Project Overview

The idea of the project is to create a Java-based application that allows people to communicate with each other. The conversation will be text-based, but possibly also via speech.

The application is designed to reduce friction in communication between project group members and co-workers. It is designed to contain all of the essential tools needed to communicate and collaborate in smaller teams or just privately between two people.

The application is primarily focused on text messaging, but aims to provide audio chat as well. While chatting the users can send different kinds of media files ranging from images to videos. The application plans to include a calendar where groups can pin important dates for meetings or events.

The chat interface provides a clean and intuitive user experience, with elements that are easy to find and pleasing for the eye. The application is divided into three main sections with the chat being the main view. On the sides the user can scroll through and manage their chat histories and friends that they've added. The user can also access profile management and application settings from the main view.

Intended Audience

Application is primarily designed for students or the general public with lower-end systems who need a lightweight program for group work or casual everyday conversations. However, it's versatile and open for anyone to use.

Main Features

- Login and registration (modifying user information, profile picture, email, etc.)
- Adding people.
- Primarily text-based conversations, but possibly also voice communication.
- Creating groups.
- Public wall where everyone can post or leave comments.
- Multiple language options (FI/EN/SWE).
- Calendar for adding events.

Project Objectives

- Develop a lightweight Java-based communication application that enables seamless interaction between individuals and groups.
- Provide communication features such as text-based messaging and media sharing.
- Allow users to create groups and share events into calendars.

- Ensure user accessibility and personalization by supporting multiple languages and offering profile customization.

Scope and Deliverables

The project scope covers the development of a lightweight Java-based communication application that provides core features for individual and group communication.

What will be included:

- User Authentication and Registration:
  - login, registration, profile management (profile picture, email, basic customization).
- Chat Functionality:
  - One-on-one text-based messaging.
  - Group messaging.
- Calendar Integration:
  - Ability to add and view events/meetings within groups.
- Social Features:
  - Adding friends.
  - Public wall for posting messages or comments.
- User Interface:
  - A clean UI built with JavaFX.
- Documentation and Testing:
  - Sprint documentation, system design documents, and code reviews.
  - Unit, integration, and user acceptance testing with JUnit.

Out of Scope

- Voice Communication:
  - While considered in planning, real-time voice chat will only be implemented if time permits. The focus will be on text communication.
- Advanced Security Features:
  - The application will cover essential authentication and protection against common vulnerabilities, but two-factor authentication, and encryption beyond baseline requirements will not be in the scope.

Project Timeline

Sprint 1 (19.08.2025 - 31.08.2025)

- Project planning (requirements and specifications).
- Trello Setup (Product Backlog / User Stories).
- Designing the UI.
- Structure for the project (MVC model).
- Github Setup.

Sprint 2 (01.09.2025 - 14.09.2025)

- Database design and implementation.
- User profile management.
- User Authentication and Registration.
- A working UI skeleton + users can register and log in successfully.

Sprint 3 (15.09.2025 - 28.09.2025)

- Post creation and feed management.
- Friends/Followers management and messing.
- Implement business logic and service layers to handle requests from the UI.
- Applying Jenkins to the project.

Sprint 4 (29.09.2025 - 09.10.2025)

- Docker setup.
- Refining the user experience.
- Extensive testing of the application.
- Writing the documentation for the application.

Resource Allocation

Team members & Roles:

- Tony Karlin
- Jarkko Kärki
- Onni Kivinen
- Joni Heikkilä

All members are equal and share the work together. Everyone contributes to design, coding, testing and documentation. Task allocation and progress tracking are managed collaboratively through Trello, where each member selects their next tasks during sprint planning or as work becomes available.

Software/Tools: Java, JavaFx, Github, Trello, Jenkins, Docker, MariaDB/HeidiSQL, JUnit, Figma, SceneBuilder

External Resources: course materials, tutorials, possible teacher guidance.

Risk Management

Risk Description	Likelihood	Impact	Mitigation Strategy
Voice communication. Implementing a working voice communication channel.	**High**	**Medium**	The strategy to mitigate the lack of voice channels is to ensure the application works seamlessly with text-only messaging, shifting focus to voice features **ONLY** if time allows.
Security. General security risks, including authentication, rate limiting, SQLInjections.	**Medium**	**High**	The strategy is to implement sufficient security from the start, including strong authentication, input sanitization, and rate limiting, to protect basic users.
Bugs and Errors near project deadline.	**High**	**High**	Test all critical features extensively from the beginning. Make an effort to leave enough time before the deadline to debug the code as a team.
Unrealistic **timelines** and changing project **requirements**.	**Low**	**High**	Prioritize important features and allocate enough resources to said features. New features must be discussed and evaluated as a team before moving forwards.

Testing and Quality Assurance

- **Testing types:** Unit–tests, integration tests, user acceptance testing.
- **Success criteria:**
  - Every component works without critical bugs.
  - The user experience is simple and clean.
  - The application performs efficiently on low-end systems.
  - The code passes all unit, integration and user acceptance tests.
- **Tools:** JUnit

Documentation

We will prepare a structured set of documentation throughout the project lifecycle. For example, a design document will be created to outline the project structure, which will include details such as the database schema, class design and entities, application UI elements, and deployment processes.

When the application becomes deployable, more in-depth documentation will be released. The later documentation will include diagrams, code reviews, testing procedures and the user stories that were followed to develop the application. Diagrams will illustrate the database relations, class structure and the use cases of the application.

The project will have documentation and reports for each completed sprint with planned biweekly meetings with the product owner. The sprint review documentation will include the planned workload for the upcoming sprint and track the progress of each team member with completed tasks, tasks that are in progress and potential postponed tasks.