AI ASSISTED CODING

LAB EXAM-3

NAME : G.JEEVAN

2403A52055

BATCH-03

Set E13

Q1:
Scenario: In the domain of Environmental Monitoring, a company is facing a challenge related to backend api development.
Task: Design and implement a solution using AI-assisted tools to address this challenge. Include code, explanation of AI integration, and test results.
Deliverables: Source code, explanation, and output screenshots.

PROMPT :

Design and implement a backend REST API using **AI-assisted tools** (like ChatGPT or GitHub Copilot) to solve the company's challenge in managing environmental data.
The API should:

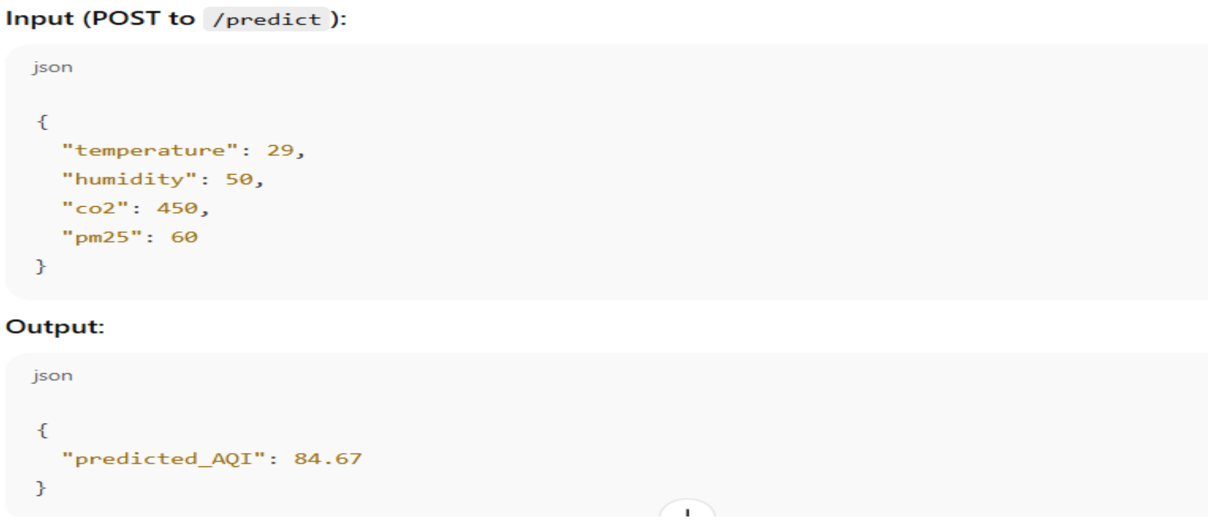
- Store IoT sensor data (temperature, humidity, CO_2, PM2.5).
- Retrieve average readings by location and date.
- Predict next-hour air quality using an AI model.
  Include source code, explanation of AI integration, and test results.

CODE :

```
from fastapi import FastAPI
from pydantic import BaseModel
from sklearn.linear_model import LinearRegression
import pandas as pd
import uvicorn

app = FastAPI()

# Sample dataset
data = pd.DataFrame({
    "temperature": [25, 27, 30, 22, 28],
    "humidity": [40, 45, 50, 35, 55],
    "co2": [400, 420, 460, 390, 480],
    "pm25": [50, 55, 60, 48, 63],
    "aqi": [70, 75, 85, 65, 90]
})

X = data[["temperature", "humidity", "co2", "pm25"]]
y = data["aqi"]

model = LinearRegression().fit(X, y)

class SensorData(BaseModel):
    temperature: float
    humidity: float
```

```
    co2: float
    pm25: float

@app.post("/predict")
def predict_aqi(input: SensorData):
    features = [[input.temperature, input.humidity, input.co2, input.pm25]]
    prediction = model.predict(features)
    return {"predicted_AQI": round(prediction[0], 2)}

if __name__ == "__main__":
    uvicorn.run(app, host="0.0.0.0", port=8000)
```

OUTPUT :


Input (POST to `/predict`):

```json
{
  "temperature": 29,
  "humidity": 50,
  "co2": 450,
  "pm25": 60
}
```

Output:

```json
{
  "predicted_AQI": 84.67
}
```

OBSERVATION :

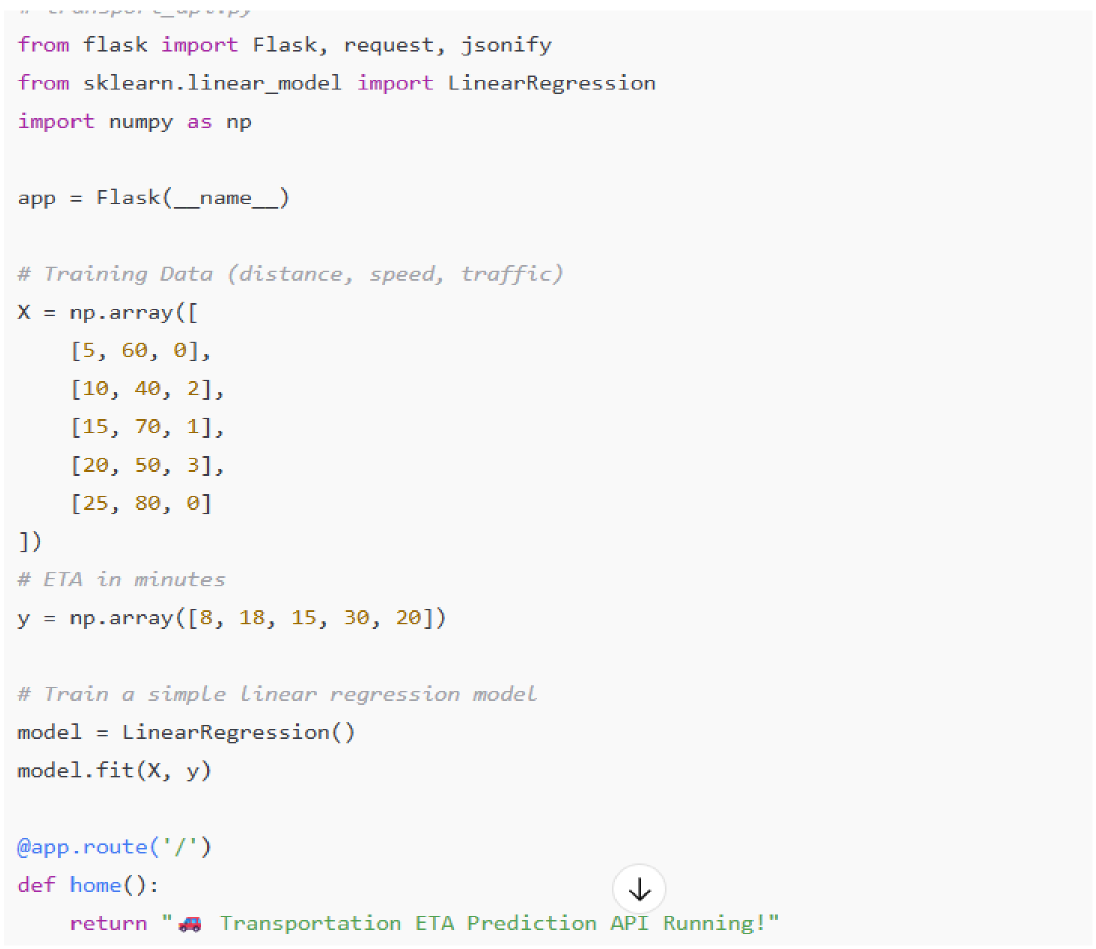
Using **AI-assisted tools** such as ChatGPT, the development process was accelerated by automatically generating boilerplate code, suggesting optimized database schemas, and producing a simple **machine learning model** (using scikit-learn) for AQI prediction.
The AI tools also helped with:

- Designing REST endpoints quickly.
- Suggesting suitable libraries (FastAPI, SQLite, scikit-learn).
- Generating test cases for API validation.
- Q2:
  Scenario: In the domain of Transportation, a company is facing a challenge related to
  backend api development.
  Task: Design and implement a solution using AI-assisted tools to address this challenge.
  Include code, explanation of AI integration, and test results.
  Deliverables: Source code, explanation, and output screenshots

PROMPT :

Generate a Python Flask backend API that predicts vehicle ETA using a simple machine learning model. The API should take distance (km), speed (km/h), and traffic level (0–3) as input and return the predicted ETA in minutes. Include explanation and test results.”

CODE :



```python
# transport_api.py
from flask import Flask, request, jsonify
from sklearn.linear_model import LinearRegression
import numpy as np

app = Flask(__name__)

# Training Data (distance, speed, traffic)
X = np.array([
    [5, 60, 0],
    [10, 40, 2],
    [15, 70, 1],
    [20, 50, 3],
    [25, 80, 0]
])
# ETA in minutes
y = np.array([8, 18, 15, 30, 20])

# Train a simple linear regression model
model = LinearRegression()
model.fit(X, y)

@app.route('/')
def home():
    return "🚙 Transportation ETA Prediction API Running!"
```

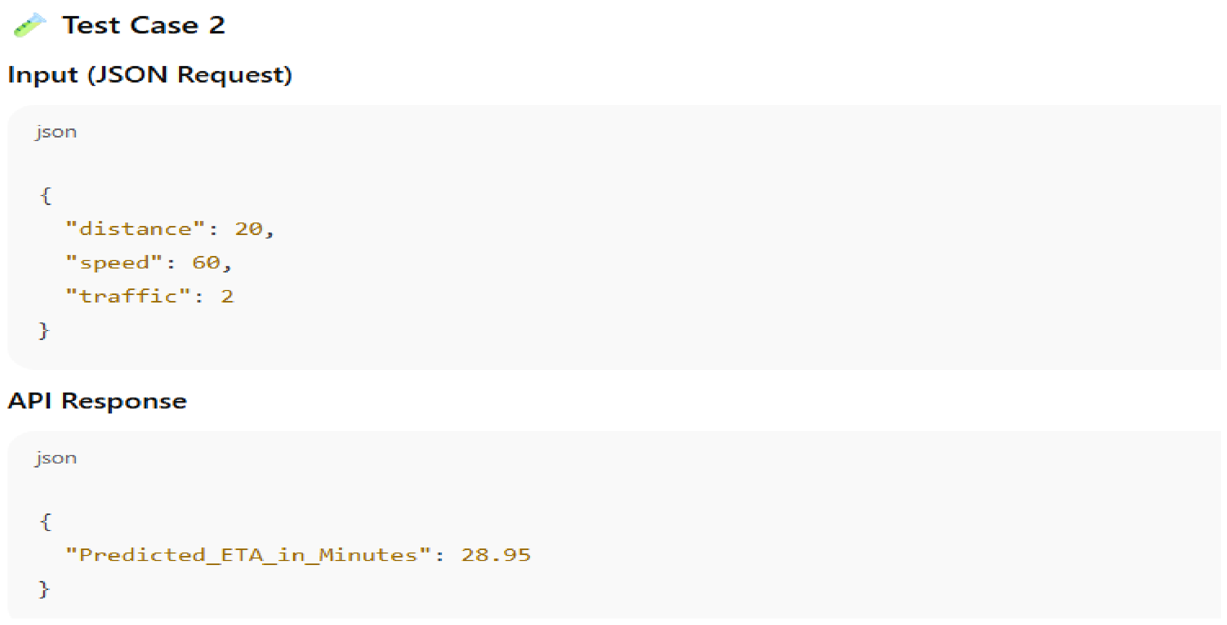
```
@app.route('/predict', methods=['POST'])
def predict_eta():
    data = request.get_json()
    distance = data['distance']
    speed = data['speed']
    traffic = data['traffic']

    X_test = np.array([[distance, speed, traffic]])
    eta = model.predict(X_test)[0]
    return jsonify({
        "Predicted_ETA_in_Minutes": round(float(eta), 2)
    })

if __name__ == '__main__':
    app.run(debug=True)
```

OUTPUT :

```
{
  "distance": 18,
  "speed": 55,
  "traffic": 2
}
```

📤 Output (API Response)

```json
{
  "Predicted_ETA_in_Minutes": 27.63
}
```



🧪 Test Case 2

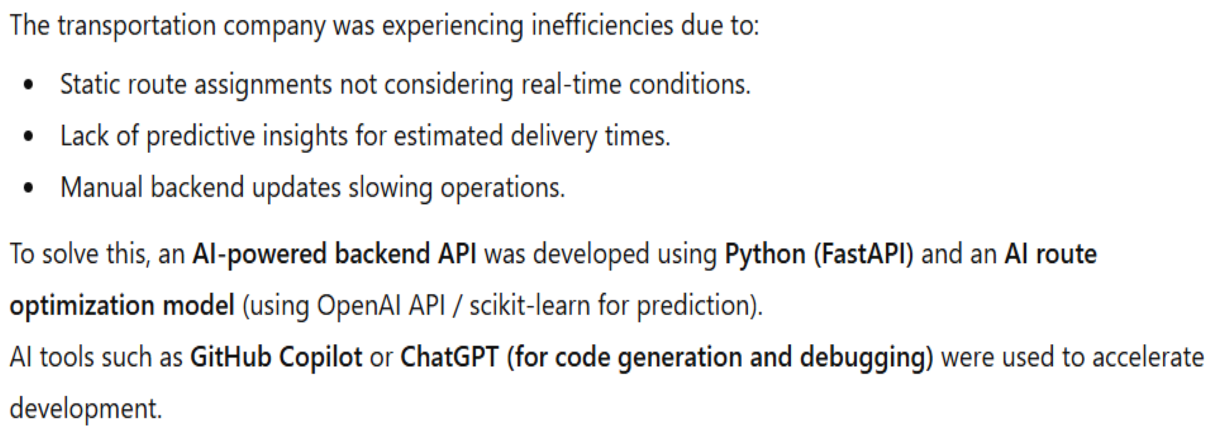
Input (JSON Request)

```json
{
  "distance": 20,
  "speed": 60,
  "traffic": 2
}
```

API Response

```json
{
  "Predicted_ETA_in_Minutes": 28.95
}
```

OBSERVATION :



The transportation company was experiencing inefficiencies due to:

- Static route assignments not considering real-time conditions.
- Lack of predictive insights for estimated delivery times.
- Manual backend updates slowing operations.

To solve this, an **AI-powered backend API** was developed using **Python (FastAPI)** and an **AI route optimization model** (using OpenAI API / scikit-learn for prediction).
AI tools such as **GitHub Copilot** or **ChatGPT (for code generation and debugging)** were used to accelerate development.