TEST AUTOMATION USING SELENIUM

A Project-II Report

Submitted in partial fulfilment of requirement of the

Degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE & ENGINEERING

BY
Shubham Gupta
(EN16CS301254)

Under the Guidance of
Prof. (Dr.) Ruchi Patel
Mr. Jitesh Agrawal

Department of Computer Science & Engineering
Faculty of Engineering
MEDI-CAPS UNIVERSITY, INDORE- 453331
May, 2020

TEST AUTOMATION USING SELENIUM

A Project-II Report

Submitted in partial fulfilment of requirement of the

Degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE & ENGINEERING

BY
Shubham Gupta
(EN16CS301254)

Under the Guidance of
Prof. (Dr.) Ruchi Patel
Mr. Jitesh Agrawal

Department of Computer Science & Engineering
Faculty of Engineering
MEDI-CAPS UNIVERSITY, INDORE- 453331
May, 2020

Report Approval

The project work **"Test Automation using Selenium"** is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the Degree for which it has been submitted.

It is to be understood that by this approval the undersigned do not endorse or approved any statement made, opinion expressed, or conclusion drawn there in; but approve the "Project Report" only for the purpose for which it has been submitted.

Internal Examiner

Name:

Designation

Affiliation

External Examiner

Name:

Designation

Affiliation

Declaration

I hereby declare that the project entitled **"Test Automation Using Selenium"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'Computer science & Engineering' completed under the supervision of **Prof. (Dr.) Ruchi Patel, Assistant Professor, Department of Computer Science and Engineering**, Faculty of Engineering, Medi-Caps University, Indore and **Mr. Jitesh Agrawal, Project Manager,** Radiance Technologies is an authentic work.

Further, I declare that the content of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for the award of any degree or diploma.

Date: _____ **Shubham Gupta (EN16CS301254)**

Certificate

We, **Prof. (Dr.) Ruchi Patel** and **Mr. Jitesh Agrawal** certify that the project entitled **"Test Automation Using Selenium"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology by **Shubham Gupta** is the record carried out by him under our guidance and that the work has not formed the basis of award of any other degree elsewhere.

Prof. (Dr.) Ruchi Patel	Mr. Jitesh Agrawal
Computer Science & Engineering Department	Project Manager
Medi-Caps University, Indore	Radiance Technologies

Dr. Suresh Jain

Head of the Department

Computer Science & Engineering

Medi-Caps University, Indore

Acknowledgements

I would like to express my deepest gratitude to Honorable Chancellor, **Shri R C Mittal,** who has provided me with every facility to successfully carry out this project, and my profound indebtedness to **Prof. (Dr.) Sunil K Somani,** Vice Chancellor, Medi-Caps University, whose unfailing support and enthusiasm has always boosted up my morale. I also thank **Prof. (Dr.) D K Panda,** Dean, Faculty of Engineering, Medi-Caps University, for giving me a chance to work on this project. I would also like to thank my Head of the Department **Prof. (Dr.) Suresh Jain** for his/her continuous encouragement for betterment of the project.

I express my heartfelt gratitude to my Internal Guide**, Prof. (Dr). Ruchi Patel**, Assistant Professor, Department of Computer Science and Engineering, Medi-Caps University, without whose continuous help and support, this project would ever have reached to the completion.

I would also like to thank my External Guides**, Mr. Jitesh Agrawal** and **Mr. Sulabh Shrivastava,** Radiance Technologies who extended their kind support and help towards the completion of this project.

It is their help and support, due to which we became able to complete the design and technical report.

Without their support this report would not have been possible.

Shubham Gupta (EN16CS301254)
B.Tech. IV Year
Department of Computer Science & Engineering
Faculty of Engineering
Medi-Caps University, Indore

Abstract

Software testing is an integral of any software development model. Any software that is developed by a company for quality assurance purposes and also to make sure that all the requests made by the customer are fulfilled.

Radiance Technologies is a US based technology firm which provides its services to various other companies by means of developing custom software.

During the period of my internship I was associated with radiance technologies under the guidance of Mr. Jitesh Agrawal and Mr. Sulabh Shrivastava and worked on testing side of as many as three software developed by radiance technologies for its client "CHEORS- Complete Health Economic Output Research Solutions".

I worked as a Quality Assurance and Quality Control Engineer at Radiance technologies and worked on functional and security testing of the software products developed by radiance for its customers.

I was involved in black box testing of the products wherein I was not provided with any code of the software or any other internal working and business logic, but my job was to test the functional and security aspects of the software and make sure that all of the functionalities get covered that were requested by the client.

I also worked on automation of test cases using selenium that will make the development process more efficient and will also help the company in monitoring the app performance once it has been delivered to the customer. It will continuously run in the background on the server and will send an email to the support team if there occurs an error even before the client can raise an escalation.

Keywords: Software Testing, Quality Assurance, Quality Control, Automation Test, Black Box Testing, Selenium.

Table of Contents

List of Figures

Abbreviations

Abbreviation	Description
IDE	Integrated Development Environment
SME	Subject Matter Expert
SLR	Systematic Literature Review
GUI	Graphical User Interface
SDLC	Software Development Life Cycle
SQL	Structured Query Language

PART – I

Chapter – 1

Introduction

1.1 Introduction

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not.

Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.[1]

Software testing can be stated as the process of verifying and validating that a software or application is bug free, meets the technical requirements as guided by its design and development and meets the user requirements effectively and efficiently with handling all the exceptional and boundary cases.

The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy and usability. It mainly aims at measuring specification, functionality and performance of a software program or application.

Software testing can be divided into two steps:

1. **Verification:** It refers to the set of tasks that ensure that software correctly implements a specific function.

2. **Validation:** It refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements.

Software Testing can be broadly classified into two types:

1. **Manual Testing:** Manual Testing includes testing a software manually, i.e., without using any automated tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug. There are different stages for manual testing such as unit testing, integration testing, system testing, and user acceptance testing.

Testers use test plans, test cases, or test scenarios to test software to ensure the completeness of testing. Manual testing also includes exploratory testing, as testers explore the software to identify errors in it.

2. **Automation Testing:** Automation Testing, which is also known as Test Automation, is when the tester writes scripts and uses another software to test the product. This process involves automation of a manual process. Automation Testing is used to re-run the test scenarios that were performed manually, quickly, and repeatedly.

Apart from regression testing, automation testing is also used to test the application from load, performance, and stress point of view. It increases the test coverage, improves accuracy, and saves time and money in comparison to manual testing.[2]

1.2 Objectives of Software Testing

- **Cost Effective Development** - Early testing saves both time and cost in many aspects, however reducing the cost without testing may result in improper design of a software application rendering the product useless.

- **Product Improvement** - During the SDLC phases, testing is never a time-consuming process. However diagnosing and fixing the errors identified during proper testing is a time-consuming, but productive activity.

- **Test Automation** - Test Automation reduces the testing time, but it is not possible to start test automation at any time during software development. Test automaton should be started when the software has been manually tested and is stable to some extent. Moreover, test automation can never be used if requirements keep changing.

- **Quality Check** - Software testing helps in determining following set of properties of any software such as
    - Functionality
    - Reliability
    - Usability
    - Efficiency
    - Maintainability
    - Portability

1.3 Significance

Software testing is very important because of the following reasons:

1. Software testing is really required to point out the defects and errors that were made during the development phases.
    - Example: Programmers may make a mistake during the implementation of the software. There could be many reasons for this like lack of experience of the programmer, lack of knowledge of the programming language, insufficient experience in the domain, incorrect implementation of the algorithm due to complex logic or simply human error.
2. It is essential since it makes sure that the customer finds the organization reliable and their satisfaction in the application is maintained.
    - If the customer does not find the testing organization reliable or is not satisfied with the quality of the deliverable, then they may switch to a competitor organization.
    - Sometimes contracts may also include monetary penalties with respect to the timeline and quality of the product. In such cases, proper software testing may also prevent monetary losses.

3. It is very important to ensure the Quality of the product. Quality product delivered to the customers helps in gaining their confidence.
    - As explained in the previous point, delivering good quality product on time builds the customer's confidence in the team and the organization.
4. Testing is necessary in order to provide the facilities to the customers like the delivery of high quality product or software application which requires lower maintenance cost and hence, results into more accurate, consistent and reliable results.
    - High quality product typically has fewer defects and requires lesser maintenance effort, which in turn means reduced costs.
5. Testing is required for an effective performance of software application or product.
6. It is important to ensure that the application should not result into any failures because it can be very expensive in the future or in the later stages of the development.
    - Proper testing ensures that bugs and issues are detected early in the life cycle of the product or application.
    - If defects related to requirements or designs are detected late in the life cycle, it can be very expensive to fix them since this might require redesign, re-implementation and retesting of the application.
7. It is required to stay in the business.
    - Users are not inclined to use software that has bugs. They may not adopt a software if they are not happy with the stability of the application.
    - In case of a product organization or startup which has only one product, poor quality of software may result in lack of adoption of the product and this may result in losses which the business may not recover from.

1.4 About the Organisation

Radiance Technologies is specialized in providing business process innovative solutions in the fields of healthcare, life science and manufacturing. Our solutions are focused on modern technologies and are enabled by a scientific, data-driven system that leverages machine learning and predictive analytics for Digital Excellence. This unique model de-risks our approach, provides better predictability, and ensures a better value per unit cost to our clients.

Radiance Technologies is group of seasoned professionals who have spent an average of 20 years in Industry before starting Radiance. We strongly believe in simplifying technology solutions and bringing business process innovation. Their expertise in ERP Cloud, Cloud Infrastructure, Product Development, Blockchain, Artificial Intelligence and Machine Learning has helped health care industry to achieve efficiency, collaboration, data and intellectual property protection and adhering to HIPAA compliance.

Chapter - 2

Software Testing

Software Testing is the evaluation of the software against requirements gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code. Software testing comprises of Validation and Verification.

2.1 Testing Objectives

Target of the test are:

- **Errors**: These are actual coding mistakes made by developers. In addition, there is a difference in output of software and desired output, which is considered as an error.
- **Fault**: When error exists, fault occurs. A fault, also known as a bug, is a result of an error which can cause the system to fail.
- **Failure**: Failure is said to be the inability of the system to perform the desired task. Failure occurs when fault exists in the system.

Testing can either be done manually or using an automated testing tool:

- **Manual**: This testing is performed without taking help of automated testing tools. The software tester prepares test cases for different sections and levels of the code, executes the tests and reports the result to the manager.
  Manual testing is time and resource consuming. The tester needs to confirm whether or not right test cases are used. Major portion of testing involves manual testing.
- **Automated:** This testing is a testing procedure done with aid of automated testing tools. The limitations with manual testing can be overcome using automated test tools.

Tests can be conducted based on two approaches:

- Functionality testing
- Implementation testing

When functionality is being tested without taking the actual implementation in concern it is known as black-box testing. The other side is known as white-box testing where not only functionality is tested but the way it is implemented is also behavior.

2.2 Testing Scope

There are two ends in which I could test the softwares: the GUI and the Algorithm. While the algorithm and business logic was not provided to me and I didn't have permission to access the code and business logic and algorithm, my primary areas of testing were limited to Functional Testing, Security Testing and GUI Testing.

The test strategy that we chose was to progress gradually. I studied the SRS document of the software given to me for testing and based on that, designed test cases for functional testing to test if the software meets the requirement of the customer.

2.3 Testing Principles

2.3.1 Black-Box Testing

It is carried out to test functionality of the program. It is also called 'Behavioral' testing. The tester in this case, has a set of input values and respective desired results. On providing input, if the output matches with the desired results, the program is tested 'OK', and problematic otherwise.

In this testing method, the design and structure of the code are not known to the tester, and testing engineers and end users conduct this test on the software.

Black-box testing techniques:

- **Equivalence class** – The input is divided into similar classes. If one element of a class passes the test, it is assumed that all the class is passed.
- **Boundary values** –The input is divided into higher and lower end values. If these values pass the test, it is assumed that all values in between may pass too.
- **Cause-effect graphing** – In both previous methods, only one input value at a time is tested. Cause (input) –Effect (output) is a testing technique where combinations of input values are tested in a systematic way.

- **Pair-wise Testing** – The behaviour of software depends on multiple parameters. In pairwise testing, the multiple parameters are tested pair-wise for their different values.
- **State-based testing** – The system changes state on provision of input. These systems are tested based on their states and input.

2.3.2 Functional Testing

Functional Testing is a type of Software Testing in which the system is tested against the functional requirements and specifications. Functional testing ensures that the requirements or specifications are properly satisfied by the application. This type of testing is particularly concerned with the result of processing. It focuses on simulation of actual system usage but does not develop any system structure assumptions.

It is basically defined as a type of testing which verifies that each function of the software application works in conformance with the requirement and specification. This testing is not concerned about the source code of the application. Each functionality of the software application is tested by providing appropriate test input, expecting the output and comparing the actual output with the expected output. This testing focuses on checking of user interface, APIs, database, security, client or server application and functionality of the Application under Test.[3]

Functional Testing Process:

Functional testing involves the following steps:

1. Identify function that is to be performed.
2. Create input data based on the specifications of function.
3. Determine the output based on the specifications of function.
4. Execute the test case.
5. Compare the actual and expected output.

2.3.3 Security Testing

Security Testing is a type of Software Testing that uncovers vulnerabilities of the system and determines that the data and resources of the system are protected from possible intruders. It ensures that the software system and application are free from any threats or risks that can cause a loss. Security testing of any system focuses on finding all possible loopholes and weaknesses of the system which might result into the loss of information or repute of the organization.[4]

Goal of Security Testing:

The goal of security testing is to:

- To identify the threats in the system.
- To measure the potential vulnerabilities of the system.
- To help in detecting every possible security risks in the system.
- To help developers in fixing the security problems through coding.

Principle of Security Testing:

Below are the six basic principles of security testing:

- Confidentiality
- Integrity
- Authentication
- Authorization
- Availability
- Non-repudiation

Major Focus Areas in Security Testing:

- Network Security
- System Software Security
- Client-side Application Security
- Server-side Application Security

2.3.4 White-box Testing

It is conducted to test program and its implementation, in order to improve code efficiency or structure. It is also known as 'Structural' testing.

In this testing method, the design and structure of the code are known to the tester. Programmers of the code conduct this test on the code.

The below are some White-box testing techniques:

- **Control-flow testing** – The purpose of the control-flow testing is to set up test cases which cover all statements and branch conditions. The branch conditions are tested for both being true and false, so that all statements can be covered.
- **Data-flow testing** – This testing technique emphasises to cover all the data variables included in the program. It tests where the variables were declared and defined and where they were used or changed.

Testing itself may be defined at various levels of SDLC. The testing process runs parallel to software development. Before jumping on the next stage, a stage is tested, validated and verified.

Testing separately is done just to make sure that there are no hidden bugs or issues left in the software. Software is tested on various levels:

Unit Testing

While coding, the programmer performs some tests on that unit of program to know if it is error free. Testing is performed under white-box testing approach. Unit testing helps developers decide that individual units of the program are working as per requirement and are error free.

Integration Testing

Even if the units of software are working fine individually, there is a need to find out if the units if integrated together would also work without errors. For example, argument passing and data updation etc.

2.4 Testing Methods Used

I applied the black-box testing methods, which mainly were limited to functional testing and also tested the software at various levels which are mentioned in Section 2.3 of this document. Both the paradigms gave me a lot of scope to work with the testing approaches.

I have worked on testing four projects from Radiance Technologies. In these projects, I got to work on testing the functionalities as well as the security testing of the application.

To do the functional testing, I looked at the software requirement documents of the project and designed test cases to make sure that all the possible scenarios possible in the application have been played out and that they function properly.

While doing security testing I used several applications like Zenmap, Arachni, WPScan and SQL Map to find out the possible vulnerabilities in the applications that I was testing.

2.5 Tools and Technologies Used

2.5.1 Zenmap

Zenmap is the official Nmap Security Scanner GUI. It is a multi-platform (Linux, Windows, Mac OS X, BSD, etc.), free, and open source application which aims to make Nmap easy for beginners to use while providing advanced features for experienced Nmap users. Frequently used scans can be saved as profiles to make them easy to run repeatedly. A command creator allows interactive creation of Nmap command lines. Scan results can be saved and viewed later. Saved scan results can be compared with one another to see how they differ. The results of recent scans are stored in a searchable database.[5]

2.5.2 Arachni

Arachni is a feature-full, modular, high-performance Ruby framework aimed towards helping penetration testers and administrators evaluate the security of modern web applications.

It is free, with its source code public and available for review.

It is multi-platform, supporting all major operating systems (MS Windows, Mac OS X and Linux) and distributed via portable packages which allow for instant deployment.

It is versatile enough to cover a great deal of use cases, ranging from a simple command line scanner utility, to a global high performance grid of scanners, to a Ruby library allowing for scripted audits, to a multi-user multi-scan web collaboration platform. In addition, its simple REST API makes integration a cinch.

Finally, due to its integrated browser environment, it can support highly complicated web applications which make heavy use of technologies such as JavaScript, HTML5, DOM manipulation and AJAX.

2.5.3 WPSCAN

WPScan is a free, for non-commercial use, black box WordPress vulnerability scanner written for security professionals and blog maintainers to test the security of their sites.

WPScan is written in the Ruby programming language. The first version of WPScan was released on the 16th of June 2011.

2.5.4 SQL Map

SQLMap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.

Chapter - 3

Projects Handled

The following chapter contains in brief the details of the projects that I worked on and about the test cases that were generated.

3.1 KnowX

This project is developed by Radiance Technologies for its client Cheors. It is an internal social networking website for the company for various subject matter experts to connect on. I did functional testing as well as security testing of this website using tools like Zenmap, Arachni and WPScan. A total of 257 test cases have been discovered, yet of which 247 have been resolved by the development team. Presently, I am working on the automation of these test cases using Selenium.

KnowX is a proprietary software developed by radiance technologies for Cheors and because of legal bindings it is not possible for me to reveal further details about the internal working of the software in the scope of this report.

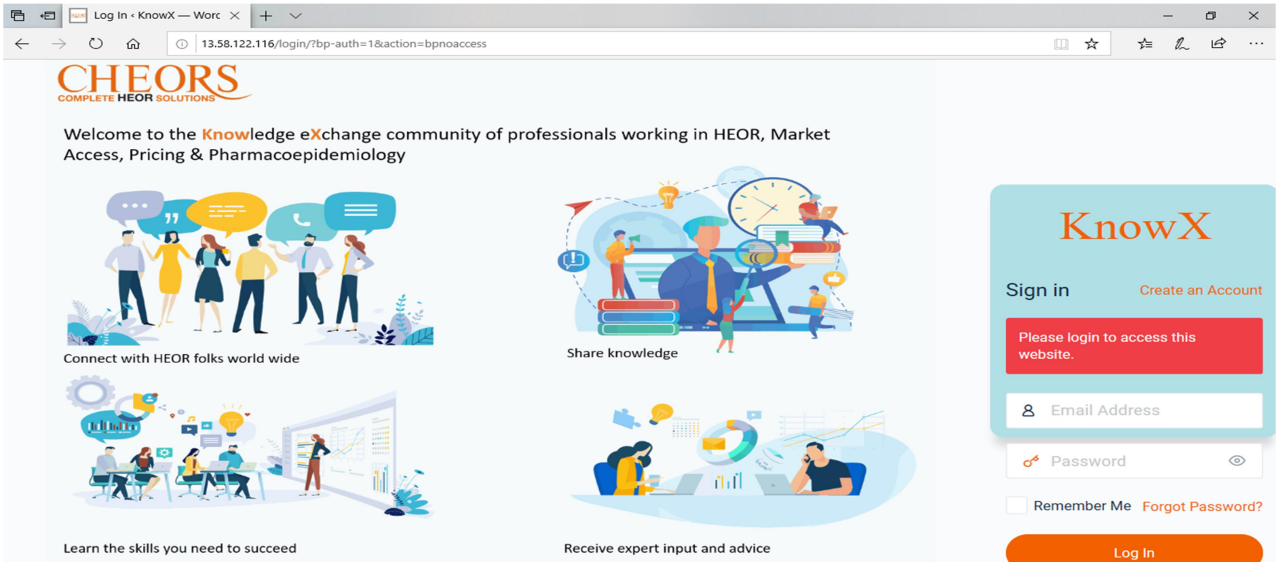

Fig. 3.1 KnowX Login Page

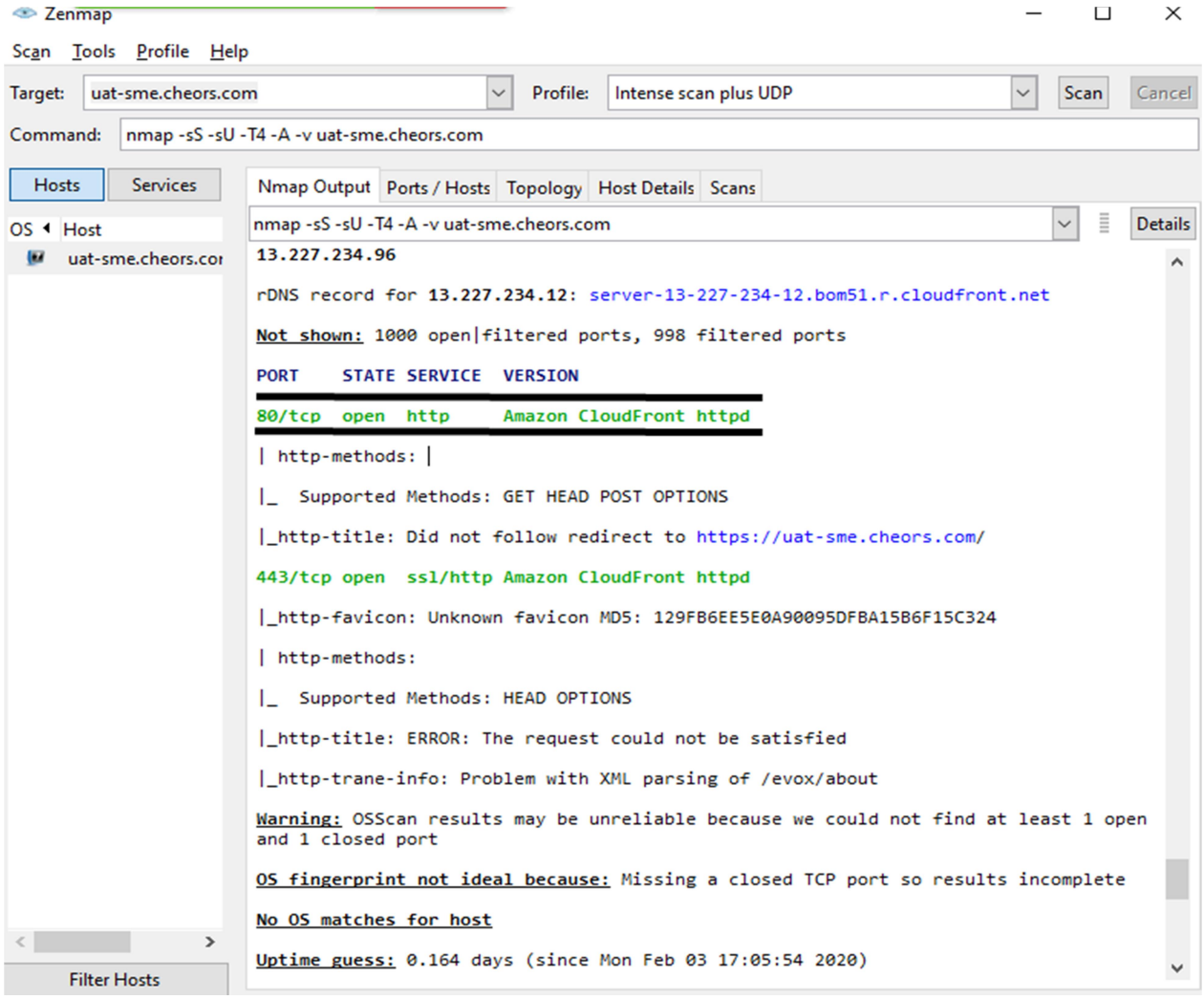

Fig. 3.2 KnowX Zenmap Report

3.2 SME

The Subject Matter Expert project is another proprietary project by Radiance Technologies for Cheors which allows the users to connect to Subject Matter Experts worldwide. The user seeking help from the SMEs can search them up and send a query form to the respective SME which suits its needs.

3.3 SLR

Systematic Literature Review is another project being developed by Radiance Technologies for Cheors. It is a Machine Learning based project that allows the client to use meta-data analysis to conduct secondary researches. This project is still in the developmental phase and I had the task to write the possible functional test cases based on the requirement document that will help in future development based on these test cases.

3.4 MAST

Mast Web-model is a cloud based platform that allows the user to use cloud a service to run complex macros on the cloud server and generate the result sparing the local machine from running extremely complex programs that are too much for the normal personal computer CPU to handle. The user can upload the input file, perform the calculation on cloud and download the results.

3.5 Project Screenshots

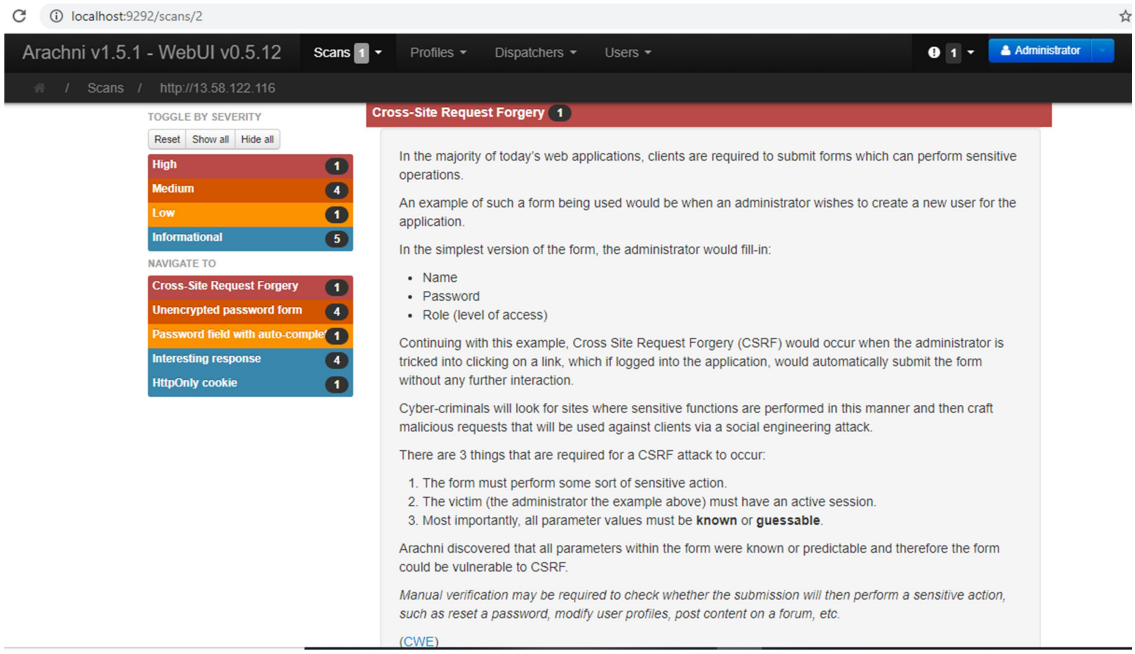

Fig. 3.3 SME App Arachni Report

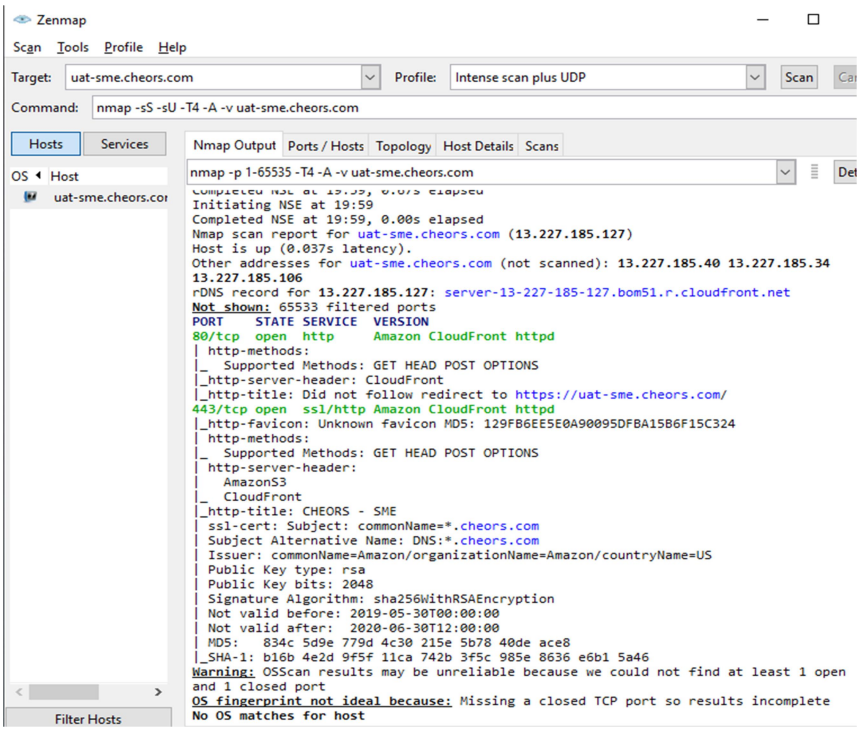

Fig. 3.4 SME App Zenmap Report (1)

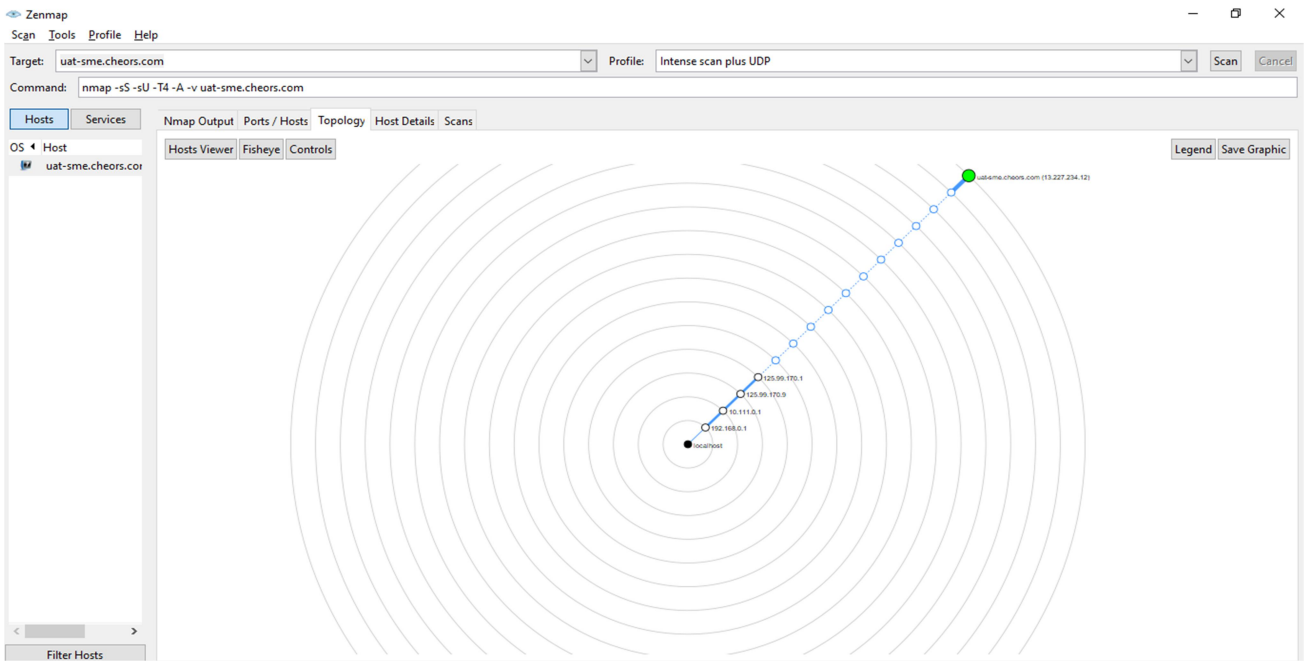

Fig. 3.5 SME App Zenmap Report (2)

3.6 Training Schedule

The internship at Radiance Technologies was a full time training with 40 hours a work week including daily meetings for updates and tracking. This was a 4 month internship during the period 20th January – 20th May 2020. This was a work from home internship in Software Testing, Quality Assurance and Quality Control.

PART - II

Chapter - 4

Automation testing

4.1 Introduction to Automation Testing

Automation Testing means using an automation tool to execute your 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 software can also enter test data into the System Under Test, compare expected and actual results and generate detailed test reports. 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.

Automated software testing is important due to the following reasons:

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

Test cases to be automated can be selected using the following criterion to increase the automation ROI:

- High Risk - Business Critical test cases
- Test cases that are repeatedly executed

- Test Cases that are very tedious or difficult to perform manually
- Test Cases which are time-consuming

The following categories of test cases are not suitable for automation:

- Test Cases that are newly designed and not executed manually at least once
- Test Cases for which the requirements are frequently changing
- Test cases which are executed on an ad-hoc basis.

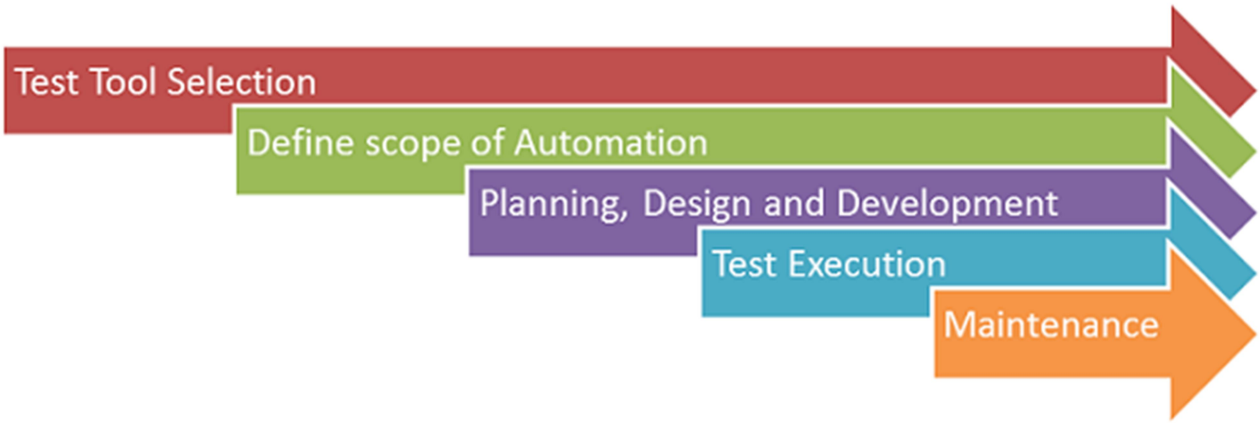

Fig. 4.1 Automation Process

4.2 Test Automation Process

The scope of automation is the area of your Application under Test which will be automated. Following points help determine scope:

- The features that are important for the business
- Scenarios which have a large amount of data
- Common functionalities across applications
- Technical feasibility
- The extent to which business components are reused
- The complexity of test cases
- Ability to use the same test cases for cross-browser testing

During planning design and development phase, you create an automation strategy & plan, which contains the following details:

- Automation tools selected
- Framework design and its features
- In-Scope and Out-of-Scope items of automation
- Automation test bed preparation
- Schedule and Timeline of scripting and execution
- Deliverables of Automation Testing

Automation Scripts are executed during this phase. The scripts need input test data before they are set to run. Once executed they provide detailed test reports.

Execution can be performed using the automation tool directly or through the Test Management tool which will invoke the automation tool.

Following are benefits of automated testing:

- 70% faster than the manual testing
- Wider test coverage of application features
- Reliable in results
- Ensure Consistency
- Saves Time and Cost
- Improves accuracy
- Human Intervention is not required while execution
- Increases Efficiency
- Better speed in executing tests
- Re-usable test scripts
- Test Frequently and thoroughly
- More cycle of execution can be achieved through automation
- Early time to market

4.3 Selenium Tool Suite

Selenium is not just a single tool but a suite of software, each with a different approach to support automation testing.[6]

It comprises of four major components which include:

- Selenium Integrated Development Environment (IDE)
- Selenium Remote Control (Now Deprecated)
- WebDriver
- Selenium Grid

4.4 Selenium IDE

Selenium IDE (Integrated Development Environment) is an open source web automation testing tool under the Selenium Suite. Unlike Selenium WebDriver and RC, it does not require any programming logic to write its test scripts; rather you can simply record your interactions with the browser to create test cases. Subsequently, you can use the playback option to re-run the test cases.

Selenium IDE is divided into different components, each having their own features and functionalities. We have categorized seven different components of Selenium IDE, which includes:

- Menu Bar
- Tool Bar
- Address Bar
- Test Case Pane
- Test Script Editor Box
- Start/Stop Recording Button
- Log, Reference Pane

1. Menu Bar

Menu bar is positioned at the top most portion of the Selenium IDE interface. The most commonly used modules of menu bar include:

- **Project Name**
  It allows you to rename your entire project.
- **Open Project**
  It allows you to load any existing project from your personal drives.
- **Save Project**
  It allows you to save the entire project you are currently working on.

2. Tool Bar

The Tool bar contains modules for controlling the execution of your test cases. In addition, it gives you a step feature for debugging you test cases. The most commonly used modules of Tool Bar menu include:

- **Speed Control Option**
  It allows you to control the execution speed of your test cases.
- **Step Feature**
  It allows you to "step" through a test case by running it one command at a time. Use for debugging test cases.
- **Run Tests**
  It allows you to run the currently selected test. When only a single test is loaded "Run Test" button and "Run all" button have the same effect.
- **Run All**
  It allows you to run the entire test suite when a test suite with multiple test cases is loaded.

3. Address Bar

This module provides you a dropdown menu that remembers all previous values for base URL. In simple words, the base URL address bar remembers the previously visited websites so that the navigation becomes easy later on.

4. Test Case Pane

This module contains all the test cases that are recorded by IDE. In simple words, it provides the list of all recorded test cases at the same time under the test case pane so that user could easily shuffle between the test cases.

At the bottom portion of the Test Case Pane, you can see the test execution result summary which includes the pass/fail status of various test cases.

Test Case Pane also includes features like Navigation panel which allow users to navigate between test cases and test suites.

5. Test Script Editor Box

Test Script Editor Box displays all of the test scripts and user interactions that were recorded by the IDE. Each user interaction is displayed in the same order in which they are performed. The Editor box is divided into three columns: Command, Target and Value.

6. Start/Stop Recording Button

Record button records all of the user actions with the browser.

7. Log, Reference Pane

The Log Pane displays the runtime messages during execution. It provides real-time updates of the actions performed by the IDE. It can be categorized into four types: Info, Error, Debug and Warn.

The Reference Pane displays the complete detail of the currently selected selenese command in the editor.

4.5 Selenium WebDriver

Selenium WebDriver is the most important component of Selenium Tool's Suite. The latest release "Selenium 2.0" is integrated with WebDriver API which provides a simpler and more concise programming interface.

The following image will give you a fair understanding of Selenium components and the Test Automation Tools.

In WebDriver, test scripts can be developed using any of the supported programming languages and can be run directly in most modern web browsers. Languages supported by WebDriver include C#, Java, Perl, PHP, Python and Ruby.

Selenium WebDriver performs much faster as compared to Selenium RC because it makes direct calls to the web browsers. RC on the other hand needs an RC server to interact with the browser.

Selenium WebDriver API provides communication facility between languages and browsers.

There are four basic components of WebDriver Architecture:

- Selenium Language Bindings
- JSON Wire Protocol
- Browser Drivers
- Real Browsers

Selenium Language Bindings/Selenium Client Libraries

Selenium developers have built Language Bindings/Selenium Client Libraries in order to support multiple languages. For instance, if you want to use the browser driver in Java, use the Java bindings. All the supported language bindings can be downloaded from the official website of Selenium.

JSON Wire Protocol

JSON (JavaScript Object Notation) is an open standard for exchanging data on web. It supports data structures like object and array. So, it is easy to write and read data from JSON.

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 REST web services.

Browser Drivers

Selenium uses drivers, specific to each browser in order to establish a secure connection with the browser without revealing the internal logic of browser's functionality. The browser driver is also specific to the language used for automation such as Java, C#, etc.

When we execute a test script using WebDriver, the following operations are performed internally.

- HTTP request is generated and sent to the browser driver for each Selenium command.
- The driver receives the HTTP request through HTTP server.
- HTTP Server decides all the steps to perform instructions which are executed on browser.
- Execution status is sent back to HTTP Server which is subsequently sent back to automation script.

Browsers

Browsers supported by Selenium WebDriver:

- Internet Explorer
- Mozilla Firefox
- Google Chrome
- Safari

Selenium WebDriver- Features

Some of the most important features of Selenium WebDriver are:

- **Multiple Browser Support**: Selenium WebDriver supports a diverse range of web browsers such as Firefox, Chrome, Internet Explorer, Opera and many more. It also supports some of the non-conventional or rare browsers like HTMLUnit.
- **Multiple Languages Support**: WebDriver also supports most of the commonly used programming languages like Java, C#, JavaScript, PHP, Ruby, Pearl and Python. Thus, the user can choose any one of the supported programming language based on his/her competency and start building the test scripts.
- **Speed**: WebDriver performs faster as compared to other tools of Selenium Suite. Unlike RC, it doesn't require any intermediate server to communicate with the browser; rather the tool directly communicates with the browser.
- **Simple Commands**: Most of the commands used in Selenium WebDriver are easy to implement. For instance, to launch a browser in WebDriver following commands are used:
  WebDriver driver = **new FirefoxDriver();** (Firefox browser)
  WebDriver driver = **new ChromeDriver();** (Chrome browser)
  WebDriver driver = **new InternetExplorerDriver();** (Internet Explorer browser)
- **WebDriver- Methods and Classes**: WebDriver provides multiple solutions to cope with some potential challenges in automation testing. WebDriver also allows testers to deal with complex types of web elements such as checkboxes, dropdowns and alerts through dynamic finders.

4.6 Selenium for Python

- Python supports the Object-Oriented Programming approach to establish the applications. It is simple and easy to learn and provides lots of high-level data structures. It is an open-source language.

- It is a high-level and interpreter scripting programing language.
- Python makes the development and debugging fast because there is no compilation step included in Python development.
- Python is very useful for automation testing because it supports multiple programming patterns.
- Python has many built-in testing frameworks such as **Pytest and Robot**, which covers the debugging and faster workflow.
- It is an interpreted language means the interpreter implements the code line by line at a time that's makes debugging easy.
- Python is Cross-platform Language; that's why it can run on different platforms like **Windows, Linux, UNIX, and Macintosh,**
- Python can be easily implemented with other programming languages such as **C, C++, Java**, etc.

Write the Selenium test script

Step 1:

In the first step, we will type the following statement to import the web driver:

```
import json
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
```

Fig. 4.2 Importing WebDriver

Step 2:

After that, we will open the Google Chrome browser and go to the target URL.

```
driver = webdriver.Chrome("C:/chromedriver.exe")
driver.get("https://dev-mast.cheors.com")
```

Fig. 4.3 Creating Browser Instance

Step 3:

In the next step, we will be maximizing our browser window size.

```
driver.set_window_size(1382, 744)
# 3 | click | name=user_name |
```

Fig. 4.4 Setting Window Size

Step 4:

Log In to the website using your credentials.

```
driver.find_element(By.NAME, "user_name").click()
# 4 | type | name=user_name | shubham.gupta@radiatechs.com
driver.find_element(By.NAME, "user_name").send_keys("shubham.gupta@radiatechs.com")
# 5 | click | name=password |
driver.find_element(By.NAME, "password").click()
        # 6 | type | name=password | Thedon@838
driver.find_element(By.NAME, "password").send_keys("Thedon@838")
```

Fig. 4.5 Logging In Using Credentials

Step 5:

Inspect various elements on the screen, find out their names and access them using their names, classes, CSS, DOM etc.

```
driver.execute_script("window.scrollTo(0,0)")
        # 10 | click | css=.col-lg-3:nth-child(1) .card-category |
driver.find_element(By.LINK_TEXT, 'studies').click()
        # 11 | click | css=li:nth-child(1) p |
driver.find_element(By.LINK_TEXT, 'studies').click()
        # 12 | runScript | window.scrollTo(0,0) |
driver.execute_script("window.scrollTo(0,0)")
        # 13 | click | css=.col-lg-3:nth-child(2) .card-category |
driver.find_element(By.CSS_SELECTOR, ".col-lg-3:nth-child(2) .card-category").click()
        # 14 | click | css=li:nth-child(1) p |
driver.find_element(By.CSS_SELECTOR, "li:nth-child(1) p").click()
        # 15 | runScript | window.scrollTo(0,0) |
driver.execute_script("window.scrollTo(0,0)")
        # 16 | click | css=.col-lg-3:nth-child(3) .card-category |
driver.find_element(By.CSS_SELECTOR, ".col-lg-3:nth-child(3) .card-category").click()
        # 17 | click | css=li:nth-child(1) p |
driver.find_element(By.CSS_SELECTOR, "li:nth-child(1) p").click()
```

Fig. 4.6 Accessing Various Elements on the Page (1)

```
        # 18 | click | css=.col-lg-3:nth-child(4) .card-category |
driver.find_element(By.CSS_SELECTOR, ".col-lg-3:nth-child(4) .card-category").click()
        # 19 | click | css=li:nth-child(1) p |
driver.find_element(By.CSS_SELECTOR, "li:nth-child(1) p").click()
        # 20 | runScript | window.scrollTo(0,0) |
driver.execute_script("window.scrollTo(0,0)")
        # 21 | click | css=li:nth-child(2) p |
driver.find_element(By.CSS_SELECTOR, "li:nth-child(2) p").click()
        # 22 | mouseOver | css=.active p |
element = driver.find_element(By.CSS_SELECTOR, ".active p")
actions = ActionChains(driver)
actions.move_to_element(element).perform()
        # 23 | mouseOut | css=.active p |
element = driver.find_element(By.CSS_SELECTOR, "body")
actions = ActionChains(driver)
actions.move_to_element(element, 0, 0).perform()
        # 24 | click | css=#ST_113 > .nc-icon |
driver.find_element(By.CSS_SELECTOR, "#ST_113 > .nc-icon").click()
        # 25 | mouseOver | css=li:nth-child(4) p |
element = driver.find_element(By.CSS_SELECTOR, "li:nth-child(4) p")
actions = ActionChains(driver)
actions.move_to_element(element).perform()
        # 26 | mouseOut | css=li:nth-child(4) p |
element = driver.find_element(By.CSS_SELECTOR, "body")
actions = ActionChains(driver)
actions.move_to_element(element, 0, 0).perform()
```

Fig. 4.7 Accessing Various Elements on the Page (2)

```
driver.find_element(By.CSS_SELECTOR, ".dt-button:nth-child(1) > span").click()
        # 28 | click | linkText=2 |
driver.find_element(By.LINK_TEXT, "2").click()
        # 29 | mouseOver | linkText=2 |
element = driver.find_element(By.LINK_TEXT, "2")
actions = ActionChains(driver)
actions.move_to_element(element).perform()
        # 30 | click | linkText=3 |
driver.find_element(By.LINK_TEXT, "3").click()
        # 31 | click | css=li:nth-child(4) p |
driver.find_element(By.CSS_SELECTOR, "li:nth-child(4) p").click()
        # 32 | click | css=li:nth-child(5) p |
driver.find_element(By.CSS_SELECTOR, "li:nth-child(5) p").click()
        # 33 | mouseOver | css=.active > a |
element = driver.find_element(By.CSS_SELECTOR, ".active > a")
actions = ActionChains(driver)
actions.move_to_element(element).perform()
        # 34 | mouseOut | css=.active:nth-child(5) > a |
element = driver.find_element(By.CSS_SELECTOR, "body")
actions = ActionChains(driver)
actions.move_to_element(element, 0, 0).perform()
        # 35 | mouseOver | css=.text |
element = driver.find_element(By.CSS_SELECTOR, ".text")
actions = ActionChains(driver)
actions.move_to_element(element).perform()
        # 36 | click | css=.text |
```

Fig. 4.8 Accessing Elements on the Page

```
driver.find_element(By.CSS_SELECTOR, ".text").click()
        # 37 | mouseOut | css=.text |
element = driver.find_element(By.CSS_SELECTOR, "body")
actions = ActionChains(driver)
actions.move_to_element(element, 0, 0).perform()
        # 38 | click | css=li:nth-child(3) .sidebar-normal |
driver.find_element(By.CSS_SELECTOR, "li:nth-child(3) .sidebar-normal").click()
        # 39 | click | css=.swal2-confirm |
driver.find_element(By.CSS_SELECTOR, ".swal2-confirm").click()
```

Fig. 4.9 Accessing Elements on the Page

4.7 Outputs

This section contains the outputs obtained by running the code on the Selenium IDE and also the outputs obtained on the Browser.

Fig. 4.10 Selenium IDE Outputs (1)

Fig. 4.11 Selenium IDE Outputs (2)

Fig. 4.12 Selenium IDE Outputs (3)

24. mouseOut on css=li:nth-child(6) p OK	14:41:03
25. click on css=li:nth-child(6) p OK	14:41:03
26. runScript on window.scrollTo(0,0) OK	14:41:03
27. click on css=li:nth-child(7) p OK	14:41:03
28. runScript on window.scrollTo(0,0) OK	14:41:04
29. click on css=li:nth-child(8) p OK	14:41:04
30. click on css=#Settings li:nth-child(1) .sidebar-normal OK	14:41:04
31. runScript on window.scrollTo(0,0) OK	14:41:04
32. Trying to find css=#Settings li:nth-child(2) .sidebar-normal... OK	14:41:04

Fig. 4.13 Selenium IDE Outputs (4)

33. click on css=#Settings li:nth-child(3) .sidebar-normal OK	14:41:15
34. mouseOver on css=.text OK	14:41:16
35. mouseOut on css=.text OK	14:41:16
36. mouseOver on css=.text OK	14:41:16
37. Trying to find css=.text... OK	14:41:16
38. mouseOut on css=.text OK	14:41:17
'login-logout-clickevents' completed successfully	14:41:17

Fig. 4.14 Selenium IDE Outputs (5)

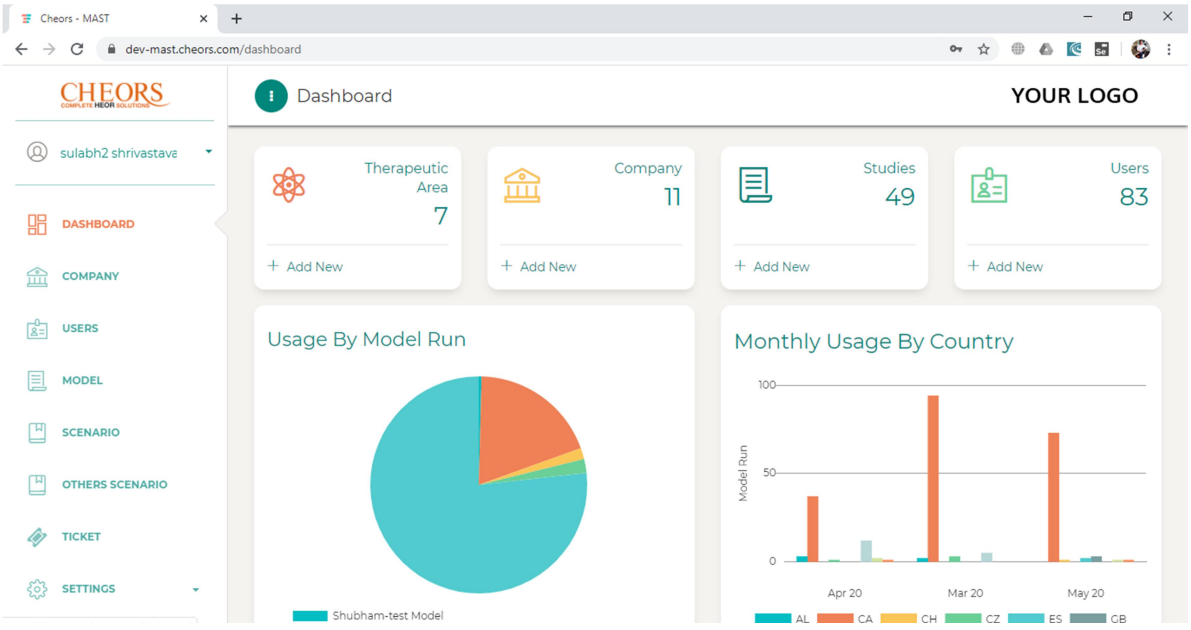

Fig. 4.15 Web App Dashboard

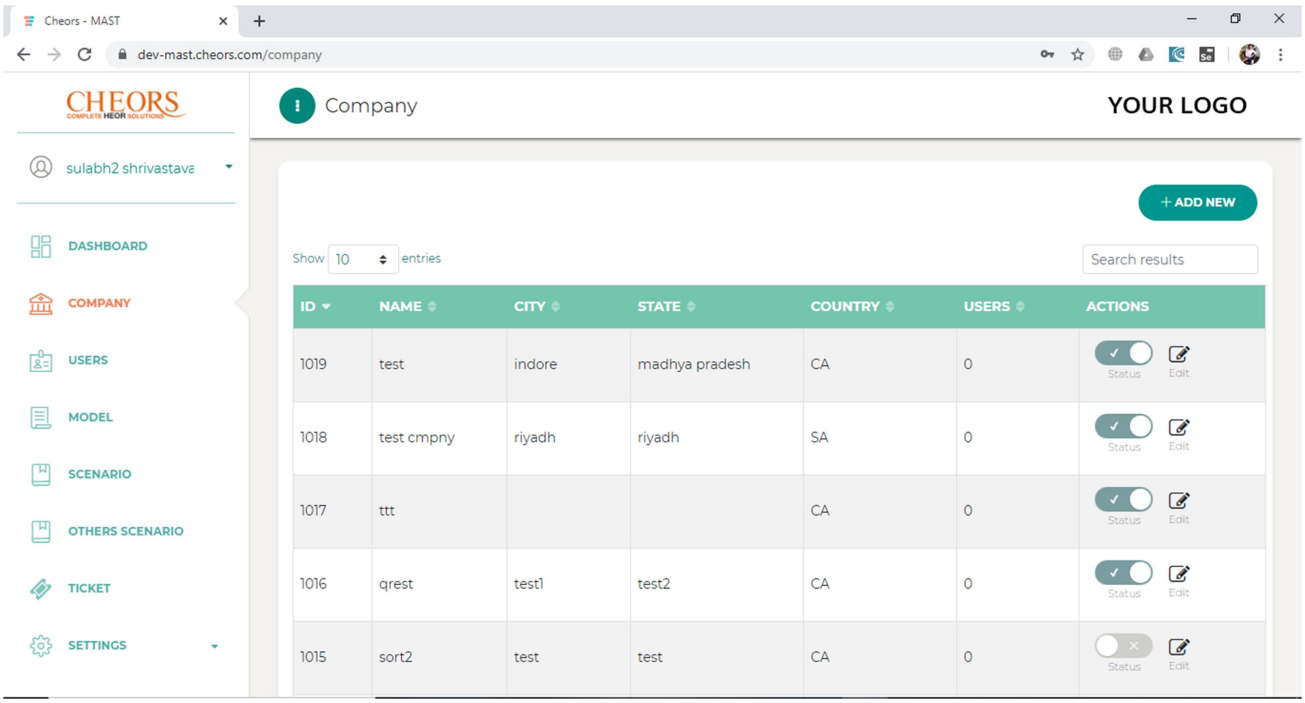

Fig. 4.16 Web App Company Page

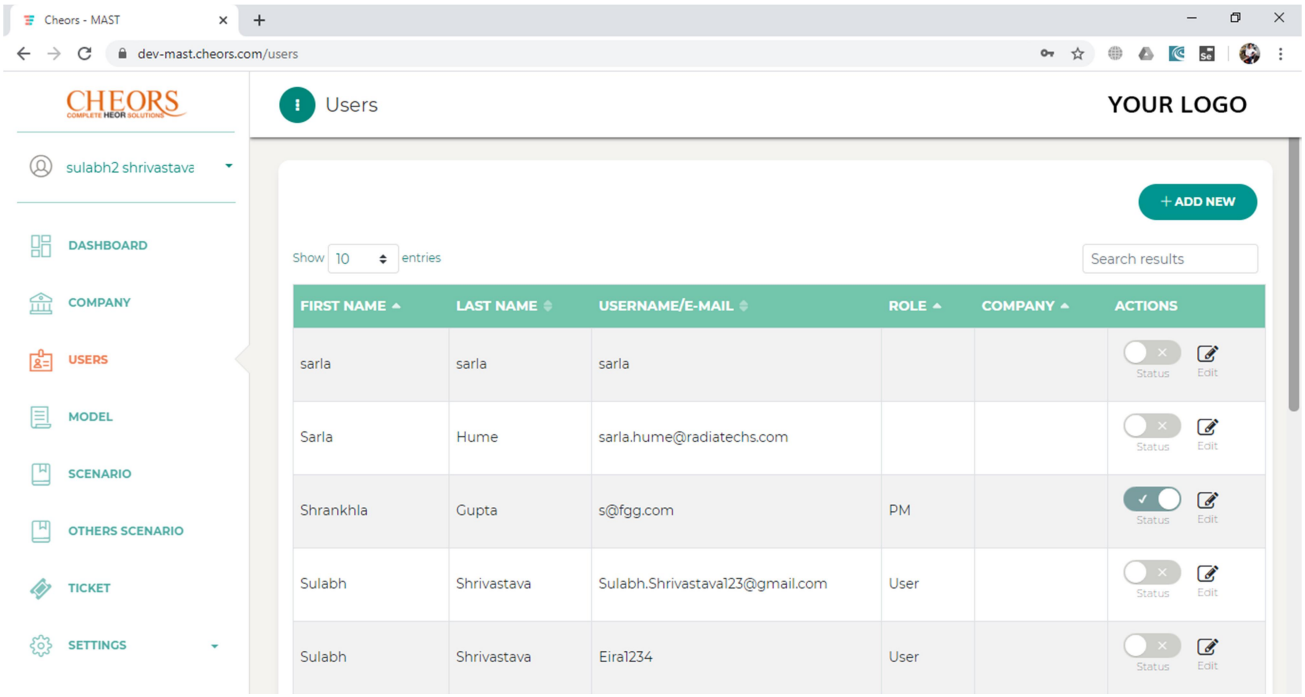

Fig. 4.17 Web App Users Page

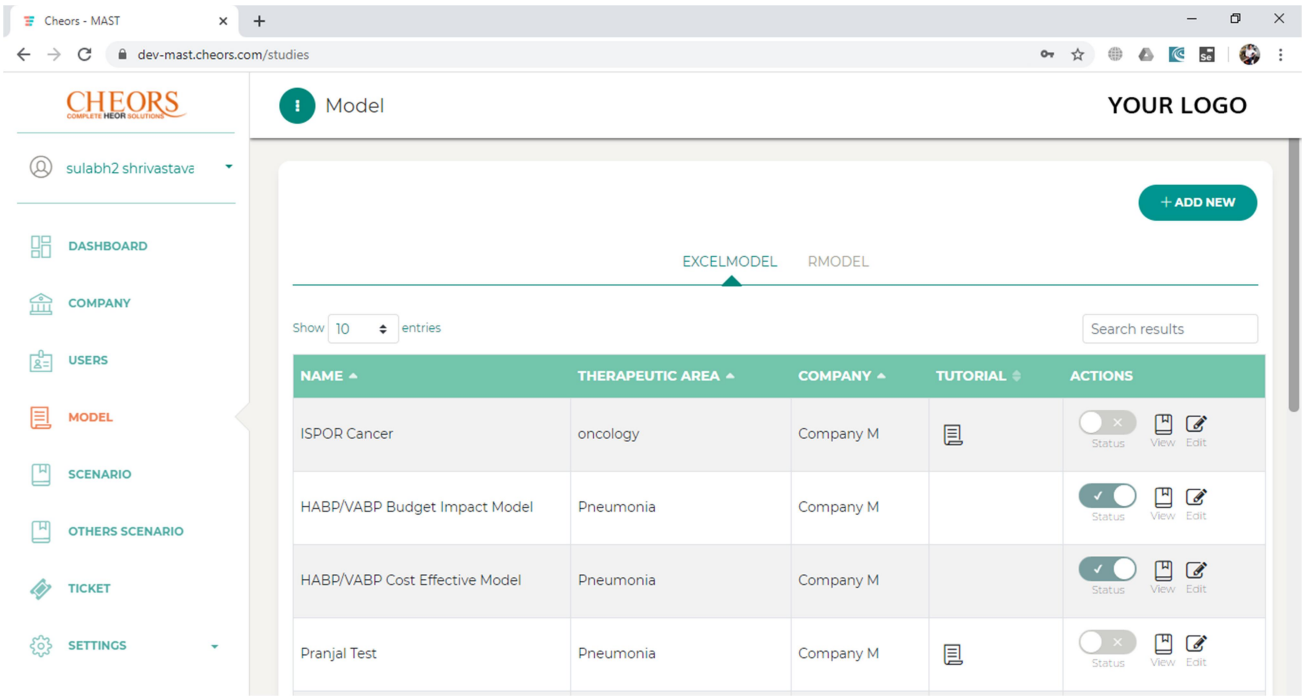

Fig. 4.18 Web App Model Page

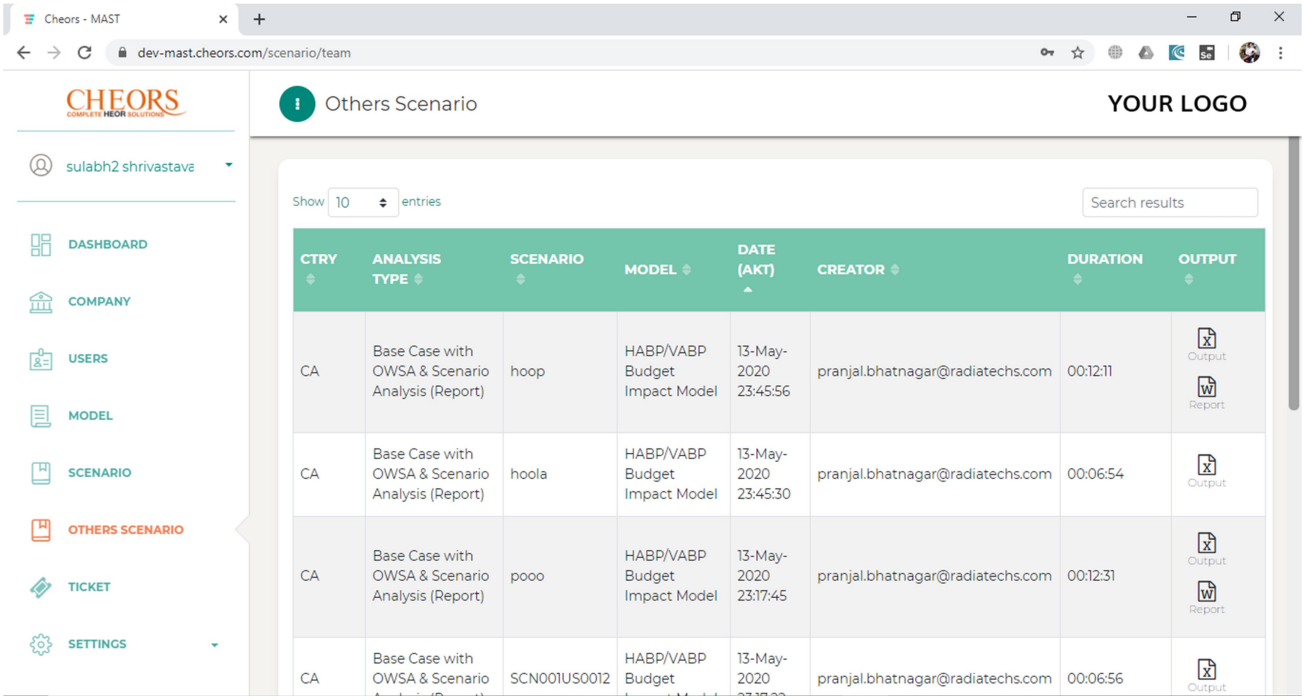

Fig. 4.19 Web App Others Scenario Page

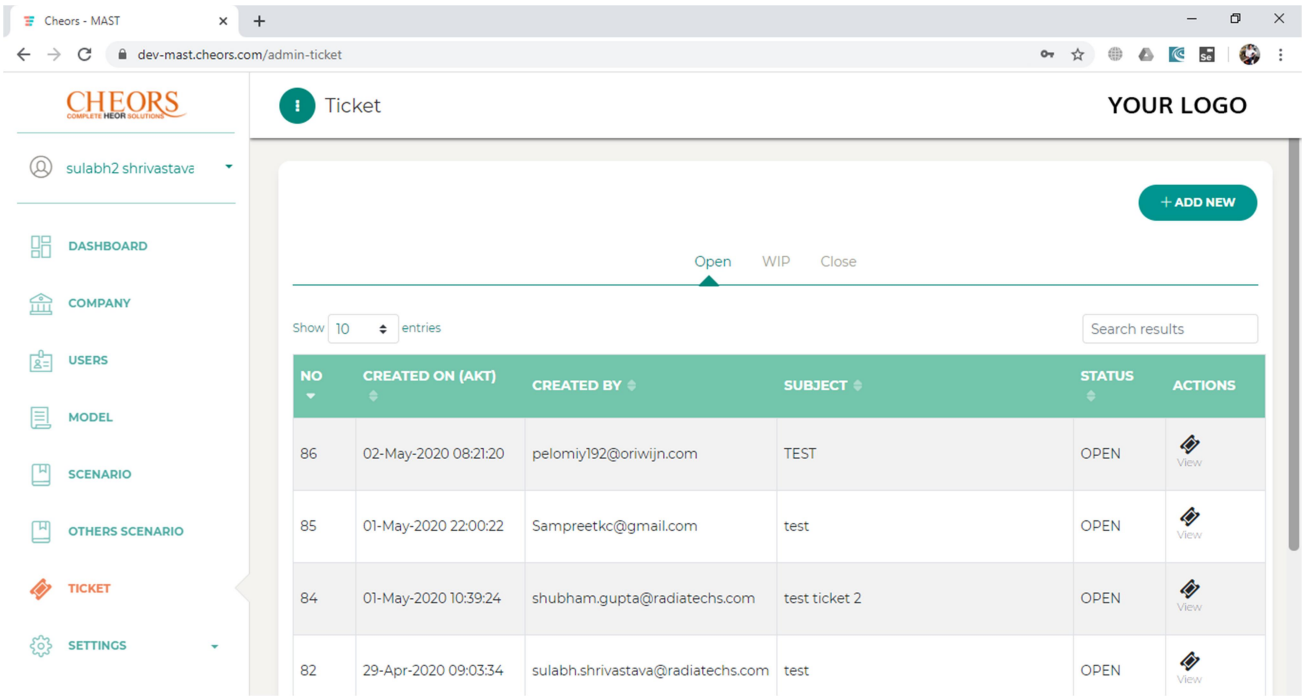

Fig. 4.20 Web App Ticket Page

Chapter - 5

Future Scope

Software testing is a rigorous process that is done continuously to ensure the quality of the software products that are finally delivered to the customer. Software testing is also necessary in the initial phases of the development of the software so that any bugs that are present in the software can be identified in the initial stages of development thus reducing the cost of the development.

The automation testing project can be extended to run on cloud server that will automatically pick execution at a specified time and will send an email to the support team before the client can catch the problem in the software.

Chapter - 6

Learning from the Training

Working on this project gave me deep insight about how things move in a professional software firm. While working in an IT company is fun, it equally challenging too.

While working on the testing and automation of test scripts of this project, I got plenty of opportunities to hone my coding skills and use my knowledge to solve software engineering problems as well as learn new things that are very useful in any software development life cycle.

This project gave me deep insights about working on the black box testing and how breaking up the project in modules helps in fast and accurate development of the test cases for the project which utilizes the talent of each member of the team to the best possible extent and deliver a quality project that meets the demand of the client in the best possible manner.

On the technical front this project taught me a great deal about test automation and dealing with CSS elements in web based applications and also the benefits of developing automation test scripts

Developing automation scripts in the project not only helps in simplifying the testing process but also reduces manual efforts and saves time.

While the technical learning greatly enhanced my skills, I also got to learn to deal with ever changing client demands and to work under the pressure of a deadline. This proved to be a very new and different experience as the exposure to deal with people gave me a lot of confidence to deal with what would happen in a full time corporate job. Dealing with client demands and deadlines also taught me how to use various SDLC models to my benefit.

Chapter - 7

Conclusion

This training was carried out in two parts. The first part was manual functional testing of software developed by Radiance Technologies where I was trained in rigorous software testing, quality control and quality assurance. The second part of the training was automation testing using Selenium to make the testing process more efficient and the bugs can be identified in earlier stages of development thus reducing the cost of development. This training was a unique experience as I got to perform software testing for the first time in my career. This training has helped me grow as a professional and I have learnt to work in a corporate culture under strict deadline and supervision. I have also learnt to handle multiple projects at a time and working with a team. This is a full time internship where I have worked 40 hours a week.

This opportunity has given me the chance to do functional testing of software, and also security testing and will advance to automation testing in the future.

I have also written test cases for the application under development that will help the development team to keep these test cases in mind and will help reduce time and energy in testing after the development is complete.

This training will also help in my professional career as a software developer while I would be able to look at the software under development with the insight of a developer and create better projects.

I have also learnt to follow strict deadlines and professionalism to be followed in meetings and corporate hierarchy.

Overall this project has been an excellent learning opportunity for me and will help me grow as an individual as well as a professional.

Chapter - 8

Bibliography and References

[1]"Software Testing - Overview - Tutorialspoint", Tutorialspoint.com, 2020. [Online]. Available: https://www.tutorialspoint.com/software_testing/software_testing_overview.htm. [Accessed: 09- Apr- 2020].

[2]"Software Testing | Basics - GeeksforGeeks", GeeksforGeeks, 2020. [Online]. Available: https://www.geeksforgeeks.org/software-testing-basics/. [Accessed: 09- Apr- 2020].

[3]"Software Testing | Functional Testing - GeeksforGeeks", GeeksforGeeks, 2020. [Online]. Available: https://www.geeksforgeeks.org/software-testing-functional-testing/?ref=rp. [Accessed: 09- Apr- 2020].

[4]"Software Testing | Security Testing - GeeksforGeeks", GeeksforGeeks, 2020. [Online]. Available: https://www.geeksforgeeks.org/software-testing-security-testing/?ref=rp. [Accessed: 09- Apr- 2020].

[5]"Zenmap - Official cross-platform Nmap Security Scanner GUI", Nmap.org, 2020. [Online]. Available: https://nmap.org/zenmap/. [Accessed: 09- Apr- 2020].

[6]"Selenium Tool Suite - javatpoint", www.javatpoint.com, 2020. [Online]. Available: https://www.javatpoint.com/selenium-tool-suite. [Accessed: 09- Apr- 2020].