Project AAYWA Full Digital Platform Documentation

1. Introduction

This document provides a **complete technical and functional specification** for the **AAYWA Digital Platform**, a role-based system supporting a 3-year social business in Rwanda that empowers **100 young women and adolescent mothers** through **nutrition-sensitive agriculture**, **organic fertilizer (compost)**, **VSLAs**, and **avocado/macadamia farming**.

The platform includes:

- A **public web landing page**
- A **role-based web dashboard** (for managers, agronomists, buyers)
- An **offline-first mobile app** (for farmers, champions, VSLA officers)
- A **RESTful backend API**
- A **PostgreSQL + PostGIS database**

All components are designed to support **avocado production**, **warehouse/storage management**, **50/50 profit-sharing**, and **VSLA financial inclusion**.

1 2. Full Project Structure

Listing 1: Project Root Directory

```
aaywa-platform/

  website/                          Public marketing site
  web-dashboard/                   Admin & role-based web app
  mobile-app/                      Flutter mobile app (offline-first)
  backend/                         Node.js + Express API
  database/                        SQL schema, migrations, seeders
  docs/                            This documentation + specs
  tests/                           Unit & integration tests
  docker-compose.yml
 .gitignore
 README.md
```

2 3. Website (Public Landing Page)

Purpose

Attract donors, partners, and public awareness.

Folder Structure

```
website/
 public/
    index.html
    assets/
        images/hero.jpg, map.png, logo.svg
        favicon.ico
 src/
     components/
        Header.jsx
        Hero.jsx
        AboutSection.jsx
        ModelSection.jsx
        ImpactSection.jsx
        PlatformSection.jsx
        PartnersSection.jsx
        BlogSection.jsx
        ContactSection.jsx
        Footer.jsx
     pages/Home.jsx
     styles/main.css
     App.jsx
```

Key Pages

- **Home**: Hero, about, model, impact, platform, partners, contact
- **Blog**: Stories from cohorts
- **Contact**: Form + info

Tech Stack

React.js + Tailwind CSS, hosted on Vercel/Netlify.

3 4. Web Dashboard (Role-Based Admin)

Purpose

Manage operations for Project Managers, Agronomists, Buyers.

Folder Structure

```
web-dashboard/
 src/
    components/layout/
       Sidebar.jsx
       Topbar.jsx
       MapView.jsx
    components/modules/
       Farmers/
       Cohorts/
       Inputs/
       Sales/
       VSLAs/
       Compost/
       Training/
       Warehouses/           NEW
           WarehouseList.jsx
           InventoryTracker.jsx
           MaintenanceLog.jsx
           UserFeeManager.jsx
           TemperatureMonitor.jsx
    pages/
       Dashboard.jsx
       FarmersPage.jsx
       WarehousesPage.jsx
       ...
    services/api.js
```

User Roles & Access

Role	Features
Project Manager	Full access: KPIs, maps, users, MEL reports
Agronomist	Farm plots, input invoices, compost batches, warehouse oversight
Buyer	View catalog, place orders, view quality certs

4 5. Mobile App (Flutter Offline-First)

Purpose

Used by farmers, champions, VSLA officers in rural areas.

Folder Structure

```
mobile-app/
 lib/
    main.dart
    models/
       farmer.dart
       cohort.dart
       input_invoice.dart
       sale.dart
       vsla.dart
       compost_batch.dart
       training_session.dart
       warehouse.dart         NEW
    screens/
       auth/
       home/
       inputs/
       sales/
       vsla/
       compost/
       training/
       warehouses/
           StorageFacilitiesMapScreen.dart
           MyStoredProduceScreen.dart
           StorageFeeHistoryScreen.dart
           TemperatureAlertsScreen.dart
    services/
        database_service.dart      Drift (SQLite)
        sync_service.dart
        api_service.dart
```

Offline Strategy

- All data saved locally via **Drift (SQLite)**
- Syncs to cloud when internet available
- Conflict resolution: server timestamp wins for financial data

5 6. Backend API (Node.js + Express)

Purpose

Central business logic, authentication, data processing.

Folder Structure

```
backend/
 src/
    config/
    controllers/
       farmerController.js
       cohortController.js
       inputController.js
       saleController.js
       vslaController.js
       compostController.js
       trainingController.js
       warehouseController.js  NEW
    routes/
       farmers.routes.js
       ...
       warehouses.routes.js
    models/
       Farmer.js
       ...
       Warehouse.js            NEW
    middleware/auth.js
    services/
       profitShareCalculator.js
       storageFeeCalculator.js NEW
    app.js
 .env
 package.json
```

Key Endpoints

- `POST /api/sales` auto-deduct inputs split 50/50
- `POST /api/warehouses/:id/store` log stored produce
- `GET /api/warehouses/inventory` view current stock
- `POST /api/storage-fees/calculate` compute user fees

6 7. Database Schema (PostgreSQL + PostGIS)

Core Tables

```sql
-- Users (all roles)
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    phone VARCHAR(20) UNIQUE NOT NULL,
    full_name VARCHAR(100) NOT NULL,
    role VARCHAR(50) NOT NULL,
    language VARCHAR(10) DEFAULT 'rw',
    created_at TIMESTAMP DEFAULT NOW()
);

-- Farmers
CREATE TABLE farmers (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(id),
    cohort_id INTEGER REFERENCES cohorts(id),
    vsla_id INTEGER REFERENCES vsla_groups(id),
    date_of_birth DATE,
    household_type VARCHAR(50),
    location_coordinates GEOMETRY(POINT, 4326)
);

-- Cohorts
CREATE TABLE cohorts (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    cropping_system VARCHAR(50),
    boundary_coordinates GEOMETRY(POLYGON, 4326)
);

-- Input Invoices
CREATE TABLE input_invoices (
    id SERIAL PRIMARY KEY,
    farmer_id INTEGER REFERENCES farmers(id),
    items JSONB NOT NULL,
    total_amount DECIMAL(10,2),
    status VARCHAR(20) DEFAULT 'pending'
);

-- Sales
CREATE TABLE sales (
    id SERIAL PRIMARY KEY,
    farmer_id INTEGER REFERENCES farmers(id),
    input_invoice_id INTEGER REFERENCES input_invoices(id),
    crop_type VARCHAR(50),
    quantity DECIMAL(10,2),
    gross_revenue DECIMAL(10,2),
    input_cost DECIMAL(10,2),
    net_revenue DECIMAL(10,2),
    farmer_share DECIMAL(10,2),
    sanza_share DECIMAL(10,2),
    sale_date TIMESTAMP DEFAULT NOW()
);

-- VSLA Groups
CREATE TABLE vsla_groups (
    id SERIAL PRIMARY KEY,
    cohort_id INTEGER REFERENCES cohorts(id),
    name VARCHAR(100),
    seed_capital DECIMAL(10,2) DEFAULT 12000
);

--  WAREHOUSE MODULE (NEW)
CREATE TABLE storage_facilities (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    type VARCHAR(50),
    location_coordinates GEOMETRY(POINT, 4326),
    capacity_kg DECIMAL(10,2),
    current_usage_kg DECIMAL(10,2) DEFAULT 0,
    user_fee_per_kg_per_week DECIMAL(10,2),
    vsla_id INTEGER REFERENCES vsla_groups(id)
);

CREATE TABLE stored_produce (
    id SERIAL PRIMARY KEY,
    warehouse_id INTEGER REFERENCES storage_facilities(id),
    farmer_id INTEGER REFERENCES farmers(id),
    crop_type VARCHAR(50),
    quantity_kg DECIMAL(10,2),
    stored_at TIMESTAMP DEFAULT NOW(),
    retrieved_at TIMESTAMP,
    storage_fee_paid DECIMAL(10,2),
    payment_status VARCHAR(20) DEFAULT 'pending'
);

```

```
-- Optional: IoT temperature logs
CREATE TABLE temperature_logs (
    id SERIAL PRIMARY KEY,
    warehouse_id INTEGER REFERENCES storage_facilities(id),
    temperature_celsius DECIMAL(5,2),
    recorded_at TIMESTAMP DEFAULT NOW()
);
```

Listing 2: SQL Schema Snippet

7 8. Key Data Linkages

Action	Tables Updated
Farmer stores avocado	`stored_produce` + `storage_facilities.current_usage_kg += X`
Sale occurs	`sales` + `input_invoices.status = 'repaid'` + `vsla_transactions`
Stipend paid	`compost_workdays.payment_status = 'paid'`
VSLA meeting	`vsla_transactions` (savings/loans)
Training session	`training_attendance`

8 9. Tech Stack Summary

Component	Technology
Mobile App	Flutter (Android/iOS), Drift (SQLite), Firebase Auth
Web Dashboard	React.js, Tailwind CSS, Recharts, Leaflet
Backend	Node.js, Express, JWT
Database	PostgreSQL 15 + PostGIS
Auth	Phone + OTP (Africas Talking API)
Payments	MTN/Airtel Mobile Money API
Hosting	Google Cloud (Johannesburg region)
Maps	Google Maps (mobile), Leaflet (web)

9 10. Core Workflows

A. Avocado Storage Flow

1. Farmer harvests avocado brings to aggregation center
2. Agronomist weighs offers storage
3. Farmer selects warehouse in mobile app enters kg
4. System logs in `stored_produce`, calculates fee

5. When sold retrieves produce finalizes fee updates VSLA balance

B. Profit-Sharing Flow

1. Sale recorded `gross_revenue = qty Œ price`
2. `input_cost` deducted `net_revenue = gross - input`
3. `farmer_share = net Œ 0.5`, `sanza_share = net Œ 0.5`
4. Settlement statement generated shared with farmer + VSLA

C. VSLA Operation

- Seed capital: 10/member recorded as opening savings
- Weekly meetings savings/loans logged in `vsla_transactions`
- Maintenance fund used for warehouse repairs

10 11. Conclusion

This platform fully digitizes **Project AAYWA**s innovative social business model, ensuring:

- **Transparency** in input repayment and profit-sharing
- **Financial inclusion** via VSLAs
- **Reduced post-harvest losses** via warehouse management
- **Scalability** through modular, cloud-based architecture
- **Accessibility** via offline-first mobile design

The system is ready for implementation and can be extended to **1,000+ farmers nationwide**.

Prepared for: AAYWA & Sanza Alkebulan Ltd.
Date: January 2026
Version: 1.0