1. Problem Statement

Landslides are one of the most devastating natural disasters globally, causing thousands of fatalities and economic losses every year. They often occur suddenly following heavy rainfall, soil saturation, or deforestation. Traditional monitoring systems rely on manual inspections, historical data, or hardware-based sensors, which are costly, geographically limited, and often fail to provide early warnings in time.

Challenges in current solutions:

- **Lack of real-time global coverage**
- **Hardware dependency in remote areas**
- **Limited predictive capability using only rainfall or slope data**
- **Low accessibility in low-bandwidth regions**

Need: A software-driven predictive system using satellite data, machine learning, and spatial analytics that provides actionable alerts to prevent loss of life and property.

---

2. System Architecture

EcoGuardian.AI operates as a Digital Twin, simulating slope stability in near real-time.

Architecture Overview:

```
Satellite APIs (Rainfall, Soil Moisture, NDVI, DEM)
          ↓
Data Ingestion & Preprocessing
          ↓
Feature Engineering (ARI, Slope, Vegetation Factor)
```

```
          ↓
Machine Learning Inference (Random Forest / LSTM)
          ↓
Risk Raster & Heatmap Generation (GEE + Mapbox)
          ↓
User Alerts (Web + SMS)
```

---

3. Data Ingestion Pipeline

EcoGuardian.AI uses only software-accessible “virtual sensors”:

Variable	Source	Frequency	Purpose
Rainfall	NASA GPM IMERG	Every 30 mins	Trigger events for potential slides
Soil Moisture	NASA SMAP	Every 2–3 days	Pre-condition soil saturation
Elevation	SRTM 30m	Static	Compute slope and runout paths
Vegetation	Sentinel-2 NDVI	Every 5 days	Adjust risk for vegetation coverage
Historical Landslides	NASA LHASA	Historical	Train predictive ML models

Data Preprocessing:

- **Temporal alignment for ARI calculation**
- **Raster reprojection to common coordinate system**
- **Interpolation for missing values in SMAP or NDVI datasets**

- **Standardization for ML input**

---

4. Feature Engineering

A. Antecedent Rainfall Index (ARI)
Measures soil saturation over the past days using decay factor `k = 0.85`:

```
def calculate_ari(rainfall_list, k=0.85):
    return sum((k**i) * rain for i, rain in enumerate(rainfall_list))
```

B. Terrain Slope Computation
Using DEM (SRTM):

```
import richdem as rd
dem = rd.LoadGDAL("srtm.tif")
slope = rd.TerrainAttribute(dem, attrib="slope_degrees")
high_risk_slopes = slope > 25
```

C. Vegetation Index Integration
NDVI reduces landslide risk in forested areas:

```
risk_modifier = 1 - NDVI  # Forested = lower risk
```

D. Soil Saturation Estimation
Combines ARI with SMAP soil moisture to quantify water load in soil layers.

---

5. Modeling Approach

Model Choice:

- **Random Forest: Baseline, interpretable, robust with sparse data**
- **LSTM: Optional, captures temporal ARI trends for time-series predictions**

Input Features:

- **ARI**
- **48hr Rainfall**
- **Soil Moisture**
- **Slope Angle**
- **NDVI**
- **Elevation**

Training & Labels:

- **Historical landslide occurrence from NASA LHASA**
- **Binary labels: Slide = 1, No Slide = 0**

Random Forest Example:

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=300,
max_depth=12, class_weight='balanced')
model.fit(X_train, y_train)
```

Risk Classification:

Probability	Status
< 0.3	SAFE
0.3–0.6	WATCH
0.6–0.8	WARNING
> 0.8	CRITICAL

---

6. Predictive Heatmap Generation

Layers:

1. **Susceptibility (Static): High slopes + weak soil**
2. **Hazard (Dynamic): ARI + SMAP data**
3. **Heatmap Rendering: Google Earth Engine raster → Mapbox GL JS frontend**

Color Scheme:

- **Green = Safe**
- **Yellow = Soil saturation rising**
- **Red = Immediate hazard**

Mapbox Pulse Effect Example:

```
map.addLayer({
  id: 'risk-heat',
  type: 'heatmap',
```

```
  source: 'risk-tiles',
  paint: {
    'heatmap-intensity': 1.5,
    'heatmap-color': [
      'interpolate',
      ['linear'],
      ['heatmap-density'],
      0, 'green',
      0.5, 'yellow',
      1, 'red'
    ]
  }
});
```

---

7. Advanced Features

A. Shadow-Model Verification:
Compares current rainfall to 10-year historical maximum to detect rare extreme events.

B. Explainable AI (XAI):

- **SHAP or rule-based explanations: “Risk 92% — 48hr Rainfall exceeds 10-year historical threshold”**

C. Low-Bandwidth SMS Alerts:

```
from twilio.rest import Client
msg = f"⚠ LANDSLIDE ALERT\nRisk: CRITICAL\nLocation: 11.45N,76.89E"
client.messages.create(body=msg, from_='+1XXXX', to='+91XXXXXXXXX')
```

---

8. System Workflow Diagram

```
[Satellite APIs] --> [Data Ingestion & Alignment] --> [Feature Engineering (ARI, Slope, NDVI)]
        --> [ML Inference (RF/LSTM)] --> [Risk Raster Generation] --> [Predictive Heatmap + Alerts]
```

---

9. Conclusion

EcoGuardian.AI provides a software-only, globally scalable, near real-time landslide prediction system.

Impact:

- **Saves lives by early warning**
- **Accessible in low-bandwidth areas**
- **Adaptable for global deployment**
- **Explainable and verifiable predictions**

Future Extensions:

- **LSTM integration for improved time-series prediction**
- **Real-time mobile dashboard**
- **Integration with local disaster management systems**

✅ This report is hackathon-ready and covers all aspects from data ingestion → modeling → heatmap → alert system.

If you want, I can also make a polished PDF-ready version with figures, flow diagrams, heatmap mockups, and tables, so it looks like a professional research submission.

Do you want me to do that next?

1. The Core Data Strategy (No Sensors Needed)

To build this without physical hardware, you will rely on **Virtual Sensors**—satellite-derived data that measures the earth's surface every few hours.

Data Type	Virtual Sensor / API	Why it's critical
Rainfall History	**NASA GPM** (Global Precipitation Measurement)	Essential for calculating the **Antecedent Rainfall Index (ARI)**.
Soil Saturation	**NASA SMAP** (Soil Moisture Active Passive)	Measures how "heavy" and "loose" the soil is before it slides.
Elevation/Slope	**ALOS PALSAR** or **SRTM**	Landslides require a slope; this provides the 3D geometry of the hill.
Land Cover	**Sentinel-2 (NDVI)**	Areas with deforested slopes (low NDVI) are at much higher risk.

---

2. The "Disaster Time" Logic: Antecedent Rainfall

The software must analyze **temporal data**, not just the current weather. A landslide rarely happens because of one storm; it happens because the soil was already "full" from rain days ago.

The "Spillover" Formula:

You should implement a Decay Function in your code to calculate the Antecedent Precipitation Index (API):

$$API_t = R_t + \sum_{i=1}^{n} (k^i \cdot R_{t-i})$$

- R_t: Rainfall today.
- k: Recession coefficient (usually **0.85**). This represents how the soil "drains" over time.
- **The Logic:** If the API_t exceeds a regional threshold (calculated from historical climate data), your software triggers an alarm.

---

3. System Architecture (The Software Flow)

1. **Ingestion:** Python scripts pull GPM and SMAP data from NASA's Earthdata servers via API.
2. **Processing:** Google Earth Engine (GEE) calculates the **Slope Gradient** from the Digital Elevation Model (DEM).
3. **Analysis:** A **Long Short-Term Memory (LSTM)** neural network (ideal for time-series data) compares the current API_t against the historical "Slide Threshold" for that specific region.
4. **Visualization:** Use **Mapbox** or **Leaflet** to render a real-time hazard heatmap.