FloodWatch

Tagline: *"See the Risk Before the Rain: AI-Powered Road Flood Alerts."*

Problem Statement

Nepal's mountainous terrain and intense monsoon rains make flooding a recurring threat to lives and infrastructure. In late September 2024, record-breaking rains inundated Kathmandu and surrounding regions, killing over 200 people and damaging dozens of major roads, blocking key highways out of the [1][2] capital . Flooding and landslides are regular hazards during the June–September monsoon, with [3][4] dozens of fatalities each year and millions of people exposed in high-risk areas . Urban growth and deforestation have worsened the problem; for example, Kathmandu's rapid expansion has disrupted natural drainage and narrowed rivers, contributing to unprecedented urban floods. These facts highlight an urgent need for localized flood risk information. Communities, travelers, and authorities in Nepal currently lack fine-grained warnings for when and where roads will flood. FloodWatch addresses this gap by predicting road-level flood risk in real time, helping people stay safe and keep the country moving during extreme weather.

Solution Overview

FloodWatch is an AI-driven platform that predicts the likelihood of road flooding on a street-by-street basis and visualizes it on a user-friendly web dashboard. Using weather forecasts, terrain and drainage data, and historical flood records, it computes flood-risk scores for each road segment ahead of heavy rains. The system highlights high-risk routes on a map and sends alerts to travelers, drivers, and local officials. By pinpointing flood hazards at the road level, FloodWatch enables safe route planning, timely road closures, and proactive response. The solution primarily helps **everyday commuters and transport authorities**: drivers can avoid flooded roads or plan alternate routes, logistics companies can adjust supply chains, and disaster agencies get actionable flood warnings for critical corridors. FloodWatch also aids city planners and emergency responders by providing a clear visual overview of vulnerable roads. In doing so, it brings AI [5][6] powered flood forecasting – a technique proven effective at scale – down to the neighborhood level for greater public safety.

Core Features and Functionalities

- **Road-Level Flood Risk Predictions:** Use machine learning models to analyze rainfall forecasts, elevation, drainage and past flood events. Output a risk score for each road segment (e.g. high, medium, low).
- **Interactive Map Dashboard:** Display a color-coded map with overlays of flood risk, current weather, and road network. Users can zoom, pan, and click on roads to view risk details.
- **Real-Time Alerts:** Provide push-notification or SMS/email alerts when flood risk exceeds thresholds on certain routes, enabling drivers to change plans. [1]
- **Route Planning and Routing Integration:** Allow users to query safe routes between two points. The system can suggest detours that avoid high-risk areas, akin to navigation apps but factoring

flood risk.

- **Historical Analysis & Reporting:** Show past flood events on the map and provide summaries of how risk levels evolve during a storm. This helps analysts refine models and users understand trends.
- **User Feedback Loop:** Include an option for users to report flooded roads (crowd-sourced hazard data), which can improve future predictions.
- **Administrative Dashboard:** Offer tools for disaster managers to filter the data (e.g. show only critical infrastructure) and export reports.

Technology Stack

- **Data Engineer / GIS Specialist:**
- *Tools:* Python, Pandas, GeoPandas; QGIS or Google Earth Engine; PostgreSQL/PostGIS database.
- *Role:* Gather and preprocess data. This member sources open datasets (e.g. Shuttle Radar Topography Mission (SRTM) for elevation, OpenStreetMap for roads, historical flood/precipitation records) and cleans/maps them into geospatial formats. They build ETL (extract-transform-load) pipelines to update the database with new weather and flood observations.
- **Machine Learning Engineer:**
- *Tools:* Python, TensorFlow or PyTorch, Scikit-learn.
- *Role:* Develop and train predictive models. This member designs the AI algorithms that take inputs like forecasted rainfall, upstream river flow, and terrain slopes to estimate flood probability on roads. They iterate on model features (e.g. soil saturation, nearby river levels) and validate accuracy. The ML Engineer integrates the model into the system so it can generate risk scores in near real-time.
- **Backend / API Developer:**
- *Tools:* Python (Flask/Django) or Node.js (Express); REST APIs; PostgreSQL/PostGIS.
- *Role:* Build the server infrastructure. This person creates the backend services that run the flood risk model on updated data, stores results in the database, and provides an API for the frontend. They implement user authentication if needed and ensure the system can handle concurrent requests (e.g. many users or sensors reporting data).
- **Frontend / UX Developer:**
-

Tools: JavaScript, React or Vue.js, Leaflet or Mapbox GL for mapping.

- *Role:* Design and code the web dashboard. This member uses mapping libraries to visualize data from the API. They create intuitive UI elements (filters, map legends, alerts) so that non-technical users can easily see flood risk on the map. They ensure responsiveness and clarity on desktop and mobile browsers.

- **DevOps / Cloud Engineer (optional in a small team):** 2

  - *Tools:* Docker, AWS/GCP, GitHub/GitLab for CI/CD.
  - *Role:* Deploy and maintain the application. This role sets up cloud servers or containers to host the API and dashboard, automates deployments, and manages logs/monitoring. They ensure the system is scalable and data pipelines run on schedule.

Each team member collaborates closely, with roles often overlapping in a hackathon. Together they use open-source tools and cloud services (e.g., AWS Lambda or Google Cloud Functions to run models on schedule, or Firebase for hosting) to accelerate development.

System Architecture and Data Flow Description

FloodWatch's architecture has three main layers: **Data & Processing**, **Backend Services**, and **User Interface**.

- **Data & Processing:** Multiple data sources feed into the system. Weather forecast APIs (e.g. OpenWeatherMap or government meteorological service) provide predicted rainfall. Terrain and drainage data (DEM from NASA SRTM) and road network data (from OpenStreetMap) supply the geographic context. Historical flood records and real-time sensor inputs (optional IoT gauges or user reports) add empirical calibration. These inputs are ingested by the Data Engineer, who preprocesses them (e.g. resampling rainfall maps to road segments, normalizing elevation values, etc.). The preprocessed data is then fed into the **AI Flood Risk Model**. This model – running as a service or scheduled job – computes a flood probability or water depth estimate for each road segment. The output is stored in a central **database** (e.g. PostGIS) along with metadata (timestamp, confidence).

- **Backend Services:** A server (built in Flask/Django or Node.js) queries the database for the latest flood risk data. It exposes a RESTful API that can return risk levels, safe-route suggestions, and historical charts on demand. For example, when the frontend requests risk for "Kathmandu Highway-1", the backend sends back the current score and recommended action (e.g. "Risk = High. Avoid during monsoon."). If an alert system is in place, the backend can also push notifications via SMS/email when risk crosses thresholds. The backend is the bridge between raw data and user view; it ensures the model runs on schedule and can also log user feedback to refine predictions.

- **User Interface (Dashboard):** The React/Vue web app calls the backend API and renders the information on a map. Color-coded overlays mark high-risk roads in red, moderate in orange, and safe roads in green. Users can click or tap any road to see details (expected flood depth, time of peak risk). The UI also provides controls to simulate different forecast scenarios or switch between "current" vs "24-hour forecast". During a live event, a timeline slider may show how the risk spreads across regions over time. Because the UI is web-based, it can be accessed on smartphones or tablets by users on the move.

This end-to-end flow (from data collection → risk computation → user alert) leverages known geospatial information system (GIS) and big data techniques in disaster management[7]. In summary: the system constantly ingests meteorological and geographic data, runs the predictive model to produce road risk scores, and then serves these results to users through an interactive dashboard. All data flow is automated, making FloodWatch a proactive early-warning tool rather than a passive map.

3

Development Workflow (10 Days)

1. **Day 1 – Kickoff & Research:** Define team roles and project scope. Gather sample datasets: download elevation maps, road network from OpenStreetMap, and test weather forecasts. Sketch the dashboard mock-ups.
2. **Day 2 – Data Pipeline Setup:** Build initial ETL scripts to import and preprocess road network and elevation data. Get a basic database (PostGIS) running.
3. **Day 3 – Weather Integration:** Implement code to fetch live weather forecasts via an API. Preprocess this meteorological data to align with map coordinates.
4. **Day 4 – Model Prototype:** Develop a simple flood model (e.g. using statistical rules or a pre-trained regression/NN). Train on any available historical flood data or terrain hazard data. Test outputs on known flood events.
5. **Day 5 – Backend & API:** Set up the server framework. Create API endpoints for "getRoadRisk" and "submitUserReport". Integrate the flood model into the backend (trigger it on new forecasts).
6. **Day 6 – Dashboard Skeleton:** Begin frontend development. Use Leaflet/Mapbox to render the map. Connect to API to fetch and display risk scores as an overlay (e.g., coloring roads by risk).
7. **Day 7 – Feature Integration:** Add UI controls: allow toggling forecast hours, filtering by risk level, displaying risk details on click. Refine the model with improvements (e.g. add more features like drainage density).
8. **Day 8 – Alerts & Feedback:** Implement alert logic (e.g. highlight alerts banner on the dashboard). If time, add an SMS or email alert via a service like Twilio. Enable a simple "I see flooding here" button on the map to collect user feedback.
9. **Day 9 – Testing & Refinement:** Run system tests using historic storm scenarios. Fix bugs in mapping or API. Optimize performance (e.g. cache API calls). Ensure the UI is responsive and understandable. Prepare example use cases.
10. **Day 10 – Finalization & Demo Prep:** Polish documentation and presentation slides. Deploy the app to a test server or GitHub Pages. Walk through the system end-to-end to ensure everything works.

Prepare a demo script highlighting how FloodWatch predicts and displays road flood risks.

This agile workflow, with clear daily goals, ensures a working prototype by day 10. Tasks are often done in parallel by different team members (for example, while the ML Engineer refines the model, the Frontend Dev can improve the map UI).

Data Sources and Tools/APIs Used

- **Road Network & Geography:** OpenStreetMap (OSM) data for Kathmandu and Nepal's highways (via OSM APIs or downloads). SRTM/ASTER global elevation data for terrain profile. Possibly NASA or USGS databases for hydrological features.
- **Meteorological Data:** OpenWeatherMap or Nepal's Department of Hydrology and Meteorology APIs for real-time and forecast rainfall. Alternatively, use NOAA or ECMWF global forecast data. Historical precipitation records for model training (from NHMP or NASA POWER).
- **Flood and Drainage Data:** Global Flood Database or national disaster reports for past flood events. (If available) river gauge readings or simulated watershed models.
- **Mapping & Visualization:** Mapbox or Leaflet for interactive maps; Mapbox GL JS or Google Maps API for base layers. D3.js or Chart.js for any in-dashboard charts. 4
- **Machine Learning & Data Processing:** Python libraries (Pandas, NumPy) for data wrangling; Scikit learn, TensorFlow, or PyTorch for building the predictive model. GeoPandas for spatial joins. QGIS for any manual GIS analysis.
- **Backend & Infrastructure:** Python Flask/Django or Node.js for the API; PostgreSQL with PostGIS extension for spatial database; Docker for environment consistency; GitHub for version control. Cloud platforms (AWS, Google Cloud) for hosting the app or using serverless functions.
- **Alerting & Communication:** (Optional) Twilio or similar SMS/email API for sending real-time flood alerts. Messaging platforms (like Slack or Telegram) for team coordination.
- **Collaboration Tools:** GitHub/GitLab for code, Google Drive or Notion for documentation, and agile tools like Trello for task tracking.

These choices prioritize open data and scalable tools. For example, OpenStreetMap provides free road maps essential for plotting flood risk without licensing issues. The use of popular ML frameworks (TensorFlow/PyTorch) and cloud services ensures the system can scale if FloodWatch grows beyond a prototype.

Potential Impact and Future Expansion

FloodWatch can save lives and resources by **improving flood awareness at the street level**. In Nepal's [4] context, where monsoon floods routinely affect over a million people each year , a tool like this helps commuters avoid hazards, and allows emergency teams to preemptively close or reinforce vulnerable roads. Timely route alerts reduce accidents and economic losses (e.g., less damage to vehicles, smoother supply chains). By visualizing flood risk, the project also educates communities about climate-related [6] dangers, complementing existing early-warning systems .

In the longer term, FloodWatch could evolve into a broader disaster resilience platform. Possible future work includes: - **Real-Time Crowd Data:** Integrate social media or user reports to validate and improve predictions during events.
- **Landslide Prediction:** Add models for predicting slope failures along mountain roads, another key hazard in Nepal.
- **Mobile App:** Develop a smartphone app for on-the-go notifications and offline map access. - **Expanded Coverage:** Apply the system to other flood-prone regions of Nepal or to neighboring countries with similar climate challenges.
- **Partnerships:** Work with Nepal's Department of Hydrology and Meteorology or NGOs to feed FloodWatch data into official disaster response networks.

By continuously incorporating new data and refining its AI models, FloodWatch has the potential to become an indispensable part of Nepal's climate adaptation toolkit. With extreme rains increasing globally, this system could also be a prototype for other regions: recent AI-based flood forecasting efforts now cover [5] hundreds of millions of people worldwide , showing strong interest in such tools. FloodWatch's focused, road-level insight would complement these initiatives by ensuring that, when heavy rains come, citizens can literally *see* the risk on every street ahead of time.

Sources: Contextual data and inspiration for this project come from recent reports on Nepal's floods and global AI flood forecasting efforts , which highlight the human and [1 2 3 4 5 6 7] infrastructural impact of monsoon disasters and the promise of data-driven prediction systems. [5]

[1] Nepal: Floods impacts driven by rapid urbanisation and climate change | PreventionWeb
https://www.preventionweb.net/news/rapid-urbanisation-and-climate-change-key-drivers-dramatic-flood-impacts-nepal

[2] More than 200 killed: What caused the devastating floods in Nepal? | Environment News | Al Jazeera
https://www.aljazeera.com/news/2024/10/2/what-caused-the-devastating-floods-in-nepal

[3] Landslides, floods kill 38 so far as monsoon rains lash Nepal | Reuters
https://www.reuters.com/world/asia-pacific/landslides-floods-kill-38-so-far-monsoon-rains-lash-nepal-2023-08-08/

[4] Nepal: 1.25 million people likely to be affected by disaster this year | PreventionWeb
https://www.preventionweb.net/news/around-125-million-people-nepal-likely-be-affected-disaster-year-ndrrma

[5 6] How Google helps others with AI flood forecasting
https://blog.google/technology/ai/expanding-flood-forecasting-coverage-helping-partners/

[7] Flood Monitoring System Architecture. | Download Scientific Diagram

https://www.researchgate.net/figure/Flood-Monitoring-System-Architecture_fig1_326946364 6