Product Requirements Document

Social Saver Bot — Turning Instagram Saves into a Knowledge Base

Owner	Hackathon Team
Version	v1.0
Date	February 19, 2026
Scope	WhatsApp → Save Link → AI Tag/Summary → Searchable Web Dashboard

Document purpose. Define a clear, build-ready spec for a hackathon MVP that meets the scoring rubric: reliable WhatsApp→website flow, useful AI tagging/summaries, clean dashboard UX, and one small “wow”.

1. Executive Summary

People save valuable Reels/Posts (workouts, recipes, design tips, coding hacks) but can’t retrieve them later because Instagram saves are hard to search. Social Saver Bot lets users forward a social link to a WhatsApp bot and automatically turns it into a searchable personal knowledge base on a web dashboard.

2. Goals and Non-Goals

Goals

• User can send an Instagram Reel/Post URL to WhatsApp bot and receive a confirmation reply.
• The link appears on a web dashboard within seconds with AI-generated category + 1-sentence summary.
• Dashboard supports fast retrieval via keyword search and category filters.
• System is robust: if caption/text extraction fails, it still saves the URL and can optionally ask for a short user note.
• Includes one “wow” feature (e.g., Random Inspiration).

Non-Goals (de-scopes for hackathon)

• Perfect Instagram caption scraping or private-content access.
• Full user account management (email/password) unless trivial.
• Rich media processing (transcoding, downloading videos).
• Enterprise-grade moderation, analytics, or multi-tenant admin tooling.

3. Users and Use Cases

Primary user: A person who shares and saves Instagram links on WhatsApp and wants a searchable archive.

Core user stories

• As a user, I can forward an Instagram link to the bot and it is saved to my dashboard.
• As a user, I can search (e.g., “pasta”, “core workout”) and find old saves quickly.
• As a user, I can filter by category (Fitness, Food, Coding, Travel, etc.).

Nice-to-have stories

• As a user, I can paste multiple links in one WhatsApp message and all are saved.
• As a user, if the system can't read the caption, I can add an optional 1-line note to improve retrieval.
• As a user, I can click "Random Inspiration" to resurface something I saved earlier.

4. Functional Requirements

4.1 WhatsApp Bot

• Inbound: accept WhatsApp messages containing one or more URLs.
• URL parsing: extract all URLs from message text; ignore non-URL text.
• Reply: send a confirmation per message (or per URL) indicating saved status and category.
• Error handling: if URL is malformed or unsupported, reply with guidance instead of failing silently.
• Optional note flow: when extraction confidence is low, ask for an optional note; accept 'skip'.

4.2 Backend Processing and AI

• Source detection: identify source type (instagram / x / blog / other) from URL domain.
• Metadata: attempt to fetch lightweight metadata (title / OpenGraph where available).
• AI: generate (a) category from a fixed taxonomy, (b) 3–8 tags, (c) 1-sentence summary.
• Validation: enforce strict JSON output from the model; apply fallback category=Other if invalid.
• Persistence: upsert a save record linked to user identity (WhatsApp phone or hashed identifier).
• Latency: target end-to-end processing within ~3–6 seconds; if slower, send a 'processing' message then a 'saved' message (optional).

4.3 Web Dashboard (React)

• Cards layout listing saved items (newest first).
• Search: keyword search across summary, tags, note, title, url.
• Filters: category chips; optional source filter.
• Card content: source icon/text, created date, category, summary, tag chips, open link button.
• Wow: Random Inspiration button picks a random item for the user.

5. Data Model (Supabase/Postgres)

Table: **saves**

Field	Type	Notes
id	uuid (pk)	Generated by DB
created_at	timestamptz	Default now()
user_phone	text	WhatsApp sender; can be hashed for privacy
url	text	Original link
source	text	instagram | x | blog | other
title	text	Optional metadata title
note	text	Optional user-provided note (fallback)
raw_text	text	Optional extracted caption/article excerpt
category	text	One of predefined categories

tags	text[]	3–8 keywords
summary	text	1 sentence

Indexes (recommended): (user_phone, created_at desc), and a text search index for summary/note/title where supported.

6. Non-Functional Requirements

• Reliability: never drop a user link; store URL even if extraction fails.
• Security/Privacy: avoid storing media; optionally hash phone numbers; keep keys in env vars.
• Usability: dashboard should be readable on desktop and mobile; search and filters should be obvious.
• Observability: log webhook requests and processing outcomes for debugging during demo.

7. Milestones (Hackathon Plan)

• Webhook + WhatsApp replies working end-to-end (even without AI).
• Supabase writes + dashboard lists saved URLs.
• AI tagging + summary integrated with JSON validation.
• Search + category filters implemented.
• Wow feature (Random Inspiration) + demo polish.
• Submission assets: screen recording + repo + architecture diagram.

8. Demo Script and Deliverables

Demo script (45–60s)

• Open WhatsApp → send an Instagram link to bot.
• Bot replies: “Saved ■ Category: … • Summary: …”
• Open dashboard → new card appears.
• Type a keyword in search → results filter instantly.
• Click “Random Inspiration” → show resurfaced item.

Deliverables: screen recording of WhatsApp→web, code repository link, and a simple architecture diagram.

Appendix: Category taxonomy (default)

Fitness, Coding, Food, Travel, Design, Business, Self-Improvement, Other