborators:
Receive login data from user	LoginScreen
send authentication request	SecurityController
receive authentication	UserController

Class name: OfferTaxiPage	
Responsibilites:	Collaborators:
use destination of user to offer route	SelectController
See potential cabpoolers	UserRatingController
waiting for potential requests	MainPage
Allow users to log out	LogoutController
Allow unsaved chat between cabpoolers and commuters	TalkController

Class name: LogoutController	
Responsibilites:	Collaborators:
confirm logout action with user	
lock the app	
redisplay LoginScreen	LoginScreen

Class name: LoginScreen	
Responsibilites:	Collaborators:
send user credentials for authentication	SecurityController
receive authentication or rejection	SecurityController
unlock app if authentication received	
count number of consecutive unsuccessful login attempts	
Disable the account for 5 minutes if unsuccessful login attempts exceeds the limit	SecurityController

Class name: RequestTaxiPage	
Responsibilites:	Collaborators:
obtain GPS location data of user	phone’s GPS, Google maps
obtain search criteria from user	User Rating Controller
send search request	SelectController
allow return to the main page of the application	MainPage
allow user to logout	LogoutController

Class name: SearchResultScreen	
Responsibilites:	Collaborators:
receive results of search as set on RequestTaxiPage	SelectController
display data to user	MainPage
allow alternate ordering of data according to destination, wait times, fare saved	LogoutController
display active users in each cab result returned	

Class name: RatingDatabase	
Responsibilites:	Collaborators:
send user ratings when search results are returned	UserRatingController
receive new ratings after ride completion	UserRatingController
incorporate new rating with stored data	
store user ratings	
update user rating	RouteController

Class name: CabpoolData	
Responsibilites:	Collaborators:
receive QR code	zebracrossing library to interface with the camera
Use QR code to match routes	SelectController
store users associated with each cab	
store destination of each cab	
send relevant cabpool data upon search request	RoutesController

Class name: UserProfile	
Responsibilites:	Collaborators:
store user information	
grant permission for users to edit own profiles	UserProfileController
display user public data	
store user ratings and comments	
allow access to ratings to display to other cabpooler and commuters	UserRatingController

Class name: LoginErrorScreen	
Responsibilites:	Collaborators:
receive failed authentication	SecurityController
keep application locked	
ask user if they ‘forgot password’	
ask for user email	
send reset password link to given email	SecurityController

Class name: UserProfileController	
Responsibilites:	Collaborators:
gather relevant user rating data to display with returned search results	RatingDatabase, MainPage
send&receive users’ profile information to the encryptor	Encryptor

Class name: SelectController	
Responsibilites:	Collaborators:
gather relevant cabpool data to display with returned search results	CabpoolData
makes edits to cabpool data regarding which routes are still available	CabpoolData
send&receive users’ selection information to the encryptor	Encryptor

Class name: UserRatingController	
Responsibilites:	Collaborators:
gather relevant user rating data, of users in cab, to display with returned search results	
send&receive user’s rating data to the encryptor	Encryptor

Class name: SecurityController	
Responsibilites:	Collaborators:
verify the password	
encrypt password	Encryptor
determine whether to display LoginErrorScreen	LoginErrorScreen

Class name: MainPage	
Responsibilites:	Collaborators:
choose to offer cabshare	OfferTaxiPage
choose to request cabshare	RequestTaxiPage
Users can edit their own profile	UserProfileController
	LogoutController

Class name: FareScreen	
Responsibilites:	Collaborators:
offer methods of payment to user	userprofile, paypal, bank
	MainPage
	LogoutController

Class name: RoutesController	
Responsibilites:	Collaborators:
controls data of the cabpoolers’ ratings and cabpooler’s data to include as a search	RatingDatabase,CabpoolData
Calculate the distance between the commuter and cabpooler	CabpoolRequestNotification
encrypt the routes information	Encryptor

Class name: CabpoolRequestNotification	
Responsibilites:	Collaborators:
receive acceptance of a search result from user	
display Cabpool request to commuter	
allow user to accept or reject the request	
display rating of the requesting user	

Class name: PaymentController	
Responsibilites:	Collaborators:
receive payment information from user	
send payment information from user to be encrypted	Encryptor
receive encrypted payment information	
store payment information	
verify encrypted payment information	Bank

Class name: PaypalCredit	
Responsibilites:	Collaborators:
store payment information	
update payment information	
return to PaymentController	PaymentController

Class name: TalkController	
Responsibilites:	Collaborators:
send the communication	
receive the communication	
communication should be encrypted	Encryptor

Class name: Encryptor	
Responsibilites:	Collaborators:
encrypt all communication between each user	TalkController
encrypt payment information	PaymentController
encrypt specific location data of each user to ensure privacy	
encrypt login information of user to protect theft of payment information	SecurityController

Class name: ServerController	
Responsibilites:	Collaborators:
send the communication to cabpool users	talkController
receive the communication from cabpool users	
communication should be encrypted	Encryptor

Class name: ScanController	
Responsibilites:	Collaborators:
read the barcode	
send the barcode data to cabpool	cabpooData

Division of Labour

Mitchell Spector: Class diagrams, use cases, Architecture and OO analysis

Yuchen Xiao: CRCs, class diagrams, System Architecture

Xue Lin: class diagrams, use case, subsystem analysis

Okay everyone just put the things here by yourself :)

Kemal Ahmed	
Sahajmeet Bhutta	
Xue Lin	
Mitchell Spector	
Yuchen Xiao	