YouTube Chat Bot with AI Agents and RAG - Project Roadmap

Project Overview

This project aims to build an interactive chatbot that can discuss YouTube video content with users. The application will extract transcripts from YouTube videos, process them using RAG (Retrieval Augmented Generation) techniques, and enable users to chat about specific sections or the entire video content.

Core Features

1. **Model Selection**: Users can select different LLM models and input their API keys
2. **YouTube Integration**: Extract transcripts from any YouTube video via URL
3. **RAG Implementation**: Generate and store embeddings of video content
4. **Chapter-Based Summary**: Auto-generate detailed video summaries with chapter breakdowns
5. **Contextual Chat**: Chat about specific chapters or the entire video
6. **Session-Based Memory**: Maintain chat history within the current session
7. **Document Upload**: Allow additional context through document uploads

Phase 1: Project Setup and Basic Structure (Weeks 1-2)

Week 1: Environment Setup and Project Architecture

- Set up the development environment with Python, virtual environment
- Initialize Git repository and project structure
- Design the application architecture (frontend, backend, database)

- Create basic UI wireframes
- Set up the project with FastAPI for backend and Streamlit for frontend

Week 2: Basic YouTube API Integration

- Implement YouTube API integration for video information retrieval
- Build transcript extraction functionality using youtube-transcript-api
- Create a basic data storage mechanism for transcripts
- Implement error handling for YouTube API requests
- Set up basic logging and monitoring

Phase 2: RAG Implementation and Core Functionality (Weeks 3-4)

Week 3: Embedding and Vector Storage

- Implement text chunking for transcript processing
- Set up embedding generation using LangChain with OpenAI or other embedding models
- Configure vector database (Chroma, FAISS, or Pinecone)
- Build embedding storage and retrieval mechanisms
- Create efficient indexing for quick content retrieval

Week 4: LLM Integration and Chapter Generation

- Implement LLM provider integration (OpenAI, Anthropic, etc.)
- Create model selection UI and API key management
- Build chapter generation algorithm using transcript timestamps and content analysis
- Develop summary generation for each chapter
- Implement caching mechanisms for API responses

Phase 3: Chat Interface and User Experience (Weeks 5-6)

Week 5: Basic Chat Functionality

- Design and implement the chat interface
- Build basic chat functionality with LangChain
- Create chat history storage mechanism
- Implement RAG-based question answering
- Add context management for chat sessions

Week 6: Advanced Chat Features

- Implement chapter-specific chat functionality
- Add memory management for session persistence
- Create chat history visualization
- Build response streaming for better UX
- Implement citation of video timestamps in responses

Phase 4: Document Upload and Additional Features (Weeks 7-8)

Week 7: Document Processing

- Implement document upload functionality
- Build document parsing for various formats (PDF, TXT, DOCX)
- Create document embedding and storage mechanism
- Implement context merging between video and document content
- Add document metadata management

Week 8: Advanced Features and Optimizations

- Implement multi-video support

- Add user preferences and settings
- Create export functionality for chat history
- Optimize token usage and API costs
- Improve error handling and edge cases

Phase 5: Testing, Deployment, and Documentation (Weeks 9-10)

Week 9: Testing and Quality Assurance

- Implement unit tests for key components
- Conduct integration testing
- Perform user acceptance testing
- Security review and optimization
- Performance optimization

Week 10: Deployment and Documentation

- Set up deployment pipeline
- Create Docker containers for easy deployment
- Write comprehensive documentation
- Create user guides and tutorials
- Final review and launch preparation

APIs and Technologies

LLM Providers:

- **OpenAI API**: For GPT models (GPT-3.5-Turbo, GPT-4)
- **Anthropic API**: For Claude models
- **Cohere API**: For Command models
- **Mistral AI API**: For Mistral models

- **Llama API**: For Meta's Llama models

Vector Databases:

- **Chroma**: Lightweight, easy to set up
- **FAISS**: Facebook AI Similarity Search, efficient for larger datasets
- **Pinecone**: Managed vector database service
- **Qdrant**: Vector search engine
- **Weaviate**: Vector search engine with GraphQL interface

YouTube APIs:

- **YouTube Data API v3**: To fetch video metadata
- **youtube-transcript-api**: To extract video transcripts

Frontend:

- **Streamlit**: For rapid UI development
- **Gradio**: Alternative for interactive UI components
- **React**: For more complex UI requirements (optional)

Backend:

- **FastAPI**: Modern, high-performance web framework
- **LangChain**: Framework for LLM application development
- **LlamaIndex**: For document retrieval and indexing

Document Processing:

- **PyPDF2/PyMuPDF**: For PDF processing
- **python-docx**: For DOCX file processing
- **Unstructured**: For handling various document formats

Database:

- **SQLite**: For lightweight development

- **PostgreSQL**: For production deployment

Implementation Challenges and Considerations

1. **Token Management**: Efficiently manage token usage to reduce API costs
2. **Chunking Strategy**: Develop optimal chunking strategy for transcript processing
3. **Context Limitation**: Handle context window limitations of LLMs
4. **API Rate Limits**: Implement proper handling of API rate limits
5. **Memory Management**: Efficiently manage session memory
6. **Security**: Secure storage of API keys and user data
7. **Scalability**: Design the system to handle multiple concurrent users
8. **User Experience**: Balance responsiveness with quality of responses

Project Extension Options

1. **Multi-modal Support**: Add image/video frame analysis
2. **User Authentication**: Add user accounts and persistent history
3. **Custom Training**: Fine-tune models on specific video content
4. **Analytics Dashboard**: Track usage and performance metrics
5. **Collaborative Features**: Allow multiple users to chat about the same video
6. **Plugin System**: Enable extensibility through plugins
7. **Voice Interface**: Add speech-to-text and text-to-speech capabilities
8. **Mobile App**: Develop companion mobile application