TEST AUTOMATION FRAMEWORK FOR THE PROCUREMENT TO PAY PROCESS IN MSD365

Arjun Mohanan Bindhu [1] and Shahid Ali [2]

[1] Genesis Energy, Automation Test Engineer, Auckland, New Zealand
[2] Department of Information Technology, AGI Institute, Auckland, New Zealand

ABSTRACT

Automation testing is an efficient way to test software functionality. The proposed test automation framework for the procurement to pay process in MSD365 ERP includes purchasing products from vendors, confirming the order, receiving the order and generating an invoice. The designed framework covers test cases, test scripts and reports of the aforementioned functionalities. Automation scripting is done using the Java programming language utilizing Selenium with the TestNG framework. This project will assist companies who are implementing MSD365 in automating procure to pay process. Techniques designed in this project are extensible to any MSD365 implementation with few minor customizations.

KEYWORDS

Scrum, Agile, Waterfall, Test Automation Framework, Selenium.

1. INTRODUCTION

The field of software development is extremely complex because of the scale and complexity of the designed software. This leads us to an even more challenging part, which is how to make sure that these highly complex software systems and the involved system modules are “fit for purpose”? Testing is the most widespread way of ensuring that the designed software implementation is free of faults and bugs [1]. In this project, an automation testing framework was implemented for the process procurement to pay in MSD365 finance and operations.

CONTOSO is an anonymous company in MSD365 ERP provided by Microsoft. This company is designed to purchase goods from various vendors and then distribute those products to its customers. MSD365 is used to maintain the company’s financial and operational data. MSD365 is most suitable for large companies who like to manage their financial and operational data in an efficient manner. One of the scope items for this is to test the functional processes by conducting end-to-end transactions. Hence, automating certain testing steps will make the testing process easier and faster [2].

Developing a testing framework which helps to automatically test the functional process for Contoso company within MSD365. This framework is designed in such a manner that it will be extendible to test similar processes for any implementation project of MSD365. The objective of this project is to develop a tailored test framework for MSD365 which is not readily available in the market and is required by many clients during implementation of MSD365. As a part of this project, testing steps for the procurement-to-pay process have been automated. The deliverable of this project is a framework for automated testing of the purchase order creation process.

This research is organized as follows. Section 2 represents literature review. Section 3 presents the methodology for this research. Section 4 covers the topics related to the execution of this project. Section 5 represents the results. Section 6 test reports of project execution are discussed. Section 7 represents the discussion of the project; section 8 covers the conclusion of the project and In Section 9 future recommendations are covered.

2. Literature Review

In this section we will review the previous literature with respect to automation processes and frameworks.

In software testing, automation is a more efficient and effective way to perform testing in web-based applications. Automating the testing process helps to reduce the time and cost of the testing process and results in producing high quality deliverables [3].

A study was conducted to improve the test automation with an AI and behavioural driven development (BDD) approach [4]. The research deployed a mixed methodology approach including the integration of visual AI and BDD in software test automation to explore the impact on test creation, execution and maintenance. The results of the study recommended the use of visual AI along with behavioural data driven.

There are numerous test automation frameworks which are used in industry, such as Ranorex, Katalon, Cucumber, Selenium and Robot framework. In this project we use the Selenium to make an automation framework because it is more suitable for applications like MSD365 [5].

Further, a test automation framework was proposed, extending the traditional strategies, for example, data-driven and page object models [6]. The study concluded that the representation of test cases in soft computing is considered to be challengeable [6].

From the above research, we conclude that there are various tools and frameworks that can be used for test automation. Each tool and method have its own limitations. To find the test creation and execution for the process, procurement to pay, we undertook this project and to compile our findings. Further, this project has the ability to deploy to any other procurement business to purchase with minor changes.

3. Research Methodology

On top of Selenium TestNG framework is used in this project to write test automation scripts. The TestNG framework supports managing and executing test cases smoothly, whereas without TestNG, the script stops working whenever an exception is thrown. Moreover, TestNG provides a reporting feature which helps to get a good overview of testing results [7]. Additionally, TestNG can also enable/disable test cases and prioritise the test run sequence.

In order to write and execute the test automation scripts in Selenium, the JAVA programming language is used. Java was selected because of its large community support, availability of extensive libraries, and platform independency [8].

Further, the methodology followed in this project is Agile, which is being widely used in the software industry, specifically Small and Medium Enterprises (SMEs) [9]. There are other well-known software development methodologies such as V-Model and Waterfall [11]. The Waterfall methodology, to some extent, is also suitable for the project to some extent where the five-week project requirements were clear. The major point not to follow the Waterfall methodology was to handle the core part of this project, which was technical. In the five-week period, working on such a technical project under industrial supervisors was mostly dynamic. Where day-to-day meetings were setting the directions and supervisors were guiding them to achieve next milestones and cope with current challenges. So, in a given situation, Scrum was more suitable for this short-term dynamic project [9]. Where iterative change in requirement happens.

Further, to manage this whole project, JIRA is used for test management and sprint planning as shown in figure A in section 4.1. JIRA is said to be one of the best and flexible tools available in the market for the above-said purposes [10]. JIRA helped with effective project planning at sprint level, which helps to deliver the project on time and helped to maintain good traceability of test cases and execution.

4. PROJECT EXECUTION

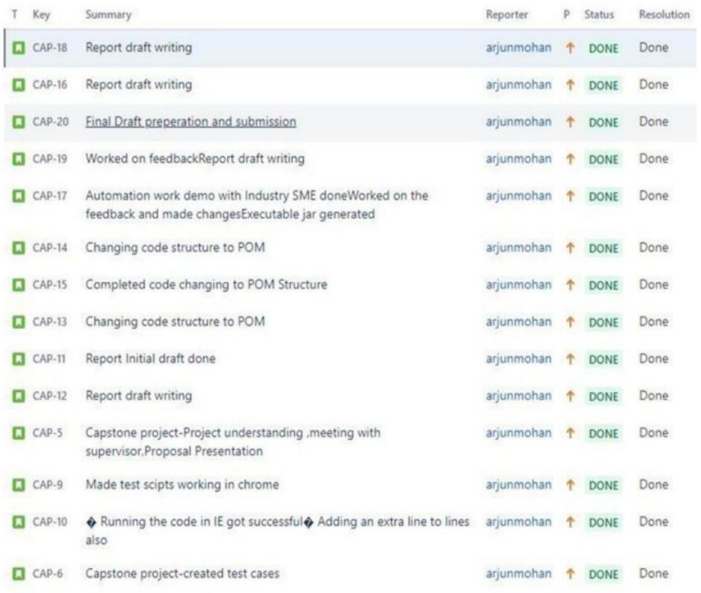
This section represents the core technical part carried out in this project. It shows the planning of a project, design, execution and implementation of automation testing to test procurement-to-pay process. A detailed five-week communication plan was created for the project as depicted in table 1.

Table 1: Five Weeks Communication Plan

Communication Meeting Type	Communication Objective	Medium	Frequency
Meeting with industry SME supervisor	Review status of the project with industry SME	Face to face / Microsoft Teams	Weekly
Ad hoc meetings	Ad hoc meetings to address critical/urgent issues	Face to face / Microsoft Teams	As required
Daily stand-up meetings with industry SME supervisor	Update daily progress	Face to face / Microsoft Teams	Daily
Meeting with project supervisor	Review status of the project with the supervisor	Face to face / Microsoft Teams	Daily

4.1. Scrum Methodology and JIRA

Sprint planning is done in JIRA (screenshot attached in Figure A). In JIRA, three sprints were created. The planning for this project was divided into three sprints. Week one is dedicated to sprints zero and two. Week three is allocated for Sprint one and the rest of the two weeks are dedicated to Sprint three. The methodology used in the project is agile scrum, where testing is done iterative with every sprint/week level planning. Daily project execution status is updated by Industry SME and feedback is taken. Changes of feedback were incorporated in the next sprint.



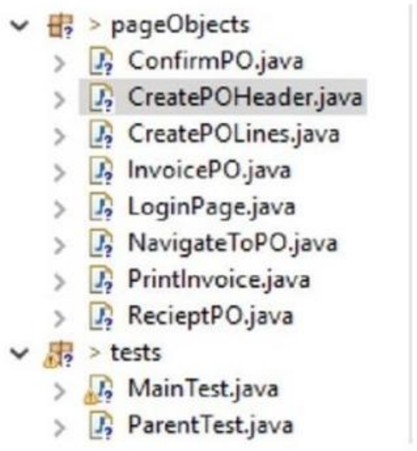
T	Key	Summary	Reporter	P	Status	Resolution
🟩	CAP-18	Report draft writing	arjunmohan	↑	DONE	Done
🟩	CAP-16	Report draft writing	arjunmohan	↑	DONE	Done
🟩	CAP-20	Final Draft preperation and submission	arjunmohan	↑	DONE	Done
🟩	CAP-19	Worked on feedbackReport draft writing	arjunmohan	↑	DONE	Done
🟩	CAP-17	Automation work demo with Industry SME doneWorked on the feedback and made changesExecutable jar generated	arjunmohan	↑	DONE	Done
🟩	CAP-14	Changing code structure to POM	arjunmohan	↑	DONE	Done
🟩	CAP-15	Completed code changing to POM Structure	arjunmohan	↑	DONE	Done
🟩	CAP-13	Changing code structure to POM	arjunmohan	↑	DONE	Done
🟩	CAP-11	Report Initial draft done	arjunmohan	↑	DONE	Done
🟩	CAP-12	Report draft writing	arjunmohan	↑	DONE	Done
🟩	CAP-5	Capstone project-Project understanding ,meeting with supervisor,Proposal Presentation	arjunmohan	↑	DONE	Done
🟩	CAP-9	Made test scipts working in chrome	arjunmohan	↑	DONE	Done
🟩	CAP-10	� Running the code in IE got successful� Adding an extra line to lines also	arjunmohan	↑	DONE	Done
🟩	CAP-6	Capstone project-created test cases	arjunmohan	↑	DONE	Done

Figure A: Jira Dashboard

4.2. Test Execution

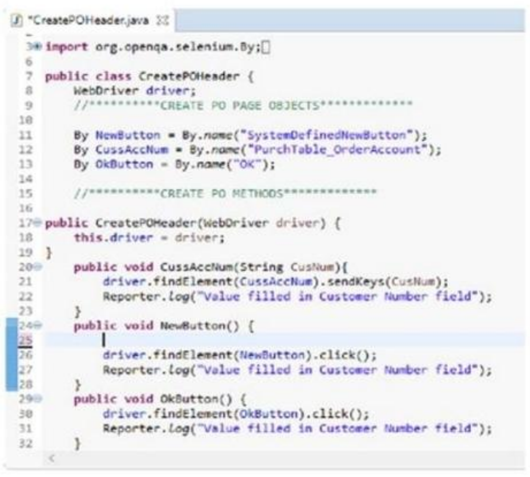
Test case Preparation: Test cases are prepared for the execution of the project. Test cases are prepared based on the end-to-end business process of procurement to pay, starting from logging in to the application to creating a new Purchase Order, confirming the order, receiving the PO and generating Invoice.

Test Scripts: Test scripts are created in a POM structure where two main packages are created. One package for Page Objects and another package for the main testing class. The whole process is divided into eight scenarios, and these scenarios are created as Page Object classes in the POM structure as shown in figure C. Each page Object class contains objects inside the pages and methods describing the actions that need to be performed, as shown in figure D. A main TestNG class is created for test execution where tests are divided and prioritized as shown in figure E.

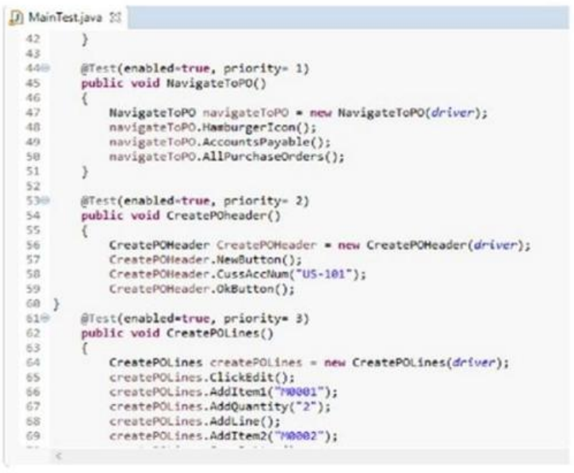


```
v > pageObjects
    > ConfirmPO.java
    > CreatePOHeader.java
    > CreatePOLines.java
    > InvoicePO.java
    > LoginPage.java
    > NavigateToPO.java
    > PrintInvoice.java
    > RecieptPO.java
v > tests
    > MainTest.java
    > ParentTest.java
```

Figure C : POM structure in high level



```java
import org.openqa.selenium.By;

public class CreatePOHeader {
    WebDriver driver;
    //**********CREATE PO PAGE OBJECTS*************

    By NewButton = By.name("SystemDefinedNewButton");
    By CussAccNum = By.name("PurchTable_OrderAccount");
    By OkButton = By.name("OK");

    //**********CREATE PO METHODS*************

public CreatePOHeader(WebDriver driver) {
    this.driver = driver;
}
    public void CussAccNum(String CusNum){
        driver.findElement(CussAccNum).sendKeys(CusNum);
        Reporter.log("Value filled in Customer Number field");
    }
    public void NewButton() {

        driver.findElement(NewButton).click();
        Reporter.log("Value filled in Customer Number field");
    }
    public void OkButton() {
        driver.findElement(OkButton).click();
        Reporter.log("Value filled in Customer Number field");
    }
```



```java
    }

    @Test(enabled=true, priority= 1)
    public void NavigateToPO()
    {
        NavigateToPO navigateToPO = new NavigateToPO(driver);
        navigateToPO.HamburgerIcon();
        navigateToPO.AccountsPayable();
        navigateToPO.AllPurchaseOrders();
    }

    @Test(enabled=true, priority= 2)
    public void CreatePOheader()
    {
        CreatePOHeader CreatePOHeader = new CreatePOHeader(driver);
        CreatePOHeader.NewButton();
        CreatePOHeader.CussAccNum("US-101");
        CreatePOHeader.OkButton();
}
    @Test(enabled=true, priority= 3)
    public void CreatePOLines()
    {
        CreatePOLines createPOLines = new CreatePOLines(driver);
        createPOLines.ClickEdit();
        createPOLines.AddItem1("M0001");
        createPOLines.AddQuantity("2");
        createPOLines.AddLine();
        createPOLines.AddItem2("M0002");
```

Figure D : POM-Page Objects & Methods

Figure E: POM Main TextNG Class

5. RESULTS

The executed results for this project are discussed one by one in this section. Figure F shows the test framework model of the project. The framework consists of the following parts:

1. Test cases
2. Test Scripts
3. Results and Reports

Test Scripts are written in a POM structure where page objects for each page are created separately and a TestNG class where all the main tests are done. Dynamic values are fed to the TestNG class. A public parentTestClassis created to run the test as a Java Application and for generating an executable jar file for the project. This Java program is then fed to web browsers through respective drivers and tests are done. With the help of the TestNG framework, reports were generated which give a broad idea about the results; tests passed, failed and skipped.

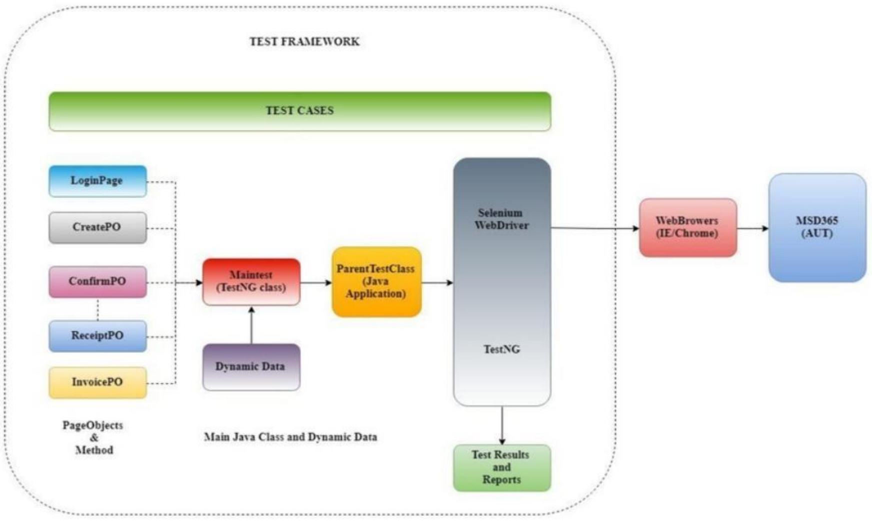


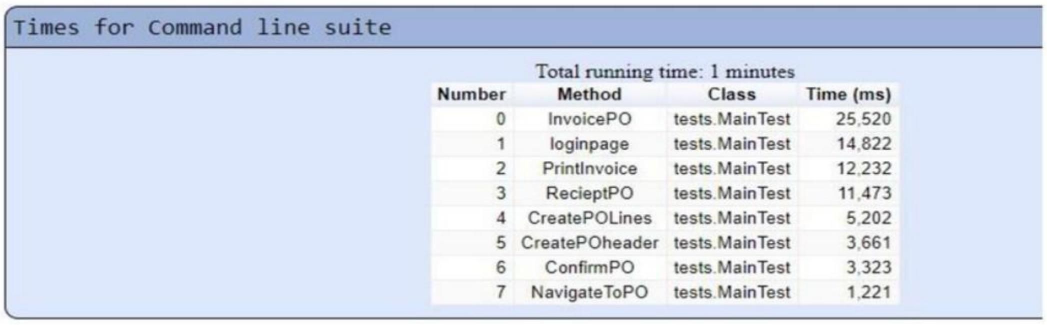
Figure F: Test Automation Framework Architecture

6. TEST REPORTS

In this section, test reports of project execution are discussed.

6.1.Time Taken for Suite Execution

The time taken for the whole automation test suite to execute is around one minute, as shown below in figure G.



Times for Command line suite

Total running time: 1 minutes

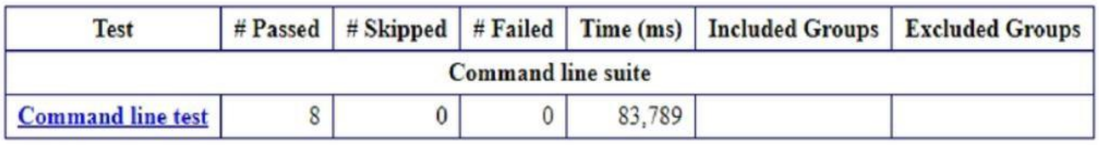
Number	Method	Class	Time (ms)
0	InvoicePO	tests.MainTest	25,520
1	loginpage	tests.MainTest	14,822
2	PrintInvoice	tests.MainTest	12,232
3	RecieptPO	tests.MainTest	11,473
4	CreatePOLines	tests.MainTest	5,202
5	CreatePOheader	tests.MainTest	3,661
6	ConfirmPO	tests.MainTest	3,323
7	NavigateToPO	tests.MainTest	1,221

Figure G: Test Execution Time

If we do this end-to-end process manually, it will take more than 3 minutes to execute all these processes. By this, it is ensured that implementation of the automation suite will improve the efficiency of testing by reducing the time.

6.2.Test Execution Status

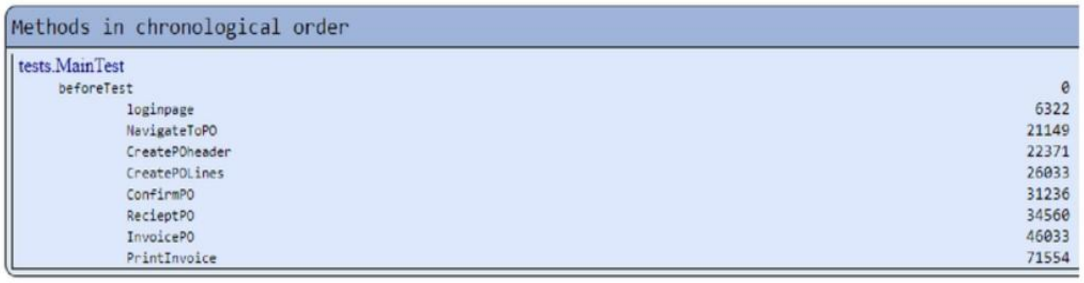
Figure H shows the test pass/fail status of the test. Here out of the eight test cases, all test cases passed and there are no skipped or failed test cases.



Test	# Passed	# Skipped	# Failed	Time (ms)	Included Groups	Excluded Groups
Command line suite						
Command line test	8	0	0	83,789		

Figure H: Test Execution Status

This is the chronological order of methods executed. This is done by the implementation of TestNG by giving priority to the test cases. The test execution order is shown below in figure I.

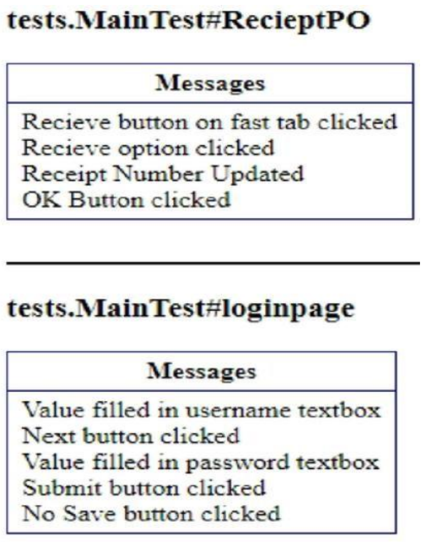


```
Methods in chronological order
tests.MainTest
    beforeTest                          0
        loginpage                    6322
        NavigateToPO                21149
        CreatePOheader              22371
        CreatePOLines               26033
        ConfirmPO                   31236
        RecieptPO                   34560
        InvoicePO                   46033
        PrintInvoice                71554
```

Figure I: Test Execution Order

6.3. Reporter Log Functionality

A functionality for getting report log of each line execution is written along the methods. This helped to get the report log as shown in figure J.



tests.MainTest#RecieptPO

Messages
Recieve button on fast tab clicked Recieve option clicked Receipt Number Updated OK Button clicked

tests.MainTest#loginpage

Messages
Value filled in username textbox Next button clicked Value filled in password textbox Submit button clicked No Save button clicked

Figure J: Reporter.log functionality Output

7. DISCUSSION

The most important aims of implementing automation testing in MSD365 were to:

1. Minimize the testing effort by making a reusable Test automation framework
2. Increasing the efficiency of the testing process 3. Reduce the test execution time

Implementation of TestNG helps the framework to generate HTML reports which give user a wider idea about the execution status (as shown in Figure H), execution time (as shown in Figure G), detailed steps in execution (as shown in Figure J) etc. TestNG also helped to prioritize and group test cases based on the business flow (as shown in Figure I). Java Language for coding: Programming language which was selected as JAVA was the most suitable language. In places where problems related to the element locator, wait time, and data provider happened, the large community support of JAVA helped to solve the issue within a short span of time.

The introduction of scrum technology in the project helped to organize and plan the whole project to different sprints and preparing the daily work schedule, communication plan etc. helped to perfectly deliver the project on time. A detailed report of work planned, and work done can be seen on table 2.

The framework shown in figure F in the CONTOSO demo environment is reusable and extensible to any MSD365 ERP implementation. With a few customizations, the framework will be reusable. Test Execution Time: Time taken for execution of test cases manually will take more than 3–4 minutes to complete, but on implementation of the Automation test framework, the execution time taken is reduced to around one minute as shown in figure G. Thereby the implementation of new framework reduced the execution time to one-fourth of manual testing effort and thereby the cost of testing as shown in figure K.

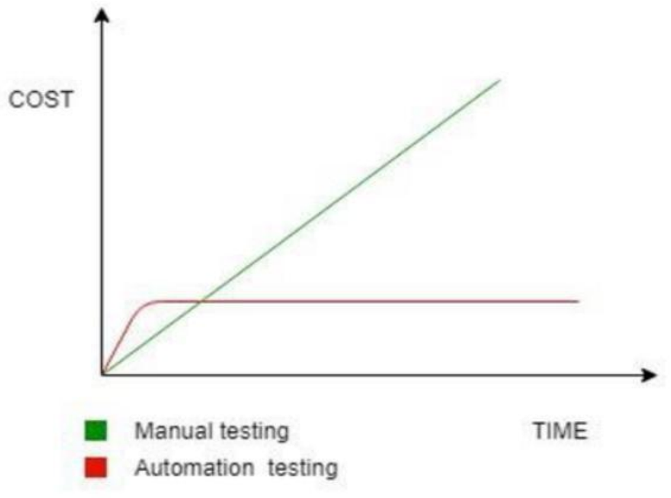


Figure :Time vs Cost Testing

8. CONCLUSION

The proposed Automation Test Framework for automating the procurement to pay process in MSD365 ERP is extensible to any company that implements MSD365 ERP. The framework includes test cases, test scripts and test results, executable jars and reports obtained after execution of the automation script. Here a TestNG-based POM framework is developed where dynamic values can be entered into the code very easily, which will help the testing process in a very effective manner. And an executable jar file (which needs Java installed on a PC) is created where any user can run the test with a click, without an IDE. Results show that this framework will help in reducing the time, increase the efficiency and reduce the cost for testing. Further, with few customizations, the framework is reusable to extend to other modules in MSD365. TestNG-based.

9. RECOMMENDATIONS

Many different customizations, improvements and add-ons have been left for implementation for future work due to lack of time frame. There are some recommendations which are planned to be included in future works are given below:

- Parametrization of dynamic values with xml/xlsx
- Changing code format to Gherkin Syntax in Cucumber for better transparency
- Finding better "static element locators' in MSD365
- Increase the scope of Automation by extending it to other modules in D365

REFERENCES

[1] Sarkar, A., Islam, S. M., & Bari, M. S. (2024). Transforming User Stories into Java Scripts: Advancing Qa Automation in The Us Market with Natural Language Processing.Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023,7(01), 9-37.

[2] Mosleh, M. A., Al-Khulaidi, N. A., Gumaei, A. H., Alsabry, A., &Musleh, A. A. (2024, August). Classification and evaluation framework of automated testing tools for agile software: Technical review. In2024 4th International Conference on Emerging Smart Technologies and Applications (eSmarTA)(pp. 1-12). IEEE.

[3] Jain, C. R., &Kaluri, R. (2015). Design of automation scripts execution application for selenium webdriver and test NG framework. ARPN J EngApplSci, 10, 2440-2445.

[4] Ragel, R. K. C., &Balahadia, F. F. (2023, November). Visual Test Framework: Enhancing Software Test Automation with Visual Artificial Intelligence and Behavioral Driven Development. In2023 IEEE 15th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)(pp. 1-5). IEEE.

[5] PaiviRaulamo,J., Mika,M.,&Vahid.,G (2017). Choosing the Right Test Automation Tool: a Grey Literature Review of Practitioner Sources. In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (EASE'17). ACM, New York, NY, USA.

[6] Swathi, B., & Tiwari, H. (2021, February). Test automation framework using soft computing techniques. In2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)(pp. 1-4). IEEE.

[7] Raghavendra, S. (2024). TestNG. InJava Testing with Selenium: A Comprehensive Syntax Guide for Automation(pp. 285-315). Berkeley, CA: Apress.

[8] Raghavendra, S. (2024). Introduction to Selenium: Java Automation. InJava Testing with Selenium: A Comprehensive Syntax Guide for Automation(pp. 1-18). Berkeley, CA: Apress.

[9] Karthik,D., (2009). Agile development: overcoming a short-term focus in implementing best practices. In Proceedings of the 24th ACM SIGPLAN conference companion on Objectoriented programming systems languages and applications (OOPSLA '09). ACM, New York, NY, USA.

[10] Fisher, J., Koning, D., & Ludwigsen, A. P. (2013, October). Utilizing Atlassian Jira for large-scale software development management. In / 4th International Conference on Accelerator & Large Experimental Physics Control Systems (ICALEPCS).

[11] Natarajan, T., &Pichai, S. (2024). Transition from waterfall to agile methodology: An action research study.IEEE access,12, 49341-49362.