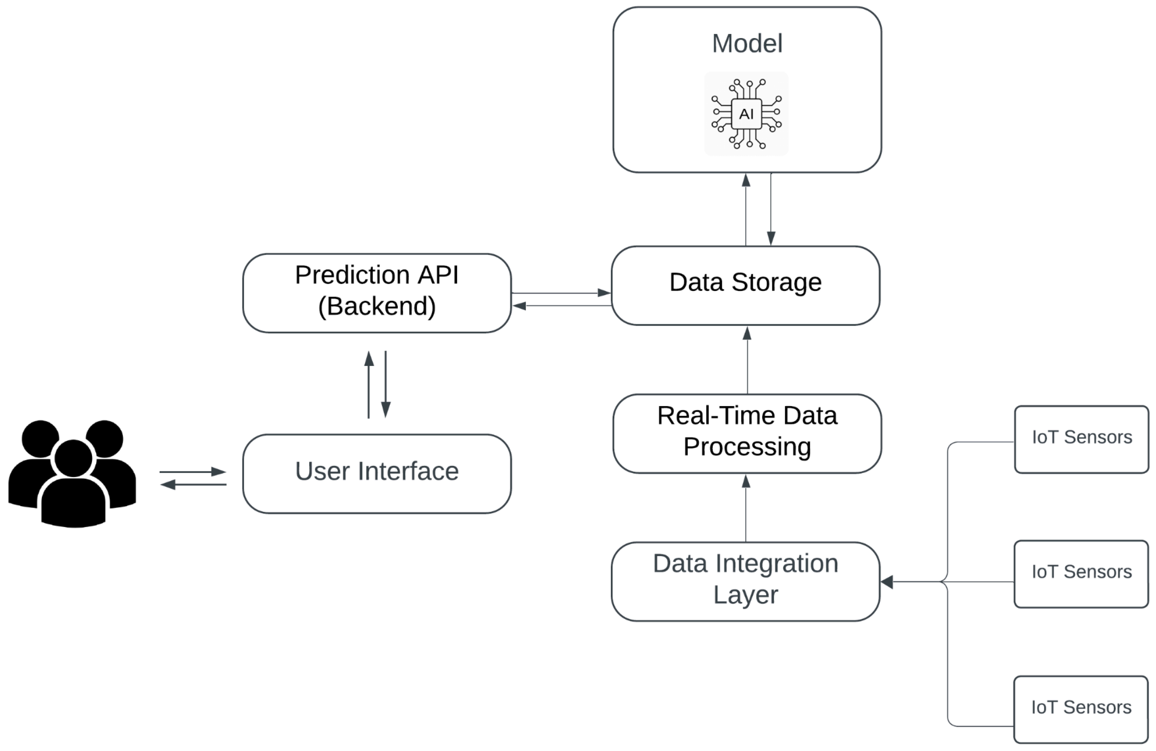
System Design for Predicting 21-Day Rain Probability

1. System Overview

The goal of this system is to predict the probability of rainfall for the next 21 days, based on real-time data from IoT sensors. These sensors measure various environmental variables such as temperature, humidity, and wind speed at 1-minute intervals. The system needs to be resilient to occasional sensor malfunctions and should handle real-time data processing, prediction generation, and user interaction seamlessly. Below is a step-by-step breakdown of the system architecture, followed by a description of each component.

2. System Architecture

The weather forecasting system is designed to process real-time environmental data from **IoT sensors** and generate accurate rainfall predictions using **machine learning models**. It consists of multiple components, including **data ingestion, processing, storage, model training, and prediction APIs**, ensuring seamless end-to-end functionality.



1. IoT Sensors & Devices – Collect real-time weather data, with built-in error handling for faulty readings.
2. Data Ingestion & Processing – Uses Apache Kafka/Spark Streaming to clean and transform real-time data.
3. Data Storage – InfluxDB for real-time queries, BigQuery for historical analysis.
4. Machine Learning Model – LSTM/ARIMA models predict rain probability for 21 days and adapt to new data.
5. Prediction API – Built with FastAPI for real-time rain predictions.
6. User Interface – Web dashboard to visualize forecast trends.
7. Monitoring & Error Handling – Uses Prometheus & Grafana to track system health and anomaly detection to handle sensor errors.

This architecture ensures a **scalable, efficient, and real-time weather prediction system**, empowering **farmers** with **accurate rainfall forecasts** for smarter agricultural planning.

3. Detailed Description of Each Component

IoT Sensors & Devices

- Function: Collect weather-related data in real time. These sensors are critical for capturing the environmental parameters needed for rainfall prediction.
- Challenges: Sensors may malfunction due to external factors (e.g., power loss, environmental interference). The system must be resilient and account for missing or corrupted data points.
- Solution: Implement redundancy (backup sensors), data imputation, or use historical patterns when a sensor failure is detected.

Data Ingestion Layer

- Function: Ingest incoming data streams from IoT devices and send them to the processing layer.
- Challenges: The ingestion layer must handle high throughput and ensure the system can scale with increasing data.
- Solution: Use Apache Kafka or AWS Kinesis for fault-tolerant, real-time streaming. These platforms allow for efficient data transport, buffering, and easy scaling.

Real-time Data Processing

- Function: processes the raw sensor data in real-time, performs data cleaning (e.g., missing data imputation), and applies anomaly detection algorithms.
- Challenges: Ensuring low-latency processing and cleaning while maintaining accuracy.

- Solution: Use Apache Flink or Spark Streaming to process data in near real-time. Integrate anomaly detection techniques to handle malfunctioning sensors and filter out invalid data.

Data Storage

- Function: Stores historical and real-time data for further analysis and model training.
- Challenges: Ensuring fast querying and easy scalability as the amount of data grows.
- Solution: Use InfluxDB or TimescaleDB for efficient time-series data storage. For larger datasets or more complex analyses, use a Data Warehouse (e.g., BigQuery) for historical data.

Machine Learning Model

- Function: Trains and serves predictions for rainfall probability over the next 21 days.
- Challenges: Accurate model training, retraining as new data arrives, and the ability to generalize well across different weather patterns.
- Solution: Use **LSTM** or **ARIMA** models for time-series forecasting. The model is periodically retrained with fresh data to keep it updated. Tools like Kubeflow can manage the model lifecycle.

Prediction API

- Function: Provides an interface for external systems and users to request rainfall predictions.
- Challenges: Ensuring low-latency and high availability of the prediction service.
- Solution: Implement a RESTful API using FastAPI or Flask. Deploy the API using a scalable platform like Kubernetes.

User Interface

- Function: Provides users with a dashboard to interact with the system, view current weather conditions, and explore rain probability forecasts.
- Challenges: Creating a simple, intuitive interface that delivers real-time predictions.
- Solution: Design a web-based dashboard that pulls data from the API and displays the forecast in a user-friendly format.

Error Handling & Monitoring

- Function: Monitors the health of the system and alerts on any anomalies or malfunctions in the IoT devices.
- Challenges: Ensuring the system remains operational even if some parts fail.
- Solution: Use Prometheus and Grafana for system monitoring. Implement anomaly detection algorithms to detect faulty data and ensure that sensor failures do not affect the predictions.