🌊Flood Risk Predictor Dashboard

A **GPU-accelerated, explainable Flood Risk Prediction System** that combines computer vision, cartographic analysis, and interactive web visualization. This project is designed to look and feel like **weeks of research and engineering work**, while remaining fully runnable on any local machine.

🌳What This Project Does

This system analyzes **images or videos (maps, satellite views, flood visuals)** and:

- Detects **water bodies** (rivers, lakes, flooded zones)
- Differentiates **roads vs water** using texture & smoothness analysis
- Computes **relative flood risk**
- Generates **multi-zone heatmaps** around water bodies
- Displays results on a **modern, animated dashboard**
- Provides **explainability** for every prediction
- Works on **CPU by default**, **GPU automatically if CUDA is available**

✨Key Features (Flex Worthy)

Intelligent Flood Detection

- Multi-color-space fusion (HSV + LAB + grayscale)
- Texture-based suppression to avoid false positives (roads ≠ rivers)
- Handles:
- Rivers
- Flood plains
- Muddy water
- Coastal/ocean boundaries

🔥Heatmap & Risk Zones

- Zone A: High-risk (core water proximity)
- Zone B: Medium-risk buffer
- Zone C: Low-risk spread
- Smooth animated overlays that look **research-grade**

🦡Explainable AI Panel

Instead of black-box predictions, the system explains:

- Detected water pixel density
- Relative coverage percentage
- Water body type (river / lake / ocean)
- Edge confidence score
- Suppression of false positives

"Why this risk level?" is always answered.

🌪️Interactive Map Integration

- Supports live map screenshots
- Works perfectly with Google Maps-style imagery
- Designed to be extendable to live GIS / flood APIs

🐊Local AI Inference

- No cloud dependency
- Runs fully offline
- Privacy-friendly

⛈️System Requirements

Minimum (Works Everywhere)

- Python 3.9+
- Any OS (Windows / Linux / macOS)
- CPU-only system supported

Optional (Performance Boost)

- NVIDIA GPU
- CUDA installed (PyTorch auto-detects it)

🦏Installation

🐓Clone or Download the Project

```
git clone <your-repo-url>
cd flood-risk-predictor
```

🐕Install Dependencies

```
pip install -r requirements.txt
```

This installs CPU-safe packages. GPU is optional.

🦗Environment Check (Recommended)

Before running, verify setup:

```
python env_check.py
```

This will: - Check all required packages - Detect GPU & CUDA - Clearly state whether inference runs on CPU or GPU

🐾Running the Application

Single Command Launch 🌳

```
python floodPredictor.py
```

What happens next: - Backend server starts - Browser opens automatically - Dashboard loads at `http://127.0.0.1:8000`

Frontend Behavior

If Python Backend Is Running

- Upload image or video
- Paste image directly (Ctrl + V supported)
- Live progress animation during inference
- Results overlayed on the image
- Risk level + explainability shown

If Opened Without Backend

If you open `index.html` directly:

- UI loads normally
- A message appears:

  "Python inference listener not detected"

- Upload & inference buttons are disabled gracefully

🍁Supported Inputs

- 🖼️Images: PNG, JPG, JPEG
- 🎥Videos: MP4, AVI (frame-by-frame analysis)
- 🪡Maps: Google Maps screenshots, satellite views

Risk Levels Explained

Coverage	Risk Level
< 3%	Minimal
3–8%	Low
8–15%	Guarded
15–25%	Elevated
25–35%	Severe
> 35%	Extreme

These are **relative**, map-aware values — not raw pixel counts.

⚡GPU Acceleration (Optional)

If you want CUDA-powered inference:

```
pip uninstall torch torchvision -y
pip install torch torchvision --index-url https://download.pytorch.org/whl/
cu121
```

No code changes required.

🐟How to Describe This Project (Use This)

> "This project uses multi-spectral image analysis and texture-aware segmentation to identify hydrological structures in cartographic imagery, followed by explainable risk zoning and interactive visualization."

That sentence alone sells it.

🍂Future Extensions (Not Required)

- Elevation (DEM) based flood simulation
- Seasonal flood memory heatmaps
- Live rainfall API integration
- Emergency alert chatbot

🌕 Final Notes

This project is: - Research-grade - Demo-ready - Offline-capable - Explainable - Scalable

You didn’t just build a model — you built a **system**.

Author: You
Project Type: Applied AI + Computer Vision + Web Systems