Create an automation workflow using n8n installed locally on your system via Docker to post content automatically on a Facebook Page and Instagram using the Gemini API. The objective is to build a content generation and publishing pipeline that runs either on a schedule or via webhook trigger.

Start by setting up n8n using Docker. You can use a simple docker-compose.yml file to run n8n locally on port 5678. Once n8n is up and running, access the editor via localhost and begin creating your workflow.

Use an HTTP Request node to connect with the Gemini API. This node should send a prompt to Gemini asking it to generate a short, engaging post for a business or personal page. Parse the response to extract the text output.

Next, create a sample Facebook page and ensure you have an access token with permissions to post. Add another HTTP Request node that uses the Facebook Graph API to post the generated text to your Facebook Page feed. Include parameters such as message content and access token. For Instagram, use the Facebook Graph API to first create a media object with an image URL and caption, and then publish it using a second HTTP request.

You can link all the nodes together using a Cron node to automate the posting on a daily or weekly basis. Alternatively, use a Webhook node to trigger the workflow manually. Make sure to test each step individually before chaining them together. The final workflow should successfully generate text using the Gemini API and post it automatically to your Facebook Page and Instagram account.

Once the assignment is complete, share a video demonstrating the workflow in action, including the successful posts made on Facebook and Instagram. The deadline to complete and submit the assignment is 5 days from the date this task is shared. This internship will be paid, but the stipend will depend on performance. Interns can opt for full-time or part-time work, and the specifics will be discussed after evaluation.

## Project Overview:

Your project’s **goal** is to **automate content generation and posting** on **Facebook** and **Instagram** using the **Gemini API**. Here's the high-level flow:

**Objective:**

* **Automate the process of creating and posting content** to your **Facebook Page** and **Instagram** using n8n and the Gemini API.
* You will:
  + Generate content using Gemini (an AI model).
  + Post the content to both social platforms (Facebook Page and Instagram).

**Workflow Steps:**

1. **Content Generation (Gemini API):**
   * You'll call the **Gemini API** to generate short, engaging text content for your social media posts.
   * The **HTTP request node** in n8n will interact with Gemini to send prompts and retrieve the generated text.
2. **Social Media Posting (Facebook & Instagram):**
   * Once the content is generated, you'll post it to **Facebook** using the **Facebook Graph API**.
   * For **Instagram**, you will first create a media object and then publish it.

### ****Day 1: Setting Up n8n and Testing HTTP Requests****

#### **Objective of Day 1:**

The goal for Day 1 was to set up **n8n** on your local system using Docker, ensure it is working properly, and test the HTTP request functionality by making API calls. This setup is essential for creating automated workflows that will later interact with the **Gemini API** to generate content and post it on **Facebook** and **Instagram**.

### ****Step 1: Setting Up n8n Locally Using Docker****

**1.1 Docker Setup:**

* **Why Docker?**  
  Docker allows you to run n8n in an isolated, consistent environment without worrying about system dependencies.
* **Steps to Set Up n8n Using Docker:**
  1. Create a new directory on your desktop (e.g., n8n-local).
  2. Create a docker-compose.yml file inside the n8n-local directory. This file defines the services needed to run n8n, including the necessary Docker containers.

version: "3”

services:

n8n:

image: n8nio/n8n

restart: always

ports:

- "5678:5678"

environment:

- N8N\_BASIC\_AUTH\_ACTIVE=true

- N8N\_BASIC\_AUTH\_USER=admin

- N8N\_BASIC\_AUTH\_PASSWORD=password

- N8N\_HOST=localhost

- N8N\_PORT=5678

volumes:

- ./n8n\_data:/home/node/.n8n

* 1. Run docker-compose up to spin up the n8n environment.
* **Why Did You Do This?**  
  By running n8n locally in Docker, you have full control over your automation workflows. This environment allows you to design and automate tasks without needing to depend on third-party services.

**1.2 Verifying n8n Setup:**

* Once n8n was set up and Docker was running, you accessed n8n via **localhost:5678**.
* You logged in successfully and reached the **workflow editor** interface, where you can start designing automation tasks.

### ****Step 2: Creating Your First Workflow in n8n****

**2.1 First Workflow:**

* You created a **simple workflow** to test n8n's functionality. Specifically, this workflow used the **HTTP Request node** to interact with an API and fetch data.
* **API Chosen for Testing:**  
  You used the publicly available test API from JSONPlaceholder. This API returns a sample JSON response about a to-do list item.

**API URL:**

arduino

CopyEdit

https://jsonplaceholder.typicode.com/todos/1

* **Steps in the Workflow:**
  1. **HTTP Request Node:**
     + Method: GET
     + URL: <https://jsonplaceholder.typicode.com/todos/1>

**JSONPlaceholder** is a free API used to provide mock data for testing and learning purposes. It’s often used by developers when they want to simulate interaction with a REST API, but they don’t need to connect to an actual backend service.

* 1. **Execution of Workflow:**
     + You executed the workflow to test if the HTTP Request node could correctly interact with the API.

**2.2 Testing the HTTP Request Node:**

* After executing the workflow, you observed the output.
* The expected response from the API should be:

json

CopyEdit

{

"userId": 1,

"id": 1,

"title": "delectus aut autem",

"completed": false

}

* The response was visible in the **Output Data** tab of the HTTP Request node.
* **Why Did You Do This?** Testing this simple HTTP request confirmed that n8n can successfully interact with external APIs. It also showed that n8n processes **JSON responses** correctly, an essential part of fetching data from APIs like the **Gemini API**.

### ****Step 3: Analyzing the Workflow Execution****

**3.1 Inspecting the Response:**

* After execution, you examined the output and found that the **JSON data** was successfully returned and parsed.
* This confirmed that n8n can handle **API responses** and retrieve useful data for future workflows.
* **What You Learned:**
  + n8n’s **HTTP Request node** works well for making API calls and handling responses.
  + You can use the same approach to make requests to other APIs, such as **Gemini** for content generation, and **Facebook/Instagram Graph API** for social media posting.

Day 2  
Enable Gemini to **generate short, engaging Instagram content** using dynamic prompts and pass the result into the automated posting workflow.

**🔑 1. Gemini API Key**

* ✅ Created API key from **Google AI Studio**
* ✅ Set the key securely in **n8n HTTP Request node credentials**
* 🔐 Ensured sensitive fields are hidden and token is reusable in workflow

**🔗 2. HTTP Request Node to Gemini**

* ✅ Method: POST
* ✅ URL: https://generativelanguage.googleapis.com/v1beta/models/gemini-pro:generateContent?key=YOUR\_API\_KEY
* ✅ Headers:

json

CopyEdit

{

"Content-Type": "application/json"

}

* ✅ Body:

json

CopyEdit

{

"contents": [

{

"parts": [

{

"text": "Write a short, catchy Instagram post about the importance of staying updated with tech tips."

}

]

}

]

}

**🛠️ 3. Parse & Extract Response**

* ✅ Used a **Set Node** (or **Function Node**) to extract:
  + output\_text = items[0].response.candidates[0].content.parts[0].text
* ✅ Stored output in a variable instagram\_caption
* ⚠️ Handled cases where API might not return candidates array (fallback response)

**🧪 4. Test & Verify**

* ✅ Ran workflow and received structured content like:

arduino

CopyEdit

"💡 Stay ahead in the tech game! Today's quick tip: Keep your software updated to avoid security risks and enjoy new features. #TechTipTuesday"

* ✅ Verified emoji usage, tone, and hashtags for Instagram-friendliness
* 🧠 Prompt refined for better clarity and engagement over multiple test runs

### Theoretical Explanation: Sending Content to Gemini API

To generate social media content using the **Gemini API by Google**, we interact with the API through an HTTP POST request. This involves sending a prompt — a specific instruction — to the API and receiving a text-based response generated by the model.

#### 1. **API Endpoint**

The Gemini API is accessed through a specific endpoint provided by Google. To use it, an API key must be obtained from **Google AI Studio**. The request is made to the endpoint:

bash

CopyEdit

https://generativelanguage.googleapis.com/v1beta/models/gemini-pro:generateContent?key=YOUR\_API\_KEY

This URL targets the gemini-pro model and includes the API key as a query parameter for authentication.

#### 2. **HTTP Request**

The API call is made using the HTTP **POST** method. This is appropriate because we are sending data (the prompt) to the server and requesting a response.

#### 3. **Headers**

To communicate properly with the API, we set the header:

pgsql

CopyEdit

Content-Type: application/json

This tells the API that we are sending JSON-formatted data.

#### 4. **Request Body**

The core of the request is the **prompt**, which is embedded in the JSON body. The prompt is the instruction we give to Gemini, such as:

"Write a short, catchy Instagram post about the importance of staying updated with tech tips."

The JSON structure for the prompt looks like this:

json

CopyEdit

{

"contents": [

{

"parts": [

{

"text": "Your prompt text here"

}

]

}

]

}

#### 5. **API Response**

The Gemini API processes the prompt using its language model and returns a **response** in JSON format. This response includes a list of candidates (possible responses) generated by the model.

To extract the generated text from the response, we parse the object:

json

CopyEdit

response.candidates[0].content.parts[0].text

#### 6. **Usage**

The extracted text is then used for publishing social media posts on platforms like **Instagram** or **Facebook**. It can be customized, edited, or directly passed to the next step of the automation workflow.