

**The Superior University, Lahore**



**TASK 7**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Course Title: | Programming for AI | | | | Course Code: | CAI601410 | Credit Hours: | 1 |
| Instructor: | Sir Rasikh | | | | Program Name: | BSDS | | |
| Semester: | 4th | Batch: | F23 | Section: | BSDSM-4A | Date: | March 17 , 2024 | |
| Time Allowed: |  | | | | Maximum Marks: | |  | |
| Student’s Name: | Rafay Noor | | | | Reg. No. | Su92-bsdsm-f23-024 | | |
| **Task 7:** **Flask API for NASA APOD (Backend and Frontend)** | | | | | | | | |

## **Overview**

This code is a simple Flask web application that fetches data from NASA's Astronomy Picture of the Day (APOD) API and return the response in JSON format.

**Breakdown of the Code:**

1. **Importing Required Modules**

from flask import Flask, jsonify

import requests

* Flask: A lightweight web framework in Python used to create web applications.
* jsonify: Converts Python dictionaries into JSON format, making API responses structured and readable.
* requests: A library used to send HTTP requests, In this case to NASA's APOD API.

**2. Initializing the Flask Application**

app = Flask(\_\_name\_\_)

Creates an instance of the Flask application.

* The \_\_name\_\_ parameter tells Flask whether the script is run directly or imported as a module.

**3. Defining API Key and Endpoint**

NASA\_API\_KEY = "Gw41GbIQqg00t5XWSiVM54C6LMH7aZGt3RTwzOUG"

NASA\_APOD\_URL = "https://api.nasa.gov/planetary/apod"

* NASA\_API\_KEY: The API key required to authenticate the request to NASA’s API.
* NASA\_APOD\_URL: The endpoint for NASA's APOD API, which returns the astronomy picture of the day along with details.

**4. Creating the Route for APOD API**

@app.route("/apod", methods=["GET"])

def get\_apod():

    params = {"api\_key": NASA\_API\_KEY}

    response = requests.get(NASA\_APOD\_URL, params=params)

    if response.status\_code == 200:

        return jsonify(response.json())

    else:

        return jsonify({"error": "Failed to fetch data"}), response.status\_code

* @app.route("/apod", methods=["GET"]): This sets up a route (/apod) that allows users to access the NASA APOD data using the HTTP GET method.
* params = {"api\_key": NASA\_API\_KEY}: Defines the parameters required to authenticate the API request.
* response = requests.get(NASA\_APOD\_URL, params=params): Sends an HTTP GET request to NASA’s APOD API with the API key.
* If-Else Block:
  + If the request is successful (status\_code == 200), the JSON response from NASA’s API is returned.
  + If the request fails, an error message is returned along with the respective status code.

**5. Defining the Home Route**

@app.route("/")

def home():

    return get\_apod()

* @app.route("/"): This defines the homepage (/).
* return get\_apod(): When the homepage is accessed, it automatically calls the get\_apod() function and displays the NASA APOD data.

**6. Running the Application**

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(debug=True)

* if \_\_name\_\_ == "\_\_main\_\_": Ensures that the script runs only when executed directly (not when imported).
* app.run(debug=True):
  + Starts the Flask application in debug mode.

This project is a **NASA Astronomy Picture of the Day (APOD) viewer** that fetches daily space images from NASA's API and displays them on a webpage. It consists of four key files:

**2. Frontend: index.html**

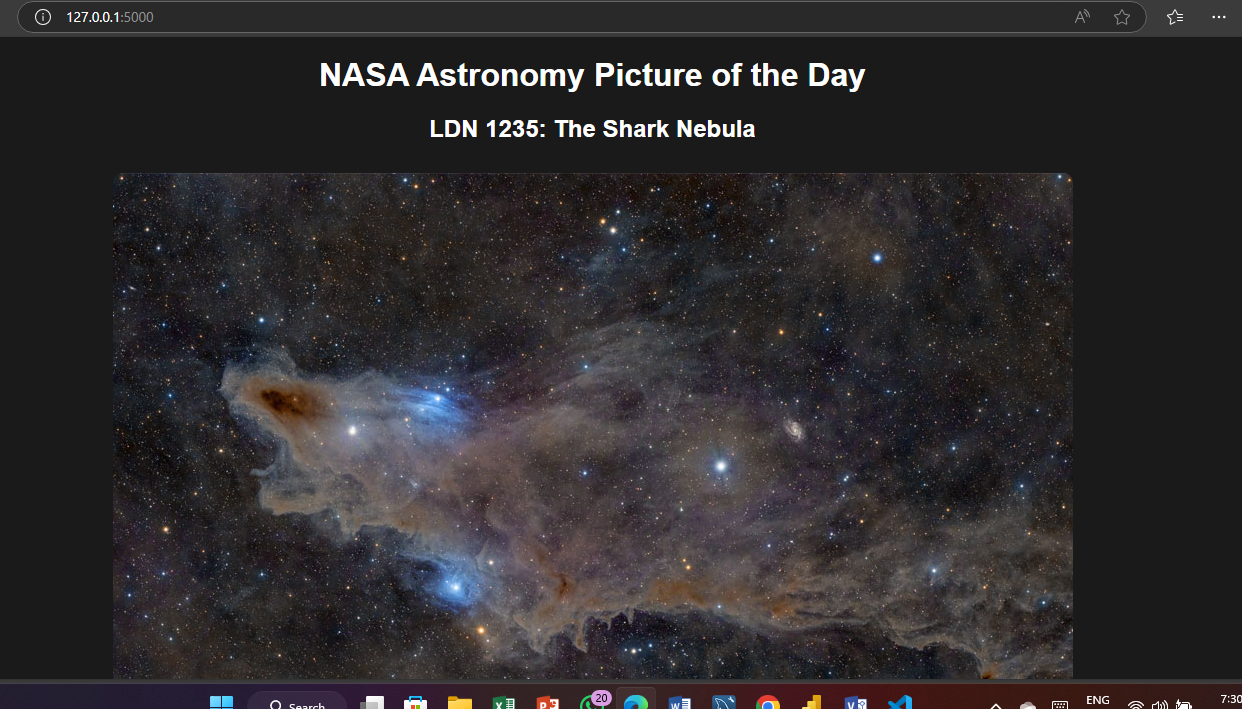
* **Why?**  
  This file provides the structure of the webpage, displaying the APOD data.
* **How?**
  + Uses an <h1> for the title.
  + Has a <div> (id="apod-container") where APOD data is displayed.
  + Uses a <script> tag to include script.js, which fetches and updates content dynamically.

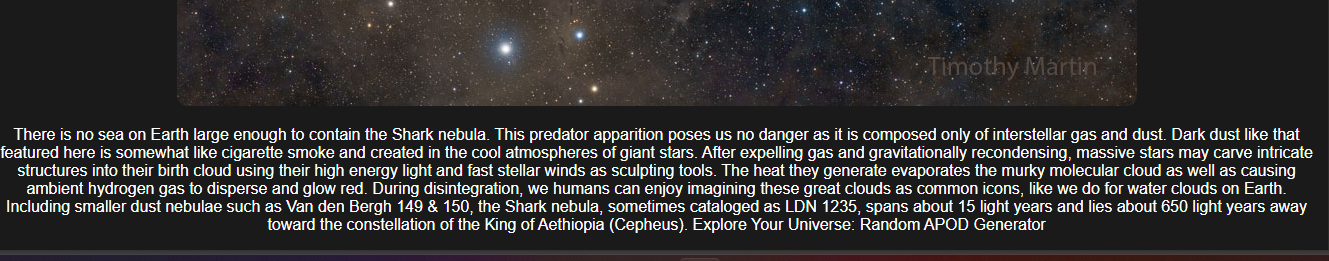
**3. Styling: styles.css**

* **Why?**  
  Defines the webpage’s visual appearance.
* **How?**
  + Uses **dark mode** (black background, white text).
  + Centers text and images.
  + Adds **rounded corners** to images for better aesthetics.

**4. Frontend Logic: script.js**

* **Why?**  
  Fetches APOD data from the backend and updates the HTML dynamically.
* **How?**
  + Waits for the page to load (DOMContentLoaded event).
  + Uses **fetch()** to request data from "/apod".
  + Updates HTML elements (title, img, explanation) with API response data.

**Example API Response (Sample Output)**

****