Architectural Blueprint for a Scalable, Culturally-Aware Genealogy Platform

Philip Leo Tambiti Walekhwa

2025-06-24

Project Vision

The WalsoftAI-Genealogy platform is a modular, narrative-first genealogy web application built on Django. Rooted in Luhya (Bakhabi) cultural structures, it aims to preserve, narrate, and visualize intergenerational family data using AI-generated stories, while honoring polygamy, remarriage, subclan logic, and dual parental lineage. It is designed to scale from individual families to whole ethnic groups, running entirely on self-hosted, privacy-first infrastructure.

Modular Architecture

App Structure

- `users/` – Role-based identity and dashboard logic (Visitor, Elder, FamilyAdmin, Researcher)
- `people/` – Core person data (names, clan, gender, birth/death, photos)
- `relationships/` – Parent-child and marriage tracking, including polygamy
- `events/` – Life events such as migration, education, initiation, etc.
- `narratives/` – AI-generated stories and narrative Q&A
- `documents/` – Uploads, photos, oral audio, and scanned records
- `admin_tools/` – Merge suggestions, versioning, audit logs, conflict resolution

Services & Helpers

- Reusable services in `services/` for dashboards, stories, data traversal
- Helper functions in `helpers/` for filters, lineage graphs, polygamy logic
- Views delegate logic, ensuring testability and thin controller pattern

Role-Based Access Control (RBAC)

Recommended Roles

1. **Visitor** – Browse public data and narratives
2. **Registered User** – Submit and edit known family data
3. **Family Admin** – Validate entries, moderate relationships
4. **Elder** – Contribute oral history, limited editing rights
5. **Narrative Editor** – Translate data into readable, structured stories
6. **Researcher** – Access anonymized data for academic analysis
7. **Moderator** – Handle flagged entries and conflict resolution
8. **System Admin** – Full access to backend, logs, settings

`users/` App Responsibilities

- CustomUser + OneToOne models (Elder, FamilyAdmin, Researcher, etc.)
- Role-specific views and dashboards
- Dashboard redirection logic based on role
- Role decorators (`@role_required(...)`, `@is_ancestor_owner()`)

User Interface Design

Styling & Layout

- Tailwind CSS for clean, responsive styling
- Role-based themes (color, icons)
- Sidebar templates stored by role in `templates/sidebars/`

Printables

- PDF story and tree exports using `xhtml2pdf`
- Jinja2-based narrative rendering templates
- School-style layout, A4-friendly

AI & Narrative Generation

- Local LLMs via `llama-cpp-python`
- Model: Phi-3 Mini or Mistral 7B (quantized)
- Templates in `narratives/prompts/*.jinja`
- Outputs tied to Person.story and editable post-generation
- Graph traversal for questions like "Am I related to…"

Data Model Blueprint

- `Person`: Full name, aliases, gender, clan, bio, photo, parent links
- `Marriage`: Partners (ManyToMany), start/end metadata, reason
- `Event`: Linked to person; types include migration, education
- `Alias`: Optional model or JSONField on Person for name variants

Logic Rules

- Dual lineage: `father` and `mother` are both FKs to Person
- Siblings inferred by shared parent(s)
- Marriage is not a gatekeeper for parenthood
- Deduplication via fuzzywuzzy and admin merge tool
- Versioning via django-simple-history

Filtering and Search

- Custom filters by subclan, name root, generation, gender
- Match by birthplace proximity or fuzzy name scoring
- Lineage filters: `filter_descendants_by_mother()` or `ancestors_of()`

Security & Compliance

- Login + role-based dashboard redirection
- Strong password policy and CAPTCHA (optional)
- .env for secrets; separate staging and prod configs
- All AI and user data run locally; no external APIs
- Django signals for audit logging

Scalability & Extensibility

- PostgreSQL with JSONB for flexible storage
- Docker for deployment and consistency
- Celery + Redis for background jobs
- REST API or GraphQL for future mobile sync
- Multi-family namespace support (Phase 5)

Continuous Improvement & Feedback

- Editable AI stories for cultural tuning
- Merge dashboard for duplicates
- Periodic integrity scans (e.g., orphaned entries, conflict links)
- UX tested with elders and youth for clarity
- Transparent versioning history for every edit

Summary

This blueprint, derived from BIVGS best practices and refined through culturally rooted design, ensures:

- Modular, DRY, testable Django architecture
- Respect for African family structure logic
- Local-first AI storytelling and query answering
- Secure and role-aware access to family data
- Path to scalable, multi-family and cross-clan growth

WalsoftAI-Genealogy is not just a database. It's a memory keeper, a story engine, and a tool for intergenerational connection.