Conversational News Bot – Fullstack Prototyp

Bc.Phan Thi Anh Tuyet

API Design for Real RAG

- Potential contract to call real RAG service:
  - Request:
    - query: user question
    - user_id: for personalization & history
    - conversation_id: to send previous context
  - Response:
    - answer: generated text
    - sources: list of { title, url, snippet }
    - latency_ms, maybe model_id
- Could define as OpenAPI/Swagger snippet for future integration

Error Handling & UX

- Backend:
    - Simplicity: mocked data → fewer failure modes
- Frontend:
    - try/catch around fetch
    - On error: assistant message
      "**Something went wrong when calling the API. Please try again.**"
    - Loading state always cleared in finally
- UX:
    - User always sees *some* feedback (loading → answer or error)

State Management

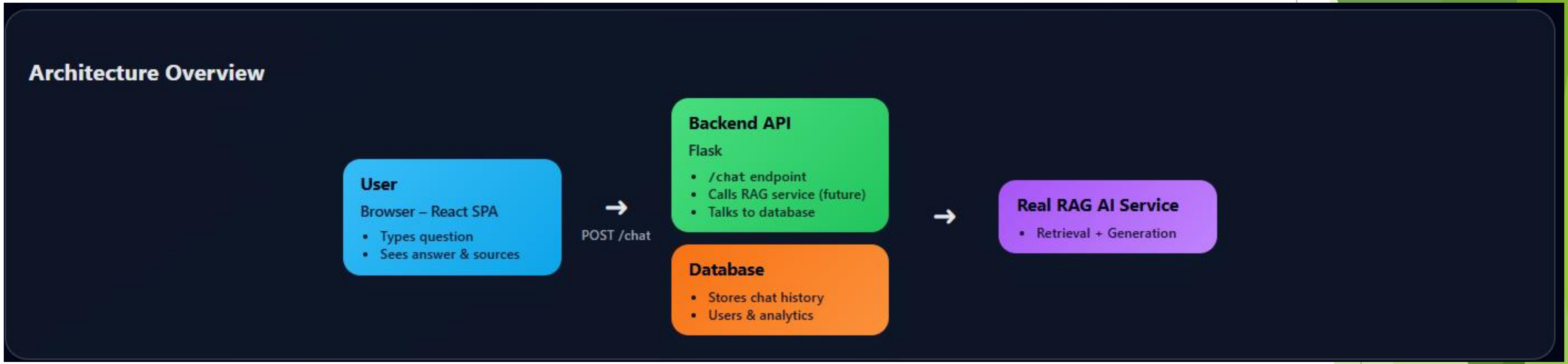
- Options for storing chat history:
- **LocalStorage,Cookies,SessionStorage:**
    - **WOULD NOT RECCOMEND,** maybe for testing
    - Not shared between devices
- **Redis cache:**
    - Very fast
    - Good for ephemeral session data
    - Not ideal for long-term analytics
    - Would reccomend for caching
- **Postgres (or similar DB):**
    - Persistent
    - Good for analytics, audit, per-user history
    - Slightly more complex schema & operations
    - Would reccomend – MySQL, MariaDB, ORACLE DB, POSTGRE SQL

Testing Approach

- Backend tests:
- Unit test for mocked RAG function:
  - Given question → returns deterministic answer + sources
- Integration test for /chat endpoint:
  - POST with a sample question → expect JSON with answer and sources
  - Explicit tests for "hello" and "thank u" flows

- Frontend tests (optional if by time):
- Component test:
  - Typing text and clicking Send adds user message
  - Mocked fetch → adds assistant message
- Snapshot/UI tests for chat bubbles

High-Level Architecture



Deployment & Scalability

- Production deployment idea:
- Package Flask app as Docker image
- Run in container platform (e.g. Kubernetes or simple VM)
- Frontend built as static bundle (React) and served via Nginx
- Most likely bottleneck:
    - Concurrent HTTP requests to the API
- Use:
    - Horizontal scaling (multiple replicas behind a load balancer)
    - Connection pooling to downstream services (RAG / DB)
- Real RAG latency:
    - Timeouts, retries, circuit breaker pattern at backend layer

Live Demo

Thank you for your attention