NEW YORK UNIVERSITY

CS9053 Project Report

Project Title:

JavaFX Chat Application

Team Members:

Koushik Ram Manoj Kumar (km6078), Meghna Manoj Nair (mm13032), Ramyaa Prasath (rp3975)

Project Overview

In the realm of software development, understanding the intricacies of Java, from its syntax to advanced features like GUIs, Threads, Networking, and Databases, forms a fundamental part of a developer's education. The project we have developed is an application that focuses on some of these concepts through a cohesive and functional Java-based chat application. This project report delves into how the project adheres to the requirements set forth for a the final project, emphasizing UI development, the integration of Java Frameworks, and the application of Java in networking and database management. It details a client-server chat application developed using Java technologies, with a focus on GUIs, networking, and multithreading. The application facilitates real-time communication between users through a graphical interface, supported by a robust server handling concurrent client connections.

What the application looks like?

The application is structured into two main components: the client and the server. The client is responsible for presenting the user interface and handling user interactions, while the server manages network communications, client connections, and the central logic for message exchange.

→ *User Interface (UI)*

The client-side of the application features a modern, user-friendly graphical interface built using JavaFX, which is Java's toolkit for building rich internet applications. The UI includes:

- ✓ Login Screen: Allows users to enter their username to join the chat.
- ✓ Chat Window: Displays the conversation in real time, where users can send and receive messages.
- ✓ User List: Shows all users currently connected to the chat.
- ✓ Notifications: Visual and auditory notifications for new messages or user activities.

The design elements include custom icons, buttons, and chat bubbles, creating an engaging and interactive user experience.

→ *Main Functionalities*

- ✓ Real-Time Messaging: Users can send and receive messages instantly with other connected users.
- ✓ User Sessions: The server tracks each user's connection and session, ensuring messages are routed appropriately among users.
- ✓ User Status Updates: Users can update their status to "Away," "Busy," or "Online," which is visible to other users, helping indicate their availability.

- ✓ Concurrency Handling: The server can handle multiple users simultaneously without performance degradation, utilizing multithreading to manage separate user interactions.
- ✓ Network Communication: Uses Java sockets for networking, enabling robust client-server communication. Data packets containing messages or commands are sent and received over the network.

Core Java Principles and Technologies Incorporated

- ***JavaFX for GUI:*** The client interface is developed using JavaFX, providing a rich interface that is both functional and aesthetically pleasing.
- ***Java Sockets for Networking:*** Both the client and server use Java's networking capabilities to establish connections and communicate. The server uses ServerSocket to listen for incoming connections, while the client uses Socket to connect to the server.
- ***Multithreading:*** Essential for handling multiple client connections and maintaining a responsive UI. The server uses threads to manage each client connection individually, allowing for simultaneous processing of client requests.
- ***Maven for Dependency Management:*** The project uses Maven, indicated by the presence of pom.xml files, which helps in managing project dependencies, building processes, and configurations.

Conclusion

This chat application showcases the effective use of Java technologies to create a scalable, responsive, and user-friendly real-time communication platform. The application leverages JavaFX for its GUI, Java sockets for networking, and multithreading to handle multiple users efficiently, making it a comprehensive example of a client-server architecture in Java.

The following paragraph details the distribution of work among the team members, highlighting their specific contributions to the project. Ramyaa and Meghna focused on the front-end, where they crafted a visually appealing user interface that's both functional and intuitive. They also worked on integrating the status update features and collaborated on compiling the project documentation and drafting the final report. Koushik took on the backend, ensuring seamless integration and robust testing of the system. He was responsible for developing test cases, debugging, and fine-tuning the performance and security of the application. This balanced distribution of tasks ensured that all team members were actively involved in both the technical and creative aspects of the project, leading to a high-quality final product.