skill.io

TECHNICAL DOCUMENTATION

Dependencies

- Flask
- requests
- os
- json
- deep_translator
- werkzeug
- PyPDF2
- chromadb
- time
- moviepy

Flow Diagrams

1. Video Translation Process

```
A[User] --> B[Upload YouTube URL]
B --> C[video_translation route]
C --> D[Clear Folders]
D --> E[Download YouTube Video]
E --> F[Extract Audio from Video]
F --> G[Transcribe Audio]
G --> H[Translate Text to Language selected]
H --> I[Generate New Audio]
I --> J[Save Transcriptions to TXT]
J --> K[Overlay Audio on Video]
K --> L[Return Translated Text and Video]
```

2. PDF Upload and Embedding Process

```
A[User] --> B[Upload PDF]
B --> C[upload_pdf route]
C --> D[Save PDF to Upload Folder]
D --> E[Extract Text from PDF]
E --> F[Create and Store Embeddings]
F --> G[Store Embeddings in ChromaDB]
G --> H[Return Upload Confirmation and Extracted Text]
```

3. Quiz Feature

```
A[User] --> B[Access Quiz Page]
B --> C[start_quiz route]
C --> D[Initialize Quiz Session]
D --> E[Return First Question]
E --> F[Submit Answer]
F --> G[quiz route]
G --> H[Check Answer]
H --> I[Update Score and Question Index]
I --> J[Return Next Question or Quiz Completion Status]
```

Code Description

1. **routes.py**

   The `routes.py` file defines the main routes and functionality of the Flask web application. It includes routes for the home page, translation, file upload, quiz, and analysis. The file also handles the core logic for uploading PDF files, extracting text, creating embeddings, handling quiz interactions, and managing video translation processes. Key routes and functionalities are:

   - **`index` Route:** Renders the homepage with subjects for recommendations.
   - **`translation` Route:** Renders the translation page.
   - **`upload_pdf` Route:** Handles PDF file uploads, extracts text from the PDF, creates embeddings, and stores them.
   - **`get_response` Route:** Translates user input, searches for relevant documents, and interacts with an AI API to generate responses.

- **`video_translation` Route:** Manages the process of translating YouTube videos by downloading the video, extracting audio, transcribing it, translating the transcription, generating new audio, and overlaying it on the video.
- **Quiz Routes:** Manages the quiz functionality, including starting the quiz, fetching questions, and displaying results.
- **`analysis` Route:** Provides recommendations based on selected subjects and skill levels.

2. audio_translation.py

The `audio_translation.py` file contains functions for processing YouTube videos, extracting and transcribing audio, translating text, generating new audio, and overlaying audio onto videos. Key functions are:

- **`download_youtube_video` Function:** Downloads a YouTube video using the `yt_dlp` library.
- **`extract_audio_from_video` Function:** Extracts audio from a video file using `moviepy`.
- **`transcribe_audio` Function:** Transcribes audio to text using Google's speech recognition.
- **`translate_text_to_tamil` Function:** Translates text to the specified language using `GoogleTranslator`.
- **`generate_new_audio` Function:** Generates audio from text using `gTTS` (Google Text-to-Speech).
- **`overlay_audio_on_video` Function:** Overlays new audio onto the video using `moviepy`.
- **`save_transcriptions_to_txt` Function:** Saves the original and translated transcriptions to a text file.
- **`clear_folders` Function:** Clears specified folders to manage temporary files.

Libraries and Technical Components

Flask and Jinja2

- **Flask:** A lightweight web framework in Python, used to create the web application. Flask is easy to set up and allows for rapid development with its simple and modular design.

- **Jinja2:** A templating engine for Python, used with Flask to render HTML templates. It allows embedding Python expressions in HTML, making it easier to create dynamic web pages.

Libraries for File Handling and Web Requests

- **os:** A standard Python library for interacting with the operating system. Used here for path manipulations, directory creation, and file handling.
- **requests:** A popular Python library for making HTTP requests. It is used for calling external APIs to fetch embeddings and chat completions.
- **werkzeug.utils.secure_filename:** A utility from the Werkzeug library, part of the Flask framework, used to secure filenames uploaded by users to prevent directory traversal attacks.

PDF Handling

- **PyPDF2:** A Python library for reading and manipulating PDF files. It is used to extract text from PDF documents.

Translation and Audio Processing

- **deep_translator.GoogleTranslator:** A library for translating text using various translation services. Here, it's used for translating text to and from different languages.
- **gtts:** The Google Text-to-Speech library, used to convert translated text into audio files.
- **moviepy:** A Python library for video editing. It is used for extracting audio from videos and overlaying new audio onto videos.
- **speech_recognition:** A library for performing speech recognition, used here to transcribe audio files into text.

Video and Audio Downloading

- **yt_dlp:** A command-line program to download videos from YouTube and other sites. It is an actively maintained fork of youtube-dl, with additional features and support for newer sites.
- **youtube_dl:** Another library for downloading YouTube videos, which can be used interchangeably with yt_dlp.

Data Management and Embeddings

- **chromadb:** A library used for managing embeddings and performing similarity searches. It allows the application to store and query document embeddings efficiently.
- **textwrap:** A standard Python library used to format text output by wrapping it to a specified width. Useful for generating readable text files.

Flask Session Management

- **session:** Part of Flask used to manage session data, which allows storing user-specific information like quiz progress and scores across multiple requests.

Key Technical Concepts

- **Routing:** Flask's routing system is used to define various endpoints (routes) of the web application. Each route is associated with a function that handles requests and returns responses.
- **File Uploads:** The application handles file uploads (PDFs) securely by checking for allowed file extensions and saving files to designated directories.
- **Text Extraction and Embeddings:** Extracted text from PDFs is processed and converted into embeddings, which are stored for later use in querying and generating responses.
- **Speech Recognition and Text-to-Speech:** Audio from videos is transcribed to text, translated, and converted back into audio in the desired language, allowing for video translations.
- **APIs:** The application interacts with external APIs (like fireworks.ai) to fetch embeddings and generate responses.

Fireworks API

The Fireworks API is an external service used to generate and manage embeddings and to facilitate chat completions. Here's a brief explanation of its functionalities and how it integrates into the application:

Key Functionalities

1. **Embeddings Generation**:
    - **Purpose**: To convert text data into a numerical format that can be efficiently stored and queried.

    - **Usage in Application**: When a PDF is uploaded, the extracted text is sent to the Fireworks API, which returns embeddings. These embeddings are then stored in the application's database for later use in querying and generating responses.
2. **Chat Completions**:
    - **Purpose**: To generate text responses based on user input and context.
    - **Usage in Application**: When a user interacts with the chatbot, the input message and relevant context (previous messages, embeddings) are sent to the Fireworks API. The API processes this data and returns a text completion that the chatbot uses to respond to the user.

Logic of Implementation

1. Uploading and Processing PDFs

- **File Upload Handling**: Users upload PDF files through the `/upload_pdf` route. The files are saved to the server, and their text content is extracted using PyPDF2.
- **Embedding Generation**: The extracted text is sent to the Fireworks API to generate embeddings. These embeddings are stored locally and in the ChromaDB collection for later use.

2. Handling YouTube Video Translation

- **Video Processing Workflow**: When a user submits a YouTube URL for translation via the `/video_translation` route:
    - **Download Video**: The video is downloaded using `yt_dlp`.
    - **Extract Audio**: Audio is extracted from the video using `moviepy`.
    - **Transcribe Audio**: The audio is transcribed to text using `speech_recognition`.
    - **Translate Text**: The transcribed text is translated to the selected language using `GoogleTranslator`.
    - **Generate New Audio**: The translated text is converted to audio using `gTTS`.
    - **Overlay Audio on Video**: The new audio is overlaid onto the video using `moviepy`.
    - **Cleanup**: Temporary files and folders are cleared.

3. Chatbot Functionality

- **User Interaction**: Users interact with the chatbot via the `/get_response` route.

- **Translation**: User messages are translated to English (or the target language) using `GoogleTranslator`.
- **Context Handling**: Relevant embeddings are retrieved based on the translated input and included in the context for the Fireworks API request.
- **Generating Response**: The Fireworks API generates a response based on the context and user input, which is then translated back to the user's language and returned.

4. Quiz Functionality

- **Starting a Quiz**: The quiz is initiated through the `/quiz/start_quiz` route, where the first question is sent to the user.
- **Handling Quiz Answers**: User answers are processed via the `/quiz/quiz` route, updating the session state and returning the next question or the final score.