Functional Specification Document (FSD)

SafiLocate – Smart Lost & Found Platform

Version: 1.0
Status: MVP Specification
Prepared for: GetRwanda / Ikoranabuhanga Rigezweho
Prepared by: Joesure (Product Manager & Technical Architect)

---

1. Introduction

1.1 Purpose of this Document

This Functional Specification Document defines the **functional requirements**, **user flows**, **system behaviors**, **data structures**, and **constraints** for the MVP release of the **SafiLocate – Smart Lost & Found Platform**.

The document ensures:

- A shared understanding across development, design, product, and stakeholders.
- Clear reference for Vibe Coding workflows (Antigravity IDE & Claude Code).
- Alignment for future phases and scalability.

1.2 Product Summary

SafiLocate is a web-based, mobile-first platform that enables individuals and businesses to:

- Report **lost items** (paid listing)
- Report **found items** (free listing)

- Search and filter listings
- Use AI-assisted categorization and tag generation
- Match lost and found items using user-submitted data
- Connect users to resolve claims
- Provide optional finder “tips” after successful return

The MVP is optimized for **Rwanda**, designed with **low-cost operations** and **simple compliance** constraints.

1.3 Goals of the Platform (MVP)

Primary Goals

- Enable users nationwide to post and find lost items quickly.
- Build a searchable, AI-enhanced database of items.
- Allow finders to report found items *without paying*.
- Let people who lost items *pay a small fee* to post their listing.
- Build user trust through transparency and simplicity.

Secondary Goals

- Create a foundation for onboarding institutions (banks, moto coops, police, venues) in future phases.
- Enable scalable architecture for mobile apps and multi-country rollout.

---

2. System Overview

2.1 Platform Architecture (MVP)

Frontend

- Framework: Next.js (recommended) or React + Vite
- Responsiveness: Mobile-first
- Components:
    - Homepage
    - Lost Item Flow
    - Found Item Flow
    - Search & Filters
    - Admin Dashboard
    - Payment Page

Backend

- Node.js + Express OR Next.js API Routes
- Database: PostgreSQL (or SQLite for initial development)
- AI integration: Lightweight tag extraction & category normalization
- Payment Integration: Flutterwave (RWF)

Hosting Options

- Vercel (frontend)
- Railway / Render / Supabase / NeonDB (database & backend)
- Replit (testing environments)

3. User Roles & Permissions

Role	Description	Permissions
Guest User	Browses the platform	Search items, view listings
Finder	Reports found items	Create found listing (free), respond to claims
Loser (Seeker)	Reports lost items	Create lost listing (paid), claim found items
Admin	Platform oversight	Approve/reject listings, ban/hide users, manage categories
Super Admin (Future)	Deep system control	Database tools, config management

---

4. Core Functional Requirements (MVP)

4.1 Homepage

Functionalities

- Display platform mission: “Smart Lost & Found for Rwanda”
- Buttons to start:
  - “Report Lost Item”
  - “Report Found Item”
  - “Search Items”
- Statistics counters (optional): total items posted, matched, etc.

4.2 Found Item Posting (Free)

Purpose

Allows users who find an item to report it quickly without barriers.

User Flow

1. User clicks **“Report Found Item”**.
2. System shows form:
   - Category (dropdown)
   - Title
   - Description
   - Location (sector/district)
   - Photo upload (optional)
   - Phone number (required)
3. On submit:
   - Backend validates fields
   - AI generates tags
   - Entry saved as `status = pending`
4. User receives confirmation with listing ID.

Functional Requirements

- Validate phone number format
- Validate required fields
- Store image (local or cloud)

- AI call for tags (optional)

4.3 Lost Item Posting (Paid)

Purpose

Monetize platform usage while maintaining fairness.

User Flow

1. User clicks **“Report Lost Item”**.
2. Form fields:
    - Category
    - Item details
    - Description
    - Image upload
    - Location last seen
    - Contact info
    - Optional “reward message”
3. On submit:
    - Create **pending lost listing**
    - Generate Flutterwave payment intent
    - Redirect to payment page
4. On payment success:
    - Listing is marked **active**

    - User receives confirmation

Functional Requirements

- Payment API integration
- Payment callback endpoint
- Ability to mark listing as active/inactive
- Handle payment failure gracefully

4.4 Search & Filter

Purpose

Allow users to find potential matching items quickly.

Functionalities

- Search bar
- Filters:
    - Category
    - Location
    - Date range
- Results:
    - Grid/list of matching items
    - Each item shows:
        - Thumbnail
        - Category

        - Title
        - Location
        - “Claim item” button

AI Enhancements

- Tag-based search
- Text-to-tag conversion
- Synonym expansion for queries

4.5 Claim Flow

Purpose

Connect seekers with finders securely.

User Flow

1. User clicks **“Claim This Item”**
2. Simple claim form:
    - Phone number
    - Optional verification message
3. Backend creates a `claim` entry
4. Notification goes to finder/admin (MVP: email or dashboard entry)

Functional Requirements

- Store claim

- Protect finder contact info
- Allow admin to review claims

4.6 Payment System (Flutterwave)

Functionalities

- Initiate payment session
- Payment callback URL
- Payment verification endpoint
- Store payment status in DB
- Allow retries

Transaction Types

Type	Fee	Notes
Lost item posting	Required	Core revenue source
Finder tipping	Optional	Post-success appreciation

4.7 Admin Dashboard

Pages

- **Login** (password ONLY for MVP)
- **Pending items**
- **Approved items**
- **Blocked items**

- **Claims Review**
- **User management** (optional MVP)

Functionalities

- Approve, reject, hide listings
- Mark suspicious users
- See basic analytics:
    - Listings per category
    - Lost vs Found breakdown

---

5. Non-Functional Requirements

5.1 Performance

- Pages must load within 1.5s on 3G mobile networks.
- Search should respond within 800–1200ms.

5.2 Security

- Rate limiting for form submissions
- Protect API routes via middleware
- Sanitize all user inputs
- Secure secret keys using `.env`

5.3 Scalability

- Stateless backend (optimal for Vercel/Cloudflare Functions)
- SQL schema designed for efficient searches

5.4 Reliability

- Automatic retries for failed AI calls
- Payment verification fallback endpoint

---

6. Database Schema (MVP)

6.1 Tables

Table: users

Field	Type	Notes
id	UUID	Primary key
phone	String	Unique
role	Enum(user, admin)	MVP may avoid full auth

Table: found_items

Field	Type	Notes
id	UUID	PK
title	String	100 chars
description	Text	

category	Enum	ID, Wallet, Phone, Document, Other
tags	Array	AI-generated
location	String	
contact_phone	String	
image_url	String	nullable
status	Enum(pending, approved, hidden)	
created_at	Timestamp	

Table: lost_items

Same as `found_items` +:

Field	Type	Notes
reward_message	String	optional
payment_status	Enum(pending, paid, failed)	
listing_status	Enum(pending, active, expired)	

Table: claims

Field	Type
id	UUID
item_id	FK to lost_items or found_items
claimant_phone	String
message	Text
created_at	Timestamp

Table: payments

Field	Type
id	UUID
user_phone	String
item_id	FK to lost_items
provider	String (“flutterwave”)
status	Enum(pending, success, failed)
reference	String
created_at	Timestamp

---

7. AI Integration Specification

7.1 Tag Generation API

Input: Item title + description
Output: JSON `{ tags: ["blue wallet", "nyamirambo", "id card"] }`

7.2 Category Normalization

Purpose: Ensure consistent database categories.

7.3 Optional Future AI Features

- Image-based object recognition
- Semantic similarity matching
- Fraud detection indicators

---

8. API Endpoints (MVP)

8.1 Lost Items

- `POST /api/lost` – create lost item
- `POST /api/lost/payment` – initiate payment
- `POST /api/lost/payment/callback` – verify payment
- `GET /api/lost/:id` – fetch details

8.2 Found Items

- `POST /api/found` – create listing
- `GET /api/found/:id` – fetch details

8.3 Search

- `GET /api/search?query=&category=&location=`

8.4 Claims

- `POST /api/claims`
- `GET /api/admin/claims`

8.5 Admin

- `POST /api/admin/items/approve`
- `POST /api/admin/items/block`
- `GET /api/admin/items/pending`

---

9. User Experience (UX) Requirements

9.1 Design Principles

- Friendly, minimal, trustworthy
- Rwandan context & colors
- Icons for categories
- Use large buttons for touch users

9.2 Accessibility

- High-contrast mode
- Minimal text walls
- Large input fields

---

10. Future Expansion (Post-MVP Features)

Phase 2

- AI-based direct matching (found → lost)
- Verified accounts (NID API when possible)
- Organization dashboards (Banks, Schools, Gyms)
- Push notifications
- Mobile app (React Native)

Phase 3

- National Lost & Found Registry
- Integration with Police & Irembo
- Insurance agency partnerships