



**AUTOMATION SOFTWARE TESTING  
USING DATADRIVEN AND POM  
FRAMEWORK**



**A PROJECT REPORT**

*Submitted by*

<b>CHINNADURAI.A</b>	<b>(620818104015)</b>
<b>GOPINATH.I</b>	<b>(620818104027)</b>
<b>GOPINATH.N</b>	<b>(620818104028)</b>
<b>MANOJKUMAR.M</b>	<b>(620818104054)</b>

*In partial fulfillment for the award of the degree*

*of*

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING  
GNANAMANI COLLEGE OF TECHNOLOGY**

**NAMAKKAL- 637018**

**ANNA UNIVERSITY: CHENNAI600025**

**JUNE2022**

**BONAFIDE CERTIFICATE**

Certified that this project report titled “**AUTOMATION SOFTWARE TESTING USING DATADRIVEN AND POM FRAMEWORK**” is the bonafide work of **CHINNADURAI.A (620818104015), GOPINATH.I (620818104027), GOPINATH.N (620818104028), MANOJKUMAR.M (620818104054)** who carried out the project work under my supervision. Certified further, that to the best of my knowledge and belief, the work reported here in does not form part of any other earlier occasion on this or any other candidate.

**SIGNATURE**

**Dr.R.UMAMAHESWARI,M.E.,Ph.D.,**

**HEAD OF THE DEPARTMENT,**

Department of CSE,

Gnanamani College of Technology,

Namakkal-637018.

**SIGNATURE**

**Mr.K.DINESH KUMAR.M.E.,**

**SUPERVISOR,**

Department of CSE,

Gnanamani College of Technology,

Namakkal-637 018.

Submitted for the University Project Viva Voce Examination held on\_\_\_\_\_

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

I feel highly honoured to extend our sincere gratitude to our beloved Chairman, **Dr.T.ARANGANNAL** and our Chairperson Smt.**P.MALALEENA**, and Vice Chairperson **Ms.MADHUVANTHINIE ARANGANNAL**, Gnyanamani Educational Institutions, **NAMAKKAL** for providing all facilities to complete this project work.

I would like to express my sincere thanks to Chief Administrative Officer **Dr.P.PREMKUMAR**, for their support to bring the best in me.

I would like to acknowledge the constant and kind support provided by our Principal **Dr.T.K.KANNAN**, for their support to bring the best in me.

We extend our thanks to our Academic Director **Dr.B.SANJAY GANDHI** , Gnanamani College of Technology, Namakkal, for motivating our project work successfully.

I feel Highly elated to thank respectable Head of the Department **Dr.R.UMAMAHESWARI**, who guided and was a pillar of support for the successful completion of the project.

I Wish to thanks to my Project Coordinator **Dr.R.C.KARPAGALAKSHMI**, for providing good suggestions and help me to make this project a success.

We are greatly indebted to our Project Incharge **Mr.K.DINESHKUMAR.,ME.** of Computer Science and Engineering department for him valuable suggestions and guidance to our project.

This project has brought out the hidden talents within me. It is a pleasure to express our gratefulness to our beloved parents for providing their support for the completion of the project and our heartfelt thanks to our entire department faculty and technical members, beloved friends, directly and indirectly who helped us during the tenure of the project.

**[CHINNADURAI.A] [GOPINATH.I] [GOPINATH.N] [MANOJKUMAR.M]**



## INSTITUTE

### **VISION**

Emerging as a technical institution of high standard and excellence to produce quality Engineers, Researchers, Administrators and Entrepreneurs with ethical and moral values to contribute the sustainable development of the society.

### **MISSION**

We facilitate our students

To have in-depth domain knowledge with analytical and practical skills in cutting edge technologies by imparting quality technical education. To be industry ready and multi-skilled personalities to transfer technology to industries and rural areas by creating interests among students in Research and Development and Entrepreneurship.

## DEPARTMENT

### **VISION**

To evolve as a Centre of Excellence to produce the most competent software professionals, researchers, entrepreneurs and academicians with ethical values in Computer Science and Engineering.

### **MISSION**

- Imparting quality education through latest technologies to prepare Students as software developer and system analyst.
- Inculcating the technological transformations for the sustainable development of society.
- Promoting excellence towards higher education, research, employability and entrepreneurship.

## **ABSTRACT**

Software Testing is a process, which involves, executing of a software program application and finding all errors or bugs in that program/application so that the result will be a defect-free software. Quality of any software can only be known through means of testing (software testing). Through the advancement of technology around the world, there increased the number of verification techniques and methods to test the software before it goes to production and off course to market. Automation Testing has made its impact in the testing process. Now-a-days, most of the software testing is done with the automation tools which not only lessens the number of people working around that software but also the errors that can be escaped through the eyes of the tester. Automation testing contains test cases which makes the work easy to capture different scenarios and store them. Therefore, software automation testing process plays a vital role in the software testing success. This study aims in knowing different types of software testing, software testing techniques and tools and to compare manual testing versus automation testing.

One of the important parts of software development life-cycle is software testing and one of which is regression testing. Regression testing is done to verify that previously detected errors have been corrected correctly and no new errors arise as a result of incorrect corrections so that the quality of the software being tested is maintained. It is important and must be done. However, the conventional method is not efficient as it is time-consuming, not reusable, and prone to error. In this paper, a comprehensive review of the automated testing approach is presented to be used by other researchers in this field of study. The review result shows that the automated testing approach is suitable to enhance the regression testing with some plausible options of tools e.g. Selenium, SAHI, and Robot-framework. Furthermore, some plausible options of execution methods are also discussed in this paper. This review further concludes that the parallel execution method is considered as a promising choice to conduct the most efficient regression testing process.

## TABLE OF CONTENTS

S.NO	TITLE	PAGE NO
	<b>ABSTRACT</b>	<b>iv</b>
	<b>LIST OF FIGURES</b>	<b>vi</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>5</b>
<b>3</b>	<b>PROJECT DESCRIPTION</b>	<b>7</b>
	3.1 Existing System	<b>7</b>
	3.2 Proposed System	<b>8</b>
<b>4</b>	<b>SYSTEM SPECIFICATION</b>	<b>9</b>
	4.1 Software Requirements	9
	4.2 Hardware Requirements	9
<b>5</b>	<b>MODULES</b>	<b>10</b>
<b>6</b>	<b>SYSTEM DESIGN</b>	<b>13</b>
	6.1 System Architecture	13
	6.2 Data Flow Diagram	15
	6.3 Use Case Diagram	16
	6.4 Framework Architecture	18
	6.5 Test Automation Lifecycle	19
<b>7</b>	<b>COMPONENTS</b>	<b>20</b>
<b>8</b>	<b>FRAMEWORKS</b>	<b>22</b>
	8.1 Data Driven Framework	22
	8.2 POM	23
	8.3 TestNG	24
	8.3.1 TestNG Reports	25
	8.3.2 Extent Reports	27
<b>9</b>	<b>MANUAL TESTING</b>	<b>28</b>
<b>10</b>	<b>AUTOMATION TESTING</b>	<b>32</b>
<b>11</b>	<b>CONCLUSION</b>	<b>36</b>
	<b>SOURCE CODE</b>	<b>37</b>
	<b>SCREEN SHOT</b>	<b>48</b>
	<b>REFERENCES</b>	<b>62</b>

## LIST OF FIGURES

<b>FIG NO</b>	<b>TITLE</b>	<b>PAGE NO</b>
Fig.6.1.1	System Architecture	12
Fig.6.2.1	Data Flow Diagram	14
Fig.6.3.1	Use Case Diagram	15
Fig.6.4.1	Framework Architecture	17
Fig.6.5.1	Test Automation Life Cycle	18
Fig.8.1.1	Data driven Framework	21
Fig.8.1.2	Register Data Sheet	22
Fig.8.2.1	Page Object Model	23
Fig.8.3.1	TestNG Diagram	24
Fig.8.3.1.1	TestNG Reporting Page	25
Fig.8.3.2.1	TestNG Extents Reports Page	26
Fig.9.1.1	Manual Testing Types Diagram	27
Fig.10.1.1	Selenium WebDriver architecture	34

# **CHAPTER 1**

## **INTRODUCTION**

Software Testing is the process to bring on the latent defects into identifiable ones. This crucial phase of the software development life cycle uncovers the hidden defects in a software product. Regardless of time-consuming and resource-hungry nature of testing, we can never ignore it. Every newly developed or modified engineering product is required to pass rigorous tests so as to ensure the quality of the developed product 5 . Software Testing utilizes approximately 40% - 50% of total resources, 30% of total effort and 50%-60% of the total cost of software development<sup>1-4</sup>. Testing phase, being a major challenge in software development, can be considered as a fair opportunity that can considerably help to improve and optimize software's cost, quality and time to market. This improvement is much desired in the present scenario when software industries are facing tough international competition and trying to shrivel their budgets and schedules 4 . In the interest of this research paper we have classified as

Software testing into two basic categories:

- a) Manual Testing
- b) Automated Testing.

Since long and now also, we are conducting manual testing of software products; in this type of testing a human tester executes the application and initiates various tests over it by interpreting and analyzing the behavior of the product on various input conditions<sup>2,6</sup>. The human tester later prepares the reports and provides comments on the quality-state of the product by comparing the actual results against the expected results. On the other hand an Automated Testing (AT) refers to the use of some standard software solutions to control the execution of test-cases on the Software Under Test (SUT)<sup>7,8</sup>. This process also involves setting up the preconditions, matching the actual results against the predicted ones and then documenting the observations according to some standard protocol<sup>9,10</sup>. Automated testing requires writing up some special computer programs to find bugs or defects in SUT. It is an excellent approach to replace the laborious and time consuming manual testing. Automated testing has various advantages and it is always



suggested for the quality improvements of the application as it provides formal test coverage, avoid human errors and speed up the test execution process<sup>11</sup>. Also, as it speeds up the execution process, it is most effective solution for meeting the strict deadlines. As a result, today, there are many commercial software tools, that allows fully automation, are available for testing purposes and lot of organizations are engaged in providing the quality assurance/testing services. But, test automation is a very critical process; a lot of factors, like which feature requires automated testing, are needed to be determined before any organization proceeds for the automation testing. Also, test automation requires high primary investments in terms of software feature analysis, scripting, tool procurement and training etc. Thus a precise analysis of the Return on Investment (ROI) from test automation is required before we start<sup>5,12,13</sup>. In this paper we have tried to identify and quantify the ways in which the test automation affects the three critical software dimensions of time, cost and quality. The flow of our paper goes in the following manner: First we have discussed the empirical annotations from the past studies over test automation. Then we have tried to model the effects of test automation on cost, quality and time to market of the software product in the subsequent sections. Afterwards the effects of test automation are calculated on three different software's, through this model.

The concept of software testing has evolved since 1970's as an important process in Software development, because through it quality of the software can be improved by checking the errors and faults present in the software. According to Burnstein [1], —Software Testing is generally described as a group of procedures carried out to evaluate some aspect of a piece of software<sup>14</sup> or —Software Testing can be described as a process used for revealing defects in software, and for establishing that the software has attained a specified degree of quality with respect to selected attributes<sup>15</sup>. The development of software is a complex activity and with the software systems growing in size the development of software has become more complex. At the same time it becomes difficult to maintain quality as the systems grow in size and complexity. As the quality becomes important concern software testing is one area to focus on in order to improve quality [2]. Software testing can be done manually and automatically. In manual testing, the testing requires human input, analysis and evaluation. Manual testing deals with human intervention, So naturally, it is prone to errors because often humans get tired of doing the process repeatedly [3]. According to Dustin et.al [4], —Automated Software Testing (AST) is a process in which the testing activities are automated which include development of test cases, execution and

verification of the test scripts and use of automated tools. In Automated Software Testing (AST) all the tests are not automated, so it is important to determine what test cases should be automated first. The advantage of Automated Software Testing (AST) is dependent on how many times a given test can be repeated [2]. If the automation is done quickly at initial stages it can lead to poorly automated tests which are difficult to maintain and vulnerable to software changes. Most of the companies are opting for Automated Software Testing (AST) to achieve benefits such as quality improvement, time to market and less human effort, but various studies show that achieving this is not easy [2]. Achieving efficient Automated Software Testing (AST) is dependent on how to perform tests within a shorter time and with less effort. Organizations are experiencing problems with maintainability and time consuming development of automated testing tools [2]. There are plenty of tools available in the industry to solve this problem. Different tools were developed depending on programming language and testing methods, but selecting a tool is required to support most of the organization's testing requirements. Literature has paid lot of attention to the field of Automated Software Testing (AST) by reporting different automation techniques, methods, approaches and tools. As there are many article published in this area, there is need to structure and evaluate the area of AST. In order to achieve this, there is need to systematically classify the different aspects of AST. Torkar [3] has conducted literature study and established few definitions related to testing and AST, based on those definitions he categorized software testing and AST. Regarding the categorization of Automated Software Testing (AST) tools there is a paper published by Sergey Uspenskiy [28]. He developed an automated test model to classify the tools so that the software tester can obtain a tool or a list of tools that is most suitable for concrete tasks. Even though these articles are very valuable to area of AST these articles do not cover all the aspects of AST such as testing levels, interfaces used (languages and scripting languages) and there is a need to fill the gap. To achieve this goal, we employed Systematic mapping research methodology to systematically categorize different aspects of AST. To get results for the above mentioned research gap; Systematic mapping is best research method as it allows to classify a large set of papers in an efficient way. 8 Automated Software Testing (AST) is not a silver bullet; it also has its limitations. It cannot be applied to every test. Berner [9] found that the majority of the new defects are detected by manual tasks not the automated ones because 60% of the bugs are found during an automated testing effort and 80% are found during the development of the tests. There are some empirical studies by

practitioners [2, 8, 9, 11, 12] on the AST benefits and challenges such as Consistency and repeatability of tests, reuse of tests, earlier time to market, expectation that automated tests will find a lot of new defects, false sense of security and maintenance. There is still need to explore the benefits and challenges of AST. In order to achieve this, we performed systematic literature review to find the empirical evidence regarding the benefits and challenges of AST. Finally, survey is conducted to find whether the benefits and challenges of AST reported in the literature are prevalent in software industry. The aim of this research paper is to systematically classify the current research literature in the area of the Automated Software Testing (AST) and to find empirical evidence regarding the AST benefits and challenges.

The aims will be satisfied by following objectives

- To classify different contributions within Automated Software Testing (AST).
- To identify the benefits of Automated Software Testing (AST) reported in the literature
- To identify the challenges of Automated Software Testing (AST) reported in the literature.
- To identify the whether the reported benefits and challenges found in the literature are prevalent in industry.

## CHAPTER 2

### LITERATURE SURVEY

Industrial softwares are released out in versions. For any version there exists many in home local builds of that version at developer's site. Thousands of software developers work together to brought up any single software. In this industrial scenario, developers usually work on the local version of the software on their machine, they modify it repeatedly, and merge their changes with the latest build ready for release. When thousands of developers simultaneously develop nightly (or weekly or monthly builds) it is nearly impossible to test every developer's local modified version build before it is merged to the main build. As the changes are generally iterative test automation is a right answer. The software grows more with more functionality added in each release. Then, why and how much to test the pre-existing functionalities is the difficult to decide.

A right degree of test automation of the pre-existing functionalities is always necessary, cost effective and time saving activity. Ramler et al in<sup>4</sup> discussed the benefits arising from the automated testing. They discussed test automation as one solution to reduce recurring testing costs and proposed a cost based model, to decide on automation strategy. In<sup>14</sup> Berner et al. claimed that in an experiment automated testing relieves expert testers from executing the same monotonous regression test suite again and again. Hence more resources (expert testers) are available for other hard testing activities which were not possible with manual testing. They also stated that timely maintenance of automated test suits is also required and this maintenance is a major drawback of automated testing.

Hoffman in<sup>15</sup> well defined a trade-off between cost and benefit of test automation by considering various ROI (return on investment) factors. From the calculations it is apparently reasonable to conclude that return from the automation are usually seen in the next release that uses it i.e. running and re-running the automated test cases yields considerable savings. Kasurinen et al. in<sup>16</sup> analyzed the industrial applications of the automated testing. They concluded with the findings that nearly 26% of the test cases are automated in industry, adoption

of test automation is a demanding effort in software industry and test automation is most commonly used for quality control and quality assurance.

Alan et al. in<sup>17</sup> discussed the benefits, challenges and applications of test automation in context of various particular domains like web applications, sensor networks and mobile phone applications etc. Bret in<sup>18</sup> stressed on adopting a development process similar to the development of standard software. They identified different criteria for selecting test cases for automation.

Li et al. in<sup>19</sup> stated that in spiral development, testing cycles are responsible for creating a demand of tools that automate testing. They focused on the design of the tools so as to produce high-quality software in shortened time. Being a step ahead, Damm et al. in<sup>20</sup> describes how test-driven development is possible in industry with test automation. They presented an approach for early defect detection in a cost effective way with the use of C++ to write both software and test tool. Amannejad et al. in<sup>21</sup> automate the integral testing process, ranging from test-case design, to test scripting, to test execution and finally test-result evaluation, in an oil and gas industry and found out that effective decisions on test automation may result in more than 100% ROI in ten rounds of test case rerunning.

A lot more work has been done in this area which emphasizes on the importance and benefits of test automation. However no work, to the best of our knowledge, has been done to formulate and calculate the direct effect of test automation on cost, quality and time

## CHAPTER 3

### PROJECT DESCRIPTION

#### 3.1 EXISTING SYSTEM

Manual testing, as the term suggests, refers to a test process in which a QA manually tests the software application in order to identify bugs. To do so, QAs follow a written test plan that describes a set of unique test scenarios. The QA is required to analyze the performance of the web or mobile application from an end user's perspective. QAs verify the actual behavior of software against expected behavior, and any difference is reported as a bug. Any new application must be manually tested before its testing can be automated. Manual Software Testing requires more effort but is necessary to check automation feasibility. Manual Testing concepts does not require knowledge of any testing tool. One of the Software Testing Fundamental is **“100% Automation is not possible”**. This makes Manual Testing imperative.

#### DRAWBACK OF EXISTING SYSTEM

- Manual testing requires human intervention for test execution.
- Manual testing will require skilled labour, long time & will imply high costs.
- Any type of application can be tested manually, certain testing types like ad-hoc and monkey testing are more suited for manual execution
- Manual testing can become repetitive and boring.

## **3.2 PROPOSED SYSTEM**

A Test Automation Framework is a set of guidelines like “coding standards, test-data handling, object repository treatment etc..”. which when followed during automation scripting produces beneficial outcomes like “increased code re-usage, higher portability, reduced script maintenance cost etc”. These are just guidelines and not rules; they are not mandatory and you can still script without following the guidelines. But you will miss out on the advantages of having a Framework. Automated testing refers to any approach that makes it possible to run your tests without human intervention. Traditional testing has been done manually. A human follows a set of steps to check whether things are behaving as expected. By contrast, an automated test is created once and then can run any time you need it. For a long time, developers have automated their unit testing. That is, the tests that check whether a given function is working properly. Then automated testing frameworks like Selenium were developed. These allow modules or entire applications to be tested automatically. These frameworks allow a test script to interact with your UI, replicating the actions of a user. For instance, they allow you to find a specific button and click it. Or locate a text entry box and fill it out correctly. They also allow you to verify that the test has completed correctly.

## **ADVANTAGES OF PROPOSED SYSTEM**

- Automation testing is use of tools to execute test cases.
- Automation Testing saves time, cost and manpower. Once recorded, it's easier to run an automated test suite.
- Automated testing is recommended only for stable systems and is mostly used for Regression testing.

## **CHAPTER 4**

### **SYSTEM SPECIFICATION**

#### **4.1. Software Requirements:**

- Operating System : Windows 10 Pro
- IDE : eclipse
- Coding Language : JAVA
- Testing Tool : Selenium
- Framework : POM, Data Driven, TestNG
- Testing Tool : Selenium

#### **4.2. Hardware Requirements:**

- Processor : Dual core processor 2.6.0 GHZ
- Hard Disk : 500 GB.
- Monitor : 14' Color Monitor
- Mouse : Optimal mouse
- Key Board : Standard Keyboard
- Ram : 4 GB



## **CHAPTER 5**

### **IMPLEMENTATION**

#### **5.1.LIST OF MODULES**

- Create account test
- Login test
- Homepage verification test
- Search box test
- Proceed to checkout test
- Payment page test
- Order confirm page test

#### **5.2.MODULES DESCRIPTION**

##### **Create Account Test**

Before We Start

You should gather all the customer requirements you can. The list of requirements which can be useful are:

- Title
- FirstName
- LastName
- Email ID
- Password
- DOB
- Address
- PhoneNumber
- Gender

## **Required fields are:/**

- First Name
- Last Name
- Email ID
- Password
- Address
- Phone Number

## **Validation for fields:**

Email: @

Password: It should have alphanumeric, Length should be 8 to 32.

PhoneNumber: Phone Number should have only Numbers, Country code is required

After successful Verification email should send to the user On Required fields \* should show.  
Now, let's begin with our test cases. Although most testers use Bugzilla or other test management tools to maintain test cases, you can also use Excel or spreadsheets.

Signup and login page by assuming some client requirements, such as:

- Username and password are mandatory fields
- There is cancel and reset button at the bottom of the form
- Radio buttons and checkboxes are placed correctly
- The limit of the Password should be 8-13 characters (alphanumeric).

## Login Test

Whenever you will be asked to write the test cases for the 'Form with some controls', you need to follow the list of rules for writing test cases as mentioned below:

Write a test case on each form object.

Written test cases should be a combination of both negative and positive test cases.

Also, test cases should always be a combination of functional, performance, UI, usability, and compatibility test cases.

Because you don't have a login page in front of you and neither you have requirements document for this login page. But the login page is such a common thing of which we can easily imagine the controls.

There can be a username, password, 'Sign In' button, Cancel Button, and Forgot Password link.

There can be one more control which is a checkbox named 'Remember me' to remember the login details on a particular machine.

## CHAPTER 6

### SYSTEM DESIGN

#### 6.1.SYSTEM ARCHITECTURE

Software testing generally involves testing the software functionality so as to identify whether the developed software fulfills the requirements of the users and developers and to identify whether there is any defect or risk. By software testing, the defects can be removed then the software will be defect-free and a high-quality product can be produced or developed.

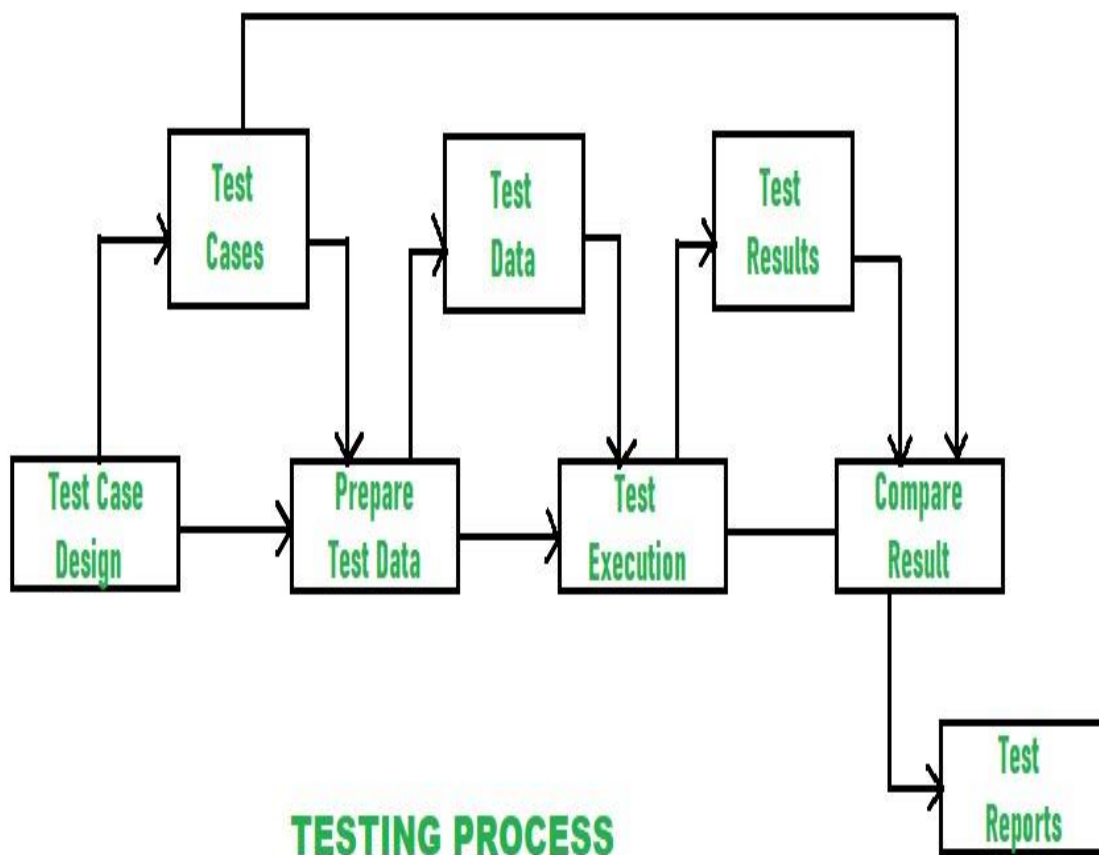


Fig.6.1.1.System Architecture

**Test Planning**

The test plan or test script is prepared. For analyzing the requirements of document (for black box) and program code (for white box), the test plans are generated.

**Test Case Design**

Creating a set of tests that are effective in testing the software is the ultimate target of test case design.

**Test Execution**

In order to obtain the test result, the test data is derived through various test cases.

**Data Collection and Comparison**

Collection and verification of test results is done whether it is complete and correct or not.

**Effective Evaluation and Report Generation**

All the above test activities are performed on the software model and the maximum number of errors is identified. Test reports are generated with the help of which we can work on the errors and find an effective way to reduce the errors in order to obtain a high-quality product

## 6.2.DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

It shows how data enters and leaves the system, what changes the information, and where data is stored.

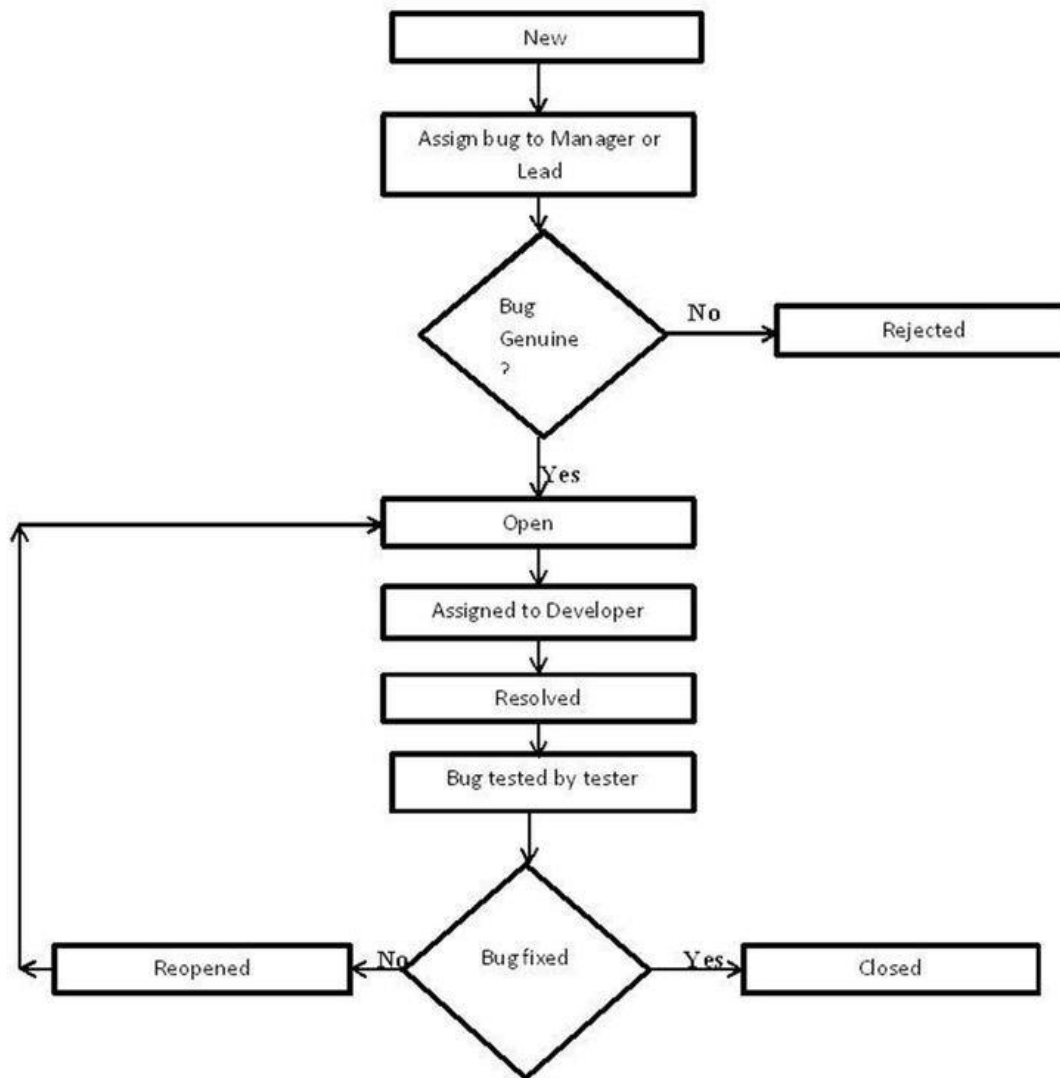


Fig.6.2.1.Data Flow Diagram

## 6.3.USE CASE DIAGRAM

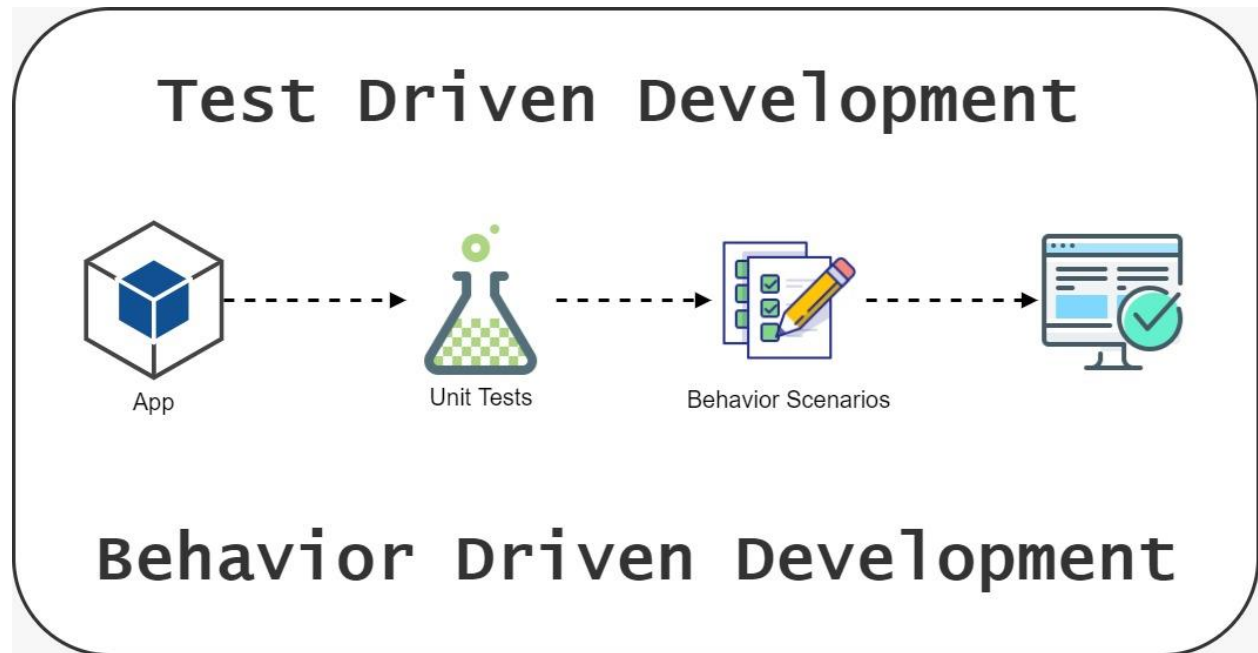


Fig.6.3.1.Use Case Diagram

### Test Driven Development

This is one of the most common test approaches. We implement our code based on unit tests. Each feature of your application should have a unit test designed before its implementation. The first time you run this test will fail. You finish your development when your test pass. We should follow the RGR development cycle.

### Advantages of using TDD

- Faster feedback regarding the new functionality
- Refactoring the code keep us safe on the way how those changes affect the code
- Reduce the number of bugs and increases productivity because we waste less time debugging
- More extensible and less coupled code since we need to separate things to be testable

## **Behavior Driven Development**

This test framework is focused on the behaviors of our application. We write scenarios to check the output with certain given states. We will get automated end to end testing. Another difference comparing with the TDD is the fact that these tests can be read by any person from the testing team and easily understand its purpose. These tests will increase the value of your project since your code and business logic would be tested in two different ways.

### **Advantages of using BDD**

- Increased cost savings since we can avoid some bugs earlier
- Better communication between testing and development teams
- Look at the whole puzzle working together and not just one piece



## 6.4.Framework Architecture

An automation framework is a platform developed by integrating various hardware and software resources and various tools and services based on a qualified set of set of assumptions. It enables efficient design and development of automated test scripts and reliable analysis of issues for the system under test.

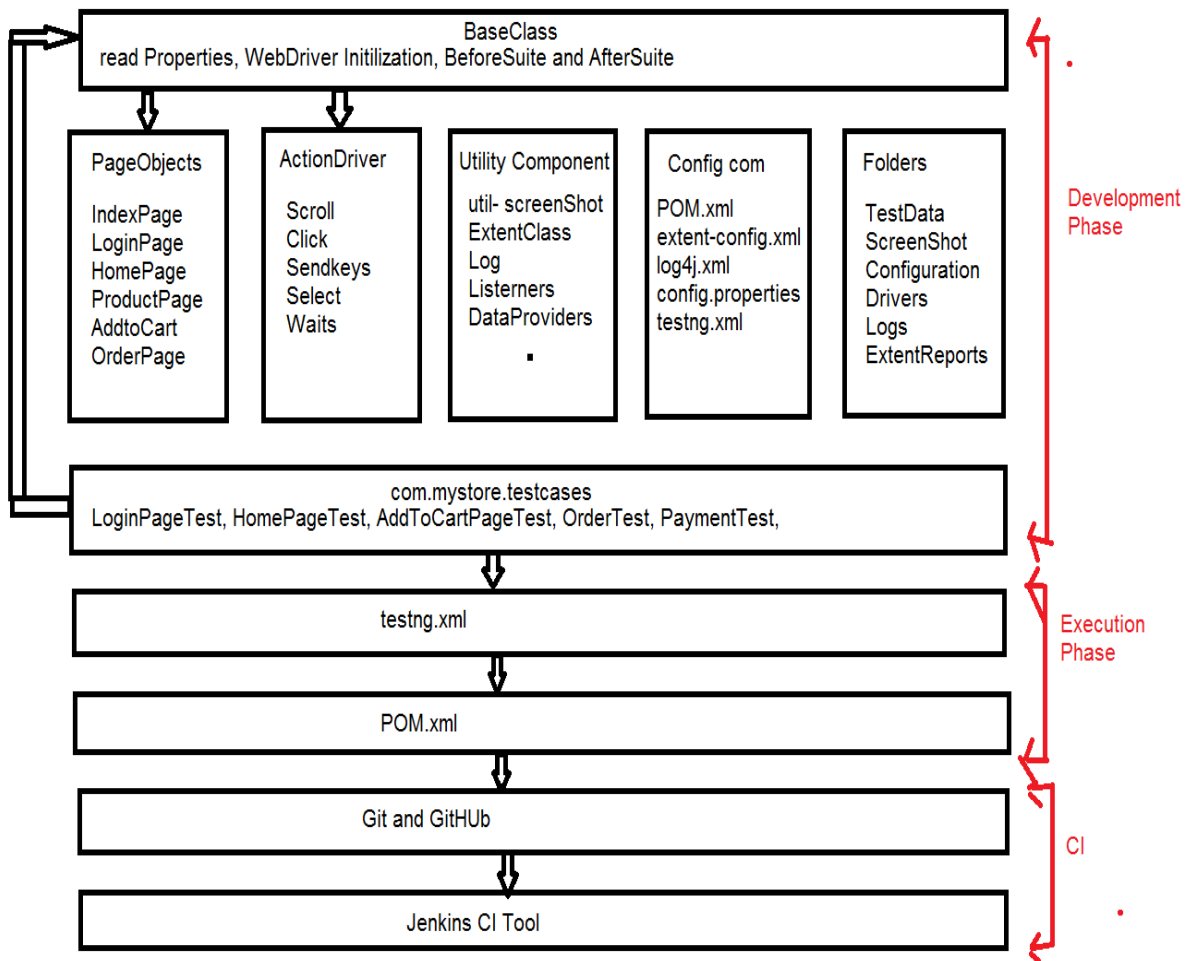


Fig.6.4.1.Framework Architecture

## 6.5.TEST AUTOMATION LIFE CYCLE

Test Automation is a basic order of testing procedure, that includes all the phases of the testing life cycle with extra stages that contain test readiness and automation execution procedure. This whole cycle is known as Automation Testing Life Cycle.

As we know that every Software Testing has to follow the Software Testing Life Cycle (STLC) to have the best outcome of software, In the same way, Automation also should follow Automation Testing Life Cycle (ATLC) to have the best automation framework and its result.

Automation Testing Life Cycle (ATLC) enables the Automation team to decide the most important factors of automation projects in advance which takes care of mistakes at an early stage in automation.



Fig.6.5.1.Test Automation Life Cycle

## **CHAPTER7**

### **COMPONENTS**

#### **JDK(JAVA DEVELOPMENT KIT)**

The Java Development Kit (JDK) is a **software development environment used for developing Java applications and applets**. It includes the Java Runtime Environment (JRE), an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) and other tools needed in Java development.

The latest version of Java is **Java 18 or JDK 18** released on March, 22<sup>nd</sup> 2022.

#### **JRE(JAVA RUNTIME ENVIRONMENT)**

A JRE is made up of a **Java virtual machine (JVM), Java class libraries and the Java class loader**. JDKs are used to develop Java software; JREs provide programming tools and deployment technologies; and JVMs execute Java programs.

#### **JVM(JAVA VIRTUAL MACHINE)**

A Java virtual machine (JVM) is a **virtual machine that enables a computer to run Java programs as well as programs written in other languages that are also compiled to Java byte code**. The JVM is detailed by a specification that formally describes what is required in a JVM implementation.

#### **SELENIUM**

Selenium is an open-source, automated, and valuable testing tool that all web application developers should be well aware of. A test performed using Selenium is usually referred to as Selenium automation testing. However, Selenium is not just a single tool but a collection of

tools, each catering to different Selenium automation testing needs. In this tutorial you will learn all about Selenium and the various types of Selenium automation testing tools.

## TESTNG

**TestNG** is an automation testing framework in which NG stands for “Next Generation”. TestNG is inspired by [JUnit](#) which uses the annotations (@). TestNG overcomes the disadvantages of JUnit and is designed to make [end-to-end testing](#) easy.

Using TestNG, you can generate a proper report, and you can easily come to know how many test cases are passed, failed, and skipped. You can execute the failed test cases separately.

## LOG4J

log4j is a reliable, fast and flexible logging framework (APIs) written in Java, which is distributed under the Apache Software License. Log4j is a popular logging package written in Java. Log4j has been ported to the C, C++, C#, Perl, Python, Ruby, and Eiffel languages.

## CHAPTER 8

### FRAMEWORKS

#### 8.1.DATA DRIVEN FRAMEWORK

Data Driven Framework is one of the popular Automation Testing Framework in the current market. Data Driven automated testing is a method in which the test data set is created in the excel sheet, and is then imported into automation testing tools to feed to the software under test.

Selenium Webdriver is a great tool to automate web-based applications. But it does not support read and write operations on excel files.

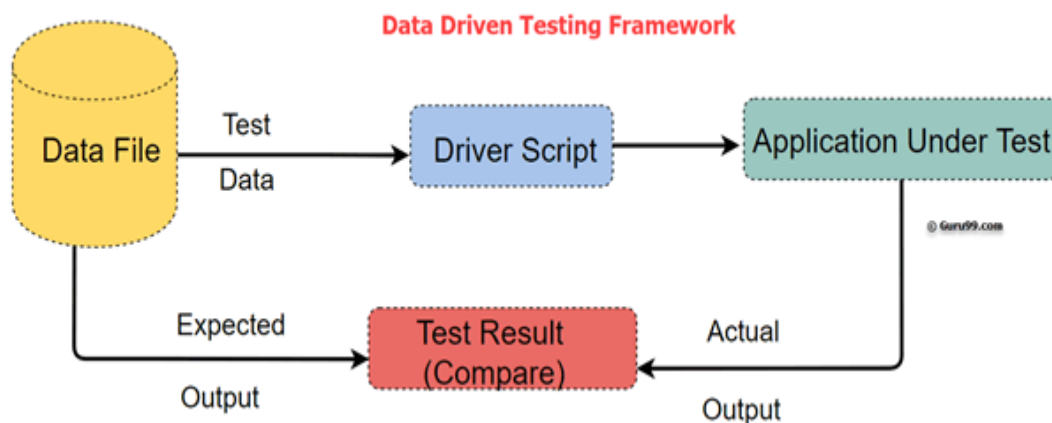


Fig.8.1.1.Data driven Framework

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	Email	Gender	FirstName	LastName	SetPassword	Day	Month	Year	Company	Address	City	State	Zipcode	Country	MobilePhone
1	automation@gmail.com	Mr	TestUser	UserTest	automation	22	7	2000	ABCDEF	EDGF123	San	Alabama	51436	United States	6379512345
2															
3															
4															
5															
6															
7															
8															
9															
10															
11															
12															
13															
14															
15															
16															
17															

Fig.8.1.2.Register Data Sheet

## 8.2.POM(PAGE OBJECT MODEL)

Page Object Model, also known as POM, is a design pattern in Selenium that creates an object repository for storing all web elements. It is useful in reducing code duplication and improves test case maintenance.

In Page Object Model, consider each web page of an application as a class file. Each class file will contain only corresponding web page elements. Using these elements, testers can perform operations on the website under test.

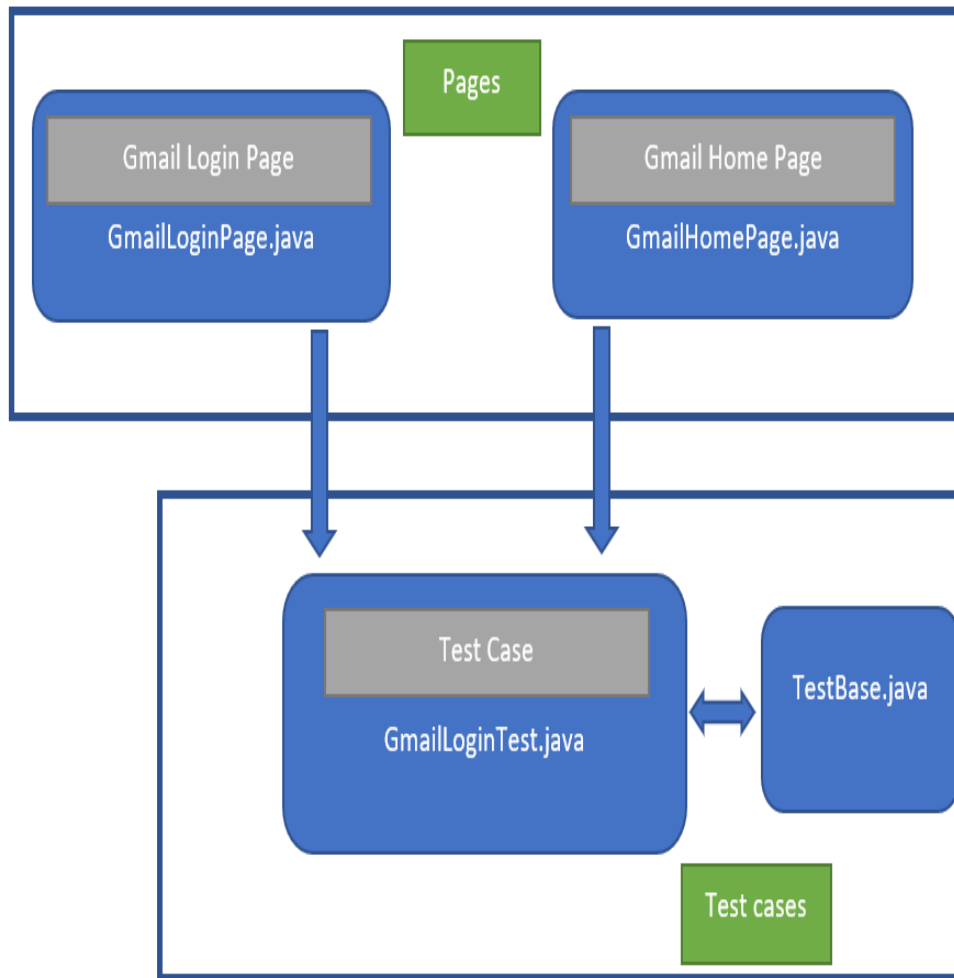


Fig.8.2.1.Page Object Model

### 8.3.TESTNG (TEST NEXT GENERATION)

**TestNG** is an automation testing framework in which NG stands for “Next Generation”. TestNG is inspired by [JUnit](#) which uses the annotations (@). TestNG overcomes the disadvantages of JUnit and is designed to make [end-to-end testing](#) easy.

Using TestNG, you can generate a proper report, and you can easily come to know how many test cases are passed, failed, and skipped. You can execute the failed test cases separately.

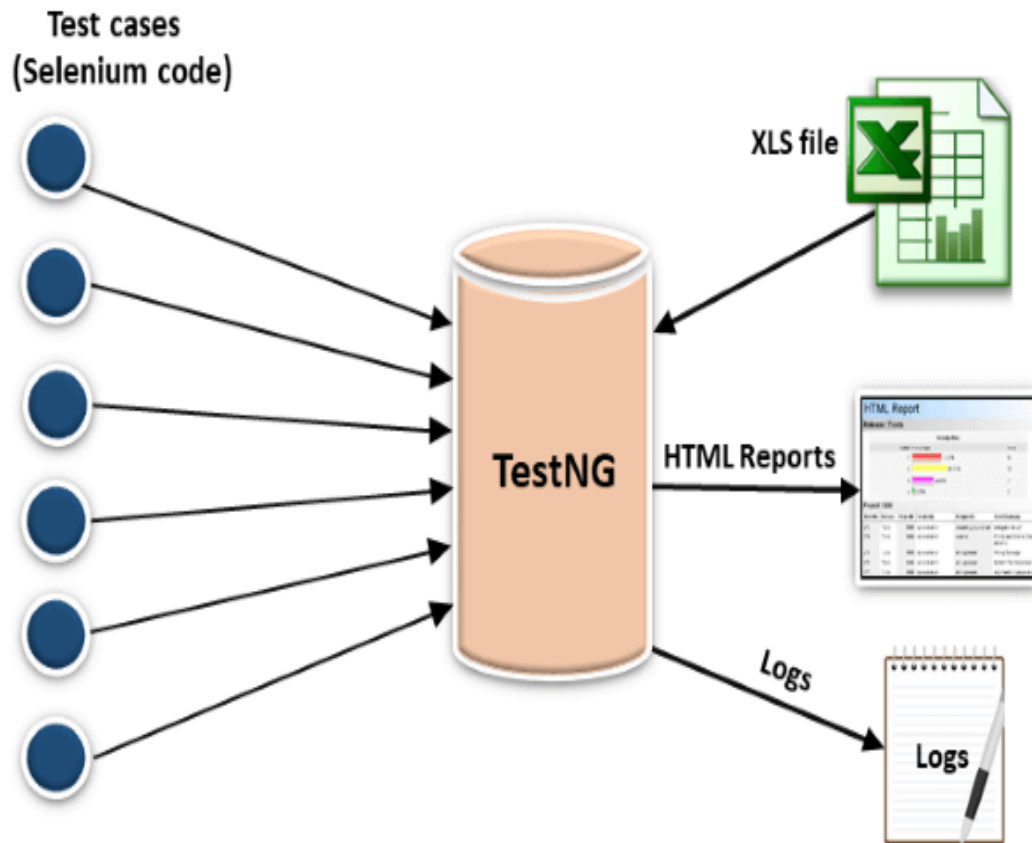


Fig.8.3.1.TestNG Diagram

### 8.3.1.REPORTING

**TestNG Reports** are the default HTML reports which are generated once the test cases are executed using TestNG. These reports help you to identify the information about test cases and the status of a project. TestNG reports in Selenium have three methods `passTest`, `failTest`, and `skipTest` to check the data about test cases.

Report generation is very important when you are doing the Automation Testing as well as for Manual Testing.



- By looking at the result, you can easily identify how many test cases are passed, failed and skipped.
- By looking at the report, you will come to know what the status of the project is.
- The TestNG will generate the default report.
- When you execute testng.xml file, and refresh the project. You will get test-output folder in that folder for reporting in TestNG.
- Right click on the emailable-report.html and select the option. Open with the web browser.

The screenshot shows a web browser window displaying the TestNG Report. The browser has tabs for 'DemoA.java', 'DemoB.java', 'TestNG reports', and 'TestNG Report'. The address bar shows the file path: 'file:///C:/Users/User/Desktop/Guru99/TestProject/test-output/index.html#'. The main content area has a blue header with 'Test results' and '1 suite'. Below this, there's a section titled 'All suites' with a sub-section 'Suite'. The 'Suite' section has a left sidebar with links: 'Info', 'Times' (highlighted), 'Reporter output', 'Ignored methods', and 'Chronological view'. The 'Info' section shows the test file path and summary: '1 test', '0 groups'. The 'Times' section shows a table of test results. The 'Results' section shows '2 methods, 2 passed' and a link to 'Passed methods (show)'.

**Test results**  
1 suite

**All suites**

**Suite**

**Info**

- C:\Users\User\Desktop\Guru99\TestProject\testng.xml
- 1 test
- 0 groups
- **Times**
- Reporter output
- Ignored methods
- Chronological view

**Results**

- 2 methods, 2 passed
- Passed methods (show)

**Times for Suite**

Total running time: 27 seconds

Number	Method	Class	Time (ms)
0	run	com.test.DemoB	14,438
1	run	com.test.DemoA	13,509

Fig.8.3.1.1.TestNG Reporting Page

## 8.3.2.EXTENT REPORTS

Extent Reports is **an open-source reporting library useful for test automation**. It can be easily integrated with major testing frameworks like JUnit, NUnit, TestNG, etc. These reports are HTML documents that depict results as pie charts.

Extent Reports class is used **to generate an HTML report on the user-specified path**. The Boolean flag indicates if the existing report needs to be overwritten or a new report needs to be created. Value 'true' is the default value, which means all the existing data will be overwritten.

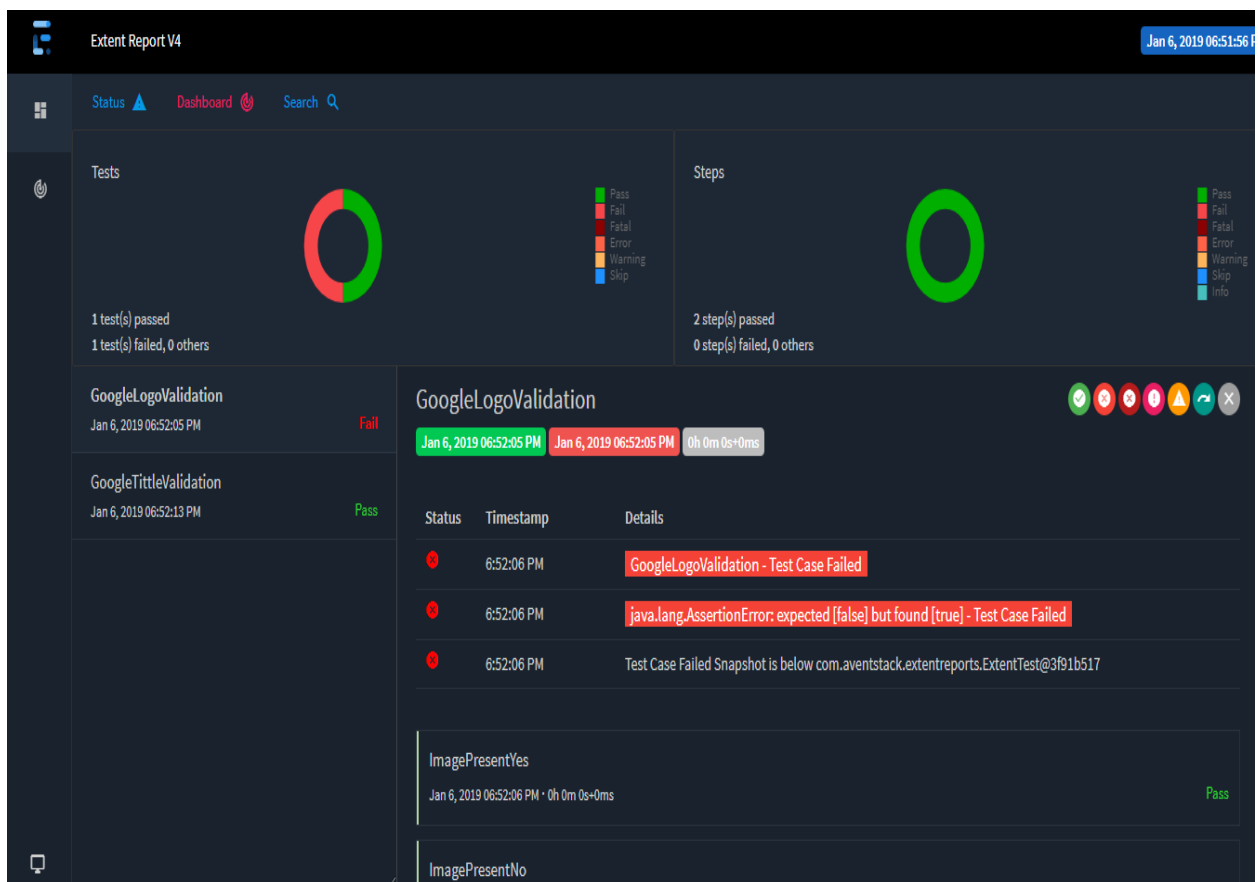


Fig.8.3.2.1.TestNG Extents Reports Page

## CHAPTER 9

## MANUAL TESTING

Manual testing is **the process of manually testing software for defects**. It requires a tester to play the role of an end user where by they use most of the application's features to ensure correct behaviour.

**Manual Testing** is a type of software testing in which test cases are executed manually by a tester without using any automated tools. The purpose of Manual Testing is to identify

the bugs, issues, and defects in the software application. Manual software testing is the most primitive technique of all testing types and it helps to find critical bugs in the software application.

Any new application must be manually tested before its testing can be automated. Manual Software Testing requires more effort but is necessary to check automation feasibility. Manual Testing concepts does not require knowledge of any testing tool. One of the Software Testing Fundamentals is “**100% Automation is not possible**“. This makes Manual Testing imperative.

### Types of Manual Testing:

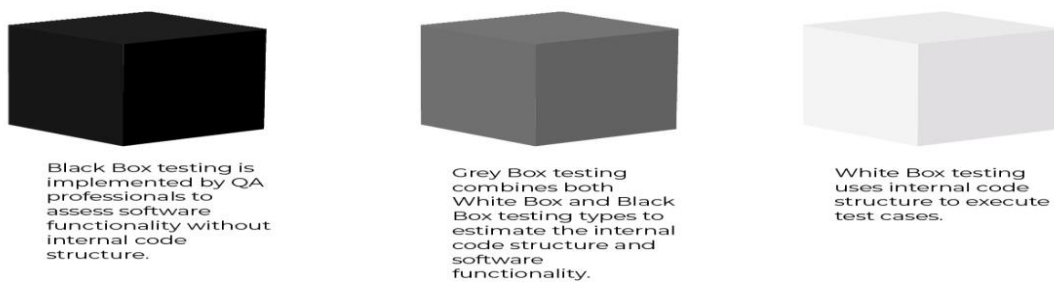


Fig.9.1.1.Manual Testing Types Diagram

## **WHITE BOX TESTING**

The white box testing is done by Developer, where they check every line of a code before giving it to the Test Engineer. Since the code is visible for the Developer during the testing, that's why it is also known as White box testing.

## **BLOCK BOX TESTING**

The black box testing is done by the Test Engineer, where they can check the functionality of an application or the software according to the customer /client's needs. In this, the code is not visible while performing the testing; that's why it is known as black-box testing.

## **GRAY BOX TESTING**

Gray box testing is a combination of white box and Black box testing. It can be performed by a person who knew both coding and testing. And if the single person performs white box, as well as black-box testing for the application, is known as Gray box testing.

Large scale engineering projects that rely on manual software testing follow a more rigorous methodology in order to maximize the number of defects that can be found. A systematic approach focuses on predetermined test cases and generally involves the following steps.

1. Choose a high level test plan where a general methodology is chosen, and resources such as people, computers, and software licenses are identified and acquired.
2. Write detailed test cases, identifying clear and concise steps to be taken by the tester, with expected outcomes.
3. Assign the test cases to testers, who manually follow the steps and record the results.
4. Author a test report, detailing the findings of the testers. The report is used by managers to determine whether the software can be released, and if not, it is used by engineers to identify and correct the problems.

A rigorous test case based approach is often traditional for large software engineering projects that follow a Waterfall model. However, at least one recent study did not show a dramatic difference in defect detection efficiency between exploratory testing and test case based testing.

Testing can be through black-, white- or grey-box testing. In white-box testing the tester is concerned with the execution of the statements through the source code. In black-box testing the software is run to check for the defects and is less concerned with how the processing of the input is done. Black-box testers do not have access to the source code. Grey-box testing is concerned with running the software while having an understanding of the source code and algorithms.

Static and dynamic testing approach may also be used. Dynamic testing involves running the software. Static testing includes verifying requirements, syntax of code and any other activities that do not include actually running the code of the program.

Testing can be further divided into functional and non-functional testing. In functional testing the tester would check the calculations, any link on the page, or any other field which on given input, output may be expected. Non-functional testing includes testing performance, compatibility and fitness of the system under test, its security and usability among other things.

## **STAGES**

### **Unit Testing**

This initial stage in testing normally carried out by the developer who wrote the code and sometimes by a peer using the white box testing technique.

### **Integration Testing**

This stage is carried out in two modes, as a complete package or as an increment to the earlier package. Most of the time black box testing technique is used. However, sometimes a combination of Black and White box testing is also used in this stage.

## **System Testing**

In this stage the software is tested from all possible dimensions for all intended purposes and platforms. In this stage Black box testing technique is normally used.

## **User Acceptance Testing**

This testing stage carried out in order to get customer sign-off of finished product. A 'pass' in this stage also ensures that the customer has accepted the software and is ready for their use.

## **Release or Deployment Testing**

Onsite team will go to customer site to install the system in customer configured environment and will check for the following points:

1. Whether SetUp.exe is running or not.
2. There are easy screens during installation
3. How much space is occupied by system on HDD
4. Is the system completely uninstalled when opted to uninstall from the system.

## CHAPTER 10

### AUTOMATION TESTING

**Automation Testing** is a software testing technique that performs using special automated testing software tools to execute a test case suite. On the contrary, Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.

The automation testing software can also enter test data into the System Under Test, compare expected and actual results and generate detailed test reports. Software Test Automation demands considerable investments of money and resources.

Successive development cycles will require execution of same test suite repeatedly. Using a test automation tool, it's possible to record this test suite and re-play it as required. Once the test suite is automated, no human intervention is required. This improved ROI of Test Automation. The goal of Automation is to reduce the number of test cases to be run manually and not to eliminate Manual Testing altogether.

**Test Automation** is the best way to increase the effectiveness, test coverage, and execution speed in software testing. Automated software testing is important due to the following reasons:

- Manual Testing of all workflows, all fields, all negative scenarios is time and money consuming
- It is difficult to test for multilingual sites manually
- Test Automation in software testing does not require Human intervention. You can run automated test unattended (overnight)
- Test Automation increases the speed of test execution
- Automation helps increase Test Coverage
- Manual Testing can become boring and hence error-prone.

## **SELENIUM IDE**

Next section of the Selenium Automation Testing tutorial covers everything you need to know about Selenium IDE. Shinya Kasatani developed the Selenium Integrated Development Environment (IDE) in 2006. Conventionally, it is an easy-to-use interface that records the user interactions to build automated test scripts. It is a Firefox or Chrome plugin, generally used as a prototyping tool. It was mainly developed to speed up the creation of automation scripts.

IDE ceased to exist in August 2017 when Firefox upgraded to the new Firefox 55 version, which no longer supported Selenium IDE. Applitools rewrote the old Selenium IDE and released a new version recently.

The latest version came with several advancements, such as:

- Reusability of test scripts
- Debugging test scripts
- Selenium side runner
- Provision for control flow statements
- Improved locator functionality

## **INSTALLING IDE**

Step 1- Open the Firefox browser

Step 2- Click on the menu in the top right corner

Step 3- Click on Add-ons in the drop-down box.

Step 4- Click on Find more add-ons and type “Selenium IDE”



Step 5- Click on Add to Firefox

## **SELENIUM REMOTE CONTROL (RC)**

Another interesting topic that this Selenium automation testing tutorial covers is Remote Control or RC. It also covers the reason why RC came into existence in the first place. Paul Hammant developed Selenium Remote Control.

Initially, Selenium-Core was called "JavaScriptTestRunner," a tool built by Jason Huggins in 2004. It was a set of JavaScript functions that interpreted and executed Selenese commands using the browser's built-in JavaScript interpreter. Selenium-Core was then injected into the web browser.

## **SELENIUM WEBDRIVER**

Next section in the Selenium Automation Testing tutorial covers everything about Selenium WebDriver. Developed by Simon Stewart in 2006, Selenium WebDriver was the first cross-platform testing framework that could configure and control the browsers on the OS level. It served as a programming interface to create and run test cases.

- Selenium test script - Selenium test script is the test code written in any of the mentioned programming languages that are interpreted by the driver.
- JSON Wire Protocol - JSON Wire Protocol provides a transport mechanism to transfer data between a server and a client. JSON Wire Protocol serves as an industry standard for various web services.
- Browser drivers - Selenium uses drivers, specific to each browser to establish a secure connection with the browser.
- Browsers - Selenium WebDriver supports various web browsers on which to test and run applications.

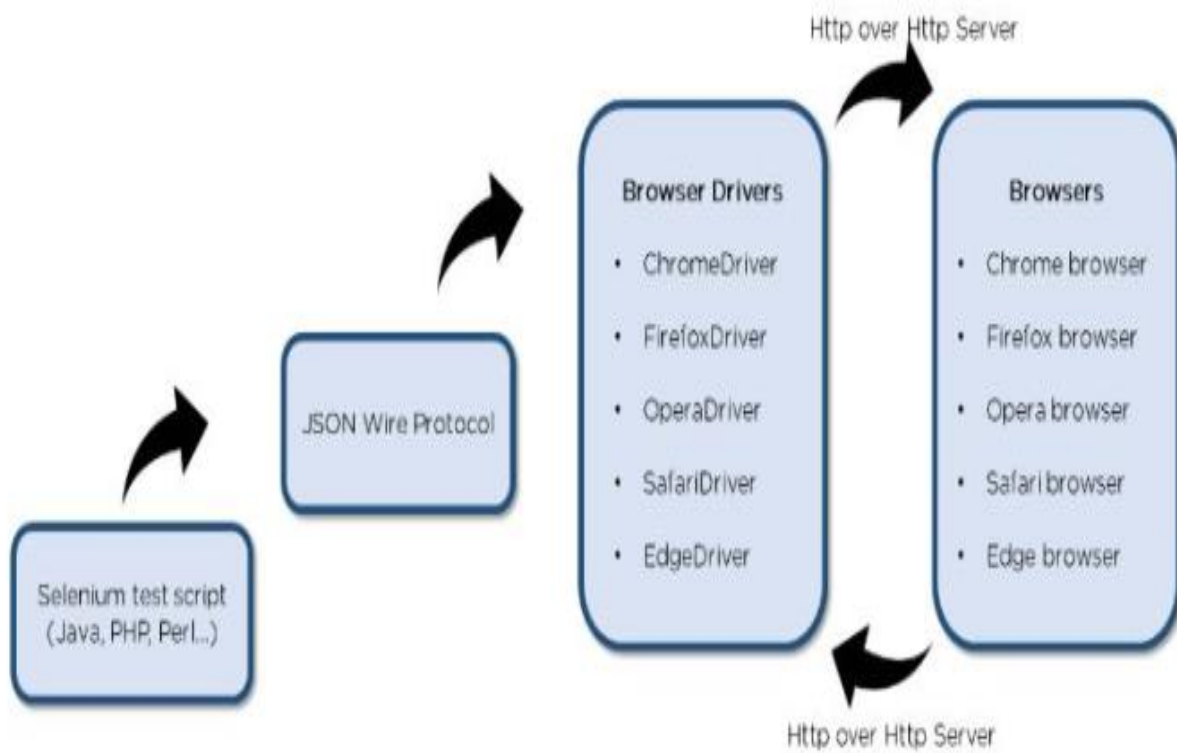


Fig.10.1.1.Selenium WebDriver architecture

## **CHAPTER 11**

### **CONCLUSION**

#### **11.1.CONCLUSION AND FUTURE WORK**

Conclusion and Future Work Software testing has a prime importance in software's verification and validation. It is important because of two main reasons, first, it assures software quality, and second, nearly 60% of the total software's cost is spend over different types of testing. Regression testing is the object of interest in this paper and we have detailed the effects of automation of regression tests over the software's cost, quality and time to market. In our experiments in we have used three different software's which are, developed in versions. We have formulated mathematical models, based on the various significant testing effort factors, to quantify the impact of test automation. Although, automation of test cases have a high implementation and maintenance costs, from our experiments we have found that, automation of test cases can give remarkable returns in the long runs where we run and rerun the automated-tests, multiple times. We have also found that test automation has positive effects on software quality. Hence we can claim that test automation increases the overall effectiveness of the testing process when we have repetitive testing tasks which are similar. This work can be extended in future by adding more variable automation cost factors in the analysis to make it more precise and accurate. The concept of automatic test data generation can also be combined with the present research.

## CHAPTER 12

### SOURCE CODE

```
package com.mystore.testcases;

import java.util.HashMap;

import org.testng.Assert;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Parameters;

import org.testng.annotations.Test;

import com.mystore.base.BaseClass;

import com.mystore.dataprovider.DataProviders;

import com.mystore.pageobjects.AccountCreationPage;

import com.mystore.pageobjects.HomePage;

import com.mystore.pageobjects.IndexPage;

import com.mystore.pageobjects.LoginPage;


import com.mystore.utility.Log;


public class AccountCreationPageTest extends BaseClass {

    private IndexPageindexPage;

    private LoginPageloginPage;
```

```

private AccountCreationPageaccountCreationPage;

private HomePagehomePage;

@Parameters("browser")

@BeforeMethod(groups = {"Smoke","Sanity","Regression"})

public void setup(String browser) {

launchApp(browser);

}

@AfterMethod(groups = {"Smoke","Sanity","Regression"})

public void tearDown() {

getDriver().quit();

}

@Test(groups = {"Sanity"},dataProvider = "email", dataProviderClass =
DataProviders.class)

public void verifyCreateAccountPageTest(String email) throws Throwable {

Log.startTestCase("verifyCreateAccountPageTest");

indexPage= new IndexPage();

loginPage=indexPage.clickOnSignIn();

accountCreationPage=loginPage.createNewAccount(email);

boolean result=accountCreationPage.validateAccountCreatePage();

Assert.assertTrue(result);

Log.endTestCase("verifyCreateAccountPageTest");

}

```

```

@Test(groups = "Regression",dataProvider = "newAccountDetailsData",dataProviderClass =
DataProviders.class)

public void createAccountTest(HashMap<String,String>hashMapValue) throws Throwable {

Log.startTestCase("createAccountTest");

indexPage= new IndexPage();

loginPage=indexPage.clickOnSignIn();

accountCreationPage=loginPage.createNewAccount(hashMapValue.get("Email"));

accountCreationPage.createAccount(

hashMapValue.get("Gender"),

hashMapValue.get("FirstName"),

hashMapValue.get("LastName"),

hashMapValue.get("SetPassword"),

hashMapValue.get("Day"),

hashMapValue.get("Month"),

hashMapValue.get("Year"),

hashMapValue.get("Company"),

hashMapValue.get("Address"),

hashMapValue.get("City"),

hashMapValue.get("State"),

hashMapValue.get("Zipcode"),

hashMapValue.get("Country"),

hashMapValue.get("MobilePhone"));

homePage=accountCreationPage.validateRegistration();

```

```

Assert.assertEquals("http://automationpractice.com/index.php?controller=my-account",
homePage.getCurrURL());

Log.endTestCase("createAccountTest");

}

}

public class LoginPageTest extends BaseClass {
private IndexPageindexPage;
private LoginPageloginPage;
private HomePagehomePage;

@Parameters("browser")
@BeforeMethod(groups = {"Smoke","Sanity","Regression"})
public void setup(String browser) {
launchApp(browser);
}

@AfterMethod(groups = {"Smoke","Sanity","Regression"})
public void tearDown() {
getDriver().quit();
}

@Test(groups = {"Smoke","Sanity"},dataProvider = "credentials", dataProviderClass =
DataProviders.class)
public void loginTest(String uname, String pswd) throws Throwable {
Log.startTestCase("loginTest");
indexPage= new IndexPage();
Log.info("user is going to click on SignIn");
loginPage=indexPage.clickOnSignIn();
Log.info("Enter Username and Password");
//homePage=loginPage.login(prop.getProperty("username"), prop.getProperty("password"));
homePage=loginPage.login(uname,pswd,homePage);

```

```

String actualURL=homePage.getCurrURL();
String expectedURL="http://automationpractice.com/index.php?controller=my-account";
Log.info("Verifying if user is able to login");
Assert.assertEquals(actualURL, expectedURL);
Log.info("Login is Sucess");
Log.endTestCase("loginTest");
}
}

public class HomePageTest extends BaseClass {
private IndexPageindexPage;
private LoginPageloginPage;
private HomePagehomePage;

@Parameters("browser")
@BeforeMethod(groups = {"Smoke","Sanity","Regression"})
public void setup(String browser) {
launchApp(browser);
}

@AfterMethod(groups = {"Smoke","Sanity","Regression"})
public void tearDown() {
getDriver().quit();
}

@Test(groups = {"Smoke"},dataProvider = "credentials", dataProviderClass =
DataProviders.class)
public void wishListTest(String uname, String pswd) throws Throwable {
Log.startTestCase("wishListTest");
indexPage= new IndexPage();
loginPage=indexPage.clickOnSignIn();
homePage=loginPage.login(uname,pswd,homePage);
}
}

```



```

boolean result=homePage.validateMyWishList();
Assert.assertTrue(result);
Log.endTestCase("wishListTest");
}

```

```

@Test(groups = "Smoke",dataProvider = "credentials", dataProviderClass =
DataProviders.class)
public void orderHistoryandDetailsTest(String uname, String pswd) throws Throwable {
Log.startTestCase("orderHistoryandDetailsTest");
indexPage= new IndexPage();
loginPage=indexPage.clickOnSignIn();
homePage=loginPage.login(uname,pswd,homePage);
boolean result=homePage.validateOrderHistory();
Assert.assertTrue(result);
Log.endTestCase("orderHistoryandDetailsTest");
}

```

```

}

public class SearchResultPageTest extends BaseClass {
private IndexPage index;
private SearchResultPagesearchResultPage;

```

```

@Parameters("browser")
@BeforeMethod(groups = {"Smoke","Sanity","Regression"})
public void setup(String browser) {
launchApp(browser);
}

```

```

@AfterMethod(groups = {"Smoke","Sanity","Regression"})
public void tearDown() {
getDriver().quit();
}

```

```
}
```

```
@Test(groups = "Smoke",dataProvider = "searchProduct", dataProviderClass =  
DataProviders.class)
```

```
public void productAvailabilityTest(String productName) throws Throwable {
```

```
Log.startTestCase("productAvailabilityTest");
```

```
index= new IndexPage();
```

```
searchResultPage=index.searchProduct(productName);
```

```
boolean result=searchResultPage.isProductAvailable();
```

```
Assert.assertTrue(result);
```

```
Log.endTestCase("productAvailabilityTest");
```

```
}
```

```
}
```

```
public class AddToCartPageTest extends BaseClass {
```

```
private IndexPage index;
```

```
private SearchResultPagesearchResultPage;
```

```
private AddToCartPageaddToCartPage;
```

```
@Parameters("browser")
```

```
@BeforeMethod(groups = {"Smoke", "Sanity", "Regression"})
```

```
public void setup(String browser) {
```

```
launchApp(browser);
```

```
}
```

```
@AfterMethod(groups = {"Smoke", "Sanity", "Regression"})
```

```
public void tearDown() {
```

```
getDriver().quit();
```

```
}
```

```

@Test(groups = {"Regression","Sanity"}, dataProvider = "getProduct", dataProviderClass =
DataProviders.class)
public void addToCartTest(String productName, String qty, String size) throws Throwable {
    Log.startTestCase("addToCartTest");
    index= new IndexPage();
    searchResultPage=index.searchProduct(productName);
    addToCartPage=searchResultPage.clickOnProduct();
    addToCartPage.enterQuantity(qty);
    addToCartPage.selectSize(size);
    addToCartPage.clickOnAddToCart();
    boolean result=addToCartPage.validateAddtoCart();
    Assert.assertTrue(result);
    Log.endTestCase("addToCartTest");

}

}

public class OrderPageTest extends BaseClass {

private IndexPage index;
private SearchResultPagesearchResultPage;
private AddToCartPageaddToCartPage;
private OrderPageorderPage;

@Parameters("browser")
@BeforeMethod(groups = {"Smoke","Sanity","Regression"})
public void setup(String browser) {
    launchApp(browser);
}

@AfterMethod(groups = {"Smoke","Sanity","Regression"})
public void tearDown() {

```

```

    getDriver().quit();
}

```

```

@Test(groups = "Regression",dataProvider = "getProduct", dataProviderClass =
DataProviders.class)
public void verifyTotalPrice(String productName, String qty, String size) throws Throwable {
    Log.startTestCase("verifyTotalPrice");
    index= new IndexPage();
    searchResultPage=index.searchProduct(productName);
    addToCartPage=searchResultPage.clickOnProduct();
    addToCartPage.enterQuantity(qty);
    addToCartPage.selectSize(size);
    addToCartPage.clickOnAddToCart();
    orderPage=addToCartPage.clickOnCheckOut();
    Double unitPrice=orderPage.getUnitPrice();
    Double totalPrice=orderPage.getTotalPrice();
    Double totalExpectedPrice=(unitPrice*(Double.parseDouble(qty)))+2;
    Assert.assertEquals(totalPrice, totalExpectedPrice);
    Log.endTestCase("verifyTotalPrice");
}
}

public class EndToEndTest extends BaseClass {

```

```

    private IndexPage index;
    private SearchResultPage searchResultPage;
    private AddToCartPage addToCartPage;
    private OrderPage orderPage;
    private LoginPage loginPage;
    private AddressPage addressPage;
    private ShippingPage shippingPage;
    private PaymentPage paymentPage;

```

```

private OrderSummaryorderSummary;
private OrderConfirmationPageorderConfirmationPage;

@Parameters("browser")
@BeforeMethod(groups = {"Smoke","Sanity","Regression"})
public void setup(String browser) {
    launchApp(browser);
}

@AfterMethod(groups = {"Smoke","Sanity","Regression"})
public void tearDown() {
    getDriver().quit();
}

@Test(groups = "Regression",dataProvider = "getProduct", dataProviderClass =
DataProviders.class)
public void endToEndTest(String productName, String qty, String size) throws Throwable {
    Log.startTestCase("endToEndTest");
    index= new IndexPage();
    searchResultPage=index.searchProduct(productName);
    addToCartPage=searchResultPage.clickOnProduct();
    addToCartPage.enterQuantity(qty);
    addToCartPage.selectSize(size);
    addToCartPage.clickOnAddToCart();
    orderPage=addToCartPage.clickOnCheckOut();
    loginPage=orderPage.clickOnCheckOut();
    addressPage=loginPage.login(prop.getProperty("username")),
prop.getProperty("password"),addressPage);
    shippingPage=addressPage.clickOnCheckOut();
    shippingPage.checkTheTerms();
    paymentPage=shippingPage.clickOnProceedToCheckOut();

```

```

orderSummary=paymentPage.clickOnPaymentMethod();
orderConfirmationPage=orderSummary.clickOnconfirmOrderBtn();
String actualMessage=orderConfirmationPage.validateConfirmMessage();
String expectedMsg="Your order on My Store is complete.";
Assert.assertEquals(actualMessage, expectedMsg);
Log.endTestCase("endToEndTest");
}

```

```

}

```

```

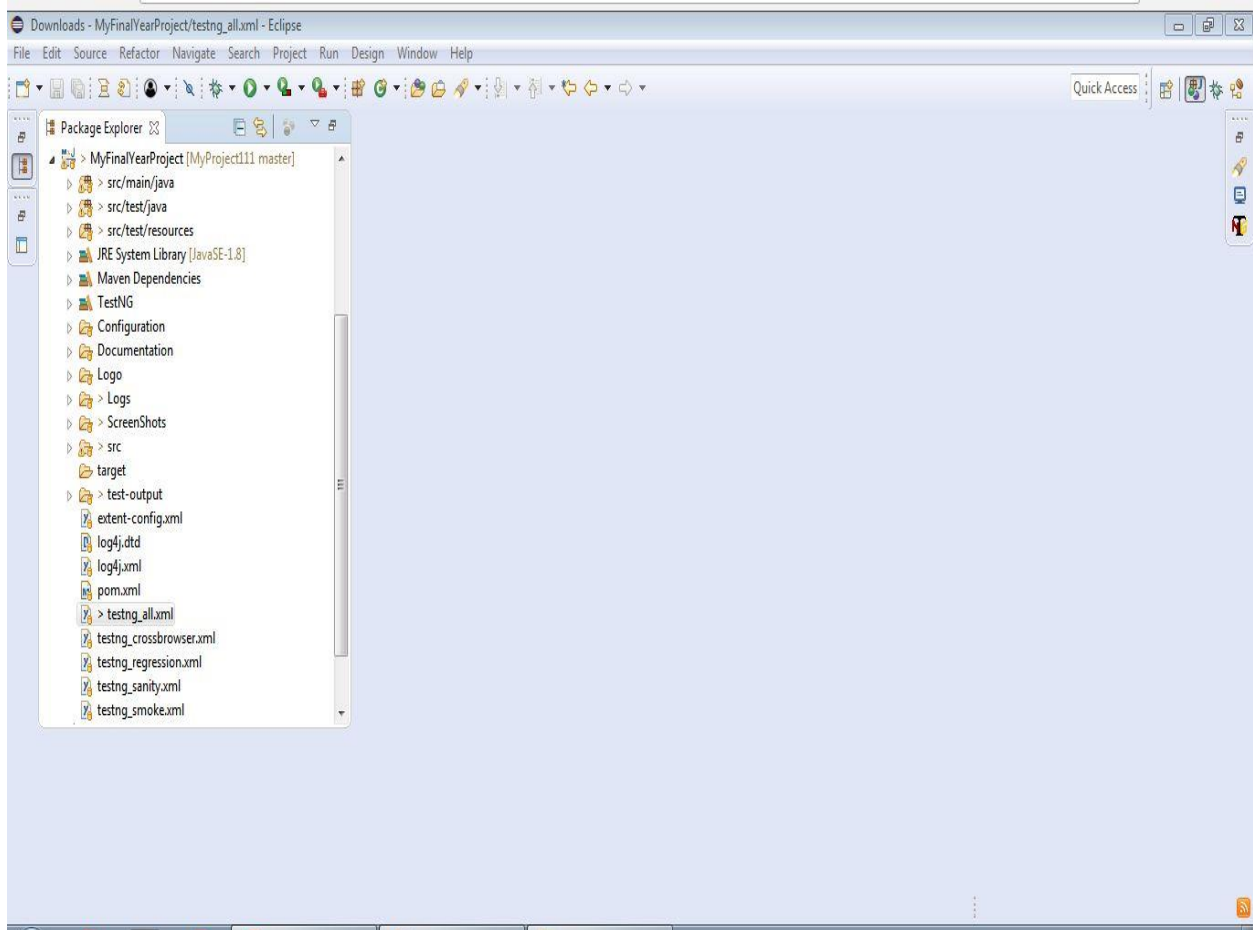
@DataProvider(name = "credentials")
public Object[][] getCredentials() {
// Totals rows count
int rows = obj.getRowCount("Credentials");
// Total Columns
int column = obj.getColumnCount("Credentials");
int actRows = rows - 1;
Object[][] data = new Object[actRows][column];
for (int i = 0; i<actRows; i++) {
for (int j = 0; j < column; j++) {
data[i][j] = obj.getCellData("Credentials", j, i + 2);
}
}
return data;
}

```

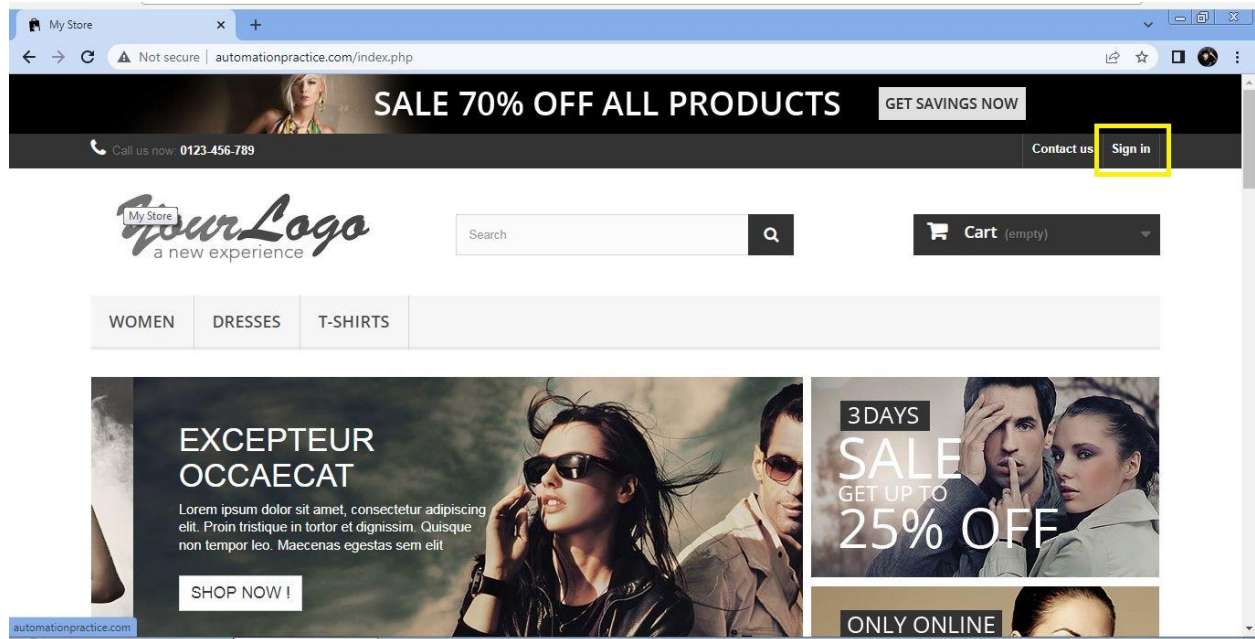
## CHAPTER13

## SCREENSHOTS

### PROJECT STRUCTURE



**CLICK THE LOG IN BUTTON**





# CREATE ACCOUNT

Login - My Store

Not secure | automationpractice.com/index.php?controller=authentication&back=my-account

WOMEN DRESSES T-SHIRTS

Authentication

AUTHENTICATION

**CREATE AN ACCOUNT**

Please enter your email address to create an account.

Email address  
admin@gmail.com ✓

Create an account

**ALREADY REGISTERED?**

Email address

Password

Forgot your password?

Sign in

## REGISTER PAGE

Browser: Login - My Store  
URL: automationpractice.com/index.php?controller=authentication&back=my-account#account-creation

State \*

Zip/Postal Code \*

Country \*

United States

Additional information

You must register at least one phone number.

Home phone

Mobile phone \*

Assign an address alias for future reference. \*

My address

**Register >**

\*Required field

## LOG IN PAGE

The screenshot shows a web browser window with the title 'Login - My Store'. The address bar displays 'Not secure | automationpractice.com/index.php?controller=authentication&back=my-account'. The page features a navigation bar with 'WOMEN', 'DRESSES', and 'T-SHIRTS' buttons. Below this is a breadcrumb trail 'Authentication'. The main content area is titled 'AUTHENTICATION' and contains two panels. The left panel, 'CREATE AN ACCOUNT', prompts the user to enter an email address and includes a 'Create an account' button. The right panel, 'ALREADY REGISTERED?', contains fields for 'Email address' (with 'admin@xyz.com' entered) and 'Password', a 'Forgot your password?' link, and a 'Sign in' button. The 'Sign in' button is highlighted with a yellow border.

Login - My Store

Not secure | automationpractice.com/index.php?controller=authentication&back=my-account

WOMEN DRESSES T-SHIRTS

Authentication

AUTHENTICATION

CREATE AN ACCOUNT

Please enter your email address to create an account.

Email address

Create an account

ALREADY REGISTERED?

Email address

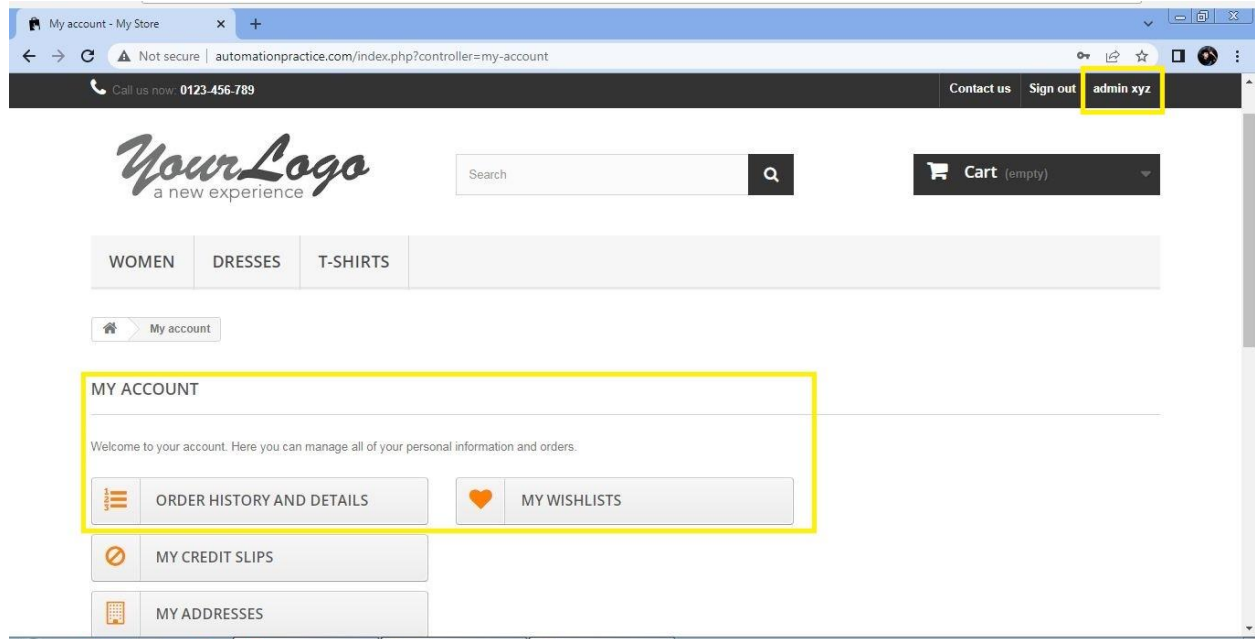
admin@xyz.com

Password

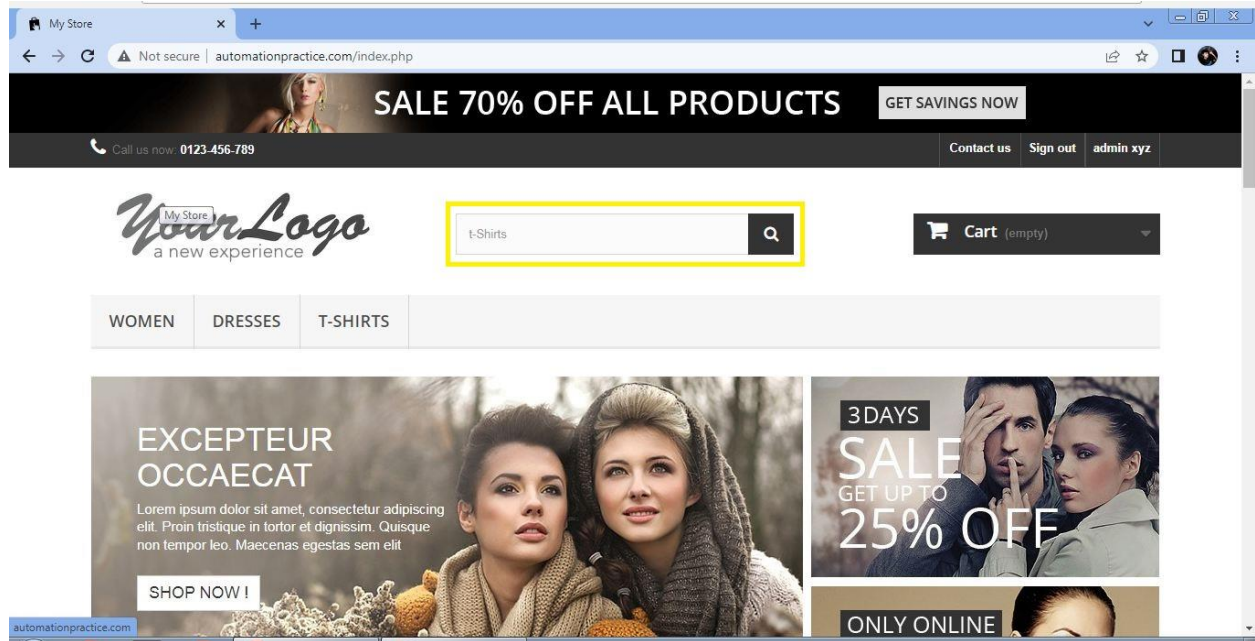
Forgot your password?

Sign in

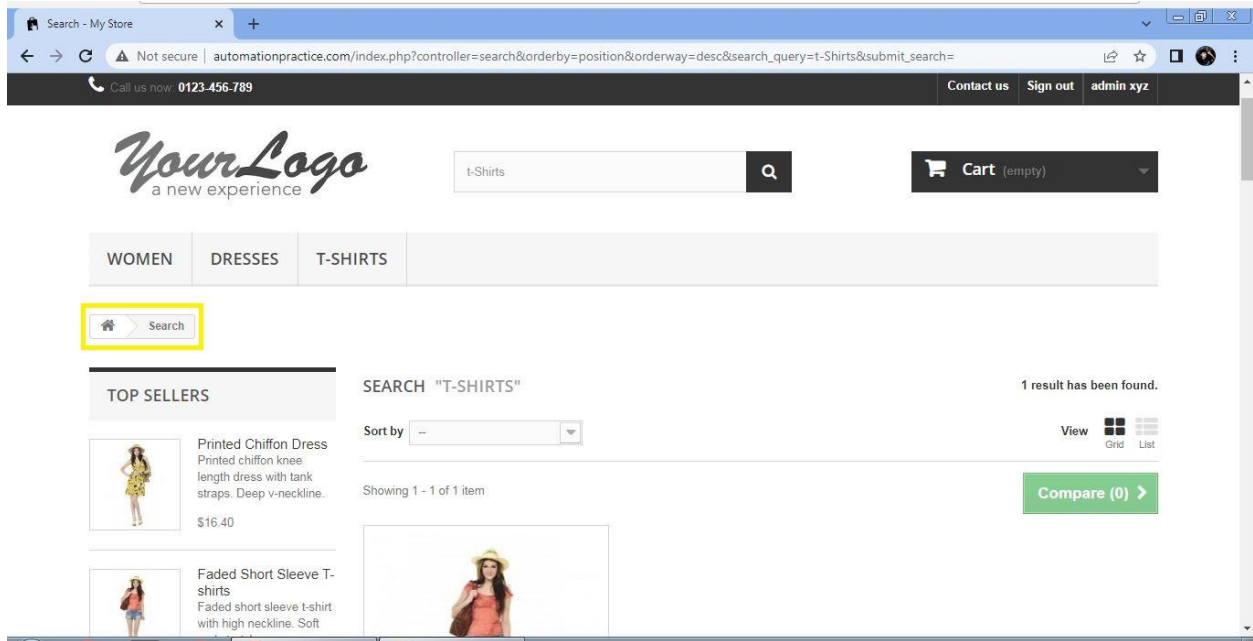
## HOME PAGE



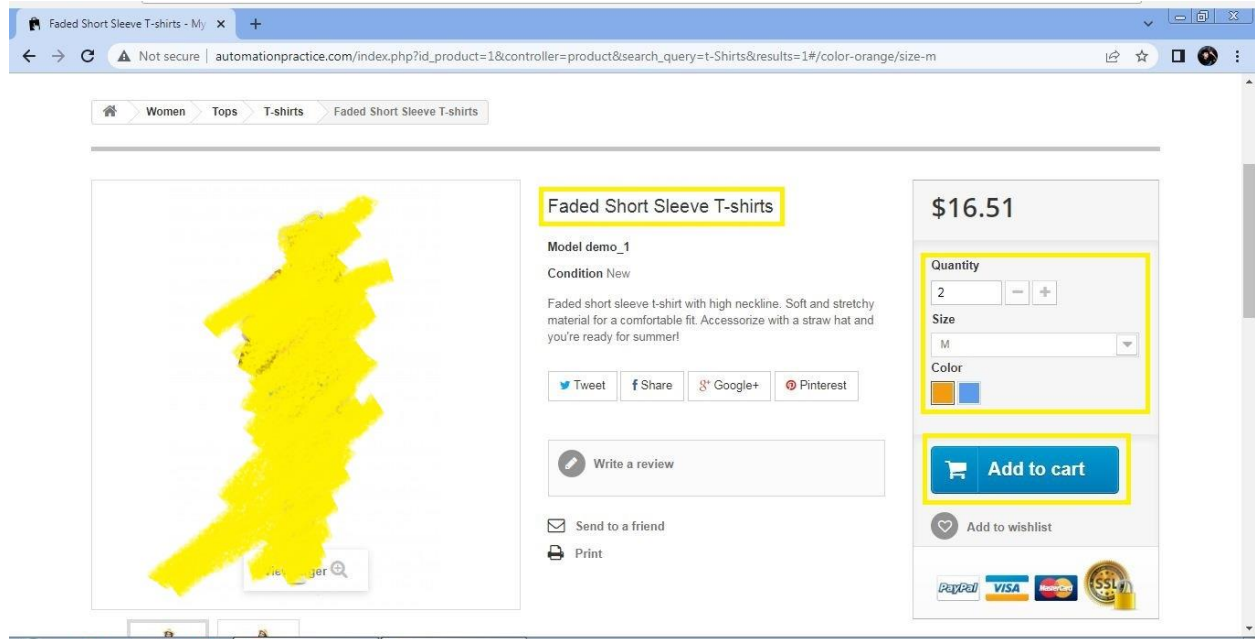
## SEARCH TEST



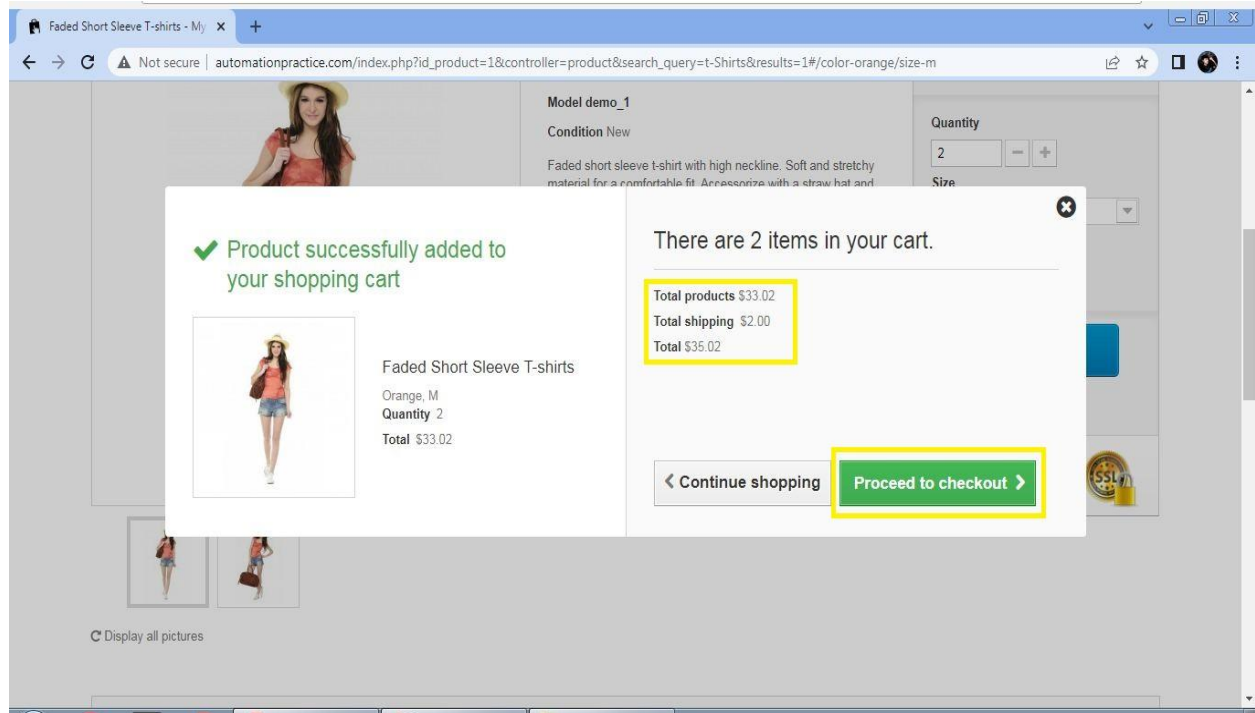
## SEARCH RESULT



## ADD TO CARD



## PROCEED TO CHECK OUT





## CONFIRM ORDER

Order - My Store



Not secure | automationpractice.com/index.php?controller=order

Your shopping cart

SHOPPING-CART SUMMARY

Your shopping cart contains: 2 Products

01. Summary 02. Sign in 03. Address 04. Shipping 05. Payment

Product	Description	Avail.	Unit price	Qty	Total	
	Faded Short Sleeve T-shirts SKU : demo_1 Color : Orange, Size : M	In stock	\$16.51	2	\$33.02	
					Total products	\$33.02
					Total shipping	\$2.00
					Total	\$35.02
					Tax	\$0.00
					TOTAL	\$35.02

# ADDRESS

Order - My Store

Not secure | automationpractice.com/index.php?controller=order&step=1

01. Summary 02. Sign in 03. Address 04. Shipping 05. Payment

Choose a delivery address:

My address

☒ Use the delivery address as the billing address.

**YOUR DELIVERY ADDRESS**

Akhilesh Rawat  
Big Data  
add  
Hapur, Florida 94000  
United States  
0707070702

Update >

**YOUR BILLING ADDRESS**

Akhilesh Rawat  
Big Data  
add  
Hapur, Florida 94000  
United States  
0707070702

Update >

[Add a new address >](#)

If you would like to add a comment about your order, please write it in the field below.

# SHIPPING

Order - My Store

Not secure | automationpractice.com/index.php?controller=order

Shipping

## SHIPPING

01. Summary


02. Sign in

03. Address

04. Shipping

05. Payment

Choose a shipping option for this address: My address

<input type="radio"/>		My carrier Delivery next day!	\$2.00
-----------------------	---	-------------------------------	--------

**Terms of service**

☒ I agree to the terms of service and will adhere to them unconditionally. ([Read the Terms of Service](#))

[Continue shopping](#)


[Proceed to checkout](#)


## PAYMENT PAGE

Order - My Store


Not secure | automationpractice.com/index.php?controller=order&multi-shipping=

01. Summary 02. Sign in 03. Address 04. Shipping 05. Payment

Product	Description	Avail.	Unit price	Qty	Total
	Faded Short Sleeve T-shirts SKU : demo_1 Color : Orange, Size : M	In stock	\$16.51	2	\$33.02
Total products					\$33.02
Total shipping					\$2.00
TOTAL					\$35.02



Pay by bank wire (order processing will be longer) >



Pay by check (order processing will be longer) >

## CHAPTER 14

### REFERENCE

1. Automated software Testing - International Software Test Institute'. [Online]. Available: [https://www.testinstitute.org/Automated\\_Software\\_Testing.php](https://www.testinstitute.org/Automated_Software_Testing.php). [Accessed: 18-Nov-2019].
2. Atlassian, 'About Code Coverage - Atlassian Documentation', 2017. [Online]. Available: <https://confluence.atlassian.com/clover/about-code-coverage-71599496.html>. [Accessed: 29-Sep-2019]
3. Aebersold Kirsten, 'Test Automation Frameworks'. [Online]. Available: <https://smartbear.com/learn/automated-testing/testautomation-frameworks/>. [Accessed: 26-Aug-2019].
4. Altexsoft.com, 'Comparing Automated Testing Tools: Selenium, TestComplete, Ranorex, and more | AltexSoft', 2018. [Online]. Available: <https://www.altexsoft.com/blog/engineering/comparing-automated-testing-tools-selenium-testcomplete-ranorex-and-more/>. [Accessed: 19-Nov-2019].
5. 'A Comparison of Automated Testing Tools | Katalon Studio'. [Online]. Available: <https://www.katalon.com/resourcescenter/blog/comparison-automated-testing-tools/>. [Accessed: 17-May-2019].
6. Brian, 'Best Automation Testing Tools for 2020 (Top 10 reviews)', 2017. [Online]. Available: <https://medium.com/@briananderson2209/best-automation-testing-tools-for-2018-top-10-reviews-8a4a19f664d2>. [Accessed: 20-Nov2019].
7. Dennis, B. H. Wixom, and R. M. Roth, Systems Analysis and Design 5th Edition, 5th Editio. USA: John Wiley & Sons, Inc., 2012.
8. 'JUnit - Wikipedia'. [Online]. Available: <https://en.wikipedia.org/wiki/JUnit>. [Accessed: 19-Nov-2019].
9. Padmini, 'Beginners Guide To Software Testing', pp. 1–41, 2013

10. Stackify, 'Code Coverage Tools: 25 Tools for Testing in C, C++, Java'. [Online]. Available: <https://stackify.com/code-coverage-tools/>. [Accessed: 29-Sep-2019]
11. Selenium HQ Browser Automation'. [Online]. Available: <https://selenium.dev/>. [Accessed: 19-Nov-2019]
12. worldqualityreport.com, 'World Quality Report 2019', 2018

**GNANAMANI COLLEGE OF TECHNOLOGY**  
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**COURSE OUTCOMES**

<b>C411 – PROJECT</b>	<b>C411.1</b>	<b>Estimate</b> the problem by applying acquired knowledge.
	<b>C411.2</b>	<b>Develop</b> the executable project modules after considering risks
	<b>C411.3</b>	<b>Choose</b> efficient tools for designing project modules.
	<b>C411.4</b>	<b>Combine</b> all the modules through effective team work after efficient testing.
	<b>C411.5</b>	<b>Elaborate</b> the completed task and compile the project report.

**CO-PO MATRICES**

<b>C411 -PROJECT</b>	<b>CO/PO</b>	<b>PO 1</b>	<b>PO 2</b>	<b>PO 3</b>	<b>PO 4</b>	<b>PO 5</b>	<b>PO 6</b>	<b>PO 7</b>	<b>PO 8</b>	<b>PO 9</b>	<b>PO 10</b>	<b>PO 11</b>	<b>PO 12</b>	<b>PSO 1</b>	<b>PSO 2</b>
	<b>C411.1</b>	3	2	2	2	2	2	-	2	3	2	2	3	3	3
	<b>C411.2</b>	3	3	2	3	3	3	-	3	3	3	2	3	3	3
	<b>C411.3</b>	3	3	3	3	3	2	2	3	3	3	2	3	2	3
	<b>C411.4</b>	3	3	3	3	3	2	3	3	3	3	3	3	3	3
	<b>C411.5</b>	3	3	3	3	3	2	3	3	3	3	3	3	3	3
	<b>C411</b>	<b>3.00</b>	<b>2.80</b>	<b>2.60</b>	<b>2.80</b>	<b>2.80</b>	<b>2.20</b>	<b>2.67</b>	<b>2.80</b>	<b>3.00</b>	<b>2.80</b>	<b>2.40</b>	<b>3.00</b>	<b>2.80</b>	<b>3.00</b>

## **LIST OF PUBLICATIONS**

Mr. DINESH KUMAR.K, CHINNADURAI.A, GOPINATH.I, GOPINATH.N, MANOJKUMAR.M **“Automation Software Testing Using Datadriven and POM Framework”** in the TNSCST Sponsored 8<sup>th</sup> International Conference on Latest Trends in Science, Engineering and Technology (ICLTSET’22) organized by Karpagam Institute of Technology, Coimbatore held on 20<sup>th</sup> & 21<sup>th</sup> of May 2022.



TNSCST Sponsored

## 8<sup>th</sup> International Conference on Latest Trends in Science, Engineering & Technology

### CERTIFICATE OF PARTICIPATION

This is to certify that Mr./Ms./Dr. **K.Dinesh Kumar** from **Gnanamani College of Technology** has presented a paper titled **Automation Software Testing Using Datadriven and POM Framework** in the TNSCST Sponsored 8<sup>th</sup> International Conference on Latest Trends in Science, Engineering and Technology (ICLTSET'22) organized by Karpagam Institute of Technology, Coimbatore held on 20<sup>th</sup> & 21<sup>st</sup> of May 2022.

  
**Dr.S.Gopinath**  
Convener

  
**Dr.D.Bhanu**  
Vice Principal

  
**Dr.P.Manimaran**  
Principal

TNSCST Sponsored

## 8<sup>th</sup> International Conference on Latest Trends in Science, Engineering & Technology

### CERTIFICATE OF PARTICIPATION

This is to certify that Mr./Ms./Dr. **Chinnadurai A** from **Gnanamani College of Technology** has presented a paper titled **Automation Software Testing Using Datadriven and POM Framework** in the TNSCST Sponsored 8<sup>th</sup> International Conference on Latest Trends in Science, Engineering and Technology (ICLTSET'22) organized by Karpagam Institute of Technology, Coimbatore held on 20<sup>th</sup> & 21<sup>st</sup> of May 2022.

  
Dr.S.Gopinath  
Convener

  
Dr.D.Bhanu  
Vice Principal

  
Dr.P.Manimaran  
Principal

TNSCST Sponsored

## 8<sup>th</sup> International Conference on Latest Trends in Science, Engineering & Technology

### CERTIFICATE OF PARTICIPATION

This is to certify that Mr./Ms./Dr. **Gopinath I** from **Gnanamani College of Technology** has presented a paper titled **Automation Software Testing Using Datadriven and POM Framework** in the TNSCST Sponsored 8<sup>th</sup> International Conference on Latest Trends in Science, Engineering and Technology (ICLTSET'22) organized by Karpagam Institute of Technology, Coimbatore held on 20<sup>th</sup> & 21<sup>st</sup> of May 2022.

  
Dr.S.Gopinath  
Convener

  
Dr.D.Bhanu  
Vice Principal

  
Dr.P.Manimaran  
Principal



TNSCST Sponsored

## 8<sup>th</sup> International Conference on Latest Trends in Science, Engineering & Technology

### CERTIFICATE OF PARTICIPATION

This is to certify that Mr./Ms./Dr. **Gopinath N** from **Gnanamani College of Technology** has presented a paper titled **Automation Software Testing Using Datadriven and POM Framework** in the TNSCST Sponsored 8<sup>th</sup> International Conference on Latest Trends in Science, Engineering and Technology (ICLTSET'22) organized by Karpagam Institute of Technology, Coimbatore held on 20<sup>th</sup> & 21<sup>st</sup> of May 2022.

  
**Dr.S.Gopinath**  
Convener

  
**Dr.D.Bhanu**  
Vice Principal

  
**Dr.P.Manimaran**  
Principal

TNSCST Sponsored

## 8<sup>th</sup> International Conference on Latest Trends in Science, Engineering & Technology

### CERTIFICATE OF PARTICIPATION

This is to certify that Mr./Ms./Dr. **Manojkumar M** from **Gnanamani College of Technology** has presented a paper titled **Automation Software Testing Using Datadriven and POM Framework** in the TNSCST Sponsored 8<sup>th</sup> International Conference on Latest Trends in Science, Engineering and Technology (ICLTSET'22) organized by Karpagam Institute of Technology, Coimbatore held on 20<sup>th</sup> & 21<sup>st</sup> of May 2022.

  
**Dr.S.Gopinath**  
Convener

  
**Dr.D.Bhanu**  
Vice Principal

  
**Dr.P.Manimaran**  
Principal

