API Documentation

API Documentation

Overview

This API is built using Flask and Flask-SocketIO and is designed to handle image processing and frame-to-video conversion tasks. The API facilitates session-based communication and supports real-time processing via WebSocket connections.

Table of Contents

Configuration

The following configuration constants are used throughout the API:

- `GENERATE_SESSION_ID_ON_CONNECT`: Whether to generate session IDs on WebSocket connect.
- `MAX_BUFFER_SIZE`: Maximum buffer size for HTTP requests (500 MB).
- `SESSION_ID_NAME`: Query parameter name for the session ID.
- `EMIT_RESULT`: Whether to emit results to the "result" event.
- `SECRET_KEY`: Secret key for Flask sessions.
- `SESSIONS_DATA`: Dictionary to store session data.
- `ALLOWED_SESSION_IDS`: List of valid session IDs.

Session Management

Generate Session ID

Function: `generate_session_id`

Generates a unique session ID using UUID.

Validate Session ID

Function: `validate_session_id`

Validates if the given session ID is allowed.

Remove Session ID

Function: `remove_session_id`

Removes a session ID from the allowed session IDs list.

Remove Session Data

Function: `remove_session_data`

Removes session data associated with a session ID.

Image and Video Processing

Convert Bytes to Image

Function: `bytes_to_image`

Converts byte data to a PIL `Image` object.

Process Image

Function: `process_image`

Processes an image and returns a dictionary containing the first pixel and image size.

Images to Video In-Memory

Function: `images_to_video_in_memory`

Converts a list of PIL images to a video stored in memory.

Group Session Frames

Function: `group_session_frames`

Groups frames by session ID and returns a video object.

Send File to Server

Function: `send_file_to_server`

Sends a file (video) from a byte stream to a specified server URL and returns the response data.

Process Video

Function: `process_video`

Processes the video data and sends it to specified APIs. This function is a placeholder for the actual video processing logic.

Append Frame to Session Data

Function: `append_frame_to_session_data`

Appends a frame to the session data.

HTTP Endpoints

Get Session ID

Endpoint: `/get_session_id`

Generates a session ID and returns it in JSON format.

Response:

```
{
  "session_id": "unique_session_id"
}
```

Batch File Processing

Endpoint: `/batch` (POST)

Processes a batch video file and sends it to multiple APIs.

Response:

```
{
  "result": {
    "goal_try": "...",
    "object_detection": "...",
    "pass_drive": "..."
  }
}
```

Options (not implemented yet)

Endpoint: `/options`

Returns the available processing options.

Response:

```
{
  "processes": ["WIP", "WIP"]
}
```

WebSocket Events

Connect

Event: `connect`

Handles the event when a client connects. Validates the session ID.

Disconnect

Event: `disconnect`

Handles the event when a client disconnects. Removes the session ID from the allowed list.

Test

Event: `test`

A simple test event to print the received data and session ID.

Process

Event: `process`

Handles the image processing event:

1. Validates the session ID.
2. Gets image bytes and validates.
3. Appends the frame to session data.
4. Groups relevant frames for the session and converts them to a video.
5. Processes the video and sends it to specified APIs.
6. Emits the result if `EMIT_RESULT` is enabled.

Utility Functions

A set of utility functions are provided to aid in session and data management:

- `generate_session_id`: Generates unique session IDs.
- `validate_session_id`: Validates if a session ID is allowed.
- `remove_session_id`: Removes a session ID from the allowed list.
- `remove_session_data`: Removes session data for a session ID.
- `bytes_to_image`: Converts byte data to a PIL `Image` object.
- `process_image`: Processes an image by extracting the first pixel and size.
- `images_to_video_in_memory`: Converts a list of images to a video.
- `group_session_frames`: Groups frames by session ID and returns a video.
- `send_file_to_server`: Sends a video file to a specified server URL.
- `append_frame_to_session_data`: Appends a frame to the session data.

Running the Server

Command:

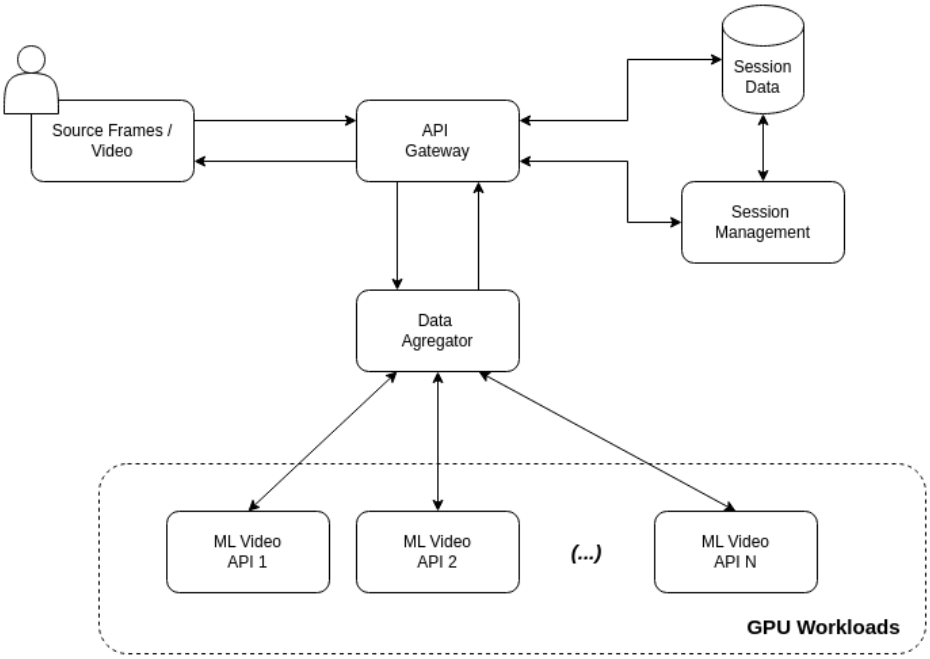
```
python app.py
```

This command runs the Flask application with Flask-SocketIO. Debug mode is disabled and `use_reloader` is set to `False`.

Additional Notes

- Ensure that the `SECRET_KEY` is set to a random and secure value.
- Error handling needs to be expanded for production use.

Architecture



Flow Diagram