

SOFTWARE VERIFICATION AND VALIDATION

CPSC-542 Project Report Fall 2018

GIRISH KUMAR RAMACHANDRA – 888253630

SURABHI KHADKE – 805676285

Darshan Shah – 888246360

CALIFORNIA STATE UNIVERSITY FULLERTON

AUTOMATION TESTING USING SELENIUM TOOL



ABSTRACT

Software testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. The purpose of **testing** can be quality assurance, verification and validation, or reliability estimation. Although crucial to software quality and widely deployed by programmers and testers, software testing still remains an art, due to limited understanding of the principles of software. The difficulty in software testing stems from the complexity of software: we cannot completely test a program with moderate complexity. Testing is more than just debugging. The purpose of testing can be quality assurance, verification and validation, or reliability estimation. Testing can be used as a generic metric as well. Correctness testing and reliability testing are two major areas of testing. Software testing is a trade-off between budget, time and quality. Here in our project we are using a Automation testing tool, **SELENIUM** and we are making use of Java to write the test script. We will be testing the basic components of a website. Since all the components are not available on a single website, we are making use of 4 websites and in a nutshell, we will be testing more than 25 components and test cases are designed accordingly.

CONTENTS

- INTRODUCTION.....4
- CONCEPT.....6
- SELENIUM AUTOMATION TOOL.....8
- PROJECT THEME.....11
- EXCEPTION HANDLING.....13
- WORKING.....14
- TEST CASES USED.....15
- OUTPUT.....28
- CONCLUSION.....31
- ROLES AND RESPONSIBILITIES.....32

INTRODUCTION

Software Testing is the process of executing a program or system with the intent of finding errors. Or, it involves any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Software is not unlike other physical processes where inputs are received and outputs are produced. Where software differs is in the manner in which it fails. Most physical systems fail in a fixed (and reasonably small) set of ways. By contrast, software can fail in many bizarre ways. Detecting all of the different failure modes for software is generally infeasible.

Software bugs will almost always exist in any software module with moderate size: not because programmers are careless or irresponsible, but because the complexity of software is generally intractable -- and humans have only limited ability to manage complexity. It is also true that for any complex systems, design defects can never be completely ruled out.

Discovering the design defects in software, is equally difficult, for the same reason of complexity. Because software and any digital systems are not continuous, testing boundary values are not sufficient to guarantee correctness. All the possible values need to be tested and verified, but complete testing is infeasible. Exhaustively testing a simple program to add only two integer inputs of 32-bits (yielding 2^{64} distinct test cases) would take hundreds of years, even if tests were performed at a rate of thousands per second. Obviously, for a realistic software module, the complexity can be far beyond the example mentioned here. If inputs from the real world are involved, the problem will get worse, because timing and unpredictable environmental effects and human interactions are all possible input parameters under consideration.

A further complication has to do with the dynamic nature of programs. If a failure occurs during preliminary testing and the code is changed, the software may now work for a test case that it didn't work for previously. But its behavior on pre-error test cases that it passed before can no longer be guaranteed. To account for this possibility, testing should be restarted. The expense of doing this is often prohibitive.

Regardless of the limitations, testing is an integral part in software development. It is broadly deployed in every phase in the software development cycle. Typically, more than 50% percent of the development time is spent in testing. Testing is usually performed for the following purposes:

- **To improve quality.**

As computers and software are used in critical applications, the outcome of a bug can be severe. Bugs can cause huge losses. Bugs in critical systems have caused airplane crashes, allowed space shuttle missions to go awry, halted trading on the stock market, and worse. Bugs can kill. Bugs can cause disasters. The so-called year 2000 (Y2K) bug has given birth to a cottage industry of consultants and programming tools dedicated to making sure the modern world doesn't come to a screeching halt on the first day of the next century. In a computerized embedded world, the quality and reliability of software is a matter of life and death.

Quality means the conformance to the specified design requirement. Being correct, the minimum requirement of quality, means performing as required under specified circumstances. Debugging, a narrow

view of software testing, is performed heavily to find out design defects by the programmer. The imperfection of human nature makes it almost impossible to make a moderately complex program correct the first time. Finding the problems and get them fixed, is the purpose of debugging in programming phase.

- **For Verification & Validation (V&V)**

Just as topic [Verification and Validation](#) indicated, another important purpose of testing is verification and validation (V&V). Testing can serve as metrics. It is heavily used as a tool in the V&V process. Testers can make claims based on interpretations of the testing results, which either the product works under certain situations, or it does not work. We can also compare the quality among different products under the same specification, based on results from the same test.

We cannot test quality directly, but we can test related factors to make quality visible. Quality has three sets of factors -- functionality, engineering, and adaptability. These three sets of factors can be thought of as dimensions in the software quality space. Each dimension may be broken down into its component factors and considerations at successively lower levels of detail. Table 1 illustrates some of the most frequently cited quality considerations.

Functionality	Engineering	Adaptability
correctness	Efficiency	Flexibility
reliability	Testability	Reusability
usability	Documentation	Maintainability
Integrity	Structure	

Table 1 – Software quality Factors

CONCEPT

Correctness testing

Correctness is the minimum requirement of software, the essential purpose of testing. Correctness testing will need some type of oracle, to tell the right behavior from the wrong one. The tester may or may not know the inside details of the software module under test, e.g. control flow, data flow, etc. Therefore, either a white-box point of view or black-box point of view can be taken in testing software. We must note that the black-box and white-box ideas are not limited in correctness testing only. Our project is basically based on Black box Testing.

Black-box testing

- The black-box approach is a testing method in which test data are derived from the specified functional requirements without regard to the final program structure. It is also termed data-driven, input/output driven, or requirements-based testing. Because only the functionality of the software module is of concern, black-box testing also mainly refers to functional testing -- a testing method emphasized on executing the functions and examination of their input and output data. The tester treats the software under test as a black box -- only the inputs, outputs and specification are visible, and the functionality is determined by observing the outputs to corresponding inputs. In testing, various inputs are exercised and the outputs are compared against specification to validate the correctness. All test cases are derived from the specification. No implementation details of the code are considered.
- It is obvious that the more we have covered in the input space, the more problems we will find and therefore we will be more confident about the quality of the software. Ideally we would be tempted to exhaustively test the input space. But as stated above, exhaustively testing the combinations of valid inputs will be impossible for most of the programs, let alone considering invalid inputs, timing, sequence, and resource variables. Combinatorial explosion is the major roadblock in functional testing. To make things worse, we can never be sure whether the specification is either correct or complete. Due to limitations of the language used in the specifications (usually natural language), ambiguity is often inevitable. Even if we use some type of formal or restricted language, we may still fail to write down all the possible cases in the specification. Sometimes, the specification itself becomes an intractable problem: it is not possible to specify precisely every situation that can be encountered using limited words. And people can seldom specify clearly what they want -- they usually can tell whether a prototype is, or is not, what they want after they have been finished. Specification problems contributes approximately 30 percent of all bugs in software

Testing automation

Software testing can be very costly. Automation is a good way to cut down time and cost. Software testing tools and techniques usually suffer from a lack of generic applicability and scalability. The reason is straight-forward. In order to automate the process, we have to have some ways to generate oracles from the specification, and generate test cases to test the target software against the oracles to decide their correctness. Today we still don't have a full-scale system that has achieved this goal. In general, significant amount of human intervention is still needed in testing. The degree of automation remains at the automated test script level.

The problem is lessened in reliability testing and performance testing. In robustness testing, the simple specification and oracle: doesn't crash, doesn't hang suffices. Similar simple metrics can also be used in stress testing.

When to stop testing?

Testing is potentially endless. We cannot test till all the defects are unearthed and removed -- it is simply impossible. At some point, we have to stop testing and ship the software. The question is when.

Realistically, testing is a trade-off between budget, time and quality. It is driven by profit models. The pessimistic, and unfortunately most often used approach is to stop testing whenever some, or any of the allocated resources -- time, budget, or test cases -- are exhausted. The optimistic stopping rule is to stop testing when either reliability meets the requirement, or the benefit from continuing testing cannot justify the testing cost. This will usually require the use of reliability models to evaluate and predict reliability of the software under test. Each evaluation requires repeated running of the following cycle: failure data gathering -- modeling -- prediction. This method does not fit well for ultra-dependable systems, however, because the real field failure data will take too long to accumulate.

SELENIUM AUTOMATION TOOL

Selenium

Selenium is a portable [software-testing framework](#) for [web applications](#). Selenium provides a playback (formerly also recording) tool for authoring tests without the need to learn a test [scripting language](#) (Selenium IDE). It also provides a test [domain-specific language](#) (Selenese) to write tests in a number of popular programming languages, including [C#](#), [Groovy](#), [Java](#), [Perl](#), [PHP](#), [Python](#), [Ruby](#) and [Scala](#). The tests can then run against most modern [web browsers](#). Selenium deploys on [Windows](#), [Linux](#), and [macOS](#) platforms. It is [open-source software](#), released under the [Apache 2.0 license](#): web developers can download and use it without charge.

History

Selenium was originally developed by Jason Huggins in 2004 as an internal tool at [ThoughtWorks](#). Huggins was later joined by other programmers and testers at ThoughtWorks, before Paul Hammant joined the team and steered the development of the second mode of operation that would later become "Selenium Remote Control" (RC). The tool was open sourced that year.

In 2005 Dan Fabulich and Nelson Sproul (with help from Pat Lightbody) made an offer to accept a series of patches that would transform Selenium-RC into what it became best known for. In the same meeting, the steering of Selenium as a project would continue as a committee, with Huggins and Hammant being the ThoughtWorks representatives.

In 2007, Huggins joined Google. Together with others like Jennifer Bevan, he continued with the development and stabilization of Selenium RC. At the same time, Simon Stewart at ThoughtWorks developed a superior browser automation tool called WebDriver. In 2009, after a meeting between the developers at the Google Test Automation Conference, it was decided to merge the two projects, and call the new project Selenium WebDriver, or Selenium 2.0.^[2]

In 2008, Philippe Hanrigou (then at ThoughtWorks) made "Selenium Grid", which provides a hub allowing the running of multiple Selenium tests concurrently on any number of local or remote systems, thus minimizing test execution time. Grid offered, as open source, a similar capability to the internal/private Google cloud for Selenium RC. Pat Lightbody had already made a private cloud for "HostedQA" which he went on to sell to Gomez, Inc.

The name Selenium comes from a joke made by Huggins in an email, mocking a competitor named [Mercury](#), saying that you can cure mercury poisoning by taking selenium supplements. The others that received the email took the name and ran with it.

Selenium IDE

Selenium IDE is a complete [integrated development environment](#) (IDE) for Selenium tests. It is implemented as a [Firefox Add-On](#) and available on [Chrome Store](#) recently, and allows recording, editing, and debugging tests. It was previously known as Selenium Recorder. Selenium-IDE was

originally created by Shinya Kasatani and donated to the Selenium project in 2006. It is little-maintained and is compatible with Selenium RC, which was deprecated.^[4]

Scripts may be automatically recorded and edited manually providing [autocompletion](#) support and the ability to move commands around quickly. Scripts are recorded in *Selenese*, a special test scripting language for Selenium. Selenese provides commands for performing actions in a browser (click a link, select an option), and for retrieving data from the resulting pages.

The 2.x version of the Selenium IDE for Firefox stopped working^[5] after the Firefox 55 upgrade and has been replaced by Selenium IDE 3.x.^[6] However users can run the older Selenium IDE on some older Firefox versions (pre-Firefox 55) or try other alternative solutions^[7]

Selenium client API

As an alternative to writing tests in Selenese, tests can also be written in various programming languages. These tests then communicate with Selenium by calling methods in the Selenium Client API. Selenium currently provides client APIs for [Java](#), [C#](#), [Ruby](#), [JavaScript](#) and [Python](#).

With Selenium 2, a new Client API was introduced (with *WebDriver* as its central component). However, the old API (using class *Selenium*) is still supported.

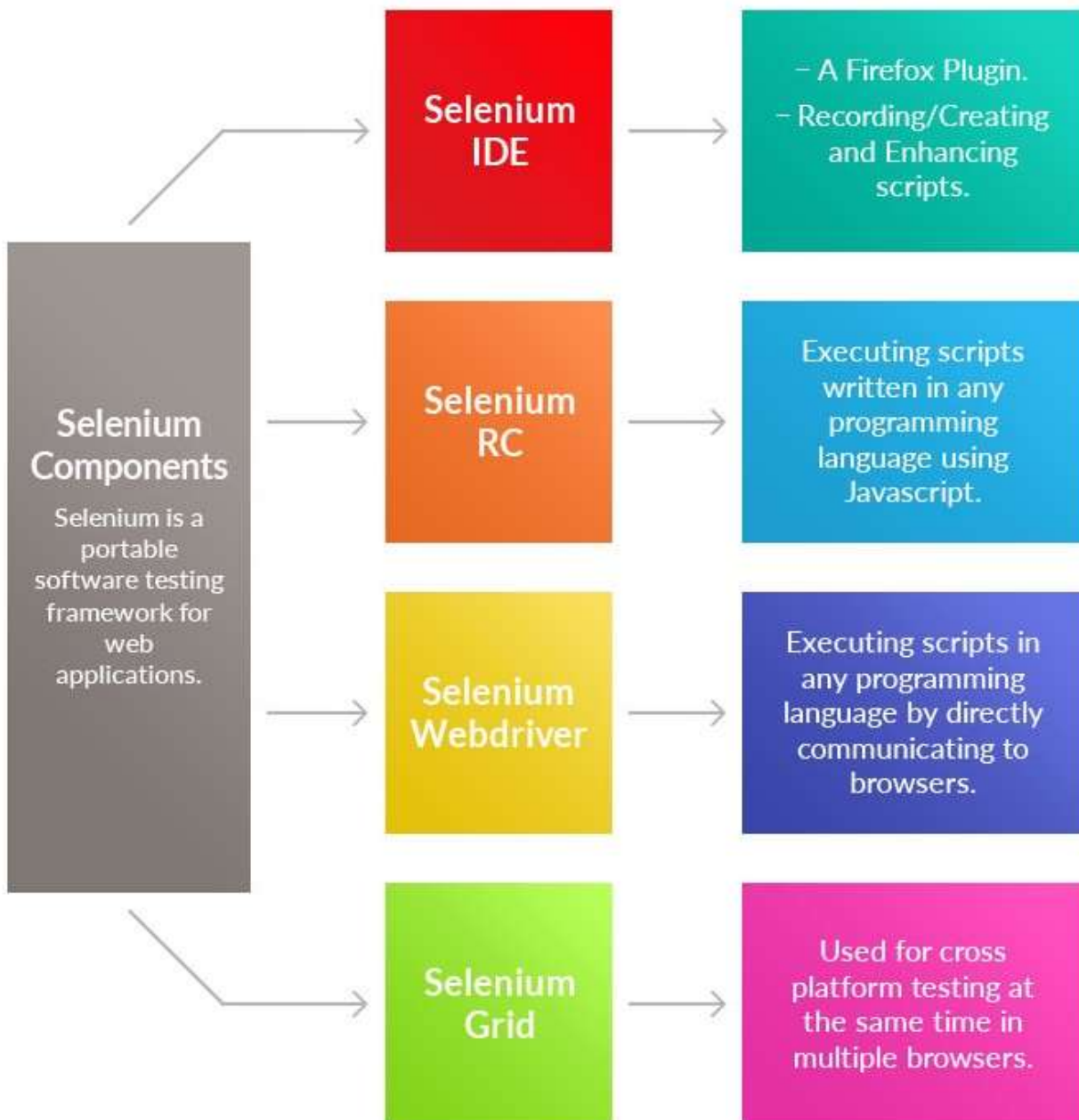
Selenium WebDriver

Selenium WebDriver is the successor to Selenium RC. Selenium WebDriver accepts commands (sent in Selenese, or via a Client API) and sends them to a browser. This is implemented through a browser-specific browser driver, which sends commands to a browser and retrieves results. Most browser drivers actually launch and access a browser application (such as [Firefox](#), [Chrome](#), [Internet Explorer](#), [Safari](#), or [Microsoft Edge](#)); there is also an [HtmlUnit](#) browser driver, which simulates a browser using the headless browser HtmlUnit.

Unlike in Selenium 1, where the Selenium server was necessary to run tests, Selenium WebDriver does not need a special server to execute tests. Instead, the WebDriver directly starts a browser instance and controls it. However, Selenium Grid can be used with WebDriver to execute tests on remote systems (see below). Where possible, WebDriver uses native operating system level functionality rather than browser-based JavaScript commands to drive the browser. This bypasses problems with subtle differences between native and JavaScript commands, including security restrictions.^[8]

In practice, this means that the Selenium 2.0 API has significantly fewer calls than does the Selenium 1.0 API. Where Selenium 1.0 attempted to provide a rich interface for many different browser operations, Selenium 2.0 aims to provide a basic set of building blocks from which developers can create their own [Domain Specific Language](#). One such DSL already exists: the [Watir](#) project in the Ruby language has a rich history of good design. Watir-webdriver implements the Watir API as a wrapper for Selenium-Webdriver in Ruby. Watir-webdriver is created entirely automatically, based on the WebDriver specification and the HTML specification.

As of early 2012, Simon Stewart (inventor of WebDriver), who was then with Google and now with Facebook, and David Burns of Mozilla were negotiating with the [W3C](#) to make WebDriver an internet standard. In July 2012, the working draft was released and the recommendation followed in June 2018.^[9] Selenium-Webdriver (Selenium 2.0) is fully implemented and supported in [Python](#), [Ruby](#), [Java](#), and [C#](#).



PROJECT THEME

To test the basic components of a website by Making use of the Automation Testing tool SELENIUM. We have tested more than 25 components, and since all the components were not available in a single website, we have made use of 4 different websites, Accordingly Test cases are written for each component, finally a Test script which can be played through Eclipse IDE.

Websites Used

www.ultimateQA.com

www.the-internet.herokuapp.com

www.demoQA.com

www.compendiumdev.co.uk

Web Components tested

1. www.ultimateQA.com

- **Button Click**
- **Search Option**
- **Toggling tabs**
- **Email alert**

2. www.the-internet.herokuapp.com

- **Checkbox**
- **File download testing**
- **Forgot password testing**
- **Drop down testing**
- **Menu form Authentication**
- **Drag and drop**
- **Hover over actions**
- **Dynamic Controls**
- **Enable Disable**
- **Dynamic Loading**
- **Notification Message**

3. www.demoQA.com

- **bookmark comment and leave a reply**

4. www.compendiumdev.co.uk

- Hire me review and enquiry
- E training review and FAQ review
- Contact us page
- Books review and Downloading samples

5. Other components tested for each website:-

- website loading issue
- current webpage title retrieval
- current webpage URL retrieval
- current page source retrieval
- clearing cookies

EXCEPTION HANDLING

Try/Catch: A method catches an exception using a combination the try and catch keywords. **Try** is the start of the block and **Catch** is at the end of try block to handle the exceptions.

```
1  try
2
3  {
4
5  // Some code
6
7  }catch(Exception e){
8
9  // Code for Handling the exception
10
11 }
```

WORKING

- I. Open Eclipse IDE.
- II. Create a java Project.
- III. Create a java Package.
- IV. Create a java Class.
- V. Import Selenium Driver and sync it to the package
- VI. Import Chrome Web driver for Selenium and sync it to the package.
- VII. Code the Test Script.
- VIII. Run the Script through Eclipse IDE,
- IX. Verify Test Cases.
- X. Handle Exceptions.
- XI. Document it manually.

TEST CASES USED

- Button Click

```
//Button Testing
try
{
    d.findElement(By.xpath("//a[@cass='et_pb_button et_pb_button_0 et_pb_bg_layout_light']")).click();
    Thread.sleep(2000);
    System.out.println("Button testing Success");
}
catch(Exception e)
{
    System.out.println("Button testing Failed");
}
```

- Search Option

```
//Search option Testing
try
{
    d.findElement(By.xpath("//div[@class='et_pb_module et_pb_sidebar_0 et_pb_widget_area et_pb_bg_layout_light clearfix et_pb_widget_area_left']//div[@id='search-2']//form[@id='search-2']"));
    d.findElement(By.xpath("//div[@class='et_pb_module et_pb_sidebar_0 et_pb_widget_area et_pb_bg_layout_light clearfix et_pb_widget_area_left']//div[@id='search-2']//form[@id='search-2']"));
    Thread.sleep(2000);
    d.navigate().back();
    d.navigate().refresh();
    System.out.println("Search option testing success");
}
catch(Exception e)
{
    System.out.println("Search option testing Failed");
}
```

- Toggling tabs

```
//Toggle Testing
try
{
    d.findElement(By.xpath("//h5[contains(@class,'et_pb_toggle_title')]")).click();
    Thread.sleep(2000);
    System.out.println("Toggle button testing success");
}
catch(Exception e)
{
    System.out.println("Toggle button testing Failed");
}
```

- **Email alert**

```
//Email Alert function Testing
try
{
d.findElement(By.xpath("//a[contains(@title,'Click to email this to a friend')]")).click();
d.findElement(By.xpath("//input[@id='target_email']")).sendKeys("xxx@gmail.com");
d.findElement(By.xpath("//input[@id='source_name']")).sendKeys("xxx");
d.findElement(By.xpath("//input[@id='source_email']")).sendKeys("xxx@gmail.com");
d.findElement(By.xpath("//input[contains(@value,'Send Email')]")).click();
Thread.sleep(2000);
System.out.println("Email alert function testing success");
}
catch(Exception e)
{
System.out.println("Email alert function testing Failed");
}
```

- **Checkbox**

```
//Checkbox Testing
try
{
d.findElement(By.xpath("//a[contains(text(),'Checkboxes')]")).click();
d.findElement(By.xpath("//form[@id='checkboxes']/input[1]")).click();
Thread.sleep(1000);
d.findElement(By.xpath("//form[@id='checkboxes']/input[2]")).click();
Thread.sleep(1000);
d.navigate().back();
Thread.sleep(2000);
System.out.println("Checkbox testing success");
}
catch(Exception e)
{
System.out.println("Checkbox testing Failed");
}
```

- File download testing

```
try
{
//File Download Testing
d.findElement(By.xpath("//a[@href='/download']")).click();
d.findElement(By.xpath("/html[1]/body[1]/div[2]/div[1]/div[1]/a[1]")).click();
Thread.sleep(2000);
d.navigate().back();
System.out.println("File download testing success");
}
catch(Exception e)
{
System.out.println("File download testing Failed");
}
```

- Forgot password testing

```
//Forgot Password Testing
try
{
d.findElement(By.xpath("//a[contains(text(),'Forgot Password')]")).click();
d.findElement(By.xpath("//input[@id='email']")).sendKeys("xxx@gmail.com");
d.findElement(By.xpath("//i[@class='icon-2x icon-signin']")).click();
Thread.sleep(2000);
d.navigate().back();
d.navigate().back();
System.out.println("Forgot password component testing success ");
}
catch(Exception e)
{
System.out.println("Forgot password component testing failed ");
}
```

- **Drop down testing**

```
//Drop Down Testing
try
{
d.findElement(By.xpath("/html[1]/body[1]/div[2]/div[1]/ul[1]/li[9]/a[1]")).click();
Thread.sleep(1000);
d.findElement(By.xpath("/html[1]/body[1]/div[2]/div[1]/div[1]/select[1]")).click();
Thread.sleep(2000);
d.navigate().back();
Thread.sleep(2000);
System.out.println("Drop down testing success");
}
catch(Exception e)
{
System.out.println("Drop down testing failed");
}
```

- **Menu form Authentication**

```
//Form Authentication Testing
try
{
d.findElement(By.xpath("//a[contains(text(),'Form Authentication')]")).click();
d.findElement(By.xpath("/html[1]/body[1]/div[2]/div[1]/div[1]/form[1]/div[1]/div[1]/input[1]")).sendKeys("tomsmith");
d.findElement(By.xpath("//input[@id='password']")).sendKeys("SuperSecretPassword!");
d.findElement(By.xpath("//i[@class='fa fa-2x fa-sign-in']")).click();
Thread.sleep(1000);
d.navigate().back();
d.navigate().back();
Thread.sleep(2000);
System.out.println("Form Authentication testing success");
}
catch(Exception e)
{
System.out.println("Form Authentication testing Failed");
}
```

- Drag and drop

```
//Drag and drop Testing
try
{
d.findElement(By.xpath("//a[contains(text(),'Drag and Drop')]")).click();
WebElement From= d.findElement(By.xpath("//div[@id='column-a']"));
WebElement To=d.findElement(By.xpath("//div[@id='column-b']"));
Actions act=new Actions(d);
act.dragAndDrop(From, To).build().perform();
Thread.sleep(1000);
d.navigate().back();
Thread.sleep(2000);
System.out.println("Drag and drop component testing success");
}
catch(Exception e)
{
System.out.println("Drag and drop component testing Failed");
}
```

- Hover over actions

```
//Hover over an Element Testing
try
{
d.findElement(By.xpath("//a[contains(text(),'Hovers')]")).click();
Actions action = new Actions(d);
WebElement we = d.findElement(By.xpath("//div[@class='example']/div[1]/img[1]"));
action.moveToElement(we).build().perform();
Thread.sleep(1000);
d.navigate().back();
Thread.sleep(2000);
System.out.println("Hover over component testing success");
}
catch(Exception e)
{
System.out.println("Hover over component testing Failed");
}
```

- Dynamic Controls

```
//Dynamic Controls Testing
try
{
d.findElement(By.xpath("//a[contains(text(),'Dynamic Controls')]")).click();
d.findElement(By.xpath("//input[@type='checkbox']")).click();
d.findElement(By.xpath("//button[contains(text(),'Remove')]")).click();
Thread.sleep(3000);
System.out.println("Dynamic control testing success");
}
catch(Exception e)
{
System.out.println("Dynamic control testing Failed");
}
```

- Enable Disable

```
//Enable-Disable Testing
try
{
d.findElement(By.xpath("//button[contains(text(),'Enable')]")).click();
Thread.sleep(6000);
d.navigate().back();
Thread.sleep(2000);
System.out.println("Enable-Disable testing success");
}
catch(Exception e)
{
System.out.println("Enable-Disable testing Failed");
}
```

- **Dynamic Loading**

```
//Dynamic Loading Testing
try
{
d.findElement(By.xpath("//a[contains(text(),'Dynamic Loading')]")).click();
d.findElement(By.xpath("//a[contains(text(),'Example 1: Element on page that is hidden')]")).click();
d.findElement(By.xpath("//button[contains(text(),'Start')]")).click();
Thread.sleep(6000);
d.navigate().back();
d.navigate().back();
Thread.sleep(2000);
System.out.println("Dynamic Loading testing Success");
}
catch(Exception e)
{
System.out.println("Dynamic Loading testing Failed");
}
```

- **Notification Message**

```
//Notification Message Testing
try
{
d.findElement(By.xpath("//a[contains(text(),'Notification Messages')]")).click();
Thread.sleep(2000);
d.findElement(By.xpath("//a[contains(text(),'Click here')]")).click();
Thread.sleep(1000);
d.navigate().back();
d.navigate().back();
Thread.sleep(2000);
System.out.println("Notification Message testing Success");
}
catch(Exception e)
{
System.out.println("Notification Message testing Failed");
}
```

- **bookmark comment and leave a reply**

```
//Bookmark ,Comment and leave a reply Testing
try
{
d.findElement(By.xpath("//a[contains(text(),'permalink')]")).click();
d.findElement(By.xpath("//input[@id='author']")).sendKeys("Girish kumar");
d.findElement(By.xpath("//input[@id='email']")).sendKeys("girish@gmail.com");
d.findElement(By.xpath("//input[@id='url']")).sendKeys("http://www.girish.com");
d.findElement(By.xpath("//textarea[@id='comment']")).sendKeys("thank you for the Article");
d.findElement(By.xpath("//button[@id='submit']")).click();
Thread.sleep(2000);
System.out.println("Bookmark,comment and leave a reply testing success");
}
catch(Exception e)
{
System.out.println("Bookmark,comment and leave a reply testing Failed");
}
```

- **Hire me review and enquiry**

```
// Hire me review and enquiry
try
{
d.findElement(By.xpath("/html[1]/body[1]/div[1]/div[1]/div[3]/div[1]/ul[1]/li[1]/a[1]")).click();
d.findElement(By.xpath("//input[@id='youremail']")).sendKeys("xxx@gmail.com");
d.findElement(By.xpath("//input[@id='yourname']")).sendKeys("Mr.X");
d.findElement(By.xpath("//textarea[@id='yourmessage']")).sendKeys("Hello ,i am looking for a job in selenium testing domain");
d.findElement(By.xpath("//input[@value='Send Consultancy Request']")).click();
Thread.sleep(2000);
System.out.println("Hire me review and enquiry testing Success");
}
catch(Exception e)
{
System.out.println("Hire me review and enquiry testing Failed");
}
```

- **E training review and FAQ review**

```
//E-training review and FAQ Review
try
{
d.findElement(By.xpath("/html[1]/body[1]/div[1]/div[1]/div[3]/div[1]/ul[1]/li[2]/a[1]")).click();
d.findElement(By.xpath("//a[contains(text(),'Selenium WebDriver With Java')]")).click();
Thread.sleep(2000);
d.findElement(By.xpath("/html[1]/body[1]/div[1]/div[2]/p[20]/a[1]")).click();
Thread.sleep(2000);
System.out.println("E-training review and FAQ review testing Success");
}
catch(Exception e)
{
System.out.println("E-training review and FAQ review testing Failed");
}
```

- **Contact us page**

```
//contact us Testing
try
{
d.findElement(By.xpath("/html[1]/body[1]/div[1]/div[1]/div[3]/div[1]/ul[1]/li[6]/a[1]")).click();
Thread.sleep(2000);
d.findElement(By.xpath("//input[@id='youremail']")).sendKeys("xxx@gmail.com");
d.findElement(By.xpath("//input[@id='yourname']")).sendKeys("xxx");
d.findElement(By.xpath("//textarea[@id='yourmessage']")).sendKeys("i wanted to enquire about selenium course");
d.findElement(By.xpath("/html[1]/body[1]/div[1]/div[2]/div[2]/form[1]/div[1]/input[2]")).click();
Thread.sleep(3000);
System.out.println("Contact us testing success");
}
catch(Exception e)
{
System.out.println("Contact us testing Failed");
}
```

- **Books review and Downloading samples**

```
//Books Review and download sample
try
{
d.findElement(By.xpath("/html[1]/body[1]/div[1]/div[1]/div[3]/div[1]/ul[1]/li[3]/a[1]")).click();
d.findElement(By.xpath("//a[@class='button blue']")).click();
d.findElement(By.xpath("//a[@class='button']")).click();
Thread.sleep(5000);
d.findElement(By.xpath("/html[1]/body[1]/div[1]/div[1]/div[3]/div[1]/ul[1]/li[3]/a[1]")).click();
d.findElement(By.xpath("//a[@class='button red']")).click();
d.findElement(By.xpath("/html[1]/body[1]/div[1]/div[2]/div[1]/p[7]/a[1]")).click();
Thread.sleep(5000);
d.navigate().back();
System.out.println("Books review and download sample testing Success");
}
catch(Exception e)
{
System.out.println("Books review and download sample testing Failed");
}
```

- Website Loading Issue

```
//Ultimate QA website components Testing
try
{
d.get("https://www.ultimateqa.com/complicated-page/");
Thread.sleep(2000);
System.out.println("Ultimate QA website loading Success");
}
catch(Exception e)
{
System.out.println("Ultimate QA website loading Failed");
}
```

```
//herokuapp website Components Testing
try
{
d.navigate().to("http://the-internet.herokuapp.com/");
Thread.sleep(2000);
System.out.println("herokuapp website loading success");
}
catch(Exception e)
{
System.out.println("herokuapp website loading Failed");
}
```

```
//demoqa website components Testing
```

```
try
{
d.navigate().to("http://demoqa.com/")(http://demoqa.com/");
Thread.sleep(2000);
d.findElement(By.xpath("//a[@class='btn btn-primary']")).click();
Thread.sleep(2000);
System.out.println("demoqa website loading success");
}
catch(Exception e)
{
System.out.println("demoqa website loading Failed");
}
```

```
// Compendium website Testing
```

```
try
{
d.navigate().to("https://compendiumdev.co.uk/default.php");
Thread.sleep(2000);
System.out.println("Compendium Website Loading Success");
}
catch(Exception e)
{
System.out.println("Compendium Website Loading Failed");
}
```

- current webpage title retrieval

```
//Getting current title of the page
try
{
String title1=d.getTitle();
System.out.println("Current title Retrival Success");
System.out.println(title1);
}
catch(Exception e)
{
System.out.println("Current title Retrival Failed");
}
```

- current webpage URL retrieval

```
//URL of current page
try
{
String Url1 = d.getCurrentUrl();
System.out.println("Current Url Retrival Success");
System.out.println(Url1);
}
catch(Exception e)
{
System.out.println("Current Url Retrival Failed");
}
```

- current page source retrieval

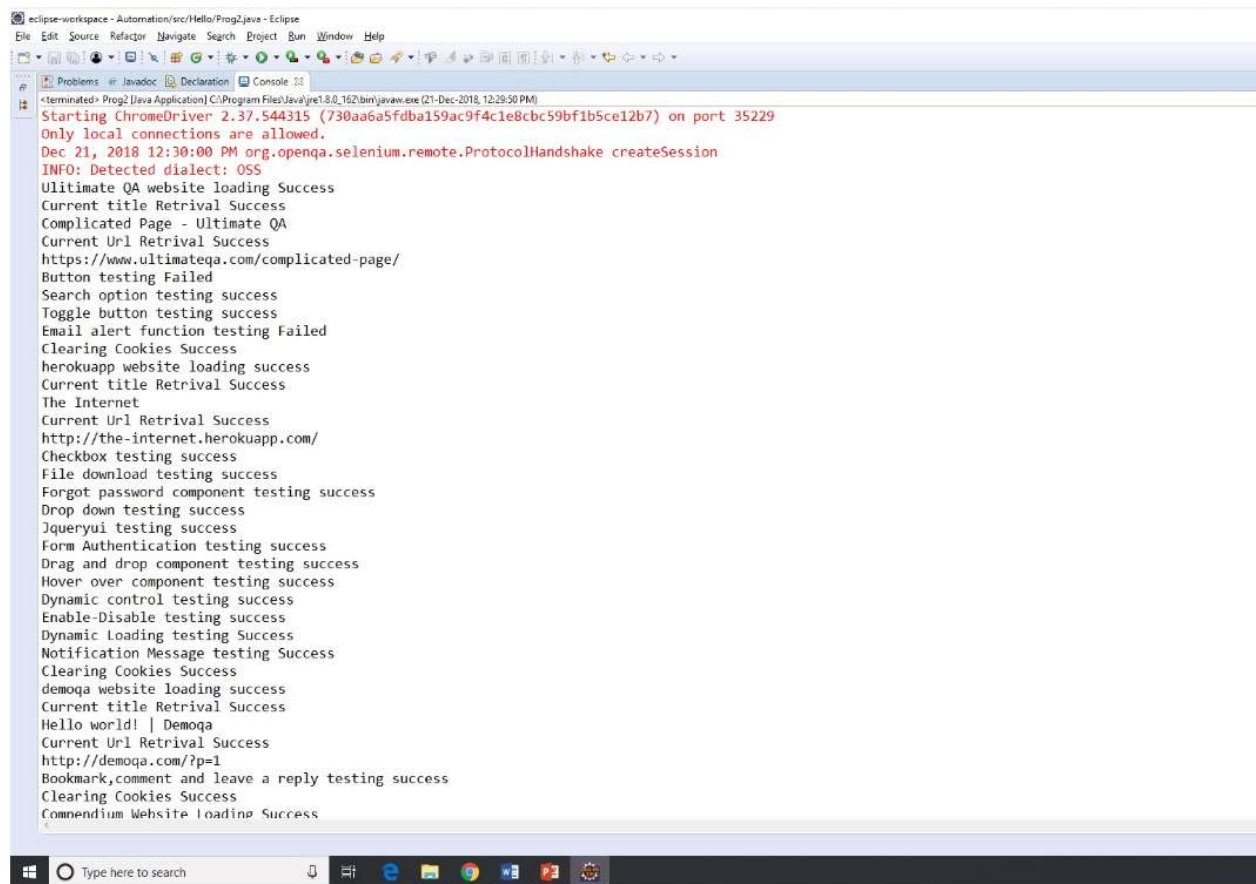
```
/*Page source of current web page
try
{
String Pagesrc1 = d.getPageSource();
System.out.println("Current page source Retrival success");
System.out.println(Pagesrc1);
}
catch(Exception e)
{
System.out.println("Current page source Retrival Failed");
} */
```

- clearing cookies

```
//Clear Cookies of this website before Navigating
try
{
d.manage().deleteAllCookies();
System.out.println("Clearing Cookies Success");
}
catch(Exception e)
{
System.out.println("Clearing Cookies Failed");
}
```

OUTPUT

- All the Test Cases are passed successfully after trial run
- Last test case has been failed deliberately in order to create an exception



The screenshot shows the Eclipse IDE's console window with the following output:

```
<terminated> Prog2 [Java Application] C:\Program Files\Java\jre1.8.0_162\bin\javaw.exe (21-Dec-2018, 12:29:50 PM)
Starting ChromeDriver 2.37.544315 (730aa6a5fdb5159ac9f4c1e8cbc59bf1b5ce12b7) on port 35229
Only local connections are allowed.
Dec 21, 2018 12:30:00 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: OSS
Ultimate QA website loading Success
Current title Retrieval Success
Complicated Page - Ultimate QA
Current Url Retrieval Success
https://www.ultimateqa.com/complicated-page/
Button testing Failed
Search option testing success
Toggle button testing success
Email alert function testing Failed
Clearing Cookies Success
herokuapp website loading success
Current title Retrieval Success
The Internet
Current Url Retrieval Success
http://the-internet.herokuapp.com/
Checkbox testing success
File download testing success
Forgot password component testing success
Drop down testing success
Jqueryui testing success
Form Authentication testing success
Drag and drop component testing success
Hover over component testing success
Dynamic control testing success
Enable-Disable testing success
Dynamic Loading testing Success
Notification Message testing Success
Clearing Cookies Success
demoqa website loading success
Current title Retrieval Success
Hello world! | Demoqa
Current Url Retrieval Success
http://demoqa.com/?p=1
Bookmark,comment and leave a reply testing success
Clearing Cookies Success
Commendium Website loading Success
```

```
eclipse-workspace - Automation/src/Hello/Prog2.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help

Problems Javadoc Declaration Console 33
<terminated> Prog2 [Java Application] C:\Program Files\Java\jre1.8.0_162\bin\java.exe (21-Dec-2018, 12:29:50 PM)

Hello world! | Demoqa
Current Url Retrieval Success
http://demoqa.com/?p=1
Bookmark,comment and leave a reply testing success
Clearing Cookies Success
Compendium Website Loading Success
Current title Retrieval Success
Software Testing Consultancy, Books and Online Training
Current Url Retrieval Success
https://compendiumdev.co.uk/default.php
Hire me review and enquiry testing Success
E-training review and FAQ review testing Success
Contact us testing success
Books review and download sample testing Failed
Exception in thread "main" org.openqa.selenium.WebDriverException: java.net.ConnectException: Failed to connect to localhost/127.0.0.1:35229
Build info: version: '3.12.0', revision: '7c6e0b3', time: '2018-05-08T15:15:08.936Z'
System info: host: 'GIRISH-LAPTOP', ip: '192.168.43.156', os.name: 'Windows 10', os.arch: 'amd64', os.version: '10.0', java.version: '1.8.0_162'
Driver info: driver.version: RemoteWebDriver
    at org.openqa.selenium.remote.service.DriverCommandExecutor.execute(DriverCommandExecutor.java:92)
    at org.openqa.selenium.remote.RemoteWebDriver.execute(RemoteWebDriver.java:543)
    at org.openqa.selenium.remote.RemoteWebDriver.execute(RemoteWebDriver.java:600)
    at org.openqa.selenium.remote.RemoteWebDriver.close(RemoteWebDriver.java:433)
    at Hello.Prog2.main(Prog2.java:593)
Caused by: java.net.ConnectException: Failed to connect to localhost/127.0.0.1:35229
    at okhttp3.internal.connection.RealConnection.connectSocket(RealConnection.java:240)
    at okhttp3.internal.connection.RealConnection.connect(RealConnection.java:158)
    at okhttp3.internal.connection.StreamAllocation.findConnection(StreamAllocation.java:256)
    at okhttp3.internal.connection.StreamAllocation.findHealthyConnection(StreamAllocation.java:134)
    at okhttp3.internal.connection.StreamAllocation.newStream(StreamAllocation.java:113)
    at okhttp3.internal.connection.ConnectInterceptor.intercept(ConnectInterceptor.java:42)
    at okhttp3.internal.http.RealInterceptorChain.proceed(RealInterceptorChain.java:147)
    at okhttp3.internal.http.RealInterceptorChain.proceed(RealInterceptorChain.java:121)
    at okhttp3.internal.cache.CacheInterceptor.intercept(CacheInterceptor.java:93)
    at okhttp3.internal.http.RealInterceptorChain.proceed(RealInterceptorChain.java:147)
    at okhttp3.internal.http.RealInterceptorChain.proceed(RealInterceptorChain.java:121)
    at okhttp3.internal.http.BridgeInterceptor.intercept(BridgeInterceptor.java:93)
    at okhttp3.internal.http.RealInterceptorChain.proceed(RealInterceptorChain.java:147)
    at okhttp3.internal.http.RetryAndFollowUpInterceptor.intercept(RetryAndFollowUpInterceptor.java:125)
    at okhttp3.internal.http.RealInterceptorChain.proceed(RealInterceptorChain.java:147)
    at okhttp3.internal.http.RealInterceptorChain.proceed(RealInterceptorChain.java:121)
```

```
eclipse-workspace - Automation/src/Hello/Prog2.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help

Problems JavaDoc Declaration Console
<terminated> Prog2 [Java Application] C:\Program Files\Java\jre1.8.0_162\bin\javaw.exe (21-Dec-2018, 12:29:50 PM)
  at org.openqa.selenium.remote.RemoteWebDriver.execute(RemoteWebDriver.java:543)
  at org.openqa.selenium.remote.RemoteWebDriver.execute(RemoteWebDriver.java:600)
  at org.openqa.selenium.remote.RemoteWebDriver.close(RemoteWebDriver.java:433)
  at Hello.Prog2.main(Prog2.java:593)
Caused by: java.net.ConnectException: Failed to connect to localhost/127.0.0.1:35229
  at okhttp3.internal.connection.RealConnection.connectSocket(RealConnection.java:240)
  at okhttp3.internal.connection.RealConnection.connect(RealConnection.java:158)
  at okhttp3.internal.connection.StreamAllocation.findConnection(StreamAllocation.java:256)
  at okhttp3.internal.connection.StreamAllocation.findHealthyConnection(StreamAllocation.java:134)
  at okhttp3.internal.connection.StreamAllocation.newStream(StreamAllocation.java:113)
  at okhttp3.internal.connection.ConnectInterceptor.intercept(ConnectInterceptor.java:42)
  at okhttp3.internal.http.RealInterceptorChain.proceed(RealInterceptorChain.java:147)
  at okhttp3.internal.http.RealInterceptorChain.proceed(RealInterceptorChain.java:121)
  at okhttp3.internal.cache.CacheInterceptor.intercept(CacheInterceptor.java:93)
  at okhttp3.internal.http.RealInterceptorChain.proceed(RealInterceptorChain.java:147)
  at okhttp3.internal.http.RealInterceptorChain.proceed(RealInterceptorChain.java:121)
  at okhttp3.internal.http.RealInterceptorChain.proceed(RealInterceptorChain.java:121)
  at okhttp3.internal.http.BridgeInterceptor.intercept(BridgeInterceptor.java:93)
  at okhttp3.internal.http.RealInterceptorChain.proceed(RealInterceptorChain.java:147)
  at okhttp3.internal.http.RetryAndFollowUpInterceptor.intercept(RetryAndFollowUpInterceptor.java:125)
  at okhttp3.internal.http.RealInterceptorChain.proceed(RealInterceptorChain.java:147)
  at okhttp3.internal.http.RealInterceptorChain.proceed(RealInterceptorChain.java:121)
  at okhttp3.RealCall.getResponseWithInterceptorChain(RealCall.java:200)
  at okhttp3.RealCall.execute(RealCall.java:77)
  at org.openqa.selenium.remote.internal.OkHttpClient.execute(OkHttpClient.java:105)
  at org.openqa.selenium.remote.HttpCommandExecutor.execute(HttpCommandExecutor.java:155)
  at org.openqa.selenium.remote.service.DriverCommandExecutor.execute(DriverCommandExecutor.java:83)
  ... 4 more
Caused by: java.net.ConnectException: Connection refused: connect
  at java.net.DualStackPlainSocketImpl.waitForConnect(Native Method)
  at java.net.DualStackPlainSocketImpl.socketConnect(Unknown Source)
  at java.net.AbstractPlainSocketImpl.doConnect(Unknown Source)
  at java.net.AbstractPlainSocketImpl.connectToAddress(Unknown Source)
  at java.net.AbstractPlainSocketImpl.connect(Unknown Source)
  at java.net.PlainSocketImpl.connect(Unknown Source)
  at java.net.SocksSocketImpl.connect(Unknown Source)
  at java.net.Socket.connect(Unknown Source)
  at okhttp3.internal.platform.Platform.connectSocket(Platform.java:125)
  at okhttp3.internal.connection.RealConnection.connectSocket(RealConnection.java:238)
  ... 24 more
```


CONCLUSION

- Software testing is an art. Most of the testing methods and practices are not very different from 20 years ago. It is nowhere near maturity, although there are many tools and techniques available to use. Good testing also requires a tester's creativity, experience and intuition, together with proper techniques.
- Testing is more than just debugging. Testing is not only used to locate defects and correct them. It is also used in validation, verification process, and reliability measurement.
- Testing is expensive. Automation is a good way to cut down cost and time. Testing efficiency and effectiveness is the criteria for coverage-based testing techniques.
- Complete testing is infeasible. Complexity is the root of the problem. At some point, software testing has to be stopped and product has to be shipped. The stopping time can be decided by the trade-off of time and budget. Or if the reliability estimate of the software product meets requirement.
- Testing may not be the most effective method to improve software quality. Alternative methods, such as inspection, and clean-room engineering, may be even better.
- Automation testing is a best way to fulfill most of the testing goals with effective resources and time.
- But be careful before using the automation tool that fulfills the requirement of the application because no any tool can fulfill 100% requirement.
- The right selection of automation tool, testing process, and team, is very important for automation to be successful.
- Manual and automation methods go hand-in-hand for successful testing.

ROLES & RESPONSIBILITIES

➤ **GIRISH KUMAR RAMACHANDRA**

- Installation of Eclipse IDE
- Working with XPath
- Exception handling Design
- Project Presentation and Report Preparation
- Integration testing
- Test cases designed
 - ✓ Button
 - ✓ Search option
 - ✓ Toggle
 - ✓ Email alert
 - ✓ JQueryui menu
 - ✓ Drag and drop
 - ✓ Hover over
 - ✓ Website loading issue
 - ✓ Current page source retrieval
 - ✓ Book review and download samples

➤ **SURABHI KHADKE**

- Deploying Selenium Driver
- Spying web Elements
- Unit Testing
- Project Presentation and Report Preparation
- Test cases designed
 - ✓ Checkbox
 - ✓ File download
 - ✓ Forgot password
 - ✓ Dropdown
 - ✓ Form Authentication
 - ✓ Clearing cookies
 - ✓ Current page URL retrieval
 - ✓ Hire me review and enquiry

➤ **DARSHAN SHAH**

- Deploying Chrome web driver
- Unit testing
- Handling ChroPath
- Project Presentation and Report Preparation
- Test cases designed
 - ✓ Dynamic controls
 - ✓ Enable disable
 - ✓ Dynamic loading
 - ✓ Notification message
 - ✓ Bookmark comment and leave a reply
 - ✓ Contact us
 - ✓ Current page title retrieval
 - ✓ E-training review and FAQ review