EcoServants® Open Grant Intelligence – Hybrid Architecture Build Plan (Intern Edition)

1) Purpose & Outcomes

Goal: Deliver a lawful, transparent, search-engine-style system that aggregates **public** grant metadata and powers the EcoServants® Grant Tracker UI.

Outcomes by end of Phase 3: - Public crawler (respecting robots.txt) indexing allow-listed domains - Normalized grants index in SiteGround MySQL - Public REST endpoint feeding WordPress Grant Tracker - Optional, opt-in Community Compute nodes contributing safe, throttled crawls

---

2) Reference Architecture (Hybrid)

```
[Allow-listed Public Sources]
   |  IRS 990/BMF  |  Grants.gov API  |  Foundation RSS/Sitemaps
|  .org/.edu/.gov pages
   v
[Connectors] → [Crawler Queue & Fetcher] → [Parser/Extractor]
                                |
                                |                     → [Normalizer → Deduper
→ Provenance]
                                v
                        [Indexer & API]
                                |
                         (MySQL on SiteGround)
                                |
          ┌─────────────────────┴─────────────────────┐
          v                                           v
 [WP Grant Tracker UI]                        [Public /wp-json/esgt/v1/
grants]
 (SiteGround)                                        (SiteGround)

Optional: Community Compute Nodes (opt-in) → contribute fetch jobs from
allow-list, with strict rate limits & logs.
```

---

3) Phased Build Plan & Milestones

Phase 0 – Foundations (Week 1) - Define **Allow-List Policy** (domains, robots adherence, crawl budget) - Draft **Ethical Crawling Guidelines** & **Community Compute Consent** - Publish **User-Agent** & `/about-bot` page - Create repo scaffolding (crawler, normalizer, API, WP plugin integration)

Milestone: Docs approved; repos created; CI lints green.

Phase 1 – Data Ingest (Weeks 2–3) - Implement **Grants.gov** & **IRS 990/BMF** connectors (API/bulk files only) - Normalize to `grants` schema; write deduper (title+funder+URL fuzzy) - Seed **MySQL schema** on SiteGround (read-write from crawler VM)

Milestone: First 1k normalized records in DB with provenance.

Phase 2 – Crawler & Policies (Weeks 3–4) - Build queue + fetcher (respect robots.txt; per-domain throttle; caching) - Parsers for **RSS/Sitemaps** then fallback **HTML extraction** - Provenance: store `source_url`, `retrieved_at`, `content_hash`, `robots_allowed`

Milestone: 10–20 allow-listed foundations parsed; zero robots violations.

Phase 3 – API & WordPress Integration (Weeks 4–5) - REST API: `/wp-json/esgt/v1/grants?status=open&state=CA&limit=50` - WP: Shortcode/blocks to list grants; cron to refresh local cache; admin tools - UI: Filters (status, funder, region), quick search, export CSV

Milestone: Live dashboard on SiteGround showing indexed opportunities.

Phase 4 – Community Compute (Weeks 6–7) - Opt-in desktop helper or CLI (signed releases) – allow-list only - Visible activity log, hard rate caps (e.g., 1 req/sec, 1000/day) - Automatic self-pause on high CPU/network; kill-switch toggle

Milestone: ≥5 volunteer nodes contributing safely with audit logs.

---

4) Team Roles & Weekly Assignments

Data Analysis & Research - Curate allow-list; verify robots.txt; annotate source fields → schema map - Write test cases for dedupe & coverage; QA data correctness

Engineering (Crawler/API) - Implement connectors, fetcher, parsers, normalizer, deduper - Ship REST endpoints; WP integration; cache and pagination

Ops & Security - Provision crawler VM; configure firewall; rotate credentials - Monitoring & alerting (uptime, queue backlog, 4xx/5xx rates)

Policy & Ethics - Maintain guidelines, consent wording, and compliance checklists - Triage takedown requests; adjust allow-list promptly

Comms/Docs - Write `/about-bot`; README; CONTRIBUTING; user tutorials - Publish weekly changelog & data coverage report

---

5) Technical Specs

5.1 Grants Schema (MySQL)

`grants` table (minimum fields): - `id` PK - `title` TEXT - `funder` VARCHAR(255) - `program` VARCHAR(255) NULL - `source_url` TEXT - `apply_url` TEXT NULL - `deadline_date` DATE NULL - `eligibility` TEXT NULL - `amount_min` DECIMAL(12,2) NULL - `amount_max` DECIMAL(12,2) NULL - `country` / `state` / `region` VARCHAR(64) NULL - `tags` JSON NULL - `source_name` VARCHAR(128) - `retrieved_at` DATETIME - `content_hash` CHAR(32) - `robots_allowed` TINYINT(1) DEFAULT 1 - `confidence` TINYINT UNSIGNED DEFAULT 80 - INDEXES: (`deadline_date`), (`funder`), (`content_hash`), FULLTEXT (`title`, `eligibility`)

5.2 REST Endpoints

- `GET /wp-json/esgt/v1/grants` – list with filters (`q`, `state`, `country`, `deadline_from`, `deadline_to`, `funder`, `tag`)
- `GET /wp-json/esgt/v1/grants/{id}` – detail
- `GET /wp-json/esgt/v1/stats` – totals, new today, coverage by domain

5.3 WordPress Integration

- Shortcodes/Blocks: `[esgt_grants filters="status,region,funder"]`
- Admin: re-index button, CSV export, mapping editor
- Roles: `grant_writer` (Editor-cap), `grant_viewer` (read-only)

---

6) Crawler Rules & Rate Limits

- **User-Agent:** `EcoServantsBot/1.0 (+https://ecoservantsproject.org/about-bot)`
- **Robots:** obey `Disallow`, `Crawl-delay`, `nofollow/noindex` metas
- **Throttle:** default 1 req/sec; per-domain budgets; exponential backoff on 429/503
- **Respectful Fetch:** HEAD when possible; cache ETags/Last-Modified; avoid heavy assets
- **No Auth:** never fetch behind logins, CAPTCHAs, or paywalls

---

7) Community Compute (Opt-In) – Requirements

- Explicit consent (+ easy disable switch)

- Allow-list enforced client-side & server-side
- Hard caps (CPU < 15%, 1 req/sec, max 1000/day)
- Zero cookies/auth; no PII collection; plaintext activity log
- Signed binaries (checksum) and auto-update with release notes

Consent snippet (UI):

> By enabling Community Compute, I agree to donate limited network requests to fetch **public** pages from the EcoServants allow-list. No personal data, cookies, or private pages are accessed. I can disable this at any time.

---

8) Monitoring & QA

- Metrics: pages fetched, unique sources, robots violations (target 0), 4xx/5xx rates, dedupe ratio
- Dashboards: Grafana/Netdata on crawler VM; weekly CSV report for interns
- Data QA: random 1% sample manual review; schema conformity tests

---

9) Risks & Mitigations

- **Robots or ToS conflicts:** maintain allow-list; immediate cease-crawl switch
- **Host load complaints:** strict throttle + contact page; remove on request
- **Data quality drift:** continuous dedupe tuning; provenance auditing
- **Abuse/IP blocks:** rotate IPs within policy; backoff & notify

---

10) Timeline (Aggressive, 7 Weeks)

- W1: Policies, repo, schema, SiteGround DB
- W2–3: Grants.gov + IRS connectors; dedupe; initial load
- W4: Crawler core; parsers for RSS/Sitemaps; first foundation domains
- W5: REST API + WP UI integration
- W6: Community Compute beta; telemetry; kill-switch
- W7: QA hardening; docs; public launch

---

11) Deliverables (Definition of Done)

- Public `/about-bot` & Ethical Guidelines
- MySQL `grants` table with ≥10k high-confidence records
- Live REST API + WordPress dashboard (filters, CSV export)
- Community Compute client (opt-in) with logs & caps
- Weekly coverage & quality report shared with interns

---

12) Intern Task Backlog (first 3 sprints)

Sprint 1 - Draft allow-list v1 (50 domains); verify robots; document contact pages - Map Grants.gov fields → schema; write normalization tests - Create `/about-bot` and user-agent docs

Sprint 2 - Implement IRS/Grants.gov connectors; seed DB - Build deduper (Levenshtein/Jaro-Winkler) + unit tests - WP shortcode prototype for listing grants

Sprint 3 - Crawler queue + RSS/Sitemap parser; caching & ETag support - REST API endpoints with filters + pagination - Community Compute UI mock + consent flow draft

---

13) Appendix – Sample SQL (MySQL)

```
CREATE TABLE grants (
  id BIGINT PRIMARY KEY AUTO_INCREMENT,
  title TEXT NOT NULL,
  funder VARCHAR(255),
  program VARCHAR(255),
  source_url TEXT NOT NULL,
  apply_url TEXT,
  deadline_date DATE,
  eligibility TEXT,
  amount_min DECIMAL(12,2),
  amount_max DECIMAL(12,2),
  country VARCHAR(64),
  state VARCHAR(64),
  region VARCHAR(64),
  tags JSON,
  source_name VARCHAR(128),
  retrieved_at DATETIME NOT NULL,
  content_hash CHAR(32) NOT NULL,
  robots_allowed TINYINT(1) DEFAULT 1,
  confidence TINYINT UNSIGNED DEFAULT 80,
  FULLTEXT KEY ft_title_elig (title, eligibility),
  KEY idx_deadline (deadline_date),
  KEY idx_funder (funder),
  KEY idx_hash (content_hash)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

EcoServants® — Learn. Act. Restore.