Software Requirements Specification for Campus Mutual Assistance Platform

1. Introduction

1.1 Purpose of Writing

This Software Requirements Specification (SRS) aims to define the development goals, functional scope, technical requirements, and acceptance criteria of the "Campus Mutual Assistance Platform". It provides a clear development basis for the R&D team and ensures the product meets core user needs in scenarios such as second-hand transactions, errand services, purchase requests, and carpooling. This document also serves as a reference for testing, operation and maintenance, and subsequent iterative optimization to ensure a standardized and efficient product development process.

1.2 Project Background

Currently, there are pain points on campus including scattered second-hand transaction channels, difficult connection of temporary services, low efficiency of travel carpooling, and ambiguous transaction addresses. Users lack a unified and secure mutual assistance service platform. Based on this, the Overload team plans to develop the "Campus Mutual Assistance Platform" to integrate multiple mutual assistance functions, break information barriers, build a trusted campus service ecosystem, and subsequently expand to community scenarios.
Relevant project information is as follows:

- Software Name: Campus Mutual Assistance Platform
- Project Initiator: MIEC EE308FZ Course Group
- Project Developer: Overload Team
- Target Users: College students initially, faculty and staff in the mid-term, and community residents in the long-term

1.3 Reference Documents

- National Standard Text of "Software Requirements Specification"
- "The Art of Construction"
- Campus User Demand Survey Report

- Product Presentation Document
- MIEC EE308FZ Course Assignment 3 Requirements Description

2. Overall Description

2.1 Goals

2.1.1 Development Intention

By integrating four core functions (second-hand transactions, errand running, purchase requests, and carpooling), build an easy-to-use, secure, reliable, and efficient campus mutual assistance platform. Solve the problems of poor circulation of campus resources and inefficient service connection, and improve the convenience of users' campus life.

2.1.2 Application Goals and Scope

- Core Goal: Achieve one-stop connection of mutual assistance services on campus, ensuring transaction security and information transparency.
- Application Scope: Cover all students of the university initially, supporting PC and H5 access; expand to 60% of the university's students and faculty in the mid-term; extend to students of other universities and community residents in the long-term.

2.1.3 Product Prospects

Become a high-frequency mutual assistance tool on campus in the short term, form a standardized campus mutual assistance service model in the mid-term, and expand into a cross-scenario community mutual assistance platform in the long term, with sustainable functional iteration and scenario extension capabilities.

2.2 Operating Environment

2.2.1 Hardware Environment

- Server: CPU ≥ 8 cores, memory ≥ 16GB, hard disk ≥ 500GB, bandwidth ≥ 10Mbps
- Client: PC devices supporting Windows 10 and above, macOS 12 and above; mobile devices supporting Android 10.0 and above, iOS 14 and above

2.2.2 Software Environment

- Backend: Java/Python programming language, MySQL database, Redis cache, Spring Boot/Spring Cloud framework
- Frontend: Vue.js/React framework, Element UI component library, responsive layout adaptation
- Testing Environment: Junit unit testing framework, Postman interface testing tool
- Deployment Environment: Docker containerized deployment, Nginx reverse proxy

3. Specific Requirements

3.1 Class Diagram

3.1.1 Core Entity Classes

- User Class: Includes attributes such as user ID, student ID/ID card number, name, phone number, password (encrypted storage), role (ordinary user/admin), authentication status, campus/community affiliation, and credit score; provides methods such as registration, login, authentication, and information modification.
- Commodity Class: Includes attributes such as commodity ID, publisher ID, commodity name, category (learning materials/idle items, etc.), description, price, images, transaction method, location, release time, and status (pending transaction/completed/removed); provides methods such as release, edit, remove, and query.
- Order Class: Includes attributes such as order ID, order type (second-hand transaction/errand/purchase request/carpool), initiator ID, executor ID, commodity/service information, price, address, status (pending acceptance/accepted/in progress/completed/canceled), creation time, and completion time; provides methods such as creation, status modification, query, and evaluation.
- Carpool Class: Includes attributes such as carpool ID, publisher ID, departure location, destination, departure time, vehicle type, available seats, registered number of people, contact information (encrypted display), fee, and remarks; provides methods such as release, registration, cancellation of registration, and query.

- Evaluation Class: Includes attributes such as evaluation ID, order ID, evaluator ID, evaluated person ID, score (1-5 points), evaluation content, evaluation time, and images; provides methods such as release and query.

3.1.2 Association Relationships

- User and Commodity: One-to-many relationship (one user can publish multiple commodities)
- User and Order: One-to-many relationship (one user can initiate/execute multiple orders)
- Commodity and Order: One-to-one relationship (one commodity corresponds to one transaction order)
- User and Carpool: One-to-many relationship (one user can publish/register for multiple carpool information)
- Order and Evaluation: One-to-one relationship (one order corresponds to one evaluation)

3.2 Feature Requirements

3.2.1 Usability

- The interface design is simple and intuitive, with core function entrances accessible within 3 steps, and new user onboarding time ≤ 5 minutes.
- Supports fast retrieval functions such as fuzzy query and category filtering, with retrieval response time ≤ 2 seconds.
- Has fault tolerance. When users enter incorrect formats, clear prompts are provided to support re-entry; the current operation can be restored after reconnection following network interruption.
- Page loading time ≤ 3 seconds, interface response time ≤ 1 second, system availability ≥ 99.9%

3.2.2 Security

- User passwords are stored using MD5 + salt encryption, and sensitive information (phone numbers, contact information) is desensitized for display.

- Implements campus identity authentication (student ID + campus email verification) and community identity authentication (ID card number + address verification); unauthenticated users have limited access to some functions.
- Order payment adopts a third-party sandbox environment (Alipay/WeChat), and no real payment information is stored.
- Supports account abnormal login reminders and password retrieval (SMS/email verification) functions.
- Prevents common security vulnerabilities such as SQL injection, XSS cross-site scripting, and CSRF cross-site request forgery.
- User behavior logs are retained for 90 days to support traceability of abnormal operations.

3.2.3 Scalability

- The system architecture adopts microservice design to support independent upgrade and expansion of functional modules.
- The database design reserves fields to support subsequent addition of attributes (such as commodity tags, service types).
- Interface design follows RESTful specifications and versioned management to ensure compatibility.

3.3 Functional Requirements

3.3.1 User Module

1. Registration and Login: Supports registration with phone number + verification code, binding of student ID/ID card number, password login; supports third-party (WeChat/QQ) quick login.
2. Identity Authentication: Campus users upload student ID and campus card photos for authentication; community users upload ID card number and address information for authentication; administrators review authentication information.
3. Personal Center: Supports viewing/editing personal information, modifying passwords, managing published/collected/order records, and viewing credit scores and evaluations.

3.3.2 Second-hand Transaction Module

1. Commodity Release: Supports uploading multiple images, filling in detailed descriptions, selecting categories and transaction methods (self-pickup/home delivery), and setting transaction addresses and prices.
2. Commodity Browsing: Supports filtering by category, price range, release time, and distance; supports keyword search and sorting (popularity/price/latest).
3. Transaction Process: Buyers place orders to lock commodities, sellers confirm orders, both parties communicate transaction details, buyers confirm receipt after transaction completion, and transaction evaluation is supported.

3.3.3 Errand and Purchase Request Module

1. Demand Release: Users publish errand (package pickup/meal purchase, etc.) or purchase requests, filling in addresses, item descriptions, time constraints, and payment amounts.
2. Order Matching: Executors can browse demand lists and take orders actively; initiators can specify specific executors to take orders.
3. Service Tracking: Supports real-time viewing of order status, executors upload service progress (such as successful pickup/delivery), and initiators can communicate in real time.

3.3.4 Carpool Module

1. Carpool Release: Supports filling in departure location, destination, departure time, available seats, fee, contact information, and other information.
2. Carpool Registration: Users can browse carpool information and submit registration applications; publishers review applications and provide contact information after approval.
3. Carpool Management: Publishers can modify carpool information, confirm registrations, and cancel carpools; registrants can cancel registrations.

3.3.5 Evaluation and Credit Module

1. After order completion, both parties can score (1-5 points) and provide text evaluations for transactions/services, supporting image uploads.

2. The system calculates user credit scores based on evaluations, serving as a reference for transaction/service connections.
3. Supports evaluation management; users can view their own evaluation records, and administrators can handle malicious evaluations.

3.3.6 Administrator Module

1. User Management: Views user lists, reviews identity authentication, and handles violating users (account suspension/function restriction).
2. Content Management: Reviews commodities/carpool information, deletes violating content, and handles user complaints.
3. Data Statistics: Views core data such as platform activity, function usage frequency, and transaction order volume.

4. Interface Prototype

4.1 Core Page Layout

4.1.1 Homepage

- Top: Search box (supports full-platform content retrieval), user avatar/login entrance, message notification icon.
- Middle: Function navigation area (four function entrances for second-hand transactions, errands, purchase requests, and carpooling with icons and brief descriptions), recommendations for popular commodities/services, and display of latest carpool information.
- Bottom: Navigation bar (Homepage, Category, Publish, Messages, Mine), About Us, customer service entrance, and privacy policy link.

4.1.2 Commodity Release Page

- Includes commodity category selection, name input box, price input box, detailed description text field, image upload area (up to 5 images), transaction method selection (self-pickup/home delivery), address selector, and publish/save draft buttons.

4.1.3 Order Detail Page

- Displays order type, commodity/service information, desensitized user information of both parties, price, address, order status, and creation time; provides operation buttons such as contacting the other party, modifying order status, and evaluation.

4.1.4 Carpool Release Page

- Includes departure location/destination input boxes (supporting address suggestions), departure time selector, vehicle type input box, available seats selector, fee input box, contact information input box (encryption prompt), remarks text field, and publish button.

4.2 Interaction Requirements

- Frontend verification is performed before all form submissions, with error prompts clearly marked below input boxes.
- Loading status feedback (such as grayed-out buttons, loading animations) is provided after clicking function buttons.
- Secondary confirmation pop-ups are required for important operations (such as confirming transactions, canceling orders).
- Supports pull-down refresh and pull-up load more functions (commodity lists, order lists, carpool lists).

5. Function Description and Acceptance Verification Standards

5.1 Detailed Function Description

5.1.1 Second-hand Transaction Function

1. Commodity Release: After selecting a category, users fill in complete information, upload images, and submit. The system verifies information integrity, generates commodity records, and sets the status to "pending transaction".
2. Commodity Browsing: Users can query commodities through categories, filter conditions, and keywords. The list page displays commodity thumbnails, names,

prices, publishers, distances, and release times; clicking enters the detail page to view complete information.

3. Order Creation: Buyers click "Place Order" on the commodity detail page, confirm order information, and submit. The system generates an order, and the commodity status changes to "locked".
4. Order Processing: Sellers receive order notifications and can choose to "accept" or "reject" the order; after acceptance, the order status changes to "in progress", and both parties can communicate through the platform's built-in chat tool; after transaction completion, buyers click "Confirm Receipt", the order status changes to "completed", and the commodity status changes to "transaction completed".

5.1.2 Errand and Purchase Request Function

1. Demand Release: Users select the service type (errand/purchase request), fill in the address, item description, time constraints, and payment amount, and submit to generate a demand order with the status set to "pending acceptance".
2. Order Acceptance and Execution: Executors browse the demand list and click "Accept Order" to change the order status to "accepted"; after completing the service, executors upload vouchers (such as pickup photos, commodity photos), and the order status changes to "completed" after confirmation by the initiator.

5.1.3 Carpool Function

1. Carpool Release: Users fill in information such as departure location, destination, and departure time, and submit to generate carpool records with the status set to "recruiting".
2. Registration and Confirmation: Other users click "Register" and submit personal information. Publishers receive notifications and review; after approval, the registration is successful, and the "number of registered people" is updated; publishers can close recruitment, and the status changes to "recruitment ended".
3. Carpool Completion: After departure, publishers can mark "departed", and mark "completed" after the trip ends.

5.1.4 Evaluation Function

Within 72 hours after order completion, both parties can evaluate the transaction/service, enter scores and text content, and support image uploads; evaluations cannot be modified after submission, and the system automatically updates user credit scores.

5.2 Input and Output Formats

5.2.1 Input Formats

- Text Input: Commodity names, descriptions, addresses, etc., must meet character length requirements (name ≤ 50 characters, description ≤ 500 characters, address ≤ 100 characters), and illegal characters are prohibited.
- Image Input: Supports JPG and PNG formats, with a single image size ≤ 5MB and a maximum of 5 images allowed for upload.
- Numeric Input: Prices and fees must be positive numbers, retaining up to 2 decimal places; the number of people and time must comply with logical rules (such as available seats ≤ 5 people).

5.2.2 Output Formats

- List Output: Commodity, order, carpool and other list information are sorted by "latest release" or "popularity", displayed in pages, with 10 items per page by default.
- Detail Output: Fully displays all relevant attributes of the entity class, images are arranged in the upload order, and status is displayed with prominent labels (such as "pending acceptance" marked in blue, "completed" marked in green).
- Prompt Output: Operation results are fed back in the form of pop-ups and top notifications, and error prompts clearly explain the reasons and solutions.

5.3 Interface Acceptance Standards

1. All pages have a unified style, consistent fonts, colors, and spacing, with no layout disorders.
2. The responsive layout is compatible with PC and mobile terminals; there is no horizontal scroll bar on mobile terminals, and key function buttons are not blocked.
3. The form verification logic is complete, error prompts are accurate, and there are no missing verification scenarios.
4. Interactions such as page loading, button clicking, and data refreshing are free of lag and respond in a timely manner.

5.4 Functional Acceptance Standards

1. Core functions (registration, login, commodity release, order creation, order acceptance, evaluation) are 100% available, with no functional blocking issues.
2. Boundary scenarios (such as account locking after 3 incorrect password attempts, unprocessed orders exceeding the time limit, and image upload exceeding the limit) have correct processing logic.
3. The system is stable in concurrent scenarios (100 people browsing commodities simultaneously, 10 people registering for carpooling simultaneously), with no data confusion.
4. No high-risk vulnerabilities are found in security testing, and sensitive information encryption storage and desensitized display meet requirements.
5. Performance indicators meet standards (page loading ≤ 3 seconds, interface response ≤ 1 second, system availability ≥ 99.9%).

6. Acceptance Verification Standards

6.1 Document Acceptance

- The Software Requirements Specification complies with the national standard format, with a complete directory, clear and unambiguous content, covering all core requirements.
- Supporting documents (interface documents, test cases, deployment manuals) are complete, formatted standardized, and can be directly used for development, testing, and operation and maintenance.

6.2 Functional Acceptance

- Full-featured testing is performed according to test cases, with a function point pass rate ≥ 98%, no critical bugs, and ≤ 2 serious bugs that can be fixed before launch.
- Core processes (such as the full second-hand transaction process, full errand service process) have no breakpoints, and users can complete operations smoothly.

6.3 Performance Acceptance

- When the number of concurrent users ≥ 1000, the system responds normally without crashes or timeouts.
- Database query response time ≤ 500ms, file upload/download speed is stable, with no obvious delays.

6.4 Compatibility Acceptance

- Compatible with the latest versions of Chrome, Firefox, and Edge on PC terminals, and mainstream browsers and WeChat built-in browsers on mobile terminals.
- No layout disorders or functional abnormalities under different resolutions and system versions.

7. Other Instructions

7.1 Description of Uncompleted Content

1. The third-party payment function currently only supports sandbox environment docking; real payment qualifications need to be applied for later, and docking will be supplemented after launch.
2. The community identity authentication module currently only supports basic information review; offline verification functions will be added later.
3. The commodity recommendation algorithm currently adopts simple category recommendation; it will be optimized to personalized recommendation based on user behavior data later.

7.2 Subsequent Improvement Plan

1. Iteration Cycle: A 2-week iteration cycle is adopted to optimize functions and experience based on user feedback.
2. Improvement Directions: Add commodity auction functions, expand service categories (such as skill exchange), optimize recommendation algorithms, and strengthen community interaction functions.
3. Document Update: Subsequent improvement content will be synchronously updated to this Software Requirements Specification, and modification records will be noted in the blog of subsequent assignments.