# Project Proposal: Distributed Video Streaming App with Kafka

1. Project Title:

Distributed Video Streaming App with Kafka

## 2. Objective:

The primary goal of this project is to develop a distributed video streaming application that allows users to upload, store, and stream videos efficiently and in real-time. The system will leverage Apache Kafka for streaming data and Minio as a distributed object storage solution to handle large volumes of videos, ensuring scalability, fault-tolerance, and performance.

## 3. Problem Statement:

With the rapid growth in user-generated content, especially videos, existing video platforms struggle to provide real-time streaming for large datasets in a scalable manner. The need for a system that can handle high throughput, low latency, and real-time video streaming is essential for modern video services. Our project addresses this need by utilizing Kafka for data streaming and Minio for distributed storage.

4. Proposed Solution:

This project will create a distributed video streaming system that:

• Allows users to sign up, log in, upload, and stream videos in real-time.

• Admins will have control over users via an admin panel.

• The application will provide fault-tolerant storage for video content using Minio and stream processing using Kafka.

• Scalable architecture to handle high video upload and streaming traffic efficiently.

## 5. Key Features:

• User Authentication: Users can create accounts and sign in.

• Video Uploading: Authenticated users can upload videos in MP4 format.

• Real-time Video Streaming: Stream uploaded videos using Kafka and provide users with video stream URLs.

• Admin Dashboard: Admin users can manage users and videos.

• Scalable & Fault-tolerant Storage: Using Minio, the system will ensure scalability and reliability for video storage.

## 6. Architecture Overview:

The architecture consists of the following key components:

1. Client-Side Application: Users can upload videos via a web interface built with HTML, CSS, and Vue.js, with backend support via Spring Boot.

2. Kafka-Upload Event: Upon uploading a video, a Kafka producer sends an event with details of the video’s location in the Minio object store.

3. Video Streaming: A Kafka consumer processes the upload event, converts the video into a streamable format, and stores it in a worker database.

4. Minio Object Storage: The system stores video files as byte objects in Minio, ensuring scalability and fault tolerance.

5. Backend: Developed with Java and Spring Boot, the backend will handle API requests for video uploads and streaming.

## 7. Technologies:

• Frontend: HTML, CSS, Vue.js, Thymeleaf (for templates).

• Backend: Java, Spring Boot.

• Data Streaming: Apache Kafka for real-time video processing.

• Storage: Minio (distributed object storage).

• Containerization: Docker for managing containerized deployments.

## 8. Development Phases:

• Phase 1: UI Development (Login, Upload, and Stream Pages).

• Phase 2: Kafka Integration (Produce and Consume Video Upload Events).

• Phase 3: Video Storage with Minio.

• Phase 4: Real-time Video Streaming.

• Phase 5: Admin Dashboard and System Testing.

## 9. Deliverables:

• Functional distributed video streaming application.

• Kafka-based real-time video processing and storage.

• Scalable backend that can handle high volumes of videos.

• Documentation of the system design and development process.

## 10. Timeline:

• Week 1-2: UI/UX Development and User Authentication.

• Week 3-4: Kafka integration for upload events and Minio storage.

• Week 5-6: Backend development for streaming and admin features.

• Week 7-8: Testing, Debugging, and Deployment.

## 11. Conclusion:

The distributed video streaming app will provide an innovative and scalable solution for handling large video datasets, ensuring real-time streaming, high availability, and fault tolerance. With the use of cutting-edge technologies like Kafka and Minio, this platform will cater to modern needs in content streaming and user management.

## 12. References:

• Spring Boot Documentation

• Apache Kafka Documentation

• Minio Documentation

This proposal is based on the architecture and system details described in the presentation, ensuring that each component and technology is aligned with the project’s requirements.