Ministry of Education of the Republic of Belarus

Institution of Education

Belarusian State University

of Informatics and Radioelectronics

Faculty of Information Technologies and Control

Information Technologies in Automated Systems Department

|  |
| --- |
| *Diploma project submission permitted* |
| The Head of the Information Technologies in Automated Systems Department |
| \_\_\_\_\_\_\_\_\_\_\_\_А.А. Naurotsky |

EXPLANATORY NOTE

Diploma Project

Automated SECOND-HAND TRADING COMPANY INFORMATION System

BSUIR 1-53 01 02 01 002 DP

|  |  |  |
| --- | --- | --- |
| Student |  | Li Boyi |
| Supervisor |  | N. V. Khajynava |
| Advisors: |  |  |
| * *from the Information Technologies in Automated Systems Department* |  | N. V. Khajynava |
| * *for the Economic feasibility study* |  | I. V. Smirnov |
| Standards Compliance Inspector |  | N. V. Batin |
| Reviewer |  |  |

Minsk 2023

**ABSTRACT**

AUTOMATED SECOND-HAND TRADING COMPANY INFORMATION SYSTEM: Diploma Project / Li Boyi – Minsk : BSUIR, 2023, – Explanatory Note – 72 p., drawings – 6 A1 sheets.

The diploma project deals with the design of a computerized system for the information system of a second-hand goods trading company. The structure of an online second-hand item trading system, the management system, the related information flows and the available computerization tools were analyzed. Solutions were provided for the computerization of several tasks addressed by the system management, such as used item posting and description management, user information management, order processing and checking, and payment gateway. These solutions include appropriate algorithms, description of information flow, database design, data validation and checkout procedures.

The software has been designed to perform these tasks. Programming and data management tools for software implementation include IntelliJ IDEA and MySQL. software operation modes have been provided for these categories of users, as well as an administrator mode for software installation and setup. Manuals for users and administrators have been prepared. A set of test cases has been prepared and described.

An economic feasibility study has been conducted to confirm the cost effectiveness of the project. The expected economic results generated by the designed software application have been calculated.

The Ministry of Education of the Republic of Belarus

BELARUSIAN STATE UNIVERSITY

OF INFORMATICS AND RADIOELECTRONICS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| Faculty | Information Technologies and Control | Department | Information Technologies in Automated Systems | |
| Speciality | 1-53 01 02 | Specialization | | 01 |

APPROVED BY

The Head of the Information Technologies in Automated Systems Department

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Signature)

*“\_\_\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023*

THE TASK

for the diploma project

to be prepared by the student Li Boyi

**1** The theme of the project

*Automated Second-Hand Trading Company Information System*

*Order No 705-c dated 22.03.2023.*

***2*** *Project preparation deadline: 01 06 2023*

**3** Initial data to the project: Computerization of tasks to be solved by an online trading platform for second-hand items: buying, posting, order processing, after-sales service. Software to be used: Windows 7, JavaScript, HTML, MYSQL, IntelliJ IDEA.

**4** Explanatory note content:

Introduction

**1** General information about company

**2** Information system software structure

**3** Implementation of information systems

**4** Economic feasibility study

Conclusion

|  |  |  |
| --- | --- | --- |
| **5** Drawings: | | |
| **1** Second-hand trading company information system organization structure (А1, 1 sheet) | | |
| **2** Second-hand trading company information system flowchart structure (А1, 1 sheet) | | |
| **3** Second-hand trading company information system IDEF0 diagrams (А1, 1 sheet) | | |
| **4** Second-hand trading company information system activity diagram (А1, 1 sheet) | | |
| **5** Second-hand trading company information system use case diagram (А1, 1 sheet) | | |
| **6** Second-hand trading company information system screenshots (А1, 1 sheet) | | |
| **6** The task for the economics feasibility study | | |
| Second-hand trading company information system makes efficient use of computer resources and simplifies the process of data calculation and information processing in the system. The system is user-friendly and highly efficient, which significantly reduces costs. | | |
| Given by |  | Smirnov Igor Viktorovich |

project progress schedule

|  |  |  |  |
| --- | --- | --- | --- |
| Project progress stage | Stage size (%) | Stage deadline | Note |
| Collection and study of materials of automated second-hand trading company information system | 20 | 10.04.2023 |  |
| Information system structure design. Development of algorithms. Database design | 30 | 18.04.2023 |  |
| Software design | 50 | 10.05.2023 |  |
| Software debugging and testing | 60 | 20.05.2023 |  |
| Economic feasibility study | 80 | 25.05.2023 |  |
| Preparation of the explanatory note and drawings | 100 | 01.06.2023 |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| The task is given on |  | Supervisor | |  | | N. V. Khajynava |
| (date)  I hereby accept the task for implementation | | |  | | Li Boyi | |

**CONTENTS**

[Introduction 6](#_Toc137209373)

[1 General Information about Company 7](#_Toc137209374)

[1.1 Organizational structure and automated second-hand trading company information system 7](#_Toc137209375)

[1.2 Existing information systems 8](#_Toc137209376)

[1.3 Comparison with similar systems worldwide 9](#_Toc137209377)

[1.4 Task statement 10](#_Toc137209378)

[2 Information System Software Structure 12](#_Toc137209379)

[2.1 Functional diagrams 12](#_Toc137209380)

[2.2 Algorithms 14](#_Toc137209381)

[2.3 Information system security 16](#_Toc137209382)

[2.4 System hardware and software requirements 22](#_Toc137209383)

[2.5 Ergonomics 23](#_Toc137209384)

[3 Implementation of Information System 24](#_Toc137209385)

[3.1 UML diagrams 24](#_Toc137209386)

[3.2 Software implementation programming tools 29](#_Toc137209387)

[3.3 Software structure 31](#_Toc137209388)

[3.4 Function description 32](#_Toc137209389)

[3.5 User’s guide 36](#_Toc137209390)

[3.6 Administrator’s guide 41](#_Toc137209391)

[3.7 Further development 46](#_Toc137209392)

[4 Economic Feasibility Study 48](#_Toc137209393)

[4.1 Characteristic of an economic case of the project 48](#_Toc137209394)

[4.2 Calculation of cost of materials for project accomplishment 48](#_Toc137209395)

[4.3 Calculation of a base salary of the personnel occupied with accomplishment of works 49](#_Toc137209396)

[4.4 Calculation of an additional salary of the contractors, including the various payments 49](#_Toc137209397)

[4.5 Calculation of deductions to social insurance 50](#_Toc137209398)

[4.6 Calculation of expenses on scientific business trip 50](#_Toc137209399)

[4.7 Calculation of common enterprise expenses 50](#_Toc137209400)

[4.8 Calculation of the complete cost value 51](#_Toc137209401)

[4.9 On level of profitability in percentage 51](#_Toc137209402)

[4.10 Calculation of the price of the project 51](#_Toc137209403)

[4.11 Calculation of the tax to value added (VAT) 51](#_Toc137209404)

[4.12 Calculation of the price of the project taking into account the VAT 52](#_Toc137209405)

[Conclusion 53](#_Toc137209406)

[References 54](#_Toc137209407)

[Appendix A (informative) Code Description 55](#_Toc137209408)

Diploma Project Register  [73](#_Toc137209409)

# **INTRODUCTION**

The development of the times, the progress of technology, the alternation of old and new items is changing day by day, every family, everyone will have updated items in the idle, how to deal with these items has also become a daily concern. Some people will get to the local second-hand market to trade, some people will be far from the second-hand market or second-hand items are not easy to carry and sell it as scrap or even throw away. This is not only bad for the environment, but also a waste of social resources.

Second-hand shopping has seen a boom with the popularity of the internet and the emergence of traditional second-hand markets on the web. Once the preserve of high street charity stores, flea markets and antique fairs, you can now find all sorts of treasures online. The internet has also greatly facilitated the growth of second-hand trading, with the convenience of the internet making more and more people accustomed to buying and selling second-hand goods online, rather than having to visit traditional second-hand markets. In order to give full play to the convenience of the Internet and establish a platform for information distribution, second-hand trading platforms were born. [1]

The birth of the second-hand trading platform makes second-hand trading no longer restricted to the second-hand marketplace. The convenience of the network allows most people to choose to publish second-hand trading information on the second-hand trading platform, without having to leave home to buy and sell second-hand items, which is very convenient for the people. It can be said that the second-hand trading network has greatly promoted the enthusiasm of the people to buy and sell second-hand goods, and also promoted the effective use of social resources.

As a used goods trading system. First of all, users can post used goods as sellers and buy goods as buyers, so user roles should have both buyer and seller roles. And the system should implement a user communication subsystem for communication between buyers and sellers, so that buyers and sellers can communicate in the communication system. The administrator can manage users, manage posted products, manage user's orders, etc. Users can also apply for arbitration of orders, and the administrator can arbitrate orders.

**1 GENERAL INFORMATION ABOUT COMPANY**

## 1.1 Organizational Structure and Automated Second-Hand Trading Company Information System

The organizational structure of a second-hand trading company information system, see Figure 1.1 below, outlines how specific activities are handled to accomplish strategic tasks, and rules, roles, and obligations are all part of these activities. The organizational structure also determines the flow of information between the various departments within the company. [2]

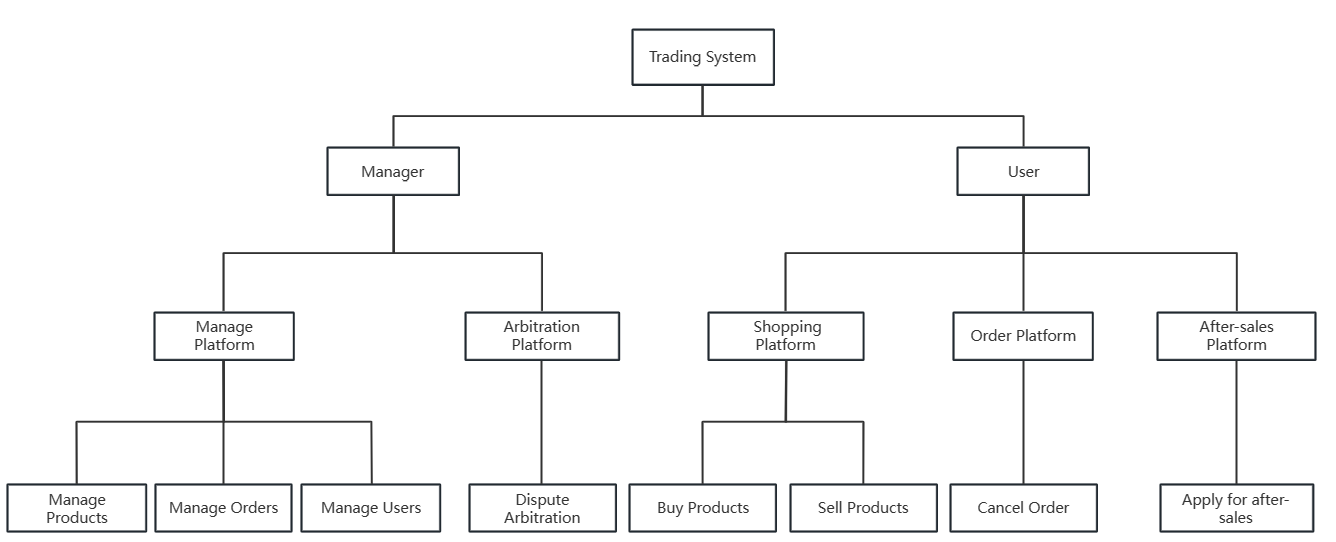


Figure 1.1 – Organization Structure in an Online Trading System

The system is structured into two sub systems, the administrator system and the user system. These two systems are used by administrators and users respectively.

The administrator can manage the products and orders posted by users in the platform and the information of users in the platform in the administrator system. The administrator can also perform the function of arbitration, and can intervene and mediate when a user raises a question about an order.

As for customers, they can browse all product descriptions, prices and choose whether to buy the product based on if they are not satisfied with the used items or if the products do not match the description, they can apply for after-sales service, return the products or get compensation. At the same time, customers can also act as sellers and release their used items to the platform for sale, and can customize the pictures, prices, descriptions, etc. of these items.

Inputs to the system include the keyboard, mouse and information used when logging in, which help users access the system with their private accounts. The pages and functions differ due to the different identities.

## 1.2 Existing Information Systems

eBay is one of the largest online auction and retail platforms in the world, founded in 1995 and headquartered in California, USA. eBay provides a trading platform for sellers and buyers to trade various goods, including but not limited to used items, new items, art, cars, electronics, fashion accessories, toys, and more.

On eBay, sellers can create their own online stores or simply list items for sale in the marketplace. Buyers can browse and search for items to purchase and bid on items in auctions. eBay also offers various tools and services for sellers, such as listing optimization, shipping and fulfillment, and customer service support.

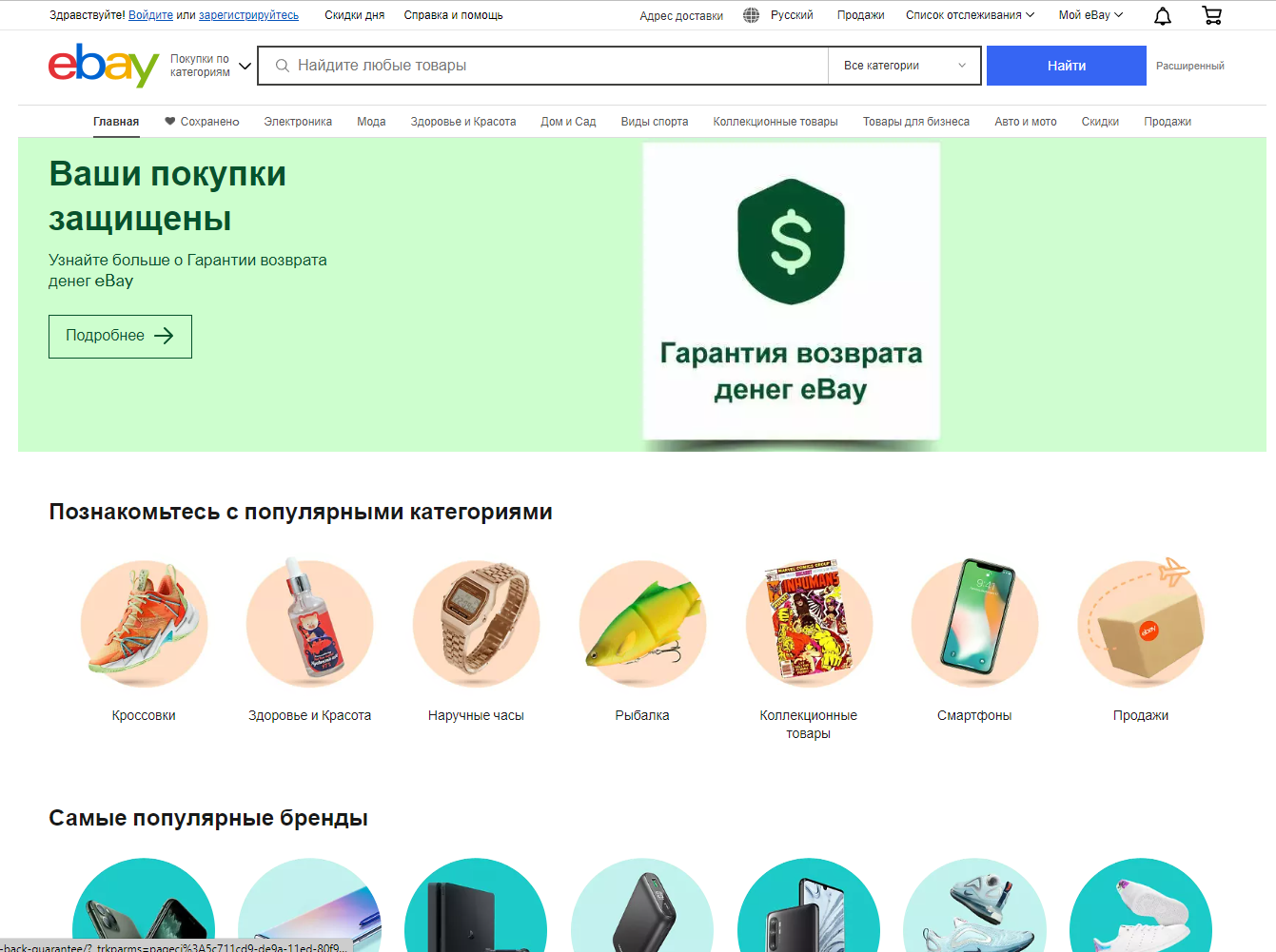


Figure 1.2 – Existing information System eBay

**1.2.1** eBay Background Management

eBay’s background management includes a variety of tools and services to help sellers optimize their listings, manage orders, process payments, and provide customer service support. eBay also has strict policies to ensure fair and safe trading practices, such as feedback ratings and buyer protection programs.

**1.2.2** Advantages of eBay

eBay offers a wide range of items for sale, including rare and hard-to-find products, providing a diverse selection that may be challenging to find in traditional retail stores. With its user-friendly interface, eBay is easily accessible from anywhere with an internet connection, making it convenient for both buyers and sellers. The auction format and global market competition on eBay often result in competitive pricing, offering potential savings compared to retail prices. Moreover, eBay's large user base allows sellers to reach a broad audience of potential buyers, making it an advantageous platform for selling niche or unique items to a wider market.

**1.2.3** Drawbacks of eBay

While eBay is a popular online marketplace, it does have certain drawbacks. Sellers may incur expenses such as listing, sales commissions and payment processing that affect their overall profits. The high level of competition among sellers may make it difficult for them to differentiate themselves and attract buyers, especially when offering similar items. Although eBay offers a buyer protection program, there is still a risk of fraud that sellers need to be aware of. In addition, eBay's strict shipping rules and guidelines may limit shipping options and increase costs. Feedback systems, while helpful in building trust, can also lead to negative feedback that can affect a seller's reputation. Technical issues such as site downtime and payment system failures can inconvenience buyers and sellers.

**1.3 Comparison with similar systems worldwide**

Table 1.1 compares the Second-Hand Trading Company Information System, eBay, Craigslist, and Facebook Marketplace. The Automated Secondhand Exchange Company Information System has the advantages of ease of use, free trading, a wide selection of product labels, and online communication, which increases flexibility but does not have a high penetration rate. eBay has the advantages of a high global reputation, a large number of users The advantages of a large number of users and a wide range of products, offering both auction and fixed-price transactions, where buyers and sellers can evaluate and provide feedback, increase trust, but the transaction process is relatively complex, with a learning curve and security issues. craigslist has the advantages of global coverage, free posting and browsing of information, and multiple item categories, but lacks an authentication and trust system, with an old interface and uneven information quality. Facebook Marketplace is based on Facebook social network, with many users and strong geography, can view sellers' personal information and reviews, and is easy to share and promote, but the transaction process and security mechanism are relatively simple, with privacy and security issues, and limited sorting and filtering functions.

Table 1.1 - The comparison with similar system

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Company System | Security Level | Communication between users | Multi-Category Products | Trading Method |
| Second-Hand Trading Company Information System | High | Yes | Yes | Fixed price |
| eBay | Medium | Yes | Yes | Auction, fixed price |
| Craigslist | Low | No | Yes | Fixed price |
| Facebook Marketplace | Medium | Yes | Yes | Fixed price |

## 1.4 Task Statement

**1.4.1** System Description

The system is designed to be a large online system for trading second-hand items. The system contains two sub-systems, the user system and the administrator system. The user system mainly undertakes the functions of user transactions and communications, while the administrator system mainly undertakes the management of user information and order information, as well as the maintenance of user rights and interests.

**1.4.2** Service Object

The system is mainly friendly to users who want to buy second-hand goods or sell second-hand goods, especially for those who have high-value second-hand items around but are unused and can sell them on the platform. It also provides convenience to those users who are keen on collecting, as these items may appear in the platform, so they can find surprises in the system.

**1.4.3** System Functions

The initial goal of the system is to facilitate the free trade of second-hand items for users. In order to provide accurate and secure services, customers phone numbers, mailing addresses, nicknames and other information need to be provided to the system, and all customers' information will be stored in the official database for security. At the same time, in order to protect user's rights, the administrator can mediate users disputes in the administrator system, so that users can trade with more confidence.

In the user system, finding used products is efficient and comfortable, as the search function is set up to find specific products based on keywords. All products are divided into different categories on the main page of the system, and buyers can add any product to the shopping cart if they want to buy it, and can check out or delete it at any time.

In the administrator system, the administrator can manage the products and orders posted by users in the platform and the information of users in the platform in the administrator can also perform the function of arbitration, and can intervene and mediate when a user raises a question about the administrator can also perform the function of arbitration, and can intervene and mediate when a user raises a question about an order.

**1.4.4** Advantages of the Automated Second-hand Trading System

Satisfaction: Having a website will provide more convenience for customers and leads. Giving customers a higher willingness to buy. Many people will be more inclined to visit a website rather than drive to your physical store to browse products. From the customer's point of view, it is better for them if they don't have to ask any questions. They can find what they are looking for on the website.

Increase customers: Most businesses are popular locally, but have a hard time attracting customers from outside the city. And it's not just limited to other cities, it's worldwide. The Internet provides a global community. With a website, business can expand more widely.

Accessibility: An offline store will have store closing hours. An online site can be accessed at any time of the day or night. People will focus on the website rather than going to the store because it is easier to access. Just make sure to post enough information about the products and services.

Access to information: A website has information traceability, where the back office of the website can see how many people have visited the website, or how many people have sent emails, etc. in the back office. And the pages can be updated and viewed at any time.

Better relationships: Having a website allows you to build better relationships with your customers. Administrators can send messages to customers instantly via email. In addition, customers can view products online and also leave feedback for the business. It is always best to send messages to customers. It is vital to build a good relationship with them.

**2** **INFORMATION SYSTEM SOFTWARE STRUCTURE**

**2.1** **Functional Diagrams**

**2.1.1** IDEF0 Diagram

The second-hand trading company information system IDEF0 diagrams see Figure 2.1 and Figure 2.2, IDEF0 diagrams shows how the system integrates the inputs, controls, outputs, and mechanisms of the functional decomposition. IDEF0 models a system as a set of activities (functions) using only two graphic symbols: boxes and arrows. [3]

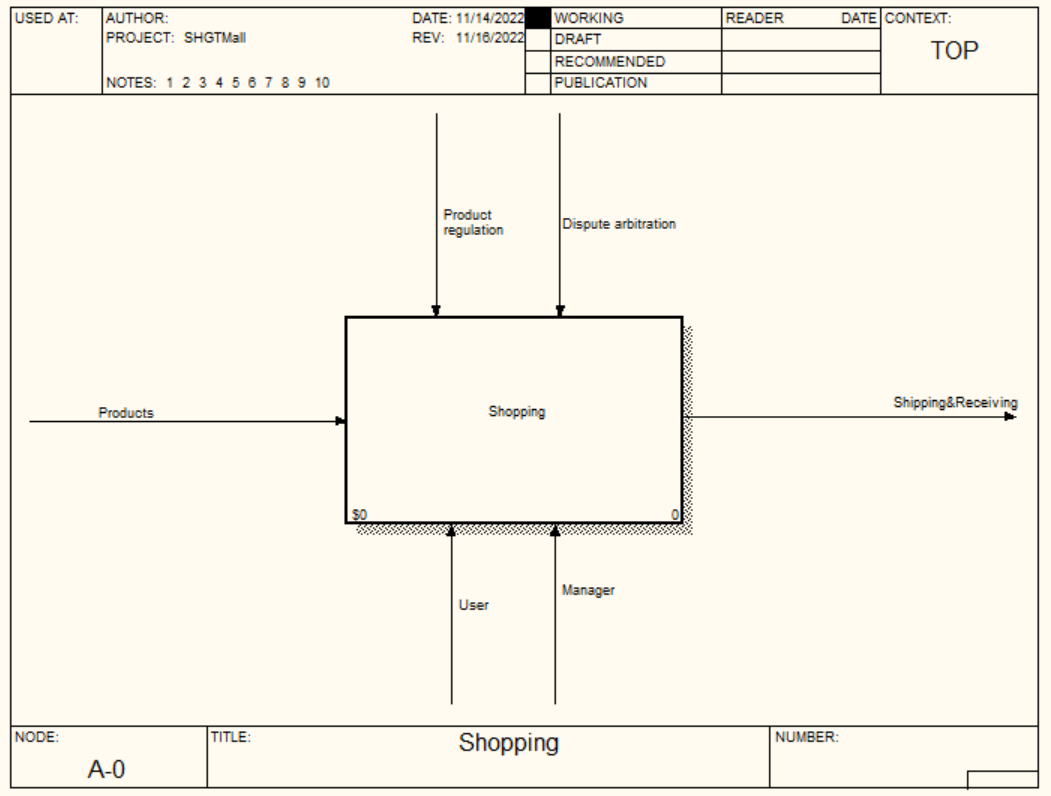


Figure 2.1 - IDEF0 Diagram

The above is the overall input and output of the system, and the decomposition diagram of the system will be shown in Figure 2.2.

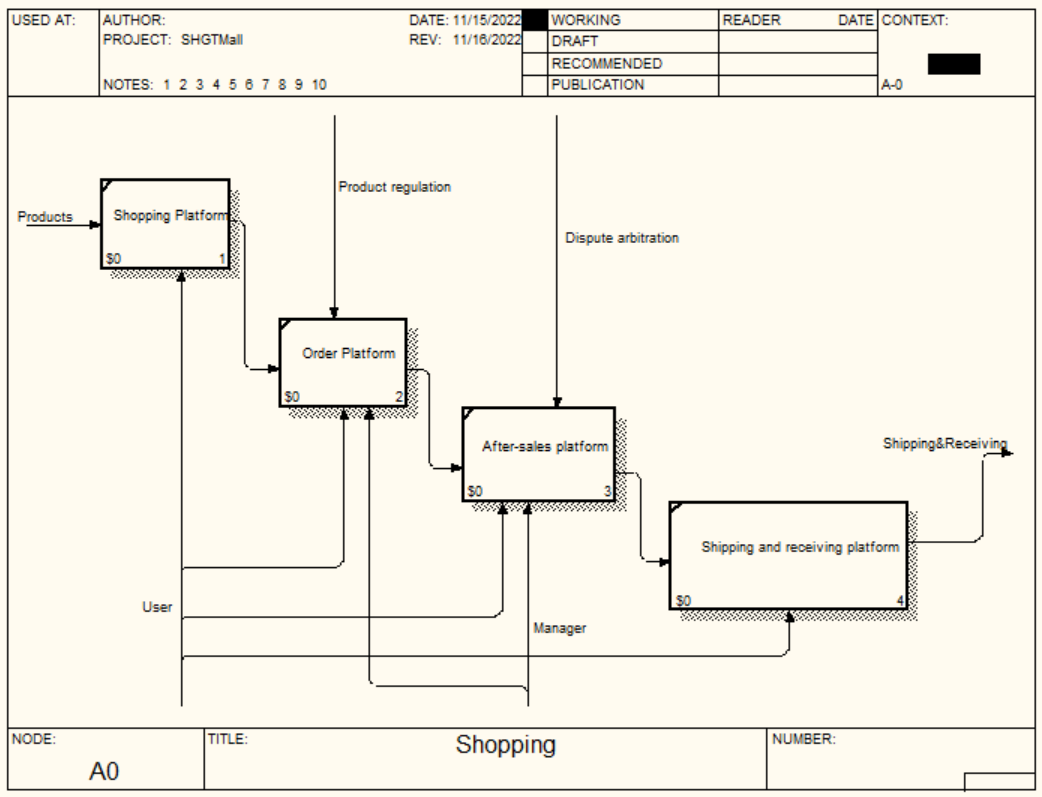


Figure 2.2 - IDEF0 Decomposition Diagram

In this second-hand trading company information system, there is an input named product. Users and administrators are used on the mechanism arrows. Goods reconciliation and dispute arbitration are on the control arrow space. The output is just shipping and delivery.

The input of the goods is the shopping platform and then goes to the order platform. The order platform is under the control of the goods verification, then the after-sales platform, which is under the control of the arbitration of disputes, followed by the delivery platform, which directly outputs the shipment and receipt of goods. The user mechanism will be used in the shopping platform, the order platform, the after-sales platform, and the shipping and receiving platform. The administrator mechanism will be used in the order platform and the after-sales platform.

**2.2 Algorithms**

Algorithm means: a limited set of rules or instructions to be followed in a computation or other problem-solving operation, or a procedure for solving a mathematical problem in a limited number of steps, often involving recursive operations. [4]

**2.2.1** Flow Chart

The following is a flowchart of the administrator of the automated second-hand trading system see Figure 2.3. The flowchart below mainly describes the two main processes in the system, The process of adding product categories and uploading banners for administrators. On the left is the process of adding product categories, administrators need to set up the details of product categories, then select the menu page for display, and finally submit, and the right branch is the process of uploading banners, administrators need to set up the details, then select the type of banner, and finally submit, and finally end these two processes.

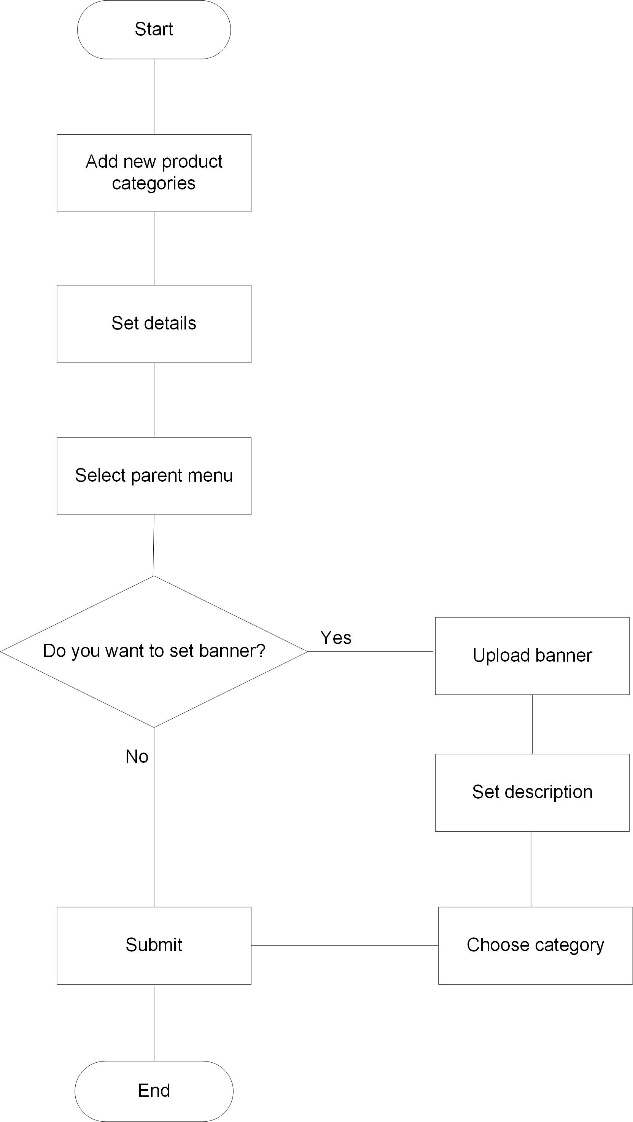


Figure 2.3 - Information System Flowchart Structure of Administrator

The following is a flowchart of the user of the automated second-hand trading system see Figure 2.4. The flowchart below mainly describes the two main processes in the system, the purchase and posting of goods by users. In the left branch is the process of buying second-hand goods, the user first selects the second-hand goods he likes, enters the selected goods to check the details of the goods, then the user makes a choice whether he wants to buy the goods, if he wants to buy then he enters the payment link to pay, then the seller will ship the goods according to the delivery address filled by the buyer, if he doesn't want to buy, he can choose to continue to choose. The right branch is for uploading used products. When uploading their used products, users can set the corresponding details of the product, such as price, picture, description, etc., and finally end these two processes.

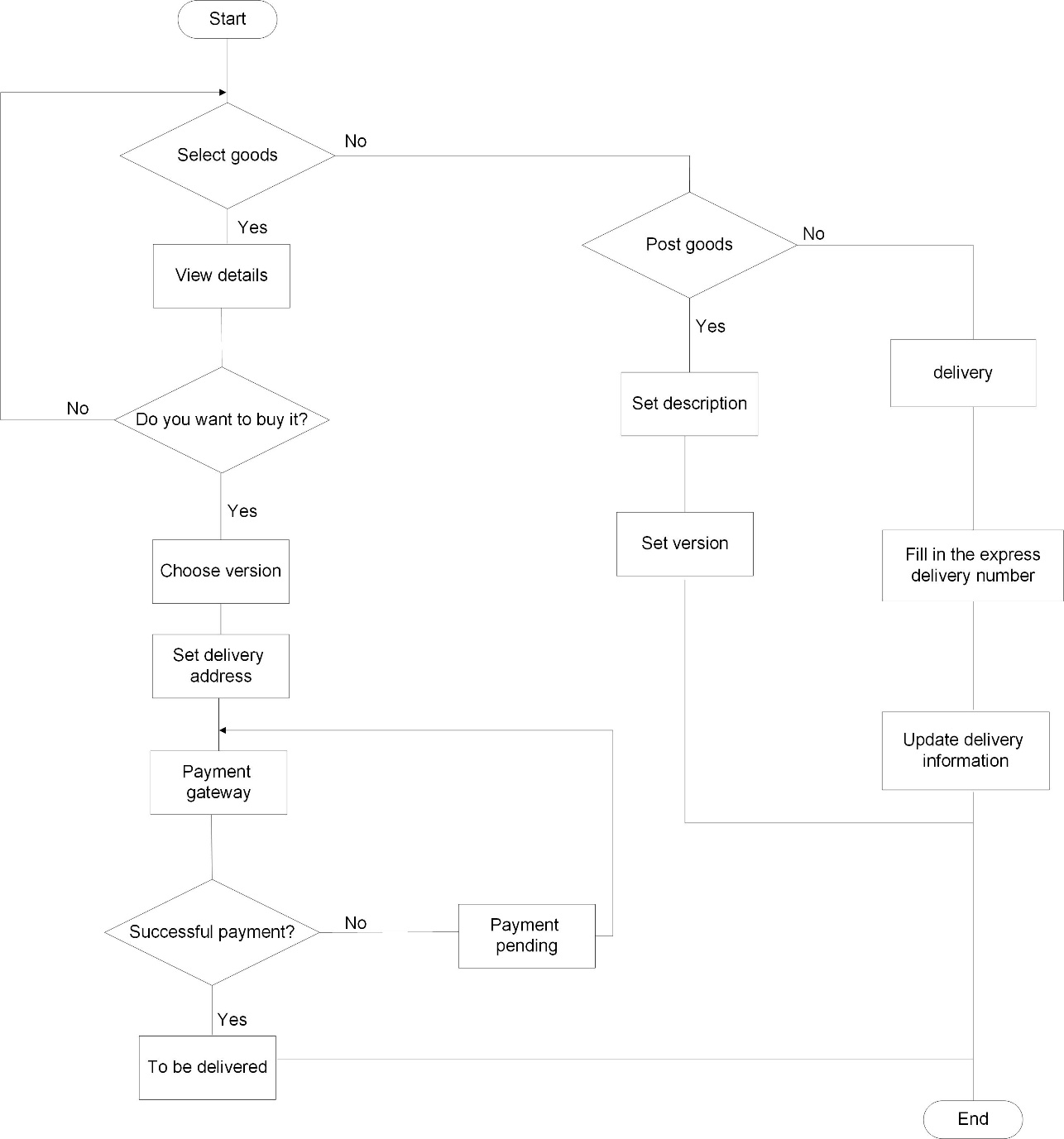


Figure 2.4 - Information System Flowchart Structure of User

**2.3 Information System Security**

**2.3.1** DFD

A data flow diagram of the second-hand trading company information system is shown in Figure 2.5. It is a visual representation of how data flows through the system or process. The main components of a DFD include:

* process: It represents a specific function or action performed on the data. Processes can range from simple calculations or transformations to more complex operations;
* data flow: It represents the movement of data from one component to another. Arrows are used to depict the direction of data flow;
* data store: It represents the storage or repository of data within the system. Data stores can include databases, files, or any other data storage mechanism;
* external Entity: It represents an external entity that interacts with the system but is outside its boundary. External entities can be users, other systems, or organizations that send or receive data from the system.

By using these components and their interconnections, the DFD provides a clear and concise overview of how data is processed and transferred through the system.

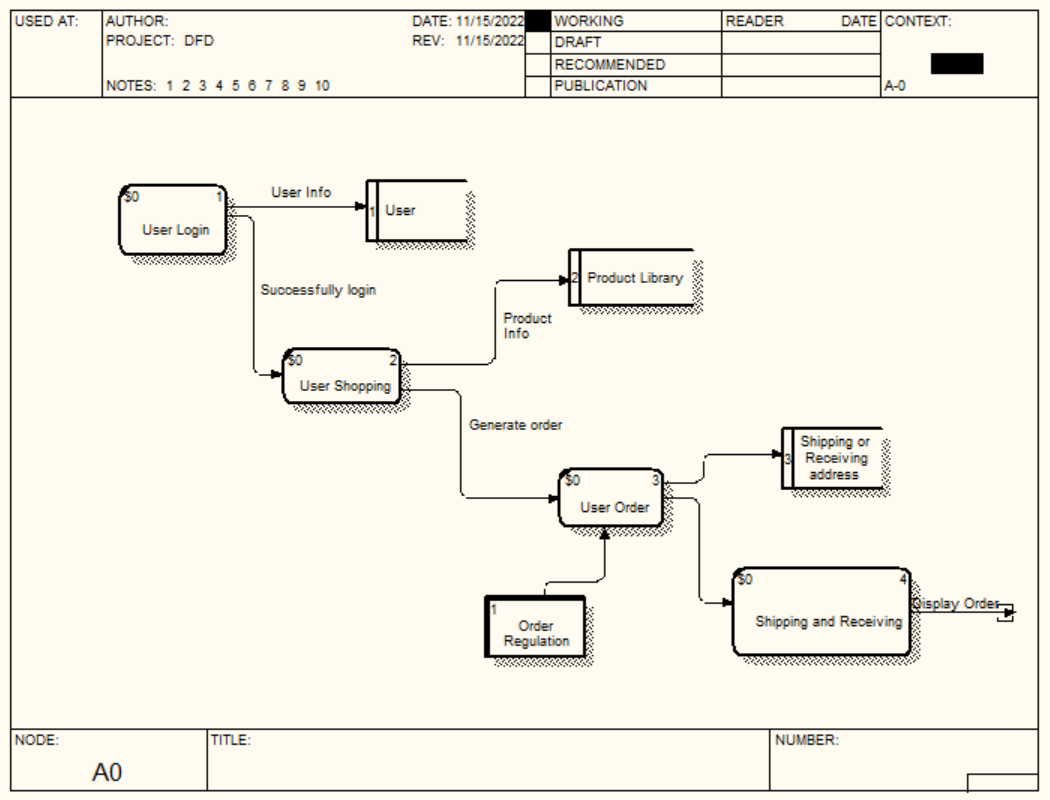


Figure 2.5 - DFD Diagram

The main process starts from user login, the user's information is provided by the external entity User, the user enters the shopping state after successful login, the information of the product is stored in the product database, the user generates an order after shopping into the user's order state, at this time the order will be processed, the seller will ship the goods to the corresponding receiving address filled by the buyer, and the process will end after the buyer receives the goods.

**2.3.2** Database

A database model is primarily a type of data model. Depending on the model in use, a database model can include entities, their relationships, data flow, tables and more. For example, within a hierarchal database mode, the data model organizes data in the form of a tree-like structure having parent and child segments. The logic model of system shows in Figure 2.6.

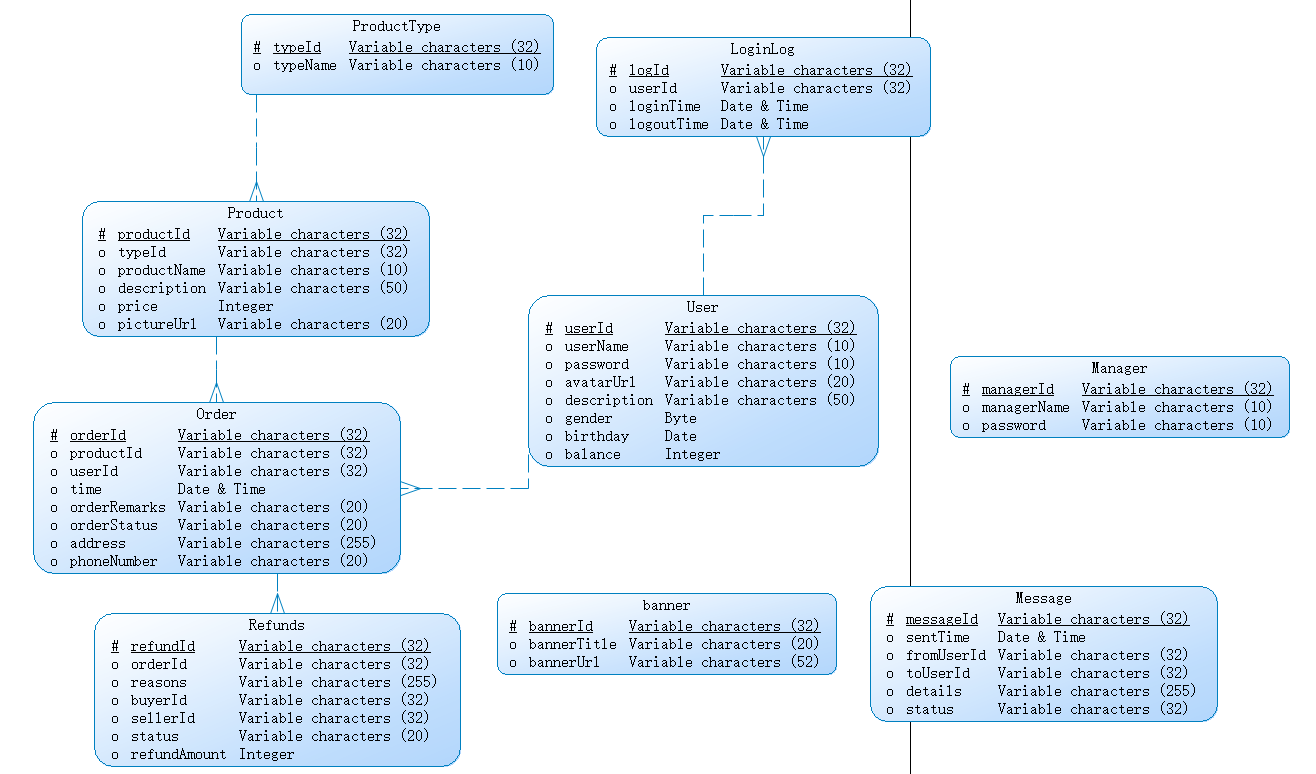


Figure 2.6 - Logic Model

For the logic model figure above, the following is related description.

Entity: There are 9 entities in the database, which are `ProductType`, `Order`, `Product`, `Refunds `, `banner`, `Message`, `Manager`, `User `, `LoginLog`.

Relationships and Foreign Keys: In total, there are 5 relationships between entities. In `Order`, the `productId` is a foreign key related with the primary key of entity `Product`, which means that an item can exist in an order.

The more details can see in the Figure 2.7.

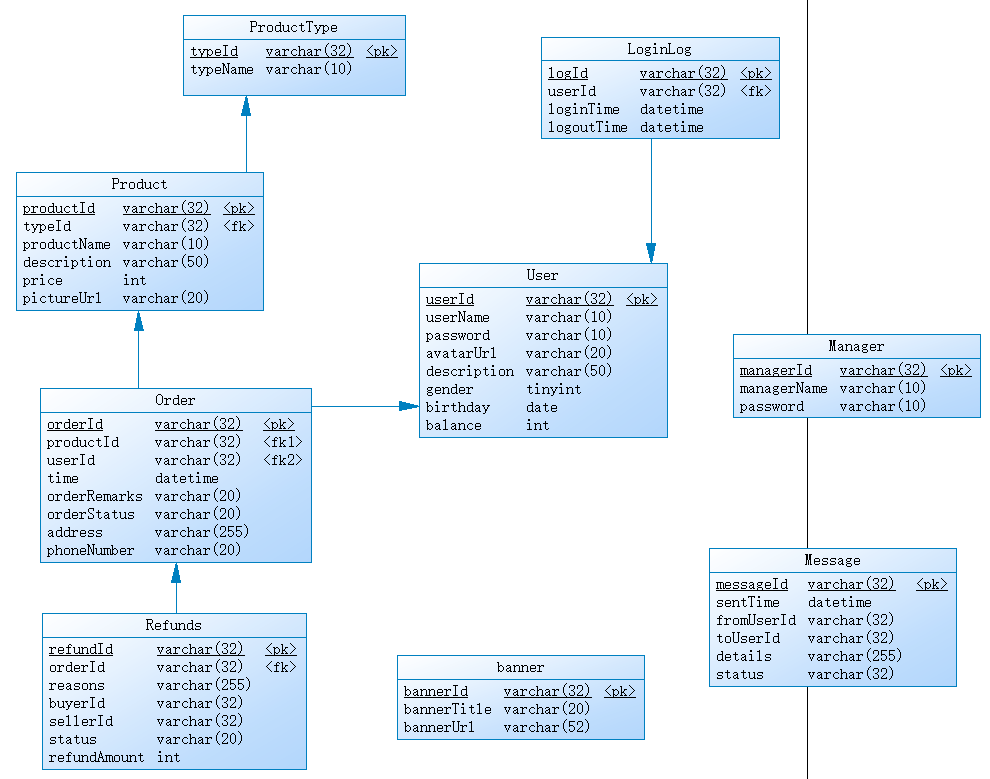


Figure 2.7 - Physical Model

The following are tables designed for the system. The “Product” table stores information about product, shown in Table 2.1. The Product table contains a foreign key “typeId”, which is the primary key of the “ProductType” table.

Table 2.1 - Structure of “Product” table

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Field size  (bytes) | Description |
| productId | Varchar | 32 | Product ID |
| typeId | Varchar | 32 | Product Type ID |
| productName | Varchar | 10 | Product Name |
| Description | Varchar | 50 | Product Description |
| Price | Int | 4 | Product Price |
| pictureUrl | Varchar | 20 | Product Picture URL |

The “ProductType” table stores information about product type, shown in Table 2.2. It mainly contains product type Id and product type name.

Table 2.2 - Structure of “ProductType” table

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Field size  (bytes) | Description |
| typeId | varchar | 32 | Product Type ID |
| typeName | varchar | 10 | Product Type Name |

The “Order” table stores information about order, shown in Table 2.3. The table contains a foreign key “productId”, which is the primary key of the “Product” table. The purpose of this foreign key is to relate the two tables so that various products can be displayed in the order.

Table 2.3 - Structure of “Order” table

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Field size  (bytes) | Description |
| orderId | varchar | 32 | Order ID |
| productId | varchar | 32 | Product ID |
| userId | varchar | 32 | User ID |
| time | datetime |  | Order Create Time |
| orderRemarks | varchar | 20 | Orders Remarks |
| orderStatus | varchar | 20 | Order’s Status |
| address | varchar | 255 | Receiving Address |
| phoneNumber | varchar | 20 | Receiving Phone Number |

The “Refunds” table stores information about daily refunds, shown in Table 2.4. The table contains three foreign keys “orderId”, “buyerId”, “sellerId”, the purpose of using foreign keys is to associate the information of buyers, sellers and orders together, enables users to perfect the use of the refund function and store relevant information in the database.

Table 2.4 - Structure of “Refunds” table

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Field size  (bytes) | Description |
| refundId | varchar | 32 | Refund ID |
| orderId | varchar | 32 | Order ID |
| reasons | varchar | 255 | Refund Reason |
| buyerId | varchar | 32 | Buyer ID |
| sellerId | varchar | 32 | Seller ID |
| status | varchar | 20 | Refund Status |
| refundAmount | int | 4 | Refund Amount |

The “LoginLog” table stores information about login logs, shown in Table 2.5. The table contains a foreign key “userId”, the purpose of using this foreign key is to record user-related log information in the system, so that the administrator can carry out relevant statistics in the background.

Table 2.5 - Structure of “LoginLog” table

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Field size  (bytes) | Description |
| logId | varchar | 32 | Log ID |
| userId | varchar | 32 | User ID |
| loginTime | datetime |  | Login Time |
| logoutTime | datetime |  | Logout Time |

The “User” table stores information about user, shown in Table 2.6. The information in the user table contains user name, user password, user password is encrypted, so even the administrator cannot see the user's password in the database, making the security of the system greatly improved, and the table also contains the user's avatar path, the user can upload an avatar to display their avatar in the front page.

Table 2.6 - Structure of “User” table

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Field size  (bytes) | Description |
| userId | varchar | 32 | User ID |
| userName | varchar | 10 | User Name |
| password | varchar | 10 | User Password |
| avatarUrl | varchar | 20 | Avatar URL |
| description | varchar | 50 | User Description |
| gender | tinyint | 1 | User Gender |
| birthday | date |  | User Birthday |
| balance | int | 4 | User Balance |

The “banner” table stores information about banner, shown in Table 2.7. Include information such as the title and URL of the banner.

Table 2.7 - Structure of “banner” table

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Field size  (bytes) | Description |
| bannerId | varchar | 32 | Banner ID |
| bannerTitle | varchar | 20 | Banner Title |
| bannerUrl | varchar | 52 | Banner URL |

The “Manager” table stores information about system manager, shown in Table 2.8. Contains information such as name and password.

Table 2.8 - Structure of “Manager” table

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Field size  (bytes) | Description |
| managerId | varchar | 32 | Manager ID |
| managerName | varchar | 10 | Manager Name |
| password | varchar | 10 | Manager Password |

The “Message” table stores information about message, shown in Table 2.9. It contains a foreign key about the user Id, which records the recipient and sender of the message.

Table 2.9 – Structure of “Message” table

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Field size  (bytes) | Description |
| messageId | varchar | 32 | Message ID |
| sentTime | datetime |  | Message Sent Time |
| fromUserId | varchar | 32 | Message Sender |
| toUserId | varchar | 32 | Message Receiver |
| details | varchar | 255 | Message Details |
| status | varchar | 32 | Message Status |

One of the main roles of foreign keys is to maintain data integrity. With foreign key constraints, you can prevent the insertion of invalid or inconsistent data into an association relationship. Foreign keys require the existence of corresponding primary or unique keys in the referenced table, which ensures that the associated data remains consistent when inserted, updated, or deleted.

Another role is to facilitate associative queries. With foreign keys, you can easily join the keys of related tables and get the associated data. Foreign keys allow association relationships to be established between multiple tables, making data retrieval and querying easier and more efficient.

**2.4 System Hardware and Software Requirements**

### 2.4.1 Hardware Requirement:

* processor: standard processor with a speed of 1.6 GHz;
* RAM:2 GB RAM or more;
* hard disk: 10 GB or more;
* monitor: standard color monitor;
* keyboard: standard keyboard;
* mouse: standard mouse.

**2.4.2** Software Requirement:

* Windows 7 or higher;
* IntelliJ IDEA, JDK;
* MySQL Workbench 8.0 CE;
* Tomcat 8.0.

**2.5 Ergonomics**

**2.5.1** Selection of colors

The system interface uses white as the background color and most of the controls are in green, making the system interface look more simple, generous and in line with most people's aesthetics.

**2.5.2** Interface style

The interface style of the system adopts a simple and efficient design concept, using larger fonts and controls to ensure that the interface is comfortable to use while

enhancing the details of the interface, and the layout of the interface uses a reasonable design, so that users can better use the system functions.

**2.5.3** Security of information storage

The information in the system is stored in the MySQL database, and key information such as passwords are stored in an encryption method that cannot be cracked in reverse, so users do not have to worry about information leakage when using the system. In addition, the payment and delivery process are very safe, with real-time monitoring in the background of the system, and if you have questions about the transaction process, you can contact customer service and the administrator to deal with.

**2.5.4** Convenience of system operation

The system operation is very simple and efficient, users only need to click the corresponding function area when using the functions, there is no unnecessary operation, and each function is simple and clear, users can quickly get started. The system's registration and login functions are also very easy to use, so new users can quickly register and use the system.

**3 IMPLEMENTATION OF INFORMATION SYSTEM**

Implementation is the stage in the project where the theoretical design is turned into a working system. The implementation phase constructs, installs and operates the new system. The most crucial stage in achieving a new successful system is that it will work efficiently and effectively.

**3.1 UML Diagrams**

UML is an acronym that stands for Unified Modeling Language. Simply put, UML is a modern approach to modeling and documenting software. In fact, it’s one of the most popular business process modeling techniques.

Mainly, UML has been used as a general-purpose modeling language in the field of software engineering. However, it has now found its way into the documentation of several business processes or workflows. For example, Activity Diagrams, a type of UML diagram, can be used as a replacement for flowcharts. They provide both a more standardized way of modeling workflows as well as a wider range of features to improve readability and efficacy.

**3.1.1** Use Case Diagram

The use case diagram of the second-hand trading company information system is shown in Figure3.1. It shows some participants and some use cases, and the relationships between them. They are mainly used for the functionality of a system, subsystem or class. Behaviors are modeled and the use case diagram shows the relationships between use cases and between participants of the same use case. The main actors (users) of this system include users and managers (administrators).

Users have such functions: Search for products, include three methods: matching keywords、search record saving、type matching. Buy products, include four steps: consignee information、cancel the order、confirm receipt、apply for a refund. Modify posted product, include three steps: modify price、modify description、modify picture. Post used products: include four steps: set description、upload picture、set price、set product type. Edit personal information, include four steps: upload avatar、edit nickname、change password、what's up. Communication with seller. Communication with buyer. Administrators can also edit users' personal information, and in addition, administrators can discern disputes raised by users, maintain a library of product types, manage products posted by users, etc.

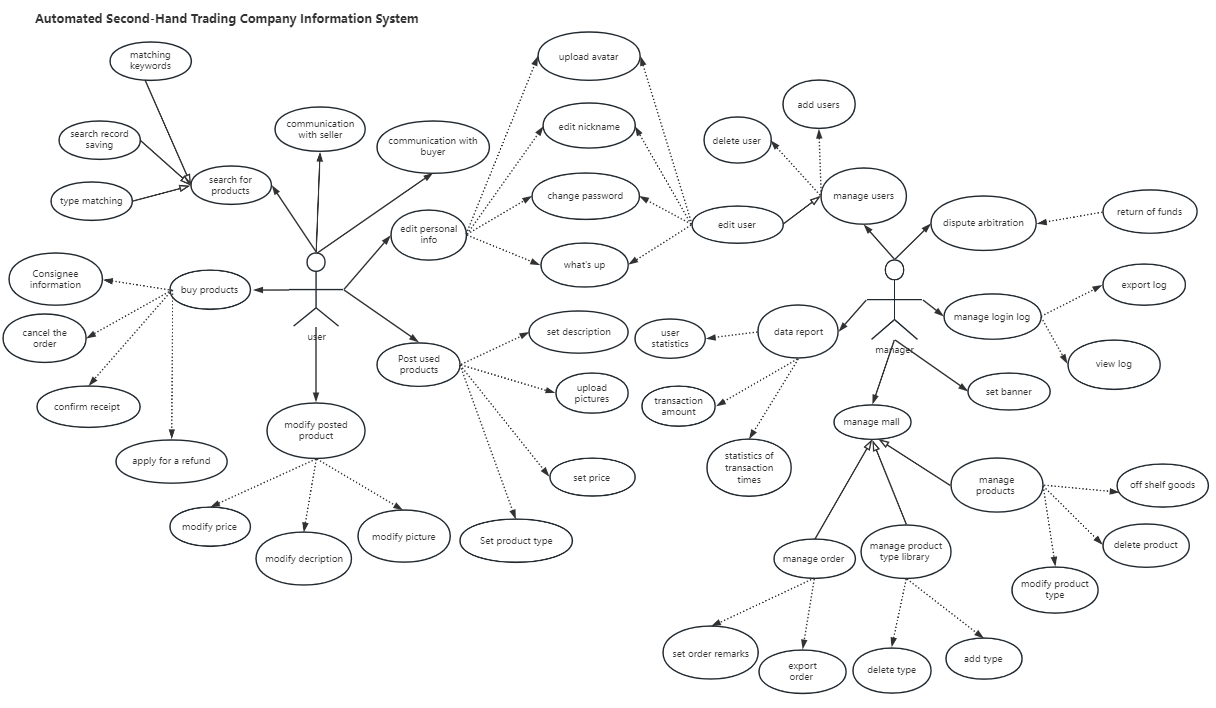


Figure 3.1 – Use Case Diagram

**3.1.2** Sequence Diagram

UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when. [5]

Sequence diagrams are commonly used during system analysis, design, and development phases to visualize and communicate the flow of interactions between various components or actors in a system. They help in understanding the order of message exchanges, identifying dependencies, and validating the correctness of the system's behavior.

In the following sequence diagram Figure 3.2, most of the operations in the system and the interactions between objects are briefly described.

First is the user's session information delivery, the user from the login, will be authorized by Main Server, when the user searches for products, the main server will extract product information from the database and displayed on the shopping platform, the user can select the used goods they need according to the information in the Shopping-Platform, then the user purchase and generate order information, the shopping platform will store the order data in the database, if the user has objections to the order, you can apply for arbitration request will be sent to the after-sales platform, which will forward the request to the administrator for processing.

After that is about the administrator's session between the transmission, the administrator from the login, will first go through the system's authorization, if the system recognizes the identity of the administrator, the administrator can enter the system, and then start the Main Server stage, in the Main Server, the administrator will receive in the After-sales platform returned about the order dispute Then the administrator can supervise the products in Shopping Platform and store the related information in database. And the administrator can set the type of products in Shopping-Platform and store the data in the database.

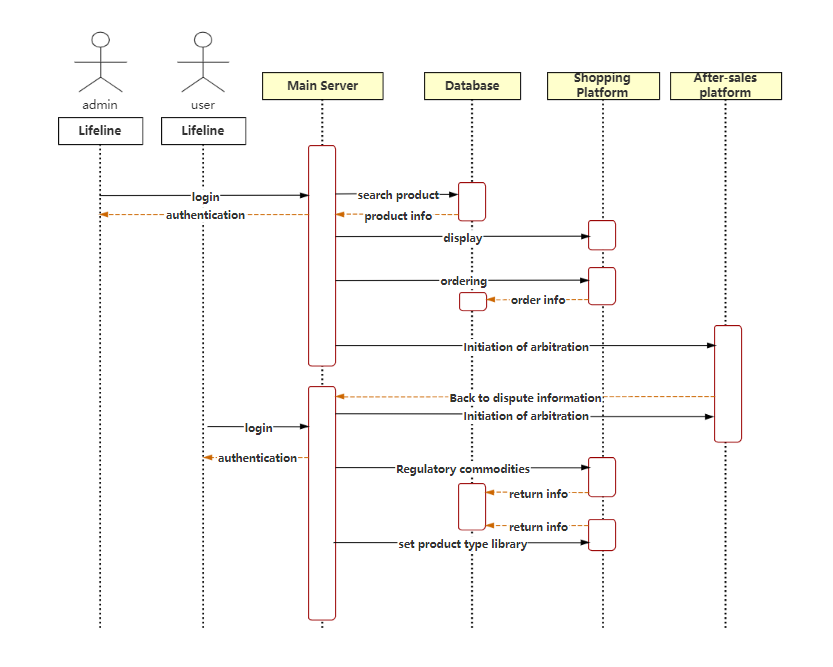


Figure 3.2 – Sequence Diagram

**3.1.3** Activity Diagram

Activity diagram is another important behavioral diagram in [UML](https://en.wikipedia.org/wiki/Unified_Modeling_Language) diagram to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity.[6]

Activity diagrams consist of the following key elements:

* activity: an activity represents a specific action or operation within the system. It can represent a single task or a group of related tasks. Activities are depicted as rounded rectangles;
* control flow: arrows or edges indicate the flow of control between activities, representing the sequence in which activities are performed. It shows the logical order of activities and any conditions or decisions that guide the flow;
* decision nodes: decision nodes, represented by diamond-shaped symbols, indicate points in the diagram where a decision or condition is evaluated. Depending on the outcome of the decision, the control flow follows different paths;
* fork and join nodes: fork nodes represent the splitting of control flow into multiple parallel paths, allowing activities to be executed concurrently. Join nodes, on the other hand, bring parallel paths back together;
* initial and final nodes: an initial node represents the start point of the activity diagram, while a final node represents the end or completion of the activity.

Activity diagrams are commonly used in system analysis, business process modeling, and software development to visualize and understand the flow of activities and actions within a system. They help in identifying dependencies, clarifying the sequence of tasks, and analyzing the overall behavior of the system or process.

The activity diagram describes the rules followed by the sequence relationship of object activities. It focuses on the behavior of the system rather than the processing process of the system. Activity diagrams can represent the situation of concurrent activities, and activity diagrams are object-oriented.

When a user uses a used goods trading system, he first logs in and then sees the shopping page homepage, on which he can select different types of goods and also choose to post his own goods, and in the process, he can communicate with the seller or buyer to discuss the specific information about the goods.

At the same time, the management of various user information, the setting of mall announcements, the handling of order disputes, and the management of products all need to be operated by the system administrator. The business flow chart of the second-hand trading company information system is shown in the Figure 3.3.

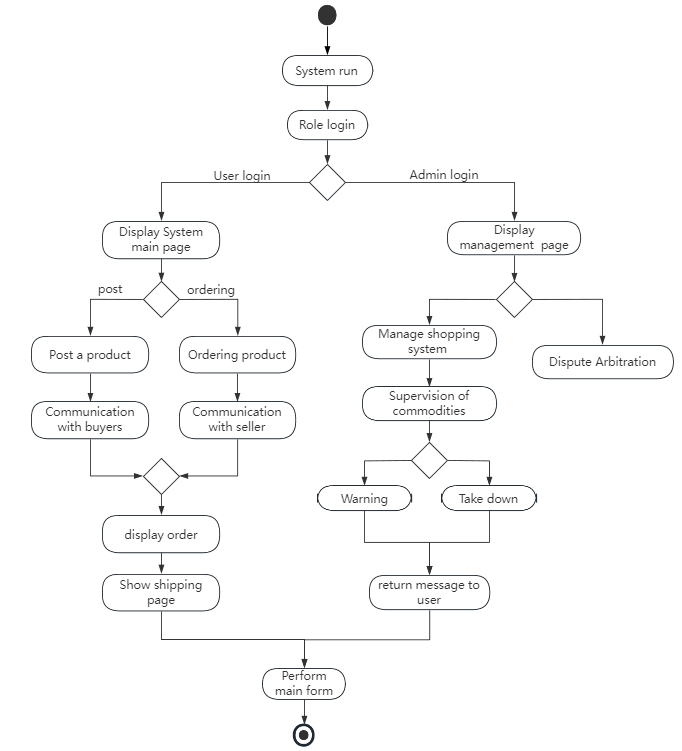


Figure 3.3 - Activity Diagram

**3.2 Software Implementation Programming Tools**

After analyzing the subject area and the structure of the system, the following tools were considered in the implementation of the program: Vue, Spring Boot, MYSQL, JS, JQUERY, CSS, HTML, Hibernate.

### 3.2.1 Vue

Vue is an open-source JavaScript framework for building user interfaces. It helps developers build interactive and flexible reusable web applications by being componentized and responsive. Vue simplifies the development process by providing automatic UI updates and a clean syntax that makes it easier to build dynamic and responsive user interfaces. It also provides features such as routing and state management, making it easier to build single-page applications and manage application data. In short, Vue is a flexible and easy-to-use framework that helps developers build efficient web applications.

**3.2.2** Spring Boot

Spring Boot is an open-source framework for rapidly building Java applications. It is based on the Spring Framework, simplifying the configuration and deployment process, allowing developers to focus more on the implementation of business logic, without spending too much time and effort on tedious configuration. Spring Boot provides automatic configuration and default conventions, you can quickly integrate common functional modules, such as database access, security authentication, web development and so on. It also provides an embedded Servlet container that makes it easy to package applications into executable JAR files and run them directly, reducing application deployment and operations and maintenance costs. Spring Boot simplifies the Java application development process and provides a fast, easy, and efficient way to build reliable enterprise-class applications. [7]

**3.2.3** Cascading Style Sheets (CSS)

CSS (Cascading Style Sheets) is a markup language used to describe the style and layout of web pages. It is used in conjunction with HTML to control the appearance and typography of web page elements through selectors and properties.CSS makes it easy for developers to change the style of elements such as text, colors, backgrounds, borders, spacing, etc. to personalize designs and visual effects. By separating style from content, CSS makes the maintenance and modification of web pages more flexible and scalable. It is one of the key technologies for building modern web pages, providing powerful style control for web design and development.

**3.2.4** jQuery

jQuery is a fast, clean and powerful JavaScript library. It simplifies the operation of common tasks such as HTML document traversal, event handling, animation effects, and AJAX interaction. By using jQuery, developers can achieve complex operations with less code and improve development efficiency. jQuery provides an easy-to-use API that makes manipulating DOM elements, handling events, performing animations, and easier and more cross-browser compatible. It is widely used in front-end development, providing developers with a convenient way to manipulate and manipulate web elements and their behavior. In short, jQuery is a popular and powerful JavaScript library that makes web development easier and more efficient.

### 3.2.5 MySQL

MySQL is an open-source relational database management system (RDBMS), which is a software tool for storing and managing large amounts of structured data. MySQL uses Structured Query Language (SQL) to manage and manipulate data. It supports multi-user concurrent access and is characterized by high performance, reliability and scalability. MySQL is widely used in Web applications and various enterprise-level applications for storing and retrieving data. It provides rich features such as data table creation, modification and querying, transaction processing, index management, security control, etc. The open-source nature of MySQL makes it easy to acquire, use and customize, making it one of the most popular relational database management systems. [8]

**3.3 Software Structure**

The structure of the automated second-hand trading company information system shown in Figure 3.4 is composed of several separate front- and back-end modules, with two API modules and two front-end modules, and the API modules are all dependent on the same core module.

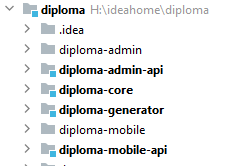


Figure 3.4 - The project structure

* diploma-admin, this module is the front-end page of the admin system written in Vue;
* diploma-admin-api, this module is the back-end API of the administrator system, which mainly handles the requests from the front-end, and the front-end page completes the front and back-end interactions through the back-end API;
* diploma-mobile, this module is the front-end page of the user system, through which users can complete the transaction of used items, etc;
* diploma-mobile-api, this module is using the back-end API of the user system, and the front-end page can realize the data interaction between front and back-end through this module. This module is using the back-end API of the user system, and the front-end page can realize the data interaction between front and back-end through this module;
* diploma-core, this module is the core of the whole project, diploma-admin-api and diploma-mobile-api both depend on this module;
* diploma-generator, this module is the project's code generator and is used to generate the project's structure as well as the base code.

**3.4 Function Description**

Below there are the description of some important functions of the automated second-hand trading company information system.

*sendSmsCode(@RequestParam String mobile) -* users can choose to use SMS verification code when logging in, this function is used to send SMS verification code.

*add(@RequestBody CartVo cartVo) -* add products to cart.

*changeDefault(@PathVariable("id") Long id,@PathVariable("isDefault") Boolean isDefault) -* users have the option to change the default address when selecting a shipping address.

*add(@PathVariable("idGoods") Long idGoods) -* mark the product as favorite.

*queryGoods(@RequestParam("idCategory") Long idCategory) -* get the list of products under the specified category.

*search(@RequestParam("key") String key) -* search for products by keywords.

*searchHot() -* search for popular products.

*getOrders(@RequestParam(value = "status", required = false) Integer status) -* get all orders for the current user.

*prepareCheckout(@RequestParam(value = "chosenAddressId", required = false) Long chosenAddressId, @RequestParam(value = "idCarts") String idCarts) -* the user is ready to settle the item.

*wxPrepare(@RequestParam("orderSn")String orderSn) -* users using Wechat pay.

*uploadAvatar(@RequestPart("file") MultipartFile multipartFile) -* users upload or change their avatar.

*save(@ModelAttribute Category category) -* set product category.

*changeIsOnSale(@RequestParam("id")Long id,@RequestParam("isOnSale") Boolean isOnSale) -* set the item to whether it can be sold or not.

The following is the most important code in the system.

*@RequestMapping(value = "sendSmsCode", method = RequestMethod.POST)*

*public Object sendSmsCode(@RequestParam String mobile) {*

*if("15011112222".equals(mobile)){//Determine whether the cell phone number meets the rules*

*return Rets.success(shopUserService.sendSmsCodeForTest(mobile));*

*}*

*if ("prod".equals(applicationProperties.getEnv())) {*

*if(StringUtil.isMobile(mobile)) {//If the rules are met*

*Boolean ret = shopUserService.sendSmsCode(mobile);//Get the result of whether to send SMS verification code*

*return Rets.success(ret);*

*}else{*

*return Rets.failure("Illegal cell phone numbers");*

*}*

*} else {*

*String ret = shopUserService.sendSmsCodeForTest(mobile);*

*return Rets.success(ret);*

*}*

*}*

The above is the function code for user login, the logic of the code is to receive the phone number passed by the front-end, firstly, match the number with the rules, check if it is in the correct format, if it is correct then send the corresponding verification code, if it is wrong then return the prompt. The user can use this function to log in to the system with the verification code when he forgets the password of his account.

*@RequestMapping(value = "prepareCheckout", method = RequestMethod.GET)*

*public Object prepareCheckout(@RequestParam(value = "chosenAddressId", required = false) Long chosenAddressId,*

*@RequestParam(value = "idCarts") String idCarts) {*

*Long idUser = getIdUser(HttpUtil.getRequest());*

*List<SearchFilter> filters = Lists.newArrayList(//Merge data into a List*

*SearchFilter.build("idUser", SearchFilter.Operator.EQ, idUser),*

*SearchFilter.build("id", SearchFilter.Operator.IN, StringUtil.splitForLong(idCarts, ","))*

*);*

The above is the first part of the code for the function where the user is ready to pay for the order in the shopping cart. The logic of this code is to receive the user's email address id and shopping cart id from the front-end, need to do validation on the passed data, and then get the corresponding object based on the id.

*List<Cart> list = cartService.queryAll(filters);*

*Address address = null;*

*if (chosenAddressId == null || chosenAddressId == 0) {*

*address = addressService.getDefaultAddr(idUser);*

*} else {*

*address = addressService.get(chosenAddressId);*

*if(address==null){//Determine if the address is empty*

*address = addressService.getDefaultAddr(idUser);//* *Get the address of the corresponding user.*

*}*

*}*

*return Rets.success(Maps.newHashMap(*

*"list", list, "addr", address*

*));*

*}*

The above is the second part of the code for the function where the user is ready to pay for the order in the shopping cart. The first part of the code has already obtained the corresponding object in the database by the id passed from the front-end. The next thing to do is to merge and store the user information with the address information, etc. The shopping cart checkout process used by the user in the system is based on this function.

*public FileInfo upload(MultipartFile multipartFile){*

*String uuid = UUID.randomUUID().toString();*

*String originalFileName = multipartFile.getOriginalFilename();*

*String realFileName = uuid +"."+ originalFileName.split("\\.")[originalFileName.split("\\.").length-1];//Add uuid as an identifier in the file name*

The above part is the preparation work for uploading files, read the file name of the file to be uploaded and add uuid as the identifier.

*try {*

*File file = new File(cfgService.getCfgValue(CfgKey.SYSTEM\_FILE\_UPLOAD\_PATH) + File.separator+realFileName);//Create a new file to store the files to be uploaded*

*if (!file.getParentFile().exists()) {//Determine if the file already exists*

*file.getParentFile().mkdirs();*

*}*

*multipartFile.transferTo(file);*

*return save(multipartFile.getOriginalFilename(),file);*

*} catch (Exception e) {*

*e.printStackTrace();*

*logger.error("save error",e);*

*}*

*return null;*

*}*

The above section is the second part of uploading a file, using exception trapping to ensure that the upload process does not cause the system to crash.

*@Configuration*

*public class CORSConfiguration {*

*@Bean*

*public WebMvcConfigurer corsConfigurer() {*

*return new WebMvcConfigurerAdapter() {*

*@Override*

*public void addCorsMappings(CorsRegistry registry) {*

*registry.addMapping("/\*\*")*

*.allowedHeaders("\*")*

*.allowedMethods("\*")*

*.allowedOrigins("\*");*

*}*

*};*

*}*

*}*

The above code is the system configuration to solve the cross-domain, to prevent the front-end and back-end separation system in the front-end and back-end data interaction when the cross-domain problem.

*@RequestMapping(value = "/export",method = RequestMethod.GET)*

*public Object export(@RequestParam(value = "mobile", required = false) String mobile,*

*@RequestParam(value = "orderSn", required = false) String orderSn) {*

*List<SearchFilter> filters = Lists.newArrayList();*

*if(StringUtil.isNotEmpty(mobile)){// Determine if mobile is empty*

*filters.add(SearchFilter.build("user.mobile", mobile));*

*}*

*if(StringUtil.isNotEmpty(orderSn)){//Determine if orderSn is empty*

*filters.add(SearchFilter.build("orderSn",orderSn));*

*}*

The above is the first part of the system to export the order report function code, the function needs to receive the cell phone number and orderSn two parameters, and determine whether these two parameters have a value, if not empty then put into the filter.

*List<Order> orderList = orderService.queryAll(filters);*

*Map data = Maps.newHashMap("list",orderList);*

*String now = DateUtil.formatDate(new Date(),DateUtil.DATE\_TIME\_FMT);//* *Create timestamps by format*

*data.put("exportTime",now);*

*data.put("userName", JwtUtil.getUsername());*

*FileInfo fileInfo = fileService.createExcel("templates/orderList.xlsx","orderlist.xlsx",data);//* *Export the report*

*return Rets.success(fileInfo);*

*}*

The above is the second part of the code for the order report export function in the system. After verifying the integrity of the data, the information about the order is taken out of the filter and a new excel sheet is created and the data is entered into excel.

**3.5 User’s Guide**

**3.5.1** User Login

As shown in Figure 3.5, in this page users can have two ways to log in, the first is to log in directly by entering the account password, the account number is the cell phone number at the time of registration, and the second is to log in by sending a verification code to the cell phone, the next demonstration of logging in via verification code in Figure 3.6.

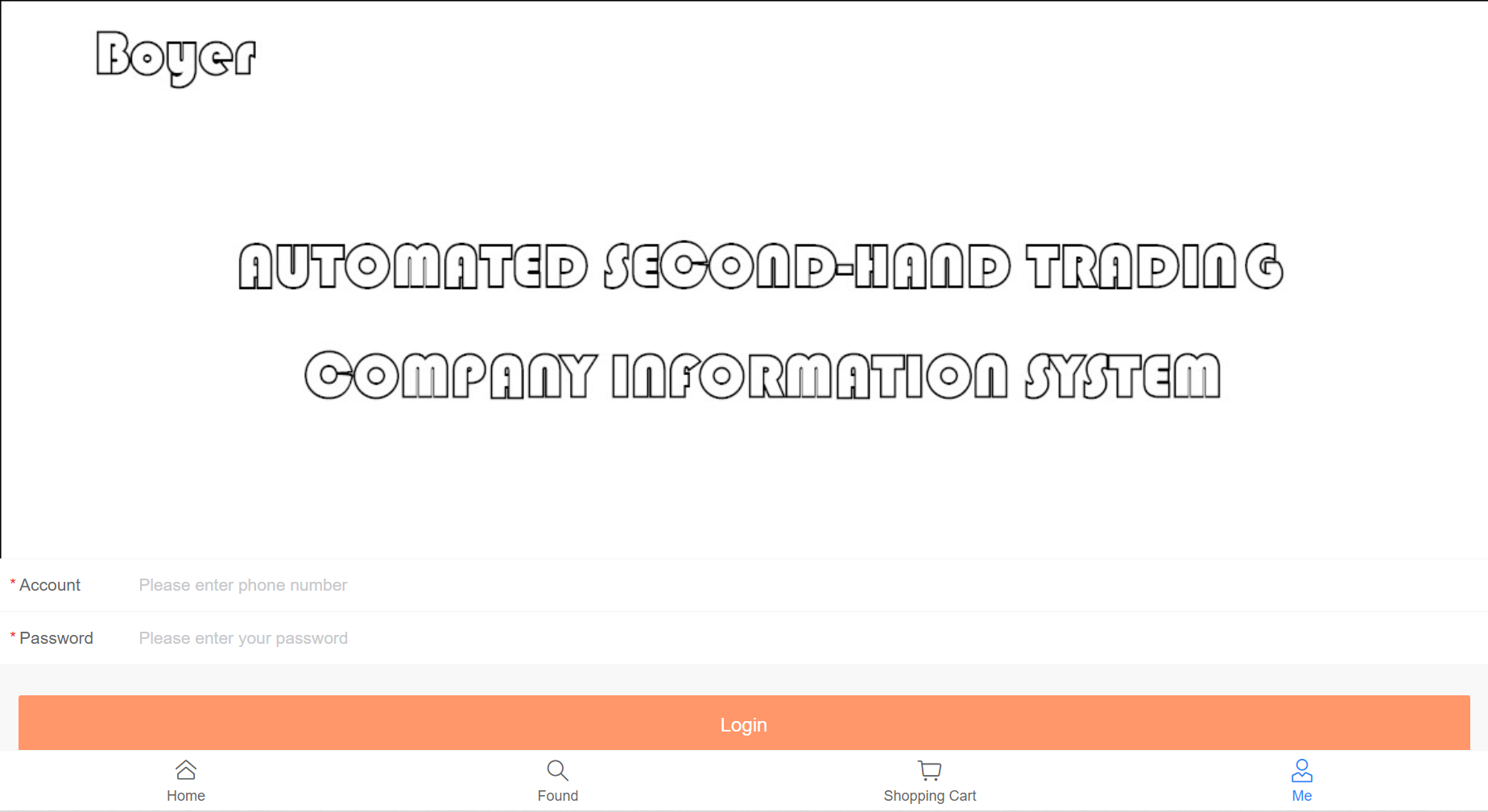


Figure 3.5 - The user login page

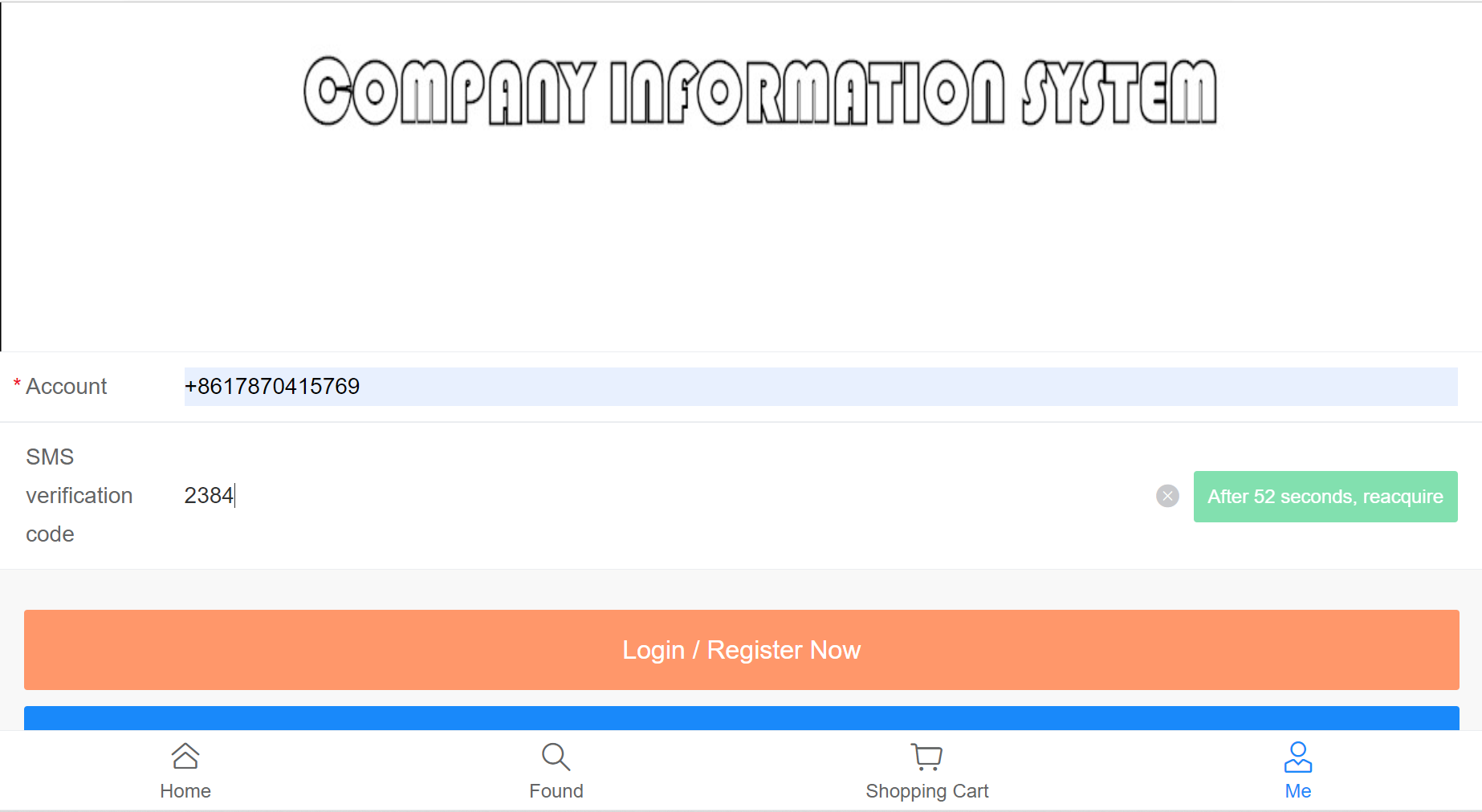


Figure 3.6 – Login by sms code

The above-mentioned users can log in after entering the SMS verification code received from the filled-in cell phone number. If the cell phone number has not been registered at this time, it will be automatically registered and given an initial password, which greatly enhances the ease of login for unregistered users when using the system.

**3.5.2** Main Page

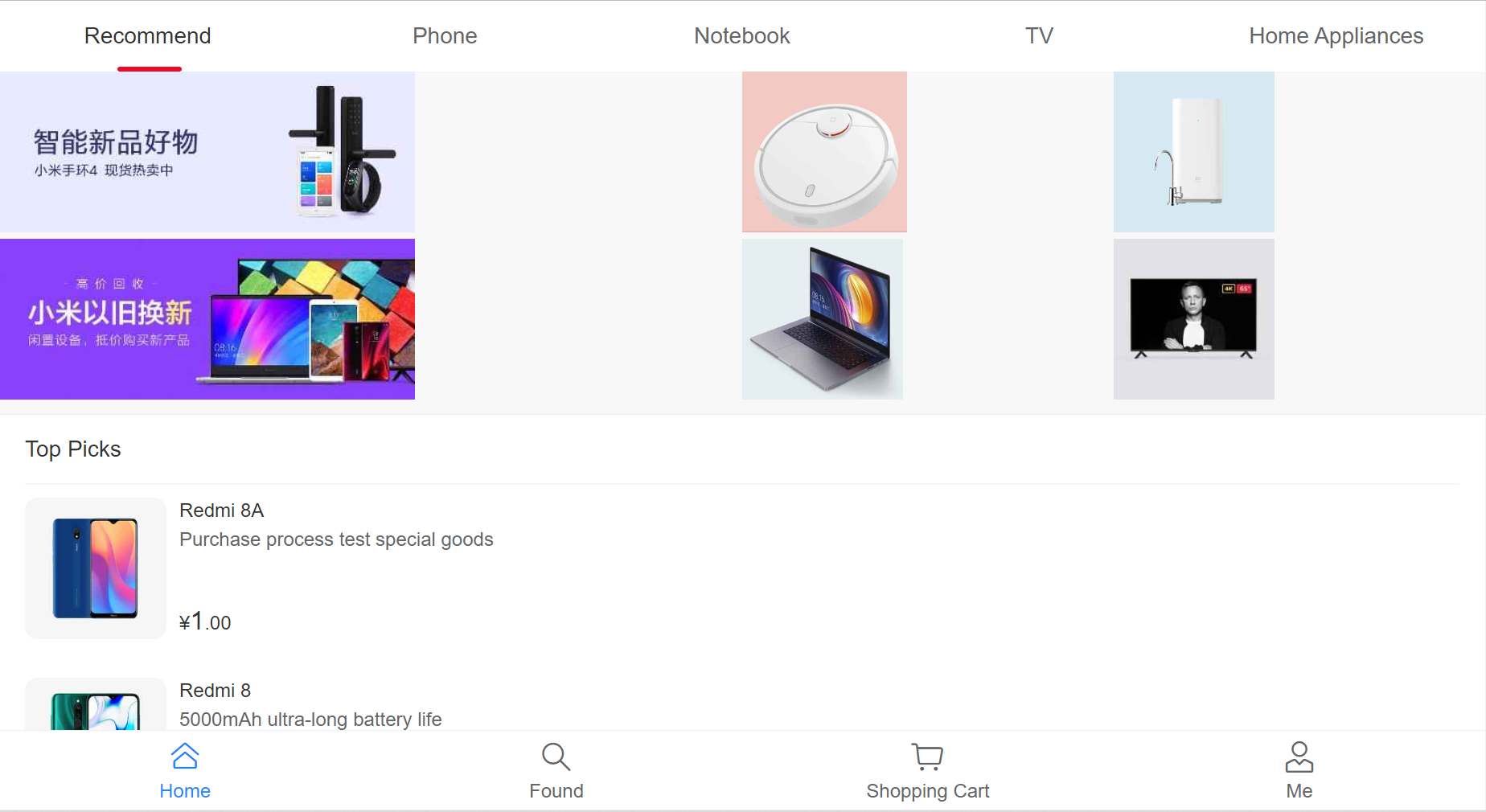


Figure 3.7 – Main page

As shown in Figure 3.7, on the home page, users can browse the products recommended by the system, which depends on the hotness of the product, or they can select the product type at the top of the page to enter a specific type to browse a specific product, and only used products of that type will be displayed on the page of the specific type. For any product on the page, the user can click to enter the product's detail page to view pictures and descriptions, etc. This is shown in Figure 3.8.

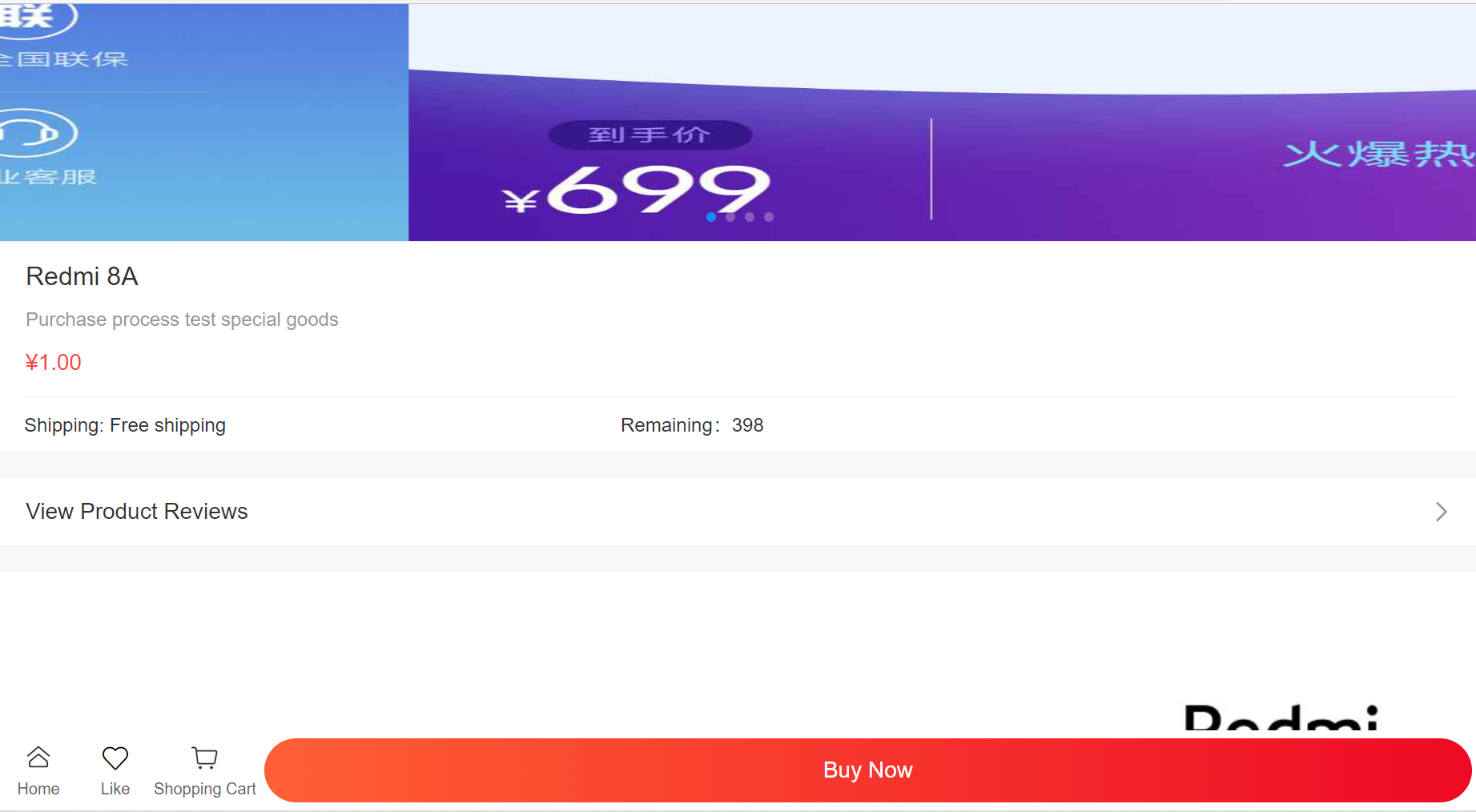


Figure 3.8 – Product Detail Page.

**3.5.3** Search product

In addition, users can search for products by keywords by clicking on the Found button underneath the main one, such as Figure 3.9, where the user enters redmi, and the system will list all products about Redmi.

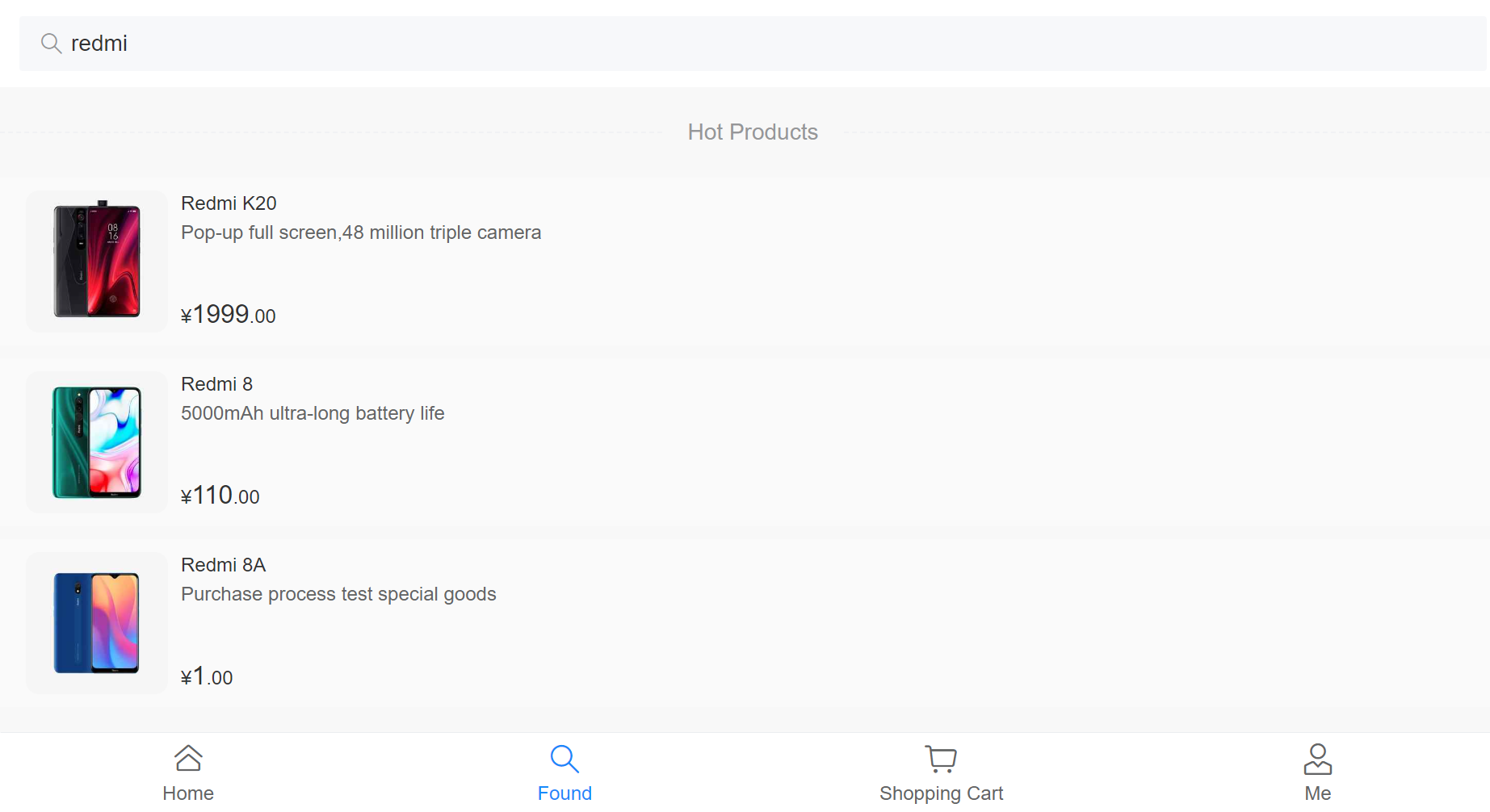


Figure 3.9 – Search products by keywords.

**3.5.4** Purchase of goods

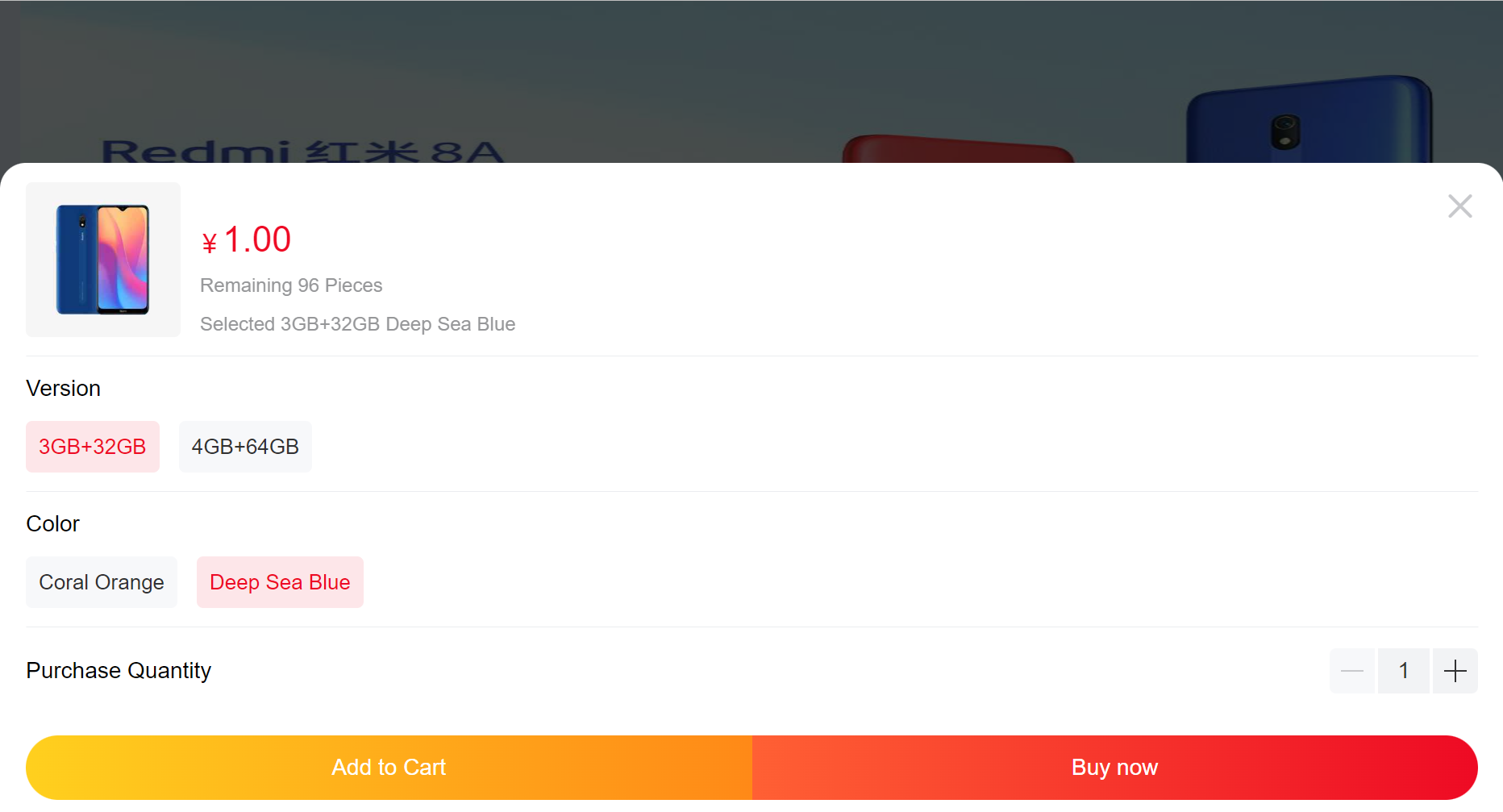


Figure 3.10 – Shopping page.

Users can click the Buy now button on the product details page to enter the purchase page, where they can select the product specifications such as Figure 3.10, and can choose to add the product to the shopping cart or buy it directly, both of which will add the product to the shopping cart as Figure 3.11.

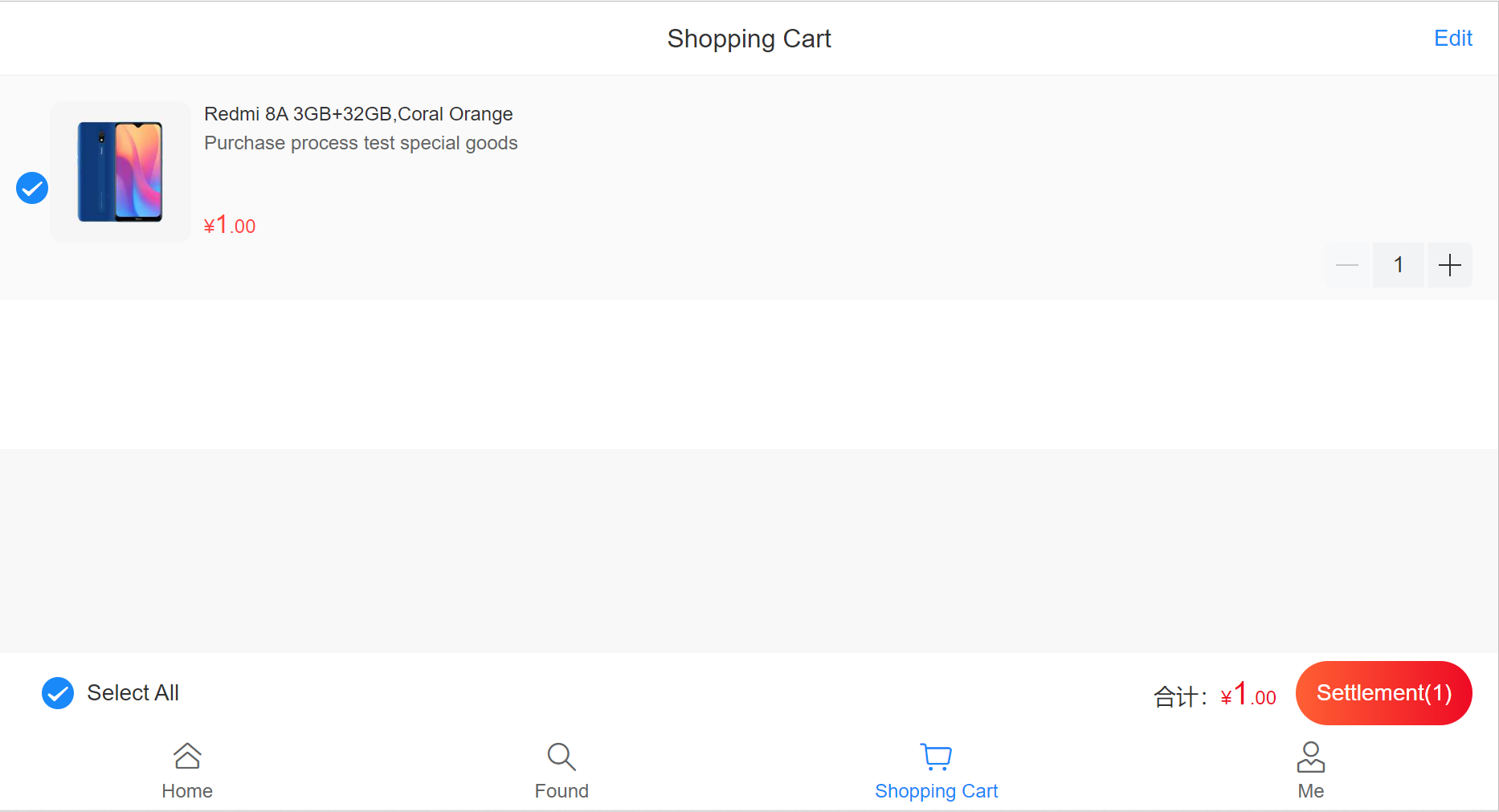


Figure 3.11 – Add product to cart or buy now.

**3.5.5** Order cancellation

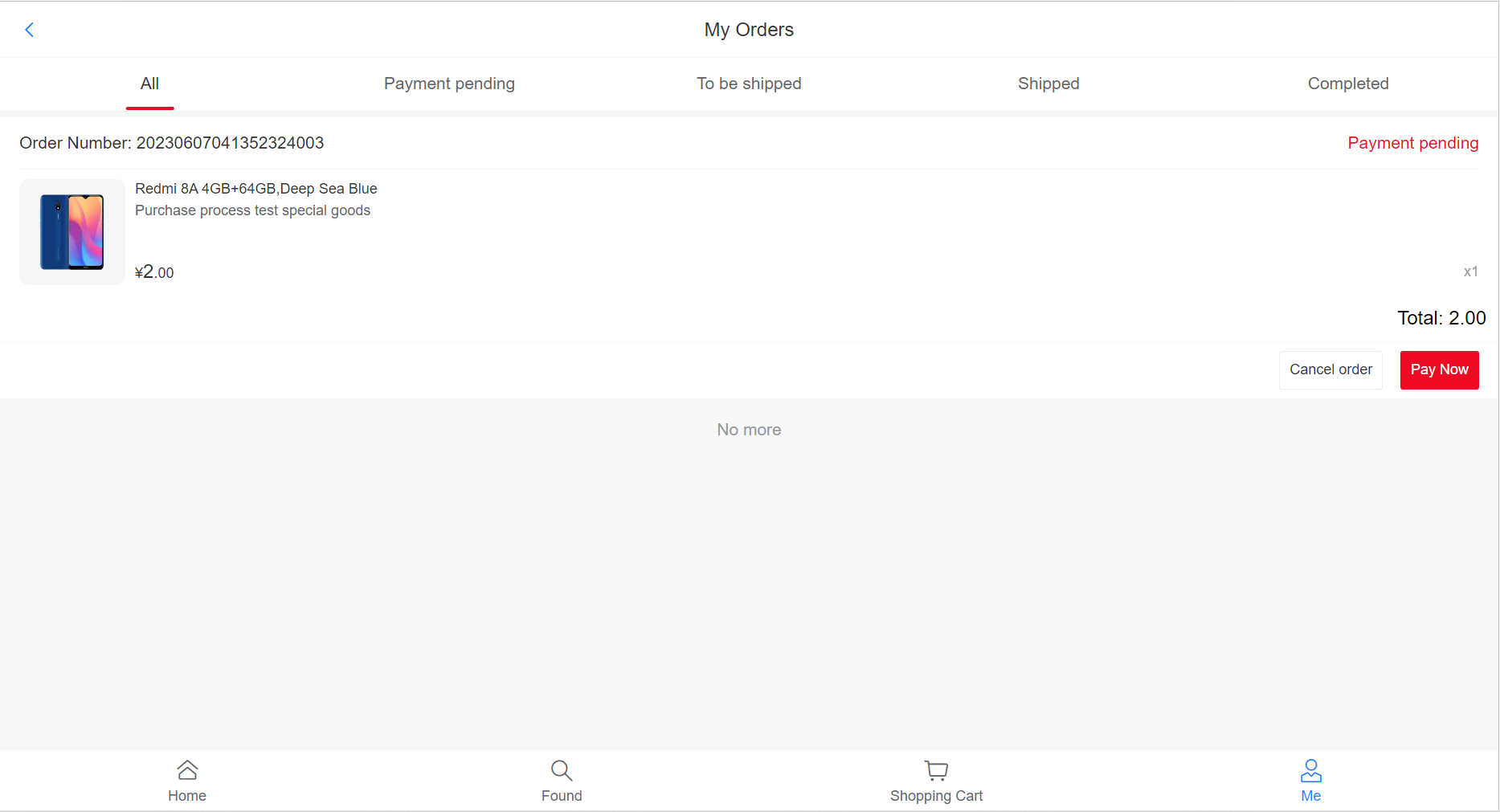


Figure 3.12 – Cancel the order

In Figure 3.12, for the user's own orders, the user has the option to cancel the order if they change their mind, and the order will be cancelled and removed from the order list when the user clicks the Cancel order button.

**3.5.6** User change password

In Figure 3.13, in this page, users can change their passwords. The old password is used for permission verification and the new password will replace the old password in the database and the next login will use the new password.

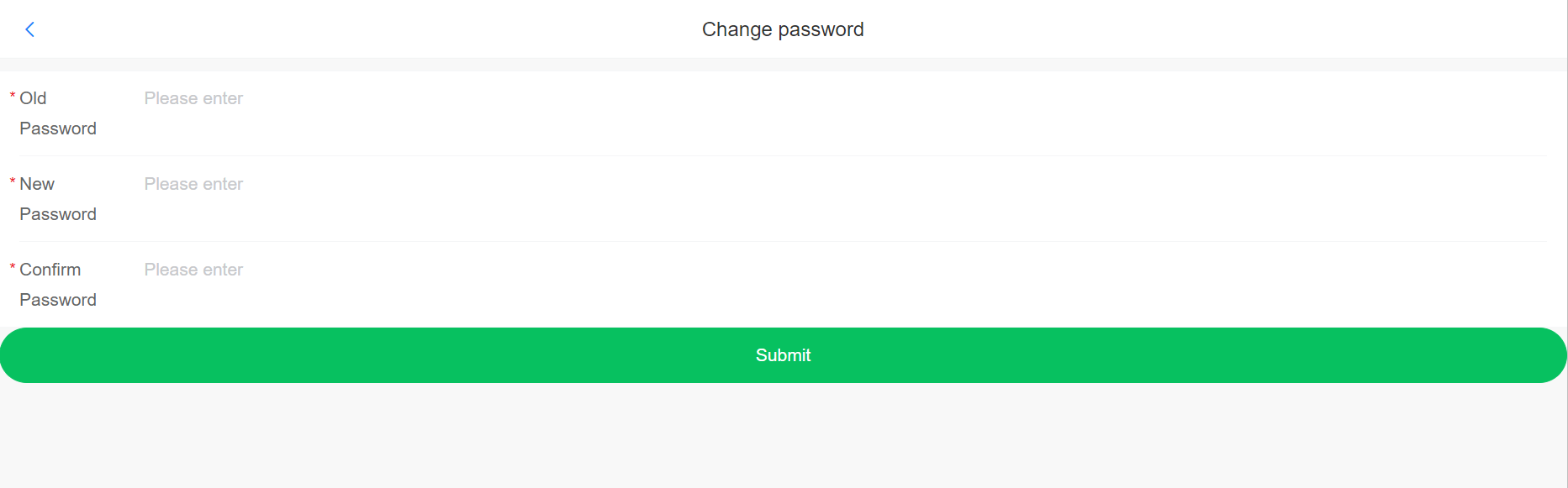


Figure 3.13 – Change password

**3.6 Administrator’s Guide**

**3.6.1** Administrator login



Figure 3.14 – Administrator login page.

On this page, administrators can log in to the administrator system by entering their account password and clicking Login.

**3.6.2** Manage Users

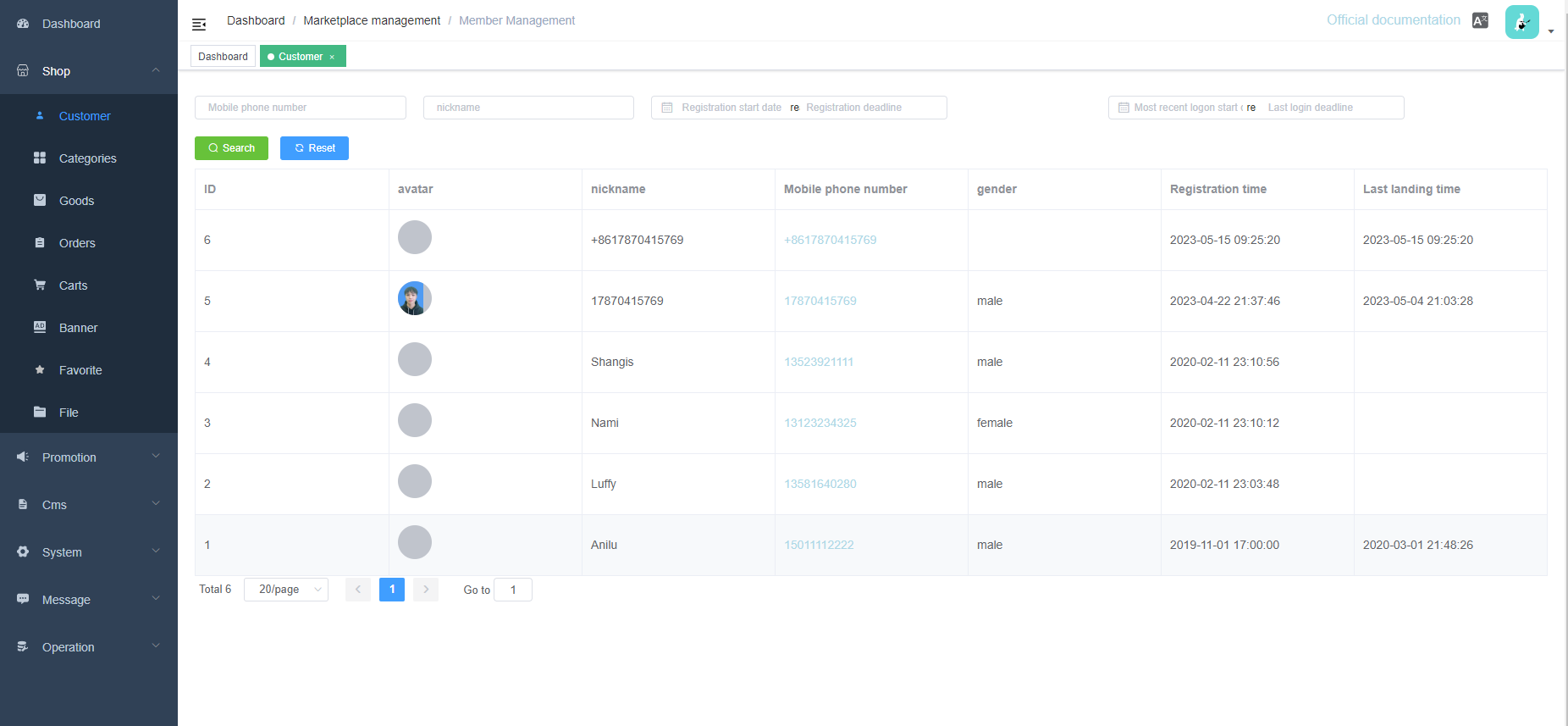


Figure 3.15– Administrator manage users

In Figure 3.15, the administrator can manage the users registered in the system and view their specific information, as shown in Figure 3.16, and can filter by keyword search, or user registration time period, which makes it easy for the administrator to observe the popularity of the system.

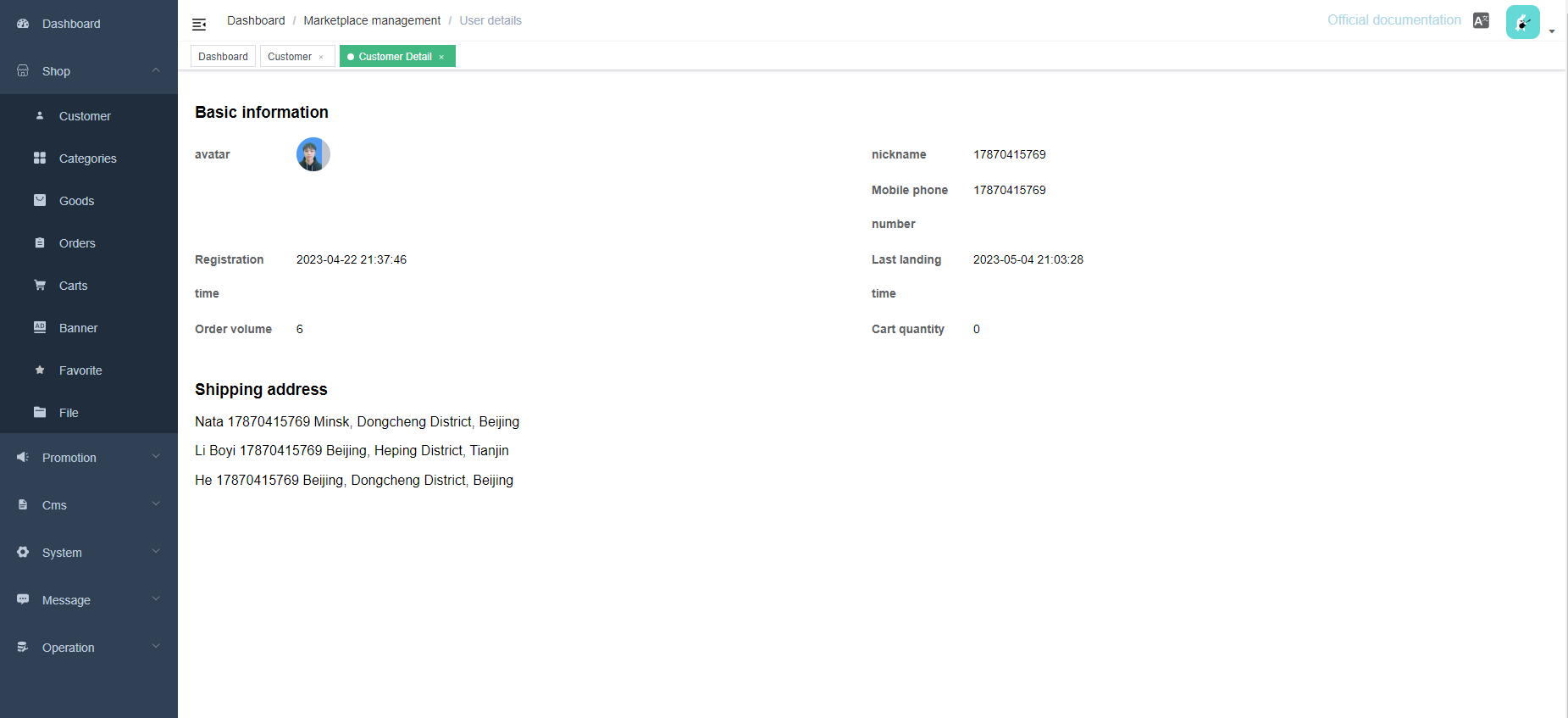


Figure 3.16 – User’s details

**3.6.3** Manage topics

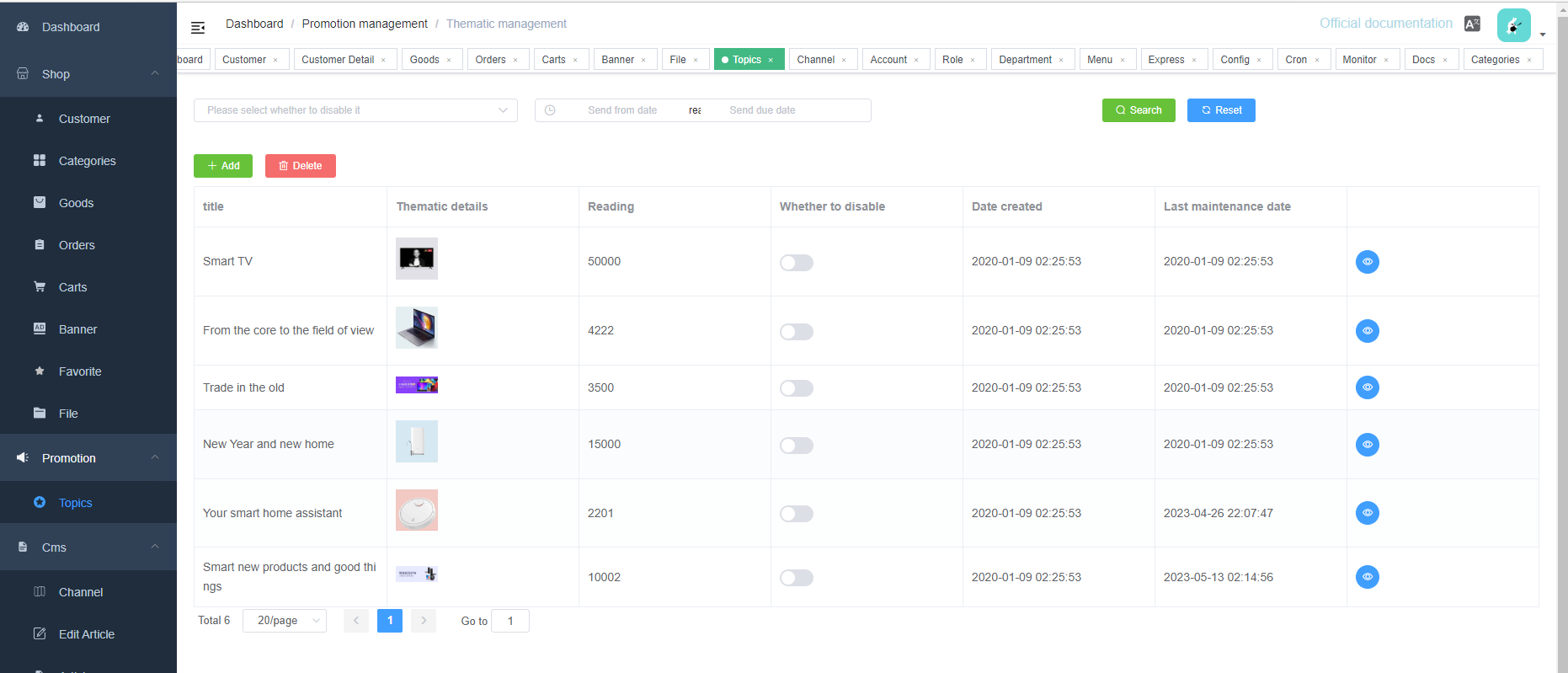


Figure 3.17 – Administrator manage topics

In Figure 3.17, the administrator can set a topic about a certain second-hand commodity, and can maintain the topic, and set the topic to be visible or invisible. If it is set to invisible, users will not see this topic in the system.

**3.6.4** Process for uploading products

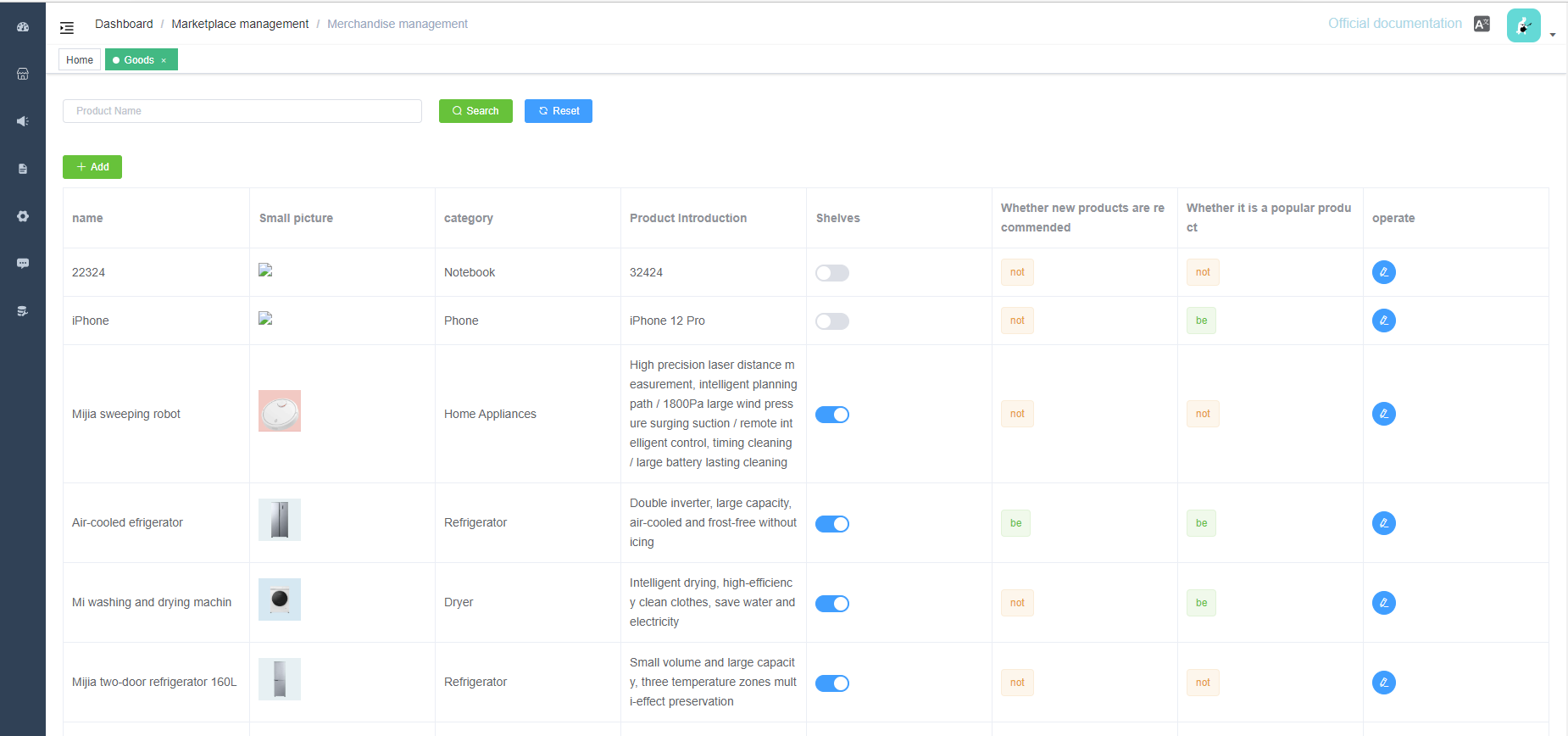


Figure 3.18 – Upload products

In Figure 3.18, the administrator clicks the Add button to add a second-hand item to the system. The operation of adding a second-hand item can be seen in Figure 3.19. The administrator needs to write the name and description of the second-hand item, then select the corresponding category, and upload it Actual pictures, etc.

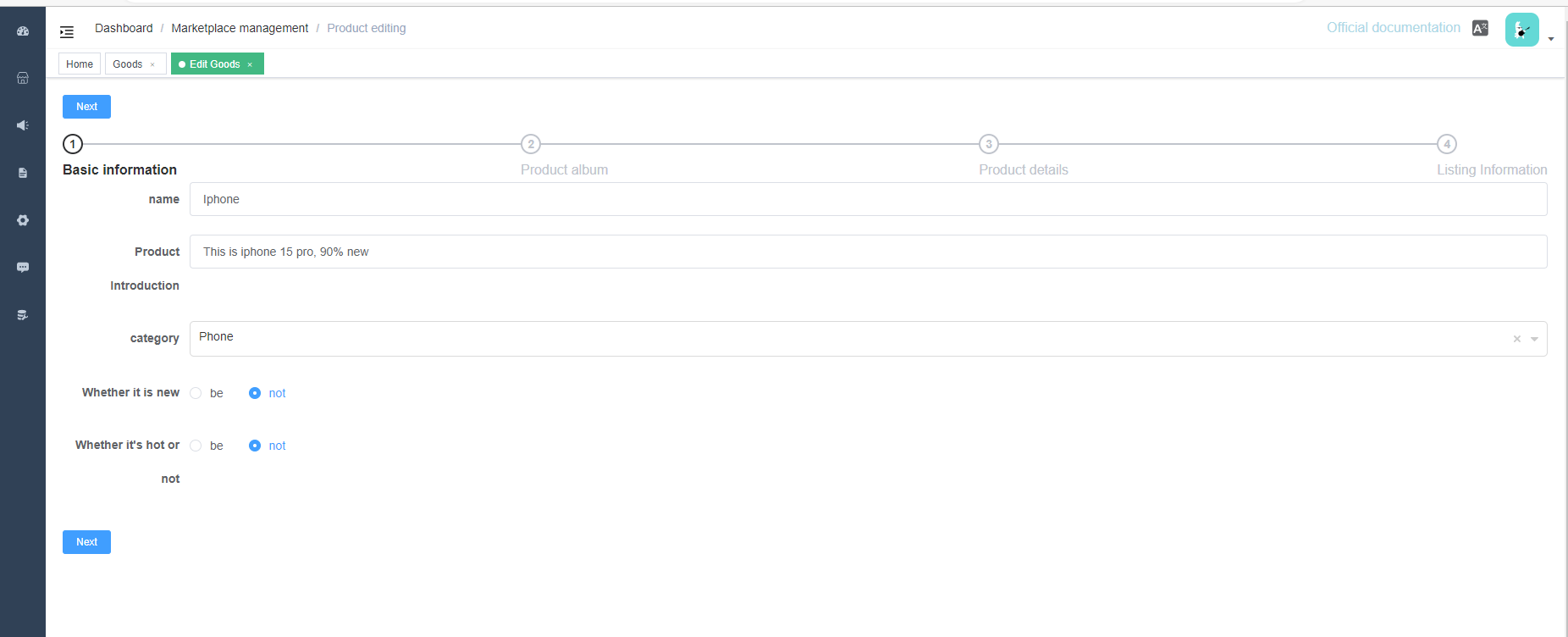


Figure 3.19 – Set up second-hand items Details

**3.6.5** Set up used product categories

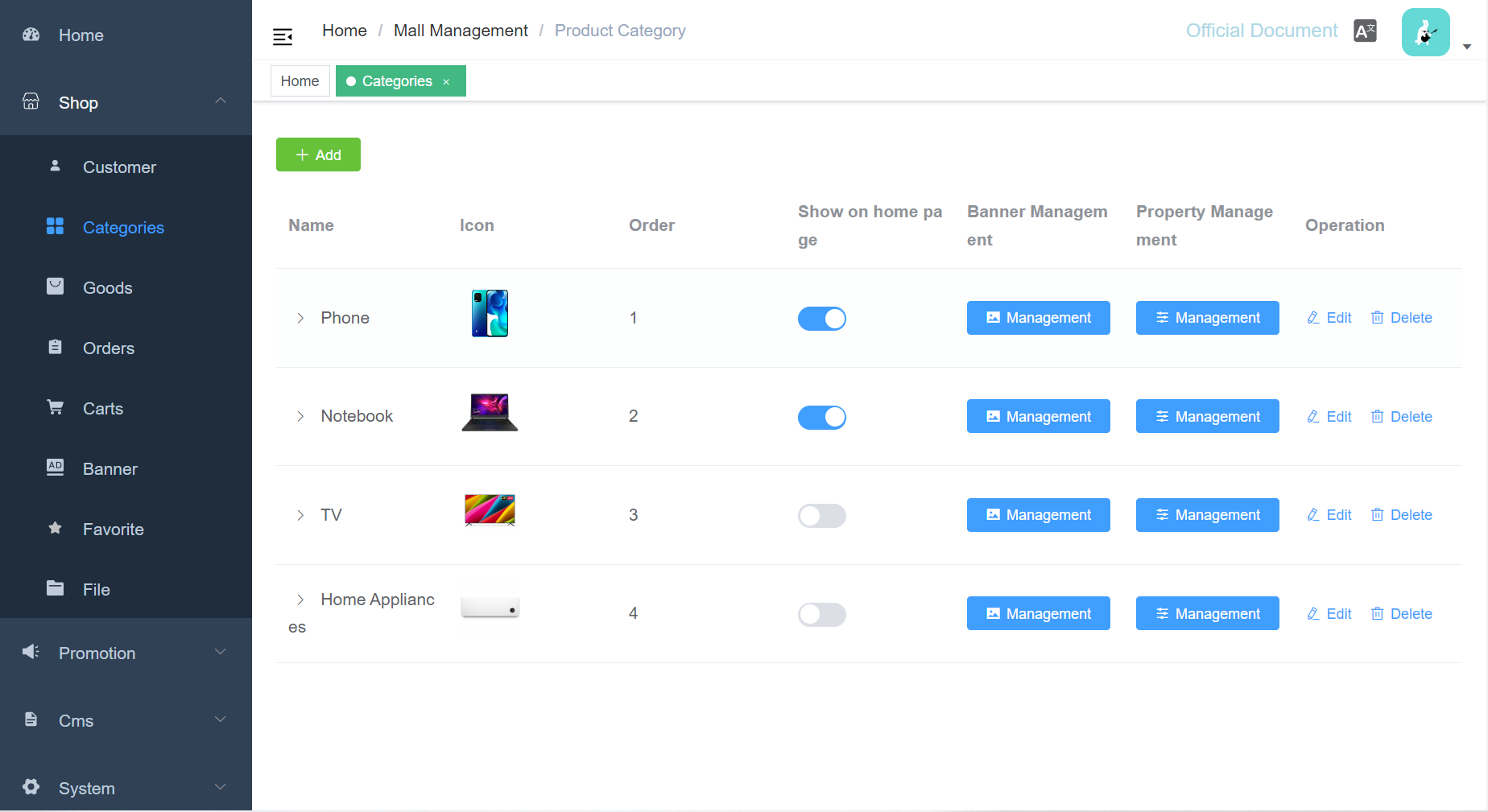


Figure 3.20 – Set up used product categories

In Figure 3.20, the administrator can add the appropriate category for the used items and click the Add button to jump to the Figure 3.21 page, where the administrator can upload the appropriate image for the category to represent the category in general and set the details about the category. This allows users to select the corresponding category tags for their used items when uploading them, making it easier for buyers to find their items.

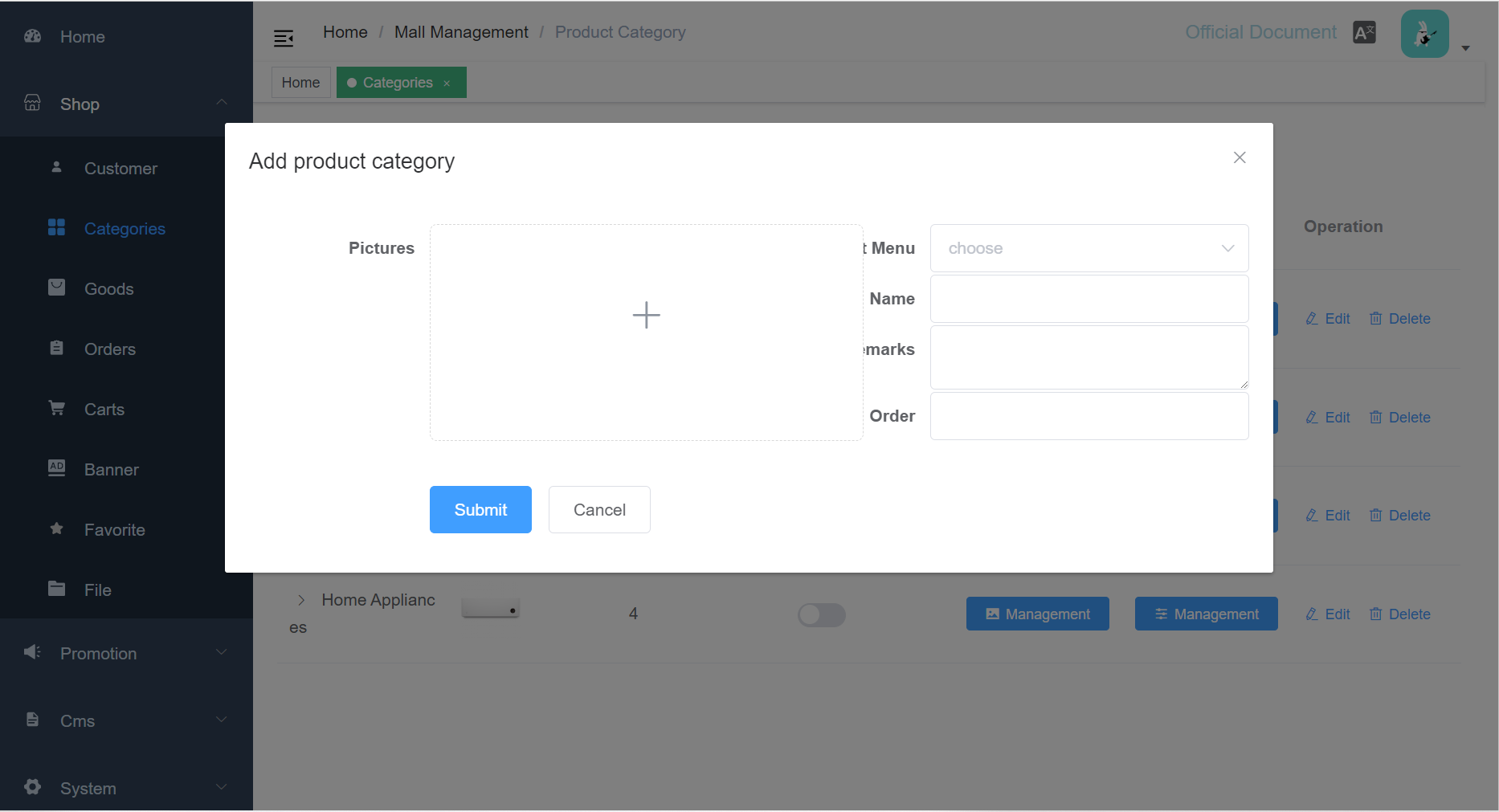


Figure 3.21 – Set category details

**3.6.6** Order managements

In Figure 3.22, the administrator can manage the recent order information. As shown in the figure, the administrator can choose to display different progress orders, such as those waiting for payment, those already paid and waiting to be shipped, those already shipped, etc., and can also manage different orders according to their time. Also, in the button of Operation, you can export the order information to excel.

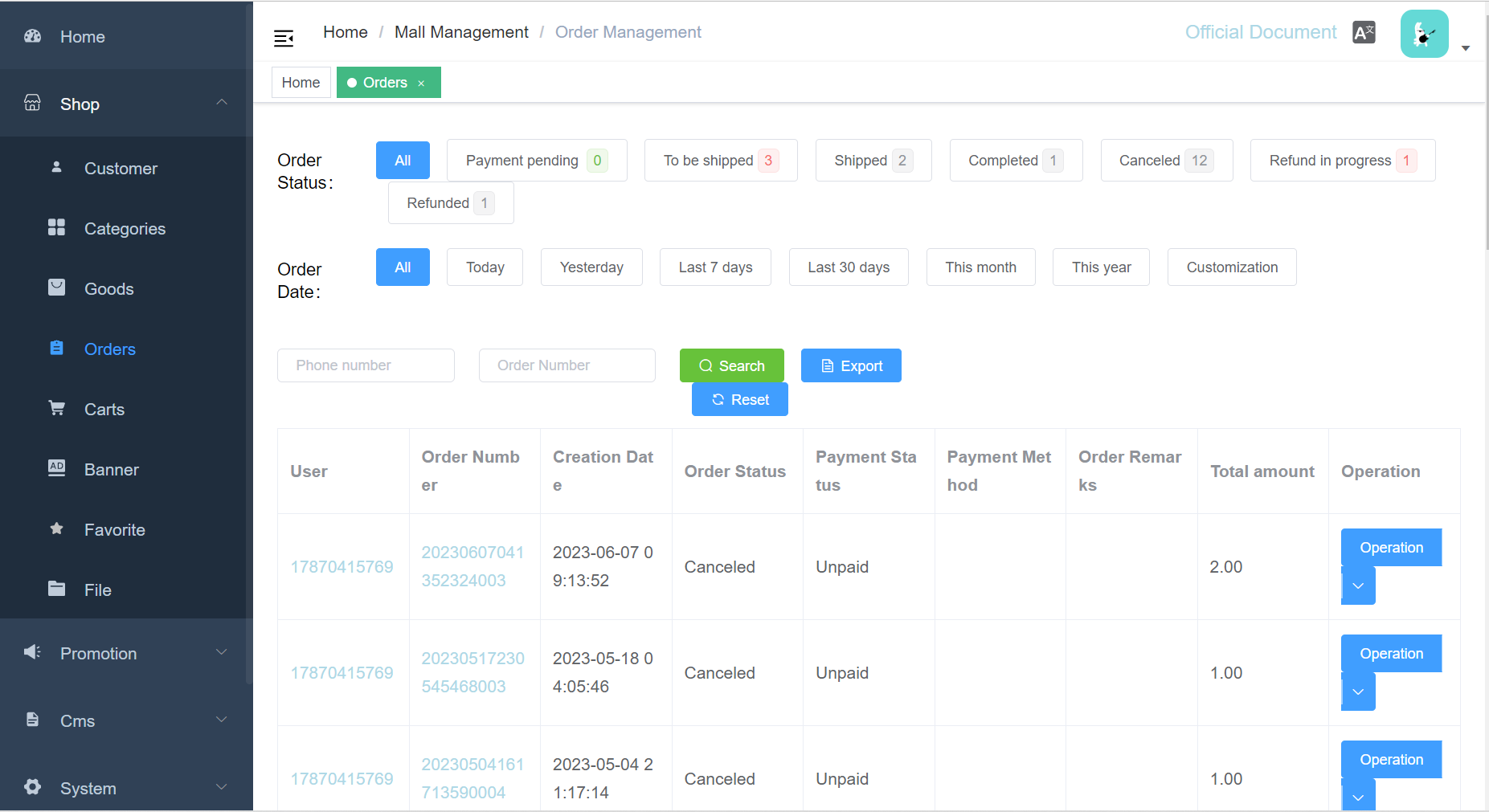


Figure 3.22 – Order managements

In Figure 3.23 is the screen after exporting the order.

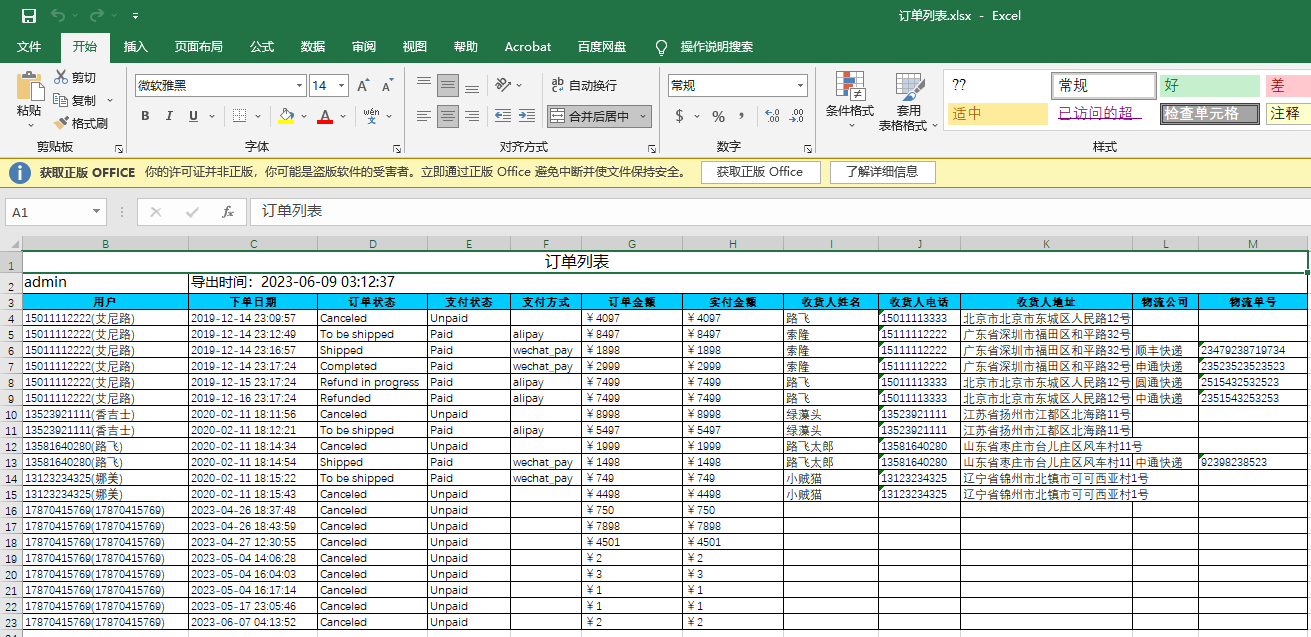


Figure 3.23 – Export orders

**3.6.7** Administrator Home

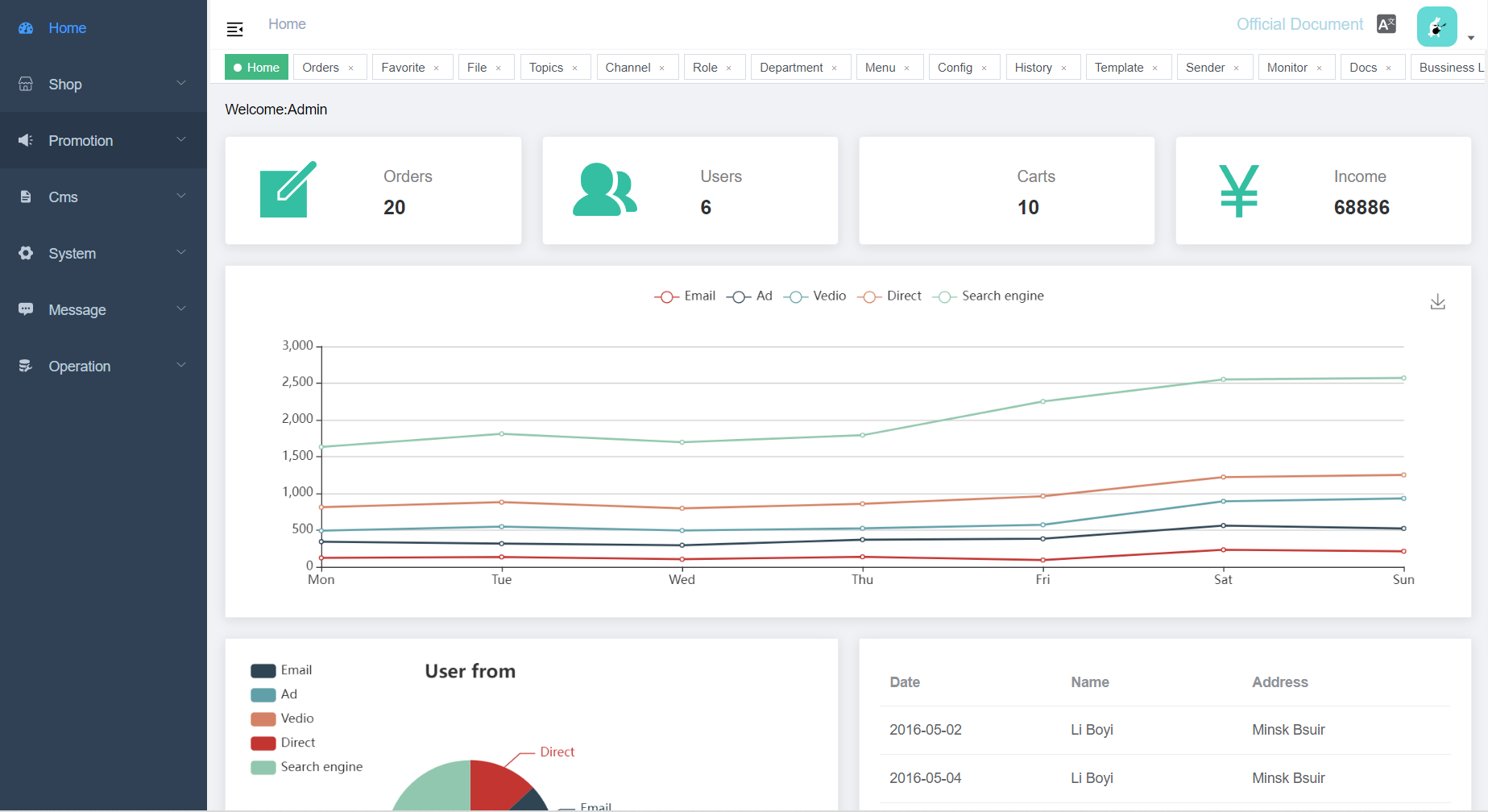


Figure 3.24 – Administrator home

In the above Figure 3.24 is the main page of the administrator, the left side is the main functional area of the administrator, the rest of the page mainly shows some statistics, such as cumulative users, or total number of orders, etc.

**3.7 Further Development**

**3.7.1** Increase the chat function between users

The current system only supports users to directly purchase second-hand items, and cannot support users to communicate with each other and discuss the details of the products. In the later development, adding this function will allow users to have more communication, and you can also find products that are more suitable for you when buying second-hand items.

**3.7.2** Add administrator arbitration function

The current system is not perfect. If users are dissatisfied after purchasing second-hand items, they cannot reasonably protect their rights and interests. Therefore, in subsequent developments, after adding the administrator arbitration function, users can apply for disputes over their purchase orders. The administrator intervenes, mediates the dispute, and finally gives the arbitration result.

**3.7.3** More payment methods

The current system only supports WeChat payment. This payment method is single and has many limitations. Considering that many users do not use WeChat, more payment methods such as PayPal and Apple Pay need to be added in the next development.

**3.7.4** Building a new payment system

The system's current payment system is not perfect and relies heavily on external payment systems, which makes the system less secure. If a payment system belonging to the system is constructed, it can make users transfer their funds directly to the platform during transactions, so that the platform can supervise the funds, and if problems such as order disputes arise, the funds can be frozen directly and await subsequent processing.

# **4 ECONOMIC FEASIBILITY STUDY**

**4.1 Characteristic of an Economic Case of the Project**

Automated Second-Hand Trading Company Information System is an online platform that allows users to trade their unwanted items without having to leave home, and offers a wide variety of options.

**4.2 Calculation of Cost of Materials for Project Accomplishment**

The estimate of costs for carrying out of scientifically research work settles payments under following clauses. Calculation is performed under the formula:

where – the coefficient considering hauling expenses

( from 1.0 to 1.10) for the project we accept KTP = 1;



– norm of the expenses a material kind on the project;

– unit of selling price of material kind, Ruble;

– Quantity of applied kinds of materials.

The estimate of costs for carrying out of scientifically research work settles payments under following clauses show in the Table 4.1.

Table 4.1 - Calculation of costs for materials

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | The name of materials | Unit of measure | The price, Ruble. | Quantity | The sum, Ruble. |
| 1 | Paper format А1 | Sheet | 6 | 9 | 54 |
| 2 | Paper format А4 | Sheet | 0.08 | 150 | 12 |
| 3 | Stationery | - | - | - | 60 |
| 4 | Materials for experiences and designing | - | - | - | 50 |
| The sum of expenses | | - | - | - | 176 |

**4.3 Calculation of a Base Salary of the Personnel Occupied with Accomplishment of Works**

The size of costs settles payments under the formula:

where – a wage rate for a day, categories of workers, Ruble;

– quantity of workers of a category;

– time of actual work of the worker of a category under the project, day;

- coefficient of awards on bonus systems;

( from 1.10 to 1.40) for the project we accept = 1.2.



Calculation of the produce in the Table 4.2.

Table 4.2- Base salary calculation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | The name of categories of workers and posts | Quantity of units, the people | Salary for one month, Ruble. | Coefficient of bonus surcharges | Expend-itures of labour, months | The sum, Ruble. |
| 1 | The supervisor of studies of the project | 1 | 1200 | 1.2 | 1 | 1440 |
| 2 | The engineer | 2 | 1800 | 1.2 | 1 | 2160 |
| The sum of expenses | | - | - | - | - | 3600 |

**4.4 Calculation of an Additional Salary of the Contractors, Including the Various Payments**

Provided by the labour law, under the formula:

Additional wages include a variety of performers stipulated by the labour legislation of the payment and is calculated according to the formula:

where Hnw – the specification of an additional salary;

from 10 to 25%, for the project it is accepted = 20%.



=360020/100= 720 Ruble.

**4.5 Calculation of Deductions to Social Insurance**

where – rate of deductions on social ensurance (tax), = 34.6%.

= (3600+720)∙34.6/100= 1494.72 Ruble.

**4.6 Calculation of Expenses on Scientific Business Trip**

We calculate the other expenses for materials scientific and technical information and the fee for the use of internet and telephone, etc.

The cost is calculated according to the formula:

where – the specification on scientific business trip expenses,

from 5 to 20%, for the project we accept = 20%.

= 3600∙20/100 =720 Ruble.

**4.7 Calculation of Common Enterprise Expenses**

Indirect cost includes the cost of management and overhead cost, calculated according to the formula:

where - the specification of indirect expenses, ≈ from 50 to 100 %, for the project it is accepted = 90 %.

=3600∙90/100 =3240 Ruble.

**4.8 Calculation of the Complete Cost Value**

The total cost of scientific and technical products is determined as the sum of all cost in all respects (clauses 1-6) as according to the formula:

= 176+3600+720+1494.72+720+3240=9950.72 Ruble.

**4.9 On Level of Profitability in Percentage**

At the average level of profitability in percent of the total cost is determined by the target profit unit of scientific and technical products according to the formula:

where - profitability level, ≈ from 10 to 30 %, for the project we accept = 30 %.

= 9950.72∙30/100 = 2985.21 Ruble.

**4.10 Calculation of the Price of the Project**

To determine an approximate (estimated) wholesale price of scientific and technical products according to the formula,

=9950.72+2985.21 = 12935.93 Ruble.

**4.11 Calculation of the Tax to Value Added (VAT)**

The Value Added Tax is determined by the formula:

where - the tax rate on vat (the tax), = 20%.

= 12935.93∙20/100 = 2587.18 Ruble.

**4.12 Calculation of the Price of the Project Taking into Account the VAT**

To determine the selling price of scientific and technical products with according to the formula:

= 12935.93+2587.18= 15523.11 Ruble.

Calculation of costs for the project and the project price are resulted in Table 4.3.

Table 4.3 - The Estimate of costs for the project

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Clauses of costs | Calculation | The sum, Ruble. |
| 1 | Materials () | Table 1 | 176 |
| 2 | Base salary () | Table 2 | 3600 |
| 3 | The additional salary () | 360020/100 | 720 |
| 4 | Deductions in population social insurance fund () | (3600+720) ∙34.6/100 | 1494.72 |
| 5 | Scientific business tripexpenses () | 3600∙20/100 | 720 |
| 6 | Common enterprise expenses(*Ркос*) | 3600∙90/100 | 3240 |
| 7 | Total the cost value (*Cn*) | 176+3600+720+1494.72+720+3240 | 9950.72 |
| 8 | Profit () | 9950.72∙30/100 | 2985.21 |
| 9 | The project price () | 9950.72+2985.21 | 12935.93 |
| 10 | The value-added tax () | 12935.93∙20/100 | 2587.18 |
| 11 | The price from the VAT () | 12935.93+2587.18 | 15523.11 |

Conclusion: Automated Second-Hand Trading Company Information System has been implemented, using the technology of front and back-end separation, where the front-end uses Vue and the back-end uses Spring boot, the cost of developing the whole system is 15523.12 Ruble.

## CONCLUSION

This explanatory note mainly describes the design process of the second-hand goods trading system, including functional diagrams and UML diagrams, the system is used by users and administrators, users have both buyer and seller status, which largely enhances the convenience of second-hand goods trading, at the same time, the system is supervised and maintained by administrators, which can make the trading system more orderly. The system is monitored and maintained by the administrator, which makes the trading system more orderly.

In the design of the system, it mainly realizes the functions of user purchasing second-hand items, publishing second-hand items, express mailing, and after-sale management, etc. The current system has been able to take up the daily online trading of second-hand items.

# **REFERENCES**

[1] Where to buy and sell second hand online? [Electronic resource]. – Access mode : https://www.which.co.uk/news/article/where-to-buy-and-sell-second-hand-online-aeGoH2H7cUyQ.

[2] What is organizational structure? [Electronic resource]. – Access mode : https://economictimes.indiatimes.com/definition/organizational-structure.

[3] What is IDEF0? [Electronic resource]. – Access mode : https://vitechcorp.com/resources/CORE/onlinehelp/desktop/Views/IDEF0.htm.

[4] What is Algorithm? [Electronic resource]. – Access mode : https://www.geeksforgeeks.org/introduction-to-algorithms/.

[5] What is Sequence Diagram? What is the Function? [Electronic resource]. – Access mode : https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/.

[6] What is Activity Diagram? [Electronic resource]. – Access mode : https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/.

[7] K. M. Koyya and B. Muthukumar, A Survey of Saga Frameworks for Distributed Transactions in Event-driven Microservices, 2022 Third International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE), Bengaluru, India, 2022, pp. 1-6, doi: 10.1109/ICSTCEE56972.2022.10099533.

[8] Birnbaum, M. H. (2010). An overview of major techniques of Web-based research. In S. D. Gosling & J. A. Johnson (Eds.), Advanced methods for conducting online behavioral research (pp. 9–25). American Psychological Association. https://doi.org/10.1037/12076-002.

# **APPENDIX A**

**(informative)**

**Code Description**

**A.1** Entity layer

The definition of the entity layer, where all entity classes are defined.

Entity of Address:

*@Entity(name="t\_shop\_address")  
public class Address extends ShopBaseEntity {*

*private Long idUser;*

*private String name;*

*private String tel;*

*private String province;*

*private String city;*

*private String district;*

*private String areaCode;*

*private String addressDetail;*

*private String postCode;*

*private Boolean isDefault=false;*

*private Boolean isDelete= false;*

*public String getWholeAddressInfo(){*

*return getProvince()+" "+getCity()+getDistrict()+getAddressDetail();*

*}*

*}*

Entity of Cart:

*@Entity(name="t\_shop\_cart")  
public class Cart extends ShopBaseEntity {*

*private Long idUser;*

*private ShopUser user;*

*private Long idGoods;*

*private Goods goods;*

*private Long idSku;*

*private GoodsSku sku;*

*private Integer count;*

*private BigDecimal price;*

*public BigDecimal getPrice(){*

*if(idSku!=null){ return sku.getPrice(); }*

*return goods.getPrice(); }*

*public String getTitle(){*

*return idSku != null?getGoods().getName()+" "+getSku().getCodeName():getGoods().getName(); }}*

Entity of Goods:

*@Entity(name="t\_shop\_goods")*

*@EntityListeners(AuditingEntityListener.class)*

*public class Goods extends BaseEntity {*

*private String name;*

*private String pic;*

*private String gallery;*

*private Long idCategory;*

*private Category category;*

*private String descript;*

*private String detail;*

*private Integer stock=0;*

*private BigDecimal price;*

*private Boolean isDelete= false;*

*private Boolean isOnSale = false;*

*private Boolean isNew = false;*

*private Boolean isHot = false;*

*private Integer likeNum;*

*}*

Entity of Order:

*@Entity(name="t\_shop\_order")*

*public class Order extends ShopBaseEntity {*

*private Long idUser;*

*private ShopUser user;*

*private String orderSn;*

*private Integer status;*

*private Long idAddress;*

*private Address address;*

*private String consignee;*

*private String mobile;*

*private String consigneeAddress;*

*private List<OrderItem> items;*

*private String message;*

*private String adminMessage;*

*private BigDecimal shippingAmount;*

*private BigDecimal tradeAmount;*

*private BigDecimal totalPrice;*

*private BigDecimal couponPrice;*

*private BigDecimal realPrice;*

*private Date shippingTime;*

*private Long idExpress;*

*private Express express;*

*private String shippingSn;*

*private Date confirmReceivingTime;*

*private String payType;*

*private Integer payStatus;*

*private String payId;*

*private Date payTime;*

*public String getStatusName(){*

*if(status!=null) {*

*return OrderEnum.get(status).getValue();*

*}*

*return null;*

*}*

*public String getPayStatusName(){*

*if(payStatus!=null) {*

*return OrderEnum.getPayStatus(payStatus).getValue();*

*}*

*return null;*

*}*

*public String getPayTypeName(){*

*if(payType!=null) {*

*return OrderEnum.get(payType).getValue();*

*}*

*return null;*

*}*

*public Boolean hasPayed(){*

*return OrderEnum.PayStatusEnum.UN\_SEND.getId().equals(payStatus);*

*}*

*}*

Entity of ShopUser:

*@Entity(name="t\_shop\_user")*

*@JsonIgnoreProperties(value = { "hibernateLazyInitializer", "handler" })*

*public class ShopUser implements Serializable {*

*private Long id;*

*private Date createTime;*

*private String mobile;*

*private String salt;*

*private String password;*

*private String nickName;*

*private String avatar;*

*private String gender;*

*private Date lastLoginTime;*

*private String wechatOpenId;*

*private String wechatNickName;*

*private String wechatHeadImgUrl;*

*}*

**A.2** Controller layer

Regarding the code of the business logic layer, the functions of the system are implemented through the business logic layer. The following describes the main business codes.

Controller of Goods:

*@RestController*

*@RequestMapping("/goods")*

*public class GoodsController extends BaseController {*

*@Autowired*

*private GoodsService goodsService;*

*@Autowired*

*private GoodsSkuService goodsSkuService;*

*@Autowired*

*private CategoryService categoryService;*

*@Autowired*

*private AttrKeyService attrKeyService;*

*@Autowired*

*private FavoriteService favoriteService;*

*@RequestMapping(value = "/queryGoods", method = RequestMethod.GET)*

*public Object queryGoods(@RequestParam("idCategory") Long idCategory) {*

*Page<Goods> page = new PageFactory<Goods>().defaultPage();*

*List<Category> categories = categoryService.queryAll(SearchFilter.build("pid", idCategory));*

*List<Long> ids = Lists.newArrayList(idCategory);*

*categories.forEach(item -> {*

*ids.add(item.getId());*

*});*

*if (ids.size() == 1) {*

*page.addFilter(SearchFilter.build("idCategory", SearchFilter.Operator.EQ, idCategory));*

*} else {*

*page.addFilter(SearchFilter.build("idCategory", SearchFilter.Operator.IN, ids));*

*}*

*page.addFilter(SearchFilter.build("isOnSale", true));*

*page = goodsService.queryPage(page);*

*return Rets.success(page);*

*}*

*@RequestMapping(value = "/search", method = RequestMethod.GET)*

*public Object search(@RequestParam("key") String key) {*

*Page<Goods> page = new PageFactory<Goods>().defaultPage();*

*if (StringUtil.isNotEmpty(key)) {*

*page.addFilter(SearchFilter.build("name", SearchFilter.Operator.LIKE, key));*

*}*

*page.addFilter(SearchFilter.build("isOnSale", true));*

*page = goodsService.queryPage(page);*

*return Rets.success(page);*

*}*

*@RequestMapping(value = "/searchNew", method = RequestMethod.GET)*

*public Object searchNew() {*

*List<Goods> list = goodsService.queryAll(Lists.newArrayList(*

*SearchFilter.build("isNew", true),*

*SearchFilter.build("isOnSale", true)*

*));*

*return Rets.success(list);*

*}*

*@RequestMapping(value = "/searchHot", method = RequestMethod.GET)*

*public Object searchHot() {*

*List<Goods> list = goodsService.queryAll(Lists.newArrayList(*

*SearchFilter.build("isHot", true),*

*SearchFilter.build("isOnSale", true)*

*));*

*return Rets.success(list);*

*}*

*@RequestMapping(value = "/{id}", method = RequestMethod.GET)*

*public Object get(@PathVariable Long id) {*

*Goods goods = goodsService.get(id);*

*List<GoodsSku> skuList = goodsSkuService.queryAll(Lists.newArrayList(*

*SearchFilter.build("idGoods", id),*

*SearchFilter.build("isDeleted", false)*

*));*

*Map skuMap = Maps.newHashMap();*

*List<Map> tree = Lists.newArrayList();*

*if (!skuList.isEmpty()) {*

*List<AttrVal> attrValList = Lists.newArrayList();*

*List<AttrKey> attrKeyList = attrKeyService.queryBy(goods.getIdCategory());*

*for (AttrKey attrKey : attrKeyList) {*

*Map treeNode = Maps.newHashMap();*

*treeNode.put("k", attrKey.getAttrName());*

*List<Map> v = Lists.newArrayList();*

*List<AttrVal> attrValListChildren = attrKey.getAttrVals();*

*attrValList.addAll(attrValListChildren);*

*for (AttrVal attrVal : attrValListChildren) {*

*v.add(Maps.newHashMap(*

*"id", attrVal.getId(),*

*"name", attrVal.getAttrVal(),*

*"plain", true*

*));*

*}*

*treeNode.put("v", v);*

*treeNode.put("k\_s", "s" + attrKey.getId());*

*tree.add(treeNode);*

*}*

*Map<Long, AttrVal> attrValMap = Lists.toMap(attrValList, "id");*

*List<Map> skuList2 = Lists.newArrayList();*

*for (GoodsSku sku : skuList) {*

*Map oneSkuMap = Maps.newHashMap();*

*oneSkuMap.put("id", sku.getId());*

*oneSkuMap.put("price", sku.getPrice());*

*String[] attrValIdArr = sku.getCode().split(",");*

*for (String attrValId : attrValIdArr) {*

*AttrVal attrVal = attrValMap.get(Long.valueOf(attrValId));*

*oneSkuMap.put("s" + attrVal.getIdAttrKey(), attrVal.getId());*

*}*

*oneSkuMap.put("stock\_num", sku.getStock());*

*oneSkuMap.put("code", sku.getCode());*

*skuList2.add(oneSkuMap);*

*}*

*skuMap.put("list", skuList2);*

*skuMap.put("price", skuList.get(0).getPrice());*

*skuMap.put("collection\_id", skuList.get(0).getId());*

*skuMap.put("none\_sku", false);*

*} else {*

*skuMap.put("price", goods.getPrice());*

*skuMap.put("collection\_id", goods.getId());*

*skuMap.put("none\_sku", true);*

*}*

*skuMap.put("tree", tree);*

*skuMap.put("stock\_num", goods.getStock());*

*skuMap.put("hide\_stock", false);*

*Map result = Maps.newHashMap(*

*"goods", goods,*

*"sku", skuMap*

*);*

*Long idUser = JwtUtil.getUserId();*

*if (idUser != null) {*

*Favorite favorite = favoriteService.get(idUser, id);*

*result.put("favorite", favorite != null);*

*} else {*

*result.put("favorite", false);*

*}*

*return Rets.success(result);*

*}*

*}*

In this Controller about Goods, the following functions are mainly implemented to get the list of products under the specified category, search products according to keywords, and query popular products.

Controller of Payment:

*@RestController*

*@RequestMapping("/pay")*

*public class PayController extends BaseController {*

*@Autowired*

*private ShopUserService shopUserService;*

*@Autowired*

*private WeixinPayService weixinPayService;*

*@Autowired*

*private OrderService orderService;*

*@RequestMapping(value = "wx/prepare",method = RequestMethod.POST)*

*public Object wxPrepare(@RequestParam("orderSn")String orderSn){*

*ShopUser user = shopUserService.getCurrentUser();*

*if(StringUtil.isEmpty(user.getWechatOpenId())){*

*return Rets.failure("not Wechat user");*

*}*

*Order order = orderService.getByOrderSn(orderSn);*

*WxPayMpOrderResult wxOrder = weixinPayService.prepare(user,order);*

*if(wxOrder!=null) {*

*return Rets.success(wxOrder);*

*}*

*return Rets.failure("data error");*

*}*

*@RequestMapping(value = "wx/notify",method = RequestMethod.POST)*

*public Object wxNotify(){*

*String msg = weixinPayService.resultNotify();*

*return msg;*

*}*

*@RequestMapping(value = "queryResult/{orderSn}",method = RequestMethod.GET)*

*public Object wxNotify(@PathVariable("orderSn") String orderSn){*

*Order order = orderService.getByOrderSn(orderSn);*

*Boolean payResult = OrderEnum.PayStatusEnum.UN\_SEND.getId().equals(order.getPayStatus())*

*&& OrderEnum.PayStatusEnum.UN\_SEND.getId().equals(order.getStatus());*

*return Rets.success(payResult);*

*}*

*}*

In this Controller about payment, it mainly implements the following functions, checking whether the current user is a WeChat user, WeChat payment callback, and querying the payment result.

Controller of Category:

*@RestController*

*@RequestMapping("/shop/category")*

*public class CategoryController extends BaseController {*

*private Logger logger = LoggerFactory.getLogger(getClass());*

*@Autowired*

*private CategoryService categoryService;*

*@Autowired*

*private CategoryBannerRelService categoryBannerRelService;*

*@Autowired*

*private AttrKeyService attrKeyService;*

*@Autowired*

*private GoodsService goodsService;*

*@RequestMapping(value = "/list",method = RequestMethod.GET)*

*public Object list() {*

*List<CategoryNode> list = categoryService.getCategories();*

*return Rets.success(list);*

*}*

*@RequestMapping(value = "/getAll",method = RequestMethod.GET)*

*public Object getAll() {*

*List<Category> categories = categoryService.queryAll();*

*return Rets.success(categories);*

*}*

*@RequestMapping(method = RequestMethod.POST)*

*@BussinessLog(value = "Edit Product Category", key = "name")*

*@RequiresPermissions(value = {Permission.CATEGORY\_EDIT})*

*public Object save(@ModelAttribute Category category){*

*if(category.getId()==null){*

*categoryService.insert(category);*

*}else {*

*Category old = categoryService.get(category.getId());*

*category.setCreateBy(old.getCreateBy());*

*category.setCreateTime(old.getCreateTime());*

*categoryService.update(category);*

*}*

*return Rets.success();*

*}*

*@RequestMapping(method = RequestMethod.DELETE)*

*@BussinessLog(value = "Delete product category", key = "id")*

*@RequiresPermissions(value = {Permission.CATEGORY\_EDIT})*

*public Object remove(Long id){*

*if (id == null) {*

*throw new ApplicationException(ApplicationExceptionEnum.REQUEST\_NULL);*

*}*

*long goodsCount = goodsService.count(SearchFilter.build("idCategory",id));*

*if(goodsCount>0){*

*throw new ApplicationException(ApplicationExceptionEnum.DATA\_CANNOT\_REMOVE);*

*}*

*categoryService.deleteById(id);*

*return Rets.success();*

*}*

*@RequestMapping(value="/getBanners/{idCategory}",method = RequestMethod.GET)*

*public Object getBanners(@PathVariable("idCategory") Long idCategory){*

*if (idCategory == null) {*

*throw new ApplicationException(ApplicationExceptionEnum.REQUEST\_NULL);*

*}*

*List<CategoryBannerRel> relList = categoryBannerRelService.queryAll(SearchFilter.build("idCategory", SearchFilter.Operator.EQ,idCategory));*

*List<Banner> bannerList = Lists.newArrayList();*

*relList.forEach( item->{*

*bannerList.add(item.getBanner());*

*});*

*return Rets.success(bannerList);*

*}*

*@RequestMapping(value ="getAttrKeys/{idCategory}",method = RequestMethod.GET)*

*public Object getAttrKeys(@PathVariable("idCategory") Long idCategory){*

*if (idCategory == null) {*

*throw new ApplicationException(ApplicationExceptionEnum.REQUEST\_NULL);*

*}*

*List<AttrKey> list = attrKeyService.queryBy(idCategory);*

*return Rets.success(list);*

*}*

*@RequestMapping(value="/removeBanner/{idCategory}/{idBanner}",method = RequestMethod.DELETE)*

*@RequiresPermissions(value = {Permission.CATEGORY\_EDIT})*

*public Object removeBanner(@PathVariable("idCategory") Long idCategory,*

*@PathVariable("idBanner") Long idBanner){*

*if (idCategory == null) {*

*throw new ApplicationException(ApplicationExceptionEnum.REQUEST\_NULL);*

*}*

*CategoryBannerRel rel = categoryBannerRelService.get(Lists.newArrayList(*

*SearchFilter.build("idCategory",idCategory),*

*SearchFilter.build("idBanner",idBanner)*

*));*

*if(rel!=null){*

*categoryBannerRelService.delete(rel);*

*}*

*return Rets.success();*

*}*

*@RequestMapping(value="/setBanner/{idCategory}/{idBanner}",method = RequestMethod.POST)*

*@RequiresPermissions(value = {Permission.CATEGORY\_EDIT})*

*public Object setBanner(@PathVariable("idCategory") Long idCategory,*

*@PathVariable("idBanner") Long idBanner){*

*if (idCategory == null) {*

*throw new ApplicationException(ApplicationExceptionEnum.REQUEST\_NULL);*

*}*

*CategoryBannerRel rel = categoryBannerRelService.get(Lists.newArrayList(*

*SearchFilter.build("idCategory",idCategory),*

*SearchFilter.build("idBanner",idBanner)*

*));*

*if(rel!=null){*

*return Rets.success();*

*}*

*rel = new CategoryBannerRel();*

*rel.setIdCategory(idCategory);*

*rel.setIdBanner(idBanner);*

*categoryBannerRelService.insert(rel);*

*return Rets.success();*

*}*

*@PostMapping(value="/changeShowIndex/{idCategory}/{showIndex}")*

*@RequiresPermissions(value = {Permission.CATEGORY\_EDIT})*

*public Object changeShowIndex(@PathVariable("idCategory") Long idCategory,*

*@PathVariable("showIndex") Boolean showIndex){*

*if (idCategory == null) {*

*throw new ApplicationException(ApplicationExceptionEnum.REQUEST\_NULL);*

*}*

*Category category = categoryService.get(idCategory);*

*category.setShowIndex(showIndex);*

*categoryService.update(category);*

*return Rets.success();*

*}*

*}*

In this product category Controller, the following functions are implemented: get all categories, edit product categories, delete product categories, set banners for products, and change the order of product display.

**A.3** Database layer

The following is the configuration of the database in the properties file in spring.

*spring.datasource.url=jdbc:mysql://localhost:3306/linjiashop?useUnicode=true&characterEncoding=UTF8&useSSL=false&allowPublicKeyRetrieval=true*

*spring.datasource.username=root*

*spring.datasource.password=root*

*spring.jpa.database-platform=org.hibernate.dialect.MySQL5InnoDBDialect*

*spring.datasource.sql-script-encoding=utf-8*

*spring.jpa.show-sql=true*

The following is the base repository implementation.

*public class BaseRepositoryImpl<T, ID extends Serializable>*

*extends SimpleJpaRepository<T, ID>*

*implements BaseRepository<T, ID> {*

*private final EntityManager entityManager;*

*private Class<T> klass;*

*BaseRepositoryImpl(JpaEntityInformation<T, ID> entityInformation,*

*EntityManager entityManager) {*

*super(entityInformation, entityManager);*

*this.entityManager = entityManager;*

*this.klass = (Class<T>) entityInformation.getJavaType();*

*}*

*@Override*

*public List<Map> queryBySql(String sql) {*

*Query query = entityManager.createNativeQuery(sql);*

*query.unwrap(NativeQueryImpl.class)*

*.setResultTransformer(Transformers.ALIAS\_TO\_ENTITY\_MAP);*

*List list = query.getResultList();*

*return list;*

*}*

*@Override*

*public List<Map> queryMapBySql(String sql) {*

*Query query = entityManager.createNativeQuery(sql);*

*query.unwrap(NativeQueryImpl.class)*

*.setResultTransformer(Transformers.ALIAS\_TO\_ENTITY\_MAP);*

*List list = query.getResultList();*

*return list;*

*}*

*@Override*

*public List<?> queryBySql(String sql, Class<?> klass) {*

*List<Map> list = queryBySql(sql);*

*if(list.isEmpty()){*

*return null;*

*}*

*List result = Lists.newArrayList();*

*for(Map map :list){*

*try {*

*Object bean = Mapl.maplistToObj(map,klass);*

*result.add(bean);*

*}catch (Exception e){*

*}*

*}*

*return result;*

*}*

*@Override*

*public List<?> queryObjBySql(String sql, Class<?> klass) {*

*List<Map> list = queryMapBySql(sql);*

*if(list.isEmpty()){*

*return Lists.newArrayList();*

*}*

*List result = Lists.newArrayList();*

*for(Map map:list){*

*try {*

*Object bean = Mapl.maplistToObj(map, klass);*

*result.add(bean);*

*} catch (Exception e) {*

*}*

*}*

*return result ;*

*}*

*@Override*

*public T getOne(ID id){*

*return findById(id).get();*

*}*

*@Override*

*public T get(String sql) {*

*List<T> list = entityManager.createNativeQuery(sql,klass).getResultList();*

*return list.get(0);*

*}*

*@Override*

*public int execute(String sql) {*

*return entityManager.createNativeQuery(sql).executeUpdate();*

*}*

*@Override*

*public Class<T> getDataClass() {*

*return klass;*

*}*

*@Override*

*public List<T> query(String sql) {*

*return entityManager.createNativeQuery(sql,klass).getResultList();*

*}*

*@Override*

*public Object getBySql(String sql, Class<?> klass) {*

*List list = queryBySql(sql,klass);*

*if(list.isEmpty()){*

*return null;*

*}*

*return list.get(0);*

*}*

*}*

**A.4** Front-end page layer

The following is the personal information page after the user logs in, where the user can view the order, the delivery status of the order, and modify the personal information, etc.

*<template>*

*<div>*

*<img class="user-poster" src="@/assets/img/banner.jpg" >*

*<van-row class="user-links">*

*<van-col span="6">*

*<van-icon name="pending-payment" @click="toOrder(1)"/>*

*Payment pending*

*</van-col>*

*<van-col span="6">*

*<van-icon name="tosend" @click="toOrder(2)"/>*

*To be shipped*

*</van-col>*

*<van-col span="6">*

*<van-icon name="logistics" @click="toOrder(3)"/>*

*Shipped*

*</van-col>*

*<van-col span="6">*

*<van-icon name="bag-o" @click="toOrder(4)"/>*

*Completed*

*</van-col>*

*</van-row>*

*<van-cell-group class="user-group">*

*<van-cell icon="records" title="Orders" is-link to="order" />*

*</van-cell-group>*

*<van-cell-group>*

*<van-cell icon="location-o" title="Address" is-link to="address"/>*

*<van-cell icon="star-o" title="Favorite" is-link to="favorite"/>*

*<van-cell icon="points" title="Point" is-link @click="sorry"/>*

*<van-cell icon="gold-coin-o" title="Coupons" is-link @click="sorry" />*

*<van-cell icon="gift-o" title="Gift" is-link @click="sorry"/>*

*<van-cell icon="setting-o" title="Setting" is-link to="setting"/>*

*</van-cell-group>*

*<br><br>*

*<van-tabbar v-model="activeFooter">*

*<van-tabbar-item icon="home-o" replace to="/index">Home</van-tabbar-item>*

*<van-tabbar-item icon="search" replace to="/search">Found</van-tabbar-item>*

*<van-tabbar-item icon="home-o" replace to="">Upload</van-tabbar-item>*

*<van-tabbar-item icon="cart-o" replace to="/cart">Shopping Cart</van-tabbar-item>*

*<van-tabbar-item icon="user-o" replace to="/user">Me</van-tabbar-item>*

*</van-tabbar>*

*</div>*

*</template>*

*<script src="./user.js"></script>*

*<style lang="less">*

*.user {*

*&-poster {*

*width: 100%;*

*display: block;*

*}*

*&-group {*

*margin-bottom: 15px;*

*}*

*&-links {*

*padding: 15px 0;*

*font-size: 12px;*

*text-align: center;*

*background-color: #fff;*

*.van-icon {*

*display: block;*

*font-size: 24px;*

*}*

*}*

*}*

*</style>*

The following is the front-end code for the display of second-hand items. Through the interaction with the back-end API, the front-end will obtain the data in the database and display it on the page in a fixed format.

*<template>*

*<div>*

*<van-tabs v-model="activeNav" @click="onClickNav">*

*<!--<van-tab title="recommend" name="0" >-->*

*<!--</van-tab>-->*

*<van-tab v-for="nav in navList" :title="nav.name" :name="nav.id" >*

*</van-tab>*

*</van-tabs>*

*<van-swipe :autoplay="3000" style="height: 200px;">*

*<van-swipe-item v-for="(image, index) in banners" :key="index" @click="clickSwipe">*

*<a :href="image.url"><img v-lazy="image.path" style="width:100%;"/></a>*

*</van-swipe-item>*

*</van-swipe>*

*<van-list*

*v-model="loading"*

*:finished="finished"*

*@load="loadMore"*

*>*

*</van-list>*

*<van-card v-for="(goods,index) in goodsList" :key="index"*

*:num="goods.stock"*

*:price="formatPrice(goods.price)"*

*:desc="goods.descript"*

*:title="goods.name"*

*:thumb="goods.img"*

*:thumb-link="goods.link"*

*@click="viewGoodsDetail(goods.id)"*

*/>*

*<br><br><br><br>*

*<van-tabbar v-model="activeFooter">*

*<van-tabbar-item icon="home-o" replace to="/index">Home</van-tabbar-item>*

*<van-tabbar-item icon="search" replace to="/search">Found</van-tabbar-item>*

*<van-tabbar-item icon="home-o" replace to="">Upload</van-tabbar-item>*

*<van-tabbar-item icon="cart-o" replace to="/cart">Shopping Cart</van-tabbar-item>*

*<van-tabbar-item icon="user-o" replace to="/user">Me</van-tabbar-item>*

*</van-tabbar>*

*</div>*

*</template>*

*<script src="./goodsList.js"></script>*

The following is the shopping cart interface code.

*<template>*

*<div class="cart">*

*<van-nav-bar*

*title="Shopping Cart"*

*:right-text="rightText"*

*@click-right="onClickRight"*

*/>*

*<van-checkbox-group class="card-goods" v-model="checkedCartItem" v-if="isLogin && cartList.length>0">*

*<div v-for="item in cartList"*

*:key="item.id" class="card-goods\_\_item">*

*<van-checkbox*

*:name="item.id"*

*></van-checkbox>*

*<van-card*

*style="margin-left:15px;"*

*:title="item.title"*

*:desc="item.goods.descript"*

*:price="formatPrice(item.price)"*

*:thumb="item.thumb"*

*>*

*<div slot="footer">*

*<van-stepper v-model="item.count" @change="stepperEvent(item,arguments)" disableInput/>*

*</div>*

*</van-card>*

*</div>*

*<br><br><br><br><br>*

*</van-checkbox-group>*

*<van-submit-bar*

*v-show="!showEdit"*

*:price="totalPrice"*

*:disabled="!checkedCartItem.length"*

*:button-text="submitBarText"*

*@submit="submit"*

*v-if="isLogin && cartList.length>0"*

*>*

*<van-checkbox v-model="checkedAll" @click="checkAll" style="padding: 0 10px;">Select All</van-checkbox>*

*</van-submit-bar>*

*<van-submit-bar*

*v-show="showEdit"*

*:price="totalPrice"*

*:disabled="!checkedCartItem.length"*

*button-text="Delete"*

*@submit="remove"*

*v-if="isLogin && cartList.length>0"*

*>*

*<van-checkbox v-model="checkedAll" @click="checkAll" style="padding: 0 10px;">Select All</van-checkbox>*

*</van-submit-bar>*

*<div class="no-data" v-if="isLogin && cartList.length ===0 ">*

*<p style="text-align: center;color:lightgray">*

*<van-icon name="cart-o" style="text-align: center" size="3rem"/><br>*

*Shopping cart is still empty</p>*

*<van-button type="default" block round @click="toHome">Go shopping</van-button>*

*</div>*

*<div class="no-data" v-if="!isLogin" >*

*<p style="text-align: center;color:lightgray">*

*<van-icon name="cart-o" style="text-align: center" size="3rem"/><br>*

*Not logged in yet</p>*

*<van-button type="primary" block round @click="toLogin">Login Now</van-button>*

*</div>*

*<van-tabbar v-model="activeFooter">*

*<van-tabbar-item icon="home-o" replace to="/index">Home</van-tabbar-item>*

*<van-tabbar-item icon="search" replace to="/search">Found</van-tabbar-item>*

*<!-- <van-tabbar-item icon="home-o" replace to="">Upload</van-tabbar-item>-->*

*<van-tabbar-item icon="cart-o" replace to="/cart">Shopping Cart</van-tabbar-item>*

*<van-tabbar-item icon="user-o" replace to="/user">Me</van-tabbar-item>*

*</van-tabbar>*

*</div>*

*</template>*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Code* | | | | | *Description* | | | | *Additional data* | |
|  | | | | | Text documents | | | |  | |
|  | | | | |  | | | |  | |
| *BSUIR DP 1-53 01 02 01 002 EN* | | | | | Explanatory note | | | | 72 p. | |
|  | | | | | Supervisor’s opinion | | | | 1 p. | |
|  | | | | | Review | | | | 1 p. | |
|  | | | | |  | | | |  | |
|  | | | | | Graphic documents | | | |  | |
| *SUIR DP 000000 001 SD* | | | | | Second-hand trading company information system organization structure | | | | Format А1 | |
|  | | | | |  | | | |  | |
| *SUIR DP 000000 002 SD* | | | | | Second-hand trading company information system flowchart | | | | Format А1 | |
|  | | | | |  | | | |  | |
| *SUIR DP 000000 003 PD* | | | | | Second-hand trading company information system IDEF0 diagrams | | | | Format А1 | |
|  | | | | |  | | | |  | |
| *SUIR DP 000000 004 PD* | | | | | Second-hand trading company information system activity diagram | | | | Format А1 | |
|  | | | | |  | | | |  | |
| *SUIR DP 000000 005 PD* | | | | | Second-hand trading company information system use case diagram | | | | Format А1 | |
|  | | | | |  | | | |  | |
| *SUIR DP 000000 006 PD* | | | | | Second-hand trading company information system screenshots | | | | Format А1 | |
|  |  |  |  |  | *BSUIR DP 1-53 01 02 01 002 D1* | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| *Ver.* | *Sheet* | *Doc. No.* | *Sign.* | *Date* | *Automated Second-Hand Trading Company Information System*  ***Diploma Project Register*** | *Letter* | | | *Sheet* | *Sheets* |
| *Prepared by* | | *Li Boyi* |  |  |  | *Т* |  | *72* | *72* |
| *Checked by* | | *Khajynava* |  |  | *Information Technologies in Automated Systems Department*  *Group 920611* | | | | |
| *Tech. Insp.* | | *Khajynava* |  |  |
| *Stand. Insp.* | | *Batin* |  |  |
| *Approved by* | | *Naurotsky* |  |  |