The University of Hong Kong

Faculty of Engineering

Department of Computer Science

COMP7704
Dissertation Title
A Smart Phone Application for Valet Parking

Submitted in partial fulfillment of the requirements for the admission to the degree of Master of Science in Computer Science

By
LUO Xianyang
3035237420

Dr. T.W. Chim
Date of submission: 30 / 11 / 2016

Abstract

Declaration of Candidate

I, the undersigned, hereby declare that the work contained in this thesis is my own original work, and has not previously in its entirely or in part been submitted at any university for a degree.

Only the source cited in the document has been used in this draft. Parts that are direct quotes or paraphrases are identified as such.

I agree that my work is published, in particular that the word is presented to third parties for inspection or copies of the work are made to pass on to third parties.

The University of Hong Kong,

LUO Xianyang

Acknowledgments

I would like to thank~

Table of Contents

lalala

Chapter 1. INTRODUCTION

1.1. Background Information

Nowadays a lot of people will go to clubs for dinners, after-work drinks or having fun with friends. Since Hong Kong is an International and fast-tempo city, it is rather common for citizens to go to places like Lan Kwai Fong after a day's work or at weekend. It is convenient for customers to drive their own cars to the hotels, clubs or bars. But as a matter of fact that Hong Kong is one of the most crowded cites in the world, it is not easy for drivers to find a parking lot quickly.

Valet parking service can help customers park their cars. It is offered by some restaurants, shopping malls, clubs and so on. A person called valet will drive a customer's car to parking lot when the customer arrives at the gate of the hotel and return the car when the customer leaves. The main advantage of valet parking is convenience. On one point, customers do not need to find a parking lot by themselves. On the other point, they do not have to walk a long way from the parking

lot to the hotel, which saves lots of time. All they need to do is just dropping their cars at drop-off point.

However, in such a fast-tempo city and such a high-tech era, some problems of traditional valet parking for a customer are:

1) must use a valet parking ticket to redeem their cars. If the ticket is lost, customer need to prove that the car belongs to him or her by showing driver license or identity card
2) may do not know where is the drop-off point for a certain hotel, restaurant or clubs and may take time to find it
3) do not know current status of the car

To solve these problems, I would like to develop a mobile application for people in Hong Kong to use valet parking service easier and more efficient.

1.2. Project Description

This project mainly focuses on help drivers enjoy better valet parking service. A customer can register our service via its phone number. After adding a new car by inputting plate, brand and color, he or she can choose a parking lot and generate a QR code. When a valet scans

the QR code successfully, the order will be generated and the customer can use the phone as a valet ticket. Whenever the customer wants to get the car back, he or she can just click the “recall” button. Then our valets will return the car back to the customer.

This project has three parts and functions are as follows

1) iOS version application for customer
    a. allows a user to register, login and reset password via phone number
    b. allows a user to add a new car, edit an existing car and delete a car. Information like plate, brand and color are requested
    c. allows a user to generate order by choosing parking lot and car
    d. allows a user to get the car back by just clicking a button
    e. allows a user to view all the historical orders
2) iOS version application for valet
    a. allows a valet to login and reset password via phone number
    b. allows a valet to add an order by scanning the QR code generating by a user

c. allows a valet to view all the opening orders

d. allows a valet to end an order by just clicking a button

3) server and database

a. processes all the http request and sends a proper response back to phone

b. stores data safely of all users, valets, cars and orders sends notification to a user then an order has been generated and ended successfully

1.3. Project Objectives

Since the traditional valet parking service has matured, so the app need to be more attractive to gain users. We have to follow the current trend of design, follow the guideline of user interface design and take some of them into consideration to fulfill the goal, which is Shneiderman's Golden Rules of Interface Design[1], Jun Gong's Guideline for Mobile Application[2] and Nurul's Threes Layers Design Guideline for Mobile Application[3].

Apart from user interface, the app should be rather easy to use. Users do not to do lots of setting or follow a complicate guideline to generate an order. There are some mobile applications in the market with

similar purpose like Meibo, Youbo and so on. After analyzing those applications, I found out that they were not that easy to use. Since there are a new technology in iOS called 3D touch which brings a new powerful dimension to Multi-Touch interface, users can enjoy the best convenience while parking their cars. Also, it is rather innovative if there is an Apple Watch application cooperating with application on the iPhone.

The final goal of this project is to change the traditional valet parking service by attractive and innovative features so our objectives can be conducted in three aspects:

1) attractability: to attract users, the application should have friendly and beautiful user interface, reasonable layout and overall clean look.
2) innovation: users are more willing to use the application by using new technology like 3D touch. This project allows user to enter the "parking now" and "current orders" views from the icon which makes it rather convenient and enjoyable.
3) connectivity: a user may enjoy more convenience if he or she has an Apple Watch. The all the parking and recall request can be

done through Apple Watch. A user do not even need to take his or her phone out the pocket.

1.4. Summary of Chapters

Chapter 1 firstly introduces the background and motivation of this project and then gives brief introduction and objectives of this project

Chapter 2 firstly introduces two similar mobile applications in the market and analyze the advantages and disadvantages.

Chapter 3 design blaalala

Chapter 4 implementation balalallala

Chapter 5 conclusion balalaalal

Chapter 2. RELATED WORK

To develop a successful application, we need to refer to the existing applications and see if how we can do it better.

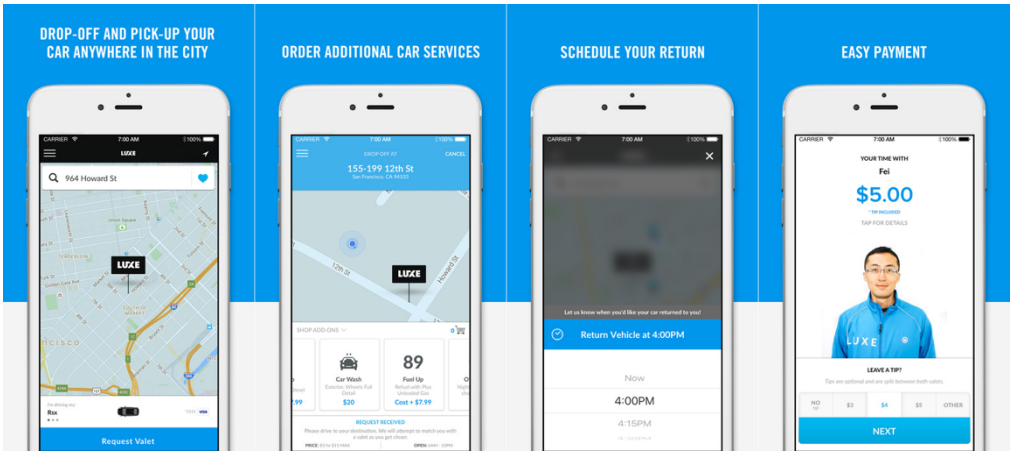
In this part, I will introduce two mobile applications offering valet parking service in the market. The first one is Luxe, which is available in American and then Youbo in China. I will firstly give a brief introduction and analyze the advantages and disadvantages of both applications respectively.

2.1. Luxe

http://www.luxe.com/

Luxe is a valet parking app available on both iOS and android. It is currently available in San Francisco, New York, Chicago, Seattle, Austin and Log Angeles. Figure 2-1 shows demo pictures for Luxe.

Figure 2-1 iPhone Screenshots for Luxe



After trying Luxe a few days, I found the advantages and disadvantages as below:

Advanages:

1) the user can drop off and gets returned anywhere and a valet will wait at the drop point. This is the best experience for a valet parking service. The user saves time since he or she does not need to find a certain drop-off point
2) the user can pay for the service using its phone, which makes it a Uber-like application. All the process can be done by the application.
3) the user interface is attractive and modern

Disadvantages:

1) the application does not use new technology like 3D touch and does not offer Apple Watch application

2) the register process is rather complicated, which requires both email account and phone number

3) all those service are only available in the USA

2.2. Youbo

http://www.uboche.com/

Youbo is a valet parking app available on both iOS and android. It is currently available in Beijing, Shanghai, Chengdu and other big cities in mainland China. Figure 2-2 shows demo pictures for Youbo

Figure 2-2 iPhone Screenshots for Youbo

Youbo offers similar services as for Luxe. After taking a real experience, I found the advantages and disadvantages below:

Advantages:

1) a valet can drive the customer to the destination and then park the car. And the customer can get the car returned at wherever he or she wants. This is rather similar to Luxe
2) a valet could help wash and refuel a customer's car
3) a user can register and login just using its phone number

Disadvantages

1) the application does not use new technology like 3D touch and does not offer Apple Watch application
2) the user interface does not look good.

Chapter 3. BACKGROUND KNOWLEDGE

There are a lot of third party frameworks to enhance the performance or to optimize the user interface of this project. This chapter mainly introduces some useful frameworks in the development. The first part talks about tools used in front-end which is the mobile application. And the second part talks about tools in back-end which is the server and database.

3.1. Tools used in Front-End

3.1.1. AFNetworking

AFNetworking is networking library used in iOS and Mac OS X development. It is built on top of the Foundation URL Loading System, extending the powerful high-level networking abstractions built into Cocoa. It is a high efficient networking module along with feature rich API which is rather easy to use. It powers some of most popular applications on iPhone and iPad.

The usage of AFNetworking is simple, which differs from the origin method offered in iOS. After initializing a session manager, the user just needs to configure some parameters, sends the request and then waits for the response. A simple post request can be implemented below:

```
AFURLSessionManager *manager = [AFURLSessionManager manager];
NSDictionary *parameters = @{@"foo": @"bar"};
[manager POST:@"http://example.com/resources.json"
success:^(AFHTTPRequestOperation *operation, id responseObject) {
parameters:parameters
    NSLog(@"JSON: %@", responseObject);
} failure:^(AFHTTPRequestOperation *operation, NSError *error) {
    NSLog(@"Error: %@", error);
}];
```

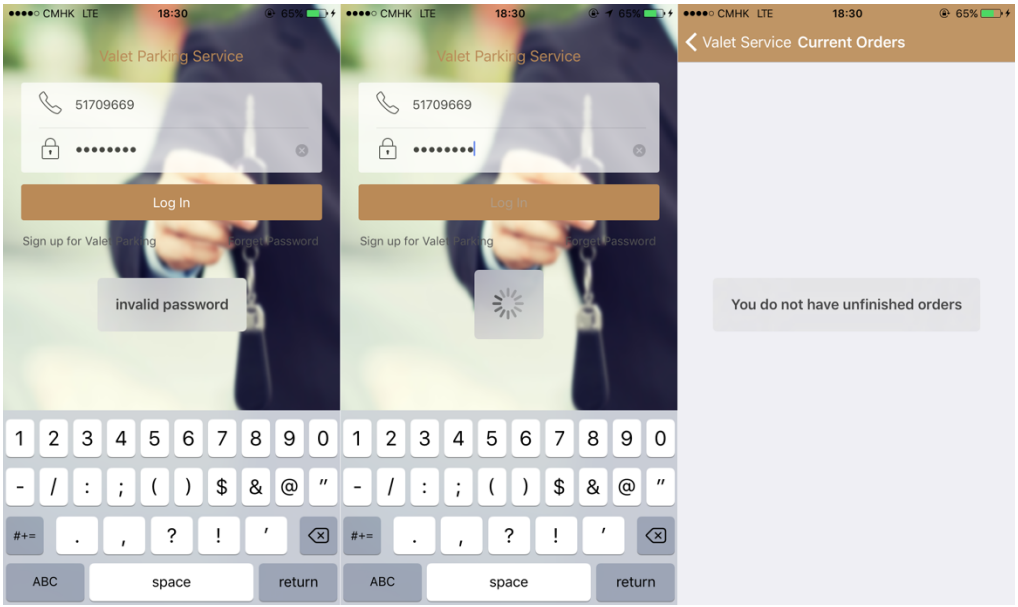
This code block sends an asynchronous request containing a parameter dictionary to the server. If the request is successful, the manager will get a *responseObject* containing request information or an *error* if the request is failed.

3.1.2. MBProgressHUD

MBProgressHUD is an iOS drop-in class that displays a translucent HUD with an indicator and/or labels while work is being done in a background thread. I use this framework in almost every view in this

project. It's easy to add a text indicator or progress indicator as shown in Figure 3-1

Figure 3-1 Screenshots for usage of MBProgressHUD



The usage MBProgressHUD is very simple by initializing an MBProgressHUD instance and show it in current view. By choosing the mode of a hud, it can display to show text or process indicator or both. The code block below shows how to add a hud to current view and hide until a certain process is finished.

```
[MBProgressHUD showHUDAddedTo:self.view animated:YES];
dispatch_time_t popTime = dispatch_time(DISPATCH_TIME_NOW, 0.01 *
NSEC_PER_SEC);
dispatch_after(popTime, dispatch_get_main_queue(), ^(void){
   // Do something...
   [MBProgressHUD hideHUDForView:self.view animated:YES];
});
```

3.1.3. JSONModel

MVC (Model, View, Controller) is widely used in object-oriented developing. A model is used to store and manage data. And the format of most data downloaded from server is JSON. In order to use this data, front-end needs to transfer it to model. JSONModel allows rapid creation of smart data models. By creating an model and inheriting it from JSONModel, the front-end can use the *initWithDictionary* method to generate instance of a model rapidly without reading each key-value from response data. The code block below shows an statement of class named *CountryModel* which is inherited from JSONModel. It has three attributes. As long as the JSON data has the same key, the application can generate a model using one line of code.

```
#import "JSONModel.h"
@interface CountryModel : JSONModel
@property (assign, nonatomic) int id;
@property (strong, nonatomic) NSString* country;
@property (assign, nonatomic) BOOL isInEurope;
@end
```

3.1.4. SMS_SDK

SMS_SDK is a framework used to send verification code to users' mobile phones. It's totally free and easy to be integrated in the application. Since our valets host customers' cars, it is important for valets to verify the identity of the user. The code block below shows how to use SMS_SDK to verify a user's phone number.

```
[SMSSDK getVerificationCodeByMethod:SMSGetCodeMethodSMS
```

```
                phoneNumber:@"51709669"
                       zone:@"852"
              customIdentifier:nil
                     result:nil];
```

3.2. Tools used in Back-End

3.2.1. MongoDB

MongoDB is a free and open-source cross-platform document-oriented database program. Being different from database like MySQL, MongoDB is classified as NoSQL database program and uses JSON-like documents with schemas. It works well with Node.js and can implement common database method like create, read, update and delete rather easily and efficiently.

Compared to MySQL, MongoDB has some main advantages as follows:

1) rich data model, dynamic schema, typed data and data locality
2) rich query language which is easier to code and read
3) development is simplified as MongoDB documents map naturally to modern, object-oriented programming languages

3.2.2. Express

Express is a fast, unopinionated and minimalist web framework that provides a robust set of features for web and mobile applications. It is the most popular framework in Node.js as for now. Because it provides

lots of powerful and useful features including organizing application's routing and providing template solutions, the user can focus on developing the functions of applications. The main advantages of express are as follows:

1) event driven: this feature enables registering various profound functionalities by connecting them to other events that are being executed once the event is triggered.
2) javascript closures: this feature allow the user to use variables that are defined in the outer calling function inside the callback body, which is extremely useful to solve the conflict while working with various applications.

Chapter 4. REQUIREMENTS AND DESIGN

In this part ~~~~~~

4.1. Requirements Analysis

4.1.1. Product Perspective

This project has front-end and back-end. The front-end is built on Xcode 8 using Objecive-C and the back-end is built with Node.js and MongoDB. And there are two applications in the front-end. One is for customers and the other one is for valets. Both customers and valets use their phone number as login account.

4.1.2. User perspective

There are two types of user – customer and valet. After a customer logging in, he or she can add a car, choose a place to park, check orders and request his or her car. After a valet logging in, he or she can access to all the opening orders, generate an order by scan the QR code on a customer's phone and end an order after returning a customer's car.

4.1.3. Functional Requirements

Using this product, a customer will be able to choose a place and choose a car to park. Here is the list of high-level functional requirements that this project is focus on. The two apps and the back-end must:

1) Authenticate and authorize a customer or a valet
2) Store a number of data records describing cars. Each record will have attributes as below:
    - Car plate
    - Car brand
    - Car color
3) Store a number of data records describing orders. Each record will have attributes below:
    - Order date and time (create, update, end)
    - Order place
    - Customer name
    - Customer phone
    - Car plate
4) Allow customers to add, edit a car by typing in plate, brand and color. Also allow customers to delete a car.

5) Allow customers to choose a place and a car to park
6) Allow customers to check all current orders and historical orders
7) Allow customers to request his or her car back
8) Allow valets to generate an order by scanning QR code
9) Allow valets to access all the opening orders
10) Allow valets to end an order

Next I will introduce the details of design.

4.2. Design

4.2.1. System Architecture

The entire system can be divided into two parts: front-end and back-end. Back-end is a server based on Node.js with MongoDB as database. All the *get* and *post* requests including registering an account, logging in, adding a car, editing a car, deleting a car, creating an order, updating an order are handled by this server. It will check every request for correctness and validity and send corresponding response back. And front-end are two mobile applications built using Xcode and Objective-C. The database used in front-end is CoreData which is native in iOS and easy to use. Figure 3-1 shows the architecture of the system

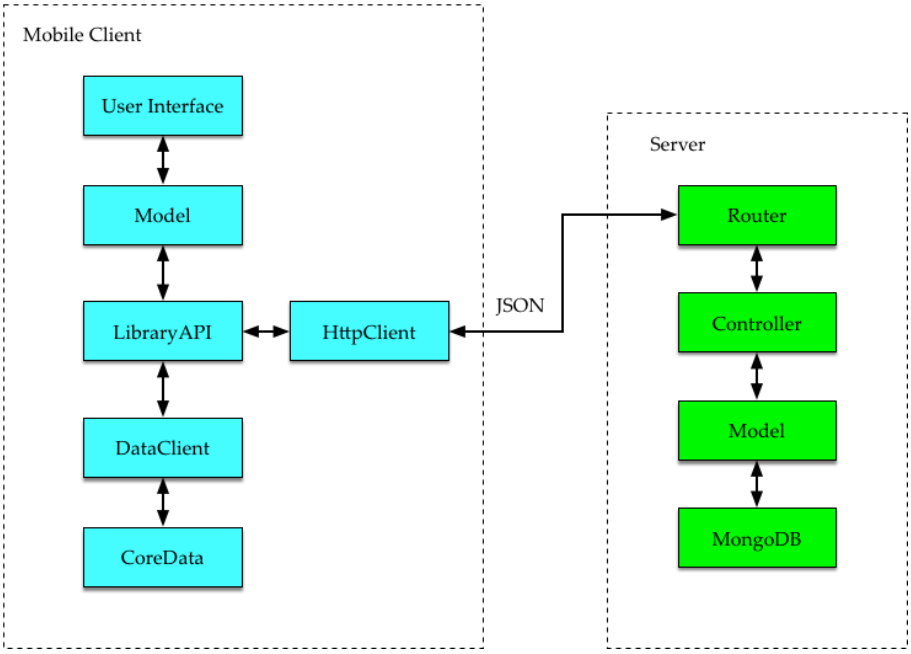


Figure 4-1 Architecture of System

4.2.2. System Workflow

The system workflow consists of two parts. The first part is the work flow in the server and the second part is the work flow in the mobile client. These two parts will be introduced separately.

4.2.2.1. Workflows in Server

Server works as back-end in this project and will handle all the http request sent from front-end. There are three major management in server: account management, car management and order management.

1) Flow chart for account management

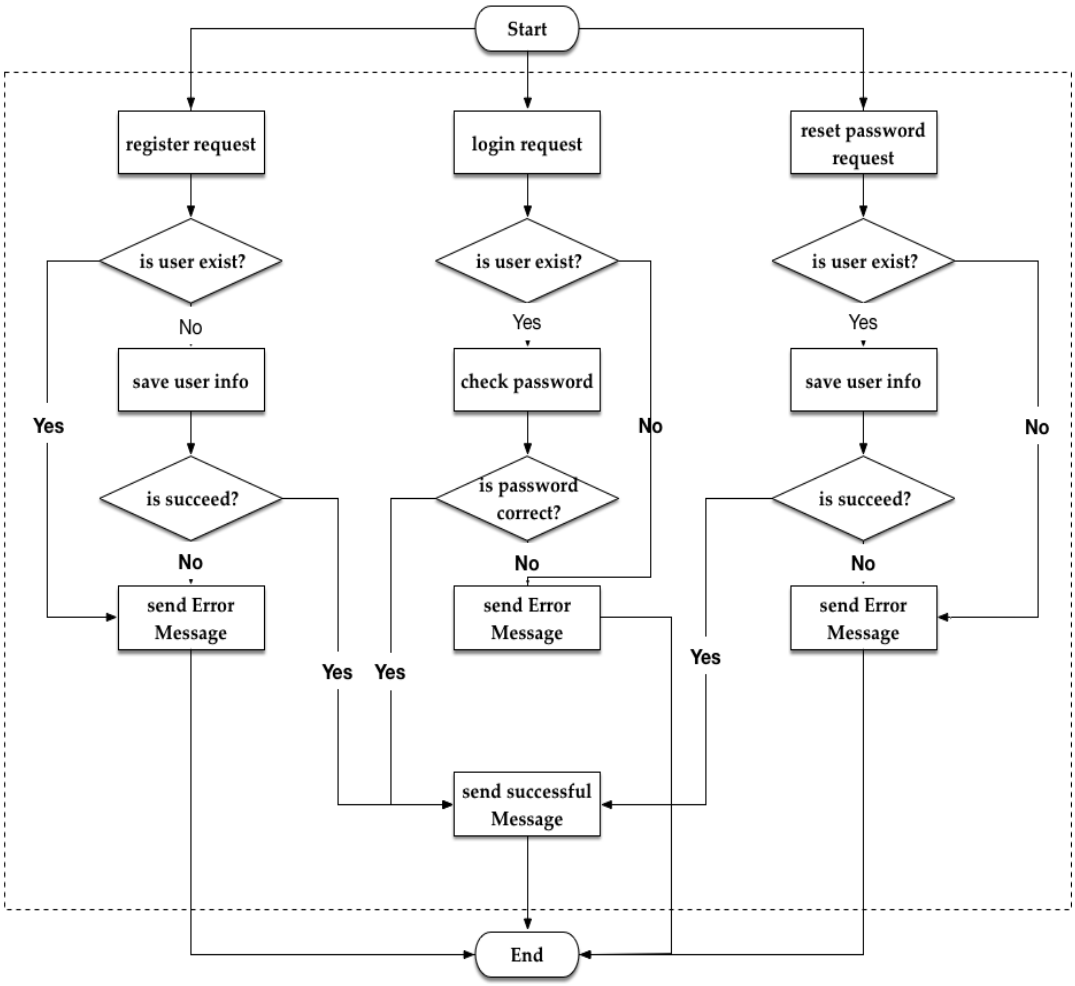


figure 4-2 flow chart for account management

As shown in figure 3-2, server needs to deal with three kinds of account management request: register request, login request and reset password request. For register request, if there exist an account with the same phone number or fail to save user file, the server will send error message. For login request, if it does not find the user with the phone number or the password is incorrect, the server will send error message. For reset password request, if it does not find the user with the phone number or fail to save new password, the server will send error message.

For other circumstance, the server will send successful message with the user model.

2) Flow chart for car management

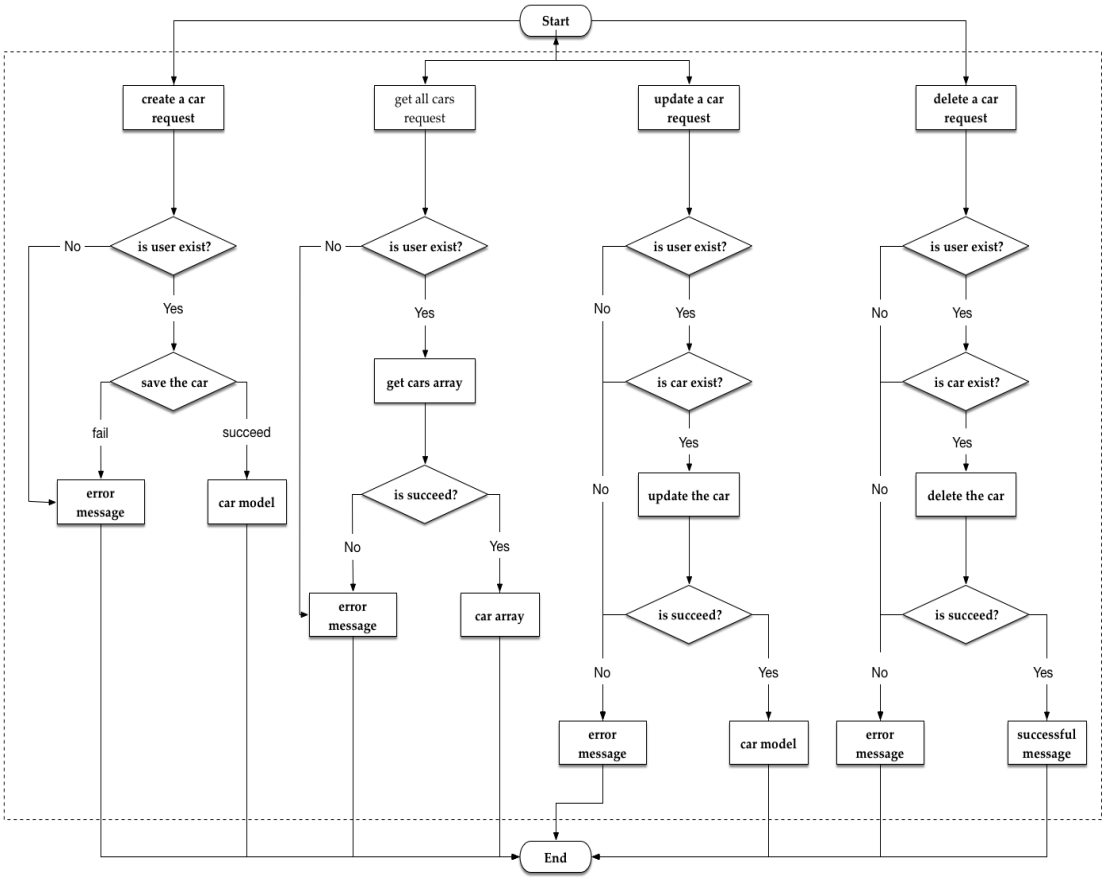


figure 4-3 flow chart for car management

As shown in figure 3-3, the server needs to deal with four kinds of car management request: create, update, delete a car and get all cars for a user. For all these four requests, server will firstly check if the user exists. If user's account can not be found, an error message will be sent to front-end. If the user's account is valid, for "create a car request, server will save this car to database. For "get all cars

request", server will get all the car models with corresponding user account from database. For "update a car request" and "delete a car request", server will check if the car exists. If the car can not be found, an error message will be sent to front-end. If the car is valid, then corresponding database method will be called. If there is any error when processing the data in the database, an error message will be sent to the front-end.

3) Flow chart for order management

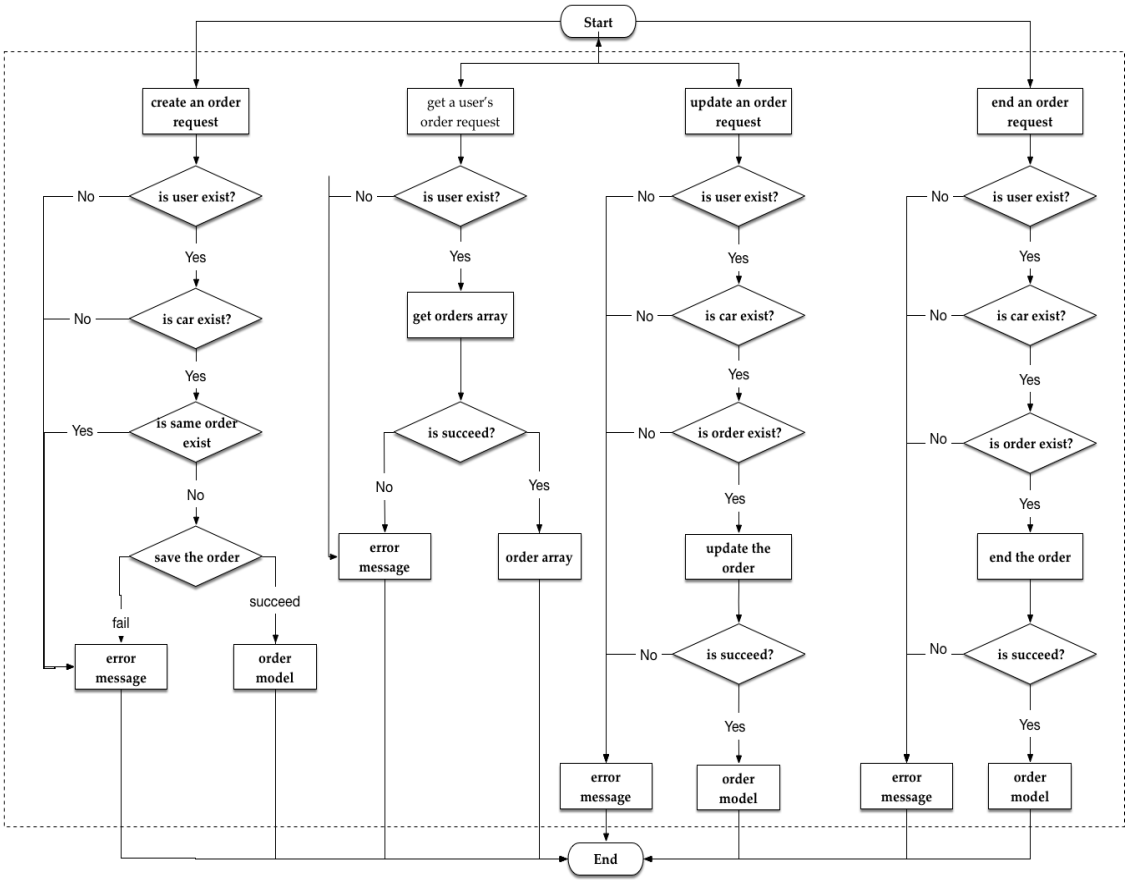


figure 4-4 flow chart for order management

As shown in figure 3-4, the server needs to deal with three kinds of order management request: create, update, read and end. For all these four requests, server will firstly check if the user exists. If user's account can not be found, an error message will be sent to front-end. If the user's account is valid, for "create an order request", server will check if the car exist and if there is an order with the same car unfinished. For "get a user's order request", server will get all the orders belong to the user and send them back. For "update an order request" and "end an order request", server will check if the car and the order exist. If not, an error message will be sent back. If the order is valid, server will update or end that order and send the order model back.

4.2.2.2. Workflows in Mobile Client

There are two major managements in front-end. One is account management and the other one is car and order management. They are less complex than that in server. Figure 3-5 shows the flow chart for mobile client.

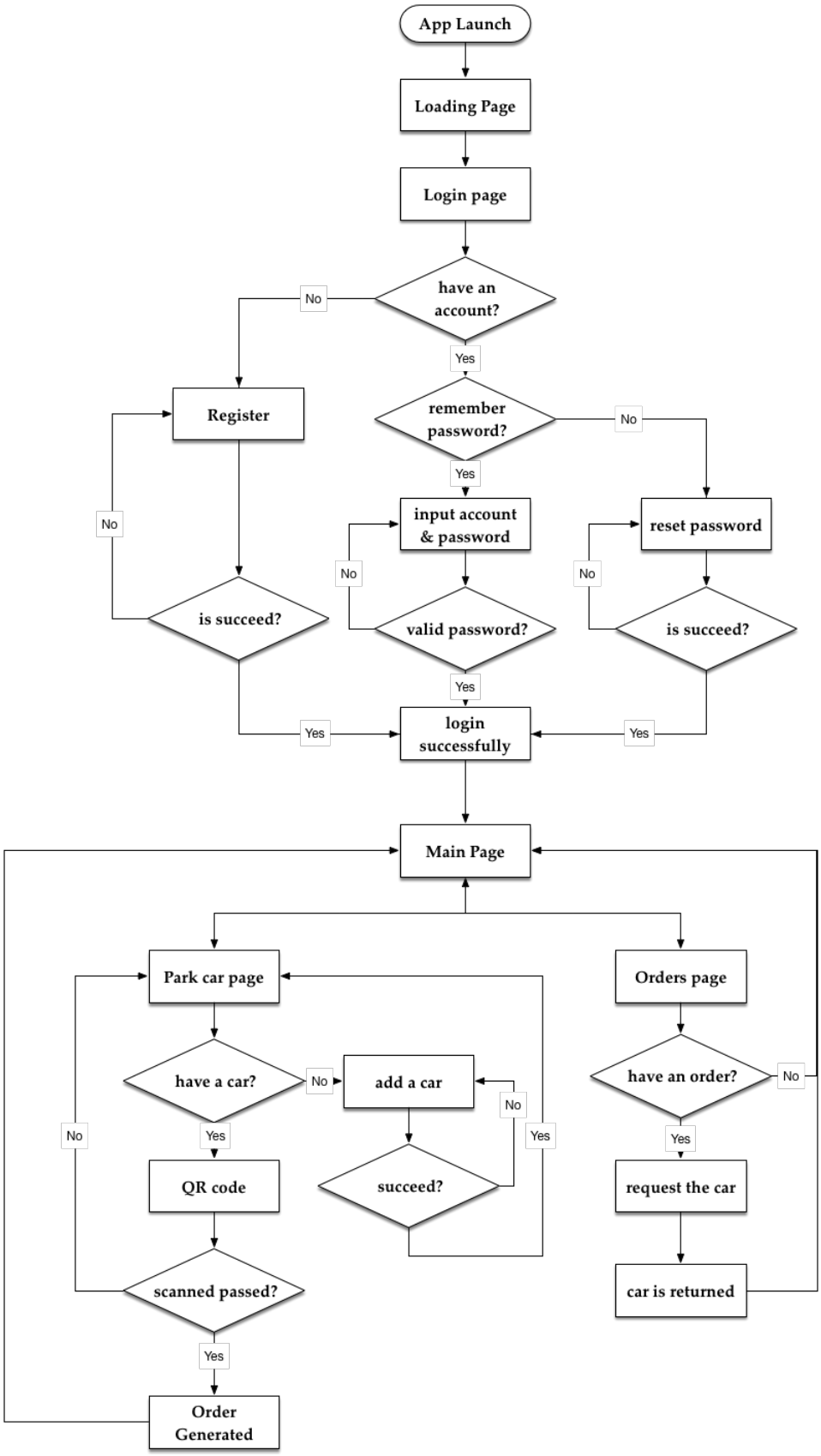


figure 4-5 flow chart for mobile client

1) Account management

   When the application is launched at the first time, it will ask the user to input account and password to login. If the user does not have an account, he or she needs to register using its phone to get a verification code. If the user forgets the password, he or she needs to set a new password using its phone to get verification code. The server will do all the verify works including check if the password is valid, if the account exists and so on.

   If the account and password are valid, the *login* view will dismiss and the main page will show. If not, it will ask the user to input account and password again.

2) Car and order management

   In the main view, a user can park its car. But if the user does not create a car in the application, he or she needs to create a car by inputting information including plate, brand and color. The server will check if a car with the same plate exist. Then the user can choose a place to park its car and a QR code will show on the screen for our valet to scan. After some checking on the

server, if all the information is valid, a new order will be generated. The application will go back to the main page.

Whenever the user wants its car back, he or she can enter the order page, choose an order and request that car. Out valet will start to drive its car back.

4.2.3. Object Oriented Design

Below I will introduce the object oriented design in this project.

4.2.3.1. Identify actors and use cases

Table 4-1 shows the actors and use cases for this project

Table 4-1 table for actors and use cases

Actor	Use Case	Use Case Description
Customer	Login	Customer logs in
Customer	Register	Customer registers the service
Customer	Reset password	Customer resets its password
Customer	Modify a car	Customer can add, update or delete a car
Customer, valet	Park a car	Customer parks its car

4.2.3.2. Use case diagrams

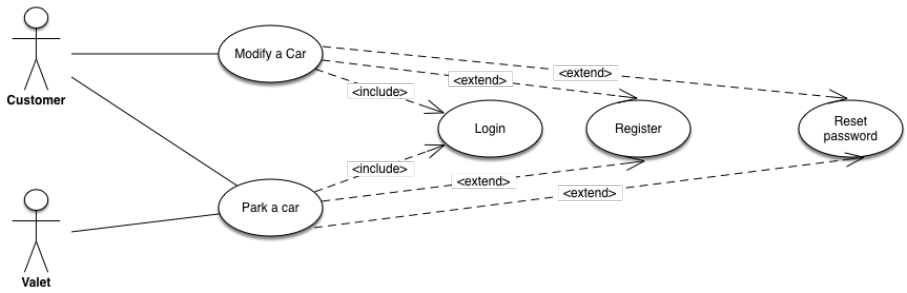


figure 4-6 use case diagram

Figure 3-6 shows the use case diagram. As it shows in the picture, there are three use cases.

1) For the “modify a car” use case, a customer can modify its cars including create, update and delete a car.
2) For the “park a car” use case, a customer can park its car if the order is generated successfully by a valet. And the user can request its car if the car is in a parking lot. Then a valet will return the car.
3) Both the use cases above need to include “login” use case.
4) If the user does not have an account, the “register” use case will be invoked.

4.2.3.3. Sequence diagrams

1) Figure 3-7 shows the sequence diagram for register and login. As shown in the figure, a customer need to register first to use our service. In this sequence diagram, all the message are asynchronous.

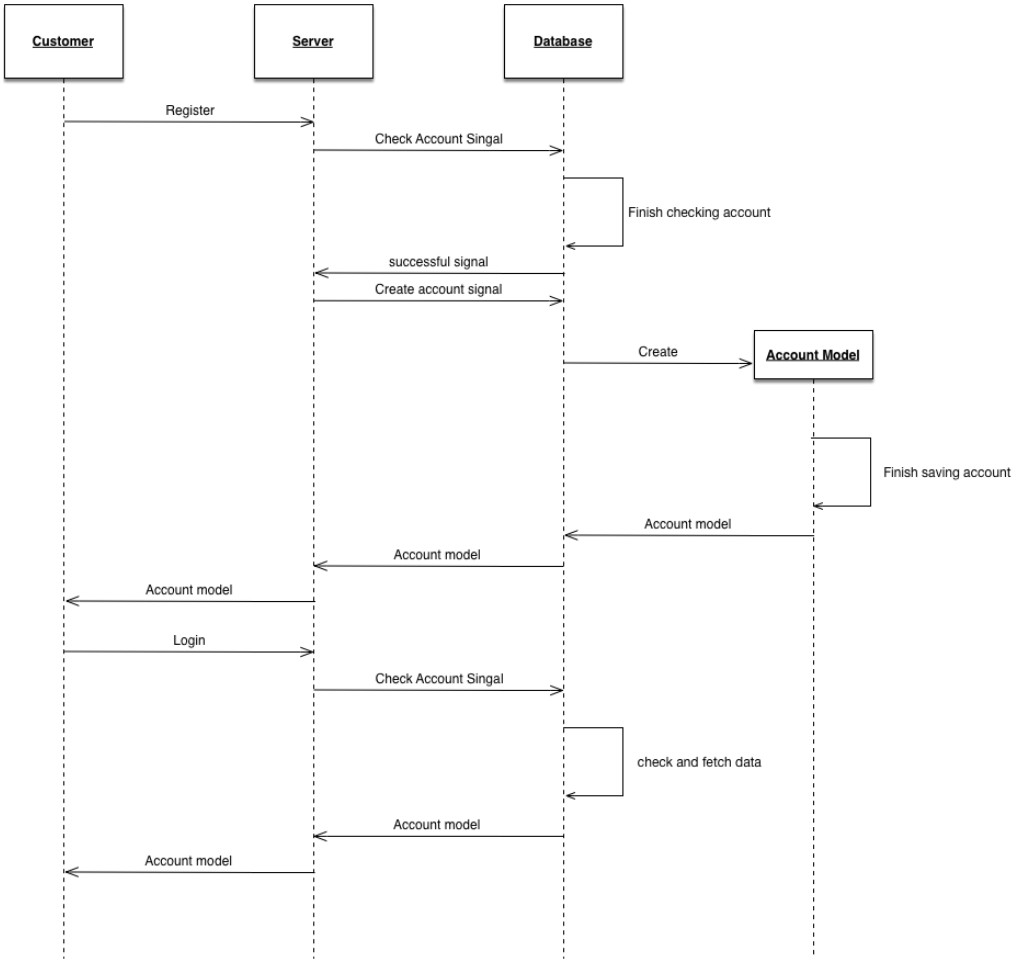


figure 4-7 sequence diagram for registration and logging in.

2) Figure 3-8 shows the sequence diagram for parking a car. As shown in the figure, a customer needs to add a car before using our valet parking service. And it is a valet who sends a an order request to server after scanning the QR code generated by a customer. The server will check each request including “add a car” and “create an order” for correctness and validity. Then the server will communicate with database and send back the corresponding model object.

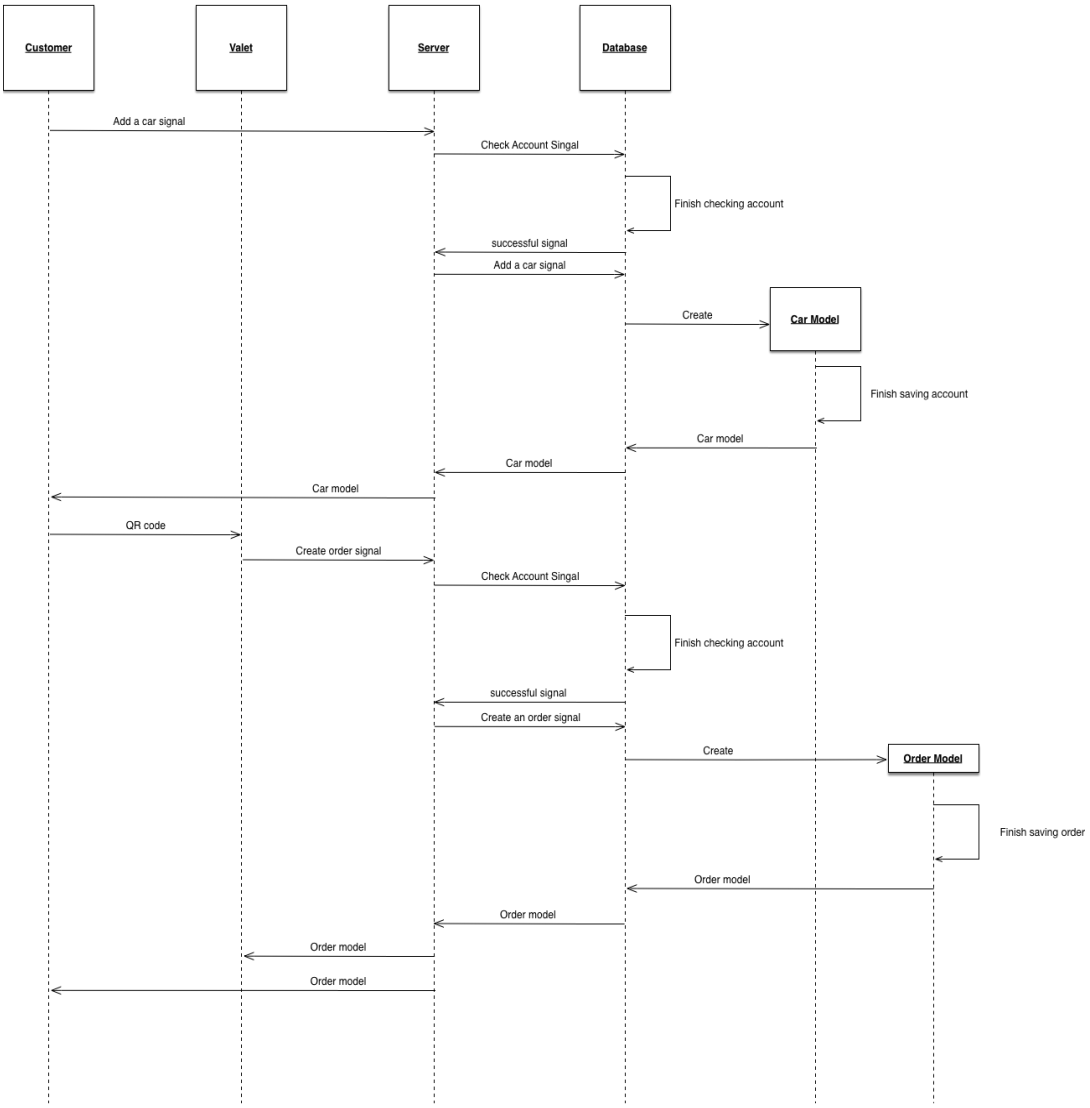


figure 4-8 sequence diagram for parking a car

4.2.3.4. Documents for design use cases

Below introduce the documents for the four use cases.

1) "Register" use case

Use Case Name:	Register
Actor(s):	Customer
Description:	This use case describes the membership registration process.

Reference:	FS-1	
Typical Courses of Events:	**Actor Action**	**System Response**
	The welcome window is currently displayed on the screen waiting for the customer to select an option.	
	Step1: Initiate this use case if a customer clicks the [Sign up for Valet Parking] button.	
		Step2: The system pops up a multi-lined window requesting the following information to be input: First Name, Last Name, Phone Number, Verification Code and Password. The window has one buttons at the bottom: [Sign up], one button at the left top corner: [Cancel] and a [Get verification code] button besides the text field for verification code. And the customer can read Terms by clicking the [Terms] button.
	Step3: The customer inputs First Name, Last Name and Phone Number then clicks the [Get Verification Code] button.	
		Step4: The system sends the phone number to Mob to get a verification code. If the format of phone number is correct, the customer will get an message with verification code

	Step 5: The customer inputs the verification code and a password then clicks the [Sign up] button.	
		Step 6: A progress indicator will display on the screen. The system sends all the information to the server to verify if the phone number is valid, if so, the server will create a user model and save all the information in the server. If the saving process is successful, the server will send the user model back to mobile client.
		Step 7: this use case concludes when the mobile client receives a successful message and the user model. The registration view will dismiss and the main page will show.
Alternative(s):	Step3: if the customer clicks [Cancel] button on the left top corner, the registration view will dismiss and the welcome page will show. This use case concludes. Step4: if the phone number or verification code is not correct, customer needs to correct the number or get verification code again. This use case goes back to Step3. Step6: if the account with same phone number already exist, the server will send back an error message and the mobile client will display a MBProgressHUD showing “This account already exists”. Then this use case goes back to step3. If server can not save the information, it will send back an error message and the mobile client will display a MBProgressHUD	

	showing “Fail to Register”. Then this use case goes back to step3.
Precondition:	None
Postcondition:	User successfully registers to use our service
Assumptions:	None

2) “Login” use case

Use Case Name:	Login	
Actor(s):	Customer	
Description:	This use case describes the login process for a user	
Reference:	FS-1	
Typical Courses of Events:	**Actor Action**	**System Response**
	Step1: Initiate this use case when a customer opens the application for the first time or if the user has logged out.	
		Step2: The system shows a welcome view with two text field requesting the user to input: Phone Number (account number) and Password. This view has three buttons: [Log in], [Sign up for Valet Parking] and [Forget Password]
	Step3: The customer inputs Phone Number and Password then clicks the [Log in] button.	
		Step4: A progress indicator will display on the screen. The mobile client sends customer’s phone number and

		password to the server for verification.
		Step 5: the server checks if the phone number and password are valid. Is so, the corresponding user model will be fetched from database and sent back to mobile client.
		Step 6: this use case concludes when the mobile client gets the response and dismisses the welcome view to show the main page.
Alternative(s):	Step3: if the customer clicks [Sign up for Valet Parking] button, a registration view will pop up. Then it goes to use case "Register". If the customer clicks [Forget Password] button, a view for resetting password will pop up. Then it goes to use case "Reset password". Step4: if the phone number and password are invalid. Server will send back an error message indicating invalid password. And the mobile client will display a MBProgressHUD showing "invalid password". Then the customer needs to correct its information. This use case goes to Step 3.	
Precondition:	This customer already has an account	
Postcondition:	None	
Assumptions:	None	

3) Reset password use case

Use Case Name:	Reset password	
Actor(s):	Customer	
Description:	This use case describes the reset password process for a user	
Reference:	FS-1	
	Actor Action	**System Response**

Typical Courses of Events:	The welcome window is currently displayed on the screen waiting for the customer to select an option.	
	Step1: Initiate this use case if a customer clicks the [Forget Password] button.	
		Step2: The system pops up a multi-lined window requesting the following information to be input: Phone Number, Verification Code and a new Password. The window has one buttons at the bottom: [OK], one button at the left top corner: [Cancel] and a [Get verification code] button besides the text field for verification code.
	Step3: The customer inputs Phone Number then clicks the [Get Verification Code] button.	
		Step4: The system sends the phone number to Mob to get a verification code. If the format of phone number is correct, the customer will get an message with verification code
	Step 5: The customer inputs the verification code and a password then clicks the [OK] button.	

		Step 6: A progress indicator will display on the screen. The system sends all the information to the server to verify if the phone number is valid, if so, the server will fetch the corresponding user model and update the password information in the server. If the saving process is successful, the server will send the user model back to mobile client.
		Step 7: this use case concludes when the mobile client receives a successful message and the user model. The reset password view will dismiss and the main page will show.
Alternative(s):	Step3: if the customer clicks [Cancel] button on the left top corner, the registration view will dismiss and the welcome page will show. This use case concludes. Step4: if the phone number or verification code is incorrect, customer needs to correct the number or get verification code again. This use case goes back to Step3. Step6: if the account is not found, the server will send back an error message and the mobile client will display a MBProgressHUD showing “Account not found”. Then this use case goes back to step3. If server can not save the information, it will send back an error message and the mobile client will display a MBProgressHUD showing “Fail to reset password”. Then this use case goes back to step3.	
Precondition:	None	
Postcondition:	User successfully resets its password	
Assumptions:	User already has registered for this service.	

4) Modify a car

Use Case Name:	Modify a car	
Actor(s):	Customer	
Description:	This use case describes the process of modifying a car	
Reference:	FS-1	
Typical Courses of Events: **Alternative(s):**	**Actor Action**	**System Response**
	The user's cars collection view is currently displayed on the screen waiting for the customer to select an option.	
	Step1: Initiate this use case if a customer clicks a cell or the [add] button on the right top of the view.	
		Step 2: if the customer wants to add a new car. A [Add a car] view will pop up with three information for customer to input: Car Plate, Car Brand and Car Color. If the customer wants to edit an existing car, a [Edit a car] view will pop up with plate, brand, color already filled. There are two buttons on this page, on is [Cancel] on the top left corner and the other one in [Done] on the top right corner.
	Step 3: the customer needs to type in the plate, choose brand and color from the system. And then click the [Done] button.	

		Step 4: the mobile client will firstly check if the user has a car with same plate locally, if not, it will display a progress indicator and send the car model to server.
		Step 5: the server will firstly check if the account is valid, if so, it will save the car model to the database. If the saving process is successful, it will send a successful message with the car model to the mobile client.
		Step 6: the mobile client gets the message and save the car model locally. This use case concludes when the saving process is finished.
	Step3: if the customer clicks [Cancel] button on the left top corner, this view will dismiss and cars collection view will show. This use case concludes. Step4: if the mobile client finds out that the user adds a duplicate car, a MBProgressHUD will display showing "You already has this car". Then the customer needs to change the plate. This use case goes back to step 3. Step6: if the account is not found, the server will send back an error message and the mobile client will display a MBProgressHUD showing "Account not found". Then this use case goes back to step3. If server can not save the information, it will send back an error message and the mobile client will display a MBProgressHUD showing "Fail to save this car". Then this use case goes back to step3.	
Precondition:	None	
Postcondition:	User successfully update a car	
Assumptions:	User already has an account	

5) Park a car

Use Case Name:	Par a car	
Actor(s):	Customer, Valet	
Description:	This use case describes the process of parking and returning a car	
Reference:	FS-1	
Typical Courses of Events:	**Actor Action**	**System Response**
	The main page is currently displayed on the screen waiting for the customer to select an option. This view is a table view with two options: [Park Now] and [Current Orders]	
	Step1: Initiate this use case if a customer clicks [Park Now].	
		Step 2: A [Park] view will show. This is a table view with two sections. The first section has four rows: Place, Name, Phone and Car. If the customer already adds at least one car, the lasted added car will be displayed. The section section is a [Confirm] button. And there is a [Back] button on the top left corner.
	Step 3: the customer chooses which car to park and clicks [Confirm].	
		Step 4: a [Placing an Order] view shows with a label and a

		QR code. The label tells the customer to show the QR code to a valet.
	Step 5: a valet scans the QR code using its application.	
		Step 6: the mobile client of valet sends the order information to server. The server will firstly check if the account and the car are valid. If so, it will check if the user has an unfinished order using the same plate. If not, the server will create a new order model and save it to database. If the saving process is successful, the server will send a successful message with the order model to mobile client.
		Step 7: the mobile client of valet goes back to main page and refresh the table to show the latest order.
	Step 8: the customer wants to get its car back. He or she clicks the [Current Orders]	
		Step 9: a [Current orders] view shows displaying all unfinished orders of this user. There are kinds of unfinished order: the one in parking lot and the one with a valet.
	Step 10: the customer clicks an order with status “in parking lot”	

		Step 11: a [Status] view shows displaying a table view showing order information. There are two sections, the first section show plate, brand and color. The second section is a [Recall] button.
	Step 12: the customer clicks the [Recall] button.	
		Step 13: the mobile client of customer sends the request to server and server will update the order status. Then server will send the new order object to customer and notify all valets that a user requests it car. The [Status] view will pop out and [Current Orders] view will show again with new car status. For mobile client of valets, the main page will refresh with new car status.
	Step 14: after the car returning the customer, a valet clicks the finished order cell	
		Step 15: a [Order] view shows with order information: customer's name, car plate. And there are a [End] button.
	Step 16: a valet clicks the [End] button	
		Step 17: the mobile client sends an end order request to server. The server will check

		if the customer, car and order are valid. If so, it will update the order send a new order model to all valets. This use case concludes.
Alternative(s):	Step2: if the customer has not added a car, an [Add a car] view will pop up. The system goes to use case [Modify a car]. Step3: if the customer clicks [Back] button on the left top corner, this view will dismiss and main view will show. This use case concludes. Step6: if the server finds that the account or car or order information is wrong, it will send an error message to valet telling the reason. The customer needs to correct its information. This use case goes back to step 3.	
Precondition:	None	
Postcondition:	User gets its car back	
Assumptions:	User already has an account	

4.2.3.5. Class diagrams

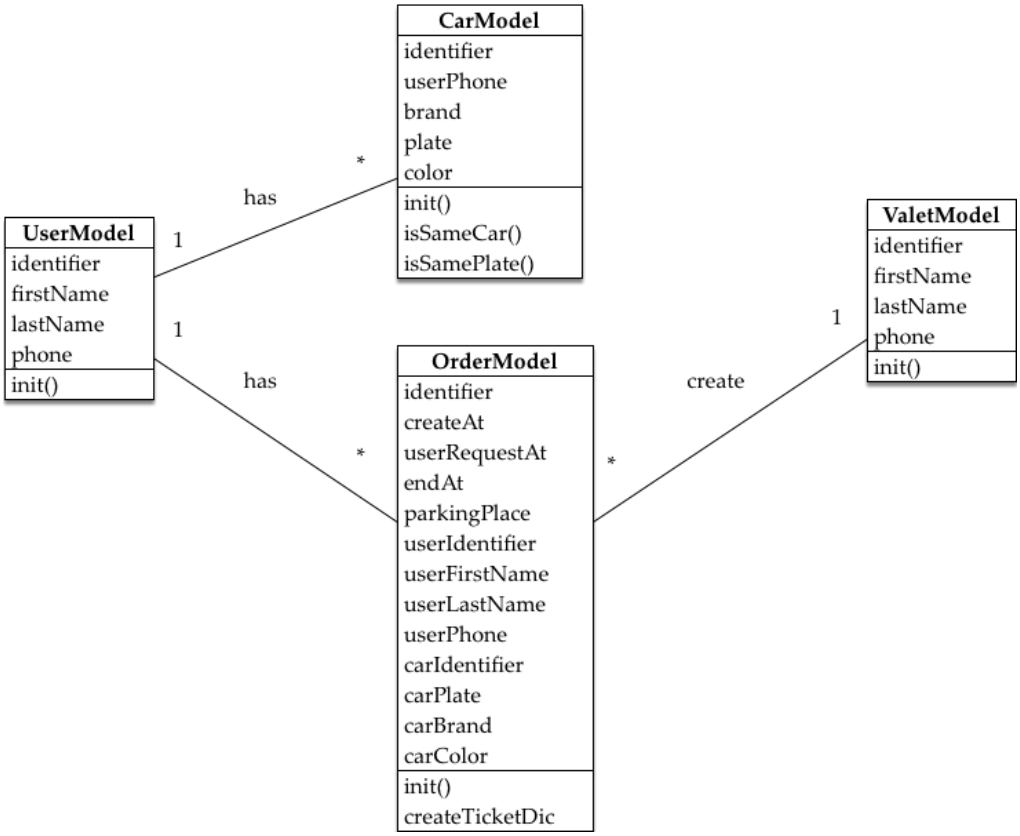


figure 4-9 class diagram

As shown in figure 4-9, the class diagram shows the relationship between different class. The *UserModel* indicates the customer. It stores identifier, first name, last name and phone of a customer. The *CarModel* stores identifier, customer's phone, brand, plate and color for a car. The *OrderModel* stores identifier, creating time, requesting time, ending time, user's identifier, user's first name, user's last name, user's phone, car's identifier, car's plate, car's brand and car's color. A *UserModel* may have multiple *CarModel* and *OrderModel*.

The *ValetModel* indicates the valet. It stores identifier, first name, last name and phone of a valet. A *ValetModel* may create multiple *OrdreModel*.

4.2.4. User Interface Design

User interface (UI) is important because a good UI will attract a user at first glimpse and keep that user. In this part, I will introduce the UI design of this project. There are mobile applications and the UI design will be introduced separately.

4.2.4.1. UI design for Customers

1) *Login/ Register/ Reset Password Pages*

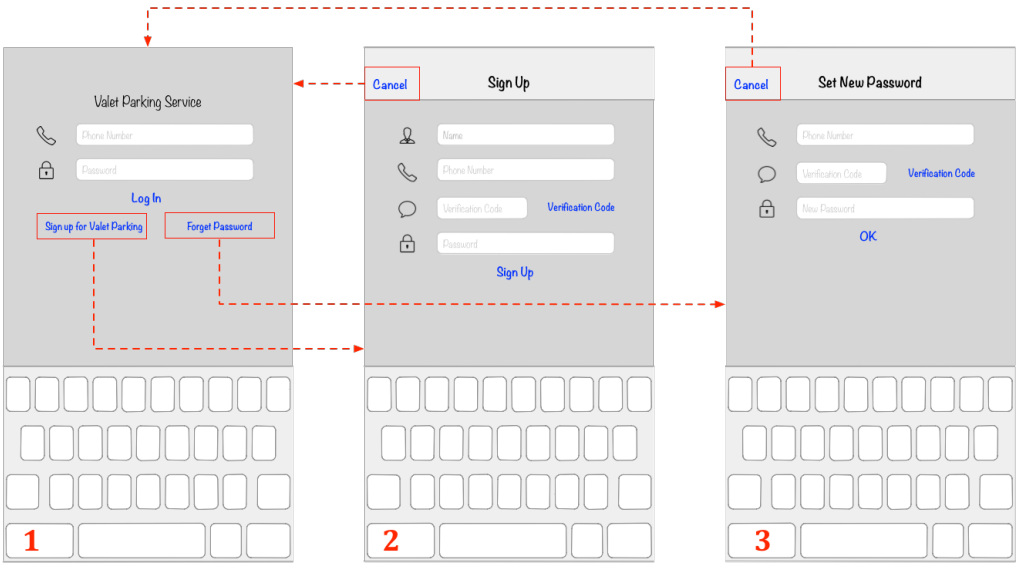


figure 4-10 UI design of *Login/ Register/ Reset Password Pages*

As shown in figure 4-10, there are three pages to display before a customer has logged in: *Login Page, Register Page* and *Reset Password Page*.

- *Login Page*: this page shows when the application if opened for the first time. Customers can login on this page by inputting phone number and password the clicking [Log In] button.
- *Register Page*: this page shows when a customer clicks [Sign up for Valet Parking] on the *Login Page.* Customers can register on this page by inputting all the information including a verification code.
- *Reset Password Page*: this page shows when a customer clicks [Forget Password] button on the *Login Page*. Customers can reset their password on this page by inputting all the information including a verification code.

2) Main function pages

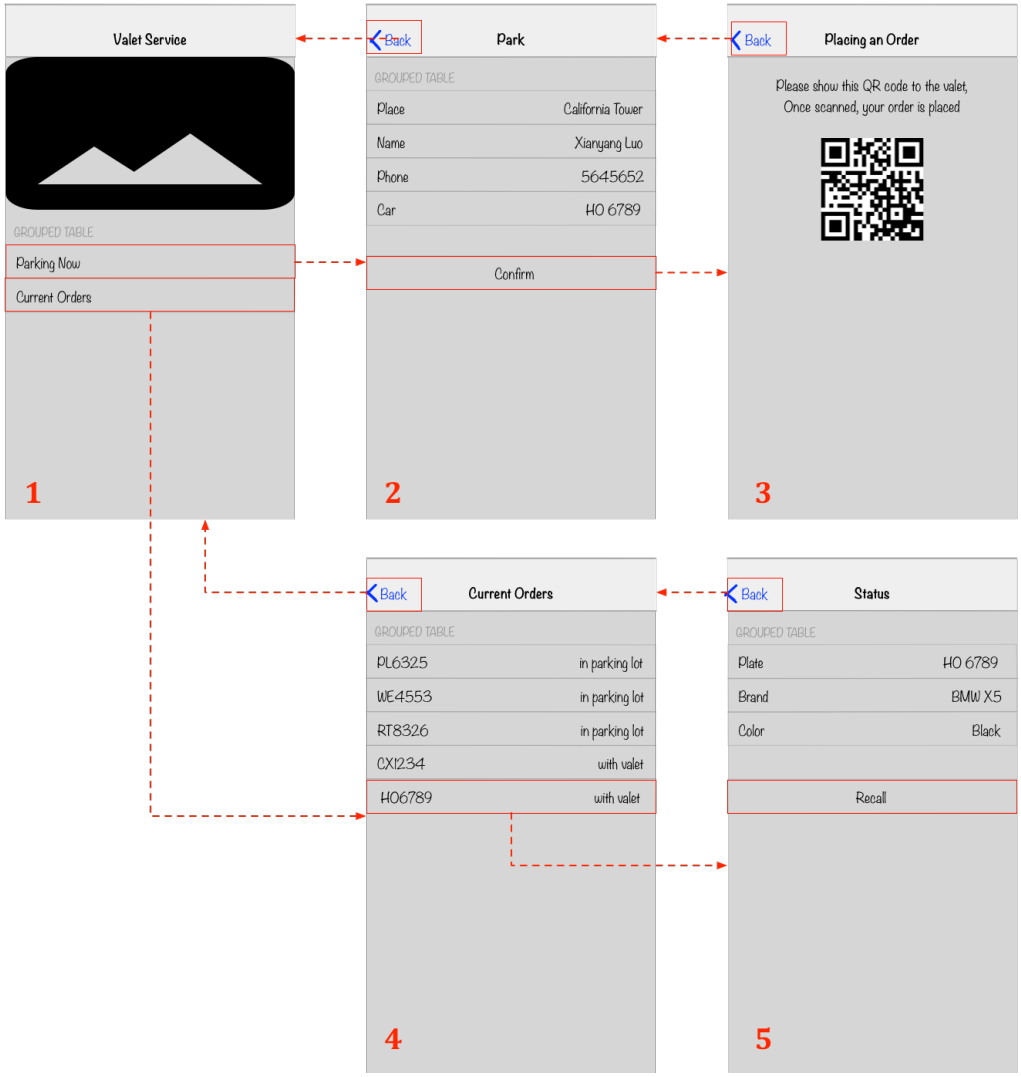


figure 4- 11 UI design of Main Functions Pages

As shown in figure 4-11, there are five pages containing main functions of this application. They are *Main Page, Park Park Page, Placing an order Page, Current Orders Page* and *Order Status Page.*

- *Main Page*: this page shows when a customer logs in. There are two parts in this page. The first part is a scrollable bar displaying images. The second part is a table view with two cells: *"Park Now"* and *"Current Orders"*.

- *Park Page*: this page shows when a customer clicks “Park Now” cell on the *Main Page*. There is a table view with two sections in this page. The first section is four rows for customer to type in information. The second section is a “*Confirm”* button.
- *Placing an order Page:* this page shows when a customer clicks “*Confirm”* button on the *“Park Page”*. This page has a label telling customer to show the QR code to a valid. The QR code below the label contains basic information for this order.
- *Current Orders Page:* this page shows when a customer clicks “*Current Orders”* in the “*Main Page”*. There is a table view in this page displaying all the unfinished orders for the customer.
- *Order Status Page:* this page shows when a customer clicks an order in *“Current Orders”* page. There is a table view with two sections in this page. The first section displays the basic information including car plate, car brand and car color. If order status is “in parking lot”, the second section is a button showing “Recall”. Customer can click this button to recall its

car. If the order status is “returning with valet”, the second section is a label displaying “returning with valet”.

3) Car Collection Pages

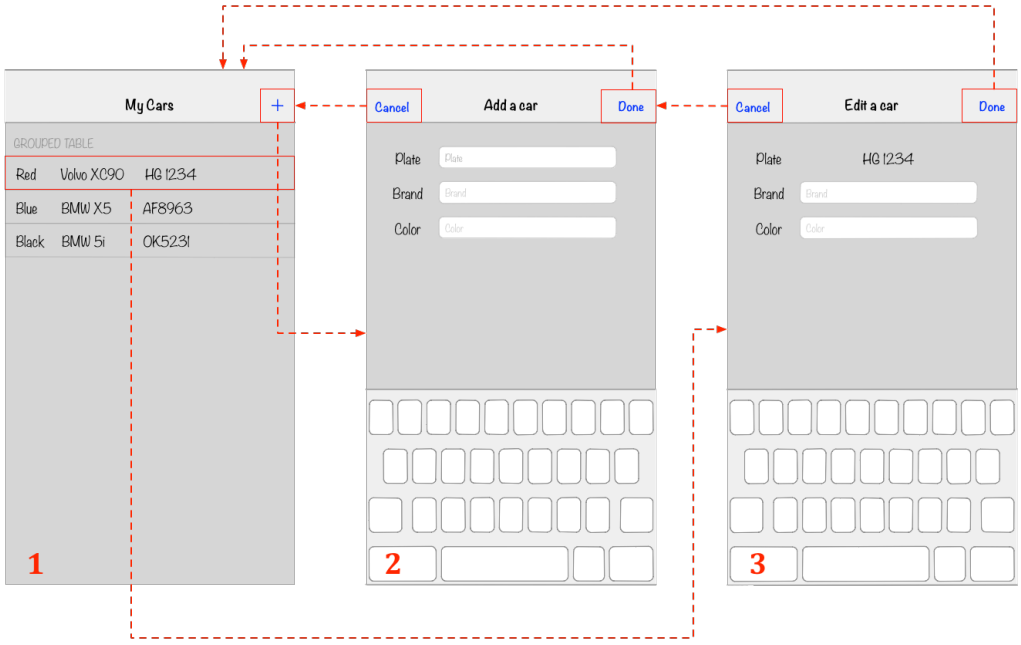


figure 4-12 UI design of Car Collection Page

As shown in figure 4-12, there are three pages for customers to view, add or update a car. They are *My Cars page, Add a car page* and *Edit a car page.*

- *My cars page*: this page shows when a customer clicks “*My Cars”* in the *Me* tab. There is a table view in this page displaying all the cars belong to the customer. Each car has three information: plate, brand and color. There is an [Add] button on the top right corner of this page.

- *Add a car page*: this page shows when a customer clicks the [Add] button on the *My cars Page*. There are three information that customer needs to input or choose and two buttons on the top. After filling all the information, the customer can click the [Done] button on the top right corner. Then the information will be sent to server to check and save. If the customer wants to cancel the process, he or she can click the [Cancel] button on the top left corner then the page will dismiss.
- *Edit a car page*: this page shows when a customer clicks a row on the *My cars Page*. The customer can change the car's brand and color. Then he or she can click the [Done] button and the information will be saved. If the customer wants to cancel the process, he or she can click the [Cancel] button on the top left corner then the page will dismiss.

4.2.4.2. UI design for Valets

The mobile clients for valets is rather simple. The *login page* and *reset password page* is almost the same with the customers'. So I will just introduce the *Main functions page* of this mobile client.

4.2.5. Database Design

Chapter 5. IMPLEMENTATION AND TEST

Chapter 6. REVIEW AND FUTURE WORK