Name: Cyril Ayegbe

NSID: cma595

Student Number: 11279069

Instructor: Ralph Deters

Course: CMPT 353

Course Section Number: CMPT-353-02

The architecture of the database consists of a MySQL database running on a Docker container. The database stores messages that belong to different channels. Each message has its unique identifier (a hash function that generates a value based on userID, channelID and time posted) message content, and relevant such as timestamp, upvotes, and downvotes. The schema was designed with normalization in mind to minimize data redundancy and maintain data integrity.

The backend of the application is built with Node.js and Express.js, which provides the RESTful API for interacting with the database. The API has endpoints for getting all messages, creating a new message, updating message vote counts, and deleting a message.

The frontend of the application is built with React.js, which allows for dynamic rendering of components. The user interface is designed using Bootstrap framework and includes a channel selection sidebar, message list, and message input box. The user can select a channel from the sidebar, and messages belonging to the channel are displayed in the message list. The user can create a new message and vote for existing messages. The frontend communicates with the backend through RESTful API endpoints and uses Axios library for making HTTP requests.

The application was meant to use Socket.IO library to provide real-time updates and chat functionality across users on the server (this was not implemented).

Overall, this architecture provides a scalable and secure solution for a multi-user chat application with real-time updates.

Partially Implemented Features

The controller logic for the search features and loading of channel messages have been written but not implemented on the front end side. As mentioned previously, fluid chat functionality was also not implemented.