UNIVERSITY AT BUFFALO

DISTRIBUTED SYSTEMS

PROJECT-2

GROUP-106

Group Members:

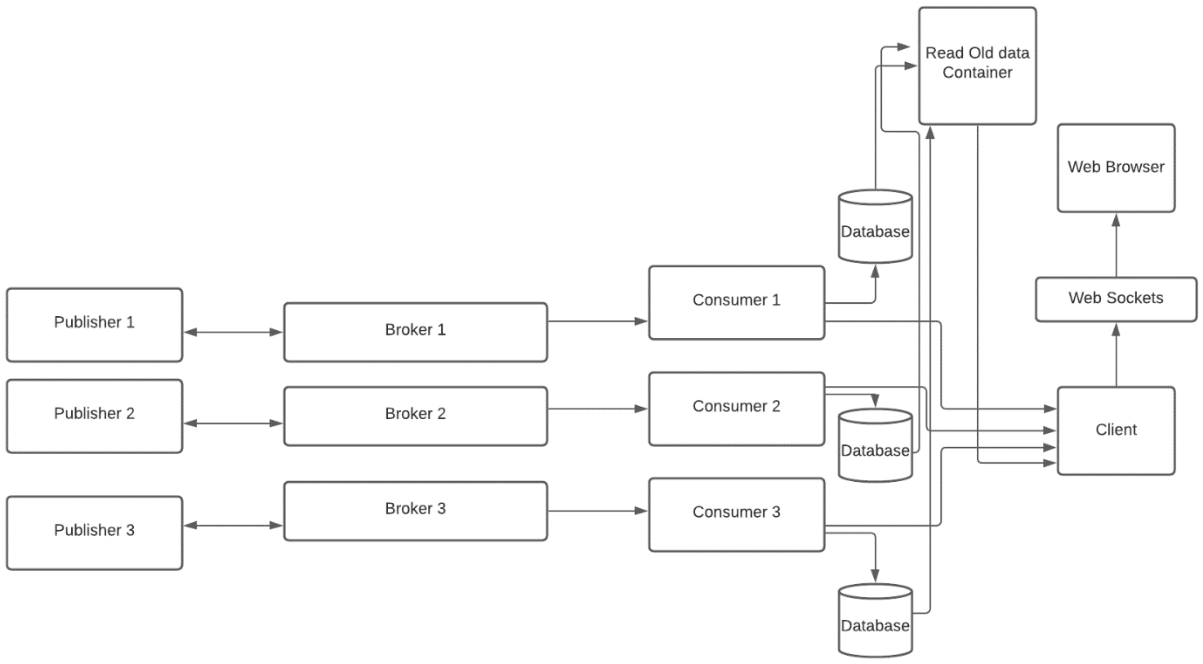
Prashant Kalyani: 50388408
Siddhesh Chourasia: 50415033

DESCRIPTION:

Kafka:

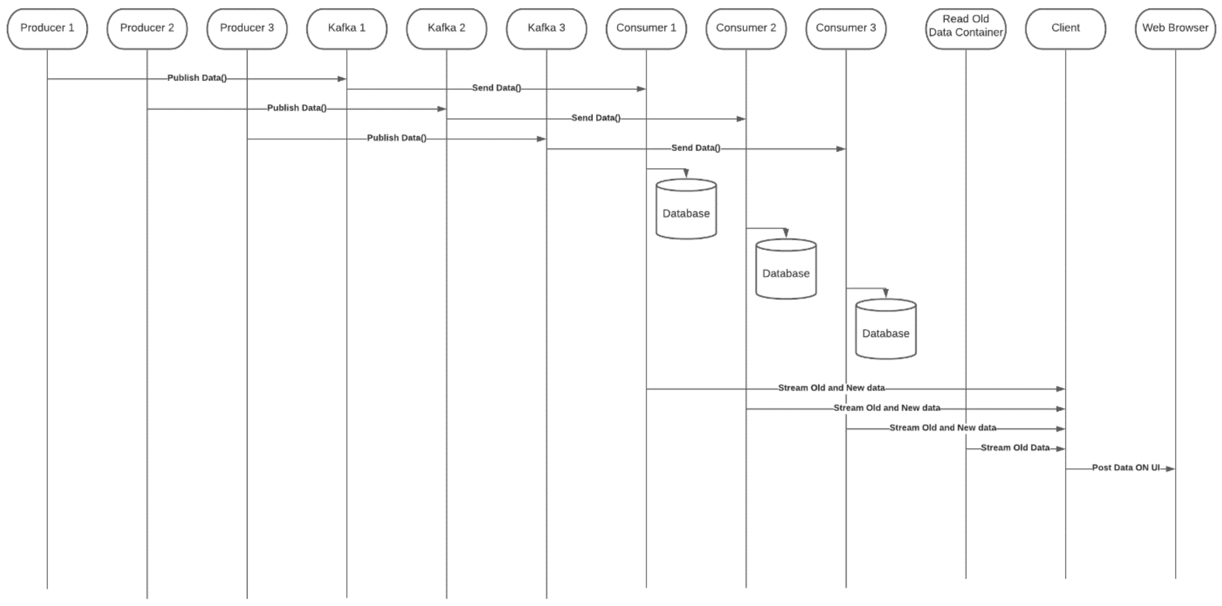
The project uses Kafka Docker images to build a publisher subscriber system Riot Gamin. There are 3 publishers, 3 Kafka brokers, 1 zookeeper, 1 flask web server, and 1 Mongo DB. The subscribers can be created on the fly as they are just HTTP clients. The GUI for the system can be used through the local hosts port 5000 via a web browser. Users have options to follow or subscribe to one or more of the 10 topics of Riot gaming.

Architecture:



ARCHITECTURE DIAGRAM

Sequence Diagram:



<u>**SEQUENCE DIAGRAM**</u>

DESCRIPTION:

Programming Language and Frameworks:

- Python
- Flask
- MongoDB
- Kafka
- JavaScript
- Web Sockets

Dockers:

Docker is an open platform for developing, shipping, and running applications. It enables you to separate your applications from your infrastructure so you can deliver software quickly.

- Docker Image - It is a file, composed of multiple layers, used to execute code in a docker container. They are a set of instructions used to create docker containers.
- Docker containers - It is a runtime instance of an image. Allows developers to package applications with all parts needed such as libraries and other dependencies.
- Docker file - It is a text document that contains necessary commands which on execution

helps assemble a Docker Image. Docker image is created using a Docker file.
- Docker compose - Compose is a tool for defining and running multi-container Docker applications.

Kafka:

Apache Kafka is a stream-processing software platform originally developed by LinkedIn, open sourced in early 2011 and currently developed by the Apache Software Foundation. It is written in Scala and Java.
Kafka is a distributed system that consists of servers and clients.

- Some servers are called brokers and they form the storage layer. Other servers run Kafka Connect to import and export data as event streams to integrate Kafka with your existing system continuously.
- On the other hand, clients allow you to create applications that read, write and process streams of events. A client could be a producer or a consumer.
- A producer writes (produces) events to Kafka while a consumer reads and processes (consumes) events from Kafka.

Producers: We have 3 producers in our Riot Gaming System. These producers generate the data and pass it onto the respective Kafka broker.

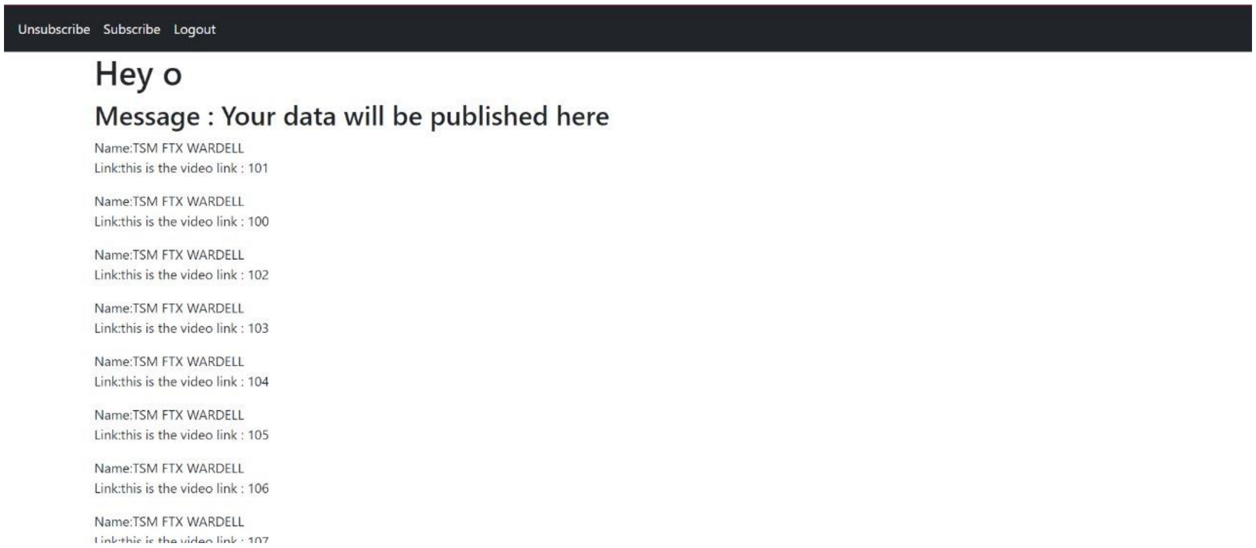
Kafka Brokers: There are 3 Kafka brokers in our system. We have kept the default replication factor to 2 and the number of partitions also to be 2.

Consumers: The data from the Kafka brokers is streamlined to the respective consumer nodes. The consumer nodes in our system checks whether the data coming from the kafka nodes is new or old one by checking the old data container where all the data is stored. If the data is new, then it passes the data to the client server.

Client: The Client is basically responsible for passing the data to the web browser using Web Sockets.

Old Data Container: The only purpose of this data container is that when the user logs out of the system and when it logs back in the system, then it will still be able to see the old posted data.

UI Screenshots:



Hey o

Message : Your data will be published here

Name:TSM FTX WARDELL
Link:this is the video link : 101

Name:TSM FTX WARDELL
Link:this is the video link : 100

Name:TSM FTX WARDELL
Link:this is the video link : 102

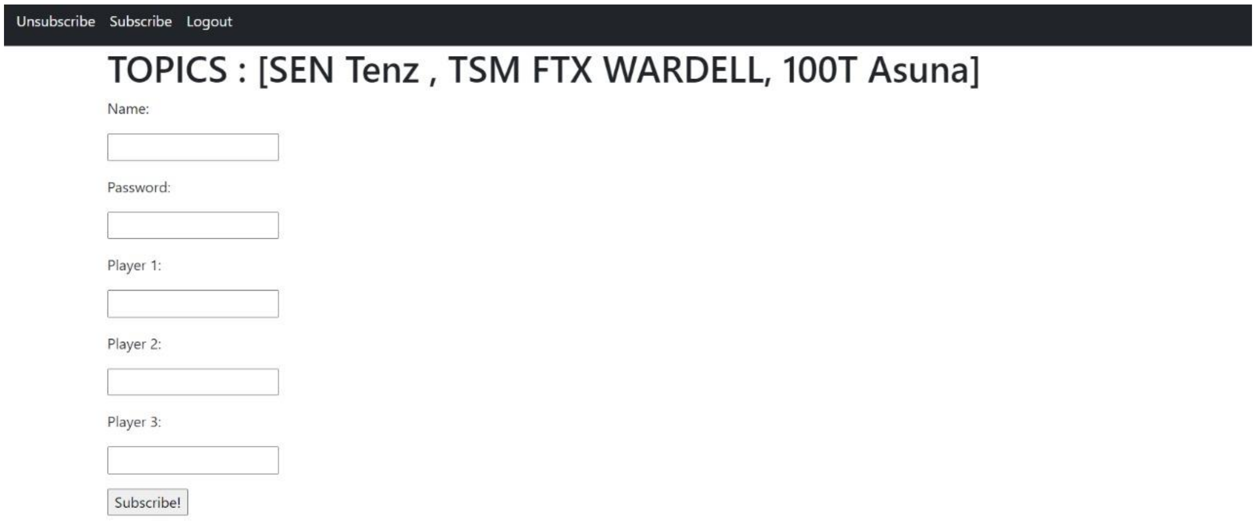
Name:TSM FTX WARDELL
Link:this is the video link : 103

Name:TSM FTX WARDELL
Link:this is the video link : 104

Name:TSM FTX WARDELL
Link:this is the video link : 105

Name:TSM FTX WARDELL
Link:this is the video link : 106

Name:TSM FTX WARDELL
Link:this is the video link : 107



TOPICS : [SEN Tenz , TSM FTX WARDELL, 100T Asuna]

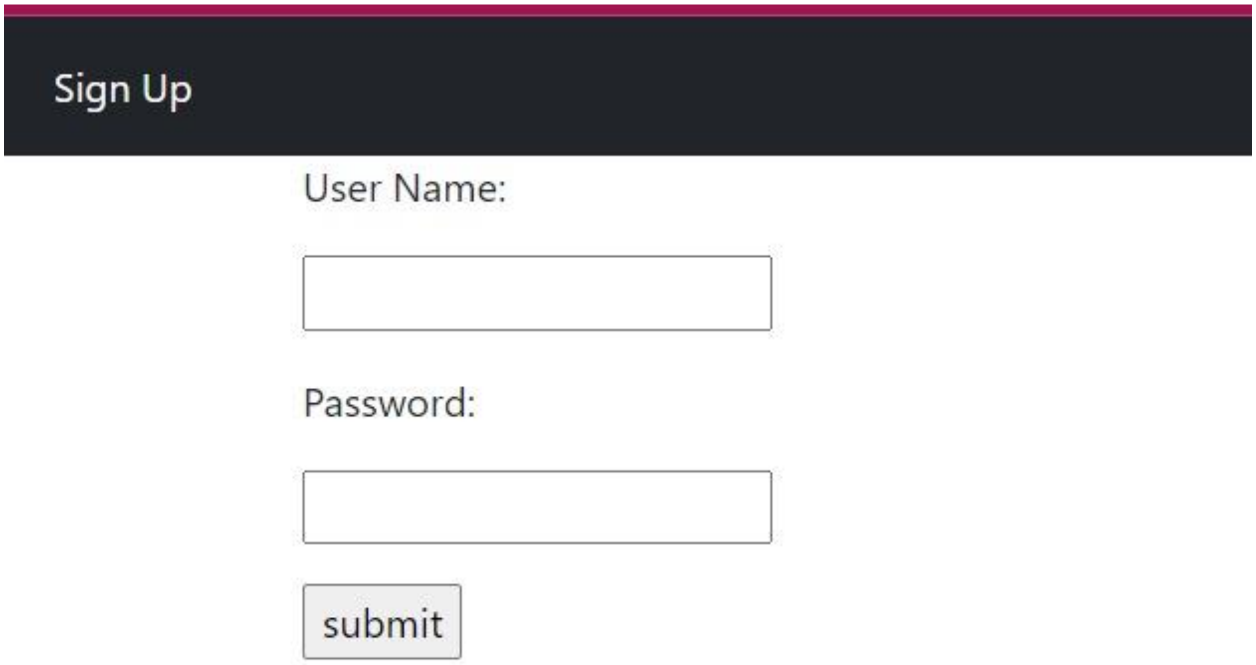
Name:

Password:

Player 1:

Player 2:

Player 3:



Steps to instantiate:

- Execute docker compose build
- Execute docker compose up
- Go to the web browser and enter **localhost:9095**

Contributions:

- **Prashant Kalyani (50%)**
- **Siddhesh Chourasia (50%)**