Core MVP Scope (Minimum Viable Product)

Frontend (React + Bootstrap + Google Maps API)

- Basic layout: map, navigation bar, zone warning
- Google Maps integration
- Markers or shaded zones for high-risk areas
- Location access prompt
- Optional: simple login/register screen

Backend (Node.js + Express.js + MongoDB/SQL)

- Serve API endpoint for flagged zones (e.g. `/api/zones`)
- Store hardcoded or scraped crime data (for demo)
- Optional: fetch news data using News API and extract locations with NLP (can be simplified)

🟩 Day 1 — Core System Setup & Basic Functionality

🕗 **08:00–09:00 — Project Setup & Planning**

- Set up GitHub repo (or GitLab)
- Create folder structure: `client/` (React) and `server/` (Express)
- Install dependencies:

    - `npm create vite@latest` (for React)
    - `express`, `cors`, `dotenv`, etc. on backend
- Test Google Maps API key and load map

🕘 **09:00–12:00 — Map UI & Geolocation (Frontend)**

- Load Google Map in React
- Center on default African city (e.g. Johannesburg, Nairobi)
- Add hardcoded **red zones** (polygons or circles)
- Use HTML5 **Geolocation API** to get user position
- Alert if user is inside a red zone

🕛 **12:00–13:00 — Lunch / Break**

🕐 **13:00–16:00 — Backend API (Node.js + Express)**

- Setup Express server
- Create `/api/zones` route that returns mock red zone data
- Add MongoDB connection (Atlas or local)
- Define simple Zone schema: `{ name, lat, lng, radius, type }`
- Optionally test a basic Python script to fetch news (store separately for now)

🕓 **16:00–18:00 — Connect Frontend and Backend**

- From React, fetch zone data from `/api/zones`
- Display zones dynamically on the map
- Trigger alert when user enters a zone from backend

🕕 **18:00–20:00 — UX Enhancements**

- Style using Bootstrap: layout, alert banners, buttons
- Add zone info card: "⚠️ High risk: Assault reports in this area"
- Optionally let user simulate location movement

🕗 **20:00–22:00 — Testing + Deploy First Demo**

- Deploy React app on **Netlify**
- Deploy backend with **ngrok** (or Azure App Service if ready)
- Test map, alerts, and API connection
- Record short screen demo or test walk-through
- Backup all your code!

🟨 Day 2 — Final Build, Stretch Features & Demo Polish

🕔 **05:00–10:30 PM** (17.5 hours)

🔥 🕔 05:00–08:00 — Early Morning Momentum

- ✅ **Bug Fixes & Refactoring** from Day 1 (UI glitches, API fetch errors, map quirks)
- ✅ **Final Testing** of core:
  - Map loads?
  - Red zones show?
  - User location works?
  - Alerts trigger?
- ✅ Add zone info overlays (crime type, time, etc.)
- ✅ Clean and comment your code

🚀 🕗 08:00–11:00 — MVP Expansion (Pick 1–2 Stretch Features)

⬇️ Suggested stretch features:

- 🔍 **NewsAPI integration** (real-time news headlines about GBV)
- 🌍 **NLP or Python script** to extract location names from news (optional; can simulate this with a JSON file)
- 📍 **Zone type categorization**: Assault, mugging, harassment, etc.
- ✍️ **Anonymous Report Form** (React form → MongoDB): Let users report new red zones

---

🍽️ 🕛 11:00–12:00 — Light Lunch / Reset

🎨 🕛 12:00–15:00 — UX/UI Polishing & Branding

- ✅ Improve alert design (Bootstrap cards, warning icons, colors)
- ✅ Add a clean banner/nav with **app name/logo** (e.g., *AfriSafe* or *HerZone*)
- ✅ Adjust mobile responsiveness (Bootstrap grid or Flexbox)
- ✅ Add a loader or “Checking zone safety…” spinner for a nice touch

---

💾 🕒 15:00–17:00 — Final Integration & Deployment

- 🔧 Final Netlify deploy of frontend
- 🔧 Azure/ngrok for backend — ensure API is reachable
- ✅ Test full flow: open site → location → alert → map zones

- ✅ Backup to Git + shareable link for judges

🧑‍💼 🕔 17:00–19:00 — Demo Script & Presentation Prep

- ✅ Prepare pitch (2–3 min):
  - **Problem**: GBV in Africa, unsafe zones
  - **Solution**: Real-time map + alerts + crowd data
  - **Tech Stack**: React, Node, Maps API, MongoDB, News API
  - **Impact**: Empowering women with proactive safety
- ✅ Optional: Prepare slides (1–3 clean slides)
- ✅ Run through your demo 2–3 times
- ✅ Record backup screen demo (in case live fails)

✨ 🕖 19:00–22:30 — Final Polish + Cooldown

- 🎁 Last UX tweaks: hover tooltips, zone legend, alert colors
- 🧪 Full walkthrough again (in different locations, if possible)
- 📝 Final project summary (GitHub README or PDF)
- 🎤 Practice smooth transitions between talking and clicking
- 💤 Shutdown at 22:30 with confidence

✅ AI Use Case Options (Choose 1 for MVP)

🧠 Option 1: AI-Powered Location Tagging from News Headlines

Goal: Use AI/NLP to extract crime-related keywords and locations from news headlines or articles.

How it works:

1. Pull articles via **NewsAPI** or use static JSON articles.
2. Run a Python script that:
    - Extracts **locations** and **GBV-related keywords** (like “rape,” “abduction,” “assault”).
    - Matches locations to lat/lng coordinates (use Google Geocoding API).
3. Add resulting points to your red zone map dynamically.

This shows the AI is **actively detecting** danger zones based on real-time news.

Libraries you can use:

- `spaCy` or `nltk` for Named Entity Recognition (NER)
- `geopy` or Google Geocoding API for converting locations to coordinates

---

🔍 Example:

python
CopyEdit

```
import spacy
from geopy.geocoders import Nominatim

nlp = spacy.load("en_core_web_sm")
geolocator = Nominatim(user_agent="zone-mapper")

headlines = [
    "Woman assaulted in Soweto train station",
```

```
    "Kidnapping reported near Cape Town CBD",
    "Protest erupts over GBV incidents in Khayelitsha"
]

zone_data = []

for headline in headlines:
    doc = nlp(headline)
    for ent in doc.ents:
        if ent.label_ == "GPE":  # GPE = Geo-Political Entity
            location = ent.text
            try:
                loc = geolocator.geocode(location)
                if loc:
                    zone_data.append({
                        "location": location,
                        "lat": loc.latitude,
                        "lng": loc.longitude,
                        "type": "news_crime"
                    })
            except:
                continue

print(zone_data)
```

Then you send `zone_data` to your backend API.

🛑 Option 2: AI-Based Threat Prediction (Simulated Model)

Goal: Use a simple ML model to **predict risk level** based on:

- Time of day
- Zone location
- News frequency

- Community reports

You can **simulate the model** using `scikit-learn` with fake or limited data (just to demo concept). Example: “If 5+ reports in area + nighttime = high risk.”

💬 Option 3: AI Assistant or Chatbot (Low Priority for Now)

If you want to add a virtual assistant later:

- Use OpenAI's API or Rasa
- Answer: “Is this area safe?” → check zone DB
- For demo: hardcode one or two Q&A pairs

But this is **not the strongest use of AI** for your red zone demo right now.

✅ Recommended for Hackathon:

👉 Go with Option 1 (AI-powered location extraction from news)

- Lightweight
- Works well with your data pipeline
- Feels very real-world and powerful

🔧 Quick Integration Plan for Day 2:

Time	Task
05:00–06:00	Install `spaCy`, set up location extractor script
06:00–07:30	Test with 5–10 news headlines (real or mock)

07:30–08:00	Store output in JSON or send to backend API
08:00–09:00	Display AI-detected red zones on the map
Optional	Label them as “AI Detected” zones in frontend

🧠 Final Touch:

In your demo/pitch, say:

> “We use an AI-based NLP pipeline to analyze news headlines and extract crime-related locations. This lets us dynamically flag new red zones even before official reports come out — providing proactive protection for users.”