Ask Mackenzie - Project Plan

heck yesss — we're building "Ask Mackenzie," a polite little brain that eats PDFs & podcast transcripts for breakfast and then helps humans book a call with the coach. I'll keep it super clear, super actionable, and a lil' bit spicy. 🫡

Purpose / Goal (aka "why are we doing this?")

- **Primary goal (money & impact):** Turn curious visitors into **qualified leads** who **book a discovery call** with Mackenzie.
- **Secondary goal (brand & trust):** Give **accurate, evidence-grounded answers** on health/nutrition/fitness (no bro-science), with **clear citations** and a warm coaching vibe.
- **Tertiary goal (learning loop):** Capture what people ask, what content supports great answers, and what nudges convert → use that data to **improve content + funnel**.

Success metrics (day-1 simple, day-30 smarter):

- **North Star:** % of sessions that click **"Book with Mackenzie"** and reach the booking page.
- Supporting: session satisfaction (thumbs-up rate), grounded-answer rate (answers with ≥1 citation), email capture rate, average messages per session, top intents answered.

End-to-End Project Plan (high level → you'll use this as your map)

0) Scope & guardrails (1 day)

- Define **topics allowed** (fat loss, muscle gain, macros, workout splits) and **topics to deflect** (medical diagnosis, eating disorders, urgent care → provide resources & "talk to a professional" guidance).
- Write a **short medical disclaimer** shown in the UI and appended to responses when necessary.

1) Data pipeline (your knowledge "pantry") (2–3 days)

- **Collect** PDFs, blog posts, Google Docs exports, and **podcast transcripts** (transcribe audio if needed).
- **Normalize** to clean text (title, author, date, URL/source).
- **Chunk** into ~700–1,000 tokens with ~150–200 token overlap; store **metadata** (source, section, tags like "nutrition", "hypertrophy").
- **Embed** chunks → store vectors + metadata in a vector DB.

MVP stack pick (cheap + chill): **Supabase Postgres + pgvector** for storage; **OpenAI embeddings** (or any strong, affordable embeddings) for vectors.

2) Retrieval-Augmented Generation (RAG) API (3–5 days)

- **Retrieve** top-k chunks by vector similarity (+ optional keyword BM25 hybrid).
- **(Nice boost)** Re-rank with a cross-encoder/reranker for better relevance.
- **Prompt** the LLM with: user question + top chunks + **answer style rules** ("sound like Mackenzie; cite sources inline; avoid medical claims; provide practical steps; end with a friendly CTA when appropriate").
- **Outputs:**
    1. final answer (grounded),
    2. list of citations with titles/links,
    3. **CTA suggestion** flag (should we show "Book a call?" button now?).

3) Safety & compliance (parallel, 1 day)

- **Intent filters**: If query looks like medical diagnosis, supplement dosing risks, or RED-flag patterns, return a **safe response** + resources + option to **book a call** instead of advice.
- **Tone & boundaries:** "Coach, not clinician." Always encourage professional consultation for medical issues.

4) Web app (2–4 days)

- **Frontend**: Minimal chat UI (think ChatGPT-lite) with message bubbles, inline citations, thumbs up/down, "Use my email to get tips," and a bold **Book** button.
- **Booking flow**: Integrate **Calendly/Cal.com** in a modal or separate page; pass UTM params (source=intent).
- **Empty state**: "Ask me anything about fat loss, muscle gain, or nutrition. I cite sources, promise."

MVP stack pick: **Next.js on Vercel** (frontend), **FastAPI or Node/Express** (backend). Keep it boring; boring ships.

5) Analytics & events (1–2 days)

- Log: page_view, message_sent, answer_cited, **book_click**, book_success, email_captured, thumbs_up/down.
- Simple destinations: Vercel Analytics + a **Google Sheet/Airtable** via webhook for early lead tracking.

6) Deployment & ops (1 day)

- Environment variables, API keys in Vercel/host.

- **Rate limits** + **request logging** (store prompts, retrieved doc IDs, model outputs with PII hygiene).

7) Evaluation & tuning (ongoing, first pass 1–2 days)

- Build a tiny **golden set** of 30–50 common questions; check: **Retrieval@k**, groundedness, and "coach-vibe" score.
- Iterate: chunk size, top-k, reranker on/off, prompt tweaks, answer template.

8) Growth experiments (week 2+)

- Trigger CTA earlier for high-intent queries ("custom plan", "coach me").
- Offer **lead magnet**: "Free 7-day protein plan" → email gate → book.

Initial Steps (setup) — do these before writing real code

Think of this like laying the gym flooring before moving in the squat rack.

A) Accounts & keys (90 min sprint)

- **Vercel** (frontend deploys), **GitHub** (repo), **Supabase** (Postgres + pgvector), **LLM provider** (OpenAI/Anthropic; start with one), **Calendly/Cal.com** (booking), **Email capture** (ConvertKit/Mailchimp or just Airtable to start).
- Create API keys; store locally via a **.env** file; plan secrets in Vercel/host.

B) Repo & folders (30–45 min)

```
ask-mackenzie/
  apps/
    web/          # Next.js (UI)
    api/          # FastAPI or Express (RAG endpoints)
  data/
    raw/          # PDFs, transcripts (never commit private files)
    processed/    # cleaned JSONL or markdown
  scripts/
    ingest/       # chunking + embeddings → Supabase
    evals/        # golden set & test runner
  docs/
    system_prompt.md
    style_guide.md  # Mackenzie's voice & tone
```

C) Database bootstrapping (45–60 min)

- Create Supabase project → enable **pgvector**.

- Tables:
    - documents(id, title, source_url, author, topic, created_at, updated_at)
    - chunks(id, document_id, chunk_index, content, tokens, embedding VECTOR, headings, tags JSONB)
    - events(id, user_id, type, payload JSONB, created_at)
    - leads(id, email, utm JSONB, created_at)
- Add indexes on (embedding), (document_id, chunk_index), (tags).
- Generate a **service role** key for server-side writes only (NEVER in the browser).

D) Content prep (2–4 hours)

- Gather PDFs + podcast audio. If you need transcripts, run a local or cloud transcription (e.g., Whisper) → **.txt** files.
- Normalize: a quick script to output **JSONL** where each item = {title, author, date, source, text}.
- Tag documents by theme (nutrition, training, mindset) to help filtering.

E) Ingestion script (2–3 hours)

- Python script that:
    1. Loads normalized JSONL,
    2. Splits into chunks (700–1,000 tokens, 150–200 overlap),
    3. Calls embeddings,
    4. Upserts into documents + chunks.
- Save a tiny **sample set** (like 3 docs) and test end-to-end before doing the whole library (save API $$).

F) System prompt & style guide (45–60 min)

- Draft **system_prompt.md**:
    1. Persona: "Supportive fitness coach (Mackenzie). Evidence-based. Friendly. Clear. No medical claims."
    2. Must: cite sources inline like [1], [2]; end with **1–2 practical next steps**; offer **"Book a discovery call"** if the user signals coaching intent.
    3. Safety: if diagnosis/medication appears → disclaim + encourage professional help.
- Draft **style_guide.md**: greeting tone, sentence length, emoji rules (lightly, not Vegas), structure template:
    1. Quick answer
    2. Why it works (science, briefly)
    3. Exactly what to do (steps)
    4. Sources
    5. Soft CTA

G) Design the UI skeleton (60–90 min)

- Decide layout:
    - Header (logo + **Book** button),
    - Chat pane,
    - Message composer with “press Enter to send,”
    - Inline citations (expand to show titles),
    - Thumbs up/down + “Was this helpful?”
- Empty state copy + disclaimer in the footer.

H) Analytics & funnels (45 min)

- Pick your early stack: Vercel Analytics + a simple **webhook → Google Sheet/Airtable** for book_click, book_success, email_captured.
- Define event names now so you don’t spaghetti later.

I) “Definition of Done” for MVP (write this down)

- Can answer 20–30 common questions with sources, under 3s P50 latency.
- Book button works and passes UTM.
- 10 test users can complete “question → book” without you babysitting.
- Log retention and error handling exist (no silent failures).

Sanity notes (because future-you will thank present-you)

- **Don’t** jam all the content on day one. Start with 10–20 high-quality sources Mackenzie trusts.
- **Don’t** over-optimize the model choice on day one. Make retrieval good first; LLM can be swapped later.
- **Do** keep a tiny **eval set** and run it after any change (chunking, top-k, prompt, reranker).
- **Do** bias toward **simple + shipped**. We iterate once leads start flowing. 💸

—----