

2CS673 Software Engineering
Scrumble Bug - Campus Exchange
Project Proposal and Planning



<u>Team Member</u>	<u>Role(s)</u>	<u>Signature</u>	<u>Date</u>
Yingtong Zhou	Team leader	<i>Yingtong Zhou</i>	Sep 25, 2024
HungHsu(Allen) Chen	Configuration Leader	<i>HungHsu(Allen) Chen</i>	<u>09/25/2024</u>
Yuanbin Man	Requirement Leader	<i>Yuanbin Man</i>	<u>09/20/2024</u>
Ang Li	Design and Implementation Leader	<i>Ang LI</i>	<u>09/20/2024</u>
Yueyihan Qi	Security Leader	<i>Yueyihan Qi</i>	<u>09/25/2024</u>
Srujana N	QA Leader	<i>Srujana N</i>	<u>09/26/2024</u>

Revision history

<u>Version</u>	<u>Author</u>	<u>Date</u>	<u>Change</u>
<u>1.1</u>	Hunghsu(Allen) Chen	<u>10/17/2024</u>	<u>Timeline update</u>

[Overview](#)

[Related Work](#)

[Proposed High level Requirements](#)

[Management Plan](#)

[Objectives and Priorities](#)

[Risk Management \(need to be updated constantly\)](#)

[Timeline \(need to be updated at the end of each iteration\)](#)

[Configuration Management Plan](#)

[Tools](#)

[Deployment Plan if applicable](#)

[Quality Assurance Plan](#)

[Metrics](#)

[Code Review Process](#)

[Testing](#)

[Defect Management](#)

[References](#)

[Glossary](#)

1. Overview - Yuanbin Man

Second-hand trading is gaining popularity among university students. This platform, tailored for BU students, enables buying and selling of used items like phones, computers, and furniture, etc. It aims to offer a convenient way to trade goods within the university, promoting sustainability and benefiting the community. Our goal is to foster an eco-friendly, green, and inclusive environment. Technically, we use the latest tech stack. Since this is a B/S system, we use Java with Spring Boot and MySQL for the backend, and Node.js with Vue for the frontend. For CI/CD, we rely on GitHub Actions, and the system is deployed on the cloud using Docker and Kubernetes.

2. Related Work - Yuanbin Man

A few well-known examples include *Facebook Marketplace*, *Craigslist*, and *OfferUp*. These platforms allow users to buy and sell items within local communities. While they serve a broad audience, they do not focus on specific groups like university students.

Facebook Marketplace: (1) Target Audience: Open to the general public, allowing people to

buy and sell within their local areas.(2) Key Features: Offers item listings with images, location-based browsing, and integrated messaging. It also allows sellers to market to a larger audience.(3) Differences: Our platform is exclusive to BU students, fostering a sense of trust and community. Additionally, we emphasize sustainability by promoting the reuse of items within the university.

Craigslist: (1) Target Audience: Open to anyone, with categories ranging from jobs to housing to items for sale. (2) Key Features: A minimalistic user interface with text-based listings, location-based search, and simple buyer-seller communication. (3) Differences: Craigslist has a broad audience, but lacks the tailored, university-specific focus and a more modern, user-friendly interface. Our platform is specifically designed for BU students, with features like verified student accounts to enhance trust.

OfferUp: (1) Target Audience: General public, with a focus on local transactions. (2) Key Features: Mobile-friendly app, location-based item search, and built-in messaging for easy buyer-seller communication. (3) Differences: OfferUp is more polished but lacks the exclusive community focus. Our platform provides a safer and more trusted environment as all users are verified BU students.

Overall, while these platforms cater to a wider audience, our system's advantage lies in creating a secure, eco-conscious, and community-driven marketplace that addresses the specific needs of university students.

3. Proposed High level Requirements - Yuanbin Man

a. Functional Requirements

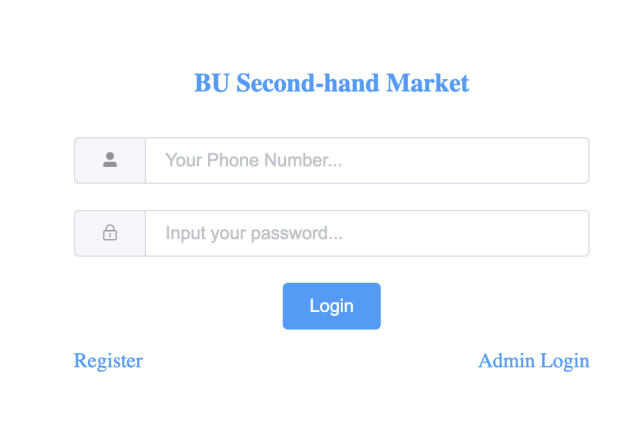
(For each functional requirement, please give a feature title and a brief description using the following format: As (a role), I want to (action), so that (value).)

i. Essential Features (the core features that you definitely need to finish):

(For each essential features, please give a rough estimation in terms of person hours or an range of person hours)

- 1) **Login Page:** As a student role, Login page with phone number and password: Includes registration and admin login functionality.

- Estimate time: 3h



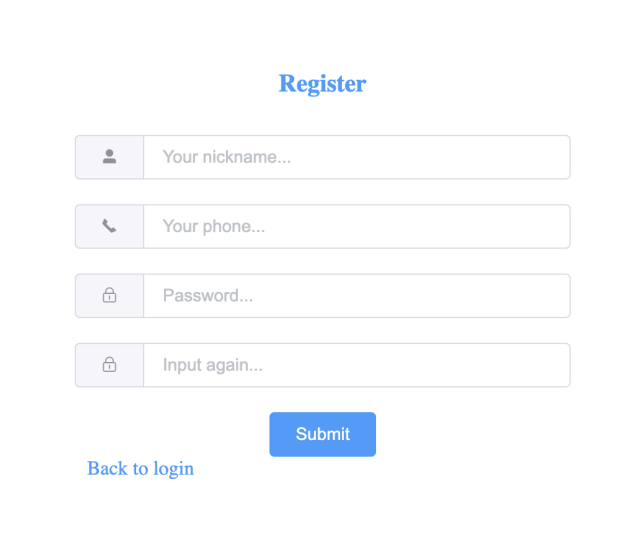
BU Second-hand Market

[Login](#)

[Register](#)
[Admin Login](#)

- 2) Register Page: Register your account with a nickname, phone number, and password. Users should be prompted to re-enter the password for confirmation before submitting the registration.

- Estimate time: 3h



Register


[Submit](#)


[Back to login](#)

- 3) Administration Login: An admin role is required. The login page should support login with phone number and password, and include both user registration and admin login functionality.

- Estimate time: 3h

Administration Managemet

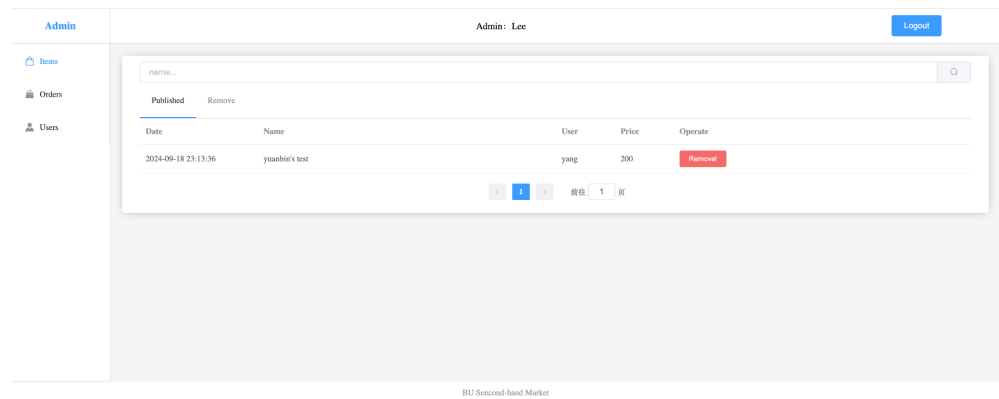




Login

[Student Login](#)

- 4) Admin page: The admin page should feature an item management subpage that displays both online and offline items with details like name, time, user, and an option to take items offline. It should also include order management, listing all market orders with full order details. Additionally, user management should list all platform users, with an option to ban users.



- 5) Index Page: The BU Second-Hand Market index page should include a search bar for users to search items by name, description, and more. The main section will display a list of items with pictures, name, description, price, location, and category. Additionally, it should feature a 'Publish Item' button, an announcement section, and a message center where users can view all messages from buyers or sellers.
- 6) Publish Item: The publish item page should include fields for title, description, category, location, price, and a picture upload option, followed by a submit button.

Publish Your Item or Announcement


0/30

0/1000

Your Location

Category

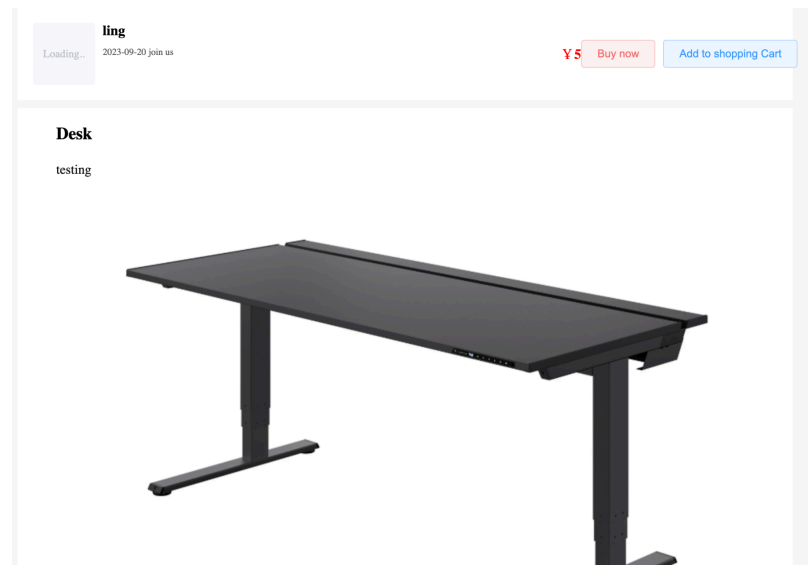
upload picture




Drag the pic here, or [Click and Upload](#)

[Publish](#)

- 7) Item Details: Display all the item's information, including the picture, seller, title, and description, along with 'Buy Now' and 'Add to Cart' buttons. Users can also leave a message for the seller.



- 8) Buy and Pay: The buy and pay page appears after a user selects an item. It displays the item's details, allows the user to choose their address, and provides payment options. The user can either cancel the purchase or proceed with payment.



Buy: iPhone 15 Pro ¥10

Receiver Address:

[Choose your address](#)

Order info (unPaid) :

Id: 172685111802410003

PayStatus: unPaid

Payment:

Create Time: 2024-09-21 00:51:58

Pay Time:

[Cancel Order](#)
[Pay Now](#)

ii. Desirable Features (the nice features that you really want to have too):

- 1) Scrolling Recommendation: The platform collects user clicks, messages, page views (PV), and unique visits (UV) to develop a recommendation algorithm that suggests items best suited to each user.
- 2) Second-hand Assistance: Utilizing the latest LLM technology, such as LLaMA, to offer an interactive chat feature that helps users quickly find items they are most interested in.

iii. Optional Features (additional cool features that you want to have if there is time):

TBD

iv. Existing Features (delete this item if your project starts from scratch)

b. Nonfunctional Requirements

i. Security requirements

4. Management Plan - Yihan

a. Objectives and Priorities

1) **Complete Core Function Development**

- **Login and Registration Features:**

As a user, you will be able to register and log in using your mobile number and password to participate in platform transactions. The login and registration process will be smooth, supporting both administrator and regular user roles.

- **Item Posting and Management Features:**

Users can post second-hand items, including the item name, description, price, and images. They can also browse and search through the item listings.

- **Admin Management Features:**

As an administrator, you will be able to log in through the backend management system and manage user information and product listings on the platform, including banning users and taking down non-compliant products.

- **Transaction and Payment Features:**

Users can select items to purchase, fill in the delivery address, and complete the payment process.

2) **Ensure System Security and User Verification**

The system should ensure the security of user and administrator information. Administrators will have higher-level permissions to manage users and items. All users must pass identity verification to ensure trust and safety within the community.

3) **Conduct Comprehensive Testing to Ensure System Stability**

- **Unit Testing:**

Each functional module will undergo unit testing to ensure the independent validity of each module.

- **Integration Testing:**

Integration testing will ensure that all functional modules work together seamlessly.

- **User Acceptance Testing:**

Invite real users to test the system, ensuring it meets user requirements and has no major bugs.

- **Improve Code Quality:**

Conduct regular code reviews to ensure code consistency and maintainability.

4) **Successfully Deploy the Platform**

The platform will be deployed in the cloud using Docker and Kubernetes, ensuring users and administrators can access and use the system at any time. The deployment will include automated CI/CD processes implemented through GitHub Actions.

5) **Optimize User and Administrator Interface Experience**

Ensure both users and administrators can smoothly use the platform's

features. The frontend will be developed using Vue.js, providing a modern, user-friendly interface with responsive design that adapts to various devices.

6) Implement Recommendation Algorithm (Optional)

Develop a product recommendation algorithm based on user behavior data (such as clicks, views, comments, etc.), improving the convenience of user transactions and increasing platform engagement.

7) Second-Hand Item Recommendation Assistant (Optional)

Utilize the latest language models (such as LLaMA) to provide a chat assistant feature, helping users quickly find items they are interested in, further optimizing the user experience.

8) Document the Project

Provide developer documentation, user guides, and API documentation to ensure the ease of future maintenance and extension.

9) Continuously Monitor and Optimize the System

Integrate continuous user feedback by regularly collecting feedback, evaluating it, and iterating to optimize system functionality quickly.

b. Risk Management (need to be updated constantly)

We have identified several key risks and formulated corresponding management strategies. In terms of **technical risks**, using the latest technology stack (such as Docker, Kubernetes, and Spring Boot) may introduce integration and configuration challenges. The team will conduct small-scale testing in advance and prepare alternative solutions. The **time risk** lies in the tight project development schedule, so core functionalities will be prioritized, while the priority of secondary features will be adjusted as needed. **Security risks** involve the protection of user personal information and transaction data. We will adopt encryption and authentication measures and conduct regular security testing. **Quality risks** will be managed through unit testing, integration testing, and user acceptance testing. **Operational risks** include performance bottlenecks and user feedback after system deployment, which we will continuously monitor and promptly optimize. Additionally, we have identified the risks of **personnel turnover** and **requirement changes**. The team will establish contingency plans, evaluate the impact of changes on the project, and ensure smooth project progress. The team will hold regular meetings to check progress, ensure each member is clear on their responsibilities, and resolve any issues quickly.

Risk Management Sheet Link: [📄 CS673_SPPP_RiskManagement](#)

c. Timeline (this section should be filled in iteration 0 and updated at the end of each later iteration) - Ang Li

Iteration	Functional Requirements(Essential/Dismissible/Option)	Tasks (Cross requirements tasks)	Estimated/real person hours
0	At this stage, we need to complete the preparatory work for the early phase of the project. This includes preliminary research, technical selection, defining specific project code functionalities, and creating the project architecture diagram. Additionally, everyone needs to confirm their local environment (including development tools and local Git repositories). All of these requirements are necessary for this phase	<ol style="list-style-type: none"> 1. Configure the local environment 2. Confirm the project type 3. Confirm the technical direction 4. Define the technical choices 5. Complete the basic architecture design 6. Define specific development goals and functionalities 	30
1	In phase 1, we need to complete the development of all specific project functionalities. Additionally, in the early stage of phase 1, we need to set up the testing environment and the backend test database. By the end of the second stage, all functionalities need to be fully developed and debugged.	<ol style="list-style-type: none"> 1. Registration 2. Login 3. Main interfaces 4. Remove 5. Personal info. 6. Comments 7. Conduct Testing environment 	80

2	<p>At this stage, we not only need to complete the bug fixes for the project but also ensure the smooth operation of all CICD processes and the overall architecture development. This includes the setup of the database, cache, servers, and container orchestration components. During this phase, we need to complete the construction of the overall project architecture and resolve any issues that may arise during the development process</p>	<ol style="list-style-type: none"> 1. Item posting 2. Message modules 3. Item purchases 4. Booking marks 5. Product management 6. Order management 7. Bug fix 8. CI/CD process 9. Databases 10. Cache layer 11. Computing servers 12. Docker & kubernetes 13. Troubleshooting 	80
3	<p>At this stage, we need to complete the development of advanced project features while ensuring the stability of the front-end, back-end, and server architecture. We also need to implement cloud monitoring to keep track of the overall project status, establish effective alert mechanisms that have been tested, and build effective high-availability solutions. Additionally, we need to optimize the architecture, such as tuning the JVM (if necessary), SQL performance analysis and optimization (if necessary), and explore ways to prevent and avoid cache breakdown and cache avalanche issues</p>	<ol style="list-style-type: none"> 1. to ensure the project stanley running on the cloud 2. Build a monitoring system. 3. Implement the alerting feature. 4. Server high availability architecture implementation 5. Databases high availability architecture implementation 6. JVM Optimization(if required) 7. SQL Optimization((if required) 8. Cache layer Optimization(if required) 	60-100

5. Configuration Management Plan - Allen

a. Tools

Version control: GitHub

IDE tools: VS code + idea

CI/CD tools: Github Action

SAST/DAST tools: GitHub Action / OWASP

Task management tools: Jira

Container tools: Docker, Kubernetes

Cloud Computing: Alibaba Cloud

b. Code Commit Guideline and Git Branching Strategy

Code Commit Guideline: Requiring pull request with at least 2 reviewers to merge branch to default branch. Will follow the coding review process and coding guidelines.

Git Branching Strategy: Github Flow - use Github Flow branching strategy as the team size is small, each member only needs to take care of their own branch and maintain the main branch in a deployable state. New branch will be created for implementing new features. Committing to a branch other than the default branch doesn't require reviewing.

c. Deployment Plan if applicable

We plan to deploy the project using a cloud service with Kubernetes. We will create a cloud container to host the Kubernetes cluster, and management will be through the Google Kubernetes Engine or Amazon EKS. We may deploy a Multi-node Kubernetes for better performance, flexibility, and scalability.

6. Quality Assurance Plan - Srujana

a. Metrics

(Describe the metrics to be used in the project to measure the quality of your software. Each metric should be measurable and quantifiable. Examples of metrics include product complexity (LOC, # of files, # of classes, # methods, cyclomatic complexity, etc.) , defect rate (# of defect per KLOC), # of test cases, test case pass rate, cost (# of person hours used), # of user stories completed, etc. **The result of these metrics should be reported in the progress report/ iteration summary sheet.**)

Metric Name	Description
-------------	-------------

Process Metric: Effort (in person hours)	Cost of person hours used (personal)
In-Process Metric: Defect Repair rate	The rate at which the defects are fixed as soon as they are found out to minimize loss
Product Metric: Size & Complexity	The number of lines of code and different tools integration along with the number of files can show product size and complexity.
Product Metric: Defect Density	the defects according to size of software in terms of lines of code or function point, etc. i.e., it measures code quality per unit. It is perfect for our second-hand trading market e-commerce website
In-Process Metric: Defect arrival pattern	The overall defect density during testing provides only a summary of the defects, while the pattern of defect arrivals offers deeper insights into varying levels of quality in the field.

b. Coding Standard

(Describe any coding standard to be used)

1. Formatting: Braces are used with if, else, for, do and while statements, even when the body is empty or contains only a single statement. We also have Block indentation: +2 spaces.

2. Naming:

Class names - Class names are written in UpperCamelCase.

Method names - Method names are written in lowerCamelCase.

Constant names -Constant names use UPPER_SNAKE_CASE

3. Comments: All code should include comments on key code lines to make it easier to read, understand, and modify.

4. Functions: Code should utilize new or separate functions to define any logic that diverges from the primary purpose of a given function. This enhances code reusability and proves beneficial in the long run for continuous integration and development.

5. File structure: Controller, Service, Repository, DAO, Model etc. packages should be defined for project structure and appropriate files should be present within them with proper annotations.

c. Code Review Process

(Everyone should review all documents to be submitted. Here you will mainly describe how the code review will be done. Who will review the code, e.g. design or implementation leader will review all code or team members review each other's code. Do you use pull requests for the code review? Is there a checklist to help review? What feedback should the reviewer provide?)

We follow a feature branch workflow for all changes. During development, we maintain open communication and iterative reviews to ensure everyone is aligned on progress. Once changes are complete, a configuration leader initiates a pull request, requiring at least two team members (including the team leader) to review and approve. Additionally, we use a GitHub Issues Kanban board ('Team 2: Scrumble Bug Kanban') to track and review work items collaboratively.

Reviewers should provide feedback on the following:

- Function simplicity: Suggest breaking down complex functions into smaller, more focused ones.
- Code reuse: Encourage the use of existing code to minimize redundancy.
- Comment clarity: Emphasize the importance of clear comments to explain non-obvious logic.
- Meaningful naming: Recommend using descriptive variable and function names instead of generic ones like 'x' or 'y'.

d. Testing

(Both manual testing and automated testing should be considered. Both unit testing and integration testing should be considered. Briefly describe the testing tools/framework to be used, the personnel involved (e.g. the QA leader will focus on the integration testing and each developer will unit test their own code), when and what types of testing will be performed, the testing objectives, etc)

1. Individually, everyone can verify their logic on their own by manually entering values for some variables and verifying the result.
2. For Unit testing we can use JUnit, Mockito, Mockaroo
3. QA Leader will use the same frameworks for automation & integration testing along with Postman to generate automatic API testing values

4. Testing objective: Test basic flow and then cover all edge cases, maintain an excel sheet to record input and output of APIs, make sure all test cases pass before completion of project, can also include database testing

e. Defect Management

(Describe the tool to be used to manage the defect (e.g github issues). The types of defects to look at. The actions or personnel for defect management.)

1. We will use github issues as a tool for defect management.
2. All team members are encouraged to participate in bug testing throughout the project. The Q&A leader will oversee this process.
3. Developers responsible for creating defective code will be directly accountable for resolving the issues. The implementation leader will provide guidance and support to all team members involved in addressing defects.

7. References

(For more details, please refer to the encounter example in the book or the software version of the documents posted on blackboard.)

<https://google.github.io/styleguide/javaguide.html>

8. Glossary

B/S system: an architecture where clients use a web browser to interact with server-hosted applications. It simplifies the client-side by only needing a browser, making it accessible, platform-independent, and easy to maintain, unlike traditional client-server models that require dedicated client software.

CI/CD: automates the software development lifecycle by integrating code frequently (CI) and delivering it to production quickly and reliably (CD).