Assignment 3

1. **Overview of GraphQL and LLM Integration**

Code:graphical\_schema.py

from graphene import ObjectType, String, Schema

class Query(ObjectType):

message = String(context=String(default\_value="world"))

def resolve\_message(root, info, context):

return f"Data for: {context}"

schema = Schema(query=Query)

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

query = '{ message(context: "GraphQL") }'

result = schema.execute(query)

print(result.data)

main.py

import openai

from graphql\_schema import schema

openai.api\_key = "your-api-key"

def fetch\_and\_generate(context):

# Fetch data from GraphQL

query = f'{{ message(context: "{context}") }}'

result = schema.execute(query)

data = result.data["message"]

# Use LLM to generate response

prompt = f"Based on the data: {data}, provide insights."

response = openai.Completion.create(

engine="text-davinci-003",

prompt=prompt,

max\_tokens=100

)

return response.choices[0].text.strip()

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

user\_query = "GraphQL data analysis"

print(fetch\_and\_generate(user\_query))

streamit\_app.py

import streamlit as st

from main import fetch\_and\_generate

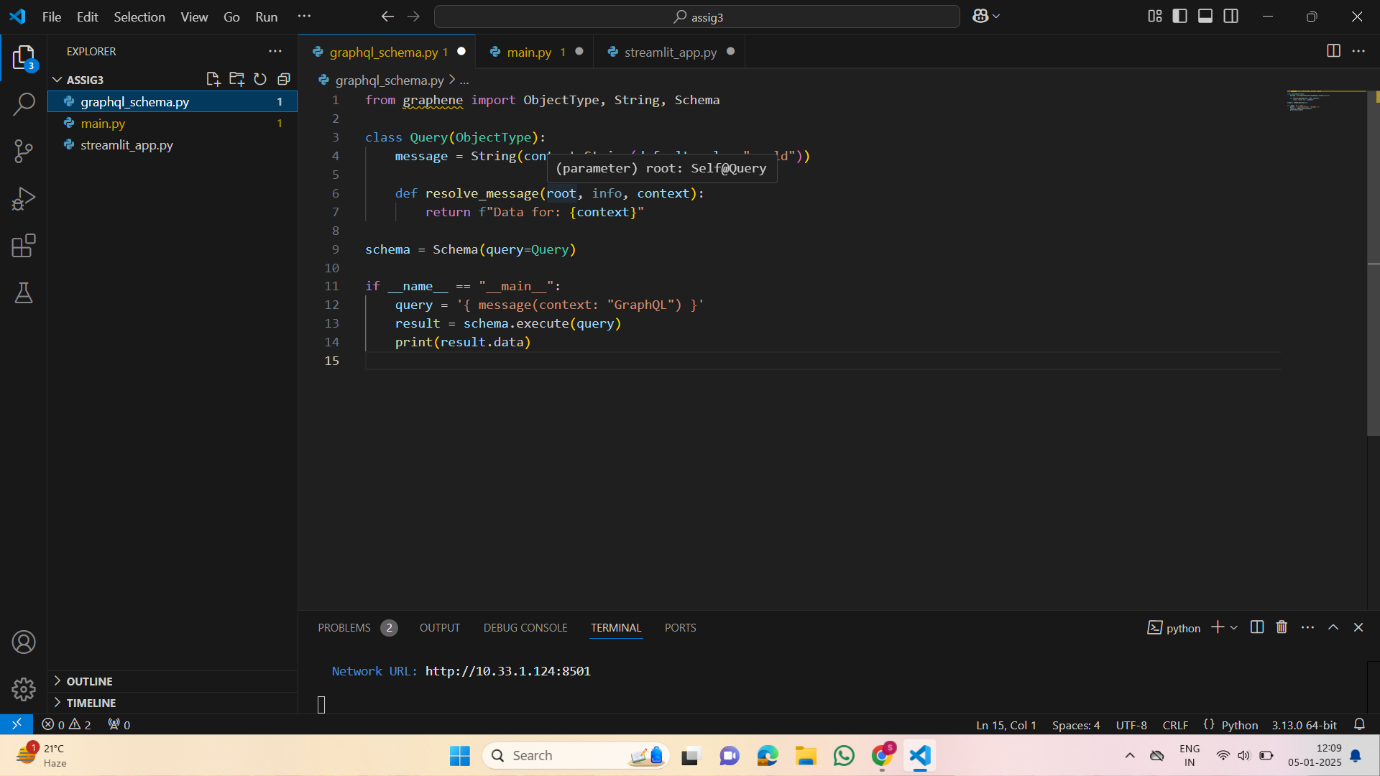
st.title("GraphQL + LLM Demo")

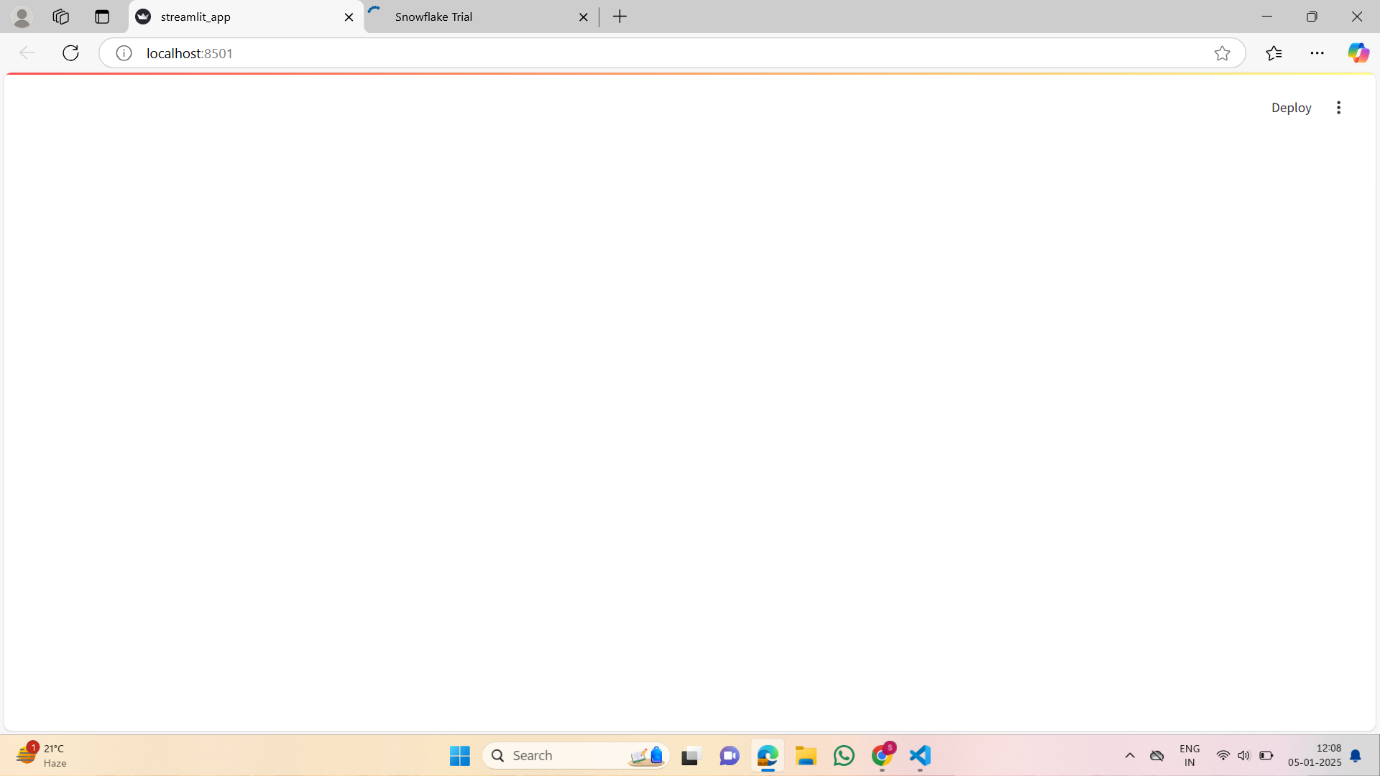
user\_input = st.text\_input("Enter your query:")

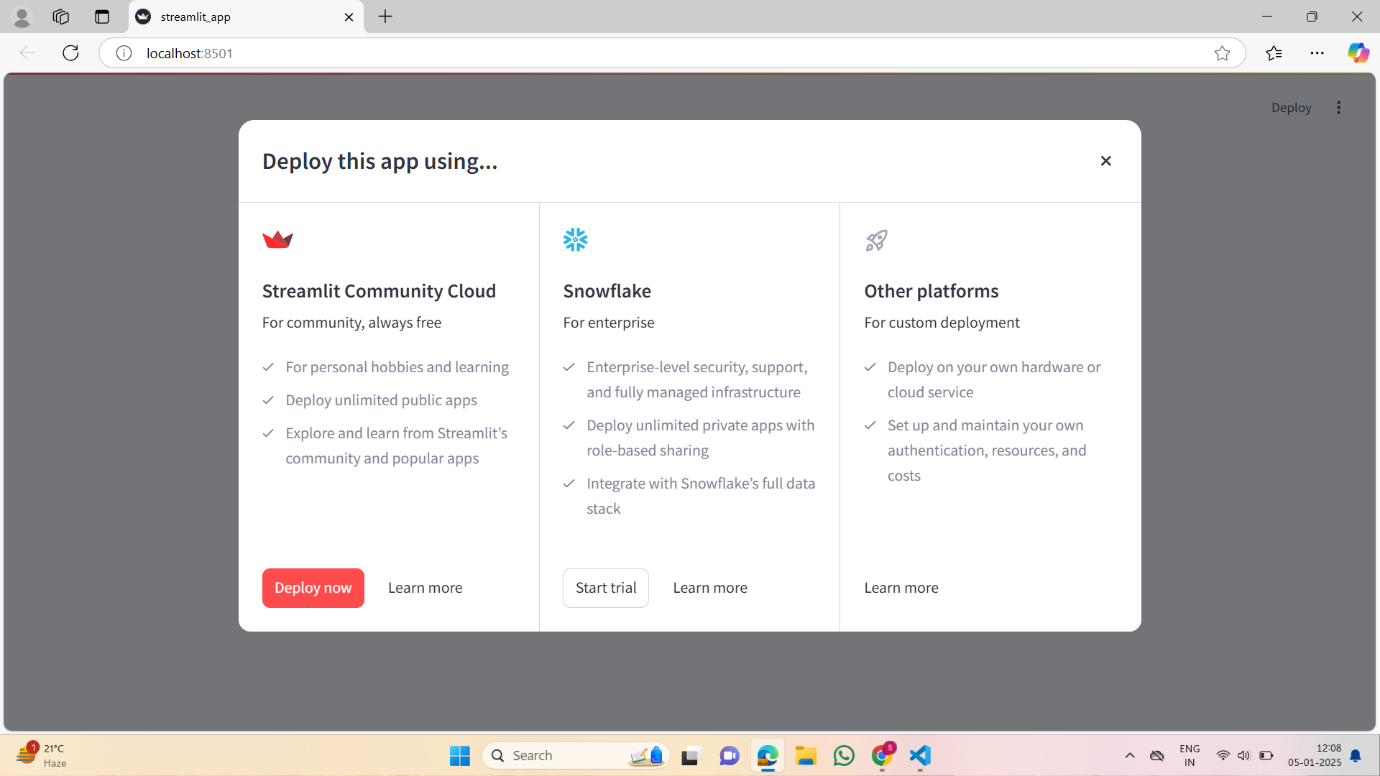
if user\_input:

response = fetch\_and\_generate(user\_input)

st.write(response)

****

****

****

1. **STRIDE Threat Modeling and Application for E-commerce Platforms**

import networkx as nx

import matplotlib.pyplot as plt

from pyvis.network import Network

import json

# Step 1: Define the Scope

assets = {

"hardware": ["Server 1", "Load Balancer", "Firewall"],

"software": ["AWS Management Console", "Database"],

"data": ["User Data", "Transaction Records"],

"users": ["Admins", "Customers", "Developers"]

}

print("Assets in the scope of this threat model:")

for category, items in assets.items():

print(f"{category.capitalize()}: {', '.join(items)}")

# Step 2: Classify Assets

critical\_assets = {

"High": ["Database", "Firewall", "Admin Panel"],

"Medium": ["Web Application", "Load Balancer"],

"Low": ["Customer Devices"]

}

print("\nAsset Classification Based on Criticality:")

for level, items in critical\_assets.items():

print(f"{level} Criticality: {', '.join(items)}")

# Step 3: Create a Visual Representation using NetworkX

graph = nx.DiGraph()

graph.add\_edges\_from([

("Customer", "Web Application"),

("Web Application", "Database"),

("Database", "Admin Panel"),

("Web Application", "Load Balancer"),

("Load Balancer", "Server"),

("Server", "Firewall")

])

plt.figure(figsize=(10, 8))

nx.draw(graph, with\_labels=True, node\_color='skyblue', node\_size=3000, font\_size=12, font\_weight="bold")

plt.title("Data Flow Diagram (DFD)")

plt.show()

# Step 4: Conduct Risk Analysis

trust\_levels = {

"Customer": "Low",

"Web Application": "Medium",

"Database": "High",

"Admin Panel": "High",

"Load Balancer": "Medium",

"Server": "High",

"Firewall": "Very High"

}

print("\nTrust Levels of Components:")

for component, level in trust\_levels.items():

print(f"{component}: {level} Trust")

# Step 5: Identify Threats

stride\_threats = {

"Spoofing": ["Customer credentials", "API access keys"],

"Tampering": ["Data in transit", "Database records"],

"Repudiation": ["Transaction logs"],

"Information Disclosure": ["Sensitive customer data"],

"Denial of Service": ["Web Application", "Load Balancer"],

"Elevation of Privilege": ["Admin access"]

}

print("\nSTRIDE Threats and Potential Targets:")

for threat, targets in stride\_threats.items():

print(f"{threat}: {', '.join(targets)}")

# Step 6: Automate Visualization using Pyvis

net = Network(height="750px", width="100%", bgcolor="#ffffff", font\_color="black")

# Add nodes and edges

nodes = ["Customer", "Web Application", "Database", "Admin Panel", "Load Balancer", "Server", "Firewall"]

for node in nodes:

net.add\_node(node, label=node)

edges = [

("Customer", "Web Application"),

("Web Application", "Database"),

("Database", "Admin Panel"),

("Web Application", "Load Balancer"),

("Load Balancer", "Server"),

("Server", "Firewall")

]

net.add\_edges(edges)

# Generate the visualization and display it in the browser

net.show("threat\_model.html")

# Step 7: Save and Iterate

threat\_model = {

"assets": assets,

"critical\_assets": critical\_assets,

"trust\_levels": trust\_levels,

"stride\_threats": stride\_threats

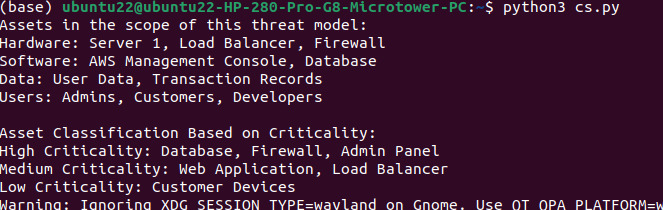
}

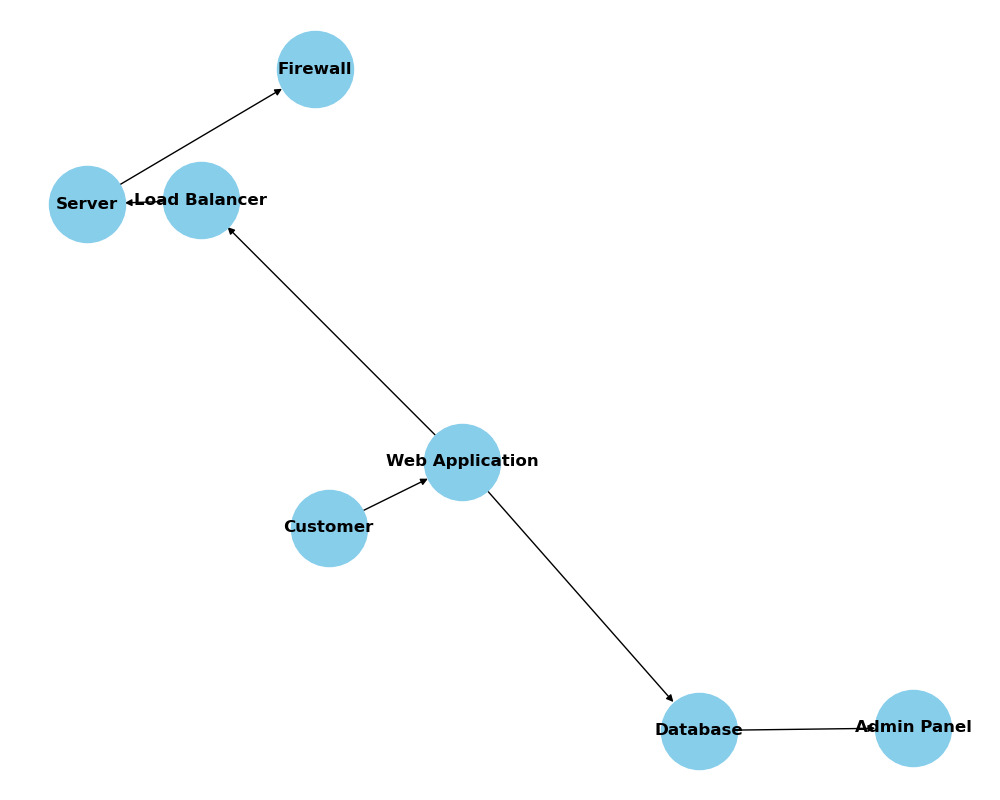
# Save to a JSON file

with open("threat\_model.json", "w") as f:

json.dump(threat\_model, f, indent=4)

print("\nThreat model saved as 'threat\_model.json'.")





1. Docker

Code

print("Hello, World!")

docker file

# Use an official Python runtime as a parent image

FROM python:3.8-slim

# Set the working directory in the container

WORKDIR /app

# Copy the current directory contents into the container at /app

COPY . /app

# Run app.py when the container launches

CMD ["python", "app.py"]

Output:

