Scala Chat Application Architecture: Leveraging Microservices and Modern Technologies

Ronak Sengupta

Project Specialist II

Payoda Technologies

Introduction

This chat application is built on a microservices architecture leveraging scala, play-framework, kafka producer and consumer to provide a scalable, reliable and efficient platform for real-time messaging.

Key Features:

- Frontend: Static web app containing signup, login, logout and chat functionality built using Play framework for UI Interaction.

- Backend API: Restful services implemented with Scala and Play Framework to handle user authentication, message routing and more.

- Microservices: Utilizing Play and Akka framework for background task and message processing.

- Kafka: Message broker for asynchronous communication between microservices, enabling scalable and resilient message processing.

- Database: MySQL used for persistent storage of user data and messages.

- Docker: Containerization for easy deployment and scalability across different environments.

Tech Stacks Used

Scala

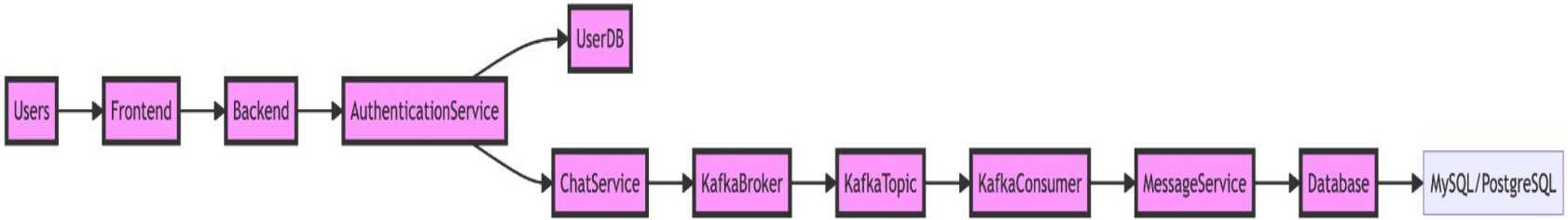
Play Framework

Docker

Kafka

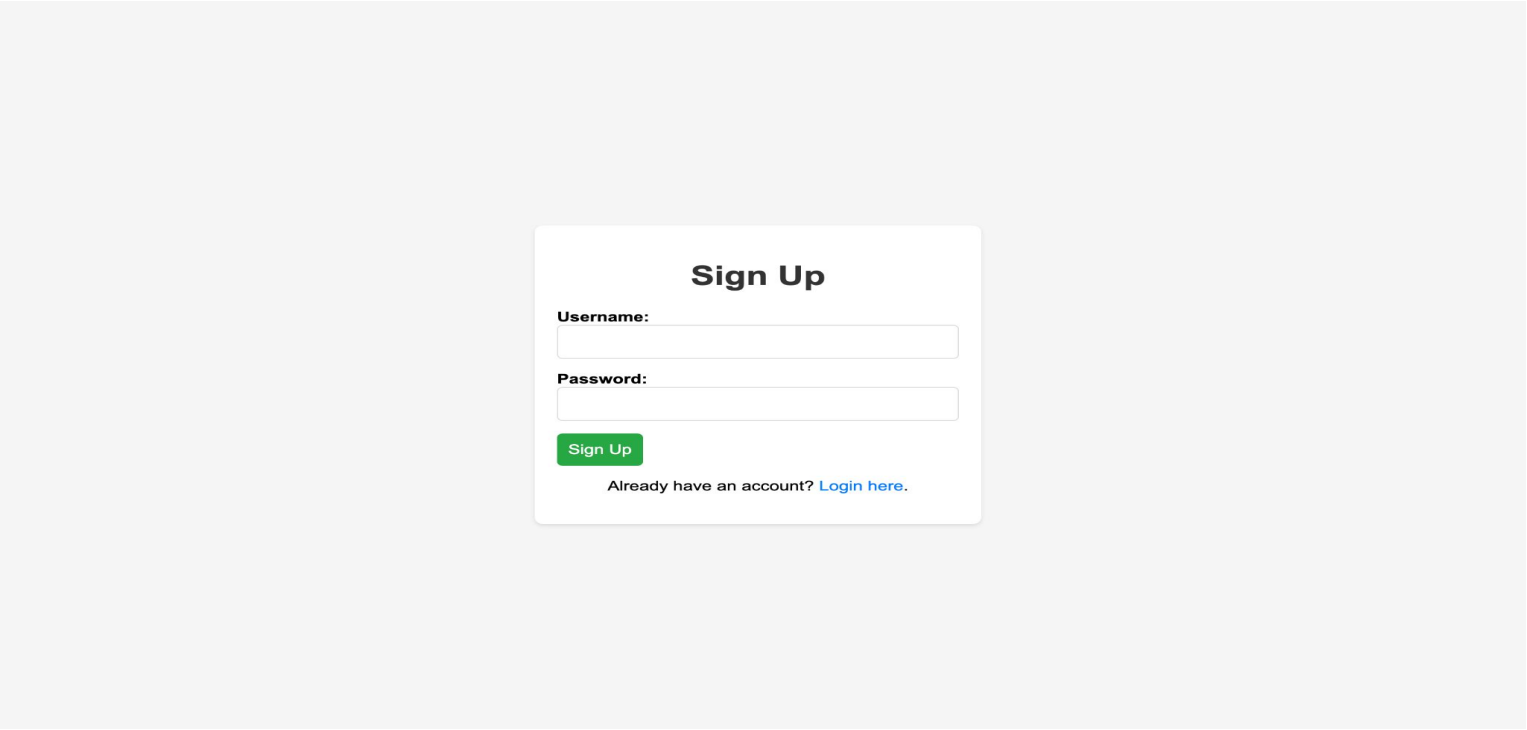
MySQL

Architectural Design



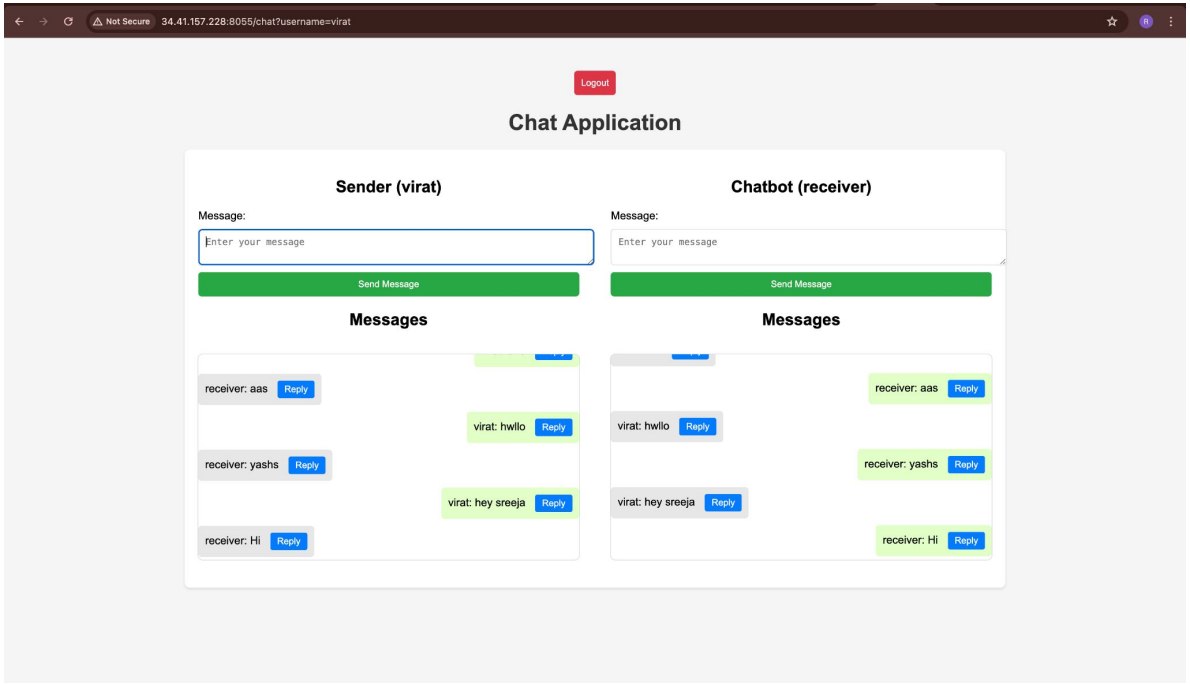
Frontend

Authentication Page



Frontend

Chat Application UI:



Backend Services

Authentication Service:

- It handles user registration and login
- Communicates with user database
- If found the correct user will redirect to chat application page
- If not found then it will show incorrect credentials

Technology used: Scala, Play Framework

Chat Service:

- Manages chat functionality where two users can exchange messages between each other,
- Uses kafka for producing and consuming chat messages.
- Saves chat messages to chat database
- From database messages are fetched and displayed in UI.

Technology used: Scala, Play Framework.

Backend: Kafka Architecture

Kafka plays a crucial role in our chat application architecture, serving as a reliable and scalable message broker for asynchronous communication between microservices.

- Kafka Producer and Consumer: Producers are responsible for publishing messages to kafka, while consumers subscribe to topics and process those messages asynchronously.

- Kafka Broker and Topic(chat-messages): Kafka brokers form the core of the kafka cluster, storing and replicating data across multiple nodes for fault tolerance. Topics represents channels through which messages are organized and distributed.

- Message processing : When a user sends a message, the kafka producer publishes it to the topic. Consumers such as microservices, subscribe to this topic, retrieve messages and process them accordingly.

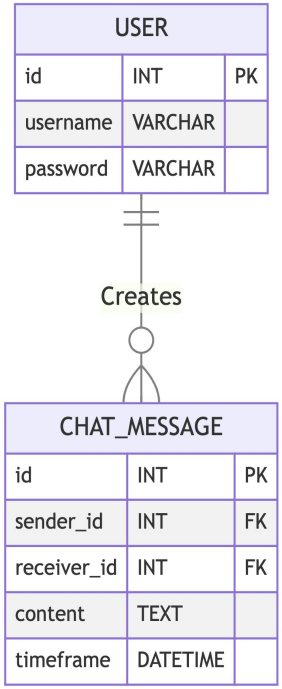
Database Architecture

User Database

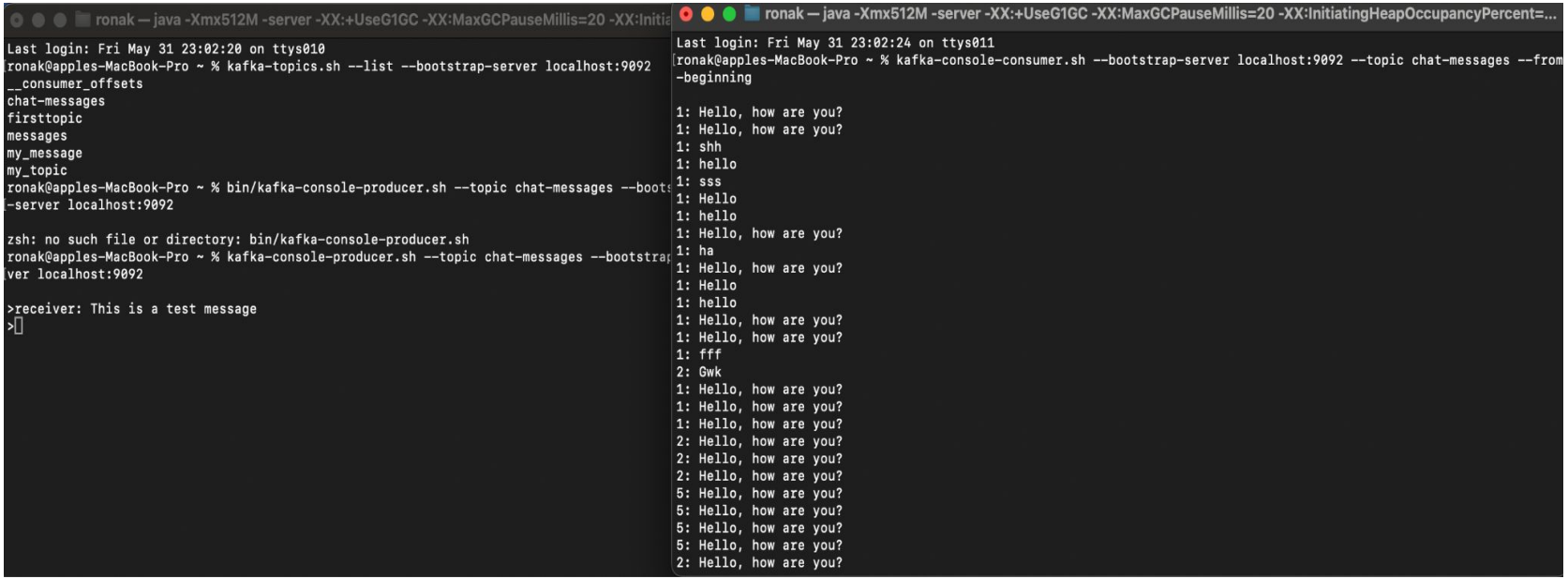
- Stores user information whenever some user sign up

Chat Database

- Stores chat messages which are consumed by kafka consumers.



Snapshots of message processing



Conclusion

My chat application architecture provides a robust and scalable platform for real-time messaging, enabling seamless communication between users.

- Designed a microservice-based architecture, leveraging Scala, Play Framework, Akka, Kafka, and MySQL to ensure responsiveness, fault tolerance and scalability. Each components are crucial in handling user interactions and message processings.

- Frontend interacts with backend API for user authentication and chat message routing. Microservices process messages asynchronously via Kafka. Database stores user data and chat messages persistently. Docker containerization facilitates deployment and scalability.

Future Enhancement:

Possible enhancements includes integrating message encryption and notification system for enriched user experience.

THANK YOU