Final Report - Messaging App

Hugo Garrido-Lestache

May 12, 2025
CSC 5201 301

Introduction

For my Final Project, I created a messaging app that allowed users to send messages to each other. This app was a web app with the abilities to send direct messages, group messages and create group chats. It also handled users login and registration. I thought this service could be a fun project to practice my skills, which could then be used to message my friends and family. This service was designed to be similar to messaging apps such as WhatsApp or Microsoft Teams.

1 Use cases

The application was developed to support the following use cases, many of which were only thought about after the initial implementation.

- User login and registration - Users needed to be able to login and register to use the app.
- User logout - Users needed to be able to logout of the app.
- List all users - Users needed to be able to see a list of all users.
- List all messages - Users needed to be able to see a list of all recent messages.
- Send direct messages to other users - Users needed to be able to send direct messages to other users.
- Create group chats - Users needed to be able to create group chats.
- Send messages to group chats - Users needed to be able to send messages to group chats.
- Add users to group chats - Users needed to be able to add users to group chats.
- Remove users from group chats - Users needed to be able to remove users from group chats.

2 Business context & Services

Our business was separated into 3 services:

- User service - This service was responsible for handling the user login and registration. It was responsible for managing the data on users in the database and ensuring that the users were authenticated.
- Message service - This service was responsible for handling the messages. It was responsible for managing the data on messages in the database and ensuring that the messages were stored and retrieved correctly.
- Group chat service - This service was responsible for handling the group chats. It was responsible for managing the data on group chats in the database and ensuring that the group chats were stored and retrieved correctly.

3 Technical Implementation

The backend of the messaging application leverages three Python-Flask microservices with MySQL databases:

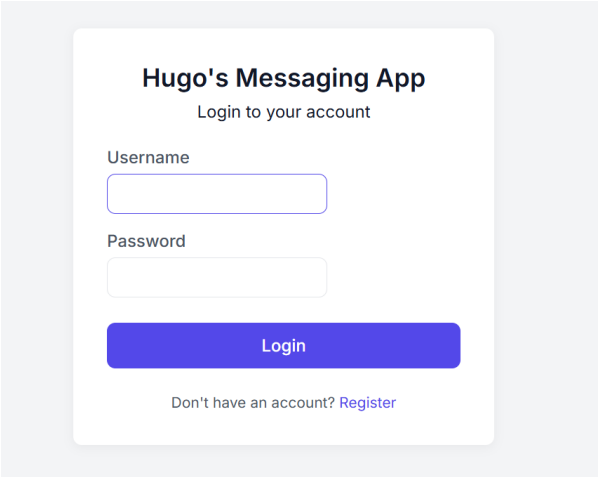
- **User Service**: Manages a `users` table with user credentials and profile information. Handles authentication, registration, and user profile queries.
- **Message Service**: Maintains two tables: `messages` for direct messaging and `group_messages` for group communications. Stores message content, sender/recipient IDs, and timestamps.
- **Group Chat Service**: Controls the `groups` table containing group metadata, including name, creator, and a JSON array of member IDs.

Inter-service communication occurs via direct API calls. An example of this is that when displaying conversations, the Message Service queries its database for message content but calls the User Service to retrieve usernames. Similarly, the Group Chat Service validates members by checking with the User Service. The application is deployed using Kubernetes for orchestration, enabling efficient load handling and service scaling. A future enhancement would be implementing message queues between services to improve comunication between them, when at a larger scale such that each service can have multiple instances. Please see the link to the repo for the code.

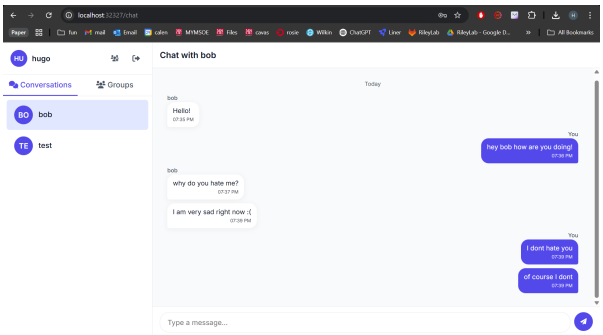
4 What I Delivered

I managed to implement all the services and endpoints to match the use cases. I realized that I had to implement extra endpoints to deal with handling the cookies and sessions when in the web app. I also realized that I had to implement endpoints to get user info, for the other services to use. Throughout creating this I settled on less traditional design of having two ways of sending messages of direct and group messages. These each had their own sections in the frontend and own endpoints, this separation helped me to implement the services easier but was not ideal.

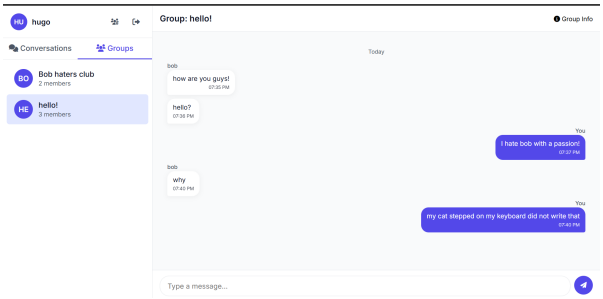
Below are some screenshots of the app in action:



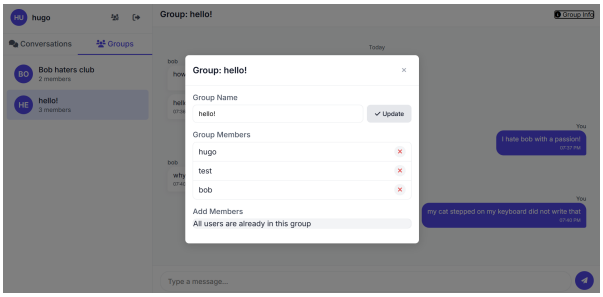
(a) User login



(b) Direct message + Home page



(c) Group chat + Home page



(d) Group chat + edit group chat

5 What I Learnt

I learnt a lot about working with Docker and managing an application with multiple services. Ensuring that the services were able to communicate with each other was challenging (passing their own services as a parameter to the other services). It was intresting having them all work together to complete the overall goal.

I also learnt a lot about Kubernetes and how to deploy an application with it. It was intresting to think abou the common data issue and having to replace that with inter-service communication.

6 What I would do if I had more time

I would not do much different, as in the end this project was more for fun and practice. If I wanted to get this project to a more useable state, there would be many features I would add. Such as security features, such as password hashing, testing scalability, and adding more features. I would also want to be able to add the ability to send files and images, and also add a media service to store them. Handeling notifications would also be a feature I would add, so users could know when they had a new message.

However if I had more time, I would likely work on a different project, as developing a messaging app is not very usefull which so many other services already providing this. I would want to create something more novel.

7 Conclusion

In conclusion, I enjoyed working on this project and it was a fun way to practice my skills. And I think I have learnt a lot from it. Goodbye!

8 Link to repo

Link to repo

https://github.com/Hugogales/microservices-final-project