



VAASAN AMMATTIKORKEAKOULU  
UNIVERSITY OF APPLIED SCIENCES

Sicheng Che

DEVELOPING A SECOND-HAND  
ONLINE SHOPPING APPLICATION  
WITH JAVA AND ANDROID

Technology and Communication  
2021

## ACKNOWLEDGEMENTS

I would like to express my gratitude to all those who have helped me to turn these ideas into something concrete.

I also would like to express my special thanks to my supervisor, Dr. Ghordrat Moghadampour, who gave me the golden opportunity to develop this wonderful project on this topic, which also gave me brilliant suggestions and I came to know about so many new things. I am also thankful to the teachers at VAMK.

Any attempt at any level cannot be satisfactorily completed without the support and guidance of my parents and friends. They gave me a lot of encouragement to keep me going and finish the project.

I would like to thank the community on Stack Overflow, GitHub, CSDN and other platforms and website. I spent a lot of time on these resource learning platforms to complete this project.

Sincerely,

Vaasa, March 15, 2021

Che Sicheng

## ABSTRACT

Author	Che Sicheng
Title	Second-hand Online Shopping Application
Year	2021
Language	English
Pages	81
Name of Supervisor	Dr. Ghordrat Moghadampour

---

With the popularity of mobile phone, great changes have taken place in the way people access and save information. Mobile phones are inseparable parts of our lives and play a very important role in our lives. Meanwhile, with the improvement of people's lives, more and more second-hand items are ignored and cannot be reused well.

The motivation of the thesis was to design and develop an online second-hand shopping application, to provide a platform for people to deal with second-hand goods.

This second-hand online trading application was implemented to be used with Java and Android technology. The application consists of two parts, the client, and the server. The client is based on the Android platform and mainly includes registration, login, buy and sell own second-hand commodity, check orders, and other functions. The server is a web application and mainly includes user management, goods category management, second-hand commodity management and order tracking.

The program was tested on an Android mobile phone and different browsers with Google Chrome and Firefox.

# CONTENTS

<b>CONTENTS.....</b>	<b>4</b>
<b>1 INTRODUCTION.....</b>	<b>12</b>
1.1 BACKGROUND.....	12
1.2 OBJECTIVES .....	12
<b>2 RELEVANT TECHNOLOGIES.....</b>	<b>13</b>
2.1 ANDROID TECHNOLOGY .....	13
2.1.1 Basics of Android Platform .....	13
2.1.2 Android UI Introduction .....	14
2.1.3 Data Interaction between Android and Web Server .....	15
2.2 MVC PATTERN.....	16
2.3 JFINAL FRAMEWORK .....	17
2.3.1 MVC in JFinal .....	18
2.3.2 ActiveRecord in JFinal .....	20
2.4 REST API.....	20
2.5 JSON .....	20
2.6 DEVELOPMENT TOOLS .....	21
2.6.1 Eclipse.....	21
2.6.2 Android Studio.....	22
<b>3 APPLICATION DESCRIPTION.....</b>	<b>23</b>
3.1 GENERAL DESCRIPTION .....	23
3.2 QUALITY FUNCTION DEPLOYMENT.....	23
3.2.1 Must-have requirements .....	23
3.2.2 Should-have Requirements .....	24
3.2.3 Nice-to-have Requirements .....	25
3.3 USE CASE DIAGRAM.....	25
3.3.1 User Use Case Diagram.....	25
3.3.2 Administrator Use Case Diagram.....	26
3.4 CLASS DIAGRAM.....	31
3.5 SEQUENCE DIAGRAM .....	32
<b>4 SYSTEM DESIGN.....</b>	<b>39</b>
4.1 SYSTEM ARCHITECTURE DESIGN.....	39
4.2 DATABASE DESIGN .....	40

4.3	GRAPHICAL USER INTERFACE (GUI) DESIGN .....	42
4.3.1	Web UI Design .....	42
4.3.2	Android UI Design.....	45
<b>5</b>	<b>IMPLEMENTATION.....</b>	<b>54</b>
5.1	DATABASE CONNECTION IMPLEMENTATION .....	54
5.2	SERVER-SIDE IMPLEMENTATION.....	56
5.2.1	User Management .....	56
5.2.2	Category Management.....	57
5.2.3	Commodity Management .....	57
5.2.4	Order Management .....	58
5.3	CLIENT-SIDE IMPLEMENTATION.....	58
5.3.1	Login.....	58
5.3.2	Registration.....	60
5.3.3	Add Commodity Shopping Cart and Purchase .....	61
5.3.4	Calling Facebook .....	63
5.3.5	Rating and Give Comments.....	63
5.3.6	Sell Commodity .....	65
<b>6</b>	<b>TESTING.....</b>	<b>67</b>
6.1	WEB APPLICATION TESTING .....	67
6.1.1	Administrator Login .....	67
6.1.2	User Management.....	68
6.1.3	Category Management.....	69
6.1.4	Commodity Management .....	70
6.1.5	Order Tracking.....	71
6.2	ANDROID APPLICATION TESTING.....	72
6.2.1	New User Registration.....	72
6.2.2	Forget Password.....	73
6.2.3	User Login .....	73
6.2.4	Add to Shopping Cart .....	74
6.2.5	Add to Watchlist .....	75
6.2.6	Launch Facebook.....	75
6.2.7	Comments .....	76
6.2.8	Sell Commodity .....	77
6.2.9	Reset Password .....	78
<b>7</b>	<b>CONCLUSIONS .....</b>	<b>79</b>
7.1	FUTURE WORK.....	79

**REFERENCES..... 80**

**LIST OF FIGURS AND TABLES**

<b>Figure 1.</b> Android architecture diagram	13
<b>Figure 2.</b> Android UI architecture diagram	14
<b>Figure 3.</b> Android UI Display	15
<b>Figure 4.</b> Data Interaction Between Android and Web Server	16
<b>Figure 5.</b> JFinal Framework	18
<b>Figure 6.</b> MVC in JFinal	19
<b>Figure 7.</b> User Use Case Diagram	26
<b>Figure 8.</b> Administrator Use Case Diagram	27
<b>Figure 9.</b> Class Diagram 1	31
<b>Figure 10.</b> Class Diagram 2	32
<b>Figure 11.</b> Administrator Login Sequence Diagram	32
<b>Figure 12.</b> Administrator Modifies Users Sequence Diagram	33
<b>Figure 13.</b> Administrator Modifies Category Sequence Diagram	33
<b>Figure 14.</b> Administrator Modifies Commodity Sequence Diagram	34
<b>Figure 15.</b> Administrator Modifies Order Sequence Diagram	34
<b>Figure 16.</b> User Registration Sequence Diagram	35
<b>Figure 17.</b> User Login Sequence Diagram	35
<b>Figure 18.</b> User Retrieve Password Sequence Diagram	36
<b>Figure 19.</b> Collect Commodity to Watchlist Sequence Diagram	36
<b>Figure 20.</b> User Add and Checkout Shopping Cart Sequence Diagram	37
<b>Figure 21.</b> User Buy Commodity Sequence Diagram	37
<b>Figure 22.</b> Sell Commodity Sequence Diagram	38
<b>Figure 23.</b> System Architecture Diagram	39
<b>Figure 24.</b> Database ER Diagram	42
<b>Figure 25.</b> Login Page	43
<b>Figure 26.</b> Management System Homepage	43

<b>Figure 27.</b> Left-Vertical Navigation	44
<b>Figure 28.</b> User Management Section	44
<b>Figure 29.</b> Category Management Section	44
<b>Figure 30.</b> Commodity Management Section	45
<b>Figure 31.</b> Order Tracking Section	45
<b>Figure 32.</b> Launcher Icon	45
<b>Figure 33.</b> Login Layout and Design	46
<b>Figure 34.</b> Sign In Layout And Design	46
<b>Figure 35.</b> Forget Password Layout and Design	47
<b>Figure 36.</b> Main page and design	48
<b>Figure 37.</b> Launch Facebook	48
<b>Figure 38.</b> Profile Layout Design	49
<b>Figure 39.</b> Shopping Cart Layout and Design	49
<b>Figure 40.</b> Buy Layout and Design	50
<b>Figure 41.</b> Watchlist Layout and Design	50
<b>Figure 42.</b> My Order Layout and Design	51
<b>Figure 43.</b> Rating and Give Comment layout	52
<b>Figure 44.</b> Sell Commodity layout	52
<b>Figure 45.</b> Reset Password Layout and Design	53
<b>Figure 46.</b> Administrator Login Successfully	67
<b>Figure 47.</b> Administrator Modifying User Successfully	68
<b>Figure 48.</b> Administrator Modifying User on Android Testing	68
<b>Figure 49.</b> Administrator Modifying Category Successfully	69
<b>Figure 50.</b> Administrator Modifying Category on Android Testing	69
<b>Figure 51.</b> Administrator Modifying Commodity Successfully	70
<b>Figure 52.</b> Administrator Modifying Commodity on Android Testing	70
<b>Figure 53.</b> Administrator Modifying Order Successfully	71



	9
<b>Figure 54.</b> Modifying Order on Android Testing	71
<b>Figure 55.</b> Successfully Login Android App	72
<b>Figure 56.</b> Update in User Management	72
<b>Figure 57.</b> Retrieve the Password	73
<b>Figure 58.</b> Error Message Display Successfully	74
<b>Figure 59.</b> Successfully Add, Delete and Pay in Shopping Cart	74
<b>Figure 60.</b> Successfully Add and Remove Commodity to/from Watchlist	75
<b>Figure 61.</b> Launch Facebook	76
<b>Figure 62.</b> Comment Successfully	76
<b>Figure 63.</b> Upload Commodity Successfully	77
<b>Figure 64.</b> Upload Commodity Successfully in Commodity Management	77
<b>Figure 65.</b> Successfully Login After Resetting Password	78
<b>Figure 66.</b> Reset the Password Successfully in User Management	78
<b>Table 1.</b> User Management Module Analysis	27
<b>Table 2.</b> Commodity Management Module Analysis	28
<b>Table 3.</b> Category Management Module Analysis	29
<b>Table 4.</b> Order Management Module Analysis	30

**LIST OF CODE SNIPPETS**

<b>Code Snippet 1.</b> Get Request Method	54
<b>Code Snippet 2.</b> POST Request Method	55
<b>Code Snippet 3.</b> Parsing and conversion of JSON data	55
<b>Code Snippet 4.</b> Local Database Connection	55
<b>Code Snippet 5.</b> Delete user by Administrator	56
<b>Code Snippet 6.</b> Edit user by Administrator	56
<b>Code Snippet 7.</b> Add new category by Administrator	57
<b>Code Snippet 8.</b> Delete commodity by Administrator	58
<b>Code Snippet 9.</b> Changes the order status by Administrator	58
<b>Code Snippet 10.</b> Identify the Login User	59
<b>Code Snippet 11.</b> Login button	59
<b>Code Snippet 12.</b> Register A New User	60
<b>Code Snippet 13.</b> Add or collect commodity to Shopping Cart or Watchlist	62
<b>Code Snippet 14.</b> Get SharedPreferences Object	62
<b>Code Snippet 15.</b> Read Data from SharedPreferences	62
<b>Code Snippet 16.</b> Calling Facebook	63
<b>Code Snippet 17.</b> Rate the Received Order	63
<b>Code Snippet 18.</b> Give Comments	64
<b>Code Snippet 19.</b> Sell Second-hand Commodity	66

**LIST OF ABBREVIATIONS**

ADT	Android Development Tool
API	Application Programming Interface
APK	Android Application Package
App	Application Program
AVD	Android Virtual Device
B/S	Browser/Server
CSS	Cascading Style Sheets
Config	Configuration
DB	Database
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
JDK	Java Development Kit
JSON	JavaScript Object Notation
JSP	Java Server Pages
MVC	Model-View-Controller
ORM	Object Relational Mapping
SDK	Software Development Kit
SQL	Structured Query Language
SMS	Short Message Service
UI	User Interface
URL	Uniform Resource Locator
WIFI	Wireless Fidelity
XML	Extensible Markup Language

# **1 INTRODUCTION**

## **1.1 Background**

With improvement of life quality, more and more second-hand items are piled up at homes or offices and cannot be better reused. A platform needs to be designed and developed so that people can manage unused items by publishing information about the items they want to sell. At the same time, every registered user also can browse and search to buy what they need.

At present, second-hand online shopping applications for second-hand items based on the Android platform are still immature in the current software market, and most second-hand shopping applications are still traditional web applications. Therefore, it will be an innovation to design and develop a second-hand online shopping mobile phone application based on the Android platform. Due to the widespread use of mobile devices, the application will enable people to easily access required commodity information and perform convenient operations. When the Web application is not available, the mobile application can also help in outdoor areas.

## **1.2 Objectives**

The project, a second-hand online shopping system, is designed and developed based on Java and Android platforms. It is convenient for users to view second-hand commodity, search for commodity by category, and buy and sell own second-hand commodity. The most effective way to use the application is to install the application on the user's mobile device.

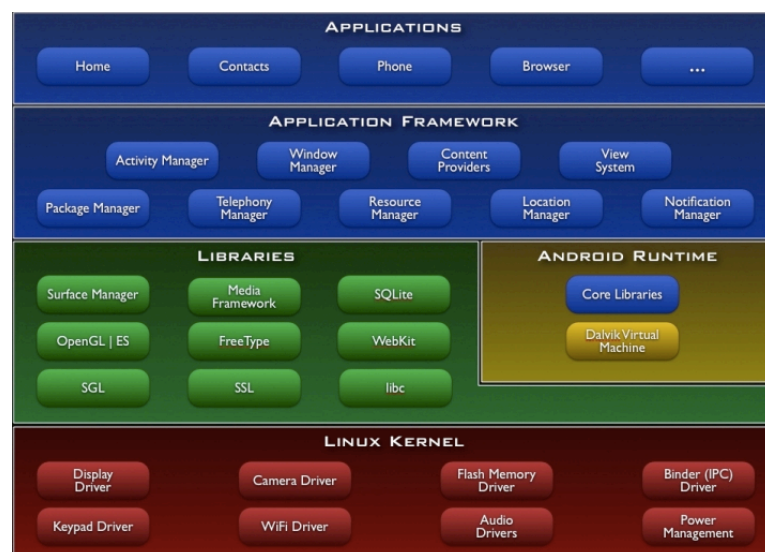
Another part of this project is the back-end management system, which can manage user, second-hand commodity and order information.

## 2 RELEVANT TECHNOLOGIES

### 2.1 Android Technology

#### 2.1.1 Basics of Android Platform

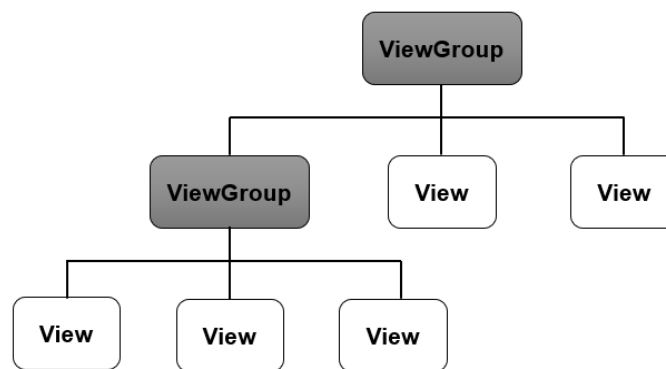
Android is a very widely used smartphone operating system and software platform, mostly used on handy mobile devices. Android is structured in a layered framework, with four layers from the top to the bottom, namely the Applications layer, the Application Framework, Libraries and Android Runtime, and the Linux Kernel. The Applications layer contains a variety of applications written in Java that interact directly with the user, such as SMS messaging clients, image galleries and web browsers. The application framework layer provides APIs for developers to develop in the application layer, enabling developers to develop applications quickly and efficiently. The system runtime library is what Android runs constitutes, specifically for the various components in the Android system. Android is based on the Linux operating system kernel and implements hardware device drivers, process and memory management, network protocol stacks, power management and wireless communication at the Linux kernel layer. The system is developed on the application layer of Android. The application involves the four main components of the application framework layer, Intent and Intent Filter and other related technologies, as well as the design of the user interface using Android's five main layouts. The Android system architecture diagram is shown in Figure 1. /1/



**Figure 1.** Android architecture diagram

### 2.1.2 Android UI Introduction

The Android UI provides its framework, with its main components being the home screen, navigation, and notifications. The application needs to ensure that they form an integral whole, which can be seen in detail in Figure 2.



**Figure 2.** Android UI architecture diagram

#### 1) Home Screen

It can be customized with application shortcuts, folders, and some small widgets. Switching between home screen panels is mainly done by swiping left and right. The most critical personal shortcuts are kept in the bottom favorites. The entire collection of applications and widgets is accessed by touching the All applications button in the center of the favorites. This is shown in Figure 3..

#### 2) All-applications

The All Applications page provides a specific view of all installed applications and window components, as shown in Figure 3.

#### 3) Switching

One of the main ways of switching between applications is through screen switching. This guides the user between the relevant tasks. The navigation bar on the right-hand side reveals which applications the user is interacting with. At the bottom, the most recent application is available to the user. Switching to the application takes place by touching it. Removing items is done by swiping to the left or right. This is shown in Figure 3. /2/



**Figure 3.** Android UI Display

### 2.1.3 Data Interaction between Android and Web Server

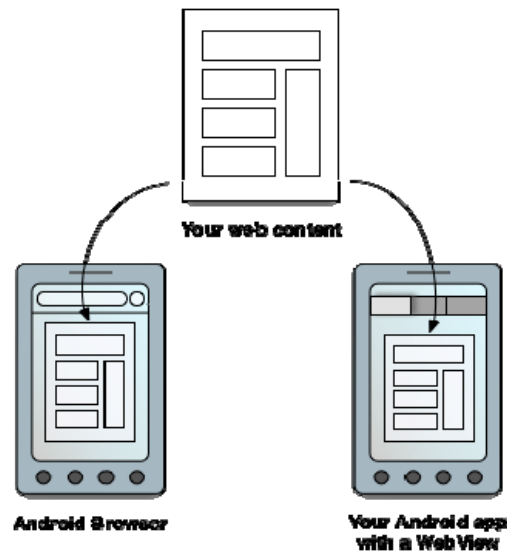
The main ways of distributing applications in Android are firstly, as a client application, which relies on the Android SDK to complete the development and is installed on the user's device with the .apk suffix; secondly, as a web application, which is developed using web standards and accessed via a web browser without installing anything. /3/

Several factors influence the final decision on the application approach, but the ability to support viewport attributes and such further simplifies web application development. In addition to this, it is possible to determine the appropriate size of the web application according to the screen size, provide images and different styles according to the screen resolution. /3/

In this case, the introduction of screen considerations can make web application development even less difficult. This is because the screens of all Android types can help to design the web pages well.

It is one of the features of the system that it is not necessary to design the application on the client side as well as the web side. It is possible to use both aspects together to develop the relevant client and to add web pages to the application. The diagram

below visualizes how web pages can be accessed from a web browser or an Android application. This is shown in Figure 4.



**Figure 4.** Data Interaction Between Android and Web Server

At the same time, it is not necessary to develop a separate Android application to reflect the website, so the design can be achieved in the following way. It is only necessary to define the appropriate interface to connect the interface to the Android application, and the API can be called via the interface JavaScript, because in this application it is possible to add interface content to the Android application via WebView and to add JavaScript to the Android API application. Additional features in the WebKit framework further enhance Android's support for screen density. This feature allows web pages to specify viewport attributes and modify image attributes and styles. Because these features are part of the Android WebKit framework, the Android browser (the default web browser provided by the platform) and WebView support the same viewport and screen resolution properties. /3/

## 2.2 MVC Pattern

MVC, short for View-Controller-Model, was created in the 1980s and is a widely used software design pattern today. It separates the input, output, and deal of an application, each being responsible for its own responsibilities.

The following is an introduction to the view, control and model layers of MVC:



### 1) View

View layer is the presentation of the web system and is the interface that connects the user to the system. The view can send requests to the model layer to query data or accept update data events passed from the model layer, thus synchronizing the user interface with the database updates, but the view layer cannot directly change the data in the model layer. After the view layer passes the request to the controller, the view does not need to care about the function calls that follow. But when the model layer is modified, the view layer responds to the model layer's changes.

### 2) Controller

The controller is responsible for receiving requests from the user, connecting the model and view layers, and bringing all three together to fulfil the user's request. When the user sends a request, the controller accepts the request but does not process the data. The controller is mainly responsible for receiving the request, calling the corresponding function module according to the request and finally calling the corresponding view to display the interface. /4/

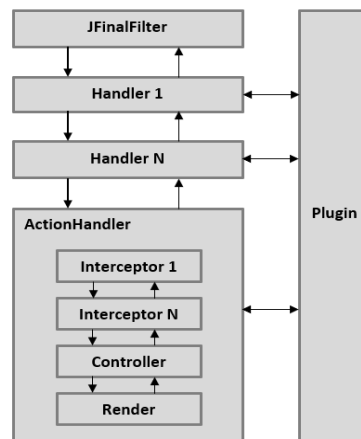
### 3) Model

The model is the core component of the pattern. It is the dynamic data structure of the application, independent of the user interface, and it directly manages the data, logic, and rules of the application. /5/ Data requests from the view layer are ultimately processed by the model layer, which in turn returns the data to the view layer, thereby updating the view layer and displaying the result data.

## 2.3 JFinal Framework

JFinal is an agile development based on the Java language WEB + ORM framework, its main design goal is rapid development, less code, simple and easy to learn, powerful, easy to extend and Restful. /6/

The framework consists of five main parts: Handler, Interceptor, Controller, Render and Plugin. Figure 5 shows the basic architecture of JFinal.



**Figure 5. JFinal Framework**

The main features of JFinal are as follows /6/:

- MVC architecture, ingenious design, simple to use.
- Convention over configuration, zero configuration, no XML
- Original DB and Record model pattern, making database development ultimate fast.
- Reloading modified Java files automatically, with no need to restart web server during development process.
- Plugin architecture with string scalability
- Complete functionality with most features of struts2

### 2.3.1 MVC in JFinal

The MVC layers in JFinal are described next.

#### 1) View

Support for normal jsp, html, but also velocity, CommonTemplate, freemarker, Smarty4j templates and other specialist display methods. These include, for example, jfreechart, JasperReport, and iReport.

#### 2) Controller

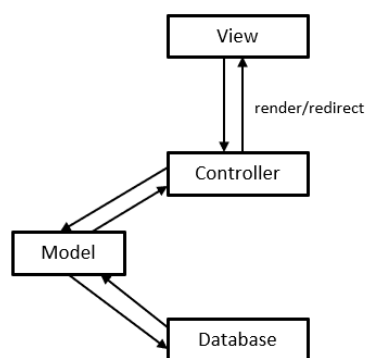
All requests are intercepted by the JFinalFilter and the Handler is called. The Handler receives all requests, including static requests, such as localhost/css/style.css or localhost/img/logo.jpg, and the Handler can change the resource to which the request is directed parameter, such as the String

target parameter. If the request is dynamic, it is handed over to the `ActionHandler`. Once in the `ActionHandler`, it will first get the `Action` object corresponding to the specific action based on the target parameters of the request mapped from the cached `ActionMapping`, which encapsulates the method name, the interceptor on the method. The controller and `Interceptor` where the method is located only intercepts requests for the action. The `ActionInvocation` is then composed based on the `Action` and the `Controller` instance and is then processed through the `invoke` in `ActionInvocation`.

This is an obvious implementation of the Command pattern. When the interceptor is called, the current controller's method corresponding to the request is invoked. Finally, `render` is responsible for rendering the corresponding page based on the current data, assembling the data into the required data format, then jumping to the specified page.

### 3) Model

JFinal model mainly completes data addition, modification, deletion, and query, like the Rails framework for database operations, mainly using a combination of `DB` and `ActiveRecord` for database operations. The main use of object-relational mapping ideas, through the reflexive operation of data tables, to complete the mapping of data table columns and class attributes, and support a variety of different types of database, different databases to generate different SQL statements. Such operations greatly simplify the database operations to improve the efficiency of system development.



**Figure 6.** MVC in JFinal

### 2.3.2 ActiveRecord in JFinal

ActiveRecord is a plug-in of the JFinal framework. It is one of the core functions of JFinal. The relational object mapping ORM is completed by ActiveRecord in JFinal. After the system is configured with the database, the data table is reselected by using object relational mapping. ActiveRecord supports many different types of databases and generates different SQL statements for different database types.

## 2.4 REST API

REST API (also known as RESTful API) is an application programming interface (API or Web API) that conforms to the constraints of the REST architecture style and can interact with RESTful Web services. REST stands for Representational State Transfer and was created by computer scientist Roy Fielding.

REST was used to describe the standard methods for creating HTTP APIs and found that the four common behaviors (view, create, edit, and delete) could be mapped directly to the GET, POST, PUT and DELETE methods already implemented in HTTP.

When a client makes a request via the RESTful API, it transmits a representation of the state of the resource to the requester or endpoint. This information or representation is passed via HTTP in one of several formats: JSON (Javascript Object Notation), HTML, XLT, Python, PHP or plain text. JSON is the most commonly used specialist programming language because, despite its name, it is language-agnostic and both humans and machines can both read it./7/

## 2.5 JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language. /8/

Any of the supported types can be represented by JSON, such as strings, numbers, objects, arrays and so on. However, objects and arrays are the two more specific and commonly used types.

Objects in JS are the contents of `{}` wrapped in brackets and have the data structure `{key1: value1, key2: value2, ...}` key-value pairs. In object-oriented languages, the key is the property of the object and the value is the corresponding value. The key name can be represented as an integer or a string. The type of the value can be of any type.

```
{ "firstName": "Brett", "lastName": "McLaughlin" }
```

Arrays in JS are the contents enclosed in square brackets `[]` and have the data structure `[ "java", "javascript", "vb", ... ]` of the index structure. In JS, arrays are a relatively special data type that can also use key-value pairs like objects, but it is the indexes that are used more often. Again, the type of value can be of any type.

```
{
  "people": [
    {
      "firstName": "Brett",    "lastName": "McLaughlin"
    },
    {
      "firstName": "Jason",   "lastName": "Hunter"
    }
  ]
}
```

## 2.6 Development Tools

### 2.6.1 Eclipse

The Eclipse source code is open in nature and contains only a framework and a set of services, however there is a standard set of plugins from which the development environment can be built, and it is a Java based extensible development platform.

To write Android projects based on this, the appropriate development environment needs to be formed first, in which the tool needs to be installed first, then the ADT

plugin on it, and then specify the correct path to the SDK must be specified, so that the Android virtual machine can be created. /9/

### **2.6.2 Android Studio**

Android Studio is an Android integrated development tool from Google, based on IntelliJ IDEA. Like Eclipse ADT, Android Studio provides integrated Android development tools for development and debugging. /10/

### 3 APPLICATION DESCRIPTION

#### 3.1 General Description

The main requirement of the application is to be able to allow users to complete online second-hand transactions on the platform. The application then needs a remote database to store user information, commodity information and order information. The system consists of a client side (Android application) and a server side (WEB application). The client and server side of the application must include the following functions.

- Client-side

The client is mainly operated by general users (not administrators). It includes functional modules, such as user registration, user login, commodity information, order information, logistics status checking and user profile modification.

- Server-side

The server side is mainly operated by the administrator. It includes functional modules, such as user information management, category management, commodity management as well as order dispatch and sign-off processing.

#### 3.2 Quality Function Deployment

Based on an analysis of the user's needs, the requirements of the application can be divided into three different categories to make the user experience better: must-have, should-have and nice-to-have.

##### 3.2.1 Must-have requirements

The must-have requirements of the application are listed below:

Client-side	<ol style="list-style-type: none"> <li>1. Register as a new user.</li> <li>2. Log in and out of the application.</li> </ol>
-------------	---

	<ol style="list-style-type: none"> <li>3. Display of commodity categories, commodity information.</li> <li>4. Users can make purchases.</li> <li>5. Display of individual user order information and viewing of logistics status.</li> </ol>
Server-side	<ol style="list-style-type: none"> <li>1. Login to the client application only by the administrator.</li> <li>2. There are four main functional sections: User management, Category management, Commodity management, Order tracking.</li> <li>3. Edit and delete users, commodity, and orders on request.</li> </ol>

### 3.2.2 Should-have Requirements

The should-have requirements of the application are listed below:

Client-side	<ol style="list-style-type: none"> <li>1. If a user forgets their password, they can retrieve it from the phone number they filled in during registration.</li> <li>2. The ability to find commodity by name keyword.</li> <li>3. The status of the commodity is "received", and the user can give some comments.</li> <li>4. Users can upload their own information about the used items they want to sell.</li> <li>5. Display of the results of the server-side's editing to improve the user experience.</li> <li>6. Users can add items to the shopping cart and watchlist.</li> </ol>
-------------	---



Server-side	Edit the information of Administrator.
-------------	--

### 3.2.3 Nice-to-have Requirements

The nice-to-have requirements of the application are listed below:

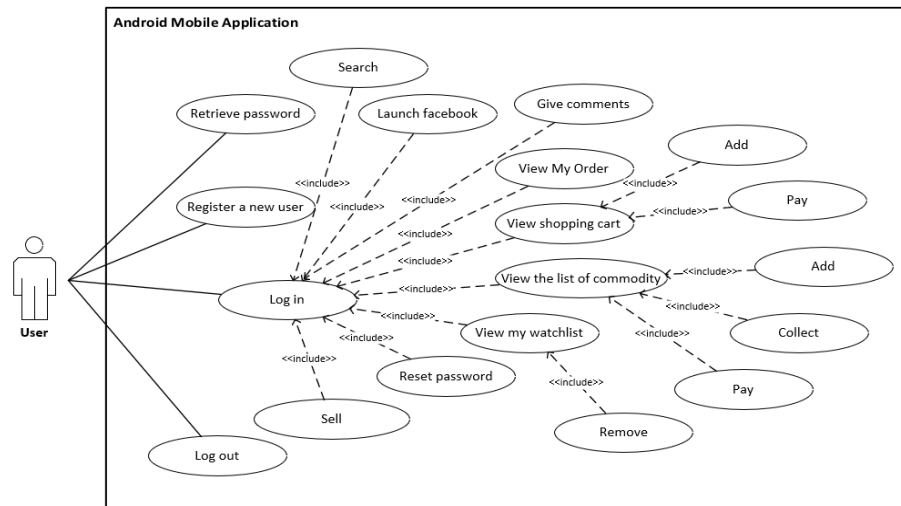
Client-side	<ol style="list-style-type: none"> <li>1. Communication between buyers and sellers by calling third party social networking software Facebook.</li> <li>2. Buyers can rate their purchases with rating, "Very Good" "Good" and "Not Good".</li> <li>3. Reviews of sold items can be displayed to all users.</li> <li>4. Users can add photos when uploading commodity.</li> <li>5. New users can add an avatar when registering.</li> <li>6. The user can reset password.</li> <li>7. Responsive UI</li> </ol>
-------------	--

## 3.3 Use Case Diagram

The system is divided into two main categories of users, namely users and administrators.

### 3.3.1 User Use Case Diagram

The user use case diagram is shown in Figure 7.



**Figure 7.** User Use Case Diagram

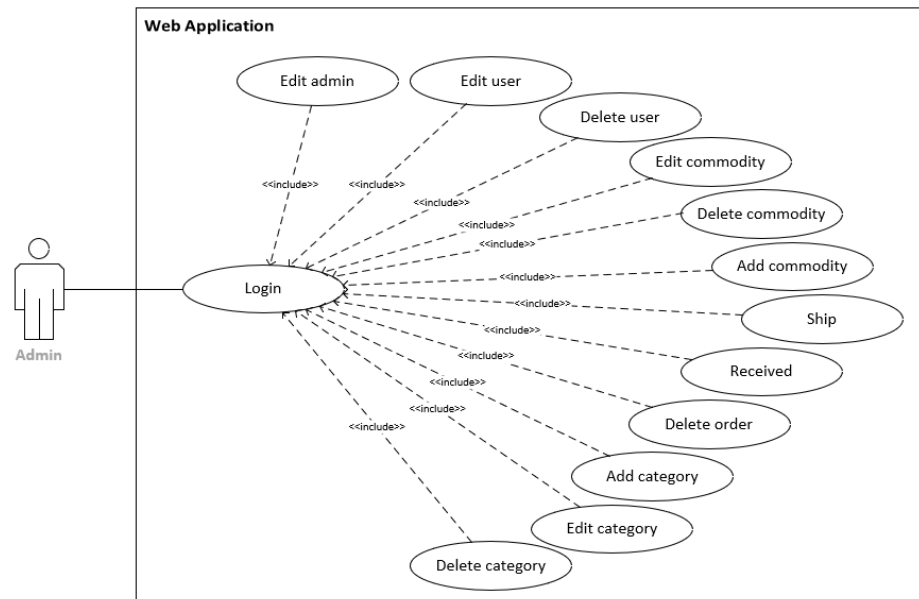
The user use case diagram shows the user's interaction with the system. The user needs to register and log in to the system. Users can register with their username and password and set their nickname and avatar. Once registered, the user can log in to the online second-hand trading application. Users do not have the right to modify the data within this application system. If the user forgets their password, they can retrieve it by using the phone number they filled in during registration.

Once logged in, the user can view information about all second-hand goods. The user can add commodity to the shopping cart, collect them and buy them. Users can also communicate with the seller by long clicking on the commodity and calling the third-party social networking software Facebook. In addition, users can view all order messages including the status of the delivery. After confirming receipt of the commodity, users can rate and review it.

Users can also upload their own second-hand items that they want to sell, which requires a photo of the item and other basic information. The password can also be reset.

### 3.3.2 Administrator Use Case Diagram

The manager (administrator) use case diagram is shown in Figure 8.



**Figure 8.** Administrator Use Case Diagram

Table 1 shows the user management module analysis.

**Table 1.** User Management Module Analysis

Case Name	User Management
Case Description	Management of user information such as editing and deleting.
Prerequisites	The system login successfully and the network connection is working.
Main Operations	<p>Editing User</p> <ol style="list-style-type: none"> <li>1) The administrator opens the system.</li> <li>2) Enter the system login screen and enter the set login account and password.</li> <li>3) Login to the system and enter the main system interface.</li> <li>4) Click on the User Management module to access the user management interface.</li> <li>5) Click the “Edit” button to edit existed user information.</li> <li>6) After filling the form, click the OK button to confirm and the user information will be edit in the database.</li> </ol> <p>Deleting User</p>

	<ol style="list-style-type: none"> <li>1) In the User management interface.</li> <li>2) Click “Delete” button to delete to the user you want to delete.</li> <li>3) Click the Confirm button to delete the user information and the corresponding user information is deleted from the database.</li> </ol>
--	---

Table 2 presents the commodity management module analysis.

**Table 2.** Commodity Management Module Analysis

Case Name	Commodity Management
Case Description	Adding and deleting commodity information is managed.
Prerequisites	System login successfully and network connection is normal.
Main Operations	<p>Editing Commodity</p> <ol style="list-style-type: none"> <li>1) The administrator opens the system.</li> <li>2) Enter the system login screen and enter the set login account and password.</li> <li>3) Login to the system and enter the main system interface.</li> <li>4) Click on the commodity management module to enter the user interface.</li> <li>5) Click “Edit” button and follow the prompts to select or enter information, such as name, category, and price.</li> <li>6) After filling the form, click the OK button to confirm and the commodity information will be edit in the database.</li> </ol> <p>Deleting Commodity</p> <ol style="list-style-type: none"> <li>1) In the Commodity management interface.</li> <li>2) Click “Delete” button to delete to the commodity you want to delete.</li> <li>3) Click the Confirm button to delete the commodity information and the corresponding commodity information is deleted from the database.</li> </ol>

. Table 3 shows the category management module analysis.

**Table 3.** Category Management Module Analysis

Case Name	Category Management
Case Description	Management of category information such as adding and deleting.
Prerequisites	The system login successfully and the network connection is working.
Main Operations	<p>Adding New Category</p> <ol style="list-style-type: none"> <li>1) The administrator opens the system.</li> <li>2) Login to the system and enter the main system interface.</li> <li>3) Click on the category management function module to enter the category management interface.</li> <li>4) Click the Add category information button on the category management interface to enter the Add category information interface.</li> <li>5) Follow the prompts and select or enter the category name and other information.</li> <li>6) After confirming that the information is correct, click the OK button to confirm the addition and the category information will be added to the database.</li> </ol> <p>Deleting Category</p> <ol style="list-style-type: none"> <li>1) You are in the category management interface.</li> <li>2) Click the Delete button to delete the category you want to delete.</li> <li>3) Click the Confirm button to delete the category information, and the corresponding category information is deleted from the database.</li> </ol>

Order management module analysis can be seen in Table 4.

**Table 4.** Order Management Module Analysis

<p>Case Name</p> <p>Case Description</p> <p>Prerequisites</p>	<p>Order Management</p> <p>Management of order information such as ship, received and deleting.</p> <p>The system logs in properly and the network connection is working.</p>
<p>Main Operations</p>	<p>Change logistics status: Shipped</p> <ol style="list-style-type: none"> <li>1) The administrator opens the system.</li> <li>2) Login to the system and enter the main system interface.</li> <li>3) Click on the order tracking module to enter the order management interface.</li> <li>4) Click on the 'Ship' button on the order tracking interface, the current order status will be 'Shipped', and the corresponding order status information will be changed in the database.</li> </ol> <p>Change the logistics status: Received</p> <ol style="list-style-type: none"> <li>1) The administrator opens the system.</li> <li>2) Login to the system and enter the main system interface.</li> <li>3) Click on the order tracking module to enter the order management interface.</li> <li>4) Click on the 'Received' button on the order management interface, the current order status will be 'Received', and the corresponding order status information will be changed in the database.</li> </ol> <p>Deleting Order</p> <ol style="list-style-type: none"> <li>1) You have entered the order tracking interface.</li> <li>2) Click 'delete' button to delete the order you want to delete.</li> <li>3) Click on the Confirm button to delete the order information and the corresponding order information is deleted from the database.</li> </ol>

### 3.4 Class Diagram

A Class diagram is a static view that describes the classes in a system, and the relationships between the classes. It is a model type, a static model type to be precise. Class diagrams represent classes, interfaces, and the collaborative relationships between them. The project has two parts, the Web-side, and the Android-side.

Figure 9 shows the class diagram of the main classes for web development. The Controller and Model parent classes are functional classes provided by the JFinal development framework. The specific location of these two packages is `com.jfinal.core.Controller` and `com.jfinal.plugin.activerecord.Model`.

Figure 10 shows the class diagram of the main development classes for Android application development projects.

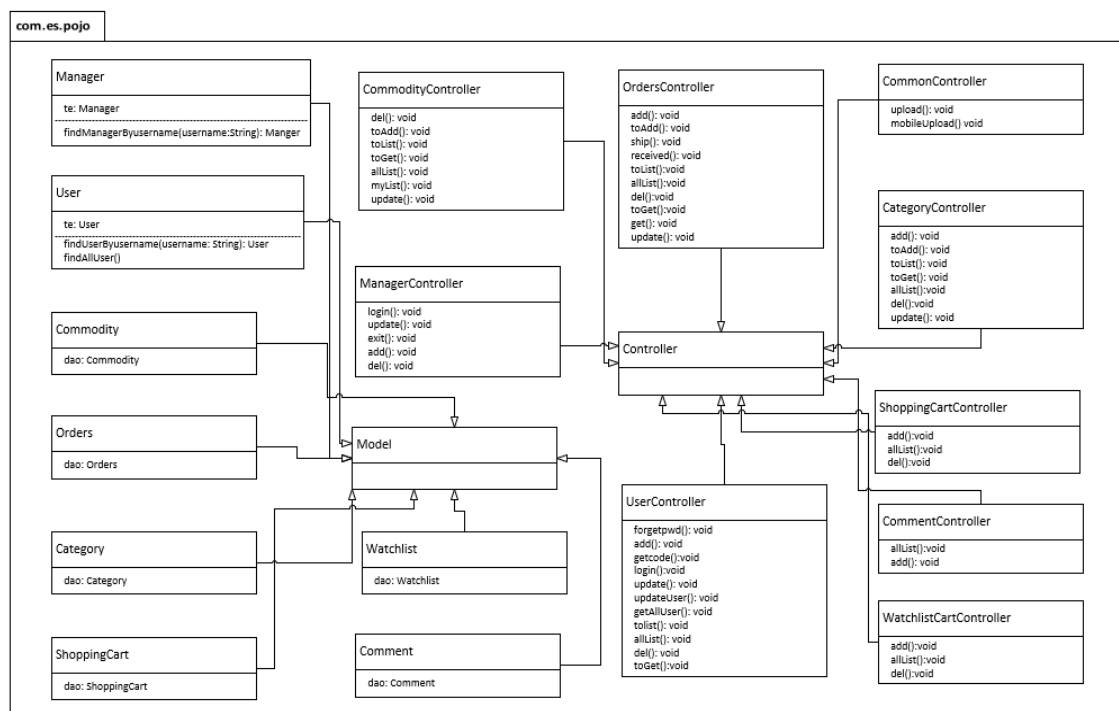


Figure 9. Class Diagram 1

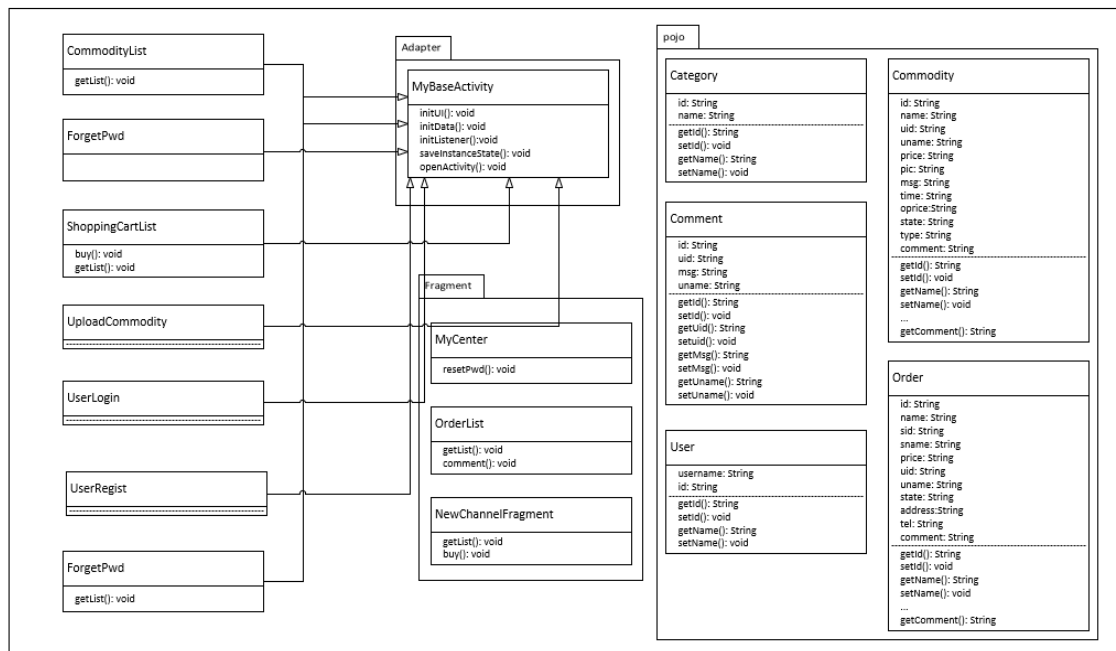


Figure 10. Class Diagram 2

### 3.5 Sequence Diagram

#### 3.5.1 Administrator Login

Figure 11 depicts a sequence diagram of how the administrator logs into the back-end management system. First, the administrator has a fixed password and username (username: admin & password: admin) in the database. Then, the administrator can use the username and password to log in to the management system.

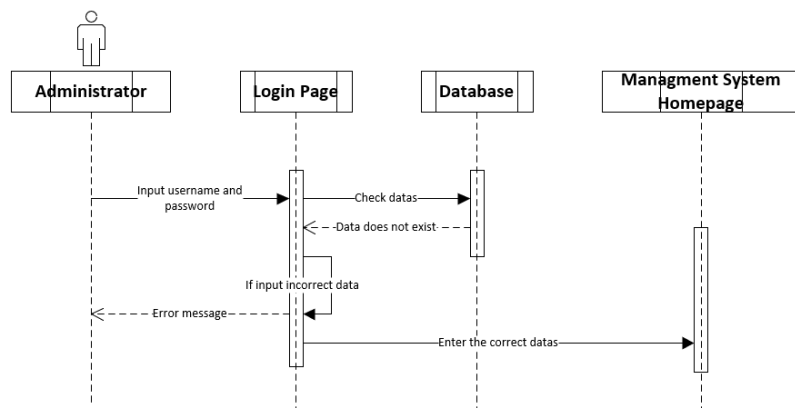
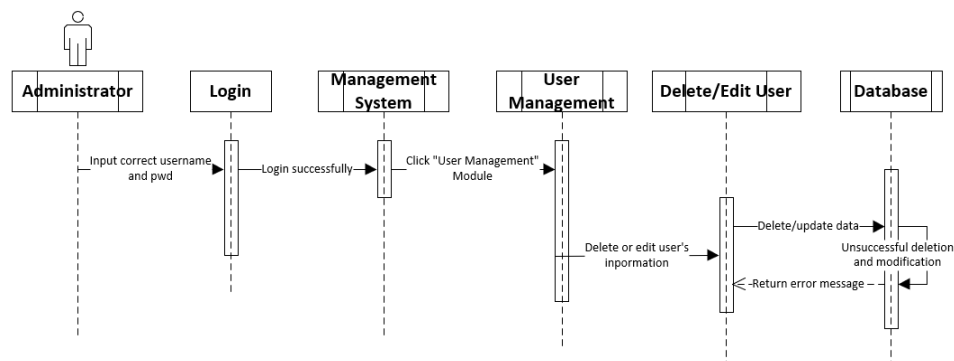


Figure 11. Administrator Login Sequence Diagram



### 3.5.2 Administrator Delete/Edit User

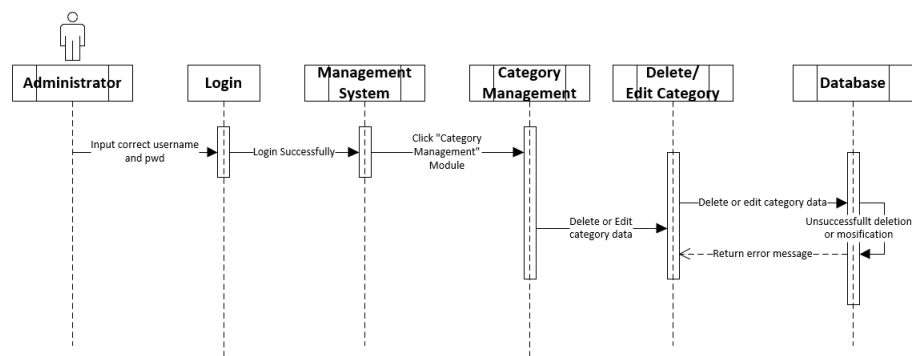
Figure 12 shows the sequence diagram of the administration of users by the administrator. After the administrator has successfully logged into the management system, he/she can delete and modify the registered users by clicking on the "User Management" section. While the administrator is operating, the data of the edited user is deleted or modified in the database by SQL statements.



**Figure 12.** Administrator Modifies Users Sequence Diagram

### 3.5.3 Administrator Delete/Edit Category

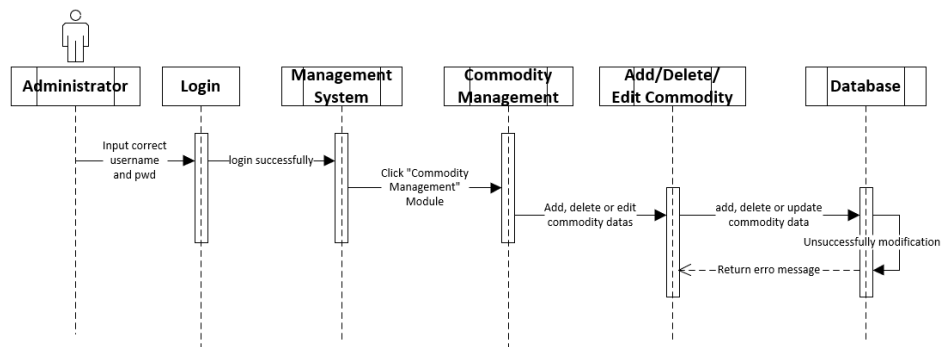
Figure 13 shows the sequence diagram for the administrator to manage the commodity categories. When the administrator logs into the management system, he/she can delete and modify the existing commodity categories by clicking on the "Category Management" section. The data of the edited category is deleted or modified in the database by SQL statements while the administrator is operating.



**Figure 13.** Administrator Modifies Category Sequence Diagram

### 3.5.4 Administrator Add/Delete/Edit Commodity

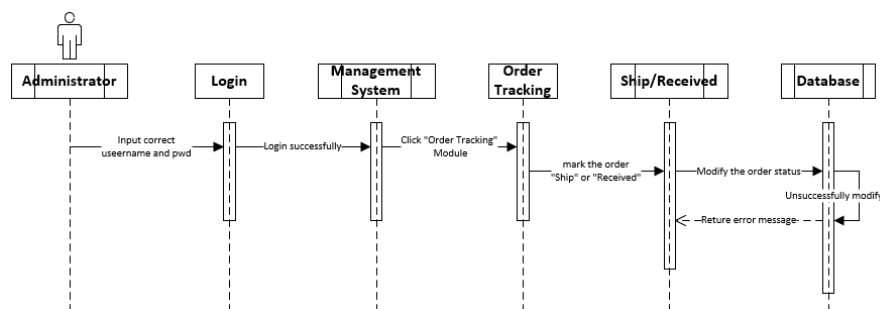
Figure 14 shows the sequence diagram for the administrator to manage commodity uploaded by users. Having logged into the management system, the administrator can delete, modify, and add existing commodity by clicking on the "Commodity Management" section. When the administrator operates, he can delete or modify the data of the edited commodity in the database through SQL statements.



**Figure 14.** Administrator Modifies Commodity Sequence Diagram

### 3.5.5 Administrator Processes Order

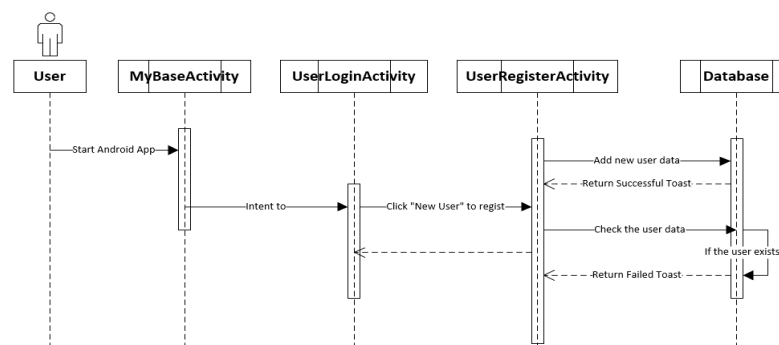
Figure 15 shows the sequence diagram of the administrator for order management. After successfully logging into the management system, the administrator can click the "Order Tracking" section to simulate the shipping and received of existing orders. During the operation of the administrator, the data of the edited order status will be deleted or modified in the database through SQL statements.



**Figure 15.** Administrator Modifies Order Sequence Diagram

### 3.5.6 User Registration

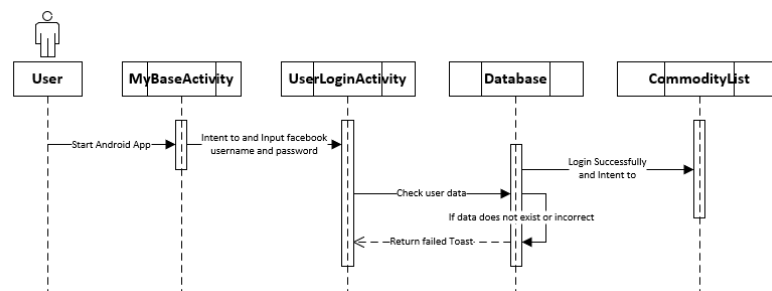
Figure 16 shows the sequence diagram of user registration. When the user starts the software, it will enter the login interface, and the user can click "New User" to register a new user. If the user already exists, the application will feedback the user's existed Toast; if successful, it will also feedback the registered successfully Toast. When the user is successfully registered, the application will return to the registration interface, and the user can log in with the registered Facebook username and password.



**Figure 16.** User Registration Sequence Diagram

### 3.5.7 User Login

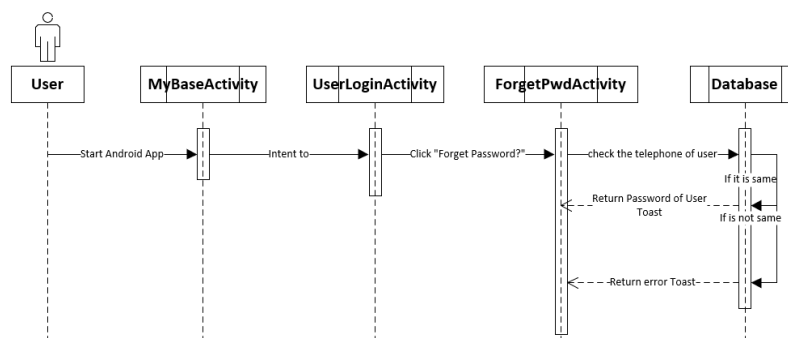
Figure 17 shows the sequence diagram of user login. When the user opens the program, he/she enters the same username and password stored in the database to successfully log in to the application homepage. If the username and password are entered incorrectly, or the account does not exist, the application will send user a Toast.



**Figure 17.** User Login Sequence Diagram

### 3.5.8 User Forget Password

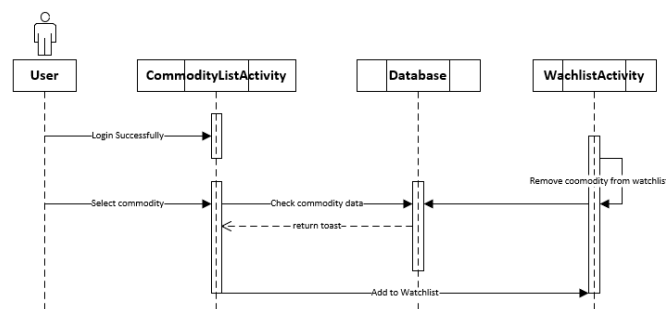
Figure 18 shows the sequence diagram of getting the password back when the user forgets the password. The user enters the login interface. In case of a forgotten password, the user can click "Forgot your password?" and fill in the Facebook username and phone number. By comparing with the phone number of the user corresponding to the database, if the phone number is the same, the user's password toast will be returned; if not, the wrong toast will be returned.



**Figure 18.** User Retrieve Password Sequence Diagram

### 3.5.9 User Collect Commodity to Watchlist

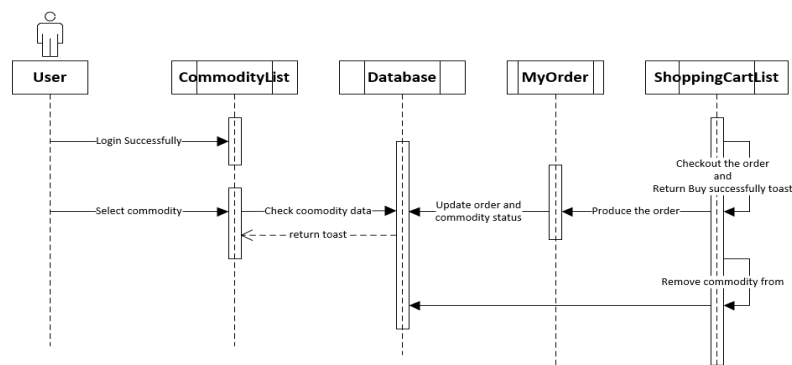
Figure 19 shows the sequence diagram of the user collecting commodity to the Watchlist. After successfully logging in, the user enters the application home page to view the commodity list. The user can click the commodity to collect it in the watchlist, and the data of the commodity will also be updated in the database. The user also can remove the commodity from the watchlist by clicking on it from the watch list.



**Figure 19.** Collect Commodity to Watchlist Sequence Diagram

### 3.5.10 User Add Commodity to Shopping Cart

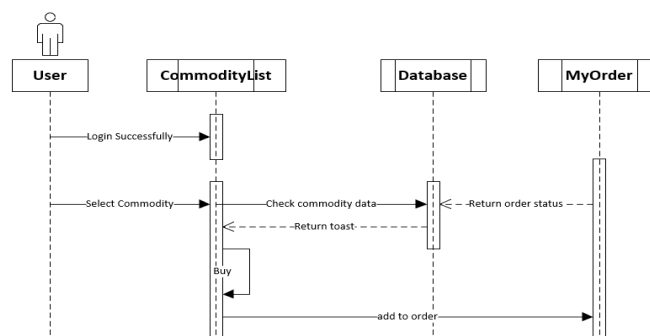
Figure 20 shows the sequence diagram of a user adding an item to the shopping cart and checking out. After the user logs in successfully, the user can click on the commodity and add it to the shopping cart. The user can view the shopping cart list by clicking the shopping cart icon on the top right of the home page. Clicking on the button below will check out the shopping cart and generate an order in "My order" at the same time. The user can also click on an item to remove it from the shopping cart. In the database, the status of commodity and orders are updated.



**Figure 20.** User Add and Checkout Shopping Cart Sequence Diagram

### 3.5.11 User Buy Commodity

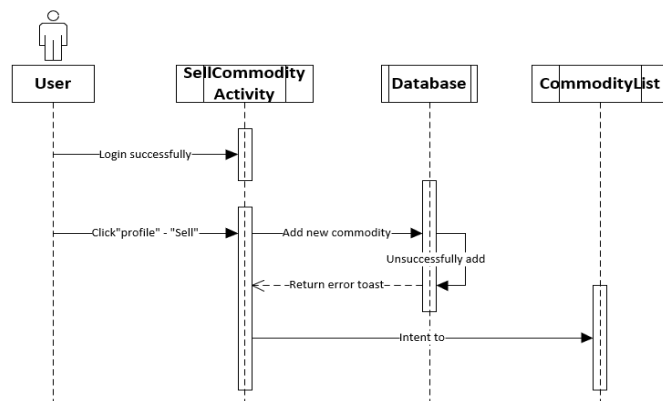
Figure 21 shows the sequence diagram for direct purchase of commodity by users. After successful login, users can click on the commodity, purchase the commodity directly and generate an order.



**Figure 21.** User Buy Commodity Sequence Diagram

### 3.5.12 User Sell Commodity

Figure 22 shows the sequence diagram of the user uploading commodity. When the user login successfully, they can click "Profile" - "Sell" to upload the commodity. The entered commodity information will be added to the database. The successfully uploaded commodity will be displayed in the Commodity List screen after refreshing.



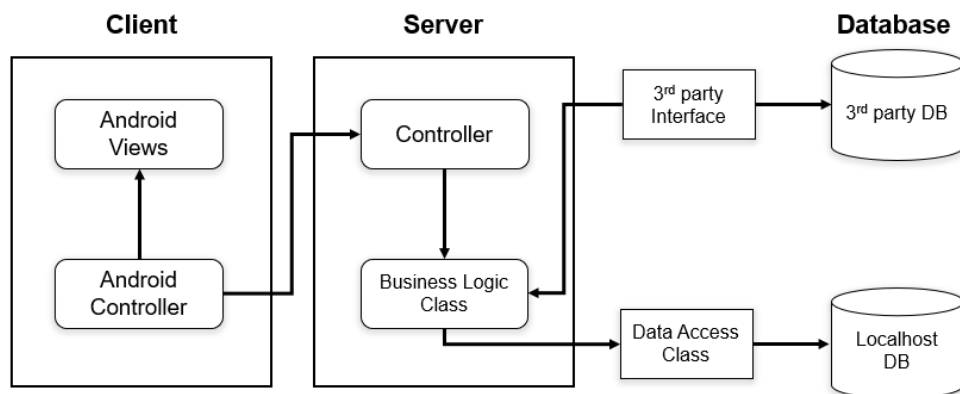
**Figure 22.** Sell Commodity Sequence Diagram

## 4 SYSTEM DESIGN

This section provides an overview of the system architecture design, database design and user interface design of the application.

### 4.1 System Architecture Design

The second-hand online shopping system generally adopts a B/S structure for system design. After registering as a user of the system, the user can interact with the backend server anytime and anywhere via Android device by Wi-Fi or mobile data network. After successfully logging in, the user can perform appropriate operations as needed. After HTTP interacts with the back-end data, the server will perform appropriate processing according to the client's request to achieve database access and send the user interface back to the mobile client to achieve smooth data interaction. In summary, the overall architecture of this system includes the Android mobile client, the backend server, and the MySQL database. The architecture of the system is shown in Figure 23.



**Figure 23.** System Architecture Diagram

As can be seen from the diagram above the system is made up of three parts, the client, the server, and the database, with the three levels taking on different roles in the system.

The client is the mobile terminal, where the user enters the system through a mobile device and establishes communication with the system. For this system, this layer is built on top of the Android-based application and includes the interface part of

the application (Android view) and the logical control of the application (Android controller). This layer obtains information by calling the server interface.

The server is a web application based on Java. Unlike traditional web applications, this layer does not contain the display of the interface, but only provides the JSON data interface for the frontend. This layer includes a control layer and a business logic layer. The control layer is responsible for providing the external interface and data definition for the interface. The business logic classes define the logic for accessing database information and the methods for accessing the database.

The data layer contains two data sources, one for accessing the local database and the other for accessing the database of the existing Android platform based used goods trading application system. Both in terms of data interaction and the design of the interface and flow, the system follows the MVC design principles, dividing the functional flow of the system into layers with independent functions. In the future, the system will need to be modified or extended. In short, it is very convenient to improve the scalability of the system.

At the same time, the system security design needs to be applied throughout the data flow to provide security for the storage, transmission, and utilization of the information system.

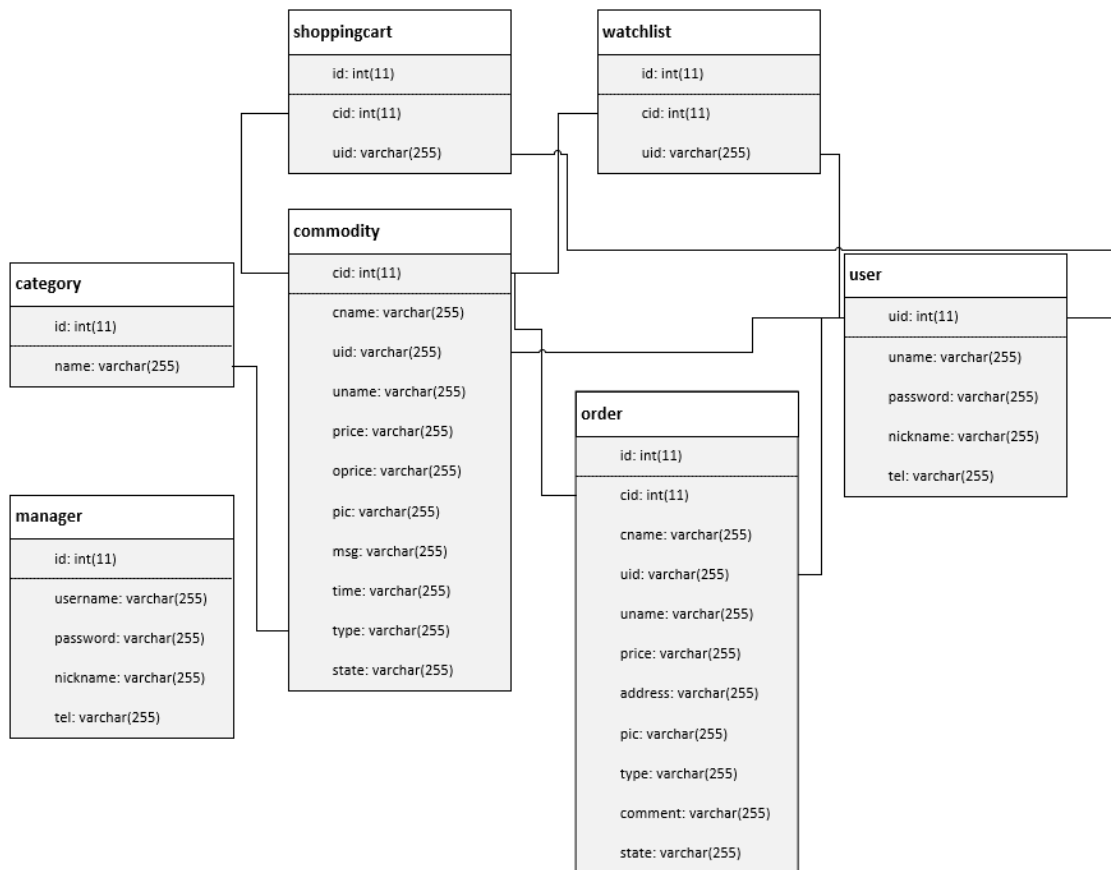
## 4.2 Database Design

The database of the application is created using MySQL. There are seven tables in the database, including manager table, user table, commodity table, category table, orders table, shopping cart table and watchlist table. Figure 24 below shows the E-R diagram of the database.

- *manager table* - The manager table is used to record and manage the manager's username, password, nickname, and telephone number.
- *user table* - The user table is used to record information about the user, which includes the user ID, username, password, nickname, and phone number. The `u_id` is the primary key of this table.



- *category table* - This table is used to record the category of second-hand commodity, including category ID and category name. The category id is the primary key.
- *commodity table* – The commodity table is used to record the information of the second-hand commodity, including commodity ID, picture, commodity name, final price, original price, description of the commodity, uploaded time, status, and category. The primary key of the table is c\_id. The table is connected to the primary key(u\_id) of the user table in such a way that both the seller's u\_id and u\_name are available.
- *order table* - The order table mainly contains basic information about the secondhand items that have been sold. The structure is similar to the commodity table, except that here the buyer's information is obtained from the user table via u\_id.
- *watchlist table* and *shopping table* - These two tables mainly record information about the second-hand commodity that have been added to the shopping cart and the watchlist. The main connection is made via the c\_id and the commodity table to get the commodity information.



**Figure 24.** Database ER Diagram

### 4.3 Graphical User Interface (GUI) Design

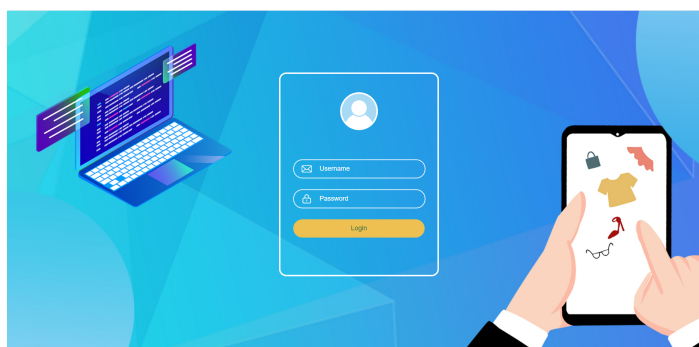
The main design languages for the front-end UI of the Web are HTML/CSS/JS, and these files need to be created in the WebRoot package. The design language for Android applications is XML, and the XML files are created in the layout package of the application project. A GUI is an interface display format for users to communicate with their computer or cell phone.

#### 4.3.1 Web UI Design

The web UI design is a ready-made front-end UI framework provided by *Layui* (<https://cdnjs.com/libraries/layers/layers>), which follows the native HTML/CSS/JS writing and organization format.

In this project, the web side is operated by the administrator and includes four main functional sections. It is mainly used to manage users, manage commodity and commodity categories, and order management.

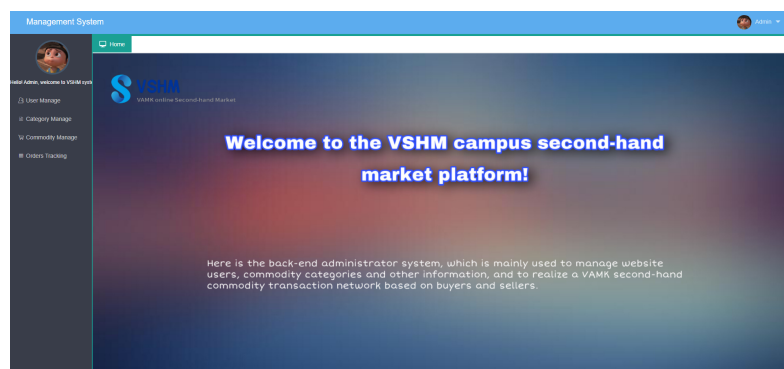
#### 4.3.1.1 Login Page



**Figure 25.** Login Page

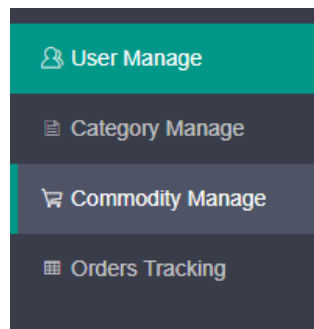
#### 4.3.1.2 Home Page

As a backend management system, the design of the web page is more focused on the realization of the function, so the functional section of the system should be obvious and to let the user have a good operating experience. Clarity and convenience are the key points of the web design of the backend management system.



**Figure 26.** Management System Homepage

The main interface mainly includes navigation, and administrator settings functional modules.



**Figure 27.** Left-Vertical Navigation

Navigation generally refers to a collection of page guidance channels, mostly in the form of menus, which can be applied to the top and side, making the whole management system the key to easy operation. A good navigation menu bar can give the user a comfortable operating experience. This system uses vertical/left navigation.

#### 4.3.1.3 Four Main Functional Sections

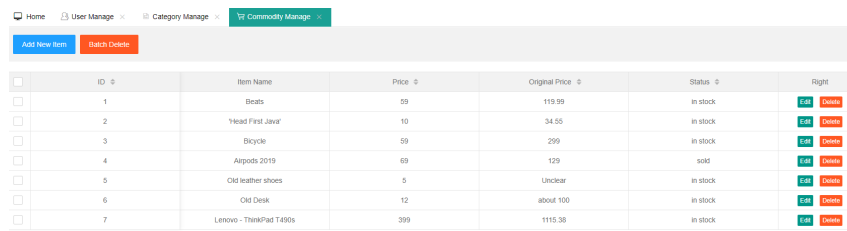
For the backend management system of second-hand online shopping, the system needs to allow the administrator to perform simple and clear operations on all users, commodity, and orders, such as editing and deleting user or commodity information and managing the status of commodity. The focus of the UI design is on clear, convenience and fast.

	ID	Username	Password	Nickname	Right
<input type="checkbox"/>	1	Che Sicheng	12345678	Che1998	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	2	Steven	steven1234	VVSHM-Steven	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	3	Andrew	andrew1234	1995_Andrew	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	4	Jessica	jessica1234	BeautyJessica	<a href="#">Edit</a> <a href="#">Delete</a>

**Figure 28.** User Management Section

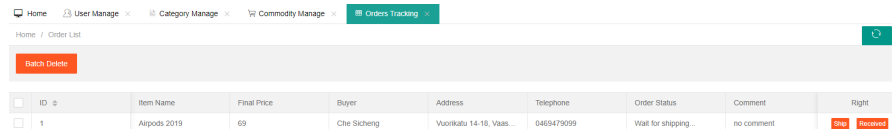
	ID	Category	Right
<input type="checkbox"/>	1	Electronic	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	2	Clothing	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	3	Book	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	4	Instrument	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	5	Furniture	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	6	Transport	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	7	Other	<a href="#">Edit</a> <a href="#">Delete</a>

**Figure 29.** Category Management Section



ID	Item Name	Price	Original Price	Status	Right
1	Beats	59	119.99	in stock	Edit Delete
2	"Head First Java"	10	34.55	in stock	Edit Delete
3	Bicycle	59	299	in stock	Edit Delete
4	Airpods 2019	69	129	sold	Edit Delete
5	Old leather shoes	5	Unclear	in stock	Edit Delete
6	Old Desk	12	about 100	in stock	Edit Delete
7	Lenovo - ThinkPad T490s	999	1115.38	in stock	Edit Delete

**Figure 30.** Commodity Management Section



ID	Item Name	Final Price	Buyer	Address	Telephone	Order Status	Comment	Right
1	Airpods 2019	69	Che Sicheng	Vuorikatu 14-18, Vaas...	04059479059	Wait for shipping...	no comment	Edit Cancel

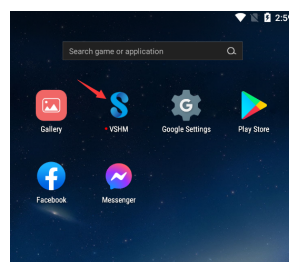
**Figure 31.** Order Tracking Section

## 4.3.2 Android UI Design

The Android side is the client side, which is the interface used by the users. As it is a service for all users, aesthetics and convenience are key to giving users a good operating experience.

### 4.3.2.1 Launcher Icon

The application is started by clicking the Launcher Icon. The Icon design is very simple and clear, with a blue "S" letter as the Icon. The name of this application (VSHM) is derived from the abbreviation of VAMK Second-Hand Market.

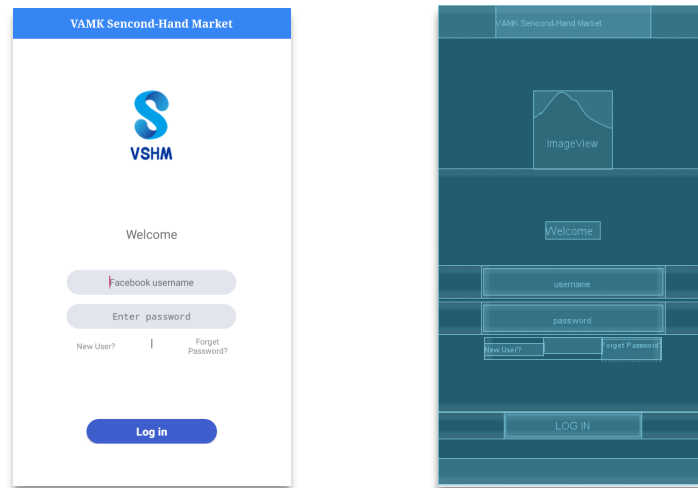


**Figure 32.** Launcher Icon

### 4.3.2.2 Login Layout

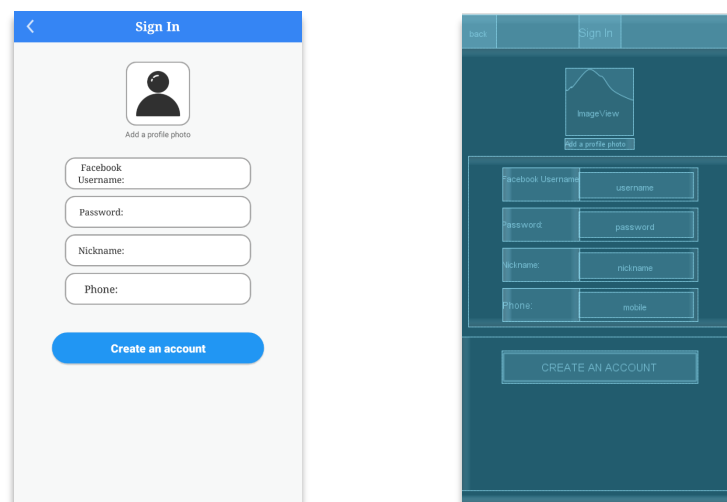
Figure 33 shows the login page of the application. The login activity consists of entering the Facebook username, password, and Login button. New users also need to register for the first time by clicking on the "New User?" button. Registered users

can log in directly with the correct username and password. If registered users forget their password, they can retrieve it by clicking on "Forget Password?" and using the phone number they provided when registering.



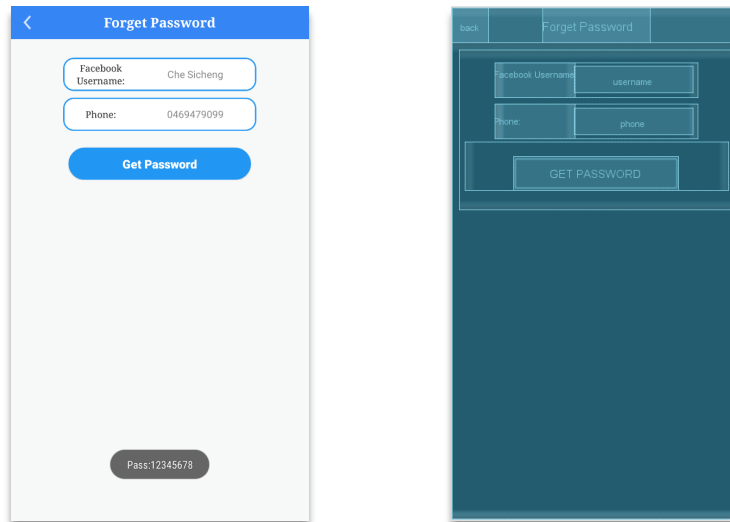
**Figure 33.** Login Layout and Design

New users will need to fill in their Facebook username, password, nickname, and phone number to register. This is because the application allows sellers and buyers to communicate with each other in a timely manner by calling on third-party social networking software during the purchase process after entering the application. Therefore, registering with a Facebook username makes it easier for buyers to find sellers and for them to communicate with each other.



**Figure 34.** Sign In Layout And Design

If a registered user has forgotten their password, they can retrieve it by using the phone number.



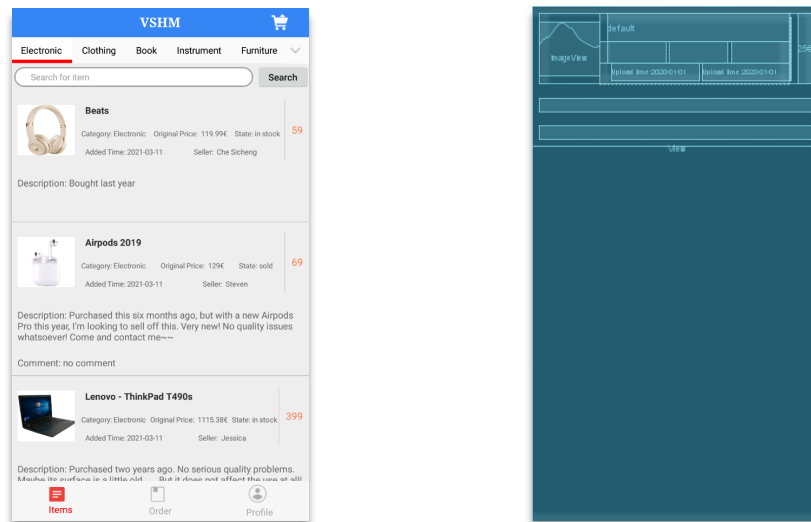
**Figure 35.** Forget Password Layout and Design

#### 4.3.2.3 Main Page

Figure 36 shows the main screen of the application. This interface needs to be designed to contain the basic functional points of the application, including information about second-hand commodity, categories, and orders,. The commodity information includes photos, sale price, original price, stock status, upload time the seller's Facebook username and the seller's description of the commodity.

A search bar is set up at the top, allowing users to conduct a fuzzy search by commodity name in each category section separately.

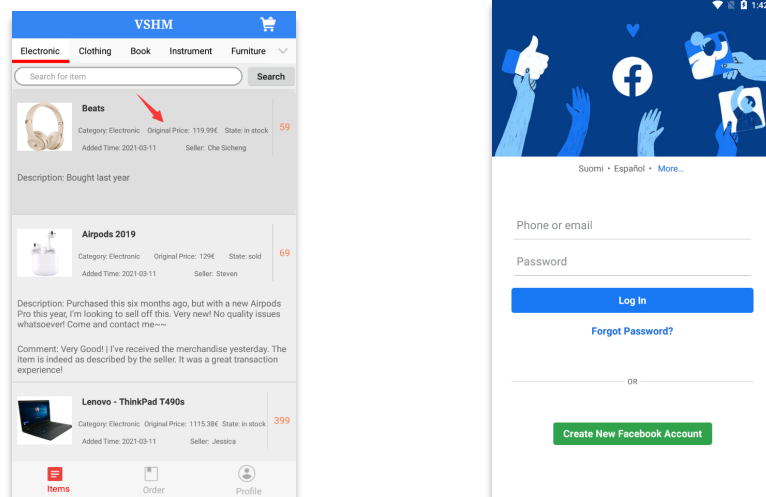
The three fragments transitions at the bottom of the main screen, will give the user a better operating experience.



**Figure 36.** Main page and design

#### 4.3.2.4 Contact Seller by Facebook

On the main page, if the user wants to find out more details about the commodity, they can call the third-party social networking software Facebook by long-clicking on the selected commodity and communicate more with the seller by searching for the seller's Facebook username. Figure 37 shows the process of calling Facebook.

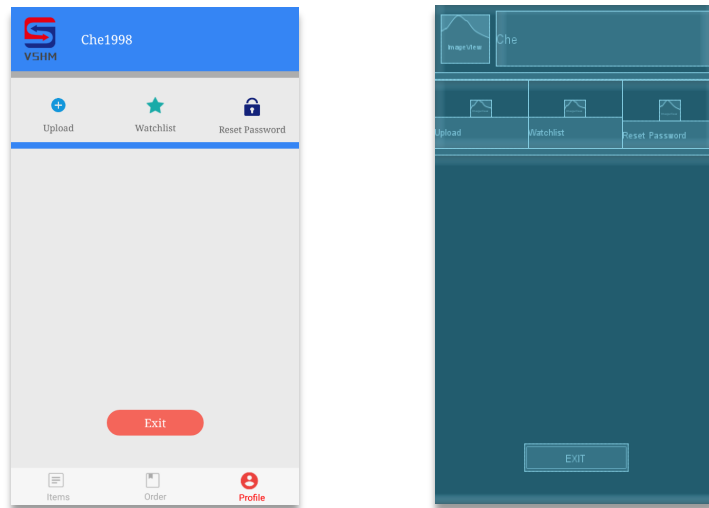


**Figure 37.** Launch Facebook



### 4.3.2.5 Profile Layout

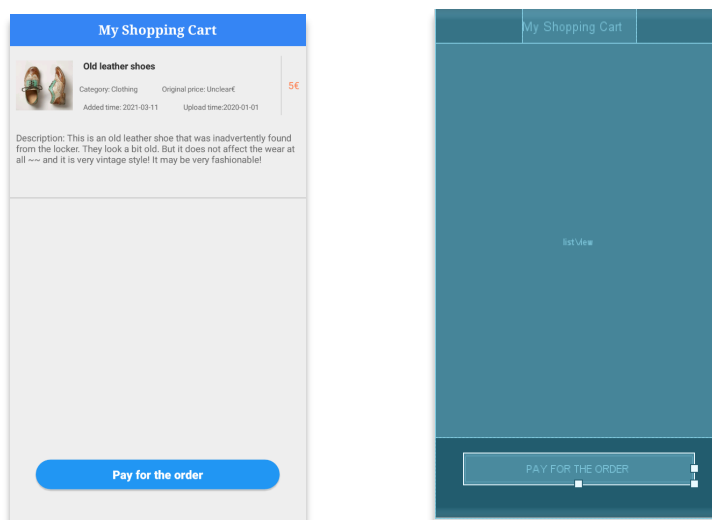
Figure 38 shows the interface of profile. This screen contains three main functions, uploading used items put up for sale, Watchlist, and resetting the user's password. From this screen it is also possible to exit the application by clicking on the 'Exit' button.



**Figure 38.** Profile Layout Design

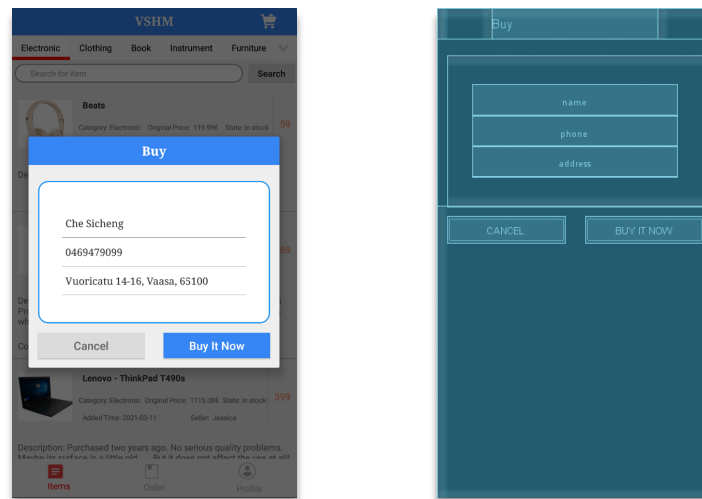
- **Shopping Cart and Watchlist Layout**

Users can perform three operations by clicking second-hand goods to be sold, namely, buy, add to shopping cart, and collect (add to watch list). Figures 39, 40 and 41 show the shopping cart, purchase and watchlist layout.



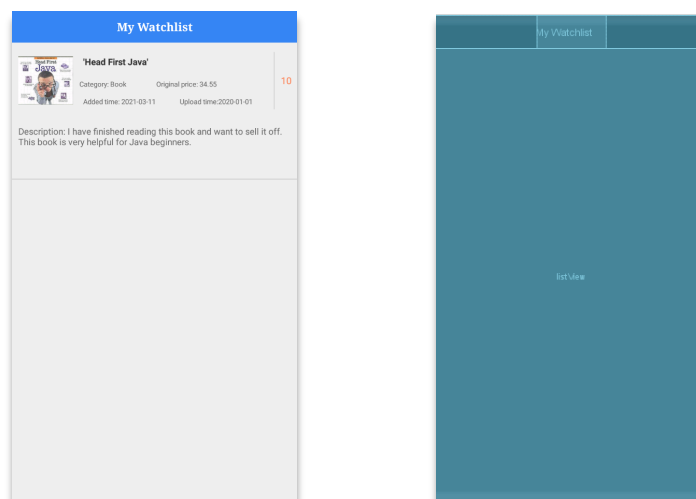
**Figure 39.** Shopping Cart Layout and Design

To make a purchase, the required credentials are username, delivery address and phone number before clicking on "Buy It Now". Users can also stop the purchase by clicking the "Cancel" button.



**Figure 40.** Buy Layout and Design

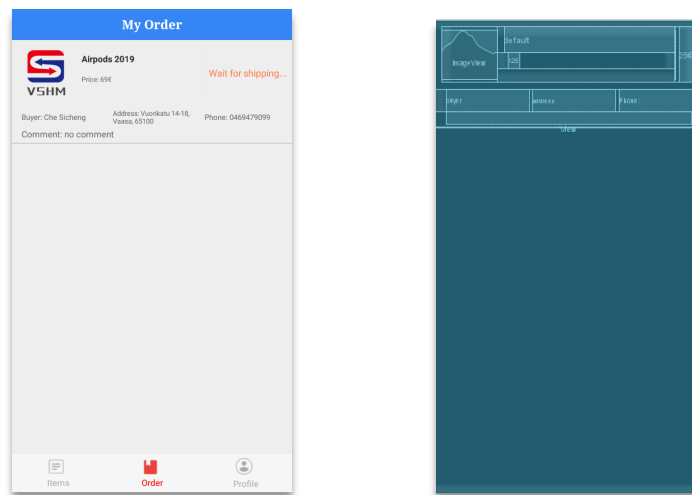
In the watchlist, users can click on a collected item to remove it.



**Figure 41.** Watchlist Layout and Design

### 4.3.2.5 My Order Layout

Having successfully purchased an item, the user will see the status of the order in My order, which generally has three statuses: Waiting for shipping, Shipped and Received. Figure 43 shows an order waiting to be shipped.

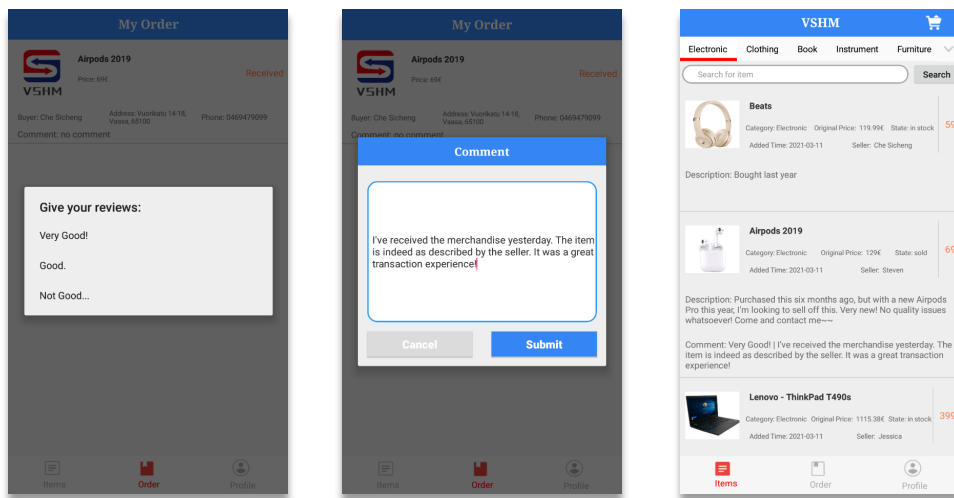


**Figure 42.** My Order Layout and Design

### 4.3.2.6 Comment Layout

Once the status of an item has been changed through the backend management system by the administrator, the user can give comments for the item once it has been received. The first thing is to rate the commodity, including “Very Good”, “Good” and “Not Good”, and then the user can enter reviews of the commodity. Once the comments have been submitted, all users will be able to see the buyer’s review of the commodity they have received on the main screen. No other user can take any action on the item.

Figure 43 shows the process by which a user evaluates an item that has been received.

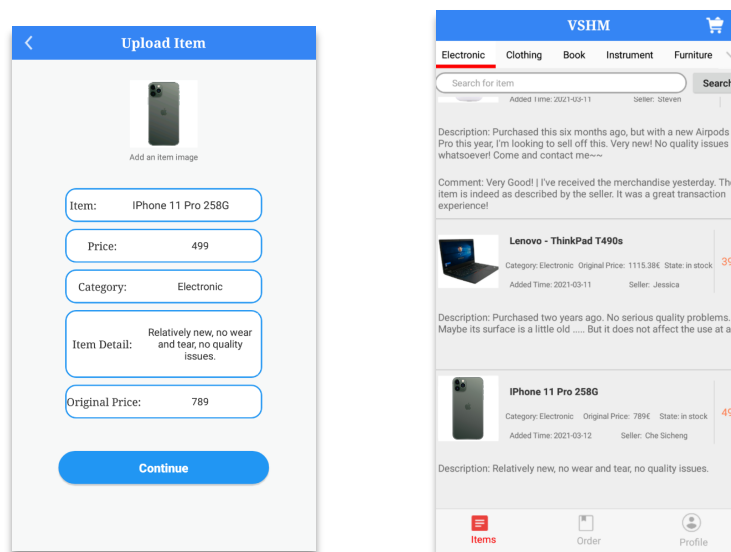


**Figure 43.** Rating and Give Comment layout

#### 4.3.2.7 Sell Commodity Layout

In this application, users need to upload second-hand goods by filling in the information of the commodity, including photos of the item, name, sale price, original price, type, description.

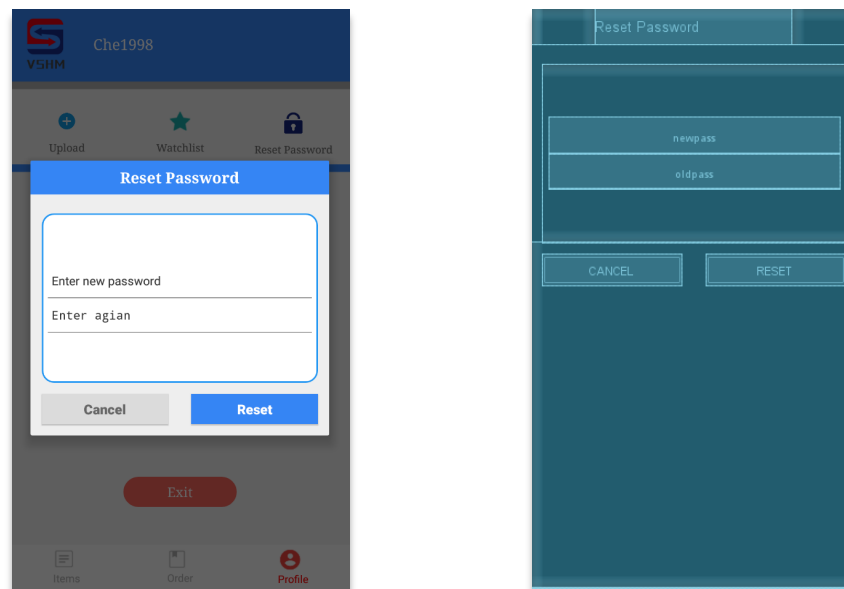
The first image in Figure 44 shows the user interface for uploading commodity, and the second image shows the result after uploading and refreshing.



**Figure 44.** Sell Commodity layout

#### 4.3.2.8 Reset Password Layout

Users can reset their passwords as needed. The user needs to enter the new password twice, and the two passwords are consistent to modify it successfully. Figure 45 shows the interface for changing the password.



**Figure 45.** Reset Password Layout and Design

## 5 IMPLEMENTATION

### 5.1 Database Connection Implementation

In this system we are using the http network protocol technology, which has two types of requests GET and POST, and for the server interaction data, the program uses JSON strings, Gson framework to parse and convert it, these methods are encapsulated in *HttpsUtil.java* and *JsonUtils.java* two tool classes, respectively. The specific code is as follows to facilitate the implementation of each functional module call.

GET request method:

```
public static String httpDoGet(String url) throws ClientProtocolException, IOException{
    String result = "";
    HttpGet request = new HttpGet(url);
    request.addHeader("Content-Type", "text/html");
    request.addHeader("charset", "utf-8");
    //Get the responding object
    HttpResponse response = new DefaultHttpClient().execute(request);
    // When the request is successful
    if(response.getStatusLine().getStatusCode()==200){
        // Get the data returned by the response server
        result = EntityUtils.toString(response.getEntity());
    }
    return result;
}
```

**Code Snippet 1.** Get Request Method

POST request method:

```
public static String doPost(String postData, String urlStr)
    throws UnsupportedEncodingException, IOException,
    KeyManagementException, NoSuchAlgorithmException {
    URL url = new URL(urlStr);
    URLConnection cnx = getConnection(url); //connect to the server

    OutputStreamWriter wr = new OutputStreamWriter(cnx.getOutputStream());
    wr.write(postData);
    wr.flush();
}
```

```

    wr.close();
    // Retrieve the result of the response
    return changeInputStream(cnx.getInputStream(), "utf-8");
}

```

### Code Snippet 2. POST Request Method

Parsing and conversion of JSON data:

```

    public static String createJsonString(Object value) {
        Gson gson = new Gson();
        String string = gson.toJson(value);
        return string;
    }

    public static <T> T getObject(String jsonString, Class<T> cls) {
        T t = null;
        try {
            Gson gson = new Gson();
            t = gson.fromJson(jsonString, cls);
        } catch (Exception e) {

        }
        return t;
    }

    public static List<?> StringFromJson(String jsondata, Type listType) {
        Gson gson = new Gson();
        ArrayList<?> list = gson.fromJson(jsondata, listType);
        return list;
    }

```

### Code Snippet 3. Parsing and conversion of JSON data

Code snippet 4 shows the online shopping system used to connect to an existing database. By writing the URL format of the JDBC protocol, the application can be connected to the local database.

```

jdbcUrl = jdbc:log4jdbc:mysql://localhost:8080/e1700704_vshm?characterEncoding=utf8&zeroDateTimeBehavior=convertToNull
user =root
password = root

```

### Code Snippet 4. Local Database Connection

## 5.2 Server-Side Implementation

### 5.2.1 User Management

After entering the correct username and password, the administrator can log into the homepage of the backend management system. In the user management module, the administrator can manage the registered user information.

The UserController class, which inherits from Controller in the control layer, has various methods that implement the management of users.

After modifying the user information by modifying the user, the corresponding user information in the database will be deleted according to the u\_id. The core code snippet is as follows:

```
public void del() {
    try {
        String[] ids = getParaValues("id");
        for (String id : ids) {
            User.te.deleteById(id);
        }
        renderJson(JsonKit.toJson(new StatusJson("200", "suc", true)));
    } catch (Exception e) {
        // TODO: handle exception
        renderJson(JsonKit.toJson(new StatusJson("500", "fail", true)));
    }
}
```

#### Code Snippet 5. Delete user by Administrator

```
public void updates() {
    try {
        getModel(User.class, "", true).update();
        renderJson(new CommonData("500", JsonKit.toJson(User.te.findById(getPara("id"))),
        ""));
    } catch (Exception e) {
        // TODO: handle exception
        renderJson(new CommonData("500", "", e.toString()));
    }
}
```

#### Code Snippet 6. Edit user by Administrator



### 5.2.2 Category Management

The `CategoryController` class, which inherits from `Controller`, has various methods for implementing category management.

In the category management module, there are mainly functions for adding, editing, and deleting. The delete and edit functions can be completed according to find the id in the category table of the database. The code snippet below shows adding new categories via PUT:

```
public void add() {
    try {
        getModel(Category.class, "", true).save();
        JSONObject js = new JSONObject();
        js.put("code", "200");
        renderJson(js.toJSONString());
    } catch (Exception e) {
        // TODO: handle exception
        System.out.println(e.toString());
        JSONObject js = new JSONObject();
        js.put("code", 500);
        js.put("msg", e.toString());
        renderJson(js.toJSONString());
    }
}
```

**Code Snippet 7.** Add new category by Administrator

### 5.2.3 Commodity Management

In the commodity management module, the functions are like those in the user management module, mainly including editing and deleting. The deleted or edited commodity will also be deleted or edited from the commodity table based on the id. The following code snippet shows the delete function:

```
public void del() {
    try {
        String[] ids = getParaValues("id");
        for (String id : ids) {
            Commodity.dao.deleteById(id);
        }
        renderJson(JsonKit.toJson(new StatusJson("200", "suc", true)));
    } catch (Exception e) {
```

```

// TODO: handle exception
renderJson(JsonKit.toJson(new StatusJson("500", "fail", true)));
}
}

```

**Code Snippet 8.** Delete commodity by Administrator

## 5.2.4 Order Management

In the order management module, the order status, including “shipped” and “received”, is obtained through the `getModel()` method, and the status updated according to the administrator's operation. At the same time, the order status in the database will also be changed according to the id.

```

public void ship(){
    try {
        getModel(Orders.class, "", true).set("state","Shipped").update();
        renderJson(new CommonData("200", "", "success"));
    } catch (Exception e) {
        // TODO: handle exception
        renderJson(new CommonData("500", "", e.toString()));
    }
}
public void received(){
    try {
        getModel(Orders.class, "", true).set("state","Received").update();
        renderJson(new CommonData("200", "", "success"));
    } catch (Exception e) {
        // TODO: handle exception
        renderJson(new CommonData("500", "", e.toString()));
    }
}
}

```

**Code Snippet 9.** Changes the order status by Administrator

## 5.3 Client-Side Implementation

### 5.3.1 Login

The start of the application is the user login interface. The user needs to fill in the Facebook username and password. This code is used to determine whether the user exists and whether the username and password are consistent with the code value

set in `CommonData.java`. If `code = "200"`, `msg` displays the information of the successful state, otherwise, `code = "500"`, `msg` displays the information of the failed state.

```
public void login() {
    User m = User.te.findUserByUsername(getPara("username"));
    if (m == null)
        renderJson(JsonKit.toJson(new CommonData("500", null, "User does not exist...")));
    else {
        if (m.getStr("pass").equals(getPara("password"))) {
            setSessionAttr("u", m);
            renderJson(new CommonData("200", JsonKit.toJson(m), "success"));
        } else
            renderJson(new CommonData("500", null, "Incorrect password!"));
        }
    }
}
```

### Code Snippet 10. Identify the Login User

```
login.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        RequestParams ps = new RequestParams();
        ps.add("username", username.getText().toString());
        ps.add("password", password.getText().toString());
        UserClient.post("user/login", ps, new AsyncHttpResponseHandler() {
            @Override
            public void onSuccess(String content) {
                super.onSuccess(content);
                CommonData data = JSON.parseObject(content, CommonData.class);
                if (data.getCode().equals("200")) {
                    User u = JSON.parseObject(data.getData(), User.class);
                    MyApplication.getApp().setU(u);
                    startActivity(new Intent(UserLogin.this, MainActivity.class));
                } else {
                    MyToastUtil.ShowToast(con, data.getMsg());
                }
            }
        });
    }
});
```

### Code Snippet 11. Login button

### 5.3.2 Registration

Users can fill in personal information as needed to register a new account, and then click "Create an Account" to successfully register. The following code identifies whether the user is already registered according to the username in database. If the user does not exist, a new user is created. Then pass data through POST.

```

public void add() throws Exception {

    User u = User.te.findFirst("select * from user where username=?",
        getPara("username"));

    if (u == null) {
        getModel(User.class, "", true).save();
        renderJson(JsonKit.toJson(new CommonData("200", null, "successfully")));
    } else {
        renderJson(JsonKit.toJson(new CommonData("500", null, "User already exists...")));
    }
}

reg.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        RequestParams ps = new RequestParams();
        ps.add("username", username.getText().toString());
        ps.add("pass", password.getText().toString());
        ps.add("nickname", nickname.getText().toString());
        ps.add("tel", mobile.getText().toString());
        ps.add("head", head_url);

        UserClient.post("user/add", ps, new AsyncHttpResponseHandler() {

            @Override
            public void onSuccess(String content) {
                super.onSuccess(content);
                CommonData data = JSON.parseObject(content, CommonData.class);
                if (data.getCode().equals("200")) {
                    MyToastUtil.ShowToast(con, "Registration success");
                    finish();
                } else {
                    MyToastUtil.ShowToast(con, data.getMsg());
                }
            }
        });
    }
});
}

```

**Code Snippet 12.** Register A New User

### 5.3.3 Add Commodity Shopping Cart and Purchase

Users can add the selected product to the shopping cart or watchlist by the sid and uid of the commodity table and the user table in the database. It should be noted that if the user purchases a commodity, the status of the commodity changes from "in stock" to "sold". The following code snippet shows the implementation of the function of adding products to the shopping cart and collect to watchlist:

```

listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, final int position, long id) {
        new AlertDialog.Builder(getActivity()).setTitle("Do you want to 'buy' or 'collect'?")
            .setPositiveButton("Collect", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    RequestParams ps = new RequestParams();
                    ps.add("sid", list.get(position).getId());
                    ps.add("uid", MyApplication.getApp().getU().getId());
                    UserClient.get("watchlist/add", ps, new AsyncHttpResponseHandler() {
                        @Override
                        public void onSuccess(String content) {
                            super.onSuccess(content);
                            if (content.equals("1")) {
                                MyToastUtil.ShowToast(getActivity(), "Successfully");
                            } else {
                                MyToastUtil.ShowToast(getActivity(), "Collected");
                            }
                        }
                    });
                }
            })
            .setNegativeButton("Buy", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    if(list.get(position).getState().equals("in stock")) {
                        gm(list.get(position));
                    } else {
                        MyToastUtil.ShowToast(getActivity(), "Sold");
                    }
                }
            })
            .setNeutralButton("Add to Cart", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    if(list.get(position).getState().equals("in stock")) {
                        RequestParams ps = new RequestParams();
                        ps.add("sid", list.get(position).getId());
                        ps.add("uid", MyApplication.getApp().getU().getId());
                    }
                }
            });
    }
});

```

```

        UserClient.get("shoppingcart/add", ps, new AsyncHttpResponseHandler() {
            @Override
            public void onSuccess(String content) {
                super.onSuccess(content);
                MyToastUtil.ShowToast(getActivity(), "Successfully");
            }
        });
    }else{
        MyToastUtil.ShowToast(getActivity(), "Sold");
    }
}
}).show();
});

```

**Code Snippet 13.** Add or collect commodity to Shopping Cart or Watchlist

After the user clicks the "Buy" text button, the application will pop up a dialog box, and the dialog box has automatically filled in the username and phone number, and the user only needs to fill in the address manually. The application uses the SharedPreferences object to store and retrieve user information.

```

SharedPreferences pref = UserLogin.this.getSharedPreferences("data",MODE_PRIVATE);
SharedPreferences.Editor editor = pref.edit();
editor.putString("user",u.getUsername());
editor.putString("tel",u.getTel());
editor.commit();

```

**Code Snippet 14.** Get SharedPreferences Object

```

SharedPreferences pref = getContext().getSharedPreferences("data",Activity.MODE_PRIVATE);
SharedPreferences.Editor editor = pref.edit();
String username = pref.getString("user", "500");
String tel = pref.getString("tel", "500");
name.setText(username);
phone.setText(tel);

```

**Code Snippet 15.** Read Data from SharedPreferences

### 5.3.4 Calling Facebook

Considering that users can learn more about the commodity, the application can call third-party software, Facebook. Buyers and sellers can communicate in real time via Facebook.

```

listView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
    @Override
    public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id)
    {
        Intent intent = new Intent("android.intent.category.LAUNCHER");
        intent.setClassName("com.facebook.katana", "com.facebook.katana.LoginActivity");
        startActivity(intent);
        return true;
    }
});

```

#### Code Snippet 16. Calling Facebook

### 5.3.5 Rating and Give Comments

The user can rate and comment on the order after receiving the commodity, and the order status must be "received". Then the user clicks on the order in "My Orders" to enable the comment function.

```

listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, final int position, long id) {
        if (list.get(position).getState().equals("Received")) {
            final String[] items = {"Very Good!", "Good.", "Not Good..."};
            AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
            builder.setTitle("Give your reviews: ");
            builder.setItems(items, new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    pl(list.get(position), items[i]);
                }
            });
            builder.show();
        }
    });
});

```

#### Code Snippet 17. Rate the Received Order

```

private void review(final orders s, final String pf) {
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
    final AlertDialog dialog = builder.create();
    final EditText msg = (EditText) view.findViewById(R.id.msg);
    Button btnOK = (Button) view.findViewById(R.id.add);
    Button btnCancel = (Button) view.findViewById(R.id.qx);
    btnOK.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
            String msgs = msg.getText().toString();
            // password!=null && !password.equals("")
            if (!TextUtils.isEmpty(msgs)) {
                RequestParams ps = new RequestParams();
                ps.add("id", s.getId());
                ps.add("comment", pf+" | " + msgs);
                UserClient.post("orders/update", ps,
                    new AsyncHttpResponseHandler() {
                        @Override
                        public void onSuccess(String content) {
                            super.onSuccess(content);
                            getlist();
                        }
                    });
            } else {
                Toast.makeText(getActivity(), "The content of the input box cannot be empty!",
                    Toast.LENGTH_SHORT).show();
            }
        }
    });

    btnCancel.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
            dialog.dismiss();
        }
    });

    dialog.show();
}

```

**Code Snippet 18.** Give Comments



### 5.3.6 Sell Commodity

Users can sell their second-hand commodity by uploading pictures and information of commodity. This is done by clicking “Sell” in Profile to upload and sell commodity.

```

@Override
protected void initListener() {
    head.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            new AlertDialog.Builder(UploadCommodity.this).setTitle("Choose portrait").setNegativeButton("", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    SimpleDateFormat df = new SimpleDateFormat(
                        "MMddHHmmssSSSS");
                    names = df.format(new Date());
                    Intent intent = new Intent(
                        MediaStore.ACTION_IMAGE_CAPTURE);
                    intent.putExtra(MediaStore.EXTRA_OUTPUT, Uri
                        .fromFile(new File(Environment
                            .getExternalStorageDirectory(), names + ".jpg")));
                    startActivityForResult(intent, 2);
                }
            }).setNeutralButton("Gallery", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    Intent intent = new Intent(Intent.ACTION_PICK, null);
                    intent.setDataAndType(
                        MediaStore.Images.Media.EXTERNAL_CONTENT_URI, "image/*");
                    startActivityForResult(intent, 1);
                }
            }).show();
        }
    });
    reg.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            RequestParams ps = new RequestParams();
            ps.add("name", name.getText().toString());
            ps.add("price", price.getText().toString());
            ps.add("msg", msg.getText().toString());
            ps.add("type", type.getText().toString());
            ps.add("yj", yj.getText().toString() + "€");
            ps.add("pic", head_url);
            ps.add("uid", MyApplication.getApp().getU().getId());
        }
    });
}

```

```
ps.add("uname", MyApplication.getApp().getU().getUsername());
UserClient.post("sp/add", ps, new AsyncHttpResponseHandler() {

    @Override
    public void onSuccess(String content) {
        super.onSuccess(content);
        CommonData data = JSON.parseObject(content, CommonData.class);
        if (data.getCode().equals("200")) {
            MyToastUtil.ShowToast(con, "Successfully");
            finish();
        } else {
            MyToastUtil.ShowToast(con, data.getMsg());
        }
    }
});
}
```

**Code Snippet 19.** Sell Second-hand Commodity

## 6 TESTING

System testing is the final and very important step in software design. It is used to check whether the quality, performance and reliability of the software meet the needs of the user.

### 6.1 Web Application Testing

#### 6.1.1 Administrator Login

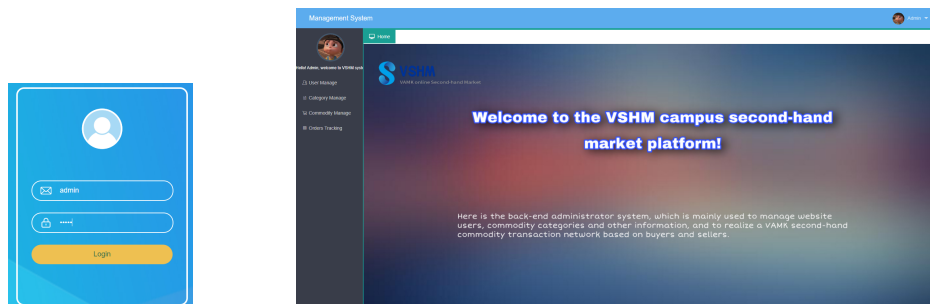
The administrator login was tested as follows:

- 1) The administrator logs in according to the specified username and password (username: **admin** & password: **admin**).
- 2) Login with empty username or password
- 3) Click the “login” button to enter the background management system.

- Expected Result

Only administrators with correct username and password can enter the management system.

- Actual Result



**Figure 46.** Administrator Login Successfully

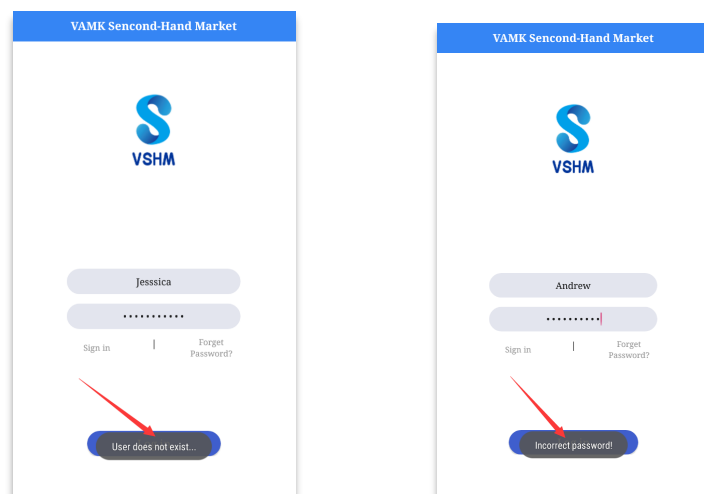
## 6.1.2 User Management

The administrator managing user was tested as follows:

- 1) Delete the information of the fourth user “Jessica”. “Jessica” logs in Android application and checks it.
  - 2) Edit the password of the third user “Andrew” from “andrew1234” to “hello1234”. “Andrew” logs in the Android application.
- Excepted Result
    - 1) Because Jessica has been deleted, Jessica can no longer log in to the Android application.
    - 2) Andrew can not only log in with the "andrew1234", the old password.
  - Actual Result

<input type="checkbox"/>	ID ↕	Username	Password	Nickname ↕	Right
<input type="checkbox"/>	1	Che Sicheng	12345678	Che1998	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	2	Steven	steven1234	VVSHM-Steven	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	3	Andrew	hello1234	1995_Andrew	<a href="#">Edit</a> <a href="#">Delete</a>

**Figure 47.** Administrator Modifying User Successfully



**Figure 48.** Administrator Modifying User on Android Testing

### 6.1.3 Category Management

The administrator managing category was test as follows:

- 1) Testing Step Add a new category “Cosmetic”.
- 2) Edit the name of the seventh category “Transport” to “Transportation”.
- 3) Delete the fourth category “Instrument”.

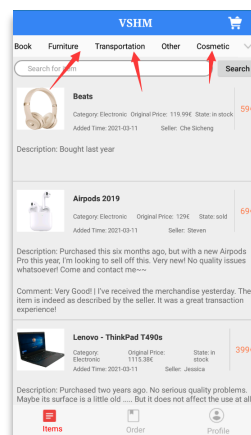
- Excepted Result

- 1) The user successfully logs in to the Android application and can see the newly added category "Cosmetics".
- 2) “Transport” category will be changed to “Transportation” on Android application.
- 3) The category “Instrument” will not be on the Android app.

- Actual Result

ID	Category	Right
1	Electronic	<a href="#">Edit</a> <a href="#">Delete</a>
2	Clothing	<a href="#">Edit</a> <a href="#">Delete</a>
3	Book	<a href="#">Edit</a> <a href="#">Delete</a>
4	Furniture	<a href="#">Edit</a> <a href="#">Delete</a>
5	Transportation	<a href="#">Edit</a> <a href="#">Delete</a>
6	Other	<a href="#">Edit</a> <a href="#">Delete</a>
7	Cosmetic	<a href="#">Edit</a> <a href="#">Delete</a>

**Figure 49.** Administrator Modifying Category Successfully



**Figure 50.** Administrator Modifying Category on Android Testing

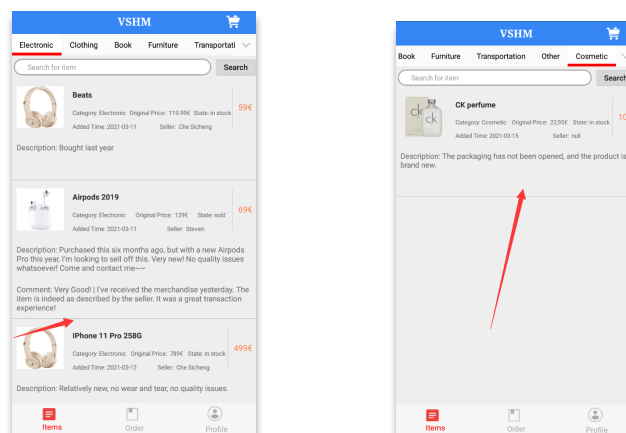
## 6.1.4 Commodity Management

The administrator managing commodity was test as follows:

- 1) Add a new second-hand commodity “CK Perfume” in “Cosmetic” category.
  - 2) Edit the price of the 8<sup>th</sup> commodity “IPhone pro 11 256G”, from 499 to 399.
  - 3) Delete the 7<sup>th</sup> commodity “Lenovo - ThinkPad T490s” .
- Excepted Result
    - 1) There will be a new item “CK perfume” in “Cosmetic” category on the Android application.
    - 2) The price of “IPhone pro 11 256G” from 499 to 399
    - 3) There will not be “Lenovo - ThinkPad T490s” in Electronic.
  - Actual Result

ID	Item Name	Price	Original Price	Status	Right
1	Beats	59	119.99	in stock	Edit Delete
2	'Head First Java'	10	34.55	in stock	Edit Delete
3	Bicycle	59	299	in stock	Edit Delete
4	Airpods 2019	69	129	sold	Edit Delete
5	Old leather shoes	5	Unclear	in stock	Edit Delete
6	Old Desk	12	about 100	in stock	Edit Delete
8	IPhone 11 Pro 256G	499	789	in stock	Edit Delete
9	CK perfume	10	22.95	in stock	Edit Delete

**Figure 51.** Administrator Modifying Commodity Successfully



**Figure 52.** Administrator Modifying Commodity on Android Testing

## 6.1.5 Order Tracking

The administrator managing order was test as follows:

- 1) Ship the “old leather shoes” for user “Andrew”.
- 2) Received the “old leather shoes” for user “Andrew”.

- Excepted Result

- 1) The user "Andrew" who has purchased the commodity can change the order status to "shipped" through the Android app.
- 2) The user "Andrew" who has purchased the commodity can change the order status to "Received" through the Android app

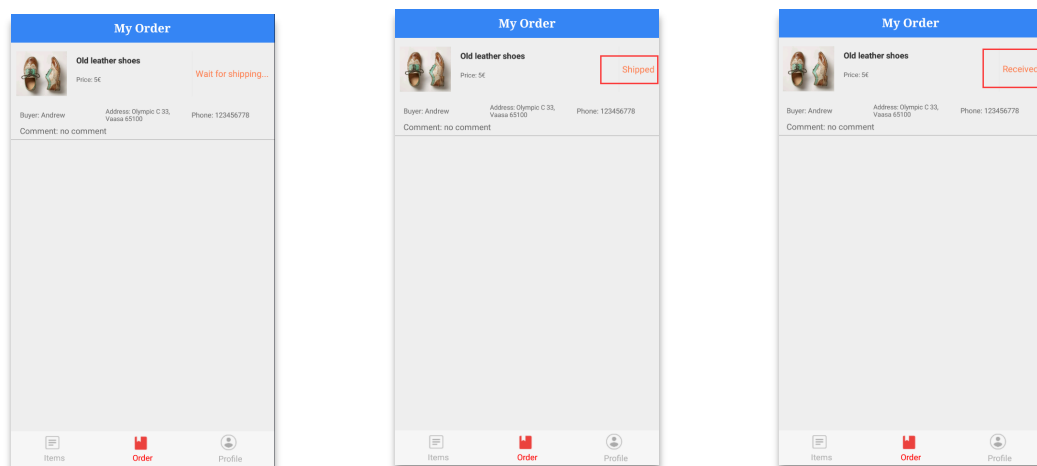
- Actual Result

ID	Item Name	Final Price	Buyer	Address	Telephone	Order Status	Comment	Right
1	Airpods 2019	69	Che Sicheng	Vuorikatu 14-18, Vaas...	0469479099	Received	Very Good! I've recei...	Ship Received
2	Old leather shoes	5	Andrew	Olympic C 33, Vaasa ...	123456778	Shipped	no comment	Ship Received

ID	Item Name	Final Price	Buyer	Address	Telephone	Order Status	Comment	Right
1	Airpods 2019	69	Che Sicheng	Vuorikatu 14-18, Vaas...	0469479099	Received	Very Good! I've recei...	Ship Received
2	Old leather shoes	5	Andrew	Olympic C 33, Vaasa ...	123456778	Received	no comment	Ship Received

**Figure 53.** Administrator Modifying Order Successfully



**Figure 54.** Modifying Order on Android Testing

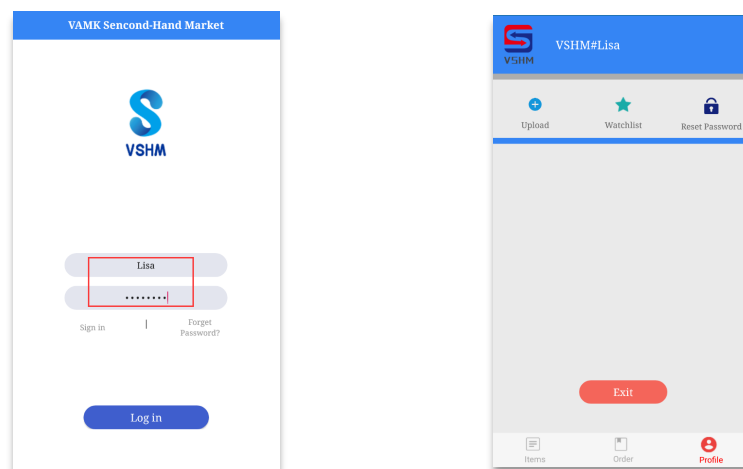
## 6.2 Android Application Testing

### 6.2.1 New User Registration

The user registration was test as follows:

Register a new user "Lisa" from the Android application, the password is "Lisa", the nickname is "VSHM#Lisa", and the phone number is "0469456789".

- Excepted Result
  - 1) The new user Lisa can successfully log in to the VSHM Android application with the account just registered.
  - 2) The user management of the Web background management system can view the information of newly registered users.
- Actual Result



**Figure 55.** Successfully Login Android App

ID	Username	Password	Nickname	Right
1	Che Sicheng	12345678	Che1998	<a href="#">Edit</a> <a href="#">Delete</a>
2	Steven	steven1234	VSHM-Steven	<a href="#">Edit</a> <a href="#">Delete</a>
3	Andrew	hello1234	1995_Andrew	<a href="#">Edit</a> <a href="#">Delete</a>
5	Lisa	lisa1234	VSHM#Lisa	<a href="#">Edit</a> <a href="#">Delete</a>

**Figure 56.** Update in User Management



## 6.2.2 Forget Password

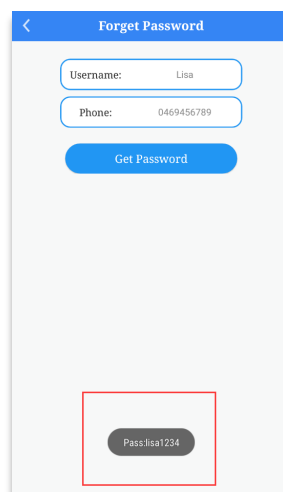
The user retrieving password was tested as follows:

User “Lisa” retrieves the password by phone number.

- Excepted Result

User “Lisa” retrieves the password by phone number, the password is displayed on the page.

- Actual Result



**Figure 57.** Retrieve the Password

## 6.2.3 User Login

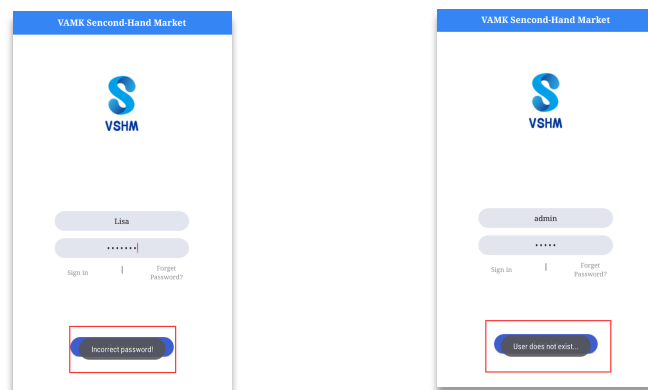
The user login was test as follows:

- 1) Login with wrong password.
- 2) Log in with a user that does not exist.

- Excepted Result

The Android application will be notified of errors.

- Actual Result



**Figure 58.** Error Message Display Successfully

## 6.2.4 Add to Shopping Cart

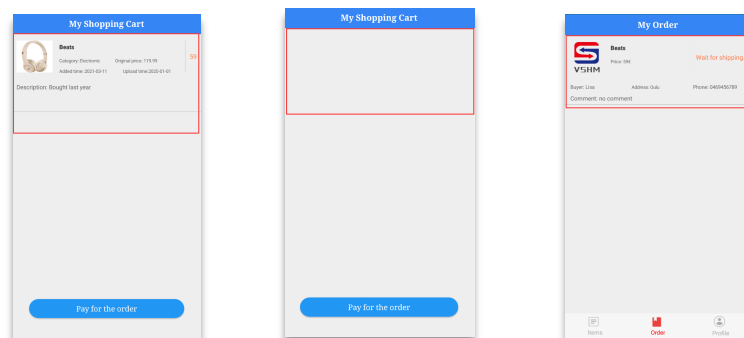
The user adding commodity was tested as follows:

- 1) The user adds "Beats" to the shopping cart.
- 2) Remove "Beats" from shopping cart.
- 3) Check the items in the shopping cart and click "Pay for the order".

- Excepted Result

- 1) Successfully added, successfully removed, and successfully purchased.
- 2) In "My Order", the user sees the "Beats" just purchased.
- 3) The administrator can see the order in "Order Tracking" in the web management system.

- Actual Result



**Figure 59.** Successfully Add, Delete and Pay in Shopping Cart

- Improvement

After the commodity is ordered, the picture disappears. This situation is very unstable and takes place often.

### 6.2.5 Add to Watchlist

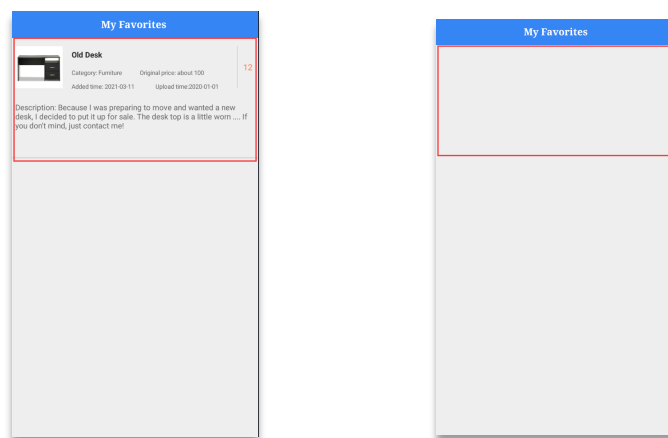
The user adding commodity was tested as follows:

- 1) The user adds "Old Desk" to the Watchlist.
- 2) User removed "Old Desk" from watchlist

- Excepted Result

Successfully add and remove.

- Actual Result



**Figure 60.** Successfully Add and Remove Commodity to/from Watchlist

### 6.2.6 Launch Facebook

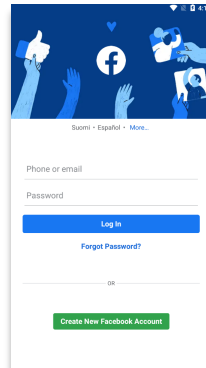
The user calling Facebook was tested as follows:

Long press the commodity to open Facebook.

- Excepted Result

Launch Facebook successfully.

- Actual Result



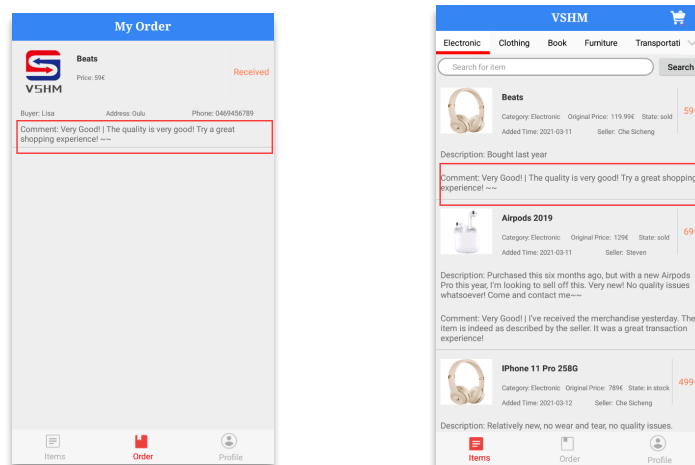
**Figure 61.** Launch Facebook

## 6.2.7 Comments

The users review was tested as follows:

The user evaluates the "Beats" in the "Received" status.

- Excepted Result
  - 1) Comment successfully.
  - 2) The content of user comments can be seen by all users.
- Actual Result



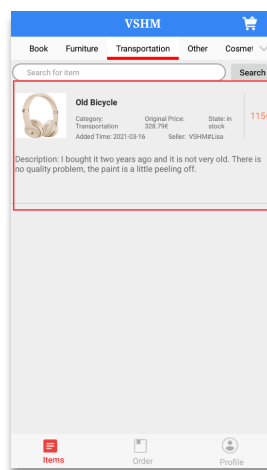
**Figure 62.** Comment Successfully

## 6.2.8 Sell Commodity

The user selling was tested as follows:

Users upload second-hand goods as required.

- Excepted Result
  - 1) Uploaded successfully, you can see it in "Items".
  - 2) After refreshing the web page of the Web management system, the administrator can see the commodity added by the user in the "Commodity Management" module.
  
- Actual Result



**Figure 63.** Upload Commodity Successfully

<input type="checkbox"/>	ID ↕	Item Name	Price ↕	Original Price ↕	Status ↕	Right
<input type="checkbox"/>	1	Beats	59	119.99	sold	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	2	"Head First Java"	10	34.55	in stock	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	3	Bicycle	59	299	in stock	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	4	Airpods 2019	69	129	sold	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	5	Old leather shoes	5	Unclear	sold	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	6	Old Desk	12	about 100	in stock	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	8	iPhone 11 Pro 256G	499	789	in stock	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	9	CK perfume	10	22.95	in stock	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	10	Old Bicycle	115	328.79	in stock	<a href="#">Edit</a> <a href="#">Delete</a>

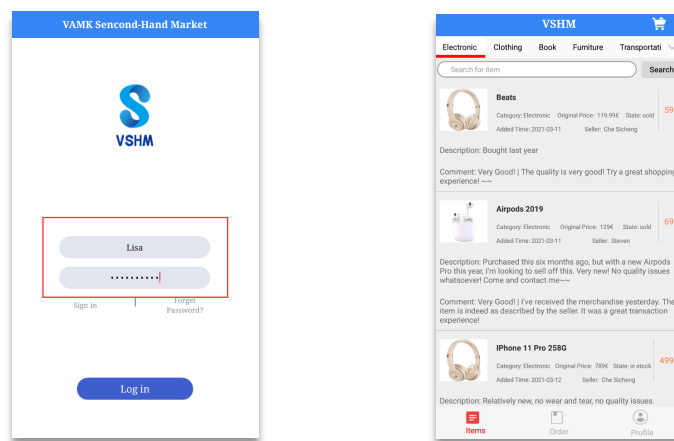
**Figure 64.** Upload Commodity Successfully in Commodity Management

## 6.2.9 Reset Password

The user resetting password was tested as follows:

The user resets his password.

- Excepted Result
  - 1) The user can successfully log in to the Android application with the reset password.
  - 2) The administrator can see the changed user password in "User Management".
  
- Actual Result



**Figure 65.** Successfully Login After Resetting Password

<input type="checkbox"/>	ID	Username	Password	Nickname	Right
<input type="checkbox"/>	1	Che Sicheng	12345678	Che1998	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	2	Steven	steven1234	VVSHM-Steven	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	3	Andrew	hello1234	1995_Andrew	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	5	Lisa	lisa111111	VSHM@Lisa	<a href="#">Edit</a> <a href="#">Delete</a>

**Figure 66.** Reset the Password Successfully in User Management

## 7 CONCLUSIONS

This project was mainly to develop an Android application to provide users with an online trading platform for second-hand commodity. The project consists of two parts, the server, and the client. For the server-side, Web Application, its main functions include managing the information of users, second-hand goods, and orders. The client is the main part of the project, an application developed based on Android. The main functions include the user registration and login, viewing of all second-hand goods, purchase of the user's favorite commodity, upload of second-hand items the user want to sell, and viewing the status of one's own orders , and commenting the commodity that have been received.

After the project development was completed, all the functions involved in must-have requirement and should-have requirement had been realized. The application was tested on different browsers and real Android devices.

During making of the thesis, a deep understanding of learning Java and Android was received. At the same time, I learned that I need more advanced studies and improvements. Due to time and learning constraints, many aspects and functions of the application still need to be enhanced. In future study, these aspects and functions will be further improved and perfected to make the application more professional and completed.

### 7.1 Future Work

Although all basic functions have been implemented after the project was completed, many aspects still need to be developed and enhanced in this application. So that other functions can be developed in the future, the following functions can be improved in future learning:

- Security for clients.
- Design and develop a buyer and seller communication function within the application, without having to call on Facebook.

## REFERENCES

- /1/ Platform Architecture. Accessed 21.03.2021  
<https://developer.android.com/guide/platform>
- /2/ Guo Hongzhi. Android application development in detail [M]. Beijing: Electronic Industry Press 2016.6 417-420. (Chinese)
- /3/ Ma Yanjun. 2015. App development based on Android system [J]. Technology and Enterprise, 22:87-88. (Chinese)
- /4/ Simple Example of MVC (Model–View–Controller) Architectural Pattern for Abstraction. Accessed 21.03.2021  
<https://www.codeproject.com/Articles/25057/Simple-Example-of-MVC-Model-View-Controller-Design>
- /5/ Burbeck, Steve. 1992. Applications Programming in Smalltalk-80:How to use Model–View–Controller (MVC)
- /6/ JFinal – manual. Accessed 21.03.2021  
<https://github.com/jfinal/jfinal-manual>
- /7/ What is a REST API? Accessed 21.03.2021  
<https://www.redhat.com/en/topics/api/what-is-a-rest-api>
- /8/ Introduction JSON. Accessed 21.03.2021  
<https://www.json.org/json-en.html>
- /9/ JSON Object. Accessed 21.03.2021  
<https://developer.android.com/reference/org/json/JSONObject.html>



/10/ Zhou Y, Yin SQ, Wang DQ, et al. 2016. Research on building an App development platform based on Eclipse and Android system[J]. Journal of Qingdao University (Engineering Technology Edition), 31(3):49-53. (in Chinese)