Instagram Auto-Poster with AI (OpenAI API Key)

This notebook automates Instagram posting with AI-generated captions using OpenAI and Instagrapi.

Setup Instructions:

1. Install required packages
2. Set up environment variables
3. Configure folder path
4. Run the automation

1. Install Required Packages

In [13]:
```
#!pip install instagrapi openai Pillow
```

2. Import Libraries

In [14]:
```
import os
import time
import random
from pathlib import Path
from instagrapi import Client
from instagrapi.exceptions import LoginRequired
from openai import OpenAI
from PIL import Image
import tkinter as tk
from tkinter import filedialog
import base64
from datetime import datetime

print("✓ Libraries imported successfully")
```

```
✓ Libraries imported successfully
```

3. Configuration Setup

Using Environment Variables (Secure Method)

FIRST TIME SETUP:

Open a PowerShell and copy/paste this before running this script:

```
setx INSTA_USERNAME "your username/email here"
setx INSTA_PASSWORD "your password here"
setx OPENAI_API_KEY "your api here"
```

Check if OK - Open another PowerShell and copy/paste:

```
echo $env:INSTA_USERNAME
echo $env:INSTA_PASSWORD
echo $env:OPENAI_API_KEY
```

IMPORTANT: Reopen Jupyter Notebook

Environment variables are only recognized once Jupyter is reloaded.

In [15]:
```python
# Read credentials from environment variables
INSTAGRAM_USERNAME = os.getenv("INSTA_USERNAME")
INSTAGRAM_PASSWORD = os.getenv("INSTA_PASSWORD")
OPENAI_API_KEY = os.getenv("OPENAI_API_KEY")

# Session file to save login state
SESSION_FILE = "instagram_session.json"

# Default folder path
DEFAULT_FOLDER = r"C:\Users\User\Desktop\Instagram Postagens API"

# Supported file extensions
IMAGE_EXTENSIONS = [".jpg", ".jpeg", ".png"]
VIDEO_EXTENSIONS = [".mp4", ".mov"]

# Validation
print("\n=== Configuration Check ===")

if not INSTAGRAM_USERNAME:
    print("❌ INSTA_USERNAME not found!")
    print("   Run in PowerShell: setx INSTA_USERNAME \"your_username\"")
else:
    print(f"✓ Instagram username: @{INSTAGRAM_USERNAME}")

if not INSTAGRAM_PASSWORD:
    print("❌ INSTA_PASSWORD not found!")
    print("   Run in PowerShell: setx INSTA_PASSWORD \"your_password\"")
else:
    print(f"✓ Instagram password: {'*' * 10} (hidden)")

if not OPENAI_API_KEY:
    print("❌ OPENAI_API_KEY not found!")
    print("   Run in PowerShell: setx OPENAI_API_KEY \"your_api_key\"")
else:
    print(f"✓ OpenAI API key: {OPENAI_API_KEY[:15]}...{OPENAI_API_KEY[-4:]}")

if INSTAGRAM_USERNAME and INSTAGRAM_PASSWORD and OPENAI_API_KEY:
    print("\n✅ All credentials loaded successfully!")
else:
    print("\n⚠️  Some credentials are missing. Please set them and restart Jupy
```

```
=== Configuration Check ===
✓ Instagram username: [redacted]
✓ Instagram password: ********** (hidden)
✓ OpenAI API key: s[redacted]

✅ All credentials loaded successfully!
```

4. Initialize Clients

In [16]:
```python
# Initialize Instagram client
cl = Client()
cl.delay_range = [1, 3]

# Initialize OpenAI client
openai_client = OpenAI(api_key=OPENAI_API_KEY)

print("✓ Clients initialized")
```

```
✓ Clients initialized
```

5. Instagram Login Function

In [17]:
```python
def login_to_instagram():
    print("\n=== Instagram Login ===")

    if os.path.exists(SESSION_FILE):
        try:
            cl.load_settings(SESSION_FILE)
            print(f"✓ Loaded session from {SESSION_FILE}")
            cl.login(INSTAGRAM_USERNAME, INSTAGRAM_PASSWORD)
            cl.get_timeline_feed()
            print("✓ Session is valid and active")
            return True
        except LoginRequired:
            print("⚠ Saved session expired, performing fresh login...")
        except Exception as e:
            print(f"⚠ Error with saved session: {e}")

    try:
        print("Logging in to Instagram...")
        cl.login(INSTAGRAM_USERNAME, INSTAGRAM_PASSWORD)
        cl.dump_settings(SESSION_FILE)
        print(f"✓ Login successful! Session saved to {SESSION_FILE}")
        print(f"✓ Authenticated as @{INSTAGRAM_USERNAME} (ID: {cl.user_id})")
        return True
    except Exception as e:
        print(f"✗ Login failed: {e}")
        return False

login_success = login_to_instagram()
```

```
=== Instagram Login ===
✓ Loaded session from instagram_session.json
✓ Session is valid and active
```

6. Folder Selection Function

In [18]:
```python
def select_folder():
    root = tk.Tk()
    root.withdraw()
    root.attributes('-topmost', True)

    print("\n•   Please select the folder with your content...")

    folder_path = filedialog.askdirectory(
        title="Enter the full path to the folder with original images:",
        initialdir=DEFAULT_FOLDER if os.path.exists(DEFAULT_FOLDER) else os.path
    )

    root.destroy()

    if folder_path:
        print(f"✓ Selected folder: {folder_path}")
        return Path(folder_path)
    else:
        print("✗ No folder selected")
        return None

print("✓ Folder selection function ready")
```

```
✓ Folder selection function ready
```

7. AI Caption Generation Function

In [19]:

```
def generate_caption_with_ai(file_path):
    try:
        print(f"\n  🤖 Generating AI caption for: {file_path.name}")

        if file_path.suffix.lower() in IMAGE_EXTENSIONS:
            with open(file_path, "rb") as image_file:
                base64_image = base64.b64encode(image_file.read()).decode('utf-8

            response = openai_client.chat.completions.create(
                model="gpt-4o",
                messages=[
                    {
                        "role": "system",
                        "content": "You are an Instagram caption expert. Create
                    },
                    {
                        "role": "user",
                        "content": [
                            {
                                "type": "text",
                                "text": "Analyze this image and create an engagi
                            },
                            {
                                "type": "image_url",
                                "image_url": {
                                    "url": f"data:image/jpeg;base64,{base64_imag
                                }
                            }
                        ]
                    }
                ],
                max_tokens=500
            )
        else:
            response = openai_client.chat.completions.create(
                model="gpt-4o",
                messages=[
                    {
                        "role": "system",
                        "content": "You are an Instagram caption expert. Create
                    },
                    {
                        "role": "user",
                        "content": "Create an engaging Instagram caption for a v
                    }
                ],
                max_tokens=500
            )

        caption = response.choices[0].message.content.strip()

        print(f"  ✓ AI caption generated!")
        return caption

    except Exception as e:
        print(f"  ✗ Error generating AI caption: {e}")
        return "New post ✨\n\n#instagram #photography #lifestyle #instagood #pl

print("✓ AI caption function ready")
```

```
✓ AI caption function ready
```

8. Image Preparation Function

In [20]:
```
def prepare_image(image_path):
    if image_path.suffix.lower() in [".jpg", ".jpeg"]:
        return image_path

    try:
        print(f"  Converting {image_path.name} to JPG...")
        img = Image.open(image_path)

        if img.mode == 'RGBA':
            img = img.convert('RGB')

        jpg_path = image_path.with_suffix('.jpg')
        img.save(jpg_path, 'JPEG', quality=95)
        print(f"  ✓ Converted to: {jpg_path.name}")
        return jpg_path
    except Exception as e:
        print(f"  ✗ Error converting image: {e}")
        return None

print("✓ Image preparation function ready")
```

```
✓ Image preparation function ready
```

9. Post Upload Function

In [21]:
```
def upload_content(file_path, caption):
    try:
        file_ext = file_path.suffix.lower()

        print(f"\n  •  CAPTION TO POST:")
        print(f"  {'='*60}")
        print(f"{caption}")
        print(f"  {'='*60}\n")

        if file_ext in IMAGE_EXTENSIONS:
```

```python
            prepared_path = prepare_image(file_path)
            if not prepared_path:
                return False

            print(f"  📤 Uploading photo: {file_path.name}")
            media = cl.photo_upload(prepared_path, caption)

        elif file_ext in VIDEO_EXTENSIONS:
            print(f"  📤 Uploading video: {file_path.name}")
            media = cl.video_upload(file_path, caption)
        else:
            print(f"  ✗ Unsupported file type: {file_ext}")
            return False

        post_url = f"https://www.instagram.com/p/{media.code}/"
        print(f"  ✓ Successfully posted!")
        print(f"  🔗 View at: {post_url}")
        return True

    except Exception as e:
        print(f"  ✗ Upload failed: {e}")
        return False

print("✓ Upload function ready")
```

```
✓ Upload function ready
```

10. Main Automation Function

In [24]:
```python
def automate_instagram_posts(folder_path, delay_between_posts=60):
    if not folder_path or not folder_path.exists():
        print("✗ Invalid folder path")
        return

    all_extensions = IMAGE_EXTENSIONS + VIDEO_EXTENSIONS
    media_files = []

    for ext in all_extensions:
        media_files.extend(folder_path.glob(f"*{ext}"))

    if not media_files:
        print(f"✗ No media files found in {folder_path}")
        return

    # Remove duplicates (same filename, different extensions)
    unique_files = {}
    for file_path in media_files:
        base_name = file_path.stem  # filename without extension
        if base_name not in unique_files:
            unique_files[base_name] = file_path
        else:
            # Prefer JPG over PNG
            if file_path.suffix.lower() in ['.jpg', '.jpeg']:
                unique_files[base_name] = file_path

    media_files = list(unique_files.values())

    print(f"\n•   Found {len(media_files)} unique media file(s) to post")
    print("="*70)
    successful_posts = 0
    failed_posts = 0

    for idx, file_path in enumerate(media_files, 1):
        print(f"\n[{idx}/{len(media_files)}] Processing: {file_path.name}")
        print("-" * 70)

        try:
            caption = generate_caption_with_ai(file_path)
            success = upload_content(file_path, caption)

            if success:
                successful_posts += 1
            else:
                failed_posts += 1

            if idx < len(media_files):
                wait_time = random.uniform(delay_between_posts, delay_between_po
                print(f"\n   ⏳ Waiting {wait_time:.0f}s before next post...")
                time.sleep(wait_time)

        except Exception as e:
            print(f"   ✗ Error processing {file_path.name}: {e}")
            failed_posts += 1

    print("\n" + "="*70)
    print("•   POSTING SUMMARY")
    print("="*70)
    print(f"✓ Successful posts: {successful_posts}")
    print(f"✗ Failed posts: {failed_posts}")
    print(f"•   Total processed: {len(media_files)}")
    print(f"⏰ Completed at: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")

print("✓ Main automation function ready")
```

```
✓ Main automation function ready
```

11. Run the Automation

In [25]:
```
if not login_success:
    print("\n❌ Cannot proceed without successful Instagram login")
    print("Please check your credentials and try again")
else:
    selected_folder = select_folder()

    if selected_folder:
        print("\n🚀 Starting Instagram automation...")
        automate_instagram_posts(
            folder_path=selected_folder,
            delay_between_posts=90
        )
    else:
        print("\n❌ No folder selected. Automation cancelled.")
```

```
•   Please select the folder with your content...
✓ Selected folder: C:/Users/User/Desktop/Instagram Postagens Python

🚀 Starting Instagram automation...

•   Found 1 unique media file(s) to post
======================================================================

[1/1] Processing: IMG_1515.jpg
----------------------------------------------------------------------

  🤖 Generating AI caption for: IMG_1515.jpg
  ✓ AI caption generated!

  •   CAPTION TO POST:
  ============================================================
The energy in the arena is electric as the Boston Celtics take on the Orlando Mag
ic. An American flag ceremony adds a patriotic touch to the court, surrounded by
fans eagerly awaiting the tip-off. It's game night magic in the heart of basketba
ll action! 🏀🇺🇸

#BasketballNight #OrlandoMagic #BostonCeltics #GameDayVibes #HoopsLife #NBAAction
#SportsFans #ArenaMagic #BasketballSeason #USA #CourtSide #LiveSports #BallIsLife
#TeamSpirit #NightOut
  ============================================================

  📤 Uploading photo: IMG_1515.jpg
  ✓ Successfully posted!
  🔗 View at: https://www.instagram.com/p/DRauIyECWsk/

======================================================================
•   POSTING SUMMARY
======================================================================
✓ Successful posts: 1
✗ Failed posts: 0
•   Total processed: 1
⏰ Completed at: 2025-11-23 20:31:56
```

📝 Notes & Best Practices

Security

- **Never** hardcode credentials in the notebook
- Use environment variables
- Keep your session file secure

Rate Limiting

- Instagram has strict rate limits
- Keep delays between posts (90+ seconds)
- Don't post too many items at once

File Requirements

- **Images**: JPG format (PNG will be converted)

- **Videos**: MP4 or MOV format
- **Size**: Keep files under 8MB

Troubleshooting

- If login fails, check credentials
- If posts fail, check rate limits
- Monitor output for errors

Responsible Use

- Use automation ethically
- Don't spam
- Respect Instagram's Terms of Service