CSAI 422 - APPLIED GENERATIVE AI

Synthesis Talk Project

Mohamed Ayman 202201208
Hana Ayman 202101348
Mariam ElSherbini 202202568

Table of Contents

1. Project Overview & Objectives

What is SynthesisTalk?

SynthesisTalk is a comprehensive AI-powered research assistant that transforms how users interact with information through:

- 🤖 **Intelligent Conversation:** Multi-turn dialogues with context awareness
- 📄 **Document Analysis:** PDF processing with AI-powered summarization
- 🔍 **Web Search Integration:** Real-time search with intelligent synthesis
- 📝 **Smart Note-Taking:** Organized research insights with export capabilities
- 🧠 **Advanced Reasoning:** Chain-of-Thought and ReAct patterns

1. Project Overview & Objectives

Core Objectives

1. **Unified Research Platform:** Integrate multiple AI tools in one cohesive interface
2. **Advanced AI Reasoning:** Implement sophisticated reasoning patterns for better insights
3. **User-Centric Design:** Make complex AI accessible through intuitive interfaces
4. **Scalable Architecture:** Build extensible system supporting multiple LLM providers
5. **Educational Demonstration:** Showcase applied generative AI principles

2. System Architecture

-> Three-Tier Architecture

PRESENTATION LAYER: React.js + Vite + TailwindCSS
• Chat Interface • File Upload • Research Tools

APPLICATION LAYER : FastAPI Backend
• Tool Orchestration • Business Logic • API Endpoints

DATA LAYER: LLM Integration + Research Tools
• NGU/Groq Providers • Document Tools • Web Search

3. Front End

Modern React Architecture

Technology Stack:

- React 18: Latest hooks and functional components
- Vite: Lightning-fast build system and development server
- TailwindCSS: Utility-first CSS framework for responsive design
- Axios: Robust HTTP client with error handling

3. Front End

Key Frontend Features

📱 Responsive Design

- Mobile-first approach with TailwindCSS breakpoints
- Adaptive layouts for desktop, tablet, and mobile devices
- Consistent user experience across all screen sizes

💬 Real-time Chat Interface

- Instant message rendering with optimistic updates
- Conversation history with auto-scroll functionality
- Loading states and typing indicators for better UX

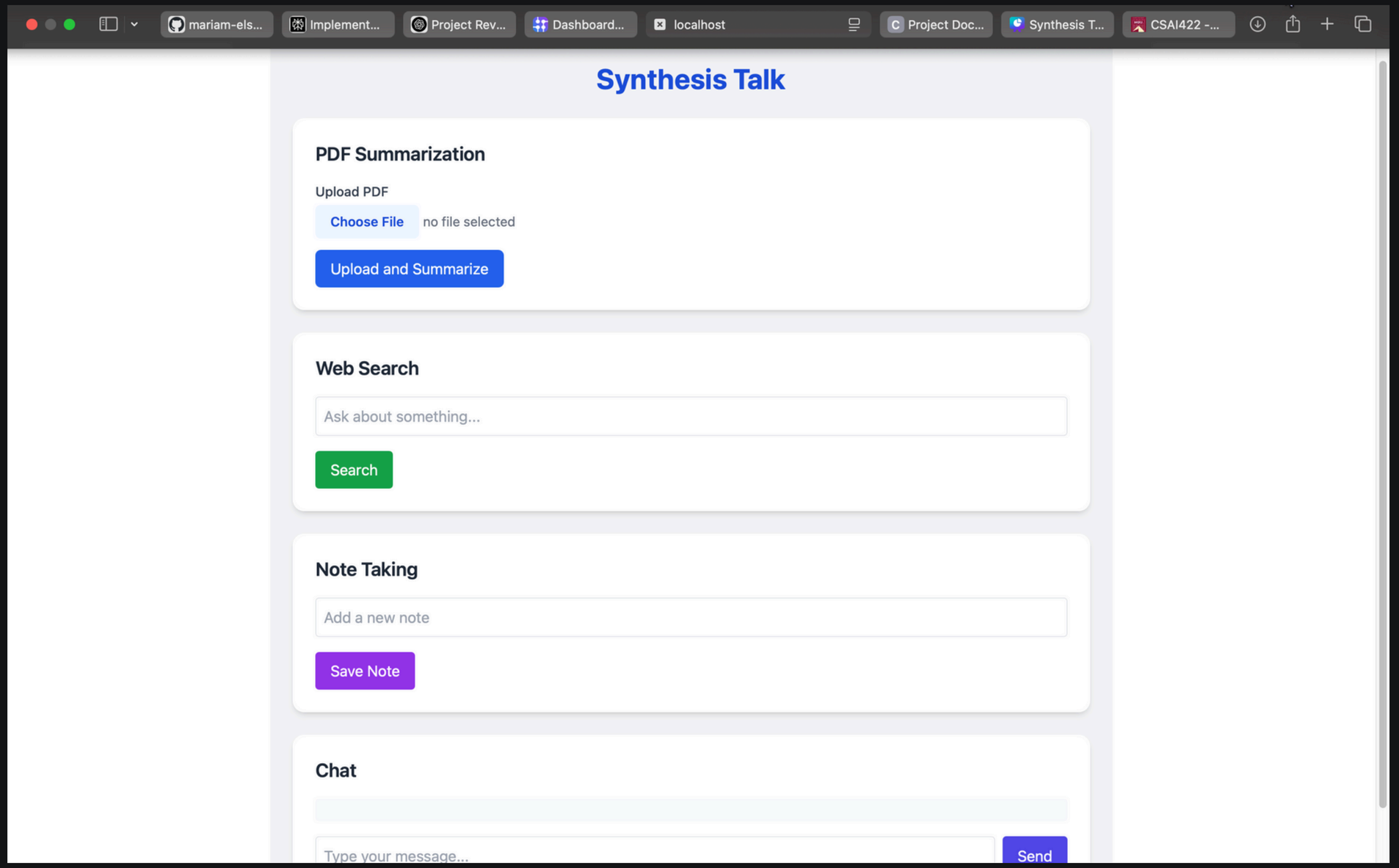
📄 File Upload System

- Drag-and-drop PDF upload with visual feedback
- File validation and error handling
- Progress indicators during processing

🎨 Modern UI Components

- Component-based architecture with reusable elements
- Consistent design system with TailwindCSS utilities
- Accessibility features and keyboard navigation support

3. Front End


mariam-els...
Implement...
Project Rev...
Dashboard...
localhost
Project Doc...
Synthesis T...
CSAI422 -...
Synthesis Talk
PDF Summarization
Upload PDF
Choose File
no file selected
Upload and Summarize
Web Search
Ask about something...
Search
Note Taking
Add a new note
Save Note
Chat
Type your message...
Send

4. Backend Systems & API Design

FastAPI Backend Architecture

Why FastAPI?

- ⚡ High Performance: Async/await support for concurrent requests
- 📖 Auto Documentation: OpenAPI/Swagger documentation generation
- 🔒 Type Safety: Pydantic models for request/response validation
- 🐍 Modern Python: Leverages latest Python features and best practices

5. Large Language Model Integration

Supported Providers

🚀 NGU (NguLlama)
- Local deployment capabilities
- Custom model support (qwen2.5-coder:7b)
- Optimized for development and testing

⚡ Groq
- High-performance inference
- Production-ready scaling
- Multiple model options (llama-3.3-70b-versatile)

🔄 OpenAI Compatible
- Standard API compatibility
- Easy integration with other providers
- Future-proof architecture

5. LLM Research Tools

1.**Document Analysis Tool**
1.Extracts content from PDFs/text files
2.Provides summaries, key points, detailed analysis

2.Web Search Tool
1.Searches for additional information
2.Supplements the LLM's knowledge with current data

3.Note-Taking Tool
1.Saves important insights automatically
2.Organizes findings with tags and timestamps

4.Explanation Tool
1.Provides multi-level explanations (basic, intermediate, advanced)
2.Adapts complexity to user needs

6. Development Insights

- This project challenged us to combine frontend engineering, backend integration, and LLM workflows into a cohesive research assistant platform.

- We gained valuable experience in bridging user interaction with complex AI reasoning pipelines, maintaining modular design, and debugging real-world tool execution issues.

7. Overcoming Technical and UX Challenges

- **LLM Integration Complexity:**
  Managing tool routing and conversation context across multiple API calls.

- **Frontend Responsiveness:**
  Designing a clean and intuitive chat UI with real-time feedback using React state.

- **Document Upload Handling:**
  Ensuring smooth PDF/text upload and feedback, with error handling and filename previews.

- **State Management:**
  Preserving context and rendering dynamic insights without breaking React’s structure.

8. Key Learnings from AI System Construction

- Learned to design multi-step **LLM workflows** (e.g., Chain-of-Thought reasoning)
- Understood how to structure APIs that trigger tools like summarization, search, and visualization
- Gained experience in building a **modular React frontend** connected to AI APIs
- Improved teamwork, Git collaboration, and communication in a cross-functional group

9. Future Directions & Resources

As SynthesisTalk evolves beyond this course, several improvements can be made to enhance its real-world performance, user experience, and long-term scalability.

Performance, Scalability, and Usability Enhancements

- **LLM Provider Optimization**
Use a more cost-effective or fine-tuned model for better speed and output quality.

- **Tool Parallelization**
Execute multiple tools (e.g., summarizer + search) in parallel to reduce response time.

- **UI/UX Improvements**
Add multi-topic research threads, dark mode, and improved file management.

- **Export & Collaboration Features**
Enable download as PDF/Word, and support multiple users sharing insights.

- **Scalability**
Containerize the backend and deploy on cloud platforms (e.g., Docker + AWS/GCP).

Project References

- React.js (Frontend Framework)
-
- Talwind CSS (Styling)
- FastAPI / Flask (Backend APIs)
- Groq / Gemini (LLM Providers)
- Recharts (Visualization Library)
- LangChain (LLM tool routing & chaining)
- GitHub (Version Control & Collaboration)

Thank You