**TEST AUTOMATION APPROACH**

**Application**

<https://buggy.justtestit.org/>

**Goals and Objectives**

The goal is to increase the effectiveness, reliability and efficiency of testing the given application. The framwork will automate the regression testcases that will reduce the number of test cases that testers have to perform manually as well as those that are challenging to test manually, therefore saving time and effort and have a wide test coverage.

**Approach**

An automation framework is developed using Selenium tool for functional testing using Java language. A testing tool Cucumber is integrated with Selenium to support Behavior Driven Developemnt (BDD) that makes it easy to read and understand the application flow and behavior in simple Business language or English text using Gherkin language. This approach to develops the Cucumber BDD framework acts as a bridge between all the involved entities like developers, testers, business analysts etc.

Selenium uses browser’s native support, webdriver, to access the browser and perform actions on the browser. Selenium webdriver captures the element’s locators on the web page that could be used in the automation framework to access the web page.

Implementation involves:

* A Maven project is created to build and manage the project. The POM.xml stores all the dependencies for cucumber, junit, selenium etc.
* Creating feature files in which feature and its corresponding scenarios are written step by step using Gherkin language following the Scenario, Given, When, Then format. Stored in Features folder with .feature extention.
* Then the Step Definitions for each of the line in feature file is created which is the backend code/script to execute those steps.
* To execute, a Test Runner file is is created that glues feature file to the step definitions.
* 3 types of Reports are generated Html, Json and Junit that shows the result of test execution.
* Other than these, a Base class is created that contains @Before and @After testcases that requires to run before each testcase like initializing driver and open browser before each testcase and closing browser and quiting driver after each testcase. In the same base class, a function to execute implicit wait is written that takes locator as a parameter and wait until that webelement is visible on the screen.
* The Driver folder contains chrome webdriver for both linux and windows.
* A java class Common is created for all the shared functions that would be used in the framework like random string generator.
* Page Object Model (POM) is implemented that maintains the object repository and stores all the Pages sepeartely with their locators and functions. These are called by step definitions to implement the action written in feature file.

The following features are covered in automation framewrok:

* Register

This feature is tested by creating random strings to fill up registration form.

* Login

This feature is tested by sending 2 sets of data, one with cirrect credentials to see successful login and other with incorrect credentials to see login fails.

* Edit Profile

The feature is tested with different datasets to test if edit profile details works fine.

* Voting

A new user is registered eveytime to test this functionality because one user can vote only once for a particular entity. So the register steps are called to register a new user, then login with that new user, vote for the entity with the comment that is being passed to the test from feature file and then verufy that the vote is submitted from final vote table.

* Logout

This feature is to test that is logout is successful from various pages/screens of application.

From the above testcases, few bugs have been identified, due to which some tests will fail in the code test execution. These bugs and the other identified bugs are recorded in the other document (bugs.odt).