Group Name: Streamline Coders

Group members: Drew Burton, Daniel Noguera, Arjay Lalonde, Kyla Rigby, Amanda Malek, Hiarimino Ralison Rakotoson

Project Idea: A chat messaging app. A login screen allows users to use a chat interface with users around the world. The users will be able to create profiles, search profiles by name, send friend requests, block users, send text, and create group chats. The server will use an API and a database to store these messages and serve it to any web client as well as handle authentication.

1. Introduction
   1. Purpose
      * Our purpose is to create a chat messaging app that would allow users to chat with other users in both direct messaging and group chats.
   2. Document Conventions
      * DB: Database
      * DBA: Database Administrator
      * ER: Entity-Diagram
      * JDBC: Java Database Connectivity
   3. Intended Audience and Reading Suggestions
      * Given how the project would function as a prototype, as it currently is it would be within the premises of the university under guidance of our professor. If brought into reality, however, it would in theory target those who wish to use a messaging app
   4. Project Scope
      * Given the nature and purpose of this project, as well as what the web app will be based on, the scope of this project would be to serve up to one hundred users by the end of the allocated time of two semesters while also providing a good user experience. If given the time, we will go beyond that initial goal of 100.
   5. References
      * https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database
2. Overall Description (Daniel)
   1. Product Perspective

Our project would provide a simple messaging application that would facilitate near-instant messaging from around the globe. It would allow users to create accounts, search for their friends, and block users. Additionally, it would not require any installation. The product would serve a similar purpose as slack/WhatsApp/discord but would not require a download.

* 1. Operating Environment

Since java is multiplatform, the server itself should be able to run on nearly any modern machine. The best machine would be a fully updated windows or Linux machine connected to the internet with a firewall. The environment will require enough ram for the program and enough storage to accommodate the DB. The operating environment of the client would not matter as long as they had access to the internet and a browser.

1. System Features (Kyla)
   1. Functional Requirements

Our chat messaging service requires user authentication and permission, which allows users to safely establish accounts, log in, and manage profiles. The messaging system will allow real-time communication, ensuring that messages are promptly sent and displayed across devices in sync. Users should also be able to send a variety of message formats, such as text, multimedia (images, videos, and files), and embedded links, as well as responses (such as emoticons) and threaded conversations to keep the context in bigger debates.

A sophisticated search tool is required, allowing users to search messages, files, and channels by keyword, person, or date. To manage permissions, role-based access control should be created, giving workspace owners and administrators control over user access, channel management, and content moderation. Furthermore, the app should be compatible with third-party tools and services, allowing users to get updates and take actions within the messaging platform, such as organizing meetings, tracking tasks, or integrating cloud storage. For usability, the software must support desktop, web, and mobile platforms while providing a consistent user experience across all devices.

1. External Interface Requirements (Drew)
   1. User Interfaces

We are choosing to implement the user facing software as a ReactJS driven web app which allows us to develop components to dynamically display user messages and the other elements of the application such as the side bar. Depending on the SQL database chosen, DBAs can interface with the data using the associated software.

* 1. Hardware Interfaces

A browser that supports HTML and JavaScript and an external cloud server that it will communicate with. Any operating environment (Mac, Windows, Linux, etc.) will be supported as long as the browser requirements are met.

* 1. Software Interfaces

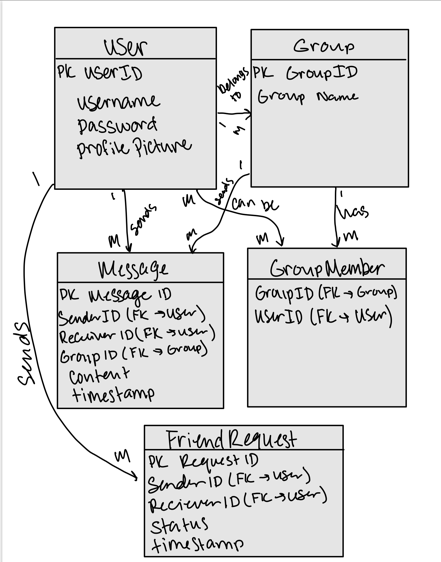
The web app with interface with the backend server using REST API communications using Spring security to authorize requests and send over chat and user data. The backend server will use JDBC to securely connect to the DA

* 1. Communications Interfaces

The web app will use HTTP as a communication method to the API. If possible, the content of the request will be encrypted, so no user can decipher hijacked messages. Deployment software will handle scaling the number of instances of the backend server in the case of high traffic.

1. Non-Functional Requirements
   1. Performance Requirements

ER Diagram:



Normalization:

The database will be normalized in order to reduce redundancies. Relationships between the entities will be maintained in a way that prevents unnecessary duplication.

* 1. Safety Requirements
     + The application will maintain a robust backup and recovery strategy to minimize data loss in a server crash. This will involve regular backups of the DA, with the most recent backup being the restored state.
  2. Security Requirements
     + Communications will be encrypted in order to safeguard user data and user authentication will be enforced by the use of a username and password being required to login. Also, the DA environment will be kept secure to prevent unauthorized access and protect client data.
  3. Software Quality Attributes
     + Availability: The application will be available to users at all times, with the exception of when there is scheduled maintenance.
     + Correctness: Messages will be sent to the intended recipient without experiencing any kind of loss or duplication. User profiles will be maintained to reflect real-time changes for users to friend/block/message the correct individual to prevent confusion.
     + Maintainability: The application will be easy for developers to maintain and will support regular updates
     + Usability: The application will be user-friendly, with features being easy to find and use