CS673 Software Engineering

Team 6 - College Street

Project Proposal and Planning

Your project Logo here if any

Team Member	Role(s)	Signature	Date
Theerarun Tubnonghee (Steve)	Team Leader/	*Theerarun Tubnonghee*	12/11/2023
Aishwarya Raja	Configuration Mgmt	*Aishwarya Raja*	12/11/2023
Nidhi Desai	Quality Assurance	*Nidhi Desai*	12/11/2023
Subhajit Das (Jeet)	Back-End Lead	*Subhajit Das*	12/11/2023
Vedant Gupta	Design/Product Implementation	*Vedant Gupta*	12/11/2023
Yin Xiancheng(Xanthus)	DevOps Lead	*Yin Xiancheng*	12/11/2023
Chenyang Lyu (Nick)	Front-End Lead	*Chenyang Lyu*	12/11/2023

Revision history

Version	**Author**	**Date**	**Change**
1.0	**Team 6**	**9/28/2023**	**Write-up**
2.0	**Team 6**	**10/19/2023**	**Requirement update**
3.0	**Team 6**	**12/11/2023**	**Requirement Update**

1. Overview

A student marketplace for all your needs. Students now struggle to adjust to a new city and have problems finding textbooks, furnishings, notes, etc. (Project name) is the solution to all the student problems. Students may offer their used furniture, textbooks, and other items, making it easier for new students to establish in a new location. The potential users would be the students in Boston, we plan to restrict the application to student email addresses. The primary technology stack that we will be implementing will be ReactJS for front-end development and Python (RestAPI) and Postgresql database for back-end development.

2. Related Work

We have decided to build an open-source online student marketplace. While doing online research for such an existing platform, we could not find something that solves the problem we're trying to solve, and it's open-source. That being said, even though there is one online student marketplace called shop swap (also built by two BU students), it's more focused on solving clothing issues among students. We are more focused on textbooks, notes, and

household items that are most necessary in any student's life. The platform like Stuvia is tailored towards study notes too, but being closed-sourced software makes it very hard to understand how it works. [our project name] is open source making it more transparent to our student and developers community.

Related Work(闲鱼，孔夫子旧书网，メルカリ):
闲鱼 Xianyu (Idle Fish): This is a second-hand marketplace app owned by Alibaba. Users can buy and sell used items, making it somewhat similar to Facebook Marketplace in its emphasis on local transactions.

孔夫子旧书网*(Kong Fu Zi Jiu Shu Wang)(focuses on 2nd-hand books):
Kongfuzi Old Book Net is the largest second-hand book trading platform in mainland China, providing second-hand books, old books, and auction services. The book categories are rich, the prices are reasonable, and the services are complete.
Here are some of the features of Kongwang:

- A wide variety of books: covering all academic fields, with more than 90 million kinds of books.
- Affordable prices: Second-hand book prices are generally lower.
- Comprehensive services: providing second-hand book trading, old book trading, auction services, etc.

メルカリ (Mercari): Japanese Online Marketplace:

- Provides a platform for individuals to buy and sell items from each other.
- Offers a wide variety of items, including new and used goods.
- Has competitive prices, generally lower than at traditional retail stores.
- Is easy to use, both for buyers and sellers.

3. Proposed High level Requirements

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:
      - Account Creation & Login/Logout (implemented): Students/Users will be able to create an account with their student email ids only, other email types will be restricted.
        - As a user, I want to sign up with my student email and create a secure password, so that I can create a user account and use the platform.
        - As a student, I want a quick and secure sign-in option using my university credentials to streamline registration

and verify users, ensuring efficiency and legitimacy.
    - As a user, I want to log in using the student email id and passowrd, so that I can use the platform features.
- Filters (implemented): User will be able to set filters for finding goods(University, Course, etc)
    - As a user, I want the ability to filter listings based on categories/tags such as textbooks, furniture, electronics, and class code, so that I can quickly find what I need at an affordable price.
- GPS: (not implement) Users will be able to put in the location to find goods near them.
    - As a user, I want to be able to filter search results based on my location, making it easier to find items available near me or near the campus.
- Create & Update (implemented): Users can publish their goods in form of post with photos and they will be able to set the status of the product(Open/Sold/On hold)
    - As a user, I want to post products for sale, allowing other users to access and view my listings.
    - As a student seller, I want to upload high-quality images of my product to attract potential buyers.
- Verification & Report: (not implement) Users will be able to report a person/post as well as verify the identity
    - As a student, I want to be able to report inappropriate listings or behavior, so that I can help maintain a safe and respectful environment
    - As a user, I want to receive an account verification email, so that I can verify my account and start browsing listings.

ii. Desirable Features (the nice features that you really want to have too):
iii. Optional Features (additional cool features that you want to have if there is time):
    - Save My Favorite (not implement)
        - As a user, I want to be able to save and revisit my favorite listings or sellers for future reference, so that I can easily track and review items or sellers I am interested in
    - Direct-Messaging (implemented)
        - As a student buyer, I want to be able to contact sellers through a messaging system to inquire about the details of the items or services they are offering, so that I can

make decision about the product

b. Nonfunctional Requirements
    - Users can setup their profile (implemented)
    - Users can view the FAQ (not implement)

4. Management Plan

a. Objectives and Priorities

(Please describe your project objectives with highest priority first. Project Goals can include but not limited to complete all proposed (essential) features, deploy the software successfully, the software has no known bugs, maintain high quality, etc) Objectives and Priorities

Objective 1 : Requirement UX/UI Analysis

Priority: High

Rationale: to provide a comprehensive requirement/solutions.

Objective 2 : Develop essential features, including account creation, login/logout, filters, create/update posts, verification/reporting, and commenting.

Priority: High

Rationale: These features are crucial for the core functionality of the marketplace, ensuring that students can efficiently buy and sell items. They directly contribute to the success of the project.

Objective 3 : Implement desirable features, such as improved GPS functionality for location pin-pointing, GPS location search

Priority: Medium

Rationale: While not essential, these features enhance the user experience and provide added value. They should be considered if time and resources permit.

Objective 4 : Successfully deploy (Project name) on the web to make it accessible to students in Boston.

Priority: Medium

Rationale: Deployment is a critical phase in the project lifecycle. This objective ensures that the marketplace is accessible to the target user base.

Objective 5 : Explore optional features like a chatbot/ direct-messaging/favorite.

Priority: Low

Rationale: Optional features can be implemented if there is time available after completing essential and desirable features. These features can make the platform more engaging but are not critical to the core functionality.

b. Risk Management (need to be updated constantly)

(Please write a summary paragraph about the main risks your group identified and how you plan to manage these risks. Then use the separate google sheet for detailed risk management. The template is provided in the same folder with this file. Please provide the link to the sheet.)

Main Risks:

Technical Challenges: There might be unexpected technical challenges during the development process, which could lead to delays.

Risk Management: We will continuously monitor the development process and have regular team meetings to address any technical challenges promptly. Detailed risk management will be documented in the provided Google Sheet: Risk Management Sheet.

User Adoption: Getting students to adopt and use the platform might be challenging, especially in the initial stages.

Risk Management: We will develop a comprehensive marketing (asking) and user engagement strategy to promote the platform among students/friends. User feedback will be actively sought and used to improve the platform's usability.

Data Security: Ensuring the security of student email addresses and personal information is crucial to compliance and user trust.

Risk Management: We will implement robust security measures, including encryption and data protection protocols. Regular security audits will be conducted to identify and mitigate vulnerabilities.

Resource Constraints: Limited time availability of developers or other resources may impact project timelines.

Risk Management: We will maintain a flexible project schedule and consider developer/team member allocation adjustments to meet project deadlines.

User Requirement Misunderstood: There might be unexpected design/user requirement that is misunderstood in team.

Risk Management: To mitigate this risk, we will establish a clear communication plan, conduct regular meetings, and document all changes to user requirements. We will also maintain a version-controlled requirements document to ensure alignment between the development team and design team.

Team Communication Challenges: Poor communication within the team may lead to misunderstandings, delays, and inefficiencies in project development.

Risk Management: To mitigate this risk, we will implement effective communication tools and channels, such as regular team meetings, status updates, and collaboration platforms. Clear roles and responsibilities will be defined, and team members will be encouraged to raise concerns or questions promptly.

.

Risk Management Sheet Link: **Team 6 of CS673_SPPP_RiskManagement**

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

Iteration	Functional Requirements(Essential/Disable/Option)	Tasks (Cross requirements tasks)	Estimated/real person-hours
0	• User requirement • System Requirement • Business requirements	- Research and design the initial version of UX/UI - User flow - Decide primary tech stack (for backend and front-end)	80
1	-Account Creation & Login/Logout -Create & Update Post	-Design database schema -Set up the development environment -Develop the UI for account creation and login/logout -Implement user authentication -Create and update posts functionality	100
2	-Filters -Commenting -FAQ1	-Implement basic filtering functionality -Add commenting functionality -Setup Deployment Environment	140
3	-Verification & Report -GPS	-Implement verification and reporting features -Integrate GPS location services	140

	-Deployment	-Deployment	

5. Configuration Management Plan

a. Tools

(In this project, we will use Git and Github as the version control tools. Please also specify any other tools to be used, e.g. IDE tools, CI/CD tools, container tools, SAST or DAST tools, and any other DevOps tools)
Tools
Version Control: Git and GitHub will be used as version control tools for tracking code changes, collaboration, and version history.

IDE (Integrated Development Environment): Visual Studio Code

CI/CD (Continuous Integration/Continuous Deployment): GitHub Actions
Container Tools: ???
SAST/DAST Tools:

- Static Application Security Testing (SAST) ???
- Dynamic Application Security Testing (DAST) ???

DevOps Tools: ???

b. Code Commit Guideline and Git Branching Strategy
(Please briefly describe criteria for the code commitment and the branching strategy used, e.g. what are the branches to be used, how the pull request will be used etc. Here is an article to give you some basic knowledge about different git branching strategies: https://www.flagship.io/git-branching-strategies/

Code Commitment Criteria:

- Commits should have clear and concise commit messages, summarizing the purpose of the change.
- Each commit should focus on a single logical change or feature.
- Code commits should follow a coding style guide (if applicable) to maintain code consistency.
- Commits should not contain any sensitive or confidential information.

Git Branching Strategy:
We will follow a branching strategy that includes the following branches:

Main/Branch: This branch represents the production-ready code. Code in this branch is stable and should only contain code that has been thoroughly tested and reviewed.

Dev/Branch: This branch is used for ongoing development work. Feature branches will be created from this branch.

FE-Dev/Branch: This branch is used for ongoing front-end development work.

BE-Dev/Branch: This branch is used for ongoing backend development work.

Feature Branches: Feature branches will be created for the development of specific features or user stories. These branches will be named descriptively and will be merged back into the development branch via pull requests.

Release/Branch (if applicable): A release branch may be created for preparing a stable release. Only bug fixes and minor adjustments are allowed in the release branch.

Hotfix/Branch (if applicable): Hotfix branches are created to address critical issues in the production code. They are merged into both the main branch and the development branch.

Pull Request Process:

- Feature branches will require code reviews from team members and be finalized by the team leader/configuration lead before merging into the development branch.
- Code reviews will focus on code quality, adherence to coding standards, and functional correctness.
- Automated tests and CI/CD pipelines will be triggered for each pull request to ensure code quality and functionality.

c. Deployment Plan if applicable

(If you plan to deploy your application (e.g. your web application), briefly describe how you plan to deploy your application).

1. Staging Environment: Create a staging environment for testing and validation of changes before they are deployed to the production environment.

2. Continuous Integration/Continuous Deployment (CI/CD): Implement CI/CD pipelines to automate the build, testing, and deployment process. Use the selected CI/CD tool (e.g., Jenkins, GitHub Actions) to ensure a seamless and consistent deployment process.

3. Containerization (if applicable): If using containerization (e.g., Docker), build container images and push them to a container registry (e.g., Docker Hub, Amazon ECR).

4. Deployment to Production: Deploy changes to the production environment following a scheduled release process or as required.

5. Monitoring and Rollback: Implement monitoring and logging to track the health of the deployed application. In case of issues, have a rollback strategy in place to quickly revert to a stable state.

6. Security and Compliance: Ensure that the deployment process adheres to security and compliance standards. Conduct security scans and audits as needed.

6. Quality Assurance Plan

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
Code Quality Metrics	Assist teams in identifying areas for improvement, tracking codebase changes over time, and ensuring that coding standards are followed. (LOC, # of files, # of classes, # methods, cyclomatic complexity, etc.)
Defect rate	Assist in identifying areas of concern, tracking the progress of defect resolution, and improving overall software quality. (Defects per KLOC)
Testing metrics	Are essential for evaluating the effectiveness of the testing process, measuring the quality of the software, and identifying areas for improvement. (# of test cases, test case pass rate)
Cost(Person-ho	Measures the total person-hours expended on the

urs used)	project.
Number of Security Vulnerabilities	Tracks the count of identified security vulnerabilities within the software

b. Coding Standard

(Describe any coding standard to be used)

1. **Naming Conventions:** We define rules for naming variables, functions, classes, and other code elements.
2. **Commenting and Documentation**: We set guidelines for adding comments and documentation to code to explain its purpose, usage, and other significant aspects.
3. **Code Readability and Maintainability**: Focus should be on making code easy to read and understand by using meaningful variable and function names, avoiding overly complex code structures, and adhering to the DRY principle.
4. **Code Reviews**: We specified procedures for code reviews and peer programming (mentioned in 6 c)
5. **Testing and Quality Assurance**: Defines the integration of testing into the development process, including rules for unit testing, integration testing, and quality assurance practices.
6. **Security**: Addresses security-related coding practices.

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 will use **Pull Requests (PRs)** for our code review process.

1. **Code Reviewers:**
    - All team members must participate in the code reviews.
    - While every team member reviews each other's code there may be a designated code reviewer for critical parts

2. **Pull Requests:**
    - The developer will create a PR after completing a task/feature. The

reviewers will discuss with the developers within the PR, and ask their questions.
- The PR should be approved by the reviewer before merging

3. **Checklist**:
   - **Functionality**: Verify that the code meets specified requirements.
   - **Code Structure**: Assess the code's organization and structure for clarity and maintainability.
   - **Documentation**: Ensure code is well-documented, including inline comments and necessary documentation updates
   - **Testing**: Confirm for presence of Unit tests

4. **Feedback**: PRs allow for review, discussions, and feedback. Team members can use inline comments to point out issues and ask questions. The feedback should be constructive and positive.

5. **Iterative Evaluation**:: Developers address the feedback and make changes. The PRs may undergo multiple reviews.

6. Ensure that only reviewed and approved code is merged

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)

Manual Testing:
Developers conduct exploratory testing throughout development.

Automated Testing:
Developers handle unit testing using Jest and React Testing Library for the frontend, and Pytest for the backend.
Integration testing will be performed by the QA leader. We will use Cypress as our tool for this purpose.

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.)

We will use **GitHub Issues** as our primary defect tool.

Types Of Defects that we will focus on:

1. **Functional Issues:** Affects the core functionality of the software and results in unexpected behavior.
2. **Feature Request**: Addresses defects introduced after adding new features/ functionality
3. **Performance issues:** Defects related to the performance of the product.
4. **Integration Issues**: Occurs when different components of the software fail to work together seamlessly.
5. **Data Defects**: Involves issues related to the handling, processing, or storage of data within the software.
6. **Compatibility**: Arises when the software fails to work correctly on different platforms, browsers, or devices.
7. **Usability Issues**: Problems related to the user interface and overall user experience.

Actions or Personnel:
Developers:

1. Responsible for resolving the defects assigned to them.
2. Prioritizes and provides input on the severity and priority of defects.

QA: Verifies the defect fixes and retests to confirm defect resolution

Team Leader: Oversees the overall defect management process, ensuring timely resolution and communication.

7. References

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

8. Glossary

(Any acronym used in the document should be explained here)