✅ PROJECT TITLE:

AI-Driven Flood and Landslide Prediction System

🎯 OBJECTIVE:

To develop a system that analyzes real-time environmental and geospatial data (rainfall, terrain, soil moisture, etc.) and predicts the risk of floods and landslides using AI/ML, providing early warnings to disaster response systems.

🛠 TECH STACK — What You’ll Need to Learn & Use

1. Backend & APIs

Node.js / Python (Flask/FastAPI) – for API, data processing pipeline.

MongoDB / PostgreSQL – to store alerts, user data, logs.

REST APIs / WebSocket – for live data and push notifications.

2. Frontend (Dashboard)

React.js / Next.js – to display warning levels, maps, graphs.

Leaflet / Mapbox / Google Maps API – for real-time mapping.

3. Machine Learning

Python (Pandas, Scikit-learn, XGBoost, TensorFlow).

Models: Random Forest, Decision Tree, or Gradient Boosting for prediction.

Optionally: Geospatial ML libraries like GeoPandas or rasterio.

4. Data Sources

Rainfall, soil moisture, river levels: Use public datasets like:

NASA EarthData

IMD (India Meteorological Department)

NOAA

ISRO Bhuvan

5. Cloud/DevOps (optional but impactful)

AWS / GCP / Render / Railway – for deployment.

Docker – to containerize your ML model + backend.

Cron jobs – for scheduled data fetch/predictions.

📋 SYSTEM REQUIREMENTS

💾 Software Requirements

Python (for ML)

Node.js (optional if using backend in JS)

MongoDB / PostgreSQL

Jupyter Notebook or VS Code

React + Tailwind (for frontend UI)

Google Maps / Leaflet API key

Git & GitHub

🧮 Hardware Requirements

Local laptop with min 8GB RAM (or use Google Colab for training).

Cloud/VM for hosting if needed (Heroku, Railway, or AWS EC2).

🔄 STEP-BY-STEP WORKFLOW

✅ Step 1: Problem Research & Dataset Collection

Study real flood/landslide zones (e.g., Kerala, Uttarakhand).

Collect datasets:

Rainfall patterns (hourly/daily)

Soil moisture

Elevation/terrain

River flow

Past disaster events

✅ Step 2: Preprocessing the Data

Clean null/missing data.

Normalize data (standard scaling, min-max).

Convert geospatial data to latitude/longitude grids (if needed).

Visualize with graphs/maps to understand patterns.

✅ Step 3: ML Model Development

Train models to predict:

Probability of flood or landslide at a location.

Techniques:

Binary Classification (Flood: Yes/No)

Regression (Risk Level: 0 to 1 scale)

Train/test split, accuracy metrics (precision, recall, F1-score).

✅ Step 4: Build Alert System (Backend)

Develop a REST API that:

Takes in sensor/live data

Runs prediction model

Sends alerts if probability > threshold

Store alerts in DB and expose via endpoints

✅ Step 5: Frontend Dashboard

Real-time map with color-coded zones (red, yellow, green).

Display:

Live data feeds

Model predictions

Alert history

Use Leaflet/Google Maps API + Tailwind for a clean UI.

✅ Step 6: Deploy & Integrate

Deploy model and backend on cloud (Render/Heroku/Railway).

Cron job for auto-refreshing predictions.

(Optional) Integrate WhatsApp/SMS API for alerting.

💡 Optional Features (if you have time):

📱 Mobile-friendly PWA (React-based).

📊 Real-time Twitter alerts scraping.

🔁 Historical comparison of disasters (trends graph).

📍 Allow public to report floods in real time via form + GPS.

📦 Suggested Folder Structure

plaintext

Copy code

/

├── backend/

│ ├── app.py (or index.js)

│ ├── model.pkl

│ └── routes/

├── frontend/

│ └── React components + maps

├── ML/

│ └── notebooks, datasets, models

├── Dockerfile

├── README.md

└── requirements.txt / package.json

🧠 Skills You’ll Learn:

Real-world geospatial data handling

Applied ML for disasters

Data pipeline & alert system development

Real-time dashboard UI

Cloud deployment + containerization

Does this project require IoT sensors or devices?

No, this project does not require IoT sensors.

It can absolutely be built without using any physical sensors or devices, by relying on publicly available environmental datasets and satellite data.

✅ Two Ways to Build This Project:

1. Without IoT (Recommended for You) – Use Public Data

You’ll collect data from online sources and simulate “real-time” input.

📡 Example Data Sources:

NASA EarthData – rainfall, soil moisture, elevation

IMD (India Meteorological Department) – rainfall data (CSV)

ISRO Bhuvan – elevation, terrain info (DEM maps)

NOAA – river levels, climate conditions

💻 Tools to Simulate Real-Time Data:

Use cron jobs or scripts to pull hourly/daily data from these sources or dummy data

Or, create mock sensors using JSON files to simulate "live updates"

Integrate that with your ML model

🧠 This approach is practical, doesn't depend on hardware, and is good for deployment on the cloud.