MANUAL TESTING BLUESTONE

ABOUT BLUESTONE:

Established in 2011, BlueStone is India's leading destination for high quality fine jewellery with strikingly exquisite designs. We aim at revolutionizing the fine jewellery and lifestyle segment in India with a firm focus on craftsmanship, quality and customer experience. In a short span of time, BlueStone has built a large family of loyal consumers in India and abroad.

We house more than 8000 unique designs for you to choose from. All these designs are crafted to perfection with utmost care giving you the flexibility to customize the product's gold purity and colour or diamond clarity to suit your needs.

Our stores have been instrumental in spreading the shine of BlueStone and bringing us closer to you. With world class experience, friendly staff and the dazzling beauty of exquisite jewellery, every store is a sparkling gem.

With an award-winning design team that pays great attention to detail, each of our products are a symbol of perfection. With cutting edge innovation and latest technology, we make sure the brilliance is well reflected in all our jewellery.

We also offer a 30-Day Money Back guarantee, Certified Jewellery and Lifetime Exchange. You can also experience luxury shopping from the comfort of your home with our complimentary Try at Home service.

BlueStone is a web-based e commerce application and it deals with all type of jewellery items and it has both mobile app support for customers to track orders and easy access.

TABLE OF CONTENTS:

- Introduction
- Project Overview
- Getting Started
    - Prerequisites
    - Installation
    - Project Setup
- Project Structure
- Writing Testcases
- Running Tests
- Test Reports
- Advantages of Manual Testing
- Disadvantages of Manual Testing
- Troubleshooting
- Conclusion

Introduction:

The BLUESTONE project is a Python automation project that utilizes the Selenium framework for automating web interactions. This documentation provides an overview of the project, instructions for installation and configuration, details about the project structure, and examples of test cases.

Project Overview:

The BLUESTONE project aims to automate various tasks on the Bluestone website. It leverages Selenium, a powerful web automation tool, to interact with web elements and perform actions such as clicking buttons, filling forms, and extracting data.

Milestone 1: Getting Started

Explain the prerequisites and setup required to run the project.

- **Activity 1:- Prerequisites**

  Before getting started with the BLUESTONE project, ensure you have the following prerequisites:

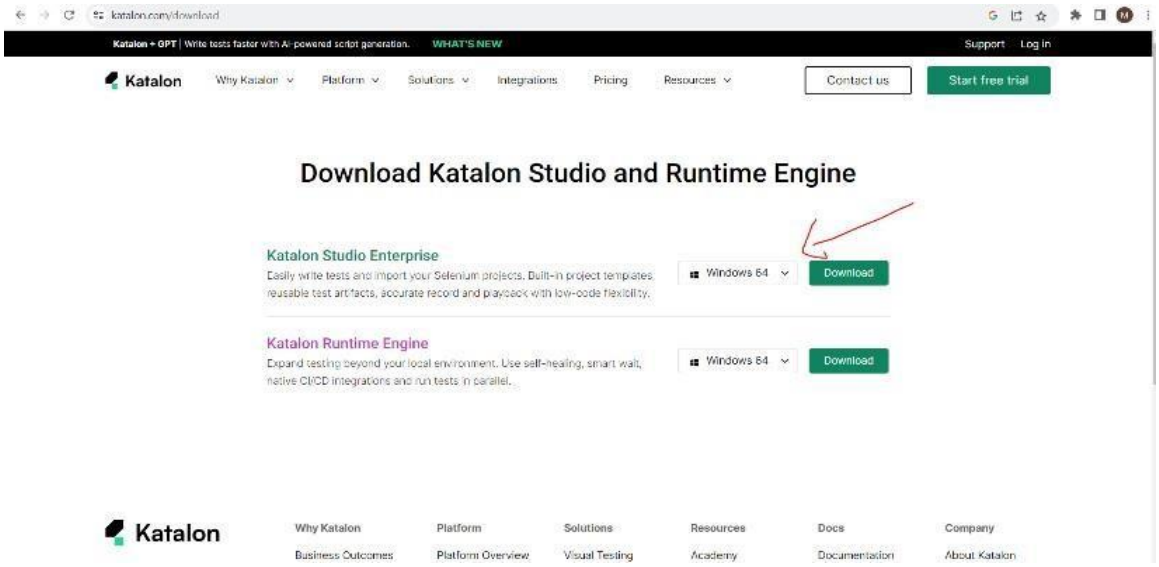
  List the software and tools that need to be installed before running the project, such as

  - Python (3.x)
  - Selenium (3.x)
  - Pytest (Latest Version)
  - Any other dependencies

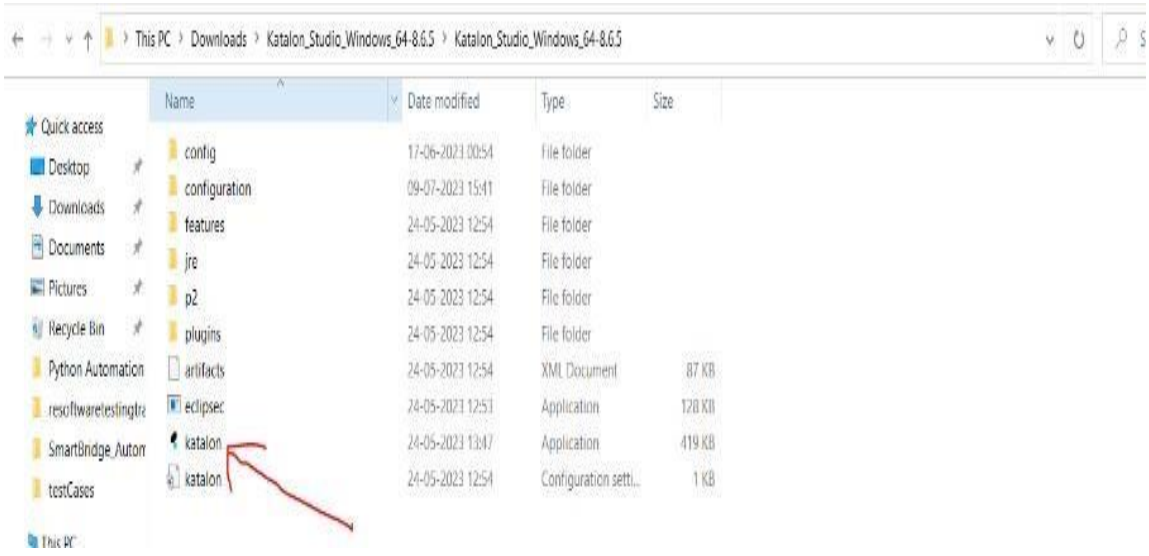
- **Activity 2:-Installation**

  To install the necessary dependencies for the actiTIME automation project, follow these steps:

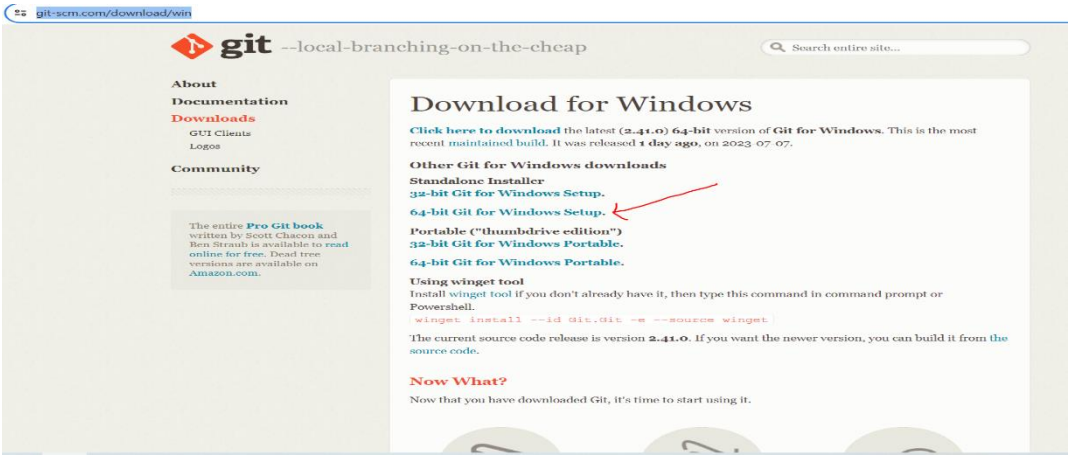
  Activity 2.1:- Install Katalon Studio: Visit the Katalon Studio website (https://katalon.com/download) and download the latest version of Katalon Studio. Fo **l**ow the installation instructions for your operating system.

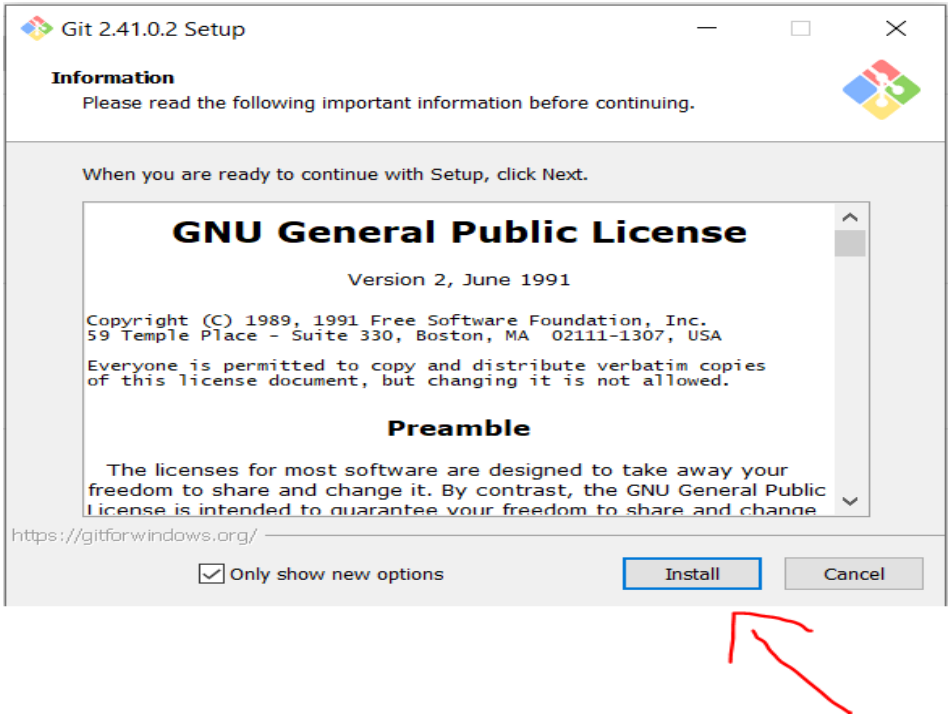


- After downloading Katalon Studio and extracting the contents from the zipfile, locate the folder where you extract edit.
- In that folder, look for the "katalon.exe" file. This file is the executable for Katalon Studio.
- Double-click on the "katalon.exe" file to launch Katalon Studio. It may take a few moments to startup.
- Once Katalon Studio is launched, you can start creating or opening your project by selecting "File" from the menu bar and choosing either "New Project" or "Open Project" based on your requirements.



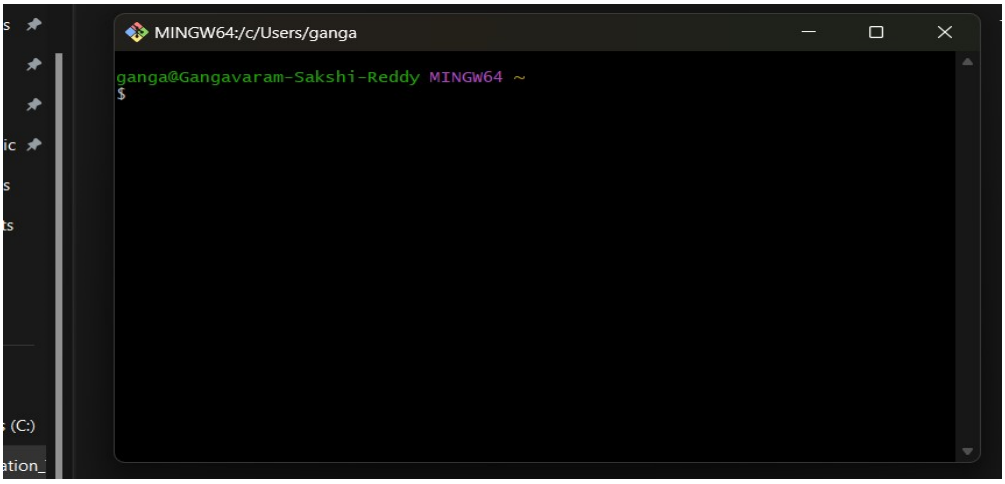
Activity 2.2:- Download and install gitbash from (https://git-scm.com/download/win).





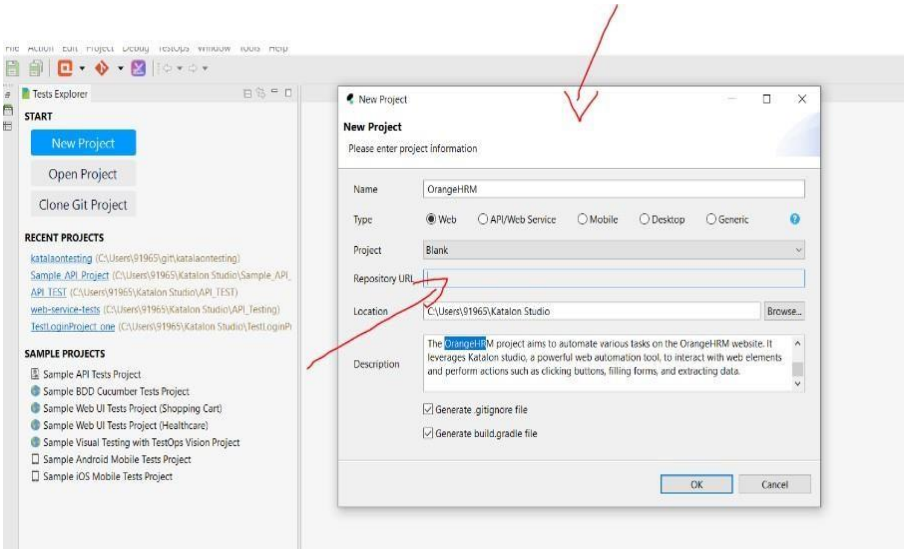
Activity 2.3 :- Go to the desired location where you want to create the folder, such as Desktop. Folder name like eg…(Automation_Testing_Project).

Open the folder- Right click with in the folder and select open gitbash here, you will get pop up like below and clone the repository based on the next steps provided below.



- **Activity 3:- Project Setup**

  Explained how to set up the project locally, including steps like:



Activity 3.1:- Clone the repository from github.

Activity 3.2:- Clone the BLUESTONE project repository from [repository URL]. https://github.com/viswatesting123/SmartBridge_Automation_Projects.g it

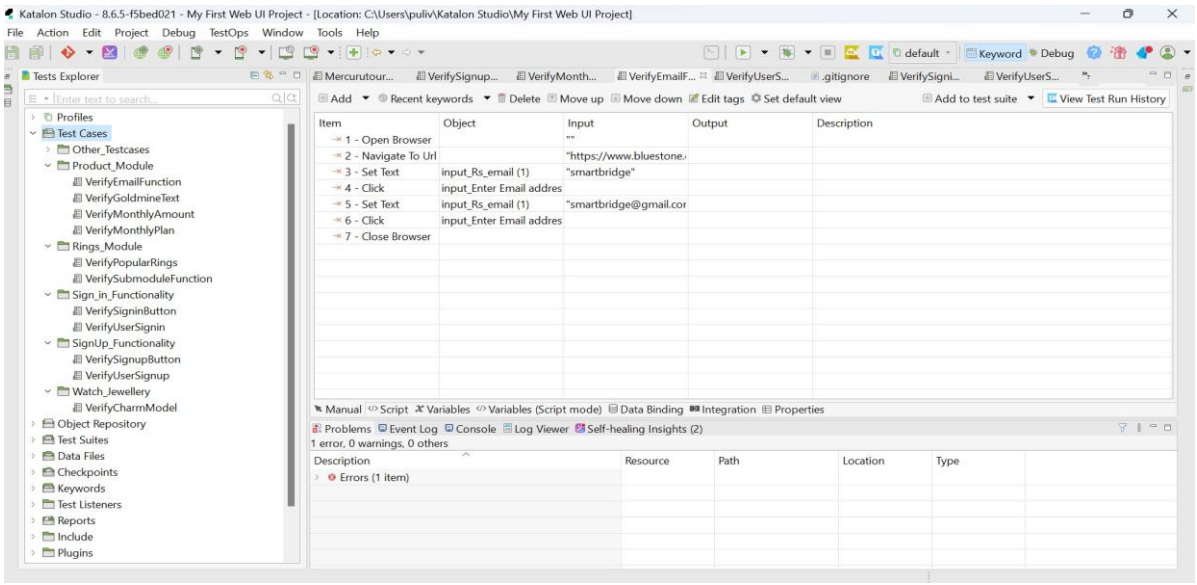
Activity 3.3:- Install all dependencies using requirements.txt (or)Requirements.txt containing all software's/libraries required for our projects. We don't need to install separately.

Activity 3.4:- RUN the command from the terminal.

Activity 3.5:- Configuration settings (if any).

Milestone 2: Project Structure

- **<u>Activity 1:- Structure of the project</u>**Explained the structure of the project, including directories and files, and their purpose



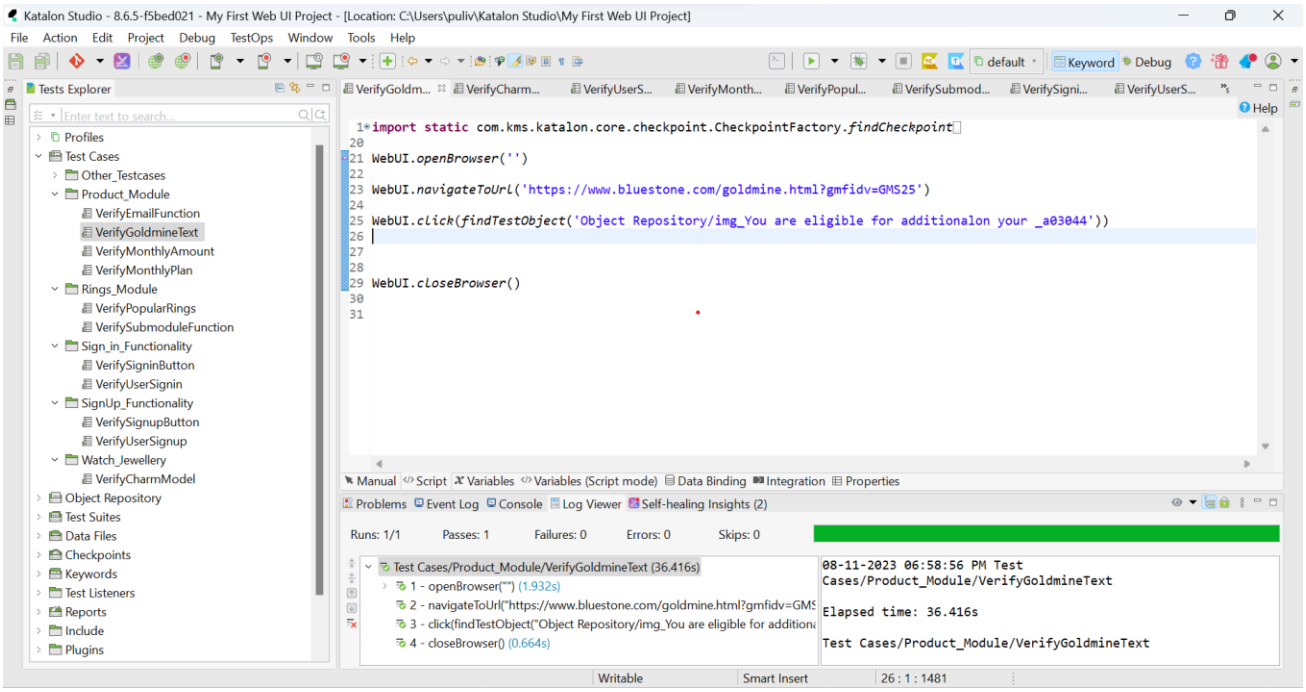
For example:

- **testcases/:** Directory containing test scripts
- **Objectrepository/:** Directory containing page object models
- **TestSuite/:** Directory containing testcase functions
- **reports/:** Directory containing test reports
- **DataFiles/:** Directory containing test data (ex: excel, csv, text, json files) related to project
- **Configuration/:** Configuration file for storing settings
- **Screenshots/:** Project testcases screenshots.
- **Plugins/:** Directory maintain the project related other dependencies.
- **Keywords/:** Directory containing global, custom keywords which are used to write testcases easily.

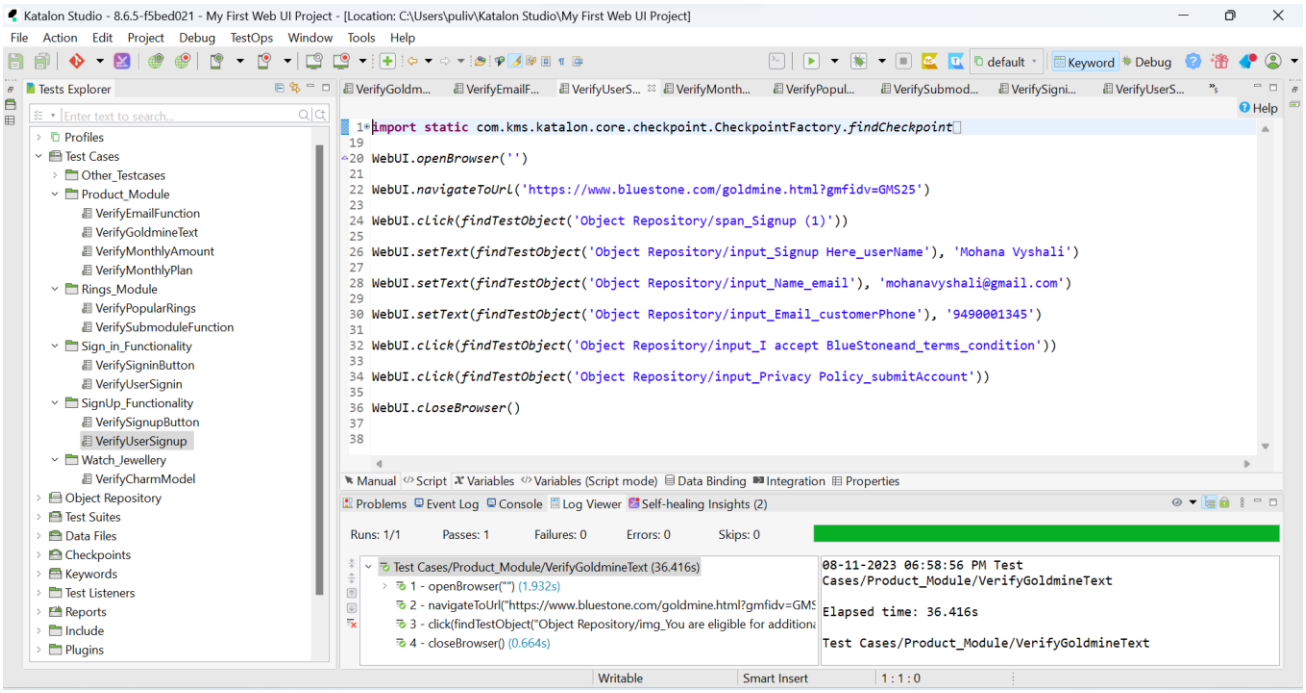
- **<u>Activity 2:-Writing Test Cases</u>**

Explained how to write test cases using Python, Selenium, and pytest.
Provide examples of different types of test cases, such as:

- Test case 1: Home Page functionality Screenshot of the code:



- Test case 2: Register new user functionality.



- Add new testcases as per Manual testcases prepared.

Include best practices for writing maintainable and readable test cases.

Milestone 3: Running Tests

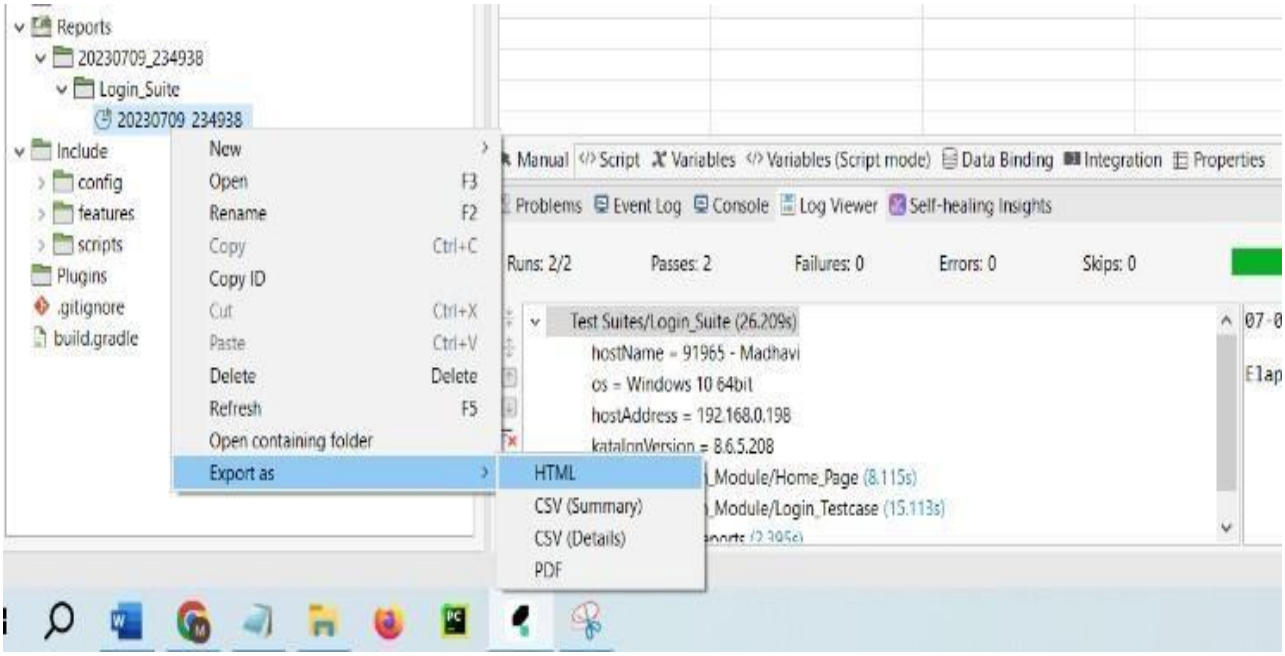
Explained how to run the test cases using Katalon studio. Include information about command-line and flags that can be used to customize test execution.

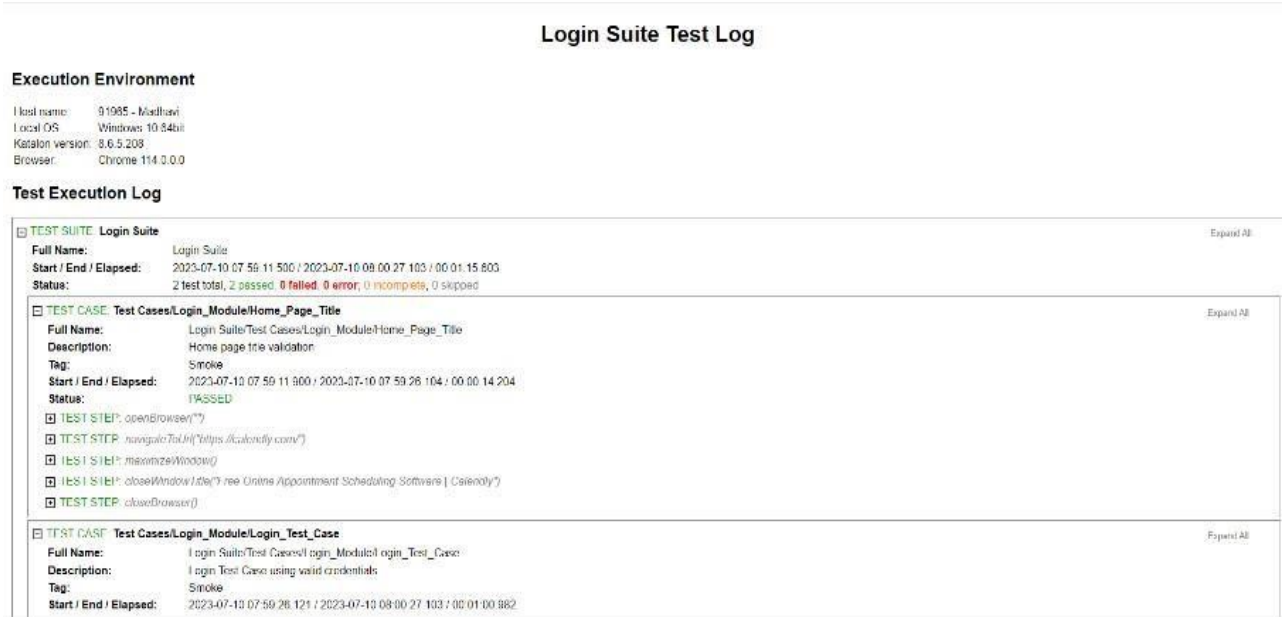
- **Activity 1:** Option to run all the test cases from the top quick access toolbar.

Milestone 4: Test Reporting

Explained how test reports are generated and where they are stored. Include information about any tools or libraries used for reporting, such as Katalon reporting-html.

- **Activity1:** reports/: Directory containing test reports





Login Suite Test Log

Execution Environment

Host name: 91985 - Madhavi
Local OS: Windows 10 64bit
Katalon version: 8.6.5.208
Browser: Chrome 114.0.0.0

Test Execution Log

TEST SUITE: Login Suite
Full Name: Login Suite
Start / End / Elapsed: 2023-07-10 07:59:11.500 / 2023-07-10 08:00:27.103 / 00:01:15.603
Status: 2 test total, 2 passed, 0 failed, 0 error, 0 incomplete, 0 skipped

TEST CASE: Test Cases/Login_Module/Home_Page_Title
Full Name: Login Suite/Test Cases/Login_Module/Home_Page_Title
Description: Home page title validation
Tag: Smoke
Start / End / Elapsed: 2023-07-10 07:59:11.900 / 2023-07-10 07:59:26.104 / 00:00:14.204
Status: PASSED

TEST STEP: openBrowser("")
TEST STEP: navigateToUrl("https://katalon.com/")
TEST STEP: maximizeWindow()
TEST STEP: closeWindowTitle("Free Online Appointment Scheduling Software | Calendly")
TEST STEP: closeBrowser()

TEST CASE: Test Cases/Login_Module/Login_Test_Case
Full Name: Login Suite/Test Cases/Login_Module/Login_Test_Case
Description: Login Test Case using valid credentials
Tag: Smoke
Start / End / Elapsed: 2023-07-10 07:59:26.121 / 2023-07-10 08:00:27.103 / 00:01:00.982

Advantages:

1. **Uses human intelligence to find errors:**

   Manual testing allows testers to use their higher cognitive abilities, such as deductive reasoning and common sense when detecting software defects. This helps them find errors that automated tests might miss.

2. **Let's testers focus on complex features and functions:**

   Using automated tests to emulate every single scenario related to a feature can be time-consuming. Manual testing allows testers to spend less time concentrating on emulating each unique user path possible and more time thinking about edge cases for complex features and functions.

3. **Tester knowledge of the project:**

   Manual testing allows testers to monitor the quality of a product throughout its development cycle by spending more time familiarizing themselves with its features and functions. This helps maintain their knowledge of the project, which will be useful if issues arise after changes have been made to the code.

4. **Detects errors outside of the code:**

Manual testing means testers look for bugs that do not affect the code, such as server response time. However, most traditional automated test tools are limited to what they can detect wthin the product's code.

5. **It helps maintain a testable system:**

   Manual testing can help testers use their expertise in detecting bugs that could cause an application to become untestable because automated tests cannot test the system they are a part of.

Disadvantages:

1. **It requires more time than automated testing:**

   Manual testing requires testers to spend additional hours manually checking every feature and function of the application. This takes up more time than running automated tests, requiring only a fraction of that time. Automated tests are also able to run overnight without any supervision.

2. **It is susceptible to human errors:**

   Manual testing is susceptible to human errors, such as bugs caused by tiredness, lack of focus, and distraction. Such bugs might not be detected if they are overlooked during testing. On the other hand, automated tests are less likely to make mistakes compared with their manual counterparts because testers no longer need to spend time manually checking for errors after each release.

3. **It is time-consuming to maintain test cases:**

   It takes more time to document all the manual errors found in an application, making it harder to keep track of changes when they are later made. This can be done effectively by maintaining automated tests instead because they only require updating if new features or functions are added.

4. **Testers need to know the product well:**

   Manual testers must have a solid understanding of a product before using it effectively. This helps them discern any errors that automated tests might miss if their knowledge is insufficient. Automated tests do not require testers to be familiar with all its features and functions.

5. **It is costly to maintain manual testers:**

Manual tests require not only time but also money since they need to be conducted by human beings (testers) instead of machines (automation tools). Automated tests are cheaper because the cost of running them does not depend on how many times they run - rather, it depends on how much time is spent coding and maintaining them.

Troubleshooting:

List common issues that may occur during project setup or test execution and provide possible solutions.

Troubleshooting BlueStone manual testing involves identifying and resolving issues that arise during the testing process. Here are some common troubleshooting steps you can follow:

- **Identify the Problem:**

  First, carefully document and reproduce the issue you encountered during manual testing. Identify the exact steps that led to the problem, including any specific inputs or configurations used.

- **Check Test Environment:**

  Ensure that your test environment is set up correctly, including the browser version, operating system, and any required plugins. Verify that BlueStone is running on the correct server and that all dependencies are properly configured.

- **Verify Test Data:**

  Double-check the test data you used during testing. Ensure that it is valid and properly set up to simulate real-world scenarios accurately.

Conclusion:

The BLUESTONE automation project provides a framework for automating various tasks within the BLUESTONE web application using Python, Selenium, and Pytest. With this documentation, you should be able to set up the project, execute tests, and extend/customize it according to your needs.