Theory and Practice of Science and Technology
2024, VOL. 5, NO. 4, 20-29
DOI: 10.47297/taposatWSP2633-456903.20240504

Design and Implementation of a Campus Secondhand Commodity Trading Platform Based on JAVA Program Design

Kaiwen Deng, Yanbin Long

University of Science and Technology Liaoning

ABSTRACT

This paper aims to design and implement a secondhand commodity trading platform for campus users, which integrates functions such as commodity information publishing, browsing, searching, trading, and evaluation. It is designed to provide a convenient and efficient channel for the exchange of secondhand items among teachers and students on campus. The system uses Java as the backend development language, utilizes the SpringMVC framework to build RESTful APIs, and implements a frontend and backend separation development model. For the database, MySQL is chosen as the data storage solution, with a reasonable database model designed to support efficient data queries and management. The frontend display uses HTML, CSS, and JavaScript technologies, combined with Ajax to achieve asynchronous data interaction, enhancing user experience. This paper elaborates on the system's design ideas, key technical implementations, database design, and system testing, and the feasibility and effectiveness of the system are verified through actual deployment.

KEYWORDS

Java; SpringMVC; MySQL; Campus Secondhand Trading Platform; Frontend and Backend Separation; RESTful API

1 Introduction

With the rapid development of Internet technology and the popularization of smart devices, campus life is becoming increasingly digital and convenient. Against this backdrop, the transaction of secondhand goods on campus has become an important market demand. Traditional secondhand trading methods suffer from problems such as information asymmetry, low transaction efficiency, and lack of trust, while online secondhand trading platforms can effectively solve these issues by providing a safe and efficient trading environment. Therefore, designing and implementing a Javabased campus secondhand commodity trading platform not only helps to promote the recycling of resources within the campus, reducing waste, but also provides students with a practical platform to exercise programming and project management skills, which has significant practical significance and research value.

At present, there are many wellknown secondhand trading platforms both domestically and internationally, such as Xianyu and Zhuanzhuan, which have achieved remarkable success in the market. However, these platforms mainly target a broader user group and have not been specifically optimized and designed for the campus user group. Within the campus, although there are some small secondhand trading groups or platforms, they often have single functions and poor user experience. Therefore, designing a secondhand commodity trading platform specifically for campus users to meet the specific needs of teachers and students is a challenging and innovative research

direction.

2 System Requirements Analysis

In the software or system development process, system requirements analysis is a crucial step, ensuring that the development team can accurately understand and meet the needs of the endusers. The following is a detailed elaboration of the system requirements analysis:

2.1 User Requirements Analysis

User requirements analysis aims to deeply understand the needs of the user group to design a system that meets user expectations.

2.1.1 User Role Division

User role division is the process of identifying different user groups in the system. By dividing user roles, we can more accurately understand the needs and behavior patterns of each user group. For example, in an ecommerce system, user roles may include ordinary buyers, sellers, administrators, etc. Each role has its unique permissions and functional requirements.

2.1.2 Functional Requirements Analysis

Functional requirements analysis further refines the specific functions required by each role based on the clear definition of user roles. These functional requirements are the core of system design, directly determining what the system can do. Functional requirements analysis should be exhaustive, specific, and as quantifiable as possible. For example, in an ecommerce system, the functions that ordinary buyers may need include browsing products, searching for products, adding to the shopping cart, placing orders, and making payments.

2.2 Business Process Analysis

Business process analysis is the process of detailing and optimizing the internal business processes of the system. It focuses on how the system supports business operations, including the various stages of the process, participants, and their interactions. Through business process analysis, it can be ensured that the system can handle business efficiently, reducing errors and redundant steps.

2.3 Data Flow Analysis

Data flow analysis focuses on the flow and transformation process of data within the system. It identifies the transmission paths, processing methods, and storage locations of data between the system and the outside world. Data flow analysis helps ensure that the system can correctly process and store data, supporting business decisions and data analysis. In data flow analysis, data flow diagrams are usually drawn to visually represent the flow and processing of data.

2.4 System Nonfunctional Requirements Analysis

Nonfunctional requirements refer to other characteristics or constraints that the system must meet in addition to functional requirements. These requirements do not directly participate in

business processing but have a significant impact on the overall performance and user experience of the system. Common nonfunctional requirements include:

Performance requirements: such as response time, throughput, and concurrent user numbers, used to measure the system's processing capabilities.

Usability requirements: ensuring that the system is easy to use, easy to learn, and can run stably in different environments.

Security requirements: protecting the system from unauthorized access, data leaks, and malicious attacks.

Maintainability requirements: ensuring that the system is easy to modify, expand, and maintain to adapt to future changes in requirements.

Portability requirements: the system should be able to run on different hardware and software platforms, reducing migration and deployment costs.

By comprehensively analyzing the functional and nonfunctional requirements of the system, it can be ensured that the development team can design a system that meets business needs and has good performance and user experience.

3 System Design

3.1 System Overall Architecture Design

The system adopts a multilayered architecture design pattern to ensure the system's scalability, maintainability, and security. The overall architecture is divided into the frontend display layer, the backend service layer, the data access layer, and the database layer. The frontend display layer is responsible for interacting with users, displaying product information, user interfaces, etc.; the backend service layer handles business logic, including user management, product publishing, browsing, searching, transaction management, and evaluation functions; the data access layer is responsible for interacting with the database, performing data creation, reading, updating, and deletion operations; the database layer stores all the data required by the system. In addition, the system also includes some auxiliary modules, such as security authentication modules, log management modules, etc., to ensure the security and traceability of the system.

3.2 System Functional Module Design

3.2.1 User Management Module

The user management module is responsible for user information registration, login, information updating, and permission management. When registering, users need to provide necessary personal information, such as username, password, contact information, etc., and verify through email or mobile phone verification code. When logging in, the system verifies the username and password entered by the user, and after successful verification, a session token (Token) is generated for subsequent request authentication. User information updates allow users to modify personal information, such as avatars, nicknames, etc. Permission management allocates different operational permissions based on user roles (such as ordinary users, administrators).

3.2.2 Product Publishing Module

The product publishing module allows users to publish secondhand product information, including product name, description, price, pictures, categories, and condition of newness. After the user fills in the product information, the system will perform preliminary verification, such as

whether the price is reasonable, whether the pictures are uploaded successfully, etc. After passing the verification, the product information will be saved to the database and synchronized to the front end for other users to browse.

3.2.3 Product Browsing and Search Module

The product browsing and search module provides product list display and search functions. Users can browse or search for products they are interested in through categories or keywords. The search results will be sorted according to certain rules (such as price, publication time) and display the product details, including pictures, prices, descriptions, etc. In addition, the module also supports product filtering functions, allowing users to filter according to product categories, price ranges, and other conditions.

3.2.4 Transaction Management Module

The transaction management module handles the transaction processes such as product purchase, payment, shipping, receipt, and refund. After the user selects the product, they can generate an order through the system and choose a payment method for payment. After the payment is successful, the seller will receive the order notification and prepare for shipping. After the buyer receives the product, they can confirm receipt or apply for a refund. The system will automatically update the order status according to the transaction status and provide corresponding notifications and reminders.

3.2.5 Evaluation and Feedback Module

The evaluation and feedback module allows users to evaluate and provide feedback on transactions. After the transaction is completed, both buyers and sellers can evaluate each other, including aspects such as product quality and service attitude. The evaluation information will be saved in the database and displayed on the product detail page and the user's personal homepage for other users to refer to. In addition, users can also provide feedback on the system itself, making suggestions for improvement and opinions.

3.3 Database Design

3.3.1 Entity Relationship Model

Based on system requirements, the corresponding entity relationship model is designed. The main entities include users, products, orders, evaluations, etc., which are interconnected through association relationships (such as onetoone, onetomany, manytomany). For example, a user can publish multiple products, a product can belongto multiple categories, an order can include multiple products, etc. It is necessary to clearly show the relationships between these entities and label each entity's attributes and foreign keys.

3.3.2 Database Table Design

Design the specific database table structure. Each table corresponds to an entity or an association relationship between entities and includes corresponding fields (such as primary keys, foreign keys, regular fields) and field types (such as integer, character, date, etc.). For example, the user table may include fields like user ID (primary key), username, password (stored encrypted), contact information, etc.; the product table may include fields like product ID (primary key), user ID

(foreign key), product name, description, price, etc. In addition, it is necessary to design appropriate indexes to optimize query performance and consider security and integrity constraints for data (such as notnull constraints, unique constraints, etc.).

3.4 Interface Design

3.4.1 RESTful API Design Principles

The system adopts the RESTful API design style, adhering to the following principles:

Use HTTP methods (such as GET, POST, PUT, DELETE) to represent operations on resources;

Use stateless communication, meaning each request contains enough information to process the request, and the server does not save any client state;

Use a hierarchical URL structure to represent the relationships between resources;

Use HTTP status codes to represent the result status of the request;

The returned data format is JSON or XML to be compatible with various frontend technologies.

3.4.2 Key Interface Description

Here is an explanation of several key interfaces:

User registration interface: Accepts the registration information submitted by the user (such as username, password, etc.), verifies it, and returns the registration result (success or failure).

User login interface: Accepts the username and password submitted by the user, verifies it, and returns the session token (Token) or error message.

Product publishing interface: Accepts the product information submitted by the user (such as product name, description, price, etc.), verifies it, saves it to the database, and returns the operation result. Product list interface: Returns the product list information based on the query conditions (such as category, keywords, etc.).

Order creation interface: Accepts the order information submitted by the user (such as product ID, quantity, etc.), verifies it, creates the order, and returns the order information.

Evaluation submission interface: Accepts the evaluation information submitted by the user (such as score, comment content, etc.), saves it to the database, and returns the operation result.

These interfaces will serve as the communication bridge between the backend service layer and the frontend display layer, implementing the various functions of the system.

4 System Implementation

4.1 Development Environment Setup

Before implementing the system, it is necessary to set up the development environment. This includes installing the Java Development Kit (JDK), Integrated Development Environment (such as IntelliJ IDEA or Eclipse), Maven (for project building and dependency management), MySQL database management system, and corresponding database client tools (such as MySQL Workbench). In addition, it is also necessary to configure the Tomcat server as the running environment for web applications.

4.2 Backend Development Implementation

4.2.1 SpringMVC Framework Setup

1. Create a Maven project: Create a new Maven project in the IDE and configure the pom.xml file,

adding dependencies such as SpringMVC, SpringJDBC, and MySQL driver.

2. Configure web. xml: Create a web. xml file in the `src/main/webapp/WEBINF` directory, configuring the SpringMVC DispatcherServlet and character set filter.

3. Configure Spring configuration files: Create Spring configuration files (such as applicationContext. xml and springmvc. xml) in the `src/main/resources` directory, configuring the data source, transaction manager, component scanning, etc.

4.2.2 Data Access Layer Implementation (DAO)

Implement the Data Access Layer using JDBC or JPA (such as Hibernate). Write DAO interfaces and corresponding implementation classes, using JDBCTemplate or JPARepository to perform database operations.

// Example: DAO implementation using SpringJDBCTemplate

```java
@Repository
public class UserDaoImpl implements UserDao {
@Autowired
private JdbcTemplate jdbcTemplate;
@Override
public User findUserById(int id) {
String sql = "SELECT * FROM users WHERE id = ?";
return jdbcTemplate. queryForObject(sql, new Object[] {id}, new BeanPropertyRowMapper<>
(User.class));
```

4.2.3 Business Logic Layer Implementation (Service)

Write Service interfaces and corresponding implementation classes to handle business logic. The Service layer will call the DAO layer to access the database.

```java
@Service
public class UserServiceImpl implements UserService {
@Autowired
private UserDao userDao;
@Override
public User getUserById(int id) {
return userDao.findUserById(id);
```

4.2.4 Control Layer Implementation (Controller)

Use SpringMVC's Controller to handle HTTP requests and responses.

```java
@RestController
@RequestMapping("/api/users")
public class UserController {
@Autowired
private UserService userService;
@GetMapping("/{id}")
public ResponseEntity<User> getUserById(@PathVariable int id) {
User user = userService.getUserById(id);
```

```
if (user != null) {
return ResponseEntity.ok(user);
} else {
return ResponseEntity.notFound().build();
```

4.3 Database Implementation

4.3.1 MySQL Database Configuration

Configure the data source in the Spring configuration file, specifying the database URL, username, password, etc.

4.3.2 SQL Script Writing and Execution

Write SQL scripts to create databases, tables, and initialize data. Use MySQL client tools or IDE database management tools to execute these scripts.

4.4 Frontend Development Implementation

4.4.1 Page Layout and Style Design

Use HTML, CSS, and possible JavaScript libraries (such as Bootstrap) to design the layout and style of the page.

4.4.2 JavaScript Script Writing

Write JavaScript scripts to handle frontend logic, such as form validation, AJAX requests, etc.

4.4.3 Frontend and Backend Interaction Implementation

Use AJAX (such as through jQuery or Fetch API) to interact with the backend, send HTTP requests, and process responses.

```javascript
// Example: Using Fetch API to send a GET request to get user information
fetch('/api/users/1')
.then(response => response.json())
.then(data => {
console.log(data); // Process the returned user information
})
.catch((error) => {
console.error('Error:', error);
});
```

Please note that the above code is only an example, and actual projects need to be adjusted and expanded according to specific requirements.

5 System Testing

After the system development is completed, comprehensive testing is a key step to ensure the quality, stability, and security of the system. The following are the main links of system testing:

5.1 Test Environment Setup

Before system testing, it is necessary to set up a test environment as similar as possible to the production environment. This includes installing necessary software (such as databases, web servers, application servers, etc.), configuring networks, setting security policies, etc. The test environment should be independent to avoid potential impacts on the production environment.

5.2 Unit Testing

Unit testing is testing for the smallest testable units in the software (such as functions, methods). It verifies whether the code unit works as expected and is independent of other code units. In Java projects, frameworks such as JUnit or TestNG are usually used to write and run unit tests. Unit tests should cover all code paths, including normal and exceptional cases.

5.3 Integration Testing

Integration testing is based on unit testing, combining various modules to test whether their interfaces and interactions are normal. Integration testing aims to discover interface errors between modules, global data structure errors, and module dependency errors. Tools such as SpringBootTest and MavenSurefire can be used to assist in integration testing.

5.4 System Testing

System testing is to test the entire system as a whole to verify whether it meets the functional, performance, security, and other requirements defined in the requirement specifications.

5.4.1 Functional Testing

Functional testing focuses on whether the system has implemented all functions as required. Testers will write test cases, simulate user operations, and check whether the system output is as expected. Functional testing includes positive testing (verifying whether the system can correctly complete expected operations) and negative testing (verifying whether the system can correctly handle unexpected inputs or exceptional situations).

5.4.2 Performance Testing

Performance testing evaluates the performance indicators of the system under different loads, such as response time, throughput, and concurrent user numbers. Testers will use tools like LoadRunner, JMeter, etc.,

5.4.3 Security Testing

Security testing checks for security vulnerabilities in the system, such as SQL injection, crosssite scripting (XSS), crosssite request forgery (CSRF), etc. Testers will use penetration testing tools, code reviews, and other methods to discover potential security issues and propose repair suggestions.

5.5 Test Result Analysis

After testing is completed, it is necessary to collect, organize, and analyze the test results. Analyze the test data to determine whetherthe system has passed the tests, which parts have problems, and

the severity of the problems. Based on the test results, write a test report summarizing the test work, proposing repair suggestions, and planning subsequent test work. If the system fails the tests, it is necessary to work closely with the development team to locate, fix, and retest the problems until the system meets all test requirements.

6 System Deployment and Operation

6.1 Deployment Environment Preparation

Before system deployment, it is necessary to prepare the corresponding deployment environment. This includes the installation and configuration of physical or virtual servers, operating systems, database management systems, web servers, and application servers, as well as ensuring the security, stability, and reliability of the network environment to meet the needs of system operation. In addition, it is also necessary to prepare necessary backup and recovery strategies to prevent data loss or system failures.

6.2 System Deployment Steps

System deployment is a multistep process, which usually includes the following main steps:

1). Environment configuration: Configure the server operating system, database, web server, and application server parameters according to system requirements.

2). Application deployment: Deploy the compiled and packaged application to the application server and perform necessary configurations, such as database connection pools, log recording, etc.

3). Data migration: If the system relies on specific data, it is necessary to migrate the data from the test environment to the production environment and ensure the integrity and consistency of the data.

4). Security reinforcement: Reinforce the system's security by configuring firewalls, enabling SSL/TLS encryption, setting access controls, etc., to improve the system's security.

5). Testing and verification: Perform the final testing of the system in the production environment to ensure that the system can run normally and meet all requirements.

6). User training: Train users on how to use the system to ensure that they can operate it proficiently and make full use of its functions.

6.3 System Operation Effect Display

After the system is successfully deployed and runs stably, the system operation effect can be displayed through various means such as system interface display, user interaction process, and system performance indicators. This helps users understand the system's functions and usage methods and evaluates the actual operation effect of the system. In addition, user feedback can also be collected for further optimization and improvement of the system.

7 Summary and Outlook

7.1 Summary of Research Work

In the summary of research work, review the entire project development process, including requirement analysis, system design, system implementation, system testing, system deployment, and operation. Summarize the main achievements and highlights of the project, such as technological innovation points, functional implementation, and performance optimization effects.

At the same time, reflect on the problems and challenges encountered during the project development and summarize the solutions and lessons learned.

7.2 Existing Problems and Insufficiencies

While summarizing the research results, objectively analyze the existing problems and insufficiencies. This may include defects in system design, shortcomings in functional implementation, performance bottlenecks, poor user experience, etc. By deeply analyzing the causes of these problems and insufficiencies, targeted suggestions and directions for future improvements can be provided.

7.3 Future Research Directions and Outlook

Looking to the future, propose directions for system improvement and development trends. These can be based on current system shortcomings and changes in user needs. For example, consider introducing new technology frameworks or tools to optimize system performance; expand system functions to meet the needs of more users; strengthen system security to cope with increasingly complex network threats, etc. At the same time, the longterm development plan and strategic goals of the system can also be formulated in conjunction with industry development trends and market demands.

Funding

Supported by the 2024 Laboratory Open Fund Project of University of Science and Technology Liaoning

References

[1] Design and Implementation of a Campus Secondhand Trading App Based on Mobile GIS [J]. Hu Kehong; Jiang Hao; Zhang Zhen. Computer Knowledge and Technology, 2020(14)

[2] Problems and Solutions in the Use of Android Studio [J]. Xie Xiquan. Computer Programming Tips and Maintenance, 2020(02)

[3] Improving the Reuse Rate of Idle Items, Building a Conservative Campus [J]. Du Xiaoxue; Li Rong; Liu Hongli; Li Ting. Science and Technology Vision, 2019(20)

[4] Development of a University Palm Secondhand Trading Software "Kan Kan Bei" Based on Bmob Backend Cloud [J]. Zhang Enci; Qu Tian; Li Cheng; Li Min; He Xin. Information Technology and Informatization, 2018(04)

[5] Design of a "Campus Flea Market" Mobile App Based on Android [J]. Ren Peihua; Xuan YuRu. Computer and Digital Engineering, 2016(11)

[6] Design and Implementation of a Campus Secondhand Goods Trading Platform Based on the Android System [J]. Lv Shuo. Peer, 2016(11)

[7] Application of Bmob Cloud Platform in Android App Development [J]. Du Wei. Communications World, 2016(03)

[8] Application and Research of Mobile GIS in Digital Campus Services [J]. Wang ShiJu; Yang Bin; Gao Guisheng; Li Maojiao. Geographic Information World, 2015(03)

[9] Application of Bmob Cloud Platform in Android App Development [J]. Zhou Ran; Gao Yuzhu. Minicomputers and Applications, 2015(01)