Documentation for `app.py` and `templates/index.html`

`app.py` Documentation

The `app.py` file is a Flask-based web application that implements a multimodal agent capable of processing text, image, and voice inputs. It leverages machine learning models, embeddings, and external APIs to provide answers related to scientific discoveries and other topics.

Overview

- **Purpose**: Handles backend logic for a web-based multimodal agent.
- **Framework**: Flask.
- **Key Features**: Processes text, image, and voice inputs; integrates face recognition, OCR, and speech transcription; generates context-aware answers.

Key Components

MultimodalAgent Class

- **Purpose**: Core logic for handling multimodal inputs and generating responses.
- **Initialization**: Sets up the language model (LLaMA-3.2-1B-Instruct), embeddings (SentenceTransformer), and face recognition tools.
- **Methods**:
  - `preprocess_image` : Converts uploaded images into a format suitable for face recognition.
  - `find_relevant_context` : Retrieves relevant context using embeddings or falls back to Wikipedia if similarity is low.
  - `generate_answer` : Uses the LLaMA model with reasoning prompts (e.g., Chain of Thought) to produce detailed answers.
  - `refine_answer` : Enhances short answers by regenerating them with additional context.
  - `process_text_input` : Processes text queries by retrieving context and generating answers.
  - `recognize_face` : Identifies known individuals in images using the `face_recognition` library.
  - `process_image_input` : Handles image inputs by either recognizing faces or extracting text via OCR (using `pytesseract`).
  - `process_voice_input` : Transcribes voice inputs (via `speech_recognition`) and processes the resulting text.

Flask Routes

- `/` : Renders the main page (`index.html`).
- `/upload_text` : Accepts POST requests with text input, processes it, and returns a JSON response with the answer.
- `/upload_image` : Accepts POST requests with an image file and optional question, processes the image, and returns a JSON response.
- `/upload_voice` : Accepts POST requests with an audio file, transcribes it, processes the text, and returns a JSON response.

External Dependencies

- **SentenceTransformer**: Generates embeddings for context retrieval.
- **LLaMA-3.2-1B-Instruct**: Language model for answer generation.
- **face_recognition**: Library for facial recognition in images.
- **pytesseract**: Optical Character Recognition (OCR) for text extraction from images.
- **speech_recognition**: Converts audio inputs to text.
- **Wikipedia API**: Fallback source for context when local data is insufficient.

Workflow

1. **Input Processing**:
   - Text: Directly processed via `process_text_input` .
   - Image: Analyzed for faces or text content via `process_image_input` .
   - Voice: Transcribed to text via `process_voice_input` .
2. **Context Retrieval**: Uses embeddings to find relevant data or queries Wikipedia.
3. **Answer Generation**: LLaMA model generates responses, refined if necessary.
4. **Response**: Results are returned as JSON to the frontend.

`templates/index.html` Documentation

The `index.html` file is the frontend interface for the multimodal agent, allowing users to submit text, image, or voice queries via a web browser. It combines HTML, CSS, and JavaScript for a responsive and interactive experience.

Overview

- **Purpose**: Provides a user interface for interacting with the multimodal agent.
- **Technologies**: HTML5, CSS3, JavaScript (with `MediaRecorder` and `fetch` APIs).

Key Components

JavaScript Functionality

- **Voice Recording**:
  - Uses `MediaRecorder` API to capture audio from the microphone.
  - **Start Recording**: Begins recording, disables start button, enables stop button, updates status.
  - **Stop Recording**: Stops recording, sends audio Blob to `/upload_voice`, displays result.
- **Text Submission**:
  - Sends text input to `/upload_text` via `fetch` with JSON payload.
  - Updates `#textResult` with the response.
- **Image Submission**:
  - Sends image file and optional question to `/upload_image` via `fetch` with `FormData`.
  - Updates `#imageResult` with the response.

Workflow

1. **User Interaction**: Users input text, upload images, or record voice queries.
2. **Data Submission**: Inputs are sent to the Flask backend via POST requests.
3. **Result Display**: Responses are shown in designated areas with status updates for voice inputs.

Summary

- `app.py`:
  - A Flask-based backend that powers a multimodal agent with text, image, and voice processing capabilities.
  - Integrates advanced tools for context retrieval and answer generation.
- `templates/index.html`:
  - A clean, user-friendly web interface for submitting multimodal queries.
  - Handles client-side logic for input submission and result display.