Project Imara: Master Technical Specification V2.0 Document Version: 2.0 (Hybrid Intelligence Edition)

Date: November 28, 2025

Author: Onyeka Nwokike

Architecture: Django Hybrid (Telegram + Web + Reasoning AI)

Infrastructure: Zero-Cost Cloud (Render + Neon + Groq + Gemini + Brevo)

1. Executive Summary & Mission Statement

Imara (Swahili for "Strong" or "Stable") is a Zero-UI Digital Bodyguard designed to protect women and girls from online gender-based violence (OGBV) directly within the messaging apps they use every day.

Our Mission: To democratize digital safety by providing an immediate, accessible, and legally robust protection tool that requires no app downloads. We protect survivors through real-time "Reasoning AI" that filters noise, offers advice for minor issues, and autonomously routes forensic evidence to authorities for serious threats.

This document serves as the single source of truth for the platform's functional requirements, data flows, and architectural standards for the Power Hacks 2025 Hackathon.

2. The "Zero-Cost" Architecture

The platform is engineered on a zero-cost, high-availability infrastructure. It decouples the Interface (Telegram/Web) from the Brain to ensure maximum accessibility. We utilize a "Hybrid Intelligence" model to optimize for both speed and visual understanding.

2.1. Compute (The Brain)

Host: Render Web Service (Free Tier).

Role: Hosts the Django Application, handles Telegram Webhooks, serves the Web Sentinel, and manages the Email Dispatcher.

Keep-Alive Strategy: External pinging (UptimeRobot) every 5 minutes to prevent the container from sleeping, ensuring the bot responds instantly during the demo.

2.2. Database (The Memory)

Engine: PostgreSQL (via Neon.tech).

Role: Stores Users, Evidence Logs, and the "Authority Rolodex."

New Model: `AuthorityContact` table. This maps locations (e.g., "Lagos", "Nairobi") and keywords to verified police/NGO email addresses.

2.3. The Hybrid AI Engine (Reasoning Core)

We use two distinct brains to handle different types of data:

1. **Text & Speed Brain: Groq (Llama-3-70b-versatile).**
   - **Role:** Handles text chats and audio transcriptions (via Whisper). It performs "Chain of Thought" reasoning to triage threats in <0.5 seconds.
2. **Visual Brain: Google Gemini 2.5 Flash.**
   - **Role:** Analyzes uploaded screenshots. It uses native optical character recognition (OCR) to read abusive texts in images and understand the context (e.g., identifying a death threat in a WhatsApp screenshot).

2.4. Communications Layer (The Dispatcher)

Provider: Brevo (formerly Sendinblue) - Free Tier.

Role: Replaces the PDF generation engine. Sends "Official Forensic Report" HTML emails directly to authorities when a high-risk threat is confirmed.

3. Core Platform Infrastructure

3.1. Server Configuration

Framework: Django 5.0 (Python 3.11).

Web Server: Gunicorn with uvicorn workers.

Static Files: Served via White Noise.

Unified Service Layer: A single `services.py` module handles logic from both Telegram and the Web Form to ensure consistent AI processing.

3.2. Asynchronous Handling (No-Redis Approach)

Constraint: The free tier cannot run Redis/Celery reliably.

Solution: Utilization of Python Threading for non-blocking operations.

Implementation: When the AI decides to "REPORT," the Brevo email dispatch runs in a background thread so the user receives an instant confirmation reply without waiting for the SMTP handshake.

4. The Reasoning Engine (The "Triage" Brain)

We do not report everything. The system acts as a triage nurse, filtering noise to prevent spamming authorities.

4.1. The Data Model (AuthorityContact)

An admin-editable model acting as the routing directory. Fields: Agency Name, Email, Jurisdiction (City/State), Tags (e.g., "Cybercrime", "Domestic Violence").

4.2. The Reasoning Workflow (The Logic)

1. **Ingestion:** User sends text (Telegram) or uploads a screenshot (Web).
2. **Analysis:**
   - *If Text:* Groq Llama-3 analyzes the intent.
   - *If Image:* Gemini 2.5 extracts text and context.
3. **The Decision (JSON Mode):** The AI returns a structured JSON object:
   - `risk_score` : (1-10)
   - `action` : "ADVISE" or "REPORT"
   - `location` : "Lagos" (extracted from context)
4. **Execution:**
   - **Case A (Low Risk - e.g., Insults):** AI returns `action: ADVISE` . The System replies with blocking tips and mental health resources. **No email is sent.**

    - **Case B (High Risk - e.g., Doxing/Threats):** AI returns `action: REPORT`. The System looks up the "Lagos" contact in the DB and dispatches the Forensic Email.

5. The Interface Layer (The Omnichannel Sentinel)

5.1. Primary Interface: Telegram Bot

The "Guardian" Forward: Users forward abusive messages to the bot. Voice Protection: Users forward Voice Notes. Groq Whisper transcribes them for analysis.

5.2. Secondary Interface: The Web Sentinel (New)

Role: A safety net for users without Telegram and a "failsafe" for the demo.

Design: A clean Django Landing Page.

- **Hero:** "Report Online Violence Securely."
- **Roadmap Grid:** Shows icons for Telegram (Active), Web (Active), WhatsApp (Coming Soon), Instagram (Coming Soon).
- **Function:** A secure form to upload screenshots/voice notes that feeds into the same AI backend.

6. The Dispatch System (Replacing Evidence Vault)

We move from passive PDF storage to active email dispatch.

6.1. The "Official" Email Template

Engine: Brevo Transactional API.

Format: A rigid, legally formatted HTML email.

Content:

- **Header:** OFFICIAL FORENSIC ALERT - PROJECT IMARA.
- **Metadata:** Case ID, Timestamp (UTC), Detected Location.
- **Evidence:** The transcribed text or attached screenshot.
- **Footer:** Chain of Custody Hash & Verification Link.

Purpose: To land directly in the inbox of a police officer or NGO worker, prompting immediate action.

7. Implementation Strategy: Role-Based Execution

To ensure the project is delivered on time, the team is divided into specialized roles.

7.1. Developer 1 (Onyeka - The Architect)

Responsibility: Core Backend & Hybrid AI Logic. Tasks:

- Setup Django Project & Neon DB.
- Implement **Groq** integration for text triage.
- Implement **Gemini 2.5** integration for image analysis.
- Write the `decision_engine` logic that routes "ADVISE" vs "REPORT".

7.2. Developer 2 (The Builder - Frontend/Integration)

Responsibility: Web Interface & Brevo Dispatch. Tasks:

- Design the "Web Sentinel" Landing Page (Bootstrap).
- Create the Report Form (handling image uploads).
- **Brevo Integration:** Build the HTML Email Template and the Python function to send it.
- Ensure the "Coming Soon" grid looks professional for the pitch.

7.3. Developer 3 (The Logic Lead - Content/QA)

Responsibility: Data Integrity & The "Rolodex". Tasks:

- **Database Seeding:** Research and input real (or demo) emails for Police/NGOs in the `AuthorityContact` table.
- **Scenario Scripting:** Create specific inputs for the demo:
  - *Input 1:* "He called me stupid." -> Result: AI Advice.
  - *Input 2:* "He posted my address on X." -> Result: Police Email.
- **Prompt Engineering:** Ensure the AI strictly follows the JSON format.

7.4. Developer 4 (The Interface/Support)

Responsibility: Bot Configuration & Presentation. Tasks:

- Set up the Telegram Bot via BotFather.

- **The "Wow" Demo:** Coordinate the live demo so the judges see the email arrive on Developer 3's phone in real-time.
- **Pitch Deck:** Update the story to emphasize "Hybrid AI" (Groq + Gemini) and "Direct Action" (Email Dispatch).

8. Feasibility & Constraints Checklist

[x] Hosting: Render Free Tier (Spins down, fixed by UptimeRobot).

[x] Database: Neon Free Tier (Scales to zero, fixed by initial ping).

[x] Text AI: Groq Llama-3 (Free Beta) for instant text reasoning.

[x] Vision AI: Gemini 2.5 Flash (Free Tier) for screenshot analysis.

[x] Email: Brevo Free Tier (300 emails/day) for dispatch.

[x] Security: Hashing ensures evidence validity.

[illegible]