Rainfall Prediction & Anomaly Detection Project Plan Phase-2

1.Objective of the Design Phase

The design phase defines the overall system structure and lays the foundation for development and deployment. It transforms research ideas into actionable components by specifying:

- Data entities and their relationships (ER Diagram)
- Logical flow of data and processes (Flowchart)
- Physical architecture for data ingestion, storage, processing, and modeling

This phase ensures that all modules are well-integrated and that the system is scalable, secure, and adaptable for future improvements.

2. System Design Overview

The system architecture is modular and cloud-enabled, structured into distinct layers:

- **Data Layer** (Ingestion & Storage)
- **Processing Layer** (Cleaning, Feature Engineering)
- **ML/DL Modeling Layer** (Prediction, Anomaly Detection)
- **Output Layer** (Visualization, Reporting, Dashboards)

This layered approach ensures clean separation of concerns and easy maintainability.

3. System Architecture Diagram

This architecture lays out the end-to-end flow of data from raw ingestion to final prediction and anomaly reporting.

Implementation Details:

- **CSV Input**: The dataset (rainfall in india 1901–2015.csv) is the primary data source. It includes monthly rainfall statistics for subdivisions of India from 1901 to 2015.
- **Data Ingestion**: The file is read using Pandas. Initial filtering and validation checks are performed to handle missing or inconsistent data.
- **Database Connection**: Using the ibm_db Python connector, structured data is inserted into IBM Db2. SQL queries ensure only cleaned, validated data is stored.
- **Storage in IBM Db2**: The database is designed with normalized tables for efficient storage, querying, and future updates. Rainfall records are linked with regions and prediction results.
- **Data Cleaning & Transformation**: Null values, outliers, and formatting errors are handled using Pandas and NumPy. Features like 'Annual Rainfall', 'Region Encoding', and 'Seasonal Aggregates' are engineered.
- **Modeling Layer**:
    - **Random Forest & Linear Regression**: Used to learn from past rainfall trends.
    - **LSTM**: Handles sequence data for time-series forecasting.
    - **KMeans**: Groups regions into rainfall clusters.
    - Models are trained, validated, and tuned using train_test_split and evaluation metrics like MSE and R^2.

- **Anomaly Detection**: Residuals (difference between actual and predicted) are computed. If residuals exceed a statistical threshold, they are flagged as anomalies.
- **Visualization Layer**: Results are visualized using Matplotlib and Seaborn—scatter plots, heatmaps, box plots, etc., help stakeholders interpret model results.
- **Optional Deployment**: The model can be hosted using Flask or deployed to IBM Cloud for real-time rainfall analysis and dashboard creation.

4. Entity-Relationship Diagram

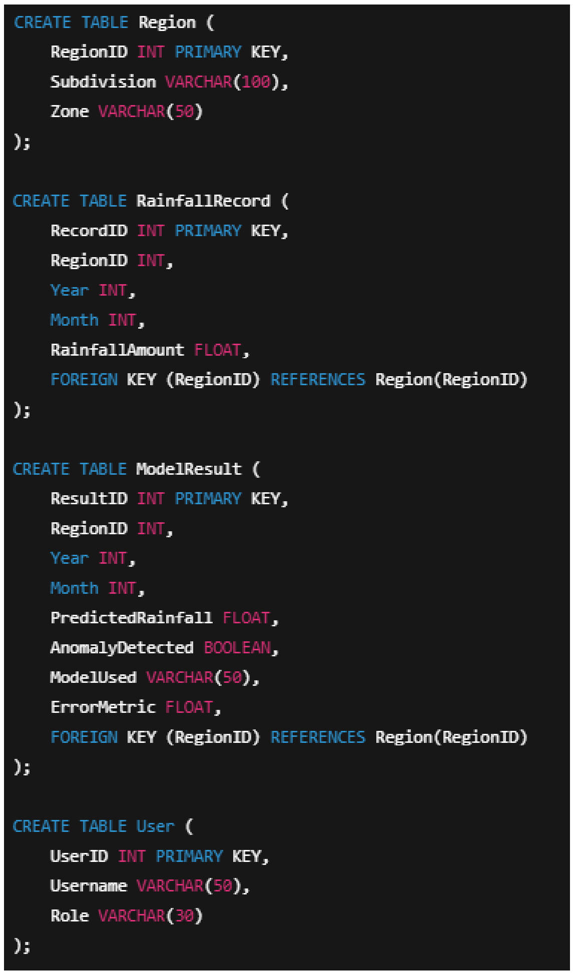
The ERD defines how the system structures and links different types of rainfall-related data in the IBM Db2 database.

Implementation Details:

- **Region Table**:
  - Stores unique subdivision identifiers and related zone information.
  - Each region is linked to multiple rainfall records and prediction outputs.
- **RainfallRecord Table**:
  - Contains raw and monthly rainfall data by region and time (year, month).
  - This table is filled during the data ingestion process after cleaning.
- **ModelResult Table**:
  - Stores output from each model for a specific region and time.

    - Includes predicted rainfall values, error metrics, model name, and anomaly status.
- **User Table** *(Optional)*:
    - Enables role-based access for future scalability.
    - Useful if the system is expanded to support dashboards or authenticated access.

This schema ensures **data normalization**, reduces redundancy, and makes future expansion easy. Foreign keys maintain referential integrity.



```sql
CREATE TABLE Region (
    RegionID INT PRIMARY KEY,
    Subdivision VARCHAR(100),
    Zone VARCHAR(50)
);

CREATE TABLE RainfallRecord (
    RecordID INT PRIMARY KEY,
    RegionID INT,
    Year INT,
    Month INT,
    RainfallAmount FLOAT,
    FOREIGN KEY (RegionID) REFERENCES Region(RegionID)
);

CREATE TABLE ModelResult (
    ResultID INT PRIMARY KEY,
    RegionID INT,
    Year INT,
    Month INT,
    PredictedRainfall FLOAT,
    AnomalyDetected BOOLEAN,
    ModelUsed VARCHAR(50),
    ErrorMetric FLOAT,
    FOREIGN KEY (RegionID) REFERENCES Region(RegionID)
);

CREATE TABLE User (
    UserID INT PRIMARY KEY,
    Username VARCHAR(50),
    Role VARCHAR(30)
);
```

Figure 1: Database Schema(Logical Design)

5. Flowchart of Rainfall Prediction Pipeline

The flowchart describes the logical steps taken by the system to go from raw data to prediction and anomaly identification.

Implementation Details:

1. **Start**: The process begins with initiating the pipeline either through a script or notebook.
2. **Load Dataset**:
   - CSV file is read using pandas.read_csv().
   - Column names are standardized; incorrect types are converted.
3. **Clean & Transform Data**:
   - Handle missing values using interpolation or deletion.
   - Create new features like cumulative rainfall or seasonal averages.
4. **Upload to IBM Db2**:
   - Cleaned data is uploaded to the RainfallRecord table using SQL INSERT operations via ibm_db.
5. **Train Models**:
   - Models are trained on past data.
   - Hyperparameters are tuned and performance metrics computed.
   - Models used: Linear Regression, Random Forest Regressor, LSTM.
6. **Generate Predictions**:
   - Trained models make predictions on test data or future unseen data.
   - Results are stored in the ModelResult table.
7. **Detect Anomalies**:
   - Calculate residuals.
   - Threshold for anomalies is set manually or via statistical measures (e.g., ±2 standard deviations).
   - Anomalous points are flagged and visualized.
8. **Visualization**:

  - Actual vs. predicted graphs.
  - Time-series plots, error distribution, and heatmaps.
  - Anomalies highlighted for quick analysis.

9. **End**: Results can be exported, stored, or used for alert generation.

6. Design Best Practices Followed

Modularity: Each component is independent, allowing for parallel development and testing.

Scalability: Cloud-native storage (IBM Db2) and use of ML pipelines ensure the system can scale with data.

Security: Design supports user authentication and role-based access if needed.

Reusability: Models and scripts are modular, so components like anomaly detection can be reused or updated independently.

Extensibility: Easy to plug in new ML models, add new regions or years, and connect to visualization dashboards.

7. Conclusion of Design Phase

The design phase has resulted in a **robust, modular, and scalable system blueprint** for rainfall prediction and anomaly detection. It defines how data will flow through the system, how it will be structured in the database, how modeling will be performed, and how results will be stored and visualized.

With the architecture, ER diagram, and flowchart fully specified, the project is well-positioned to move into full implementation, testing, and deployment.