**AI-based API Aware Image and Video Generation System**

This project provides a workflow for generating AI-driven API aware images and videos based on text prompts, storing the generated content and logs in databases, and serving the results through a web interface.

**Overview**

1. **Text-to-Image and Text-to-Video Generation:**  
   Using OpenAI’s DALL·E 3 and RunwayML’s video generation model, this system takes a user prompt and generates 5 images and 5 corresponding videos. The **image\_video\_generation.py** code uses user text prompt to generate image by using DALL-E-3 image generator, once image is generated it gets downloaded in to generated\_content/<user\_id> directory as image\_1.png so on. Later that image is converted into base64 format which along with text prompt together sent to RunwayML API to generates a 5 seconds video which is later downloaded into the same directory in the same file name as of the reference image (used as prompt earlier) say video\_1.mp4 format. However, more images and videos can be generated but it will incur too much cost. For example, in the **image\_video\_generation.py** code search for the function **generate\_images\_and\_videos**(user\_id, prompt, **n=5**, image\_size="1024x1024", video\_duration=5) and change value of n.
   * Images are stored in generated\_content/<user\_id>/image\_x.png.
   * Videos are stored in generated\_content/<user\_id>/video\_x.mp4.
2. **Storing and Managing Content:**  
   The generated content details (prompts, file paths, status, timestamps) are stored in a database (ai\_generation.db). Once generation completes, the status is updated to "Completed".
3. **User Access and Web Page Display:**  
   A Flask web application (web.py) allows users to access their generated images and videos by providing a user\_id. If the content is still processing, it displays a "Processing" page. Once completed, it shows a gallery of images and playable videos.

User actions (login attempts and content views) are logged in a separate database (user\_logs.db).

1. **Notifications:**  
   When the content generation is complete, the system prints a notification message to the terminal, including a link where the user can view their content.

**Directory Structure**

ai-generation-system/

├─ generated\_content/

│ └─ <user\_id>/

├─ templates/

│ ├─ gallery.html

│ └─ processing.html

├─ main.py

├─ web.py

├─ config.py

├─ database.py

├─ image\_video\_generation.py

├─ notifications.py

├─ requirements.txt

└─ .env

* generated\_content/: Holds generated images and videos for each user.
* templates/: Contains HTML templates for the gallery and processing pages.
* main.py: CLI script to trigger generation and send notification once done.
* web.py: Flask web application serving the generated content and logging user actions.
* database.py: Manages the ai\_generation.db for storing user content and metadata.
* config.py: Loads environment variables and configuration.
* image\_video\_generation.py: Contains functions for generating images and videos.
* notifications.py: Handles user notifications (in-terminal message).
* requirements.txt: Lists Python dependencies.
* .env: Contains API keys and other secrets (not checked into version control).

**Prerequisites**

* Python 3.11.2
* A .env file with the following variables (use your own key for ChatGPT and RunwayML):

OPENAI\_API\_KEY=<your\_openai\_api\_key>

RUNWAYML\_API\_SECRET=<your\_runwayml\_api\_key>

* The required Python packages:

pip install -r requirements.txt

* Export Runwayml api (on Linux/Mac, in windows it does not work) otherwise it does not work, instead of saving it in .env file as below. Use your own key.

export RUNWAYML\_API\_SECRET=key\_0521990e96f3f1f718abd6f2e73d96d8d56e198ff49e21f261d1c79a9c25e47a02d815601936facd5d7735ad60bd5d1f2f3c013a26025b2cdd9e3c8de125a0b6

**Usage**

1. **Content Generation:**  
   Run main.py to start the generation process:

python main.py <user\_id> "<prompt>"

For example:

python main.py user123 "A serene landscape with mountains and a river at sunset."

This will:

* + Insert a new record into ai\_generation.db with status "Processing".
  + Generate 5 images and 5 videos using the prompt.
  + Update the database record to "Completed" once done.
  + Print a terminal notification indicating that content is ready.

1. **Serving the Content:** Start the Flask server:

python web.py

Then access the content in your browser:

http://localhost:5000/?user\_id=user123

* + If still processing, you’ll see a "Processing" page.
  + Once completed, a gallery of images and videos will be displayed.

1. **Logs:**
   * The user\_logs.db database logs user interactions:
     + "Login" when a user visits /?user\_id=...
     + "View Content" when a user views an image or video file.

**Notes**

* Ensure that generated\_content/ directory is writable.
* The code currently uses eval() to parse lists of file paths stored as text in the database. For a production environment, prefer json.loads() for safer parsing.
* The RunwayML and OpenAI generation may take some time. The processing page remains until the main.py script completes generation.
* The notification is currently in-terminal only. You can extend notifications.py for email or other services if desired.

**License**

This project is provided as-is for demonstration purposes. Adjust and extend it to meet your own requirements.