Initial Report of a Distributed Chat Application

Team Zero members:

Zulean, Tiberiu Iustin
Darwish, Emad Mahmoud Nagui
Esener, Iris
Li, Zhuoling

November 18, 2019

Part 1 - Project Description

1.1 Aims

1. Priority 1
   - 1.1. Clients register with the server - client details are saved in the database.
   - 1.2. Registered clients can login, and once authenticated can do the following:
     - 1.2.1. Clients can search for other clients in the system, also see the status of other clients (e.g. online, offline and busy).
     - 1.2.2. Clients can fetch their contact list (groups and friends) and chat list.
     - 1.2.3. Clients can initiate a chat with and send messages to other clients, via the server.
     - 1.2.4. Clients can receive messages from other clients, via the server.
       - 1.2.4.1. If a client was to receive a message while they were offline, the message will be sent on login (queued messages).
2. Priority 2
   - 2.1. Group chat functionality - a client can initiate a chat and send messages to a group of more than one client.
     - 2.1.1. The recipients will automatically be placed in the group chat until they choose to leave.
   - 2.2. To send an email on registration to validate email address
3. Priority 3 (implemented last)
   - 3.1. End to end encryption - to be implemented on the clients, so the server cannot intercept or decrypt messages.
     - 3.1.1. Public key encryption to determine shared keys between 2 clients.

3.1.2. Symmetric encryption with shared keys to encrypt and decrypt messages between 2 clients.

3.1.3. Keys to be kept in clients local storage only.

3.2. Message storage and retrieval - chat information is stored in the database, and history can be retrieved on request.

3.2.1. Text messages can be of arbitrary size.

3.2.2. Recent messages (e.g. messages sent within the last week) cached locally for quick request.

4. Priority 4 (optional)

4.1. User and group avatars/pictures

4.2. Multimedia messages - besides plain text, message such as pictures, emoji and voice record can be sent.

4.3. Message status - clients can see if their messages are received or read and also see typing status of other clients.

4.4. QR code for adding new friends - rather than search by user name, clients can find another clients quickly after scanning their QR code.

1.2 Progress so far (November 18, 2019)

We have chosen to follow an Agile software development approach, with weekly meetings to evaluate tasks. So far we have determined the architecture for our system and database (may be subject to minor changes in future iterations), and implemented the bare bones of server and client applications.

Figure 1 below shows the structure of the database that is stored server side.

1.2.1 Web app

The web app has been implemented with Vue.js for the UI, and javascript embeded inside html pages. The app has a RESTful server with SpringBoot that is responsible for mapping and routing. For individual clients to interact with the main application server, they fetch the web app from the RESTful web server, and then communicate with the main application server in JSON messages using javascript's web sockets.

1.2.2 Android app

The Android application (developed using Java with XML for front-end layouts) has established a Websocket connection test to a remote dummy server and successfully sent and received JSON requests using an internally built JSON construction tool. The app has an SQLite local database attached used to store and retrieve previous client chats. Several functionalities are implemented (registration, login, sign out, unregister, sending messages) and need to be orchestrated with the server. There have been added unit tests for anti SQL injection input validation checks (using regex) alongside instrumented tests for internal tools check.

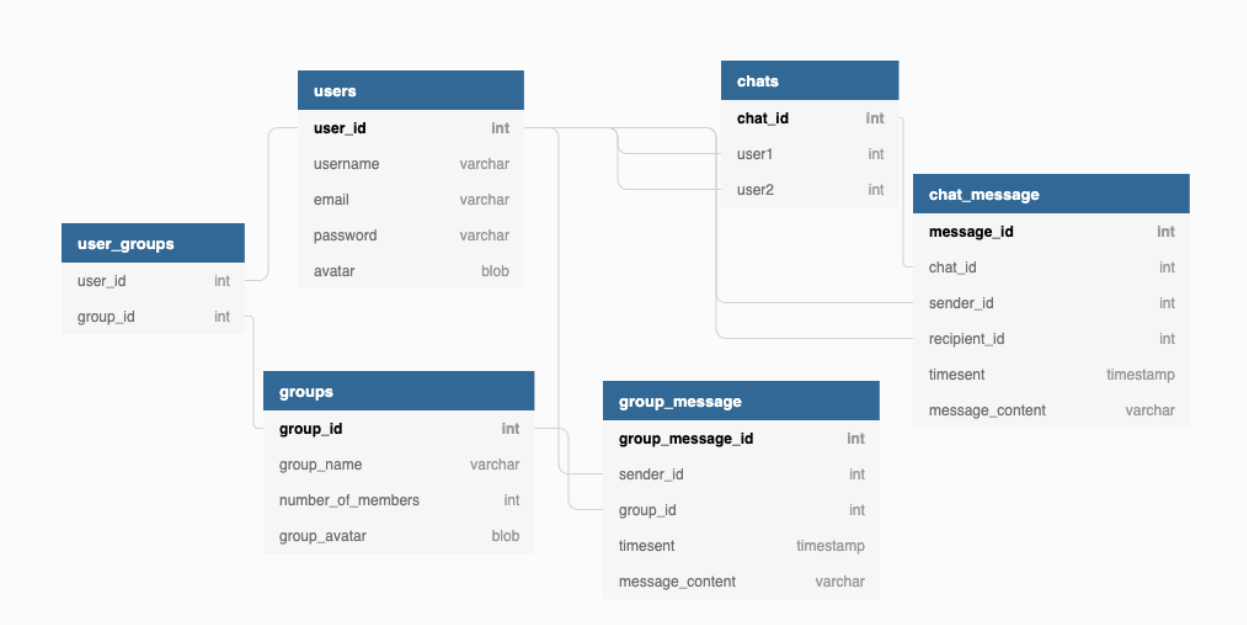

Figure 1: Database structure

1.2.3 Server

The main server, implemented in Java, can currently receive REGISTER, LOGIN, TEXT, and UNREGISTER requests from clients in a JSON format via a web socket. The server appropriately responds to these requests by creating a user in the PostgreSQL database, checking login details, sending the text message to the recipient user (storing for future retrieval if they are offline), or deleting the user from the database.

1.2.4 Deploying to Heroku

We chose to use Heroku as a platform to deploy our server and database. So far, the database (in PostgreSQL) has been deployed to Heroku.

In the coming weeks we intend to deploy the application server as well so that full integration's tests can commence.

1.3 Schedule

Our current schedule estimates are as follows:

Priority 1 aims - Complete by November 30, 2019

Priority 2 aims - Complete by December 14, 2019

Priority 3 aims - Complete by January 31, 2020

Priority 4 aims (if time) and final testing/debugging - Complete by February 29, 2020

Part 2 - Project Organization

2.4 Team collaboration

Methods of team collaboration:

- Weekly meetings for discussion and planning (occasional sub-weekly meetings for pair programming and brainstorming ideas)
- Whatsapp group chat for regular daily communication
- Google docs for document draft and brainstorm collaboration
- Trello for task management
- Git and GitHub for software development version control
  - GitHub Wiki for relevant Client - Server communications information

2.5 Member roles

Although all members are free to help out in any area of development, we have taken on general roles along the following lines.

- Zhuoling Li: Web client application
- Tiberiu Zulean: Android client application
- Iris Esener and Emad Darwish: Server application and database development / deployment

2.6 Handling of conflicts

Having agreed to equally share the allocated points for the project, we believe all members are equally invested in the quality of the project. All members of the team have agreed that should there be a conflict, we will resolve it peacefully with each other and take democratic votes when required. In the case that it is absolutely necessary, an outsiders opinion may be sought.