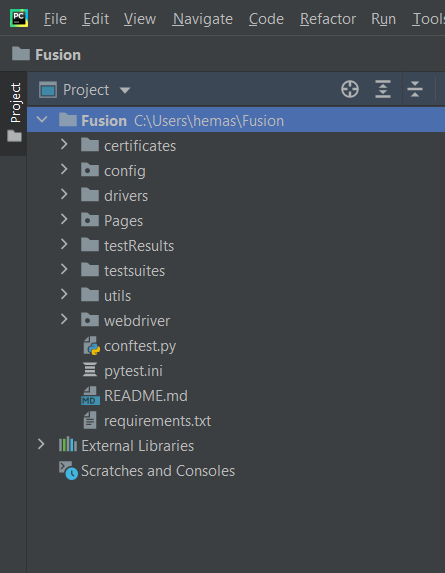
Table of Contents

1. Details of the Packages 2
2. Config packages 3
3. Drivers package 3
4. Pages package 4
5. testResults package 4
6. testsuites package 5
7. utils package 6-8
8. WebDriver package 8
9. Conftest file 9
10. Pytest.ini file 10
11. Requirements.txt 11
12. Project Pre-requisites 11-13

Different types of Packages and config files are listed below.

* config
* drivers
* Pages
* testResults
* testsuites
* utils
* webdriver
* conftest.py
* pytest.ini
* requirements.txt

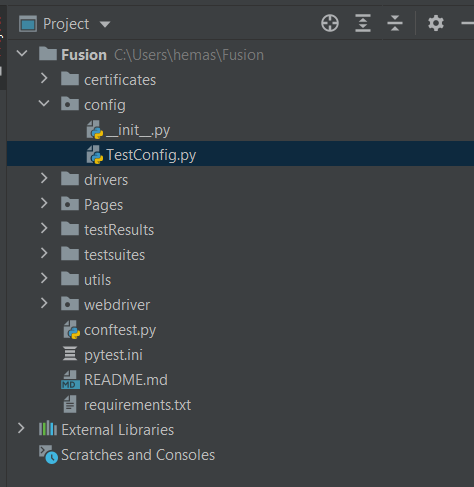


**Config:**

All the testcase config’s data are initialized in TestConfig.py.

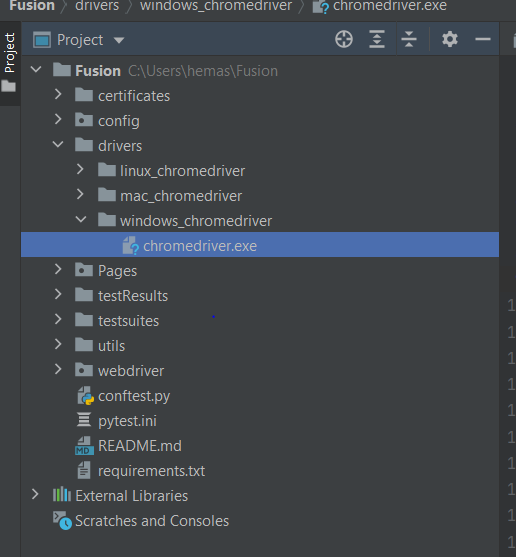
Ex: test\_data.json file contains the login credentials.

"login\_data": [  
 {  
 "username": "Admin",  
 "password": "admin123"  
 }  
]



**drivers:**

Driver’s executable files are present in this folder for different platforms and browsers.



**Pages:**

Automation classes are created for each functionality available in web pages. The Appropriate methods are available in appropriate classes.

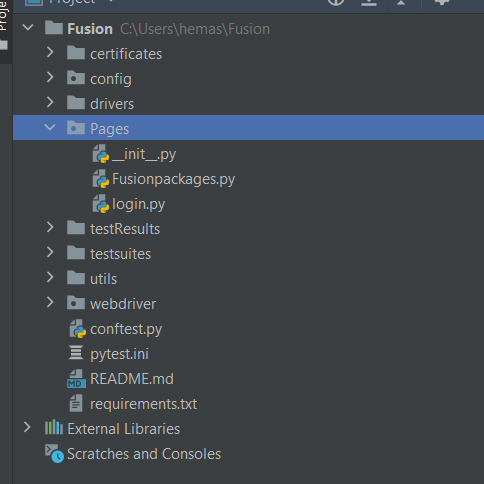
**New functionality for existing feature in UI**

Create a method for that functionality in existing python feature class.

**New feature in UI**

Create the new python class with the feature name, under this location.

EX: HrmAdmin



**testResults:**

Execution of test results are generated in the form of reports, html file and logs.

**Steps for Allure Reporting**:

Install the Java and NodeJS and set the path variable

Run the below command in cmd:  
 npm install -g allure-commandline

Import allure in your test file

Run the Pytest command with allure path

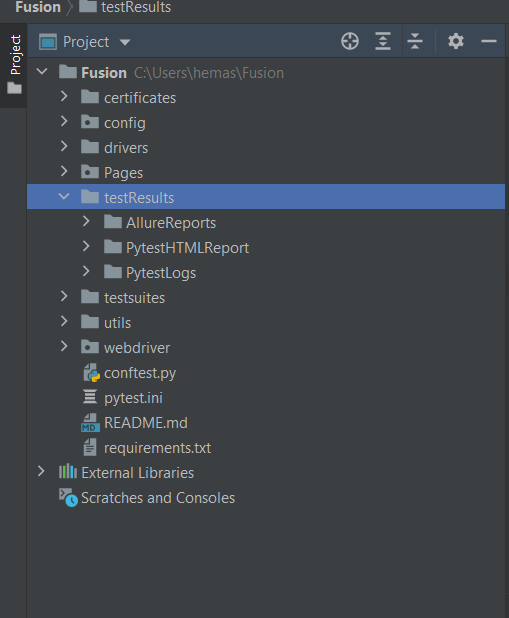
pytest test\_xyz.py --alluredir=path\_where\_you\_want\_to\_save\_reports

Run below command to launch the report

allure serve report\_path

A screenshot of a computer

Description automatically generated



**testsuites:**

Testsuites are available under this directory for all the features.

**Existing feature**

Create a testmethod in the existing testsuite file, with proper title and marker details.

**New Feature**

Create a new testsuite file with feature name and create the testmethod with all details.

**Regression Folder**--- Sample test script to automate the web application.

How to integrate the LaunchBrowser class and WebDriverHelper class

**Api** --- Sample test script to automate the service api.

**test\_api\_database\_connect.py ---** Sample test script to make a connection with database.

The tests will validate the functionality of each method without needing a real database connection, thanks to the use of unittest.mock.

A screenshot of a computer program

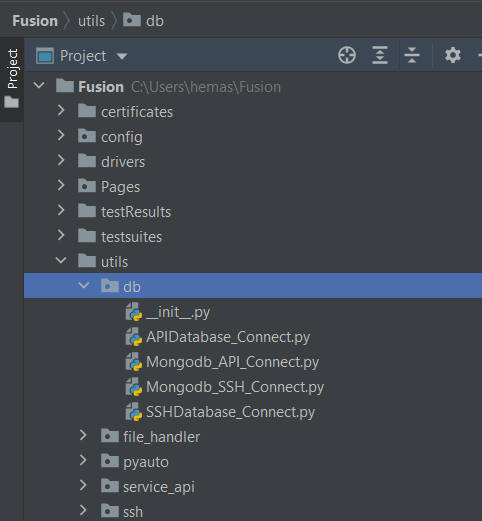
Description automatically generated

**Utils:**

If the expected methods or data’s not available in utils file, we are adding it in appropriate util functionalities.

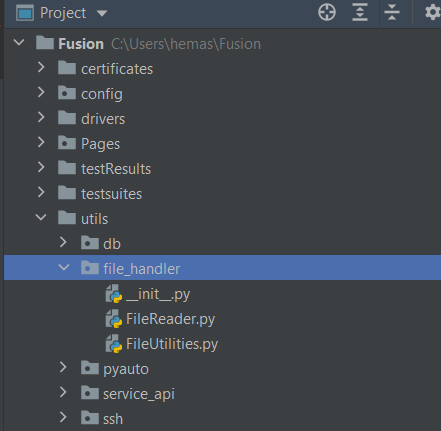
**DB utils**

The common reusable codes to connect db and access data from db, codes are available under this section.



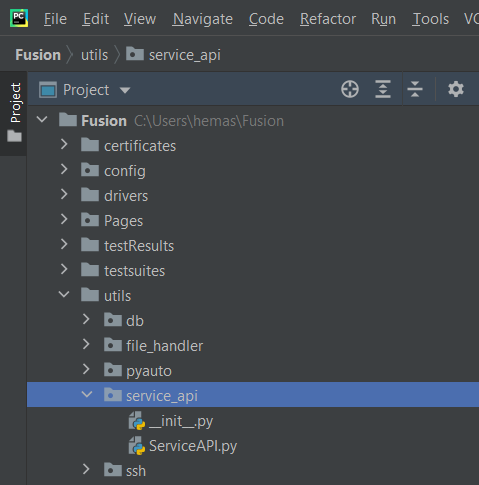
**File Handler utils**

Reusable methods available to read data from file and write data to the given file.



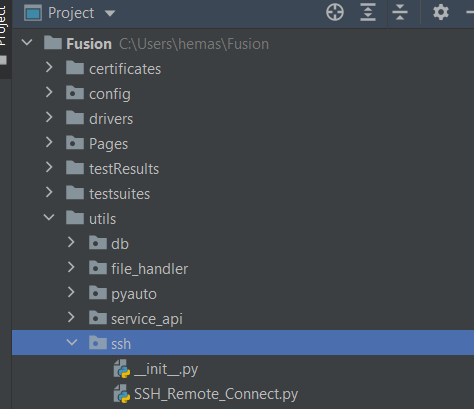
**Service API utils**

Reusable methods available for GET, PUT, POST and DELETE rest API Call’s.



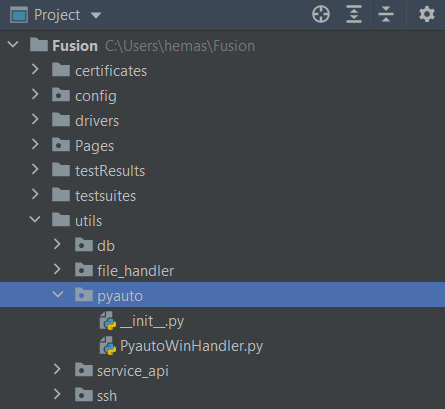
**SSH utils**

Reusable methods available to connect the remote machine through SSH commands.



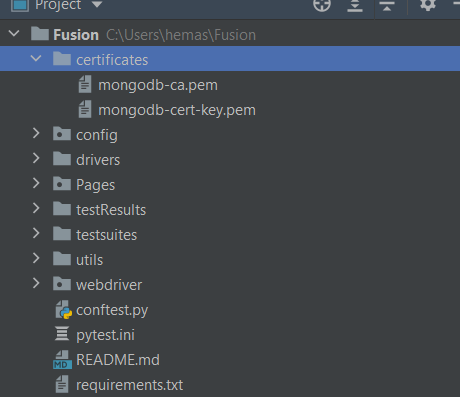
**Pyauto util**

Common method to handle window popup.



**Certificates**

Under this directory the certificates are available, so we can reuse it wherever it required.



**WebDriver:**

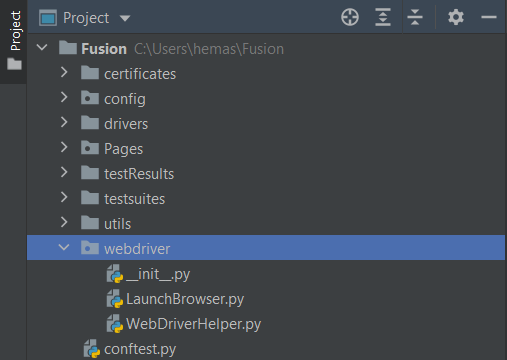
Reusable methods are available to launch browser and selenium webdriver commands.

The LaunchBrowser class is to handle browser setup and initialization and contains methods for launching and configuring a web browser (such as Chrome, Firefox, etc.) before performing automated UI tests.  
  
**LaunchBrowserNew**.py---- Updated code with new version of selenium

The WebDriverHelper class is to serves as a utility/helper to encapsulate common operations or interactions with the browser, such as clicking elements, sending text, waiting for elements to appear, or capturing screenshots. It helps to avoid repetitive code by providing reusable methods to interact with web elements during automated UI testing

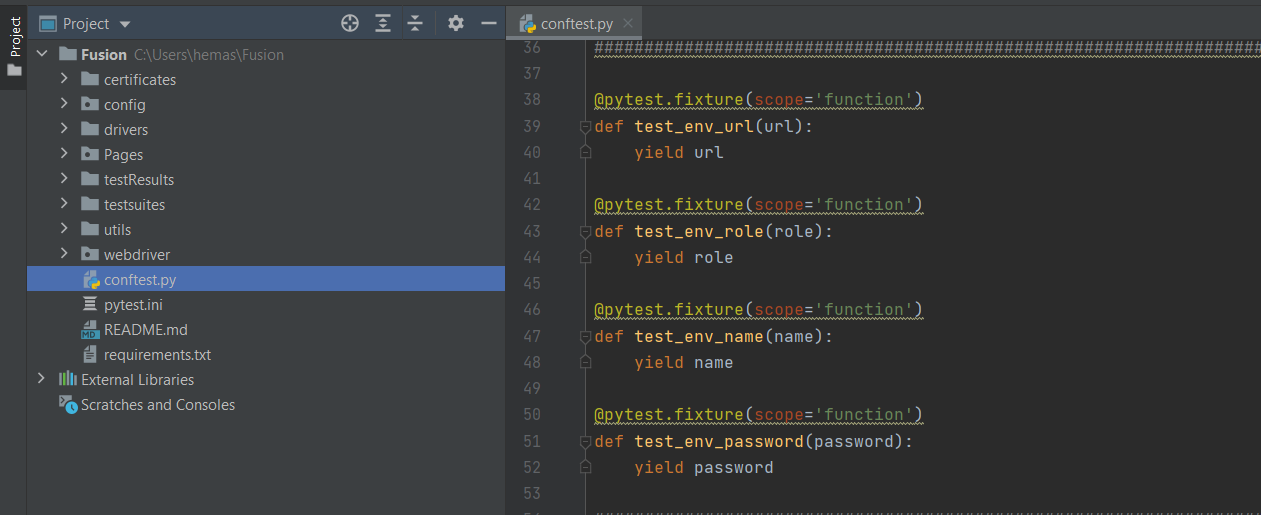
**WebDriverHelperNew**.py---- Updated code with fix of reusable methods and handle drive in init method.

50+ reusable methods.



**Conftest.py:**

* conftest.py is used to import external plugins or modules. By defining the global variable, pytest will load the module and make it available for its test. Plugins are generally files defined in your project or other modules which might be needed in your tests.
* Fixtures are a potential and common use of conftest.py. The fixtures that you will define will be shared among all tests in your test suite.
* Also we can specify hooks such as setup and teardown methods and much more to improve your tests.



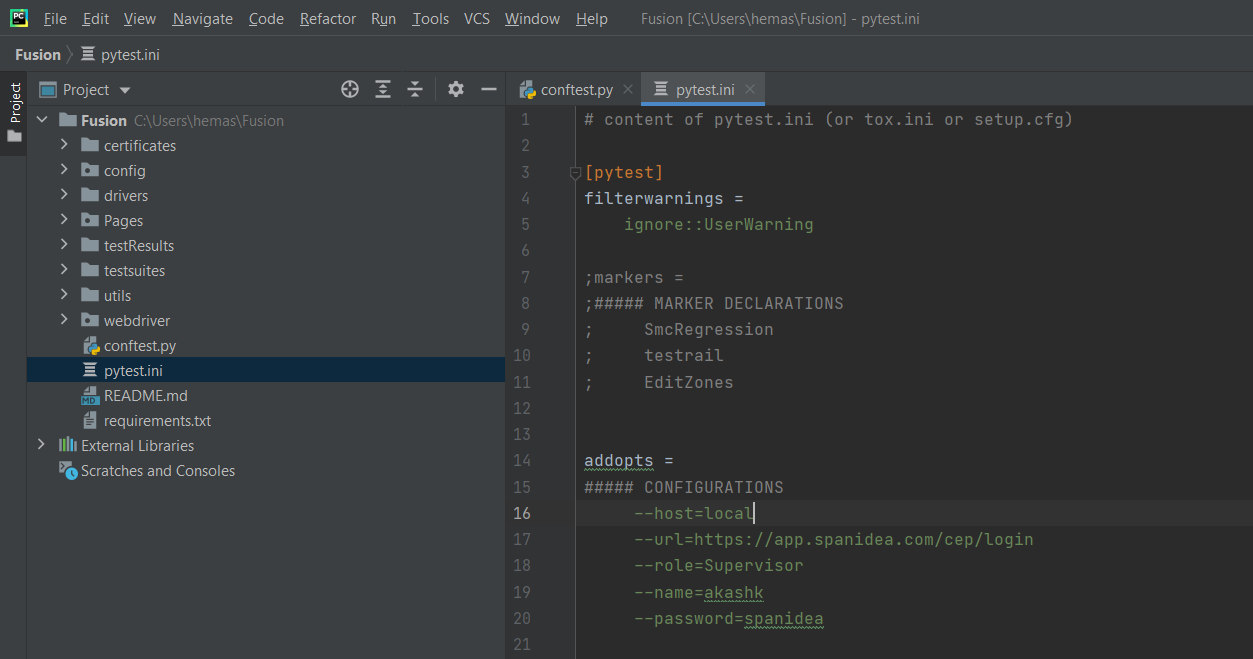
**pytest.ini:**

Command line options and Configuration file settings are available in pytest.ini.

**Test Execution parameters**

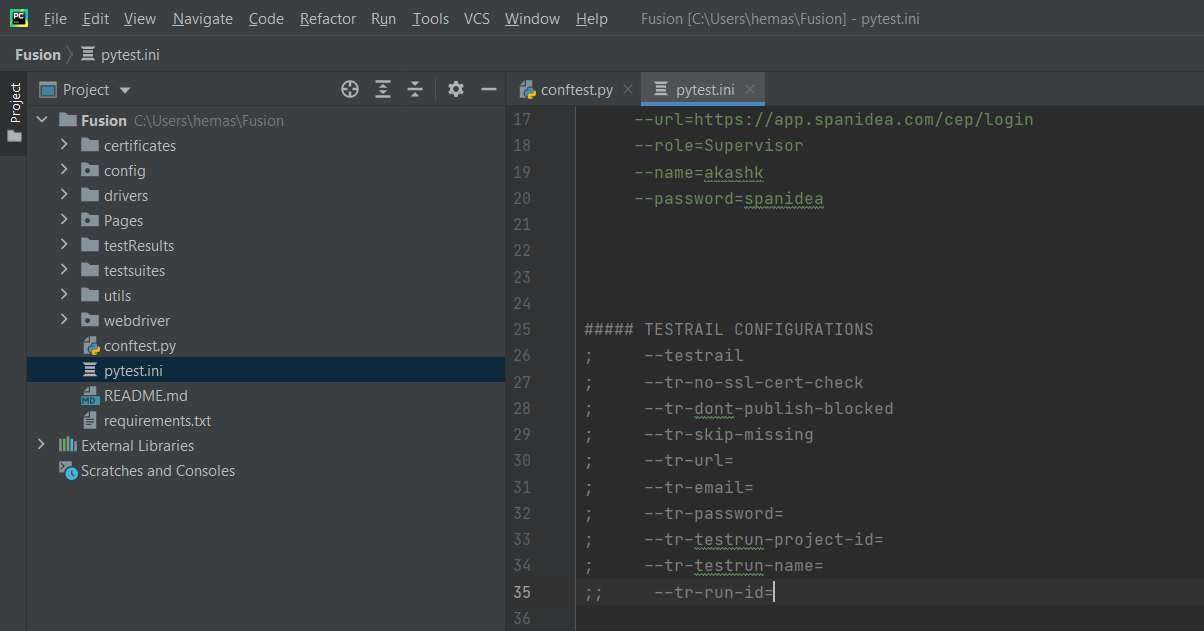
Test Execution parameters are available in pytest.ini file. These files take precedence over other files.

Please change the configuration data based on your requirement.



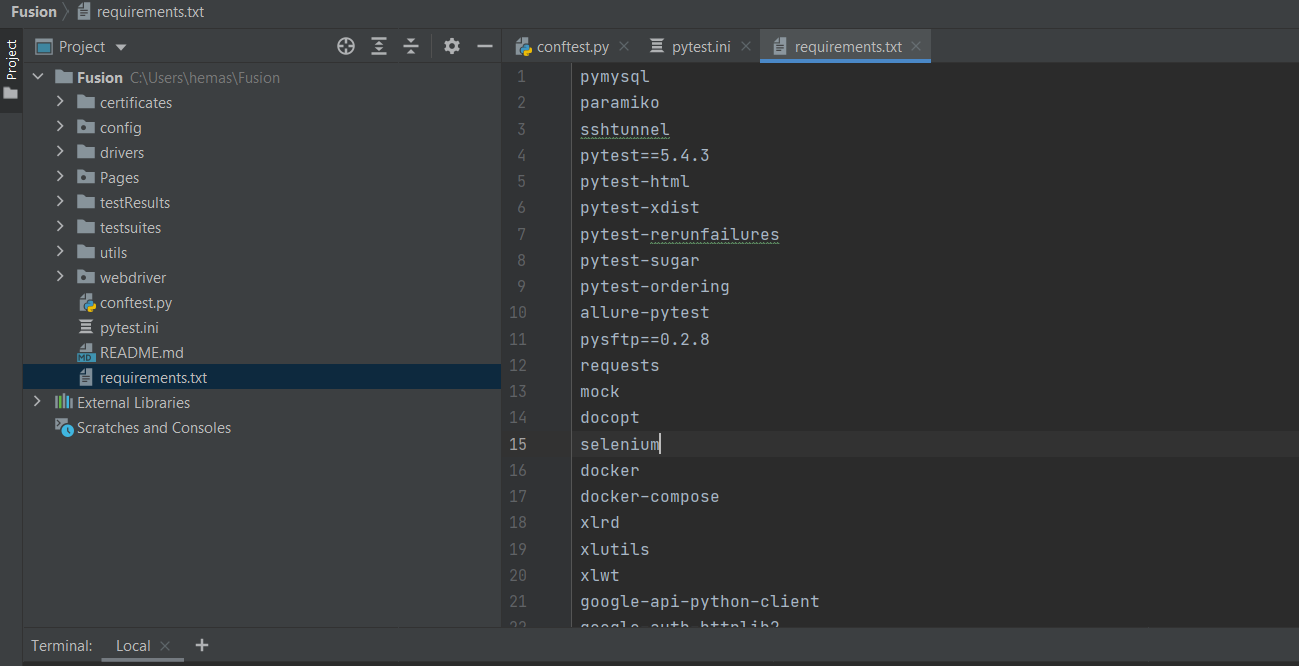
**TestRail:**

To Upload the test result in testrail, please uncomment below parameters in pytest.ini file. Also change the parameters values based on your requirement.



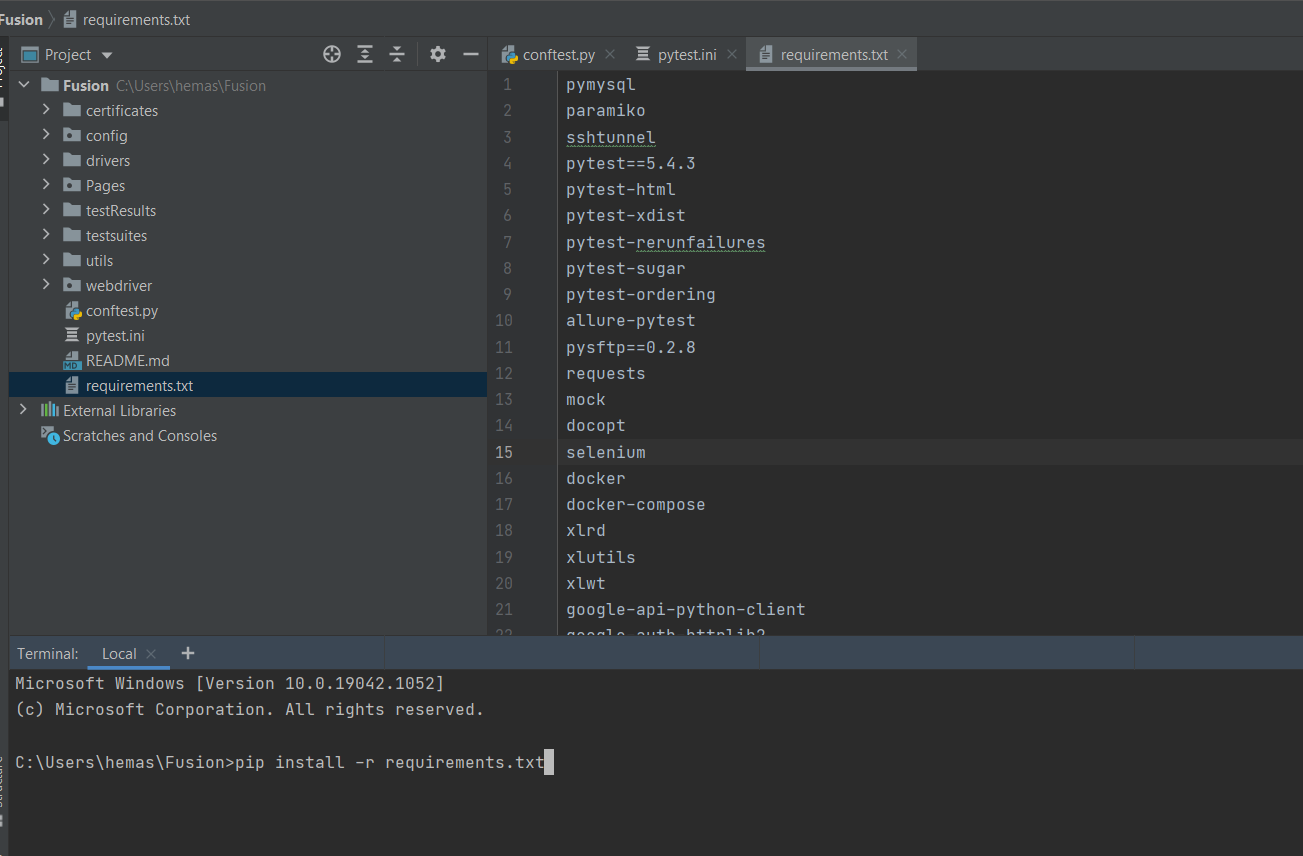
**Requirements.txt:**

Requirements.txt file is used for specifying python packages and libraries are required to run the project. This file will automatically update the package details with version when someone installs a new package, also python provides a way to install those version packages directly from the requirements.txt file, so you need not to manually install each & every package. Typically the requirements.txt file is located in the root directory of your project.



**Project Pre-requisites:**

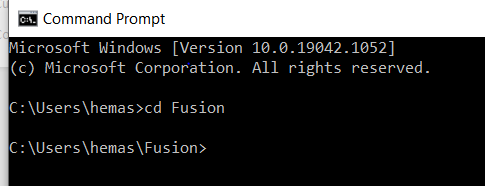
1. Install Python and set path in environment variables.
2. Install Pycharm and set path in environment variables.
3. Install git
4. Git checkout the project in pycharm from git VCS.
5. Install the requirements.txt which will execute the project dependencies



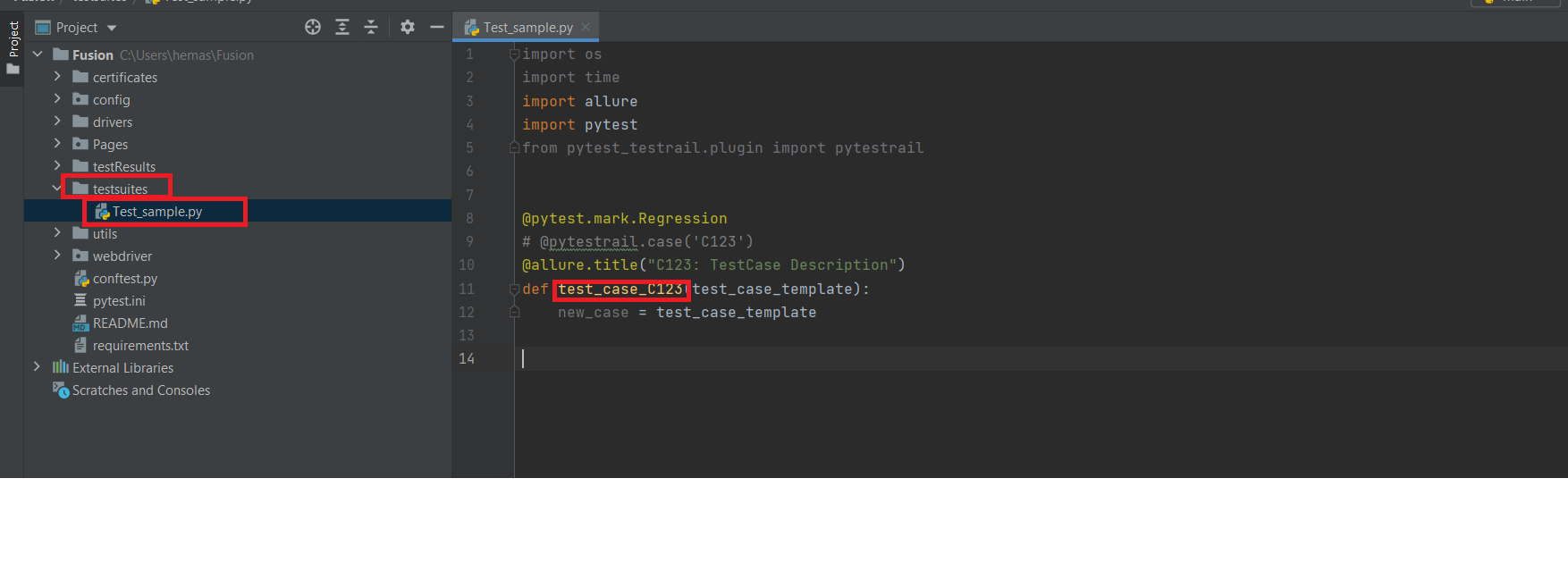
1. Run test from command line

Steps to run the specific test from specific file

* + Set the path in command prompt, till project base directory.
  + Open command prompt.



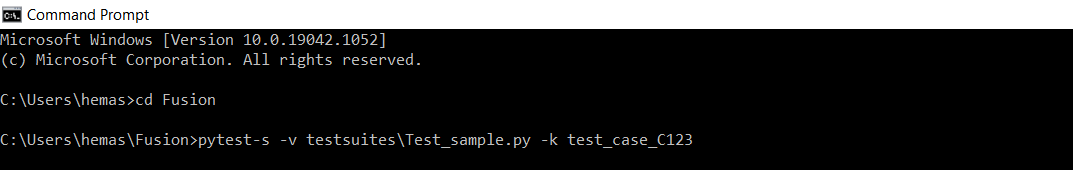
* + Select the pytest file from base directory
  + Select the test from pytest file.



Enter the pytest run command in command prompt, with below format

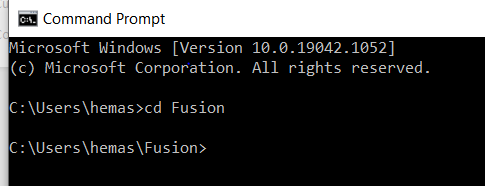
Pytest -s -v <TESTFILE PATH> -k <SINGLE TEST METHOD Name>

Example:

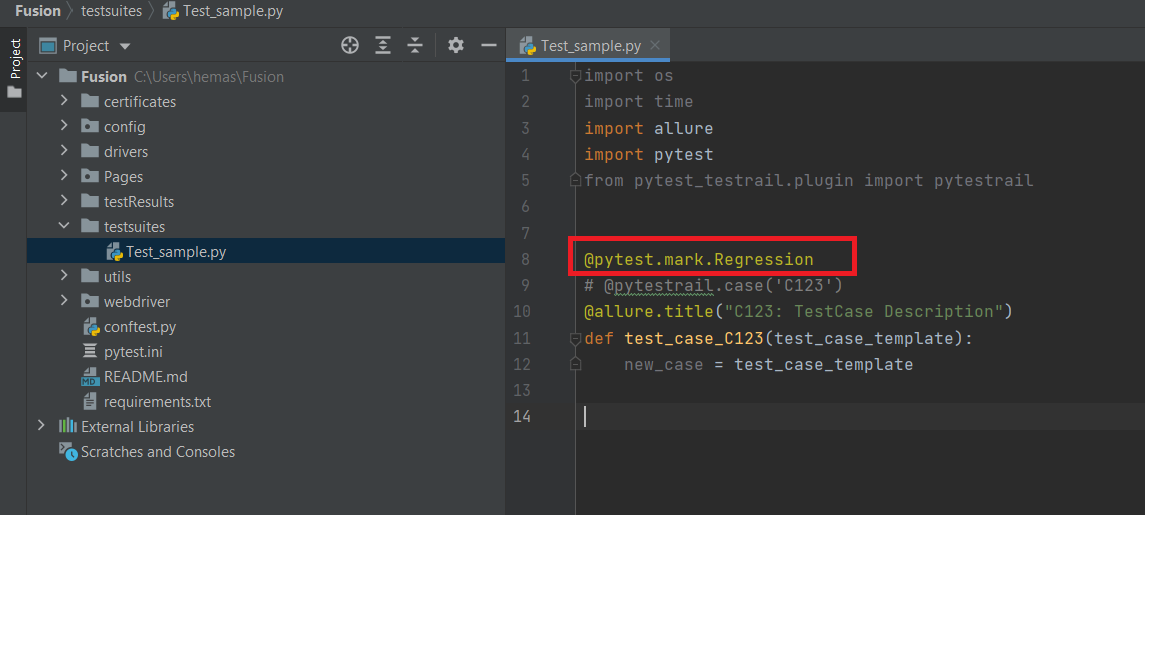


**OR**

1. Run Automation script from command line using marker(marker is to group the testcases and run test with single command)
   * Set the path in command prompt, till project base directory.
   * Open command prompt.



* + Select the marker



Enter the Pytest run command in command prompt, with below format

Pytest -s -v -m <MARKER NAME>

Example:

