Perfect—let’s make this *concrete*. You already know the steps; now I’ll give you the **blueprint**: what the website will look like, what features it will have, and how all the parts connect. Think of this like an **architectural drawing** before you lay bricks.

**📐 Website Blueprint for “Legal AI Assistant”**

**1. User Journey (Flow)**

1. **Landing Page** → User understands what the platform does.
2. **Search Page** → User types query (“alimony after divorce”).
3. **Results Page** → AI answer with:
   * Plain-English explanation
   * Key legal sections
   * Important cases
   * Next steps (general guidance)
   * Sources (clickable)
4. **Source Viewer** → Opens law text / case extract in full.
5. **User Tools** → Save answer, download PDF, switch language.
6. **Help & Disclaimer** → Remind: “Not legal advice, just information.”

**2. Core Pages & Features**

**🔹 A) Landing Page**

* Hero text: *“Understand laws in plain language, backed by real cases.”*
* Search bar (direct access).
* Buttons: “Start Search” / “How it works.”
* Minimalist design → trust-building.

**🔹 B) Search Page**

* **Search Box** (big and centered).
* Filters on the left:
  + Jurisdiction (India / Delhi HC / Supreme Court, etc.)
  + Topic area (Family, Criminal, Property, Consumer)
  + Year range (cases after 2015).
* Option: “Explain like I’m [15 | 21 | Lawyer]” toggle.
* Optional: Upload a PDF (judgment/notice) for summary.

**🔹 C) Results Page**

**Layout (grid-based, 2-column):**

* **Left column → AI Answer**
  + **In simple terms:** bullet summary in plain English.
  + **Key sections of law:** clickable links (e.g., HMA Section 24).
  + **Important cases:** list with 1-line takeaway + [citation].
  + **What to do next:** (general, not advice).
  + **Disclaimer:** “Not a substitute for lawyer consultation.”
* **Right column → Sources Panel**
  + Expandable case extracts.
  + Full bare act sections.
  + Direct link to official judgment PDFs.
* Buttons: “Save answer”, “Download PDF”, “Share”.

**🔹 D) Source Viewer**

* Side-by-side view:
  + Left: highlighted law section/case paragraph cited.
  + Right: full official text/document.
* “Copy citation” button.
* Date & jurisdiction shown clearly.

**🔹 E) Saved Answers / Dashboard (optional in MVP)**

* User logs in → can see previous queries.
* Download all answers as PDFs.
* Organize by topic (“Family Law”, “Consumer Disputes”).

**🔹 F) Help & Disclaimer Page**

* Clear FAQ:
  + “Is this legal advice?” → No.
  + “Where does data come from?” → Official judgments, bare acts.
  + “How accurate is it?” → Updated regularly.
* Contact (for feedback/reporting errors).

**3. UI / UX Design Blueprint**

* **Design philosophy:** clean, trustworthy, minimal.
* **Colors:** Blue (trust), White background, Black text.
* **Typography:** Serif for titles (law feels authoritative), Sans-serif for body (easy reading).
* **Card-style containers** with rounded corners for answers.
* **Animations:** Smooth fade-in for results, not flashy.
* **Mobile-friendly** (lawyers/students often use phones).

**4. Architecture (Behind the Scenes)**

**Frontend (User side)**

* **Framework:** Next.js + TypeScript
* **Styling:** TailwindCSS (fast, clean)
* **UI Components:** ShadCN UI (cards, buttons, modals)
* **Search UX:** Autocomplete for common topics

**Backend (Brains)**

* **API layer:** FastAPI (Python)
* **Search & Retrieval:**
  + **Keyword search:** Elasticsearch / Postgres Full Text
  + **Semantic search:** Vector DB (PGVector or Weaviate)
* **LLM Orchestration:** LangChain / LlamaIndex (RAG pipeline)
* **Database:** PostgreSQL (metadata: case title, citation, court, date, chunk\_id)
* **File storage:** S3 bucket / GCP bucket (for raw PDFs, scans).

**AI Workflow**

1. User query → Backend
2. Query expansion (synonyms, legal keywords)
3. Search DB → Retrieve top relevant chunks (law sections + case extracts)
4. Feed to LLM with strict prompt → AI generates structured response
5. Response + citations returned to frontend

**Security & Guardrails**

* Don’t store user queries with PII.
* Prompts enforce: “No personal legal advice.”
* Safety layer: detect sensitive queries (domestic abuse, child issues) → show helpline numbers.

**5. Database Blueprint (Simplified Schema)**

**Table: laws**

* id
* title (e.g., “HMA Section 24”)
* jurisdiction
* section\_text
* year
* url

**Table: cases**

* id
* case\_name
* citation (e.g., “AIR 2010 SC 1234”)
* court
* date
* paragraph\_text
* url

**Table: embeddings**

* id
* chunk\_text
* vector (embedding array)
* source\_type (law/case)
* source\_id

**6. Tech Stack Summary**

* **Frontend:** Next.js, Tailwind, ShadCN
* **Backend:** FastAPI, LangChain/LlamaIndex
* **Search:** Postgres + PGVector OR Weaviate
* **Data:** Crawled official bare acts + case law (via scraping/OCR)
* **Infra:** Docker, any cloud provider
* **LLM:** GPT-4o, LLaMA 3, or other depending on budget

**7. MVP Milestones**

1. ✅ Build landing page + search box.
2. ✅ Ingest *only Divorce/Alimony laws + 50–100 landmark cases*.
3. ✅ Enable keyword + vector search.
4. ✅ Set up RAG pipeline (query → retrieve → LLM → structured response).
5. ✅ Display answer with citations + source viewer.
6. 🚀 Launch closed beta with 10–20 test users.

💡 Think of this blueprint as your **map**. Now, depending on your skill level and team, I can also:

* Draw a **system architecture diagram** (boxes + arrows: frontend → backend → DB → LLM).
* Or design a **UI wireframe** (how each screen looks, like a mockup).

👉 Do you want me to sketch a **visual architecture diagram** first, or a **screen-by-screen wireframe** for the website?