Software Engineering Project Plan

Group 6: Ade Aiho, Heta Hartzell, Mika Laakkonen, Jonne Roponen

1. Introduction

In today's world of online dating, speed dating has emerged as a popular format for individuals seeking meaningful connections in person. However, organizing and managing these events manually presents significant challenges, including inefficient participant enrollment and matchmaking processes, as well as difficulty in tracking interests. The lack of automation often results in delays, confusion, and missed opportunities for participants.

This project aims to develop a comprehensive software solution that streamlines speed dating events through automation and intuitive features. Designed primarily for Tatskatytöt-hosted events, our application simplifies participant registration, interest tracking, session coordination, and post-event matchmaking, ensuring a seamless and enjoyable experience for all users.

Our vision is to create a visually engaging, user-friendly platform that minimizes manual tasks, enhances participant engagement, and optimizes event management. Key features of the system include automated matchmaking, an interactive "Active View" during sessions, profile management and administrative tools for session oversight. Additionally, the software will incorporate SMS notifications and an intuitive help section to support users throughout the event process.

By implementing this solution, we aim to improve efficiency, reduce frustration for organizers, and aid more meaningful connections in speed dating environments. This project utilizes technology to enhance the dating experience while preserving the authenticity of in-person interactions, making speed dating events more engaging and well-organized.

2. Project Organization

Efficient and clear project organization is essential for a successful development process and outcome. This section outlines the team’s structure, communication strategies, and stakeholder involvement.

2.1 Team Structure

Our team is organized in the following way:

- UI Designers: Heta and Ade are responsible for designing a user-friendly and visually appealing interface using Figma.
- Backend Developers: Mika and Jonne are primarily responsible for server-side development.
- Frontend Developers: Heta and Ade are primarily responsible for implementing the UI designs and connecting them to the backend.

- Scrum Master (changes for every sprint)
    - Sprint 1: Heta
    - Sprint 2: Mika
    - Sprint 3: Jonne
    - Sprint 4: Ade

While these roles provide a general structure, they are not definitive. Team members will take on different tasks to balance the workload and assist others if needed to ensure the project stays on schedule.

2.2 Communication Plan

Effective communication ensures transparency and efficiency throughout the project. The team will communicate through:

- Daily Scrums: Short meetings to discuss progress, challenges, and priorities. If needed, these meetings can be longer to discuss major updates, unresolved issues or other relevant matters.
- Sprint Planning Meetings: Held at the start of each sprint to review backlog, assign tasks and set goals.
- Sprint Review Meetings: Held at the end of each sprint to showcase progress, gather feedback and discuss next steps.

The main communication platform is Discord, where also links to all documentation will be maintained. The team will also meet in-person based on their schedules. The sprint reviews will be held on Zoom. Progress is tracked with Trello and GitHub is used for code management.

2.3 Stakeholder Involvement

The primary stakeholder in our project is Tatskatytöt. They are informed and involved through:

- Bi-weekly Updates: Reports to summarize progress, demonstrate features and discuss next steps.
- Final Presentation**:** A comprehensive walkthrough of the application’s features and usage.

Stakeholders are encouraged to provide feedback during updates and reviews, ensuring the project stays aligned with their expectations.

3. Risk Analysis

Risk analysis and management is essential to ensure the project succeeds and that the team is aware of unexpected troubles. The below **table 1.** highlights and summarizes some of the most relevant risks associated with this software engineering project. As this is a student-run project, some risks regarding topics such as budget, lack of developers, and external risks are either eliminated or reduced significantly. However, many risks are still

apparent or even more likely, especially those related to inexperience, such as poor code quality.

Description	Likelihood	Impact	Mitigation Strategies
Illness of team member	medium	medium	The likelihood of the risk occurring is nigh impossible to reduce. To manage members' illness, the ill member needs to inform the group about their wellbeing, and if they are in working condition or not. Depending on the timetable and level of illness, another member might need to take up their work or project goals may need to be re-evaluated to reduce the chance of the project failing.
Data corruption or loss	low	high	Files should be backed up in a cloud and version control (Git) used for programming.
Malfunction or breaking of hardware	low	high	Members should take good care of their hardware and be aware of any backup options should something break. In the event hardware breaks, members need to take it to repair.
Poor quality code	medium	medium	Code should be tested and bug-fixed frequently. Code should follow best practices and standards to mitigate bugs and errors, and improve readability.
Aggressive deadlines	medium	medium	Aggressive deadlines can lead to incomplete features and team member burnout. Sprints should be planned with realistic expectations and estimates in mind to reduce this risk. Missed features can also be implemented during a later sprint, should there be one.
Scope creep: the project's scope changing to something it was not originally.	medium	medium	Project scope should be reviewed during each sprint to identify unnecessary changes and keep it in line with customer expectations.
Poor communication between team	high	low	Poor or lack of communication is bound to occur. This can lead to duplicate or mismatched work. To

			mitigate this risk scrums should be held often and to-do's should be updated frequently in Trello.
Huge client requirements	high	low	The client might require more of the project than can be delivered. In this scenario, the project should be executed according to the group's expectations, estimates and capabilities, rather than the client's.

Table 1. The table outlines potential risks associated with the project as well as their likelihood, impact, and mitigation strategies.

4. Hardware and Software Resource Requirements

4.1 Hardware

As this is a student project with only 4 group members, the hardware requirements are relatively shallow. Each member needs their own computer with enough performance to run IDE's and most other light- to medium-software. This is ideally a laptop, as working locations change frequently. Furthermore, each member needs some form of transport to travel between locations. Finally, each member needs a mobile phone for communication purposes.

4.2 Software

The project's software requirements are much more complex and demanding than the hardware ones. As this is a student project, the group may encounter restrictions or limitations on what software, API's, and other useful tools are available. This is mainly due to budget constraints and lack of experience, yet some software is accessible with Metropolia's student credentials.

4.2.1 Programming Languages and Libraries

The project will be written in the Java programming language, as it is a mandatory project requirement. This limits what the group wants to accomplish in certain aspects, yet is non-negotiable. Additionally, a database software solution is needed. To conform with project requirements, this project will use mySQL and the Jakarta Persistence API for object relational management, as it is a Java library. For the user interface, the JavaFX client application platform will be used, as it is easy to integrate into a Java project and complies with project requirements.

4.2.2 Development Tools and Platforms

Members will use the IntelliJ integrated development environment to write the program. Not only is it meant for Java programming, but is also familiar to group members from prior courses. Additionally, Github will be used for version control throughout the project.

4.2.3 Testing

JUnit tests will be written for database connectivity and CRUD operation testing. Later down the line, Jenkins will likely also be implemented to automate processes related to testing, building and deployment.

4.2.4 Deployment, Operation, and Monitoring

Docker will be used to containerize the application and simplify deployment. Kubernetes will be used to gather customer feedback and tickets, and to get important information about application usage, such as usage and traffic. Finally, DataDog will be used for monitoring, particularly to get data about performance, errors, and security.

5. Work Breakdown

Work breakdown structure (WBS) seen in **image 1.** outlines the tasks necessary to complete the project. It has a short description of each task, dependencies between tasks, and members responsible for each task.

Google docs link to the image WBS

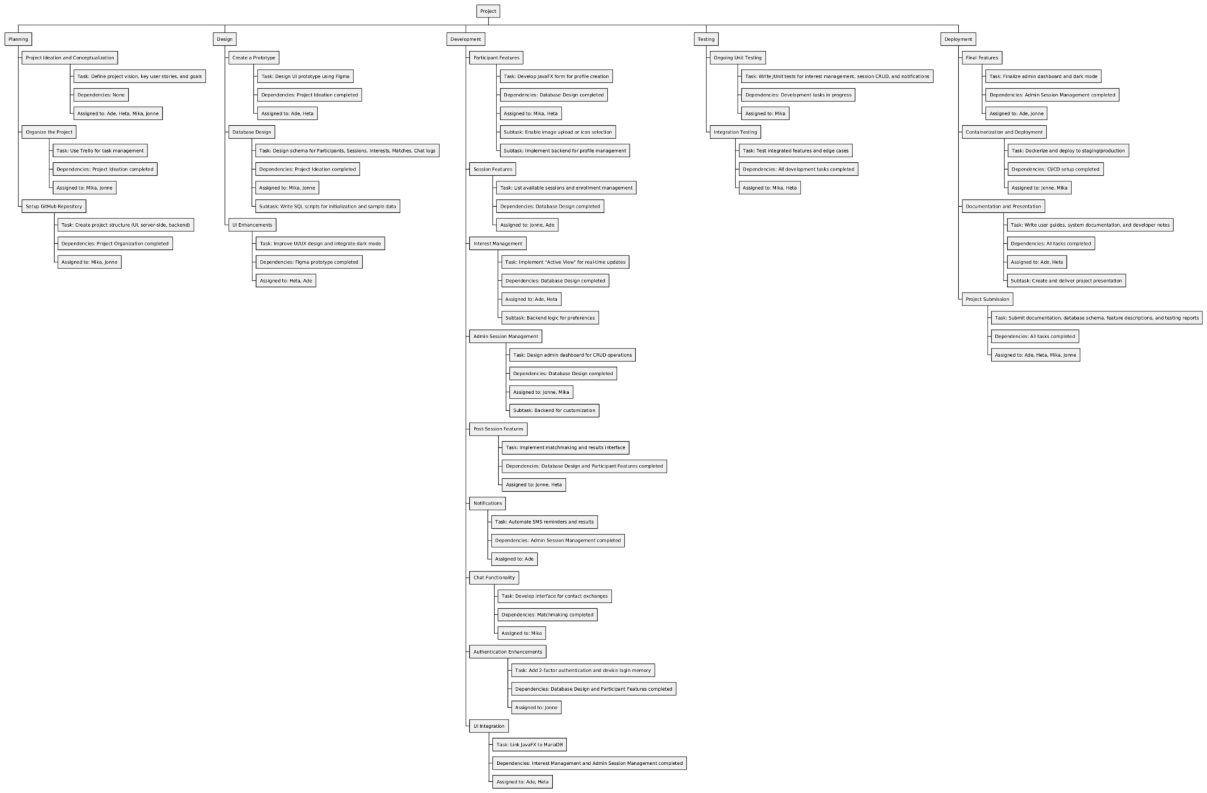

Image 1. Work Breakdown Structure.

6. Project Schedule

Project schedule outlines key phases of the project. It includes dates for sprint planning, reviews, and important milestones along with a list of tasks for each sprint.

6.1 Sprint 1 - Project Setup and Initial Design

- Sprint duration: 13.01.2025 - 27.01.2025
- Sprint planning: 13.01.2025
- Tasks:
    1. Project ideation.
    2. Organize the project using Trello for task management.
    3. Setup GitHub repository.
    4. Create a prototype of the application using Figma
    5. Design database schema and create needed tables (participants, sessions, interests, matches).
- Sprint review meeting: 27.01.2025
- Milestone: Complete project ideation, prototype creation, and database schema design.

6.2 Sprint 2 - Core Feature Implementation

- Sprint duration: 27.01.2025 - 10.02.2025
- Sprint planning: 27.01.2025
- Tasks:
    1. Develop basic JavaFX UI for login, registration, and profile creation.
    2. Implement CRUD operations for participant profiles.
    3. Build interest management features (add/update interests in real-time).
    4. Create an admin dashboard for managing sessions (create, update, delete).
    5. Link JavaFX UI to MariaDB for data display and manipulation.
    6. Write JUnit tests for session and interest management.
- Sprint review meeting: 10.02.2025
- Milestone: Integrate CRUD functionality, basic session/interest management.

6.3 Sprint 3 - Advanced Features and CI/CD Integration

- Sprint duration: 10.02.2025 - 03.03.2025
- Sprint Planning: 10.02.2025
- Tasks:
    1. Implement a post-session matchmaking results page.
    2. Improve UI/UX design with usability enhancements and a help section.
    3. Connect frontend and backend.
    4. Develop admin features for managing users and matches.
    5. Implement a randomized profile image rotation for better user experience.
    6. Write JUnit tests for matchmaking logic.
    7. Set up CI/CD pipelines for automated testing and deployment.
- Sprint Review Meeting: 03.03.2025
- Milestone: Complete matchmaking, UI/UX enhancements, and Jenkins setup.

6.4 Sprint 4 - Finalization and Deployment

- Sprint duration: 03.03.2025 - 14.03.2025
- Sprint Planning: 03.03.2025

- Tasks:
    1. Do extensive testing to ensure overall system functionality.
    2. Write additional JUnit tests for edge cases.
    3. Containerize the application using Docker.
    4. Prepare a final project presentation showcasing key features of the application.
    5. Submit final project documentation including database schema, feature descriptions, and test reports.
- Sprint Review Meeting: 14.03.2025
- Milestone: Finalize system testing, complete deployment, and submit documentation.

7. Monitoring and Reporting Mechanisms

With effective and consistent communication, monitoring and reporting as well as structured tools and methods, the team ensures that the project stays organized, adaptable and on track to meet deadlines and expectations.

7.1 Progress Tracking

The team practises Agile principles for effective tracking. Each sprint has specific milestones and deliverables to provide a framework to evaluate progress and make adjustments as needed. The team uses Trello to sort tasks into "To Do," "Doing," "Done" and “Postponed” categories, with deadlines for every team member. The team will also share individual progress and challenges during daily scrum meetings.

7.2 Reporting

Clear and regular reporting ensures that the team and the stakeholders are aligned with the progress.

The primary way of discussing updates within the team is during the daily scrum meetings. Any major updates or issues outside the meetings will be communicated via Discord or in person.

The stakeholders are updated bi-weekly with a summary of progress, feature demonstrations and plans for upcoming work. A final report is delivered at the end of the project, showcasing its features and usage.