Report 2: System Design for Rain Prediction

1. Introduction

This report outlines the design of a system to predict rain probabilities using real-time data from IoT sensors. The goal is to predict whether it will rain in the next 21 days, providing valuable information for farmers. The system processes real-time weather data, predicts rain, and delivers results to end users. We also discuss how the system handles potential issues like sensor malfunctions.

1.1 Objective

To develop a system that provides accurate rain predictions in real-time by processing data from weather IoT devices, predicting the likelihood of rain, and making predictions available to users through a web interface.

1.2 Requirements

- **Real-time Processing**: The system processes data in real time (every minute).
- **Error Handling**: The system should account for sensor malfunctions and missing data.
- **User Interaction**: End users can access predictions and interact with the system.

2. System Architecture

The system consists of the following key components:

2.1 IoT Sensors

- **Role**: Collects real-time weather data such as temperature, humidity, and wind speed.
- **Error Handling**: In case of missing or faulty data, the system uses imputation techniques (e.g., mean imputation).

2.2 Data Preprocessing

- **Role**: Handles missing values, normalizes data, and prepares features for the machine learning model.
- **Error Handling**: Missing data points are filled using statistical techniques like mean or median.

2.3 Machine Learning Model (Flask API)

- **Role**: Hosts the trained machine learning model to predict rain based on the preprocessed data.
- **Error Handling**: If data is insufficient for prediction, the system returns a default prediction or requests more data.

2.4 Web Interface (HTML/PHP)

- **Role**: Provides the user interface where end users can view predictions.
- **Data Flow**: Displays rain probabilities and allows users to interact with the system.

2.5 End User (UI)

- **Role**: End users access the system through a web browser or mobile app.
- **Interaction**: Users input data and view predictions, such as whether it will rain in the next 21 days.

3. Data Flow

The flow of data within the system is as follows:

1. **IoT Sensors** collect weather data and send it to the **Data Preprocessing** component.
2. The preprocessed data is passed to the **ML Model**, which makes predictions about rain probability.
3. **Web Interface** delivers these predictions to the **End User**, who interacts with the system.

The system ensures that predictions are accurate by updating the model periodically with fresh data.

4. Error Handling and Sensor Malfunctions

Given the nature of IoT sensors, there's a chance for data anomalies or malfunctions. The system handles these in the following ways:

- **Imputation**: Missing or faulty data points are filled with statistical estimates (mean, median).
- **Fallback Mechanism**: If a sensor consistently fails, the system alerts the user and may switch to historical data or external weather forecasts.
- **Data Validation**: The system validates incoming data to ensure it is within expected ranges (e.g., temperature should not be below absolute zero).

This ensures that predictions remain reliable even in the case of sensor errors.

5. Scalability and Reliability

The system is designed to scale easily:

- **Cloud Deployment**: The backend, including the Flask API and model, is deployed in the cloud (e.g., AWS), allowing it to handle growing amounts of data.
- **Microservices Architecture**: By separating components like data preprocessing, prediction, and user interface, the system can scale effectively. New sensors or features can be added without disrupting the existing workflow.

Additionally, redundancy is built in to ensure that the system remains operational even if individual sensors or components fail.

6. Conclusion

This report describes the architecture of a system designed to predict rainfall probabilities based on real-time weather data from IoT sensors. By preprocessing the data, making predictions through machine learning models, and delivering results via a web interface, the system provides an efficient way to support farmers with accurate rain predictions. The system handles sensor errors and malfunctions through imputation and validation, ensuring reliability in real-world scenarios.