Auri Laitinen, Niko Mehiläinen, Samuel Sarimo

Shout! - Social Media Platform

13. Social Networking Platform

Metropolia University of Applied Sciences

Ohjelmistotuotantoprojekti 1 TX00EY27-3007

Project Plan

Group 1

30.09.2025

Contents

1 Project Overview

Project Title:
Social Networking Platform

Problem Summary:
Current social media platforms are too open and you feel disconnected from your closest friends. There is pressure to create perfect content and get as many likes and comments as possible. This eventually leads to not being as active as you would like to be.

Intended Audience/Users:

- Users who prefer private messaging apps for broadcasting instead of open social media platforms.
- Groups (Friends, work, sports teams etc.)

Main Features/Components:

- Register and sign in
- User profile management : Users can change username, profile picture and password.
- Post : Users can share a text with or without an image to their feed.
- Like : Users can like posts.
- Comment : Users can comment to posts.
- Follow : Users can follow other users.

2 Project Objectives

The primary objective of the project is to develop a user-friendly, compelling and relaxed social media platform, where users can freely share their daily thoughts.

To achieve this goal, we will focus on the following key objectives:

- Successfully implement all of the main features.
- Design and implement an intuitive, modern and user-friendly interface.
- Develop a well-structured and scalable code base using best practices.
- Ensure that the application runs seamlessly without any major bugs.
- Get 10+ registered users on launch.

3 Scope and Deliverables

Inclusions:

- User authentication.
- Basic social media actions. (post, like, comment, follow, unfollow)
- User profile modification, including changing username, profile picture and password.
- Database for storing user data and other application data, including likes, posts, comments and followers.
- User interface created with JavaFX.

Exclusions:

- Direct messaging between users.
- Support for posting videos.
- Real time notifications or updates.
- Mobile or web-based version of the application.
- Advanced security measures such as two-factor authentication or email verification.

4 Project Timeline

This section briefly outlines the project timeline and the planned goals. It also describes the focus of each sprint.

4.1 Project Duration

The project has four sprints, each lasting two weeks. Making the total duration of the project eight weeks. In every sprint, the team works on specific objectives to manage tasks, and track progress.

Each sprint is followed by a sprint review, where the development team, together with the SCRUM Master and Product Owner, reviews the progress to ensure the project stays on track and meets the desired requirements.

4.2 Key milestones and deadlines

Each sprint is carefully planned, and the team works to meet the requirements of the software development life cycle. The life cycle consists of five phases: Planning, Requirements Gathering and Analysis, Design, Implementation, and Testing & Deployment. These five phases are divided evenly across the four sprints. The timeline below provides an overview of the entire development schedule, including key milestones and deliverables.

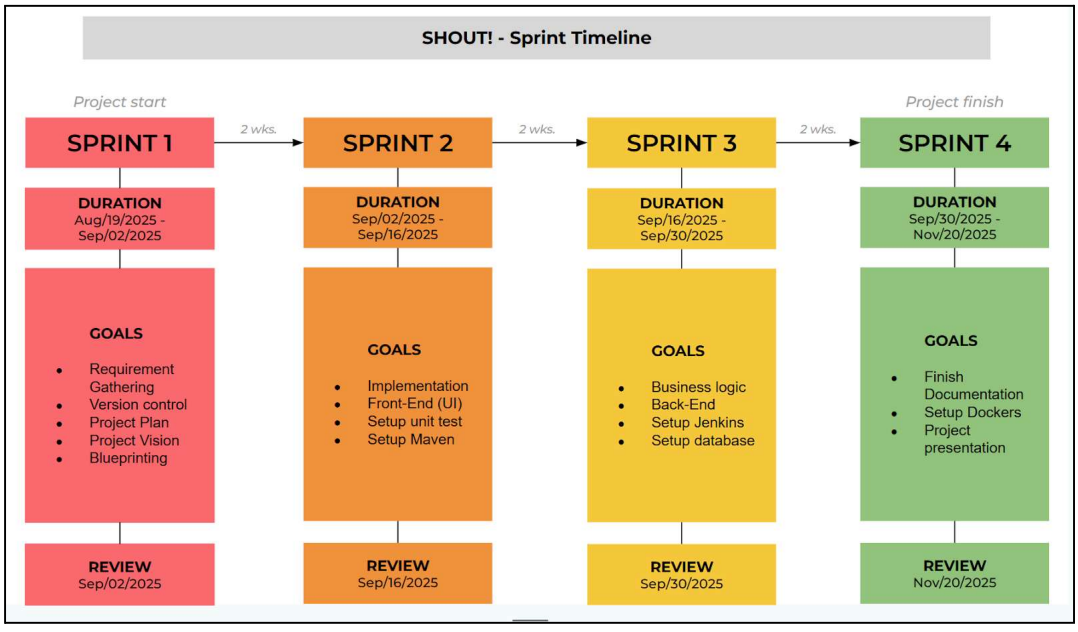

Figure 1: Project timeline with sprint breakdowns.

Sprint 1

In the first two weeks, the development team focuses on planning, gathering requirements, and designing. The main goal is to create a clear plan for the project. This includes deciding what the project will do, who it is for, what technologies will be used, and making the first blueprints of the user interface and database design.

Sprint 2

The second sprint will mostly focus on the front-end of the project. It is very important that the first sprint is done well because a well-prepared and planned product affects the entire project. During this sprint, the development team begins setting up the first unit tests and configures Maven, which is used for managing project dependencies and building the application.

Sprint 3

At the start of the third sprint, the development team has likely solved the biggest problems and knows what is still missing from the project. The main focus will be on building the business logic and finishing the database. The team will also set up Jenkins, which is used to automate builds and testing. It's also important to leave some time to finish tasks from earlier sprints, since not everything may be done on time.

Sprint 4

In the last sprint, the implementation should be mostly finished, with only some final adjustments needed. The last thing added to the project will be Docker, which is used to package the application so it can run consistently on any system. The focus will be on finalizing the documentation and updating the project plan and vision to match the finished product. The development team will make final tweaks and prepare to deploy the project for presentation.

5 Resource Allocation

5.1 Development Team

The production team consists of three members: Auri, Niko, and Samuel. They contribute equally to the project, and as the software is further developed, the team discusses which aspects of the software each member would like to focus on.

Although all three members are equal, at the start of each two-week sprint one is chosen to act as Scrum Master, taking responsibility for the team's schedule. That member leads the software development process for the duration of the sprint.

5.2 Technologies Employed

The project will be mainly implemented using Java and its frameworks, such as JavaFX and Jakarta Persistence API. The database will be MariaDB, a type of SQL database. User authentication will be handled using Java Authentication and Authorization Service (JAAS).

To support development, the project will also use tools like Maven, Jenkins, and Docker. Maven helps manage project dependencies and builds, Jenkins automates testing and continuous integration, and Docker ensures the application runs consistently across different environments.

5.3 Ideologies and Methods Applied

Throughout the software's development, the team focuses on following the best practices. Documentation is prioritized, and effective communication between developers is seen as the key to delivering a successful project.

To ensure the development stays on schedule and all team members remain informed about its progress, the team follows the SCRUM framework. By using Agile practices, the team addresses unexpected obstacles while working closely with the product owner to ensure the software meets the requirements.

6 Risk Management

This section identifies potential risks to the project's success and outlines strategies to mitigate them.

Database Schema Changes:

- Evolving project requirements may necessitate changes to the database schema during development.
- Likelihood: High.
- Impact: Medium.
- Mitigation strategies: Design the initial DB schema and backend code with flexibility in mind.

Time management and feature creep:

- Underestimation of how long some project tasks take to complete and unforeseen feature additions.
- Likelihood: medium.
- Impact: medium.
- Mitigation strategies: Start off small and do the initial design as bare bones as possible, focus on delivering MVP (minimum viable product).

Life events and getting sick:

- Possibility of a team members' unexpected illness, flu or some other event that prevents ability to work.
- Likelihood: High.
- Impact: Medium.
- Mitigation strategies: Don't get sick. Wash hands, try to rest enough in your free time. Eat vitamin D.

7 Testing and Quality Assurance

Our approach to Quality Assurance will be proactive and encompassing, focusing on preventing UX issues and ensuring a stable, functional application. We will use Jenkins and the JUnit framework.

Types of Testing:

- **Unit Testing:** We will write JUnit tests for all backend service classes, utility methods, and core logic (e.g., user authentication, post creation, friend following).
- **Integration Testing:** We will test the functionality of different integration points of our application.

Criteria for Success:

- **Unit & Integration Tests:** All unit and integration tests should pass (≥ 90% pass rate) before new code is merged into the main branch.
- **Feature Completion:** All features defined by the MVP are implemented and functional.
- **Performance:** The application must feel responsive. Key actions (loading the feed, posting a message) should execute within an acceptable time frame (< 2 seconds for most operations).

8 Documentation and Reporting

8.1 Technical Documentation

- **Inline comments** will be made to explain complex algorithms or non-obvious code decisions.
- **JavaDoc**
- README.md that comprehensively documents and instructs in the use of the software and how to setup required components like Maven, Jenkins, Docker and MariaDb for development.

8.2 User Documentation/Manual

- A brief user manual will be made that details and describes the use of the software from a users point of view.

8.3 Progress Reporting

Progress will be reported according to the instructions on the Moodle materials. These include sprint reviews and the final presentation.