Appointment Scheduling Agent — Architecture Diagram

```
Appointment Scheduling Agent — Architecture Diagram

Below is a single-page architecture diagram and accompanying notes that include all requested components

- Conversation agent flow
- Calendly API integration
- RAG pipeline for FAQs
- Tool calling (availability check, booking)
- Context switching mechanism (scheduling ↔ FAQ)
- Error handling paths

---------------------------------------------------------------------

MERMAID DIAGRAM (raw format)

flowchart LR
  subgraph Client
    A[User (Web Chat UI / Mobile)]
  end

  subgraph Frontend
    B[Chat UI (Vite React)
    - input box, bubbles
    - shows suggestions
    - retries/errors]
  end

  A --> B
  B --> |POST /api/chat/message| C[Backend API Gateway (FastAPI)]

  subgraph AgentLayer
    C --> D[Conversation Orchestrator / Router]
    D --> E[LLM / Policy Layer (Prompting)]
    D --> F[Session Store (in-memory / Redis)]
    D --> G[RAG Retriever]
    G --> H[FAQ Document Store (clinic_info.json / vector DB)]
    D --> I[Tool Caller]
    I --> J[Calendly Integration (mock/real)]
    I --> K[Availability Tool]
    I --> L[Booking Tool]
  end

  E --> D
  F --> D
  G --> D
  J --> I
  K --> I
  L --> I

  subgraph Persistence
    M[SQLite (bookings.db)]
    N[Logs / Monitoring]
  end

  J --> M
  L --> M
  D --> N

  NOTE: Context switching
    - Intent detection (schedule / reschedule / cancel / faq)
    - If FAQ -> short-circuit to RAG Retriever
    - If Schedule -> run scheduling flow with Tool calls
    - Maintain session state for multi-turn

  ERROR HANDLING:
```

```
    - Tool failure -> fallback
    - LLM unclear -> clarification
    - Calendly down -> retry or phone number
-----------------------------------------------------------------

Explanation:
(removed for brevity in PDF; can be re-added on request)
```