Onion Express

# System Analysis Document

Yang LI
Zhongjin LUO
Guohui YANG
Yiqun LIN
Yirui WANG
Xinying WU

School of Software Engineering
**Tongji University**
Group 4

June 23, 2017

# Contents

# 1 Introduction

Onion Express® is a system built for logistic companies, which provides them with a solution to logistics tracking, goods packing, goods distribution, after-sales management, data storage, information processing, etc.

## 1.1 Purpose

These days as B2C business is increasing rapidly, the growth of logistics business is also remarkable. The enormous market demand brings logistics companies opportunities as well as the challenge. Facing such kind of condition, this project is aimed at improving the efficiency of field personnel and customer satisfaction of a logistics company by building a cross-platform system.

## 1.2 Definitions

As Jobs has ever said, People don't know what I really want at all, until your products are in their eyes. This project is specially designed for an independent logistics companies like UPS. The business scope is limited within China. To be more precise, the express is only available in Jiangsu, Zhejiang and Shanghai at the beginning. Temporarily private orders are not covered in the business scope, which means the express company corporates with e-commercial companies only with the cash-on-delivery express or normal express. The system focuses on logistics service without regard to O2O, bulk cargo or self-support e-business. Timing express might be expanded in future.

## 1.3 System Overview

The actors in the system are classified as *Postman, E-business, Customer service, Customer* and *Agent*. *Customer* and *Agent* are generalized as Receiver. The *Postman* has access to this system only on mobile devices while *Customer* has access both on browsers as well as mobile devices. *E-business* offers orders periodically. *Customer service* helps to deal with tasks cannot be done only by the system.
Web application and iOS application provide different functions for different users to enhance user experience and have some humanization design(e.g. using different colors to mark tasks as reception or delivery in postmans app). Besides basic functions, the system also

provides some advanced functions, like printing invoices. Different offline payment methods are supported. And the customers telephone number is hidden to protect his/her privacy.
The postman is equipped with a multifunctional special device, when the customer receives his/her package, he/she can use this device to pay by card and can also press thumb on it to sign digitally, besides, the device helps collect postmans GPS location accurately.
The system considers all the 8 scenarios, including sending the package, paying for the product, signing the package and so on. To integrate the system, two scenarios are added. One is creating the orders, at the beginning of the entire flow. Another is dealing the order manually, to reduce errors caused by the system and handle other unanticipated situations. That can improve the stability of the system and in consideration of the relatively small scale of users in the early stage, robot customer service is not necessary. It can be taken into consideration when the business is expanding to a certain stage.
This project also designs several user interface mock-ups on the website and on mobile devices. Core functions are exhibited in these mock-ups, for example, the dispatch list interface.
Nonfunctional requirements and further explanations on security, performance, data storage and computing, tracking the package, maintenance and others are detailed in supplementary Specification.

# 2 Requirements

## 2.1 Use Case Modeling

### 2.1.1 Global Use Case

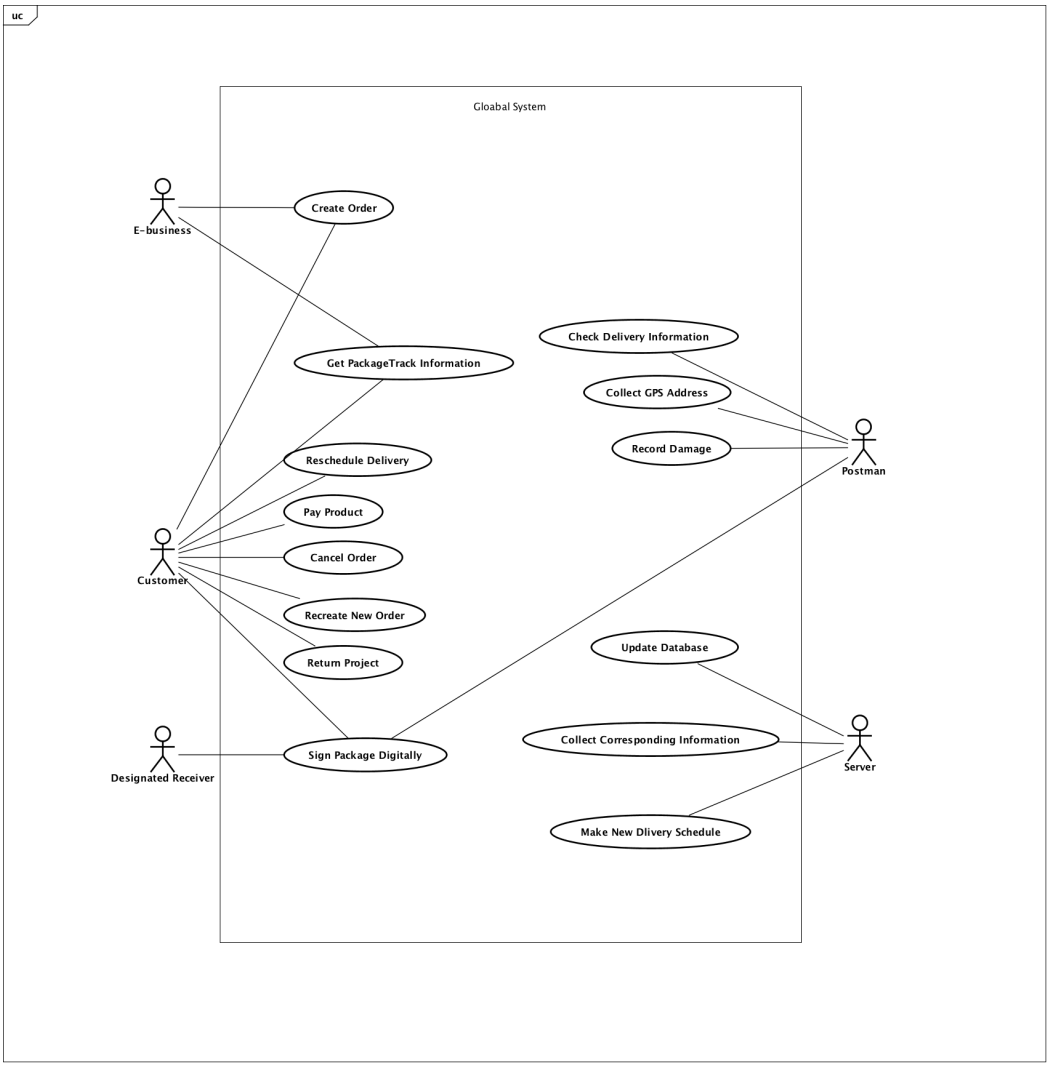



Figure 2.1: Global Use Case Diagram

### 2.1.2 Specification of Use Cases

#### Scenario One

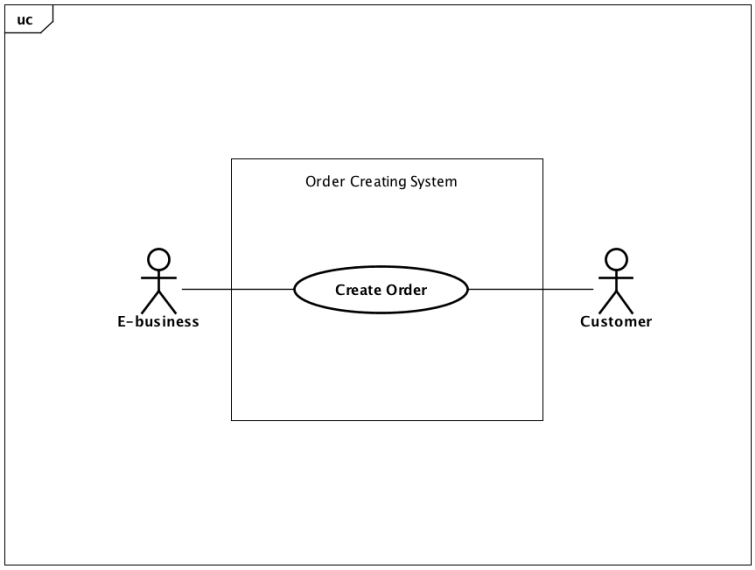



Figure 2.2: Scenario One Use Case Diagram

| Use Case Name | Create Order |
|---|---|
| Related Requirements | Scenario One |
| Goal in Context | The customer or E-business requests to create order. |
| Preconditions | The customer buys products from E-business or wants to send packages. |
| Successful End Condition | Server creates order according to the information provided by E-business & Customer. |
| Failed End Condition | Server refuses to create an order. |
| Primary Actors | E-business and Customer |
| Secondary Actors | None |
| Trigger | E-business & Customer sends the related information to Server. |
| Main Flow | Step 1 : Customer & E-business Registers Information to Server.<br>Step 2 : Customer & E-business Sends Information to Server.<br>Step 3 : Server checks whether the information is completed.<br>Step 4 : Server creates orders automatically according to the information above.<br>Step 5 : E-business & customer transfers products to Logistics company and pays for the delivery. If the customer doesnt pay before, Logistics company will pay for products for the customer in advance.<br>Step 6 : Logistics company transports packages to different regional distribution centers. |
| Extensions | The request of creating order is rejected. |

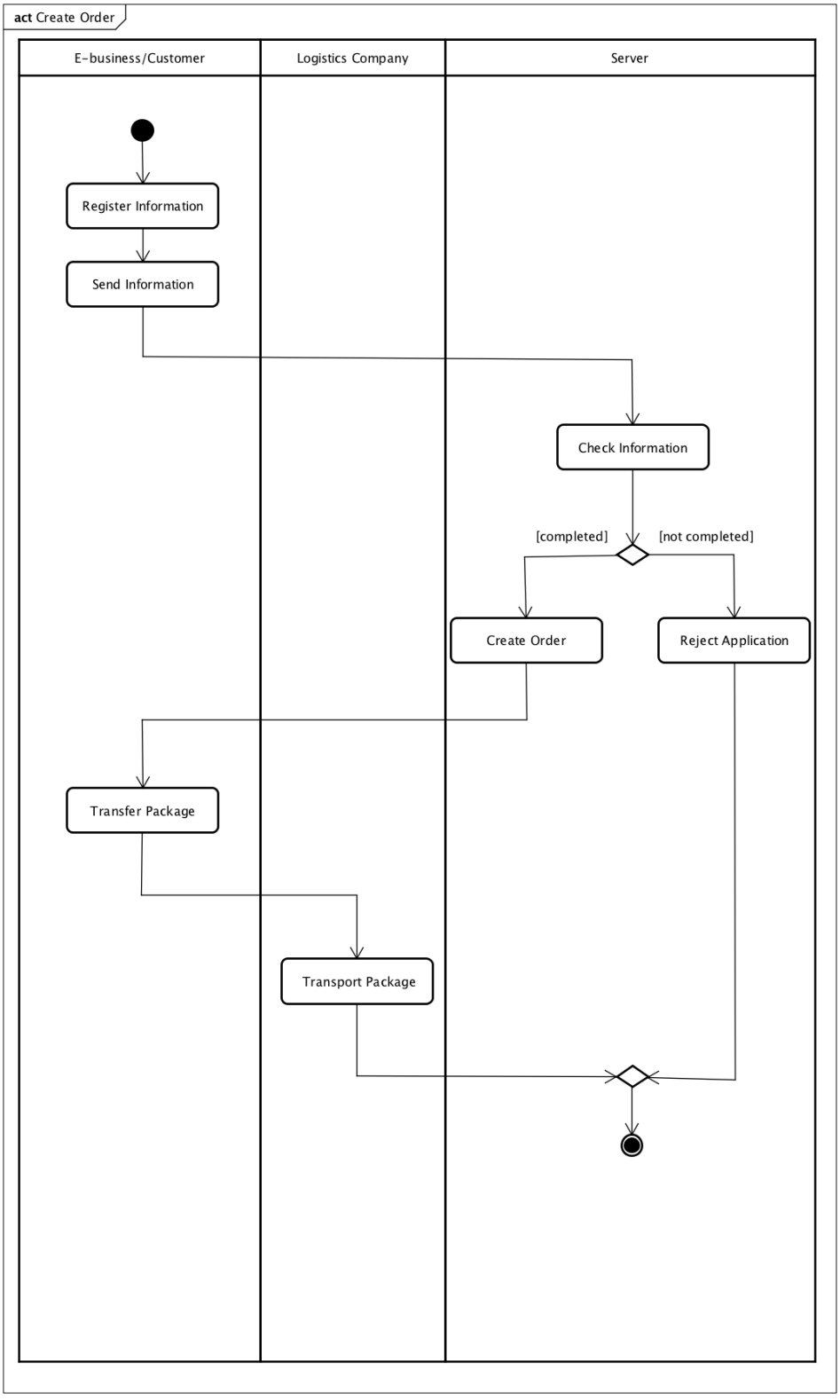



Figure 2.3: Scenario One Activity Diagram

### Scenario Two

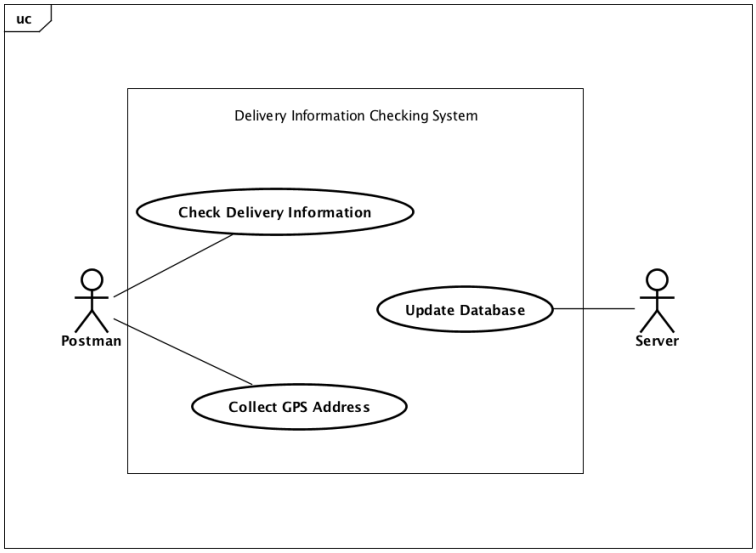



Figure 2.4: Scenario Two Use Case Diagram

| | |
|---|---|
| Use Case Name | Collect GPS Address |
| Related Requirements | Scenario One |
| Goal in Context | Postmen collect the GPS address of the customer. |
| Preconditions | Postmen have delivered packages to the destination. |
| Successful End Condition | Server Updates database successfully. |
| Failed End Condition | None |
| Primary Actors | Postman |
| Secondary Actors | Server |
| Trigger | The GPS address of the customer is not in the database. |
| Main Flow | Step 1 : Postmen inspect that whether the GPS address of the customer is in the database.<br>Step 2 : If the GPS address of the customer is not in the database, postmen will collect the GPS address.<br>Include : Update database. |
| Extensions | None |

| Use Case Name | Check Delivery Information |
|---|---|
| Related Requirements | Scenario Two |
| Goal in Context | Postmen accept delivery task assigned by the system. |
| Preconditions | Orders have been created and packages have been transported to the regional distribution centers. |
| Successful End Condition | Postmen check delivery information and send packages to the customer. |
| Failed End Condition | None |
| Primary Actors | Postman |
| Secondary Actors | None |
| Trigger | Postmen accept an delivery task. |
| Main Flow | Step 1 : Logistics Company assigns task to Postman.<br>Step 2 : Postmen accept delivery task.<br>Step 3 : Postmen log in the system.<br>Step 4 : Postmen check delivery information including tracking number, destination, personal information about receivers, QR code for payment (if the customer choose to pay on-site) and so on.<br>Step 5 : Postmen go to the regional distribution center and get the package.<br>Step 6 : Postmen deliver it to the destination.<br>Step 7 : Postmen inform customers to take packages. At the same time, Postmen will Inspect Address, if the GPS address of the customer is not in the database, they will collect the GPS address. |
| Extensions | None |

| Use Case Name | Update Database |
|---|---|
| Goal in Context | Server Updates database. |
| Preconditions | Postmen collected the GPS address of the customer and sent to server. |
| Successful End Condition | Server Updates database successfully. |
| Failed End Condition | Server fails to update database. |
| Primary Actors | Server |
| Secondary Actors | None |
| Trigger | The GPS address of the customer is not in the database. |
| Main Flow | Step 1 : Server Updates database. |
| Extensions | None |

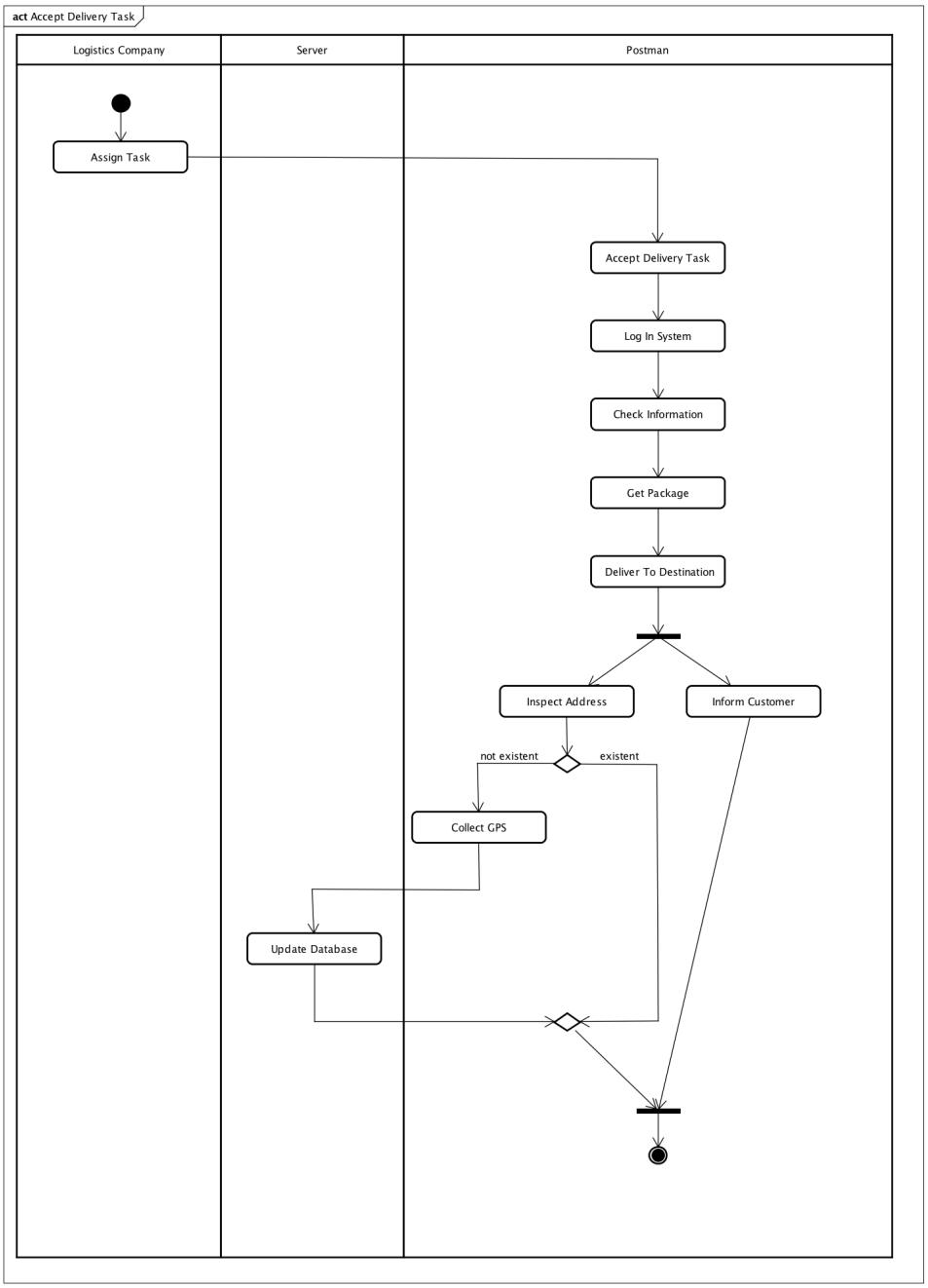



Figure 2.5: Scenario Two Activity Diagram

### Scenario Three

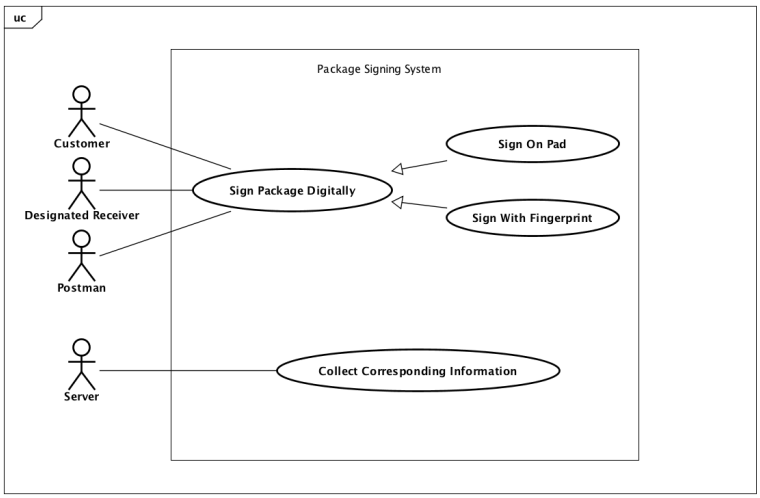



Figure 2.6: Scenario Three Use Case Diagram

| Use Case Name | Collect Corresponding Information |
|---|---|
| Related Requirements | Scenario Two & Three |
| Goal in Context | Collect information and archive related data. |
| Preconditions | Signing package is finished successfully. |
| Successful End Condition | The order is marked as complete; Relavant information is gathered. |
| Failed End Condition | The order is unfinished; Inspect system exception. |
| Primary Actors | Server |
| Secondary Actors | None |
| Trigger | System sends server a message that informs the package delivery is done. |
| Main Flow | Step 1 : Information including the data of received products, customer acceptance and some other transaction details are collected by the server. All necessary related data is archived to database.<br>Step 2 : Server requests to mark this order as complete. |
| Extensions | Step 1.1 : Error appears upon the procedure of data collection.<br>Step 1.2 : Data collection is undone. |

| Use Case Name | Sign on Pad |
|---|---|
| Related Requirements | Scenario Two & Three |
| Goal in Context | A customer or the receiver designated by him/her requests to sign his/her name (or other proofs of identity) on Pad. |
| Preconditions | The package has delivered to the customer-specified location on time. |
| Successful End Condition | The customer receives the package; Package delivery success. |
| Failed End Condition | The customer fails to get the package; Package should be returned or reschedule delivery. |
| Primary Actors | Customer, Package receiving representative. |
| Secondary Actors | Postman |
| Trigger | Postman requests customer to sign on pad with the agreement of customer. |
| Included Cases | Confirm Reception, Collect Corresponding Information |
| Main Flow | Step 1 : Customer checks if the parcel is delivered right and in good condition in person.<br>Step 2 : Customer signs on pad.<br>Step 3 : Postman confirms the package is received by customer or his/her representative.<br>Step 4 : All corresponding information including the data of the received product and customer acceptance will be transferred back to the server; The order will be marked as complete.<br>Include : Collect Corresponding Information |
| Extensions | Step 1.1 : The customer is not convenient to sign the package and asks someone to receive for him/her.<br>Step 2.1 : The representative of customer request to sign the package.<br>Step 2.2 : A confirming message is send to customer and he/she should ensure the package is received by the person he/she designated.<br>Include : Confirm Reception<br>Step 2.3 : The representative signs on pad. |

| Use Case Name | Sign Package Digitally |
|---|---|
| Related Requirements | Scenario Two & Three |
| Goal in Context | A customer requests to sign the package. |
| Preconditions | The package has delivered to the customer-specified location on time. |
| Successful End Condition | The customer receives the package; Package delivery success. |
| Failed End Condition | The customer fails to get the package; Package should be returned or reschedule delivery. |
| Primary Actors | Customer, Package receiving representative. |
| Secondary Actors | Postman |
| Trigger | Postman sends customer the request of signing the package. |
| Included Cases | Confirm Reception, Collect Corresponding Information |
| Main Flow | Step 1 : Customer checks if the parcel is delivered right and in good condition in person.<br>Step 2 : Customer signs the package digitally.<br>Step 3 : Postman confirms the package is received by customer or his/her representative.<br>Step 4 : All corresponding information including the data of the received product and customer acceptance will be transferred back to the server; The order will be marked as complete.<br>Include : Collect Corresponding Information |
| Extensions | Step 1.1 : The customer is not convenient to sign the package and asks someone to receive for him/her.<br>Step 2.1 : The representative of customer request to sign the package.<br>Step 2.2 : Step 2.2 : A confirming message is send to customer and he/she should ensure the package is received by the person he/she designated.<br>Include : Confirm Reception<br>Step 2.3 : The representative signs the package digitally. |

| Use Case Name | Sign with Fingerprint |
|---|---|
| Related Requirements | Scenario Two & Three |
| Goal in Context | A customer or the receiver designated by him/her requests to sign his/her name (or other proofs of identity) on Pad. |
| Preconditions | The package has delivered to the customer-specified location on time. |
| Successful End Condition | The customer receives the package; Package delivery success. |
| Failed End Condition | The customer fails to get the package; Package should be returned or reschedule delivery. |
| Primary Actors | Customer, Package receiving representative. |
| Secondary Actors | Postman |
| Trigger | Postman requests customer to sign via fingerprint with the agreetment of customer. |
| Included Cases | Confirm Reception, Collect Corresponding Information |
| Main Flow | Step 1 : Customer checks if the parcel is delivered right and in good condition in person.<br>Step 2 : Customer signs with fingerprint.<br>Step 3 : Postman confirms the package is received by customer or his/her representative.<br>Step 4 : All corresponding information including the data of the received product and customer acceptance will be transferred back to the server; The order will be marked as complete.<br>Include : Collect Corresponding Information |
| Extensions | Step 1.1 : The customer is not convenient to sign the package and asks someone to receive for him/her.<br>Step 2.1 : The representative of customer request to sign the package.<br>Step 2.2 : A confirming message is send to customer and he/she should ensure the package is received by the person he/she designated.<br>Include : Confirm Reception<br>Step 2.3 : The representative signs with fingerprint. |

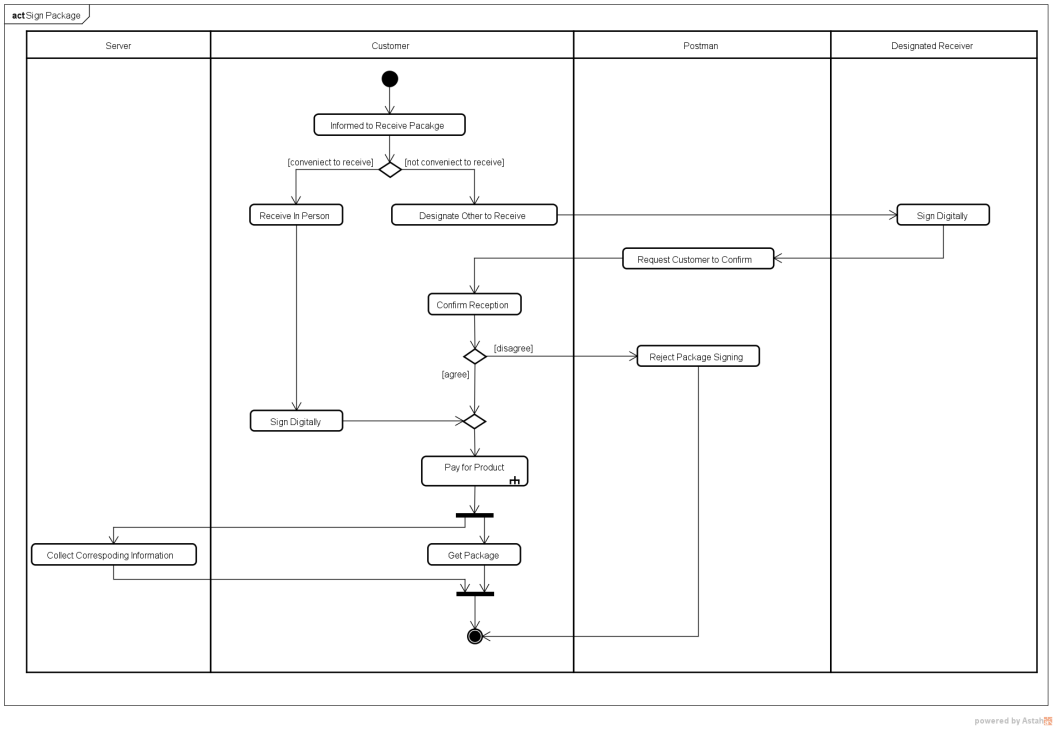



Figure 2.7: Scenario Three Activity Diagram

## Scenario Four

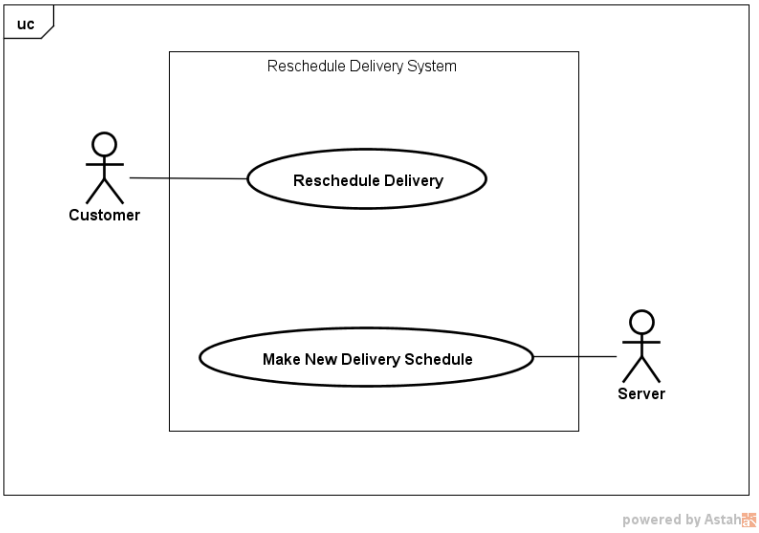



Figure 2.8: Scenario Four Use Case Diagram

| Use Case Name | Make New Delivery Schedule |
|---|---|
| Related Requirements | Scenario Four |
| Goal in Context | Rearrange package delivery. |
| Preconditions | No one receives the package. |
| Successful End Condition | A new delivery schedule is made; Parcel would be sent again. |
| Failed End Condition | Delivery reschedule failed. |
| Primary Actors | Server |
| Secondary Actors | Customer, Postman |
| Trigger | Server make a request of reschedule package delivery. |
| Main Flow | Step 1 : Server puts forward a new schedule.<br>Step 2 : Server sends the new plan to customer to see if customer is available then.<br>Step 3 : Customer replies yes.<br>Step 4 : Postman sends the package again according to new schedule that specified by customer. |
| Extensions | Step 3.1 : Customer replies no.<br>Step 3.2 : Made new delivery schedule again, or cancel the order. |

| Use Case Name | Reschedule Delivery |
|---|---|
| Related Requirements | Scenario Four |
| Goal in Context | Rearrange package delivery. |
| Preconditions | Customer does not receive the package. |
| Successful End Condition | A new delivery schedule is made; Parcel would be sent again. |
| Failed End Condition | Delivery reschedule failed. |
| Primary Actors | Customer |
| Secondary Actors | Server |
| Trigger | Customer requests to reschedule package delivery. |
| Main Flow | Step 1 : Customer requests the server to reschedule package delivery, sends a new time that he/she is available (and a new location if he/she needs).<br>Step 2 : Server makes a new delivery plan.<br>Include : Made New Delivery Schedule<br>Step 3 : Customer agrees.<br>Step 4 : New Schedule would be executed. |
| Extensions | Step 3.1 : Customer is not satisfied with the new schedule.<br>Step 3.2 : Made new delivery schedule again, or cancel the order. |

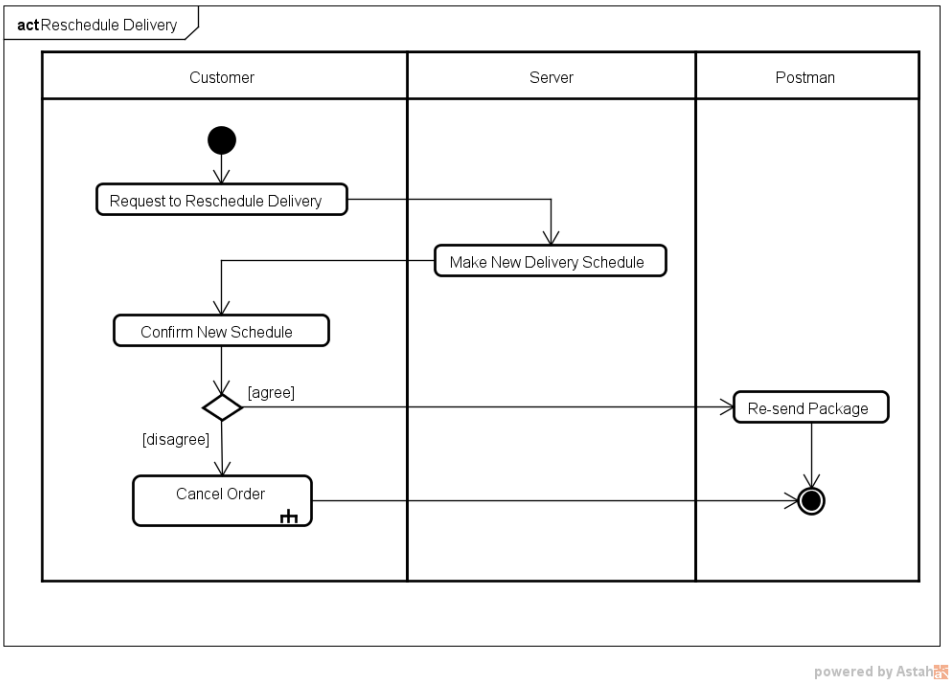



Figure 2.9: Scenario Four Activity Diagram

## Scenario Five

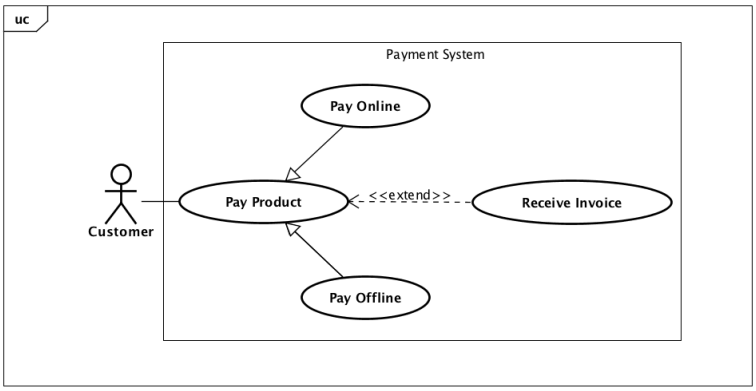



Figure 2.10: Scenario Five Use Case Diagram

| Use Case Name | Pay Online |
|---|---|
| Related Requirements | Scenario Five |
| Goal in Context | The customer wants to pay for the product online after he/she has placed the order. |
| Preconditions | The customer has placed the order, and paying online is provided. |
| Successful End Condition | The payment is completed. |
| Failed End Condition | The order has been canceled. |
| Primary Actors | Customer |
| Secondary Actors | None |
| Trigger | The customer places the order. |
| Main Flow | Step 1 : Customer logins the Third-party Trade system.<br>Step 2 : Customer checks the amount in the system.<br>Step 3 : Customer enters the passwords to pay.<br>Step 4 : Customer finishes the payment.<br>Step 5 : E-business receives the payment. |
| Extensions | None |

| Use Case Name | Pay for Product |
|---|---|
| Related Requirements | Scenario Five |
| Goal in Context | The customer needs to pay for the product after he/she has placed the product or received the product. |
| Preconditions | The customer has placed the order. |
| Successful End Condition | The payment is completed. |
| Failed End Condition | The order has been canceled or the customer has not received the product for some reasons. |
| Primary Actors | Customer |
| Secondary Actors | None |
| Trigger | The customer places the order or receives the product. |
| Main Flow | Step 1 : Customer chooses one acceptable way to pay for the product.<br>Step 2 : Customer checks the amount.<br>Step 3 : Customer finishes the payment.<br>Step 4 : E-business receives the payment. |
| Extensions | Step3.1 : Customer asks for a invoice about the product. |

| Use Case Name | Pay Offline |
|---|---|
| Related Requirements | Scenario Five |
| Goal in Context | The customer wants to pay for the product offline after he/she has received the product. |
| Preconditions | The customer has received the product, and paying offline is provided. |
| Successful End Condition | The payment is completed. |
| Failed End Condition | The customer has not received the product for some reasons. |
| Primary Actors | Customer |
| Secondary Actors | Postman |
| Trigger | The customer receives the product. |
| Main Flow | Step 1 : Postman checks the amount with the customer.<br>Step 2 : Customer chooses one way to pay, by cash or by credit card.<br>Step 3 : Customer finishes the payment.<br>Step 4 : Postman records the information about the payment.<br>Step 5 : The logistics company transfers the money to E-business. |
| Extensions | Step 2.1 : Customer chooses to pay by cash.<br>Customers pays for the product by cash.<br>Postman checks the amounts of money and confirms the payment in the app.<br>Step 2.2 : Customer Chooses to pay by credit card.<br>Postman provide the POS device for the customer.<br>Customer gives the postman his/her Visa card or Union-Pay card.<br>Postman swipes the card and enters the amount of money.<br>Customer enters the password.<br>The bank system adds money to the logistics companys account.<br>Step 4.1 : Postman submits the money to the logistics company. |

| Use Case Name | Provide Invoice |
|---|---|
| Related Requirements | Scenario Five |
| Goal in Context | E-business provides the invoice about the product for the customer. |
| Preconditions | The customer has completed the payment and asks for the invoice. |
| Successful End Condition | The Invoice has been sent to the customer. |
| Failed End Condition | E-business rejects the request. |
| Primary Actors | E-business |
| Secondary Actors | Customer |
| Trigger | Customer asks for the invoice about the product. |
| Main Flow | Step 1 : Customer choose one way to receive the invoice, digital invoice sent by e-mail or a paper invoice sent by post.<br>Step 2 : E-business sends the invoice to the customer.<br>Step 3 : Customer receives the invoice and confirm whether the information is consistent. |
| Extensions | Step 1.1 : Customer chooses to receive a digital invoice sent by e-mail.<br>Customer fills in the e-mail address in the app.<br>Step 1.2 : Customer chooses to receive a paper invoice sent by post.<br>Customer fills in the mailing address in the app. |

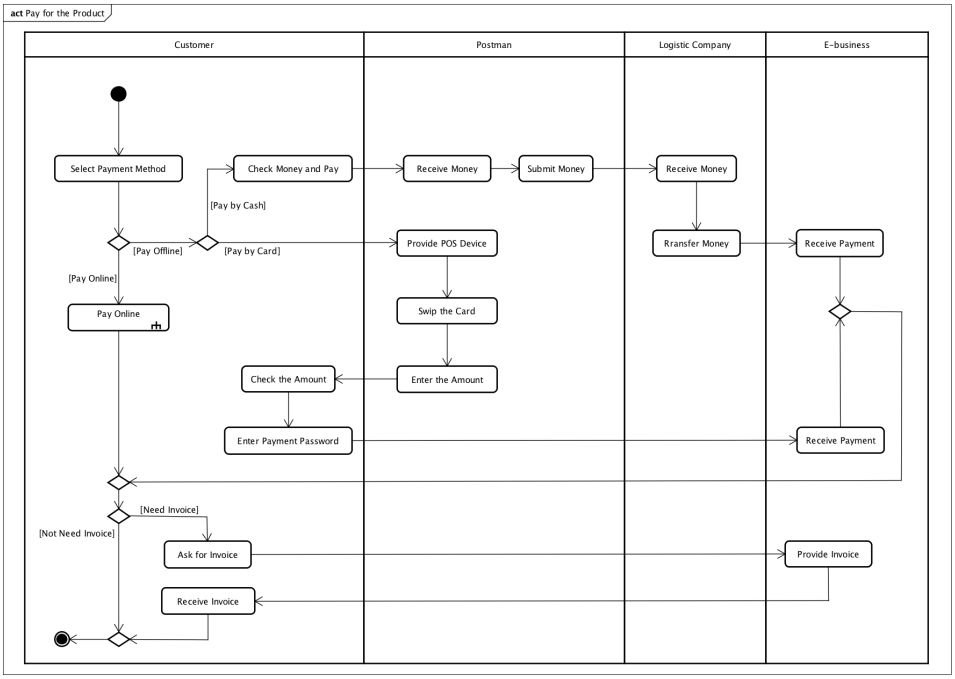



Figure 2.11: Scenario Five Activity Diagram: Pay The Product

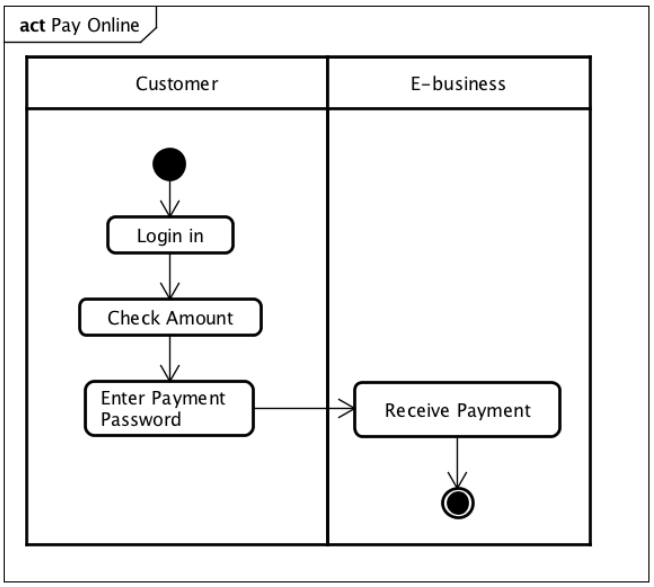



Figure 2.12: Scenario Five Activity Diagram: Pay Online

### Scenario Six

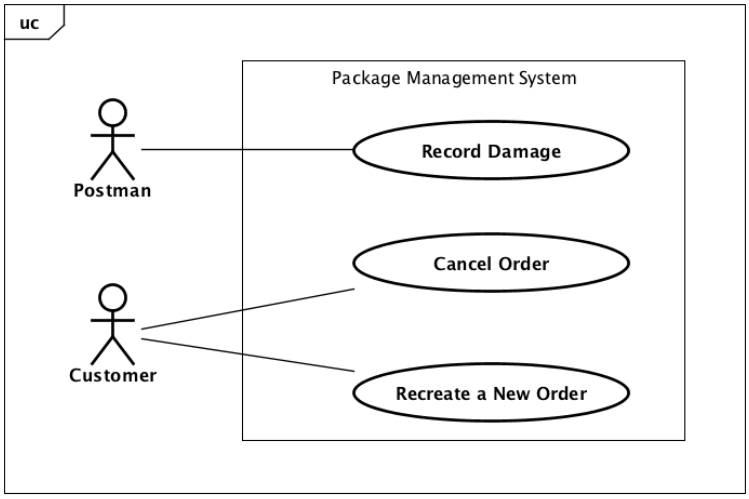



Figure 2.13: Scenario Six Use Case Diagram

| Use Case Name | Record Damage |
|---|---|
| Related Requirements | Scenario Six |
| Goal in Context | Postman records the information of the damaged or lost package. |
| Preconditions | The package was damaged or lost. |
| Successful End Condition | Postman records the infomation successfully. |
| Failed End Condition | The information of the package was lost as well. |
| Primary Actors | Postman |
| Secondary Actors | None |
| Trigger | The package was damaged or lost. |
| Main Flow | Step 1 : Postman records the information of the package.<br>Step 2 : The logistics company receives the corresponding information.<br>Step 3 : The logistics company sends a message to the customer, explaining the situation and making an apology to the customer. |
| Extensions | None |

| Use Case Name | Cancel Order |
|---|---|
| Related Requirements | Scenario Six |
| Goal in Context | The customer wants to cancel the order for some reasons. |
| Preconditions | The customer has placed the order. The package has not been shipped or the package was damaged or lost during process of delivery. |
| Successful End Condition | The logistics company passes the request. |
| Failed End Condition | The request is rejected. |
| Primary Actors | Customer |
| Secondary Actors | None |
| Trigger | The customer makes the request of canceling the order. |
| Main Flow | Step 1 : Customer logins the logistics system.<br>Step 2 : Customer selects the button "cancel the order".<br>Step 3 : Customer explains the reason.<br>Step 4 : The logistics company receives the request and passes the request. |
| Extensions | None |

| Use Case Name | Recreate a New Order |
|---|---|
| Related Requirements | Scenario Six |
| Goal in Context | The customer wants to recreate a new order without any extra fees because the old package was damaged or lost. |
| Preconditions | The old package was damaged or lost and the customer has received the message from the logistics company. |
| Successful End Condition | The new order is placed successfully. |
| Failed End Condition | The new order is non-compliant and rejected. |
| Primary Actors | Customer |
| Secondary Actors | None |
| Trigger | The customer makes the request of recreating a new order. |
| Main Flow | Step 1 : Customer receives the message about the infomation of the damaged or lost package.<br>Step 2 : Customer logins the logistics system.<br>Step 3 : Customer chooses to deliver a new package.<br>Step 4 : Customer fills in the basic information and recreates a new order without any extra fees.<br>Step 5 : The logistics company receives the new order and passes the request. |
| Extensions | None |

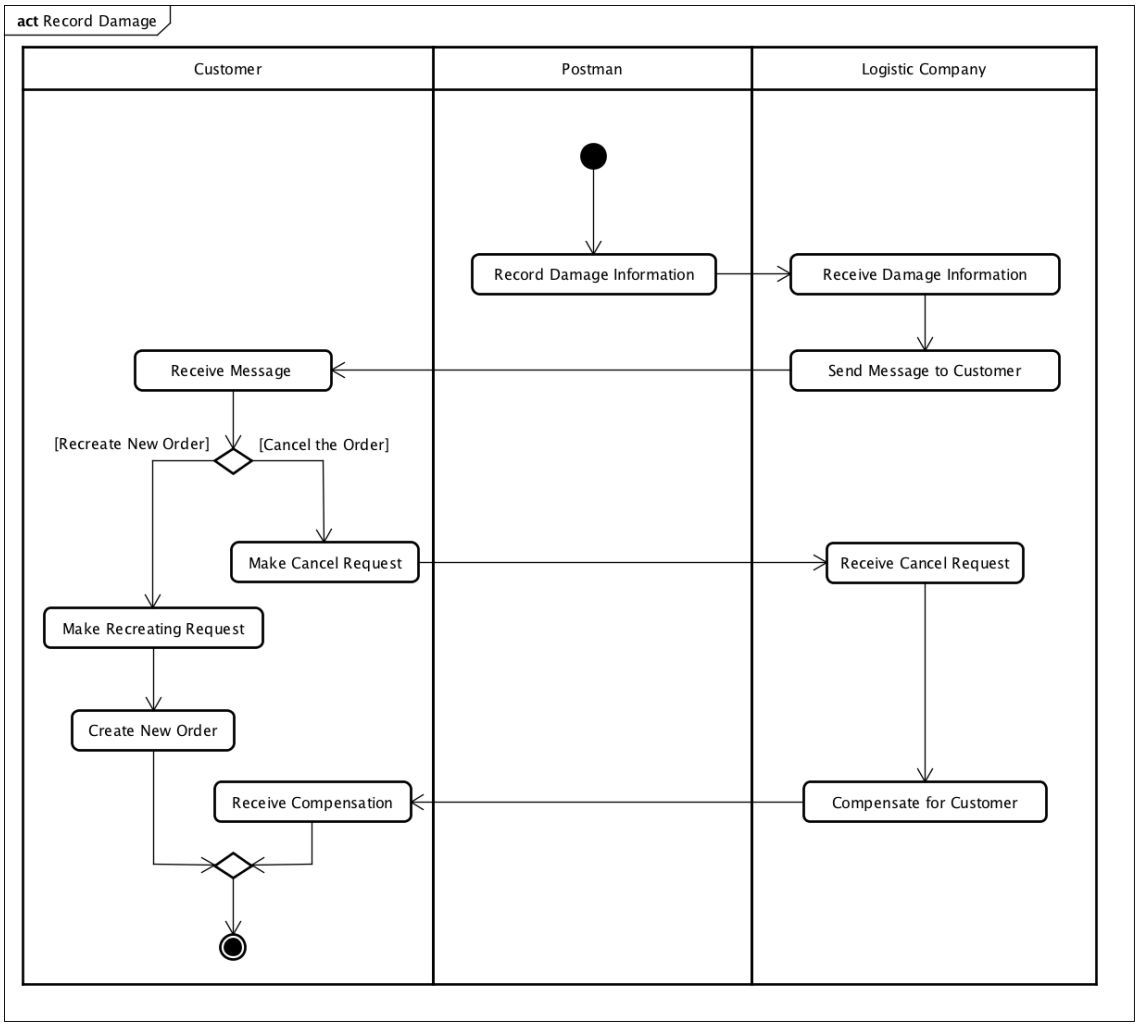



Figure 2.14: Scenario Six Activity Diagram

### Scenario Seven

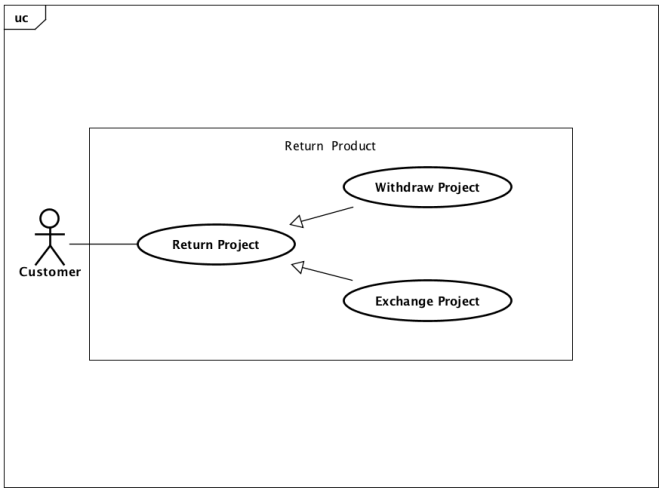



Figure 2.15: Scenario Seven Use Case Diagram

| Use Case Name | Withdraw Product |
|---|---|
| Related Requirements | Scenario Seven |
| Goal in Context | A customer returns the product and the payment is reimbursed. |
| Preconditions | The customer wants to withdraw the product the or she unsatisfied with after receiving within a month. |
| Successful End Condition | The order is completed and the payment is reimbursed to the customer. |
| Failed End Condition | The order is not completed and the payment is not reimbursed to the customer. |
| Primary Actors | Customer |
| Secondary Actors | None |
| Trigger | The customer choose to withdraw the product. |
| Main Flow | Step 1 : Customer makes the returning request.<br>Step 2 : Server checks the returning request.<br>Step 3 : The package is delivered.<br>Step 4 : The E-business receives and checks the pack.<br>Step 5 : The E-business agrees the returning.<br>Step 6 : Server make the new delivery schedule. |
| Extensions | Step 2.1 : The database does not verify the details.<br>Step 3.1 : The package fails to be delivered.<br>Step 5.1 : The E-business doesn't agree the withdrawing. |

| Use Case Name | Exchange Product |
|---|---|
| Related Requirements | Scenario Seven |
| Goal in Context | A customer returns the product and the delivery schedule is made again upon mutual agreement. |
| Preconditions | The customer wants to exchange the product the or she unsatisfied with after receiving within a month. |
| Successful End Condition | The delivery schedule is made again upon mutual agreement. |
| Failed End Condition | The delivery schedule fails to be made again upon mutual agreement. |
| Primary Actors | Customer |
| Secondary Actors | None |
| Trigger | The customer choose to exchange the product. |
| Main Flow | Step 1 : Customer makes the returning request.<br>Step 2 : Server checks the returning request.<br>Step 3 : The package is delivered.<br>Step 4 : The E-business receives and checks the pack.<br>Step 5 : The E-business agrees the returning.<br>Step 6 : Server make the new delivery schedule. |
| Extensions | Step 2.1 : The database does not verify the details.<br>Step 3.1 : The package fails to be delivered.<br>Step 5.1 : The E-business doesn't agree the exchange. |

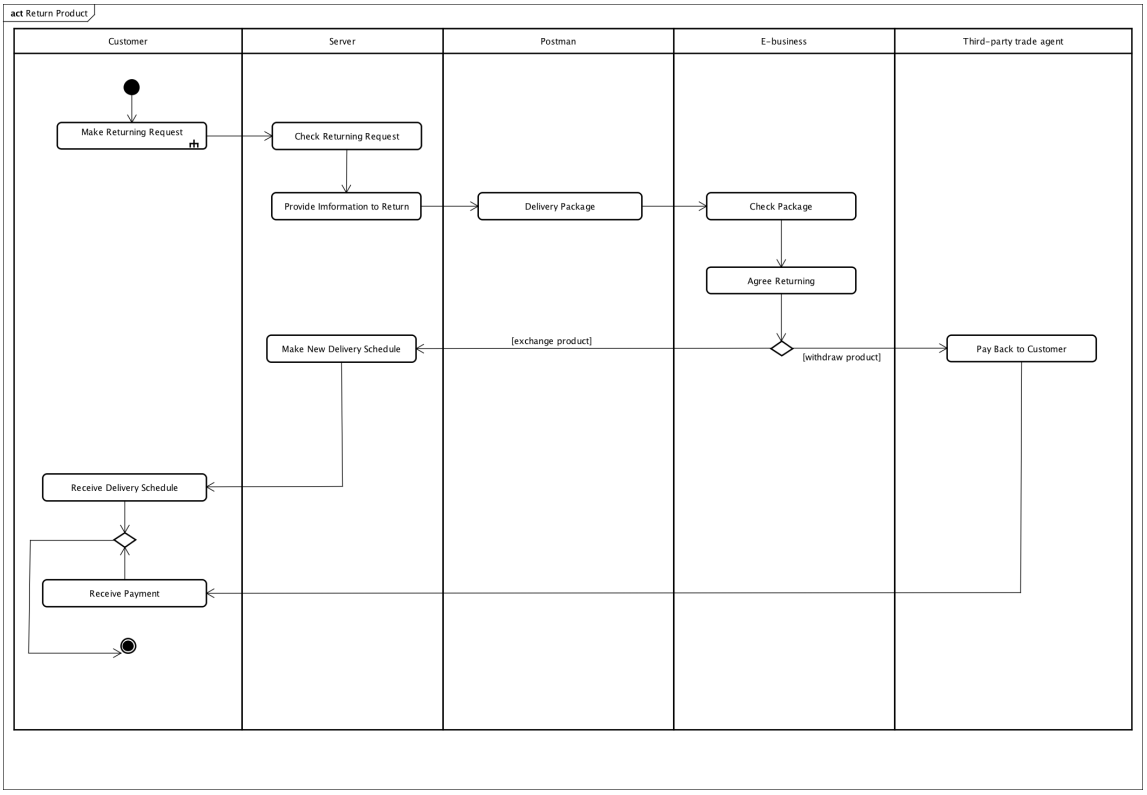



Figure 2.16: Scenario Seven Activity Diagram: Return Product

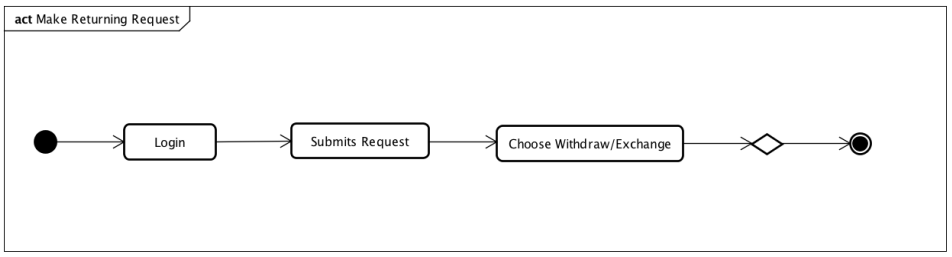



Figure 2.17: Scenario Seven Activity Diagram: Make Returning Request

### Scenario Eight

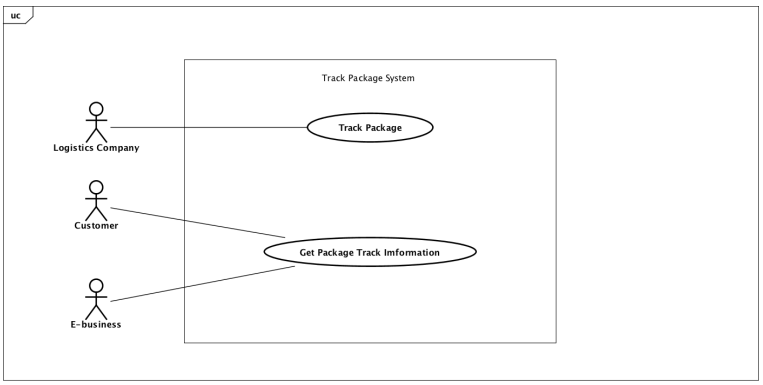



Figure 2.18: Scenario Seven Use Case Diagram

| Use Case Name | Track Package |
|---|---|
| Related Requirements | Scenario Eight |
| Goal in Context | The logistics tracks all the packages, collecting the location at fixed period and inserting new addresses of destinations into the database. |
| Preconditions | The customer or the e-business company deliver the package. |
| Successful End Condition | The logistics tracks all the packages and feeds back to the customer or the E-business. |
| Failed End Condition | The logistics fails track all the packages and feed back to the customer or the E-business. |
| Primary Actors | Logistics |
| Secondary Actors | None |
| Trigger | Postman receives the package from the customer. |
| Main Flow | Step 1 : Postman receives and expresses the package to the logistics.<br>Step 2 : Logistics delivers the package to the express station.<br>Step 3 : The express station scans the bar code.<br>Step 4 : Server records the express station information of the package.<br>Step 5 : Logistics delivers the package to the next express station. |
| Extensions | Step 3.1 : The express station fails to scan the bar code and gets a new bar code. |

| Use Case Name | Get Package Track Information |
|---|---|
| Related Requirements | Scenario Eight |
| Goal in Context | The customer or the e-business can get track Information of the package delivered. |
| Preconditions | 1. The returning request is admitted.<br>2. The customer or E-business delivers the returning product.<br>3. The package has not been transferred to the receiver.<br>4.The customer or the E-business has obtained the tracking number.<br>5.The logistics tracks all the packages, collecting the GPS location at fixed period and inserting new GPS addresses of destinations into the database automatically. |
| Successful End Condition | The customer or the e-business successfully tracks the package he or she delivered. |
| Failed End Condition | The customer or the e-business successfully failed to track the package he or she delivered. |
| Primary Actors | Customer, E-business |
| Secondary Actors | None |
| Trigger | The customer or the E-business logins the system. |
| Main Flow | Step 1 : The customer or the E-business logins the system.<br>Step 2 : The customer or the E-business inputs the tracking number.<br>Step 3 : The customer or the E-business get package track information. |
| Extensions | Step 1.1 : The customer or the E-business failed to login the system with the wrong account and password. |

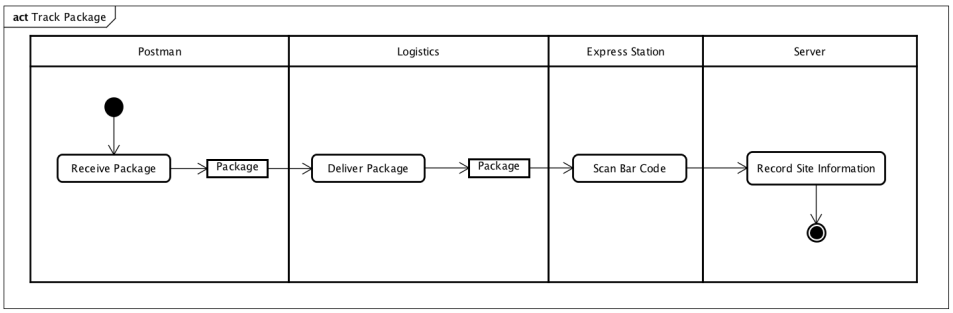



Figure 2.19: Scenario Seven Activity Diagram: Track Package

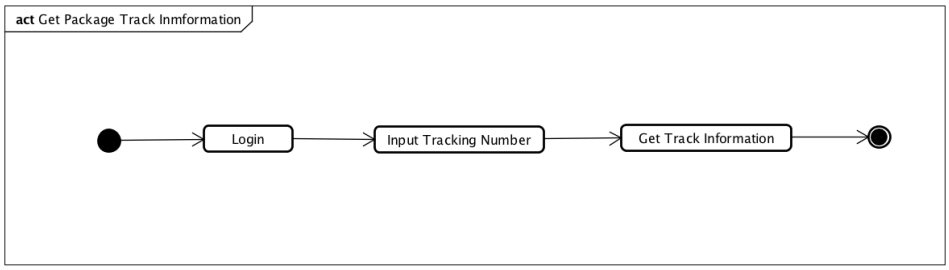



Figure 2.20: Scenario Seven Activity Diagram: Get Package Track Information

## 2.2 Supplementary Specification

### 2.2.1 Security

The system should avoid the database being attacked and data being taken advantage of by the wicked.

#### Access and Data Integrity

1. The authorization of access to the system of postmen, customers and customer servers should be classified and announced clearly. With certain authorization, different users have limited access to data and operation.
2. The server should use anti-virus software.
3. Firewalls and network protection are necessary, and they should be updated in time.
4. The atomic processes in the database will ensure the accuracy of the database.

#### Encryption

1. The session should not be transmitted in DNS.
2. All texts and messages should be encrypted with Encryption Algorithm such as RSA, 3DES or IDEA.
3. Two keys are used to identify a certain user. One public key is used for encryption and anther private key is used for decryption. The key is a completely random mix of letters.
4. The session will record the activity of the customer, and if the customer has no operation for 5 minutes, he or she will log out the system automatically.
5. After customers log out the system, all the private information(cookies) will be cleaned.

#### Digital Certificates

1. We use digital certificates as a replacement of user names and passwords, for example, SSL Certificates. It will be used automatically with the permission of users.
2. The IP address or location where users log in the system will be recorded and when the account is used beyond their regular locations, the user will get alarmed.

#### Digital Signatures

1. Users should log in the system with a password. Our system will test its complexity. If it is too simple, the system will remind the users to complicate it. That involves cryptography.
2. We use a message digest to ensure the integrality of the data.
3. If necessary, we can extend our fingerprint system to login system.

### 2.2.2 Performance

1. The information of the package, including the real-time position, Order-ID, the postman etc., should be checked by customers in 3 seconds with at most 0.1% error rate.
2. The payment should be confirmed in 2 seconds by the system from the moment when the third party trade agent sends the message or the postmen report the payment.
3. The order created by customers should be processed in 15 minutes.
4. The orders obtained from e-business should be processed every hour(about 5,000 orders).

5. Information of the delivery such as the phone number, the address, the receiver and others should be updated and checked by postman in 1 min.

6. This system allows the e-business to create batch orders which can be sent at regular time.

7. The estimate of delivery time should be accurate with the max uncertainty in 2 days.

8. The expectation should be sent to custom service in 2 min from the time a postman reports it.

9. This systems unavailable time should be controlled in 20 minutes in a year.

10. To offer the best user experience, a content delivery network should be used by this system.

### 2.2.3 Data Storage and Computing

1. To store a huge amount of data, distributed database should be used. And it should use Homogeneous Distributed Databases Management System.

2. Considering that there may be an enormous number of visitors and inquiries at the same time, the system must implement cloud computing service.

3. The system can support as many as 1500 times of visits per second.

4. There must be a copy of the database, including device entity, software, data and even employees, in order to prevent some unpredictable disasters.

5. If the database is destroyed, the copy should be enabled in 3 hours.

6. The data can be in English, Chinese, Japanese, French and Korean.

### 2.2.4 Track the Package

1. In order to track the package, the GIS system should be applied, with the help of the GPS system. The system gets geographic information from a third party system, and get the position of postmen who deliver the package through the system of postmen. And this system should match both kinds of the information and show it to users of the system.

2. The system for postmen should upload the position of the postman automatically every 2 hours, through 3G, 4G or WLAN network.

3. If the locations of postmen are missing for 4 hours, the system should inform the custom servers, and custom servers will contact with postmen.

### 2.2.5 Maintenance

1. The distributed database should be maintained by the employees of our own company including the employees of the standby database every day when the visiting traffic is not heavy.

2. The software for custom service, customer and postmen and the system itself should be maintained by our employees.

3. The geographic information source should be multiple, in case that one of the sources is unavailable.

4. The engineers from the company offered DBMS will maintain our system every year.

5. An integrated scheme to deal accidents, for example, the crash of database, is necessary.

### 2.2.6 Others

1. The architectures of the postman app and the customer app are B/S and C/S, but that of custom service is C/S for safety.

2. Our system can be used in iOS and Android on mobile devices and in a normal browser on PC(Windows/macOS/Unix).

3. Anticipated development time is two months.

# 3 Analysis

## 3.1 Architectural Analysis

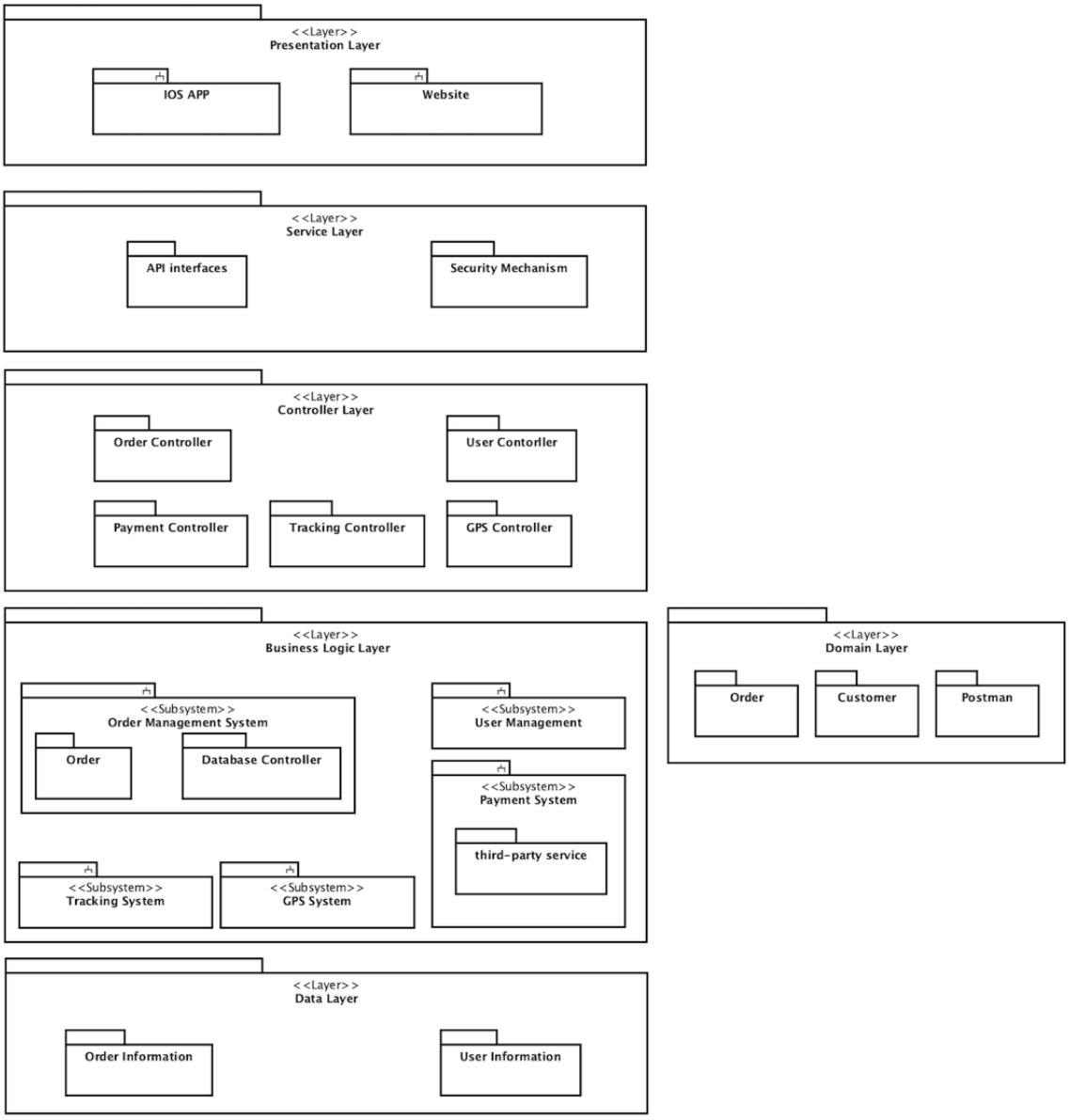



Figure 3.1: Package Diagram

### 3.1.1 System Structure

The architecture has 5 layers, which are extended from the 4-tier architecture. We have taken extensibility, code reusability into consideration. 5 layers are as follows.

#### Presentation Layer

The first layer is Presentation Layer. This layer provides a user interface to translate the logistics information to user friendly representation. It contains Web front-end and iOS app, serving as user's entry of the system. It transfers the information the transport layer and call the API, provided by the transport layer. The layer in out system consists of two subsystems, iOS app and Website. As for the platforms consideration, the website will take advantage of Bootstrap framework, using the language of HTML, CSS and JavaScript. The iOS app will use the Cocoa Touch, Media, Core Services and Core OS in the development.

#### Service Layer

The second layer is Service Layer. This layer works as a coordinator of our system. It provides some Application Programming Interface(API) to the Presentation Layer, including Web API, iOS API and external API. The external API provides interfaces to some third-party services, such as Alipay API provides the services for pay via Alipay Account. This layer plays a role of delivering the request from Presentation Layer to Business Logic Layer. We have designed protocols and different frameworks to implement those API, including Spring Web and UIKit.

#### Business Logic Layer

The third layer is Business Logic Layer. This layer contains several controllers including Order Controller, User Controller, Dispatch Management and Third-Party Service. In our system, these controllers will process the data and make package the update UI operation. This layer is one of the part of MVC(Model-View-Controller) framework.

#### Data Layer

The fourth layer is Data Layer. This layer define the data model of our system, including data access and data maintenance. It contains Order Information and User Information which is entity in our database system. In this layer, we will take advantage of different database and handle different type efficiently(e.g. structural and non-structural data). We will use Oracle DBMS and SQL language. For the data access, we will choose the Hibernate and NHibernate framework.

### Domain Layer

The last layer is Domain Layer. The Domain Layer contains 3 packages including Order, Customer, Postman. This layer contains the concrete information stored in our system. It provides services for the Business Logic Layer by helping it get concrete information. We separate these information into this independent layer, so that there will be less data redundancy and the efficiency of reading or writing will be improved. This layer is also related with the Data Layer, It needs the service and support of the Data Layer.

## 3.2 Analysis Model

### 3.2.1 Global Class Diagram

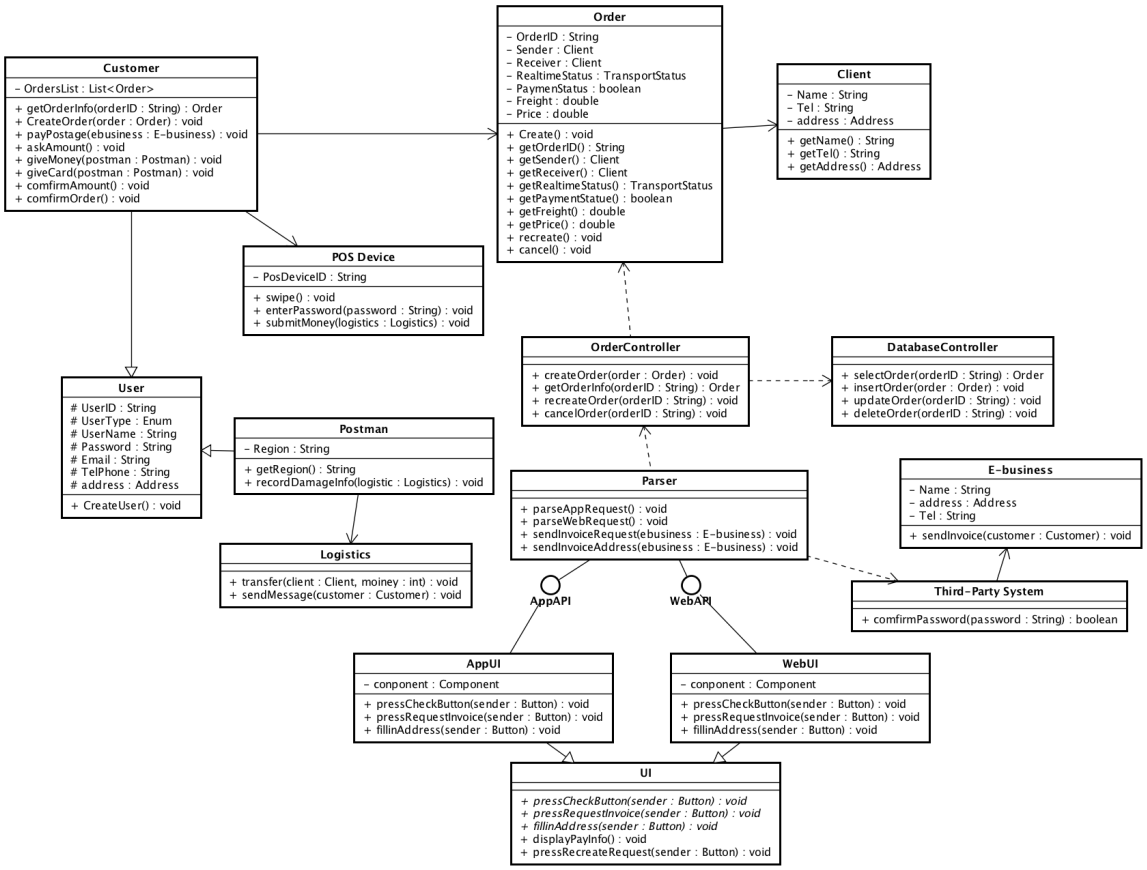



Figure 3.2: Global Class Diagram

The first part is the domain layer, which contains the User Class, Order Class and Client Class. Thus, Customer and Postman is the subclass of User and implement the method. The User Class records the basic information of a user. User ID is the primary key of Use

and User Type is an enum consisting of [**Customer, Postman, Superuser**]. The user can login the system by email or ID or telephone number and password. In the newly created class, it's instance stores the detailed information of the order, including user's name, phone number, and address. This class is designed as a self-define data type and can be used to compare the information of users stored in our system.
The class Customer has a corresponding order list and the attributes to sign the package. The Postman has the attribute region to identify the areas where he is in charge of. We define the public API of User to provide services for the controller.
The class Order record the basic information of an order. The Order ID is the primary key of Order. Since the sender or the receiver may not register an account in the system, the Order class records the name, Tel number and address of sender and receiver. The class Order is designed for containing information about our orders and all the basic services will be offered by the controller, such as [**createOrder(), getOrderInfo()**]. When the user tracks the package, System will return the real-time address of the order.
The second part is the controller classes. All the requests from UI are sent to the Parser. The Parser provides Order Controller APIs to web UI and app UI. It will parse the request and delivery the request to corresponding controller. The DatabaseController collects the necessary information to record in the database and handles the request and returns the result to UI. For example, when the user tracks the package by using the app UI, the app UI will call the app API and sends the request to Parser then to the OrderController.
The third part is the boundary classes containing the UI class. The AppUI provides a lot of functions for users, which is the same with the WebUI.

### 3.2.2 Use Case Realization

#### Scenario One

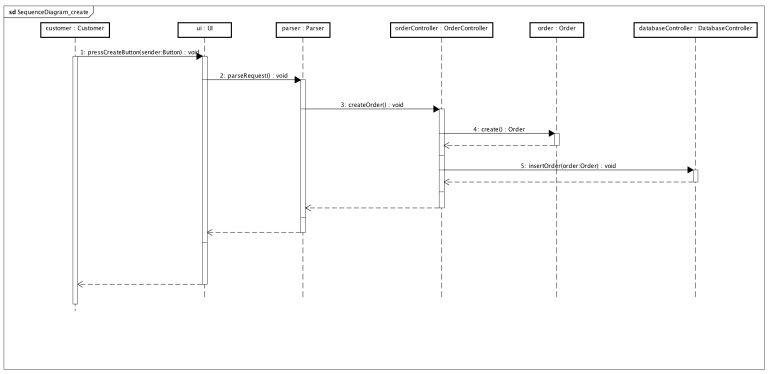



Figure 3.3: Scenario One Sequence Diagram: Create

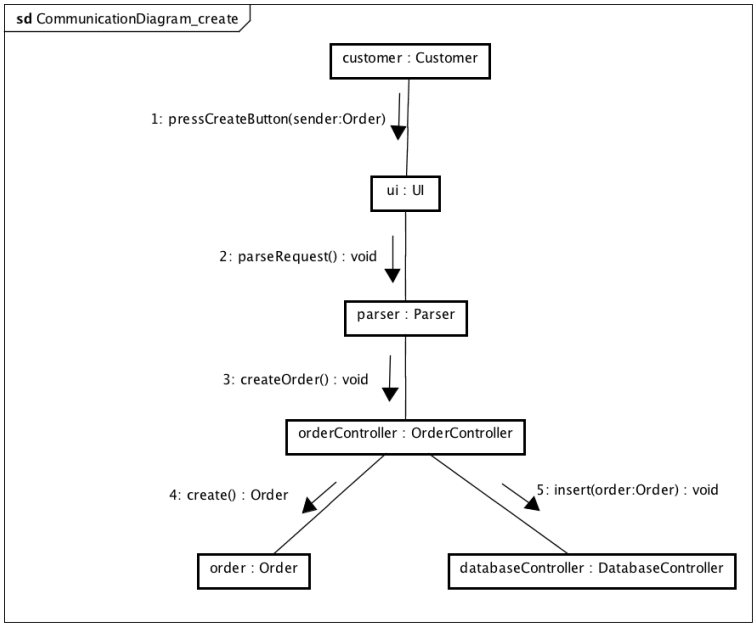



Figure 3.4: Scenario One Communication Diagram: Create

As seen in the communication diagram or sequence diagram, the customer can create order in our order management system. The customer will create order by pressing the corresponding buttons on user interface. After getting the request, user interface will send the request to the parser. Then, the parer send messages to the order controller to create order. The order controller, as seen from its name, will send the creating messages to the class order and control its creation of order, then new order will be created and inserted to database by database controller.

### Scenario Two

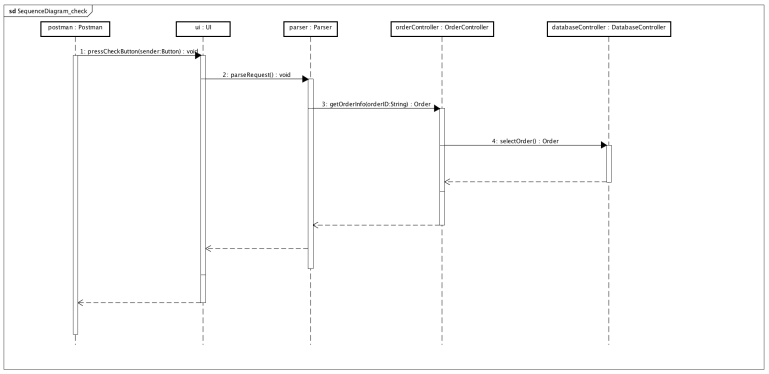



Figure 3.5: Scenario Two Sequence Diagram: Check

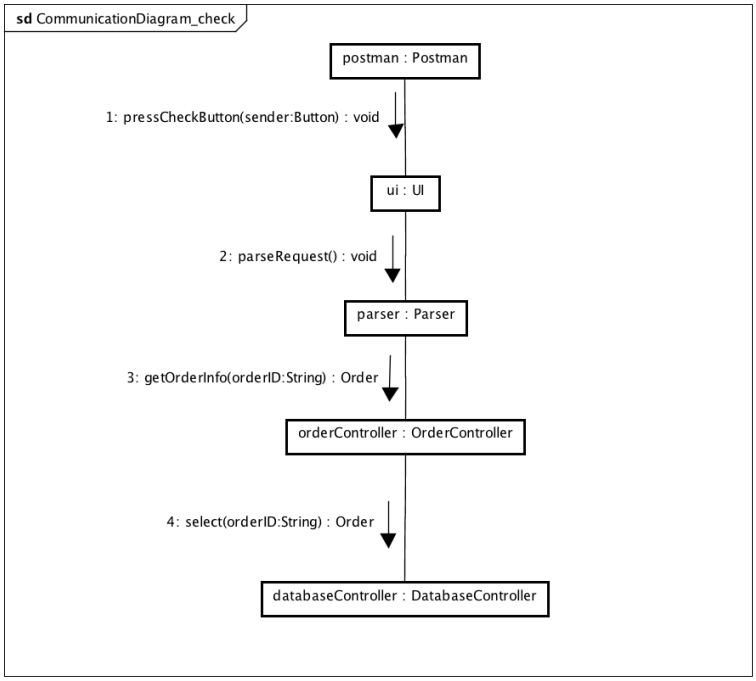



Figure 3.6: Scenario Two Communication Diagram: Check

As seen in the communication diagram or sequence diagram, postman can check order information in our order management system. Postman will check order information by pressing the corresponding buttons on user interface. After getting the request, the user interface will send the request to parser by calling the API. The parser will analyze the request and send messages to the order controller. Then the order controller will call database controller to select corresponding order information from database.

### Scenario Three & Four

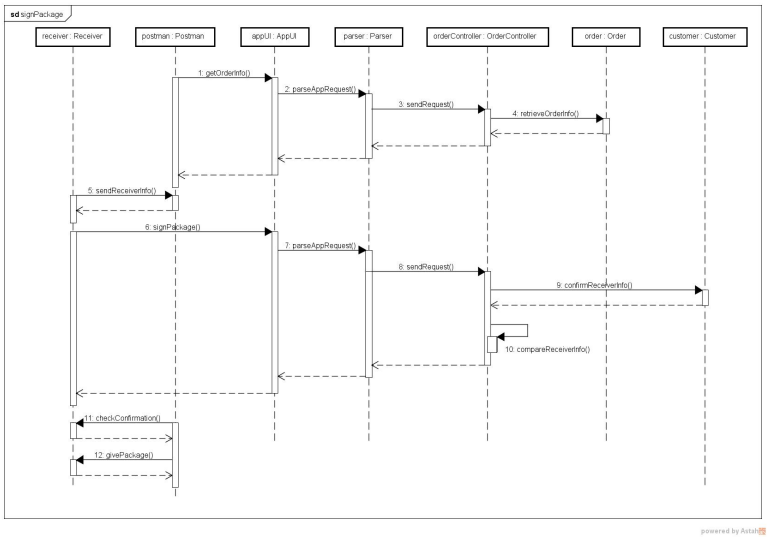



Figure 3.7: Scenario Three & Four Sequence Diagram: Sign Package

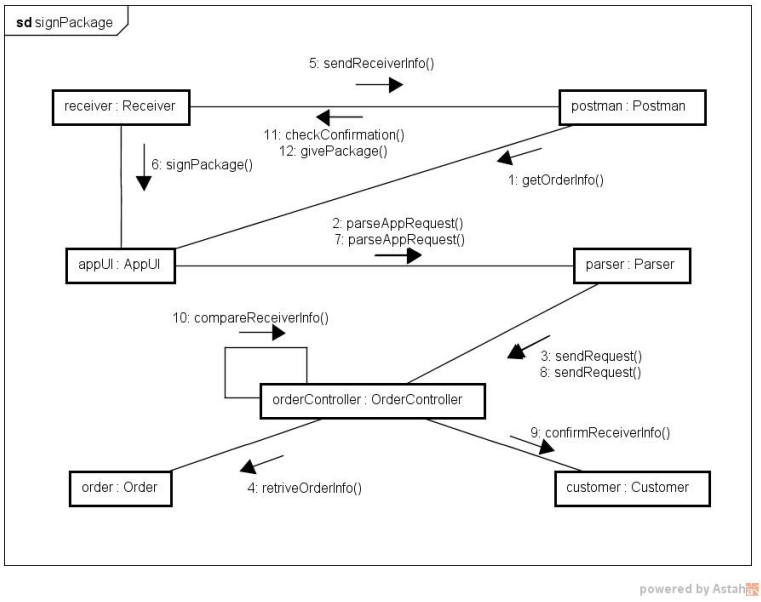



Figure 3.8: Scenario Three & Four Communication Diagram: Sign Package

The process of signing package is shown as the sequence diagram below. First of all, the postman needs to know information of the customer. As a result, a request of retrieving the order information will be sent by postman. The parser here is to analysis requests from UI

and transmit to the controller. Information, including customer's name and phone number, will be returned to the postman. After that, the receiver is supposed to inform the postman her information. In order to verify if she is the designated person or customer herself, a confirmation of customer is required. If so, the receiver will have access to sign the package; otherwise, the demand of signing the package should be rejected.

### Scenario Five

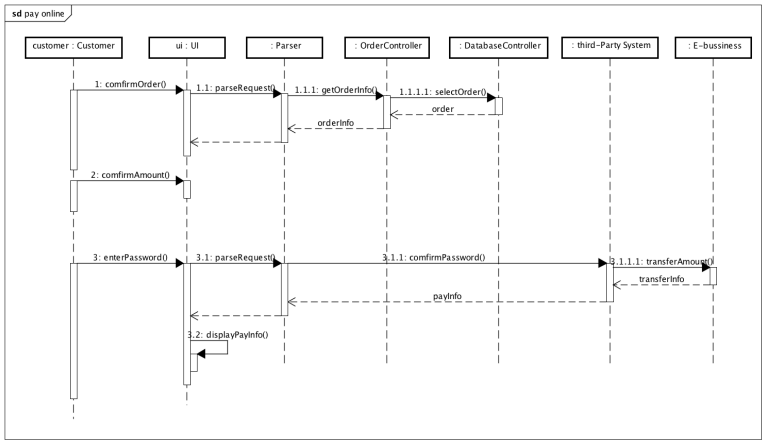



Figure 3.9: Scenario Five Sequence Diagram: Pay Online

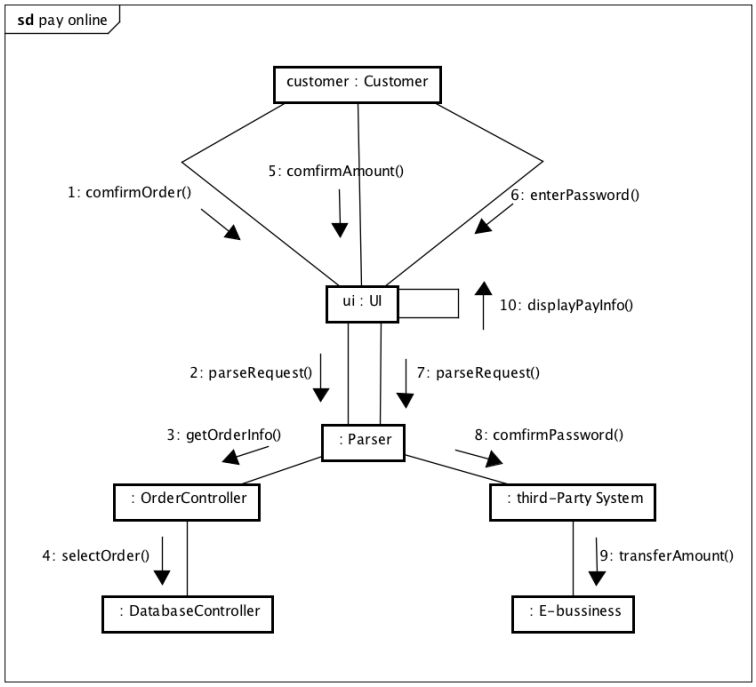



Figure 3.10: Scenario Five Communication Diagram: Pay Online

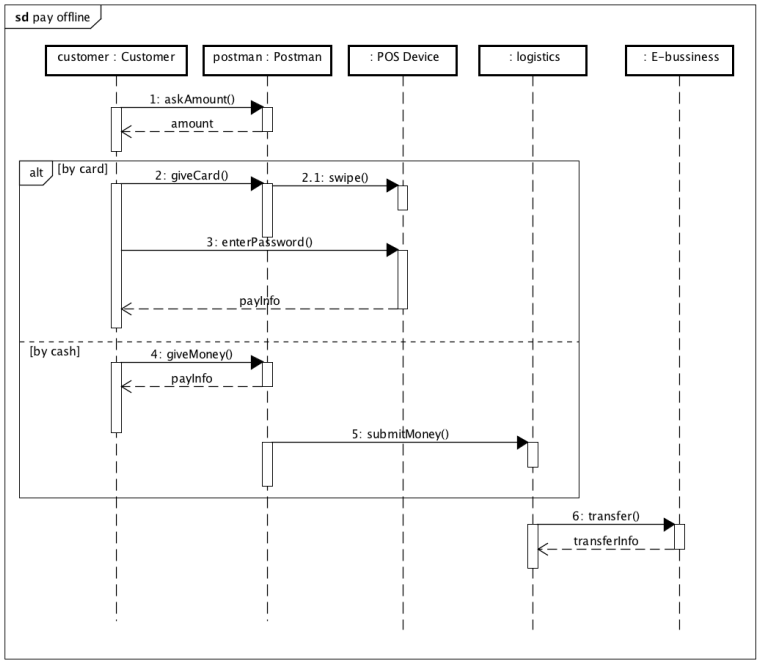



Figure 3.11: Scenario Five Sequence Diagram: Pay Offline

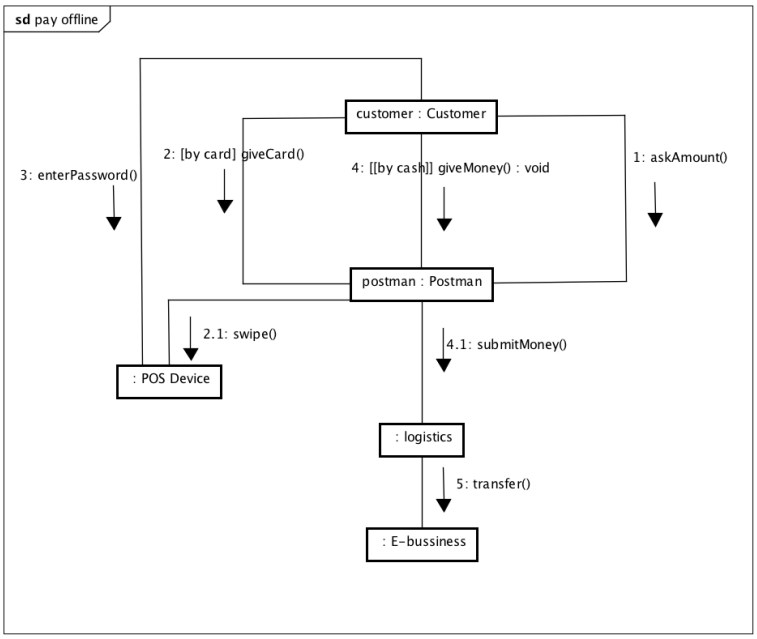



Figure 3.12: Scenario Five Communication Diagram: Pay Offline

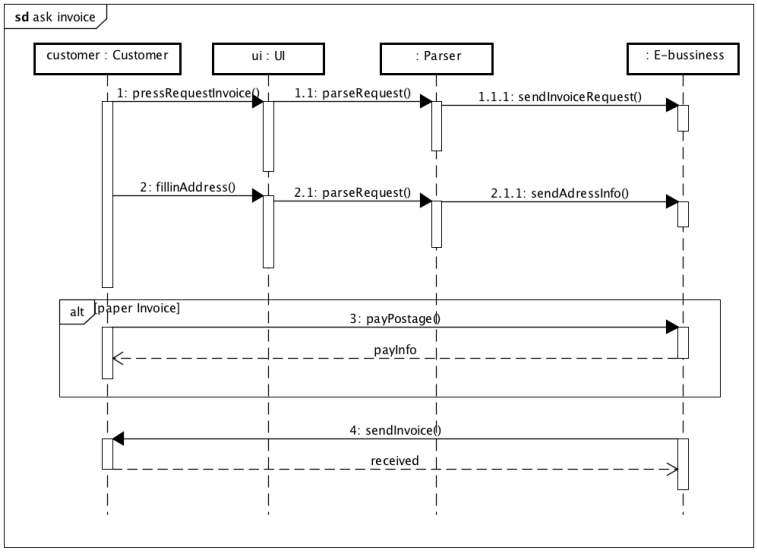



Figure 3.13: Scenario Five Sequence Diagram: Ask Invoice

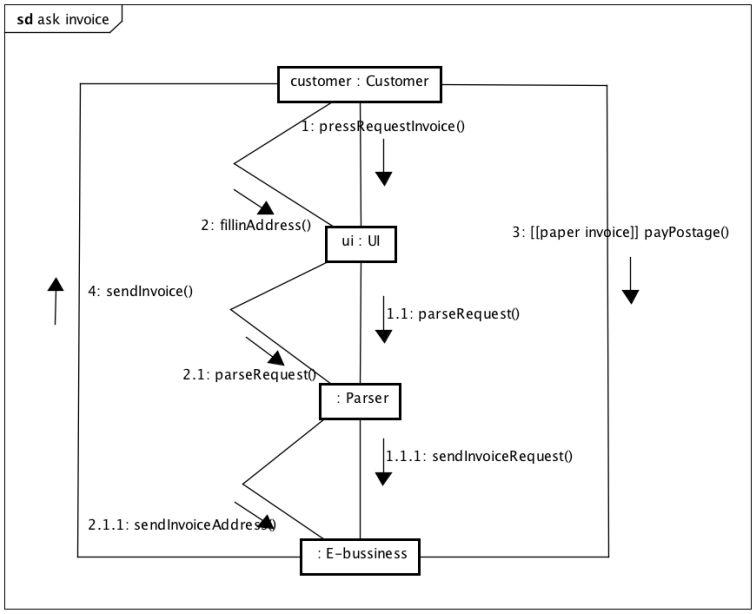



Figure 3.14: Scenario Five Communication Diagram: Ask Invoice

After the customer creates an order, he or she should choose one way to pay for the product, pay online or pay offline. If the customer chooses to pay online, he or she should enter the payment password after confirmed the order. And the Parser will parse the request and send the password to the third-Party System. Then the third-Party System will response the payment information(success or failed) and the ui will display the payment information. On the other hand, if the customer chooses to pay offline, he or she needs to pay after received the product. The customers pays for the product by cash or by card after checked the amount with the postman. Then the postman submits the money to the logistic company. After the third-Party System or logistic company received the amount, it will transfer money to the E-business.
After paid the product, the customer has the opportunity to ask the E-business for an invoice. He or she fills in the address in the app, and then the Parser will parse the request and send the address to the E-business. If the customer wants a paper invoice, he or she needs to pay an extra postage for the invoice. After that, the E-business will send the invoice to the customer.

## Scenario Six

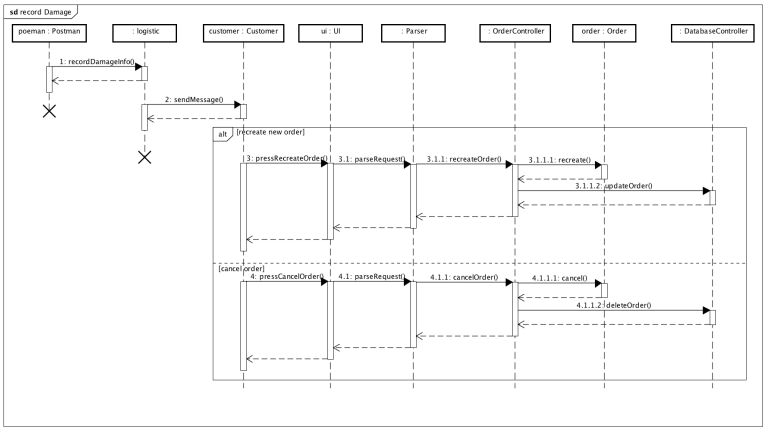



Figure 3.15: Scenario Six Sequence Diagram: Record Damage

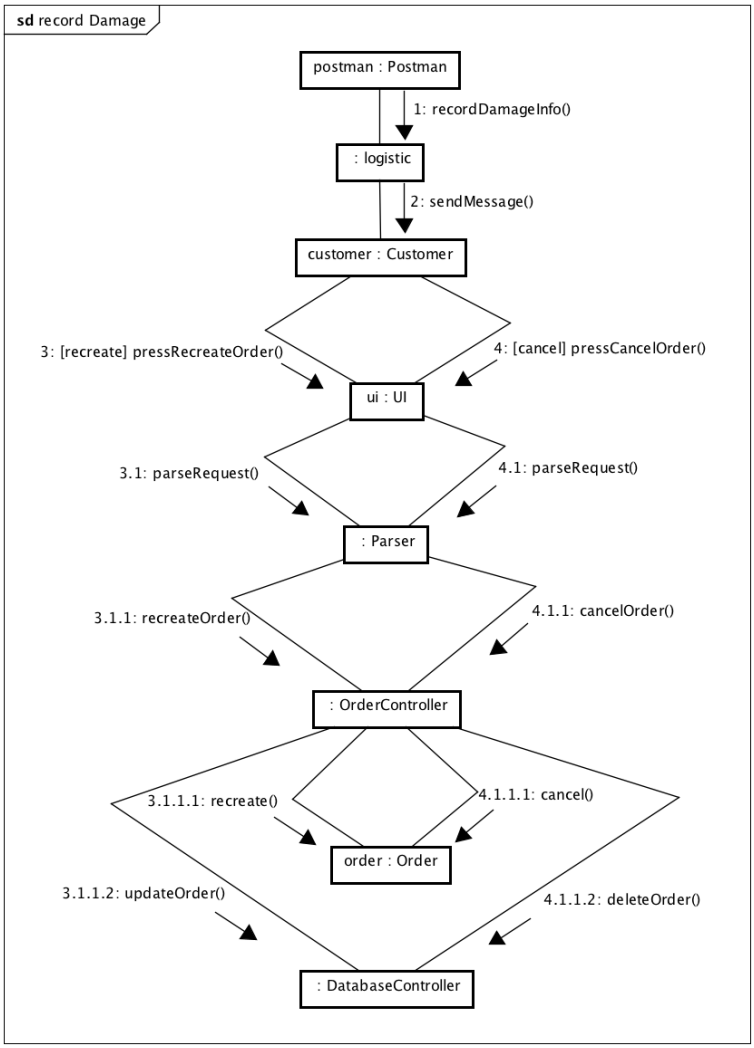



Figure 3.16: Scenario Six Communication Diagram: Record Damage

If the package is damaged or lost, the postman records the damages and sends the damages information to the logistic company. Then the logistic company will send an apology message to the customer. After received the message, the customer can choose to recreate a new order or cancel the order. He or she needs to press the recreate or cancel button in the app, and then the Parser will parse the request and send a signal to the OrderController. After received the signal, the OrderController will execute a function to update or delete the order information in the database.

### Scenario Seven

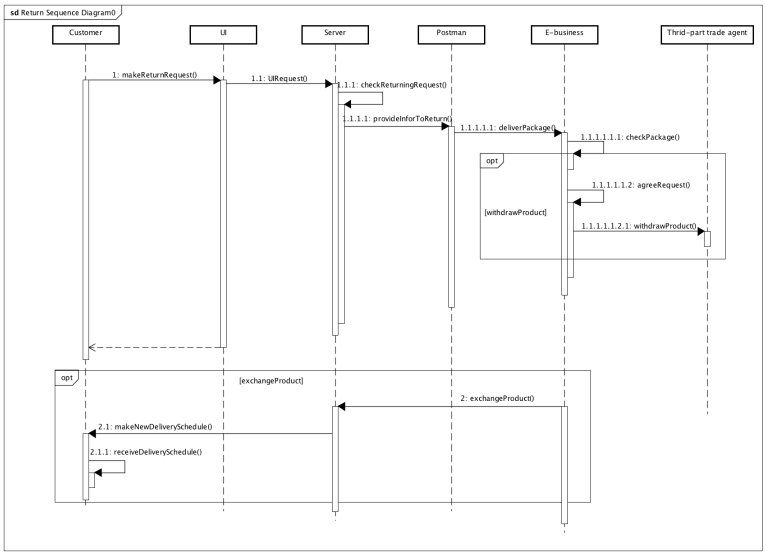



Figure 3.17: Scenario Seven Sequence Diagram: Return

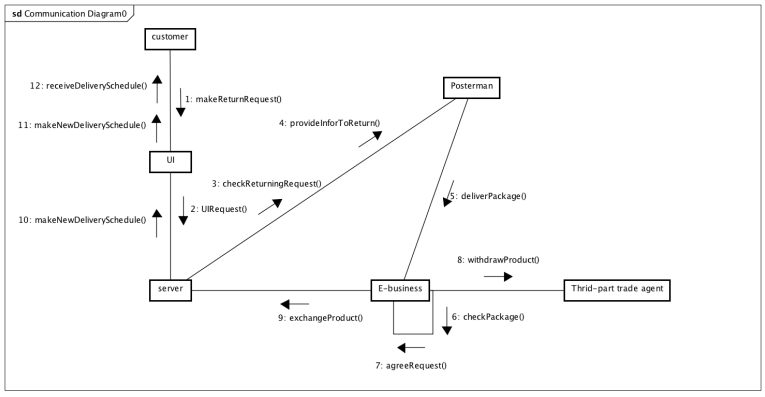



Figure 3.18: Scenario Seven Communication Diagram: Return

If the customer is unsatisfied with the product, he/she can return the product. Customers need to apply for returning through both App UI and Web UI. Then Parser will recognizes that request and assigns it to OrderController so Order Domain will return a label containing the address to send it back and the postage.
If the customer chooses to cancel the order, after a similar process, the order will be completed. Else if he/she chooses to withdraw the order, the original order will be recorded through **withdrawProduct()** and then be completed.

### Scenario Eight

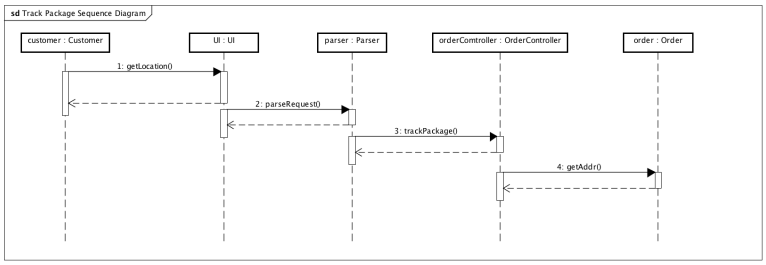



Figure 3.19: Scenario Eight Sequence Diagram: Track

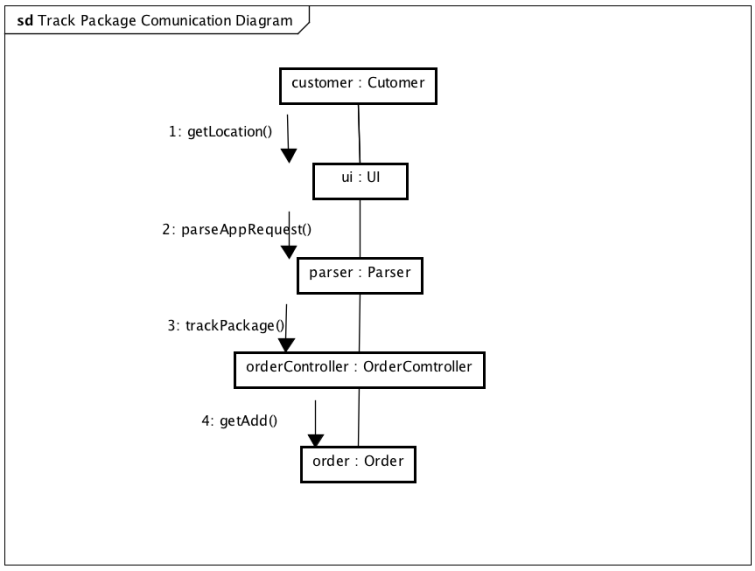



Figure 3.20: Scenario Eight Communication Diagram: Track

Customers and Customer Service have the right to track the product. Customer will invoke the function on UI. Messages will be sent to Parser in URL link. After parsing the link, Parser invokes the trackPackage function of OrderController. OrderController will then get the location by invoking the function of order entity. After getting the real time address, the system will return this information to the UI who invokes the function.

# 4 Design

## 4.1 Subsystems

### 4.1.1 List of Interfaces

#### Order Management Subsystem

This subsystem offers interfaces to controller for customers and E-business companies to create order, check order information and update order information. The corresponding interfaces are as follows.

```
Order create(string orderID, TransportStatus rStatus, bool
 pStatus, double freight, double price)
```

This interface is used to create order entity. It will receive all orders information. The return value indicates whether the creating process is successful, if return value is not NULL, the creating process is successful, and if return value is NULL, the creating process is unsuccessful.

```
Order selectOrder(string orderID)
```

This interface is used to select order from database. It will receive the order ID. The return value indicates whether the selecting process is successful, if return value is not NULL, the selecting process is successful, and if return value is NULL, the selecting process is unsuccessful.

```
bool insertOrder(Order order)
```

This interface is used to insert order into database. It will receive the order entity. The return value indicates whether the inserting process is successful, if return value is true, the inserting process is successful, and if return value is false, the inserting process is unsuccessful.

```
bool updateOrder(Order order)
```

This interface is used to update order of database. It will receive the order entity. The return value indicates whether the updating process is successful, if return value is true, the updating process is successful, and if return value is false, the updating process is unsuccessful.

```
bool deleteOrder(string orderID)
```

This interface is used to delete order in database. It will receive the order ID. The return value indicates whether the deleting process is successful, if return value is true, the deleting process is successful, and if return value is false, the deleting process is unsuccessful.

### User Management Subsystem

This subsystem offers interface to user to create or modify his informations and was inherited. by Class Customer and Postman.

```
User createUser(string userName, string password, string
 email, string telephone, Address address)
```

This interface is used to create user via userName, password, email, telephone and address. Once created, it will generate an userId automatically to identify and store the whole user info in database.

```
User modifyUser(template<T> userInfo)
```

This interface is a virtual function implemented by different sub-functions. User can modify his informations through sending the new value of the Class User.

### Package Sign Subsystem

This subsystem allows customer to interact with controller to finish package signing.

```
bool signPackage(Customer customerInfo, Order orderInfo)
```

This function will be invoked when a receiver wants to sign the package. Information of customer and her order will be sent to back end. A boolean value will be returned as a mark which indicates the status of signing process, that is, success or not.

```
bool confirmReceiverInfo(Customer receiverInfo, Order
 orderInfo)
```

This interface receives the information of receiver and corresponding order, sends them to customer in order to confirm if the receiver has access to the package. It returns a boolean value that means the verification is successful or not.

### Payment Subsystem

This subsystem offers interfaces to controller for customers and postmen to finish payment(including package and freight). The corresponding interfaces are as follows.

```
bool payOnline(string orderNumber, string password)
```

This interface is used in the scenario that customers pay online using third-party system(e.g. Alipay) and the third-party will check the payment and return the payment information(successful or failed). If the order and password are right, it will return true and amend the state of the order. Otherwise it will return false.

```
bool payByCash(string orderNumber, double amount)
```

This interface receives the orderNumber and the amount of the order as parameters and return payment information(success or fail). This is used in the receivers pay by cash and postmen check the payment.

```
bool payByCard(string orderNumber, double amount, string
 password)
```

This interface is used in the scenario that receivers pay by credit card. Postmen will check the order with the customers and then the amount and password will be entered into the POS device and return the payment information(success or fail).

```
Invoice provideInvoice(E-business ebusiness, string
 orderNumber, InvoiceType type, Address address)
```

This interface is used in the scenario that the customers need an invoice. If a customer asks for an invoice, he or she should send the orderNumber, invoice type(paper or digit), and corresponding address to the E-business company. Then the E-business company will return an invoice to the customer.

```
bool transferAmount(E-business ebusiness, double amount)
```

After the logistic company or third-party system received the amount from customers, they need to transfer the amount to the E-business company to finish this order payment.

### Tracking Subsystem

This subsystem offers interfaces to controller for customers and e-business companies to track their packages. The corresponding interfaces are as follows.

```
string getLocation(string orderNumber)
```

By sending request of customer to this function, the customer will get position information from App UI or Web UI.

```
bool parseAppRequest(string orderNumber)
```

By UI passing request to Parser to parse the request, UI will get position information from Parser.

```
Address trackPackage(string orderNumber)
```

By Parser passing request to OrderController, Parser will get position information from OrderController.

```
Address getAddress(string oderNumber)
```

By OrderController passing request to Order, OrderController will get position information from Order.

#### Return Products Subsystem

This subsystem is used to handle the withdrawal or exchange request from the customer. This subsystem offers interfaces to controller for postmen and customer service to deal with the returning. The corresponding interfaces are as follows.

```
bool sendWithdrawInfo(string orderNumber, string reason)
```

By sending the order number and the withdrawal reason to this function, the corresponding information will be changed in the system. The return value decides whether the message has been sent successfully.

```
bool sendExchangeInfo(string orderNumber)
```

By sending the order number and the withdrawal reason to this function, the corresponding information will be changed in the system. The return value decides whether the message has been sent successfully.

```
string newDelivery(string info)
```

By sending the order number and the withdrawal reason to this function, the corresponding information will be changed in the system. The return value decides whether the message has been sent successfully.

### 4.1.2 Detailed Subsystem: Order Management

Our Order Management Subsystem provides some interfaces such as create, selectOrder, insertOrder and so on. The OrderController class require following interfaces. For example, if a customer want to create an order, it will call the createOrder function of Order controller, and then use the create function of Order class interface to create an order and the insertOrder function of Database Controller to insert a new order into the database.
As the class diagram shows, our system will get the external order information in the format of Pair from customers and E-business type if it creates orders for E-business. By invoking the interface in orderController class, it will call the create function in order and insertOrder function to insert the data in database. This sequence diagram shows the process of creating the order for the customer. After getting the request from the customer, the system will

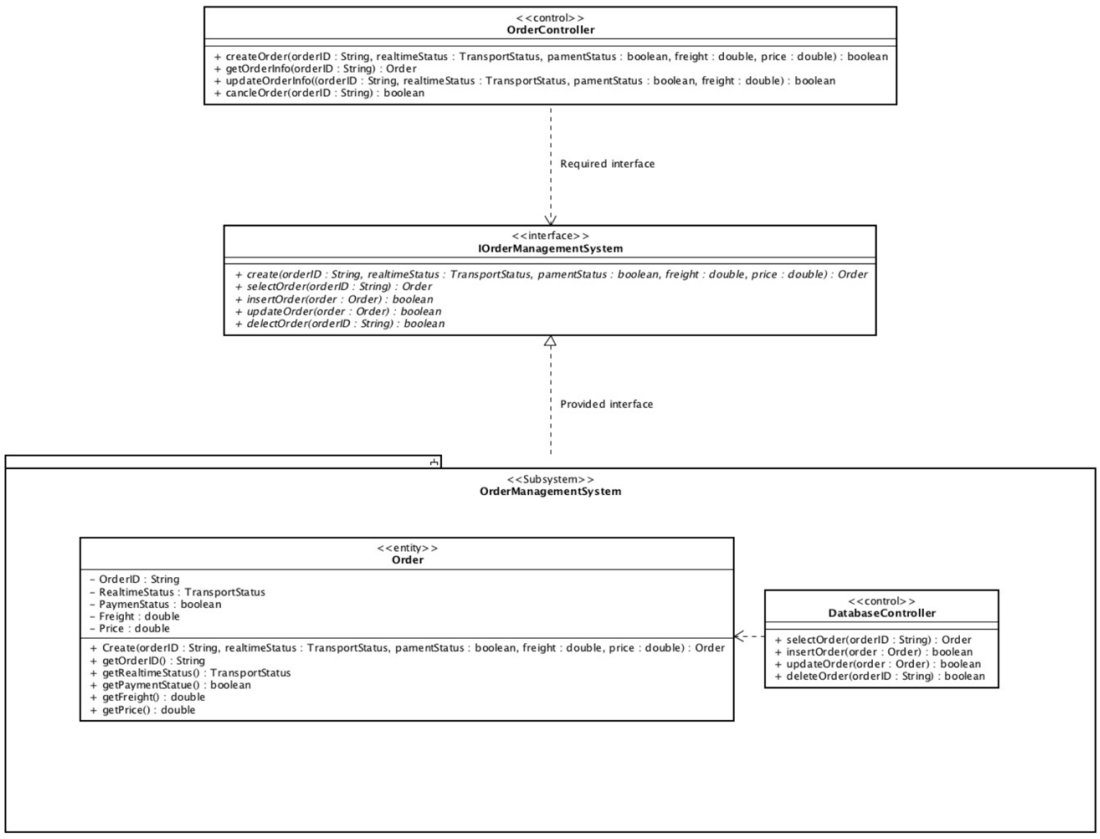



Figure 4.1: Class Diagram: Order Management Subsystem

check if the customer already exists in our system. If the customer does exist, the system will add the external order information provided by the customer into our system. And return the result to the customer.
This sequence diagram shows the process of creating order list for the E-business. In this subsystem, it will create the order list by creating one single order each time. Until the loop of creating a single order finishes, it will return the whole order list back to the E-business. For creating a single order, the process is the same as creating the order for the customer, as showed above. By checking the existence of the customer, the system will add the new orders information of the customer.

### Protocol

Mainly we would use TCP/IP (Internet Protocol Suite) and the common four layers: link layer, transport layer, internet layer and application layer.

- Link Layer: We adopt Ethernet and particularly take MAC (Medial Access Control) model in our system.
- Transport Layer: We employ IP to ensure the transportation of data and ARP & RARP as the transformation between IP address and the physical address. In addition, considering our system is used around the regions of Jiangsu, Zhejiang and Shanghai, we use IGMP to process the trans-regional transportation.
- Internet Layer: Because in OrderController, the content is orders, we decided use UDP and through the error handling to ICMP. If the transportation failed, ICMP in Transport Layer will send error message.
- Application Layer: In this layer because there is no need to assure real-time communication. Besides, it is unpractical and will cost a lot, we use HTTP/HTTPS.

### JSON Format of Order

```
{
 "orderNumber" : string,
 "orderDate" : string,
 "orderSenderContacts" : {
  "userID" : string,
  "userType" : enum,
  "userName" : string,
  "email" : string,
  "telephone" : string,
  "address" : {
```

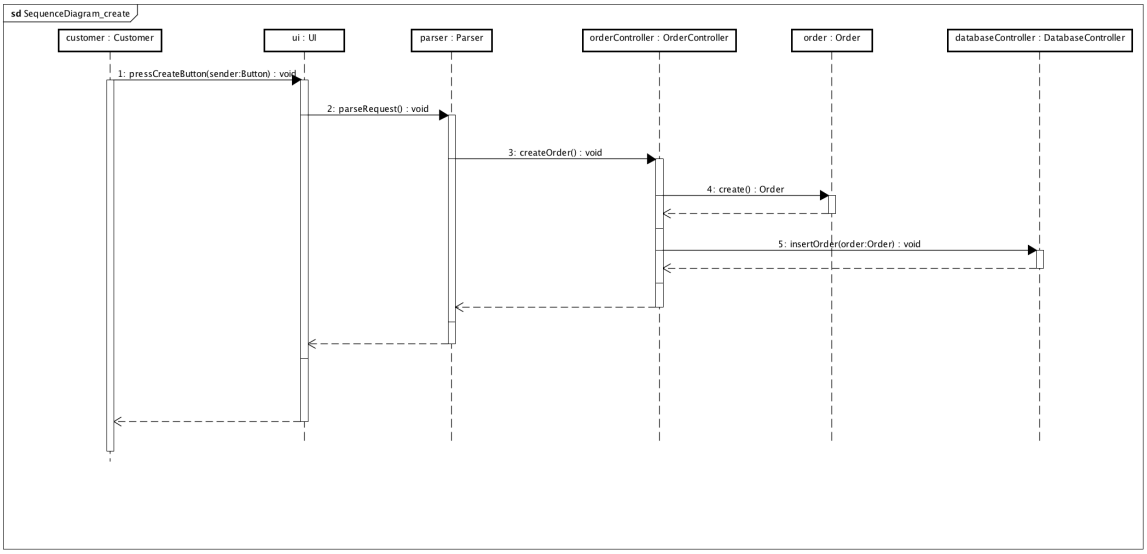



Figure 4.2: Sequence Diagram: Create

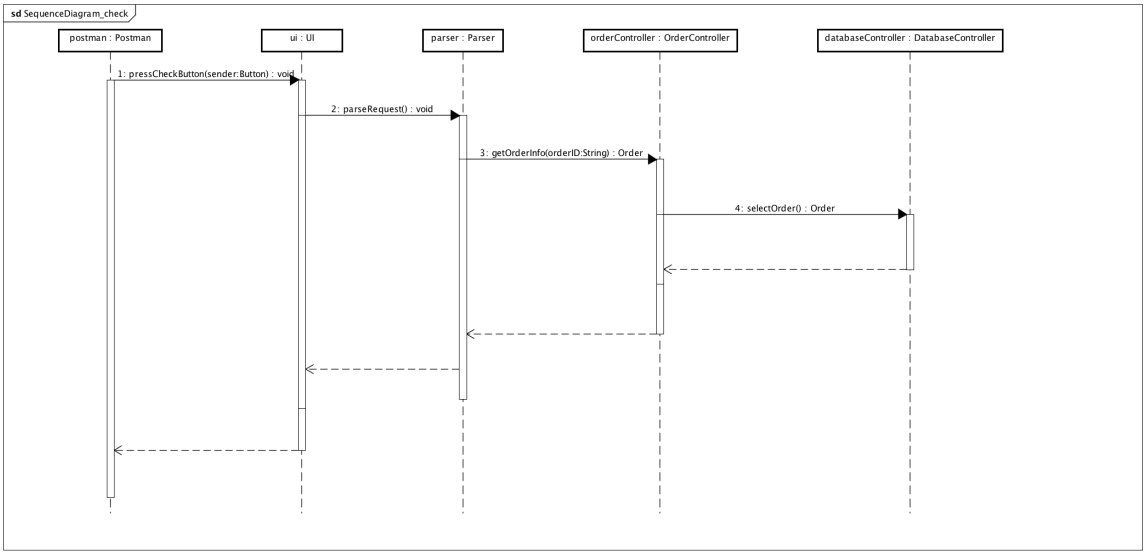



Figure 4.3: Sequence Diagram: Check

```
    "country" : string,
    "state" : string,
    "city" : string,
    "district" : string,
    "road" : string,
    "building" : string,
    "room" : string
    }
  },
 "orderReceiverContacts" : {
  "userID" : string,
  "userType" : enum,
  "userName" : string,
  "email" : string,
  "telephone" : string,
  "address" : {
    "country" : string,
    "state" : string,
    "city" : string,
    "district" : string,
    "road" : string,
    "building" : string,
    "room" : string
    }
  },
 "orderPackage" : Package
}
```

### 4.1.3 Interface between Our System and External Systems

As shows above, we have designed create order interface for communication between our system and external systems. The external systems include the E-business websites and the customer.
The Encryption class provides interfaces for the third party messaging system. These interfaces are showed in the security mechanism.

## 4.2 Mechanisms

### 4.2.1 Communication Mechanism

Our system provides both AppAPI and WebAPI to the User Interface. The parser will parse the request and forwarding to the OrderController or Third-Party System according

**Order Format**

| | | | |
|---|---|---|---|
| Order ID | | Freight | |
| Sender Name | | Sender Tel. | |
| Sender Address | | | |
| Receiver Name | | Receiver Tel. | |
| Receiver Address | | | |
| Real-time Address | | | |
| | | | |
| Product Information | | | |
| | | | |
| Payment Status | | Sign Status | |

Figure 4.4: Order Format

to the content. The OrderController will use the interface provided by the Order Class. The communication will be provided by using its interface. The sequence diagram is used to describe the message receiving process of the subsystem's interface. This interface will receive the messages and then create responses, send back to front-end.

### 4.2.2 Security Mechanism

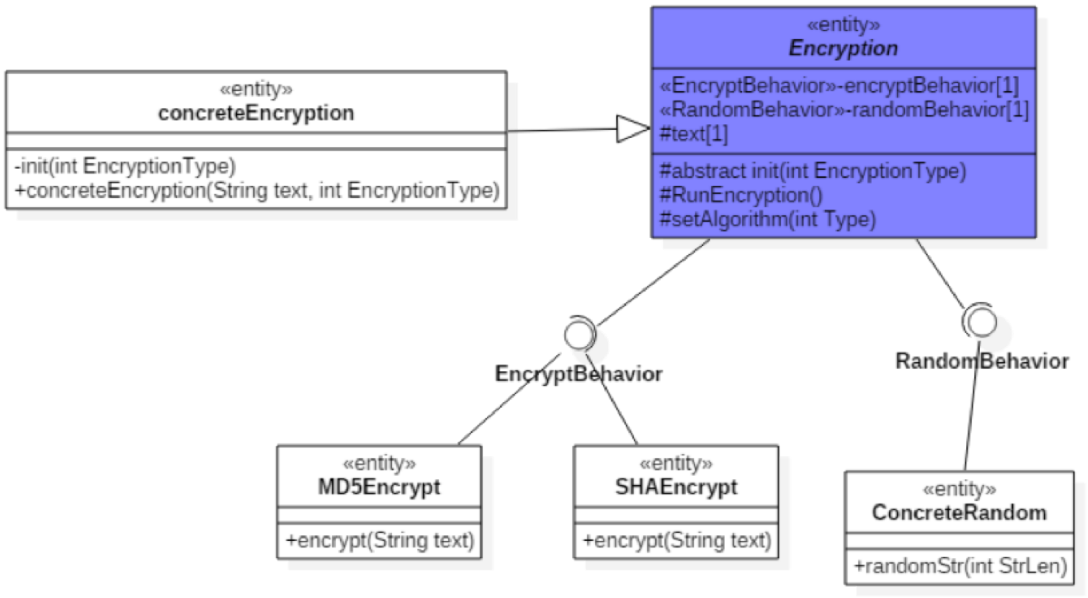



Figure 4.5: Security Mechanism

Our system provides two interfaces in this mechanism and defines two corresponding functions, including encryption and random string generation. There are two entity classes MD5Encrypt and SHAEncrypt which are designed for implementing the interface Encrypt-Behavior. Inheriting from the abstract class Encryption, the entity class ConcreteRandom is used to implement the RandomBehavior interface and will invoke these two interfaces.
Encryption is an abstract class, without the **encryption** operation and **randomStr** operation. Thus, two attributes which are two interfaces are private. What is more is that we add a string attribute to store the content, an abstract operation **init** for subclass to init the text attribute and set algorithm at the beginning, a **runEncryption** operation to precess the encryption and a **setAlgorithm** operation to set the algorithm.
The sequence diagram of this security mechanism shows the encryption process. The system will generate a string randomly according to the length defined in the Encryption class. The **encrypt** interface will put together the original password and the random string to complete the encryption, thus, return the encrypted string. This encrypted string is used to identify the user's identity by comparing the input of password with what stores in our database.

## 4.3 Use Case Realizations

### 4.3.1 Sign the Package

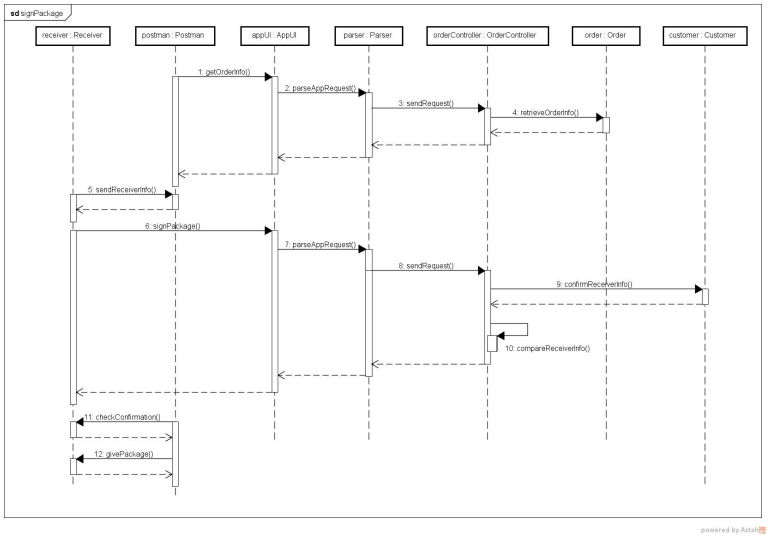



Figure 4.6: Sequence Diagram: Sign the Package

In this use case, the postman uses his mobile device to let the receiver sign the package. Since the receiver is the generalization of the customer and the third-party agent, we add additional controller: Signature system, to control the signing way: fingerprint. After receiving and parsing the request in the Dispatch Controller from Postman's App UI, it will use the Order Controller to compare the receiver information and use the Signature System to compare the signature or fingerprint with what store in the system. We have defined a series of **get** functions and corresponding parameters to implement this use case. All the attributes except the orderID are defined as protected, so that the security is guaranteed. We define the signature, fingerprint and order list attribute of the class Customer as private ones to improve security. And the aggregation as well as the relationships such as aggregation, dependency are showed in the class diagram.
The postman will get the receivers information by using the Postman's App UI. This boundary class will call its interface **getReceiverInfo()**. The corresponding request will pass through Dispatch Controller, Order Controller and Order class. Each class will call its own interface showed in the class diagram and retrieve the order and receiver information finally. During the signing process, the postman's App UI will call its interface **signPackage()**. If the package is signed by the agent, the system will identify the information and compare them by using the order controller and calling compareIdentifyInfo(). If the package is to be signed by the customer himself, the customer has alternative choices, including signature and fingerprint. We will submit the signature or fingerprint by calling the interface **submitSignature()**, provided by the signature system. And pass the corresponding information to the

signature service which will compare the signature or fingerprint with what are stored in our system. If the corresponding information are matched with each other, the order controller will call its interface **confirmDelivery()**. Then the order class will use **updateStatus()** to complete the package delivery and return success signal to the **getStatus()** interface.

### 4.3.2 Track the Package

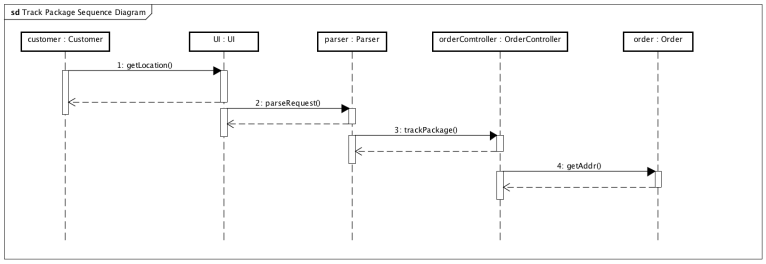



Figure 4.7: Sequence Diagram: Track the Package

The process of signing package is shown as the sequence diagram above. First of all, the postman needs to know information of the customer. As a result, a request of retrieving the order information will be sent by postman. The parser here is to analysis requests from UI and transmit to the controller. Information, including customer's name and phone number, will be returned to the postman. After that, the receiver is supposed to inform the postman her information. In order to verify if she is the designated person or customer herself, a confirmation of customer is required. If so, the receiver will have access to sign the package; otherwise, the demand of signing the package should be rejected.
When the Customer and the Customer Service track the package by using the Customer's App UI or Web UI, the UI will call the API implemented by dispatch controller and send the Request by the HTTP. The request contains the order id and the content-type is JSON. The parser parses the request and sends the order id to Order Controller. Order Controller gets the order information from the database. And if the order exists, Order Controller will get the location info of the order and sends it to the Address Controller. The Address Controller will get the addresses from the location information by a loop and put them in an order list. This process will come to an end when the prelocationid is null. After this process, the Address Controller return the Address list to the customer.
After invoking the interfaces parseWebRequest(URL url, JSON info) and parseAppRequest(URL url, JSON info) of Dispatch Controller, the system will get the tracking request from the customer or the customer service. It will use the getOrderFromDB(String ordered) provided by the Order Controller to get the package, required to track by the customer or the customer service. If the package exists in the database, it will invoke the getAddress() and getPreLocation() interfaces to get the current and former addresses of the package by its order id. Then the system will return the Address list to the customer or the customer service.

# 5 User Interface

## 5.1 Mobile Devices(iOS)

### 5.1.1 Log in Page

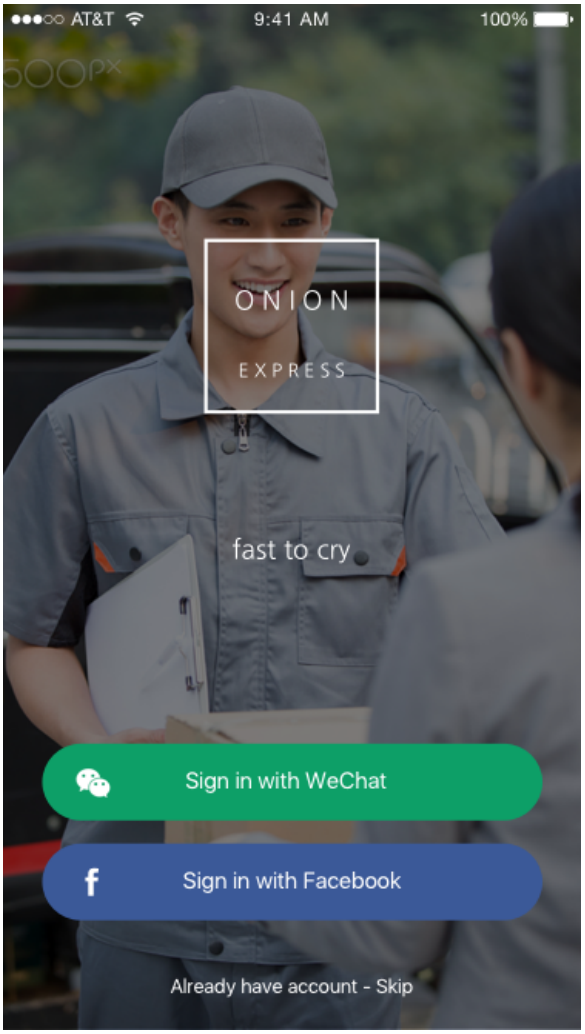



Figure 5.1: Log in

This is the login page of our app, for convenience, users can log in via WeChat or

Facebook, which is the popular social account around the world. Through the account, we will record user's information in the server, and can sync data to provide a better user experience.

### 5.1.2 Main Page

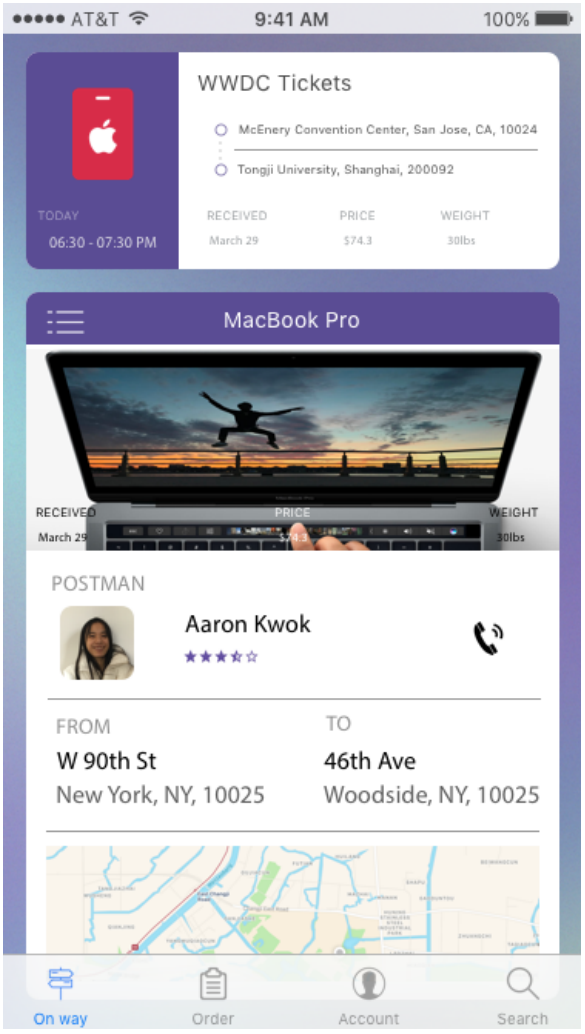



Figure 5.2: On way

This is the main page of our app. It shows the user at all glance all the current package, and click on the user after the show the details of the package. Details of the courier to provide a contact, historical evaluation, and the location of real-time display on the map, easy to track users. In addition, the basic information about the package is provided: including estimated time of arrival, delivery costs and weight.

### 5.1.3 Search Page

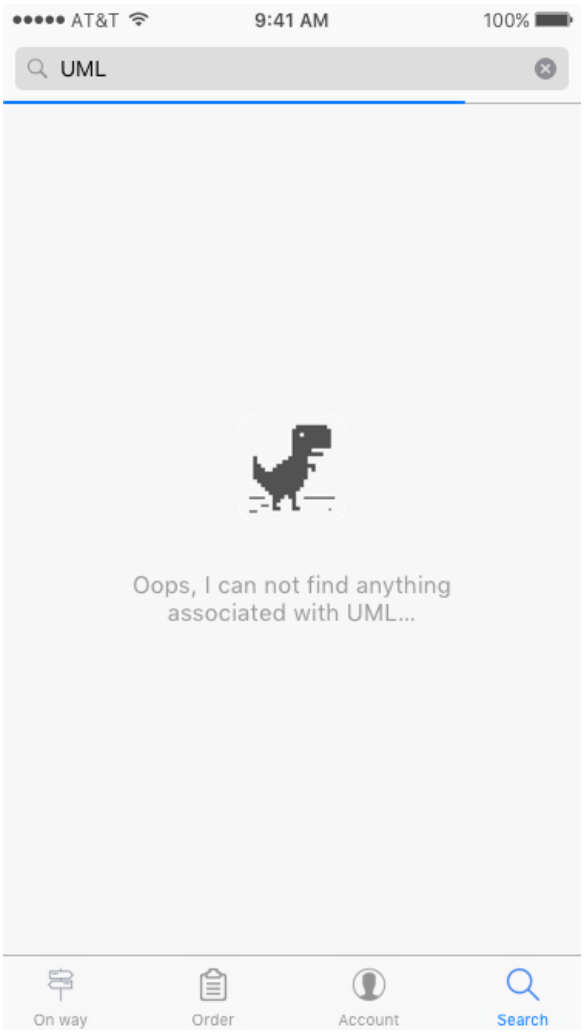



Figure 5.3: Search

This is the search page of our app. User can type everything he/she want to. And we will search both locally and in the server.

### 5.1.4 Order Page


Figure 5.4: Order

This is the order page of our app. It shows all the packages both on the way and delivered. Like the main page, it can show details and ordered with timeline.

### 5.1.5 Account Page

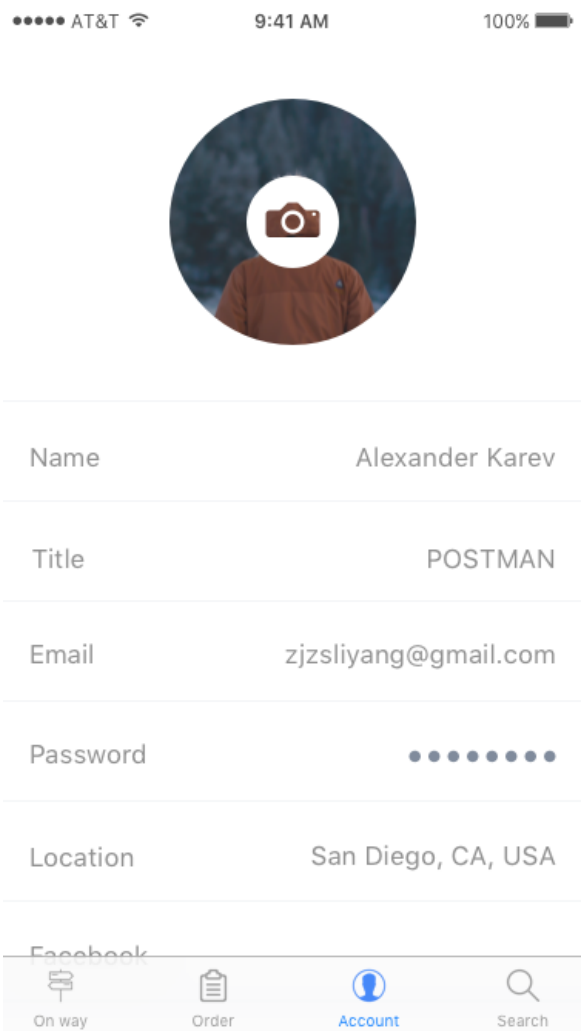



Figure 5.5: Account

This is the account page of our app. It shoes the basic information of users, including name, title, email, password, location and social account. Users can change settings there and it's simple and concise.

## 5.2 Website

There is basic information of the system showed on the page. The services can be ordered on this website after the user logs in. The custom can search for their packages' particulars after logging in. In addition, the significant notations of the Onion Express® are showed on

the web page, such us the forbidden objects etc. Any common browsers of the website can contact the Onion Express® company and know about the company freely.

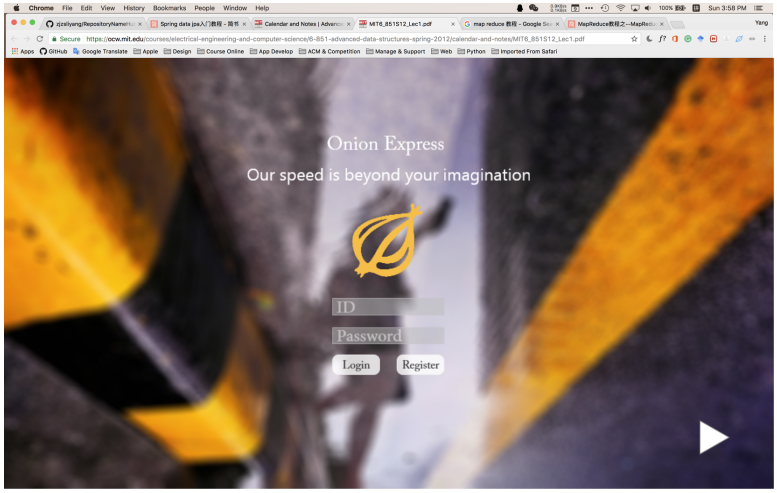



Figure 5.6: Index1

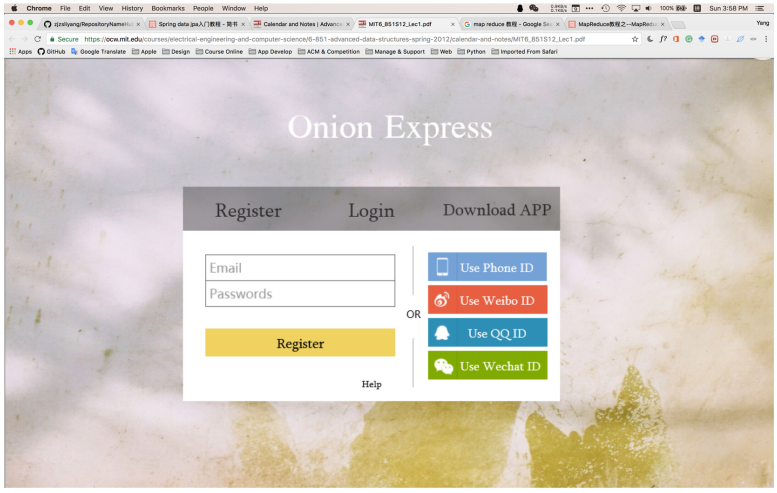



Figure 5.7: Index2

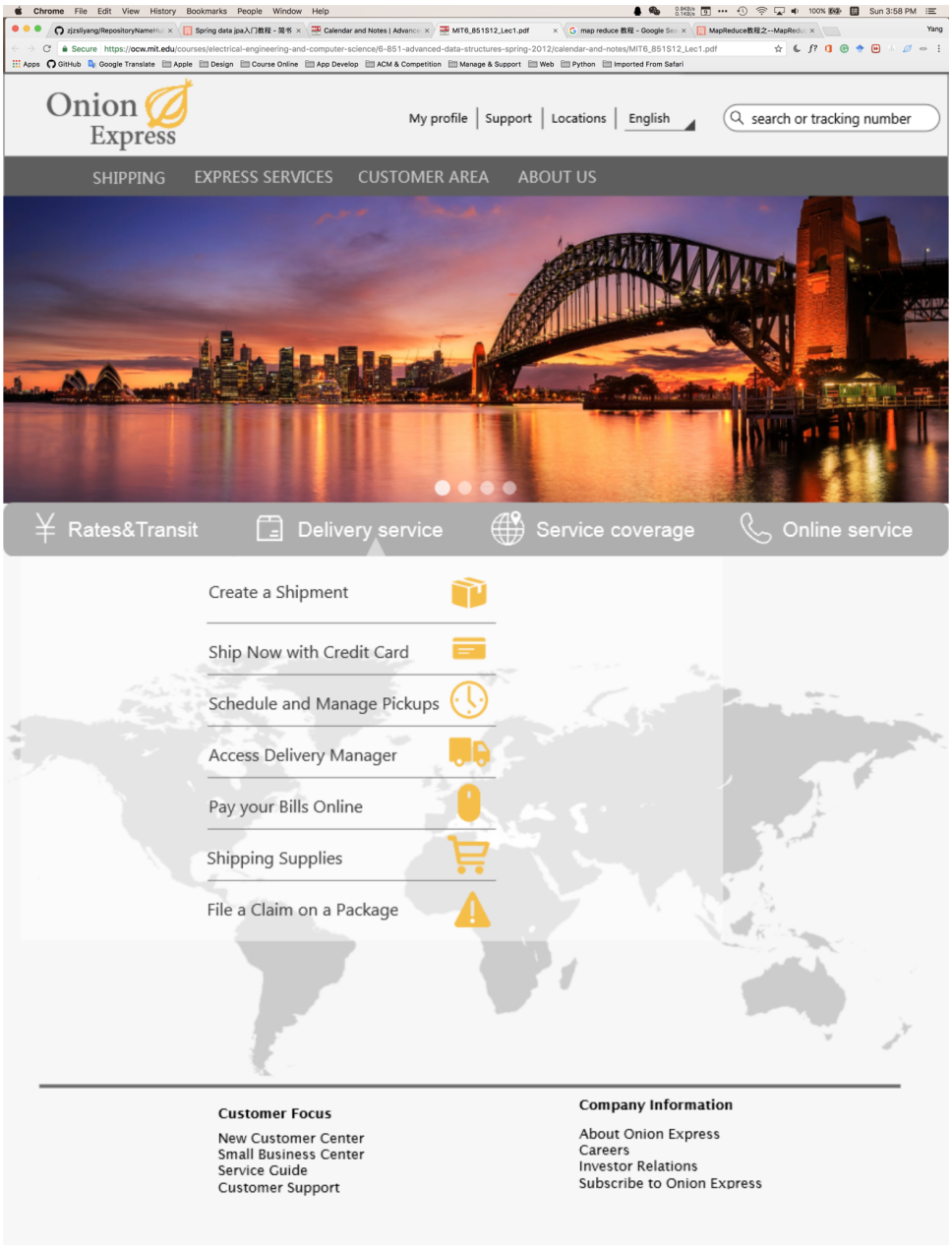
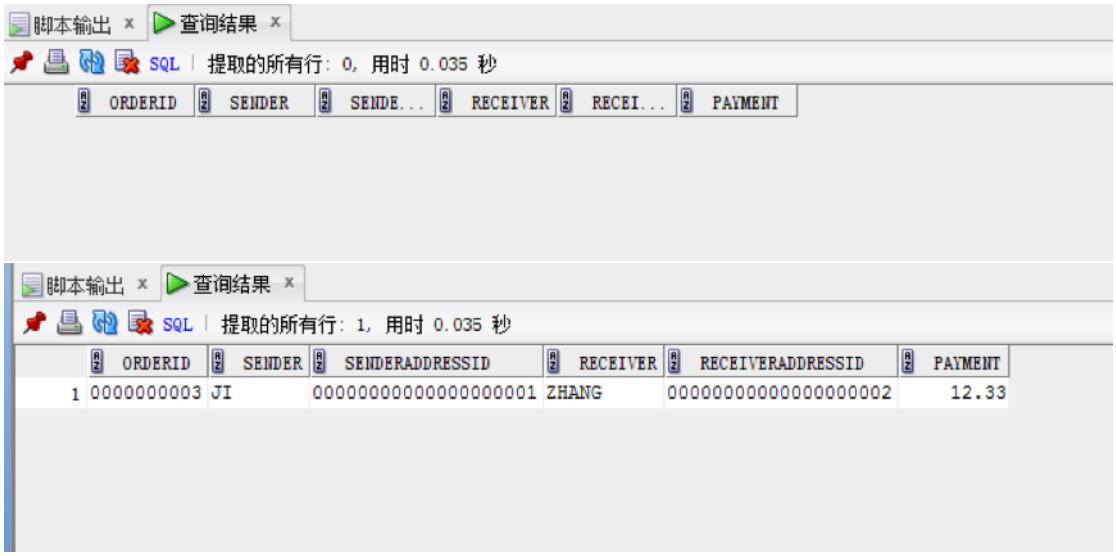



Figure 5.8: Index3

# 6 Prototype

We build the connection with Oracle and the attributes of the connection, including user name, password and the dialect.




We create the mapping between the Order table in Oracle and the class Order of Java code. And specify the configuration file and starts one session with the database.
After the instantiation of the Order class, the data has been stored in the database after the operations.
Thus, encryption part is shown as follow:

```
DBPassword: 堉"  墊□?□K?7=,□
True
```

In order to realize extendibility and reusability, the encryption mechanism of our system is designed be using the strategy pattern. We have defined 2 interfaces, which are encryption behaviors and generation of random string. An abstract class Encryption is defined and its method is designed for implementing the 2 interfaces. We may define a class inherited from the Encryption class and define the two interfaces in detail. In our prototype, we design two concrete class, named as MD5Encrypt and SHAEncrypt. And 2 other concrete classes are

defined for the generation of random string and encryption of the password. The detailed test is in the test file.
The source code is showed in the compressed package.

# 7 Glossary of Terms

**after-sales service**

Also called customer service, after sales service is the provision of service to customers before, during and after a purchase.

**article**

The material in the package which is sent by a normal customer.

**bi-directional read**

The information can be read in both direction.

**cash-on-delivery express**

The sale of goods by express where payment is made on delivery rather than in advance.

**claim**

When packages are damaged or lost, customers have right to ask for compensation.

**courier**

A courier is a person who delivers messages, packages, and mail. Here it refers to postmen.

**customer service staff**

The staff in the logistics company serving customers.

**damaged express item**

The package that is damaged during express.

**decision support system(DSS)**

A decision support system is a computer-based information system that supports business or organizational decision-making activities.

**delivery**

A single task to send the package to a customer.

### delivery terminal

The destination of the delivery where the receiver receive and sign the package.

### dispatch list

The digital list of information of the packages to be delivered in postmens port.

### distribution center

A station in a large district to transfer packages to the regional distribution center.

### door-to-cfs

From the shipper factory or warehouse to the destination or the Container freight station of the discharging port.

### door-to-door

From the shipper factory or warehouse to the consignee's factory or warehouse.

### Electronic Data Interchange(EDI)

Electronic Data Interchange is an electronic communication method that provides standards for exchanging data via any electronic means.

### Electronic Order System(EOS)

Electronic Order System is to meet demand instantly, with perfect quality and punctuality.

### express item

Packages to be delivered.

### express item tracking system

A subsystem in our system to track the packages with GIS automatically.

### express network

A service network within the scope to help delivery.

### express waybill

An express receipt given by the carrier to the shipper acknowledging receipt of the packages being shipped and specifying the terms of delivery.

### first time delivery

The first time for particular postman to send the package to a position.

### Global Position System(GPS)

The Global Positioning System is a space-based navigation system that provides location and time information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites.

### Geographic Information System(GIS)

A geographic information system is a system designed to capture, store, manipulate, analyze, manage, and present all types of spatial or geographical data.

### handheld terminal

Handheld terminal refers to the portable data processing terminal with some particular features. Here it refers to mobile phones with our app.

### inquiry

The customer logins the system or connects with customer service staff to get information about the order, operation instruction etc.

### Integrated Services Digital Network(ISDN)

Integrated Services for Digital Network is a set of communication standards for simultaneous digital transmission of voice, video, data, and other network services over the traditional circuits of the public switched telephone network.

### interchange receipt

A voucher to certify that the customers or e-business commits articles or products to the logistics company for delivery.

### Invoice(INV)

An invoice is a commercial document issued by a seller to a buyer, relating to a sale transaction and indicating the products, quantities, and agreed prices for products or services the seller had provided the buyer.

### Just-in-time logistics(JIT logistics)

Just-in-time logistics is a modern logistics method based on the JIT management philosophy.

### lost express item

The package that is lost during express.

### order number

The number generalized when the order is created.

**order processing**

A series automatic operation in system to deal the order, such as creating an order, completing an order and so on.

**package**

The material to be delivered after customers or the e-business company create orders.

**product**

The material in the package which is ordered by customers from the e-business company.

**receiver**

Generalized from Customer and Agent, the person receiving and sign in the package directly.

**redelivery**

When no one can sign the package, the postman will carry it back to the delivery terminal and the order will be rescheduled in the system.

**redirect express item**

When customer changes the destination or the destination is out of scope, the package will be reassigned.

**regional distribution center**

The substation in a certain region of the logistics company to assign packages to postmen.

**return**

If customers are unsatisfied with the product, he or she can send it back with a label from system.

**sender**

The customer or the e-business company who sends the package.

**serial number of express**

i.e. the tracking number of packages in the system.

**sign in**

The receiver sign the package and get it.

**sorting**

The packages in the regional distribution center are sorted to transfer to corresponding postmen or the packages in the distribution center are sorted to transport to regional distribution centers.

### tracking number

Especially for tracking the real-time GPS location of the package.

### withdrawal

If the customer is unsatisfied with the product and has sent it back, he or she can choose withdraw the order and the payment will be reimbursed.

# 8 Contributions

More information, please visit us on GitHub

| | | |
|---|---|---|
| 1452559 Yang LI | iOS UI, Document | 17% |
| 1453645 Zhongjin LUO | Scenario One & Two Diagrams | 17% |
| 1451229 Guohui YANG | Scenario Three & Four Diagrams | 17% |
| 1552705 Yiqun LIN | Scenario Five & Six Diagrams | 17% |
| 1552651 Yirui WANG | Scenario Eight & Global Diagrams | 17% |
| 1552677 Xinying WU | Web UI, Scenario Seven Diagrams | 17% |

# Bibliography


[1] 830-1998, *IEEE Recommended Practice for Software Requirements Specifications*, IEEE, Oct 1998.

[2] 29148-2011, *Systems and software engineering – Life cycle processes –Requirements engineering*, ISO/IEC/IEEE International Standard, Dec 2011.

[3] Russ Miles, Kim Hamilton, *Learning UML 2.0*, O'REILLY, 1st edtion, April 2006.

[4] Jim Arlow, *UML 2.0 and the Unified Process: Practical Object-oriented Analysis and Design*, ADDISON WESLEY, 2nd edition, 2005.

[5] Karl Eugene Wiegers, Joy Beatty, *Software Requirements*, Microsoft Press, 3rd edition, 2013.

[6] Craig Larman, *Applying UML and Patterns*, Pearson Education International, 3rd edition, 2005.

[7] Simon J. Bennett, Steve McRobb, Ray Farmer, *Object-oriented Systems Analysis and Design Using UML*, McGraw-Hill Education, 2nd edition, Dec 2001.

[8] Yunjie TAN, *Thinking in UML*, China Water Conservancy Hydropower, 2nd edition, March 2012.