Post Genius - Project Documentation

# Introduction

Post Genius is a Django-based AI-powered social media automation platform. It helps users generate brand-friendly captions and images using OpenAI's GPT-4 and DALL·E models, and post them directly to Facebook and Instagram through secure OAuth integration.

# Technology Stack

* **Backend:** Django (Python) — Robust and scalable web framework for rapid development.
* **Frontend:** HTML, CSS, JavaScript — Responsive and interactive user interfaces.
* **AI Integration:**
  + **GPT-4** by OpenAI — Generates engaging, brand-friendly captions.
  + **DALL·E** by OpenAI — Creates stunning visuals from text prompts.
* **Authentication & Social Posting:** Facebook OAuth via the Facebook Graph API — Secure login and social media integration.
* **Database:** SQLite for development (easily upgradable to PostgreSQL or MySQL) — Managed via Django ORM.
* **Hosting:** Runs locally (localhost) during development; compatible with platforms like **Render**, **Railway**, or **Heroku** for deployment.

# Features & Functionality

* **User Authentication**  
  Secure registration, login, and logout system using Django's built-in authentication framework.
* **User Profiles**  
  Users can view and update their profile details, including avatar and bio.
* **AI-Powered Caption Generation**  
  Generate social media captions based on user input using OpenAI GPT-4.
* **AI Image Generation**  
  Create images from text prompts using OpenAI DALL·E.
* **Engagement Prediction**  
  Predict potential likes and shares of captions using sentiment analysis via TextBlob.
* **Facebook and Instagram Integration**  
  OAuth login with Facebook Graph API, including long-lived token storage.
* **Direct Social Media Posting**  
  Post generated captions and images directly to Facebook Pages or Instagram business accounts.
* **Data Export**  
  Export generated captions and image metadata as downloadable CSV files.
* **User Dashboard**  
  A centralized dashboard for viewing and managing all previously generated content.

# Setup Instructions

1. Clone the project and install dependencies:  
    pip install -r requirements.txt
2. Create a .env file with the following keys:
   1. OPENAI\_API\_KEY=your\_openai\_key
   2. FB\_APP\_ID=your\_facebook\_app\_id
   3. FB\_APP\_SECRET=your\_facebook\_app\_secret
   4. FB\_REDIRECT\_URI=http://localhost:8000/facebook/callback/
3. Run migrations:  
    python manage.py makemigrations && python manage.py migrate
4. Run the server:  
    python manage.py runserver

## How to Get Your OpenAI API Key

**1. Create/Open Your OpenAI Account**

* Go to: <https://platform.openai.com/signup>
* If you already have an account, log in here: <https://platform.openai.com/login>

**2. Access the API Keys Dashboard**

* Once logged in, go to the **API Keys** section:  
  <https://platform.openai.com/account/api-keys>

**3. Generate a New API Key**

* Click on the **“+ Create new secret key”** button.
* Give it a name (optional, e.g., PostGenius App) and click **Create secret key**.
* **Copy the key immediately** — it will look like this:  
  sk-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

**4. Add the Key to Your Project's .env File**

* Create or open your .env file and add the key:
* OPENAI\_API\_KEY=sk-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

## How to Get Your Facebook App Credentials (for OAuth)

**Step 1: Create a Facebook Developer Account**

1. Go to <https://developers.facebook.com/>
2. Log in using your personal Facebook account.
3. Click **“Get Started”** (if you haven’t already registered as a developer).

**Step 2: Create a New App**

1. From the top menu, click on **“My Apps”** → **“Create App”**.
2. Choose the app type:  
   Select **"Business"** or **"Consumer"**, depending on your use case.
3. Fill in the app details:
   * **Display Name:** e.g., Post Genius
   * **Contact Email:** Your valid email
4. Click **“Create App”**.

**Step 3: Get FB\_APP\_ID and FB\_APP\_SECRET**

1. Once the app is created, you’ll be taken to the **App Dashboard**.
2. On the **“Settings > Basic”** page:
   * Copy your **App ID** → this is your FB\_APP\_ID
   * Click **“Show”** next to **App Secret** → this is your FB\_APP\_SECRET
3. Save both securely and don’t share them publicly.

**Step 4: Set the Redirect URI**

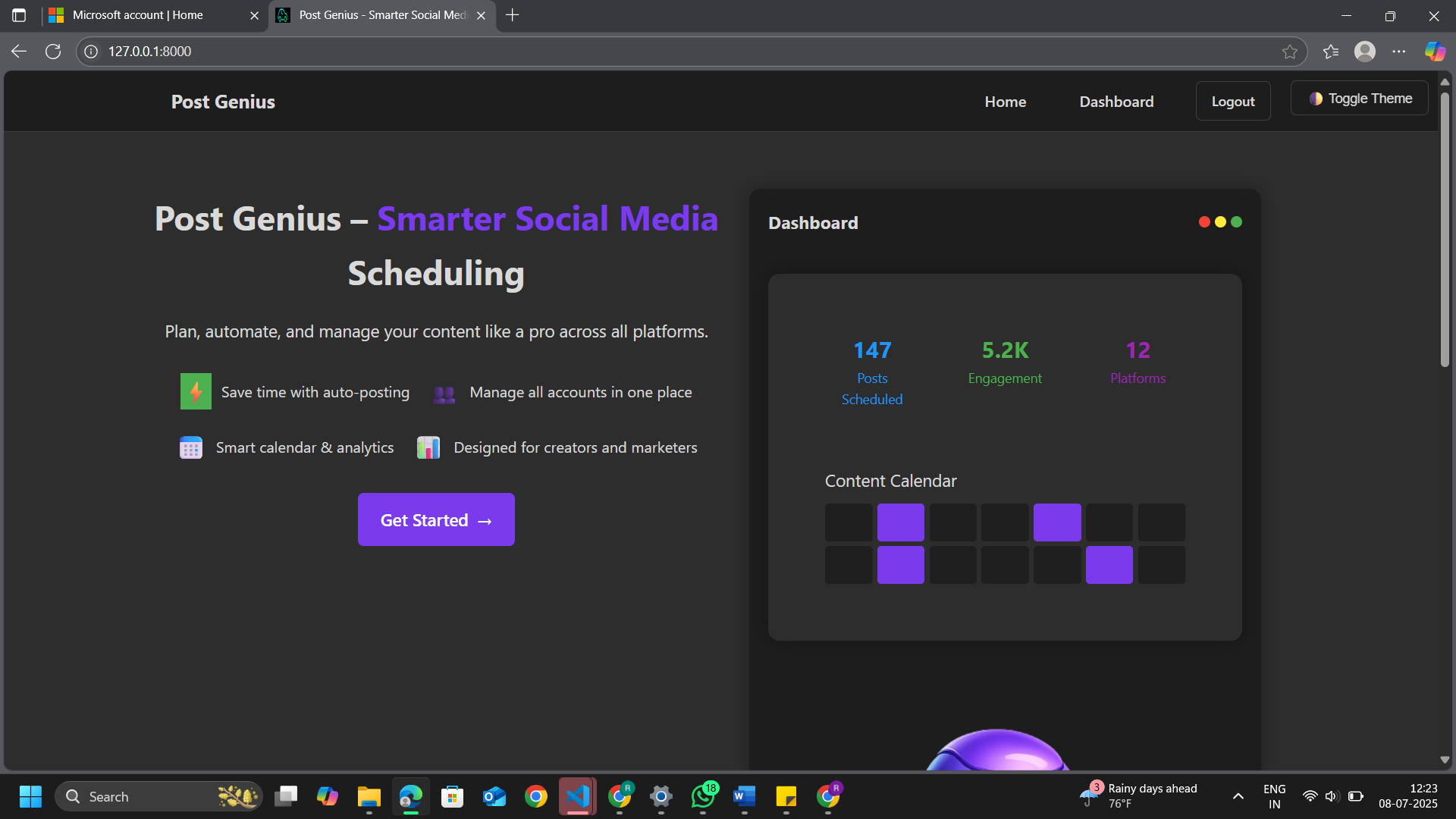
1. In the left sidebar, go to **"Facebook Login" > "Settings"**.
2. Under **"Valid OAuth Redirect URIs"**, enter:  
   http://localhost:8000/facebook/callback
3. Click **Save Changes**.  
   This becomes your:  
   FB\_REDIRECT\_URI=http://localhost:8000/facebook/callback

# Working Overview

**1. Landing Page**

**What Happens:**

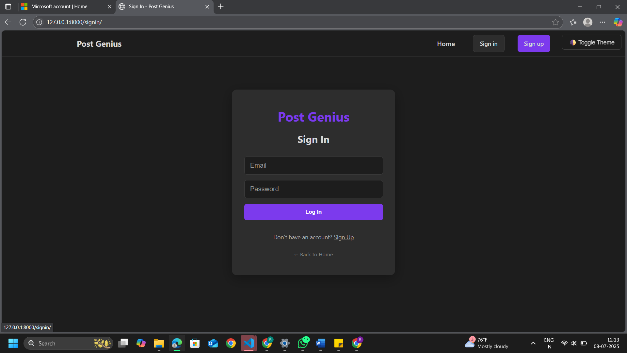
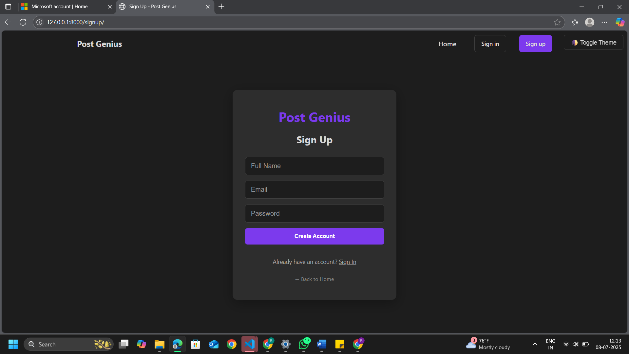
* User lands on a clean homepage with options to sign up or log in.
* Navigation bar includes links to pricing, demo, FAQ, and about.



**2. User Registration & Login**

**What Happens:**

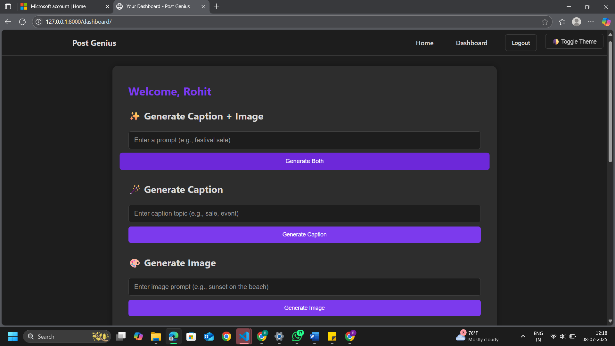
* Users can register with a username, email, and password.
* Login uses email and password to authenticate.
* Upon login, they’re redirected to the dashboard.



**3. Dashboard**

**What Happens:**

* Users see their name, history of generated captions and images.
* They can generate new captions and images, and export past work.
* Two forms: caption + image generation
* Captions listed below with timestamp
* Thumbnails of previously generated images



**4. Generate AI Caption**

**What Happens:**

* User enters a **topic** like “Winter Sale”.
* The backend sends this to OpenAI GPT-4 API.
* A catchy caption is returned and saved to history.

**Backend Logic:**

openai.ChatCompletion.create(

model="gpt-4",

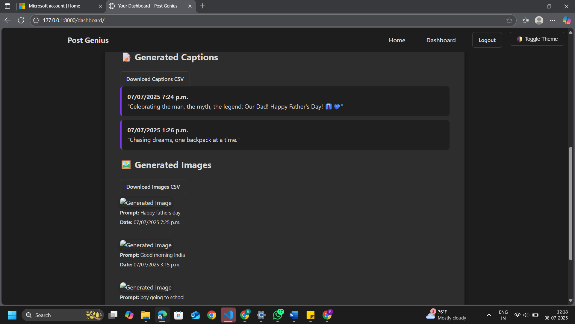
messages=[

{"role": "system", "content": "Generate catchy, brand-friendly captions."},

{"role": "user", "content": "Generate a caption for: Winter Sale"}

]

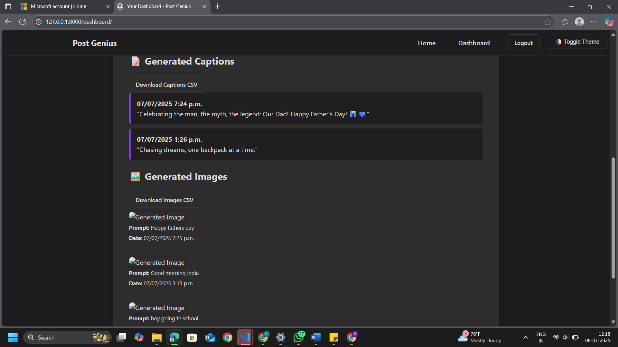
)



**5. Generate AI Image**

**What Happens:**

* User enters a **prompt** like “robot dancing under neon lights”.
* Sent to OpenAI DALL·E model using the image generation API.
* Image URL is returned and stored in the database.



**6. Predict Engagement**

**What Happens:**

* TextBlob is used to analyze sentiment polarity of the generated caption.
* Likes and shares are **simulated** based on sentiment and length.

**7. Connect Facebook via OAuth**

**What Happens:**

* User clicks “Connect Facebook”.
* They are redirected to Facebook’s OAuth login page.
* After approval, a token is saved securely.

**8. Post to Facebook / Instagram**

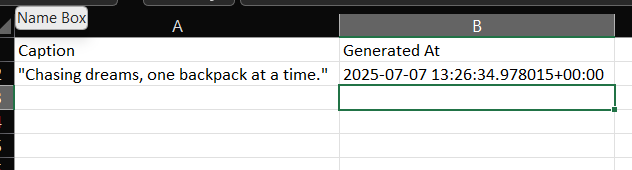
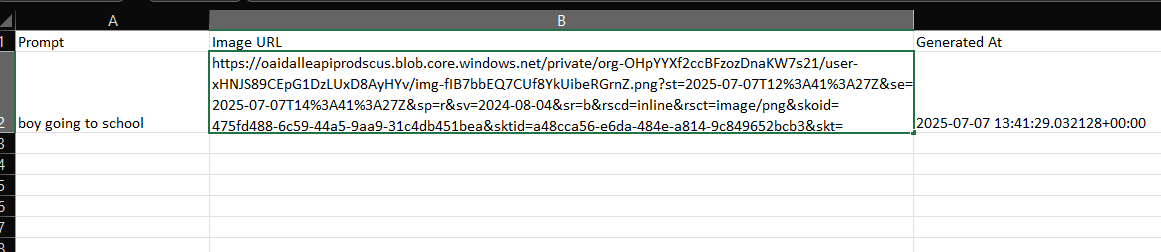
**What Happens:**

* User selects platform: Facebook or Instagram
* Chooses caption + image
* The app sends the post through Facebook Graph API

**9. Export to CSV**

**What Happens:**

* Users can export their captions or image data as .csv files.
* Useful for reporting or reuse on other platforms.

**** 

**10. User Profile & Avatar**

**What Happens:**

* Users can upload an avatar and write a bio.
* This is stored under UserProfile and editable from the dashboard.

# Example Captions and Images

Caption Example: Boy going to school

“Chasing dreams, one backpack at a time”

Image Example: Boy going to school



# Common Issues and Fixes

* No table: Run migrations with `makemigrations` and `migrate`.
* Token not found: Ensure Facebook is connected before posting.
* OpenAI error: Upgrade SDK and use the structured `OpenAI()` client.
* Replicate removed: Replaced with OpenAI DALL·E integration.

# Admin Panel Access

The Django Admin interface lets you manage all backend data including users, captions, images, and social tokens.

## Admin URL

http://localhost:8000/admin/  
Change the domain when deployed (e.g., <https://yourdomain.com/admin/>)

## Admin Credentials (for development)

| **Field** | **Value** |
| --- | --- |
| Username | rahul |
| Password | rahul@generator |

## Creating a Superuser (if not yet created)

If the admin user hasn’t been created yet, run the following command from your project root:  
python manage.py createsuperuser

## Recommendations

* Do **not** expose the default admin credentials in production.
* Use a strong password and enable 2FA on your host platform if supported.
* Use Django’s ALLOWED\_HOSTS and SECURE\_\* settings for production hardening.