National AI-Optimized Disaster Platform for Sri Lanka

Hackathon Submission: Problem, Solution, Technical Architecture, and PoC Guidance

docs/problem.md

Problem Statement

Sri Lanka faces significant challenges with disaster response and logistics during floods, cyclones, landslides, and other natural emergencies. Key issues include:

- **Outdated Communication:** Citizens and responders lack up-to-date, localized risk alerts and real-time instructions.
- **Resource Bottlenecks:** Relief supplies are often misallocated due to poor visibility of on-ground needs and changing threat zones.
- **Slow, One-way Notifications:** Current government communications are broad, delayed, and often fail to reach mobile-first users promptly.
- **Lack of Citizen Agency:** Citizens cannot easily report urgent dangers, request immediate help, or receive actionable AI-assisted guidance tailored to their situation.
- **Fragmented Data:** Ministries and emergency responders lack a transparent, unified system for monitoring needs, supply movements, and community feedback.

Impact:
These gaps cause avoidable loss of life, inefficiency in relief deployment, and erode public trust during crises. Fast, accurate, and inclusive disaster communication and resource matching are critical for resilience in a climate-vulnerable nation like Sri Lanka.

docs/solution.md

Solution Overview

1. Citizen Disaster Mobile App

- **Live Disaster Index:**
  - Intuitive, color-coded risk map of all active and potential disasters (floods, storms, landslides), with icons and danger levels by locality.
- **Real-time Alerts & SOS:**
  - Automated push notifications for new/existing disasters; “SOS” one-tap emergency request tied to geolocation.
- **Need & Safety Reporting:**

    - Citizens report their status (“I’m safe/danger”, “Need food/medical”, “Blocked road”) via forms, voice, or photo in-app or via SMS.
- **AI Safety Assistant:**
    - Built-in chatbot for “Can I travel to X?”, “Is Y river safe for swimming today?”, or “Where is the nearest shelter?”—with guidance personalized to disaster context, location, and user vulnerability.
- **Offline-First & Multilingual:**
    - Supports Sinhala, Tamil, and English; caches risk maps for offline access.

2. Government Admin Web Dashboard

- **Disaster Management Portal:**
    - Officials update disaster areas, upload safety advisories, approve bulk notifications, and monitor risk clusters via intuitive interface.
- **Resource Allocation & Needs Heatmap:**
    - Visualizes reported needs by type and location; synchronizes supply status from logistics APIs.
- **AI Supply Optimizer:**
    - Recommends warehouse/relief deployment based on evolving need clusters, road blockages, and supply perishing rates.
- **SOS & Incident Monitor:**
    - Real-time feed of citizen SOS signals, prioritized by severity and geographical proximity.

3. Transparency & Feedback Layer

- **Allocation Tracker:**
    - Live dashboard for both ministries and the public, showing relief shipments in-transit, delivery confirmation, and heatmaps of fulfilled/unfulfilled needs.
- **Feedback and Correction Loop:**
    - Citizens can confirm aid delivery or flag missed/duplicate support, closing the loop.

DPI Stack Integration

- **SLUDI:** Secure authentication for citizens, responders, and admins.
- **NDX:** Consent-based sharing and pull of disaster, demographic, supply chain, and contact info data.
- **PayDPI:** If financial assistance or digital payments (e.g., relief top-ups) required.

Impact

- **Faster, life-saving response** to disaster threats and citizen emergencies.
- **Optimized, transparent resource flows** that reduce wastage.

- **Empowered communities** with two-way, accessible disaster communication.
- **Scalable for global export** as a disaster resilience platform.

Technical Architecture

High-Level Architecture Components

Layer	Component	Tech Stack Example	Description
User Frontend	Mobile App (Citizen)	Flutter/Kotlin/React Native	Disaster alerts, reporting, AI chat, SOS
	Web App (Gov/Admin)	React/Vue + Node/Express	Dashboard, map editing, resource mgmt
API Gateway	Unified REST API, WebSocket Push	Node.js/FastAPI, GraphQL	Auth, data exchange, real-time updates
AI/Logic Layer	Disaster Risk & Resource Engine	Python (scikit/transformers)	Risk color-coding, supply optimization
	Conversational AI (Assistant)	Rasa/Dialogflow	Natural-language Q&A, safety advice
Backend & Data	Core Relational DBs & GIS	PostGIS, MongoDB	Disaster zones, needs, supplies
	Messaging & Queue	Firebase, Twilio	SMS/notification delivery
DPI Stack	SLUDI, NDX, PayDPI Integration	OAuth2, REST	Auth, data consent, digital payments
Security		HTTPS, JWT, RBAC	End-to-end encryption, access control
DevOps	Containerization, K8s	Docker, Kubernetes	Deploy at scale, auto-heal, easy update

Communication Flows

- **Mobile App ↔ API Gateway ↔ Disaster Data Layer (NDX)**
  - Live data push/pull to citizens and agent apps, risk and supply updates, report submission
- **Admin Web Dashboard ↔ API ↔ Data Layer ↔ AI Engine**
  - Add/update disasters, deploy real-time warnings, manage supply algorithms
- **AI Assistant ↔ Citizen Data ↔ Open Knowledge + NDX**
  - Q&A about personal and geographic risk; routes to live agent if AI cannot help
- **All ↔ Security/Ops**
  - SLUDI-backed login, fine-grained permission, auditing

src/ (Proof-of-Concept Code Guidance)

A minimal working prototype (POC) could be composed of:

- Flutter or React Native mobile app:
  - Home screen listing disaster events from NDX sandbox
  - Color-coded risk map component (pulls sample disaster data via API)
  - SOS button posts to NDX
  - Simple chat interface powered by Dialogflow or Rasa with dummy Q&A
- Node.js or Python backend:
  - REST endpoints to serve sample disaster/event data, accept status and SOS posts, and expose analytics API
  - Websocket/Push API for disaster notifications
- React admin dashboard:
  - Form to add/edit disasters (updates NDX), control notifications, monitor submitted needs (testing with sample CSV/JSON data)
- SLUDI integration for mock login, and one PayDPI-triggered test payment for demo (optional)
- README.md:
  - Setup, build, run instructions; API samples; user credentials for test logins

Submission Checklist

- **docs/problem.md:** Clear, evidence-backed explanation of disaster communication/logistics pain point and societal impact.
- **docs/solution.md:** End-to-end platform description: citizen app, AI advisor, admin dashboard, DPI utilization, and value proposition.
- **src/:** Demo (even basic) of live risk map, SOS, reporting, chat, and gov-side event control.
- **README.md:** Complete instructions for local setup and test.

Conclusion

This platform addresses a high-priority, under-served need with AI-empowered, DPI-integrated technology. It is highly suited for hackathon demonstration and pilot adoption by government, and is designed for scale and real-world impact.

1. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/83149372/c51bb48a-186d-4be4-b145-8b54294daa2e/Pilot-Government-Application-Pack-for-the-ReviveNation-Hackathon_v1.pdf
2. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/83149372/f4684f4d-c678-4de8-b1c4-7766b23c7ed1/Hackathon-Proposal-Submission-Guidelines.pdf