Student Guider Chatbot Documentation

# 1. Project Overview

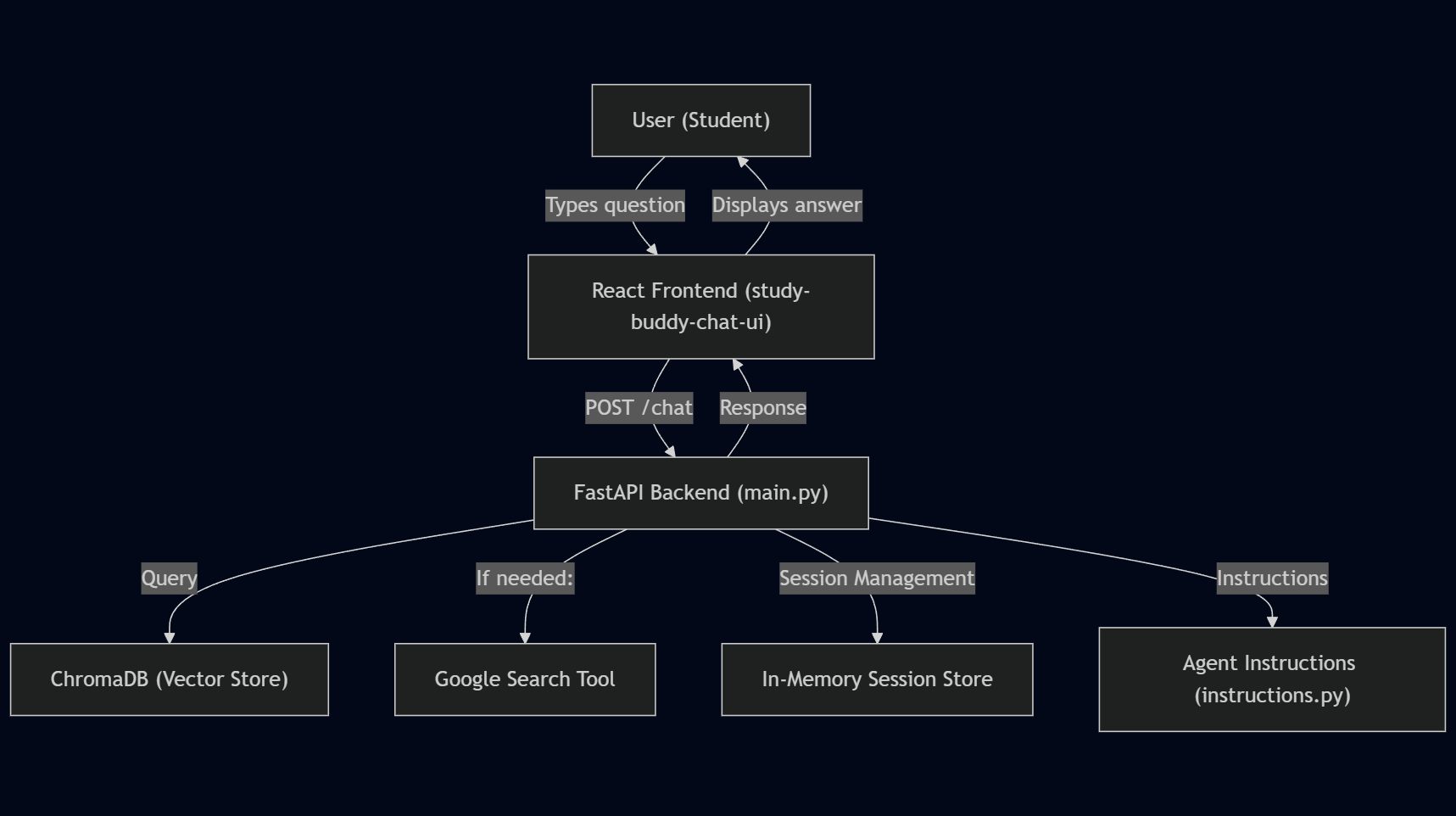
The Student Guider Chatbot is an AI-powered assistant that helps students with information about studying abroad, including programs, scholarships, eligibility, and application processes. The system consists of a Python backend (API and agent logic) and a modern React frontend (chat UI).

# 2. System Architecture

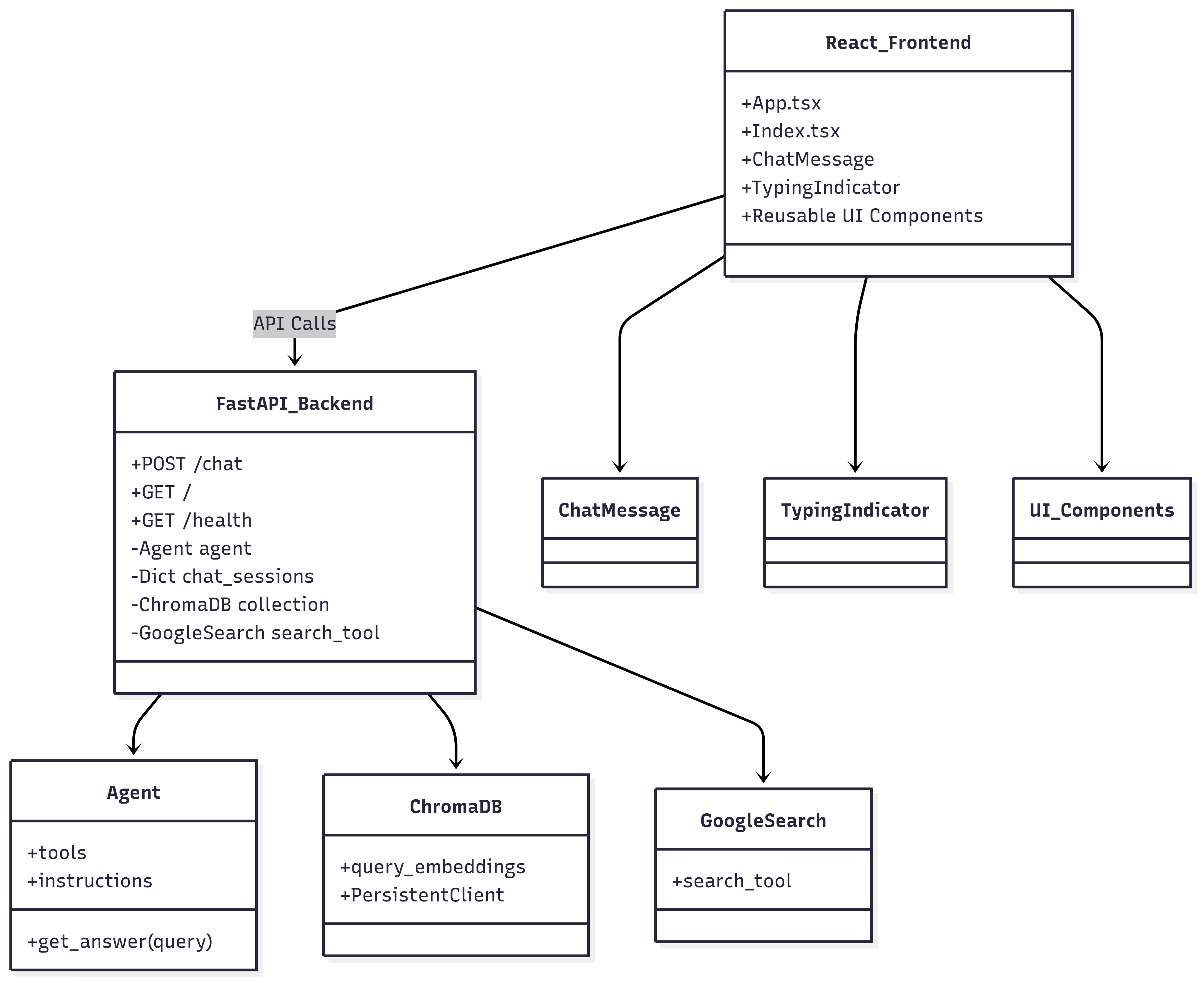
The system is composed of:

- React Frontend (study-buddy-chat-ui)  
- FastAPI Backend (main.py)  
- ChromaDB Vector Store  
- Google Search Tool (fallback)

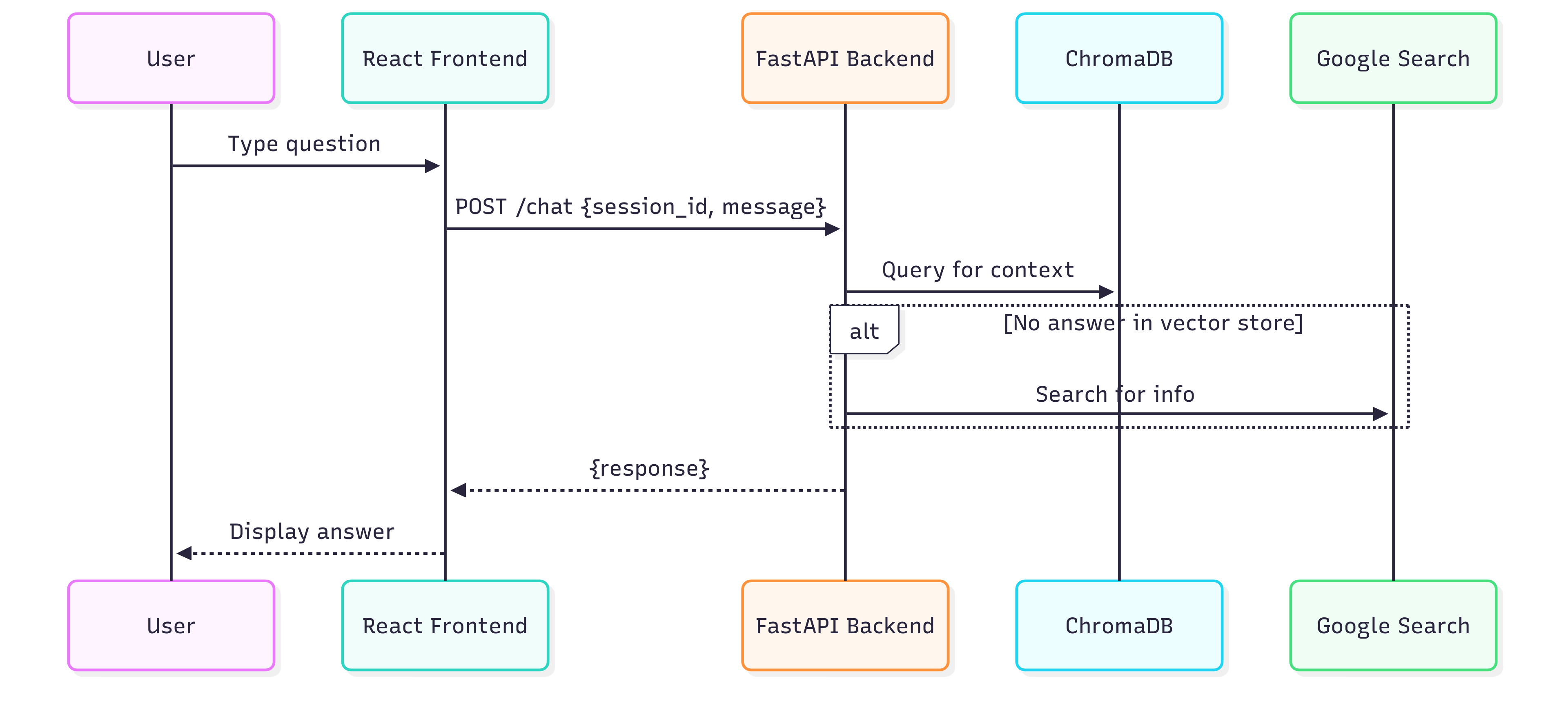
## System Flowchart

Shows the flow from user to frontend, backend, vector store, and back.

## Class Diagram

Shows main backend and frontend classes/components and their relationships.

## Sequence Diagram

Step-by-step message flow from user to assistant and back.

## Entity-Relationship Diagram

Session/message structure.

# 3. Backend (Python/FastAPI)

Features:  
- Conversational AI agent for study abroad guidance  
- Retrieval-augmented generation using a vector store (ChromaDB)  
- Live internet search fallback for up-to-date information  
- Session-based chat with in-memory session management  
- REST API built with FastAPI  
  
API Endpoints:  
- POST /chat: Main chat endpoint. Accepts { session\_id, message } and returns { session\_id, response }  
- GET /: Welcome and health check  
- GET /health: Service health check  
- OPTIONS /chat: CORS preflight  
  
Agent Logic:  
- Uses Google Gemini API for embeddings and content generation  
- Retrieves relevant context from ChromaDB using semantic search  
- Follows strict behavioral guidelines (see instructions.py), including asking clarifying questions, using vector store as primary knowledge, with internet search as fallback, explaining academic terms, and maintaining a supportive, student-friendly tone  
  
Session Management:  
- Sessions are tracked in-memory (dictionary keyed by session\_id)  
- Each session stores a list of message objects (role/content)

# 4. Frontend (React/Vite)

Features:  
- Modern chat interface for interacting with the assistant  
- Session persistence using sessionStorage  
- Auto-scrolling, typing indicator, and error handling for smooth UX  
- Reusable UI components (buttons, textareas, toasts, tooltips, etc.)  
- React Router for page navigation  
  
Main Components & Structure:  
- src/App.tsx: Main app entry, sets up providers and routes  
- src/pages/Index.tsx: Main chat page, handles message state, API calls, and UI  
- src/pages/NotFound.tsx: 404 page  
- src/components/: Chat message and typing indicator components  
- src/components/ui/: Large library of reusable UI primitives (button, toast, dialog, etc.)  
- src/hooks/: Custom React hooks (e.g., for toast notifications)  
- src/lib/utils.ts: Utility functions  
  
UI/UX Flow:  
1. User opens the app and sees a greeting  
2. User types a question and presses Enter  
3. The message is sent to the backend; a typing indicator is shown  
4. The assistant's response appears in the chat  
5. User can start a new session at any time

# 5. Setup & Running

Backend:  
1. Install dependencies: pip install -r requirements.txt  
2. Set up environment variables (e.g., GOOGLE\_API\_KEY)  
3. Run the FastAPI server: uvicorn main:app --reload  
  
Frontend:  
1. Navigate to study-buddy-chat-ui/  
2. Install dependencies: npm install  
3. Start the development server: npm run dev

# 6. Extending & Customizing

- To add new knowledge: Update the vector store using set\_vector\_store.py or similar scripts  
- To change agent behavior: Edit instructions.py  
- To add new UI features: Create new components in src/components/ or src/components/ui/

# 7. Appendix

Python Requirements:  
- streamlit  
- httpx  
  
Frontend Stack:  
- Vite  
- TypeScript  
- React  
- shadcn-ui  
- Tailwind CSS