# PYTEST-BDD GUIDE

Automation Setup Guide

# Table of Contents

# Document Control

## Revision

| Version | Author | Comments | Date |
|---|---|---|---|
| 0.1 | MUHAMMAD SIBTAIN HAIDER | Initial Draft | AUGUST 24th 2023 |
| | | | |

## Approval

| Name | Designation Department and Section | Signature |
|---|---|---|
| | | |

# Introduction

## Purpose

Purpose of this documentation is to guide about the Pytest-BDD framework, what all tools and technologies are used in its installation with an example web UI test suite and discuss its advantages.

# What is Pytest-BDD ?

Pytest-BDD is a Python BDD framework used for agile software development technique that promotes collaboration between non-technical or business stakeholders with the developers and testers.

Pytest-BDD implements a subset of the Gherkin language to enable automating project requirements testing and to facilitate behavioral driven development.

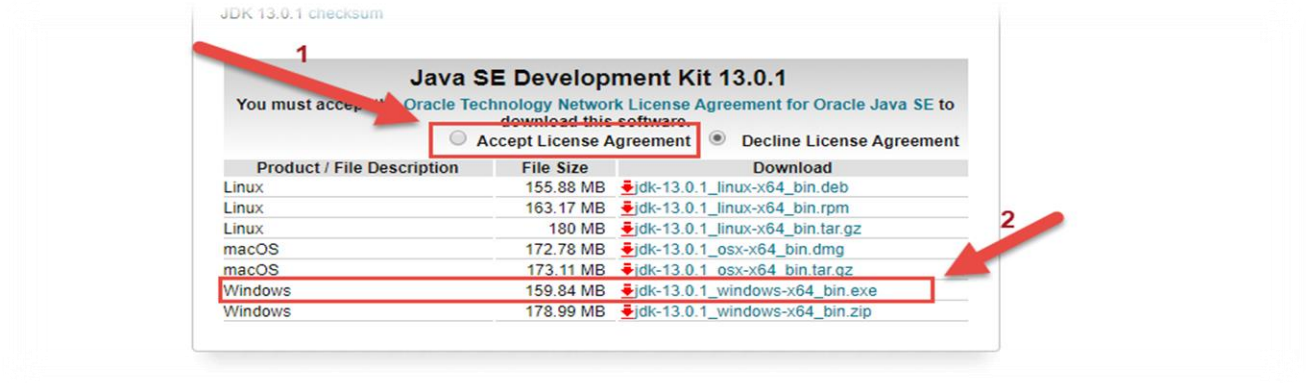
# Pre-Requisites

- Java JDK
- Python
- Pytest
- Selenium
- Allure
- IntelliJ IDEA Ultimate Version

## Java JDK

Download Java JDK from link

https://www.oracle.com/technetwork/java/javase/downloads/jdk13-downloads-5672538.html




After downloading the .exe file, open it and install it. To verify, if it is installed correctly, open "**command prompt**" and write 'java –version' command on it. You will be able to see the java version installed.

## Python

Download Python from link

Download Python | Python.org

After downloading the .exe file, open it and install it.

## Pytest

Download Pytest using the pip command

```
pip install -U pytest
```

## Selenium

Download Selenium using the pip command

```
pip install selenium
```

## IntelliJ IDEA Ultimate Version

Download IntelliJ IDEA from link

https://www.jetbrains.com/idea/download/#section=windows


After downloading the .exe file, open it and install it.

# Project Configuration

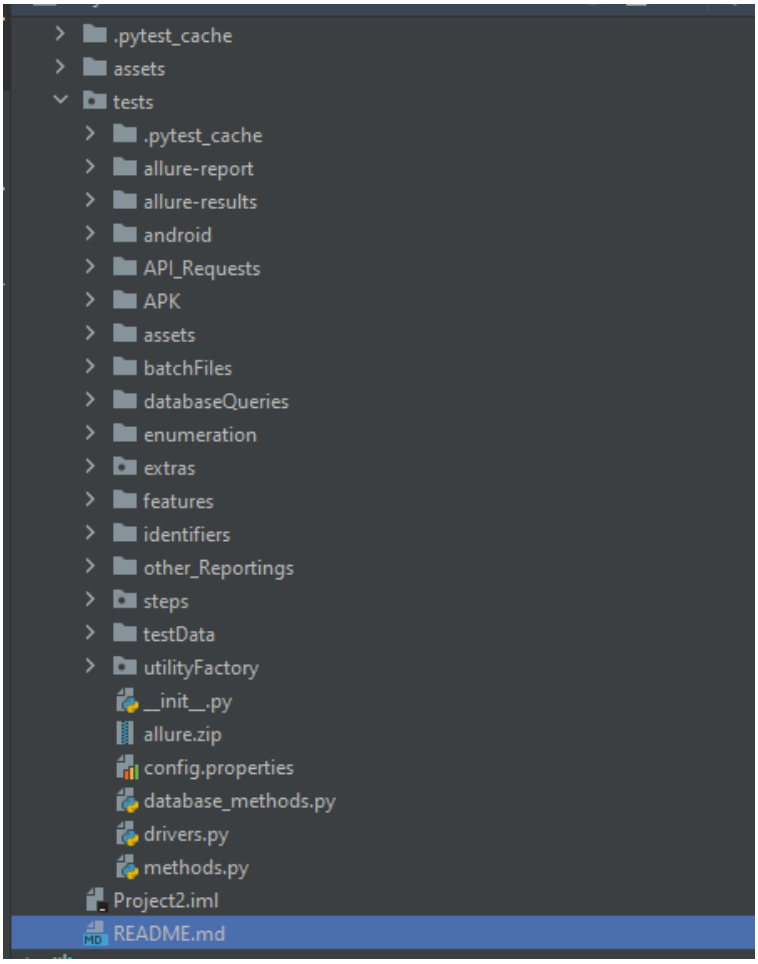
Considering you have the project in a zip file. Follow the following steps to configure the project:

- Extract the project and place in some folder and copy its path.
- Open IntelliJ and select '**Import Project**'.
- Paste the path and locate to Selenium-Script folder.
- Select it and click OK.
- Select '**Create project from existing sources**' and click next.

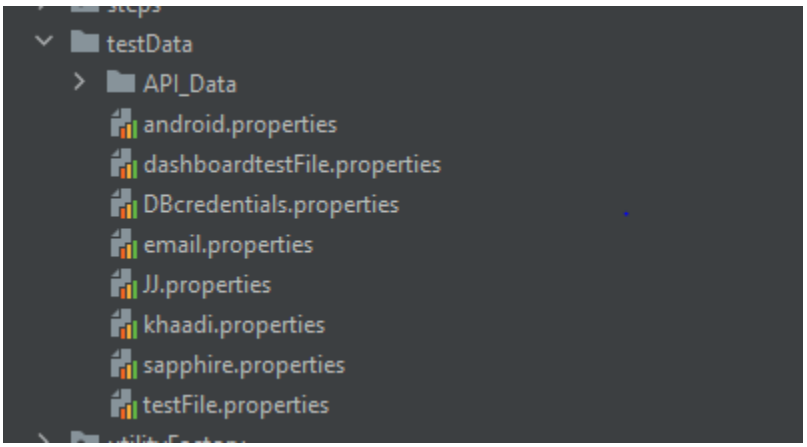
## Data Configuration

In IntelliJ, click on project explorer navigate to "**/test/**" folder you will be able to see multiple folders under directory:

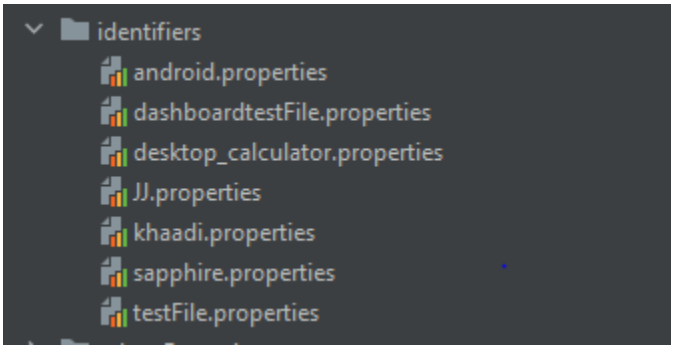
1. testData
2. Identifiers
3. Enumeration


.pytest_cache
assets
tests
.pytest_cache
allure-report
allure-results
android
API_Requests
APK
assets
batchFiles
databaseQueries
enumeration
extras
features
identifiers
other_Reportings
steps
testData
utilityFactory
__init__.py
allure.zip
config.properties
database_methods.py
drivers.py
methods.py
Project2.iml
README.md

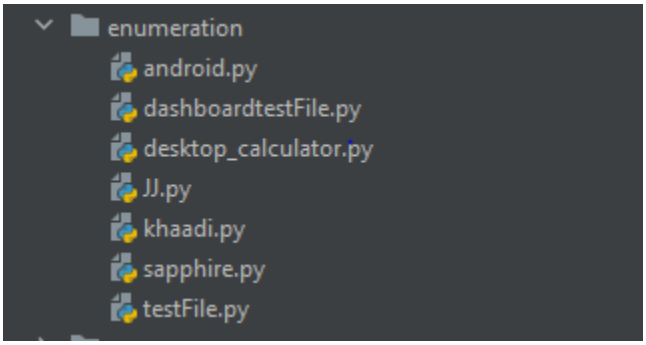
## testData


testData
API_Data
android.properties
dashboardtestFile.properties
DBcredentials.properties
email.properties
JJ.properties
khaadi.properties
sapphire.properties
testFile.properties

## Identifiers



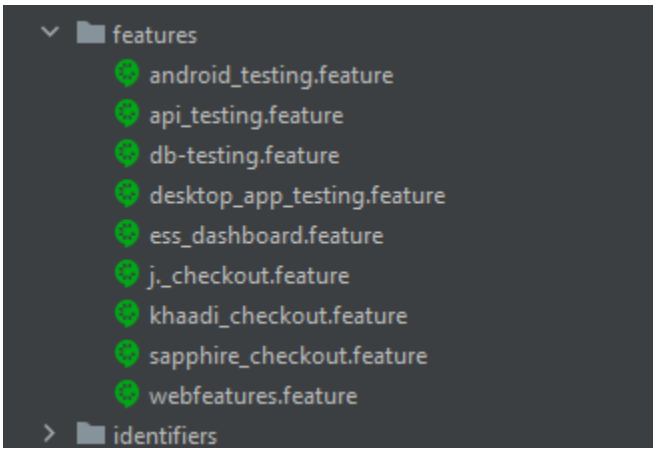

## Enumeration



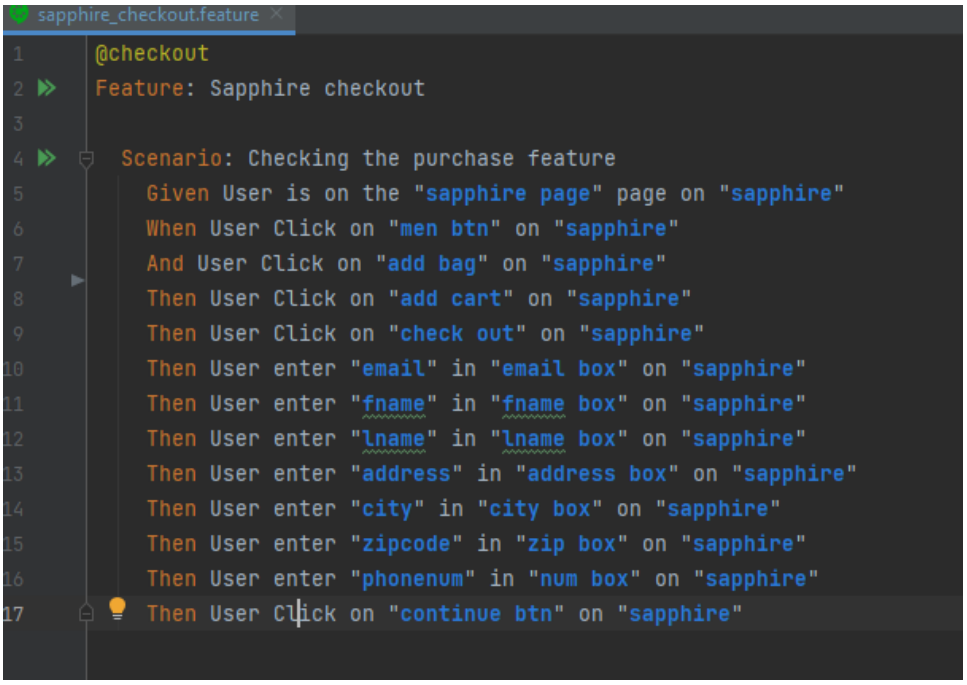
# Business Use Case and Reporting

## Feature File

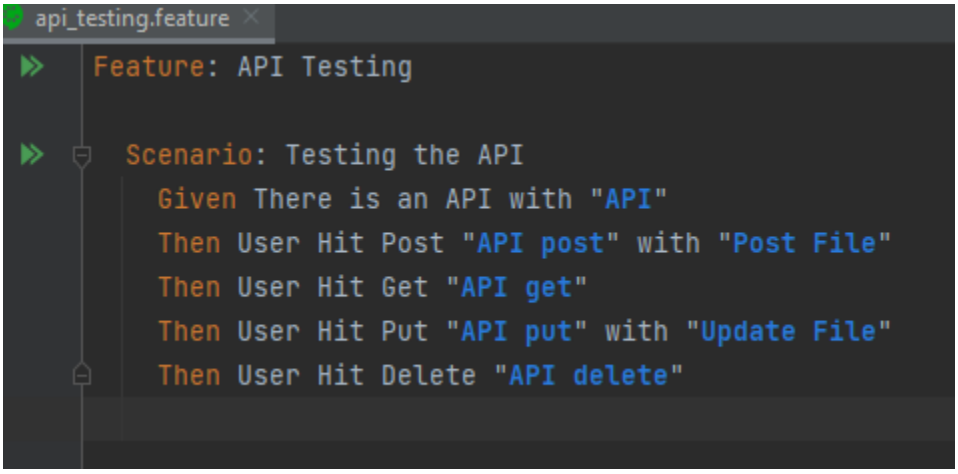
The Feature file is a high-level description of a software feature, and to group related scenarios or Test cases, we write our test case steps.




### sapphire_checkout.feature

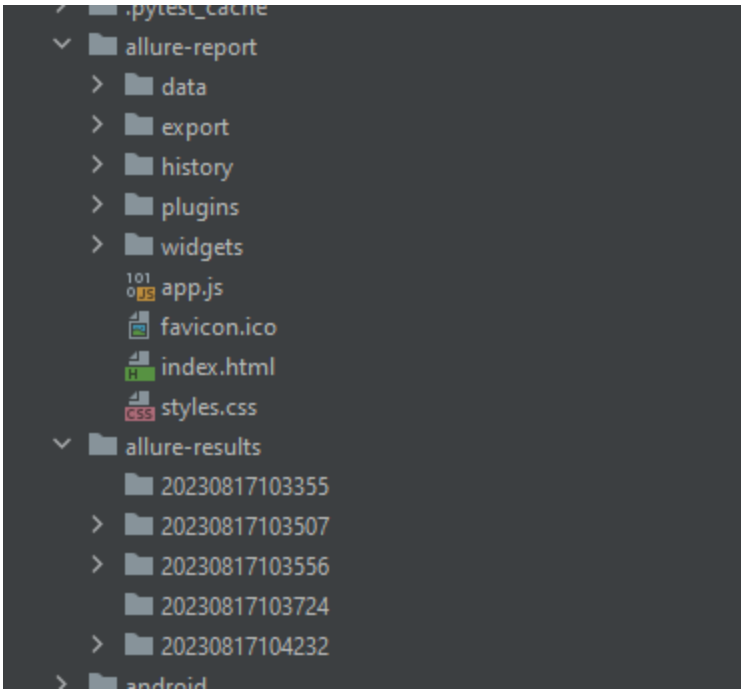


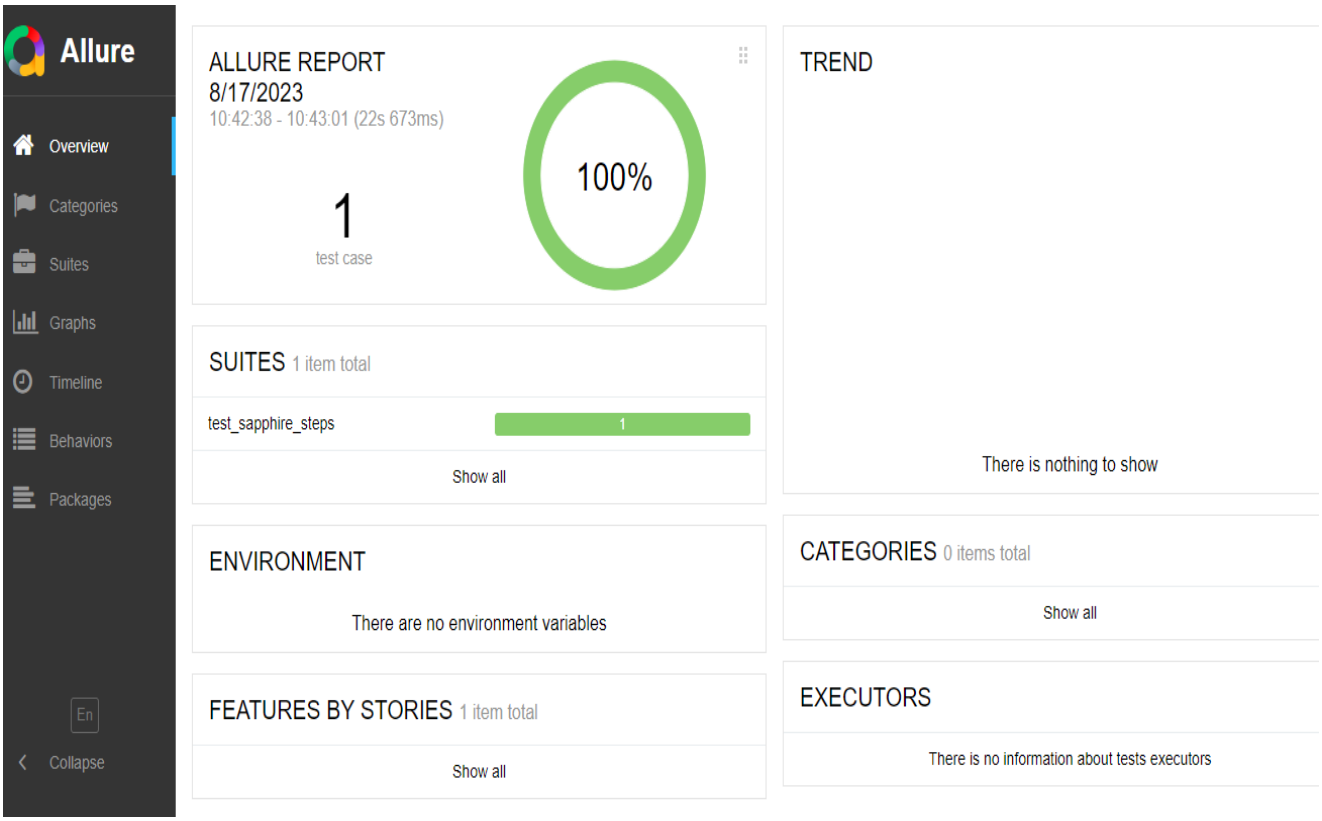
Api_testing.feature




## Report

After execution the scenario you can see the report in **target/executionReports** directory


.pytest_cache
allure-report
data
export
history
plugins
widgets
app.js
favicon.ico
index.html
styles.css
allure-results
20230817103355
20230817103507
20230817103556
20230817103724
20230817104232
android




# How to Write Steps

To steps, navigate to the “tests directory” then create a new python directory named steps and then create a new file named conftest.py and write the common steps there.

```python
from allure_commons.types import AttachmentType
from pytest_bdd import given, when, then, parsers
from tests import drivers, methods
import allure
from appium.webdriver.common.touch_action import TouchAction
import os

current_script_path = os.path.abspath(__file__)
path1 = os.path.dirname(os.path.dirname(current_script_path))


SibtainHaider
@allure.severity(allure.severity_level.NORMAL)
@given(parsers.parse('User is on the "{web_name}" page on "{testfile}"'))
@when(parsers.parse('User is on the "{web_name}" page on "{testfile}"'))
@then(parsers.parse('User is on the "{web_name}" page on "{testfile}"'))
def browser_navigation(web_name, testfile):
    web_name_edit = methods.data_mod(web_name)
    testfile_edit = methods.file_mod(testfile)
    path2 = "/testData/{}.properties".format(testfile_edit)
    url = methods.get_data(path1 + path2, 'details', web_name_edit)
    drivers.driver.get(url)
    allure.attach(drivers.driver.get_screenshot_as_png(), name="url_browse", attachment_type=AttachmentType.PN
```

This file will be automatically imported in all the test files.

# How to Write Tests

To write tests, navigate to the “tests directory” then open python directory named steps and then create a new file named test_*.py and write the test as follows:

```python
from pytest_bdd import scenario


SibtainHaider
@scenario('../features/sapphire_checkout.feature', "Checking the purchase feature")
def test_sapphire():
    pass
```

Where the first parameter of the scenario function is the location of the feature file and the second parameter is the name of the Scenario to be tested.

# How to Run Tests

To run tests, navigate to the “test_file.py” and then click on the button as follows: