YOUMU BOOKSTORE

PART.3 DESIGN MODEL

2019.6.7 | by: Group 3

1. Overview

1.1 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.

1.2 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.

1.3 System Overview

1.3.1 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.

1.3.2 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.

1.3.2 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.

1.4 Progress

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.

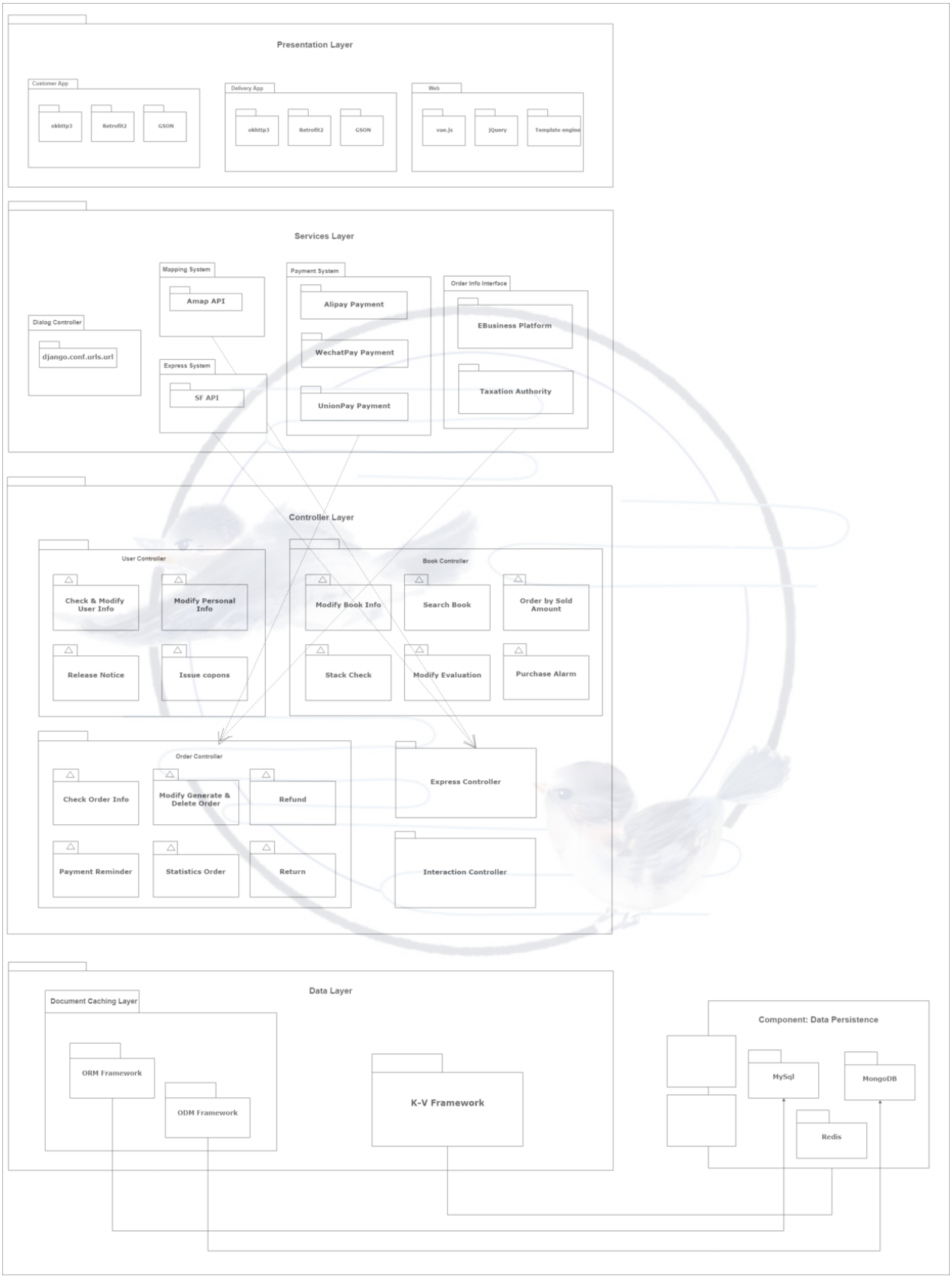
As for the prototyping, we have already created a demo connecting the database and our login subsystem interface. Users are able to register and log in though the browse. The coordinate data of him or her will be stored perpetually.

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.

2. Architecture Refinement

2.1 Global Architecture



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

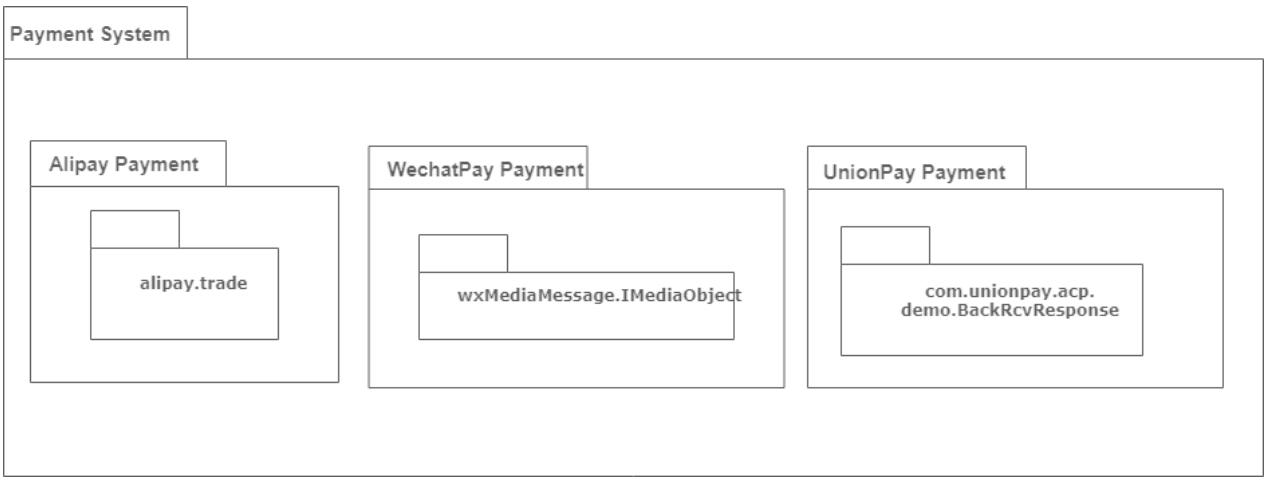
And the library we import in the subsystem is listed here

- **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;

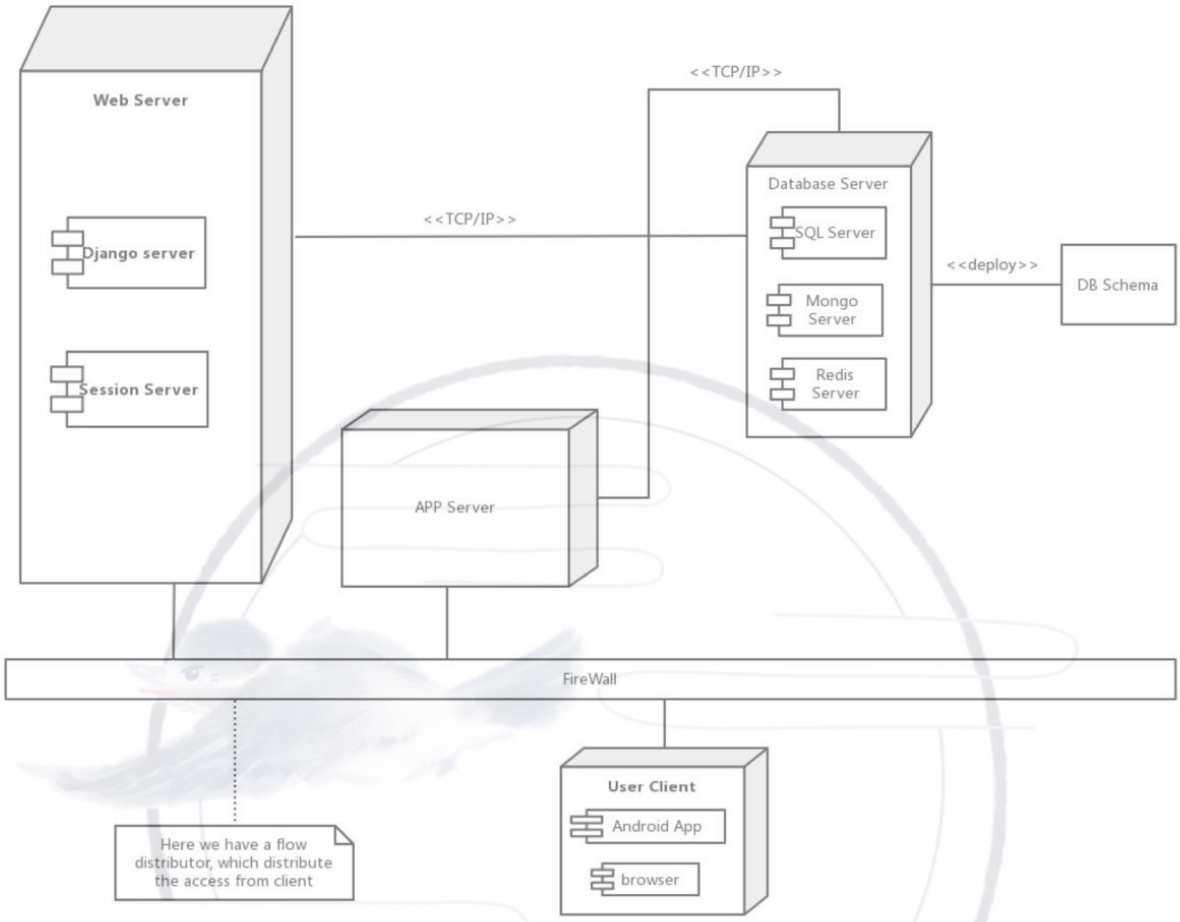
MongoDB for storing mass data that is less relational and has less demand of ACID, such as GPS location data from all GPS sensors in each collecting period (30 seconds). MongoDB enables those kinds of data to be stored and accessed more efficiently.

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.

- 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	+create(dict) +save() +update(dict) +delete(dict) +get(dict) +filter(dict) +sort(dict) +check(dict)	ODM Framework ORM Framework K-V Framework	
	IScheduleModel			
	IAddressModel			
	IPaymentModel			
	IInvoiceModel			
	IUserModel			
	IDiscountModel			
	ITaxModel			
	IInventoryModel			
DataAccess	MySQL	+SELECT(args) +INSERT(args) +UPDATE(args) +DELETE(args)		
	MongoDB			
	Redis			

2.3 Interfaces Between Our System and External Systems

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

2.3.1 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":"SF201608081055208281",
    "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"
        }
    ]
}
```

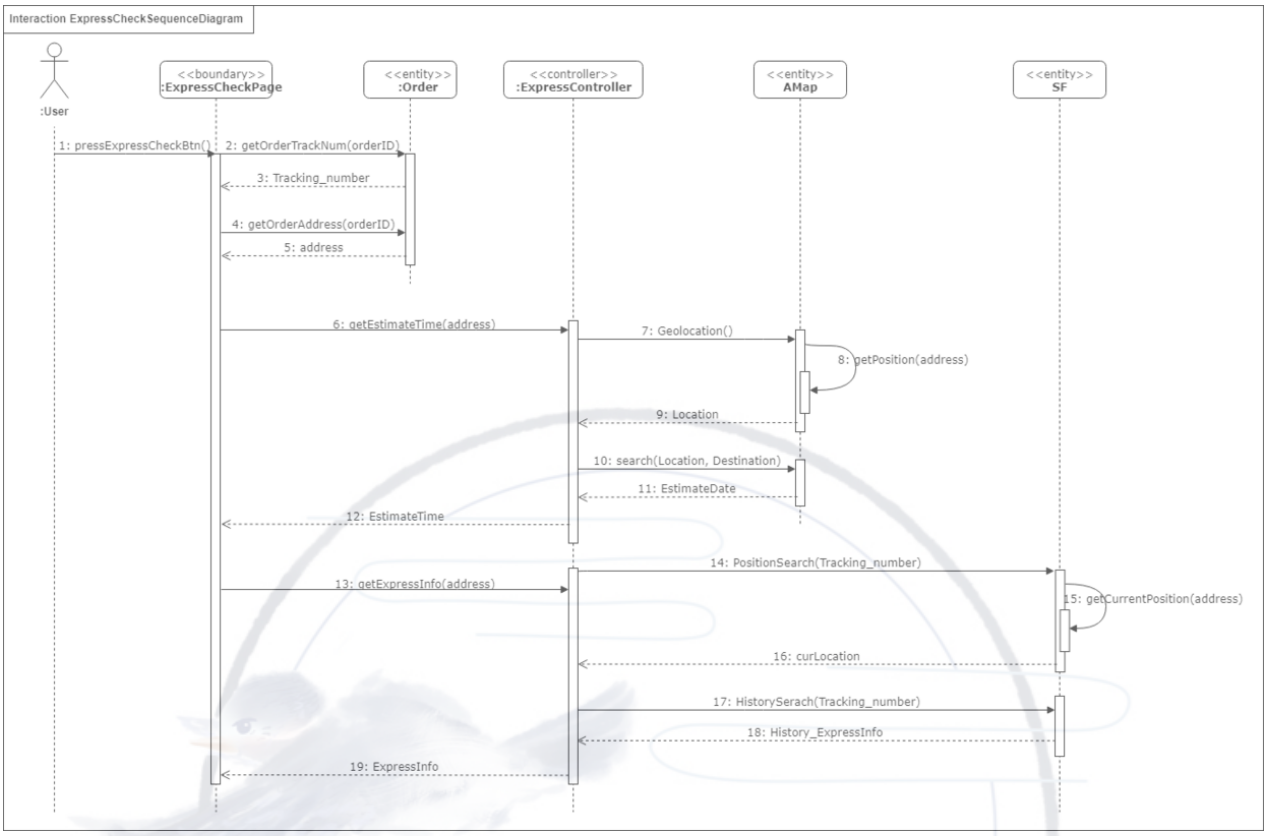
- JSON return example

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

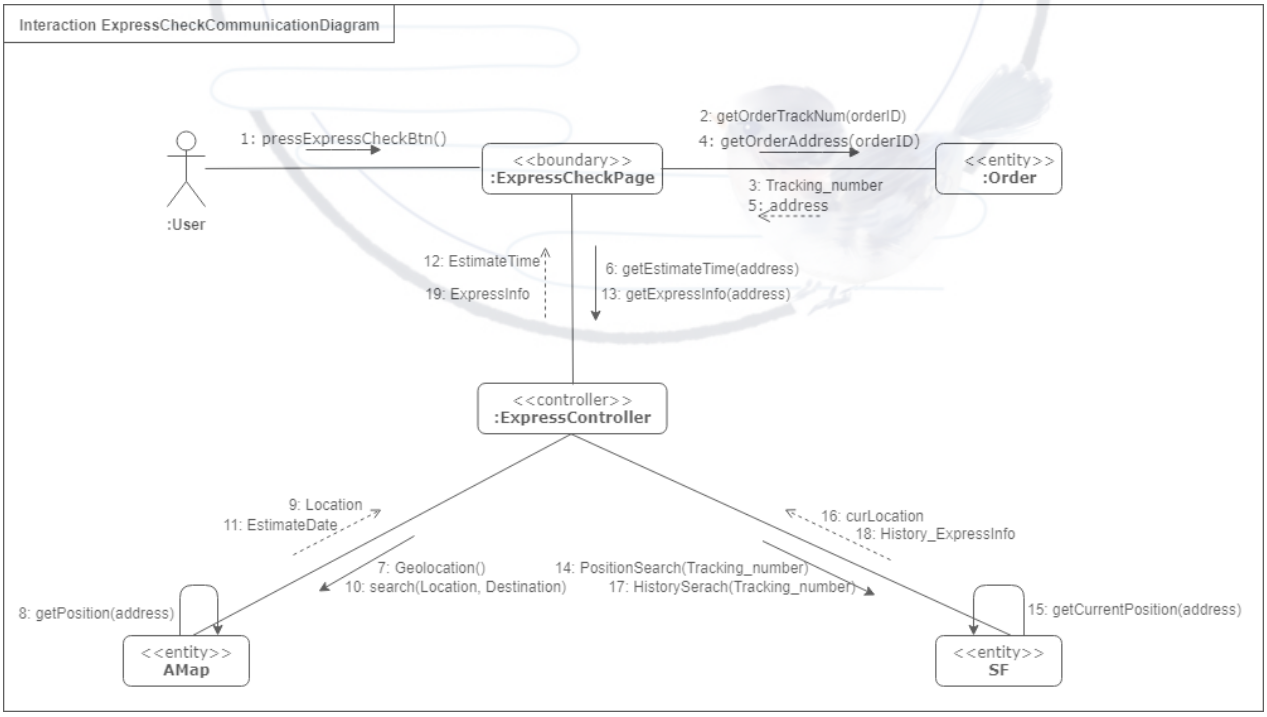
- API interface call

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

2.3.2 Sequence Diagram



2.3.3 Communication Diagram



2.4 Order Query Subsystem and its Interfaces Specification

specify the data and database in datail

2.4.1 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.

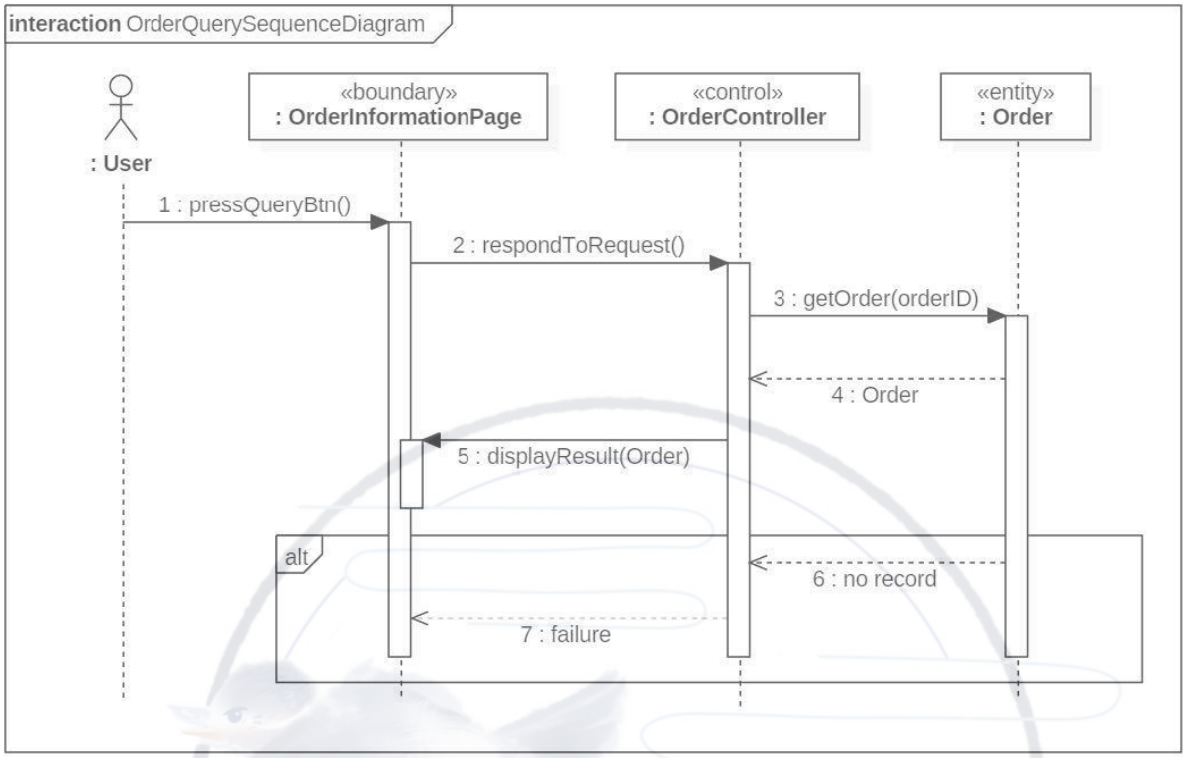
- The interfaces we designed are mainly showed as follows:
  - `respondToRequest():bool`
    This interface is implemented in the order query boundary class. It is invoked after the user press the query button. Then it will call function to execute the query.
  - `getOrder(orderID: string): Order`
    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”
}
```

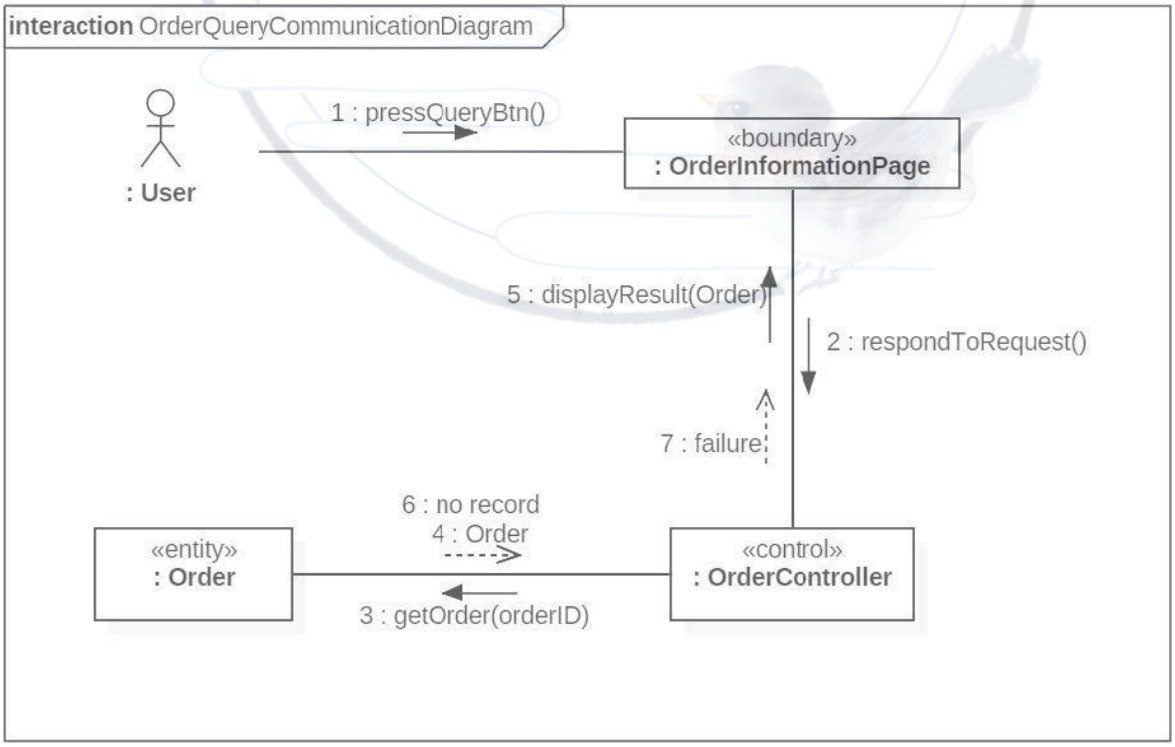
  - `displayResult(order: Order):bool`

    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.

2.4.2 Sequence Diagram



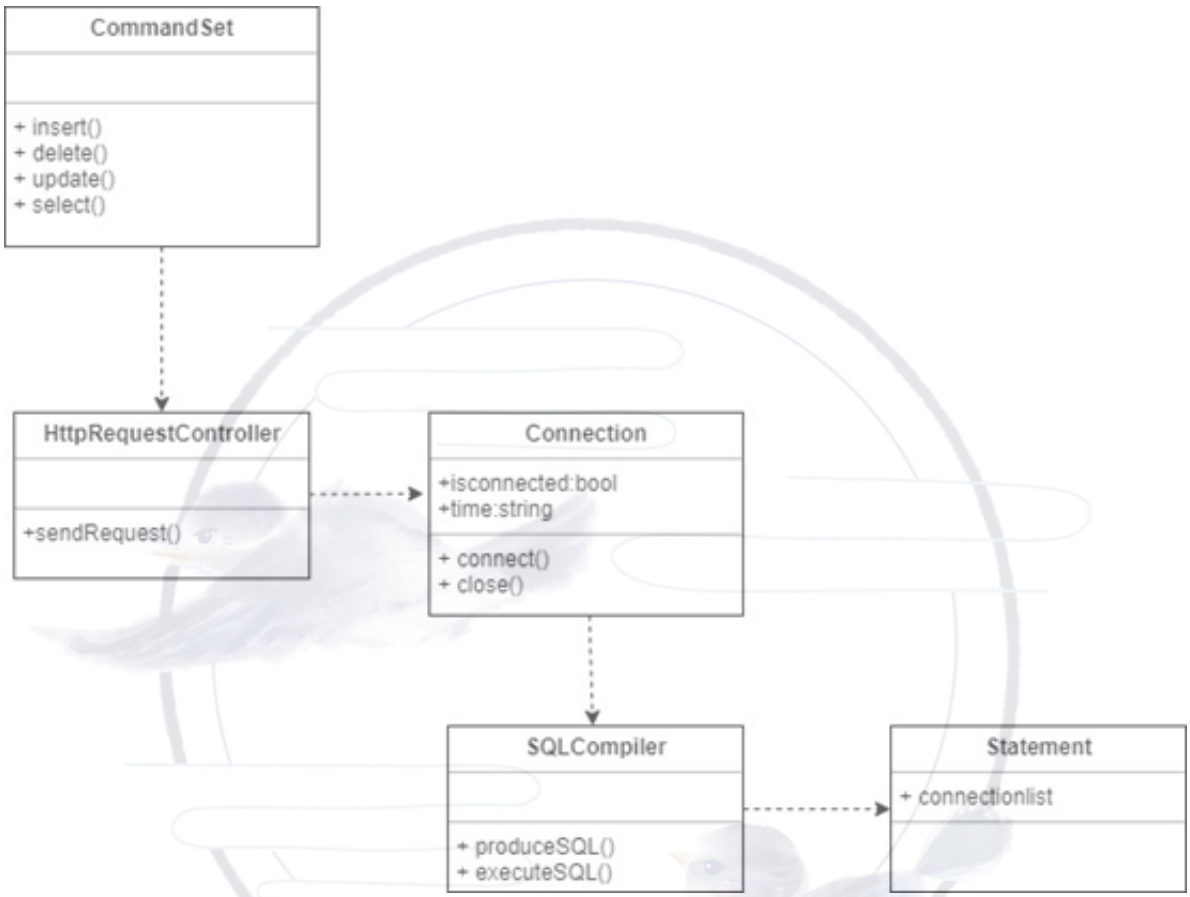
2.4.3 Communhication Diagram



3.Design Mechanisms

3.1 Persistence

3.1.1 Class Diagram



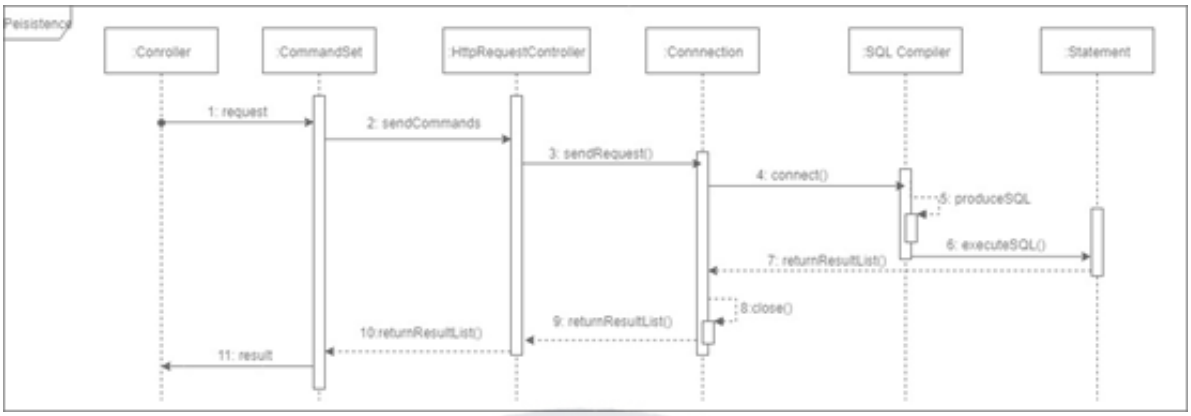
There are three parts of the above class diagram:

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.

3.1.2 Sequence Diagram

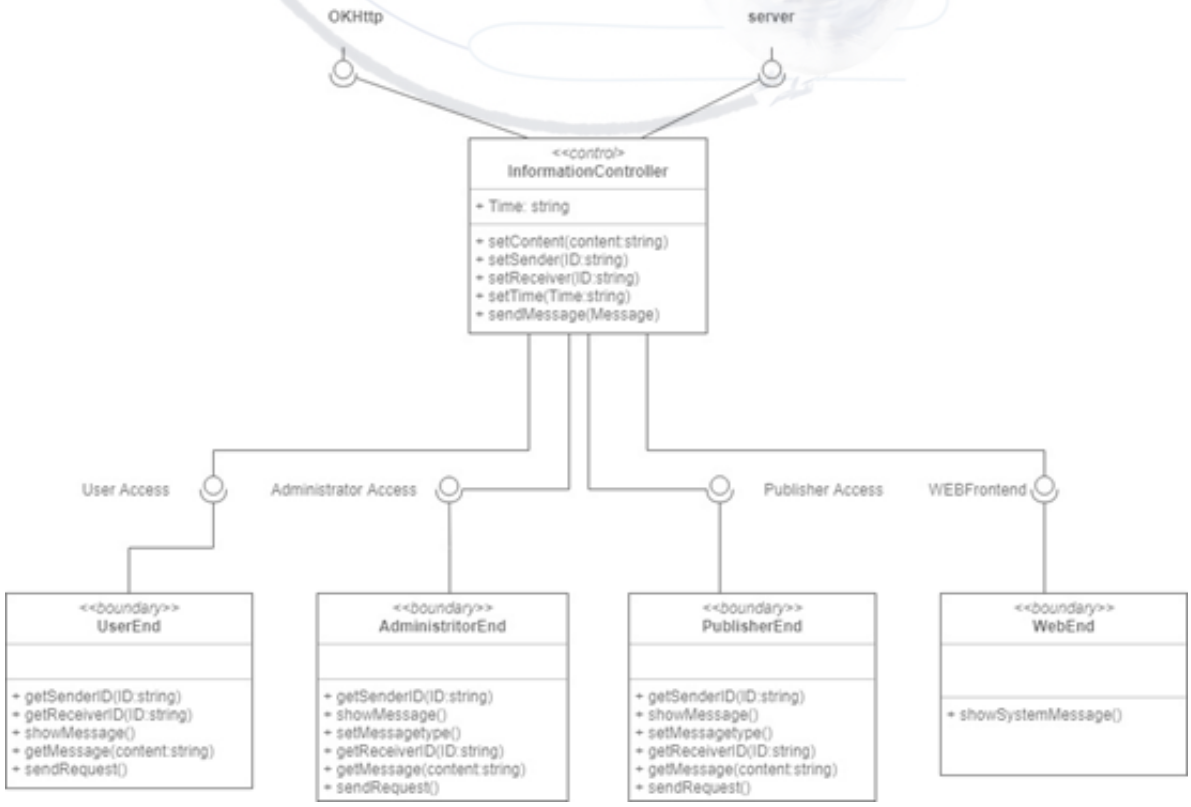


After receiving the request from front end,
the CommandSet will send command to the HttpRequestController. The latter will then send request to the back end. The connection with the database will be built at same time of the production and execution of SQL. When the result list is returned from SQL compiler,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

3.2 Information Exchange

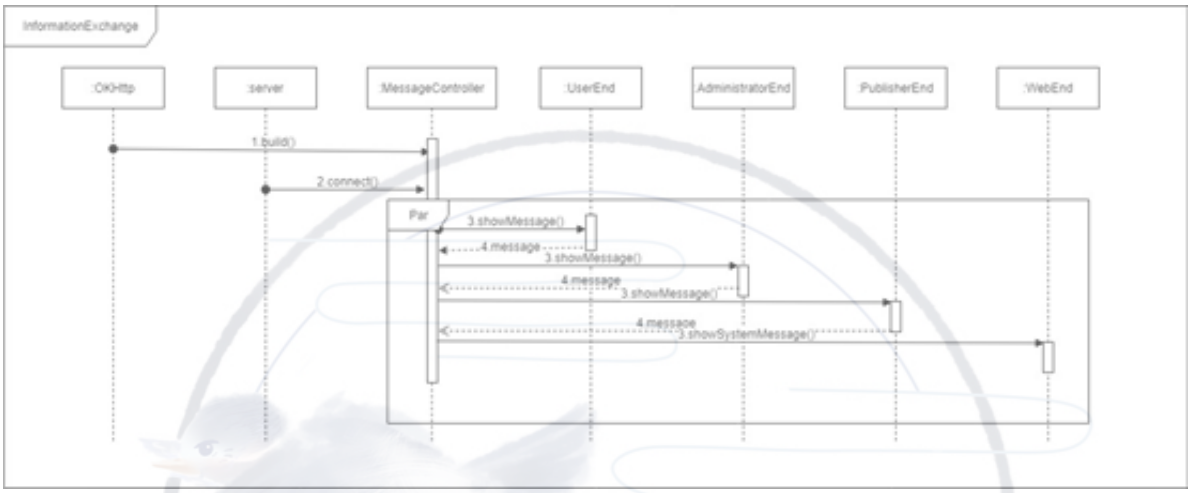
3.2.1 Class Diagram

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.



3.2.2 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.

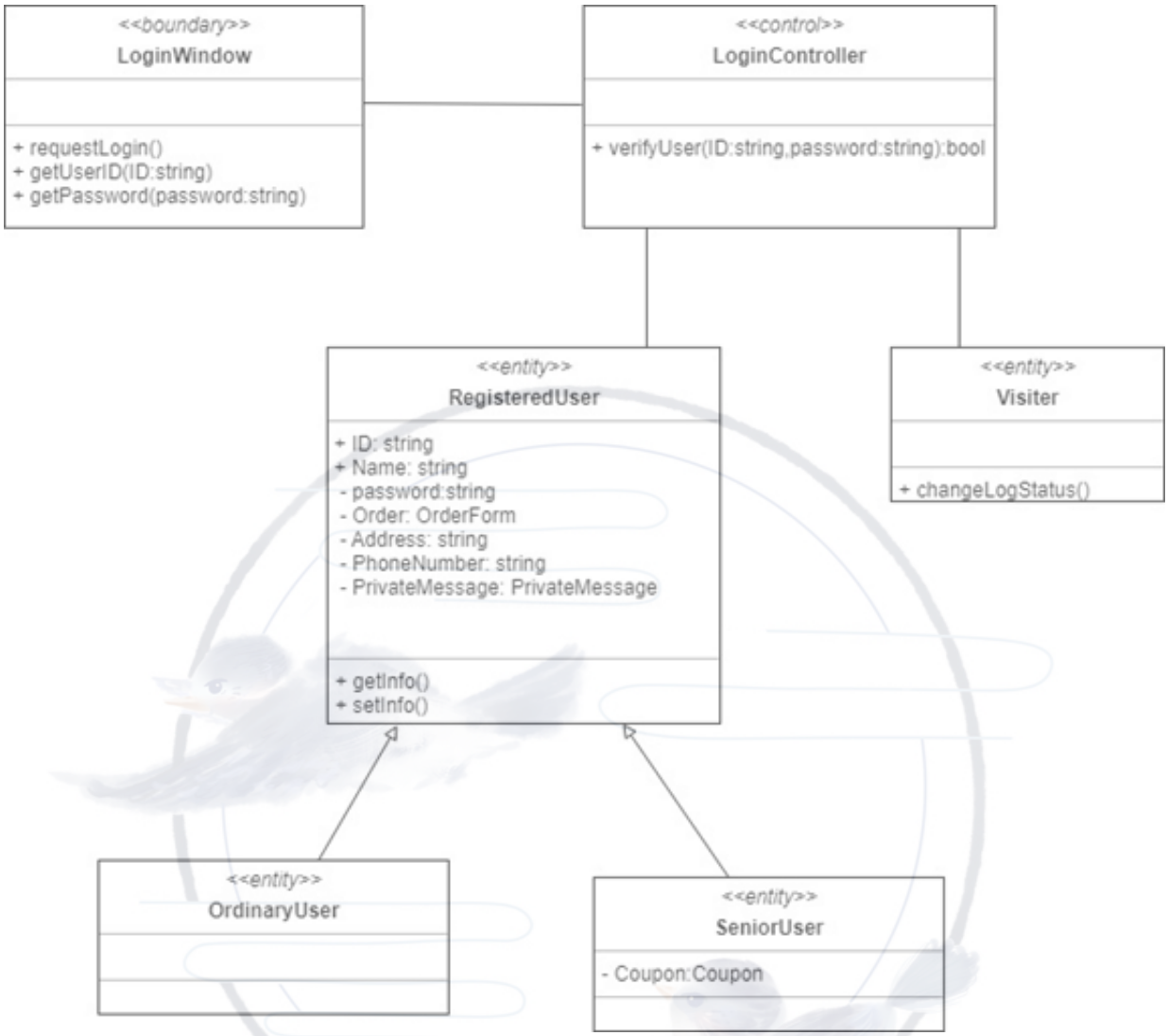


4.Use Case Realization

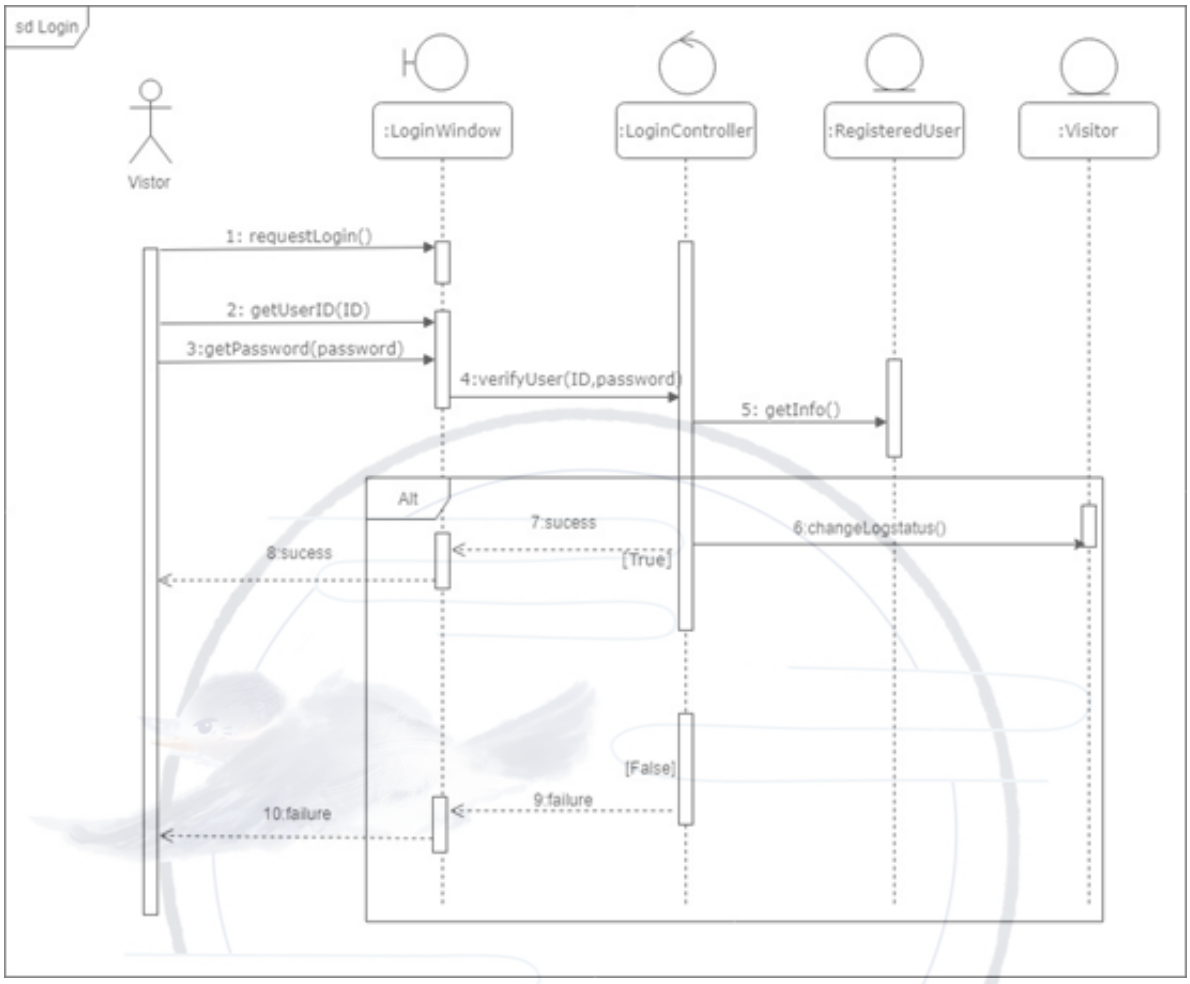
4.1 Login

4.1.1 Login Realization

4.1.2 Class Diagram



4.1.3 Sequence Diagram

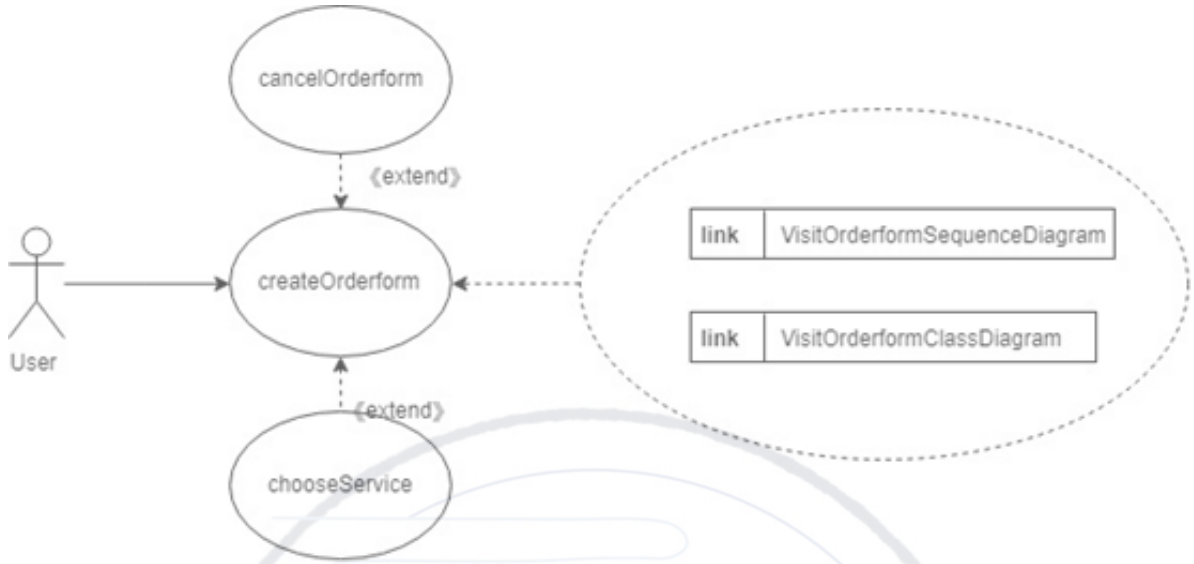


In the Login use case, the communiaction between the RegisteredUser class, Visitor class and database is showed as mechanism Persistence.

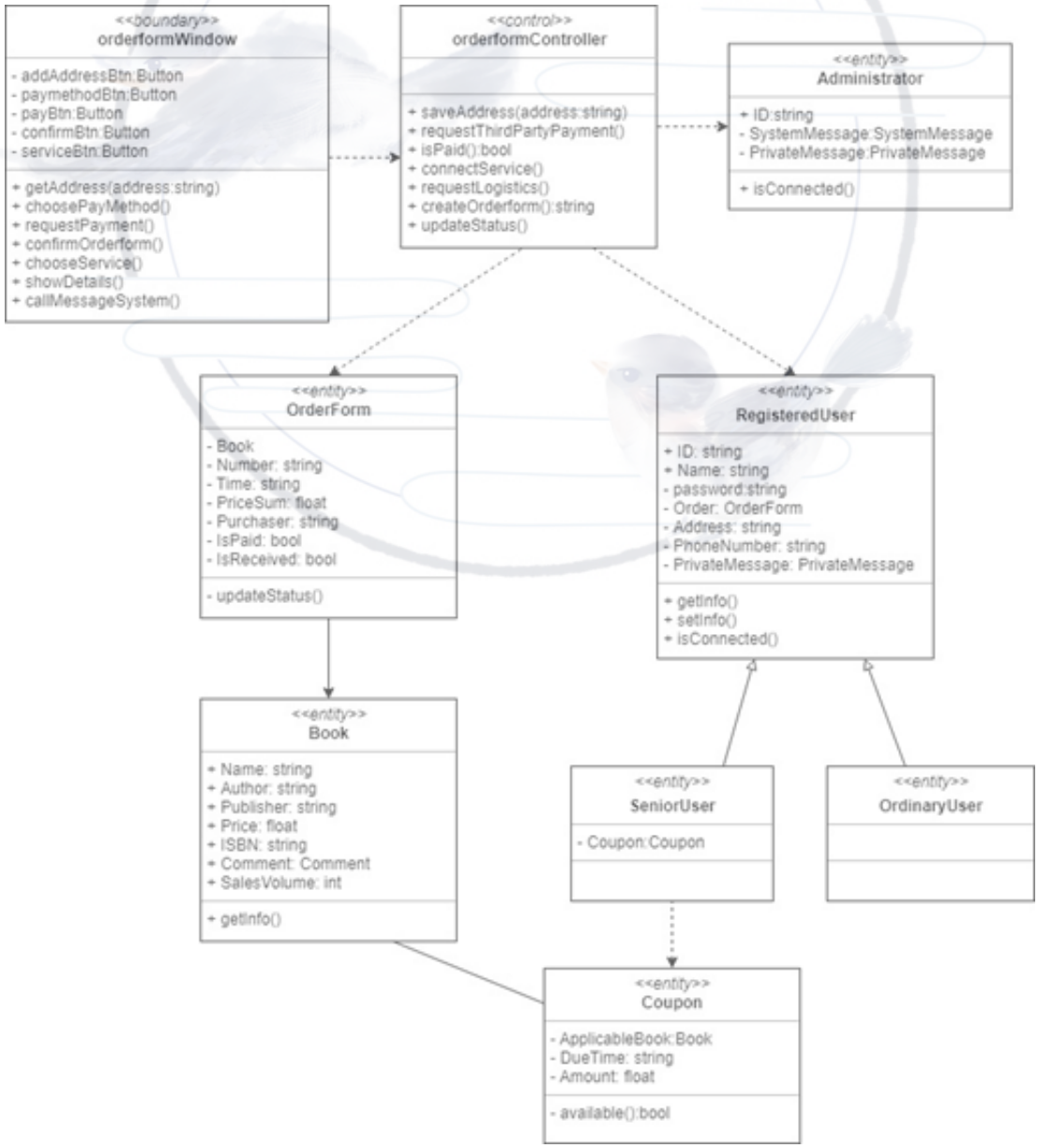
If the Visitor wants to log in, he should first request to log in and then put his ID and password in the LoginWindow, and then the window will send these information to the LoginController. The LoginController will check the database of users and verify the information is true or not. If it is true, 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.

4.2 Visit Order form

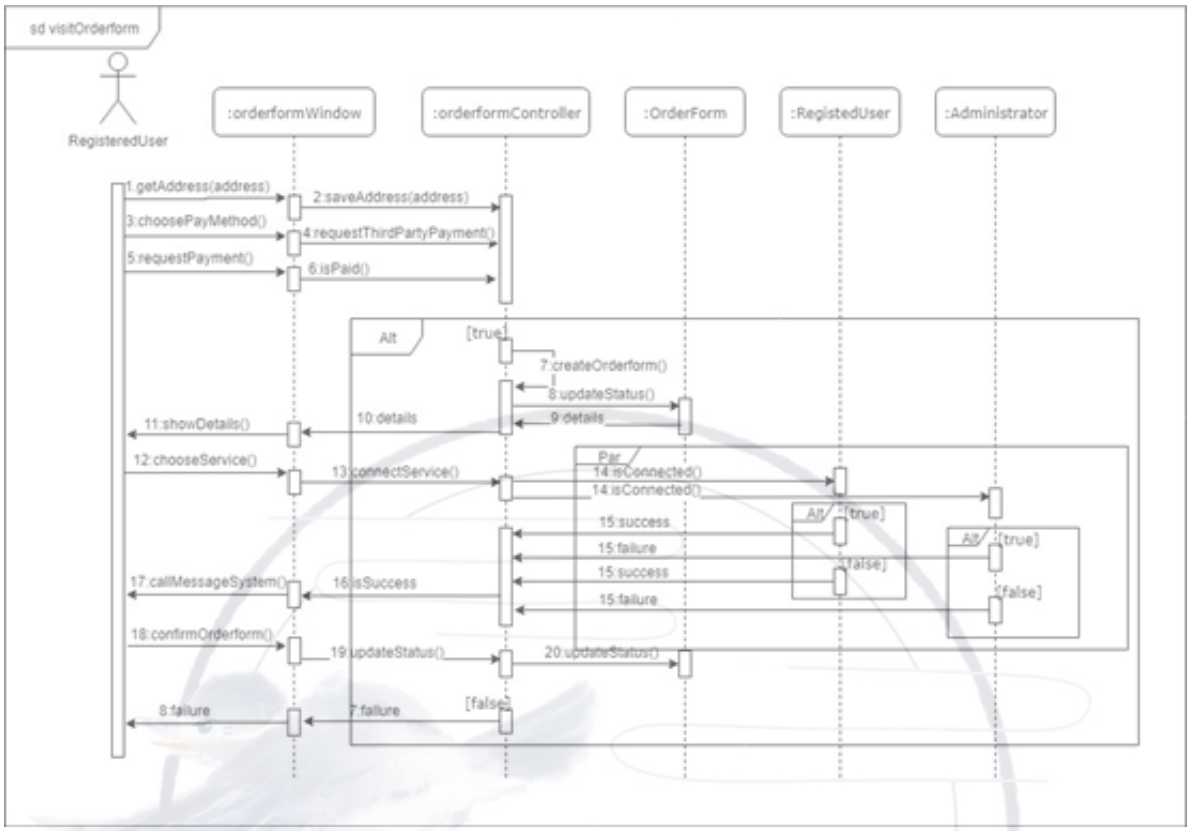
4.2.1 VisitOrderform Realization



4.2.2 Class Diagram



4.2.3 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.

5. Prototyping

5.1 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

5.2 Register Example

5.2.1 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.

5.2.2 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

5.2.3 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);
    });
    }

  }
}
```

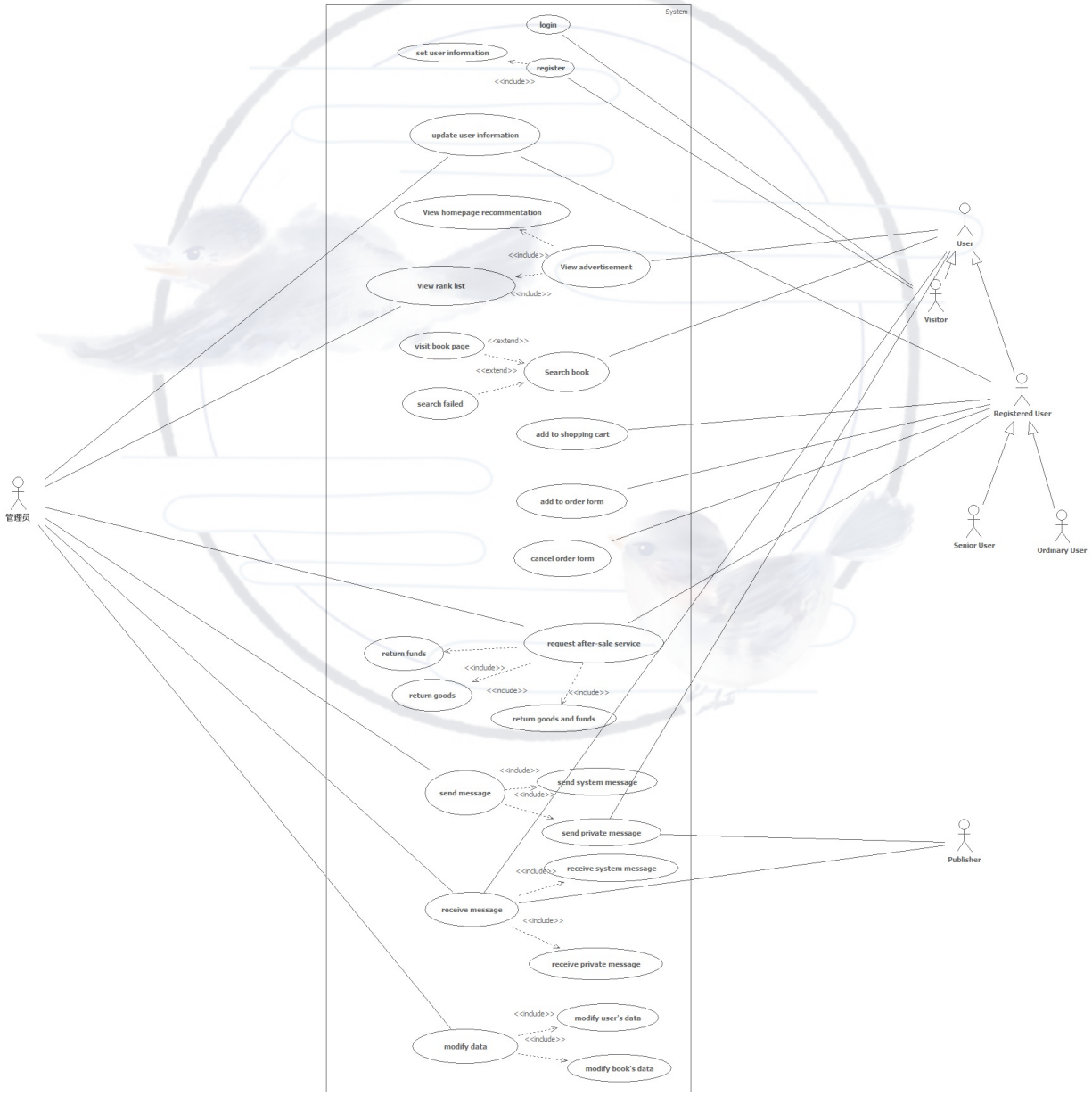
This part shows how to handle click method as a whole. It will judge if the input is valid, them send the callback to view layer

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.

6. Update Use Case Model and Analysis Model

6.1 Use Case Model

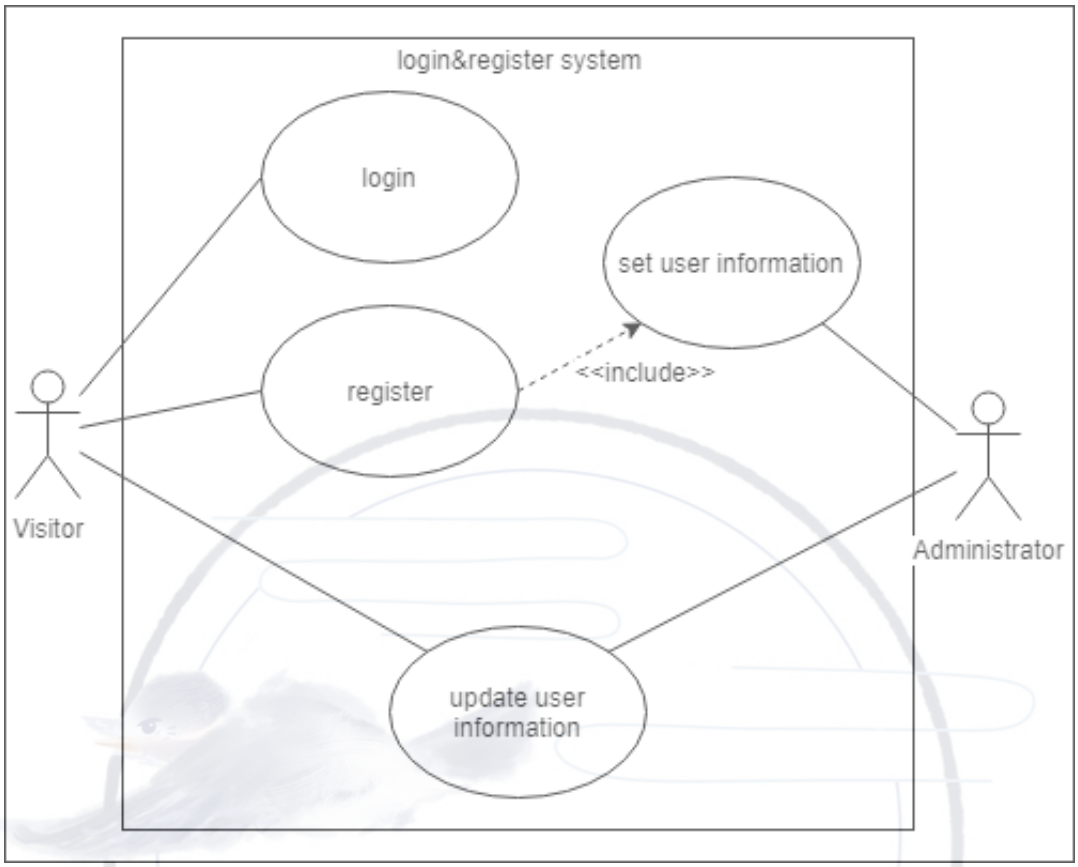
6.1.1 Global System Use Case



6.1.2 Subsystem Use Case

login & register Subsystem

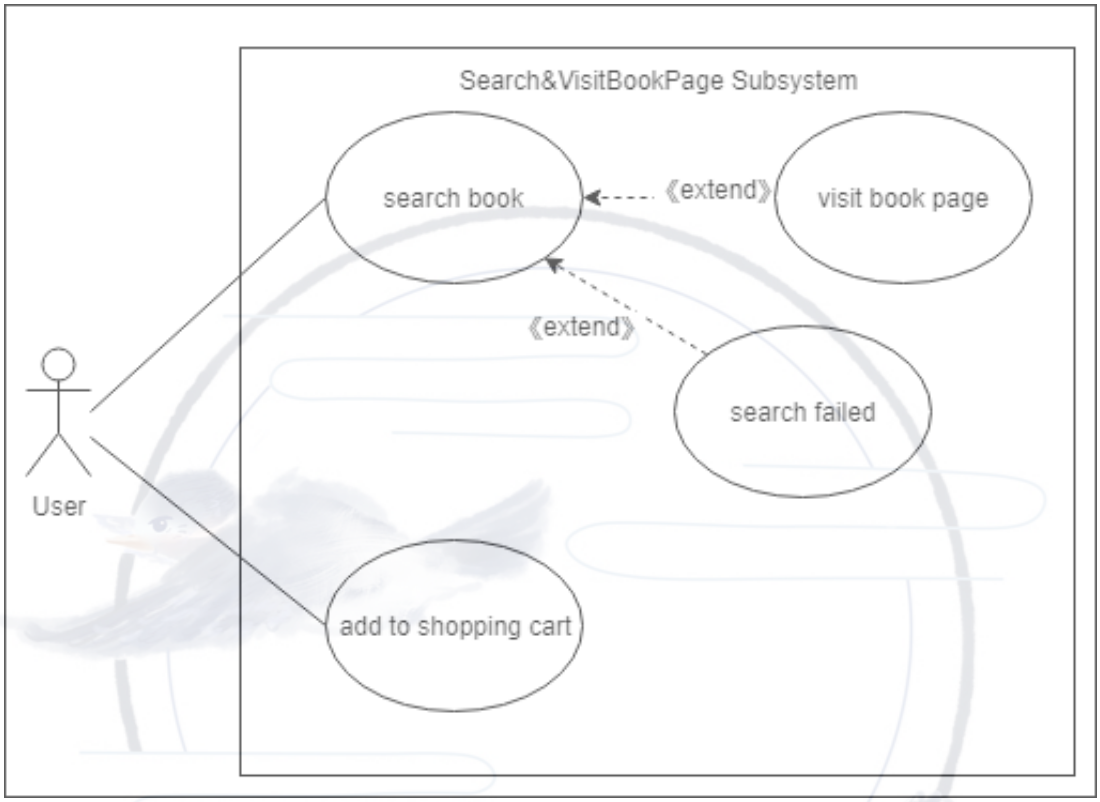
- Use case diagram



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

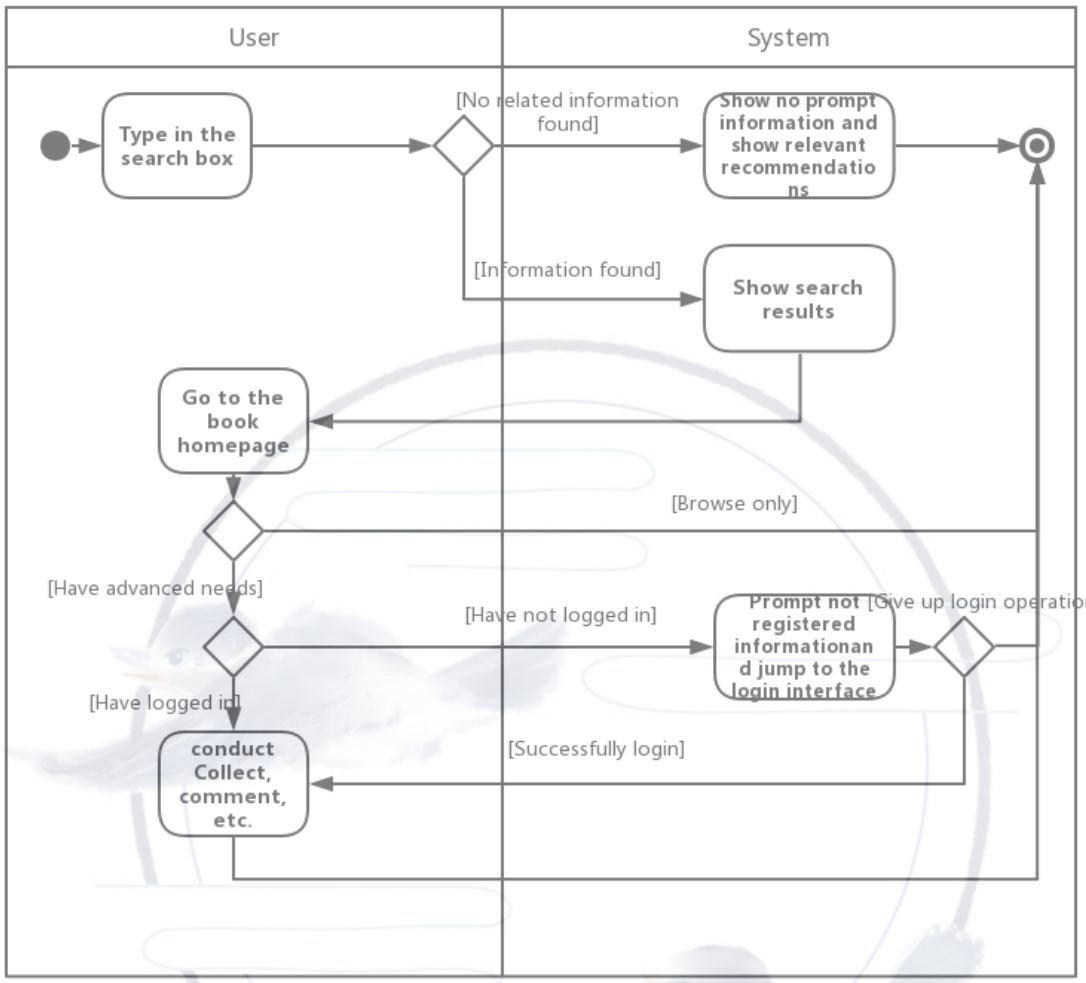
search & visitBookPage Subsystem

- Use case diagram



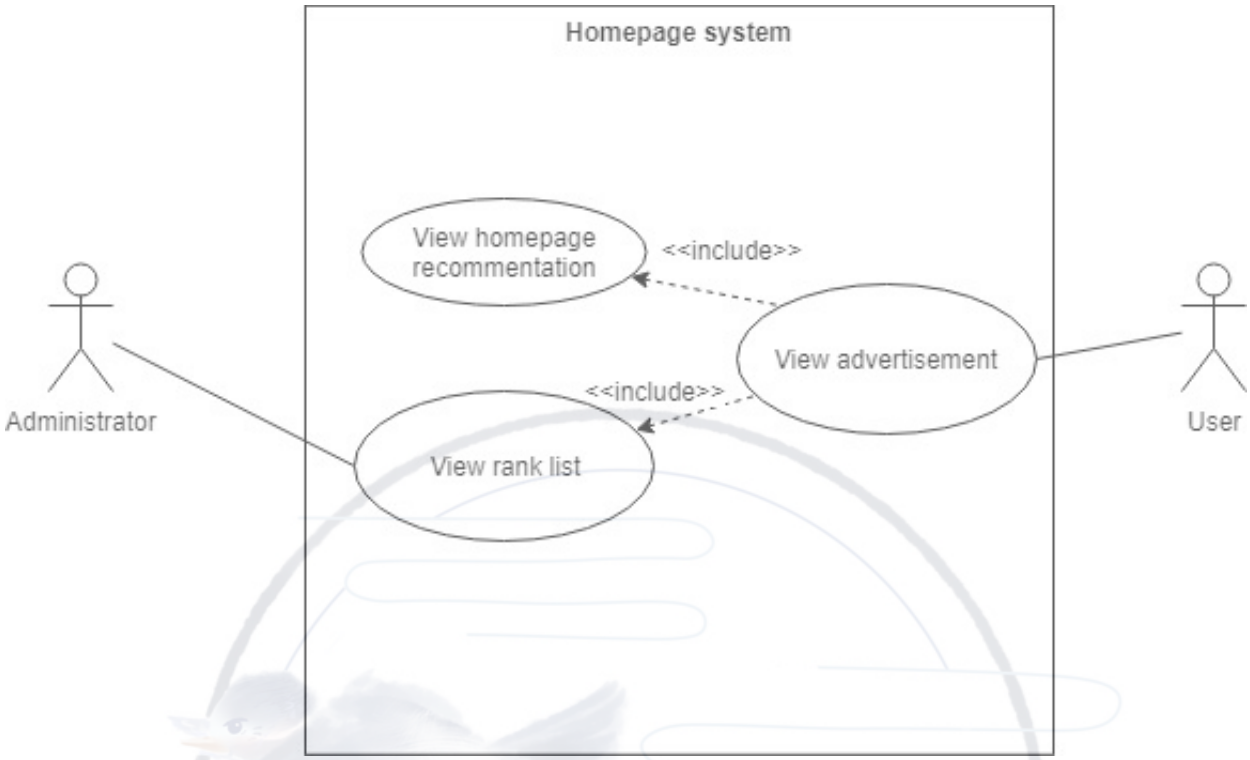
Use case: Search & VisitBookPage Subsystem
Actor: Customer
Precondition: Connect to the internet and successfully access the page or app
Basic flow: A. The user enters the content in the search box (author name or book name or publisher name) B. Return search results a) The book or author does not exist 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



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.

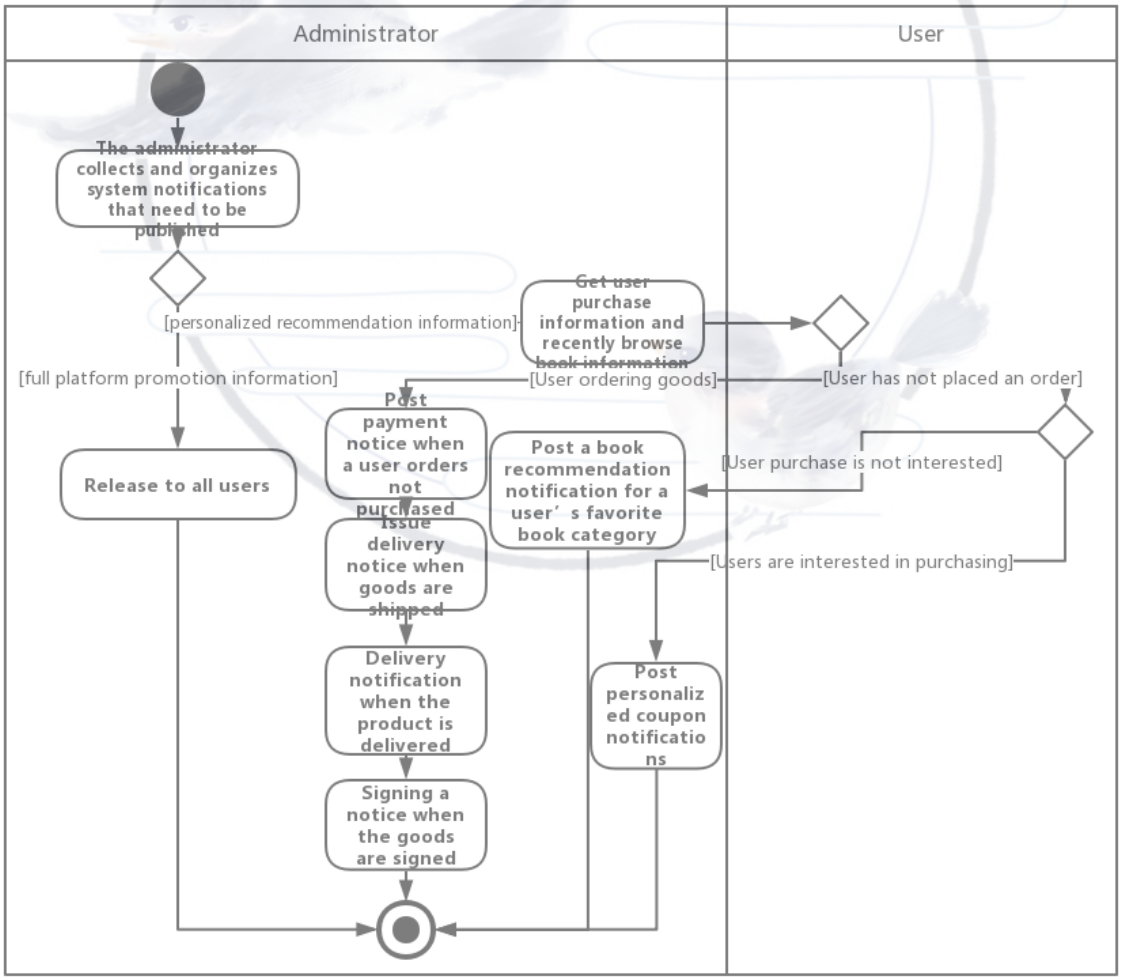
Message Subsystem

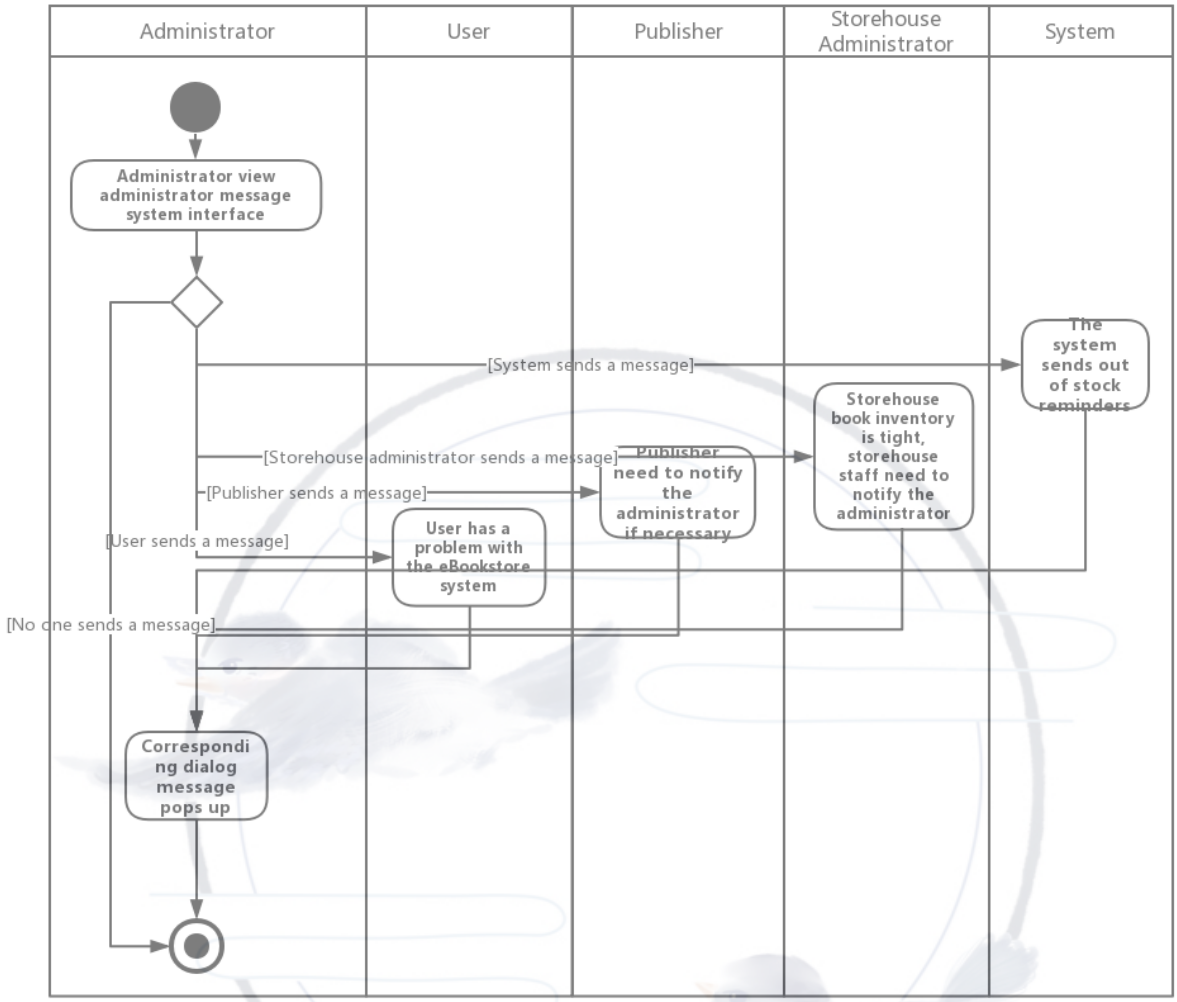
- Use case diagram

Use case: SEND MESSAGE
Actor: User, Publisher, Administrator
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.

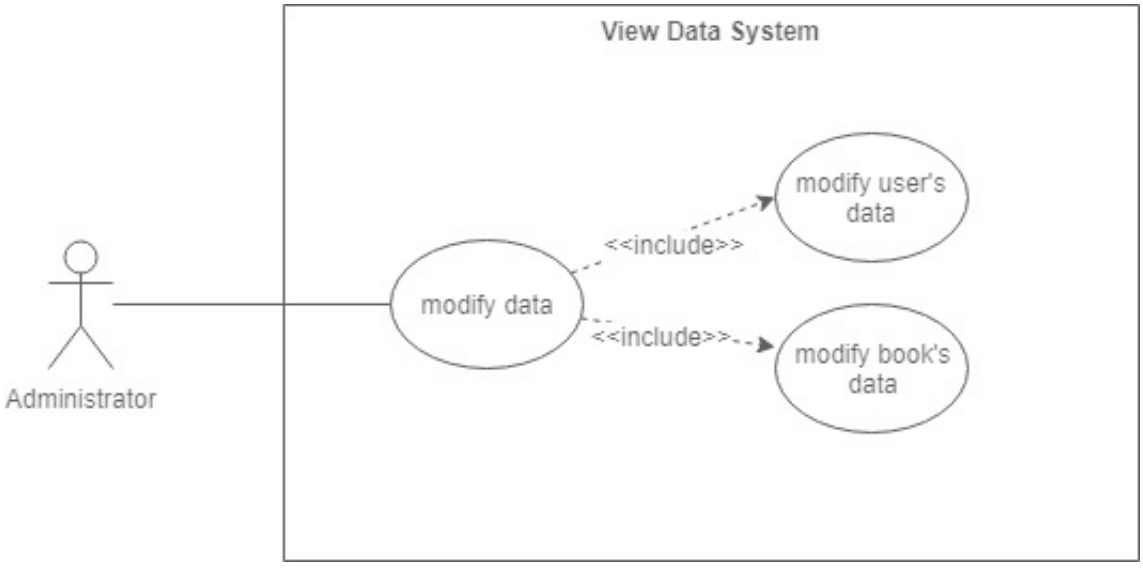
- Activity Diagram





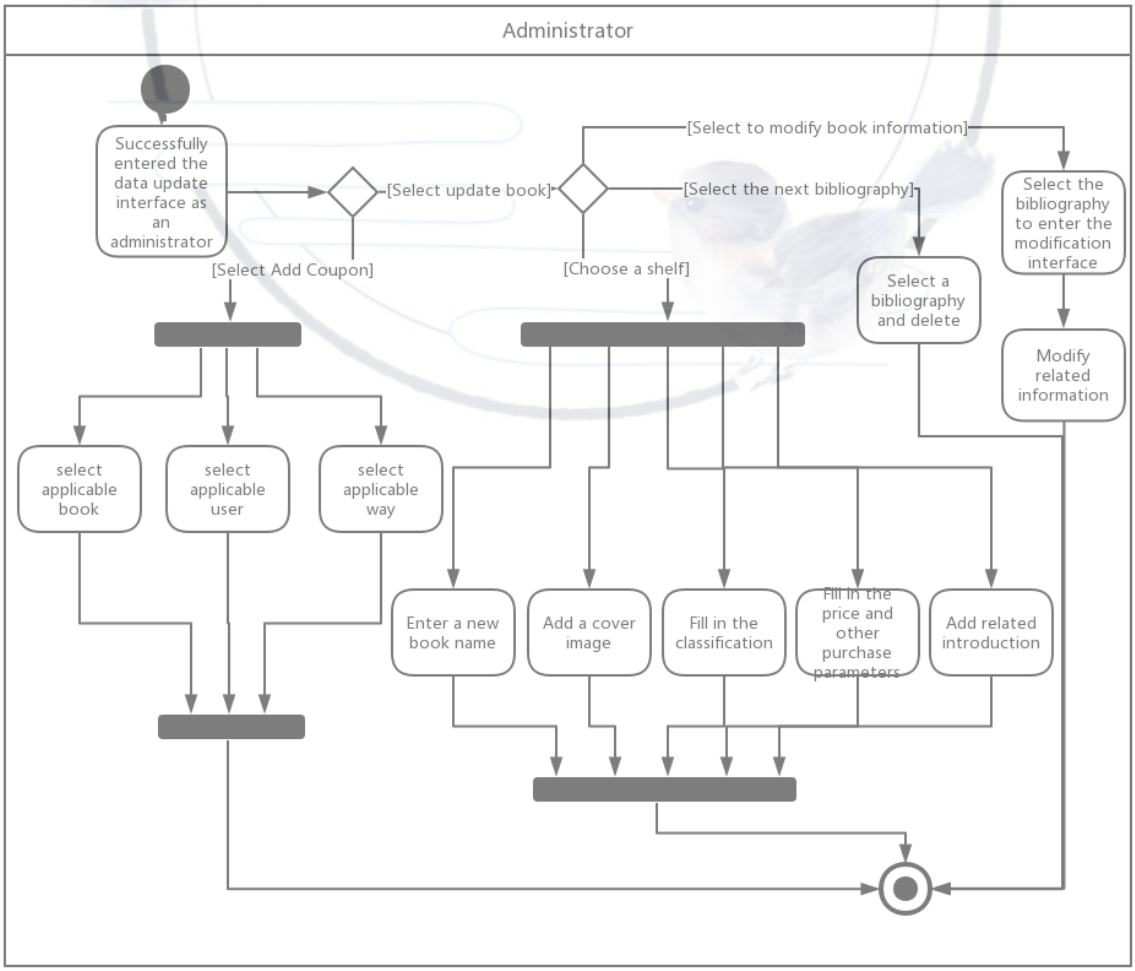
View Data Subsystem

- 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.

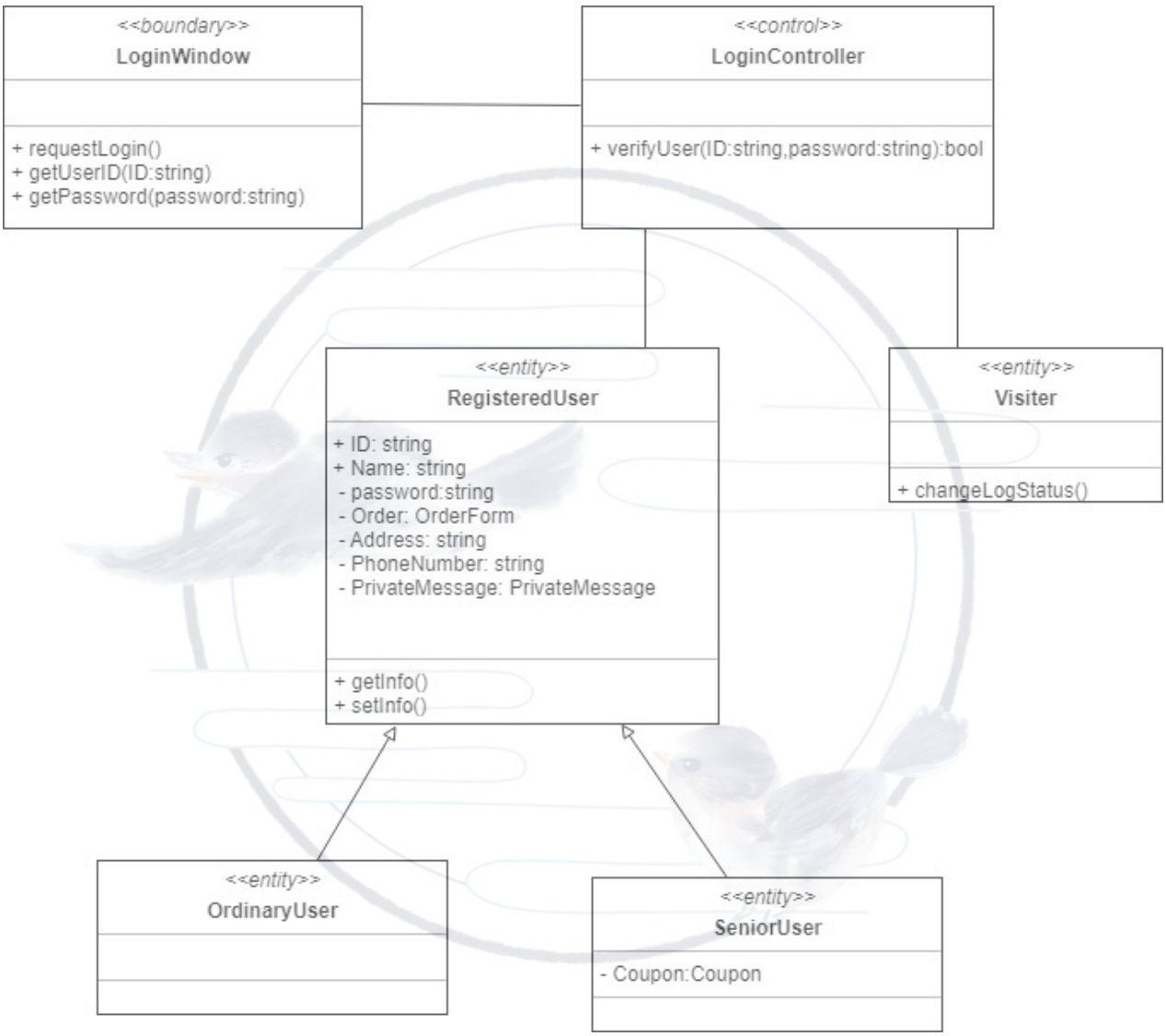
- Activity Diagram



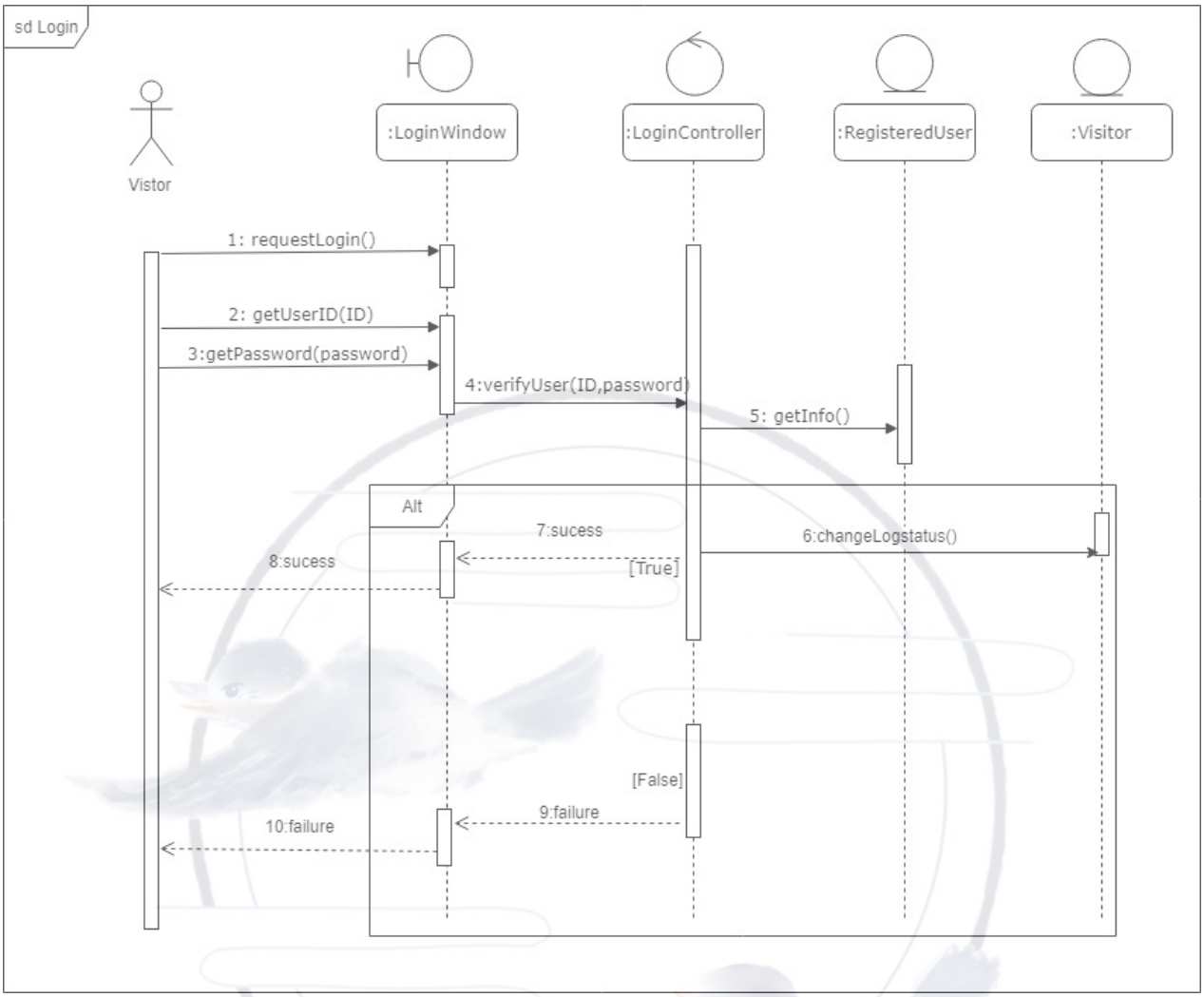
6.2 Analysis Model

6.2.1 Login

Class diagram

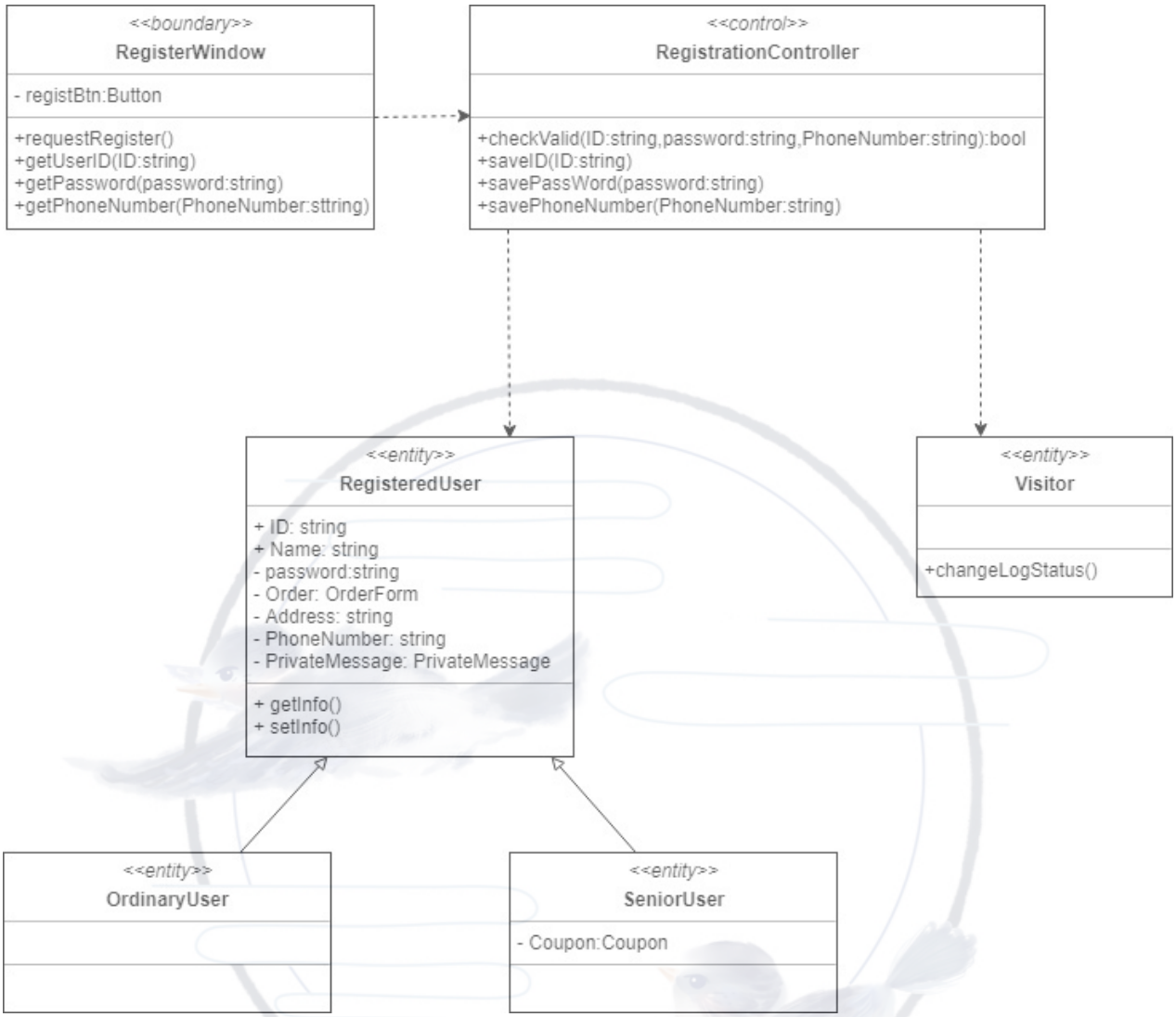


Sequence Diagram

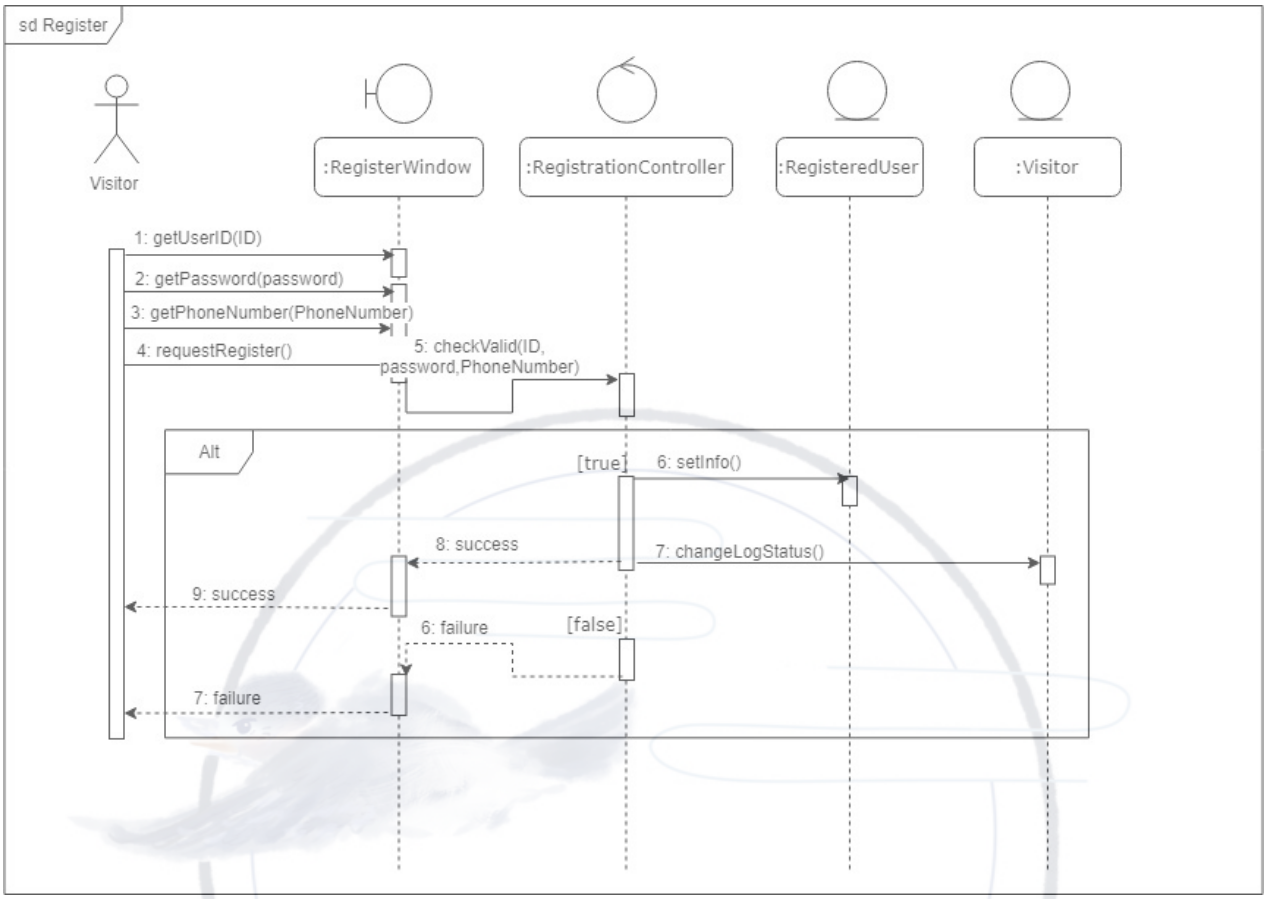


6.2.2 Register

Class Diagram

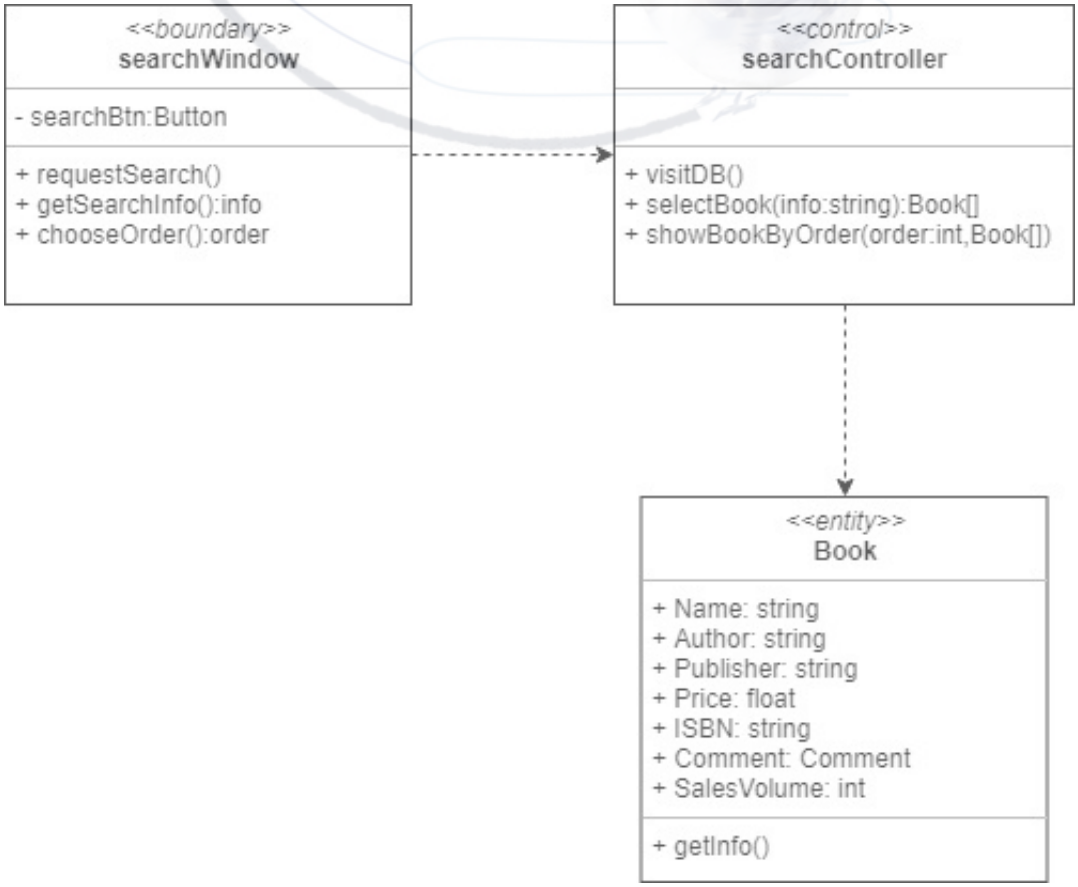


Sequence Diagram

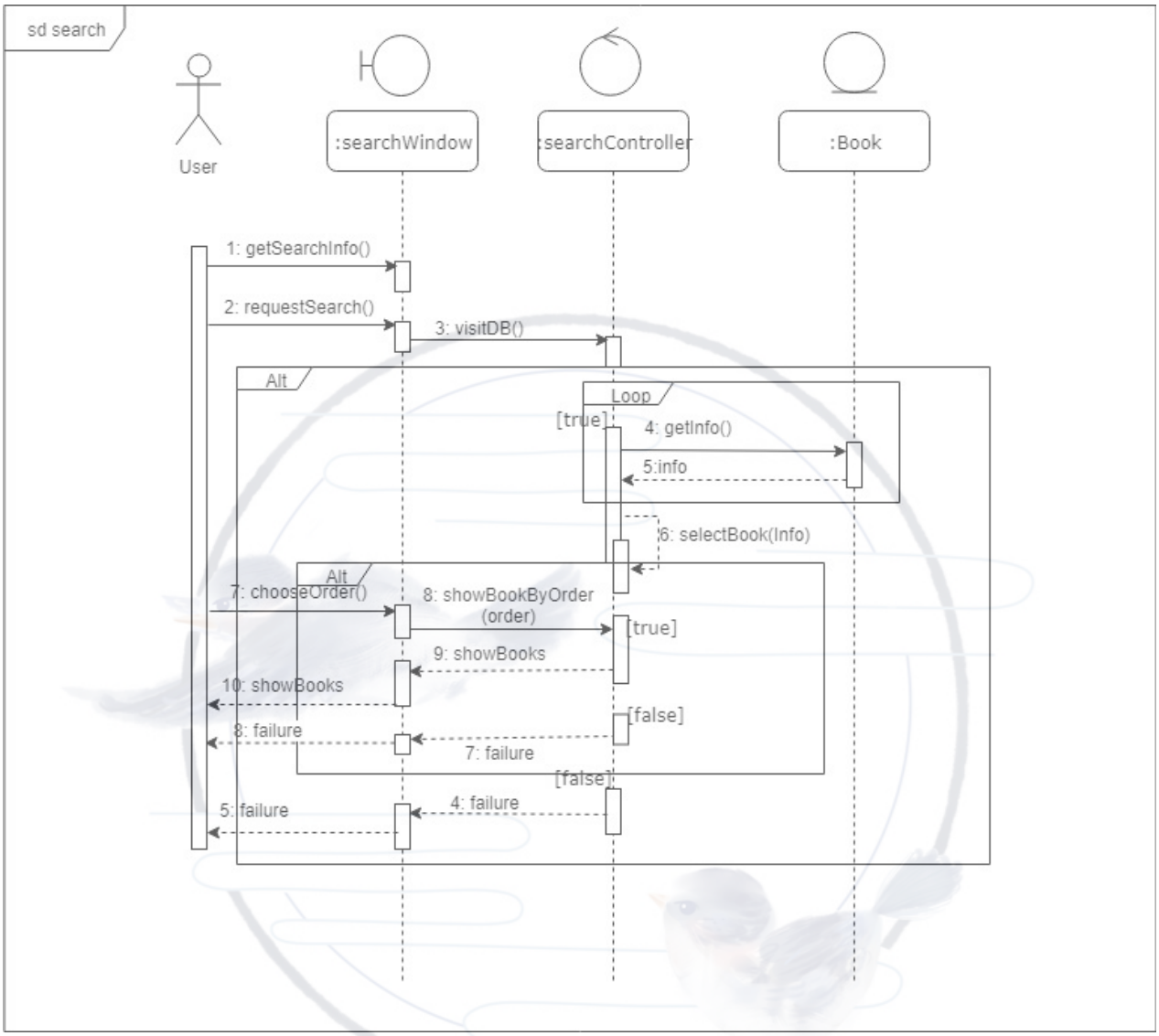


6.2.3 Search

Class Diagram

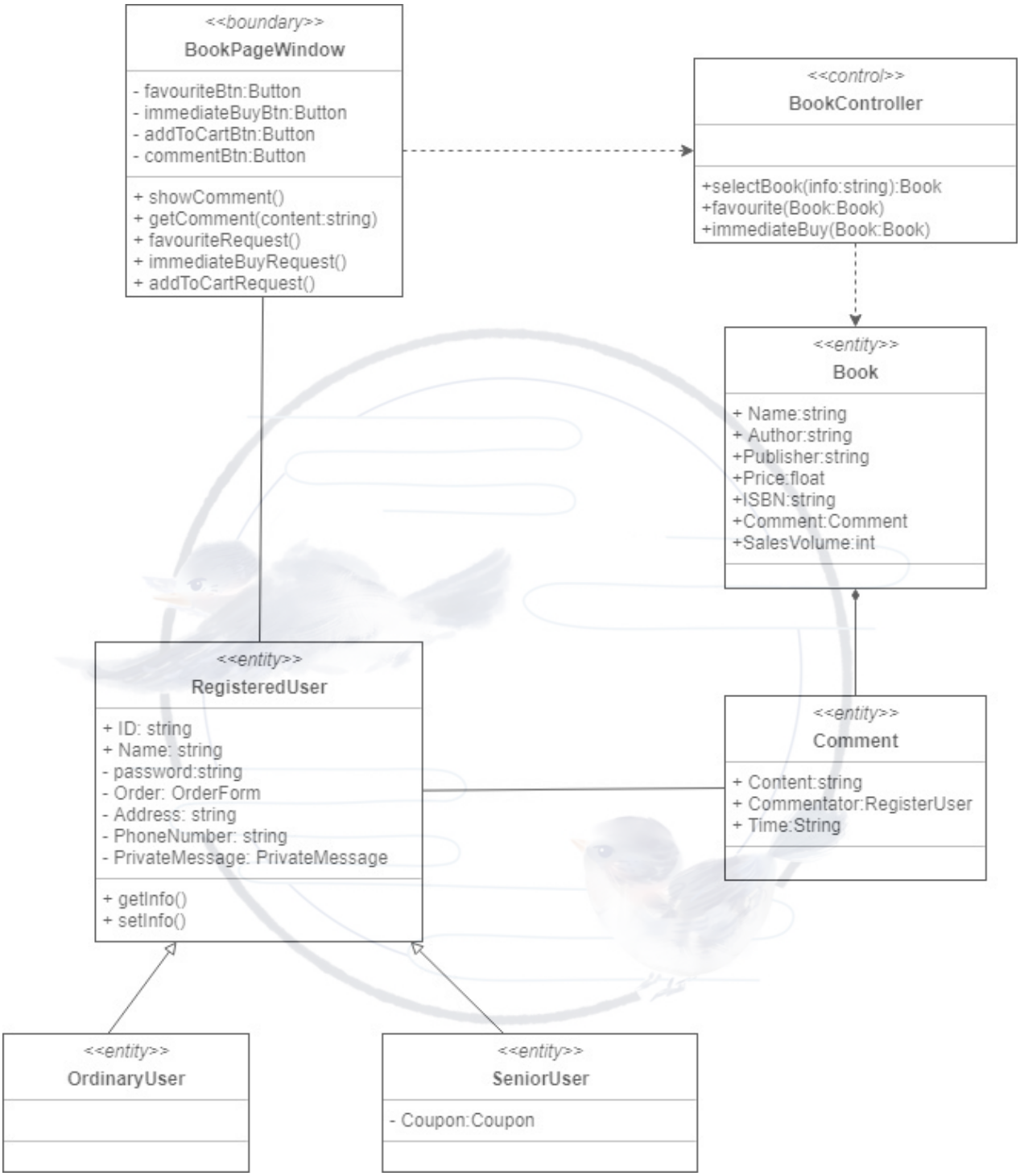


Sequence Diagram

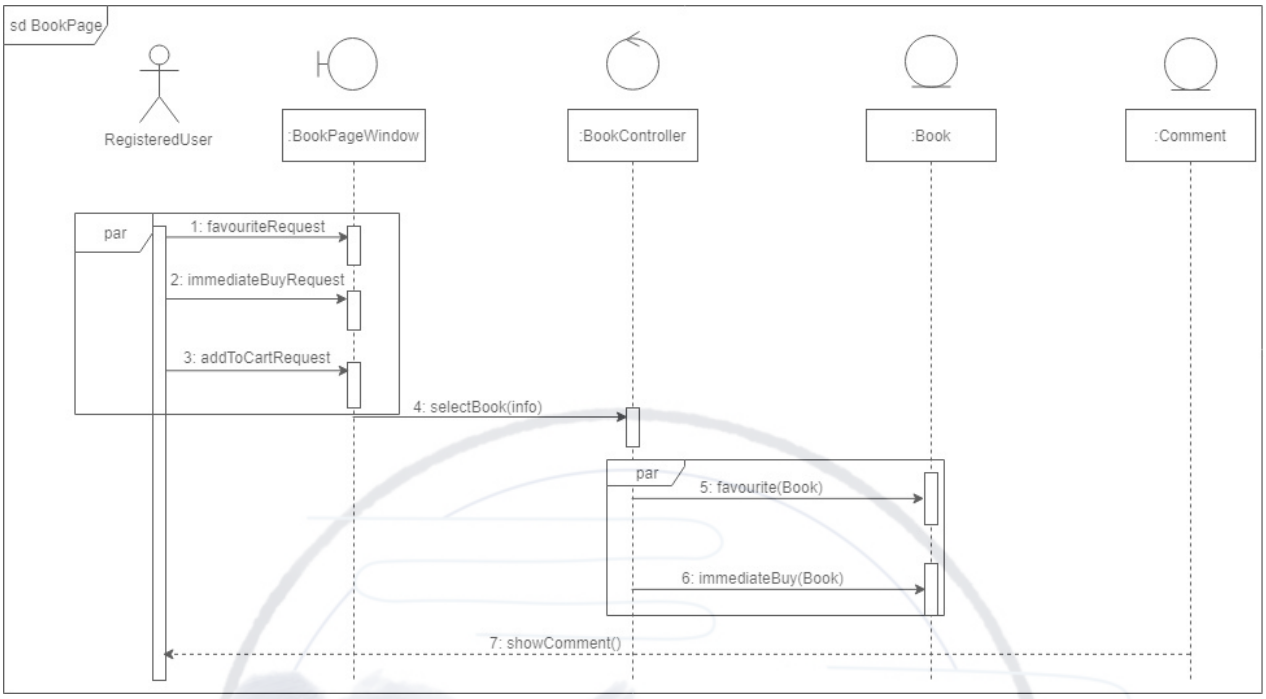


6.2.4 viewBookPage

Class Diagram

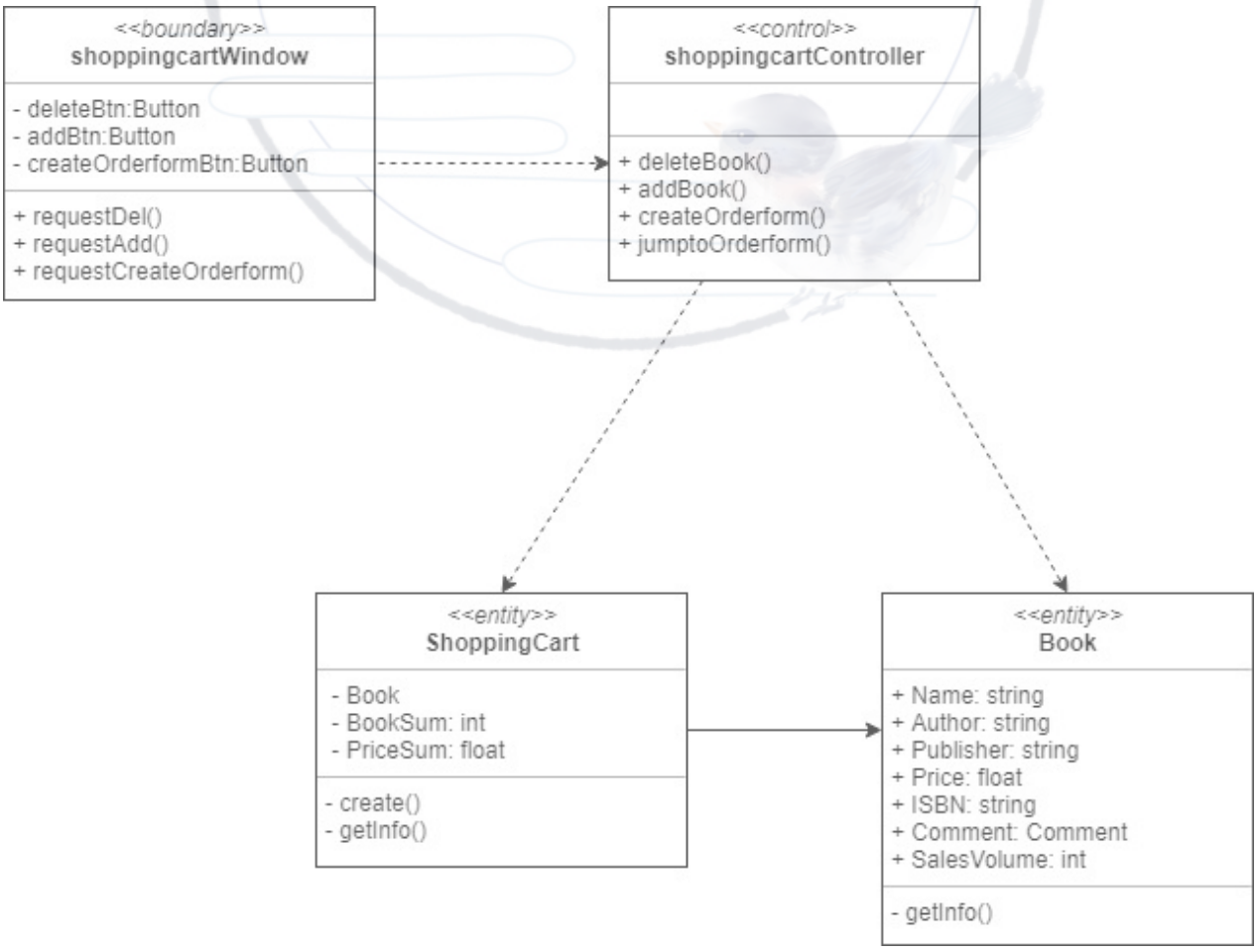


Sequence Diagram

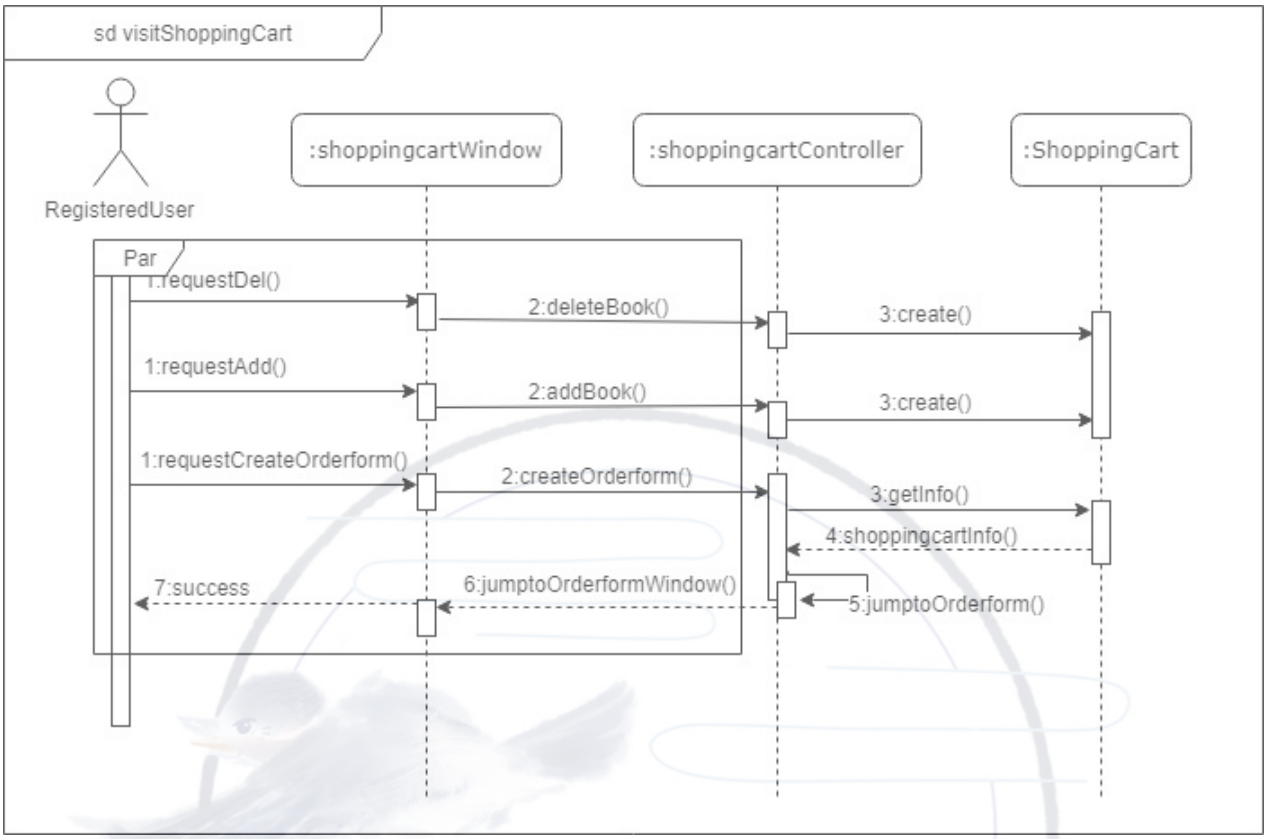


6.2.5 viewShoppingCart

Class Diagram

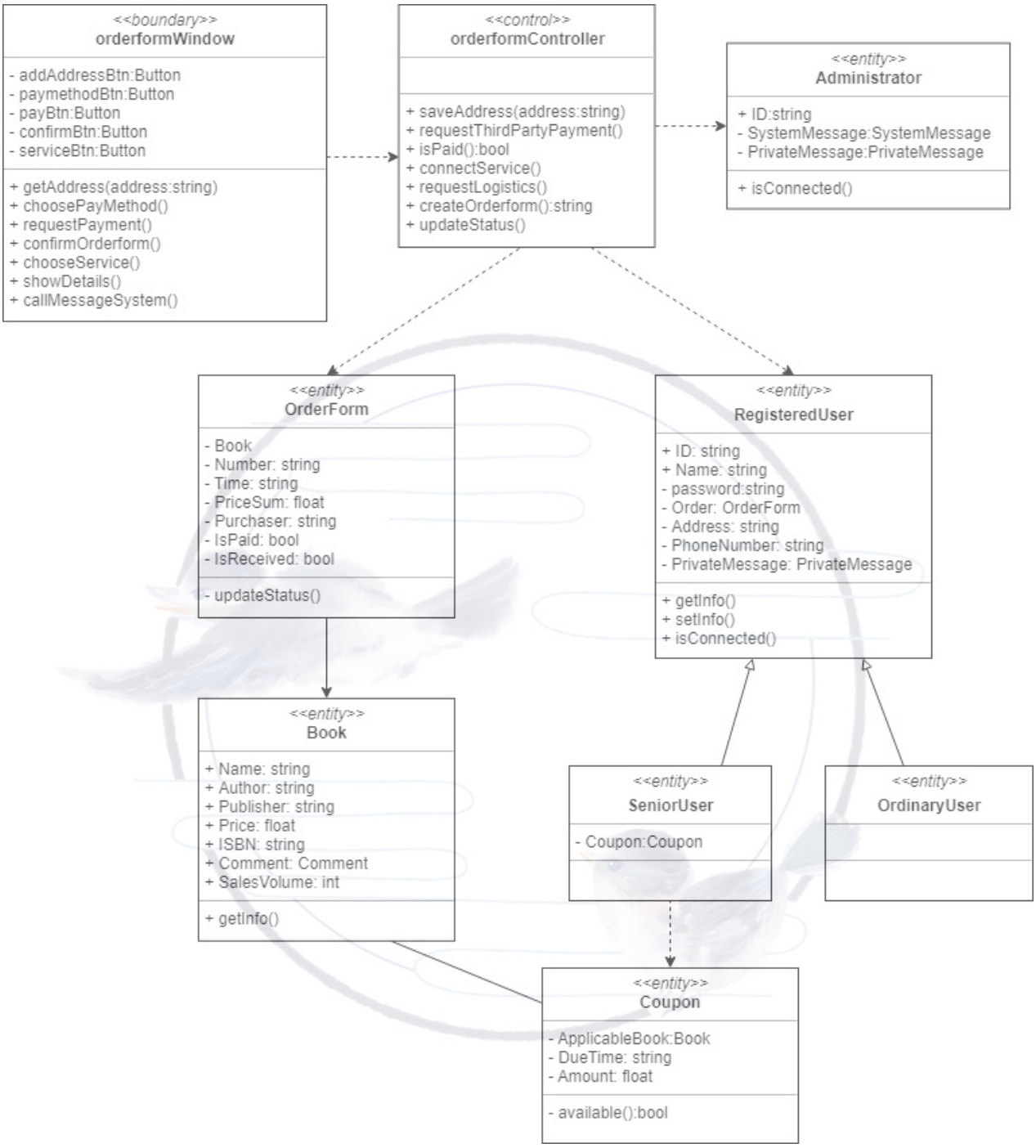


Sequence Diagram

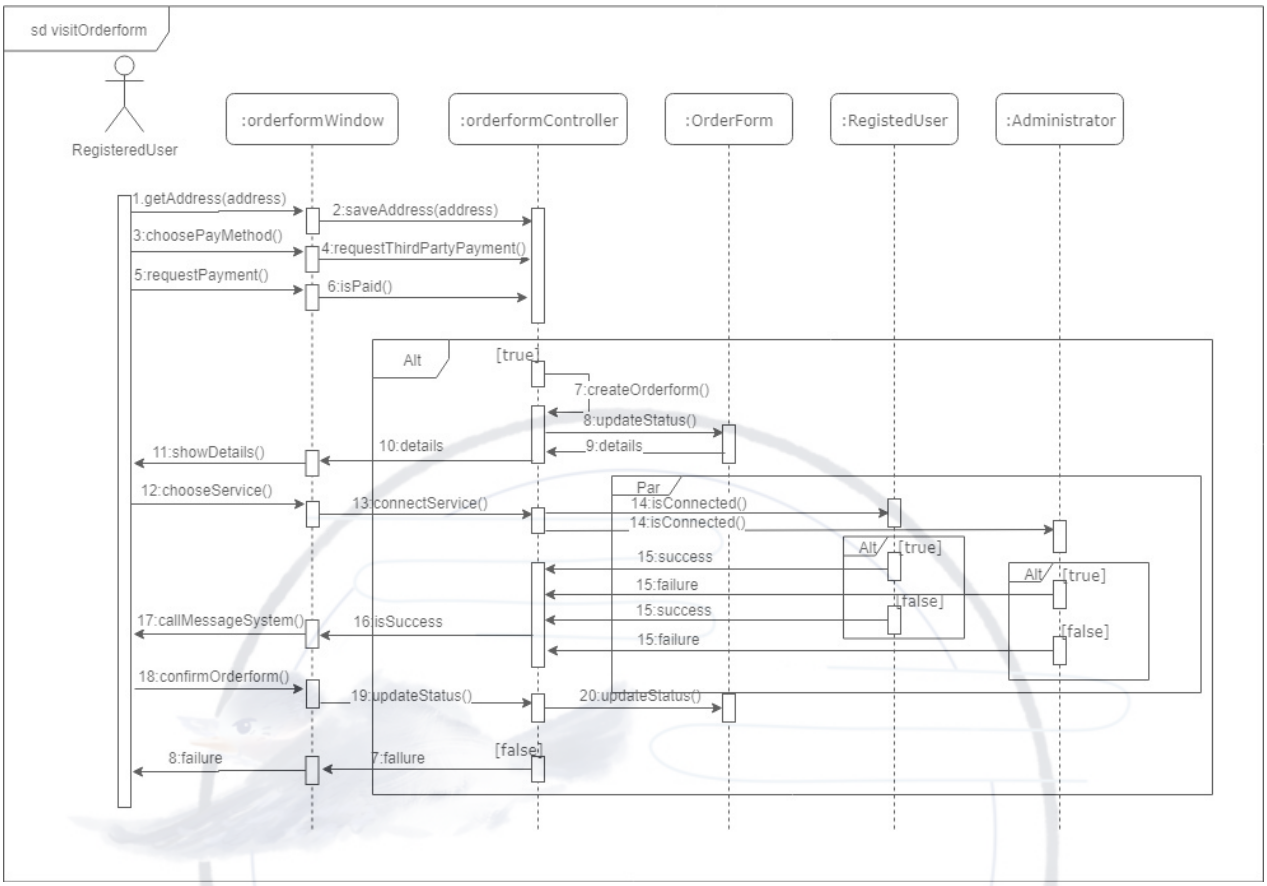


6.2.6 visitOrderForm

Class Diagram

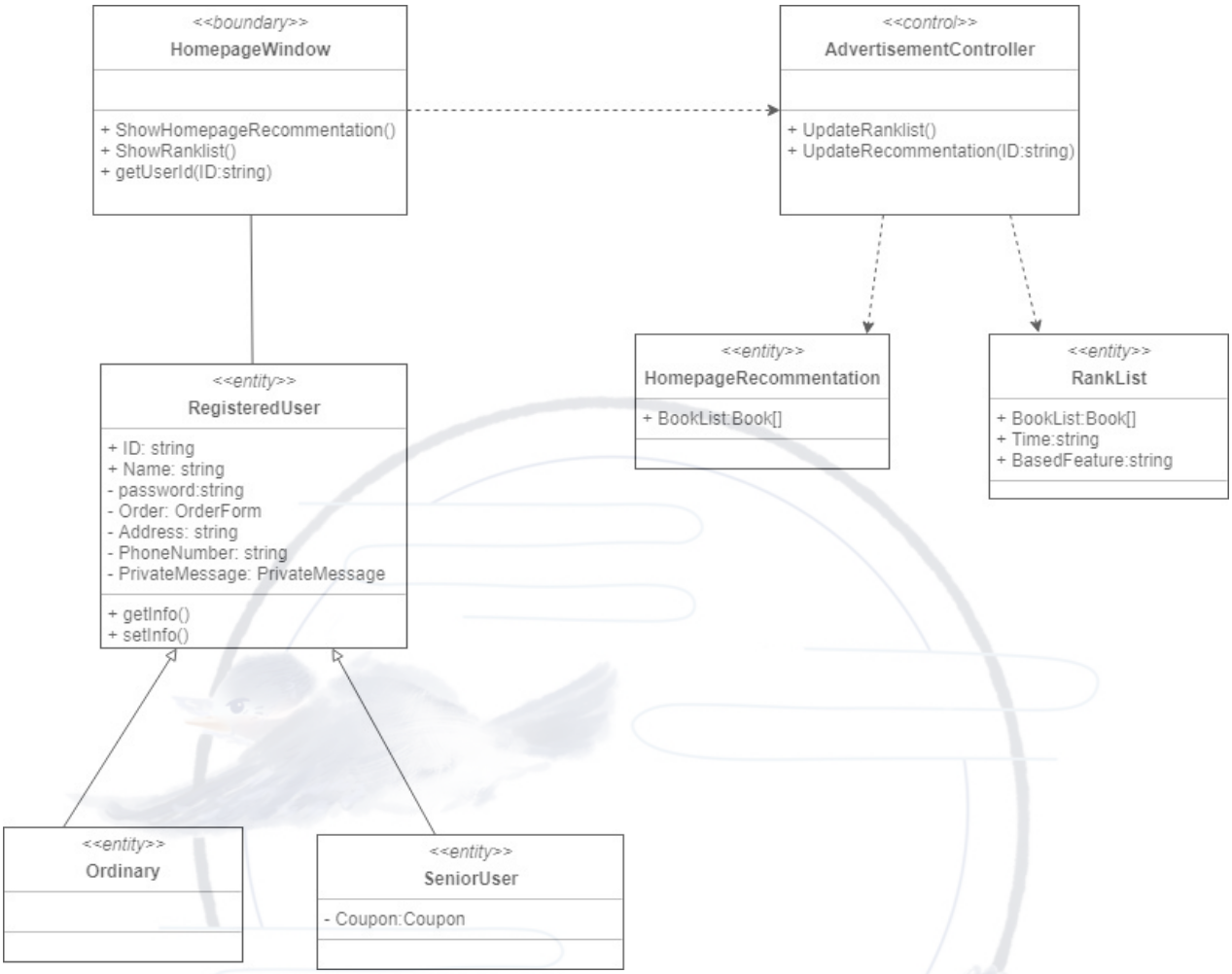


Sequence Diagram

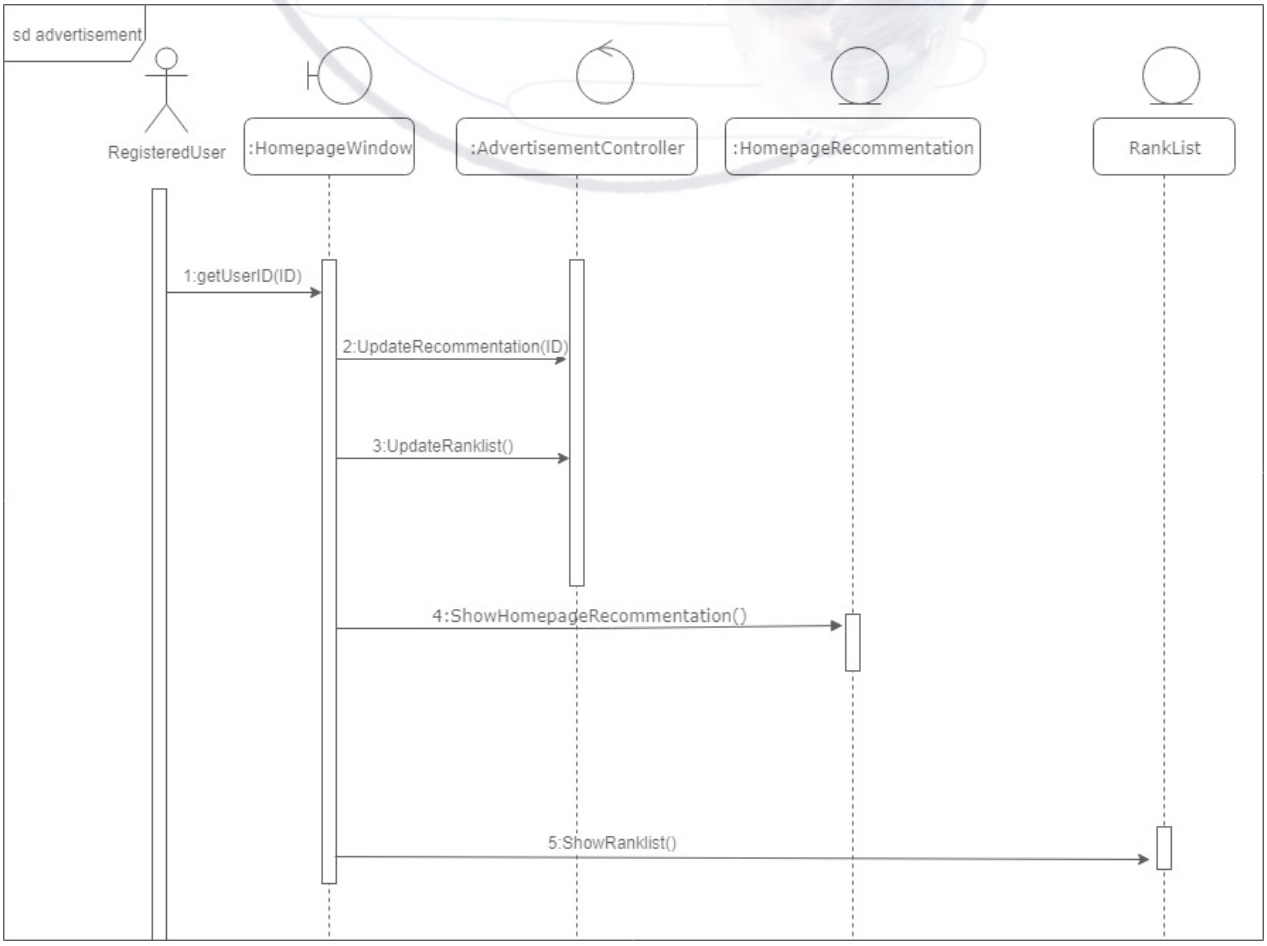


6.2.7 Advertise

Class Diagram

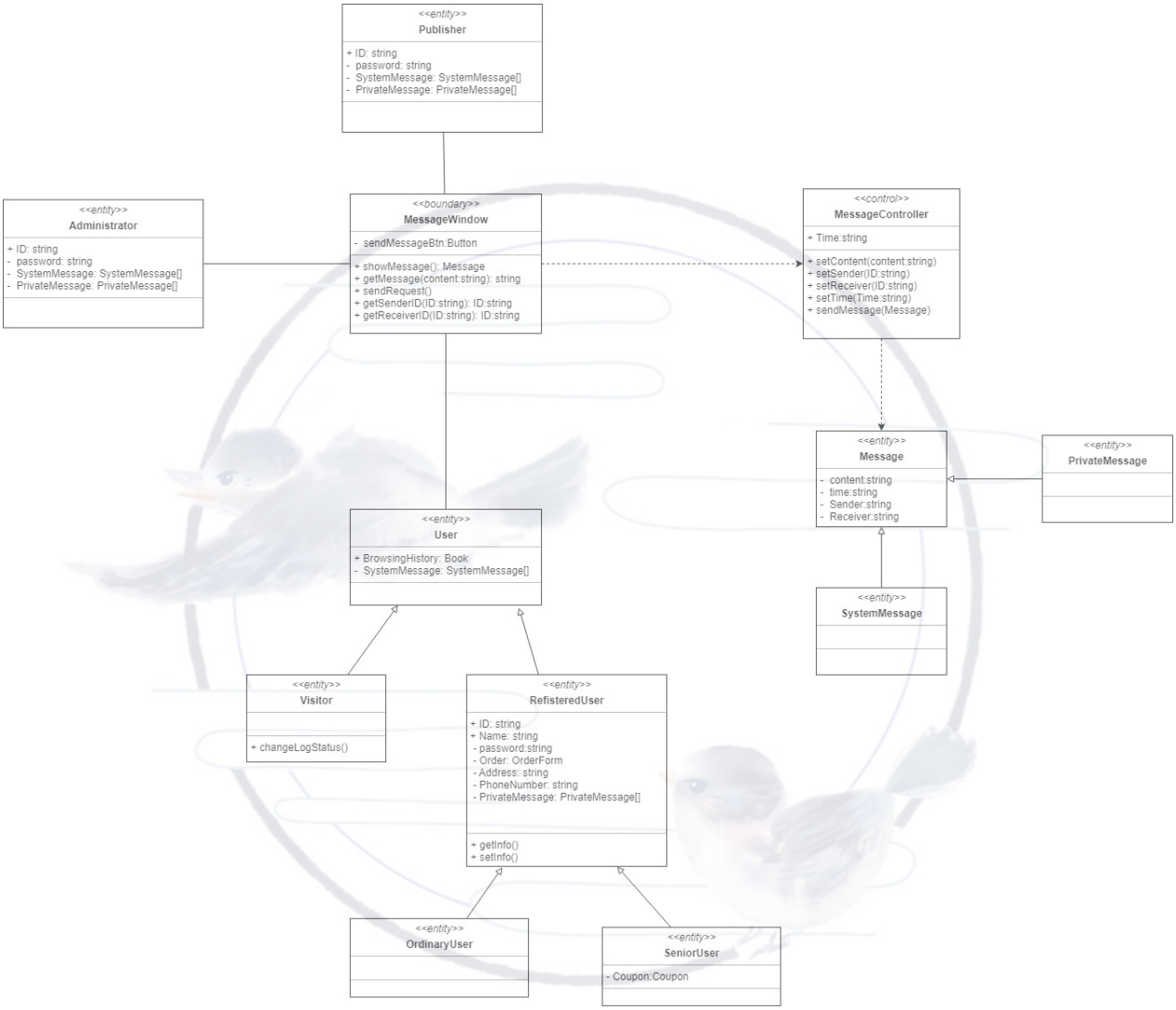


Sequence Diagram

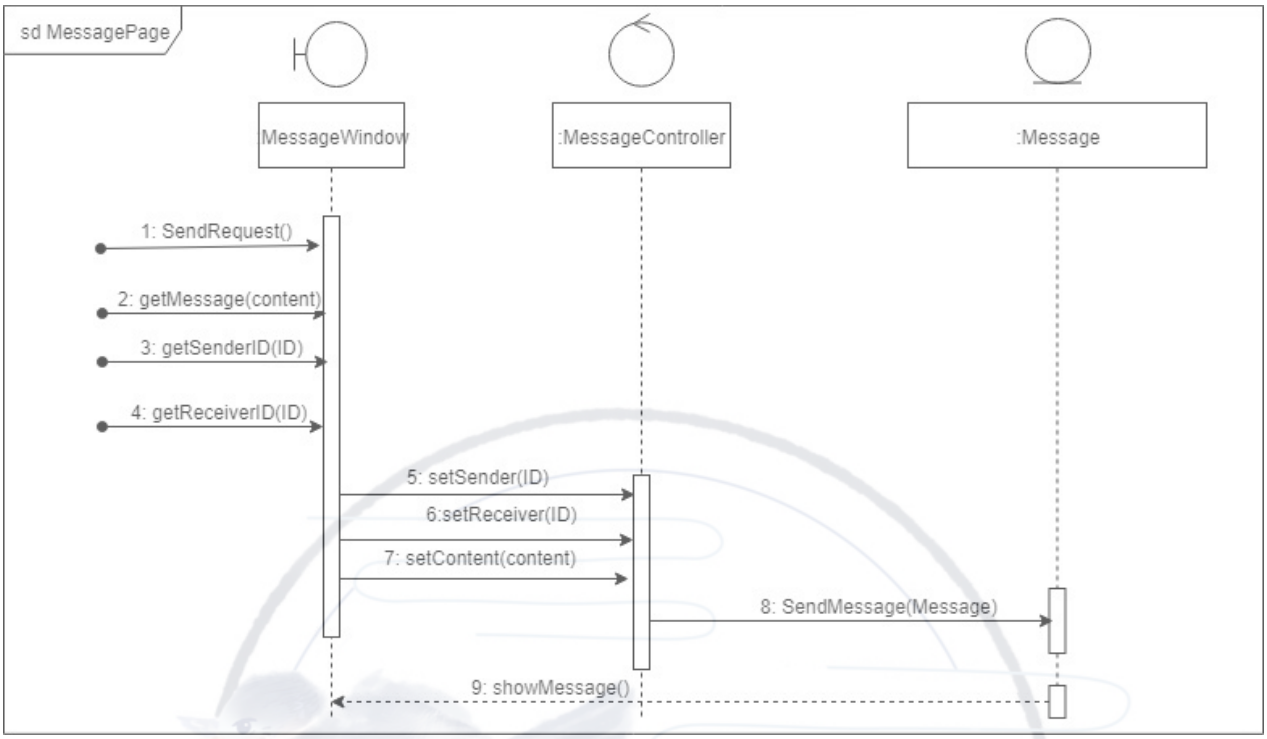


6.2.8 viewMessage

Class Diagram

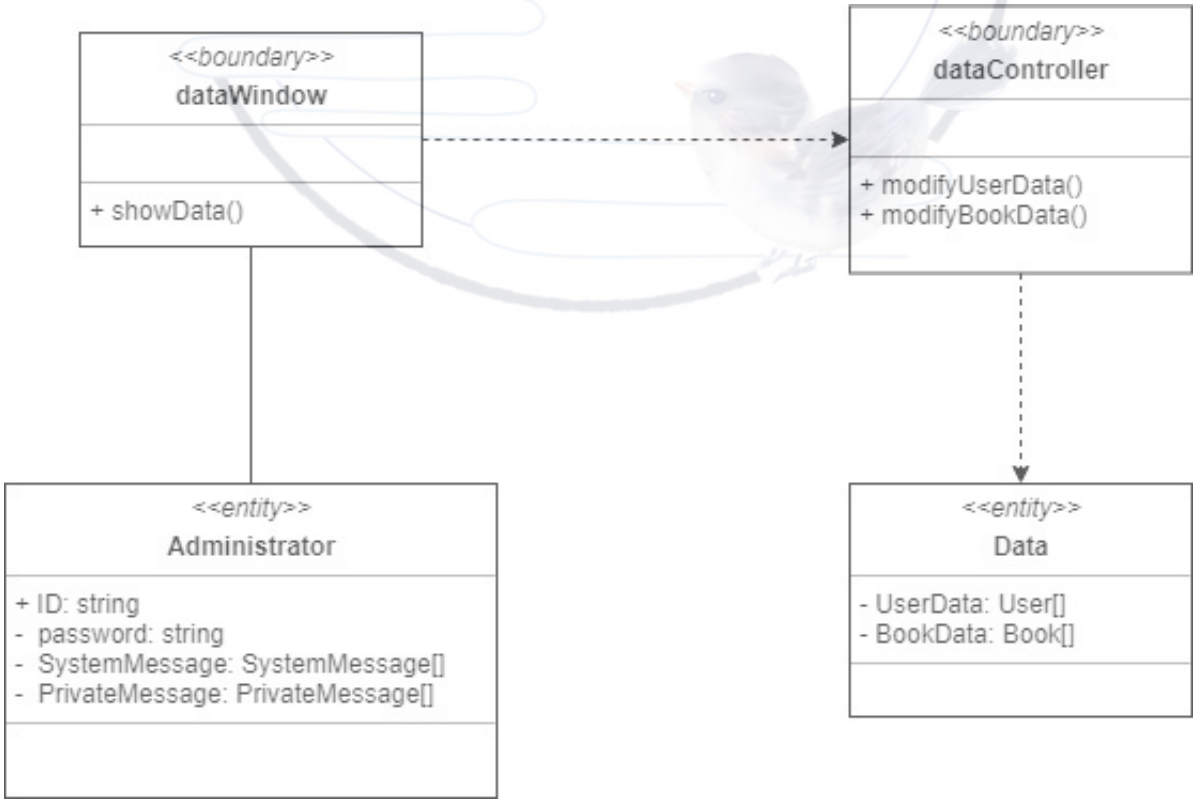


Sequence Diagram

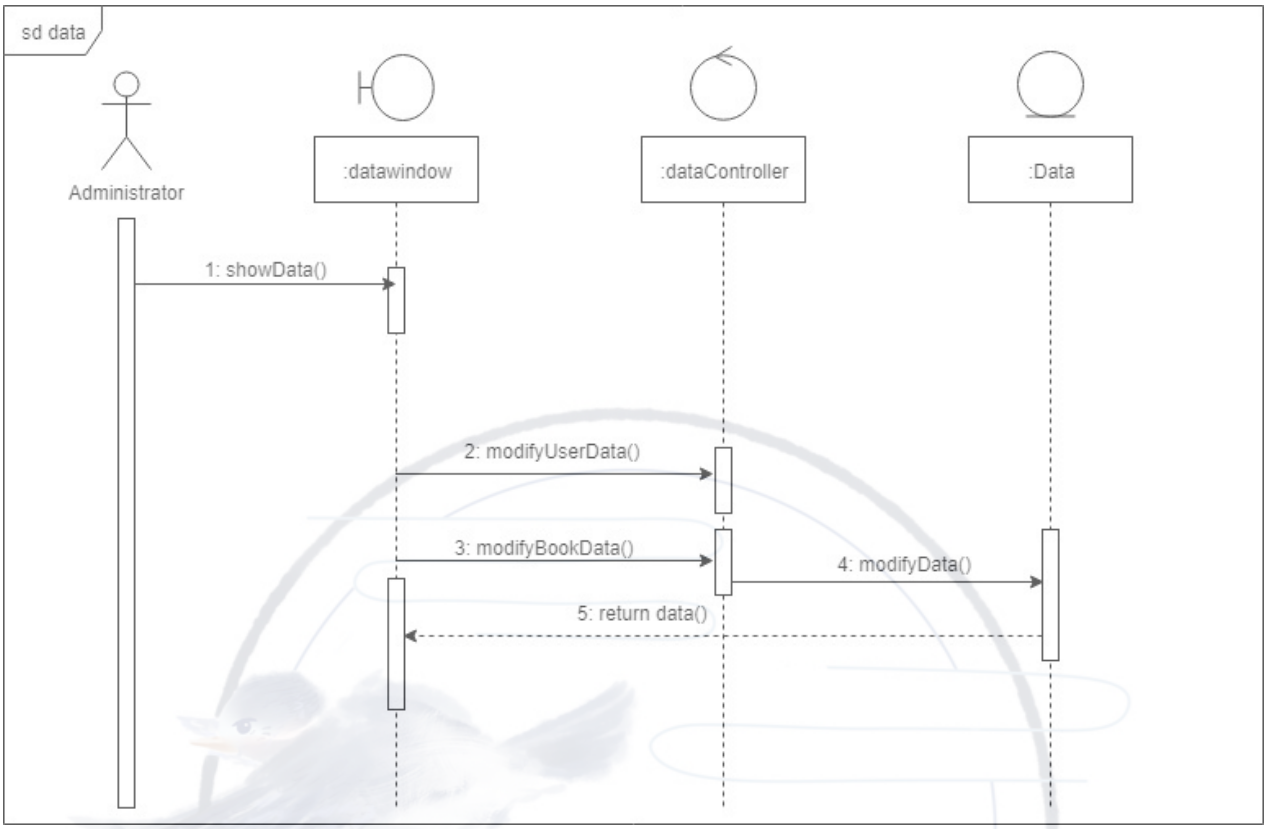


6.2.9 analysisData

Class Diagram



Sequence Diagram



7. Design Model

7.1 Architecture

Our **refined architecture** has four layers: Presentation Layer, Services Layer, Controller Layer and Data Layer, using Django Web framework and following the Models-Template- Views architecture pattern.

Considering the Presentation Layer, first we have Customer APP and Delivery APP, and they both have web version frontend to serve for customers, publishers and courier respectively. Management on the serve-side is based on web frontend.

Service Layer provides unified API interfaces of all functionalities for both web clients and mobile client, including unified API interfaces from Amap API to serve for Mapping System; and from Alipay, Wechat and Paypal Payment API to serve for Payment System, on both web and mobile clients. Order info interfaces employs EBusiness Platform and Taxation Authority.

In order to let upper layer get access to the database more clearly and easily, our Controller Layer is divided into three main controllers: User, Book, Order.There are Express Controller and Interaction Controller as well. Models are divided according to classification in subsystems, and are in correspondence with classes in database.

Data layer including document caching and data persistence, the former using Redis for its high performance; the latter using MySQL and MongoDB to store mass data.

7.2 Design Mechanism

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.

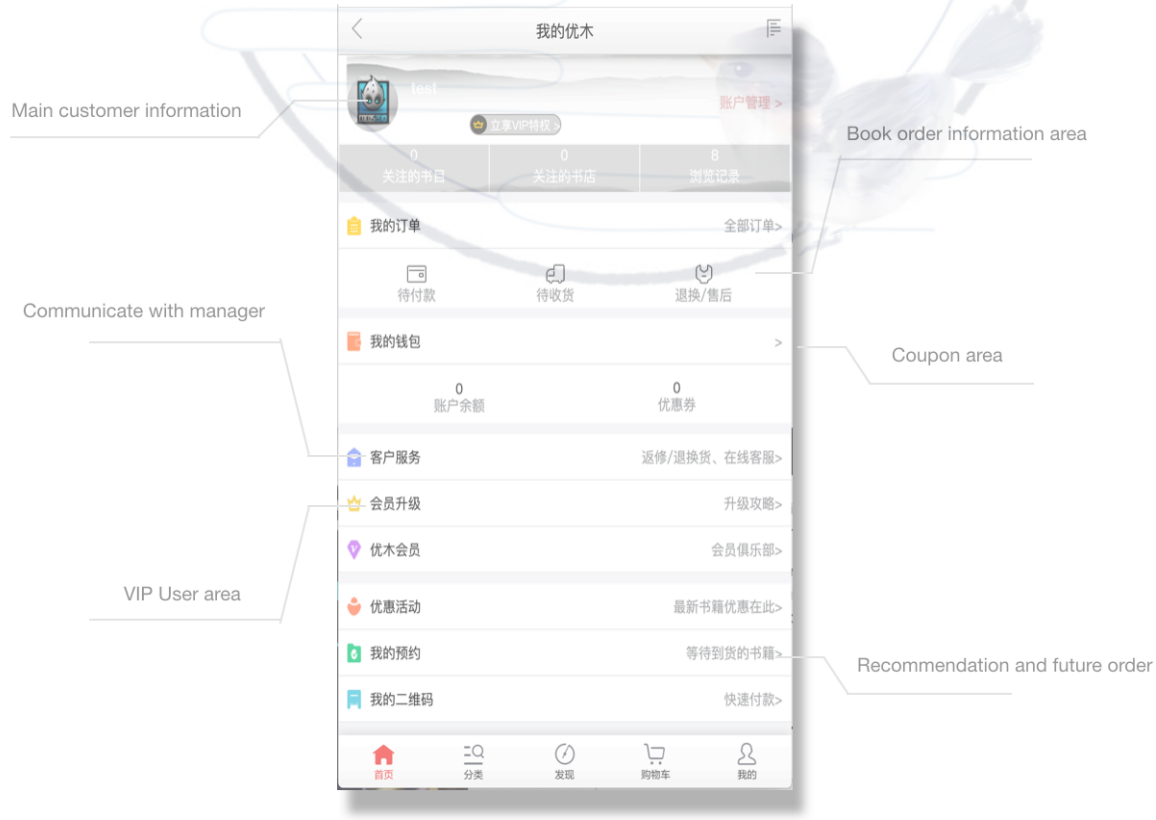
Another is the Information Exchange mechanism, using visitOrderform as interface specification. When user wants to have communications with administrators, request from APP and Web frontend will be distributed to the service layer through this communication.

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.

7.3 Detail Interfaces

7.3.1 Mobile Login

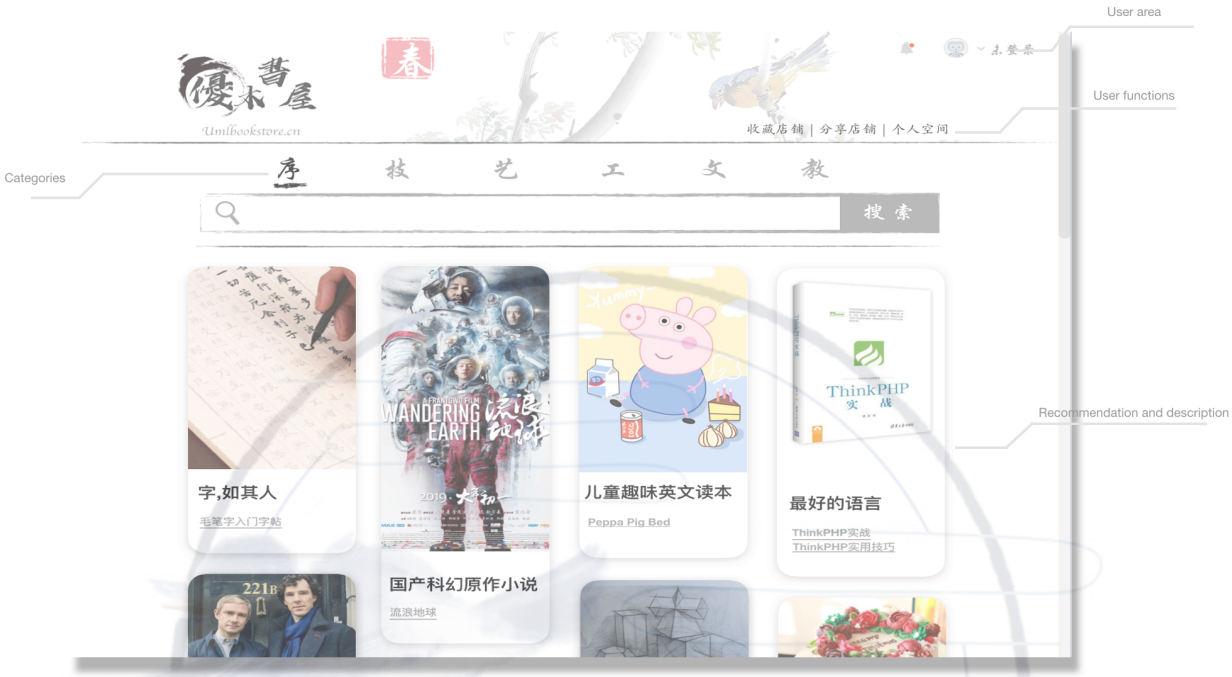
7.3.2 Mobile Pesonal Page



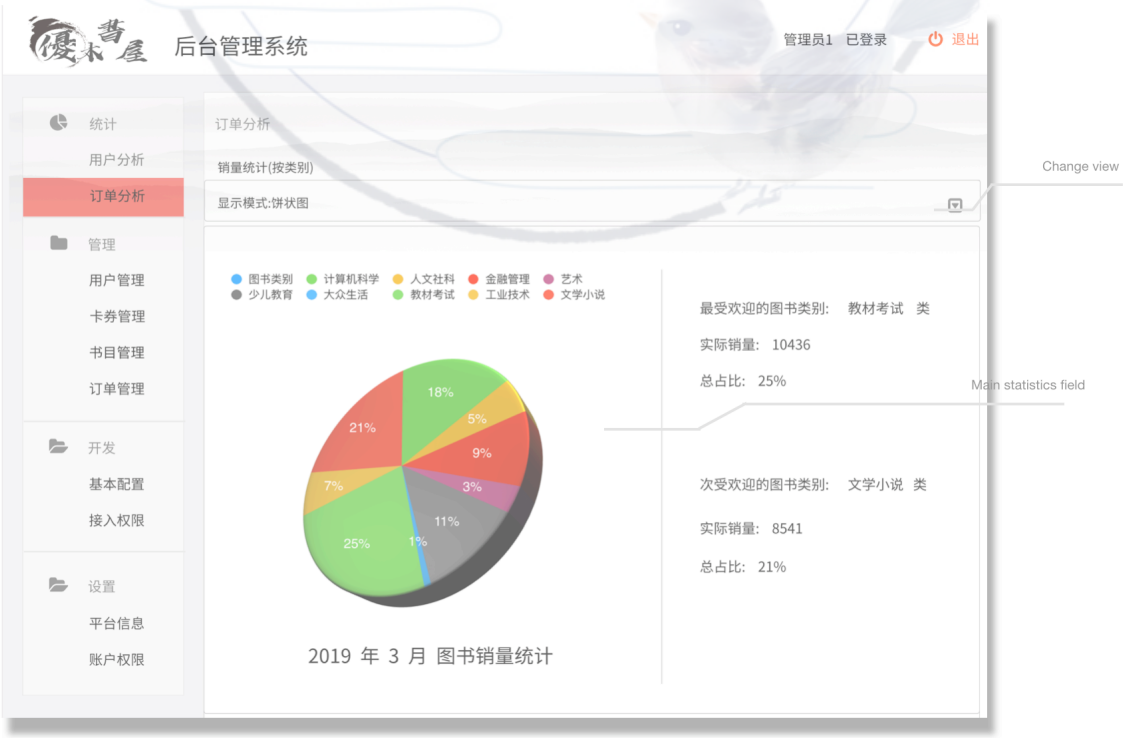
7.3.3 Moblie Shopping Cart

7.3.4 Moblie Payment

7.3.5 PC Main Page



7.3.6 Background statistics



7.3.7 Background Add Book

8. Attributes & References

- Attributes

杨乐(Le Yang): Update Use Case & Analysis Model, Design Mechanisms, Use Case Realization(Two Detailed Use case)

马思腾(SiTeng Ma): Update Use Case & Analysis Model, Design Mechanisms(Persistence, Communication), Use Case Realization, Update Overvies(Prupose, Definion, System Overview)

陈开昕(KaiXin Chen): PPT and Prepare for presentation, Organize Total Document

张喆(Zhe Zhang): Architectural Refinement(Global Architecture, Subsystem and Interfaces, Tow Detailed Subsystem), Organize Total Document

王舸飞(GeFei Wang): Interfaces, Prototyping(Model, Views, Template)

- References

https://sce.uhcl.edu/helm/RationalUnifiedProcess/process/workflow/ana_desi/co_amech.htm

http://plantuml.com/zh/sequence-diagram

https://www.tutorialspoint.com/uml/uml_architecture.htm

https://blog.csdn.net/fanxiaobin577328725/article/details/51700528

.etc