Research Article

A New Hybrid Software Testing Automation Framework

Meral BOZDEMİR[1], Turgay Tugay BİLGİN [2], Kader NİKBAY OYLUM[3]*
[1] Mert Yazılım, 0000-0003-0199-6009, meralbozdemir@trex.com.tr
[2] Bursa Teknik Üniversitesi, 0000-0002-9245-5728, turgay.bilgin@btu.edu.tr
[3]Mert Yazılım, 0000-0002-5218-9218, kaderoylum@trex.com.tr
* Sorumlu Yazar: kaderoylum@trex.com.tr; Tel: 444 3 468

(First received January 19, 2023 and in final form March 24, 2023)

Reference: Bozdemir, M., Bilgin, T., T., Oylum K., N. A New Hybrid Software Testing Automation Framework. The European Journal of Research and Development,3(1), 200-211.

Abstract

Nowadays, with the development of software technologies in all areas of life, software testing, which is an indispensable need of the software life cycle, has become open to development and change. The replacement of manual testing of software products with test automation systems that minimise the human error margin provides the most important example of the transformation of testing processes into a dynamic structure.

This paper details a hybrid approach that utilizes tools such as Java, Maven, Test-NG, etc. used in a traditional test automation. The proposed hybrid approach is based on Java's "Write once, run anywhere" approach. First of all, all HTML elements used in web applications are determined and these elements are saved in a model called PageFactory. PageFactory is a Page object model concept that allows managing elements from a single file. These saved elements are transferred to the next layer, the Test Layer architecture, using the relevant methods in the Page Layer model, and the methods in this layer are invoked and executed. This proposed hybrid approach provides reusability, easy maintenance, cost reduction, increased performance, better quality software products, ease of use and time savings.

Keywords: *Software Engineering, Software Testing, Hybrid Test Environments*

1. Introduction

The importance of software testing is increasing day by day with the inclusion of software products in all areas of life. In this context, minimising the errors that may occur in the testing of software products will undoubtedly make a positive contribution to increasing the quality of the product.

Bhondokar et al. 2015 proposed a hybrid test automation framework that minimises human intervention and can be easily applied to different organisations with minimal changes in the future [1]. In Wang and Du's study, a Selenium and JMeter based software testing framework was designed for web applications, and it was found that this proposed framework increased the development efficiency of software products [2]. In another framework model proposed by Kim, Na and Ryoo, the architecture of a test automation framework that combines reusable components is presented [3].

In this paper, a new hybrid framework architecture is presented in the field of software test automation. In the second part of the paper, the technologies used, in the third part, the proposed framework design, in the fourth part, the working steps of the proposed design, and in the last part, the results and evaluations obtained from the study are mentioned.

2. Used Technologies

2.1. Java

Java is a widely accepted object-oriented programming language developed for writing portable, operating system independent, portable programs. Java is an open source programming language that is step-by-step executed and compiled [4].

2.2. Selenium

Selenium is an open source and free browser test automation tool. It supports all browsers and three main operating systems, Windows, Apple and Linux, and runs smoothly. It also allows the use of multiple programming languages such as C#, Python, Java for test scripts [5].

The page object model (POM) used in Selenium is a very common design model

used in test automation projects. Thanks to this model, code repetition is prevented and the maintenance of existing codes is much easier. Figure 1 shows an example of Page Object Model.

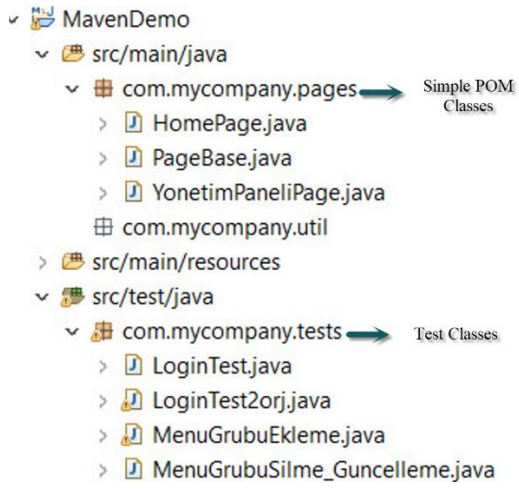

Figure 1: Page Object Model Example

2.3. Maven

Maven is an automation and build tool used to create and manage Java projects professionally. It enables the elimination of library dependency and IDE dependency in software. Maven also supports various software languages such as C#, Ruby, Scala, and Python [6].

2.4. TestNG

TestNG is a test framework developed with JUnit and NUnit logic. The most important differences from the other tools mentioned are; It allows all tests to be performed independently of each other, creates test reports, and allows the error messages of the tests that have errors as a result of the test to be added to the report. Some of the advantages of TestNG;

- Multi thread test support,
-Easy grouping of test scenarios,
- Parameter usage
- Flexible test configuration
- Data Driven testing with @DataProvider
- Plugin support for IDEs

- Logging
- Support for dependency testing [7].

3. Proposed Framework Design

The system architecture developed in this study is shown in Figure 2. The architecture consists of three main parts. These sections are configuration files, data provider and report generator sections. The test automation codes within the architecture are based on the "Object Oriented Programming (OOP)" approach.

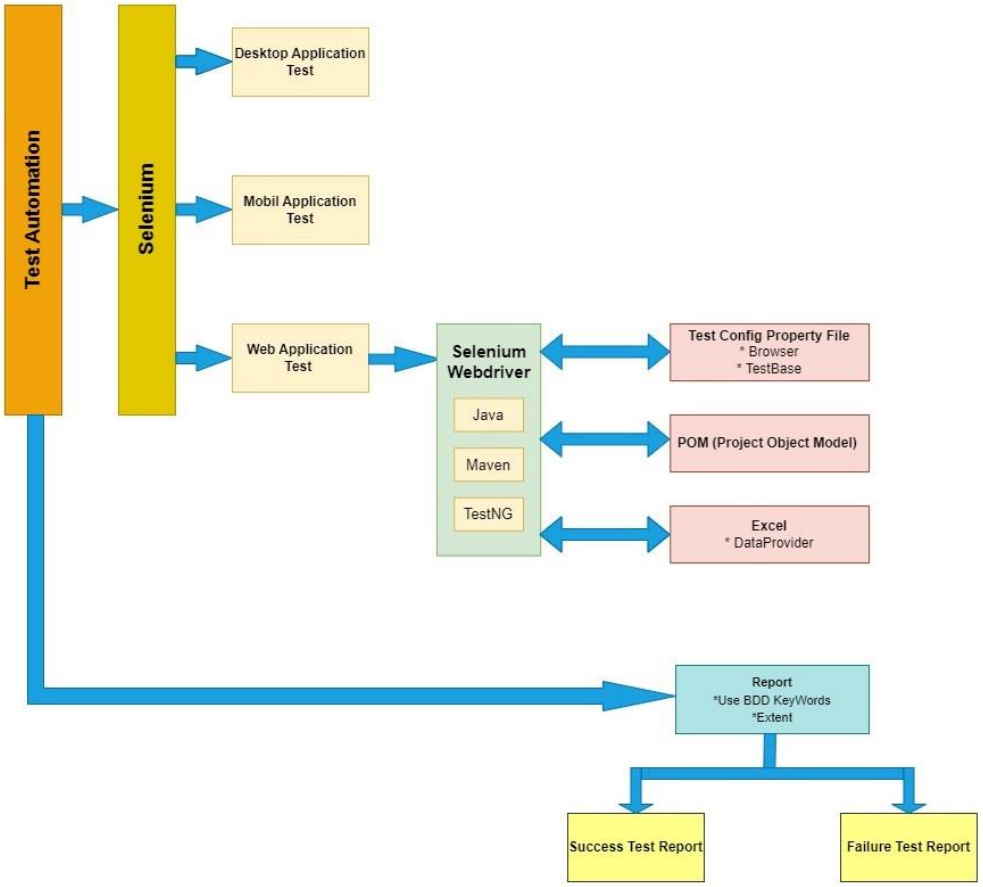

Figure 2: General Architecture of The System

3.1. Configuration Files

The file specified with the title "Test Config Property File" is the part where the general configuration settings of the tests are made for test automation (Figure 3).

```
#browser=Chrome
#browser=ChromeLocal
#browser=ChromeHeadLess
browser=FIREFOX

environment=test

qatest.baseUrl=http://test1.itcareercenter.us
test.baseUrl=http://test1.itcareercenter.us
dev.baseUrl=http://test1.itcareercenter.us
prod.baseUrl=http://test1.itcareercenter.us


excelFile = TestDataFile.xlsx
dataSheet=TestCaseDataSheet
```

Figure 3: Test Config Property File

The parameters in the Test Config Property File depend on two Java classes named TestBase and Browser. The TestBase class contains the code that runs the environment and the URL connected to this environment. It also provides the creation of the DataProvider and the enhanced Extend Report. Another task is to define repetitive codes in TestBase using TestNG Annotations. These annotations are presented in Table 1.

Table 1: TestNG Annotations

Annotation	**Description**
@BeforeSuite	It will run before the execution of all test methods in the project.
@AfterSuite	It will run after the execution of all test methods in the project.
@BeforeTest	It is executed before the start of all test methods of existing classes of the respective class.
@AfterTest	It is executed after the start of all test methods of the existing classes of the respective class.
@BeforeClass	Executed before the first method of the current class is called.
@AfterClass	It'll be called after the execution of all test methods of the current class.
@BeforeMethod	It's executed before any test method runs.
@AfterMethod	It runs after the execution of any test method.

Browser class contains the codes that enable the browsers in the Test Config Property File to run. Whichever browser is selected in the Test Config Property File file, that browser runs, and each of them can also be run on a different browser when parallel testing is required.

3.2. Project Object Model (POM)

It is an XML file containing the necessary information and configuration details for the build process about the project by Maven. This file is actually a project file that contains and manages information such as project dependencies, plugins or targets that can be run, build profiles, project version, description, developers, mailing lists. Figure 4 shows the contents of the POM file.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>Tracfone</groupId>
    <artifactId>TracfoneAutomation</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>API and Selenium WebDriver</name>
    <description>Use to automate Company Tests</description>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    </properties>

    <dependencies>
        <dependency>
            <groupId>com.github.wnameless</groupId>
            <artifactId>json-flattener</artifactId>
            <version>0.2.2</version>
        </dependency>


        <!-- https://mvnrepository.com/artifact/log4j/log4j -->
        <dependency>
            <groupId>log4j</groupId>
```

Figure 4: Project Object Model (POM) Example

3.3. Data Provider

It is a component providing data reading from data sources such as sql, text, excel. Firstly, the method name created as shown in Figure 5 is defined in the Excel file. The method name is found using DataProvider and the method parameters in the next column are taken. The specified process continues until the parameters in the relevant field are finished. When the related parameters are finished, if there is a new method in the next row, the same process cycle continues. With this approach, hundreds of tests can be run using a single DataProvider.

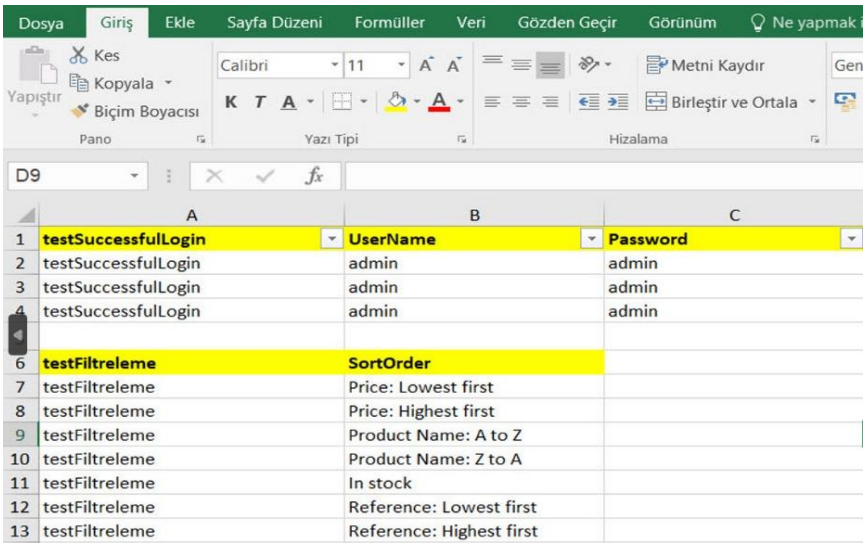

Figure 5: Example of Excel File

3.4. Report Builder

The report generator which is one of the architectural components is an open source, HTML reporting library for Java and .Net programming languages that enables the automatic generation of test reports in Selenium test automation [8]. The advantages of Extent reports can be listed as follows;

- More customisable than other report types,
- It can capture screenshots at each test step,
- The time and date information for the executed test can be displayed in the report,
- All test scenarios can be run on a single report screen and the results can be easily monitored,
- It can be easily integrated with frameworks such as JUnit, NUnit and TestNG,

An example of a successfully completed test report generated by Extent report is shown in Figure 6.

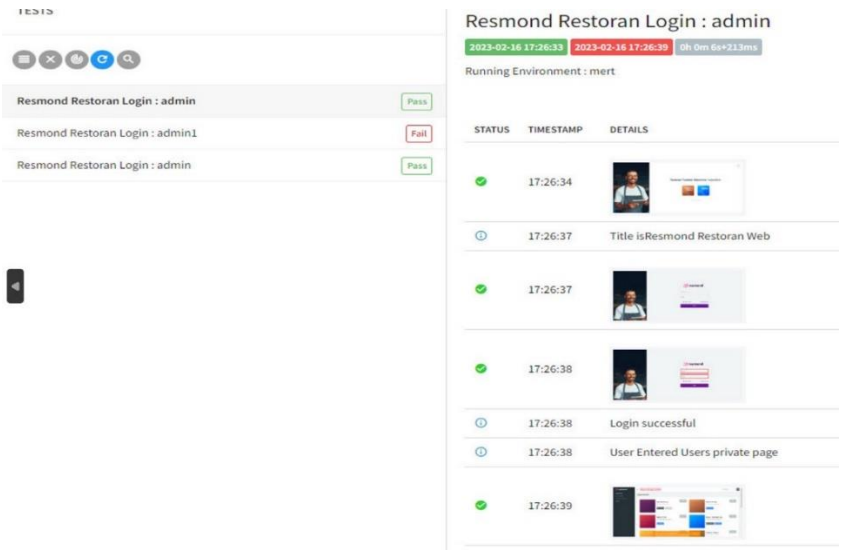

Figure 6: Successful Test Example

An example of an errored test report generated by Extent report is shown in Figure 7.

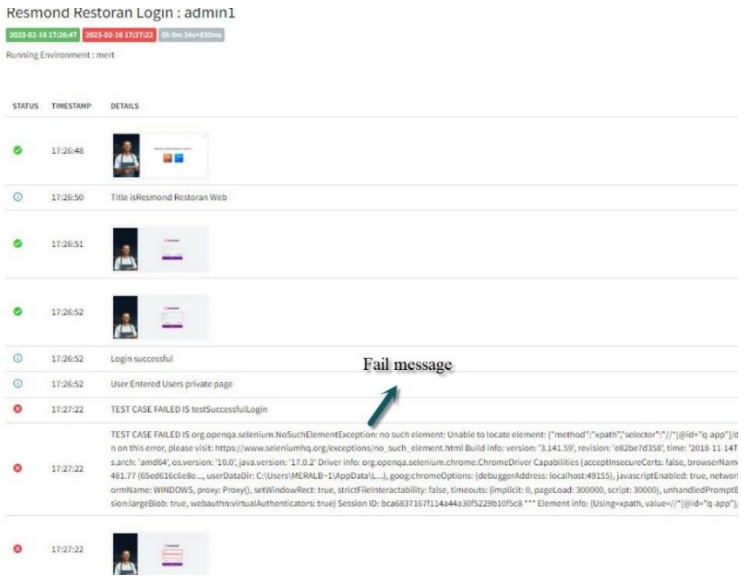

Figure 7: Fail Test Example

Another approach to test report generation is open source Cucumber's Behaviour Driven Development (BDD) approach, which prepares test reports in a colloquial language structure. BDD creates test reports using keywords such as "Given", "When", "Then" [9].

4. Steps of Operation of the Proposed Design

The working steps of the proposed design are shown in Figure 8. The study consists of four main sections. These sections are; Web Application UI, Page Factory, Page Layer, Test Layer.

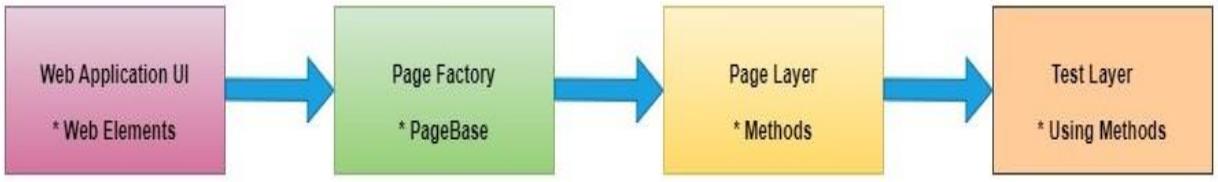

Figure 8: Operation Steps of The Proposed Design

4.1. Web Application UI

Web Application UI section is the area where the application to be tested is located. Each HTML element that we will operate on the relevant page is called an element, these elements are components such as pictures, text, buttons, figures, tables on the page. Firstly, all web elements on the page are determined. They are added as elements to the WebElementi class in Selenium. Figure 9 shows the elements on a sample page.

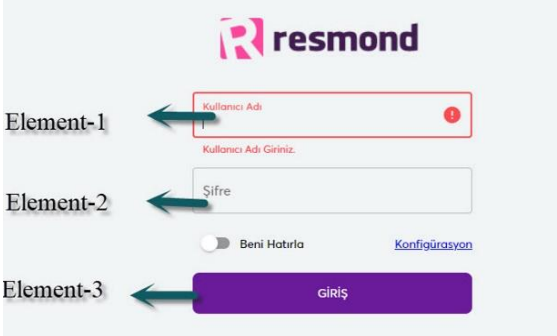

Figure 9: Application UI Example

4.2. Page Factory

It is a Page object model concept that allows to create an object repository in the project, sustainable, reusable and allows to manage elements from a single place. After the elements to be used in the application to be tested are determined, these elements are named and saved with @Findby in Page Factory as the 2nd stage. In this way, the location of the element is determined. Figure 10 shows a sample page factory class.

```
// PAGE FACTORY
    //loginkismi----------------------------------------------------------------------------------------
    // user name
    @FindBy(xpath= "//*[@id=\"q-app\"]/div/div/div/div/div/div[2]/div/form/div/label[1]")
    WebElement user;

    // password
    @FindBy(xpath = "//*[@id=\"q-app\"]/div/div/div/div/div/div[2]/div/form/div/label[2]")
    WebElement pass;

    // giris button
    @FindBy(xpath = "//*[@id=\"q-app\"]/div/div/div/div/div/div[2]/div/form/div/div[3]/button/span[2]/span")
    WebElement grsbtton;
```

Figure 10: Page Factory Example

4.3. Page Layer

Page Layer is the layer where all the pages to be tested in the application are located, the elements named with @FindBy in Page Factory are used in the relevant methods in Page Layer.

4.4. Test Layer

After the elements are used in methods in Page Layer, these methods are called and executed in Test Layer. The method of putting a dot at the end of the method to be executed in Java is also included in the Test Layer here. Thus, other methods can be called without the need for another test page. Figure 11 shows a sample Test Layer code example.

```
@Test(dataProvider = "dataProvider")
public void testSuccessfulLogin(String username, String password) throws IOException {

    ExtentReports extent = this.extent;
    extentTest = extent.startTest("Resmond Restoran Login : "+username);
    extentTest.setDescription("Running Environment : " + testConfig.getProperty("environment"));


testResult= homePage.clickYonetimPaneliLink(extentTest, jobNo, date).clicksaveLink().loginAs(jobNo,date, username, password, extentTest
            .isLoginSuccesful(jobNo,date,username, extentTest);

    Assert.assertTrue(testResult, "Error:Login is not Succesful for user " + username);
```

Figure 11: Test Layer Example

5. Results and Discussions

In this study, a framework is proposed for fast and accurate software testing. Thanks to this framework, it is ensured that the setup of the test environment is created quickly by defining which test will be run in which browser and environment. In addition, changes and updates are easily performed with a single point of control. This ensures ease of adaptation and maintenance.

Another benefit is that in the existing automation tests, DataProvider must be run repeatedly for each test, while with this proposed hybrid framework, it is enough to run DataProvider once for the desired number of tests. With this automated work, recurring code complexity has been prevented.

Another advantage of the framework is that screenshots of each step can be taken, test scenario steps can be added, and the duration of the actual test can be seen in the generated reports. In this way, a detailed test report that can be easily analyzed is generated.

Thanks to the easy management of the proposed framework, new staff to be included in the test teams can be easily adapted to the team. In the future studies, it is aimed to adapt the proposed test automation framework not only for web applications but also for the testing of mobile and desktop software.

References

[1] Bhagyashree Bhondokar, Pooja Ranawade, Snehal Jadhav, Mayuri Vibhute, ve MIT College of Engineering, Pune, "Hybrid Test Automation Framework for Web Application", *Int. J. Eng. Res.*, c. V4, sy 04, s. IJERTV4IS041292, Nis. 2015, doi: 10.17577/IJERTV4IS041292.

[2] Fei Wang ve Wencai Du, "A Test Automation Framework Based on WEB", içinde *2012 IEEE/ACIS 11th International Conference on Computer and Information Science*, Shanghai, May. 2012, ss. 683-687. doi: 10.1109/ICIS.2012.21.

[3] E. H. Kim, J. C. Na, ve S. M. Ryoo, "Implementing an Effective Test Automation Framework", içinde *2009 33rd Annual IEEE International Computer Software and Applications Conference*, Seattle, Washington, USA, 2009, ss. 534-538. doi: 10.1109/COMPSAC.2009.188.

[4] Java website, https://www.java.com/, (Date of access: February 2023)

[5] Selenium website, https://www.selenium.dev/documentation/, (Date of access: February 2023)

[6] Maven website, https://maven.apache.org/, (Date of access: February 2023)

[7] TestNG website, https://testng.org/doc/, (Date of access: February 2023)

[8] Extent Reports website, https://www.extentreports.com/, (Date of access: March 2023)

[9] Cucumber website, https://cucumber.io/, (Date of access: March 2023)