**CS673 Software Engineering** 

**Scrumble Bug - Campus Exchange**

**Project Proposal and Planning**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Yingtong Zhou | Team leader | [Yingtong Zhou](mailto:cutejiu@bu.edu) | Sep 25, 2024 |
| HungHsu(Allen) Chen | Configuration Leader | *HungHsu(Allen) Chen* | 09/25/2024 |
| Yuanbin Man | Requirement Leader | *Yuanbin Man* | 09/20/2024 |
| Ang Li | Design and Implementation Leader | *Ang LI* | 09/20/2024 |
| Yueyihan Qi | Security Leader | [Yueyihan Qi](mailto:qyyh@bu.edu) | 09/25/2024 |
| Srujana N | QA Leader | *Srujana N* | 09/26/2024 |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[Overview](#_g6igqliy7rm)

[Related Work](#_bf21eadgjj29)

[Proposed High level Requirements](#_rgyo4hi9stmq)

[Management Plan](#_ts358bsdtbcv)

[Objectives and Priorities](#_nxeeppkjxgn4)

[Risk Management (need to be updated constantly)](#_tk7yixobah8p)

[Timeline (need to be updated at the end of each iteration)](#_iksrndohvx29)

[Configuration Management Plan](#_j5uvivmxqcsp)

[Tools](#_dzly5b9kz982)

[Deployment Plan if applicable](#_sd8zu6r3jisd)

[Quality Assurance Plan](#_vra5ptwu59qx)

[Metrics](#_vwjduhc9wuah)

[Code Review Process](#_hx3eaiwb8v3m)

[Testing](#_l9xnpmd6hh0y)

[Defect Management](#_5amsh8h9f0c7)

[References](#_pd9euov6m4du)

[Glossary](#_ty3i2nqffhtc)

# Overview - [Yuanbin Man](mailto:ybinman@bu.edu)

*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.*

# Related Work - [Yuanbin Man](mailto:ybinman@bu.edu)

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.

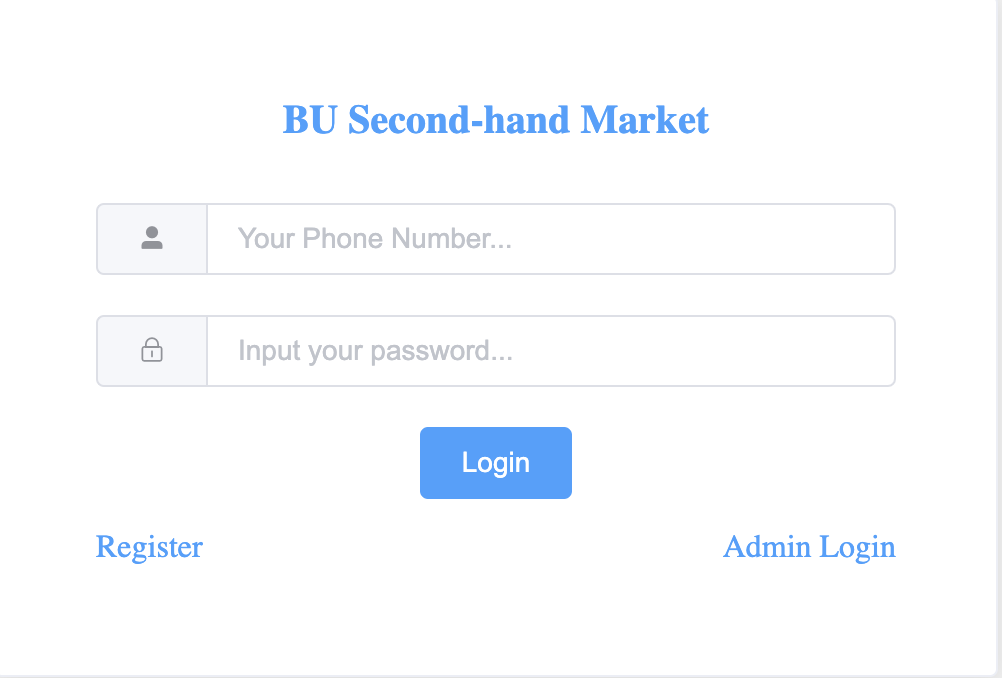
# Proposed High level Requirements - [Yuanbin Man](mailto:ybinman@bu.edu)

* 1. 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).)
     1. 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)

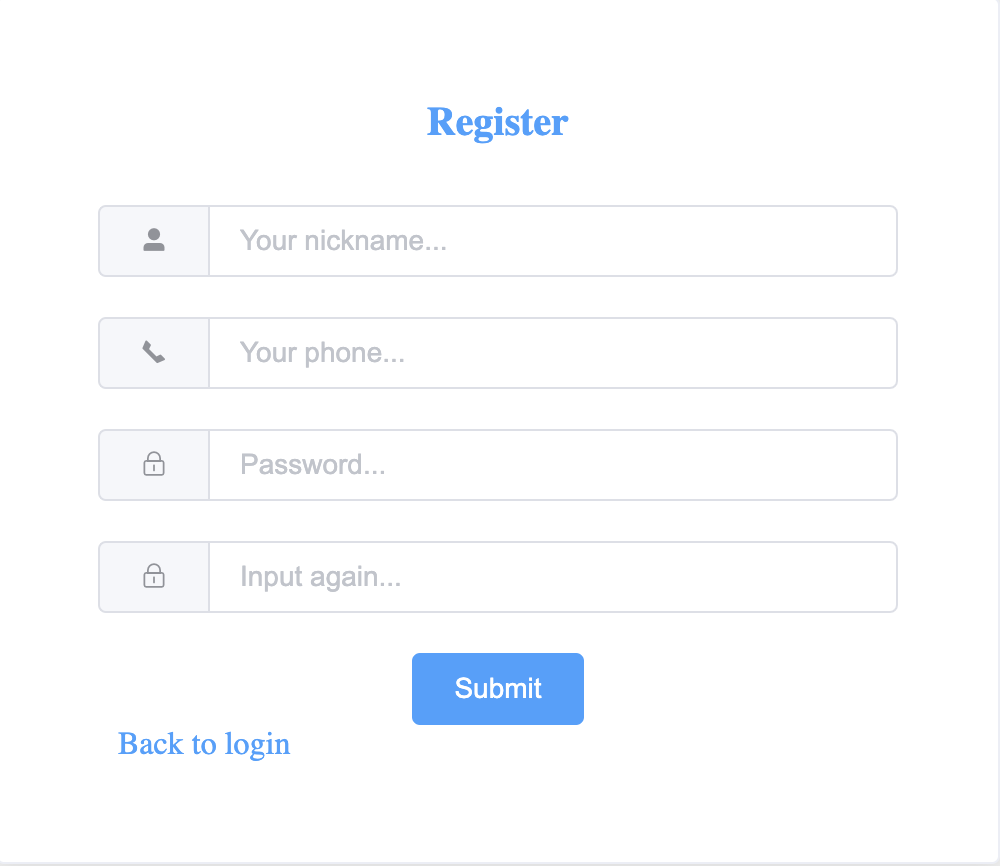
1. ***Login Page***: As a student role, Login page with phone number and password: Includes registration and admin login functionality.

* Estimate time: 3h



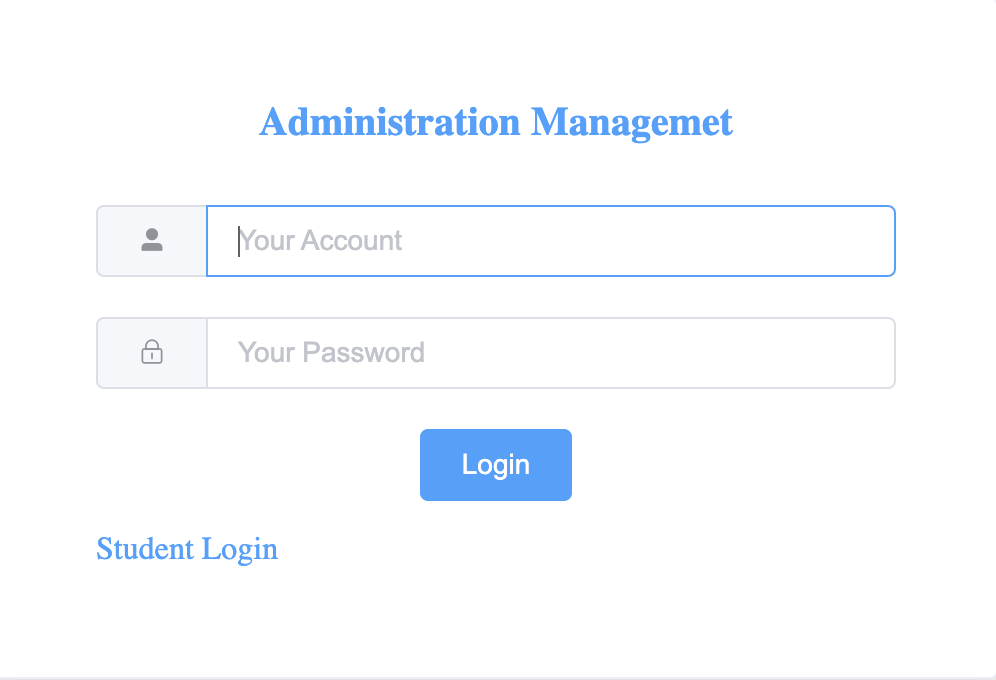
1. Register Page: Register your account with a nickname, phone number, and password. Users should be prompted to re-enter the password for confirmation before submitting the registration.

* Estimate time: 3h

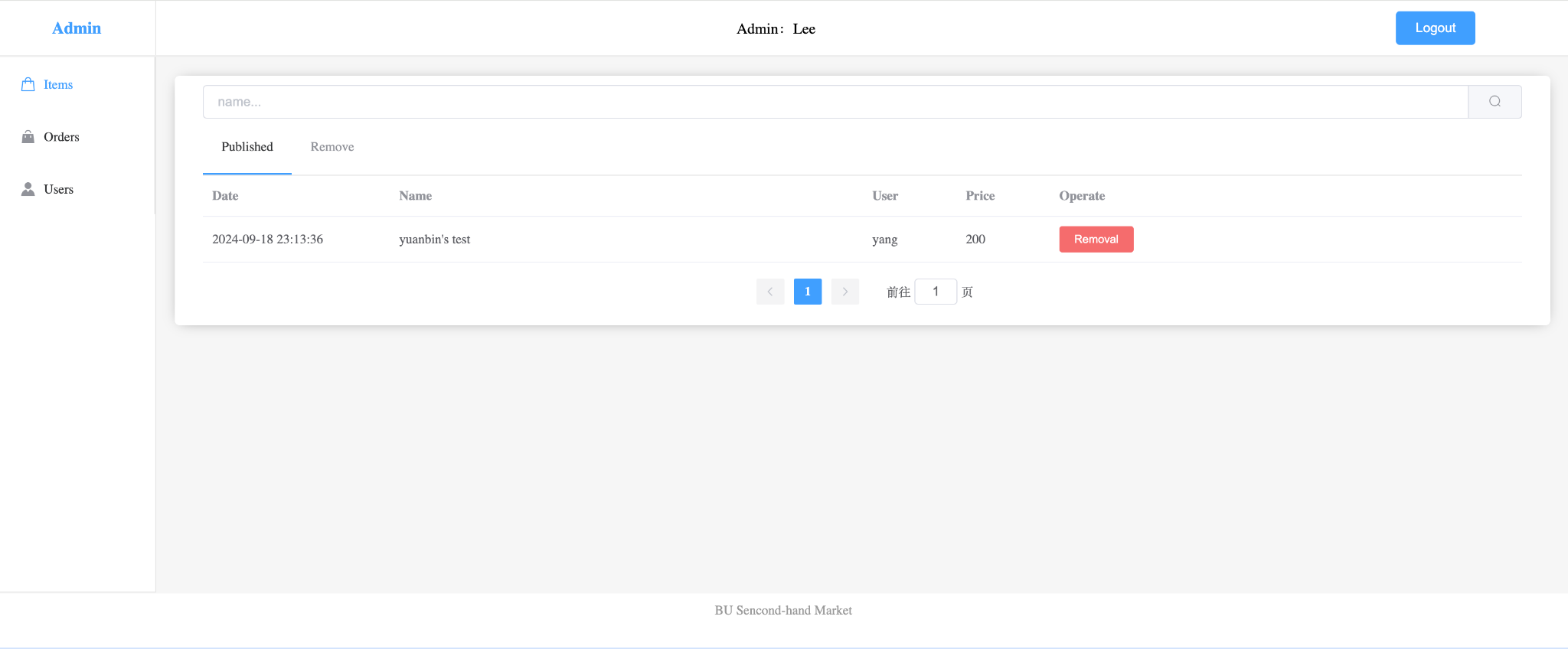


1. Administration Login: An admin role is required. The login page should support login with phone number and password, and include both user registration and admin login functionality.

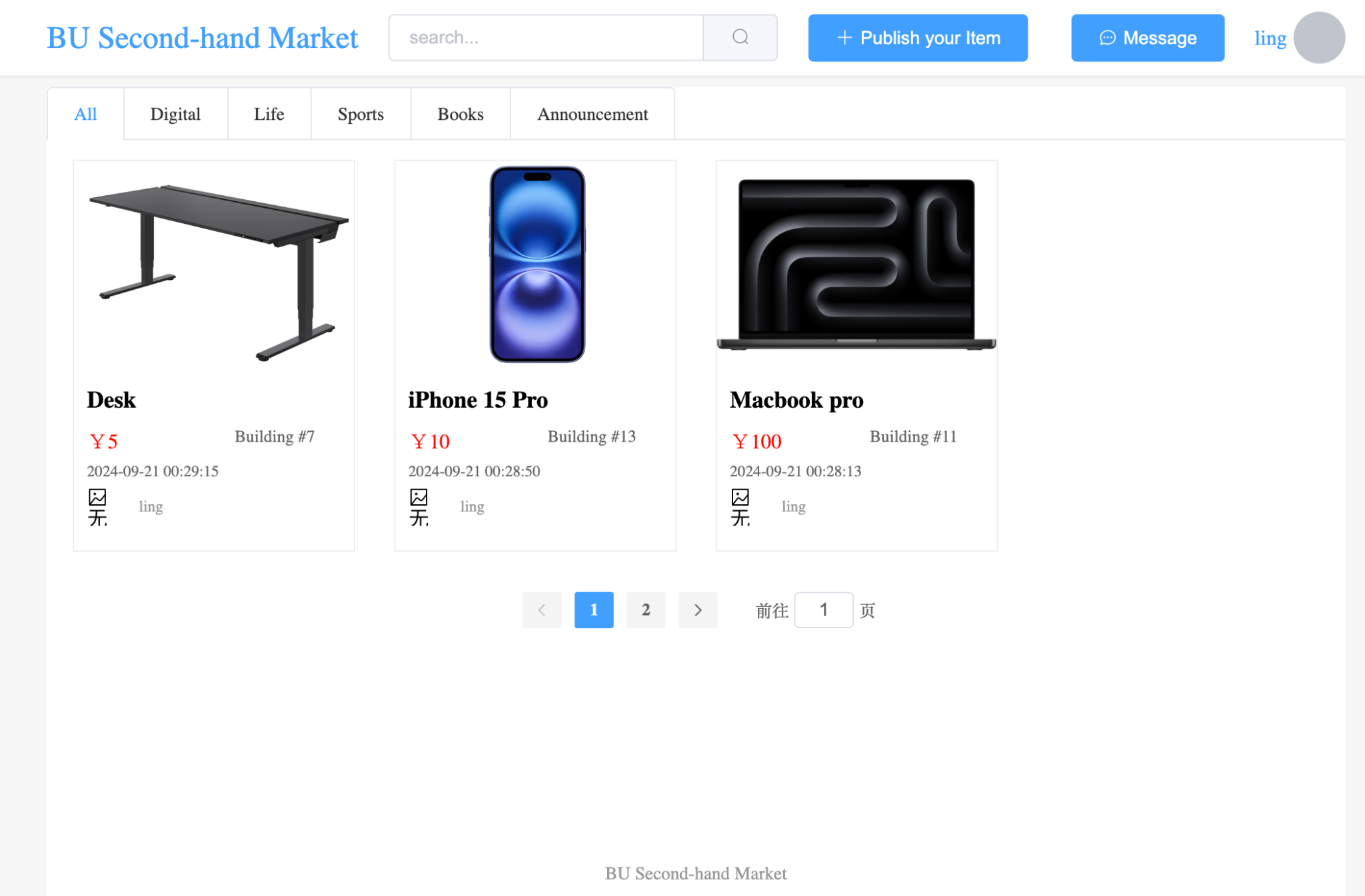
* Estimate time: 3h



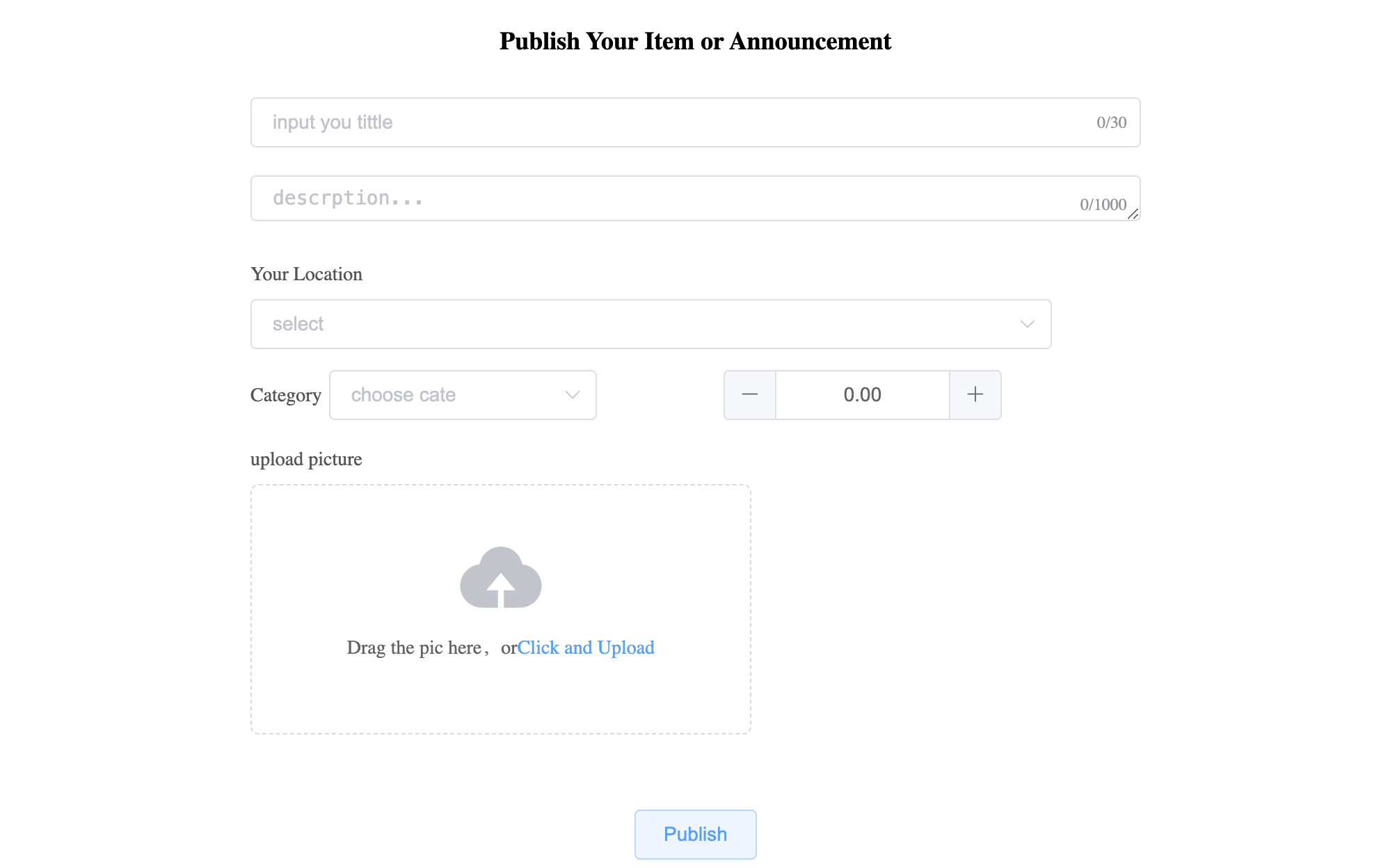
1. Admin page: The admin page should feature an item management subpage that displays both online and offline items with details like name, time, user, and an option to take items offline. It should also include order management, listing all market orders with full order details. Additionally, user management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



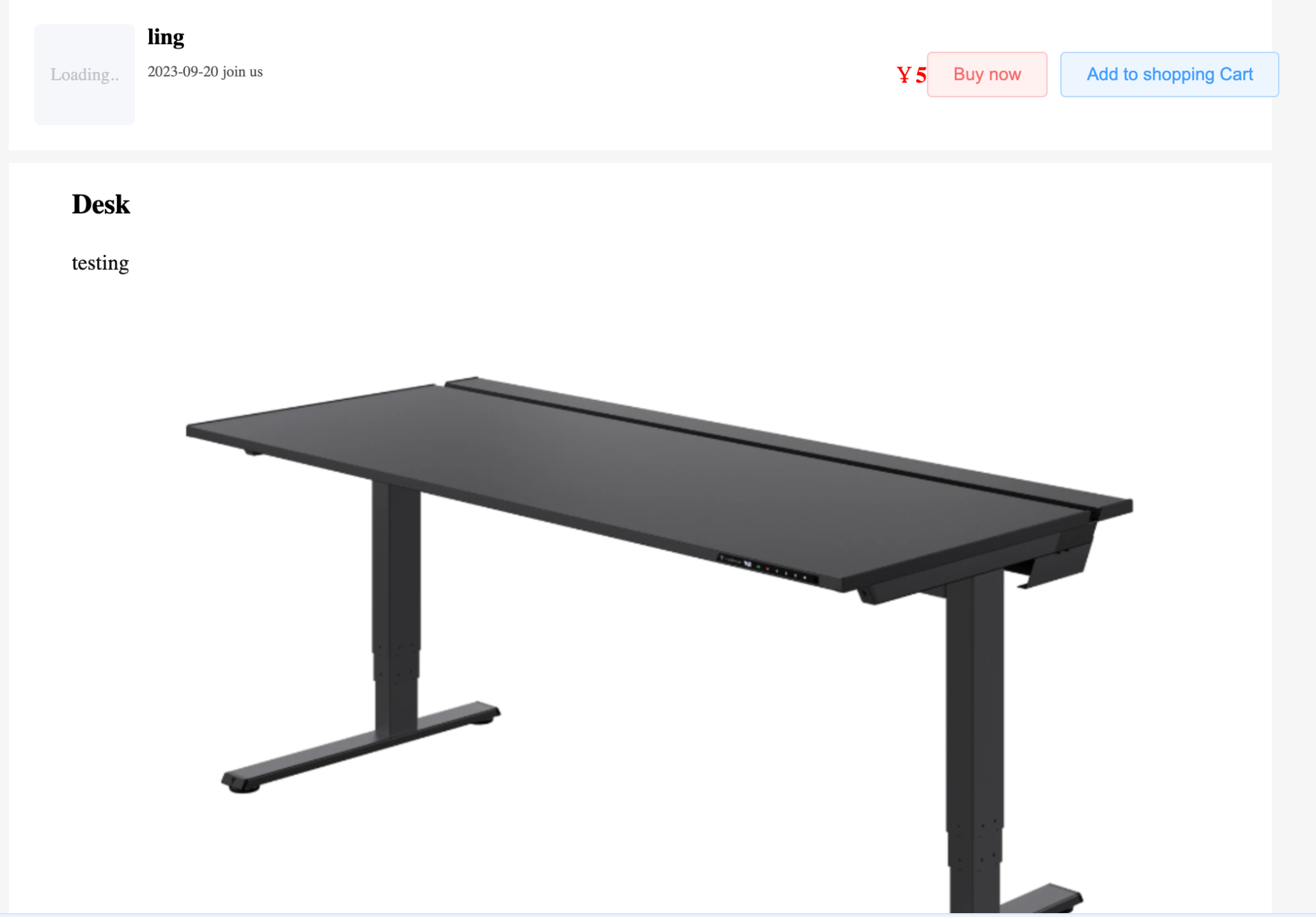
1. Index Page: The BU Second-Hand Market index page should include a search bar for users to search items by name, description, and more. The main section will display a list of items with pictures, name, description, price, location, and category. Additionally, it should feature a 'Publish Item' button,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



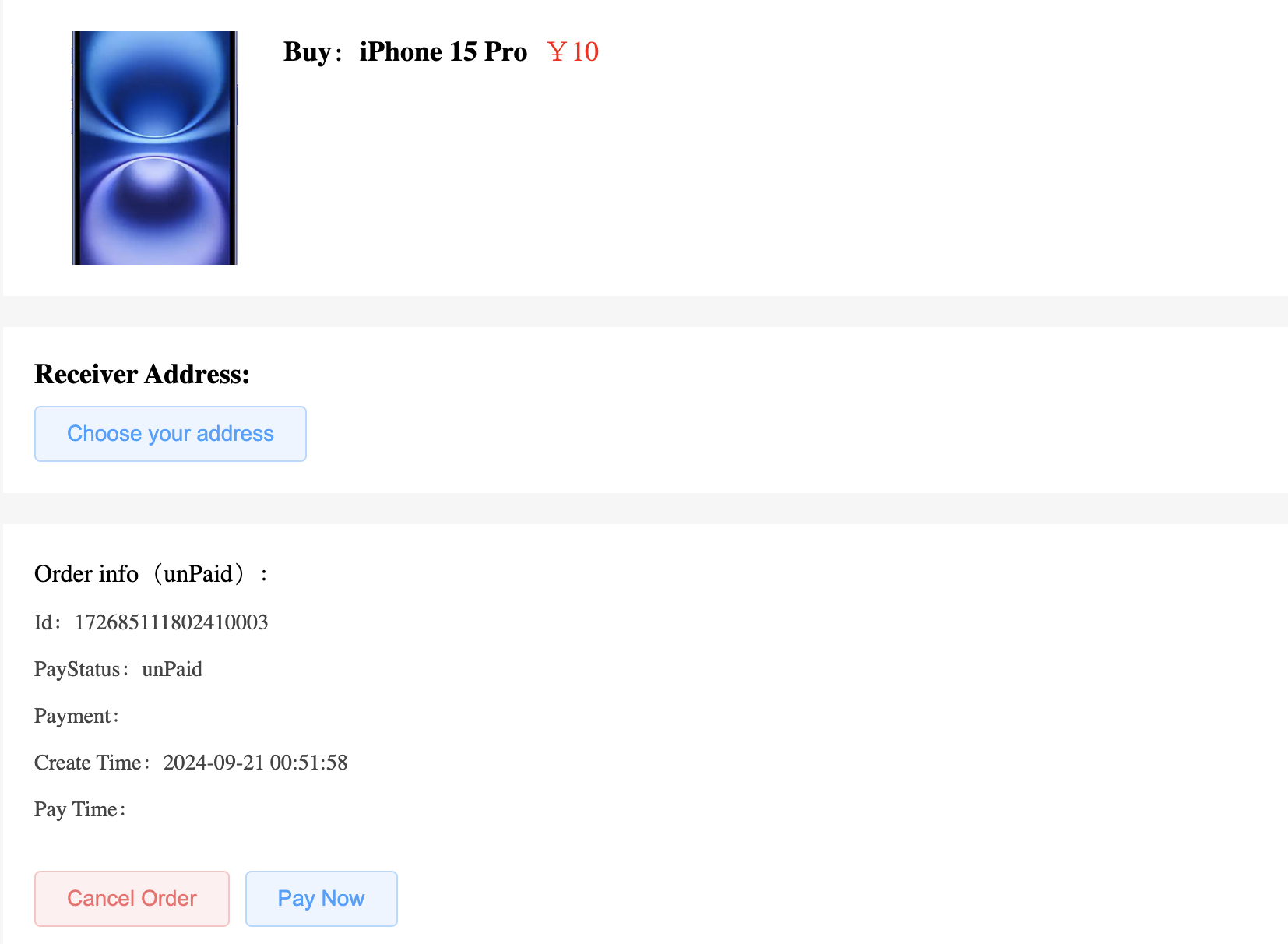
1. Publish Item: The publish item page should include fields for title, description, category, location, price, and a picture upload option, followed by a submit button.



1. 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.



1. 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.



* + 1. 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.
   * 1. Optional Features (additional cool features that you want to have if there is time):

TBD

* + 1. Existing Features (delete this item if your project starts from scratch)
  1. Nonfunctional Requirements
     1. Security requirements

# Management Plan - Yihan

## 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.

1. **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.
2. **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.

1. **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.
2. **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.
3. **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.
4. **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.
5. **Document the Project**Provide developer documentation, user guides, and API documentation to ensure the ease of future maintenance and extension.
6. **Continuously Monitor and Optimize the System**Integrate continuous user feedback by regularly collecting feedback, evaluating it, and iterating to optimize system functionality quickly.

## 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](https://docs.google.com/spreadsheets/u/0/d/1EOHJxfbe1r2A6FCcdZZyWTnh4tPsCqbDHbEYIv8BXms/edit)

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

| Iteration | Functional Requirements(Essential/Disable/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 | 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. | 1. Registration 2. Login 3. Main interfaces 4. Item posting 5. Message modules 6. Item purchases 7. Booking marks 8. Remove 9. Personal info. 10. Comments 11. Product management 12. Order management 13. Conduct Testing environment | 60 |
| 2 |  |  |  |
| 3 |  |  |  |

# Configuration Management Plan - Allen

## Tools

Version control: Github

IDE tools: VS code + other tools that based on personal preference

CI/CD tools: Github Action

SAST/DAST tools: GitHub Action / OWASP

Task management tools: Jira

Container tools: Docker, Kubernetes

Cloud Computing: AWS or Google Cloud

* 1. 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 guideline.

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.

## Deployment Plan if applicable

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

# Quality Assurance Plan - Srujana

## 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 |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

* 1. Coding Standard

(Describe any coding standard to be used)

## 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?)

## 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)

## 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. )

# 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>

# 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).