Term: Fall 2024 **Subject:** Computer Science & Engineering (CSE) **Number:** 512

Course Title: Distributed **D**atabase **S**ystems (CSE 512)

GROUP PROJECT PROPOSAL

Project Title:

Content Recommendation System for Social Media Explore Page

Team Name and Members:

Disco DeewaneS (DDS)

Deep Rodge: 1229519137

Jayati Goyal: 1227629236

Uttam Kumar 1229602753

1. Introduction

1.1 Background

In the age of social media, providing personalized content recommendations is critical for user engagement. Social media platforms rely on content delivery systems that recommend posts, videos, or articles tailored to individual users' interests. Distributed database systems play a key role in scaling these operations by managing vast amounts of user data across geographically dispersed servers. Current content recommendation systems often struggle with ensuring real-time personalization at scale while maintaining consistency,

availability, and fault tolerance. This project aims to build a prototype that optimizes these areas for the Explore page of social media applications.

1.2 Problem Statement

The problem is to design a content recommendation system for social media's Explore page, ensuring efficient data partitioning, replication, and fault tolerance across a distributed database. The system must handle thousands of users and content items, providing real-time recommendations with high availability and scalability.

1.3 Objectives

- To explore and implement efficient data distribution techniques that balance load across nodes, improving response times and resource utilization.
- To design and implement data partitioning, replication strategies, and fault-tolerance mechanisms specifically tailored for content recommendation.
- To evaluate the performance of the system in terms of scalability, availability, and query processing efficiency.

2. Project Description

2.1 System Design

The proposed system will rely on a distributed architecture to store user interaction data, content metadata, and recommendation algorithms. Key components include:

Data Partitioning: User and content data will be partitioned based on geographical location, user activity levels, and content type to distribute the load effectively.

Replication: User data will be replicated across multiple nodes to ensure availability and fault tolerance, with a master-slave replication model for consistency.

Fault Tolerance: We will implement a system where data writes will be acknowledged only when they are stored across multiple nodes to ensure durability.

Query Processing: The recommendation engine will query the database using distributed algorithms to rank and display content personalized for each user. This may involve sharding user preferences and performing distributed joins.

2.2 Implementation Plan

The project will be implemented using:

Languages: Python (Backend) and JavaScript (Frontend)

Databases: PostgreSQL will be the core database system and Neo4j for graph-based content recommendations.

Frameworks: Flask/Django for the backend, React for the frontend, and SQLAlchemy for ORM with PostgreSQL.

2.3 Data Strategy

We will use synthetic user interaction data (likes, shares, comments) to simulate real-world social media data. PostgreSQL will store both user interaction data and content metadata. Sensitive information will be anonymized to ensure compliance with data privacy regulations.

3. Methodology

3.1 Technique 1: Data Partitioning

We will use horizontal partitioning to split user interaction data across different regions, improving scalability and reducing query latency.

3.2 Technique 2: Replication Strategy

A master-slave replication approach will be used to balance read/write load and ensure availability during node failures.

3.3 Technique 3: Consistency Model

We will implement eventual consistency for user content updates, optimizing for availability in case of network partitions.

3.4 Technique 4: Recommendation Algorithm

A collaborative filtering algorithm will be employed, utilizing both user-item interaction data and content similarity to generate personalized recommendations.

3.5 Technique 5: Fault Tolerance Mechanism

The system will employ multi-node replication and a failure detection mechanism using heartbeat protocols to recover from node failures quickly.

4. Evaluation Plan

4.1 Metrics for Evaluation

- **Response Time**: Measure the time taken to fetch and display personalized content.
- **Throughput**: Evaluate the system's ability to handle concurrent user requests.
- **Fault Tolerance**: Test system recovery time from node failures and data consistency post-recovery.
- **Scalability**: Assess system performance as the user base and content size increase.

4.2 Expected Outcomes

- A scalable distributed content recommendation engine capable of handling thousands of users and queries.
- High availability and fault tolerance with minimal downtime.
- Optimized data partitioning for faster query processing and real-time recommendations.

5. Timeline

Include a timeline with key milestones and deadlines. You can use a table format or a Gantt chart to illustrate your schedule.

Milestone	Start Date	End Date	Team Member Responsible For
Project Proposal Submission	10/10/2024	10/13/2024	All
System Design and Plan	11/13/2024	10/20/2024	All
Backend development	10/21/2024	11/9/2024	Deep Rodge
Frontend Development	11/10/2024	11/19/2024	Jayati Goyal
Integrating Backend and Frontend	11/20/2024	11/25/2024	Uttam Kumar
Testing and Evaluation	11/26/2024	11/30/2024	All
Final Report and Video Creation	12/01/2024	12/02/2024	All

6. Conclusion

This project addresses the need for scalable and fault-tolerant content recommendation systems in social media applications. By implementing a distributed database system, we aim to provide efficient query processing and high availability, making real-time content personalization feasible for large-scale social media platforms. Our solution has the potential to contribute significantly to the field of distributed databases by demonstrating a balance between consistency, availability, and performance.

References

1. ChatGPT (Help with System Design)
2. https://www.algolia.com/blog/ai/the-anatomy-of-high-performance-recommender-systems-part-iv/
3. https://www.scylladb.com/2020/10/20/making-a-scalable-and-fault-tolerant-database-system-partitioning-and-replication/