Indian Law AI Chatbot

Streamlit-based legal Q&A assistant with a self-refreshing case-law dataset and multi-key LLM failover

A production Streamlit chatbot that answers questions on Indian law — the Bharatiya Nyaya Sanhita, Bharatiya Nagarik Suraksha Sanhita, Bharatiya Sakshya Adhiniyam, the Constitution of India, and case law from the Supreme Court and all 24 High Courts — grounded in a **20,700+ entry curated Q&A dataset** built by parsing raw legislative text and court judgments. The app streams Gemini responses with **automatic multi-key failover** and retry/continue logic, renders LLM-generated charts inline, persists chat history per conversation, and keeps its dataset current via a **fully automated daily GitHub Actions pipeline** that pulls fresh judgments from official RSS feeds with zero manual intervention. Built and maintained end-to-end across ~3,300 lines of Python.

20.7K	24+	3	~3.3K
Q&A DATASET ENTRIES	COURTS / TRIBUNALS TRACKED	INDEPENDENT SCRAPER SOURCES	LINES OF PYTHON

KEY CONTRIBUTIONS

- **Streaming chat engine with multi-key API failover** — a round-robin Gemini API key rotator with quota-aware failover, exponential backoff across retries, and auto-continue logic that transparently resumes responses cut off by output-token limits, hiding provider-level rate limiting from the end user.
- **Self-refreshing legal dataset via GitHub Actions** — a daily cron workflow pulls new Supreme Court, High Court, and tribunal judgments straight from Indian Kanoon's official RSS feeds (no API key, no rate limit), auto-committing new entries so the knowledge base stays current with no manual upkeep.
- **Offline regex-based Q&A extraction pipeline** — parses raw statutory text (BNS, BNSS, BSA, Constitution) directly into 20,000+ structured Q&A pairs with no LLM calls, plus a secondary LLM-assisted extraction path for chunked judgment/case-law text.
- **Multi-source legal web scraping** — three independent scrapers (indiacode.nic.in acts, IndianKanoon case law, legislative.gov.in) each normalize heterogeneous HTML into the same Q&A schema and merge into one unified dataset.
- **LLM-driven data visualization** — the model is instructed to emit structured chart JSON (bar, horizontal bar, pie, line, timeline) alongside prose answers, which is parsed out of the response and rendered as themed Plotly figures embedded directly in the chat.
- **Persistent multi-conversation history** — a SQLite-backed (WAL mode) chat/message store supporting multiple named conversations, auto-generated chat titles, and per-message regenerate/continue actions.
- **Custom dark-themed UI** — a hand-built Streamlit "Aether Neon" theme with a GSAP/Three.js/Anime.js animation layer, a custom SVG icon set, and safe markdown-to-HTML chat rendering with copy-to-clipboard.
- **Dual deployment targets** — configured for both Streamlit Cloud (secrets.toml) and Render (env vars via render.yaml), with automatic environment detection for API key sourcing at startup.

TECH STACK

Languages/Core: Python, Streamlit | **AI/LLM:** Google Gemini API (streaming, key rotation, retry/continue) | **Data:** SQLite, JSON, regex-based ETL | **Visualization:** Plotly | **Automation:** GitHub Actions (cron) | **Scraping/Parsing:** urllib, XML/RSS, HTML parsing | **Frontend:** Custom CSS, GSAP, Three.js, Anime.js | **Deployment:** Streamlit Cloud, Render