Name: Palak Pramod Keni (pk2539), Jiayu Liu (jl13683)
Professor : Dr. Dean Christakos
Subject: Introduction to Java
15 Dec. 2023

JAVA FINAL PROJECT

PROJECT TITLE: One Room Chat App
PROJECT DESCRIPTION: This project aims to develop a real-time multi-user chat application with a single chat room that allows users to communicate with each other. The application will have a **React**-based front-end, a **Java Spring Boot** back-end with a **PostgreSQL** database and **Kafka** as streams processing component.

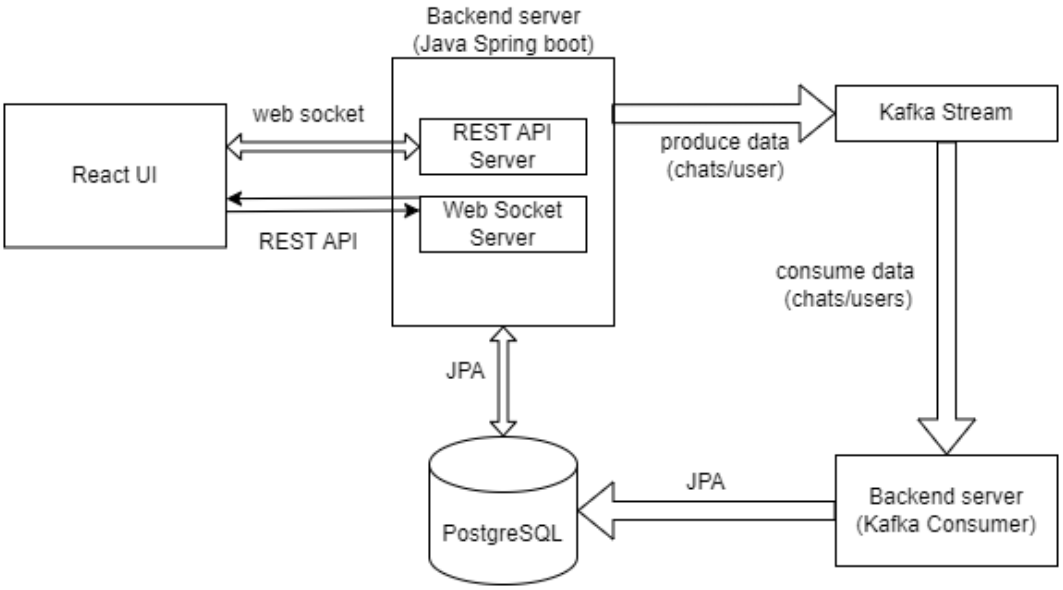
PROJECT FEATURES:

- User registration and authentication
- Dashboard for multiple users to login into a single chat room
- Send and receive messages in real-time in the chat room using web sockets
- PostgreSQL database to store user details and chat messages
- Kafka server for queueing messages into the database
- React front-end server with robust UI/UX
- Spring Boot REST APIs for front-end to interact with back-end

JAVA ADVANCED FEATURES:

- Java Spring
- JDBC
- Message queues
- Web Sockets

ARCHITECTURE:



The application will follow a layered architecture with the following key components:

Front-end: The React front-end will be the user interface for login and signup as well as the messages dashboard. The chat dashboard is connected with web sockets for a bi directional client server real-time communication. The login and signup pages use REST API endpoints to connect to the back-end for user authentication registration and authentication. The previous chat history is also fetched to the front end with REST API calls.

Back-end:

The first back end server is a Java Spring Boot app with REST API server and web sockets server. This server also acts as a producer for the Kafka stream (i.e. to put things to the queue waiting for future operation). It also connects to DB directly but will only fetch data for the frontend to display and user authentication. The main purpose of this server is to perform fetch operations, handle web socket connections and produce other requests in the kafka queue.

The second backend server is also a Java Spring Boot app and will act as a consumer to consume from the kafka stream. The main purpose of this server is to perform CRUD operation in the database.

Streams Processing: The Kafka stream server is responsible for queuing jobs the first server produces, and waiting for the second server to consume the jobs and put things to the database. It happens asynchronously to make the system more robust when there's a high volume of requests, for reliability and scalability.

Database: PostgreSQL database is used to persist the data of the application. There are two tables in the database (one_room_chat_app) used for this application: users to store user details and chat to store the chat messages.

FILES IN THE SUBMISSION:

- finalDemo.mp4 (Video demo)
- Java-Project-Documentation.pdf (Current documentation)
- oneRoomChat (code base which contains all the code)
    - one-chat-room-app-ui (Front end code)
    - oneRoomChatAppKafkaConsumer (kafka consumer server code)
    - one-room-chat-application (backend server code)
    - sql-script (PostgreSQL DB scripts)
    - docker-compose.yaml (docker-compose script to compile and run code in docker containers)
- oneRoomChat-x86 (script to run code in windows or linux (non-ARM based systems))
    - sql-script (PostgreSQL DB scripts)
    - docker-compose.yaml (docker-compose script to run code in docker containers)
- oneRoomChat-ARM (script to run code in MACs (ARM based systems eg. Mac M1, Mac M2))
    - sql-script (PostgreSQL DB scripts)
    - docker-compose.yaml (docker-compose script to run code in docker containers)

STEPS TO RUN THE APPLICATION:

- Install docker & docker compose in your machine.
- There are two ways to run the code
    - Create docker container in your machine and run the containers
        - Steps
            1. Go to oneRoomChat folder and run the following in a terminal/cmd:
               docker compose up -d
    - Download container directly from dockerhub and run containers
        - Incase of windows and linux (non ARM based systems)
        - Steps
            1. Go to oneRoomChat folder and run the following in a terminal/cmd:
               docker compose up -d

        - Incase of Mac systems (ARM based systems)
        - Steps
            1. Go to oneRoomChat folder and run the following in a terminal/cmd:
               docker compose up -d

We would recommend following step 2 to download docker images directly and run the code as this is relatively less time consuming. Alternatively, you also try to run Step 1, but it will take more time as it will download and also compile.