Image Generation & Payment App Documentation

May 17, 2025

Project Overview

Defining the structure and purpose of this project, outlining a web application enabling users to generate and download images via the Unsplash API. Incorporating user authentication, search history management, and a premium payment system integrated with Stripe. Free users are limited to a set number of image generations, while paid users gain unlimited access.

Features

- **User Authentication**: Signup, login, and logout functionalities using an SQLite database.
- **Image Search & Download**: Users can search and download images from Unsplash using keywords.
- **Search History**: Each user's search records are saved in a database and displayed on the dashboard.
- **Payment Integration (Stripe)**: Free users are limited to a set number of images; a premium subscription unlocks unlimited generations.
- **Session Management**: Streamlit session state maintains the user's login status.

Technologies Used

- **Python**: Core programming language.
- **Streamlit**: Web UI framework for rapid prototyping.
- **SQLite**: Local database for user and history management.
- **Stripe API**: Payment gateway integration.
- **Unsplash API**: Image search and download functionality.
- **dotenv**: For managing environment variables.

Environment Setup

Creating a .env file in the root directory and adding Stripe keys:

```
SECRET_KEY=your_stripe_secret_key
PUBLISH_KEY=your_stripe_publishable_key
```

The Unsplash API key is hardcoded as API$_{K}EYinthescript.Replaceitwithyourownifneeded.$

Project Structure Overview

- **AuthSystem class**: Handles user signup, login, logout, and premium status management.
- **HistorySystem class**: Manages search history recordsadding, deleting, and retrieving.
- **ImageDownloader class**: Communicates with the Unsplash API to search and download images.
- **Stripe Checkout function**: Creates a Stripe payment session for premium access.
- **Streamlit UI**: Controls the frontend interface, including login/signup, image search, payment upgrade, and history display.

Usage Instructions

1. User Authentication

- **Signup**: Enter a unique username and password, then click Signup.
- **Login**: Enter your credentials and click Login.
- **Logout**: Click the Logout button on the dashboard.

2. Image Search & Download

Once logged in, type a keyword to search for images. Click "Generate Image" to fetch an image from Unsplash. The Download button allows saving the image locally. Free users can generate up to 5 images. Paid users have unlimited generation.

3. Payment Upgrade

Upon reaching the free image limit, an option to upgrade to Premium appears. Clicking "Upgrade to Premium" redirects to the Stripe payment page. Successful payment unlocks unlimited access.

4. Search History

Previous search queries and timestamps are listed. Each record can be deleted individually.

Database Schema

users Table

Column	Type	Description
id	INTEGER	Primary Key (auto increment)
username	TEXT	Unique username
password	TEXT	User password
paid	INTEGER	Premium status (0 or 1)

history Table

Column	Type	Description
id	INTEGER	Primary Key (auto increment)
username	TEXT	Username who generated the image
image$_{q}uery$	TEXT	Keyword used for image search
generated$_{a}t$	TEXT	Timestamp of image generation

Important Functions & Classes

AuthSystem

- `signup(username, password)`: Registers a new user.
- `login(username, password)`: Validates login credentials.
- `logout()`: Logs out the current user.

- `mark`$_{p}aid(username) : Markstheuseraspremium.$

HistorySystem

- `add`$_{r}ecord(username, query) : Savesanewsearchrecord.$

ImageDownloader

- `download`$_{i}mages(query) : DownloadsanimagefromtheUnsplashAPI.$

create$_{c}heckout_{s}ession()$

Creates a Stripe checkout session URL.

How to Run

1. Install dependencies:

```
pip install streamlit stripe python-dotenv requests
```

2. Set environment variables in the .env file. 3. Run the app:

```
streamlit run app.py
```

4. Access the app via browser at http://localhost:8501.

Notes & Future Enhancements

- Passwords are stored in plaintext for simplicity; consider hashing for production.
- Stripe URLs and keys are configured for localhost; update URLs for deployment.
- Add more detailed error handling for API requests and database operations.
- Improve UI/UX for a better user experience.
- Add pagination or advanced search filters for images.