Session 1: Virtual Environment, Git Workflow, and Selenium

1. Virtual Environment:

The session began with a discussion on the importance of using virtual environments in Python to isolate dependencies for each project.

- Creation and Activation:
    - Demonstrated how to create and activate a virtual environment using:
    - python -m venv myenv

For Windows, the activation command was:

myenv\Scripts\activate

- Managing Dependencies:
    - Explained how to install the necessary libraries within the virtual environment.

2. Git Workflow:

An overview of essential Git commands was provided, including git init, git add, git commit, git push, and git pull.

- GitHub Integration:
    - Showed how to push a project to GitHub and manage repositories effectively.

3. Selenium Automation with Python:

- Installation of Selenium and WebDriver:
    - Installed Selenium and Chrome WebDriver to automate web interactions.
- Simple Automation Script:
    - Demonstrated a basic Selenium script to automate actions in a browser:
    - from selenium import webdriver
    - driver = webdriver.Chrome()
    - driver.get("https://qxf2.com/selenium-tutorial-main")
    - name = driver.find_element(by="id", value="name")
    - name.send_keys("Qxf2")
- Element Interaction:

    - Explained how to locate elements on a webpage and perform actions such as sending keys using Selenium.

Session 2: API Testing with Postman and Python Requests Library

1. Postman API Testing:

The session introduced the basics of API testing using Postman.

- GET and POST Requests:
    - Demonstrated how to send GET and POST requests using Postman, and how to validate the responses.

2. API Testing with Python Requests Library:

- Using Python's Requests Module:
    - Showed how to perform API testing using Python's requests module.
    - Executed a series of API calls to a local server (http://127.0.0.1:5000/).
- Sample GET Request:
    - Demonstrated a sample GET request:
    - import requests
    - response = requests.get("http://127.0.0.1:5000/cars", auth=("qxf2", "qxf2"))
    - print(response.status_code)
    - print(response.json())
- Sample POST Request:
    - Showed how to send a POST request to add a new car:
    - response = my_session.post(url=base_url + 'cars/add', json={
    -     'name': 'Gwagon',
    -     'brand': 'Gwagon',
    -     'price_range': '90-200lacs',
    -     'car_type': 'sedan'
    - }, auth=(username, password))
- Assertions for API Response Validation:
    - Implemented assertions to ensure the correctness of API responses:

    - assert response.status_code == 201, f"Expected status 201, got {response.status_code}"
    - assert 'message' in response_content and 'successfully' in response_content['message'].lower(), "Car addition failed"
    - Confirmed that the car count increased after the POST request.

Conclusion:

Today's session provided an in-depth understanding of key concepts in automation and API testing. We covered:

- How to set up a virtual environment for managing project dependencies.
- The basics of Git for version control and collaboration.
- Automating web interactions with Selenium.
- Performing API testing using both Postman and Python's requests library.

The session was interactive, with hands-on examples and practical applications of assertions to validate API responses.