My Stack

**1. Desktop App Shell**

* **Electron (JavaScript/TypeScript)**  
  – Cross-platform desktop app (Win/Mac)  
  – Easy to bundle UI + backend logic
* **Alternatives:** Tauri (Rust+JS) for smaller binaries, but Electron’s ecosystem is more mature.

**2. Front-end UI**

* **React + TypeScript**  
  – For the embedded Chat-Bar panel in SPSS  
  – Component-based, easy to style and update
* **UI Framework:** Tailwind CSS (rapid styling)
* **Animations:** Framer Motion (for pulsing highlights).

**3. SPSS Integration Layer**

* **Python** (with SPSS Python Essentials)  
  – Directly execute SPSS syntax and read/write .sav files
* **UI Automation Fallback:** PyAutoGUI or AutoHotKey  
  – Click-level control for menu navigation when needed.

**4. Backend & AI**

* **Node.js (TypeScript)**  
  – Orchestrates between UI, SPSS-Python bridge, and AI
* **AI Engine:** OpenAI API (GPT-4) or Anthropic Claude by API  
  – LLM for context parsing, prompt engineering, and result interpretation
* **Local Caching/Fine-Tuning:**  
  – Store common test-templates; optionally fine-tune small LLM for stats terminology.

**5. Data & State Management**

* **SQLite** (embedded)  
  – Store project context, chat history, cleaned-data snapshots
* **Redux or Zustand** for UI state.

**6. Subscription & Licensing**

* **Stripe** for payment processing (student & institutional plans)
* **License Server** (Node.js) to issue and verify API keys or JWTs.

**7. Packaging & Updates**

* **Electron Builder** for installers and auto-updates
* **Sentry** (or similar) for error tracking and usage analytics

**Why This Stack?**

* **Speed to market:** Electron + React gets us running fast.
* **Deep SPSS hooks:** Python essentials let us call SPSS directly, ensuring reliability.
* **Rock-solid AI:** OpenAI/Claude APIs give us world-class language understanding.
* **Modular for exits/partnerships:** Clean separation ensures we can license the core engine or spin it into partners’ ecosystems.