**Food Delivery App Backend Data Engineering Project Documentation**

**1. Introduction**

This document outlines the backend data engineering aspects of a food delivery application. The project focuses on integrating weather data into the order processing pipeline to potentially apply additional charges based on unfavourable weather conditions.

**2. System Architecture**

The system will consist of the following components:

* **Mobile App:** Users place orders for food delivery through the mobile app.
* **Order Database:** Stores order details (e.g., items, customer information, delivery address).
* **Weather API:** Provides real-time weather data for a given location (OpenWeatherMap or similar service).
* **Data Processing Engine:** Processes order data, retrieves weather information, and calculates potential extra charges.
* **Delivery Management System:** Manages delivery logistics and interacts with delivery personnel.

**3. Data Flow**

1. **Order Placement:** Users submit orders through the mobile app, which are then stored in the Order Database.
2. **Order Processing:**
   * The Data Processing Engine extracts new orders from the Order Database.
   * For each order, the system extracts the delivery address (city name).
   * The Data Processing Engine calls the Weather API using the city name.
3. **Weather Data Retrieval:**
   * The Weather API returns weather data relevant to the city (e.g., temperature, precipitation).
4. **Extra Charge Calculation:**
   * Based on pre-defined rules (e.g., heavy rain, extreme temperature), the system determines if an extra charge applies.
   * If applicable, the extra charge is calculated and added to the order total.
5. **Order Update:**
   * The updated order information, including potential extra charges, is sent back to the Order Database.
6. **Delivery Management:**
   * The Delivery Management System receives the updated order details and initiates the delivery process.
   * The mobile app can display the updated order information to the user, including any extra charges.

**4. Data Engineering Tasks**

* **Data Extraction:** Design and implement mechanisms to extract new orders from the Order Database at regular intervals. This might involve streaming data pipelines or scheduled queries.
* **API Integration:** Develop code to interact with the Weather API securely, including handling API keys and rate limits.
* **Data Transformation:** Parse and process the retrieved weather data to extract relevant information for extra charge calculations.
* **Data Enrichment:** Enrich order data with weather information and calculated extra charges.
* **Data Storage:** Update the Order Database with the enriched order details.
* **Job Scheduling:** Implement mechanisms to schedule data processing tasks regularly (e.g., using Databricks Jobs).
* **Error Handling:** Design robust error handling for potential issues like API failures, missing data, or invalid weather conditions.

**5. Technologies**

* Programming Languages: Python (likely)/pyspark
* Data Processing Engine: Apache Spark
* Database: Relational Database (e.g., MySQL, PostgreSQL) or Delta Lake for Order Database
* Weather API: OpenWeatherMap
* Cloud Platform: Databricks

**6. Benefits**

* **Improved Delivery Efficiency:** Understanding weather conditions can help optimize delivery routes and avoid delays during adverse weather.
* **Enhanced Safety:** Extra delivery charges during bad weather can incentivize caution for delivery personnel and potentially benefit customer safety.
* **Transparency:** Users are informed about potential extra charges upfront, promoting a fair and transparent pricing model.

**7. Future Considerations**

* **Machine Learning:** Integrate machine learning models to predict weather conditions based on historical data and improve extra charge prediction accuracy.
* **Real-time Integration:** Explore real-time weather data integration for more dynamic extra charge calculations.
* **User Preferences:** Allow users to set preferences for weather-related extra charges (e.g., opt-in or opt-out).

**8. Conclusion**

This data engineering project integrates weather data into the food delivery app's backend to potentially add extra charges based on unfavourable weather conditions. By leveraging data processing frameworks and APIs, the system can enhance delivery efficiency, promote safety, and provide transparency to users.