|  |
| --- |
| 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: 11 / 11 / 2016 |

Abstract

Nowadays lots of people drive to hotels, clubs or restaurants after working. One of big issues for these people is to find a parking lot around the destination. And one of concerns for these customers is saving time and convenience. Although some places offer valet parking service, it is not that convenient in a big city like Hong Kong. Cause customers need to find the drop-off point, keep the parking ticket and can not recall their vehicles in advance.

To solve these problems, this project develops a smart phone application on iOS to help customers to use valet parking service. Users can create their cars in the application and place an order to park their cars. The application can act as a parking ticket for customers to redeem their cars, which is rather convenient for car owners. Using this application can save both customers and valets a lot of valued time.



**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.

The University of Hong Kong,

LUO Xianyang

**Acknowledgements**

First of all, I would like to thank my supervisor Dr. T.W.Chim. Dr. Chim gave me advice and support during the whole dissertation. And he could always answer my questions fast and helpful, either through email or whatsapp. Without his great help, I can not finish this project.

Besides my supervisor, I would like to thank my second examiner Dr. Ronald H.Y. Chung. He gave me creative questions, insightful advice and great guide.

Also I would like to thank my classmates. I can discuss the idea and design with them and I improve the project with their creative ideas.

Last but not least, I must thank my family who gave my great support when I encountered difficulties.

**Table of Contents**

1. **INTRODUCTION**
   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 [1]. 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. must drop off or return at certain point
4. do not know where his or her car is parked and do not know how long it has parked

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. **Project Description**

This project mainly focuses on helping drivers enjoy better valet parking service. A customer can register to use our service via its mobile phone number. After signing in, he or she can request dropping its vehicle anywhere in the service area. If there is an available valet nearby, our system will generate an order for user. Customer will get valet’s basic information like profile image, name and mobile phone number. And the valet will also get customer’s information, the drop off point and a route to that position. When our valet meets the customer, he will show his name card to the user to verify identity. Then the valet will park the vehicle and the customer can do its own business. Whenever the customer wants to get the vehicle back, he or she can just pick a address in the service area and our valets will return the vehicle back to the customer.

This project has three major parts as below:

1. iOS version application for customer
   1. allows a customer to sign up, sign in and reset password via mobile phone number
   2. allows a customer to pick a place in the service area to drop off its vehicle or get its vehicle back
   3. allows a customer to check the status of its vehicle
   4. shows a route from customer’s position to meet point
   5. allows the user to view all the historical orders
2. iOS version application for valet
   1. allows a valet to login and reset password via mobile phone number
   2. allows a valet to see all the current drop off orders and return orders
   3. shows customer’s basic information
   4. shows a route from valet’s position to meet point
   5. allows a valet to update status of an order
3. server and database
   1. processes all the http request and sends a proper response back to customers and valets
   2. stores data safely of all users, valets, cars and orders
   3. sends notification to a user when the status of an order is changed
   4. **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[2], Jun Gong’s Guideline for Mobile Application[3] and Nurul’s Threes Layers Design Guideline for Mobile Application[4].

Apart from user interface, the app should be rather easy to use. Users do not need to follow a complicate guideline to generate an order[5]. 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 is 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 customers, the application should have friendly and comprehensive user interface, reasonable layout and overall clean look.
2. innovation: customers are more willing to use the application by using new technology like 3D touch. This project allows customer to request service rather fast from the icon which makes it rather convenient and enjoyable.
3. connectivity: a customer may enjoy more convenience if he or she has an Apple Watch[6]. If all the parking and recall request can be done through Apple Watch. A cusomter does not even need to take his or her phone out the pocket.
   1. **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. The first one is Luxe operating in the USA and the second one is Youbo operating in mainland China.

Chapter 3 talks about the background knowledge of this project including third party frameworks used in mobile clients and server.

Chapter 4 introduces the detail design for this project. It will firstly give the system requirements then introduces the whole design for this project including the system architecture, the system work flows, use case design, UI design and database design.

Chapter 5 talks about the implementation for this project including server side and two mobile clients. For mobile clients of customers, it will introduce some major modules and then comes with the implementation of user interface.

Chapter 6 discusses about the future work.

1. **RELATED WORKS**

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.

* 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.

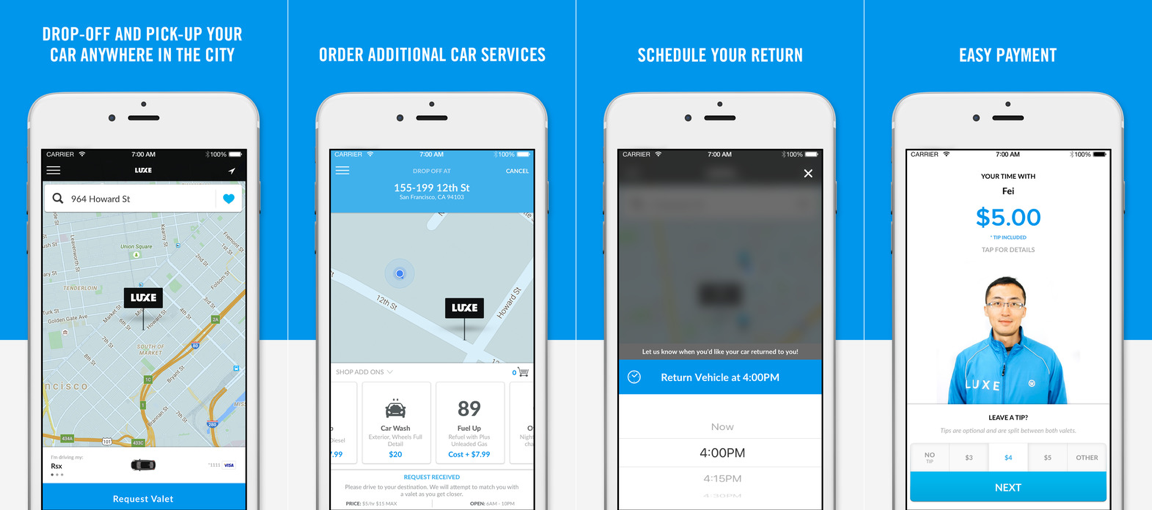


Figure 2-1 iPhone Screenshots for Luxe

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

Advantages:

1. the customer 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 customer saves time since he or she does not need to find a certain drop-off point
2. the customer can pay for the service using its phone, which makes it an 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
   1. **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. a customer can only drop off and get its vehicle return at certain position but not everywhere around the service area
2. the application does not use new technology like 3D touch and does not offer Apple Watch application
3. the user interface does not look good.
4. **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. Google Maps SDK for iOS**

<https://developers.google.com/maps/documentation/ios-sdk/>

Google map is one of the most powerful maps and google map sdk offers map service which is fast to integrate. It powers lots of great application on both iOS and android platform such as UBER, Didi and Runtastic. The user can add map to its application after obtaining an API key. A simple demo of google map with a certain location is shown below:

|  |
| --- |
| self.mapView = [[GMSMapView alloc] init];  GMSCameraPosition \* camera = [GMSCameraPosition cameraWithLatitude:22.2860547 longitude:114.13799 zoom:16];  self.mapView = [GMSMapView mapWithFrame:CGRectZero camera:camera];  self.mapView.delegate = self;  self.view = self.mapView; |

This code block creates an instance of *GMSMapView*. Then it creates a camera with latitude, longitude and scale size. After setting the camera property of map, the view will show the location according to the coordinate.

In this project, the main view is built on google map. Customer can see its position, vehicle’s position, a route from its location to drop off point and so on.

**3.1.2. Google Places API for iOS**

<https://developers.google.com/places/ios-api/>

Google places API offers data from the same database used by Google Maps and Google+ Local. It features over 100 million businesses and points of interest that are updated frequently through owner verified listings and user-moderated contributions. A user can get information of places with place ID or coordinate. A simple demo of google places API is shown below:

|  |
| --- |
| self.geocoder = [[GMSGeocoder alloc] init];  [self.geocoder reverseGeocodeCoordinate:position.target  completionHandler:^(GMSReverseGeocodeResponse \* response, NSError \* error)  [self.mapSearchPlaceView setParkAddress:response.results[0].lines[0]];  }]; |

The code block creates an instance of *GMSGeocoder*. The *position* variable is an instance of *CLLocation*. The geocoder can use the coordinate of the location to get the address of it. The format of the address is “1-2 Queen Victoria Street”.

In this project, I use this API to get address of a location and create a view allowing customers to search an address in Hong Kong.

**3.1.3. AFNetworking**

[**http://afnetworking.com/**](http://afnetworking.com/)

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[7]. 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**

[**https://github.com/jdg/MBProgressHUD**](https://github.com/jdg/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

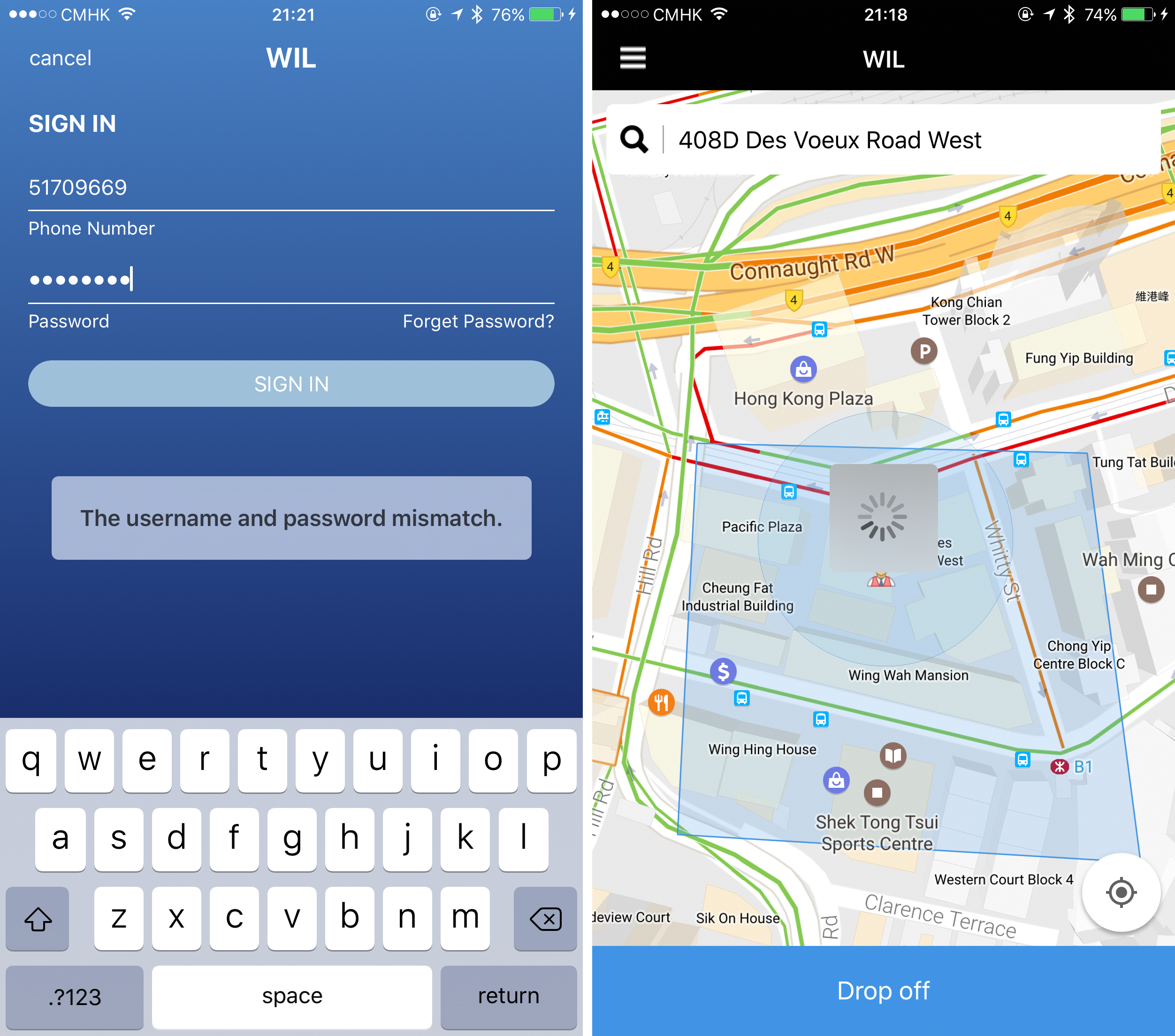


Figure 3-1 Screenshots for usage of MBProgressHUD

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];  }); |

**3.1.5. RESideMenu**

<https://github.com/romaonthego/RESideMenu>

RESideMenu is an iOS style side menu with parallax effect. Nowadays many iOS applications in the market has a side menu. Users can open the menu by tapping a button or drag the screen from the edge. The advantage of this design is saving space of the screen since the space is very valuable. In this project, I use RESideMenu to create the menu options for user. User can choose function views like payment, orders and contact from the menu.

**3.2. Tools used in Back-End**

**3.2.1. LeanCloud**

1. **REQUIREMENTS AND DESIGN**

In this part, I will introduce the requirements and design for this project. At first, I will talk about the requirements analysis including product perspective, user perspective and functional requirements. Then comes with design for this project including system architecture, system workflow, object oriented design, user interface design and database design.

**4.1. Requirements Analysis**

**4.1.1. Product Perspective**

I develop the front-end and back-end of this project. The front-end is built on Xcode 9 using Objective-C and the back-end is built with LeanCloud. And the front-end has two different applications on iOS platform. One is for customers and the other one is for valets. Customers can use their mobile phone number to sign up, sign in and reset password. And valets can use their mobile phone number to sign in and reset password.

**4.1.2. User perspective**

There are two types of user in this project: customer and valet. Customers are who uses our valet parking service and valets are who park and return vehicle for customers.

After a customer signing in, he or she can see available valets around, choose anywhere in the service area to park, check its order status, check valet’s information if he or she has an unfinished order, request its vehicle back anywhere in the service area and other basic settings. After a valet signing in, he or she can access to all opening orders related to him or her, check customer’s information and location, get an route to the drop off or return point, update the status of an order and other basic settings.

**4.1.3. Functional Requirements**

Hers is a list of high-level functional requirements that this project is focus on. The back-end must:

1. Authenticate and authorize a customer or a valet
2. Store a number of data records describing customers and valet. Each record will have attributes below:
   * First name
   * Last name
   * Mobile phone number
   * Profile image URL
3. Store a number of data records describing customers’ locations and valets’ locations. Each record will have attribute below:
   * User’s identifier
   * User’s coordinate
   * User’s status
4. Store a number of data records describing orders. Each record will have attribute below:
   * Customer’s identifier
   * Drop off valet’s identifier
   * Return valet’s identifier
   * Create time
   * End time
   * Drop off location
   * Return location
   * Order status
5. Store photos and URLs of photos
6. Return corresponding response to front-end

The front-end must:

1. Allow customers to view its location and valets nearby
2. Allow customers to choose a place in the service area to drop off or return its vehicle
3. Show the route from customer’s location to meet point
4. Allow customers to check order status
5. Allow customers to view valet’s basic information and call valet
6. Allow valets to view all the unfinished orders related to him or her
7. Allow valets to check customer’s basic information
8. Allow valets to update order status
9. Show the route from valet’s location to meet point

Next I will introduce the details of design.

**4.2. Design**

**4.2.1. System Architecture**

The entire system can by divided into two parts: front-end and back-end. Back-end is a server and database based on LeanCloud. It handles all the http request including signing up, signing in, fetching valets’ locations, uploading customers’ locations, creating an order, updating an order, sending notifications to user and so on. 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 4-1 shows the architecture of the entire system of this project.

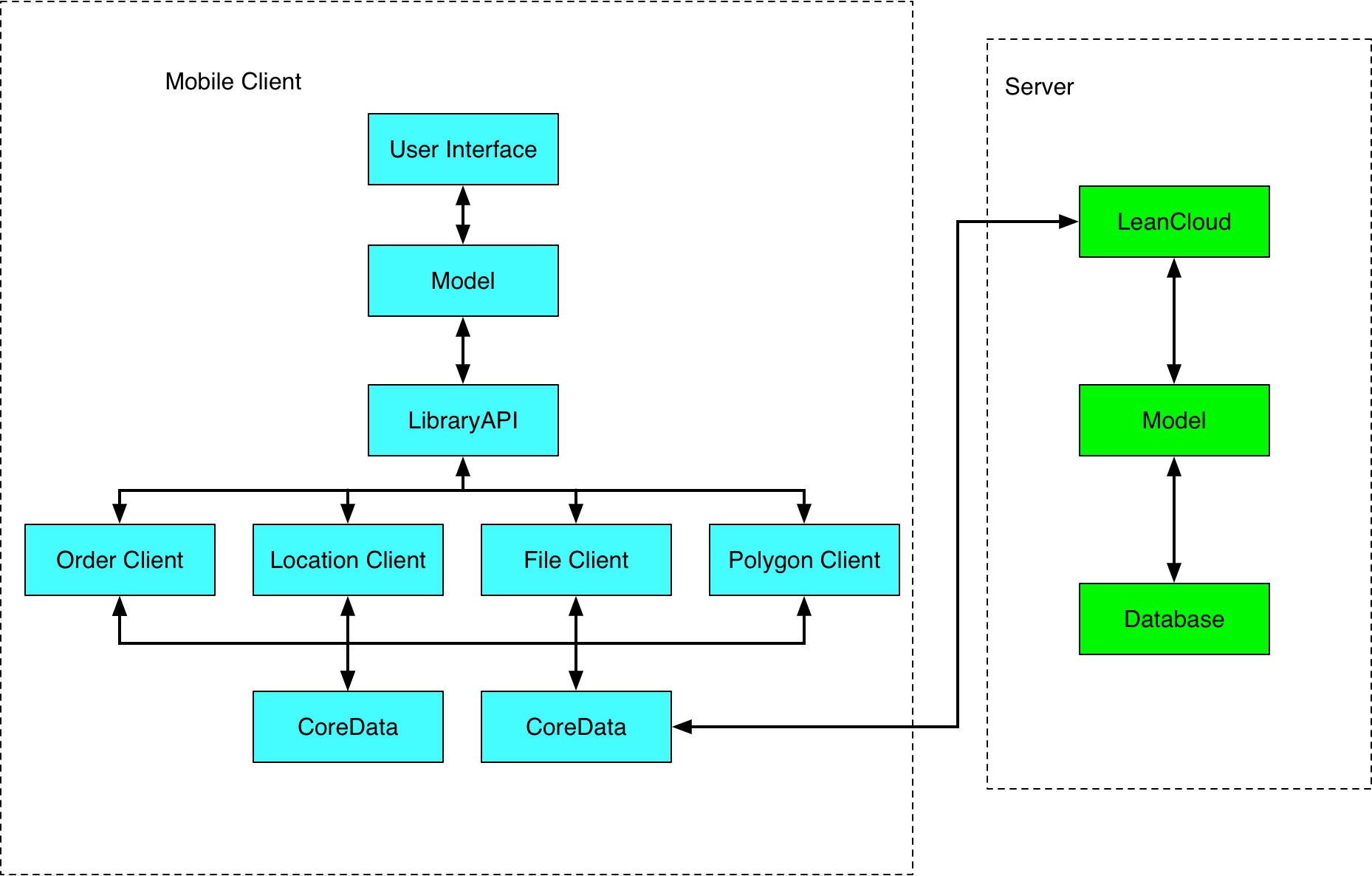


Figure 4-1 Architecture of System

**4.2.2. System Workflow**

The system workflows consist of two parts. The first part is the workflow in the back-end and the second part is the work flow in the front-end. These two parts will be discussed separately.

4.2.2.1. Workflows in Back-End

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

1. Flow chart for account management

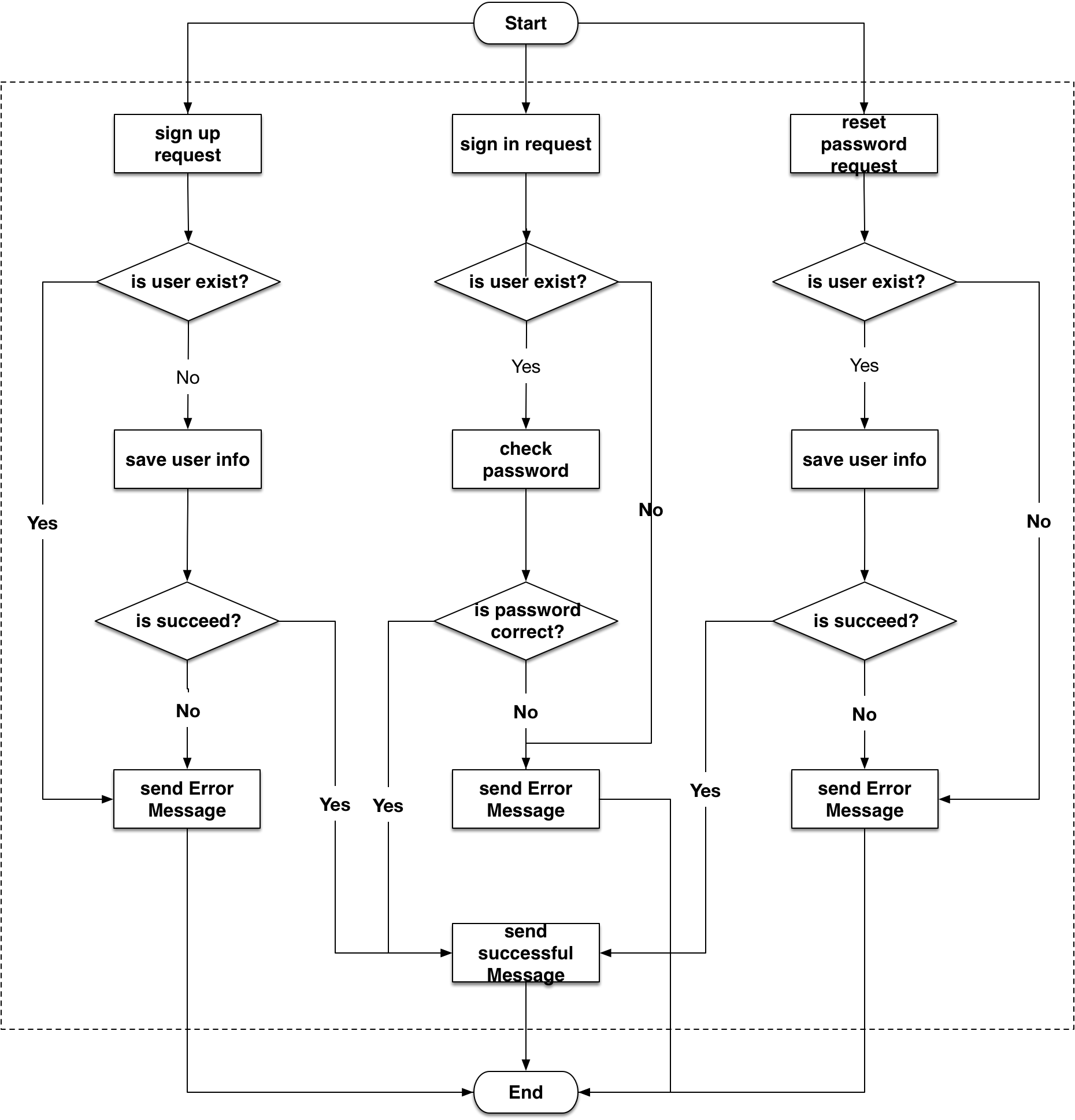


Figure 4-2 Flow chart for account management

As shown in figure 4-2, server needs handle three different request related to accounts management: sign up request, sign in request and reset password request.

For sign up request, if there exist an account with the same mobile phone number or user name, the server will send error message back to front-end. If sign in process is success, the server will send an *User* object back to front-end.

For sign in request, if the user enter the correct account and password, the server will send a successful message back to front-end. If the account and password mismatch or cannot find the user, the server will send the corresponding error message back to front-end with the *User* model. Then user needs to sign in again.

For reset password request, if the user enter the correct verification code and a new password, then the server will change renew user’s password and return a successful message along with the *User* model. If the server cannot find the user or the verification code is wrong, then it will send back a corresponding error message.

1. Flow chart for order management

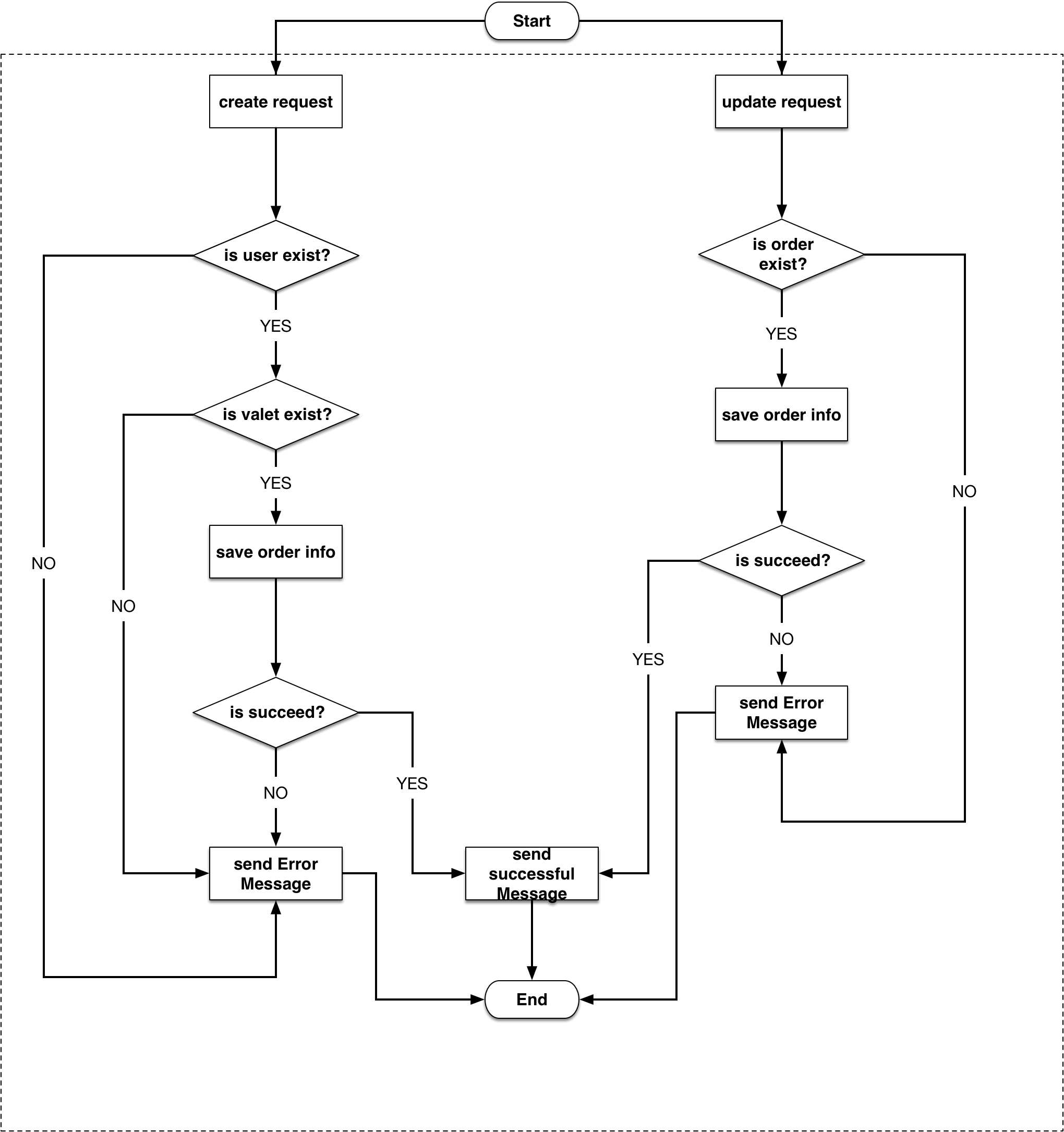


Figure 4-3 flow chart for order management

As shown in figure 4-3, the server can handle two kinds of order management request: create and update.

The creating order quest has user’s information, valet’s information, drop off point coordinate. So the server will firstly check if the customer exists, if so, it will check if the valet exists. Final step is save this order in the database. If saving process is successful, server will send the order object back to front-end. If any step occurs error, server will send corresponding error message back to front-end.

After getting the order object, the front-end will save this object locally for later use. If a customer or a valet wants to update an order such as cancel or end an order, the front-end will send an update order request along with the order identifier and next order status. After getting the request, the server will firstly check if the order exists, if so, the server will update the status of the order. If saving process is successful, server will send the order object back to front-end. If cannot find the order cannot save the order, the server will send an error message back to front-end.

1. Flow chart for location management

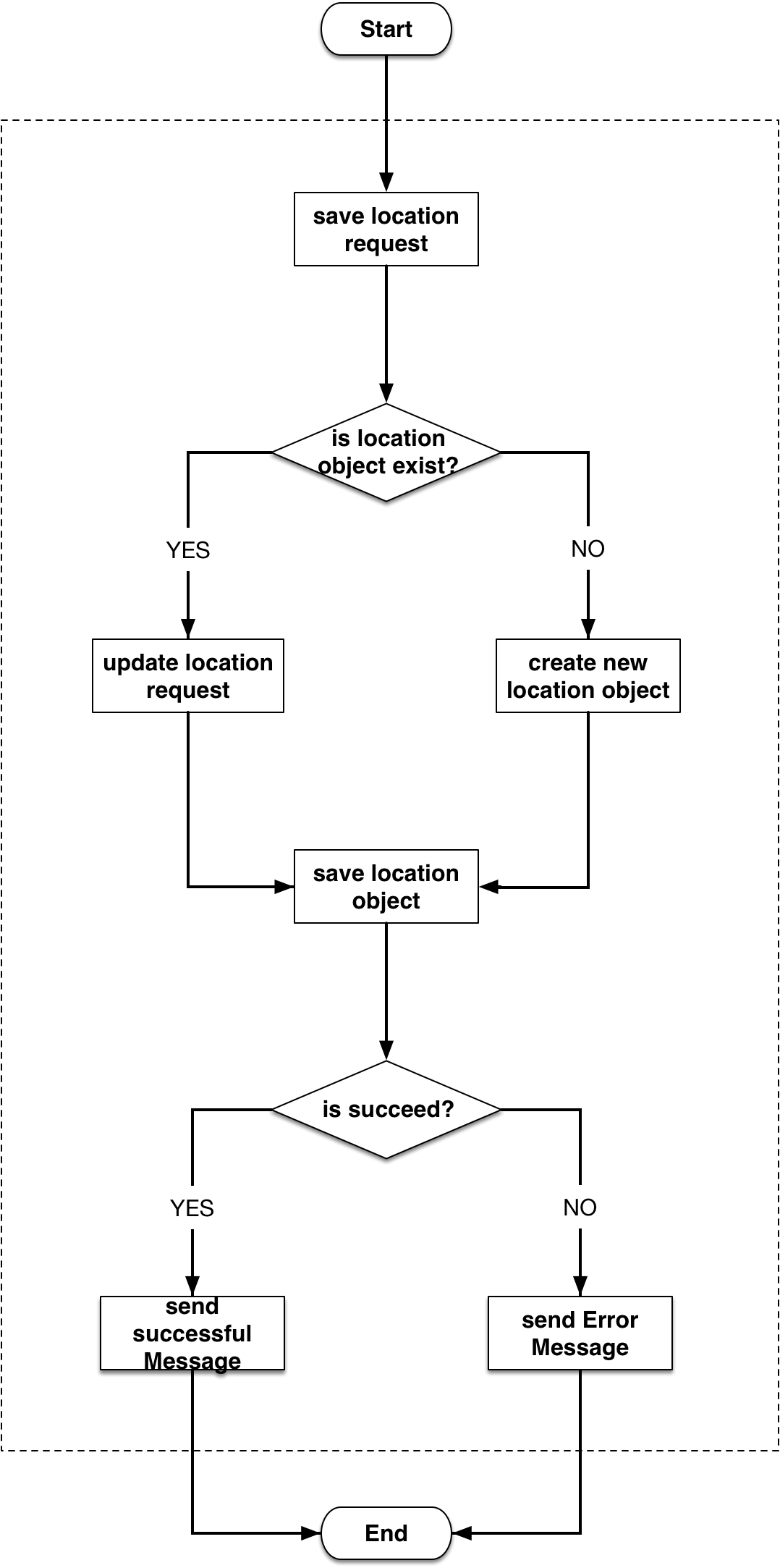


Figure 4-4 Flow chart for location management

When a customer or a valet opens the application, their locations are uploaded to server every 10 seconds. Each user has only one location object on the server and the location object is updated in real time.

As shown in figure 4-4, when the server gets a request of saving location object, if will firstly check if the location object exists in the database. If so, it will update the location object with the new coordinate. If not, it will create a new location object and save all the information in the database. If there is no error, the server will send a response to front-end with the location object. If not, the server will send a corresponding error message to front-end.

4.2.2.2. Workflows in Front-End

The two mobile applications work as front-end in this project. One is for customers and the other one is for valets. There are three major managements in the front-end: account management, order management and location management. Figure 4-5 shows the flow chart for front-end.

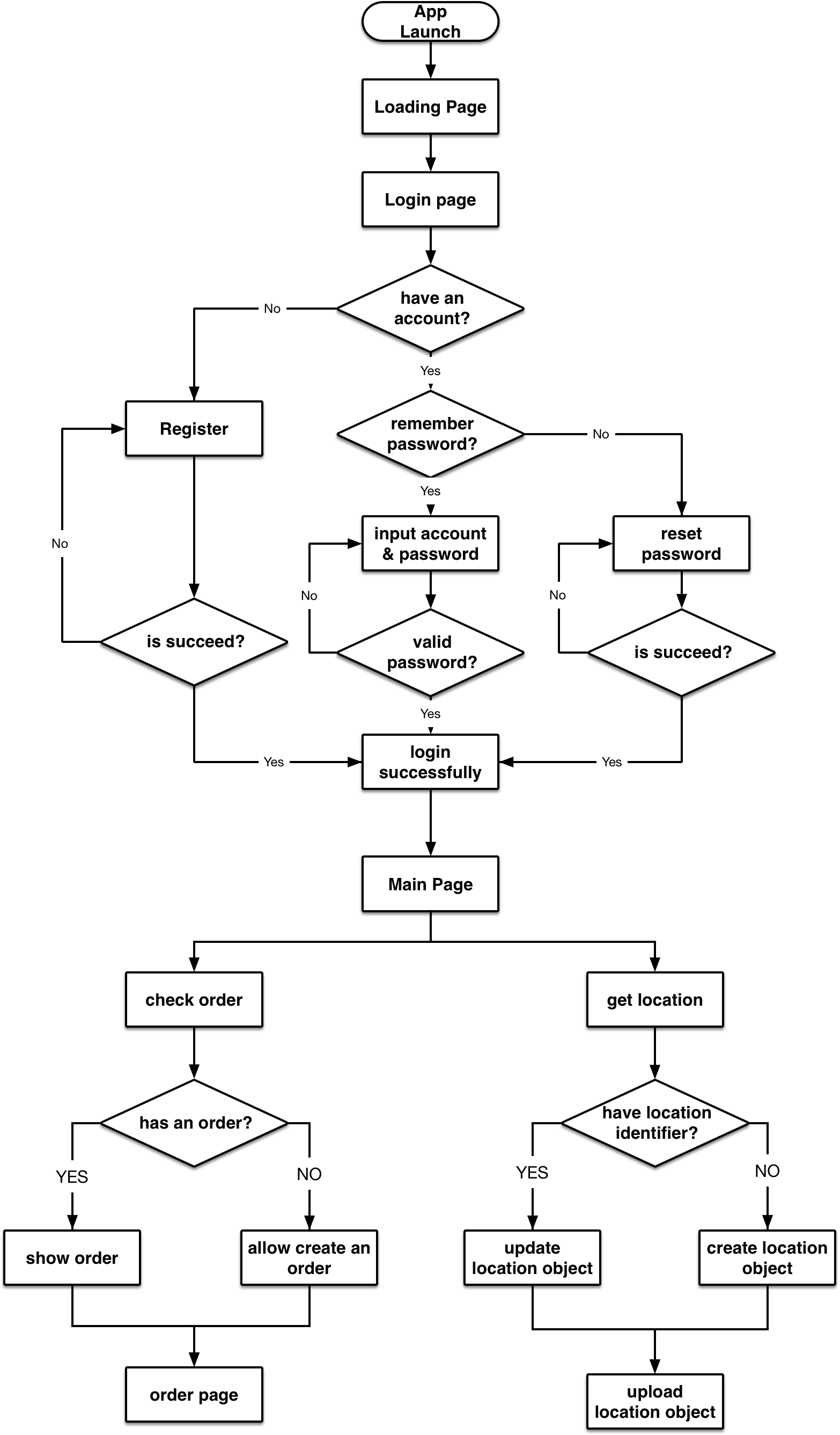


Figure 4-5 Flow chart for front-end

1. Account Management

When the application is launched for the first time, it will show an introduction page and ask user to sign up or sign in. If the user does not have an account, he or she needs to sign up using its mobile phone number. Our system will send a verification code to that phone and user needs to fill in basic information. After successfully signing up, the instruction view will disappear and main view will show. If the user has an account, he or she can sign in using its mobile phone number and password. After successfully signing in, the main view will show up. If the mobile phone number and password mismatch, the application will ask the user to input mobile phone number and password again. If the user forgets its password, he or she can use its mobile phone number to get a verification code to reset password.

1. Order Management

In the main view, a customer can park its vehicle. An order has 10 statuses:

* Undefined: the application does not know if the user has an unfinished order
* None: the user does not have an unfinished order
* Dropping off: the user is dropping off it vehicle
* Parking: a valet is parking user’s vehicle
* Parked: the vehicle is parked
* Requesting back: the user wants its vehicle back
* Returning back: a valet is returning user’s vehicle back
* Waiting for payment: service is finished and waits for user’s payment
* Finished: user pays the bill and the order is finished
* Cancel: the user cancels the order when dropping off

When the application for customer is launched, it will ask the server if the user has an unfinished order. During this time, the status of order is *undefined*. If the user does not have an unfinished order, the order’s status will change to *none.* And the user can request our service now. The flow chart for order status is shown in figure 4-6.

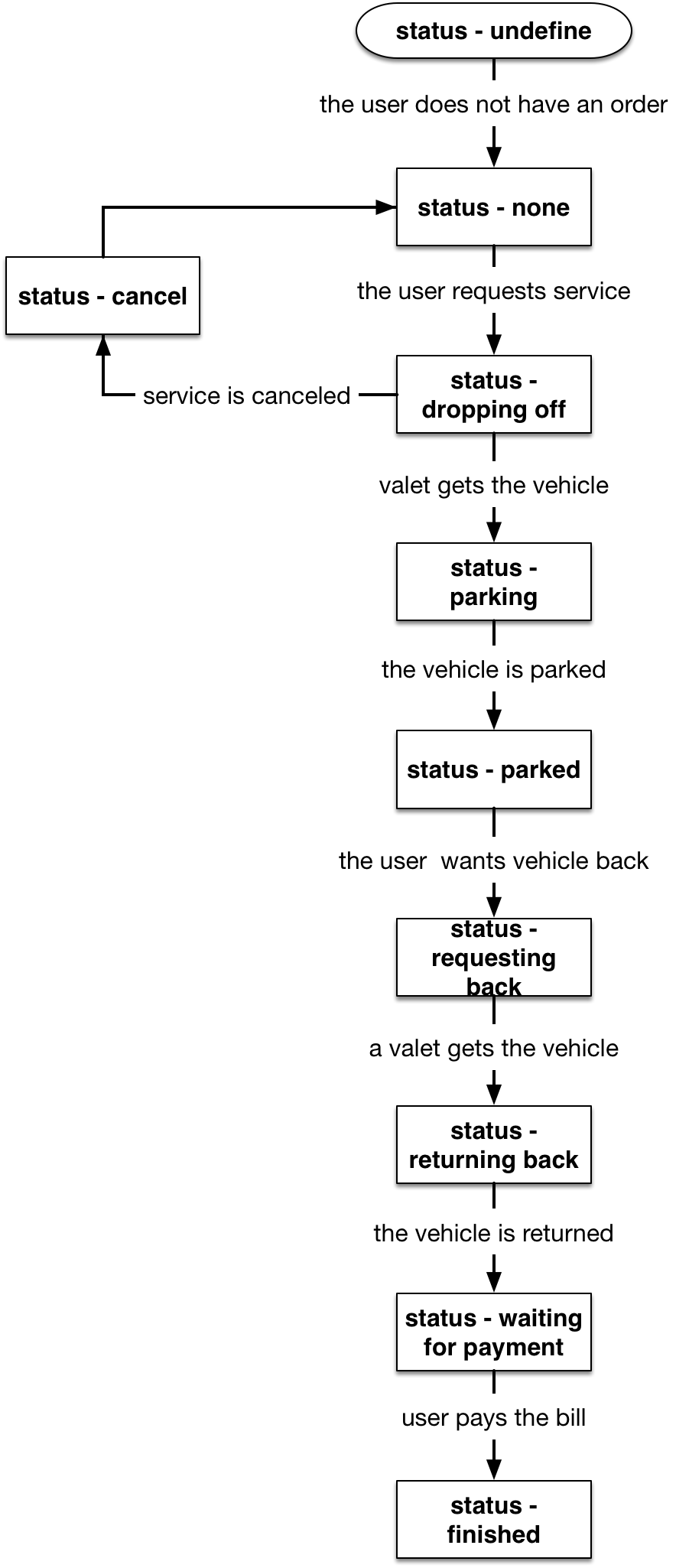


Figure 4-6 Flow chart for order status

1. Location Management

The location object in server saves users’ and valets’ locations. Attributes for each location object are:

* User’s name
* User’s mobile phone number
* User’s identifier
* User’s coordinate
* User’s status (only for valet)

When the applications for customers or valets are launched, they will start to get user’s location. And applications will check if the user has a location object on the server, if so, they will get the identifier of that object. When the iPhone pass user’s location to our applications, they will update the location object if the user already has an location object or create a new location object and upload it to server if not.

The location object for valet has an attribute *status*. This attribute indicates the status of the valet. It can be *busy* or *available*. *Busy* means the valet is parking to returning a vehicle. And *available* means the valet is ready for service.

**4.2.3. Object Oriented Design**

Below I will introduce the object oriented design of 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 | Sign in | Customer signs in |
| Customer | Sign up | Customer signs up the service |
| Customer | Reset password | Customer resets its password |
| Customer, valet | Park a vehicle | Customer parks its vehicle |
| Customer, valet | Return a vehicle | Valet returns customer’s vehicle |

4.2.3.2. Use case diagrams



Figure 4-7 Use Case Diagram

Figure 4-7 shows the use case diagram. As it shows in the picture, there are four use cases as below:

1. For the “park a vehicle” use case, a customer can use our service anywhere in the service area if he or she does not have an unfinished order and there are valets available around.
2. For the “return a vehicle” use case, a customer can request its vehicle back anywhere in the service area. Our system can choose a nearest valet to serve the customer.
3. For “sign in” use case, it is included in the “park a vehicle” use case
4. If the user does not have an account, the “sign up” use case will be invoked
5. If the user forgets its password, the “reset password” use case will be invoked. User can use its mobile phone number to reset password.

4.2.3.3. Sequence diagrams

1. Figure 4-8 shows the sequence diagram for “sign up”. All the messages in this sequence diagram are asynchronous.

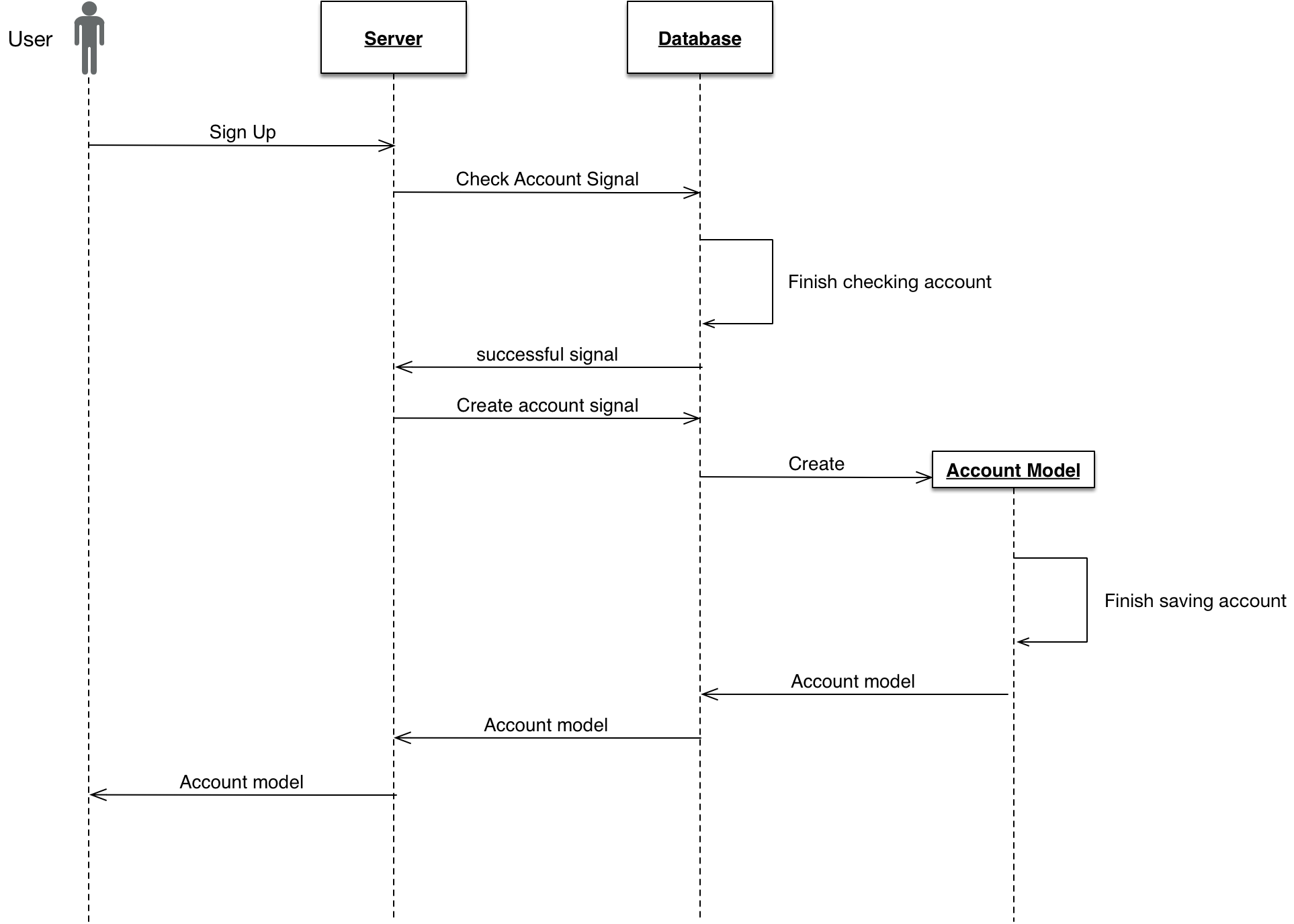


Figure 4-8 Sequence diagram for “Sign Up”

As shown in the figure 4-7, customer will firstly send a sign up request to server with its information including name, mobile phone number, password and verification code. And the server will check with the database if the information is valid. If so, database will create an account and server will send this account object back to front-end.

1. Figure 4-9 shows the sequence diagram for “sign in”. All the messages in this sequence diagram are asynchronous.

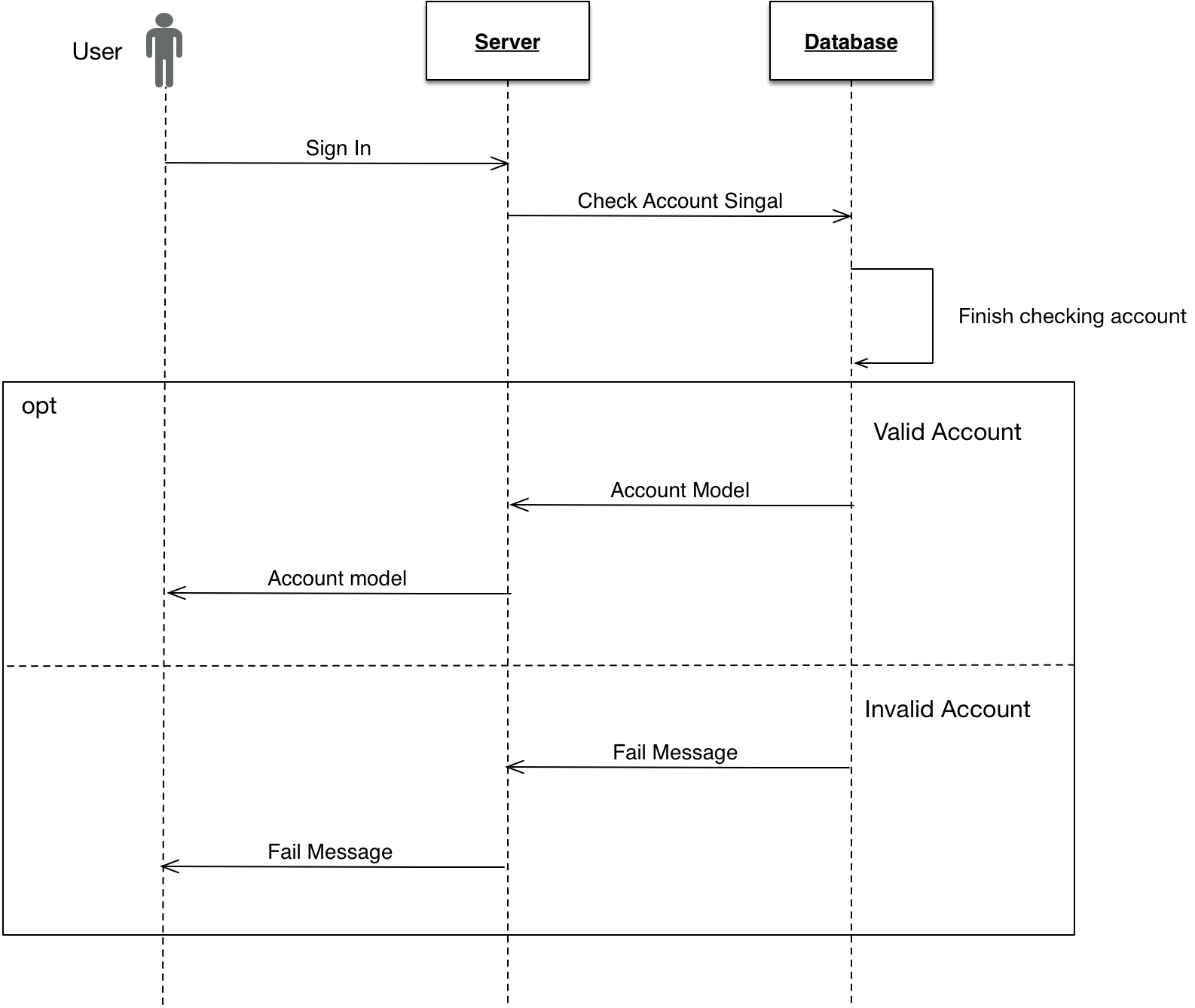


Figure 4-9 Sequence diagram for “Sign In”

As shown in the figure 4-9, a user can sign in using its mobile phone number and password. Server will check the validity of the account with the database. If the account is valid, server will send back the account object. If not, server will send back a corresponding error message.

1. Figure 4-10 shows the sequence diagram for “reset password”. All the messages in this sequence diagram are asynchronous.

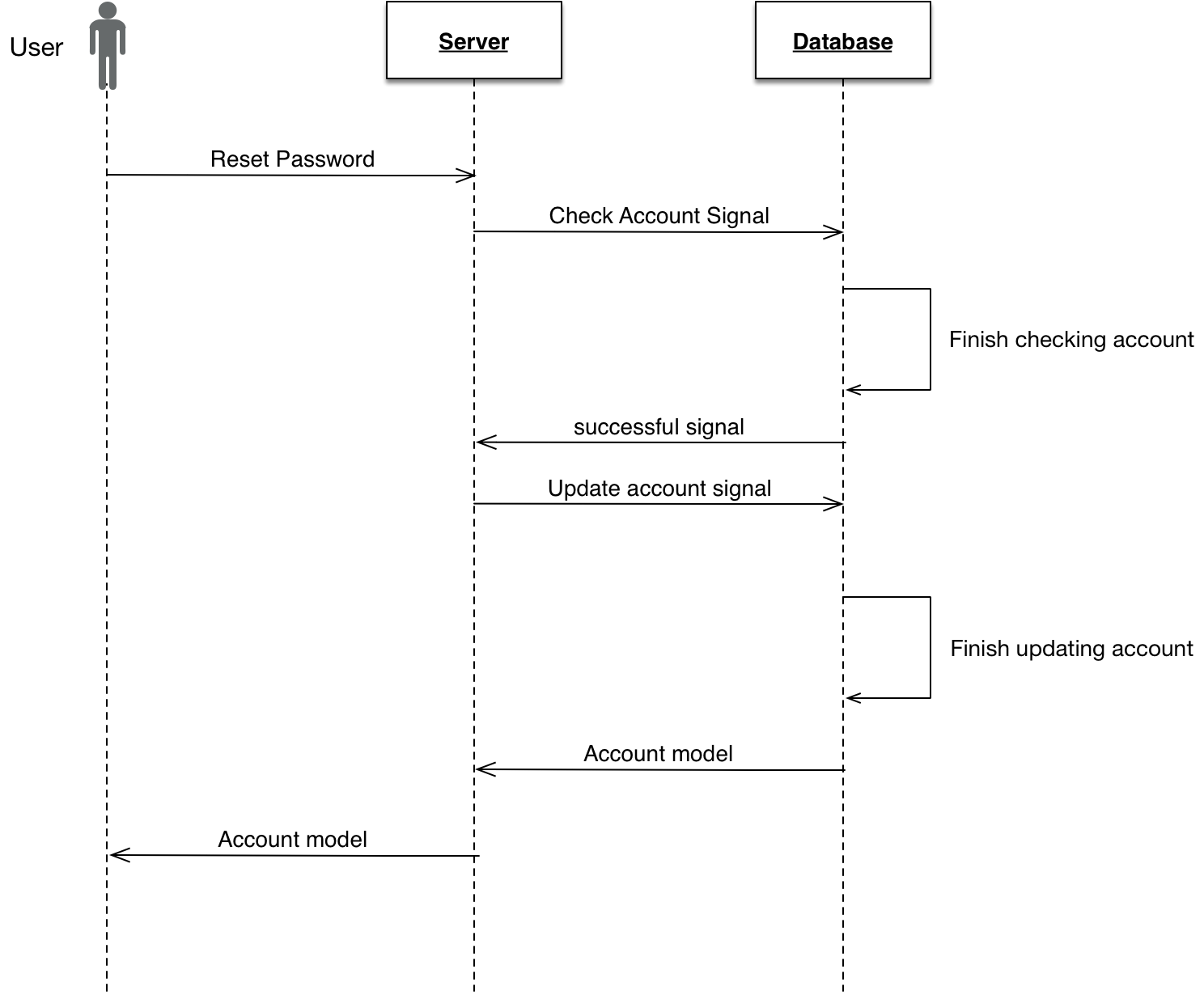


Figure 4-10 Sequence Diagram For “Reset Password”

As shown in figure 4-10, a user can reset password using its mobile phone number and a verification code. If the account and information is valid, the server will update user’s password and send a new account object back to front-end.

1. Figure 4-11 show the sequence diagram for “park a vehicle”. All the messages in this sequence diagram are asynchronous

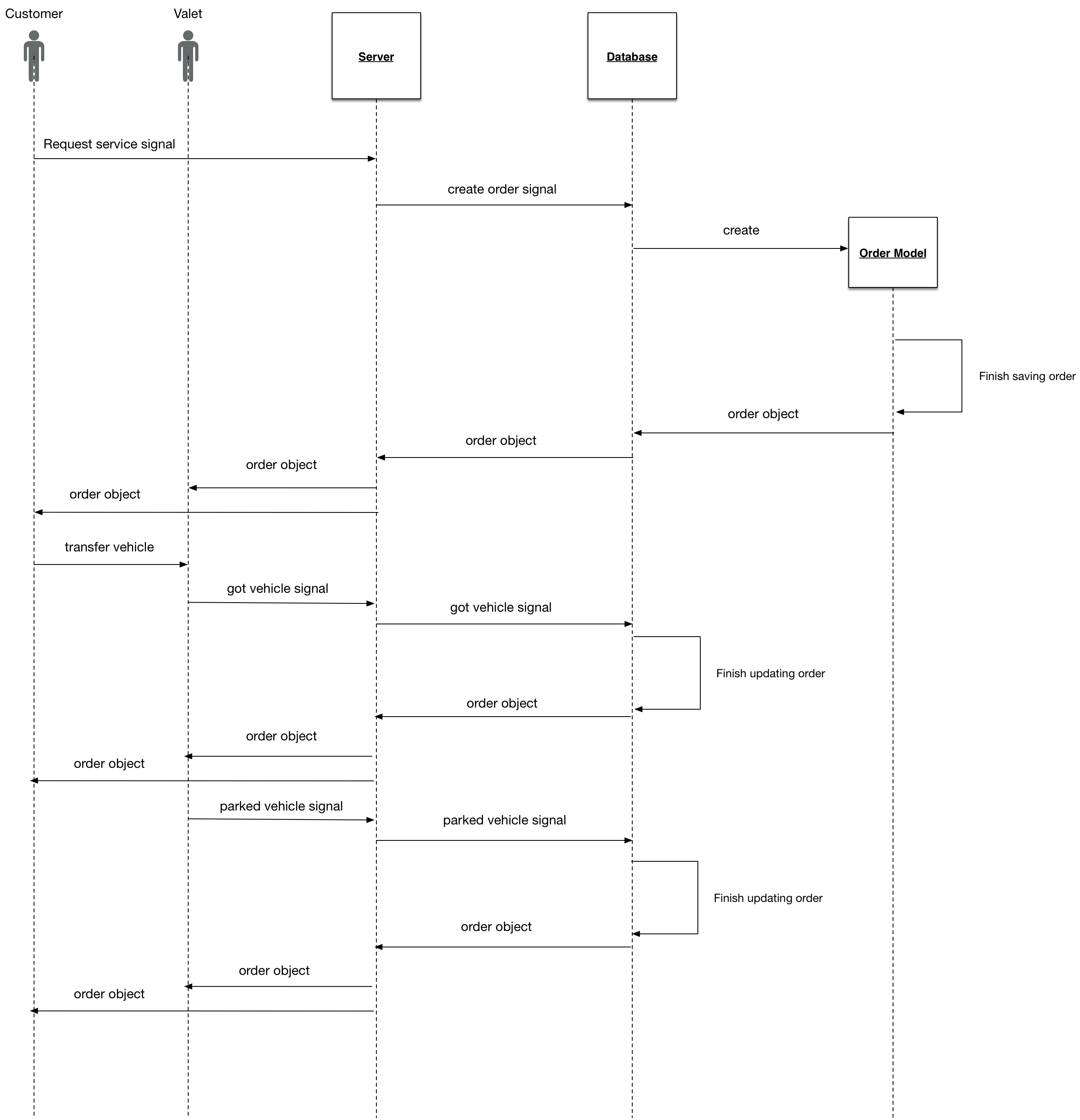


Figure 4-11 Sequence diagram for “Park a vehicle”

As shown in the figure 4-11, the use case “park a vehicle” contains steps below

1. A customer requests service and send a request to server.
2. Server checks the account and creates an order for this customer. If the order is created successfully, the order object will be sent to both customer and valet
3. Customer transfers its vehicle to valet
4. Valet gets the vehicle and sends a request to server to update order status
5. Server checks and updates the order. Then it will send the updated order to both customer and valet
6. Valet parks the vehicle and sends a request to server to update order status
7. Server checks and updates the order. Then it will send the updated order to both customer and valet

1. Figure 4-11 show the sequence diagram for “return a vehicle”. All the messages in this sequence diagram are asynchronous

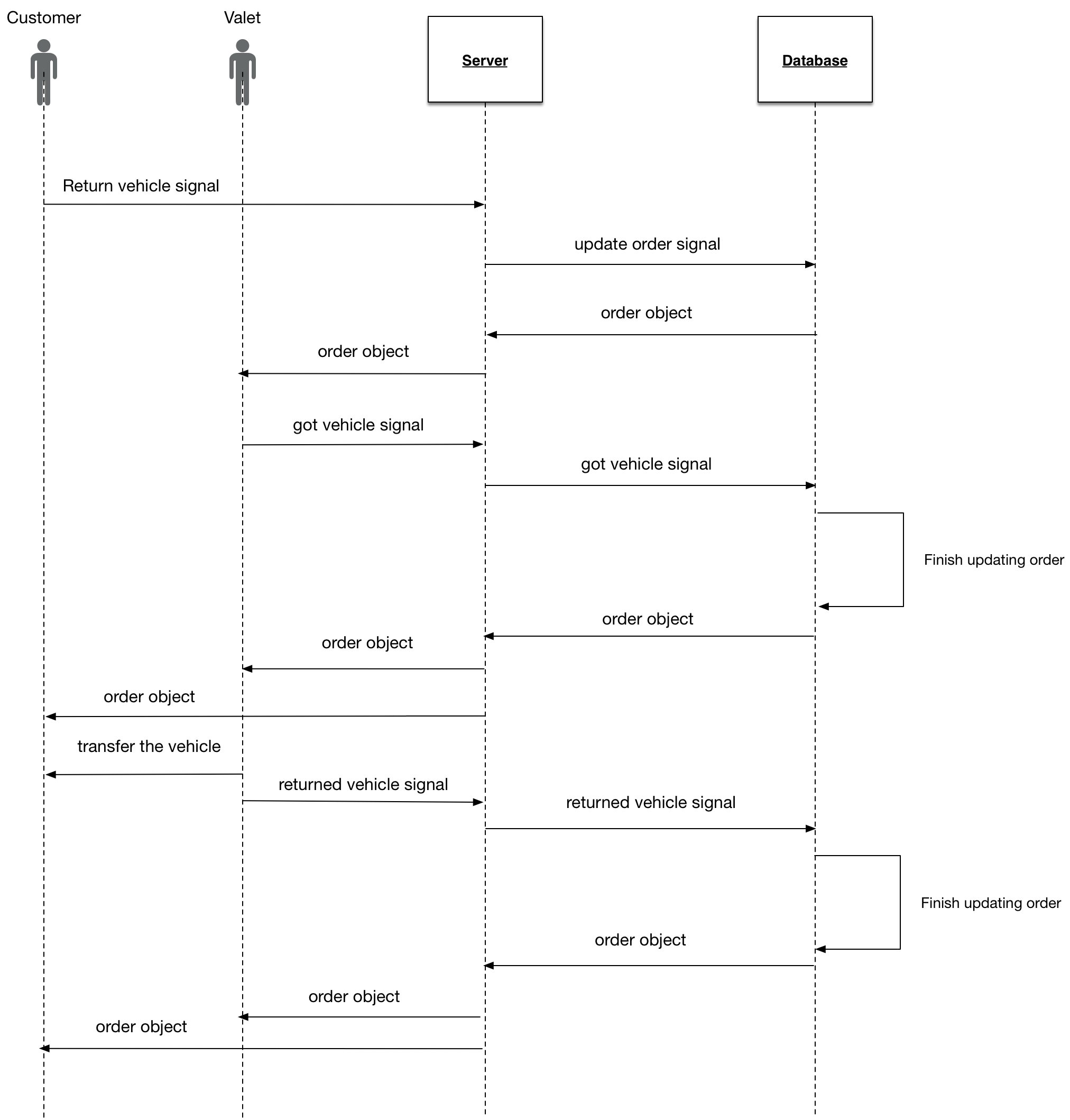


Figure 4-12 Sequence Diagram for “Return A Vehicle”

As shown in the figure 4-12, the use case “return a vehicle” contains steps below

1. A customer wants its vehicle back and sends an request to server
2. Server gets the request, checks the order and assigns the nearest valet for this customer. If the order is updated successfully, the updated order object will be sent to both customer and valet
3. Valet gets the order and he or she will go to parking lot. When the valet gets the vehicle, he or she will send a request to server to update the order status
4. Server checks and updates the order. Then the updated order will be sent to both customer and valet
5. Valets transfers the vehicle to customer. Then the valet will send an request to server to update the order
6. Server checks and updates the order. Then the updated order will be sent to both customer and valet

4.2.3.4. Documents for design use cases

Below introduce the documents for the five use cases.

1. “Sign up” use case

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Sign up | |
| **Actor(s):** | Customer | |
| **Description:** | This use case describes the membership registration process. | |
| **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 | |

1. “Sign in” use case
2. “Reset password” use case
3. “Park a vehicle” use case
4. “Return a vehicle” use case

4.2.3.5. Class diagrams

1. **IMPLEMENTATION AND TEST**
2. **REVICW AND FUTURE WORKS**