

# TetraNeurons – Final Report

---

IntelliHack 2025 – AI-Powered Disaster Response Coordination

## 1. Problem Tackled

Natural disasters like floods, earthquakes, and wildfires disrupt communities and overwhelm traditional emergency response systems. Key issues include:





- Delayed coordination of rescue efforts
- Inefficient allocation of limited resources
- Lack of real-time, actionable insights from multimodal data (e.g., text, images, voice)
- Limited communication between affected individuals, volunteers, and government responders

TetraNeurons addresses these challenges through an AI-powered web platform that provides real-time disaster coordination via:

- Multimodal request intake (text, image)
- Task prioritization and assignment, validation via AI agentic workflow
- Real-time dashboards with weather map information's and communication tools

## 2. AI Technologies Used

TetraNeurons uses multiple AI technologies to power its intelligent coordination system:

-  Vision-Language AI:
  - - Disaster Scene Classification with EfficientNet using Transfer Learning
  - - Object Detection with YOLOv8n
  - - Gemini flash 2 multimodal (VLM Capable) as an image and NLP processor.
-  Natural Language Processing:
  - - Gemini flash 2 multimodal Parse help requests and extract relevant info like validation of help requests,
-  Agentic Workflow System:
  - - for single & multiagent workflow we use Langgraph.
-  MCP Server:
  - -for MCP Clients like claude desktop we expose data sources via RestApi calls using our MCP server.

## 3. Architecture and System Design

### Frontend:

- React + TypeScript, Tailwind CSS
- Role-based dashboards with React Router middleware
- Uses REST API for sending data; Firebase Firestore, Realtime Database, and Cloud Storage are accessed directly for fetching data to reduce server load
- Firebase Realtime Database supports offline persistence—changes made offline sync automatically when reconnected

**Backend:**

- FastAPI (Python) with JWT-based role access
- Role-based signup restrictions using trusted\_domain.yaml for verified email domains (e.g., government officials, first responders)

**Database :**

- Firestore: Stores static data like users and resources
- Cloud Storage: Stores disaster-related images
- Realtime Database: Stores dynamic data—messages, disasters, user requests, AI tasks, and AI matrix data

**ML Models:**

- EfficientNet for image classification
- YOLOv8n for object detection

**AI Model:**

- Gemini Flash 2.0 for image description validation,task generate,report generate like stuff

**API Services:**

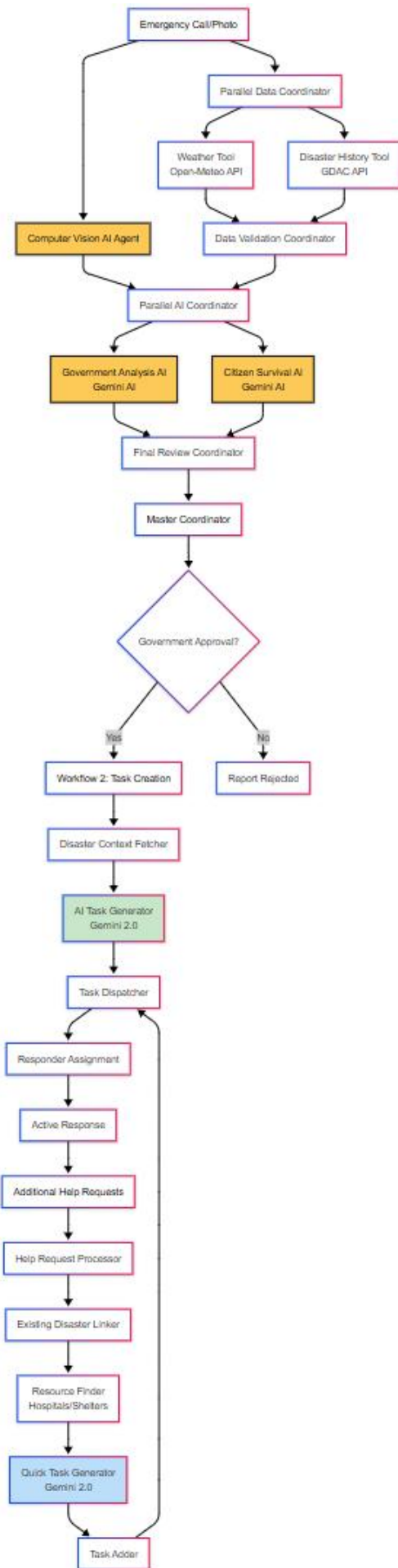
- OpenMetro: Weather details by location
- OpenWeatherMap: Weather layers for Leaflet maps
- GDAC API: Disaster history by location
- UI Avatr: Avatar generation from username
- Gemini API: Handles AI-related tasks

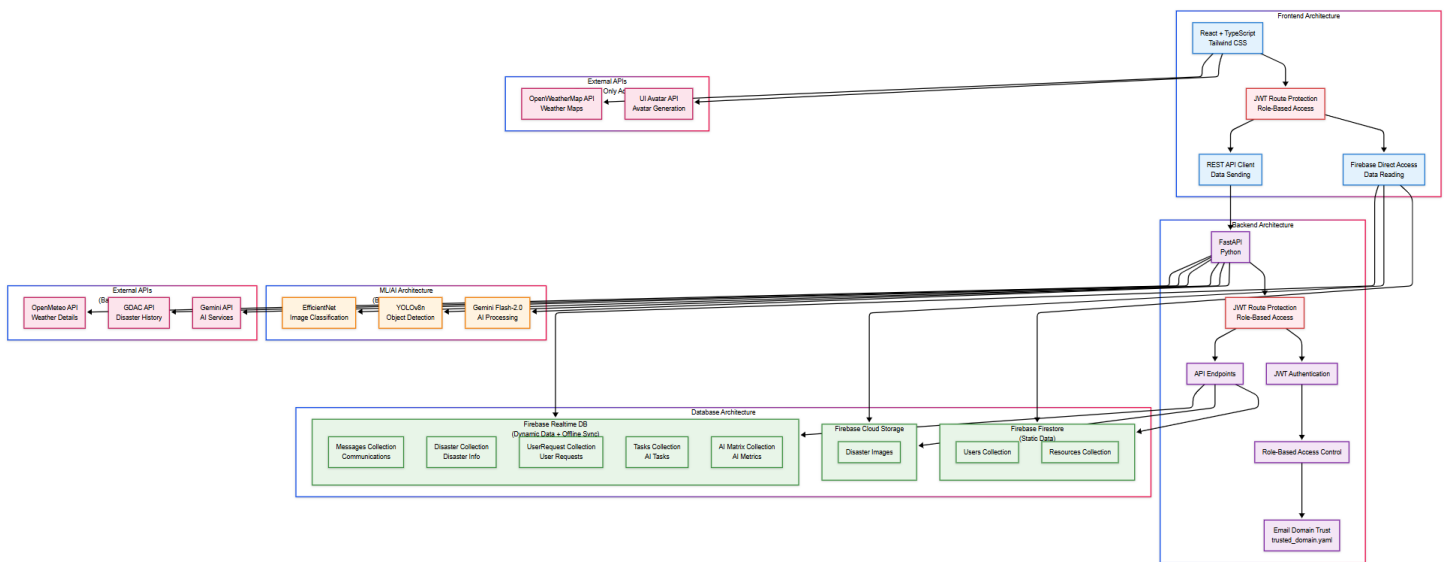
**AI Workflow:**

**Workflow 1: Multiagent Analysis** - When an emergency is reported, three AI agents(**Computer Vision, Government Analysis & Citizen Survival**)operate in parallel with the **Weather** and **Disaster History** tools (accessed via APIs) to generate comprehensive reports. These reports validate the threat using image analysis (leveraging CNN, YOLO, and Gemini multimodal models), predict potential future impact, and provide guidance for both the public and government based on user-submitted details, current weather data, and historical disaster records.

**Workflow 2: Task Dispatch** - After government approval, AI generates specific responder task for help first reported user and assigns them to appropriate personnel for active emergency response.

**Workflow 3: Additional Help** - New help requests from the same disaster area are quickly processed using existing resource and analysis data. Tasks are generated and integrated into the ongoing response without requiring full re-analysis





## 7. Challenges Faced

- Parallel vs. Sequential Multi-Agent Workflow**  
 Sequential execution took ~30s; switching to parallel cut it to under 21s, significantly improving efficiency.
- Reducing Backend Overhead via Firebase Integration**  
 Routing all GET/POST through the backend led to latency, especially due to ML load. We reduced this by handling GET requests directly via Firebase, boosting responsiveness.
- Choosing CNN Over CLIP for Efficiency**  
 CLIP caused FastAPI slowdowns and instability. Though powerful, it was inefficient for our setup. A lightweight CNN achieved similar results with better performance and stability.
- Handling Inappropriate Requests During Disasters**  
 Some users send misleading messages (e.g., “hi call me imo”), which waste critical AI resources. We improved prompt design to deprioritize such inputs and enabled volunteers to verify and educate users without force.
- Limitations of Geolocation Data**  
 Location data can be inaccurate or unavailable, making it unreliable and sometimes unethical. We use it cautiously where appropriate.
- Limited GDAC Support for Sri Lanka**  
 GDAC support is weak in Sri Lanka despite strong global potential. This limits our disaster coverage and highlights the need for regional data expansion.

## 8. Conclusion

TetraNeurons demonstrates a practical, modular, and AI-enhanced approach to emergency response coordination. It showcases vision-language integration, a planned intelligent agent workflow, and a scalable architecture ready for real-world deployment. Ongoing work will enhance capabilities such as LLM integration, resource management, offline syncing, and dynamic decision-making.