YouTube Summarizer Project Report

Prepared by: Likhita Yerra , Remi Allam

Date: April 16, 2025

A Comprehensive Analysis of the YouTube Summarizer Application Built with FastAPI and MongoDB

Contents

1 Project Overview

The YouTube Summarizer is a web application developed using FastAPI, designed to generate concise summaries of YouTube videos by leveraging their transcripts and OpenAI's language models. Additionally, it supports note-taking and bookmarking functionalities, providing a robust tool for managing video-related information. The system uses MongoDB for persistent storage and includes a mock database for testing, ensuring scalability and efficient asynchronous operations.

1.1 Key Features

- **YouTube Video Summarization**: Extracts transcripts using the `youtube-transcript-api` and generates summaries with OpenAI's GPT-3.5-turbo model.
- **Bookmark Management**: Enables creation, retrieval, and deletion of bookmarks with titles, URLs, descriptions, and tags.
- **MongoDB Integration**: Stores data using ODMantic for ORM, with a mock database fallback for testing.
- **RESTful API**: Provides endpoints for CRUD operations on summaries, bookmarks, and notes.
- **CORS Support**: Facilitates cross-origin requests for frontend integration.

2 System Architecture

The application adopts a modular architecture with clear separation of concerns, ensuring maintainability and scalability.

2.1 Backend Components

- **FastAPI Application** (`main.py`):
  - Initializes the FastAPI app with CORS middleware.
  - Mounts static files for frontend content.
  - Manages MongoDB connections via lifespan events.
  - Includes health check and root redirection endpoints.
- **Database Layer** (`db.py`):
  - Uses `motor` for asynchronous MongoDB interactions and `odmantic` for ORM.
  - Implements a `MockEngine` for testing.
  - Provides `fix_mongo_ids` for JSON serialization.
- **Schema Definitions** (`schema.py`):
  - Defines MongoDB models (`Note`, `YouTubeSummary`, `Bookmark`).
  - Uses Pydantic for input validation.
  - Configures datetime serialization.
- **CRUD Operations** (`crud.py`):

  - Implements asynchronous functions for transcript fetching, OpenAI API requests, and summary/bookmark management.
  - Handles YouTube URL parsing and error handling.

- **API Routes** (`router.py`):
  - Defines RESTful endpoints with input validation and error responses.
  - Supports query parameters (e.g., tag filtering).

2.2 External Dependencies

- **YouTube Transcript API**: Fetches video transcripts.
- **OpenAI API**: Generates summaries using GPT-3.5-turbo.
- **MongoDB**: Provides persistent storage with a mock fallback.

3 Technical Details

3.1 Technologies Used

- Python 3.8+
- FastAPI: Asynchronous web framework.
- MongoDB: NoSQL database.
- ODMantic: MongoDB ORM.
- Motor: Asynchronous MongoDB driver.
- Pydantic: Data validation.
- httpx: Asynchronous HTTP client.
- youtube-transcript-api: Transcript extraction.
- python-dotenv: Environment variable management.

3.2 Dependencies

Key dependencies include:

- `fastapi==0.115.11`
- `odmantic==1.0.2`
- `motor==3.7.0`
- `pydantic==2.10.6`
- `httpx==0.28.1`
- `python-dotenv==1.0.1`

3.3 Data Models

- `Note`: Contains a `content` field.
- `YouTubeSummary`: Stores `url` and `summary`.
- `Bookmark`: Includes `title`, `url`, `description`, `tags`, and `created_at`.

3.4 API Endpoints

Method	**Endpoint**	**Description**
POST	`/notes/`	Create a new note
POST	`/youtube-summary/`	Create a YouTube video summary
GET	`/youtube-summaries/`	List all YouTube summaries
GET	`/youtube-summary/{id}`	Get a specific YouTube summary
DELETE	`/youtube-summaries/{id}`	Delete a YouTube summary
POST	`/bookmarks/`	Create a new bookmark
GET	`/bookmarks/`	List bookmarks, optionally filtered by tag
DELETE	`/bookmarks/{bookmark_id}`	Delete a bookmark

4 Implementation Highlights

4.1 Asynchronous Programming

- Uses `asyncio` and `motor` for non-blocking operations.
- Wraps synchronous `youtube-transcript-api` calls in a thread pool.

4.2 Error Handling

- Handles errors for MongoDB connections, OpenAI API requests, and transcript fetching.
- Falls back to a mock database if MongoDB fails.
- Returns HTTP exceptions (e.g., 404).

4.3 Security

- Uses environment variables for sensitive data.
- Implements CORS middleware.

4.4 Testing Support

- `MockEngine` simulates CRUD operations for testing.
- Simplifies development and CI/CD pipelines.

5 Limitations

- **Transcript Availability**: Depends on YouTube video transcripts.
- **OpenAI Dependency**: Requires API key and incurs costs.
- **Mock Database**: Lacks persistence and advanced queries.

- **Video ID Parsing**: Limited to common YouTube URL formats.
- **No Authentication**: Unsuitable for multi-user scenarios.

6 Potential Improvements

- Add JWT or OAuth2 for authentication.
- Support multiple summary lengths or custom prompts.
- Implement caching for transcripts and summaries.
- Develop a React-based frontend.
- Add rate limiting to API endpoints.
- Enhance URL parsing for broader compatibility.
- Add MongoDB indexes for faster queries.

7 Conclusion

The YouTube Summarizer is a robust, extensible application for generating video summaries and managing bookmarks and notes. Its asynchronous architecture, modular design, and error handling make it suitable for small to medium-scale deployments. With enhancements like authentication and caching, it could become a production-ready tool for content creators, researchers, and students.