****

**Name: Ahsan Ali**

**Roll No: 061**

**Task: 7,8**

**Section: BSAI-4B**

**Subject: PAI Lab**

**Submitted To: Sir Rasikh**

**Title:**

**Vehicle Info App using Flask & VIN API**

**Objective:**

To design and implement a web-based application using Flask that takes a VIN (Vehicle Identification Number) as input, fetches vehicle details from a free API, and displays the results in a user-friendly interface.

**Tools & Technologies Used:**

| **Tool/Library** |  |  | **Purpose** |
| --- | --- | --- | --- |
| Python |  |  | Backend programming language |
| Flask |  |  | Web framework |
| HTML/CSS |  |  | Frontend layout and styling |
| JavaScript |  |  | Frontend validation and interaction |
| NHTSA API |  |  | Free vehicle info API |
| Requests |  |  | API integration with Python |

**Free API Used:**

* API: National Highway Traffic Safety Administration (NHTSA)
* Endpoint:  
  https://vpic.nhtsa.dot.gov/api/vehicles/DecodeVinValuesExtended/{vin}?format=json
* Data Extracted:
  + Make
  + Model
  + Year
  + Fuel Type
  + Engine Model
  + Vehicle Class

**Project Workflow:**

1. User Interface:
   * A clean form where the user enters a 17-character VIN.
2. Form Submission:
   * Form sends data via POST to Flask backend.
3. Backend Process:
   * VIN is validated.
   * Flask makes a request to the NHTSA VIN API.
   * JSON response is parsed to extract required fields.
4. Response Display:
   * Information is displayed using Jinja in an elegant card layout.
5. Validation:
   * Client-side JavaScript ensures VIN is exactly 17 characters.

**Screenshots:**

**Code: app.py**

from flask import Flask, render\_template, request

import requests

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

@app.route('/', methods=['GET', 'POST'])

def index():

    vehicle\_data = None

    error = None

    if request.method == 'POST':

        vin = request.form['vin'].strip()

        if len(vin) != 17:

            error = "VIN must be exactly 17 characters long."

        else:

            url = f"https://vpic.nhtsa.dot.gov/api/vehicles/DecodeVinValuesExtended/{vin}?format=json"

            try:

                response = requests.get(url)

                results = response.json()['Results'][0]

                vehicle\_data = {

                    'Make': results.get('Make', 'N/A'),

                    'Model': results.get('Model', 'N/A'),

                    'Year': results.get('ModelYear', 'N/A'),

                    'Fuel Type': results.get('FuelTypePrimary', 'N/A'),

                    'Engine': results.get('EngineModel', 'N/A'),

                    'Class': results.get('VehicleType', 'N/A')

                }

            except Exception as e:

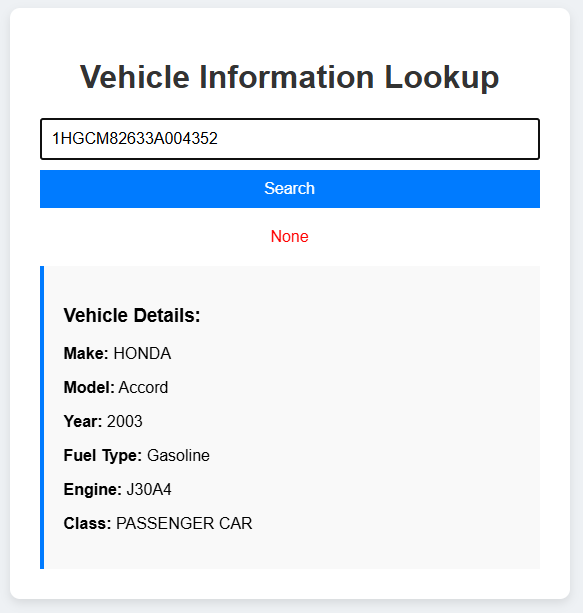
                error = "Failed to fetch vehicle data."

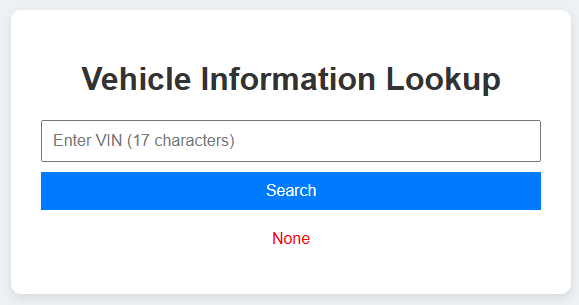
    return render\_template('index.html', vehicle=vehicle\_data, error=error)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

**Output:**

****

****

**Sample VIN for Testing:**

* VIN: 1HGCM82633A004352
* Result:
  + Make: Honda
  + Model: Accord
  + Year: 2003
  + Fuel Type: Gasoline
  + Engine: 4-cylinder
  + Class: Passenger Car

**Folder Structure:**

vehicle-info-app/

├── app.py

├── templates/

│ └── index.html

├── static/

│ ├── style.css

│ └── script.js

└── requirements.txt

**Testing and Validation:**

| Test Case |  |  | Result |
| --- | --- | --- | --- |
| Valid VIN input |  |  | Vehicle data shown correctly |
| Short VIN (invalid) |  |  | Error shown |
| Empty input |  |  | Validation alert |
| API unavailable |  |  | Error handled gracefully |

**Conclusion:**

* Successfully built a real-time VIN decoder web app using Flask and a free API.
* Integrated backend API calls, frontend design, and JavaScript validation.
* Learned to process JSON, handle HTTP requests, and improve UX with styling and form feedback.