TEST AUTOMATION SOLUTION FOR INSTITUTE STUDENT MANAGEMENT SYSTEM

Preety Joshi[1] and Shahid Ali[2]

[1]The Testing Consultancy Software Assurance Company, Auckland, New Zealand

[2]Department of Information Technology, AGI Institute, Auckland, New Zealand

ABSTRACT

Testing is considered a key element in any business, especially in product development. Similarly, software development testing plays a key role in ensuring the quality of products. The challenge to perform manual testing is enormous as it takes huge time and effort, especially in rapidly development environments. To tackle this challenge tech experts, deploy automation testing. The focus of this research is to test web-based application via automation scripts. This research presents a proposed solution for Student Management System Test Automation Solution (SMS TAS). The solution is developed to test the functionality of Student Management System automatically. The institute SMS was a web-based application and was developed to manage students, staff, enrolments and other related tasks.

KEYWORDS

Student Management System, Test Automation Solution, Application Programming Interface, Page Object Model, Cucumber.

1. INTRODUCTION

Test automation is the most efficient and effective approach to increase the productivity of the business. The focus of this research is to perform test automation of the institute Student Management System (SMS), to improve the quality of services provided by the institute. The methodology for creating Test Automation Solution (TAS) has been selected after evaluation of tools that best suit the project requirement. Institute SMS web application provides a secure login to institute authorized administrators. After login each administrator can reach to SMS dashboard, where administrator can create, edit and delete records related to four main categories which are organization, academics. staff and enrolment. Institute SMS TAS is designed and developed to automate functional testing of an application of SMS. This automated test suite can be used to perform regression testing on SMS, which will help the institute to save time and cost on regression testing.

SMS TAS architecture follows advanced design practices. The solution is based on layered architecture, which provides a robust solution to test SMS. Technologies used to develop this solution are Java programming, Selenium WebDriver, Cucumber and Maven. There are three layers of this test automation solution. The first layer is SeleniumWrapper which includes robust and reusable methods (launchBrowser, launchApplication, quitBrowser, click, etc.) to automate user actions on web browser and create consistent logging. The second layer is Page Objects, which includes element locators of web application pages (institute SMS). The third layer is feature files, which includes tests in Behavior Driven Development/Gherkin style [8]. The scope of this testing solution includes development of SeleniumWrapper, Page Objects of institute SMS, write test scenarios in BDD style and develop test automation scripts for institute SMS. The testing covers the features/functionality such as Valid Login, Invalid Login, Logout, and

DOI: 10.5121/csit.2025.151607

Dashboard header, Dashboard Left Panel, Add Organization Institution, Add Academics Module, Add Staff Detail and Add Enrolment Intake.

This research paper is organized as follow: Section 2 focuses on the literature review of the test automation. Section 3 is focused on the research execution for this research. Section 4 explains execution results for this research. Section 5 provides the discussion on the results of this research. In section 6 recommendations for future research are provided. Finally, in section 7 conclusion to the research is provided.

2. LITERATURE REVIEW

The software automation testing has been considered more efficient and convenient to perform testing on the web-based application [1]. There are some automations testing frameworks available in the market such as Data-driven, keyword driven and Hybrid Framework which can be used to develop an effective framework [2]. The institute SMS Test Automation Framework has been designed by using Selenium which is an open-source tool. Although there are a lot of other tools which are available such as HP-QTP, TOSCA, LoadRunner and testComplete. Selenium is considered portable and web applications supporting different browsers [3]. Selenium was developed by Jason Huggins in 2004 [3]. The automation framework can be developed in many languages such as .Net, Php, java etc. but for this project java has been used, which is an open source and platform independent language [4]. For writing the test cases there were options available such as writing manual test cases in excel, or tools available such as Enterprise Tester where test case can be written. For this framework test cases are written with the help of Cucumber, which is a tool based on Behaviour Driven Development, this tool is used to write acceptance tests for web-based application written in Gherkin style which is easily understandable by all stakeholders in the project [5]. The framework helps to keep logging automatic, consistent and re-usable. In this framework, the data driven testing has been achieved by providing test data from Cucumber Feature files. Further, parameterization has been achieved by Cucumber Examples table, which helps to use the same code to test multiple sets of data for a test scenario [5].

The extent report is the library for selenium WebDriver, which generate impressive execution reports. These reports can be integrated with TestNG or Cucumber to execute the results and get great visualisation within the project to visualize test results graphically. A scrum approach has been followed in this project, which will assist split work in an iterative process. The result of each iteration gives an increment of a product and risk reduction. In addition to this, adopting Scrum methodology brings transparency in the project [6]. Moreover, there are other approaches like Waterfall or V-model. Waterfall has its own disadvantages like if any change is required to be implemented, it must be done once all the phases have been achieved. These models are not suitable for such projects because of less flexibility. Each phase is dependent on the other and time consuming. It is considered an inflexible model. V-Model [7] on the other hand does not provide a transparent path for the problems which are identified at the testing phase.

3. RESEARCH EXECUTION

The research execution steps for this research are given below.

This section presents the communication plan for 5 weeks sprint planning shown in Table 1. The entire process of test planning to test closure activities has been explained under this section. The institute SMS TAS is the outcome of sprint planning which has been distributed in 5 sprints. In each sprint multiple tasks have been performed**.**

Sprint 1			
Tasks Covered in Sprint	Status	Priority	Est. hrs
Project Test Planning	Done	High	8
Research on tools	Done	Medium	8
Creation of Proposal	Done	High	8
Installation of tools	Done	Medium	6
			Total = 30

Sprint 2			
Tasks Covered in Sprint	Status	Priority	Est. (Hrs)
Designing of scripts	Done	High	8
Exploratory Testing on institute SMS Website	Done	Medium	6
Research on Project Report	Done	High	8
Creation of feature file	Done	Medium	8

Sprint 3			
Tasks Covered in Sprint	Status	Priority	Est. (Hrs)
Research on Project Report	Done	High	7
Writing Report	Done	Medium	6
Writing automation scripts	Done	High	7
Compiling Execution Results in Report	Done	Medium	10

Sprint 4			
Tasks Covered in Sprint	Status	Priority	Est. (Hrs)
Execution and implementation	Done	High	7
Creation of Sprint Table	Done	Medium	8
Writing automation scripts	Done	Medium	6
Cocluding Recommendations and Suggestions for report	Done	Medium	9

Sprint 5			
Tasks Covered in Sprint	Status	Priority	Est. (Hrs)
Writing Project Execution part	Done	High	7
Creation of reports and analysing	Done	Medium	8
Compiling report and improvements	Done	High	6
Meeting exit criteria and finalising	Done	Medium	9

Table 1: Sprin Planning Communication Plan

3.1 Test Planning

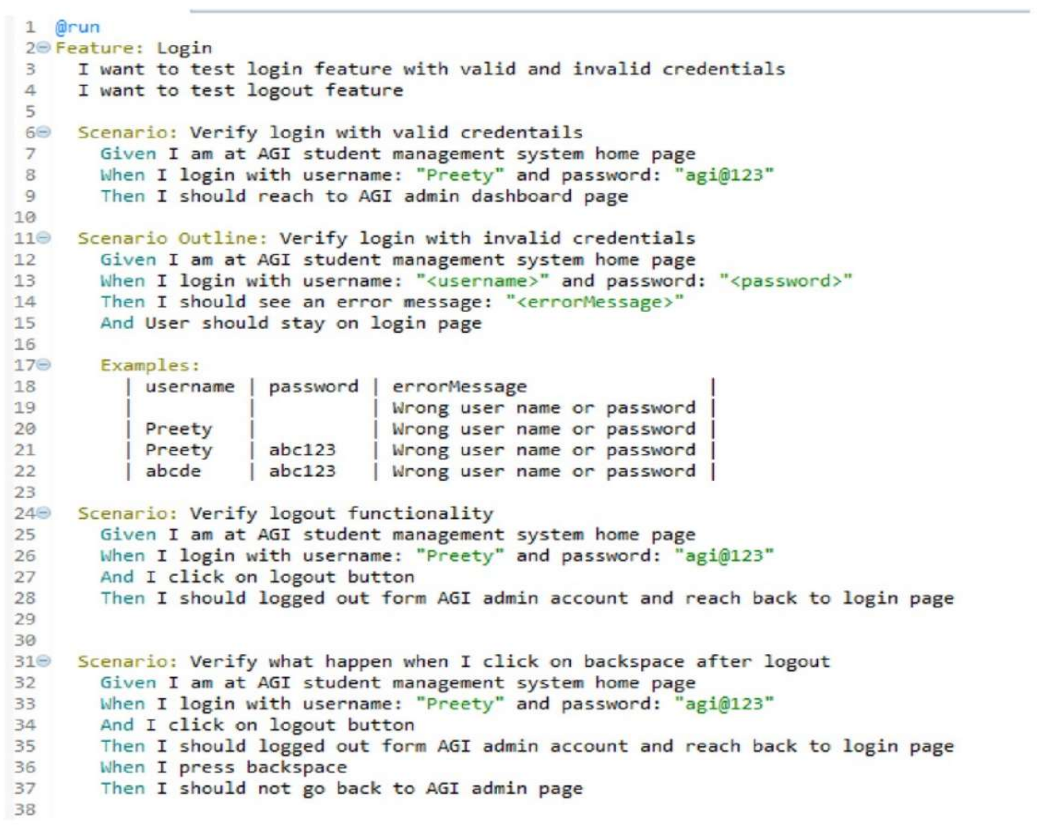
This was the first step before proceeding towards further implementations. Test objectives were identified at this stage and the decision was made on resources and testing efforts required. The objective of this task is to design a plan to develop layered test automation solutions and automate test scenarios (Valid Login, Invalid Login, Logout, New Institution, Edit Institution, Delete Institution, Search Institution, New GPO and Delete GPO) of institute SMS web application.

3.2 Analysis

Test conditions identified on the basis of test objectives. Due to unavailability of a specification, exploratory technique [9] used to identify test conditions. Prioritization of test conditions has been done to identify the most important test conditions. These identified test conditions have been used for preparing test data, test input
and test outcomes.

3.3 Design

Test cases were created in cucumber feature files which exercise identified test conditions. As our approach is to create tests is scenario based, therefore logical test case approach preferred. Each test case has an objective, preconditions, test data, expected result and post-conditions. Figure 1 represents the login feature file and test scenarios. Login feature file covers all the features related to login and logout functionality for institute SMS. In figure 1 login scenarios with valid and invalid credentials are written. The Scenario outline covers multiple sets of test data for invalid credentials with BDD style test cases:

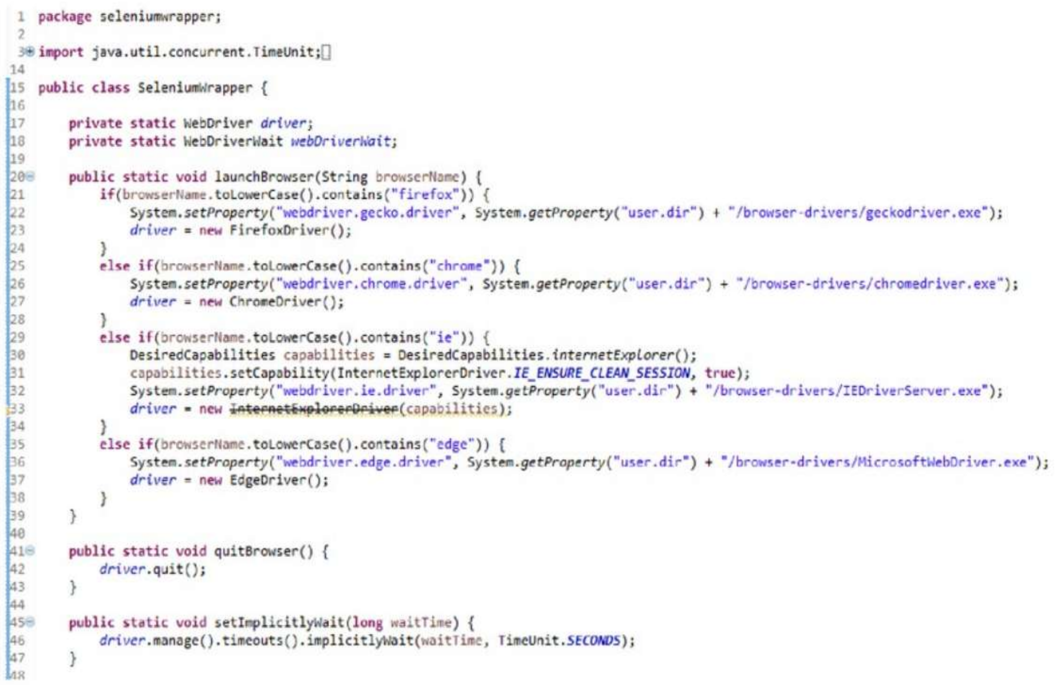


```
@run
Feature: Login
  I want to test login feature with valid and invalid credentials
  I want to test logout feature

  Scenario: Verify login with valid credentails
    Given I am at AGI student management system home page
    When I login with username: "Preety" and password: "agi@123"
    Then I should reach to AGI admin dashboard page

  Scenario Outline: Verify login with invalid credentials
    Given I am at AGI student management system home page
    When I login with username: "<username>" and password: "<password>"
    Then I should see an error message: "<errorMessage>"
    And User should stay on login page

    Examples:
      | username | password | errorMessage               |
      |          |          | Wrong user name or password |
      | Preety   |          | Wrong user name or password |
      | Preety   | abc123   | Wrong user name or password |
      | abcde    | abc123   | Wrong user name or password |

  Scenario: Verify logout functionality
    Given I am at AGI student management system home page
    When I login with username: "Preety" and password: "agi@123"
    And I click on logout button
    Then I should logged out form AGI admin account and reach back to login page


  Scenario: Verify what happen when I click on backspace after logout
    Given I am at AGI student management system home page
    When I login with username: "Preety" and password: "agi@123"
    And I click on logout button
    Then I should logged out form AGI admin account and reach back to login page
    When I press backspace
    Then I should not go back to AGI admin page

```

Add figure 1: login Feature

3.4 Implementation

Once the test design is completed then automation scripts are written. Test cases divided into different test suites like login/logout related tests are in different feature files and institutions related tests are in different feature files. As far as implementation is concerned, test scripts are managed in separate packages. Each package is responsible for different tasks. For example, seleniumwrapper contains implementations based on Selenium, pages contain implementation to create page objects and steps contain implementation for step definition. Below are the examples of these three packages. Figure 2 shows Selenium Wrapper class, figure 3 shows one page object class and figure 4 shows step definition class. These classes are explained in the results section 4.

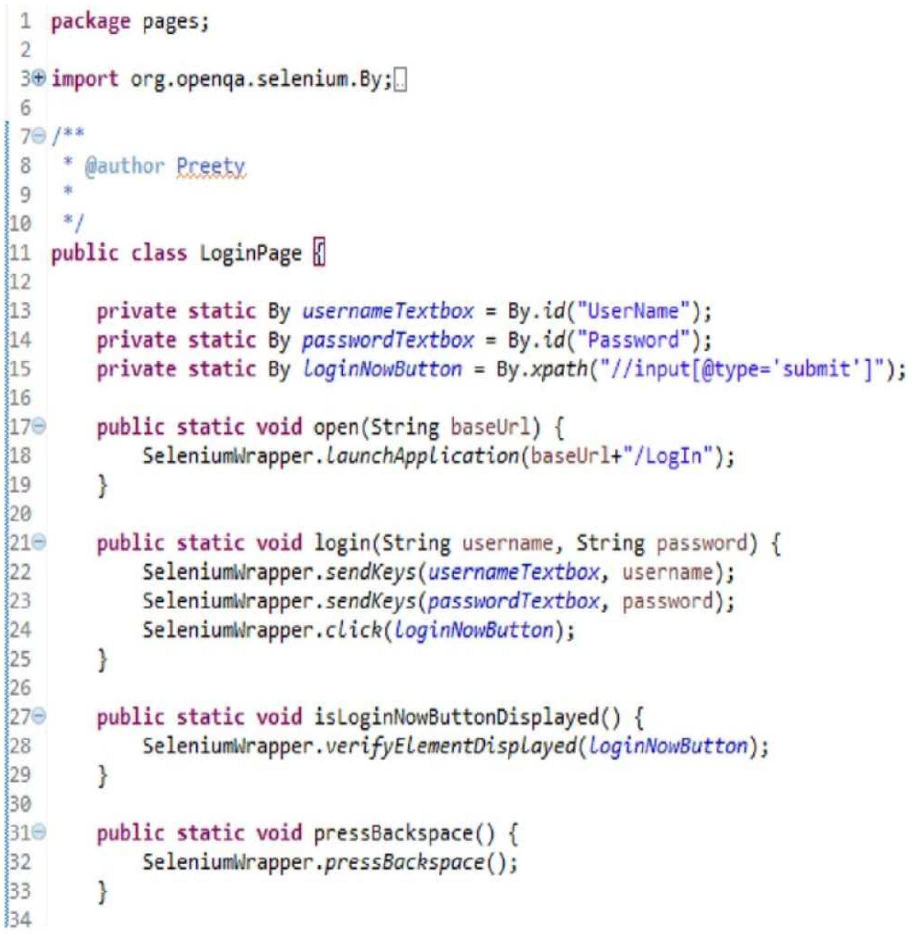


```java
package seleniumwrapper;

import java.util.concurrent.TimeUnit;

public class SeleniumWrapper {

    private static WebDriver driver;
    private static WebDriverWait webDriverWait;

    public static void launchBrowser(String browserName) {
        if(browserName.toLowerCase().contains("firefox")) {
            System.setProperty("webdriver.gecko.driver", System.getProperty("user.dir") + "/browser-drivers/geckodriver.exe");
            driver = new FirefoxDriver();
        }
        else if(browserName.toLowerCase().contains("chrome")) {
            System.setProperty("webdriver.chrome.driver", System.getProperty("user.dir") + "/browser-drivers/chromedriver.exe");
            driver = new ChromeDriver();
        }
        else if(browserName.toLowerCase().contains("ie")) {
            DesiredCapabilities capabilities = DesiredCapabilities.internetExplorer();
            capabilities.setCapability(InternetExplorerDriver.IE_ENSURE_CLEAN_SESSION, true);
            System.setProperty("webdriver.ie.driver", System.getProperty("user.dir") + "/browser-drivers/IEDriverServer.exe");
            driver = new InternetExplorerDriver(capabilities);
        }
        else if(browserName.toLowerCase().contains("edge")) {
            System.setProperty("webdriver.edge.driver", System.getProperty("user.dir") + "/browser-drivers/MicrosoftWebDriver.exe");
            driver = new EdgeDriver();
        }
    }

    public static void quitBrowser() {
        driver.quit();
    }

    public static void setImplicitlyWait(long waitTime) {
        driver.manage().timeouts().implicitlyWait(waitTime, TimeUnit.SECONDS);
    }
```

Figure 2: Selenium Wrapper Class

Page Object Class: This class contains the locators and methods for all the in-scope institute SMS web page. Methods in these classes are reusable and can be called when required under step definition classes. There are four classes created to fulfill the test coverage as per testing scope:

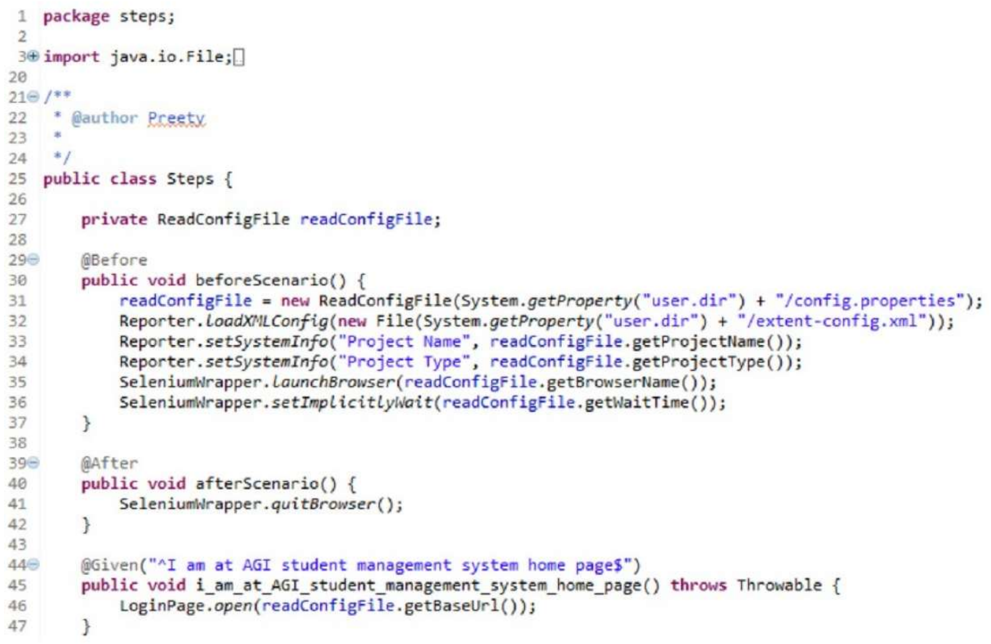
- LoginPage
- AdminDashboardMasterPage
- InstitutionPage
- GraduateProfileOutcomePage



```java
package pages;

import org.openqa.selenium.By;

/**
 * @author Preety
 *
 */
public class LoginPage {

    private static By usernameTextbox = By.id("UserName");
    private static By passwordTextbox = By.id("Password");
    private static By loginNowButton = By.xpath("//input[@type='submit']");

    public static void open(String baseUrl) {
        SeleniumWrapper.launchApplication(baseUrl+"/LogIn");
    }

    public static void login(String username, String password) {
        SeleniumWrapper.sendKeys(usernameTextbox, username);
        SeleniumWrapper.sendKeys(passwordTextbox, password);
        SeleniumWrapper.click(loginNowButton);
    }

    public static void isLoginNowButtonDisplayed() {
        SeleniumWrapper.verifyElementDisplayed(loginNowButton);
    }

    public static void pressBackspace() {
        SeleniumWrapper.pressBackspace();
    }

```

Figure 3: Page Object Class

Step Definition Class: This class contains step definition of test scenarios written under feature files. Steps: Contains test scripts under step definition methods. This class calls methods from page objects to perform the action on web page. This layer works with config.properties file to get configuration data e.g. browserName, baseUrl etc.



```
package steps;

import java.io.File;[]

/**
 * @author Preety
 *
 */
public class Steps {

    private ReadConfigFile readConfigFile;

    @Before
    public void beforeScenario() {
        readConfigFile = new ReadConfigFile(System.getProperty("user.dir") + "/config.properties");
        Reporter.loadXMLConfig(new File(System.getProperty("user.dir") + "/extent-config.xml"));
        Reporter.setSystemInfo("Project Name", readConfigFile.getProjectName());
        Reporter.setSystemInfo("Project Type", readConfigFile.getProjectType());
        SeleniumWrapper.launchBrowser(readConfigFile.getBrowserName());
        SeleniumWrapper.setImplicitlyWait(readConfigFile.getWaitTime());
    }

    @After
    public void afterScenario() {
        SeleniumWrapper.quitBrowser();
    }

    @Given("^I am at AGI student management system home page$")
    public void i_am_at_AGI_student_management_system_home_page() throws Throwable {
        LoginPage.open(readConfigFile.getBaseUrl());
    }
```

Figure 4: Step Definition Class

3.5 Execution

After completing test automation implementations, test execution has been performed. The execution is analyzed carefully to check automated tests should run as per expectation and should fail where they should be failed.

3.6 Evaluating exit criteria and reporting

The test scenarios, test scripts and result report analyzed carefully to check that all the scenarios planned to perform testing have been covered and tested. The failed test scenarios are reported to institute SMS concern persons with recommendations
and improvements.

3.7 Test closure activities

The institute SMS TAS and along with guidelines is shared with institute.

4. Results

4.1 Framework architecture

The layers of TAS architecture have been explained under this section:

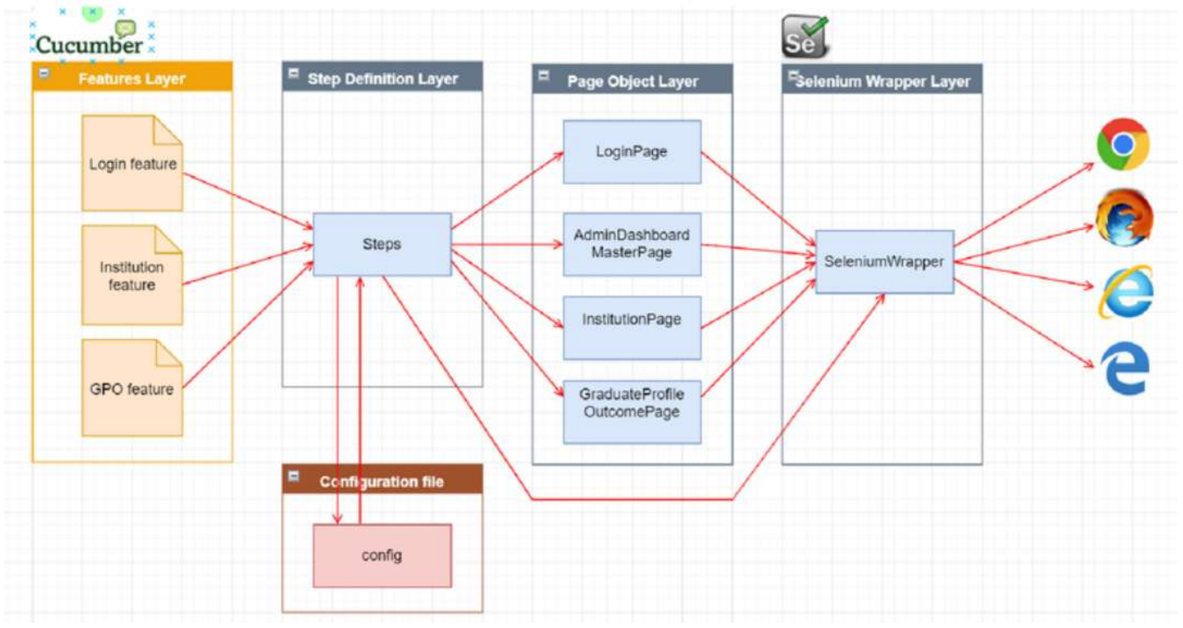


Figure 5: TAS Architecture

TAS for institute SMS is compatible with four browsers which are Mozilla Firefox, Internet Explorer, Microsoft Edge and Google Chrome. It has been observed that the execution speed on Google Chrome browser is faster followed by Internet Explorer. On Edge and Firefox, it is working as much similar in speed. Test Scripts can run on one browser at a time by changing the browser name in the configuration file.

Features Layer (package name: runner)

This layer contains BDD style test scenarios. The test scenarios written in BDD style are understandable for non-technical people and provide live documentation for test scripts. There are three feature files created to fulfill the test coverage as per testing scope:

1. Login feature: Contains test scenarios for valid login, invalid login and logout functionality.
2. Institution: Contains test scenarios for new, edit, delete and search institution.
3. GPO: Contains test scenarios for new and delete GPO (Graduate Profile Outcome).

Step Definition Layer (package name: steps)

This layer contains step definition of test scenarios under feature files. This layer is a connector between feature files and test scripts. There is one class created to fulfill the test coverage as per testing scope. This class contains test scripts under step definition methods. This class calls for methods from page objects to perform action on web page. This layer also communicates with config.properties file to get configuration data
(browserName, baseUrl, waitTime, etc.)

Page Object Layer (package name: pages)

This layer contains locators and methods for each institute SMS web page. The methods in these classes are reusable and can be called unlimited time under step definition classes.There are four classes created to fulfill the test coverage as per testing scope:

1. LoginPage: Contains locators and methods for institute SMS login page.

2. AdminDashboardMasterPage: Contains locators and methods for header and left panel under institute SMS admin dashboard page.
3. InstitutionPage: Contains locators and methods for institute SMS institution page.
4. GraduateProfileOutcomePage: Contains locators and methods for institute SMS graduate profile outcome page.

Selenium Wrapper Layer (package name: seleniumwrapper)

This layer contains selenium WebDriver based methods to perform actions on web browser. The methods in these classes are reusable and can be called unlimited time under step definition and page object classes. There is a class created to fulfill the test coverage as per testing scope:

SeleniumWrapper:

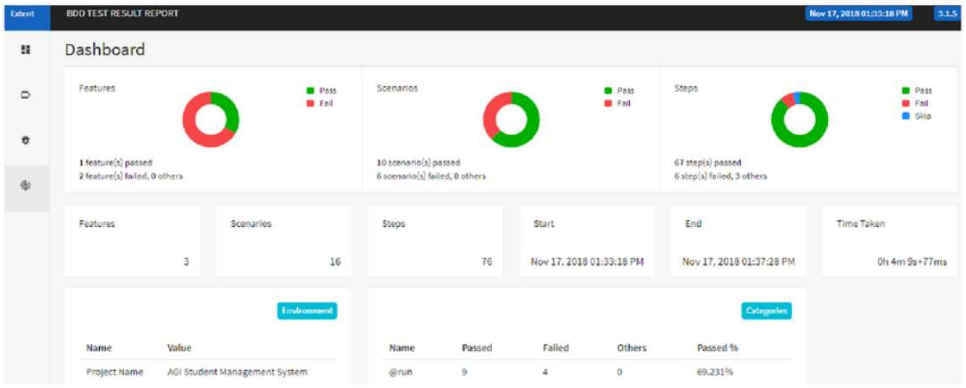
The SeleniumWrapper contains methods to communicate with web browser. The methods are as below:

a) launchBrowser (String browserName)
b) quitBrowser ()
c) setlmplicitlyWait (long waitTime)
d) launchApplication (String url)
e) click (By locator)
f) clearValue (By locator)
g) sendKeys (By locator, String value)
h) verifyElementDisplayed (By locator)
i) verifyElementNotVisible (By locator)
j) pressBackspace ()
k) isElementSelected (By locator)

Runner package

This package is not present in the architecture diagram. There is one class in this package i.e. TestRunner. This class contains code to run tests. Cucumber related plugins are also managed in this class.

4.2 To evaluate the efficiency and effectiveness of the proposed framework Test Automation Solution, its execution was conducted on institute SMS. The extent report has been generated to view the results of automation. This is evident in the extent report that the number of steps covered while automation was 76 in total time of 4m+9s+77ms. The time taken by automation testing using Selenium WebDriver was quick. The percentage test scenario passed is 69.231 and failed was 30.769. Figure 6 shows extent reports result.


Extent BDD TEST RESULT REPORT Nov 17, 2018 01:33:18 PM 3.1.5

Dashboard

Features
🟩 Pass
🟥 Fail
1 feature(s) passed
2 feature(s) failed, 0 others

Scenarios
🟩 Pass
🟥 Fail
10 scenario(s) passed
6 scenario(s) failed, 0 others

Steps
🟩 Pass
🟥 Fail
🟦 Skip
67 step(s) passed
6 step(s) failed, 3 others

Features	Scenarios	Steps	Start	End	Time Taken
3	16	76	Nov 17, 2018 01:33:18 PM	Nov 17, 2018 01:37:28 PM	0h 4m 9s+77ms

Environment

Name	Value
Project Name	AGI Student Management System

Categories

Name	Passed	Failed	Others	Passed %
@run	9	4	0	69.231%

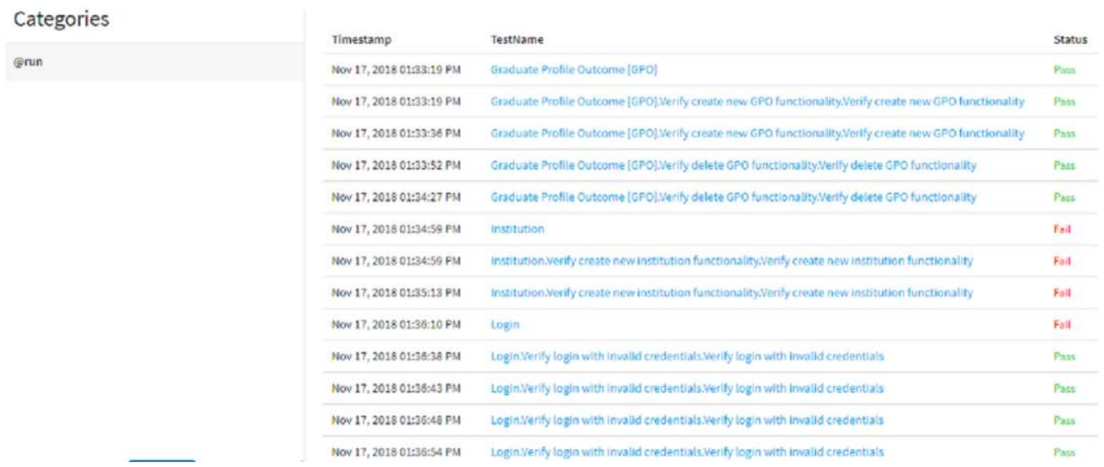
Figure 6: Extent Report

4.3 Figure 7 shows the categories of scenario created for SMS functionality testing along with their features, scenario name and status (Pass/Fail).

4.3.1 The scenario for new institution has failed because every time a user tries to create a new institution. The expected result is the creation of an institution, but it does not meet the expected criteria. Therefore, this test scenario has failed.

4.3.2 Login with invalid credentials should fail and therefore it is meeting the expected criteria.

4.3.3 When the user logout form institute SMS application and on home page click on the back button, user should not be directed to the Admin dashboard. But the user is directed to the admin dashboard, so the scenario has failed.


Categories

@run

Timestamp	TestName	Status
Nov 17, 2018 01:33:19 PM	Graduate Profile Outcome [GPO]	Pass
Nov 17, 2018 01:33:19 PM	Graduate Profile Outcome [GPO].Verify create new GPO functionality.Verify create new GPO functionality	Pass
Nov 17, 2018 01:33:36 PM	Graduate Profile Outcome [GPO].Verify create new GPO functionality.Verify create new GPO functionality	Pass
Nov 17, 2018 01:33:52 PM	Graduate Profile Outcome [GPO].Verify delete GPO functionality.Verify delete GPO functionality	Pass
Nov 17, 2018 01:34:27 PM	Graduate Profile Outcome [GPO].Verify delete GPO functionality.Verify delete GPO functionality	Pass
Nov 17, 2018 01:34:59 PM	Institution	Fail
Nov 17, 2018 01:34:59 PM	Institution.Verify create new institution functionality.Verify create new institution functionality	Fail
Nov 17, 2018 01:35:13 PM	Institution.Verify create new institution functionality.Verify create new institution functionality	Fail
Nov 17, 2018 01:36:10 PM	Login	Fail
Nov 17, 2018 01:36:38 PM	Login.Verify login with invalid credentials.Verify login with invalid credentials	Pass
Nov 17, 2018 01:36:43 PM	Login.Verify login with invalid credentials.Verify login with invalid credentials	Pass
Nov 17, 2018 01:36:48 PM	Login.Verify login with invalid credentials.Verify login with invalid credentials	Pass
Nov 17, 2018 01:36:54 PM	Login.Verify login with invalid credentials.Verify login with invalid credentials	Pass

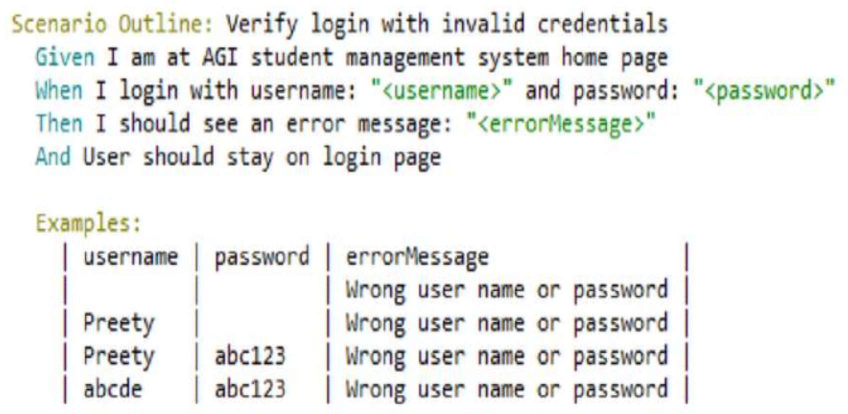
Figure 7: Feature, Scenario name and Status

5. Discussion

Regression testing is a very important part of Software Development Life Cycle. Institute SMS web application is under development, therefore new features/functionalities were kept added by developers. Due to these ongoing additions and changes to the application, there are high chances that existing features/functionalities could affect (break). Regression is the only way to test that existing features should not be affected because of new implementations. Performing regression testing before each release is an expensive and time-consuming task. Institute SMS TSA perform regression testing can be performed automatically by taking less time and providing confidence to management and stakeholders before each new release that all existing features are working as expected. Scrum methodology adoption for the institute SMS TAS has made test designing, implementation and execution easier. The testing activities have been performed iteratively. Below are the few contributions of the institute SMS TAS for filling the existing gaps:

- The institute SMS TAS is developed using scrum methodology. All the tasks of testing life cycle are achieved in sprints. By adopting this methodology over the traditional approach of waterfall and V- model, would allow making changes in the TAS when required. It also helps to build a better approach to handling software development. Moreover, it is considered fast and can adapt changes quickly [7].
- The regression testing effort is reduced by implementing this solution because in this solution we have test scenarios automated, which are required to run for regression testing before each release of institute SMS.

- Cucumber implementations provide live documentation for test scripts/cases within the testing solution; therefore, stakeholders can easily understand what is in the test scripts and how much test coverage has been achieved. However, the test cases written based on another testing framework are only test scripts, therefore it is required to write tests in some document (excel, word, etc.) separately to make it understandable for non-technical people.
- By using Cucumber Examples table, any test scenario can be run with unlimited set of test data without writing additional test scripts. As far as manual testing is concerned, it is very time consuming to test a single test scenario with 100 different sets of test data. It may take a day to complete. However, with the help of the institute SMS TAS, we just need to add test data under the Cucumber examples table, and all tests will run automatically by taking less than an hour. Figure 8 represents an example of a Cucumber example table which covers the invalid credentials for login functionality with multiple sets of data.



```
Scenario Outline: Verify login with invalid credentials
  Given I am at AGI student management system home page
  When I login with username: "<username>" and password: "<password>"
  Then I should see an error message: "<errorMessage>"
  And User should stay on login page

  Examples:
    | username | password | errorMessage              |
    |          |          | Wrong user name or password |
    | Preety   |          | Wrong user name or password |
    | Preety   | abc123   | Wrong user name or password |
    | abcde    | abc123   | Wrong user name or password |
```

Figure 8: Cucumber Examples Table

The institute SMS web application explored, and test scenarios have been automated as per its scope in the introduction of this research. While exploring the end-to-end functionality of institute SMS application, some weak areas have been identified which are mentioned below:

- Login sessions are not secured. After logging out from application, when user clicks the backspace button, it takes the user back to his/her session. Moreover, the user can also perform all actions on a user dashboard. It is a major security and functional bug and needs to be resolved.
- The new institution functionality is not working as expected. User failed to create a new institution. It is a major functional bug and blocker for further testing (edit, delete and search institution), therefore, it needs to be resolved.
- Login with invalid credentials brings an error message “Wrong username or password” in simple black color. For better usability it should be in red colour, so that it gives clarity to the user that something wrong happened.

6. RECOMMENDATIONS

After conducting this research study, the following recommendations are suggested below:

- Institute SMS TAS is developed by using java libraries and core java concepts. To improve it further, Object Oriented Programming concepts can be implemented to make the solution compatible for parallel execution.
- SeleniumWrapper layer can be extended with more methods.
- More features/test scenarios can be automated to get the maximum coverage.
- A method to capture screenshots when any test scenario failed can be added in future.
- Pre-conditions and post-conditions can be managed by making a direct connection with database.

7. Conclusion

This research covers the explanation of institute SMS TAS, which is developed to perform regression testing on institute SMS. The solution is based on layered architecture and page object model. The solution is robust, scalable and easily maintainable. The solution will help the institute to reduce the time and cost on regression testing for institute SMS. All technologies used to develop this solution are open source (Selenium WebDriver. Cucumber, Maven, and Extent Reports). The scrum methodology is used to manage the work product within 5 weeks. The bugs/issues found during exploratory testing reported in this research, along with the recommendations to improve this solution further.

7. References

[1] Vila.E., Novakova.G., & Todorova.D. (2017). Automation Testing Framework for Web Applications with Selenium WebDriver: Opportunities and Threats. Jn Proceedings of the International Conference on Advances in Image Processing (CAIP 2017). ACM, New York, NY, USA, 144-150.

[2] Laukkanen, P. (2006). Data-driven and keyword-driven test automation frameworks. Master's thesis. Helsinki University of Technology.

[3] Sharma, M., & Angmo, R. (2014). Web based automation testing and tools. International Journal of Computer Science and Information Technologies, 5(1), 908-12.

[4] Angmo, M. R., & Sharma, M. M. (2014). Selenium Tool: A Web based Automation Testing Framework.

[5] Soeken, M., Wille, R., & Drechsler, R. (2012, May). Assisted behavior driven development using natural language processing. In /nternational Conference on Modelling Techniques and Tools for Computer Performance Evaluation (pp. 269-287). Springer, Berlin, Heidelberg.

[6] Schwaber, K. (2004). *Agile project management with Scrum*. Microsoft press.

[7] Mahalakshmi, M., & Sundararajan, M. (2013). Traditional SDLC Vs Scrum Methodology A Comparative Study. International Journal of Emerging Technology and Advanced Engineering, 3(6), 192-196.

[8] Srinivas, S., & Goel, L. (2025). Designing and Implementing Robust Test Automation Frameworks using Cucumber BDD and Java. *arXiv preprint arXiv:2505.17168.*

[9] Pfahl, D., Yin, H., Mäntylä, M. V., & Münch, J. (2014, September). How is exploratory testing used? a state-of-the-practice survey. In *Proceedings of the 8th ACM/IEEE international symposium on empirical software engineering and measurement* (pp. 1-10).