LinkedIn

Upendhar

Single Point Solution

Software Developer Intern

Objectives

Manual Testing

Test Plan

Test Case

Test Case Review

Defect Report

AutomationTesting

Login Functionality

Folder Structure

Data Driven Testing

HTML Reports

Allure Reports

Mobile Testing

Android studio

VYsor

AVD & APK

CI/CD

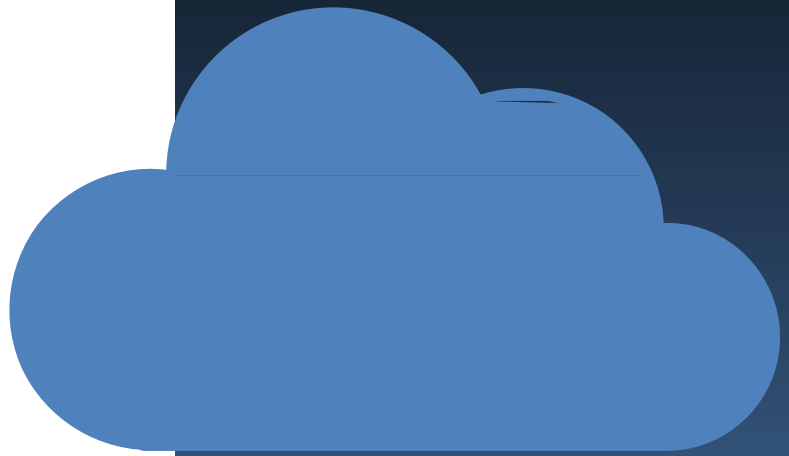
Jenkins

GIT

GIT Hub

Setups

- ❑ Selenium Python 4.26.1
- ❑ PyCharm 2024.3
- ❑ Allure Reports 2.29.0
- ❑ Unit Test 4.1.2
- ❑ PYTest 7.10.2
- ❑ Node JS v23.4.0



Manual Testing

Manual testing is a software testing approach where testers manually evaluate software or application quality without the help of automated testing tools or test scripts. Testers interact with the system like how an end user would identify bugs, defects, and issues in the software that create friction in user experience.

Test Deliverables

1.Test Plan

2.Test Case

3.Defect Report

Features

- Join now
- Sign in
- Sending connection Request
- Creating post
- Job search
- Add Skills
- URL customization

Test Plan Document

LinkedIn Portal

1. Test Plan Identifier

File Name: 1.0_LinkedIn_tplan_v1.0.doc

Location: Hyderabad

2. Document History

REV	Author	Edits	Date
1.0	Upendhar,Team Leader,Project Manager	Initial Draft	12/12/2024

3. Intorduction

LinkedIn is the world's largest professional networking platform. It is designed to connect professionals across industries, allowing them to showcase their skills, experience, and accomplishments while building relationships with others in their field.

Launched in **2003**, LinkedIn is widely used for professional networking, job searching, and sharing industry insights.

4. Features to be Tested

Medium priority features are features that have been stable and High Priority features are features that have been recently changed.

Feature To be Tested	Priority	Comments
Join now	High	
Sign in	High	
Sending Connection Request	Medium	
Creating Post	High	
Job Search	High	
Add Skills	Medium	
URL Customization	Low	

5.Features NOT to be Tested

Help contents will not be tested.
Installation and presentation will not be tested.
Support of the other architecture also will not be tested

6.Test Strategy

The test strategy for LinkedIn.1.0 will include manual testing with automated testing where time/need permits.Automated testing selenium with python will be used to test the basic Interface objects, keyboard and mouse naviga-

Test Plan Document

tion or any redundant tasks best translated to automation. Automated testing will be limited to a Win-64 platform because of Automation tool limitations.

- Sanity test will include an very short battery of tests to ensure the LinkedIn1.0 runs/launches and can connect to the back end database. Test execution will happen as individual component testing. These tests will be fully automated.
- Acceptance tests will include a short battery of tests that will only touch major features to ensure LinkedIn1.0 will operate smoothly at a basic level. These tests will be executed on Windows 11, Windows NT 4, The goal will be to automate all of these tests.
- Full Functional tests will include a comprehensive test of all features listed in this document.These test will include automation with time allowing.

Functionality Testing

- Profile management: Creating, updating, and deleting profiles.
- Connections: Sending, accepting, and rejecting connection requests.
- Messaging: Sending, receiving, and managing messages.
- Job search: Applying for jobs, saving jobs, and viewing recommendations.
- Content sharing: Posting updates, articles, and multimedia.

Usability Testing

The usability testing will be accomplished by verifying the information in each window are accurate. Menus, icons and toolbar functionality will be tested as applicable to the navigation and results panes. Multi Window Overlapping will be tested because product supports opening of multiple documents.

Robustness/Reliability Testing

Test the GUI for correct keyboard and mouse navigation, windows, menus, buttons, etc. These tests will include Keyboard only navigation of LinkedIn. 1.0, Mouse only navigation. Correctness and existence of warning/error dialog boxes. The correct/expected functionality of buttons, icons and menus

Stress Testing

Not applicable

7. Test Deliverables

Test plan	1.0_LinkedIn_tplan.doc
Test Specification	1.0_LinkedIn_tspec.doc
Test Automation Testing is Used	TBD
Test Checklist	TBD
Test Logs	TBD
Test tools	Selenium Suite,JIRA

8. Testing Tasks & Resource Needs

Use of MS ACCESS database.
Write Manual Test plan.
Write Test Scenario.
Write Test Cases
Setup workspace.
Develop Test script.

Test Plan Document

Intel: Intel ~400mz, with 96mb* of Ram for NT workstation client.
Intel ~400mz, with 64mb *of Ram for 98 workstation client.
(* Automation needs an additional 128MBs to function)

9.Schedule

Test Documentation

Document	*Development Time*		*Sign-off Date*	*Dependencies*
	Person-Days	*Completion Date*		
Test Plan	1 day	Today	09/12/24	Functional Spec. Review
Test Specifications	1days			Test plan, Functional Spec, Review
Test Logs	TBD			
Test Summary Report	TBD			
Checklist	TBD			

Test Development

Deliverable	*Development Time*		*Dependencies*
	Person-Days	*Completion Date*	
Functional Test Suite	1		
Automation build			

Test Execution

Test Activity	*Execution Time (Person-Days)*			*Dependencies*
	Alpha	Beta	FCS	
Acceptance	1 Days	1 Days	1 Days	
Functional	Functionality	1	1		
	Usability	All High to Medium test cases			
	Robustness				
	Stress				
	Multi Inst.				
Regression					
Documentation Review		1?	1?	1?	
Total Test Cycle		1	1	1	

10.Issues/Risk and Contingencies

Risks:

Environment Instability: Delays in setting up the test environment could impact the schedule- VM is not working port no.-5103

Resource Availability: Limited availability of testers could delay test execution.

Late Requirement Changes: Changes in requirements during the testing phase could lead to additional testing efforts.

Contingencies:

Backup Resources: Have backup resources ready to join the testing effort if required.

Flexible Schedule: Allocate buffer time in the schedule to accommodate unexpected delays.

Test Plan Document

Regular Communication: Hold daily stand-up meetings to quickly address any issues that arise

11.Approvals

POSITION *Approval required	NAME	SIGNATURE	REVIEWED	APPROVAL
Tech Lead	Upendhar		Raghavendra BN	Raghavendra BN
Project Manager	Upendhar		Raghavendra BN	Raghavendra BN
Client Approval	Upendhar		Raghavendra BN	Raghavendra BN

Presented By:

A.Upendhar

End of document

Test Case Document

Requirement ID	Scenario ID	Test Case ID	Test Case Title	Precondition	Test Data	Test Steps	Expected Result	Actual result	Status
S1_001	T_001	T_001 A	Verify Join now functionality	User Should have valid Email Id	URL- https://www.linkedin.com/signup Email - aaa@gmail.com Password - ****** First name - Upendhar Last name - A	1.open the chrome wrobser 2.Enter the URL 3.click join now hyperlink 4.Enter the Email 5.Enter the password 6.click agree button 7.Enter the first name and last name 8.Solve the puzzle for security verification 9.Enter the Location 10.click next Button 11.Enter most Recent Job title 12.click continue Button 13.Enter the code sent to the Email 14.Click Agree and continue check box 15.click Next 16.Enter the job title and job location 17.Click the I'm open to remote work Check box 18.click the next button 19.Account has been created Successfully	User's account should crated Successfully		Passed
S2_001	T_002	T_002 A	Verify Sign in functionality using Email or Phone number	User is registered in LinkedIn	URL- https://www.Linkedin.com/login Email - aaa@gmail.com Password - ******	1.Open the chrome browser 2.Enter the URL 3.Click on the sign in button in the top right corner 4.Enter the Email or phone number	User is successfully logged in and can access their account		

4	S3_001	T_003	T_003 A	Verify Job Search Functionality	User should Signed into LinkedIn	URL - https://www.Linkedin.com/feed Email - aaa@gmail.com	1. Open the Chrome browser 2. Enter the URL 3. Sign in to the LinkedIn account 4. Click the job tab 5. Enter the location	Job Search done Successfully and results shown		Passed
5	S4_001	T_004	T_004 A	Verify Creating Post	User should Signed into LinkedIn	URL - https://www.Linkedin.com/feed Email - aaa@gmail.com	1.Open the chrome browser 2.Enter the URL 3.Sign in to the LinkedIn account 4.Scroll until you find "start a post by writing with AI" 5.Click the options 6.Create a post by typing the content and add attachment if needed 7.click post button 8.post Created Successfully	User should Have created post Successfully		Passed
6	S5_001	T_005	T_005 A	Verify Saved Item	User should Signed into LinkedIn	URL - https://www.Linkedin.com/my.items	1.Open the chrome browser 2.Enter the URL 3.Sign in to LinkedIn account 4.Go to the feed Section 5.Click "Saved items" in the left corner 6.You can find the saved items in "My items" section	User can find saved items Successfully		Passed

Test Case - Peer Review

Raghavendra Bn <raghu.astepahead@gmail.com>

Dec 10, 2024, 2:42 PM (1 day ago)

to me

Dear Upendhar Ambati,

I hope this message finds you well. I wanted to take a moment to thank you for sharing the documents related to the MakeMyTrip project. I have received the Test Cases, Bug Report, and Test Plan Document that you have prepared. I truly appreciate the effort and detail you have put into compiling these documents, as they are crucial for the project's success.

I will be reviewing the documents thoroughly to ensure that all aspects of the project are covered and that the quality standards are met. Once I have completed my review, I will provide you with detailed feedback and any suggestions for improvement if necessary. If I come across any points that require clarification or further input from your side, I will make sure to reach out to you promptly.

Thank you once again for your hard work and dedication to this project. Your contributions are highly valued, and I look forward to collaborating with you further to achieve our goals.

Good To GO

Best regards,
Raghavendra Bn

Defect Report

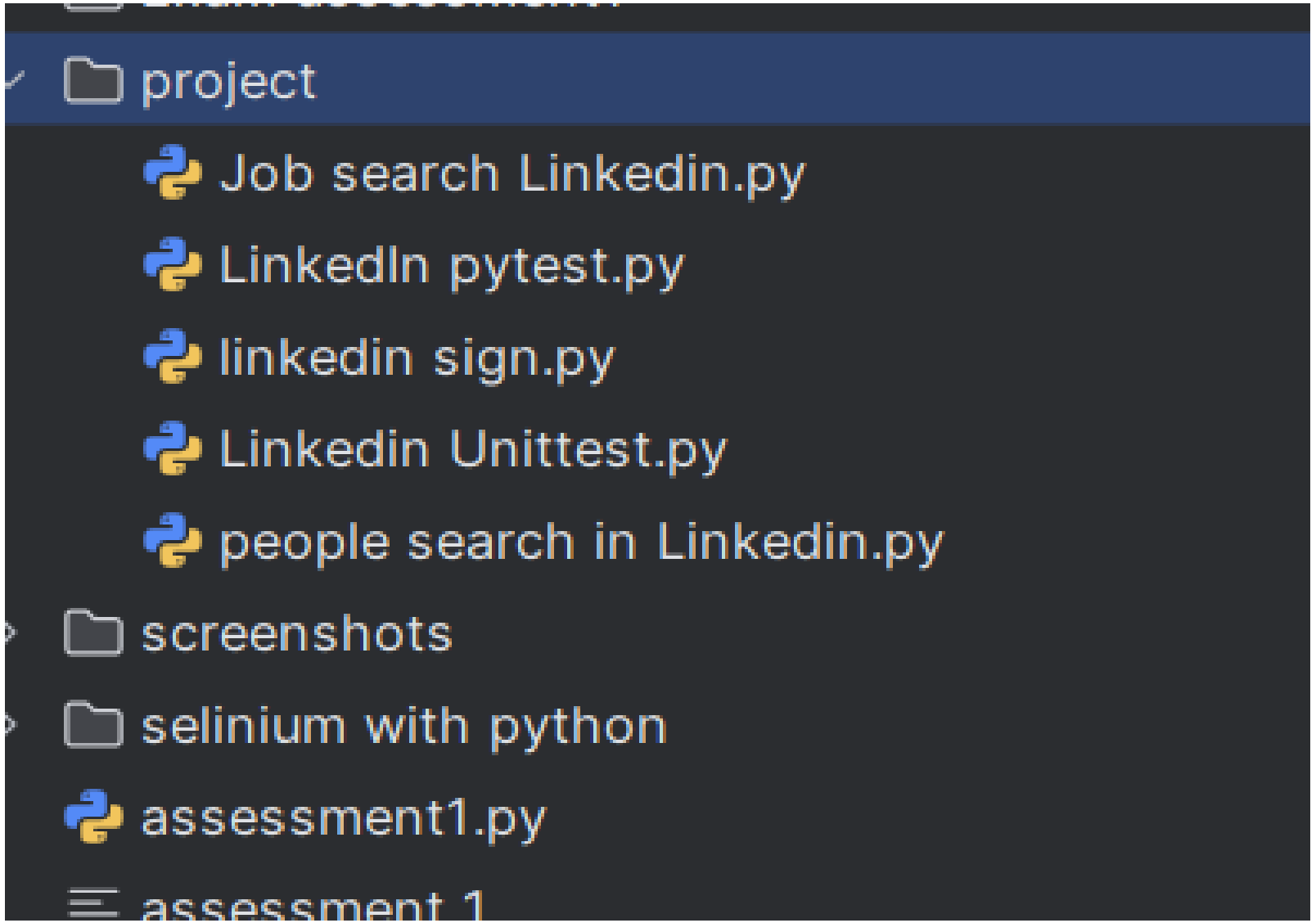
	A	B	C	D	E	F	G	H
1								
2		**Make LinkediIn defect report- Cycle 1 Testing**						
3		**Prepared By**	Upendhar ambati					
4								
5								
6	**Defect ID**	**Defect Description**	**Steps to reproduce**	**Severity**	**Status**	**Created by**	**Assigned to**	**Screenshot**
7	**1**	"The job application form does not submit when using the LinkedIn mobile app on Android devices."	1.Login to the LinkedIn mobile app. 2.Navigate to a job posting. 3.Fill out the application form and click "Submit"	1-Medium	Open	Upendhar	RaghavendraBN	
8	**2**	"When typing in the search bar on Safari, no suggestions appear, even for common queries."	1.Open LinkedIn on Safari. 2.Start typing a keyword in the search bar.	1- Medium	Open	Upendhar	RaghavendraBN	
9	**3**	"The 'Who's viewed your profile' section displays fewer views than the email notification reports."	Open LinkedIn. Navigate to "Who's viewed your profile."	1-Medium	Open	Upendhar	RaghavendraBN	
10								

Automation Testing & Page Object Model

Automation testing involves using software tools to execute pre-scripted tests on a software application before it is released into production. This helps in identifying defects, ensuring the software behaves as expected, and improving efficiency by reducing manual testing efforts.

The **Page Object Model (POM)** is a design pattern in automation testing that creates an object repository for web UI elements. It enhances test maintenance and reduces code duplication by encapsulating the page elements and actions in separate classes, making the tests more readable and easier to manage.

Folder Structure



sign.py

linkedin sign.py

```python
from selenium import webdriver
from selenium.webdriver import Keys
from selenium.webdriver.common.by import By
import time
driver = webdriver.Chrome()
driver.get("https://www.linkedin.com/signup/cold-join")
driver .maximize_window()
driver.find_element(By.ID, value: "email-or-phone").send_keys("208r1a05i2cse@gmail.com")
driver.find_element(By.NAME, value: "password").send_keys("Lasmaiah@5014")
driver.find_element(By.XPATH, value: "/html/body/div[1]/main/div[2]/div[1]/form/div[4]/button")
time.sleep(15)
actual_title = driver.title
expect_title = "Sign in to your account, for the best experience"
if actual_title==expect_title:
    print("login is successful......well done python")
else:
    print("login sucessfully.....very god my boy")
```

Job search.py

Job search Linkedin.py

```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys

# Set up the webdriver
driver = webdriver.Chrome()
driver.get("https://www.linkedin.com/signup/cold-join")

# Login to LinkedIn
username = "208r1a05i2cse@gmail.com"
password = "Lasmaiah@5014"

username_input = driver.find_element(By.ID, value: "email-or-phone").send_keys("208r1a05i2cse@gmail.com")
password_input = driver.find_element(By.NAME, value: "password").send_keys("Lasmaiah@5014")
login_button = driver.find_element(By.XPATH, value: "//button[text()='Sign in']")
login_button.click()

# Navigate to job search page
job_search_link = driver.find_element(By.LINK_TEXT, value: "Jobs")
job_search_link.click()

# Enter job search keywords
job_search_input = driver.find_element(By.XPATH, value: "//input[@aria-label='Search by keyword']")
job_search_input.send_keys("Software Engineer")

# Enter location
location_input = driver.find_element(By.XPATH, value: "//input[@aria-label='Search by location']")
location_input.send_keys("New York")

# Click on search button
search_button = driver.find_element(By.XPATH, value: "//button[text()='Search']")
search_button.click()

# Verify job search results
job_search_results = driver.find_element(By.XPATH, value: "//h3[text()='Job search results']")
print("Job search results found")

# Close the browser
driver.quit()
```

Connectppl.py

people search in Linkedin.py

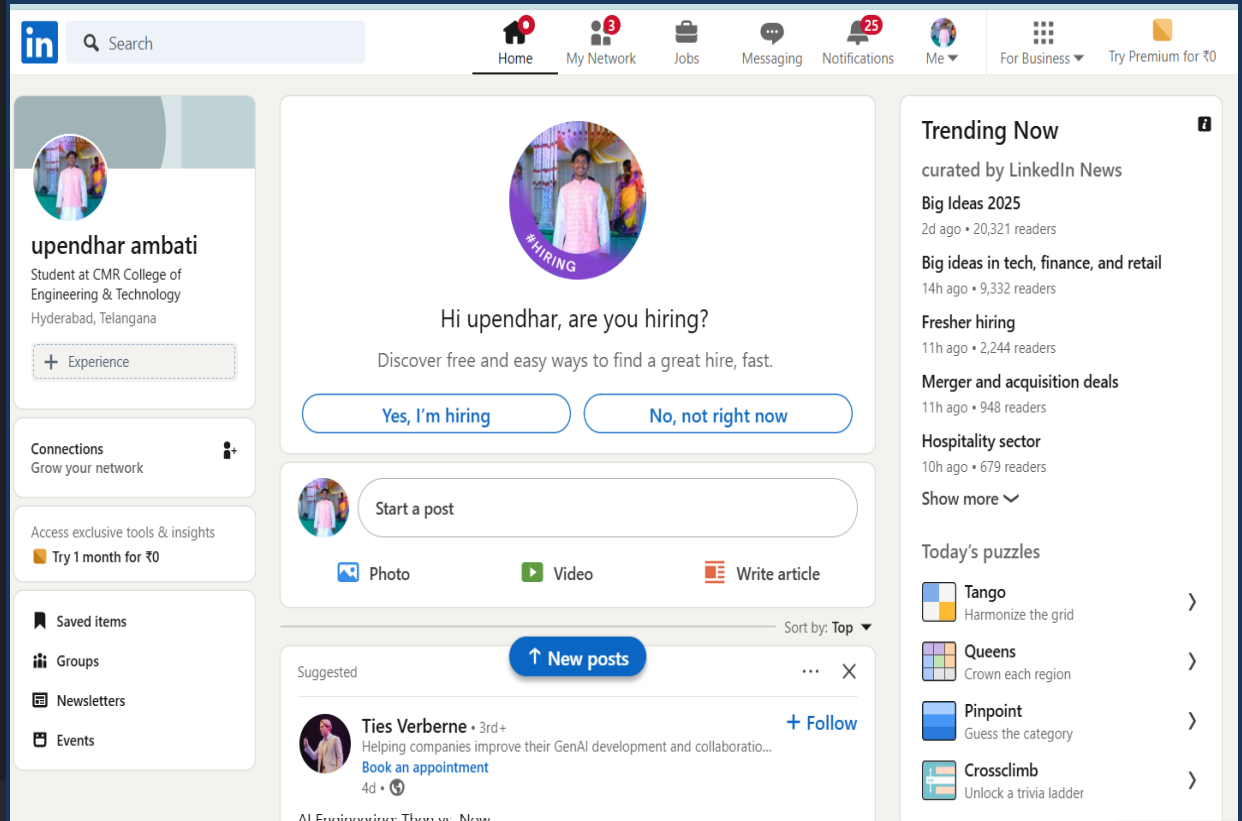
```python
from selenium import webdriver
from selenium.webdriver.common.by import By
import time
# Set up the webdriver
driver = webdriver.Chrome()
driver.get("https://www.linkedin.com/signup/cold-join")

# Login to LinkedIn
username = "208r1a05i2cse@gmail.com"
password = "Lasmaiah@5014"

username_input = driver.find_element(By.ID, value: "email-or-phone").send_keys("208r1a05i2cse@gmail.com")
password_input = driver.find_element(By.NAME, value: "password").send_keys("Lasmaiah@5014")
login_button = driver.find_element(By.XPATH, value: "//button[text()='Sign in']")
login_button.click()
# Navigate to people search page
people_search_link=driver.find_element(By.LINK_TEXT, value: "People")
people_search_link.click()
```

```python
# Enter people search keywords
people_search_input = driver.find_element(By.XPATH, value: "//input[@aria-label='Search by name, keyword, or title']")
people_search_input.send_keys("Jyothika")

# Click on search button
search_button = driver.find_element(By.XPATH, value: "//button[text()='Search']")
search_button.click()

# Verify people search results
people_search_results = driver.find_element(By.XPATH, value: "//h3[text()='People search results']")
print("People search results found")
# Close the browser
driver.quit()
```

Sign Test.py

linkedin sign.py

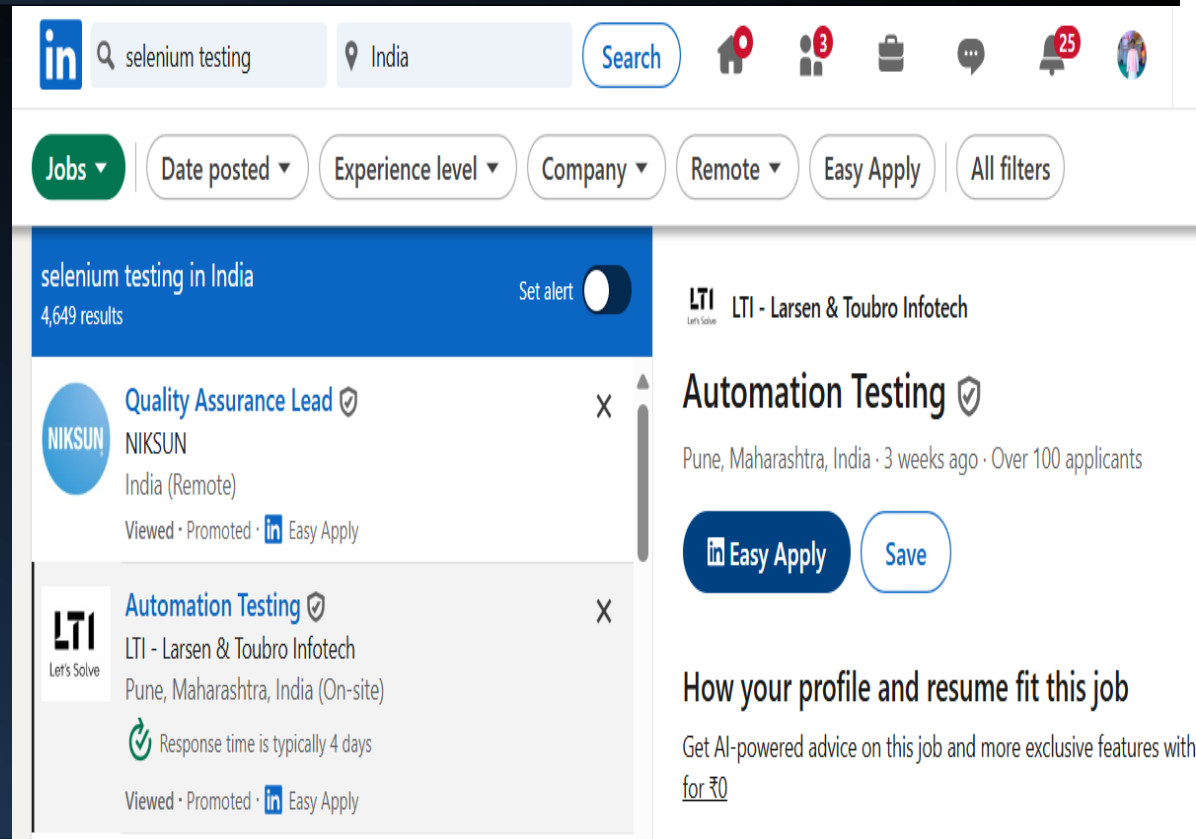
```python
from selenium import webdriver
from selenium.webdriver import Keys
from selenium.webdriver.common.by import By
import time
driver = webdriver.Chrome()
driver.get("https://www.linkedin.com/signup/cold-join")
driver .maximize_window()
driver.find_element(By.ID, value: "email-or-phone").send_keys("208r1a05i2cse@gmail.com")
driver.find_element(By.NAME, value: "password").send_keys("Lasmaiah@5014")
driver.find_element(By.XPATH, value: "/html/body/div[1]/main/div[2]/div[1]/form/div[4]/button")
time.sleep(15)
actual_title = driver.title
expect_title = "Sign in to your account, for the best experience"
if actual_title==expect_title:
    print("login is successful......well done python")
else:
    print("login sucessfully.....very god my boy")

```



Job searchTest.py

Job search Linkedin.py

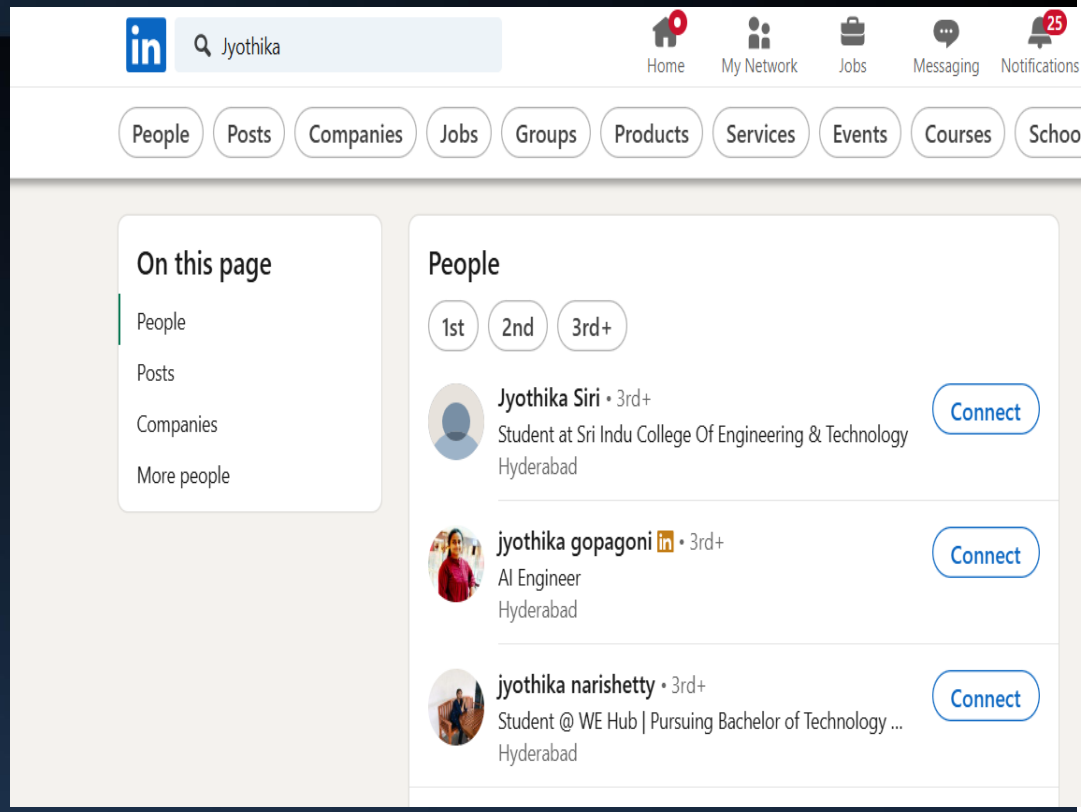
```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys

# Set up the webdriver
driver = webdriver.Chrome()
driver.get("https://www.linkedin.com/signup/cold-join")

# Login to LinkedIn
username = "208r1a05i2cse@gmail.com"
password = "Lasmaiah@5014"

username_input = driver.find_element(By.ID, value: "email-or-phone").send_keys("208r1a05i2cse@gmail.com")
password_input = driver.find_element(By.NAME, value: "password").send_keys("Lasmaiah@5014")
login_button = driver.find_element(By.XPATH, value: "//button[text()='Sign in']")
login_button.click()

# Navigate to job search page
job_search_link = driver.find_element(By.LINK_TEXT, value: "Jobs")
job_search_link.click()
```



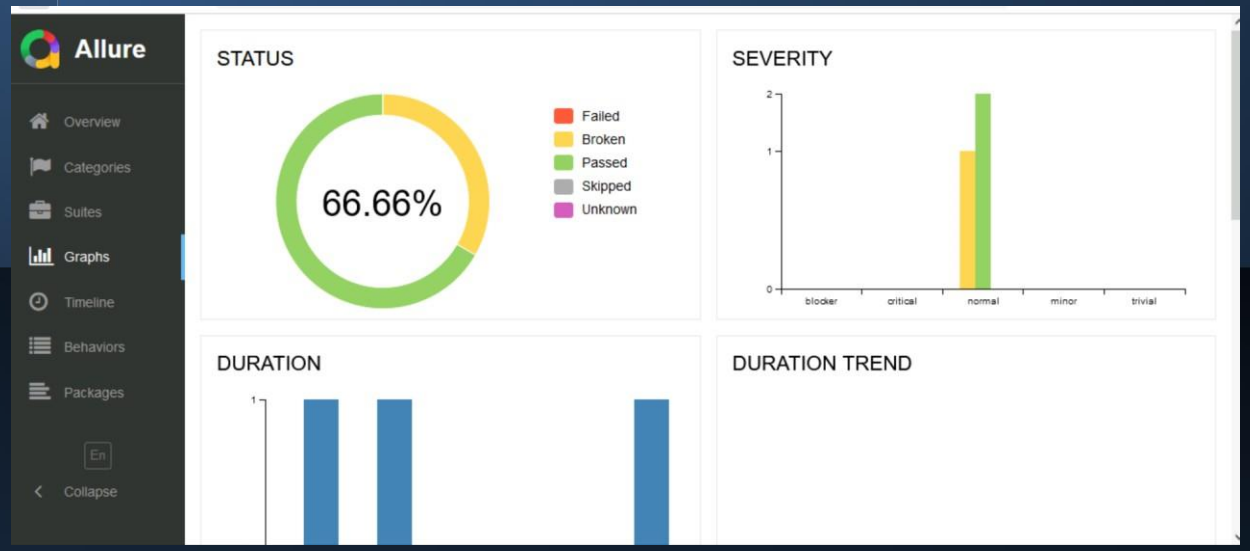
Connect ReqTest.py

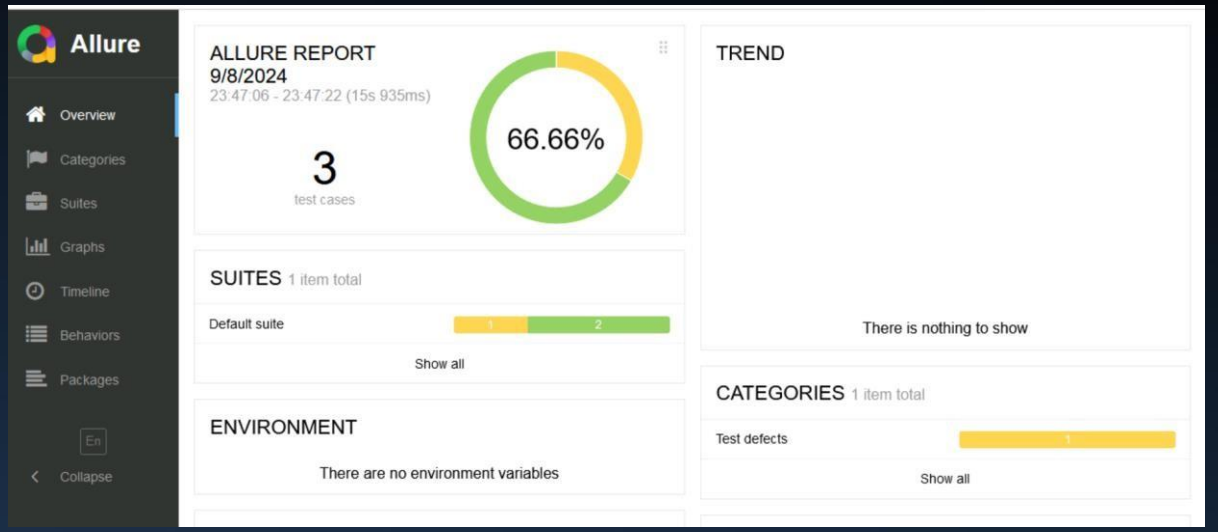
people search in Linkedin.py

```python
from selenium import webdriver
from selenium.webdriver.common.by import By
import time
# Set up the webdriver
driver = webdriver.Chrome()
driver.get("https://www.linkedin.com/signup/cold-join")

# Login to LinkedIn
username = "208r1a05i2cse@gmail.com"
password = "Lasmaiah@5014"

username_input = driver.find_element(By.ID, value: "email-or-phone").send_keys("208r1a05i2cse@gmail.com")
password_input = driver.find_element(By.NAME, value: "password").send_keys("Lasmaiah@5014")
login_button = driver.find_element(By.XPATH, value: "//button[text()='Sign in']")
login_button.click()
# Navigate to people search page
people_search_link=driver.find_element(By.LINK_TEXT, value: "People")
people_search_link.click()
```



Allure Report





Mobile Testing

Mobile Automation Testing is the process of using automation tools and frameworks to test mobile applications for functionality, usability, performance, and reliability. It involves creating automated test scripts to validate the behavior of a mobile app on various platforms (e.g., Android, iOS) and devices (e.g., smartphones, tablets, emulators, simulators).

Key Aspects of Mobile Automation Testing

1. **Platforms**:

- Android
- iOS

2. **Application Types**:

  - Native Apps: Built specifically for a platform using platform-specific tools (e.g., Swift for iOS, Kotlin for Android).
  - Web Apps: Accessed via a mobile browser.
  - Hybrid Apps: Combine elements of both native and web apps.

Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android app development, created by Google. It is built on JetBrains' IntelliJ IDEA software and is designed specifically for creating Android applications. It provides developers with tools and features to write, debug, test, and deploy Android apps efficiently.

Key Features:

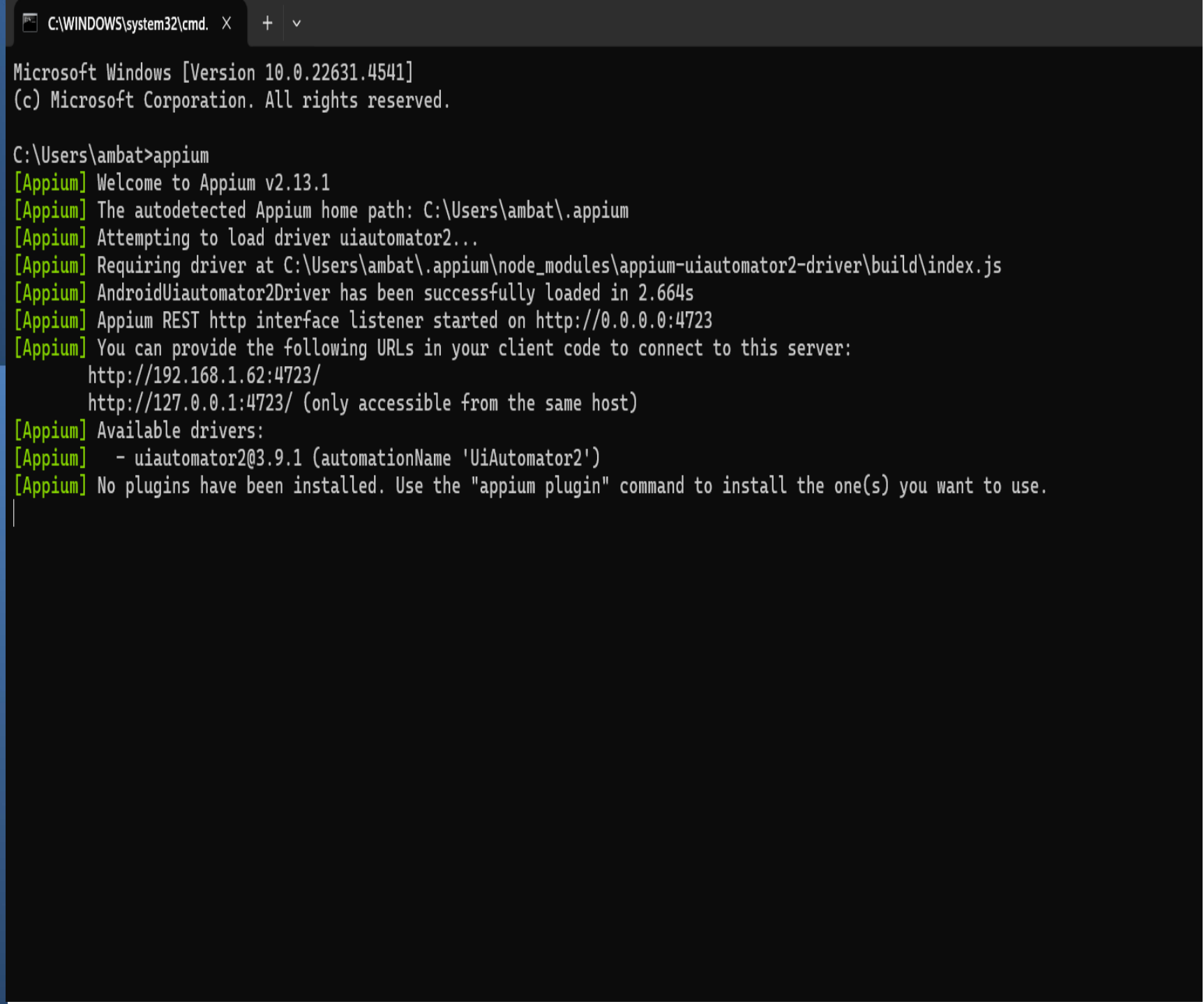
1.Code Editor: Offers advanced code completion, refactoring, and syntax highlighting for Kotlin, Java, and C++.

2.Layout Editor: Includes a drag-and-drop interface for designing app layouts with real-time preview functionality.

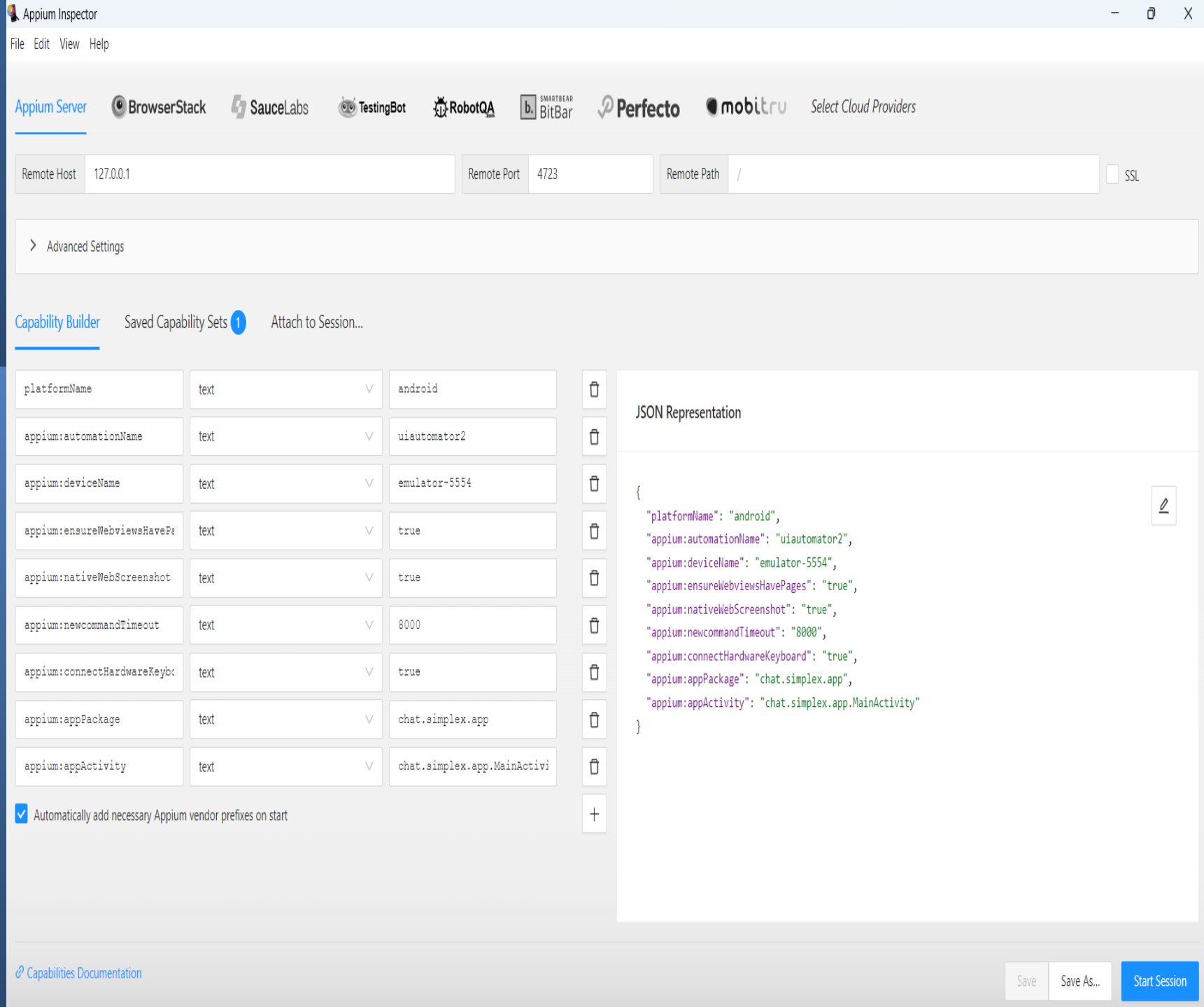
3.Emulator: A built-in Android Emulator allows developers to test apps on virtual devices with various configurations.

4.Build System: Uses Gradle for managing project dependencies and building APKs.

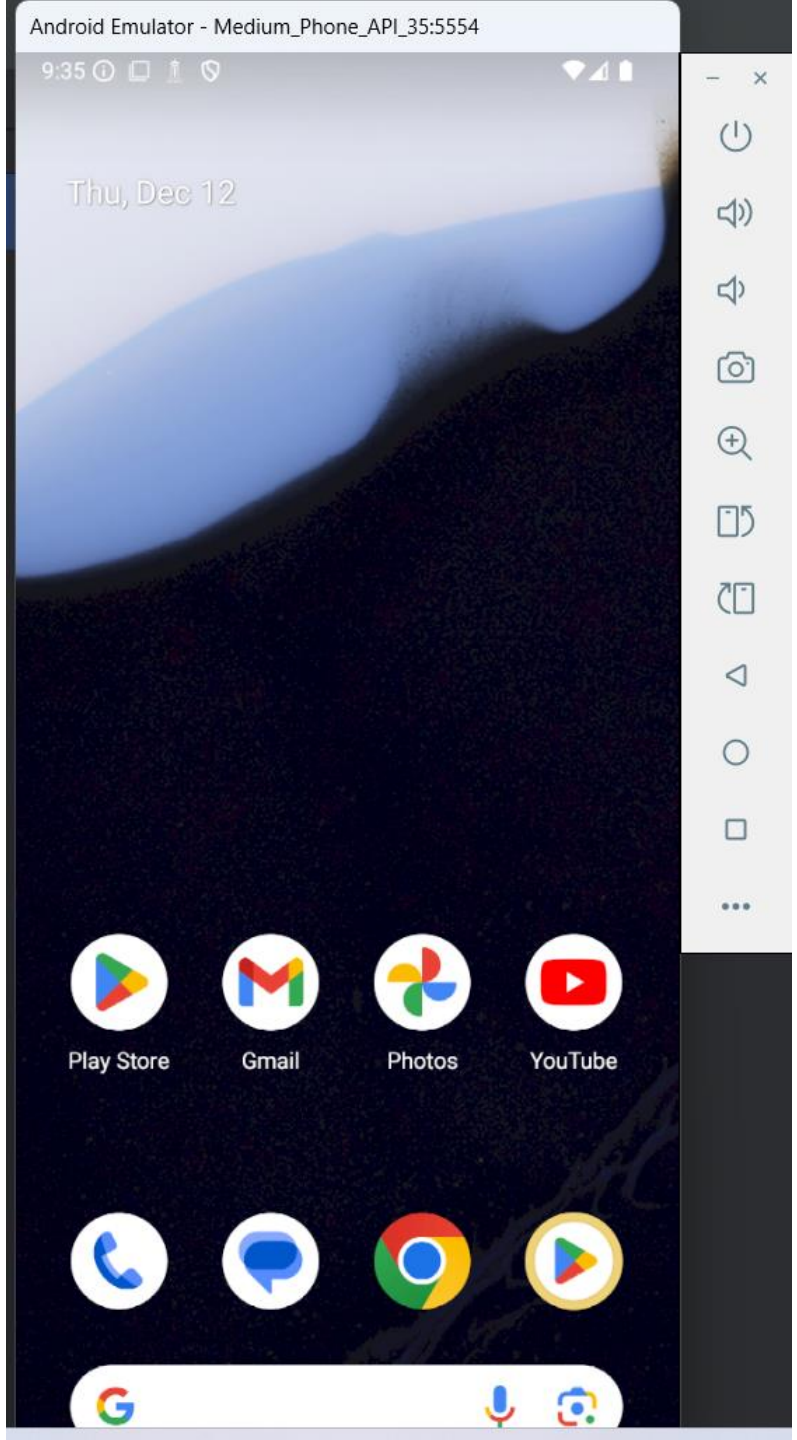
lunching Appium Server



Appium Inspector



Android Emulator



Continuous Integration(CI):

Continuous Integration (CI) is a software development practice where developers frequently integrate code changes into a shared repository. Each integration is automatically verified by building the application and running automated tests to detect issues early. CI aims to improve software quality and reduce the time required to deliver updates by catching integration errors as soon as they occur.

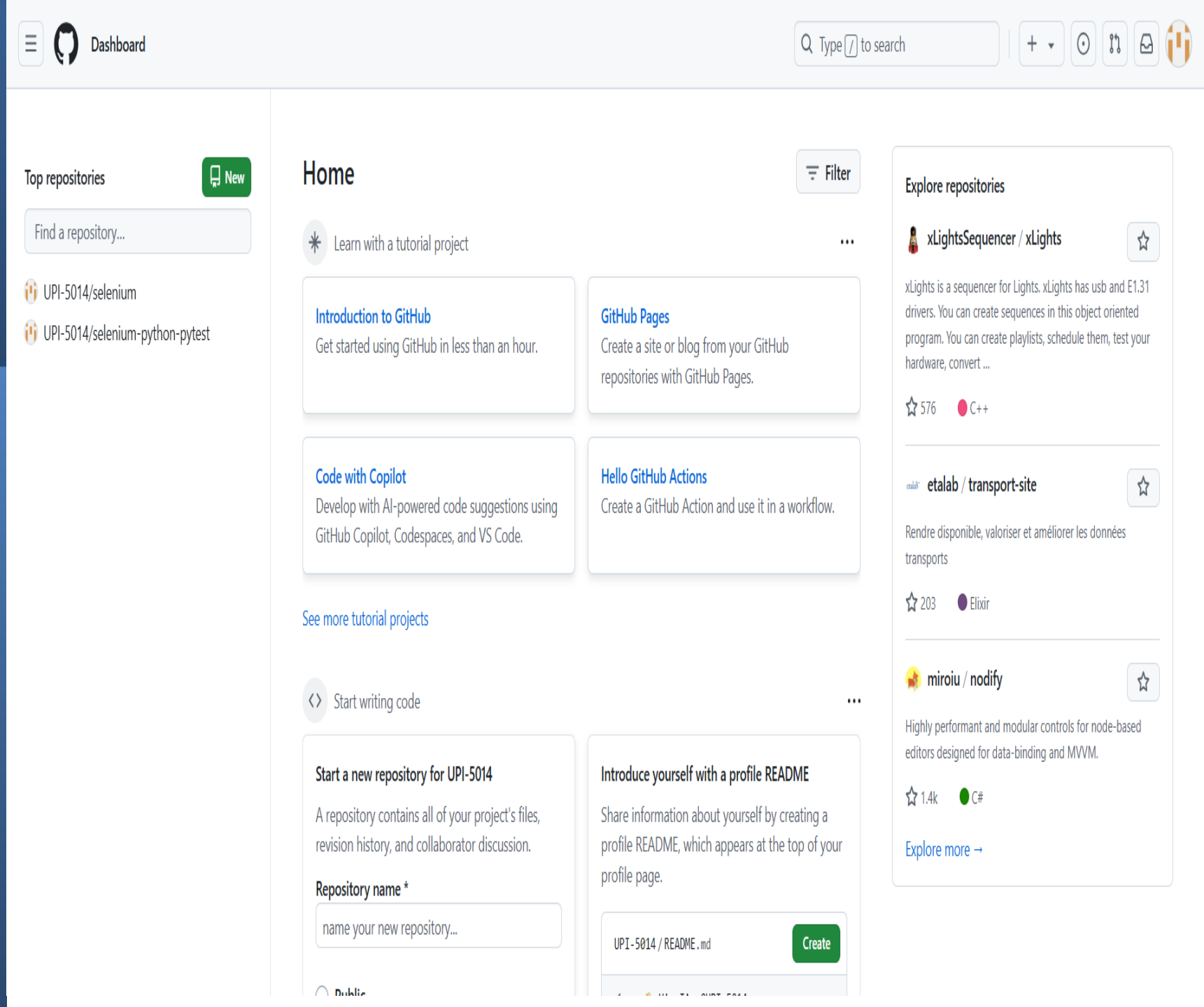
Key Components of CI:

1. **Version Control System (VCS):** Code changes are pushed to a shared repository (e.g., GitHub, GitLab, or Bitbucket).
2. **Build Automation:** A CI server (like Jenkins, GitHub Actions, or CircleCI) triggers automated builds whenever new code is committed.
3. **Automated Testing:** Unit, integration, and sometimes functional tests are executed to validate the new changes.
4. **Feedback:** Developers receive immediate feedback on the status of their code (build success, test results, or errors).

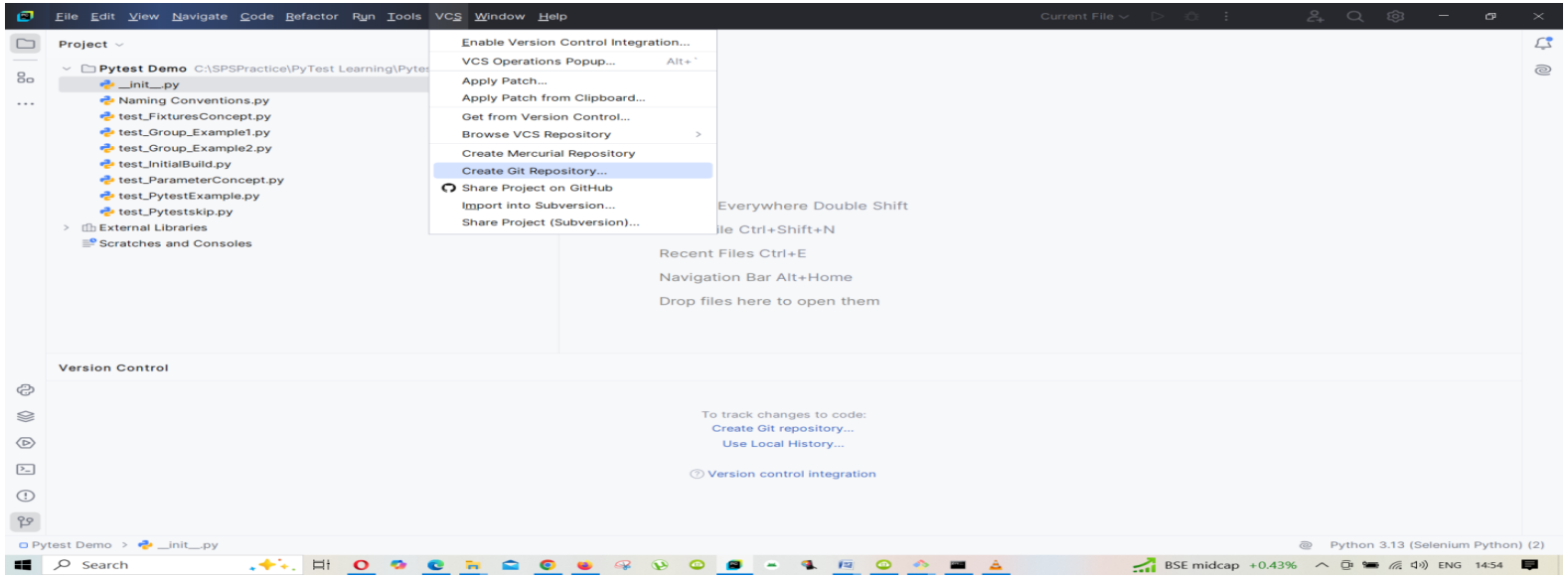
Continuous Development

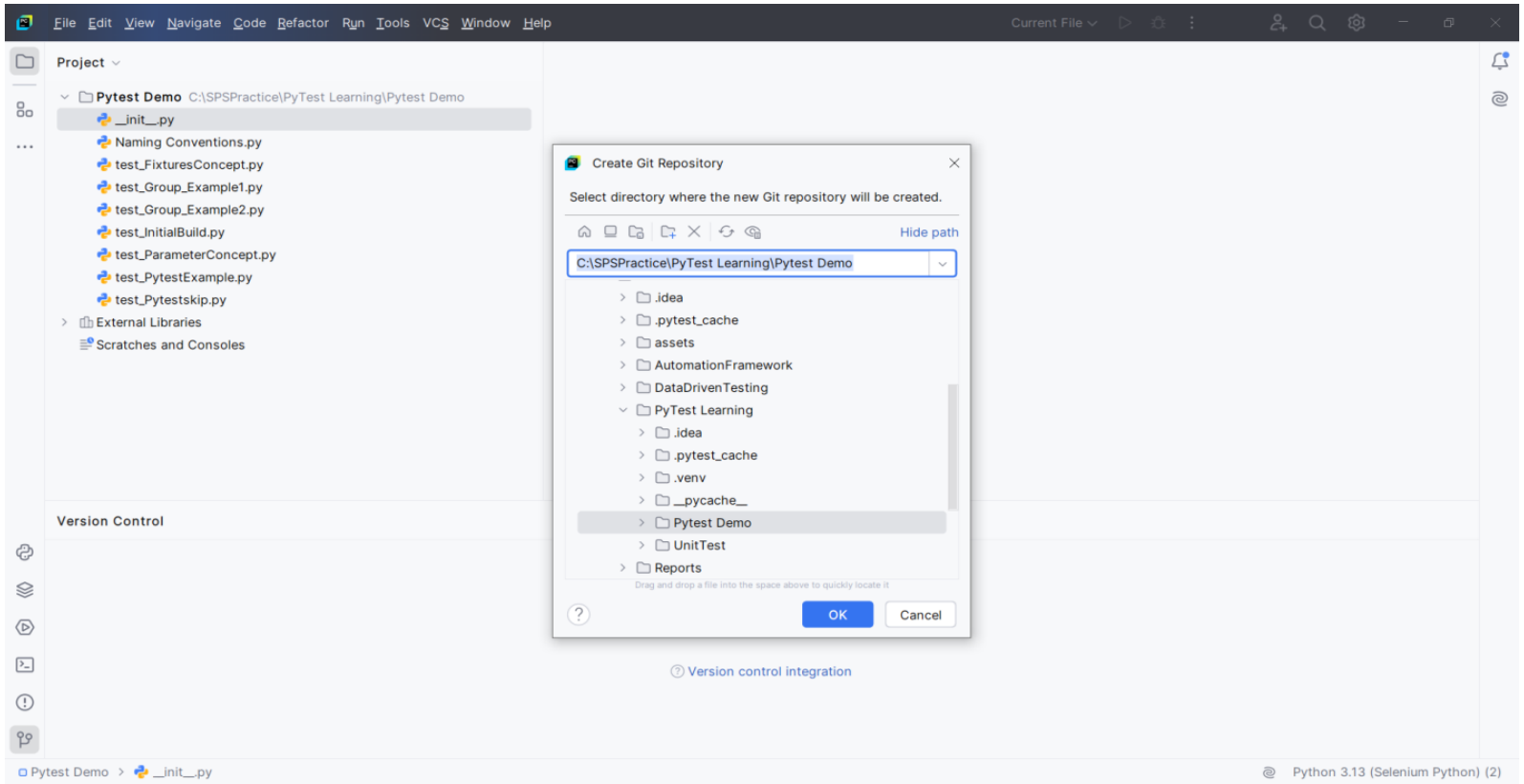
Continuous Development is a software engineering approach that extends **Continuous Integration (CI)** to include **Continuous Delivery (CD)** and sometimes **Continuous Deployment**. It focuses on automating the entire software delivery pipeline, from code integration to testing, deployment, and feedback collection. The goal is to enable faster, more reliable software updates and ensure new features or fixes reach users quickly and efficiently.

Lunching Github

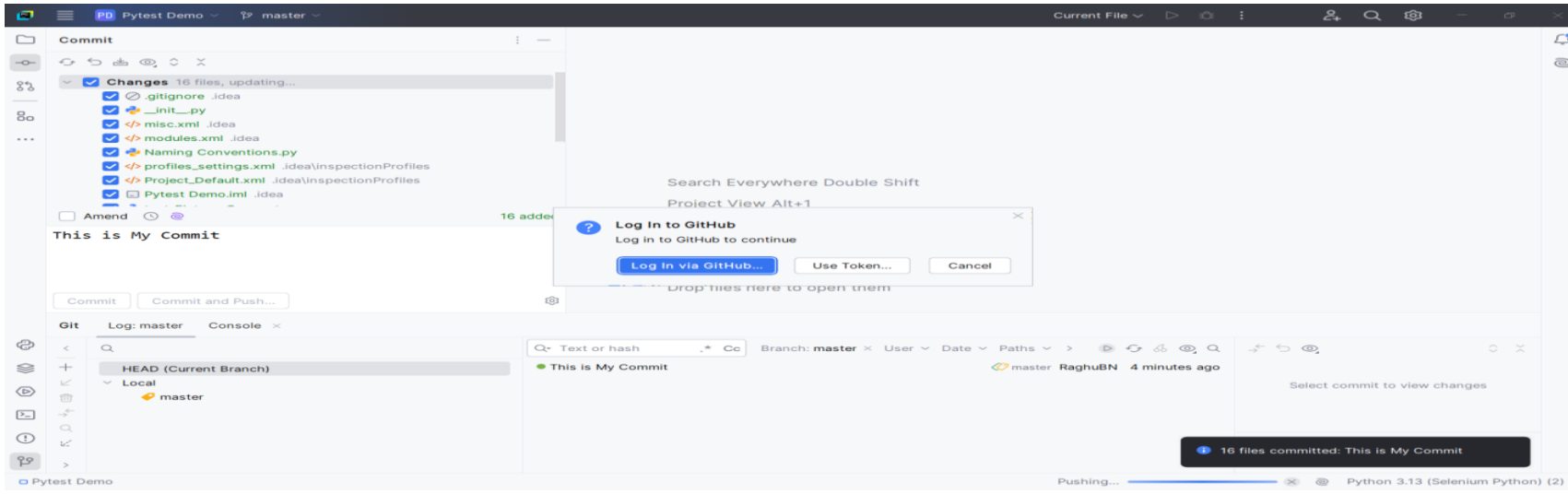


Push the code to Github



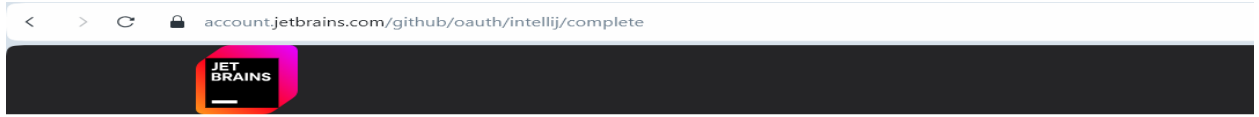


Push the code to Github



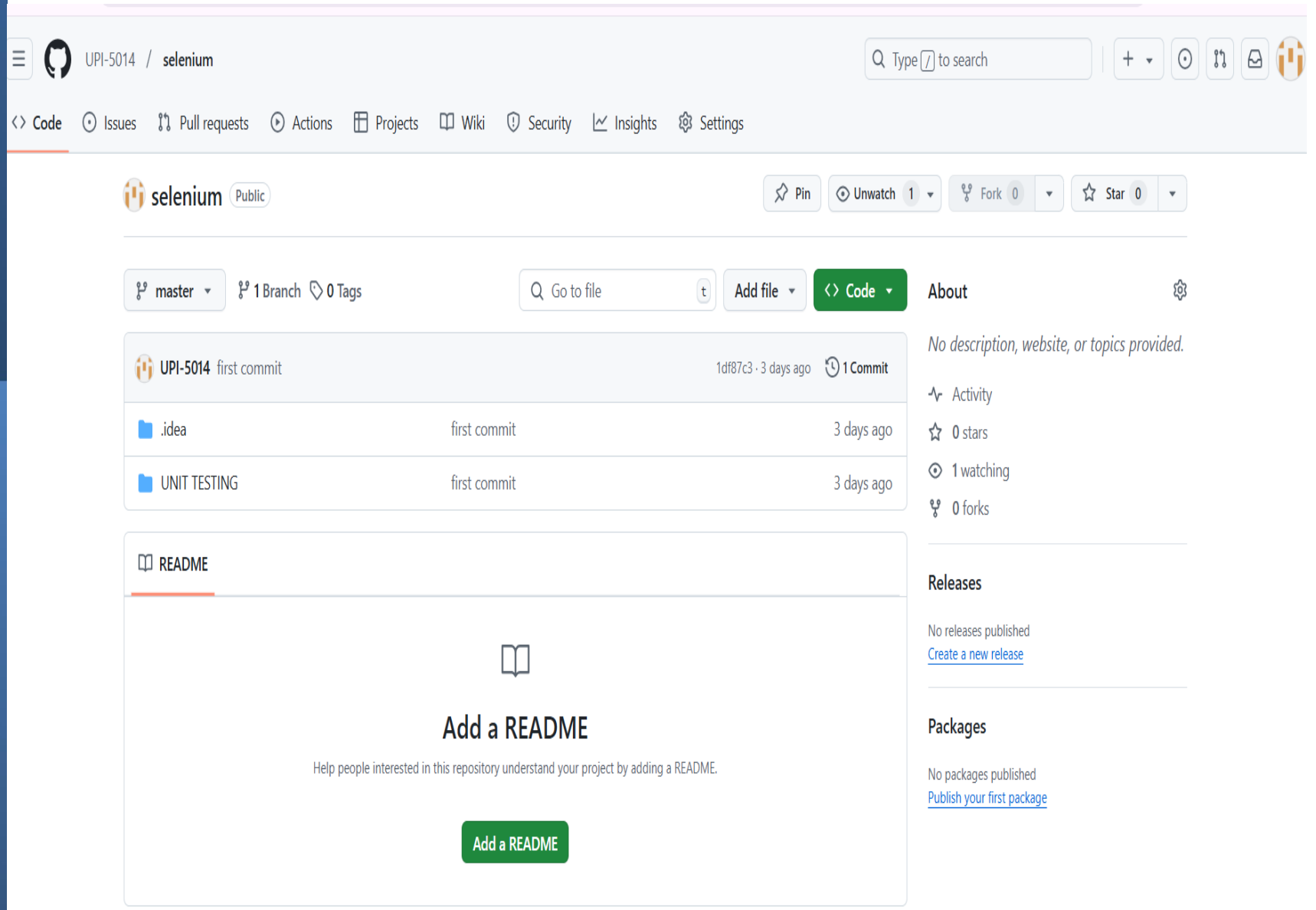
Please continue only if this page is opened from a JetBrains IDE.

Authorize in GitHub

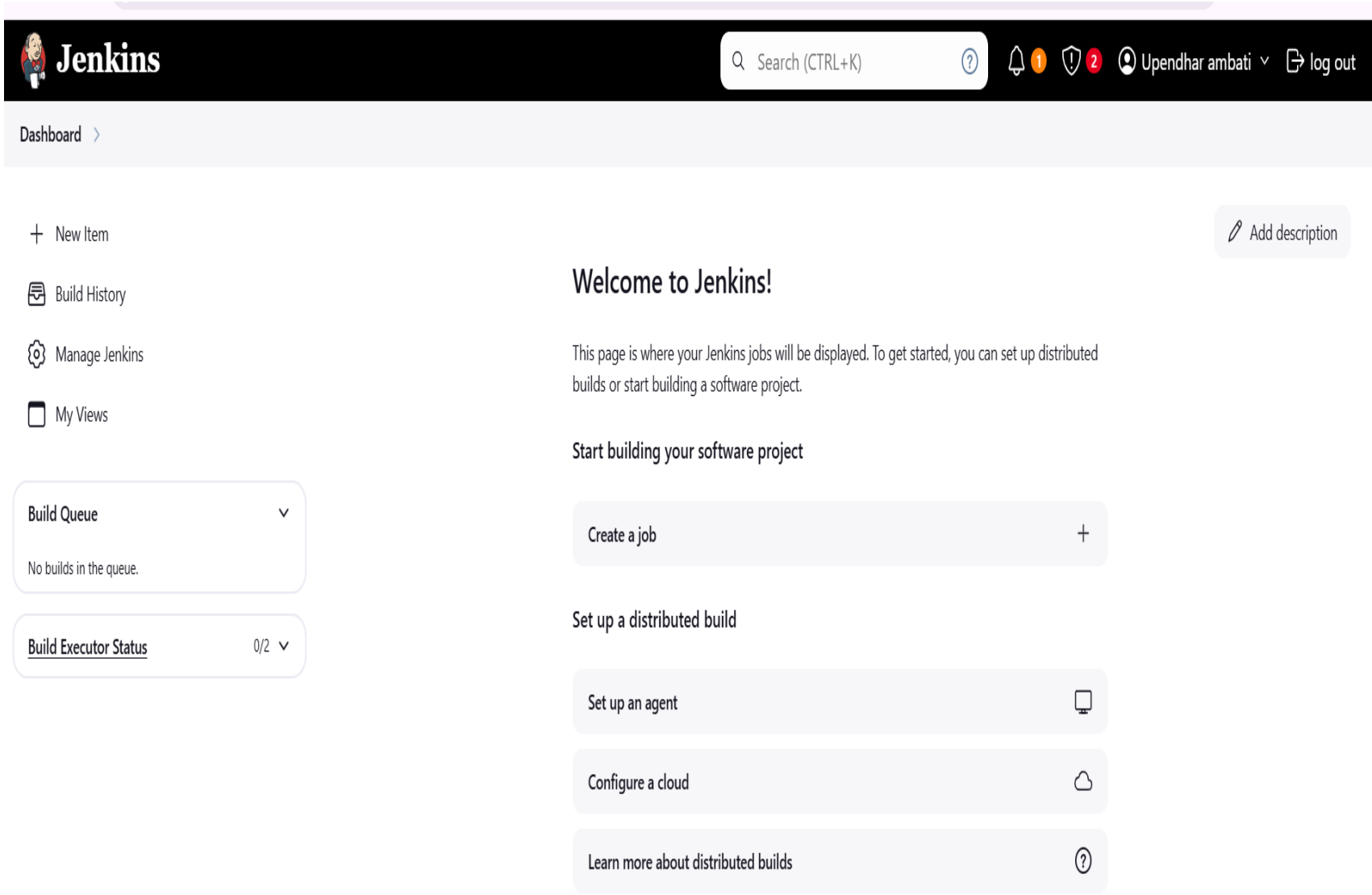


You have been successfully authorized in GitHub. You can close the page.

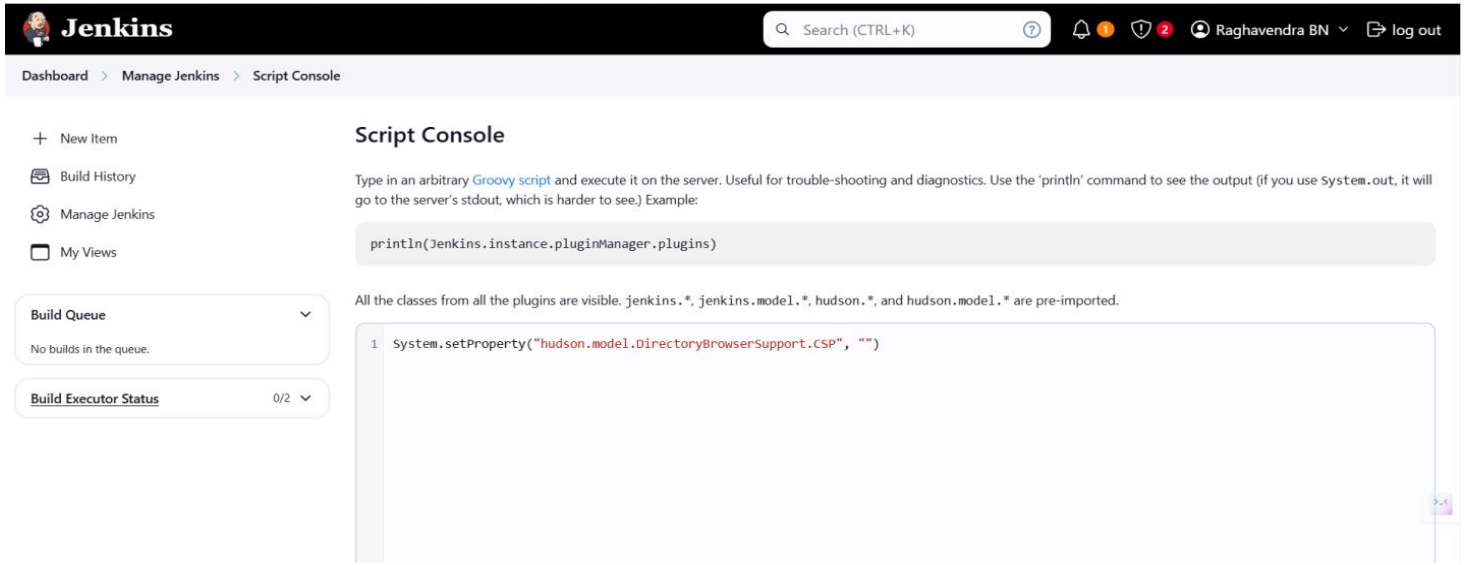
Successfully code is pushed into the Github



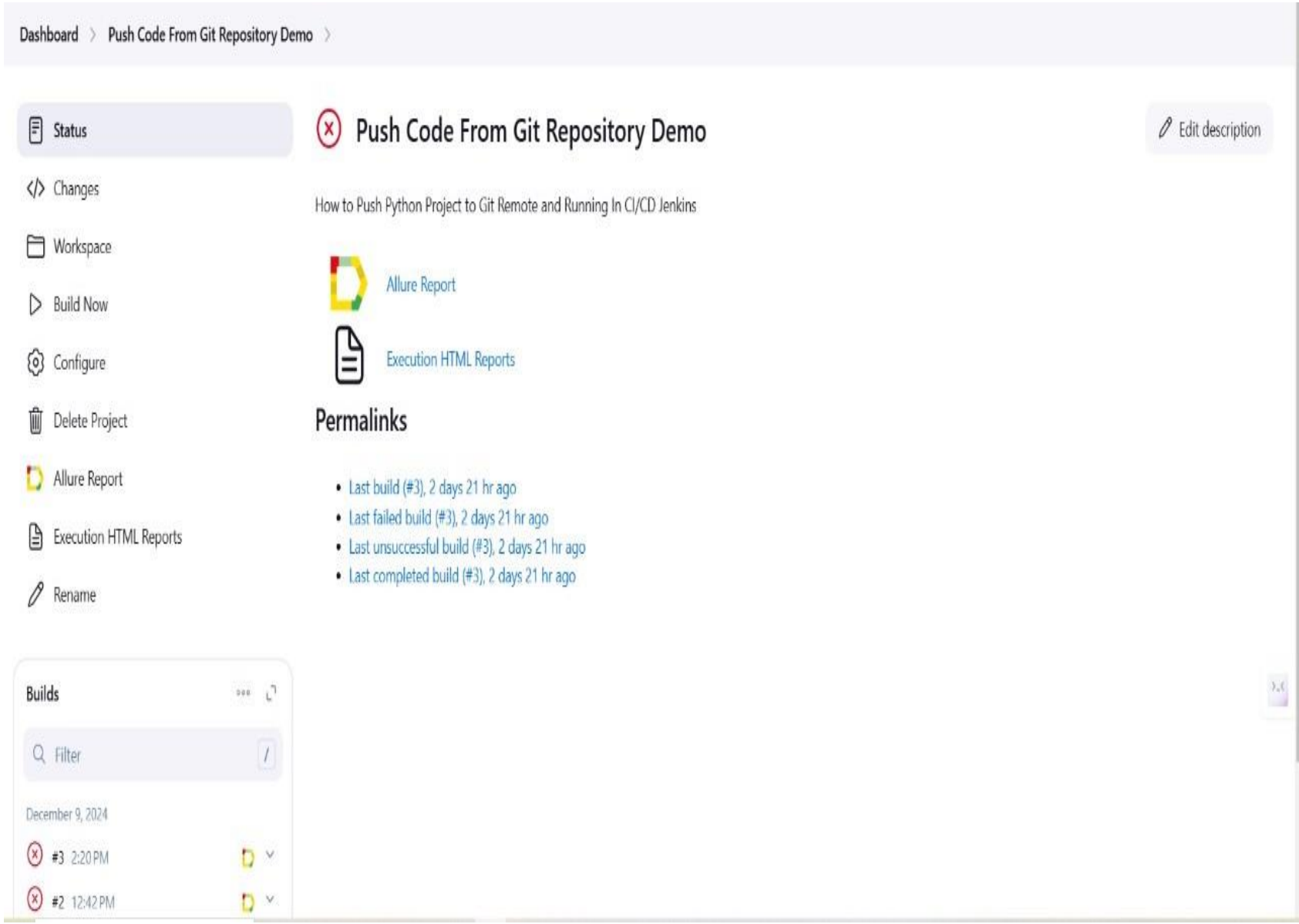
Lunching Jenkins



Jenkins script console



Push Code From Git Repository



Thank You