Software Project Management Plan (SPMP)

CS673 Team-5 Project

TerrierConnect

Introduction

The Software Project Management Plan (SPMP) for TerrierConnect, developed by CS673 Team-5, is a detailed guide to help manage the project effectively from start to finish. This document explains the steps, tools, and processes that will be used to make sure the project is completed on time, within budget, and meets the required quality standards.

The SPMP's primary goal is to maintain team organization by outlining roles, duties, deadlines, and important activities in detail. It facilitates effective communication and teamwork by ensuring that everyone is working towards the same objectives. We can also track our progress, manage risks, maintain quality, and stay within the project's scope with the use of this plan.

Organization

Team structure, roles, and responsibilities

Members	**Roles**	**Responsibilities**
Yi Zeng	Project Manager, Full Stack developer.	Manages the project progress, work distribution, documents, meetings and code reviewing. Participate in front-end and back-end development according to workload.
Chaojin Guo	Frontend Developer / UX designer	Implements a responsive web interface in React.js Using figma to design and improve the user interface.

		Contributes on documents like design, plan, requirements, and analysis.
Aradhana Mehra	Backend Developer / API Integration/ Quality Assurance	Implements robust backend APIs and functionalities by Django. Database design and relevant function implementation. Quality assurance and testing.
Tianchi Wu	UX designer / Frontend Developer	Implements a responsive web interface in React.js and Material UI. Using figma to design and improve the user interface. Contributes on documents like design, plan, requirements, and analysis.
Damodhar Pai	Quality Assurance Lead / Backend Developer / API Integration	Implements robust backend APIs and functionalities by Django. Database design and relevant function implementation. Quality assurance and testing.

Project management tools

- Git: Version control and source code management.
- GitHub: Remote repository, SCIs management.
- Google Drive: Collaborating documents, recording revises.
- Docker: Back-end deployment and development environment synchronization tool.
- Nginx: Web-Serve, front-end development tool.
- Discord: Communication tool within team members.
- Jira: Tasks and progress management.
- Selenium: For automated testing.

- JMeter: For performance testing.

Risk management

In order to ensure that the project can be developed and deployed properly, we analyze and manage possible risks in advance. Since our project is a website system, and according to our design, this system will include two parts, the frontend and the backend. Such a design improves the flexibility of the project, but also increases the difficulty of development and management. The developers of the frontend and backend need to cooperate more and synchronize the latest progress of the project development in a timely manner. In this way, the development of the frontend and backend parts can be maintained in a relatively balanced state.

Potential risks

1. The development of the frontend and backend uses different frameworks. Due to the lack of familiarity of the developers with the framework, the progress of the project is slow.
   Likelihood: 7 Impact: 10 Cost of Managing: 6
   Priority: 24
2. Because of the separation of the frontend and backend, if the developers of the frontend and backend do not design the specifications of the API interface in advance before starting the project development, it may cause problems when the frontend and backend are integrated.
   Likelihood: 6 Impact: 8 Cost of Managing: 4
   Priority: 60
3. Since the frontend and backend use two different programming languages, two different test plans need to be written during the testing part, which increases the complexity of the project and slows down the development progress of the project.
   Likelihood: 8 Impact: 3 Cost of Managing: 5
   Priority: 120
4. In our plan, our project will be deployed on Docker, which means we will use a lot of middleware to deploy this project. Using more middleware means we will have to write and manage more configuration files, and also deal with possible problems.
   Likelihood: 5 Impact: 3 Cost of Managing: 2
   Priority: 96
5. At present, our estimate of the project scale is based only on our analysis. It is possible that during the actual development process, we will find that the scale of the project is much larger than we expected, which will affect our estimate of the project delivery time.
   Likelihood: 9 Impact: 4 Cost of Managing: 6
   Priority: 84

Solution

Each of the potential risks listed above needs to be addressed, and if not properly addressed, each of these issues may have an impact on the progress and results of the project. So for each potential problem, we should propose a solution.

We plan to develop the project in multiple steps. In the early stage, we will ensure that the technology stack used in the project can be effectively connected. After completing this part of the work, we will continue to develop based on the environment that can already run. This can avoid the problem of difficulty in integrating various components in subsequent development.

During the development process, the frontend and backend will cooperate to decompose the project into multiple components. After the development of each component is completed, the connection between the tests can work normally before proceeding to the next step of development. Similarly, the test code of each component will be carried out after the component is written. In this way, all functions of each component can be tested as much as possible.

Since we will divide the project into multiple modules and gradually implement functions, this will also help us better manage the development progress of the project. The development of core functions will have a higher priority. This will ensure that the system is in a fully usable state when the project is delivered.

Scheduling

Schedule management plan

- Weekly meetings: Will be scheduled on Wednesday once and optional once on Monday. Communicate any absences or unavailability to the group with appropriate time to make changes or adjustments as needed.
- Timeline: Progress and timeline tracked using Jira to visualize task dependencies, durations, and milestones.

Timeline of this semester

We created a timeline (Gantt Chart) on Jira, it is an approximate plan, changing with the current situation. We scheduled main tasks during this semester, including but not limited to:

- SPMP Document

- Requirement Analysis Document
- Design Document
- Midterm Presentation Slide
- Iteration 1
- Iteration 2
- Final Project Presentation Slide

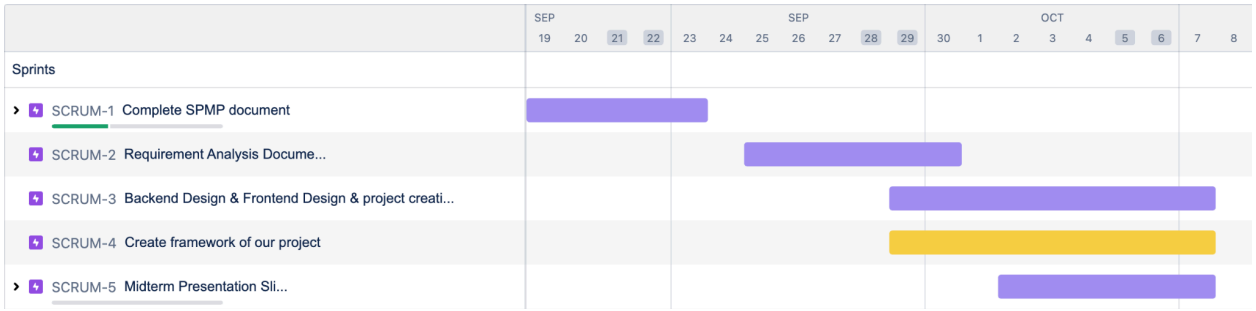

Figure 1. Timeline from Sep 19 to Oct 8

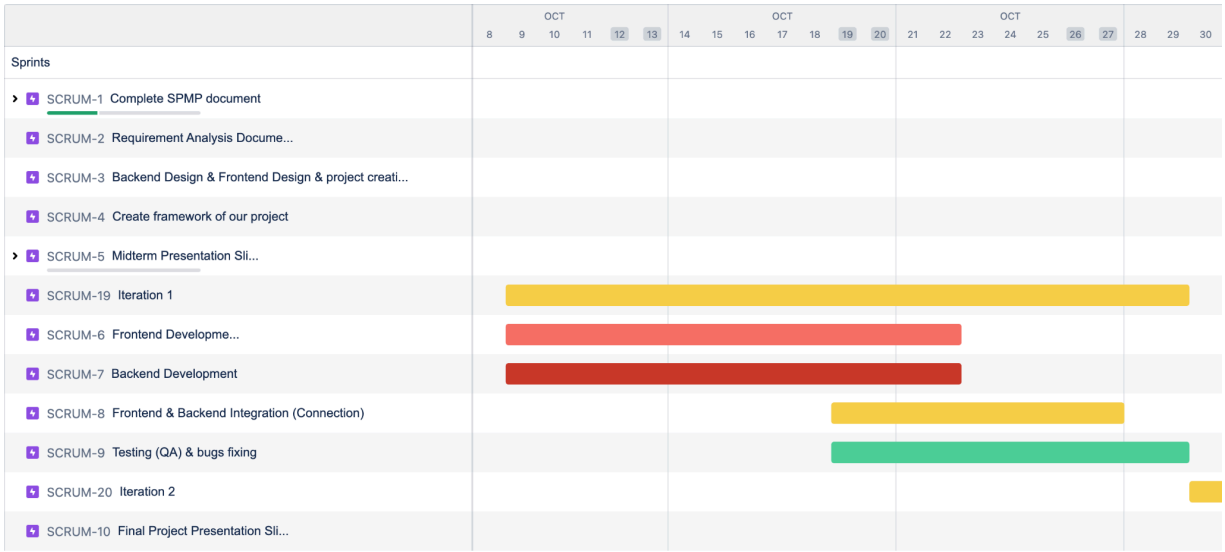

Figure 2. Timeline from Oct 8 to Oct 30

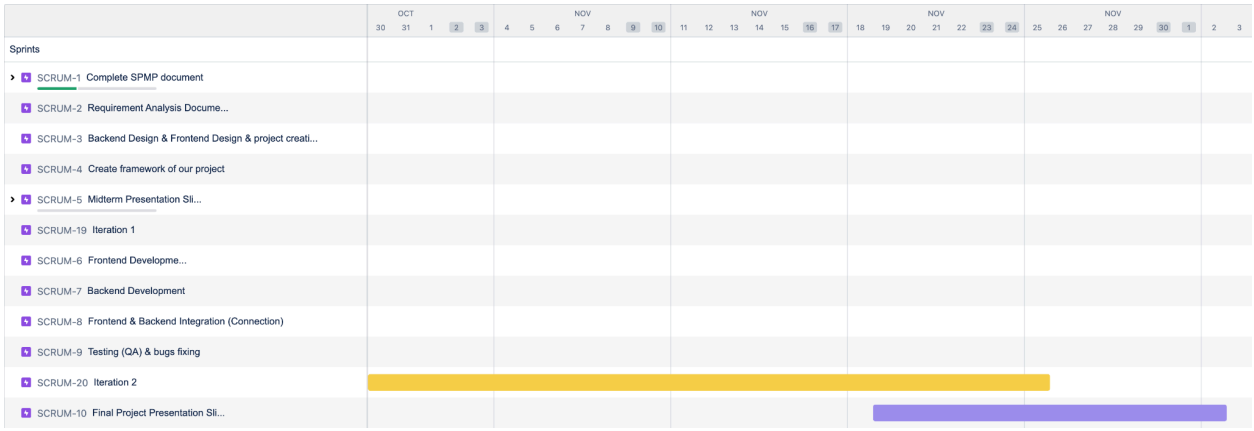

Figure 3. Timeline from Oct 30 to Dec 3

Estimation

Here's a rough breakdown of how the 1,000 to 1,200 LOC for the Terrier Connect project could be distributed across the major components:

1. Front-End (React):
   UI Implementation: Login, post creation, event display.
   Estimated LOC: 500 to 600 lines.

   Google Maps Integration: Adding map functionality.
   Estimated LOC: 80 to 100 lines.

   Event and Post Interaction: Posting events and filtering.
   Estimated LOC: 100 to 150 lines.

   Total Front-End: 680 to 850 LOC.

2. Back-End (Django, RESTful API):
   User Authentication: Login, registration.
   Estimated LOC: 80 to 100 lines.

   Event Management and API Implementation: Event posting, profile management.
   Estimated LOC: 100 to 150 lines.

   Database Handling: Interactions with posts, events, and user data.
   Estimated LOC: 50 to 80 lines.

   Total Back-End: 230 to 330 LOC.

3. Database (SQLite):
   Schema Design: Defining tables for users, posts, events.
   Estimated LOC: 50 to 80 lines.

   SQL Queries: Data retrieval and manipulation.
   Estimated LOC: 30 to 50 lines.

   Total Database: 80 to 130 LOC.

4. Testing and Deployment:
   Unit Tests: Key functionalities like login, posting, location sharing.
   Estimated LOC: 80 to 100 lines.
   Deployment Scripts: Docker, Nginx.
   Estimated LOC: 50 to 80 lines.

   Total Testing & Deployment: 130 to 180 LOC.

   Rough Breakdown Summary:
   Front-End: 680 to 850 LOC.
   Back-End: 230 to 330 LOC.
   Database: 80 to 130 LOC.
   Testing & Deployment: 130 to 180 LOC.

Effort Estimation (COCOMO Model):

Using the COCOMO model for effort calculation:

1. Effort = a·KLOC^(b), where:
   a=2.4
   b=1.05
   KLOC = Thousands of Lines of Code.

   Effort Calculation:
   For 1,000 LOC (1.0 KLOC):
   Effort = 2.4 · (1.0)^1.05
   Effort ≈ 2.53 person-months.

   For 1,200 LOC (1.2 KLOC):
   Effort = 2.4 · (1.2)^1.05
   Effort ≈ 3.34 person-months.

2. Duration Estimation:
   Using the formula:
   Duration = c · Effort^(d), where:
   c=2.5, d=0.38.

   Duration Calculation:
   For 1,000 LOC:
   Duration = 2.5 · (2.53)^0.38
   Duration ≈ 3.3 months.

   For 1,200 LOC:
   Duration = 2.5 · (3.34)^0.38
   Duration ≈ 3.7 months.

3. Cost Estimation:
   Assuming a cost of $8,000 per person-month:

   Cost Calculation:
   For 1,000 LOC:
   Cost = 2.53×8,000 = $20,240.

   For 1,200 LOC:
   Cost = 3.34×8,000 = $26,720.

Summary of the Revised Estimation:

Effort:
For 1,000 LOC: 2.53 person-months.
For 1,200 LOC: 3.34 person-months.

Duration:
For 1,000 LOC: 3.3 months.
For 1,200 LOC: 3.7 months.

Cost:
For 1,000 LOC: $20,240.
For 1,200 LOC: $26,720.

Key Assumptions:

Effort and Duration: The estimates assume an organic development environment (small team, familiar technology).
Parallel Development: Adding more developers won't significantly reduce the timeline due to coordination overhead.

Documentation and monitoring

In this project, we will employ a range of monitoring and controlling mechanisms to ensure adherence to the Software Project Management Plan (SPMP). The mechanisms used will include reporting tools, review processes, audits, and status updates, as outlined below. These tools and methods will enable continuous oversight and management of project activities and deliverables.

Reporting and Communication Plan

Information	From	To	Time Period Communicated
Status Report	Project Team	Project Manager	Weekly
Status Report	Project Manager	Project Team	Weekly
Project Review	Project Team	Software Manager	As Needed
Document Updates	Project Team	Project Manager	As Needed
Integration Updates	Backend Team	Frontend Team	As Needed
QA Testing Results	Project Team	Project Manager, QA Lead	Weekly (during test phase)

Reporting Mechanisms

- **Jira**: Jira will be the primary tool for tracking tasks, progress, and issues throughout the project. Each work package will have its own corresponding task in Jira, where updates are made regularly by team members.
- **Progress Reports**: Each team member will submit a weekly progress report through the **CS673_Team5_ProgressReport** Excel document, available in the shared Google Drive. This will detail the individual contributions, next week's plan (optional), and any blockers encountered during the week.
- **Meetings**: Weekly stand-up meetings will be held on zoom to discuss progress, blockers, and next steps. These meetings will involve the entire project team and the project manager. Monthly project reviews with the project manager will also be conducted to track overall progress.
- **GitHub**: All code contributions will be version-controlled through Git and GitHub. Code reviews and change tracking will be part of the continuous integration system managed through GitHub.

Quality Management

Quality Objectives

Product Quality:

1. Ensure all the required features and functions are implemented as specified.
2. Design a clean, intuitive and user-friendly interface.
3. Keep the quality of codes. Write code that is reusable, maintainable, and easy to read.

Process Quality:

1. Follow established coding standards and best practices throughout the development process.
2. Update the relevant documents in time to record the process of development.

Quality Assurance

QA Activities:

1. Code Reviews: Regular peer reviews of code to ensure adherence to coding standards and identify potential issues early.
2. Automated Testing: Implement unit tests, integration tests, and end-to-end tests to catch defects early.
3. Manual Testing: Conduct exploratory testing and usability testing to identify issues that automated tests might miss.

4. Security Audits: Perform regular security audits to identify and mitigate vulnerabilities.

Roles and Responsibilities

QA Lead: Oversees the QA process, ensures standards are met, design test cases and coordinates testing activities.

Developers: Participate in code reviews and write unit tests.

All members: Follow the direction and work distribution from QA Lead. In principle, developers mainly perform testing on the modules they are in charge of.

Quality Control

QC Techniques

1. Static Code Analysis: Use tools to analyze code for potential errors and adherence to coding standards.
2. Automated Testing Tools: Utilize tools like Selenium and others for automated testing.
3. Performance Testing Tools: Use tools like JMeter to test the performance and load handling of the platform.

Inspection and Testing

1. Unit Testing: Test individual components for correctness.
2. Integration Testing: Ensure that different components work together as expected.
3. System Testing: Validate the entire system's functionality and performance.
4. User Acceptance Testing (UAT): Conduct testing with actual users to ensure the platform meets their needs and expectations.

Continuous Improvement

Feedback Mechanisms

1. User Feedback: Collect feedback through surveys, user interviews, and feedback forms.
2. Bug Reports: Encourage users to report bugs and issues through a dedicated bug tracking system.

Process Improvement

1. Regular Reviews: Conduct regular reviews of processes and practices to identify areas for improvement.

2. Retrospectives: Hold retrospectives after major milestones to discuss what went well and what could be improved.
3. Training and Development: Provide ongoing training for team members to keep skills up-to-date and improve overall quality.