WalsoftAI-Genealogy: Ancestral Narrative Web App – Technical & Cultural Blueprint

Walekhwa Tambiti leo Philip

WalsoftAI-Genealogy: Ancestral Narrative Web App – Technical & Cultural Blueprint

1. Project Overview & Vision

Project Name

WalsoftAI-Genealogy
A culturally-rooted, AI-enhanced family history platform for the Luhya community and beyond.

Vision Statement

The WalsoftAI-Genealogy platform is a scalable web application designed to preserve, narrate, and explore African family lineage — starting with the Luhya (Bakhabi) people of Busia, Kenya. It goes beyond visual trees by using AI to generate **natural language stories** from family data, helping current and future generations reconnect with their ancestral roots.

We aim to solve a pressing problem: as elders pass away, intergenerational knowledge — names, clans, family ties, and origin stories — is disappearing. This app ensures those stories are captured, structured, and beautifully retold.

Purpose

- Preserve African genealogy and oral history
- Provide younger generations with understandable family narratives
- Respect African marriage structures (polygamy, remarriage)
- Avoid assumptions based on Western lineage models
- Offer an AI-powered natural storytelling interface (free, local models)
- Scale from one family to an entire ethnic group's genealogical system

---

Core Features (MVP Phase)

- Add/edit individuals (name, gender, clan, birth/death, location, biography)
- Track both **maternal and paternal** lineage
- Support **multiple marriages** (polygamy, divorce, remarriage)
- Visualize family tree structure
- Generate flowing **ancestral stories** using offline AI
- Prevent duplicate entries through smart matching
- Secure, permission-controlled editing

What Makes It Different

- AI-generated **narratives**, not just data
- Rooted in **Luhya cultural structures** (e.g., subclans, totems, polygamy)
- Designed for **offline/local AI**, no reliance on paid APIs
- Built for **sustainability and collaboration** across generations

---

2. Architecture & System Modules

System Architecture Overview

The WalsoftAI-Genealogy platform is a modular Django-based web application with layered responsibilities:

```
[Client (Browser/UI)]
↓
[Frontend Layer: HTMX + Tailwind or React]
↓
[Django Views & API Layer (DRF)]
↓
[Django Models → PostgreSQL Database]
↓
[AI Narrative Engine (local LLM wrapper)]
```

Core Modules

1. Person Module

Captures all details about individuals, including: - Names (first, middle, last) - Gender - Clan/Subclan - Birth/death info - Parental relationships (father, mother) - Place of origin - Free-form biography - Profile photo

2. Relationship Module

Tracks: - Parent-child relationships (dual lineage) - Marriages (monogamous, polygamous, remarried) - Sibling inference from shared parent(s)

3. Marriage Module

Supports: - Multiple spouses - Date of marriage and optional end - Cause of separation (death, divorce)

4. Story Engine Module

Uses local AI to: - Summarize family data into readable prose - Generate biography-like outputs - Answer narrative queries like "Who is Philip?" or "Am I related to Joseph Wafula?"

5. Search & Deduplication Module

Prevents duplication by: - Fuzzy name matching - Birthplace and date cross-check - Aliases and nicknames - Admin-reviewed merge suggestions

6. User & Role Module

Controls: - Who can add/edit/delete people - Who can confirm relationships - Admins vs trusted family contributors - Logs of all edits for transparency

7. Deployment Module

Handles: - Virtual environments - Gunicorn + systemd services - NGINX configuration - Static/media handling - Secure domain & HTTPS setup

8. AI Serving Module

Supports: - LLM execution (local) - Story prompt templating - Model switching (e.g., Phi-3, GPT4All, Mistral) - Optional: RAG or graph-based question answering

---

Future Modules (Optional Later Phases)

- Ancestral map visualizer using Leaflet or Mapbox
- Audio story upload + Whisper transcription
- Clan and totem tree visualizer
- Automated relationship graph traversal (e.g., "How are we related?")

---

3. Tech Stack & Tools

The WalsoftAI-Genealogy platform is built entirely with **free and open-source tools**, designed for sustainability, offline capability, and cultural sensitivity.

Tool	Purpose

Backend

Tool	Purpose
Python 3.11+	Core programming language
Django 5.x	Web framework for rapid development
Django REST Framework	For APIs (to mobile apps or React frontend)
PostgreSQL	Primary database (supports advanced querying, JSONB fields)
Gunicorn	WSGI server for running Django in production
NGINX	Reverse proxy + static/media file serving
Certbot (Let's Encrypt)	Free SSL certificate management
systemd	Service manager for Gunicorn process

Frontend

Tool	Purpose
HTMX	Server-driven interactivity (fast MVP)
Tailwind CSS	Clean, modern styling with minimal effort
Alpine.js (optional)	Lightweight reactivity for HTMX
React (optional future)	For full SPA experience or advanced interactivity

---

AI/Narrative Engine

Tool	Purpose
llama-cpp-python	Run free, local LLMs (e.g., Mistral, Phi-3, GPT4All)
transformers (HuggingFace)	Load or run pre-trained open-source LLMs
LangChain (optional)	For chaining logic in LLM storytelling
Jinja2	For story prompt templating
fuzzywuzzy / rapidfuzz	For deduplication and smart name matching

---

DevOps & Deployment

Tool	Purpose
OpenSSH	Secure remote server access
tmux or screen	Manage long-running server/AI processes
Supervisor (optional)	Monitor AI or worker services
UFW	Simple firewall for server hardening
cron	Automated backups, cleanup, cert renewals

Optional Libraries (Future Integration)

Tool	Use Case
networkx	Graph-based relationship traversal (e.g., find common ancestors)
Leaflet.js / Mapbox	Visualize migration origins (e.g., "from Bugiri, Uganda")
Whisper.cpp	Local speech-to-text for elder audio recordings
spaCy / NLTK	Text parsing or enrichment of stories
GraphQL	Alternative API layer for efficient frontend interaction

---

Local AI Model Suggestions

Model	Notes
Phi-3-mini (ONNX or GGUF)	Lightweight, small memory footprint (~1GB)
Mistral 7B (quantized)	Good quality, ~4GB in GGUF format
GPT4All-J	Versatile, community-supported
LLaMA 2 or 3 (quantized)	Advanced reasoning, needs more resources

All models must be downloaded and run using `llama-cpp-python` with no API keys required.

---

4. Data Modeling & Relationship Logic

At the heart of WalsoftAI-Genealogy is a flexible, culturally-aware data structure that can capture: - Maternal and paternal ancestry - Polygamous and sequential marriages - Migration and origin history - Free-form narrative content

---

Core Models

1. `Person`

Represents each individual in the genealogy database.

```
class Person(models.Model):
    full_name = models.CharField(max_length=255)
    first_name = models.CharField(max_length=100, blank=True)
    middle_name = models.CharField(max_length=100, blank=True)
    surname = models.CharField(max_length=100, blank=True)
    gender = models.CharField(max_length=10, choices=[('male', 'Male'), ('female', 'Female')]
    date_of_birth = models.DateField(null=True, blank=True)
    date_of_death = models.DateField(null=True, blank=True)
    place_of_origin = models.CharField(max_length=255, blank=True)
    clan = models.CharField(max_length=100, blank=True)
    subclan = models.CharField(max_length=100, blank=True)
    father = models.ForeignKey('self', null=True, blank=True, on_delete=models.SET_NULL, rela
    mother = models.ForeignKey('self', null=True, blank=True, on_delete=models.SET_NULL, rela
    biography = models.TextField(blank=True)
    photo = models.ImageField(upload_to='photos/', null=True, blank=True)
    created_at = models.DateTimeField(auto_now_add=True)
```

2. `Marriage`

Captures relationships including polygamy and remarriage.

```
class Marriage(models.Model):
    partners = models.ManyToManyField(Person, related_name='marriages')
    date_of_marriage = models.DateField(null=True, blank=True)
    end_date = models.DateField(null=True, blank=True)
    reason_for_end = models.CharField(max_length=255, blank=True)
```

3. `Event`

Captures additional facts like education, migration, or life events.

```
class Event(models.Model):
    person = models.ForeignKey(Person, on_delete=models.CASCADE)
    event_type = models.CharField(max_length=100)  # e.g., 'education', 'migration'
    description = models.TextField()
    date = models.DateField(null=True, blank=True)
```

Relationship Logic

- **Dual lineage** is supported via `father` and `mother` foreign keys.
- **Siblings** are inferred from shared parent(s).
- **Marriages** can involve more than two individuals (polygamy).
- Children are linked only to *individuals*, not to the `Marriage` object, to avoid rigid structures.

Deduplication and Validation

To avoid duplicate entries (e.g., "John Okiya" vs "John Meshak Okiya"):

- Use `fuzzywuzzy` or `rapidfuzz` to compute similarity
- Warn users during creation: "Possible duplicate: John Meshak Okiya (b. 1950, from Busia)"
- Allow aliases and nicknames

Versioning (Audit Trail)

- Use `django-simple-history` or `django-reversion`
- Every edit to a `Person` or `Marriage` is tracked
- Admins can restore or compare changes

---

5. AI Integration & Narrative Generation

Purpose

The WalsoftAI-Genealogy platform uses local, free AI models to transform structured family data into **natural, flowing stories**. These narratives go beyond technical lineage charts — they bring ancestors to life in culturally grounded prose that's understandable and memorable.

Key Use Cases

1. **Generate Person-Based Narratives**

Example:
"John Meshak Okiya was born to Peter and Alice in Busia. He was the third child in a family of 12. His father also had a second wife, Nancy…"

2. **Summarize Family Lineage**

Example:
"Peter married Alice and Nancy. Alice bore five children and Nancy had seven. Together, they formed the extended household of Peter, whose own father, Dennis, was one of four siblings…"

3. **Answer Genealogical Questions**

Examples:
- "Who is Philip?"
- "Am I related to Joseph Wafula?"
- "Is Anazio Walekhwa part of my family line?"

AI Tools and Technologies

Model Runner: `llama-cpp-python`

- Runs local large language models (LLMs) directly on your server
- No internet or API required
- Supports `.gguf` quantized models for high performance
- Lightweight and production-safe

Installation:

```
pip install llama-cpp-python
```

Example Integration:

```
from llama_cpp import Llama

llm = Llama(model_path="/models/phi3.gguf", n_ctx=2048)
prompt = render_template("person_story_prompt.jinja", context_data)
output = llm(prompt)["choices"][0]["text"]
```

Primary Model: Phi-3 Mini

Feature	Details
Model Name	`Phi-3 Mini (gguf)`
Size	~1.8GB
Speed	Very fast on low-resource CPUs
Quality	Excellent for short narratives
Hosting	Self-hosted, local-only

Store models in: `/home/philip/models/phi3-mini.gguf`

Prompt Templating

Narrative prompts are built using **Jinja2**, drawing from the `Person`, `Marriage`, and `Event` models.

Example Prompt Template:

```
{{ person.full_name }} was born to {{ person.father.full_name }} and {{ person.mother.full_na
He was the {{ person.birth_order }} child in a family of {{ person.siblings_count }}.
{{ person.father.full_name }} had {{ father_spouse_count }} spouses including {{ father_othe
{{ person.full_name }}'s grandfather, {{ paternal_grandfather }}, came from {{ paternal_orig
```

Story Handling

- **On-demand story generation**: triggered via button or query
- **Stored summaries**: saved into a `Person.story` field after generation
- **Editable**: users can refine the AI-generated narrative if needed

---

Query Handling

All genealogical queries (e.g., "Am I related to X?") are processed in two stages:

1. **Data Traversal**: use relationship logic (parent → grandparent, etc.) to find connection path
2. **Narrative Generation**: AI turns the results into readable explanations

---

Narrative Quality Checks

- All prompts include fallback handling (e.g., unknown parents)
- Generated text is post-processed to fix common AI grammar issues
- Stories are tied to structured data — updates will refresh stories when requested

---

Model Storage & Runtime

- Store models in `/home/philip/models/`
- Load models at boot or on-demand using `supervisor` or `systemd`
- LLM runs on localhost and exposes a function call or minimal API

Security and Privacy

- Stories are never sent to external APIs
- All AI runs locally to preserve family privacy
- Role-based access controls prevent unauthorized generation/editing

---

6. Deployment & Server Setup

This platform is designed to run on a secure, self-managed Ubuntu server using your own hosting infrastructure — no cloud lock-in, no external dependencies.

Server Requirements

Component	Recommended
OS	Ubuntu 22.04 or 24.04 LTS
RAM	Minimum 4 GB (8 GB preferred for AI models)
CPU	2+ vCPUs (4 for smoother AI model runtime)
Disk	20 GB minimum (plus 5–10 GB for models/media)
Access	Full SSH access with sudo privileges
Public IP	e.g. `193.71.134.212` (confirmed)

Directory Structure

```
/home/philip/
    apps/
        genealogy/              # Django project
        models/                 # AI models (gguf)
        logs/                   # Gunicorn, Nginx logs
        static/                 # Static files
        media/                  # Uploaded files
```

Core Software Stack

Install with:

```
sudo apt update && sudo apt install -y \
python3 python3-venv python3-pip \
nginx ufw curl git unzip wget \
postgresql postgresql-contrib \
certbot python3-certbot-nginx \
tmux
```

Python & Django Setup

```
cd ~/apps
python3 -m venv genealogy_venv
source genealogy_venv/bin/activate

pip install --upgrade pip
pip install django djangorestframework psycopg2-binary \
llama-cpp-python jinja2 fuzzywuzzy django-simple-history
```

Gunicorn Service

Create service:

```
sudo nano /etc/systemd/system/genealogy_gunicorn.service
```

Paste:

```
[Unit]
Description=Gunicorn daemon for Genealogy App
After=network.target

[Service]
User=philip
Group=www-data
WorkingDirectory=/home/philip/apps/genealogy
ExecStart=/home/philip/apps/genealogy_venv/bin/gunicorn \
    --access-logfile - \
    --workers 3 \
    --bind unix:/run/genealogy.sock \
    genealogy.wsgi:application

[Install]
WantedBy=multi-user.target
```

Then:

```
sudo systemctl start genealogy_gunicorn
sudo systemctl enable genealogy_gunicorn
```

NGINX Configuration

Create config:

```
sudo nano /etc/nginx/sites-available/genealogy
```

Paste:

```
server {
    listen 80;
    server_name genealogy.walsoftai.com;

    location /static/ {
        alias /home/philip/apps/genealogy/static/;
    }

    location /media/ {
        alias /home/philip/apps/genealogy/media/;
    }

    location / {
        proxy_pass http://unix:/run/genealogy.sock;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    access_log /home/philip/apps/logs/genealogy_access.log;
    error_log /home/philip/apps/logs/genealogy_error.log;
}
```

Enable & restart:

```
sudo ln -s /etc/nginx/sites-available/genealogy /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl reload nginx
```

SSL via Let's Encrypt

```
sudo certbot --nginx -d genealogy.walsoftai.com
sudo certbot renew --dry-run
```

AI Model Setup

Download your AI model (e.g., Phi-3 Mini) from Hugging Face:

```
mkdir ~/apps/models
cd ~/apps/models
wget https://huggingface.co/microsoft/Phi-3-mini/resolve/main/phi-3-mini.gguf
```

Useful Tools

Tool	Purpose
`tmux`	Run persistent LLM processes
`supervisor`	(Optional) Manage LLM or custom scripts
`namei -l path`	Debug NGINX/static/media permissions
`tail -n 50 logfile`	Monitor logs
`ufw allow`	Open firewall ports

Daily Maintenance Tips

- Backup PostgreSQL:

  ```
  pg_dump genealogy_db > backup_$(date +%F).sql
  ```

- Check static/media permissions:

  ```
  sudo chmod -R o+rX /home/philip/apps/genealogy/static
  sudo chmod -R o+rX /home/philip/apps/genealogy/media
  ```

- Rotate logs and clear AI cache periodically

---

7. Data Integrity, Roles & Checks and Balances

In a collaborative, culturally sensitive genealogy platform, **data correctness, trust, and accountability** are critical. This section defines how WalsoftAI-Genealogy maintains clean, accurate, and respectful family data.

Key Risks We Must Prevent

Risk	Description
Duplicate entries	Same person added with variations (e.g., "John Okiya" vs "Meshak John Okiya")
Inconsistent lineage	Conflicting parent-child relationships
Unauthorized editing	Unverified users modifying sensitive ancestral records
Data loss or corruption	System errors or accidental deletions
Culturally inappropriate assumptions	Imposing Western rules (e.g., monogamy-only structures)

Strategies for Accuracy and Safety

1. Smart Duplicate Detection

- Fuzzy matching using `fuzzywuzzy` on:
  - Full name variants
  - Nicknames / aliases
  - Place of origin + DOB range
- Prompts user during entry:
  "This person may already exist: Meshak Okiya, born in Busia. Do you want to review this record instead?"

2. Aliases & Spellings Support

- Store alternative names in a `Person.aliases` JSONField
- Capture tribal or religious name variations

3. Structured Relationships

- Each person has explicit `mother` and `father`
- Children are inferred from reverse links
- Siblings are calculated dynamically (same parent(s))

4. Polygamy & Sequential Marriage Support

- Multiple spouses linked via `Marriage` model
- Each `Marriage` has metadata (start/end date, reason)
- Children are not bound to a specific `Marriage`, but to parents directly

5. Admin Merge Tool

- Admin dashboard suggests likely duplicates
- Option to "Merge A into B", showing:
  - Field conflicts
  - Relationship overlaps
  - Confirm or override decision

6. Edit History & Versioning

- Use `django-simple-history` to track all changes to:
  - Person
  - Marriage
  - Event
- Changes are logged with timestamp and user info
- Admins can view or restore previous versions

Roles & Permissions

Role	Capabilities
Admin	Full control: add/edit/delete/merge, approve users, generate stories
Contributor	Add/edit family members, submit corrections, cannot delete or merge

Role	Capabilities
Viewer	Read-only access to trees and narratives
Research Mode (future)	External historians can browse de-identified data for cultural studies

All roles are assigned via Django Admin or a custom user management panel.

Security Controls

- Strong password policy
- CAPTCHA for public forms (if exposed)
- Two-step verification (future enhancement)
- Role-based view restrictions
- Auto-log out after idle period

Integrity Checks (Periodic)

Check	Frequency	Method
Duplicate name/person	Weekly	Fuzzy match scan
Orphaned records (no parent)	Monthly	Query + admin prompt
Broken relationships	Daily	Validation rules on save
Stories missing or outdated	Monthly	Regenerate using updated templates

---

Backups & Recovery

- PostgreSQL backups run via `cron` daily
- Static/media folder zipped weekly
- AI model files backed up manually on change
- Backups stored in: `/home/philip/apps/backup/`
- Recovery process is documented in `README.md`

8. Roadmap & Development Phases

The WalsoftAI-Genealogy platform is intentionally structured for long-term growth. Below is the development roadmap, broken into clear, progressive phases.

Phase 1: Core MVP (Family Tree + Editor)

Goals: - Basic person data entry (name, gender, clan, photo) - Add/edit/delete individuals - Track parents, spouses, children - Dual-lineage support (mother and father) - Visual tree view using HTMX or React - Admin-protected backend - Fuzzy matching to prevent duplicate entries

Outputs: - Family tree builder interface - Data model with integrity controls - Basic backup and log structure

---

Phase 2: AI Narrative Engine

Goals: - Generate flowing, readable biographies using local AI - Model selection: **Phi-3 Mini** - Prompt templating via Jinja2 - Store generated story in DB (`Person.story`) - Edit/refresh story manually - CLI or API-based LLM wrapper

Outputs: - Story-generation engine - Prompt template system - Story editor in UI

Phase 3: Ancestral Q&A System

Goals: - Handle natural questions like: - "Who is Philip?" - "Am I related to X?" - Parse query → search DB → generate response - Use graph traversal to trace relationships - Respond via AI in full sentences

Outputs: - Q&A form or chat-like interface - Simple relationship graph backend - Connection explainer logic

---

Phase 4: Cultural Extensions

Goals: - Add place of origin maps (e.g., Bugiri, Uganda) - Clan/subclan relationships - Cultural notes (e.g., totems, rituals, idioms) - Elders' audio stories + transcription (Whisper) - Browse by subclan, migration path, or surname root

Outputs: - Map viewer (Leaflet.js) - Audio-to-text converter - Clan registry model

---

Phase 5: Multi-Family & Community Mode

Goals: - Multi-family support (each with own namespace) - Invite family members to contribute - Role-based dashboards - User registration + moderation - Community leaderboard for contributions

Outputs: - Registration system - Contribution history tracker - Family selector in dashboard

---

Phase 6: Advanced Features & Optimization

Goals: - Merge suggestion dashboard - Conflict resolution tools - Relationship tracing via networkx - Admin analytics: growth, accuracy, usage

Outputs: - AI-augmented moderation dashboard - Integrity scoring system - Timeline and statistics view

---

Long-Term Vision

- Scale across Luhya clans and other ethnic groups
- Become a cultural preservation platform for Kenya and Africa
- Publish family storybooks, clan trees, and oral histories
- Train AI on Luhya dialect stories for preservation

The roadmap remains open to revision — but the structure ensures smooth growth without technical debt.