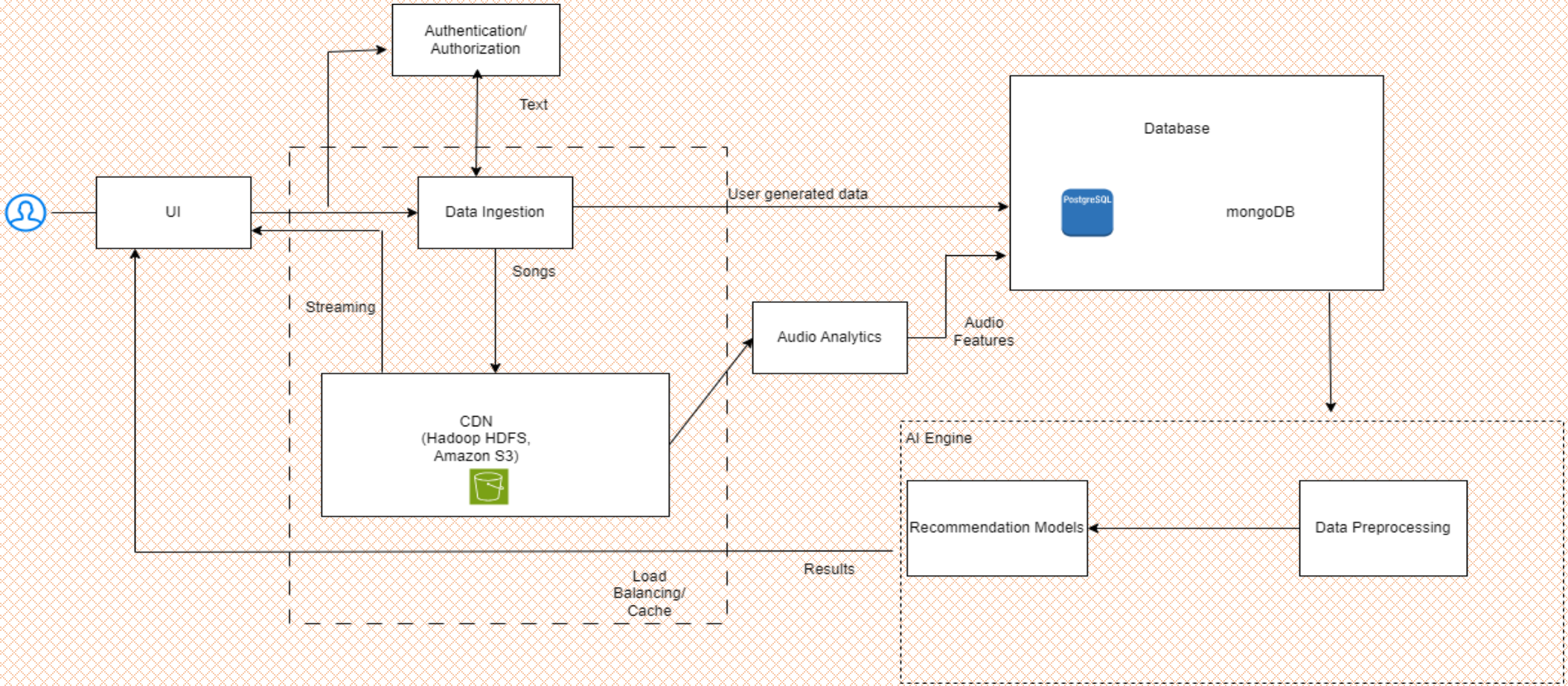
Data Engineer Technical Challenge- eMoodie

Mohammad Raeez

System block diagram



SYSTEM BLOCKS

1. **User Interface**:
    1. Mobile app, desktop app, web app for users.
    2. Dashboard for music creators to upload songs.

2. **Authentication and Authorization**:
    1. User registration and login.
    2. Role-based access control for music creators and listeners.

3. **Database**:
    1. Store user profiles, preferences, and song metadata.
    2. Use a relational database (e.g., PostgreSQL) for structured data.
    3. Use NoSQL databases (e.g., MongoDB) for semi-structured and unstructured data (e.g., user-generated comments)

SYSTEM BLOCKS

4. Data Ingestion:

1. Ingest songs and their metadata from music creators.
2. Store audio content in a distributed file system (e.g., Hadoop HDFS or Amazon S3).
3. Extract audio features (e.g., tempo, danceability) using audio analysis tools and store them in a separate database.

5. Recommendation Engine:

1. Collaborative filtering, content-based filtering, and hybrid recommendation models.
2. ML models trained on user preferences, song properties, and audio features.
3. Real-time and batch recommendation pipelines.

6. Caching and Load Balancing:

1. Use caching (e.g., Redis) for frequently accessed data like user profiles and recommendations.
2. Load balancers to distribute user requests across application servers.

SYSTEM BLOCKS

7. Data Preprocessing Pipelines:

- Preprocess user data, song metadata, and audio features to feed into the recommendation engine.
- Normalize and clean data, handle missing values, and perform feature engineering.

8. Scalability:

1. Implement horizontal scaling for application servers and databases.
2. Use container orchestration tools like Kubernetes for managing microservices.
3. CDN for efficiently serving audio content to users.

9. Security:

1. Implement SSL/TLS for data encryption.
2. Regularly update and patch system components.
3. Implement access controls, input validation, and secure API endpoints.
4. Protect against DDoS attacks and data breaches.

10. Monitoring and Analytics:

1. Implement monitoring tools for system health and performance (e.g., Prometheus, Grafana).
2. Collect user behavior data for analytics and improving recommendations.

Potential Scalability and Security Issues:

- Data Volume
- Real-time Processing
- User Interactions
- Data Privacy
- Authentication
- Content Security
- Data Breaches
- DDoS Attacks
- API Security

Potential Tradeoffs

- **Latency vs. Recommendation Quality**
- **Scalability vs. Cost**
- **Data Privacy vs. Personalization (UX)**
- **Real-Time vs. Batch Processing**
- **Security vs. Usability**

Main Challenges that can be expected

- **Data Volume and Scalability**
- **Real-time** recommendations
- **Audio Analytics**
- **Maintaining High Availability**
- **Data Storage Costs**
- **Cost Management of the system**

Part 2

- **Q1**

```
SELECT name, price FROM products ORDER BY price ASC LIMIT 1;
```

This selects name and price from the products table, orders the products by price in ascending order (lowest price first), and then limits the result to the first row, which represents the product with the lowest price.

Alternatively, can use Rank function to do the same

- Q2

```
SELECT AVG(order_cost) AS average_order_cost
FROM (
        SELECT (price * quantity + tax_rate + shipping_rate) AS order_cost
        FROM orders
    ) AS subquery;
```

This query calculates the order cost for each order by multiplying the price by quantity and then adding tax_rate and shipping_rate. It then calculates the average of these order costs to determine the average order cost.

Thank You!