ChatGPT

United for the Missing – Developer Brief

This document contains all the information required to implement **unitedforthemissing.com** — a WordPress-based platform for coordinating efforts to locate missing persons. It is intended for the engineering team responsible for the custom plugin and n8n automation. All names and keys are consistent across sections; keep them unchanged when creating code or workflows.

---

1. Overview & Goals

Purpose: Build a public portal to centralize missing-person cases, disseminate verified information, collect sightings, and apply AI to prioritize leads. The system must be trustworthy, transparent, and respectful of privacy.

Target Users: - **Families & Guardians** — submit cases and receive status updates. - **General Public** — search cases, submit sighting reports, and learn how to help. - **Investigators & NGOs** — use the AI summaries, risk scores, and lead triage to focus resources.

Business Model: Nonprofit project funded by donations and grants. No advertisements or monetization from case data. Public donations support hosting, AI services, and volunteer coordination. Volunteers may contribute to case vetting and translations.

2. WordPress Site Structure

1. **Pages** (create in the admin dashboard):
2. `/search` – embed the search and map via shortcodes `[u4m_search]` and `[u4m_map]`.
3. `/submit` – form to submit a case via `[u4m_submit_case]`.
4. `/report` – form to submit a sighting via `[u4m_report_sighting]`.
5. `/help`, `/resources`, `/about`, `/press`, `/blog`, `/contact`, `/safety`, `/privacy`, `/terms`, `/faq` — static content pages using copy provided earlier.
6. `Missing Persons` archive and single templates are provided by the plugin (archive slug `/missing`).
7. **Menus & Settings**:
8. Set permalink structure to “post name”.
9. Configure primary menu with “Search”, “Report a Sighting”, “Help”, etc.
10. Install reCAPTCHA v3 for forms and a cache plugin (exclude `wp-json/u4m/v1/*`).
11. Create WordPress roles `Case Manager` and `Reviewer` with capabilities defined in plugin.
12. Publish privacy policy and terms; add cookie banner if required.
13. **Security & Privacy**:
14. Enable HTTPS and HSTS.

15. Limit REST API writes (`/u4m/v1/cases*`) to authenticated users with `edit_posts` capability.
16. Public API route `/u4m/v1/sightings` accepts sighting reports without login.
17. Use Application Passwords for n8n to authenticate via Basic Auth.
18. Redact or mask personally identifiable information when consent is not provided (handled in `class-u4m-security.php`).

3. Plugin Specification

Implement a custom plugin **united-for-the-missing** (folder `united-for-the-missing`) that registers the data structures, REST API endpoints, and UI hooks. The plugin's namespace prefix is `U4M` to avoid conflicts.

3.1 Directory Layout

```
united-for-the-missing/
├─ united-for-the-missing.php    # bootstrap
├─ uninstall.php                 # cleanup (removes options)
├─ readme.txt                    # standard WordPress info
├─ languages/                    # .mo/.po for translations
├─ assets/
│  ├─ css/u4m-frontend.css      # frontend styles
│  ├─ js/u4m-frontend.js        # client JS (search, map, forms)
│  └─ js/u4m-admin.js           # admin JS for blocks
├─ templates/
│  ├─ single-missing_person.php
│  ├─ archive-missing_person.php
│  ├─ page-submit-case.php
│  ├─ page-report-sighting.php
│  └─ parts/                    # reusable markup fragments
├─ includes/
│  ├─ class-u4m-plugin.php      # orchestrator; enqueues assets, registers
hooks
│  ├─ class-u4m-cpt.php         # post types
│  ├─ class-u4m-tax.php         # taxonomies
│  ├─ class-u4m-meta.php        # meta fields & REST registration
│  ├─ class-u4m-rest.php        # REST routes & handlers
│  ├─ class-u4m-admin.php       # custom admin screens (optional)
│  ├─ class-u4m-templates.php   # template loader & page templates
│  ├─ class-u4m-shortcodes.php  # shortcodes for search/map/forms
│  ├─ class-u4m-blocks.php      # Gutenberg blocks (map, case list)
│  ├─ class-u4m-cron.php        # scheduled tasks & IA triggers
│  ├─ class-u4m-security.php    # privacy and PII handling
│  └─ helpers.php               # helper functions (e.g. block renders)
```

3.2 Custom Post Types (CPTs)

1. **missing_person** – Public CPT for missing persons. Registered with custom capabilities (`edit_missing_person` , etc.) and archive slug `/missing` . Supports title, editor, thumbnail,

excerpt, author. Metafields are stored via `register_post_meta` in `class-u4m-meta.php` and exposed via REST.

2. **case_update** – Non-public CPT for internal case updates (events, leads, official statements). Shown in the admin but not publicly accessible except via `get_case` REST call. Used to build timelines.

3. **sighting_report** – Non-public CPT for sighting reports. Created via the public REST endpoint. Contains triage fields and status; accessible in admin for moderation.

4. **resource** – Public CPT for support/legal resources. Archives at `/resources`.

3.3 Taxonomies

- `status` – Active, Located, Deceased, Unknown (non-hierarchical).
- `risk_level` – Low, Medium, High, Critical (non-hierarchical).
- `region` – geographical regions, can be hierarchical (continent → country → state).
- `age_group` – Baby, Child, Teen, Adult, Senior (non-hierarchical).

3.4 Meta Fields

Meta keys follow a prefix convention (`mp_` for missing person, `ai_` for AI generated, `cu_` for case update, `sr_` for sighting report). All meta fields are registered via `register_post_meta()` with `show_in_rest=true` so they appear in JSON responses.

missing_person meta keys:

Key	Type	Purpose
`mp_full_name`	string	Full legal name (required)
`mp_aliases`	array	Known aliases/nicknames
`mp_photo_main`	integer	Attachment ID for main photo
`mp_gallery`	array	Attachment IDs for gallery
`mp_date_missing`	string (YYYY-MM-DD)	Date of disappearance
`mp_time_missing`	string	Time if known
`mp_age_when_missing`	number	Age at disappearance
`mp_current_age_estimate`	number	Estimated current age
`mp_gender`	string	Gender identity
`mp_nationality`	string	Nationality
`mp_last_seen_address`	string	Street or specific address
`mp_last_seen_city`	string	Last known city
`mp_last_seen_region`	string	Region/State

Key	Type	Purpose
`mp_last_seen_country`	string	Country
`mp_last_seen_lat`	number	Latitude (decimal)
`mp_last_seen_lng`	number	Longitude (decimal)
`mp_distinguishing_marks`	string	Description of scars/tattoos
`mp_clothing_last_seen`	string	Description of clothing
`mp_circumstances`	string	Narrative of disappearance
`mp_contact_agency_name`	string	Contact law-enforcement agency
`mp_contact_phone`	string	Contact phone number
`mp_case_number`	string	Official case number
`mp_sources`	array (URL+text)	Source citations
`mp_privacy_consent`	boolean	User consent to display PII
`ai_risk_score`	number (0–100)	AI computed risk score
`ai_summary`	string	AI generated summary
`ai_tags`	array of strings	AI suggested tags
`ai_next_best_actions`	array of strings	Suggested next actions
`ai_last_refresh`	string (datetime)	Last update timestamp

case_update meta keys:

Key	Type	Purpose
`cu_case`	integer	Parent missing_person ID
`cu_type`	string	lead, media, location, statement, official_update
`cu_text`	string	Narrative of the update
`cu_media`	array	Attachment IDs
`cu_event_time`	string (datetime)	Timestamp of update
`cu_lat`, `cu_lng`	number	Coordinates, if relevant
`ai_delta_summary`	string	AI summarization of change

sighting_report meta keys:

Key	Type	Purpose
`sr_case`	integer	Parent missing_person ID
`sr_reporter_contact`	string	Reporter’s contact (optional)

Key	Type	Purpose
`sr_channel`	string	web, phone, email, social
`sr_text`	string	Description of sighting
`sr_time`	string (datetime)	Time of sighting
`sr_lat`, `sr_lng`	number	Location
`sr_evidence_files`	array	Attachments (photos, docs)
`ai_triage_score`	number	AI priority 0–100
`ai_flags`	array	Flags (duplicate, inconsistent, sensitive)
`sr_status`	string	queued, vetted, forwarded, dismissed

3.5 REST API Endpoints

Endpoints are namespaced under `u4m/v1`. Use these from n8n via Basic Auth.

- **GET** `/u4m/v1/cases` – list cases. Parameters: `q` for search query, `pp` for results per page. Returns an array of shaped case objects: `{id,title,excerpt,link,status,risk,last_seen,city,country,thumb}`.
- **GET** `/u4m/v1/cases/{id}` – get full case data. Returns shaped case plus `meta` and `terms` arrays. Publicly accessible.
- **POST** `/u4m/v1/cases` – create new case (draft). Requires authentication. Pass JSON with any of the `mp_*` and `ai_*` fields. Returns the post ID.
- **POST** `/u4m/v1/cases/{id}` – update existing case. Requires authentication. Accepts `mp_*`, `ai_*`, and `post_status`. Use to publish cases after review.
- **POST** `/u4m/v1/sightings` – submit sighting report from the public. Accepts `sr_*` fields. Always creates `pending` `sighting_report`. Public access; no auth.
- **POST** `/u4m/v1/recalc/{id}` – trigger an AI recomputation for a given case ID. Requires authentication. This triggers the `u4m_recalc_ai` action hook; n8n should listen to this via webhook.

3.6 Templates & Shortcodes

- `single-missing_person.php` – defines the case page layout. Includes hero header with name, last seen city/date, and "Report a sighting" button; summary content; map area via `[u4m_map]`; AI panel showing `ai_summary`, `ai_risk_score`, and `ai_last_refresh` with disclaimer; timeline of case updates; how to help section; contact sidebar.

- `archive-missing_person.php` – lists all cases. Uses loop output of `u4m-card` style cards with photo, name, city/date, and risk badge.

- `page-submit-case.php`, `page-report-sighting.php` – wrappers for the shortcodes to output the dynamic React forms. Shortcodes in `class-u4m-shortcodes.php` return `<div>` placeholders; JavaScript populates these.

- **Shortcodes**:

- `[u4m_search]` – renders the search list of cases.
- `[u4m_map]` – renders the map container (JS fetches case locations and draws markers/clusters with Leaflet or Mapbox).
- `[u4m_submit_case]` – dynamic form for authorized users to submit a case via AJAX to `POST /u4m/v1/cases`.
- `[u4m_report_sighting]` – dynamic form for the public to post sighting reports via `POST /u4m/v1/sightings`.
- **Blocks**: optionally provide Gutenberg blocks in `blocks/` folder (e.g. `u4m/map` and `u4m/case-list`) registered in `class-u4m-blocks.php`. Use `helpers.php` to define server-side render functions like `u4m_render_map()`.

3.7 Frontend JavaScript (`assets/js/u4m-frontend.js`)

This script attaches to DOM elements created by shortcodes and loads data from the REST API. Key functions:

```
// Load case list for search page
async function loadCases() {
    const root = document.getElementById('u4m-search');
    if (!root) return;
    const res = await fetch(U4M_CFG.rest + 'cases');
    const data = await res.json();
    root.innerHTML = data.map(c => `
      <article class="u4m-card">
        <img src="${c.thumb || ''}" alt="">
        <h3><a href="${c.link}">${c.title}</a></h3>
        <p>${c.city || ''} · ${c.last_seen || ''}</p>
        <div class="u4m-risk r-${Math.floor((c.risk || 0) / 25)}">${c.risk
|| 0}</div>
      </article>`).join('');
}

// Load map and plot markers (to be completed)
function loadMap() {
    const el = document.getElementById('u4m-map');
    if (!el) return;
    // Initialize Leaflet/Mapbox here and fetch cases to plot coordinates
from mp_last_seen_lat/lng
}

// Handle forms via REST
async function hookForms() {
    const submit = document.getElementById('u4m-submit');
    const report = document.getElementById('u4m-report');
    // Build dynamic forms; send JSON to API endpoints; handle validation &
recaptcha
}
```

```
document.addEventListener('DOMContentLoaded', () => {
    loadCases();
    loadMap();
    hookForms();
});
```

You must integrate Leaflet/Mapbox for interactive maps and clusters, and handle address search with geocoding if needed. All API endpoints require the nonce `wp_create_nonce('wp_rest')` which is exposed via `U4M_CFG.nonce`.

3.8 AI & Cron Hooks

- `class-u4m-cron.php` defines `u4m_hourly` and `u4m_recalc_ai` hooks. On plugin activation, it schedules `u4m_hourly` events every hour using `wp_schedule_event()`.
- When `POST /u4m/v1/recalc/{id}` is called, it triggers the `u4m_recalc_ai` action. The plugin's `queue_recalc()` method should call an external webhook (e.g. an n8n webhook) passing the case ID to compute updated `ai_*` fields.
- `u4m_hourly` can perform maintenance tasks: reorder high-risk cases, clean spam sightings, or send notifications.

3.9 Security & PII Handling

- In `class-u4m-security.php`, filter `pre_update_postmeta` to redact or mask PII when `mp_privacy_consent` is false. Example masks the contact phone number to show only the first 6 digits followed by obscured characters.
- Additional measures such as sanitizing user input, validating file uploads, and logging meta changes are recommended.

3.10 Uninstall Script

`uninstall.php` runs on plugin removal. It should delete plugin options but not remove posts by default. Only remove the custom scheduled hooks and plugin settings.

4. n8n Automation Workflows

The automation layer uses **n8n** to orchestrate geocoding, AI summarization, risk scoring, deduplication, and notification flows. All HTTP nodes should use Basic Auth credentials tied to an Application Password for a WordPress user with `edit_posts` capability. Base URL is `https://your-domain/wp-json/u4m/v1/`.

4.1 Workflow: `u4m-case-ingest`

Trigger: HTTP Webhook (e.g. `/u4m/ingest`) or a manual trigger from the admin interface when a case is submitted.

Steps: 1. **Validate Input** – Ensure required fields (`mp_full_name`, `mp_date_missing`, `mp_last_seen_city`, etc.) are present. 2. **Geocode Address** – Use Nominatim or Google Maps API to convert `mp_last_seen_city/country` to latitude and longitude (`mp_last_seen_lat`, `mp_last_seen_lng`). 3. **AI Summary & Tags** – Call an LLM (OpenAI or similar) with the case narrative to generate an objective summary (`ai_summary`), risk tags (`ai_tags`), and recommendations

(`ai_next_best_actions`). 4. **Risk Score** – Compute `ai_risk_score` via a custom formula (e.g. missing duration, age group, known violence) scaled 0–100. 5. **Create/Update Case** – Send JSON to `POST /u4m/v1/cases` or `/u4m/v1/cases/{id}` with the `mp_*` and `ai_*` fields. If the case already exists, update `post_status` to `publish` after review. 6. **Notify Case Manager** – Send a Slack/Telegram/email message to notify staff that a new case is ready for review.

4.2 Workflow: `u4m-recalc-ai`

Trigger: HTTP Webhook (e.g. `/u4m/recalc`) configured in WordPress to fire when `POST /u4m/v1/recalc/{id}` is called.

Steps: 1. **Get Case Data** – Use `GET /u4m/v1/cases/{id}`. 2. **Recompute** – Regenerate `ai_summary`, `ai_next_best_actions`, and `ai_risk_score` based on updated information and case updates. 3. **Update Case** – `POST /u4m/v1/cases/{id}` with new `ai_*` fields and update `ai_last_refresh` with current timestamp. 4. **Notify** – If the risk score increases significantly (e.g. by ≥15 points), send a critical alert to case managers.

4.3 Workflow: `u4m-sighting-triage`

Trigger: Scheduled (every 5 minutes). Another option is to expose a custom REST route listing new `sighting_report` posts where `sr_status = queued`.

Steps: 1. **Fetch Reports** – Query WordPress for `sighting_report` posts with `sr_status = queued` (create a custom endpoint or use WP_Query via the WordPress API). 2. **Triage AI** – For each report, compute `ai_triage_score` (0–100) and `ai_flags` (e.g. duplicate if text & location resemble an existing report; inconsistent if location contradicts known timeline; sensitive if content mentions violence). Use a LLM or rule-based algorithm. 3. **Geocode** – If `sr_lat/lng` are empty but the text contains a recognizable address, geocode it. 4. **Decide** – If the triage score is ≥70 and flags do not suggest dismissal, set `sr_status = vetted` or `forwarded` and inform the relevant agency; otherwise mark `sr_status = dismissed`. 5. **Create Case Update** – For high-confidence reports, create a `case_update` (with `cu_type = "lead"`, `cu_text` summarizing the sighting, `cu_event_time` equal to `sr_time`, coordinates, and `ai_delta_summary` summarizing the difference). Then call the `u4m-recalc-ai` webhook for that case.

4.4 Workflow: `u4m-news-monitor`

Trigger: Scheduled (every 6–12 hours).

Steps: 1. **Loop Active Cases** – Fetch all missing_person posts with `status = active`. 2. **Search External Sources** – Use a news API or Google Custom Search to look for articles mentioning the person's name and last known location. 3. **Named Entity Recognition (NER)** – Extract dates and locations from new articles. Compare to existing `case_update` timeline; if new, create a `case_update` with `cu_type = "media"` or `"official_update"` depending on the source. 4. **Recalculate** – Trigger `POST /u4m/v1/recalc/{id}` to update risk score and summary.

4.5 Workflow: `u4m-deduplication`

Trigger: Daily scheduled job.

Steps: 1. **Fetch All Cases** – Query all `missing_person` posts. 2. **Check Duplicates** – Use a similarity metric combining name, date_missing, and city to detect potential duplicates (e.g. Levenshtein distance < 3 and same date/location). Use text embeddings to compare `mp_circumstances`. 3. **Mark Duplicates** – For duplicates, flag them or merge into a single record after manual verification. Post a message to the Case Manager for review. Do not automatically delete cases.

4.6 Additional Considerations

- Use environment variables for API keys (geocoding, OpenAI, Slack, etc.)
- Implement exponential backoff for external API calls.
- Log n8n workflow runs and store error details for debugging.
- Use n8n's built-in credentials store for Basic Auth to WordPress.

5. AI Functionality Phases

1. **V1 – Baseline**
2. Geocoding addresses.
3. Summarizing cases and extracting tags.
4. Calculating risk scores from objective signals (e.g. age, missing duration, region risk profile).
5. Triage sighting reports (duplicate detection, spam filtering).
6. Flagging inconsistent dates/locations.
7. **V2 – Enhanced**
8. Semantic similarity search to cluster related cases.
9. Map clustering and heatmap for likely areas based on timeline and reports.
10. Recommend next actions (e.g. contacting local businesses, reviewing CCTV footage).
11. Alert when relevant news articles appear.
12. Age progression imaging (external service with explicit consent).
13. **V3 – Multimodal & Investigative**
14. Similarity matching across text and image embeddings.
15. Generate hypotheses with supporting evidence while documenting the chain of reasoning.
16. Provide investigators with an interactive panel showing AI reasoning with transparency.

6. Coding Guidelines & Quality

- Use WordPress coding standards (camelCase for variables, snake_case for functions; prefix functions with `u4m_` to avoid conflicts).
- Keep logic modular within classes; limit global functions to helpers.
- Escape all output (`esc_html`, `esc_url`) and validate/sanitize user input.
- Include translation functions (`__()`, `_e()`) using the text domain `u4m`.
- Write docblocks for all functions and classes.
- Use version control (Git) and include `readme.txt` with plugin headers and changelog.

7. Summary for the Developer

- **Implement the WordPress plugin** per the detailed structure above. Register CPTs, taxonomies, meta fields, REST routes, shortcodes, blocks, cron hooks, and security filters. Style frontend output via CSS and create dynamic UI with JS.
- **Deploy the site** on WordPress 6.5+ with HTTPS. Create necessary pages and assign shortcodes. Set up roles, menus, privacy policies, and security measures. Test REST API endpoints.
- **Set up n8n** workflows for case ingestion, AI recompute, sighting triage, news monitoring, and deduplication. Use Basic Auth to communicate with WordPress. Integrate geocoding and AI services. Log and monitor workflow outcomes.
- **Iterate on AI functions** across three phases (V1–V3) using consistent meta keys. Provide human-readable explanations for AI outputs and ensure they are clearly marked as informational, not conclusions.
- **Respect privacy and consent** in all aspects: mask PII when consent is absent; publish only verified information; provide mechanisms to correct or remove data per user rights.
- **Document your code** thoroughly and keep naming consistent (`u4m_` prefix). Provide comments and README files for maintainability. Ensure compatibility with future WordPress updates.

With this brief, the engineering team should be able to build the complete system—frontend, plugin, and automations—efficiently and professionally.
