**Session1:**

**Setting Up and Testing a FastAPI Project with GitHub Actions**

1. Activate the Virtual Environment

Before proceeding, activate the virtual environment:

source ~/profile/scripts/activate {profile}

2. Navigate to the Project Directory

Move into the profile directory, which is already created and pushed to the remote GitHub repository:

cd ~/projects/profile

3. Add requirements.txt and test\_main.py

Ensure that both requirements.txt (for dependencies) and test\_main.py (for testing) are placed inside the profile directory.

4. Install Dependencies

Run the following command to install all required dependencies from requirements.txt:

pip install -r requirements.txt

5. Run Tests Using Pytest

Execute the test script using:

pytest test\_main.py

If errors occur, upgrade the necessary dependencies using:

pip install --upgrade fastapi

pip install httpx

Then, run the test again:

pytest test\_main.py

6. Commit and Push Changes to GitHub

Once the test is successful, add and commit the changes:

git add .

git commit -m "added requirements.txt and test\_main.py"

git push

7. Add a main.yml File for GitHub Actions

Create a main.yml file inside the .github/workflows/ directory. This file will define the CI/CD workflow for GitHub Actions.

8. Commit and Push the Workflow File

After adding the workflow file, stage, commit, and push the changes:

git add .

git commit -m "added yml file"

git push

9. View Workflows in GitHub Actions

Go to the Actions tab in your GitHub repository. All workflows will be displayed, showing their status and execution logs.

**Session2:**

**Deploying Project on Render and Setting Up Selenium for Test Automation**

1. Deploy the Project on Render

Ensure the FastAPI project is successfully deployed on Render following the previous setup instructions.

2. Deactivate Any Active Virtual Environment

Before setting up test automation, deactivate the currently active environment:

Move to the main directory:

cd ~

Deactivate the environment:

deactivate

3. Create a New Directory and Virtual Environment

Create a new directory named test\_automation:

mkdir test\_automation

Create a virtual environment inside test\_automation:

python -m venv ~/test\_automation

Activate the new environment:

source ~/test\_automation/scripts/activate {test\_automation}

4. Install Selenium

Once the new environment is activated, install Selenium using:

python -m pip install selenium

5. Open Python Interpreter in Git Bash

Launch the Python interactive shell:

python -i

6. Use Selenium WebDriver to Open a Website

Inside the Python shell, use Selenium to open qxf2.com/selenium-tutorial-main:

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 services")

7. Add CSS and XPath Extensions to Chrome

Install the CSS Selector and XPath Finder extension in Chrome.

Use this extension to get the CSS selector of elements by ID for Selenium automation.