優木書屋

Class : System Analysis and Design

Group No : Group 3

Team Member : 张喆， 陈开昕， 马思腾， 王舸飞， 杨乐

Date : 2019.6.22

1.Overview

1.1 Requirement

A new business will be initiated: a virtual bookstore. The system to support it must manage the acquisition and selling processes of the company. Access for customers and management people must be accomplished through a Web site. The user can access it via PC or mobile device.

When a book is ordered, it is delivered immediately if available in stock, or else, the book is ordered to a publisher, and a compatible deadline is informed to the customer. The system shall handle the return orders. The system must calculate taxes and delivery fee as well as applying discounts to the sale when applicable. The system should be able to monitor the inventory and issue the warning when certain products need to be replenished.

The system must allow a manager to generate reports on bestselling books, and on most profitable customers, as well as suggest books for buying based on past customer's interests. Furthermore the system must have the capability of predicting the sales in order to provide better decision (inventory, reordering products, etc.) with the solid foundation.

Constraints:

a) Customers would pay by credit card, Alipay, or WeChat.

b) The bookstore will deal only with new books, not used ones.

c) Access to the system will be available through a web site via PC, mobile devices etc.

1.2 Background

As the booming of network in recent years, e-commerce has been widely accepted by a variety of the Internet users. Being a part of which, online book store system is developing rapidly as well.

In a form of website or application, online book store can be taken advantage to sell books. Online book store opens one after another with the broad adaptation of e-commerce in the aspect of book selling. For one thing, a majority of people have formed the habit that shop online. The number of which has been increasing continuously. For another thing, from the aspect of shopping items, books are relatively cheap and standard, so it is of few risks to buy a book and online purchasing can be accepted easily. Comparing to the traditional book store, the online book store have a lot of advantages. As for sellers, they can not only avoid the limitation and blindness while ordering books, but also surmount the obstacles of high expanses, tough management and costly ordering. As for customers, they are capable of learning details, looking through samples and ordering books swiftly. It is difficult for offline transaction to have the above advantages.

All in all, developing an online book store system is available as well as fashionable.

1.3 Purpose & Definition

Based on the fundamental functions of online book store and our brainstorming, the participants of the online book store system involve users (visitors, registered user, senior user), administrators, publishers and third-party payment enterprises.

According to the advanced design, the online book store system can maintain the following functions:

1. As for users: have access to check and alter part of the personal information, be able to search, add to shopping cart, add to order form and ask for after-sale service, can share own reading schedule and comments.
2. As for administrators: update the details of each book, control the basic settings of the system, send messages to users in time and get touch with publishers in order to inform

them to supply the stock.
3. As for publishers: receive the timely messages which come from administrators and feedback the stock information.
4. As for third-party payment enterprises: give authority to the payments in the order forms.
5. Keep limits of authority distinct so that it is conducive to the protection of data and intimacy securities.
6. Find latent users in order to expand our customers.

To coordinate with the online book store's style, we are expecting to provide a cozy environment for users to stroll in the book store at the same time of feeling the pleasure of reading besides make it possible that users are able to easily buy a book online. By using our online book store system, users can attain a positive shopping and reading experience.

Moreover, as an online book store, we are determined to endow a sense of belonging to every user by forming a unique culture. So we have designed an UI of chinoiserie and have entitled our store YOUMU Book Store.

1.4 System Overview

1.4.1 System Structure

The high level architecture is divided as four different layers which have their own certain functions, named as Presentation Layer, Services Layer, Controller Layer and Data Layer.

1.4.2 Design

The customers will send request from the browser on his/her device, then the browser accesses the targeted web pages through Uniform Resource Locator (URL), which colloquially termed a web address. After retrieving the request, the View invokes the model to retrieve information from the backend database. The tuples in the database will be returned as an object, then the view will process them, select the appropriate templates, padding with the appropriate parameters, and finally send the HTML web page to the browser. In fact, our architecture is framework based, Views and models are implemented at different levels.

1.4.3 Description of Implementations

We choose to use the popular CSS and JavaScript frameworks, such as Vue2.0 and jQuery to design our HTML web pages. jQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. It is a free, open-source software using the permissive MIT license. Web analysis indicates that it is the most widely deployed JavaScript library by a large margin. Vue2.0 is a **progressive framework** for building user interfaces. Unlike other monolithic frameworks, Vue is designed from the ground up to be incrementally adoptable. The core library is focused on the view layer only, and is easy to pick up and integrate with other libraries or existing projects. Based on Google's V8 engine, Node.js is an event-driven I/O server-side JavaScript environment. Simply, Node.js runs in the backend written by JavaScript. We take advantage of Express as framework to develop server and what's more, to apply to the Web application, attaching great functions to our online book store system. As for the Persistence, we use the MySQL as DBMS to maintain the data.

1.5 Progress for Each Time

A1 => A2

For the use case diagram, we integrated the original user and administrator use case diagrams together, and deleted some redundant relationships, increasing the types and generalization of users. For activity diagrams, in order to clarify the behavior of different participants in the activity, we added swim lanes for each activity map and optimized the layout of the activities. For the system snapshot, in order to match the Chinese style theme of "Yu Mu Shu Wu", we redesigned a corresponding UI and added a mobile prototype. In order to clarify the system architecture, reduce system redundancy and facilitate teamwork, we layered the whole system and gave the system package diagram, and analyzed and clarified the architecture of the whole system. For the design of the domain model, we designed the concept class based on the principle of focusing on the application side rather than the system design, drawing the overall concept class diagram, adding the relationship between the classes, and finally sorting into tables.

A2 => A3

Since the last project, a great progress has been made. The English version is totally developed, which means English is adapted to each of our diagrams as well as word descriptions.

As for the architecture, according to the result of our brainstorming, we have refined the architecture with considering the ways of implementation based on the original 4 layers. Each layer has specific dependent platform. What's more, a label of subsystems and interfaces is listed to demonstrate the operations and interfaces clearly. And several samples are presented to show the details of our process.

As for the design mechanisms, we have searched a lot of relative reports to learn about it and make use of some of them to our online book store system. Persistence mechanism and Information Exchange mechanism are exemplified specifically with their class diagram, sequence diagram and description, including the suitable implementation platform.

As for the use case realization, the above design mechanism is combined with the architecture, presented through the use case.

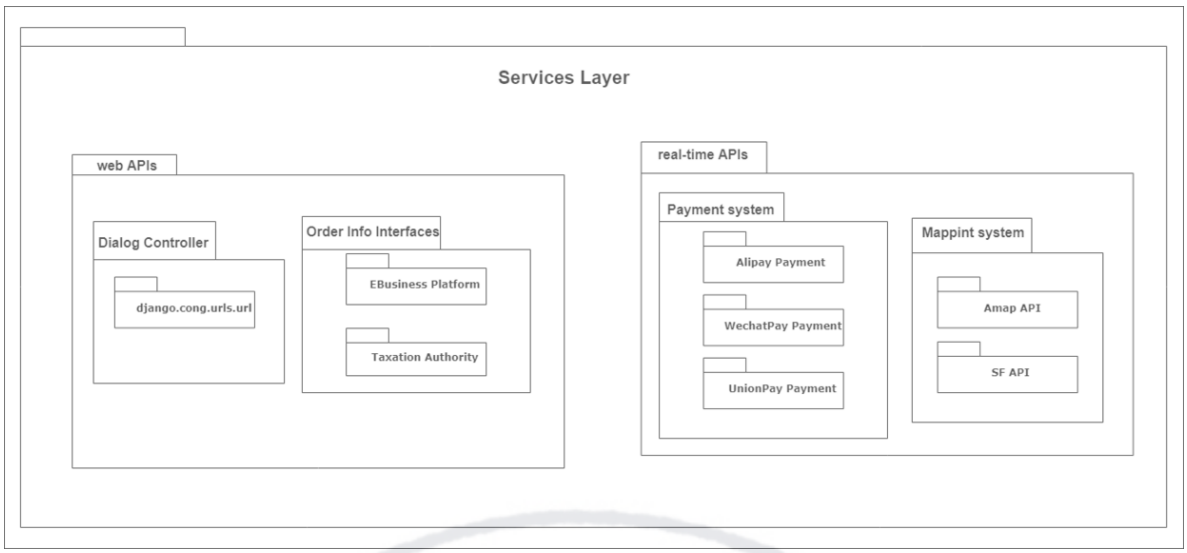
As for the prototyping, we have already created a demo connecting the database and our login subsystem interface. Users are able to register and log in though the browse. The coordinate data of him or her will be stored perpetually.

Apart from the use case model we made in advance, we analyzed the whole system and separated it to 9 subsystems, suppling the class diagram, the sequence diagram, etc. The class diagrams are updated from the previous analysis class diagrams, being suppled with functions and boundary or control classes.

As for the design mechanism, the architecture and interfaces are presented, especially the detailed interfaces showing the major pages of our online book store system. The interpretation is shown in these pages.

A3 => A4

For the final solution and self reflection, we first make some improvement of our primary architecture. The basic principle we use in architectural design is layered architecture which is the most common architecture and has become almost a standard norm. We introduce asychronus mechanism on the web front end by using asychronous design version APIs, such as AJAX and jQuery. What's more, we update the service layer by splitting it into two parts: the web APIs(containing dialog controller and order info interface) and the real-time APIs(containing mapping system and payment system). The updated layer is shown as below:



We also add a Web Application Firewall at the outside boundary of the infrastructure layer. The WAY middleware controls all data access to our API. Then we add a model layer between controller layer and data layer in order to search data more efficiently by accessing the database more clearly and easily.

Furthermore, we make critical design for data storage and classes. First, we decided to use MySQL, MongoDB and Redis together to store our data considering the three kinds of data in our system which are data that is highly relational, data that is less relational and data that is performance sensitive. Second, we create a class called senior user derived from normal user in order to help reduce the data that needs to be stored and transfered for coupons.

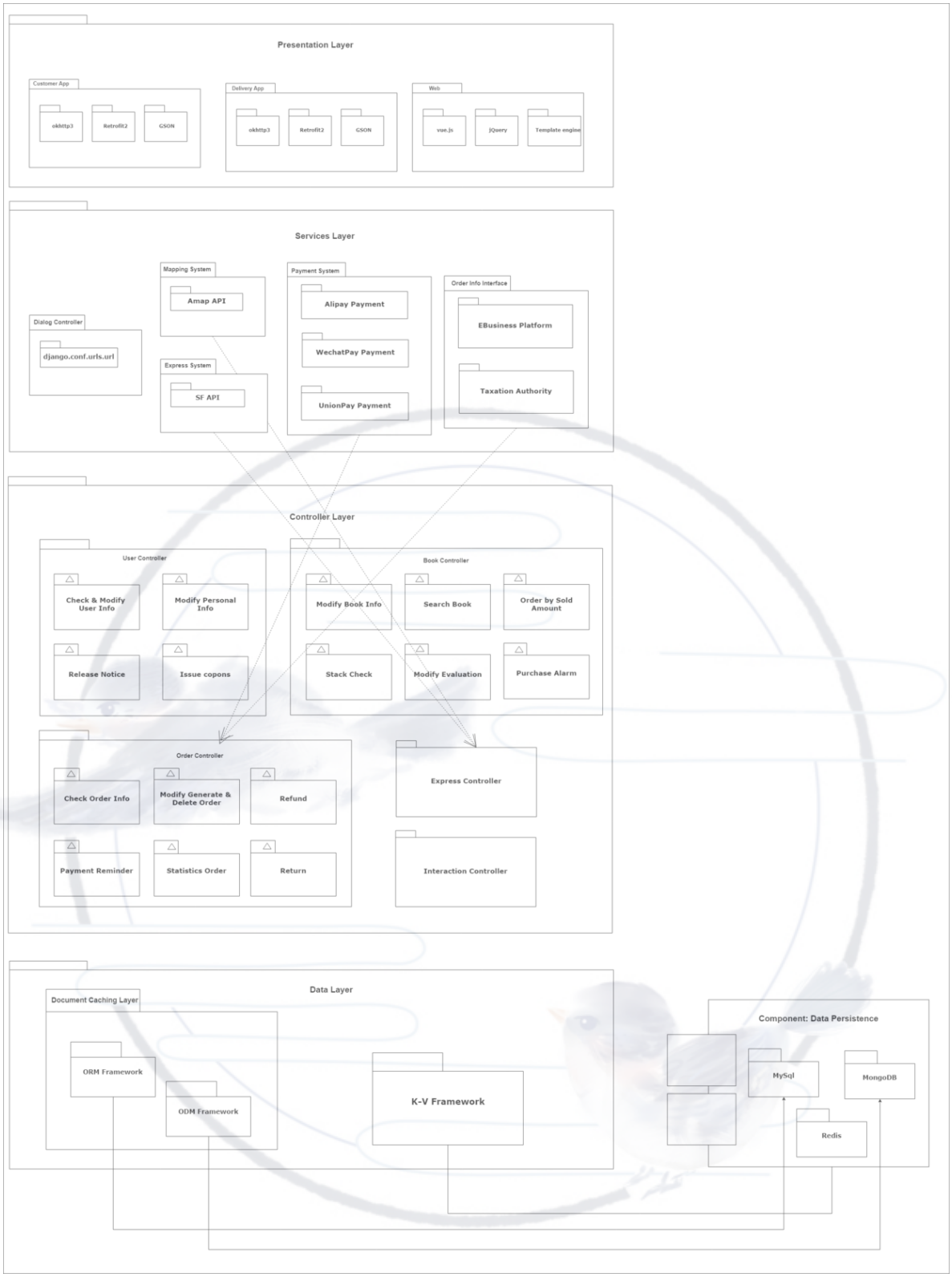
Also, we find some open issues for our system. The first is improving robustness which is very important for our system. We need to improve the robustness in architecture design and program coding. The main way to improve the robustness in architecture design is to exchange time and space for performance. And to improve the robustness for coding, we can utilize some program static analysis tools such as SONAR, PMD and so on. The second open issue is for maintainability and expandability, take the processing of online payment as an example, our system class provides an interface and concrete class and whenever we want to change the way we pay, we can only extend the functinality of this subsystem.

Finally, it comes to our self reflection. We reflect our project in teamwork, developing tools, presentation and resources and we also reflect what each of our group members has achieved. It is the group discussion from time to time that help us to progress. And we find a good developing tool, draw.io. Most importantly, we want to thank our professors for their directions.

2.Architecture

2.1 Global Architecture

2.1.1 Architecture Diagram



2.1.2 Architecture Description

The high level architecture is divided as four different layers which have their own certain function , and also the majority of them correlative with others, they named as **Presentation Layer**, **Services Layer**, **Controller Layer** and **Data Layer**.

- **Presentation Layer**

The Presentation Layer is in charge of providing user interface and handle user interactions. We renew it from the last assignment which we called it as UI Layer and the packages were also renew from the Mobile Terminal UI and Web Terminal UI.

On the web frontend, we choose the popular CSS and JavaScript frameworks, such as **vue.js** and **jQuery** to design our HTML web pages.

> **Why do we choose vue.js?**
>
> Vue is a progressive framework for building user interfaces. Unlike other large frames, Vue is designed to be applied from the bottom up. Vue's core library focuses only on the view layer, making it easy to get started and easy to integrate with third-party libraries or existing projects. On the other hand, Vue is also fully capable of driving complex single-page applications when used in conjunction with modern toolchains and various support libraries.

> **Why do we choose jQuery?**
>
> jQuery is a JavaScript tool library (class library) that gets a whole set of defined methods by encapsulating native JavaScript functions. Integrates the power of JavaScript, CSS, DOM and Ajax.
>
> We can write only a little code to gain a wonderful effect.

The **Template Engine** is a tool for parsing corresponding type template files and then dynamically generating view files consisting of data and static pages. It responds to various parsing actions through tags, and dynamically displays the corresponding data to a specified location by means of variable occupancy.

We have also imported the **template engine of Django** in the realization of web frontend, which serves as the templates of MTV architecture pattern.

Since our Customer APP and Delivery APP are based on Android, frameworks like okhttp3, Retrofit2, GSON will be used in those modules.

> **Okhttp**
>
> Mainstream web request framework
>
> The whole process is: Convert the constructed Request to Call through OkHttpClient, then perform asynchronous or synchronous tasks in RealCall, and finally send out the network request and get the returned response through some interceptor interceptor.
>
> **Why do we choose OkHttp?**
>
> 1) Support http2, share all requests for one machine to share the same socket
>
> 2) Built-in connection pool, support connection multiplexing, reduce latency

3) Support transparent gzip compression response body

4) Avoid duplicate requests through caching

5) Automatically retry the host's other ip when the request fails, automatically redirect

6) Easy to use API

Retrofit2

Retrofit2 is simply a network request adapter that translates a basic Java interface into an HTTP request via a dynamic proxy and sends the request through OkHttp. It also has great scalability, support for various format conversions and RxJava.

GSON

GSON is a Java class library provided by Google to map between Java objects and JSON data. You can convert a Json character into a Java object or convert a Java to a Json string.

Why do we choose GSON?

1) fast and efficient

2) the amount of code is small, simple

3) object-oriented

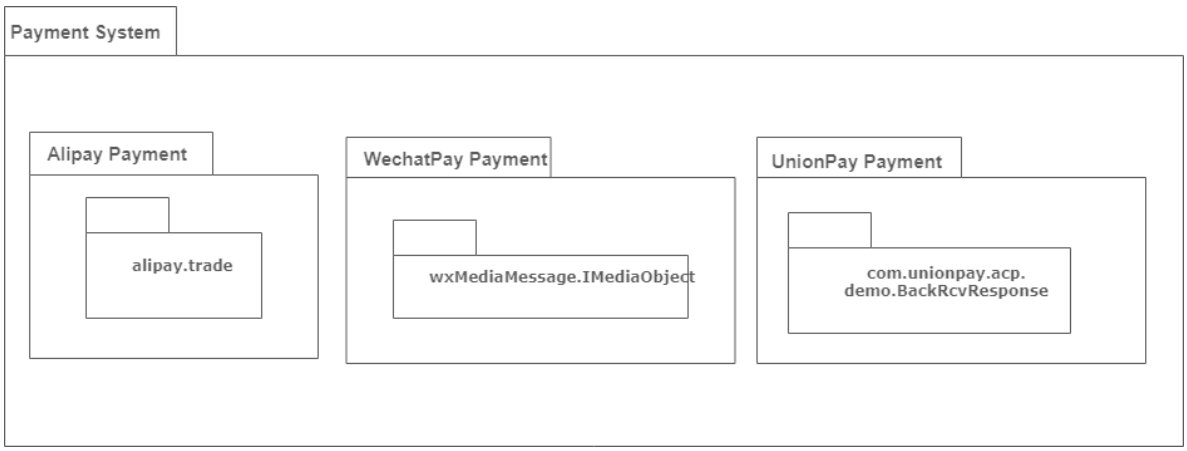
4) data transfer and analysis is convenient

- **Services Layer**

It provides unified API interfaces of all functionalities for both web clients and mobile clients. The web APIs are provided in the form of JSON services based on HTTP/HTTPs.

In the Dialogue Controller subsystem, we import Django's url library **"Django.conf.urls.url"**. The **Amap API** is called in the mapping system, and the **SF API** is called in the Express System which is used for checking logistics information such as: Where the books now I bought? How far is it from me? etc.

The Payment System supports three kinds of payment modes, **Alipay**, **WechatPay** and **UnionPay**. We read the official document of the API and designed adapters for those to help us to use it conveniently.



- **Controller Layer**

In this layer, we build three main controllers: User Controller, Book Controller, Order Controller. And we also hold a Delivery Schedule and Django_view.

In User Controller, it mainly contain the behavior that the users interact with the system. The user can check and modify his or her information, modify personal information, releases notices(this is only for administrator) and get issue coupons.

In Book Controller, it contain modify the book information, search book, order by sold amount, stack check(if close to the limit amount, we should send a message to the publisher), modify the evaluation and purchase alarm.

In order Controller, it contain check order information , modify generate or delete order, refund, payment reminder, statistics order and return.

We also create a Express Controller, which is powered by Amap API and SF API. It is used for customer to check his or her express information and can get the location of the books he or she brought at the first time.

- **Data Layer**

The Data Access Layer is in charge of all data accessing operations. It provides unified data accessing interfaces for different data models. The system introduces three different databases for storing data:

MySQL for storing most part of the data, which is highly relational and has relatively high demand of ACID;

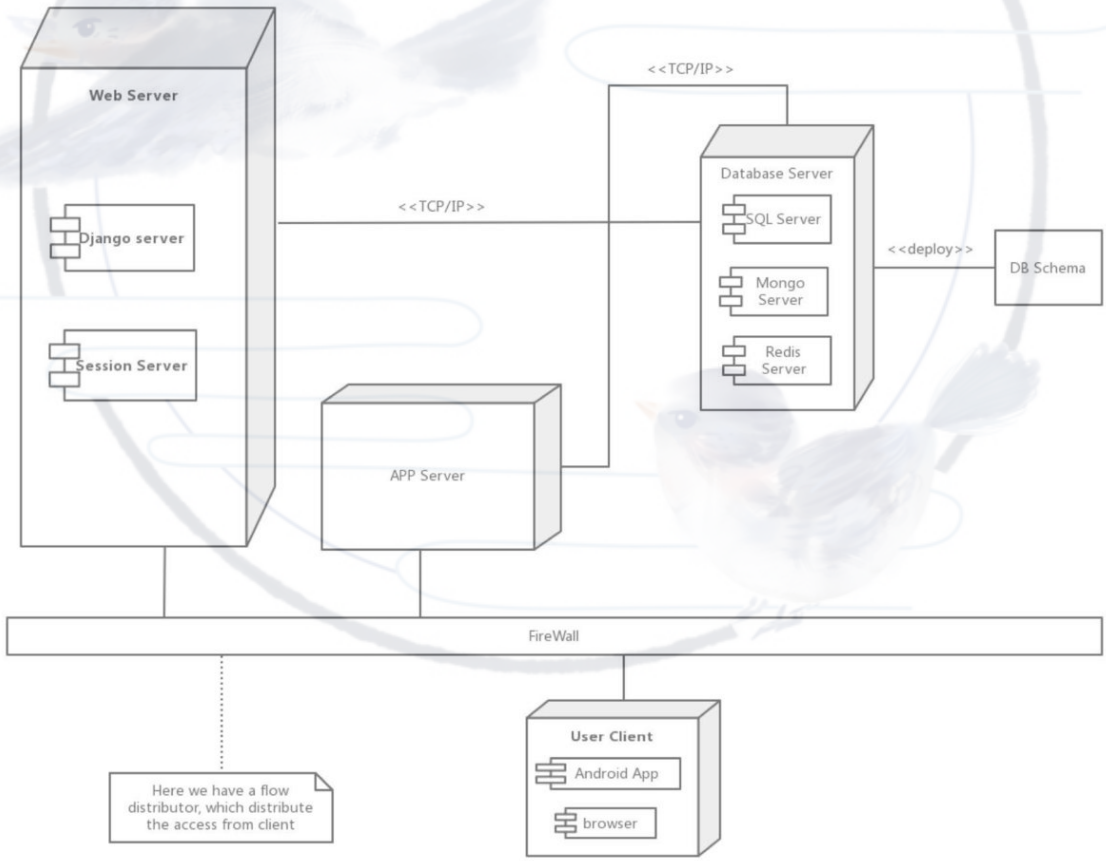
MongoDB for storing mass data that is less relational and has less demand of ACID, such as GPS location data from all GPS sensors in each collecting period (30 seconds). MongoDB enables those kinds of data to be stored and accessed more efficiently.

Finally, the system requires **Redis**, which is an in-memory (however data can be persistent) high performance Key-Value database, which is used to store cache and sessions that are highly performance sensitive. Notice that Redis also supports a rich kind of data structures, which can significantly boost the calculation of some kind of ranking related tasks.

For MongoDB, the system uses ODM (Object Document Mapping) frameworks to map objects to MongoDB queries. For MySQL, the system uses ORM (Object Relational Mapping) frameworks to map objects to SQL queries.

ORM and ODM frameworks provides better readability while fundamentally prevents SQL injections. Both ODM and ORM frameworks exposes interface under the caching layer. In this kind of architecture, the caching layer is transparent to the developers. Developers need not to care about whether to access the cache or database. This can simplify the development process while keeping high performance. The cache layer is dependent on Redis (and its drivers) of course.

2.1.3 Deployment Diagram



- User Client

We hold two kinds of service mode for User Client: Android app and browser.They are provided services by different servers.

- firewall

There is a flow distributor on the firewall which distribute the assess from the client. All operations on the server are filtered through the firewall.

- web server

The session server is designed for provide authentication services. Page routing and rendering are finished by Django server.

- database server

The web server and app server share one database server, where redis provides key-value cache and improves authentication speed. Besides, SQL server and Mongo server will assume relational and non-relational data storage services respectively.

> **Why do we choose Mongo Server?**
>
> MongoDB Atlas delivers the world's leading database for modern applications as a fully automated cloud service engineered and run by the same team that builds the database. Proven operational and security practices are built in, automating time-consuming administration tasks such as infrastructure provisioning, database setup, ensuring availability, global distribution, backups, and more. The easy-to-use UI and API let you spend more time building your applications and less time managing your database.

> **Why do we choose Redis Server?**
>
> Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes with radius queries and streams. Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.

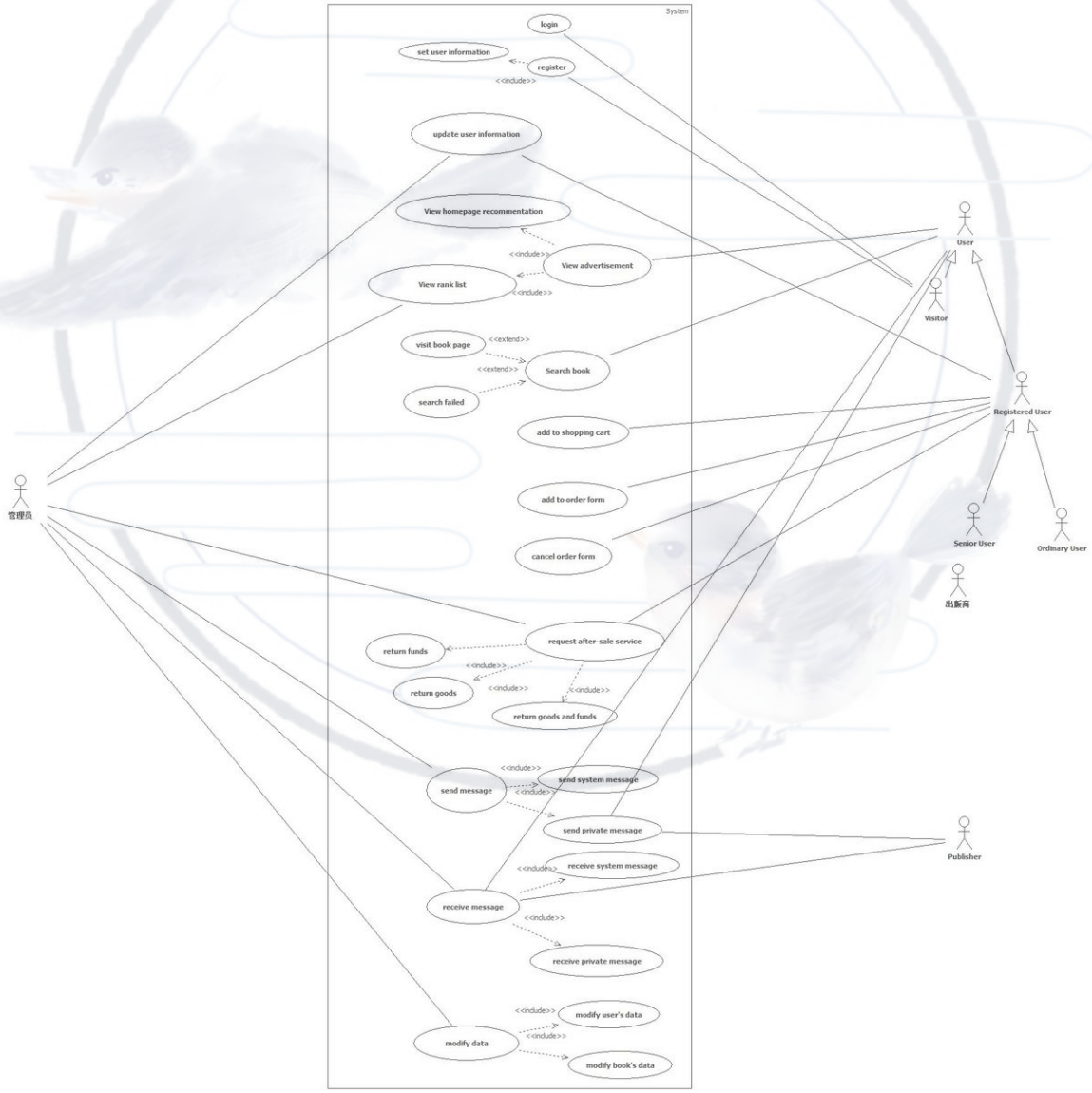
2.2 Subsystems and Interfaces

SubSystem	Provided Interfaces	Operations	Required Interfaces	Operations
APP	Customer Access Administrator Access	+getExpress() +getPosition() +HistorySearch(Tracking_number)	OKHttp	+OkHttpClient() +client.newCall(request).execute() +response.isSuccessful() +response.body() +onFailure() +onResponse()
Web	WEBFronted	+POST(args)		
Map	Map	+getMap(location) +getAddress() +getLocation(address) +getEstimateTime(locatioin,address)	Amap API	+Map(id,option) +getCenter() +setView(center,zoom) +routePath(startOption,stopOption,sucessback,errback,option)
Express	Express	+getExpress() +getPosition() +HistorySearch(Tracking_number)	SF API	+getOrderServiceRequestXml(ExpressOrder expressOrder) +CallExpressServiceTools.callSfExpressServiceByCSIM(requestXml) +shunFengCallback(String content)
Payment	Payment	+payOnline(orderID,paymentWay,receiptType,coupon) +payOffline(orderID,userID,paymentWay,receiptType ,coupon) +getPaymentReceipt(orderID) +getPaymentInformation(orderID) +refund(orderID) +choosePayMethod() enterPassWord() +showPayment()	AlipayPayment	+DefaultAlipayClient(args) +AlipayTradePrecreateRequest(args) +AlipayTradePayRequest(args) +AlipayTradeRefundRequest(args)
			WechatPayment	+POST(args) +IMediaObject(args) +wxpay.unifiedOrder() +wxpay.orderQuery() +wxpay.refundQuery() +wxpay.downloadBill() +WXPayUtil.mapToXml() +wxpay.isPayResultNotifySignatureValid() +WXPayUtil.xmlToMap()
			UnionPayment	+AcpService.sign() +SDKConfig.getConfig().getBackRequestUrl() +AcpService.validate() +rspData.get()
Order	Orderlinformation	+checkPayment() +confirmPayment() +getOrderLocation() +createInvoice(title, items, type) +check(userID, signature) + check(userID, fingerprint)	EBusinessPlatform	+getNewOrder():Order
			TaxationAuthority	+createNewInvoice(Order,title,timestamp):invoiceID +getInvoice(invoiceID):Invoice
	OrderManagement	+getOrder(orderID) +getOrderList(orderIDList) +updateOrder(orderID,commandType,args) -setWithdraw(orderID) -judgeReplace(order)		
User Management	Customer_Operation	+register(name,password,ID) +login(ID,password) +setInfo() +favourite(Book:Book) +selectBook(info:string) +immediateBuy(Book:Book) +setContent(content:string) +sendMessage(Message)		
	Administrator_Operation	+login(ID,password) +setInfo() +setContent(content:string) +sendMessage(Message) +modifyUserData() +modifyBookData()		
	Publisher_Operation	+register(name,password,ID) +login(ID,password)+setInfo() +setContent(content:string) +sendMessage(Message)		
Model	IOrderModel IScheduleModel IAddressModel IPaymentModel IInvoiceModel IUserModel	+create(dict) +save() +update(dict) +delete(dict) +get(dict) +filter(dict) +sort(dict)	ODM Framework ORM Framework K-V Framework	

	IDiscountModel	+check(dict)		
	ITaxModel			
	IInventoryModel			
DataAccess	MySQL	+SELECT(args) +INSERT(args) +UPDATE(args) +DELETE(args)		
	MongoDB			
	Redis			

3.Use Case Modelling

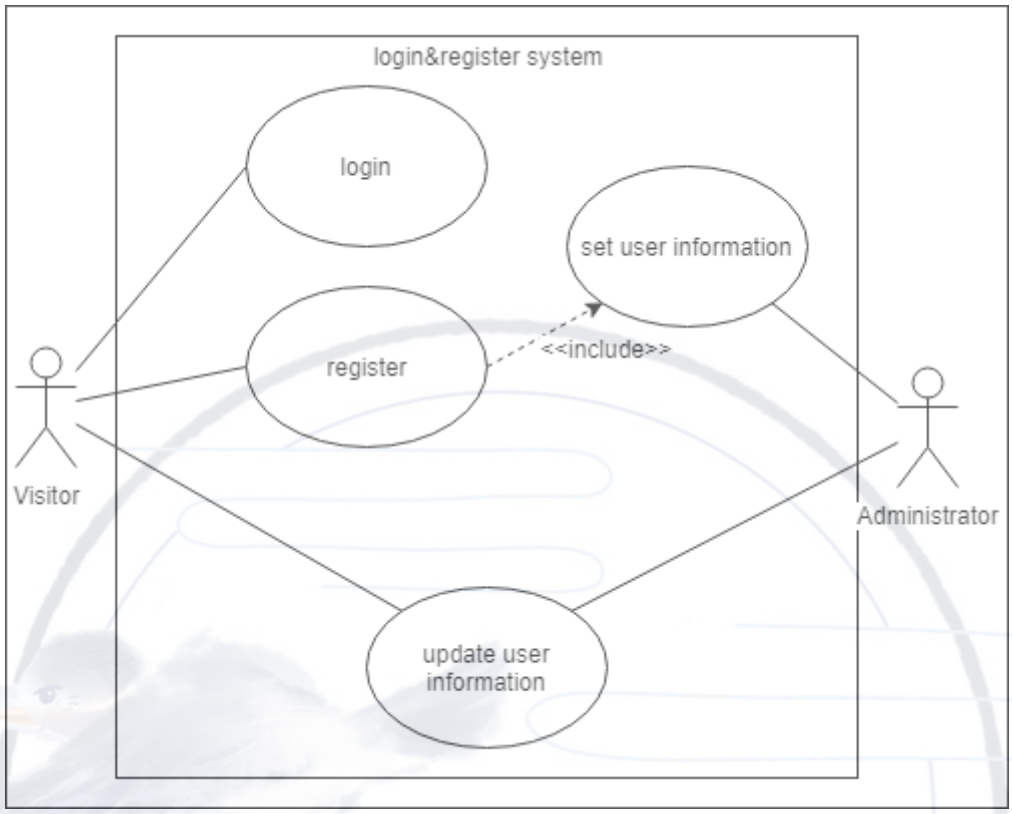
3.1 Global System Use Case



3.2 Subsystem Use Case

3.2.1 login & register Subsystem

- Use case diagram



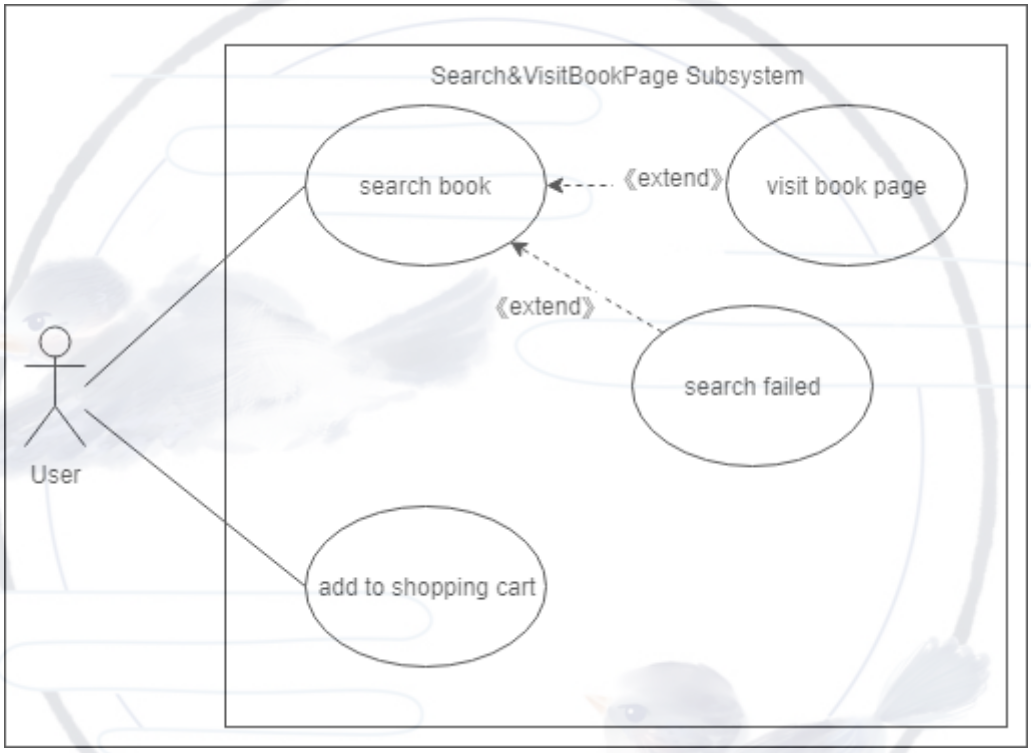
Use case: Login & Register System
Actor: Customers, Administrator, Publisher
Basic flow: 1. Customers/Administrator/Publisher select the button "Log in". 2. Customers/Administrator/Publisher input their username and password into the system. 3. Customers/Administrator/Publisher enter the system successively.
Alternative flow: 1. If the customer is a new one who enter our system, he or she are supposed to select the button "Register" and then input required information and set password for their accounts. 2. Customers enter the password wrongly, the system will allow them to have tries for another 3 times. 3. Customers forget the password, then the system will offer a security code to customers, asking customers to reset the password. 4. Administrators forget the password, he or she can use his or her ID to reset the password again. 5. Publishers forget the password, he or she can send a message to administrators to send their password again.

Use case: Login & Register System

Post condition:
Customers have access to checking up the information of the order, paying for the bill, and signing the package.

3.2.2 search & visitBookPage Subsystem

- Use case diagram



Use case: Search & VisitBookPage Subsystem

Actor: Customer

Precondition: Connect to the internet and successfully access the page or app

Use case: Search & VisitBookPage Subsystem

Basic flow:

A. The user enters the content in the search box (author name or book name or publisher name)

B. Return search results

a) The book or author does not exist and returns "The bookstore does not have this author, book or publisher"

1. Display a list of related books related to the book

b) The book or author exists

1. Display simple information about the book (title, author, price, publisher)
2. Similar results appear at the bottom of the search results

c) Search results can be adjusted by sorting (sales descending/ascending, rating descending/ascending)

C. The user clicks on the book to access the book homepage.

D. The book's home page displays the details of the book (cover preview, title, author, price, publisher, reader comments)

E. Collecting books

a) The user is not logged in and jumps to the login screen.

b) User has logged in, add books to favorites

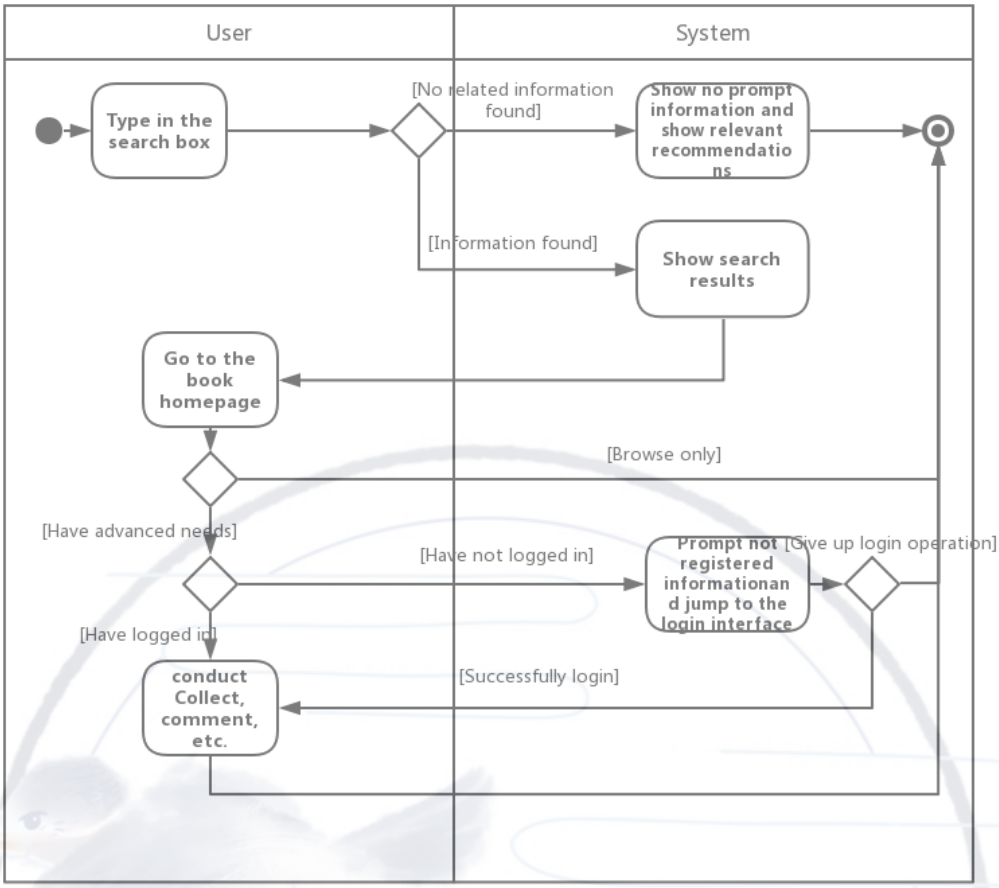
F. Show comments for this book

a) The user is not logged in and can only view comments

b) User login

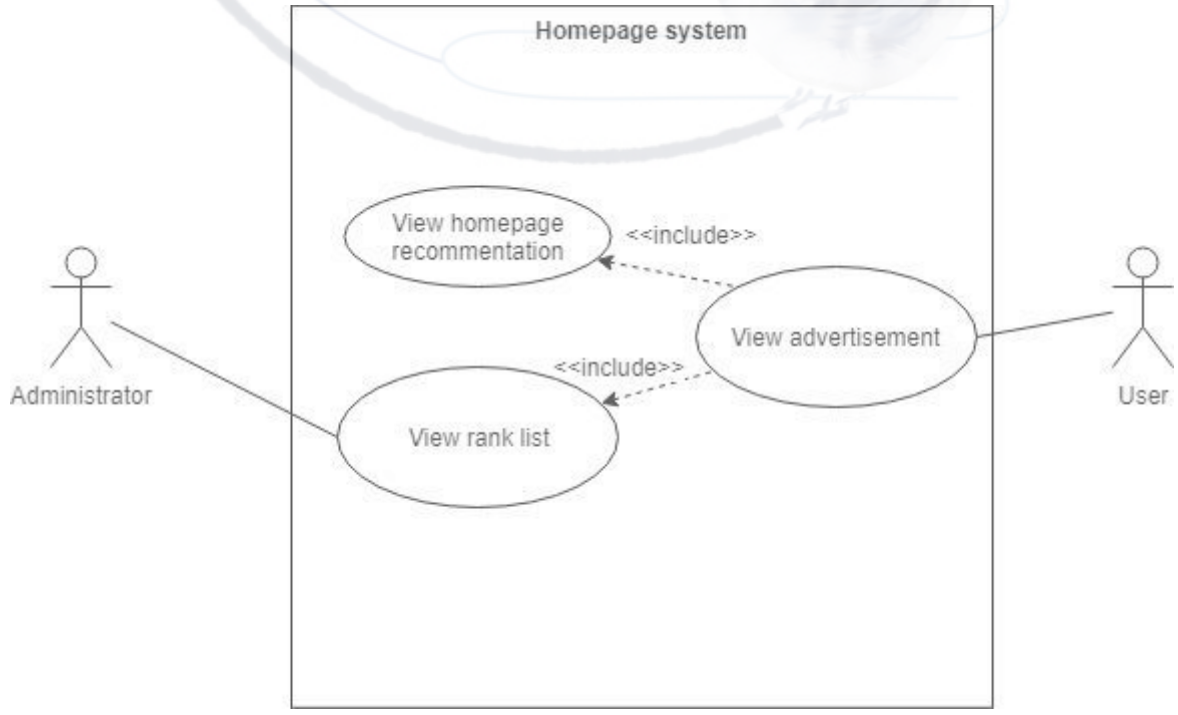
1. The book has not been purchased, only comments can be viewed
2. Purchased the book to comment on the book or respond to comments from others

- Activity Diagram



3.2.3 Homepage Subsystem

- Use case diagram



Use case: View ADVERTISEMENT

Actor: User, Administrator

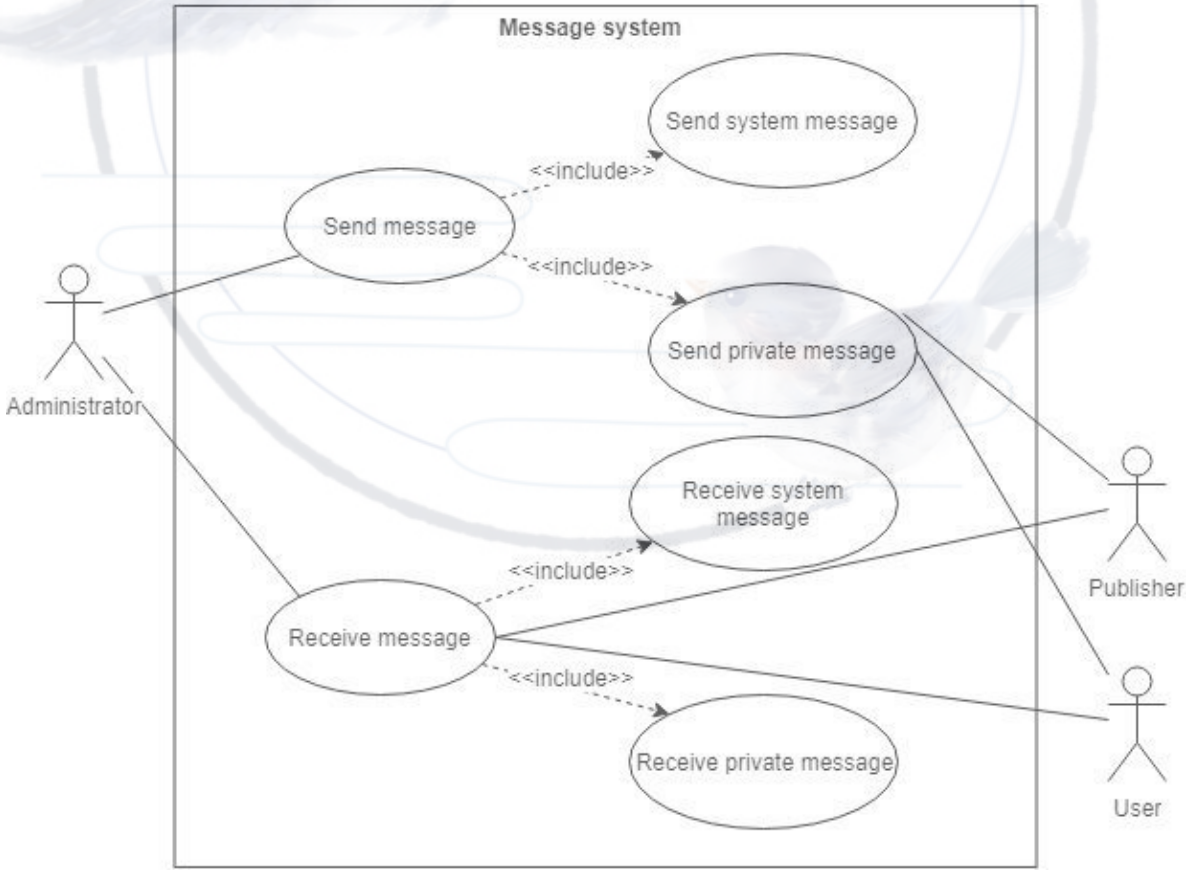
Precondition: User/Administrator login

Flow of events:
1. User/Administrator view our rank list and homepage recommentation.
2. User/Administrator click on the book he/she want to further learn about.
3. User/Administrator enter the selected book's page successively.

Post condition:
Customers can learn more about the book and have some actions on it.

3.2.4 Message Subsystem

- Use case diagram



Use case: SEND MESSAGE

Actor: User, Publisher, Administrator

Use case: SEND MESSAGE

Precondition: For private message, Administrator, Publisher and User know the ID of each other.

Flow of events:

1. The Administrator, Publisher or User send private message
   A. The Administrator, Publisher or User open the message window.
   B. The Administrator, Publisher or User set the receiver's ID.
   C. The Administrator, Publisher or User enter content in the message box.
   D. The Administrator, Publisher or User click on "send" button.
   E. The receiver receive message.
2. The Administrator send system message
   A. The Administrator open the message window.
   B. The Administrator set the type "system message".
   C. The Administrator enter content in the message box.
   D. The Administrator click on "send" button.

Use case: RECEIVE MESSAGE

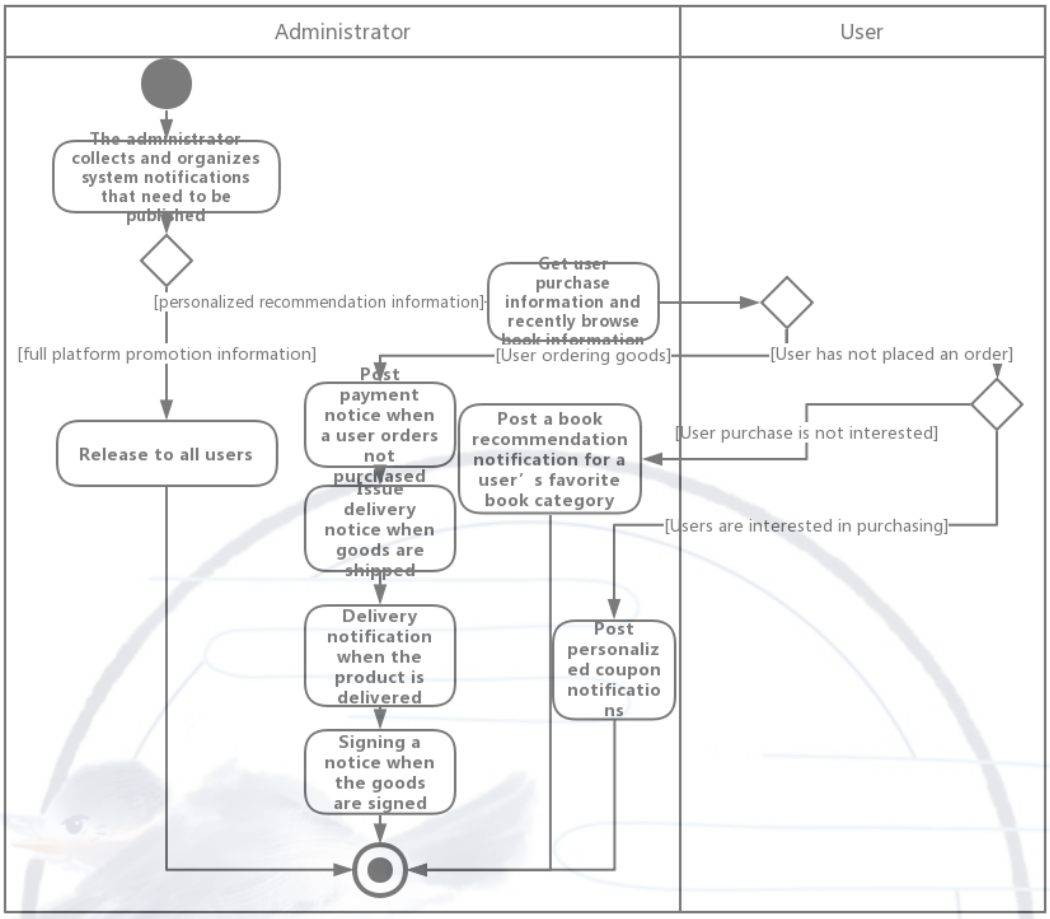
Actor: User, Publisher, Administrator

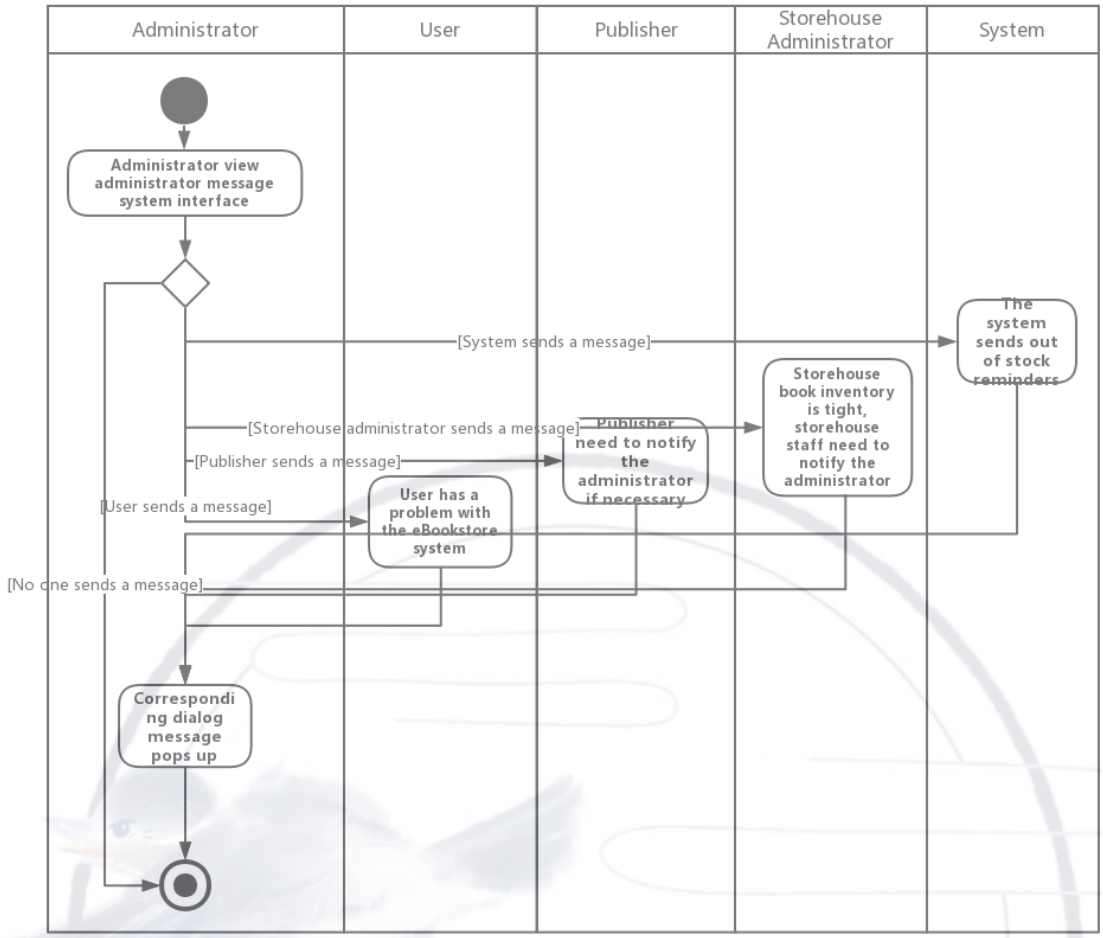
Precondition: Administrator, Publisher and User know the ID of each other and send message.

Flow of events:

1. The Administrator, Publisher or User receive private message
   A. The Administrator, Publisher or User open the message window.
   B. The Administrator, Publisher or User view the message content.
2. The Publisher or User receive system message
   A. The Administrator view the homepage and see the system message.

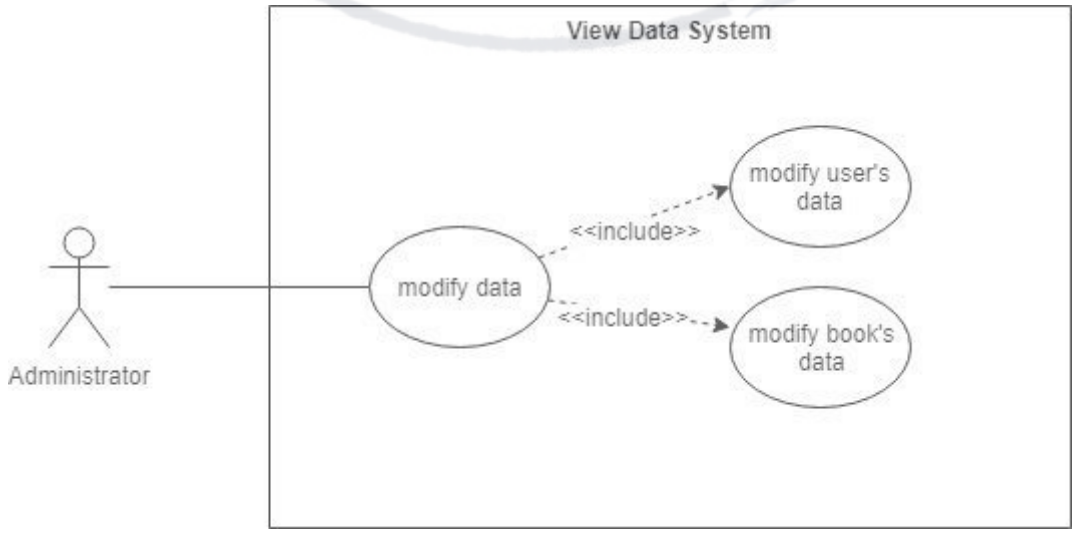
- Activity Diagram


Administrator
User
The administrator collects and organizes system notifications that need to be published
[personalized recommendation information]
[full platform promotion information]
Release to all users
Get user purchase information and recently browse book information
[User ordering goods]
[User has not placed an order]
Post payment notice when a user orders not purchased
Post a book recommendation notification for a user's favorite book category
[User purchase is not interested]
[Users are interested in purchasing]
Issue delivery notice when goods are shipped
Delivery notification when the product is delivered
Signing a notice when the goods are signed
Post personalized coupon notifications



3.2.5 View Data Subsystem

- Use case diagram



Use case: View DATA

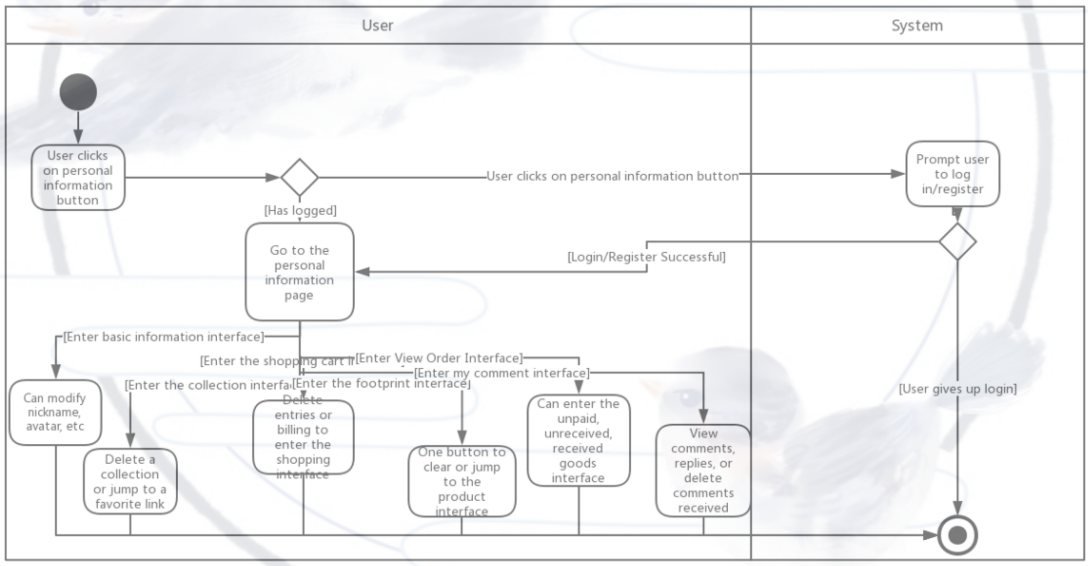
Actor: Administrator

Precondition: Administrator login and is verified.

Flow of events:
1. Administrator view data analysis window.
2. Administrator choose user's data.
   A. Administrator check if the user's data is right and tell the user to correct it.
   B. Adminstrator modify the user's data like change some user into superior user.
3. Administrator choose book's data.
   A.Administrator check if the book's data is right and correct it.
   B. Administrator update book's data like modifying its price.

Post condition:
Customers can learn more about the data and have some actions on it.

- Activity Diagram



3.3 Glossary of Terms

Term	Explanation
Registration	Visitors fill in their personal information, create exclusive account information, and assign accounts to them. After registration, visitors become registered users of the system.

Term	Explanation
Login	Registered users use personal account passwords to access personal accounts to obtain appropriate user rights and to use user functions such as purchases, favorites, comments, and so on.
Administrator	That is, the e-bookstore system is backstage, and the e-bookstore system is used to interact with other participants of the system, and some data of the system can be modified.
Advanced users	Registered users can become advanced users after multiple purchases of books, advanced users are marked as profitable, and administrators can issue coupons or push book advertisements to advanced users.
Tourists	No system account is registered, only information can be browsed and communicated with the administrator, but there are no user functions such as comments, favorites, shopping carts, purchases, etc.
Publisher	The e-book store's source of the book's goods, the administrator can adjust the purchase volume, replenish the source of goods, etc. by interacting with the publisher.
Coupon	The administrator issues it to a profitable senior user and can be used to exchange a discount.
Book page	The book page displays detailed information about the book, including price, author, publisher, comment, etc. Users need to enter the book page to comment, collect, add to the shopping cart, purchase, etc.
Shopping cart	Registered users can add books to their shopping carts for temporary storage, and then they can purchase all the books in the shopping cart by billing.
Order	An order is generated when the user settles, and the payment information, the delivery address, and the like are displayed.
Homepage recommendation	On the homepage of the e-book store, you can display the recent hot-selling or discounted books. Users can access the corresponding book pages by clicking directly.
Hot sales ranking	The system statistics are automatically generated according to the purchase order of the books in the near future. The user can view the hot-selling behaviors and select the books to provide reference for them. The administrator can view the hot-selling rankings to know the sales of the books and adjust the purchase volume of the books.
Collecting books	Users can collect favorite books and purchase them later in the book.

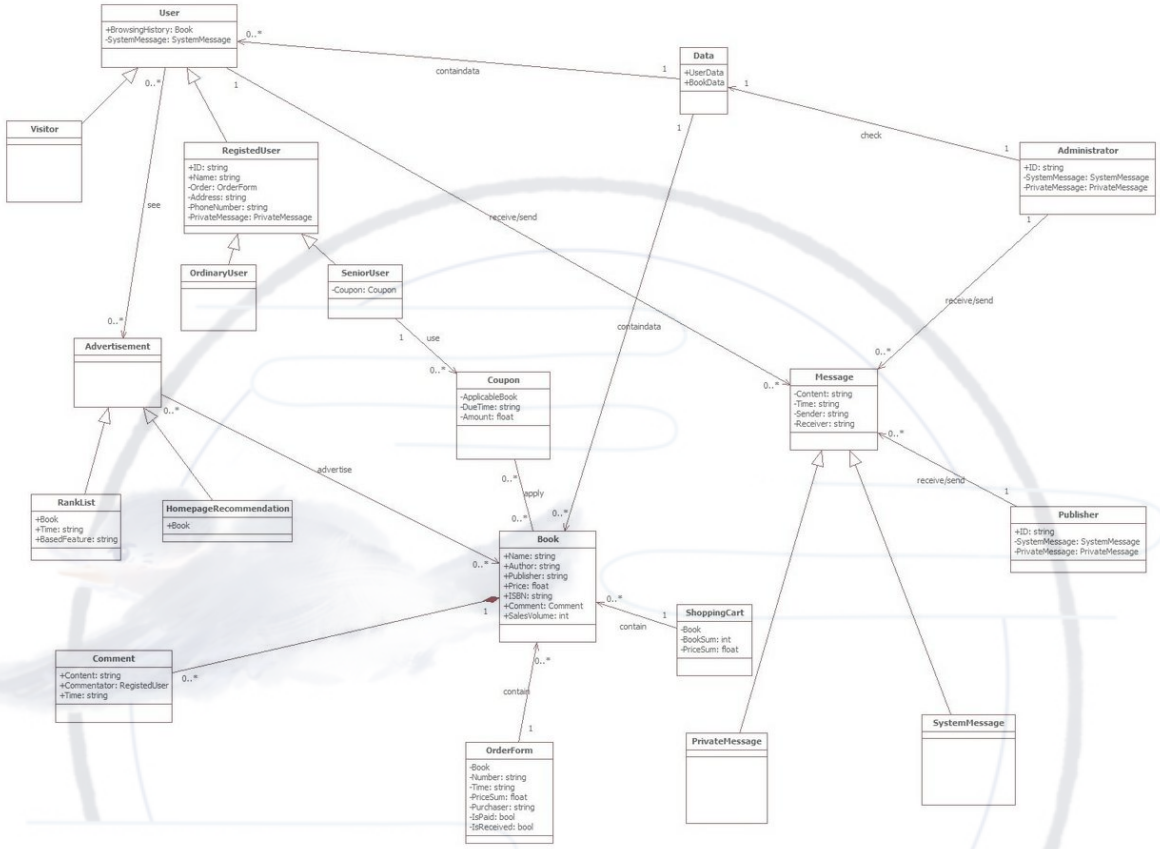
Term	Explanation
After sales service	A return service that can be requested after a user purchases a book.
Freight and tax rates	When you purchase a book online, you must deduct a certain fee based on the tax rate and the shipping cost when you mail it.
Personal information	Basic information filled out by registered users, including personal details, shipping address, etc.

3.4 Supplementary Specification

Item	Explanation
Goal	The purpose of this document is to determine the requirements for an online bookstore system. This supplemental specification lists the requirements that are not easily readable in the use case model use case. Supplementary specifications and use-case models work together to complete a complete set of requirements for the system.
Area	This specification applies to online bookstore systems, which are intended for users who wish to purchase books at the bookstore. This specification defines the non-functional requirements of the system; such as reliability, availability, performance, security, and functional requirements that are common across multiple use cases. (Function requirements are defined in the use case specification.)
Functionality	No.
Availability	The user interface of the online bookstore system on the mobile phone side should be adapted to the mobile phone interface, and the user interface of the online bookstore system on the pc end should be compatible with the operating system corresponding to the computer.
Reliability	The main system must be running 98% of the time. The system must be up and running during the online bookstore run.
Security	The system should prevent users or administrators from changing any valid information other than permissions. In addition, both the clerk level and the store owner level administrator can change the bibliographic information, but the store owner level takes precedence.

4.Domain Model

4.1 Class Diagram



4.2 Relationships Between Classes

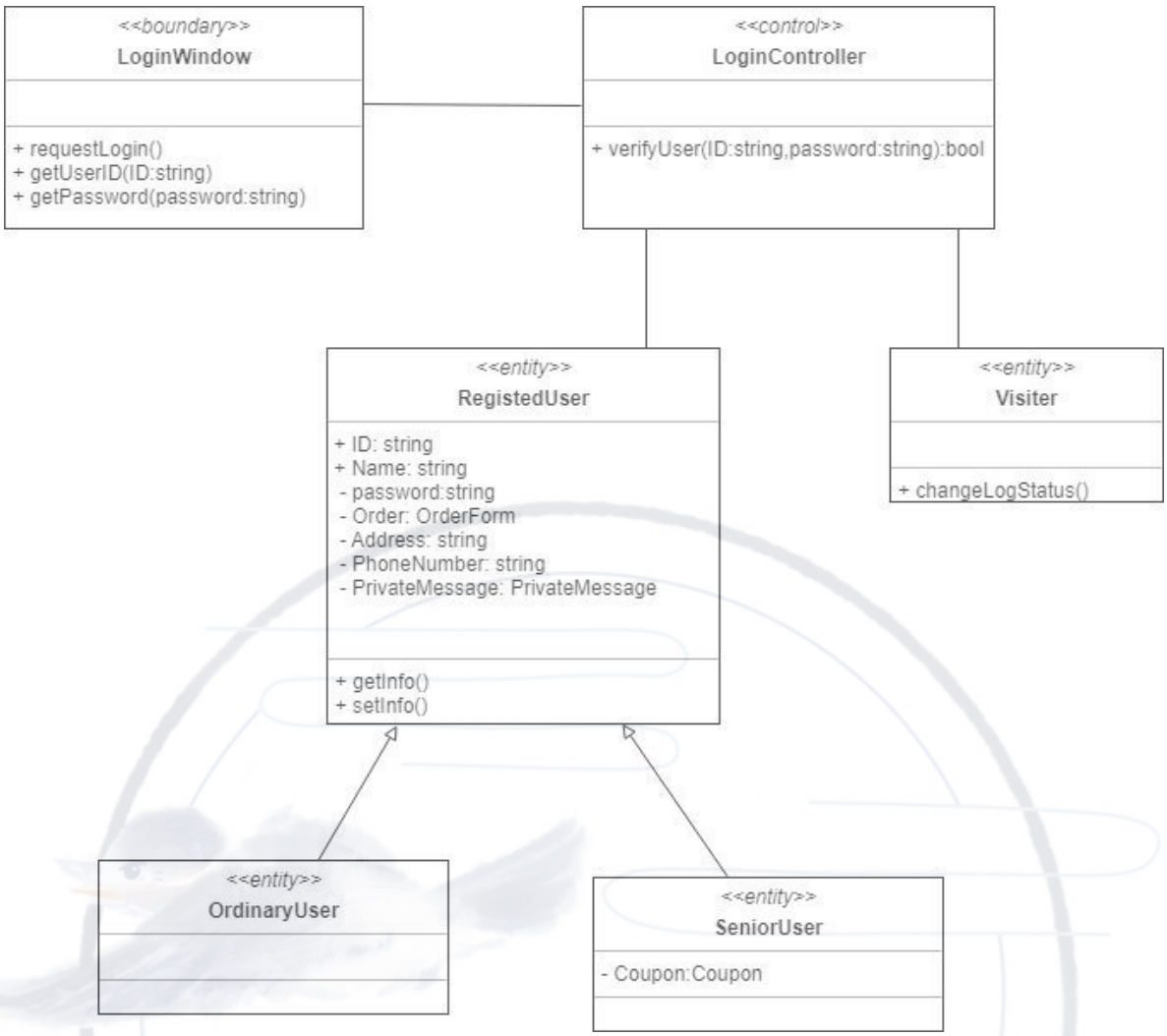
Conceptual symbol	Attributes
User	+BrowsingHistory:Book +SystemMessage:SystemMessage
Visitor	+BrowsingHistory:Book +SystemMessage:SystemMessage
RegistedUser	+BrowsingHistory:Book +SystemMessage:SystemMessage +ID:string +Name:string +Order:OrderForm -Address:string -PhoneNumber:string -PrivateMessage:PrivateMessage
OrdinaryUser	+BrowsingHistory:Book +SystemMessage:SystemMessage +ID:string +Name:string +Order:OrderForm -Address:string -PhoneNumber:string -PrivateMessage:PrivateMessage
SeniorUser	+BrowsingHistory:Book +SystemMessage:SystemMessage +ID:string +Name:string +Order:OrderForm -Address:string -

Conceptual symbol	Attributes
	PhoneNumber:string -PrivateMessage:PrivateMessage +Coupon:Coupon
Advertisement	
RankList	+Book:Book +Time:string +BasedFeature:string
HomepageRecommen-dation	+Book:Book
Coupon	-ApplicableBook:Book -DueTime:string -Amound:float
Book	+Name:string +Author:string +Publisher:string +Price:float +ISBN:string +Comment:Comment +SalesVolume:int
Comment	+Content:string +Commentator:RegistedUser +Time:string
OrderForm	-Book:Book -Number:string -Time:string -PriceSum:float -Purchaser:string -IsPaid:bool -IsReceived:bool
ShoppingCart	-Book:Book -BookSum:int -PriceSum:float
Data	+UserData +BookData
Administrator	+ID:string -SystemMessage:SystemMessage -PrivateMessage:PrivateMessage
Message	-Content:string -Time:string -Sender:string -Receiver:string
PrivateMessage	-Content:string -Time:string -Sender:string -Receiver:string
SystemMessage	-Content:string -Time:string -Sender:string -Receiver:string
Publisher	+ID:string -SystemMessage:SystemMessage -PrivateMessage:PrivateMessage

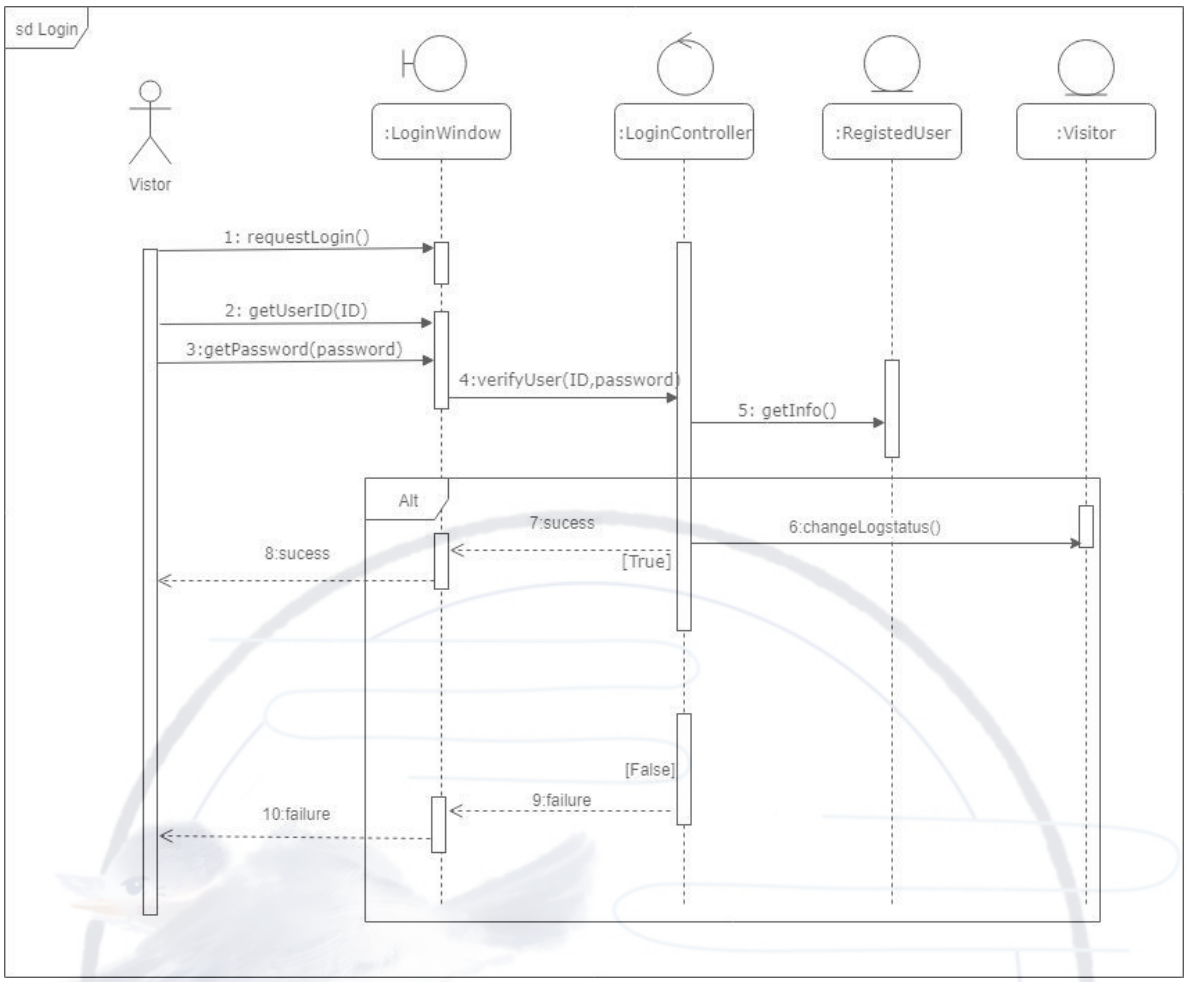
5.Analysis Model

5.1 Login

Class diagram

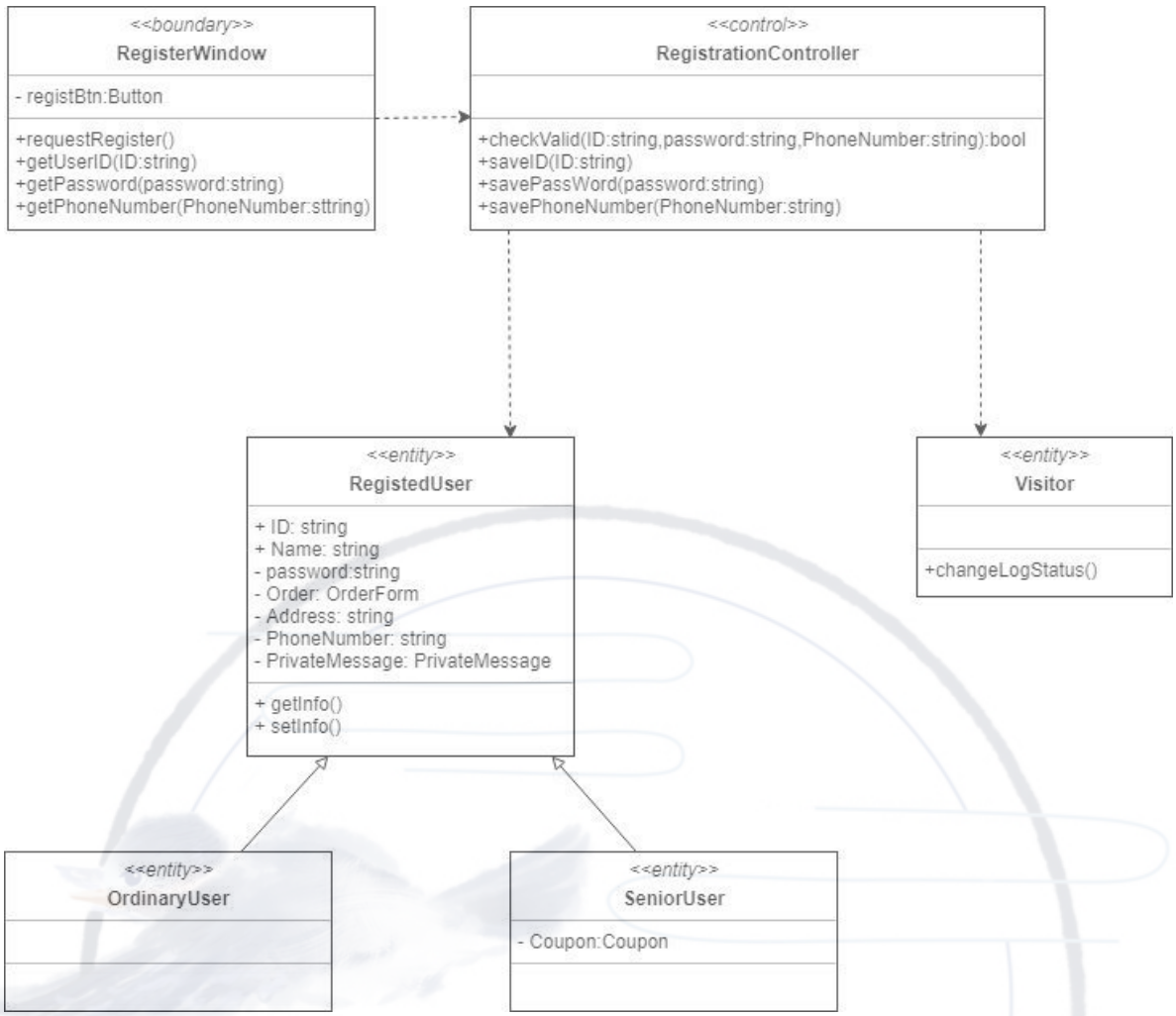


Sequence Diagram

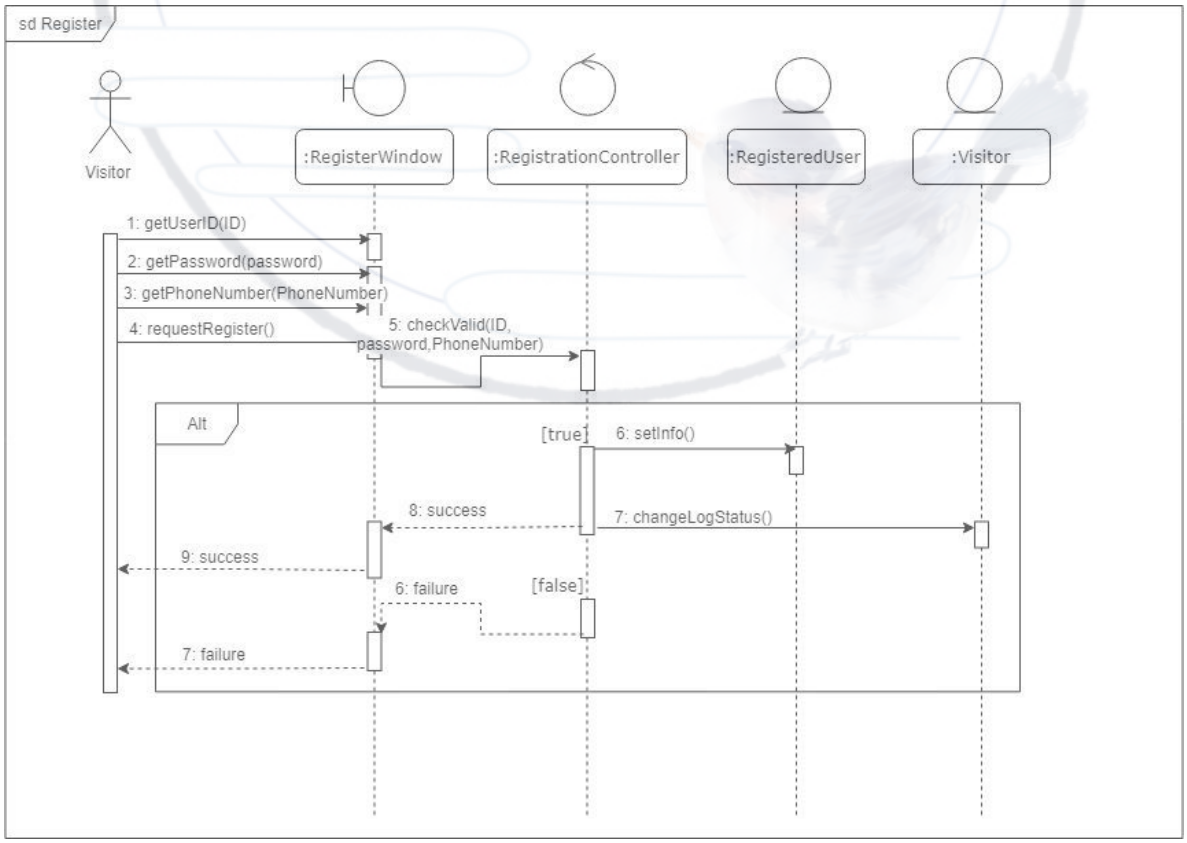


5.2 Register

Class Diagram

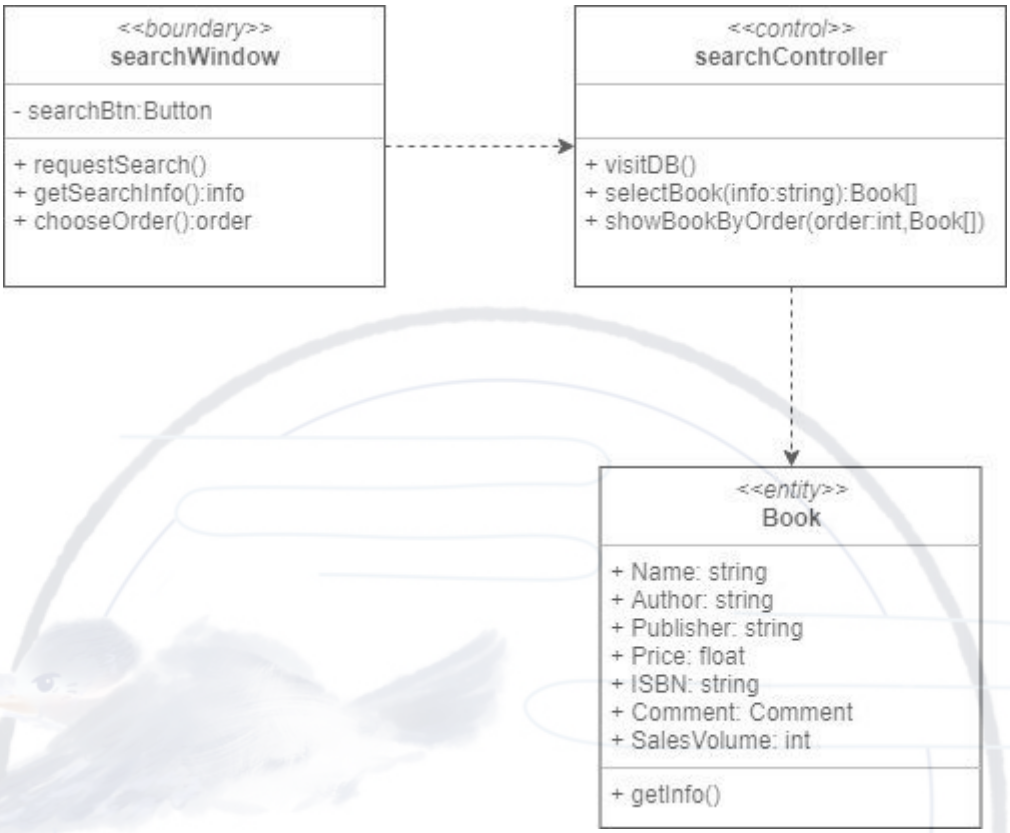


Sequence Diagram

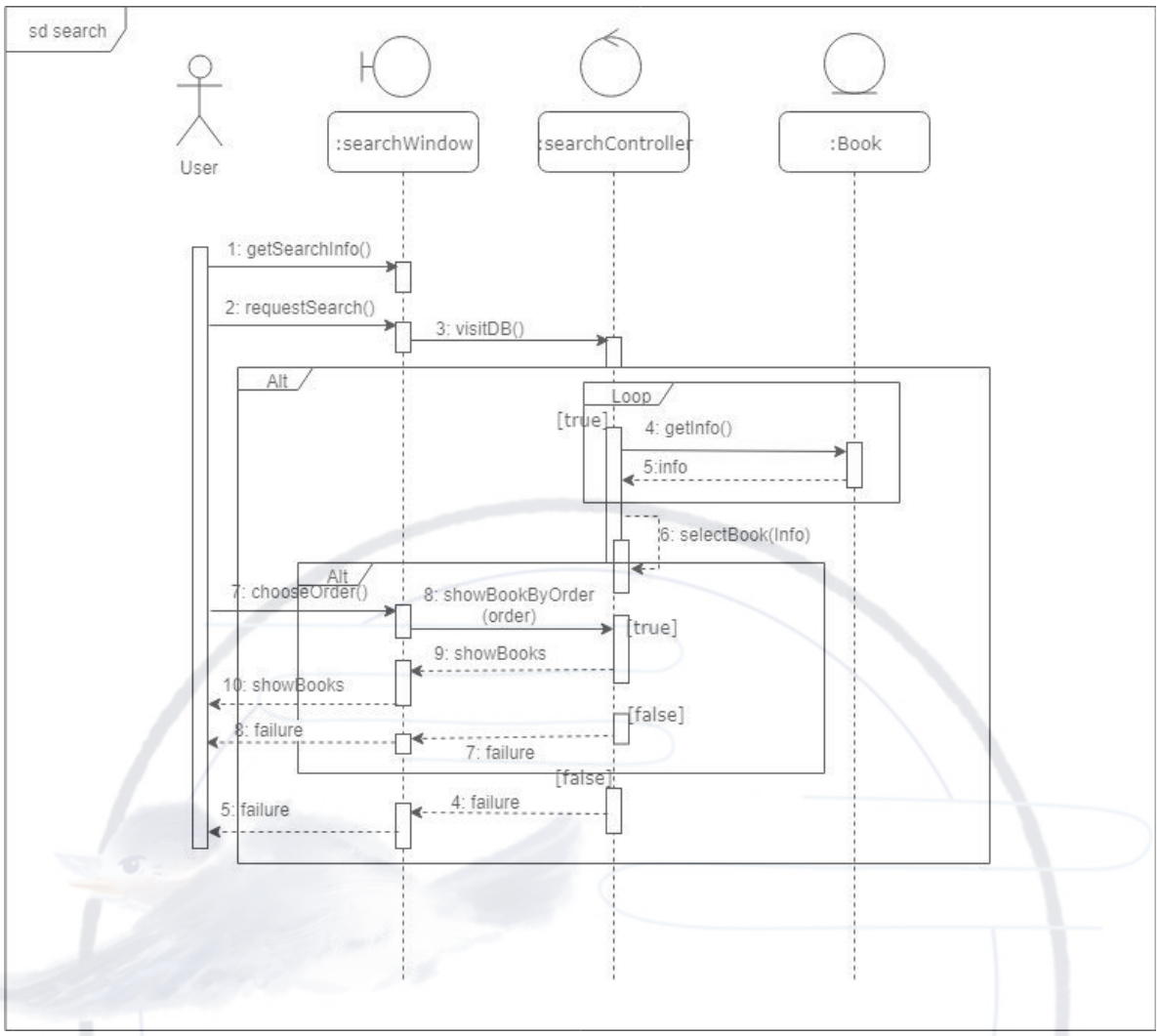


5.3 Search

Class Diagram

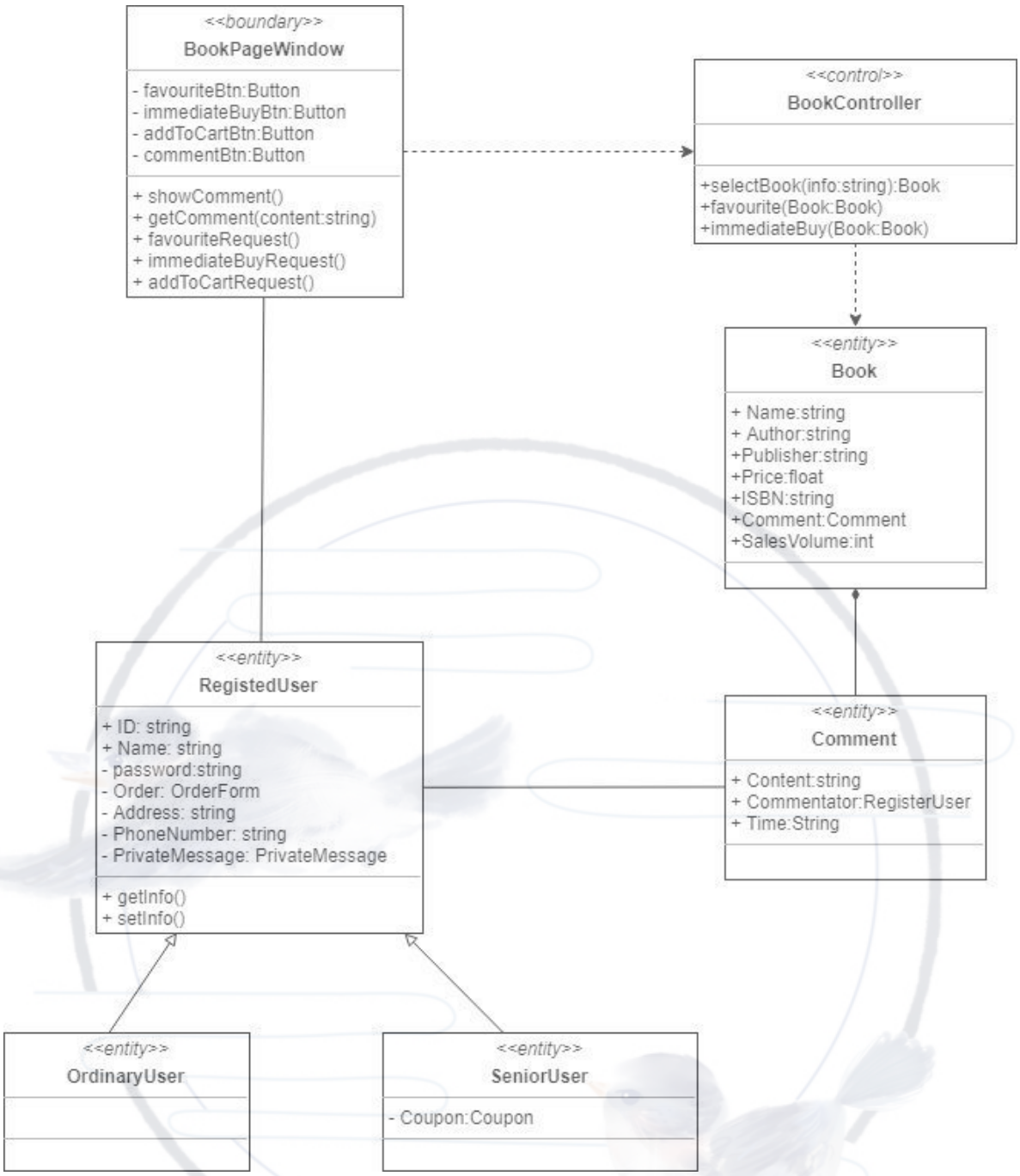


Sequence Diagram

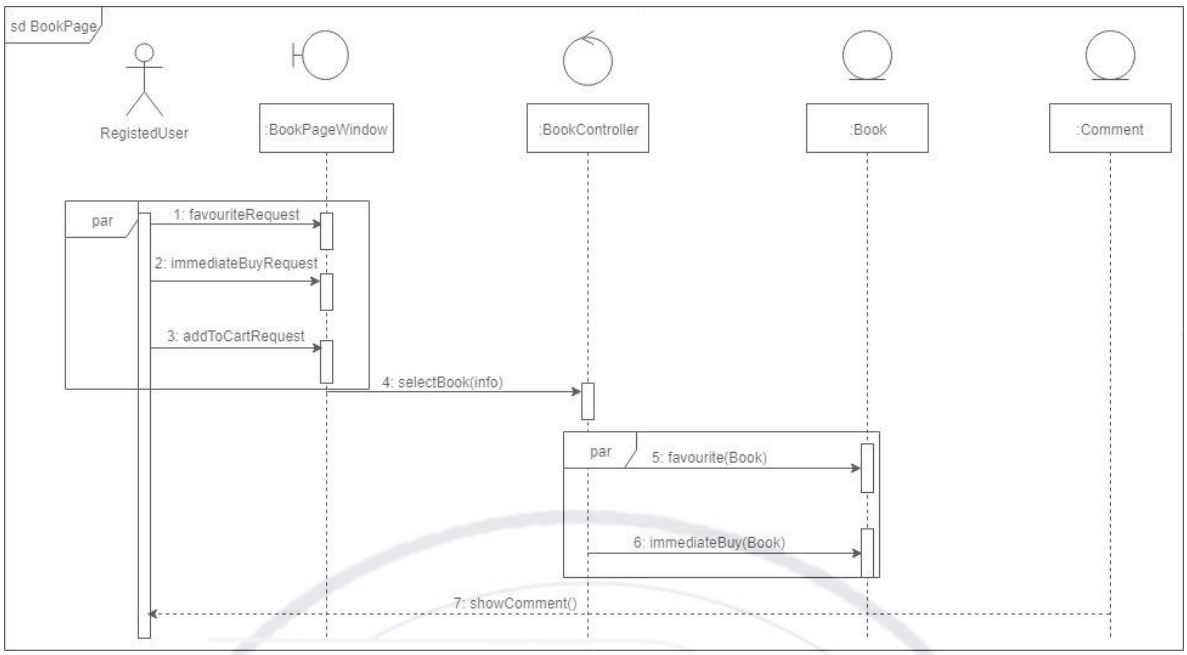


5.4 viewBookPage

Class Diagram

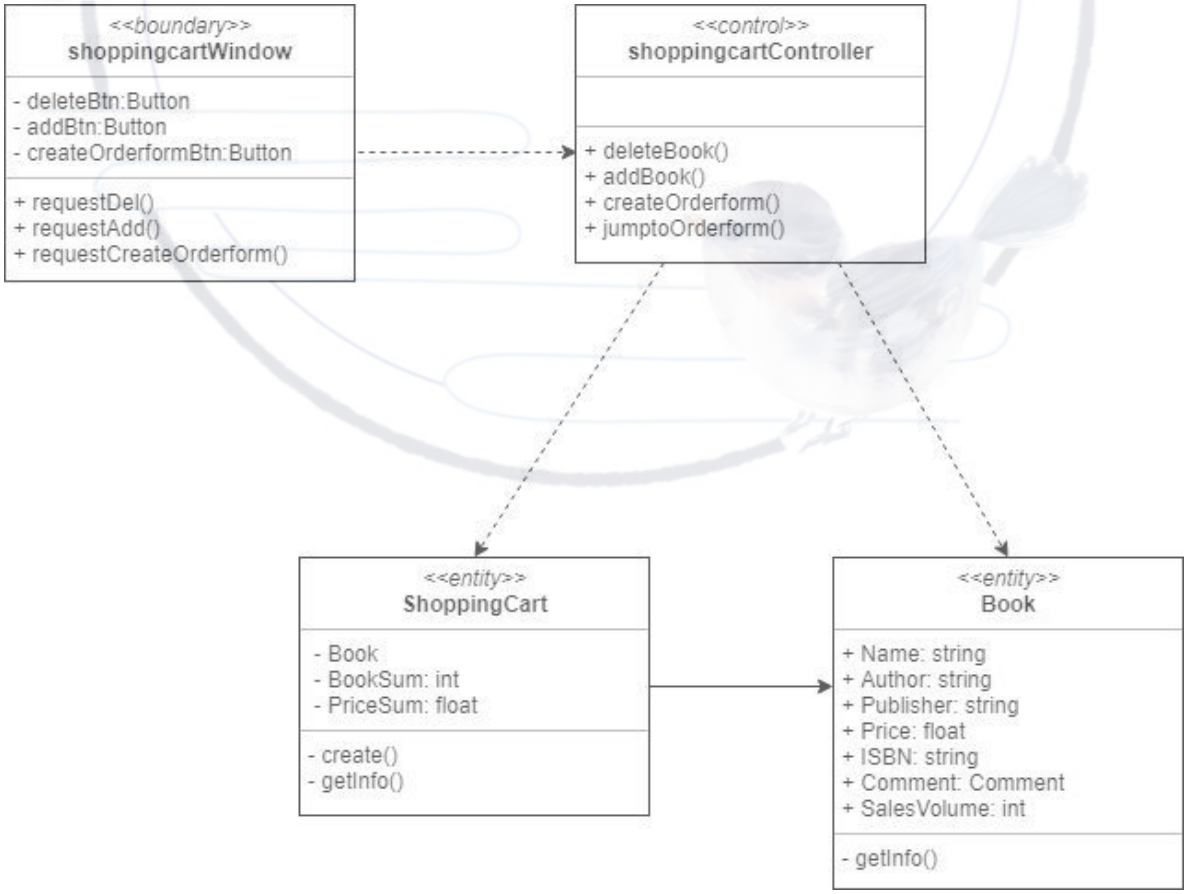


Sequence Diagram

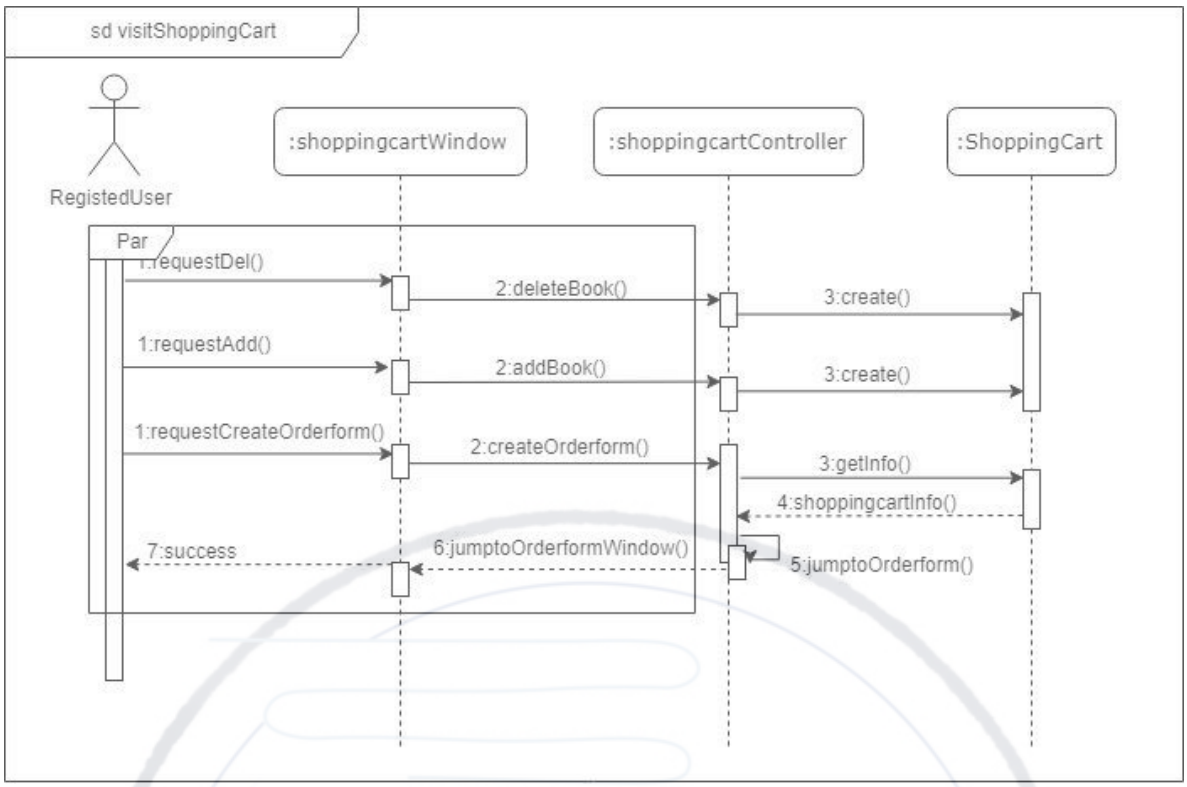


5.5 viewShoppingCart

Class Diagram

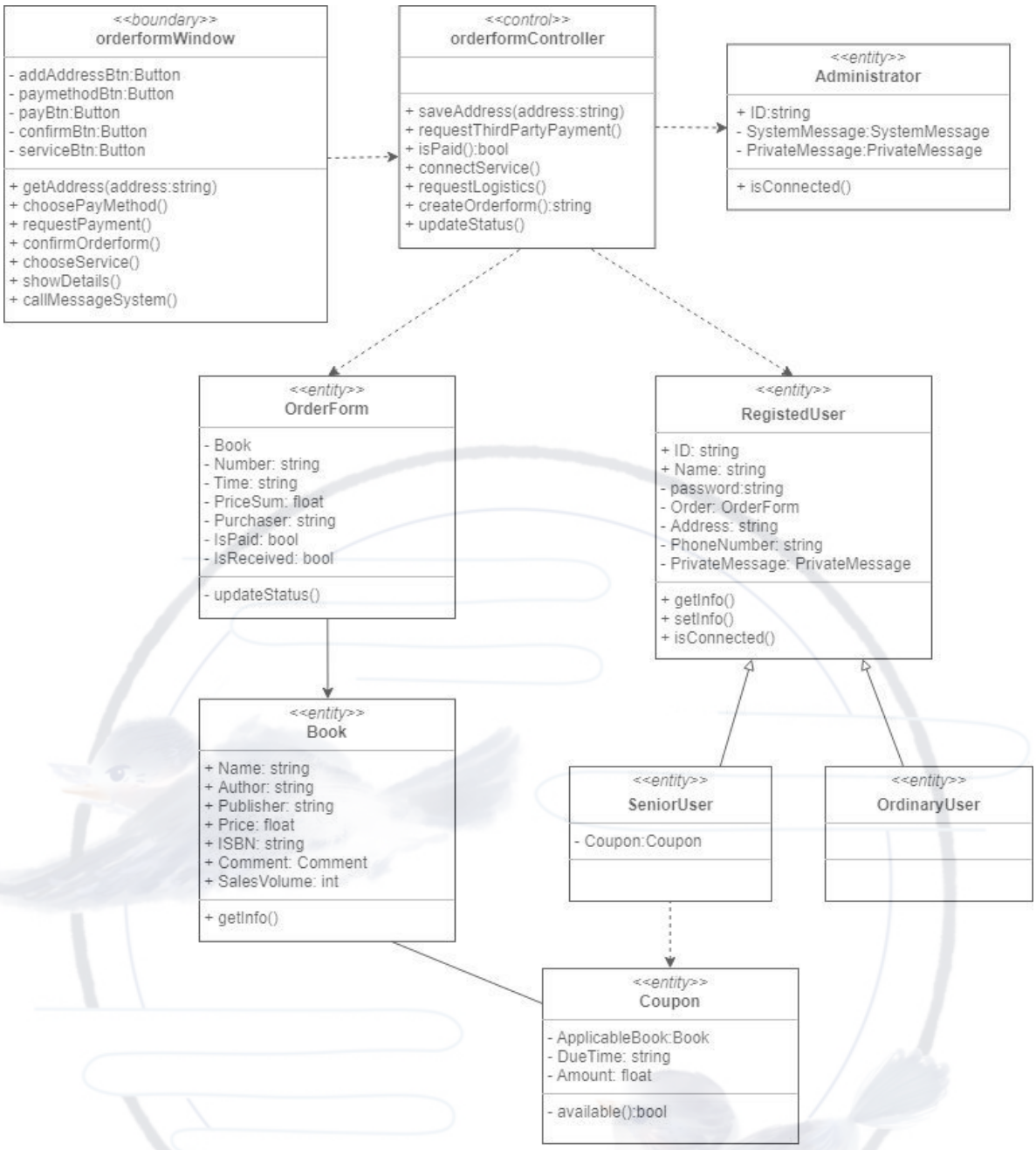


Sequence Diagram

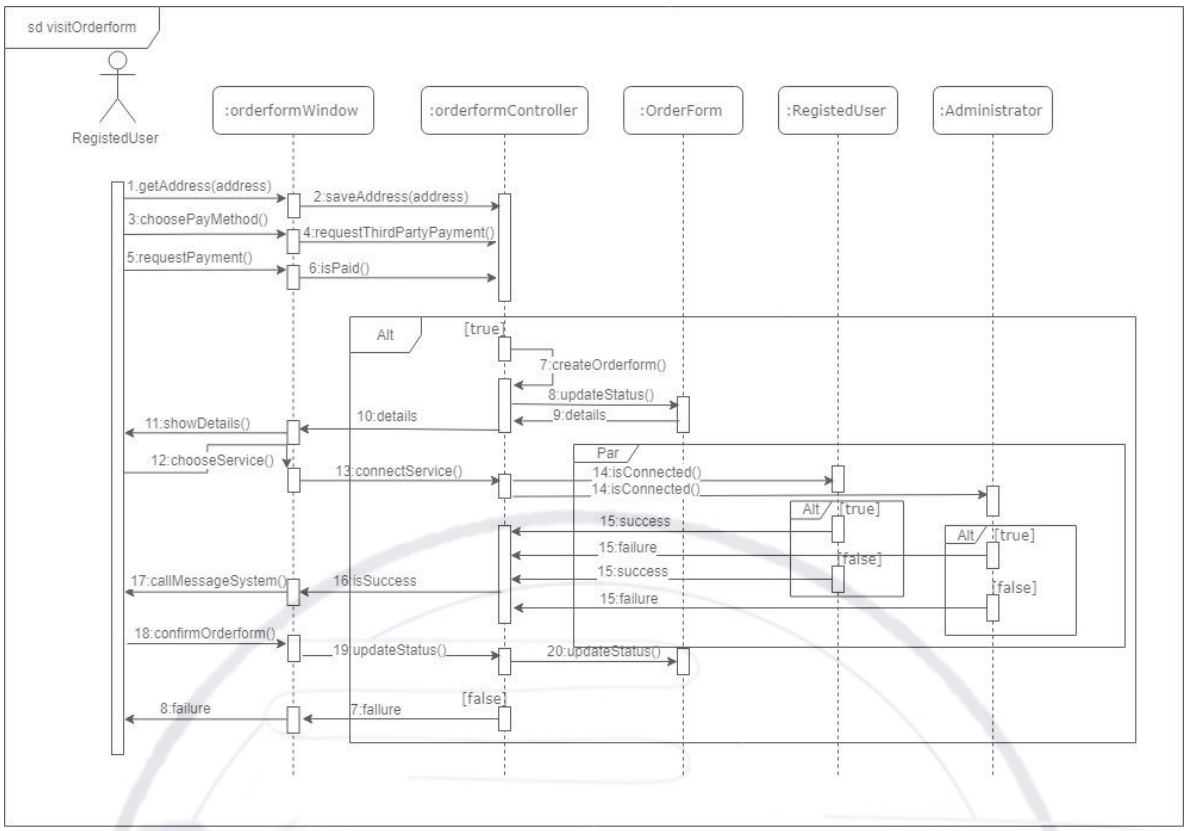


5.6 visitOrderForm

Class Diagram

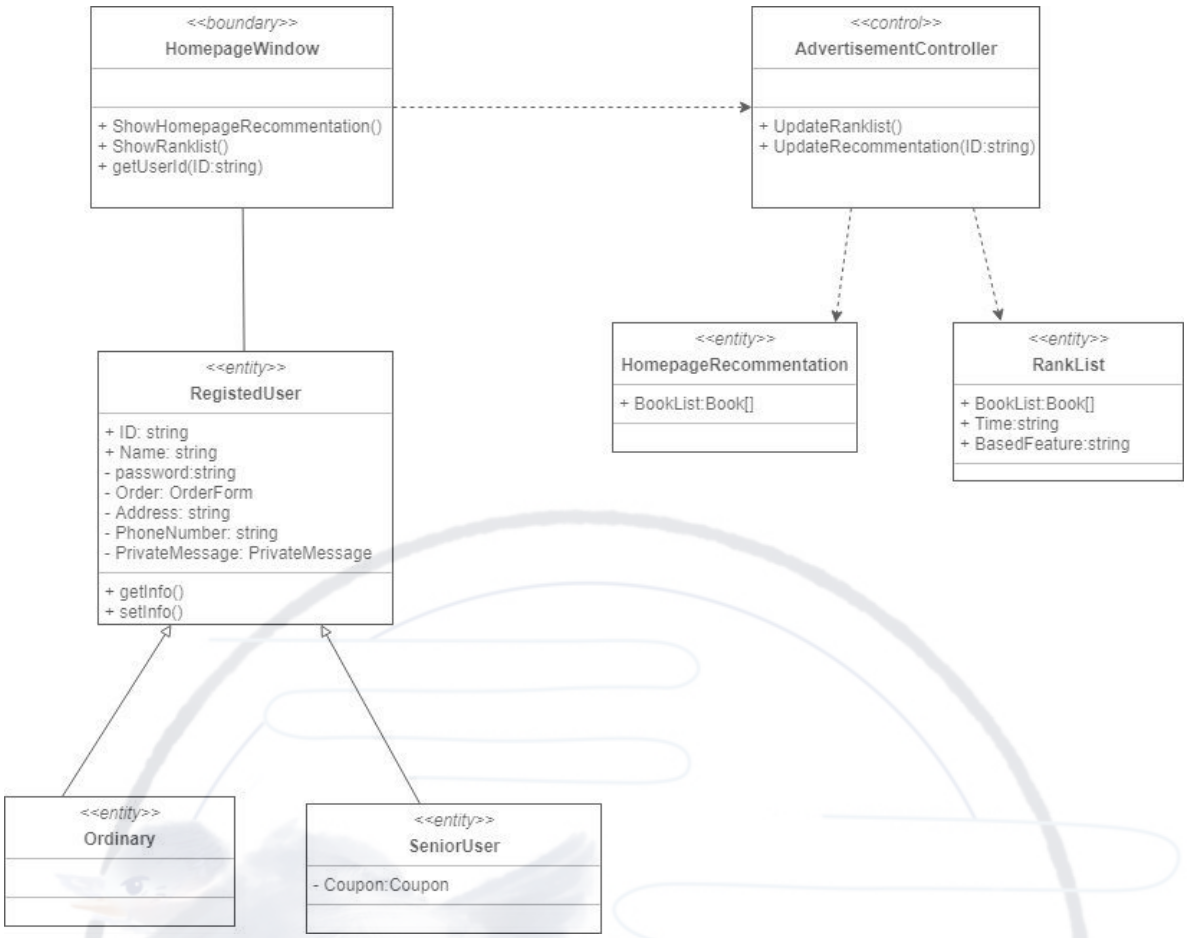


Sequence Diagram

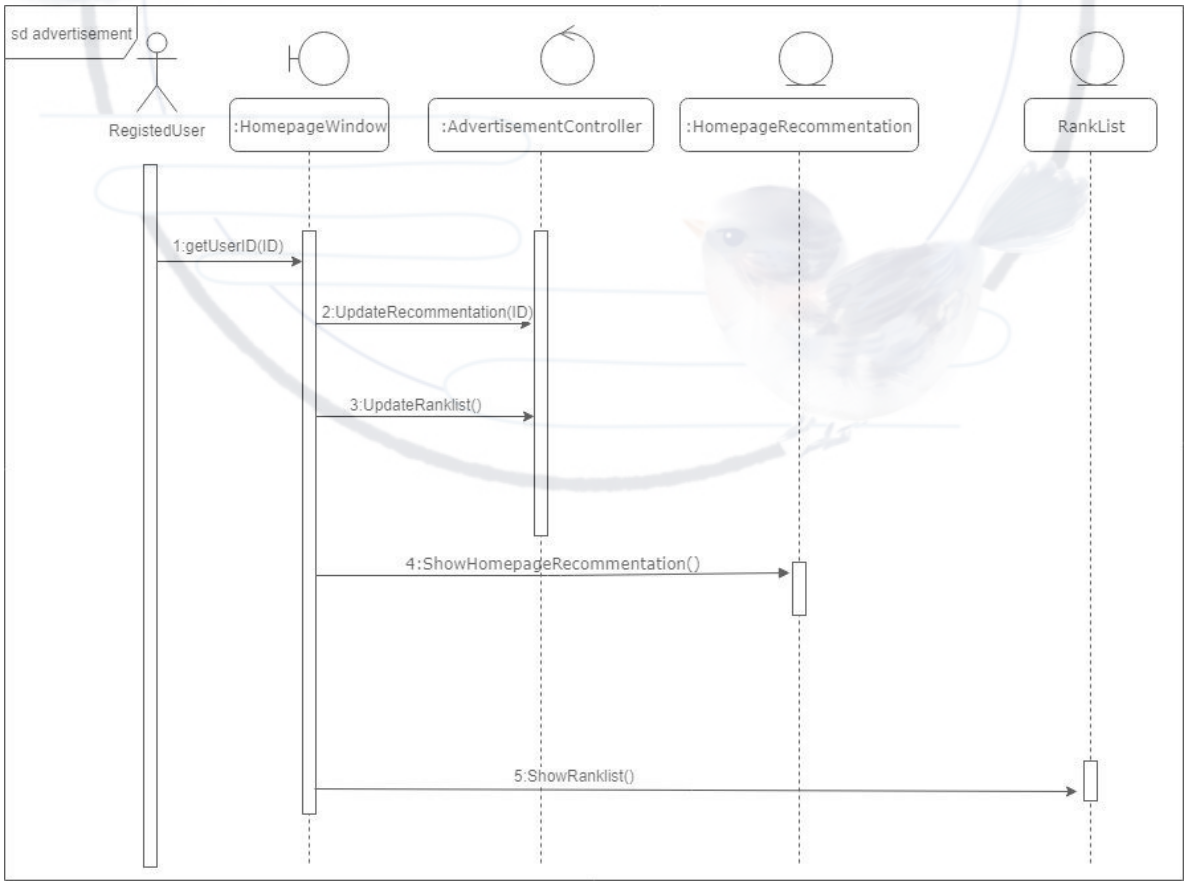


5.7 Advertise

Class Diagram

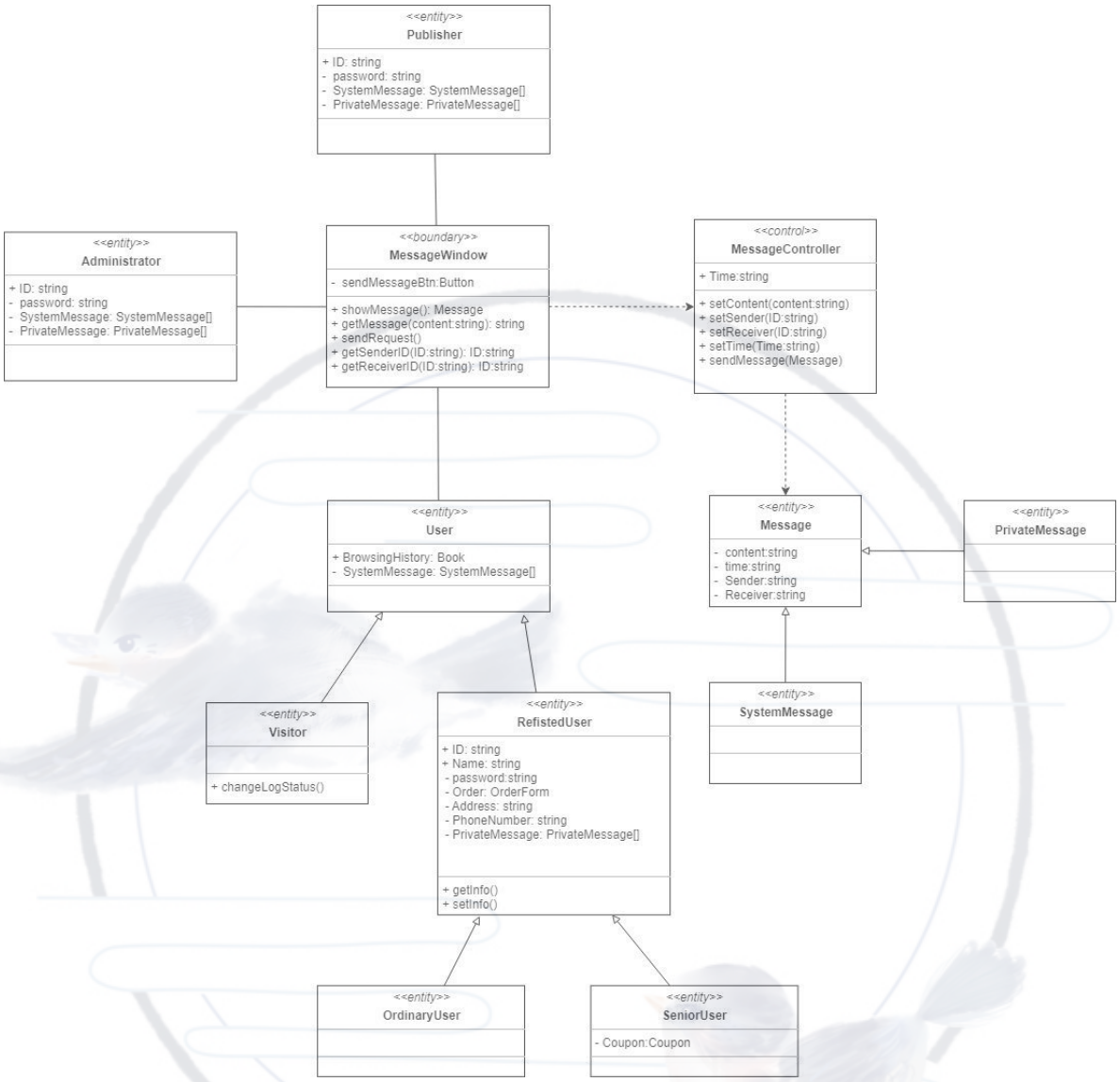


Sequence Diagram

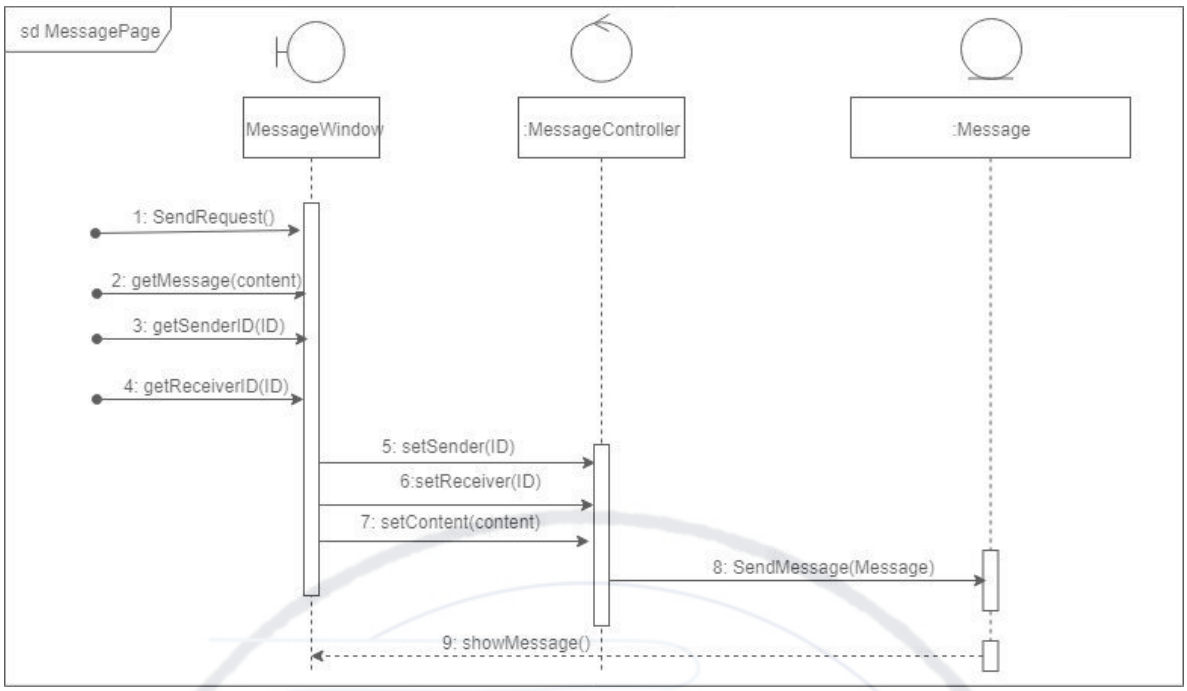


5.8 viewMessage

Class Diagram

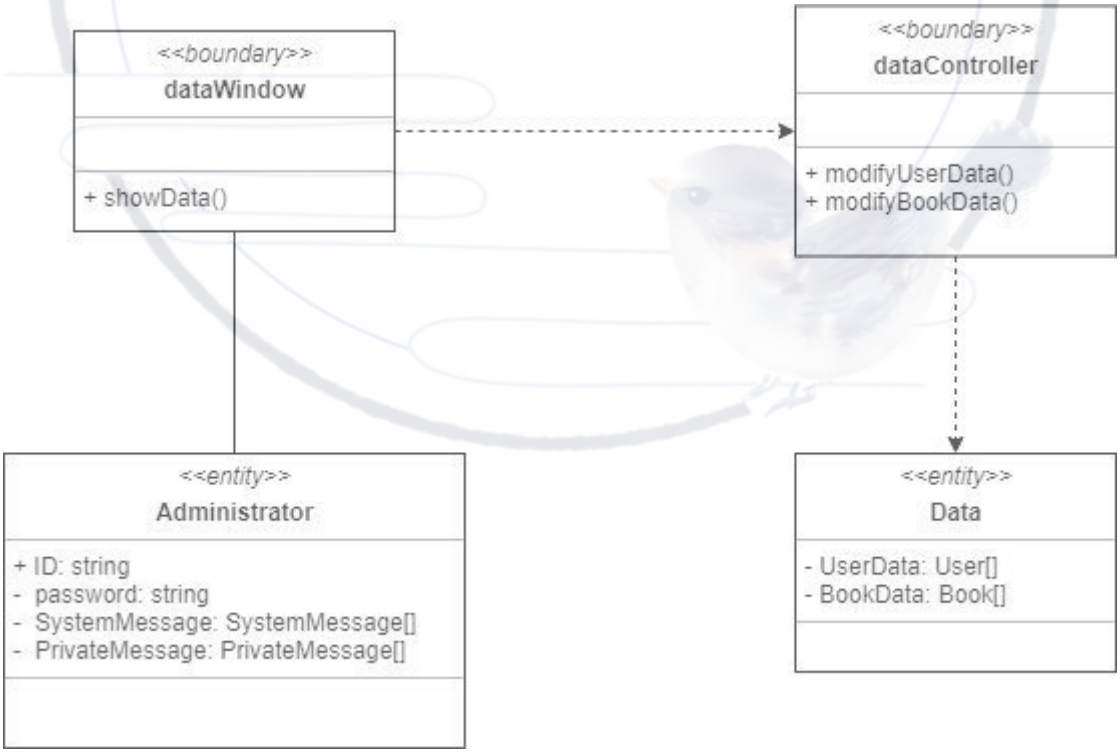


Sequence Diagram

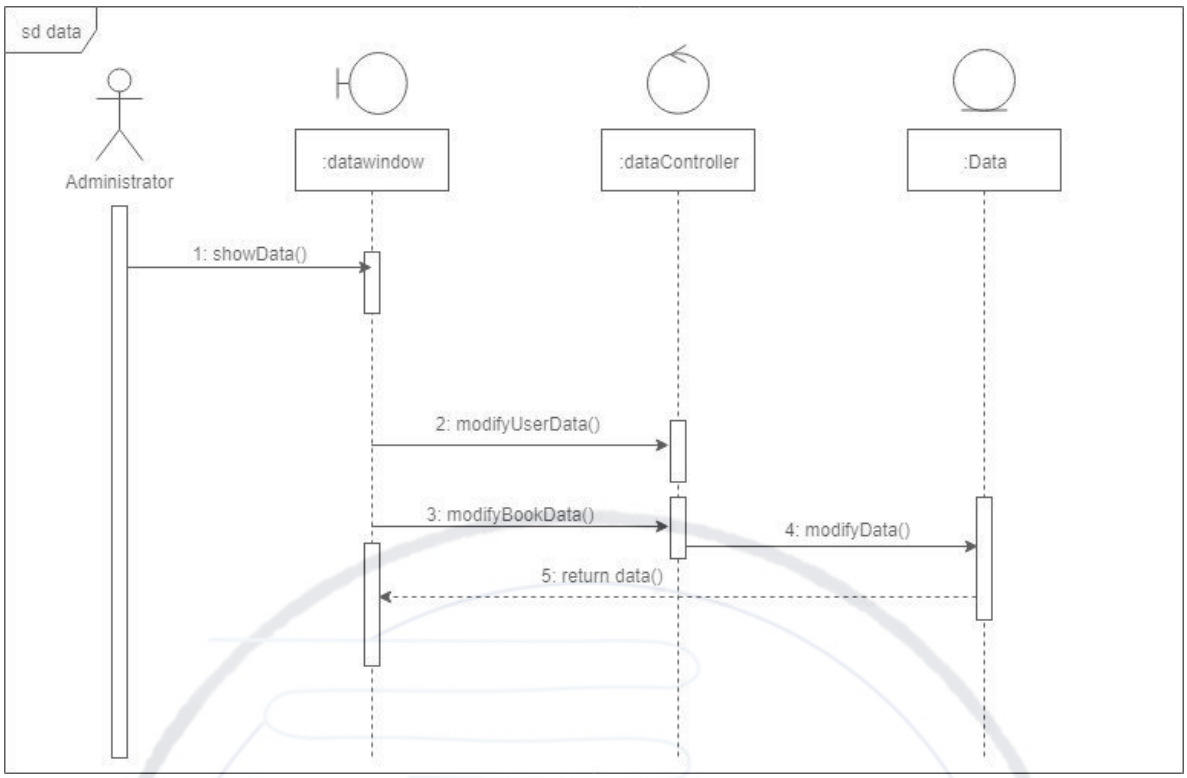


5.9 analysisData

Class Diagram



Sequence Diagram



6.Design Mechanism

6.1 Overview

For the critical design mechanisms, we set Login as interface specification. In the login use case, the communication between the RegisteredUser class, Visiter class and database is showed as mechanism Persistence. Customer can enter the relative information in the login page to make this.

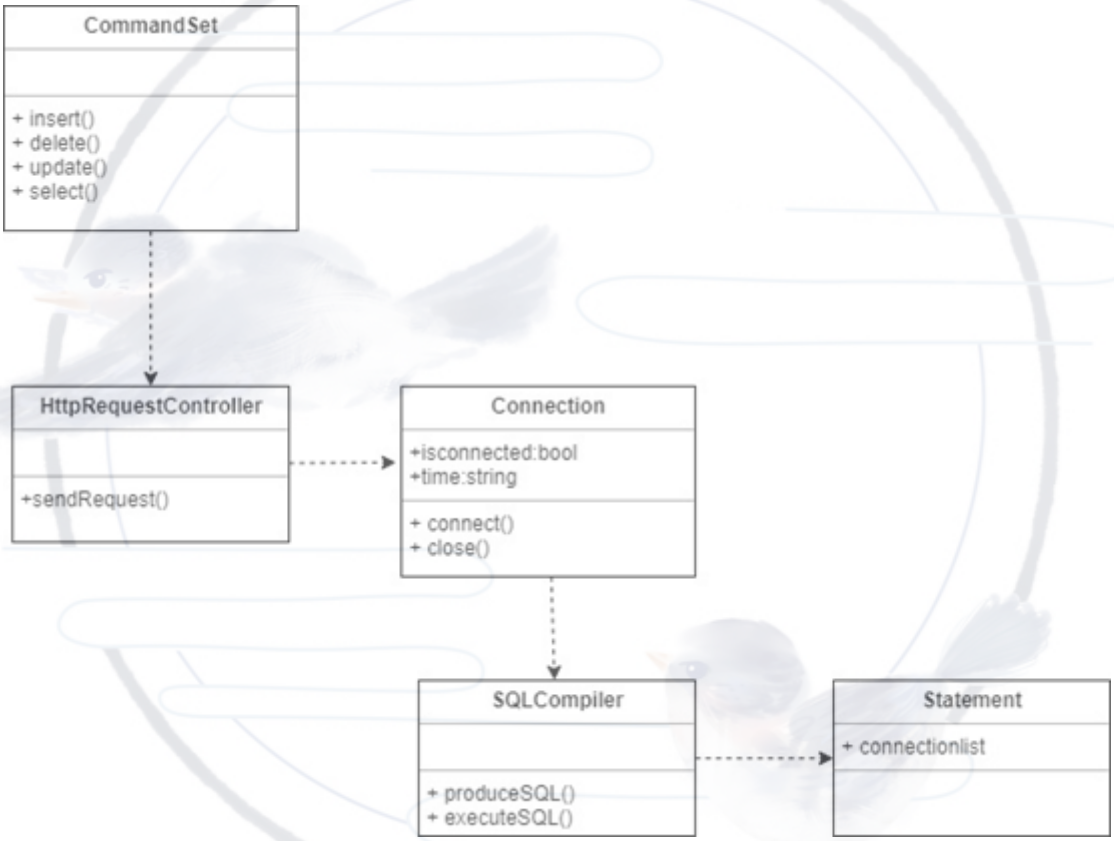
To realize this use case, Request from Customer APP frontend in presentation layer will be sent to the LoginController class of the User controller in controller layer, then go into our persistence mechanism arriving at certain database. Finally return the results layer-by-layer. Database entirely being separated from frontend and controller, which receives network request and builds or closes the connections, makes our design model perfectly persistent.

Another is the Information Exchange mechanism, using visitOrderform as interface specification. When user wants to have communications with administrators, request from APP and Web frontend will be distributed to the service layer through this communication.

To realize the use case, when a customer using wants to visit order form, after entered the address and chosen the third party payment method, the frontend first call a function to send a request to the orderformController, getting to the order controller interface int the controller layer. And if he would like to connect administrator, our client send request through User Access or Administrator Access to InformationController and connected to our server and OKHttp to achieve this.

6.2 Persistence

- Class Diagram



There are three parts of the above class diagram:

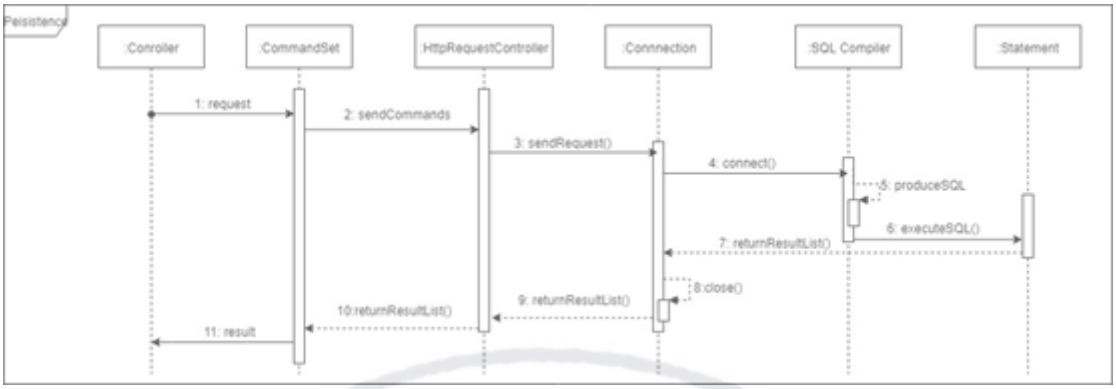
1. CommandSet can handle the requests from front end.
2. HttpRequestController and Connection can separate the database from the front end, moreover, they can make use of the functions named connect and close to keep the persistence of database.
3. SQLCompiler and Statement can get touch with real operation of database.

What the use of the classes are as follows:

1. CommandSet: Sort the front end's requests and process to the operation of database.
2. HttpRequestController: Send http requests and receive response.
3. Connection: Build and close connection with database.
4. SQLCompiler: Compile the given operation to SQL and execute it in the database.

5. Statement: Record the connections with database and contain some other attributes.

- Sequence Diagram

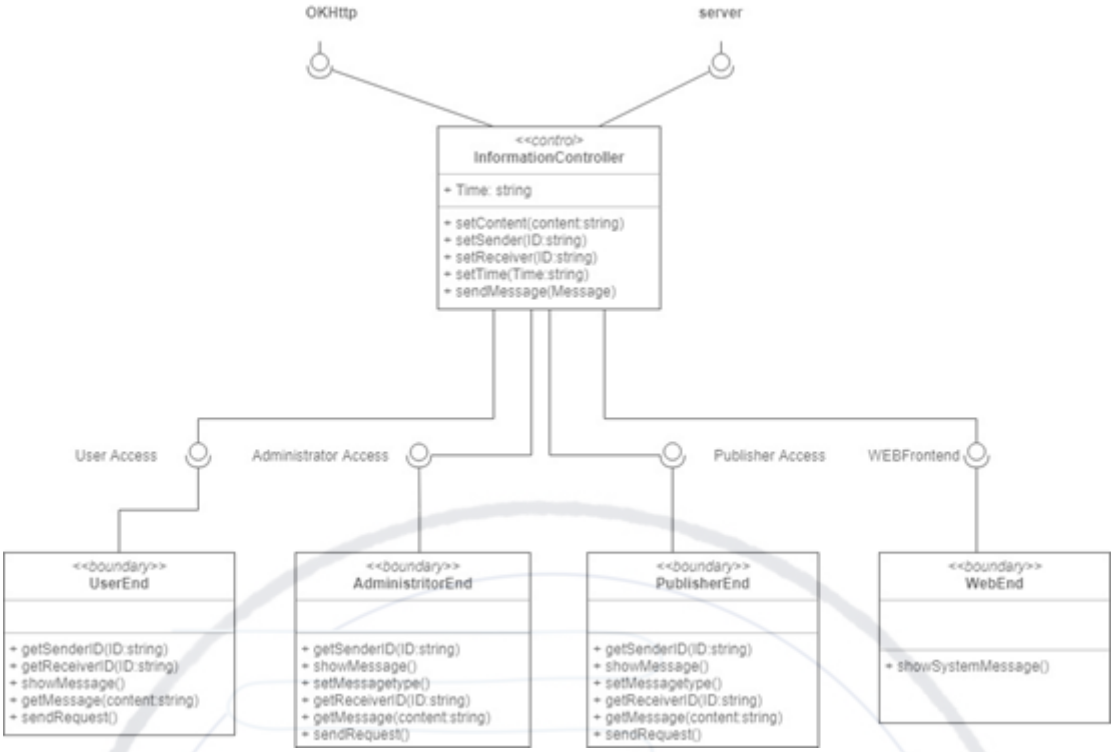


After receiving the request from front end, the CommandSet will send command to the HttpRequestController. The latter will then send request to the back end. The connection with the database will be built at same time of the production and execution of SQL. When the result list is returned from SQL compiler, the connection close simultaneously and return the result straightforwardly to the front end.

6.3 Information Exchange

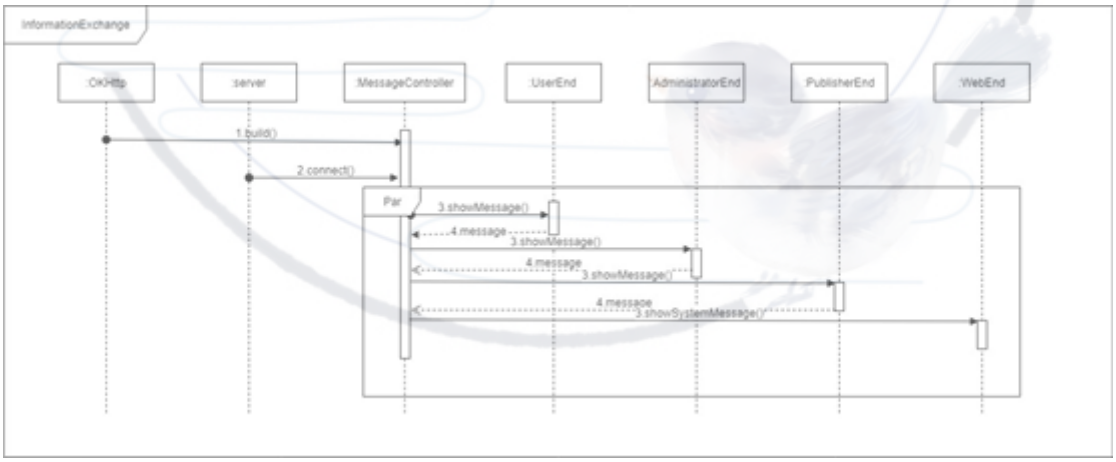
- Class Diagram

In the process of communication among User, Administrator, Publisher and Web page, we need the information exchange mechanism to handle this. Our View Message system provides User Access, Adminitrator Access and Publisher Access for our clients. And the Web page is used for system message, and it uses WEBFronted. There classes can use the API to contact with InformationController. And the InformationController is connected to Okhttp and our server.



- Sequence Diagram

Firstly, the MessageController of the web will build a client by calling the build method of the OKHttpClient. At the same time, it will try to connect the server as well. The MessageController will also get the connection and make it available to every end. The UserEnd, AdministratorEnd and PublisherEnd can be used by users, administrators, publishers in order to send messages to the message end, that is, the MessageController. It will process the messages and show them on the corresponding ends.



7.Realization

7.1 Interfaces

7.1.1 Interfaces Between Our System and External Systems

specify the 3rd party API as well as the map services in detail

Express Check Subsystem

If the customer want to know: Whether the books are shipped? Where the books I brought yesterday? How far is it from me? and so on. He or she can use this subsystem to know these information.

After order query, the query page will save the order's receiver's tracking number and adress in the cookie, so that the express check page can get it by reading the cookie and the query will get the express information by calling the SF API , other information for address especially the estimate time by calling the AMap API and interfaces in sequence:

- Get customer code and check code

Apply for the api interface: https://qiao.sf-express.com/index.html

We can get the *url*, *clientCode*, *checkword*

url：*http://bsp-oisp.sf-express.com/bsp-oisp/sfexpressService*

- XML message description

1. Request an XML message:

   The service attribute defines the "service name"; the Head element defines the "customer code"

```
<Request service="server_name">
<Head>customer code</Head>
<Body>message description XML</Body>
```

2. Response XML message:

   The value of the Head element is "OK" or "ERR"; OK means the transaction is successful, ERR means that the system or business is abnormal, the transaction fails; for the batch trading scenario, it can only be success/failure, no partial success/partial failure

```
<Response service="server_name">
<Head>OK|ERR</HEAD>
<Body>Correct corresponding data XML</Body>
<ERROR code="NNNN">Error details</ERROR>
```

- JSON request example

```
{
    "ShipperCode":"SF",
    "OrderCode":"SF20160808105520828l",
    "LogisticCode":"3100707578976",
    "PayType":"1",
    "ExpType":"1",
    "CustomerName":"",
    "CustomerPwd":"",
    "MonthCode":"",
    "IsNotice":"0",
    "Sender":{
        "Name":"1255760",
        "Tel":"",
        "Mobile":"13700000000",
        "ProvinceName":"广东省",
        "CityName":"深圳市",
        "ExpAreaName":"福田区",
        "Address":"测试地址"
    },
    "Receiver":{
        "Name":"1255760",
        "Tel":"",
        "Mobile":"13800000000",
        "ProvinceName":"广东省",
        "CityName":"深圳市",
        "ExpAreaName":"龙华新区",
        "Address":"测试地址2"
    },
    "Commodity":[
        {
            "GoodsName":"book-name"
        }
    ]
}
```

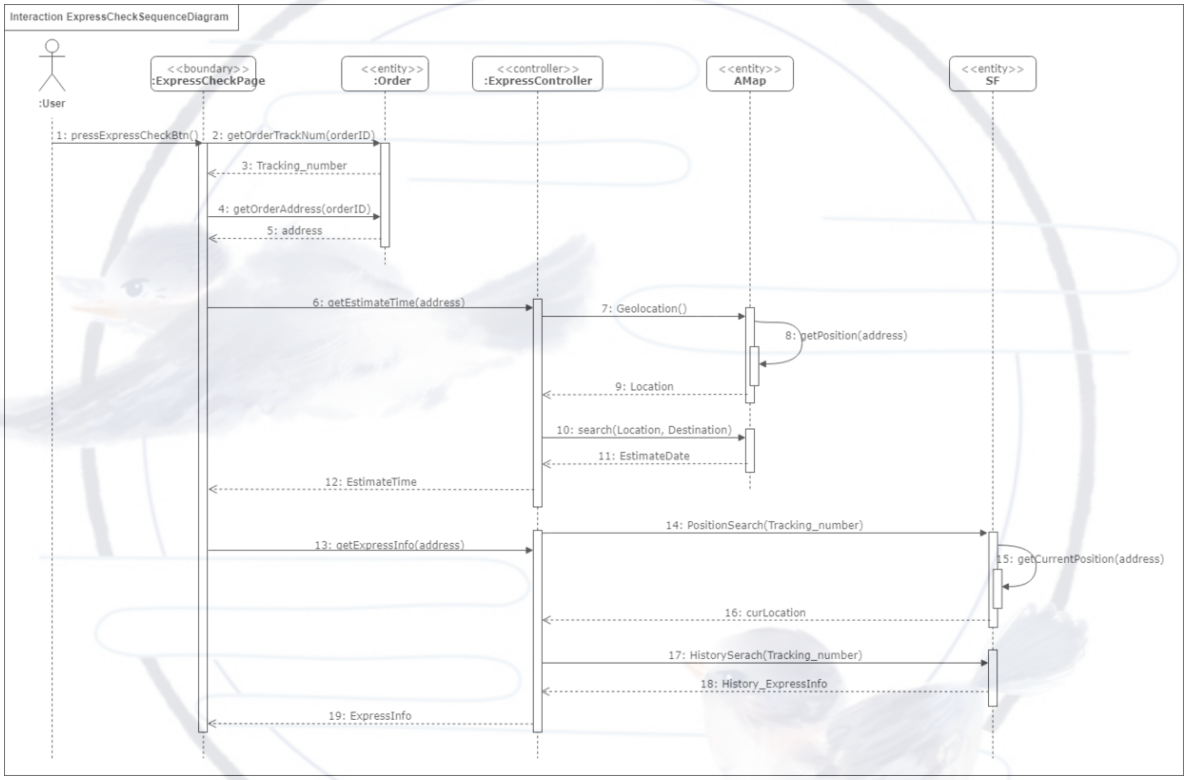
- JSON return example

```
{
    "EBusinessID": "1151847",
    "UpdateTime": "2016-08-09 16:42:38",
    "Success": true,
    "Reason": ""
    "EstimatedDeliveryTime":"2016-8-12"
}
```

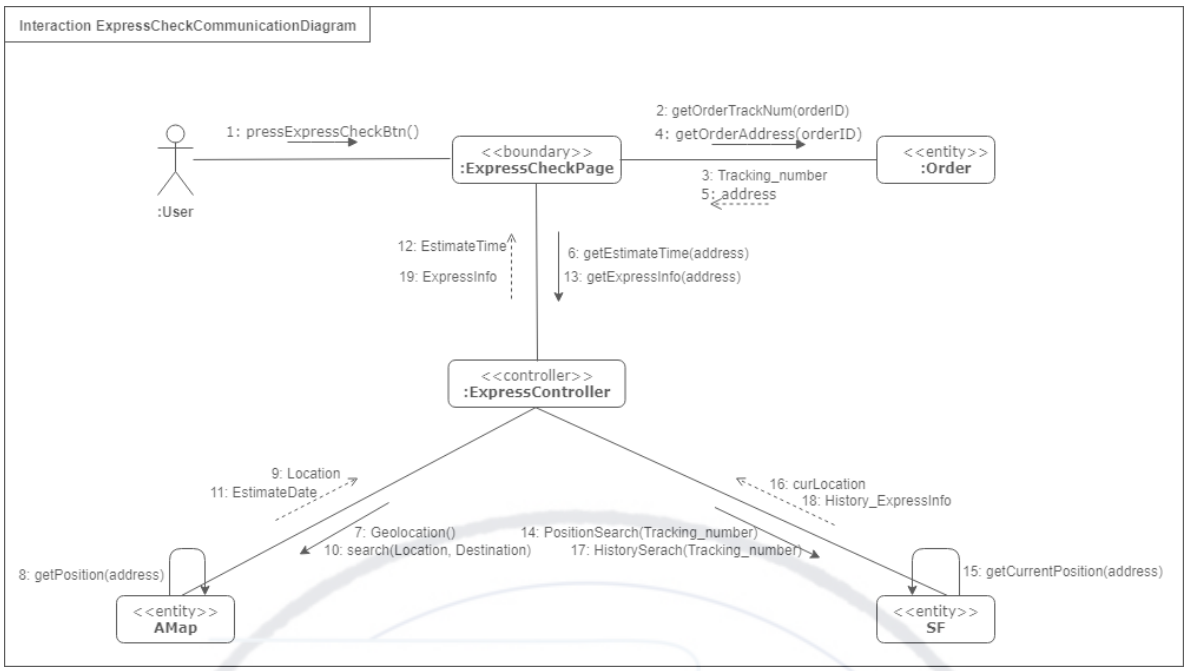
- API interface call

```
requestXml = getOrderServiceRequestXml(params);
        methodName = "orderSerivce";
requestXml = getOrderSearchServiceRequestXml(params);
        methodName = "orderSearchService";
requestXml = getRouteServiceRequestXml(params);
        methodName = "routeService";
//post request
SfExpressResponse response =
(SfExpressResponse)XMLUtil.convertToObject(SfExpressResponse.class,re
sultStr);
```

Sequence Diagram



Communication Diagram



7.1.2 Order Query Subsystem and its Interfaces Specification

specify the data and database in detail

Order Query Subsystem

- If the customer want to query his orders, he can take the following steps:

1. Input the order id and press the query button in the order information Page.
2. With the **Order ID** attached, the page will then send a request to the **Order Controller**.
3. Once the controller get the request, it will got to the **Order Database** to get the information and then send it back to the Page and the Page will show it to the customer.
4. If the order id doesn't exist, the controller will return failure information.

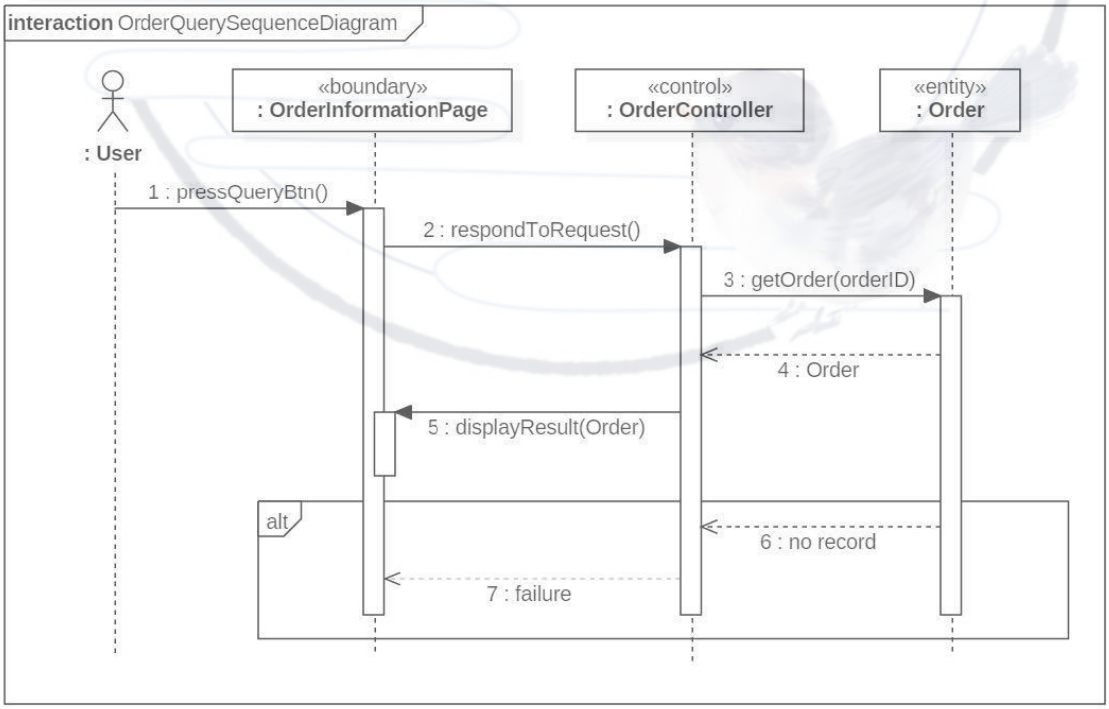
- The interfaces we designed are mainly showed as follows:
  - `respondToRequest():bool` This interface is implemented in the order query boundary class. It is invoked after the user press the query button. Then it will call function to execute the query.
  - `getOrder(orderID: string): Order` This interface is used to query order by order id. It will receive the order id and then send a HTTP request with the order id to the back-end API. the API will integrate the order id into SQL and execute the order query in the Order database and return the query result as a json object to this function. And the function will analyses it and return the result.

```
Json Format of Order:
{
“orderId”: “orderIDNumber”
“sender”: “senderName”
“receiver”: “receiverName”
“signStatus”: true/false
“deliveryStatus”: true/false
“payStatus”: true/false
“receiverAddress”: address object
“currentLocation”: address object
“senderTEL”: “senderTelephone”
“receiverTEL”: “receiverTelephone”
}
```

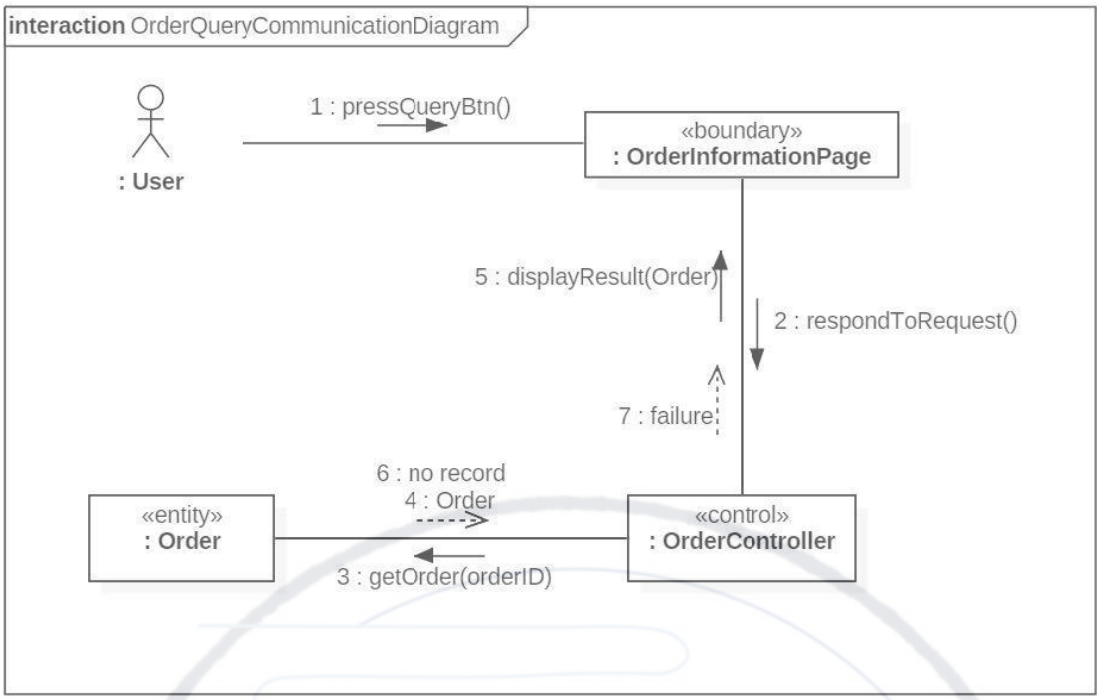
- `displayResult(order: Order):bool`

  This interface is implemented in the order query boundary class. This interface will receive the result of query and display it fitting the webpage design. If the query result is empty, it will display a message that the order id doesn’t exist in the database.

Sequence Diagram



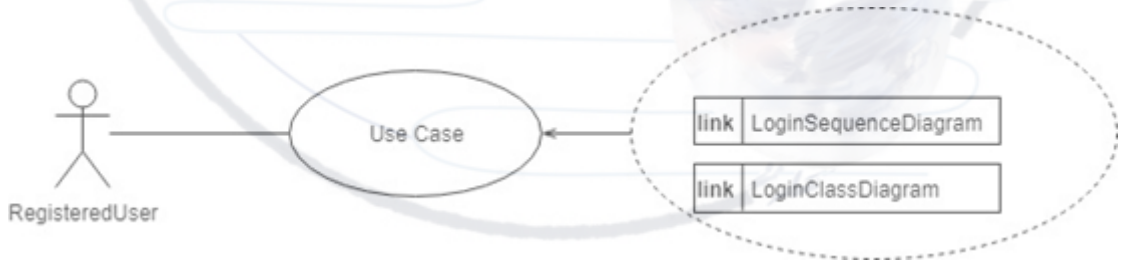
Communication Diagram



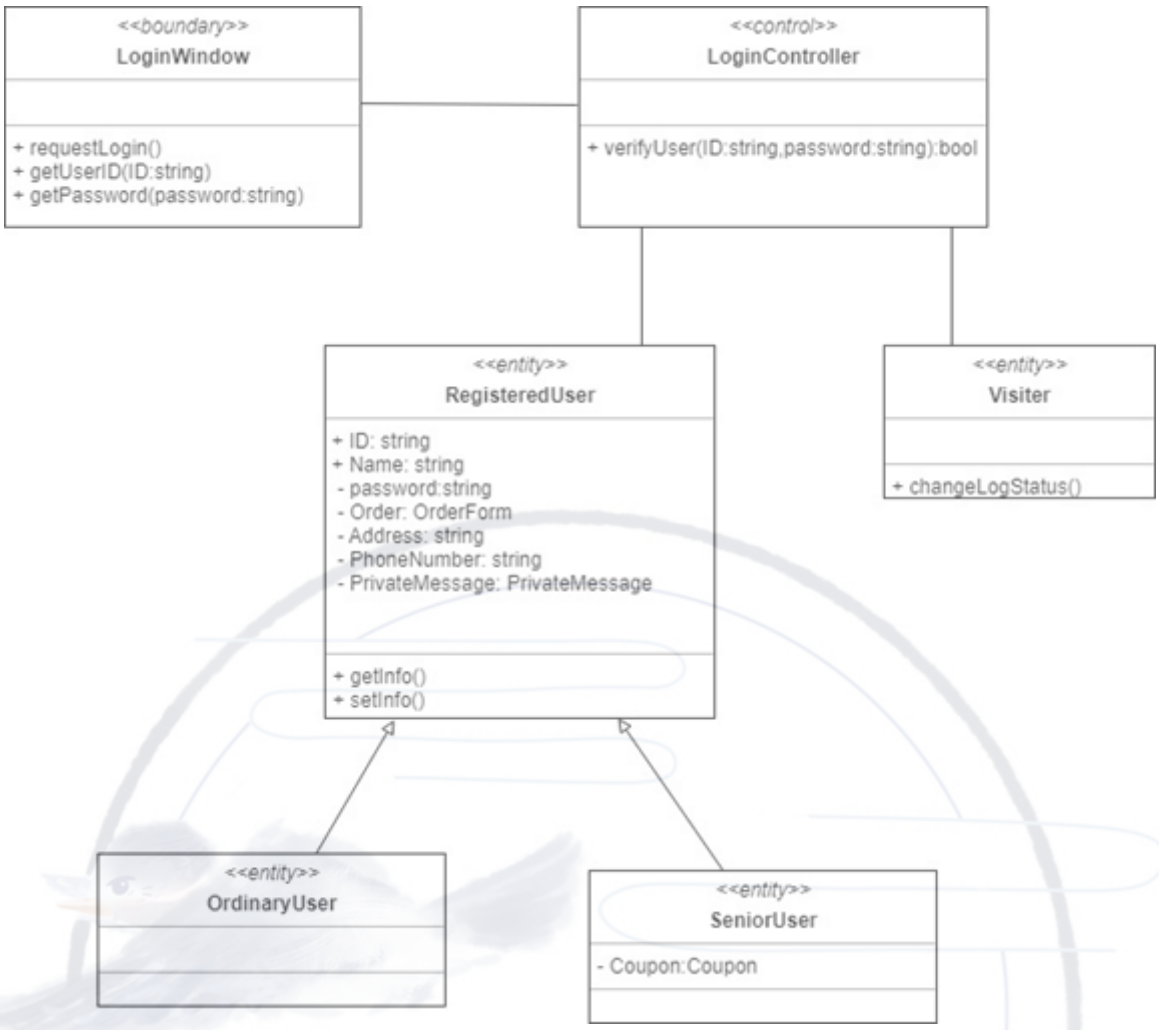
7.2 Use Case Realization

7.2.1 Login

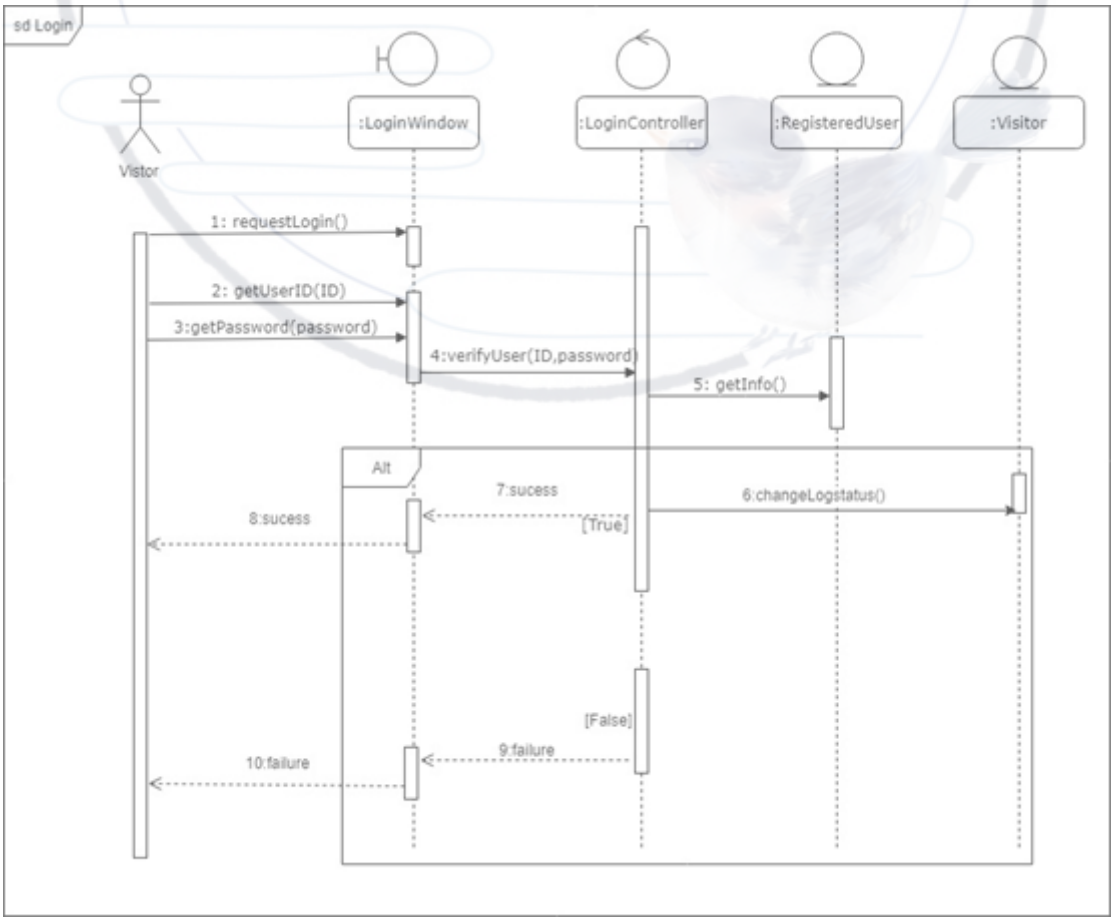
- Login Realization



- Class Diagram



- Sequence Diagram



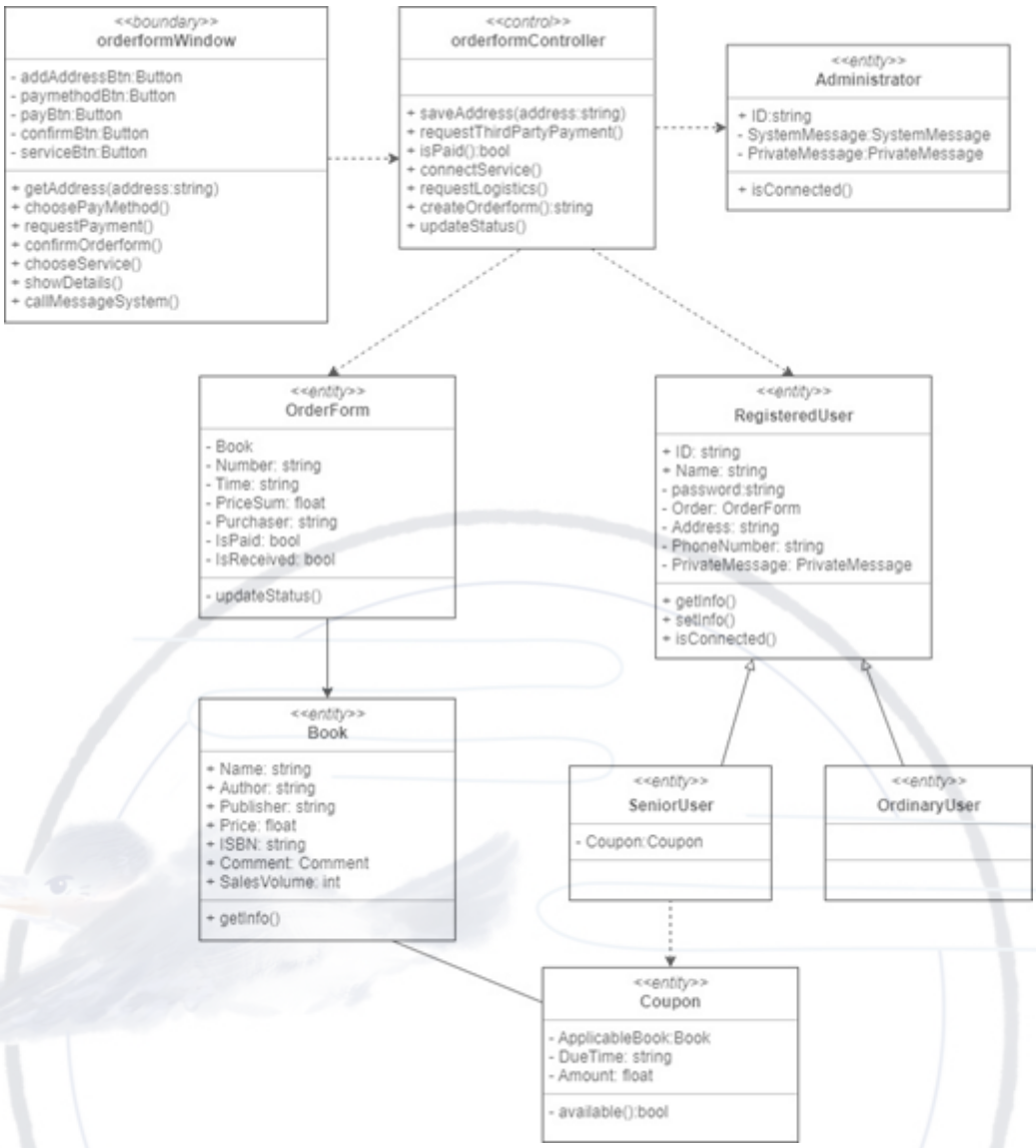
In the Login use case, the communiaction between the RegisteredUser class, Visitor class and database is showed as mechanism Persistence.

If the Visitor wants to log in, he should first request to log in and then put his ID and password in the LoginWindow, and then the window will send these information to the LoginController. The LoginController will check the database of users and verify the information is true or not. If it is true, the visitor can then change his logstatus and turn into a registed user. But if the information is false, the LoginWindow will send a failure message.

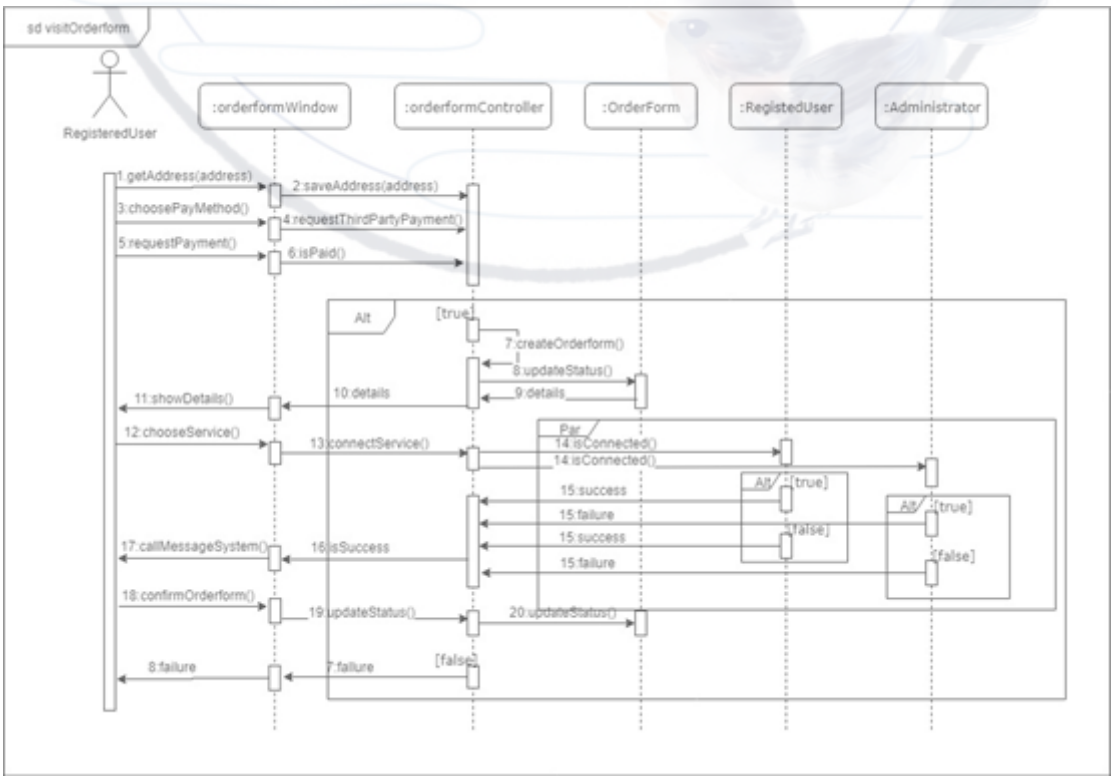
7.2.2 Visit Order Form

- VisitOrderform Realization

- Class Diagram



- Sequence Diagram



In the createOrderform use case, the communication between the OrderForm class and database is shown as mechanism Persistence. And that between RegisteredUser class and Administrator class is designed as mechanism Information Exchange.

If the registeredUser wants to confirm the order form, he or she will have to enter the address and choose third party payment method, which along with the order form fundamental information are stored perpetually. After this, the window will show the details of order form.

The registeredUser can also choose after-sale service to communicate with administrator, which requests the message system.

8.Prototyping

8.1 Description

In the prototyping part, we implement three functions: Register, Login and Search mainly via vue and node.js in localhost webserver. Both of the two javascript frameworks are very popular recently.

As for vue, it is a front end javascript framework which is highly efficient and easy to maintain. By using declarative rendering, component system and vue-router, it allows developer to focus on the data and logic instead of DOM operation. Ad it is responsive, it can also help releasing brower's resources.

As for node.js, it is a javascript runtime built on Chrome's V8 engine. It uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.

As it is a demo that contains few data, we choose mysql as our database management system

8.2 Register Example

8.2.1 Interact with Database

```
route.post('/reg', (req, res) => {

    let mObj = {};
    for (let obj in req.body) {
        mObj = JSON.parse(obj);
    }
    let regName = mObj.regName;
    let regPasswd = mObj.regPasswd;
    regPasswd = common.md5(regPasswd + common.MD5_SUFFXIE);
    const insUserInfo = `INSERT INTO user(user_name,login_password,user_number) VALUES('${regName}','${regPasswd}','${regName}')`;
    delReg(insUserInfo, res);
});
/*
 *deal user register
 */
function delReg(insUserInfo, res) {
    db.query(insUserInfo, (err) => {
        if (err) {
            console.error(err);
            res.send({ 'msg': '服务器出错', 'status': 0 }).end();
        } else {
            res.send({ 'msg': '注册成功', 'status': 1 }).end();
        }
    })
```

this snippets define how the front end interact with data base via '/reg'. Fisrt, it will recieve two parameters from json named regName and regPasswd. Second, in order to secure the information while transfering, we will use md5 method to encrypt regPasswd. Next, it will insert it into database and find if the user is successfully registered.

8.2.2 Front End

```
<template>
  <div class="m_r">
    <header class="top_bar">
        <a onclick="window.history.go(-1)" class="icon_back"></a>
        <h3 class="cartname">注册优木账号</h3>
    </header>
    <main class="user_login_box">
        <div class="login_dialog">
            <div class="_username">
                <input type="text" name="regname" placeholder="用户名/邮箱/手机号" class="user_input" v-model="regname">
            </div>
            <div class="_username u_passwd">
                <input type="password" name="regpasswd" placeholder="请输入密码" class="user_input" v-model="regpasswd">
            </div>
            <div class="_username u_passwd">
                <input type="password" name="regpasswd_ag" placeholder="请再次输入密码" class="user_input" v-model="regpasswd_ag">
            </div>
            <div class="login_box">
                <a @click="goSearch()" class="btn_login">注册</a>
            </div>
        </div>
    </main>
  </div>
</template>
```

This part define the out look of the page together with vue template

8.2 3 Click Method

```
methods:{
  goSearch(){
    let _this = this;
    if(_this.regname ==''){
      alert('请输入手机号');
    }else if(_this.regpasswd == '' || _this.regpasswd_ag == ''
      alert('请输入密码');
    }else if(_this.regpasswd!==_this.regpasswd_ag){
      alert('两次输入的密码不一致');
    }else{
      _this.$http.post('/reg',{
        regName:_this.regname,
        regPasswd:_this.regpasswd
    }).then((res)=>{
      if(res.status == 200){
        _this.regInfo = res.data;
        if(_this.regInfo.status == 1){
          //reg success, go to this login page
          window.history.go(-1);
        }else{
          alert('注册失败');
        }
      }else{
        alert('出现错误');
      }
      console.log(res);
    },(err)=>{
      console.log(err);
    });
    }

  }
}
```

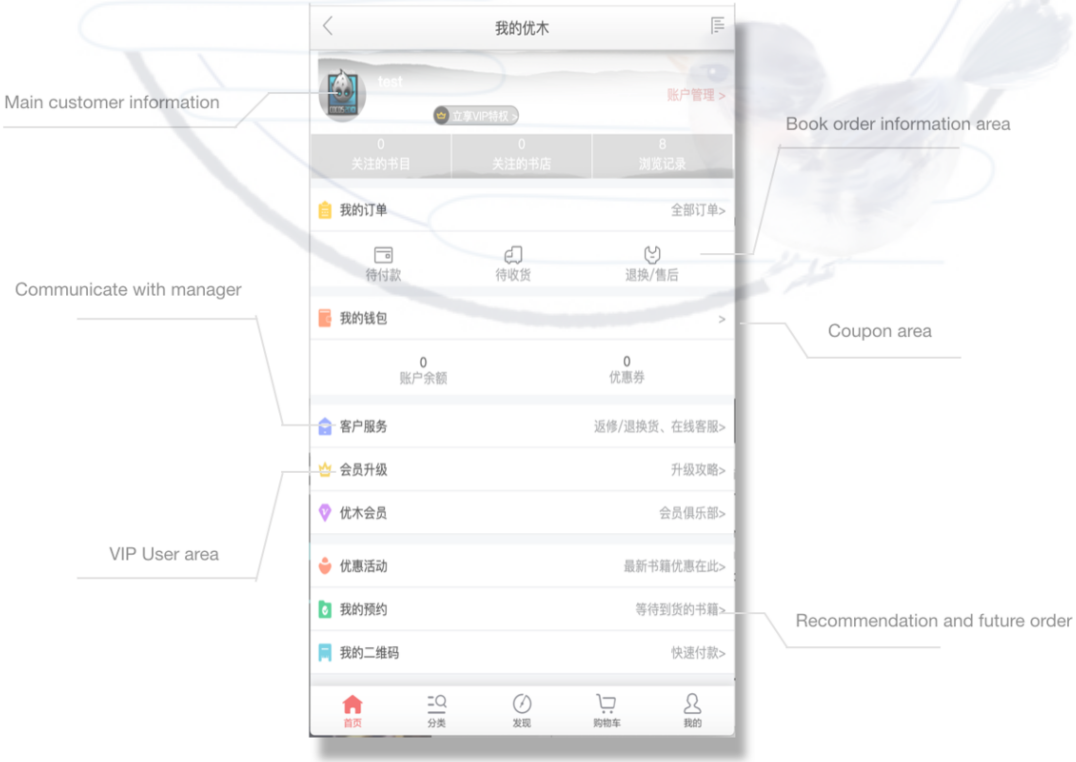
This part shows how to handle click method as a whole. It will judge if the input is valid, them send the callback to view layer

Through the functions and frameworks showed below, we can allow a user to register in a mobile web page, send the response and data to different layers and show the feedback to user.

8.3 Initial Snapshot

8.3.1 Mobile Login

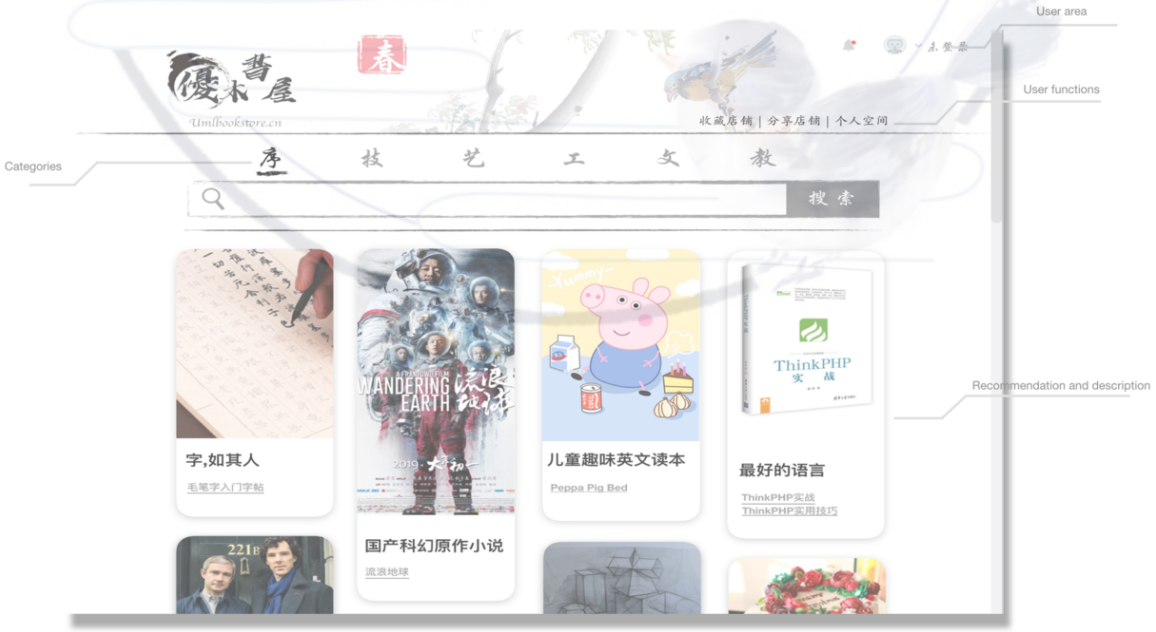
8.3.2 Mobile Personal Page



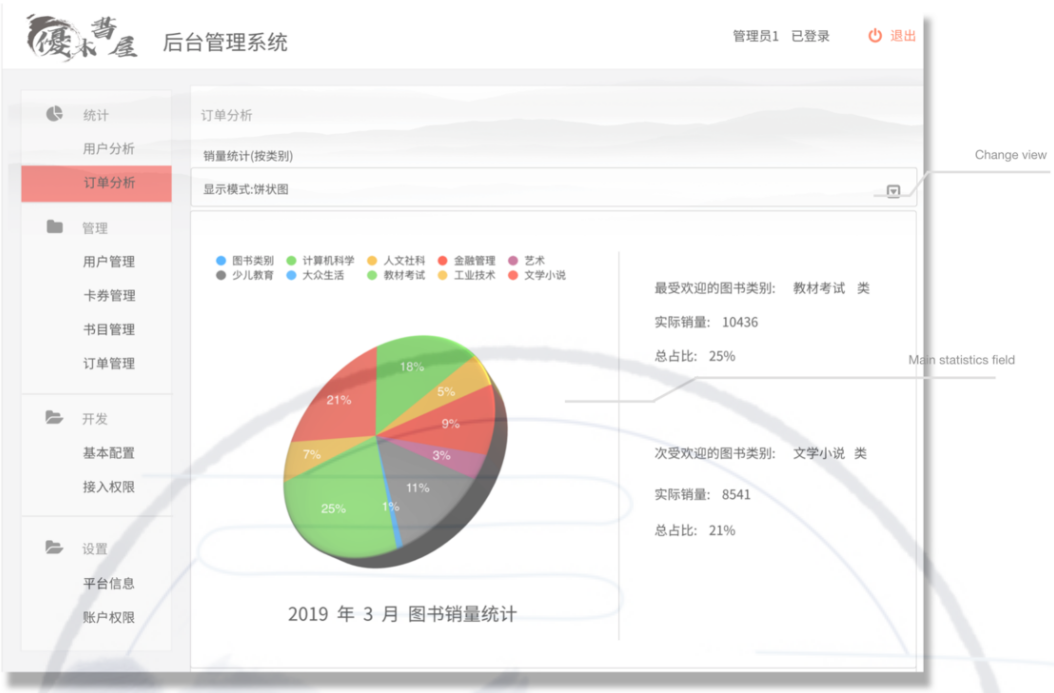
8.3.3 Moblie Shopping Cart

8.3.4 Moblie Payment

8.3.5 PC Main Page



8.3.6 Background statistics



8.3.7 Background Add Book

9.Quality

9.1 Architecture Style

The basic principle we use in architectural design is layered architecture. A layered architecture is the most common architecture and is also known as an n-tier architecture. Many companies and companies have used this architecture in their projects for many years, and it has become almost a standard norm.

So in our architecture there are four layers: Presentation Layer, Services Layer, Controller Layer and Data Layer. Layering does not need to know how other layers do it. For example, the controller layer does not need to know how the data layer queries the database. On the contrary, when the controller layer calls the specific method of the data layer, it only needs to pay attention to whether part of the data or all the data is needed. This is what we call the separation of concerns, which is a very powerful feature to let each layer is responsible for its responsibilities.

9.2 Design patterns

To achieve our web platform, we mainly use Django and Vue.js, so in this part we will mainly talk about the design patterns of this two frameworks.

As for Django, it follows the model-view-template(MVT) pattern, which can be treated as a special form of MVC. It consists if an object-relational mapper(ORM) that mediates between data models, and the workflow if MVT pattern is as follows:

1)Web Server(middleware) receives a HTTP request

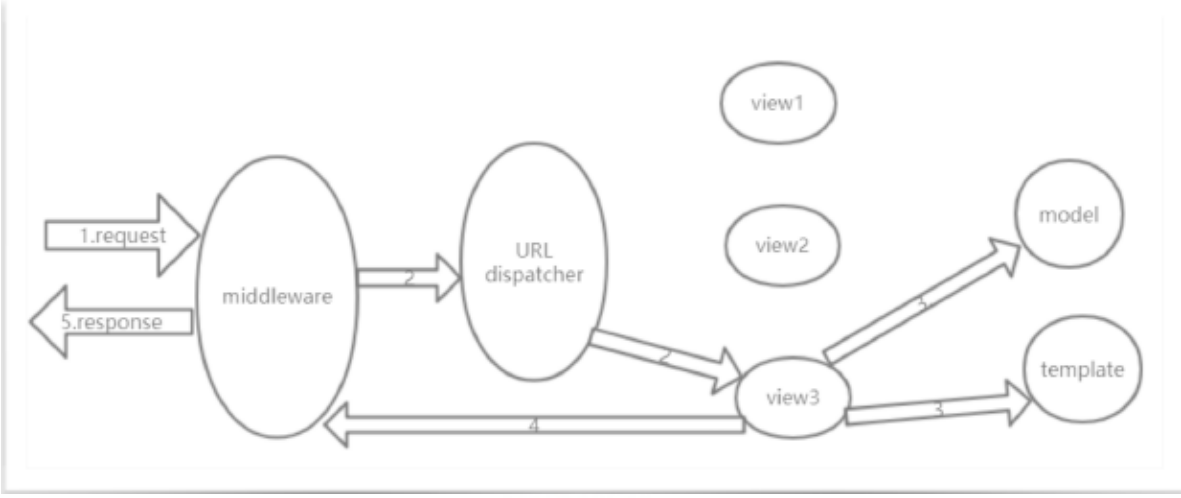
2)URL dispatcher dispatches the request to specific views

3)View functions call corresponding models to access data and invoke the corresponding template to present the page to user

4)View functions finishes its task and return the HTTP response to middleware

5)Web Server(middleware) sends the response back to client

As for Vue.js, it is a progressive framework for building user interfaces. Unlike other large frameworks, Vue is designed from the bottom up. The core library of Vue focuses only on the view layer, making it easy to get started and easy to integrate with third-party libraries or existing projects.



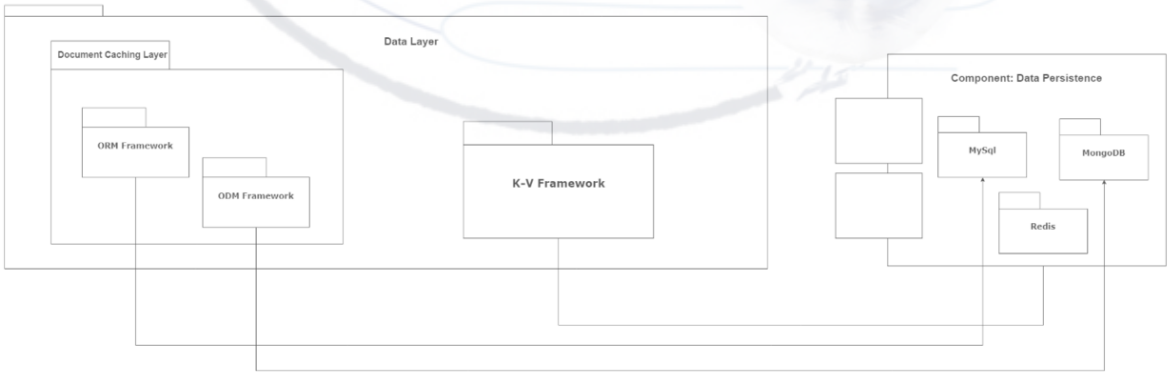
9.3 Critical decisions

6.1 Data Storage

To choose a suitable storage method for our project, we first analyze the data types that we may encounter: data that is highly relational and has great demand of ACID, data that is less relational and data that is highly performance sensitive like cache and sessions. After discussion, we decide to use MySQL, MongoDB and Redis together to store the three kinds of data above.

For MongoDB, the system uses ODM(Object Document Mapping)frameworks to map objects to MongoDB queries and for MySQL, it uses ORM(Object Relational Mapping)frameworks to map objects to SQL queries.

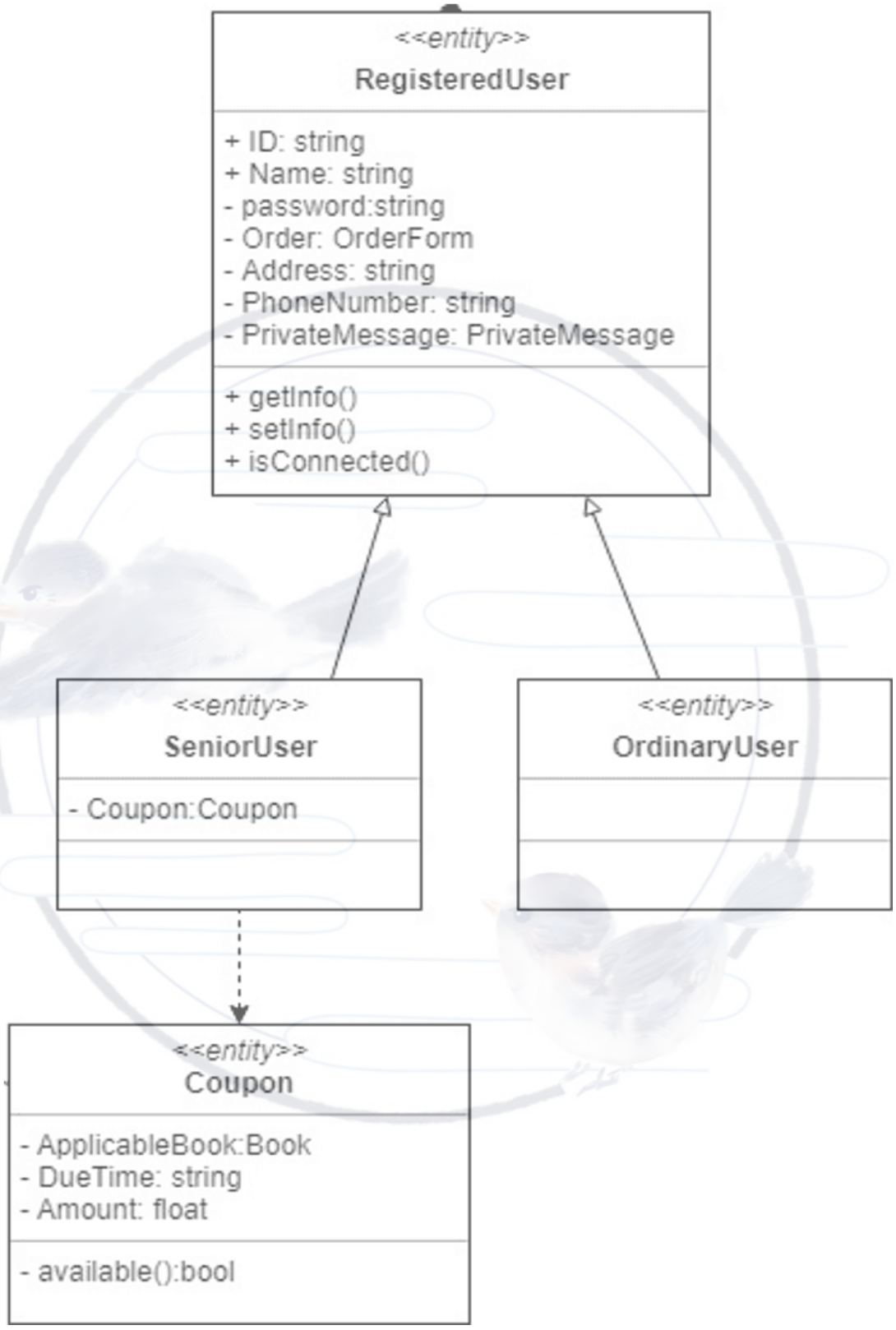
Both the two methods and database management systems are indispensable and can make the queries and update more efficient.



6.2 Senior User

When trying to define all the classes that contained in our project, we create a class called senior user derived from normal user. This class has the privilege to possess coupons and can interactive with the manager.

As we are a private bookstore, we do not need to create various kinds of coupons and create a special kind of user can help reduce the data that needs to be stored and transferred.



10.Improvement

10.1 Limitation

10.1.1 User experience: Lack of special handling mechanism for ordinary users

In our system we distinguish between ordinary users and senior users by issuing coupons or not, which means ordinary users can never get coupons unless they become senior users by some ways. This mechanism is bound to be conducive to senior users, while that keeps the senior users being absorbed in our online book store. But for the ordinary users, the book store is not that attractive. The lack of special handling mechanism for ordinary users may lead to the loss of new users who can not buy books firstly in a lower price.

10.1.2 Lower level design: A rough design of database.

During our analysis and design work, we did not design a relational database systematically but only made a rough design of database in several times. This mirrors the lack of design in data layer and model layer. Without a design of the whole database structure, we only considered the functions of each entity class, which led to some repetition and some omissions of methods among entity classes. This rough design really reduced our efficiency because we unified the functions and operations among different data models more than once. And we have learned from this limitation that the systematical design of the whole structure should always come first.

10.2 Potential Improvements

In the third assignment, we designed a primary Architecture and we made some improvement of it in this assignment.

10.2.1 Design asynchronous version APIs on the Presentation Layer

Introduce asynchronous mechanism on the web frontend. The updated web frontend uses the asynchronous design version APIs, such as AJAX and jQuery.

For example, the working flow for a specific procedure:

1) At first, the customer wants to get contract list from the customer frontend.

2) Then the customerFrontend will send a GET or POST request to the contractController.

3) The controller will get the contract list from the database of contract.

4) The controller returns the serialize of the contract list to the CustomerFrontend.

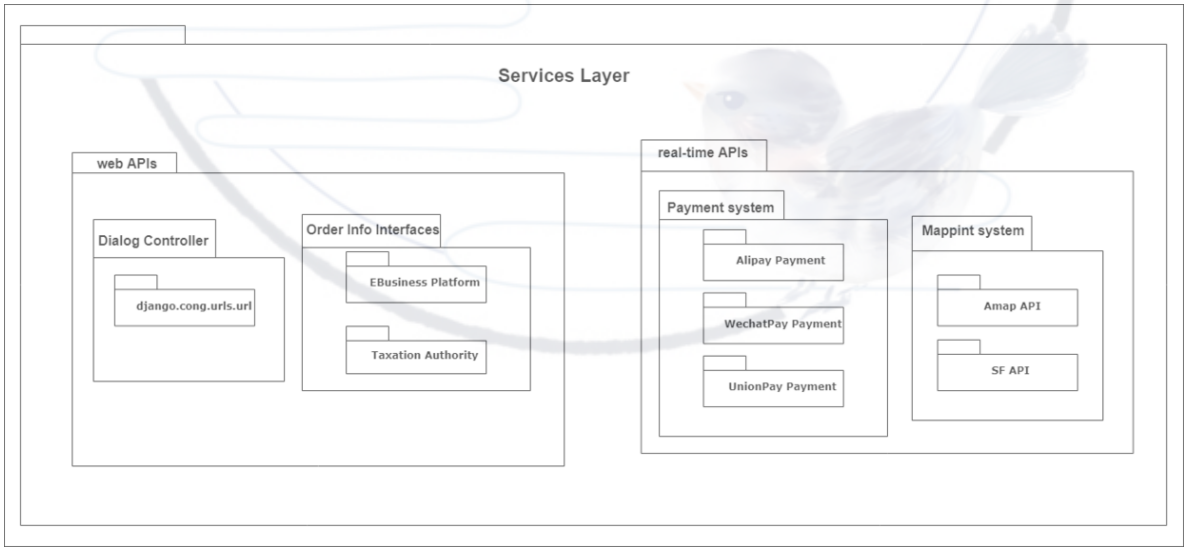
5) The CustomerFrontend updates only the part which needs to update.

10.2.2 Update the Service Layer by splitting it into two parts

We know that the Service Layer provides unified API interfaces of all functionalities for both web clients and mobile clients.

The updated API is split into two parts: the web APIs(containing Mapping System and Payment System) and the real-time APIs(containing Dialog Controller and Order info Interface).

The updated layer is shown as below:



The web APIs are provided in the form of JSON services based on HTTP/HTTPS. Web API interface serve functionalities that fit the request-and-response model. Most of the functionalities are provided by this kind of API.

The real-time APIs are provided in the form of JSON services based on WebSocket protocol, which allows modern web browsers (such as IE10+, Chrome and Firefox) to access it using HTML5 WebSocket feature.Functionalities that requires “server side push” are provided by this kind of API.

One typical example is the package real-time tracking functionality.

10.2.3 Add a WAF at the outside boundary of the Infrastructure Layer

Considering the safety issues, WAF (Web Application Firewall) can be added at the most outside boundary of the Infrastructure Layer.

The WAF middleware controls all data access to API.

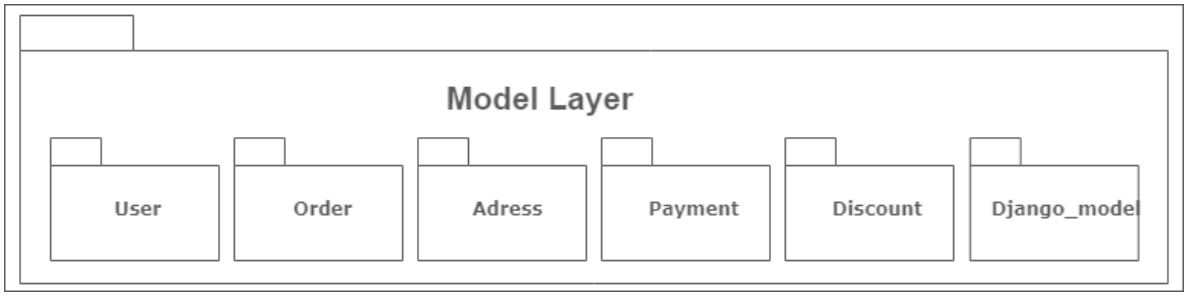
The WAF should at least provide limiting rating mechanism and application-layer access control (for example, IP blocking) functionality, aims to reduce security ricks from large scale brute-force / dictionary based enumerations. CAPTCHAs and access-blockings are both kinds of rate limiting mechanism.

10.2.4 Add a model layer to get data more efficiently

In order to search data more efficiently, we add a model layer between Controller layer and Data layer.

Models are divided according to classification in subsystems, and are in correspondence with classes in database. We design this layer to let upper layer get access to the database more clearly and easily.

The model layer is shown as below:



10.3 Open Issues

10.3.1 Robustness

Improve robustness of our design model includes architecture design, program coding.

Architecture design is a very large range, generally we are not particularly real-time business to exchange time for performance, space for performance and other programs. For example, we can store some original data about books first in the database, until the middle of the night when the pressure is small, and then we deal with them further.

For inquiries, with the aim to read more and write less, we generally take CQRS (Command Query Responsibility Segregation) program, as reading may also use the search engine technology and we do not use distributed transactions, but take the compensation mechanism in order to coordinate multiple system changes.

As for program codes, which is the last remaining thing to be seen, it is necessary for programmers to write robust codes. It can be partly measured by the judgement of the null value of string. Moreover, timely de-allocating the objects, avoiding writing dead loops to implement sleep, avoiding writing a possible death cycle. When writing code, programs can fully utilize some Program Static Analysis tools to find bugs and improve quality of codes, such as SONAR, PMD and so on.

10.3.2 Maintainability & Expandability

Take the processing of online payment as an example, our system provides an interface and concrete class. For various kinds of payment platforms, the system encapsulates the payment subsystem. And whenever we want to change the way we pay, we can only extend the functionality of this subsystem.

Thus, we can extend the other functions according to the original subsystem just in the similiar way.

10.3.3 Form a Book Dispatching Schedual System

As the bookstore grows bigger, it may need more and more methods to fetch tht books and send them. If there are certain somebody to handle the dispatching job, then it is really unnecessary and a waste of resources.

Form a dispatching schedual system can not only save the cost and resources but provide the 24-hours analysis for the current stock and orders, by which bookstore can know whether it need to buy books from the publishers or send books to customers in time. We may also add some machine learning algorithms to make the system smarter and more efficient.

10.4 Development Plan

Because it need hard work to build the front end and the user interface, we plan to have two of our group members to learn some knowledge about the implement of the front end and the user interface and after that they will implement corresponding subsystems through calling required and provided interfaces. The learning and implement time will take one and a half months.

For the design of database and back end, it is done by three of our group members. Two of them may spend one week to design ER diagrams and turn them into tables. We are going to dispatch the left person to build and manage database of MYSQL, Redis and MongoDB. As we think it will not be too hard, so we plan to finish this in two weeks.

Then, for the sever connection and the connection between back end and front end, as it is very important, this part is distributed to the back end team and front end team together. This part of work is to deal with requests and return information, so the back end need to coordinate with the front end. As we may encounter some problems, two months is necessarily needed.

Finally, after the above work has been done, we need to test our system, debug and complete deficiencies that have been found during the test process. This part may take up one month or more because the whole group has to do that together and be responsibility for every subsystem.

11.Course Reflection

11.1 Teamwork

For the teamwork, actually it shouldn't be neglected, because it is so important for our efficiency. At First, we were completely in the dark about project and so worried about it. But it is the group discussions from time to time that help us to make process. We first have a brainstorm about the project, and it truly help us to find the direction. Then We made plans and divided our team into small groups to finish the job. Finally, as the professor told us, we do this big project step by step in an iterative way. Therefore, just as the old saying goes, "where there's a will, there's a way." We learned new things and make process step by step and now we are proud of our final project. We can say that our efforts are worth the time.

11.2 UML Developing Tool

For the UML developing tool, our group members are using different tools at first. Three members use StarUML and two members use processon. These two tools have their advantages and disadvantages. StarUML is more convenient and processon is more beautiful. For our third assignment, we find a new tool named draw.io. We use it to draw our architecture system diagram and class diagram, because its style is good and it is easy to use. But our use case diagram are drawed with StarUML, because we can simply drag diagrams into the diagram and it can generate global diagram automatically by using StarUML.

11.3 Presentation

There are totally 4 presentations for all the groups in this team. We can always make full use of our presentation time because we are Group 3. Because we do not to be normal, in addition to the our assignments, we present what makes us outstanding every time. Most importantly, we try our best to make our work conciseness and correctness. What's more, we organize our slides logically and pleasant to watch. At first our presentations are in Chinese but then we use English to present our idea because our class is in English.

11.4 Materials

Our professors direct us in our class, but we still need to searching for lots of materials for reference. In addition to our textbook, we use Google and our library's site on the Internet to look for materials, which is truly helpful. Now, though we are still not very professional, we have learned a lot things about object-oriented analysis and design.

11.5 Self-reflection

Zhe Zhang(张喆) [Team Leader]

In this semester, as the team leader of Group 3, I have witness the whole System Analysis & Design for our online bookstore —— YouMu(UM) bookstore.

At the beginning of our design, we don't know how to start the whole things, which means you have got an assignment to build a system like Dangdang or other online bookstore, you don't even know where to start(Interfaces?Function?Architecture?or other thing)

At the first class this semester, our professor Cao told us what is an ideal design, and what's the development mode of our common design and also we get a global view of "What is UML and what is Object-Oriented Analysis and Design".

And when we do our Assignment 1, we start with with a brainstorm, everyone shared his standpoints and we list all the thing there. Then we discuss all the items step by step in detail, and we confirm the users, use cases, main function of our bookstore, and more things. Nonetheless, on the first presentation, we found ourselves forgot something more personal, which mean we regard the origin task as a project and homework we must finish, not a real system we build our self.

So, we polish the first document into fresh version, which add our own idea for our online bookstore: We name the bookstore as YouMu bookstore, which the letter in Chinese is UM, it represent the content we take in class —— UML. And since the second presentation, we create our own PPT template, which reflect our idea of the bookstore —— Chinoriserie.

We have tried so many tools to draw the UML chart more efficient and beautiful, and meanwhile, we learned a lot in use these software and website.

As for the references, we searched quite a lot in the website and library. Because at the beginning of the design, we cannot hold a clear mind, so we read some article which introduce how to design analysis and design a system, and to be honest we also take example by some blogs and article, but I swear we done all the thing by ourselves and we just used them as references.

As the team leader, I have learnt something more this semester —— teamwork. You should master the whole thing we are supposed to do prior to your team members, and you should organize the short meeting nearly three times a weak, I tried to allocate the task in equal, and I do think what I done well was that I allocated different work for each one in each assignment. So I think all of my team member have learned many concept in UML, not only one part. And we changed 4 individuals for each presentations, this will also help them grasp more skill.

In the end of this course, I want to say that both the Professor Cao and Professor Liu did an extraordinary job, which made me enjoy this course. One thing is special in my mind, in the first presentation, Professor Liu waited until 10:30pm to listen all the group and she even stop during the 3 hours, and the essential thing was that she gave an evaluation for each group, so we can find our deficiency and we can complete ourselves. I must say THANKS to our teachers, You are our role model.

I hope we will take our analysis and design into real work, and we can finish our own YouMu online bookstore one day!

Thanks Professor Cao and Professor Liu again, and I want to say to my team members that you have worked hard, thanks everyone!

Siteng Ma(马思腾)

From my perspective, UML is useful but not easy to be taken advantage. The online bookstore conception gets mature by the discussion and cooperation of us five. Except for the knowledge taught during the class, there are a variety of principles for us to learn by doing and searching in the internet. By designing the subsystem and drawing plenty of diagrams, I realize the importance of UML after implementing the whole system. The vary steps to refine the whole project are miserable but essential. We have overturned the models several times and replaced with a better one. So that we have finally established a UML model which is not that bad. We five boys consolidate firmly and learn much by doing these four assignments. All in all, I benefit a lot through the team work in this semester.

Gefei Wang(王舸飞)

During the whole project, although I mainly focus on prototyping, presentation and some activity diagrams, I also learned a lot while communicating and discussing with all my team members. Choosing which architecture and framework to use requires us to have a clear understanding about the advantages and disadvantages of all the methods. As a sophomore that dose not have so many experiences, I searched a lot information on the Internet and referred many books that recommended by our professor. After getting my own part done, we also shared our knowledge with other team members to make sure that we are always keeping in touch.

As for the limitations, I think we still lack of real experiences, which means we did not have many choices when implementation. Actually, as the user's requirements will also change, it is acceptable that we will have to make changes to our previous architecture, and that is exactly the way a good product is made.

Le Yang(杨乐)

During the process of developing our project, I was mainly responsible for design the use case model, analysis model. At first I was totally in the dark about what to do, but with the direction of our professor and help from my team members, I improved myself a lot and finish my job. The main shortcoming of me is that I was too hurried and sometimes neglected the feelings of other team members, so I need to overcome this in the future because the feelings of other team members is very important for good team work. Here, I want to thank my team members and my director, for they made me do better than I could do.

Kaixin Chen(陈开昕)

Participating in a group to complete the uml project is a very rare experience. Through this period of experience, I learned how to get along with others, how to hand over and complete tasks, how to be a team member to make the team's progress more efficient. At the same time, I also learned a lot of basic techniques and methods of system analysis and design. Know the system builds, function points, use cases, and the various tools and methods used to achieve these goals during development. For example I tried several UML developing tools during this semester, among them StarUML is a powerful and free UML developing tool and can clearly show the structure of our project and the relationship between classes. However, taking the whole group work into consideration, it's more convenient to develop UML via ProcessOn. Really apprciate for this experience.

12.Contributions of Team Member

Zhe Zhang(张喆) [Team Leader]

- First presentation & PPT
- Global Architecture: Architecture Diagram, Description, Deployment Diagram
- Subsystems and Interfaces in Architecture
- Realization: Two Detailed Interfaces
- Potential Inprovements(update the diagram)
- Organize Documents(Second, Third, Final)

Gefei Wang(王舸飞)

- Second presentation & PPT
- Prototyping: Description, Register Example * 3
- Initial Snapshot * 7
- Implement the Login System(codes)
- Design Patterns
- Critical Decisions

Kaixin Chen(陈开昕)

- Third presentation & PPT
- Glossary of Terms, Supplementary Specification
- Architecture Style
- Open Issuses * 3
- Potential Inprovements
- Organize documents(First, Third)

Le Yang(杨乐)

- Forth presentation
- Global System Use Case
- Subsystem Use Case: login & register Subsystem, search & visitBookPage Subsystem
- Domain Modal: Class Diagram (part), Relationships Between Classes(part)
- Analysis Model(part)
- Design Mechanism: Persistence
- Use Case Realization: Login
- Development Plan

Siteng Ma(马思腾)

- Forth PPT
- Subsystem Use Case: Message Subsystem, View Data Subsystem
- Domain Modal: Class Diagram (part), Relationships Between Classes(part)
- Analysis Model(part)
- Design Mechanism: Information Exchange
- Use Case Realization: Visit Order Form

- Limitation

13.References

Reference	Description
《UML与Rational Rose 2003从入门到精通》	This book introduces the basics of UML language in detail, as well as the application of UML in object-oriented software system analysis and design, and explains the object-oriented analysis and design process through rich examples, inspiring readers how to learn in UML language.
《Applying UML and Patterns》	From this book we further understand and identify analysis and design and concepts and understanding. Object-oriented analysis emphasizes the discovery and description of objects in the problem domain, creating domain descriptions from the perspective of objects. Therefore, when doing analysis, we choose to start with the big problems embodied in the use case diagram, such as how users shop correctly, and gradually split the basic objects and relationships, and analyze related domain models and relationships.
《UML2面向对象分析与设计》	We focus on the parts of demand analysis, design principles and patterns, architecture design, and component design. Further understanding the methods and considerations of requirements analysis, learning the drawing methods of related expressions such as domain models and concept class diagrams, and understanding the form of the representation structure of the package diagram.
https://blog.csdn.net/fanxiaobin577328725/article/details/51700528	From the above article, we further learn the meaning and detailed introduction of the package diagram, including but not limited to name, element, visibility, import and export, and the relationship between package and package and generalization. This has played an important role in building our own architectural analysis.
https://blog.csdn.net/gstrong298/article/details/22623509	The article introduces the knowledge of concept class diagrams. We have learned the methods of finding conceptual classes, the differences between conceptual class diagrams and design class diagrams, the drawing methods of class diagrams, the meaning of associations and properties, etc., which provide a great reference for us to draw conceptual class diagrams.
http://plantuml.com/zh/sequence-diagram	The article tells us how to draw various timing diagrams and the various syntaxes and functions in the sequence diagram. It has important reference value for our uml mapping.

Reference	Description
https://www.tutorialspoint.com/uml/uml_architecture.htm	Any real-world system is used by different users. The users can be developers, testers, business people, analysts, and many more. Hence, before designing a system, the architecture is made with different perspectives in mind. The most important part is to visualize the system from the perspective of different viewers. The better we understand the better we can build the system. This page help us learn the architecture of ump development.
https://blog.csdn.net/fanxiaobin577328725/article/details/51700528	This article describes the drawing methods and considerations of the package diagram and the details. It is very helpful for the drawing and attention of our system architecture logic package diagram.