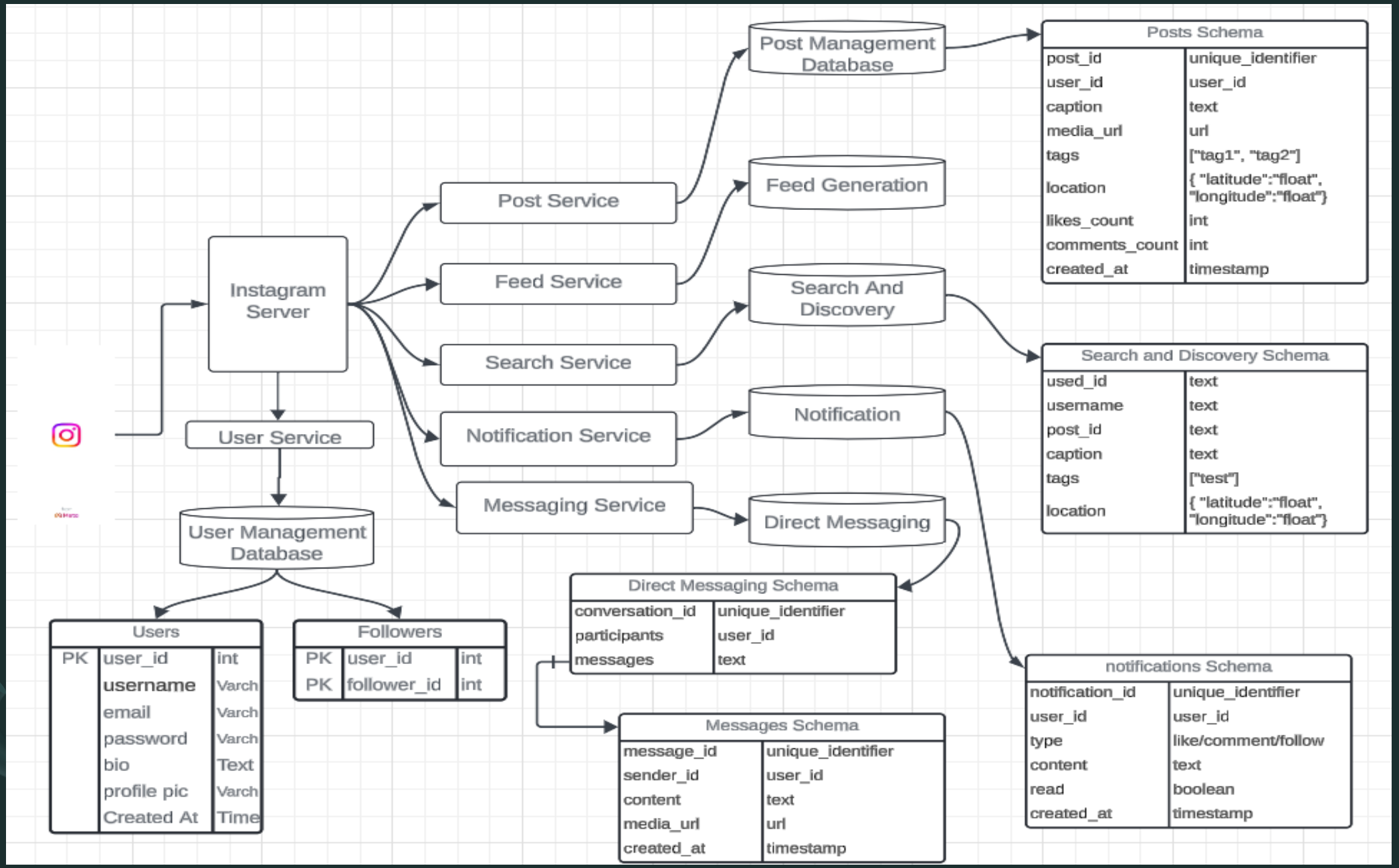
DBMS Assignment

Group 4:

1. Piyush Sunil Borse 25PGAI0026
2. Prateek Majumder 25PGAI0027
3. Bhawana Thawarani 25PGAI0137
4. Prajwal Wagh 25PGAI0109
5. Yuvraj Singh Srinet 25PGAI0019

Low Level Design with schema

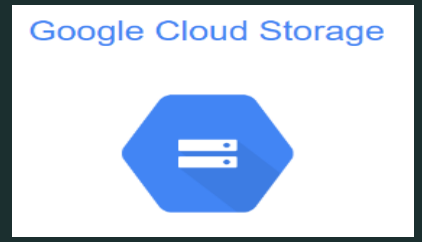


- # User Management

- SQL Database (MySQL) for storing user profiles, authentication data, and relationships.
- Reason: MySQL is a highly reliable and widely-used relational database that supports ACID transactions, ensuring data integrity. Its robust support for relational schemas and strong community backing make it a suitable choice for managing user data.

- # Post Management

- Media Storage (NoSQL): Use Google Cloud Storage for storing photos/videos.
- Reason: Google Cloud Storage offers scalable and secure object storage with high availability and durability, making it ideal for storing large volumes of media files.



Feed Generation

Redis (NoSQL): : Store feeds in Redis for fast retrieval and update in real-time with RabbitMQ.

Reason: Using Redis for caching ensures quick feed retrieval, while RabbitMQ handles real-time updates efficiently.

Search and Discovery

Indexing: Use Algolia to index user profiles, posts, and hashtags.

Reason: Algolia's advanced search features and real-time indexing capabilities enhance the search and discovery experience, providing fast and relevant results.

Notifications

Real-Time Processing: Use RabbitMQ to publish and subscribe to notification events.

Reason: RabbitMQ ensures reliable and timely delivery of notification events, supporting high-throughput and real-time processing requirements.

Storage: Store notifications in DynamoDB for fast retrieval.

Reason: DynamoDB's fast and predictable performance, coupled with its scalability, makes it suitable for storing large volumes of notification data.

Direct Messaging

Message Delivery: Use WebSockets for real-time message delivery.

Reason: WebSockets provide a persistent connection for real-time communication, ensuring instant message delivery and updates.

Storage: Store chat history in Firebase Realtime Database, with messages partitioned by conversation ID.

Reason: Firebase Realtime Database supports real-time data synchronization and offline mode, enhancing the user experience for chat applications.

- **Scalability Considerations**
- **Database Scaling :**

Sharding: Partition large datasets in Firestore and DynamoDB to distribute the load.

Reason: Sharding helps distribute the database load across multiple nodes, improving performance and scalability.

Replication: Use master-slave replication in MySQL for read scalability.

Reason: Replication improves read performance and availability by distributing read traffic across multiple replicas.

Consistency: Use eventual consistency in DynamoDB for messaging and notifications.

Reason: Eventual consistency provides higher availability and performance, suitable for use cases where immediate consistency is not critical.

- **Load Balancing:**
- Use Nginx or HAProxy to distribute incoming traffic across multiple instances of each microservice.
    - **Reason:** Load balancers help distribute traffic evenly, preventing overloading of individual servers and improving fault tolerance.
- **Fault Tolerance:**
- Deploy services across multiple availability zones.
    - **Reason:** Deploying across multiple availability zones enhances fault tolerance and availability, protecting against zone-level failures.
- Implement auto-scaling groups to handle traffic spikes.
    - **Reason:** Auto-scaling ensures that the system can handle sudden increases in traffic by automatically adding or removing resources.
    - **Data Partitioning and Consistency:**
- Use consistent hashing for partitioning data across multiple nodes.
    - **Reason:** Consistent hashing ensures even data distribution and minimizes data movement when adding or removing nodes.
- Ensure data consistency with distributed transactions where necessary (e.g., two-phase commit for critical operations).
    - **Reason:** Distributed transactions ensure data consistency across multiple nodes, critical for operations that require strong consistency.

Thank You!!! 😊