Test Automation Framework Technical Document

Contents

[I. Test Automation Framework Overview 3](#_Toc98872522)

[II. Framework Components 4](#_Toc98872523)

[III. Scalability and Maintenance 6](#_Toc98872524)

[IV. Technology Used 7](#_Toc98872525)

[V. Local Execution 8](#_Toc98872526)

[VI. Limitations 9](#_Toc98872527)

# Test Automation Framework Overview

Graphical user interface

Description automatically generated

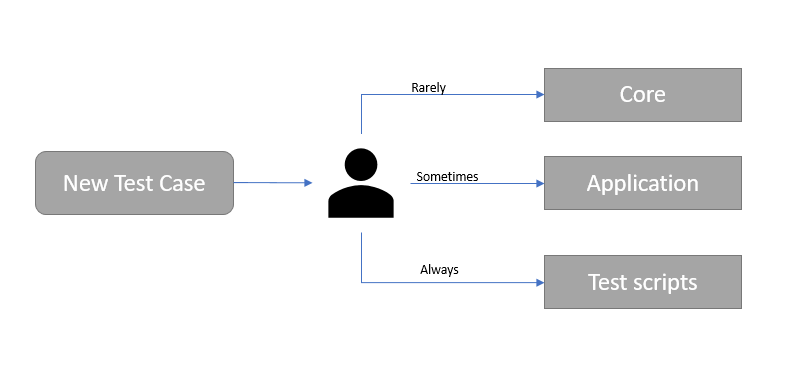
The illustration above gives a quick summary how the Automation Framework functions for this exercise. There are 3 main components Core, Application and Test Scripts (details are explained in section II).

# Framework Components

* 1. **Core** – contains base codes and generic codes that can be used anywhere.
     1. Core libraries – These are the main libraries used in automation. This includes Unit, API/Service Testing and UI Automation.
        + GUI Testing Lib– Used in UI Automation. Selenium library is used to interact with the browser drivers that is responsible in interfacing with the actual browsers.
        + API/Service Testing Lib – Core component was not implemented in this exercise
        + Cucumber – this is used for the behavioral driven development. Which is - a process that encourages collaboration among technical and non-technical persons in a software project. Cucumber utilizes Gherkin as the business language which are bound into a java code.
  2. **Application** – contains application specific codes.
     1. Hooks – Module was implemented to load configuration and necessary data needed for a specific type of test. This utilizes Cucumber annotations. This lets you perform code executions before and after certain scenarios and steps.
     2. Page Objects – Used in UI automation only. These are object repositories for the UI elements. The page objects will act as an interface for the page of application under test.
     3. Configurations – These are application configurations in yaml files specific to applications
     4. Data files – Place holder for future data
  3. **Test scripts** – contains the application specific tests scripts
     1. UI Automation – Test scripts written for User Interface automation. Simulates how a manual user do testing in Browsers. The framework supports UI automation for Chrome only. The framework can be extended and add support for different browsers with different configurations.
     2. API Automation or Service Automation – Core component was not implemented in this exercise
     3. Gherkin – The domain specific language. Describes what is being tested.
     4. Feature Files – This is the entry point for the Tests. This is where the Gherkin Test cases are written.
     5. Steps – These are the step definitions containing the Java code. These are the codes linked into the Gherkin steps.

# Scalability and Maintenance

As mentioned in Section 1, there are 3 main sections of the automation framework namely Core, Application and Test Scripts. If you add new test cases, you can see the frequency below on what components they need to change. See illustration below:



|  |  |
| --- | --- |
| Areas | Criteria of Update |
| CORE | only updated for core functionally changes. Like changes in the web driver |
| APPLICATION | only updated when there are new pages,  configurations for an application |
| TEST SCRIPTS | will be updated for every test case created |

As illustrated above, core and application are not always updated. Framework is trying to achieve decoupling of the key functionalities that can be re-used all throughout out the script development.

The framework utilizes the Page Object Model Class pattern and Page Factory in Selenium to perform operations on UI elements.

# Technology Used

|  |  |
| --- | --- |
| **IDE** | IntelliJ (JDK 16) |
| **Programming Language** | Java |
| **UI Auto** | Selenium |
| **BDD** | Cucumber |
| **API Automation** | Not implemented |
| **Build Tool/Dependency Manager** | Maven |
|  |  |

# Local Execution

* + - * Pre-requisites:
      * Java JDK 16 (Other JDK versions were not tested)

Text

Description automatically generated

* + - * IntelliJ IDEA IntelliJ IDEA 2021.2 (Community Edition)

Graphical user interface, text

Description automatically generated

* + - * Steps :

1. Load the project in IntelliJ
2. Install dependencies located in Pox.xml
3. Run Tests
   1. Navigate to feature files > IDEA Run Tab > Run ‘Scenario xx’

# Limitations

Lacks custom logger

Does not provide support for Headless executions for UI

Does not provide support for other browsers

Lacks custom Reporting/Dashboard