S	
E1	
E2	
For Office Use Only	

Masters Project Final Report

(MCS)

2019

Project Title	**Automated Test Script Executing Tool**
Student Name	**LDSS Karunarathna**
Registration No. & Index No.	**2014/MCS/039** **14440395**
Supervisor's Name	**Dr. GDSP Wimalarathne**

For Office Use ONLY

Automated Test Script Executing Tool

A dissertation submitted for the Degree of Master of Computer Science

LDSS Karunarathna

University of Colombo School of Computing

2019

Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: LDSS Karunarathna

Registration Number: 2014/MCS/039

Index Number: 14440395

____

Signature: Date:

This is to certify that this thesis is based on the work of Mr. LDSS Karunarathna under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name: Dr. GDSP Wimalarathne

____

Signature: Date:

Abstract

Software testing process is a critical aspect of any software development process as it is essential to verify the software product for its proper functionality with the expected quality level. With the recent developments happening in software testing domain, test automation has been attracting a lot of attention from the industry as a result of urgent need to increase the productivity of the process while maintaining the accuracy.

The present day test automation tools (MarathonITE, Selenium, and Sikuli) have a wide variety of features which makes them more suitable for different contexts. However, most of them has several problems in common. Some significant problem areas identified are; need to access the code for GUI component identification, not being able to execute batches of test scripts, requirement of customized test reports for high quality outcomes, and not having compatibility with other technologies used in the product etc. Though these problems have been resolved in certain tools, there was not a single tool which has catered to all the issues. Thus, through this research project it was intended to come up with a test automation tool which could help to address all the aforementioned issues.

Image processing based GUI component identification is gaining a lot of popularity in the context of test automation. Thus this project used a supporting tool called Sikuli which could perform the initial tasks such as test script creation. Then the developed tool through this research project will include the functionalities of executing a flow of test scripts (which were created by the supporting tool), and generating customized test reports based on test statistics etc. The specialty of this automated test script executing tool is that it does not require the code level access and doesn't require a customized framework to be compatible with the other technologies.

The prototype of the tool is developed using key technologies and tools such as Java, Python, OpenCV, and Sikuli APIs. The tools underwent several testing processes to ensure that it is serving the purpose in the correct way. Further, this prototype of the tool was also subjected to the evaluations of the stakeholders of the test automation domain such as QA leads, QA engineers and Software engineers etc.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ATS - Automated Test Script

FADEC - Full Authority Digital Engine Control

IPBCI -Image Processing Based Component Identification

API – Application Program Interface

QA – Quality Assurance

PRINCE2- Projects in Controlled Environments, version 2

SDK- Software Development Kit

JRE- JAVA Runtime Environment

STLC – Software Testing Life Cycle

GUI – Graphical User Interface

DAG - Directed Acyclic Graph

NN- Neural Network

NASA - National Aeronautics and Space Administration

IEEE – Institute of Electrical and Electronic Engineers

IBM - International Business Machines

UML - Unified Modeling Language

IDE - Integrated Development Environment

API - Application Program Interface

OOD – Object Oriented Design

Chapter 1: Introduction

Chapter 1 – Introduction

1.1 Chapter overview

This chapter intends to provide an overall idea of the research project titled as 'Automated Test Script Executing Tool'. The chapter will begin with the elaboration of the problem domain which justify the need to have a new test automation tool with enhanced functionalities. This is followed by the definition of the aim and objectives of the project, basics of requirement analysis process, overview about the development of the prototype, testing, critical evaluation and the documentation aspects of the proposed solution.

1.2 Problem statement

Software testing has become an integral part of the software engineering process in today's context. The customer's expectation on obtaining the highest possible quality product for the price being paid has resulted in increasing attention towards ensuring the quality aspect of the final software product. Throughout the entire development process of the software, testing is involved in various phases. As the amount of testing being done is increasing at a rapid pace, it is very common to see that various types of automation tools are being introduced to the market, to automate the test cases being performed on the product. This has drastically resulted in enhanced accuracy and the improved efficiency in the software testing process.

There are several tools to Automate Web clients (Eg: Selenium [1]). But there are only few software automation tools for Java/Swinging applications (Eg: Marathon [2]) at the moment.

However the solutions which are present at the moment to serve this purpose have some drawbacks which limit the scope of expanding the efficiency furthermore. Also, most of these solutions still require a certain amount of human involvement during the test case automation procedure. For an example, most of the automated tools still require the user to define the components of a software product before starting the test case automation process. In this context, there are some instances where the components are just labeled as unidentified or duplicate resulting in complicated maintenance issues.

The aforementioned drawback is overcome by another software test automation tool called 'Sikuli' [3], where it has the capability to automate anything that appears on the screen. This tool does not require code level access and uses image recognition techniques to control and identify GUI components (Described in detail in Chapter 2).

Yet, even 'Sikuli' still has several weaknesses which should be overcome to provide a smoother testing mechanism. Even though it can automate a single test script, it cannot execute a bunch of test scripts at the same run. Also it cannot generate the test reports which contains the status of the tests (pass or fail), and types of errors etc. This deprives the ability to decide whether the software product under test is verified or not.

The research work currently being done under this area is as follows.

A Study on an Intelligent General-Purpose Automated Software Testing Suite [4] shows that this manual software testing can be automated with the aid of an automated software testing scheduler and a log file error analyzer. This has enabled the testing engineer to keep track of the test process even from an outside location through a report generated during the process.

Another software testing automation framework [5] being published by IEEE suggests a multiplatform, multi-language approach for the reuse the services that can be used to automate major activities in the testing process. Here reusing has been identified as a potential solution to efficiently automate resource-intensive test suites and was actually tested in IBM.

Analysis of a software automation test protocol [6] also covers a deep analysis of software automation testing, and the definition, characteristics and functions which will be very useful in coming up with a suitable solution to the problem being addressed through this project.

Using these concepts and research work material currently available, it is assumed that a new approach which is more effective could be introduced for the automated software testing process

1.3 Project Aim and objectives

To design, develop and evaluate a software tool which can automate the execution of multiple software test scripts during one test cycle, using a readily available image processing based component identification tool. The new test tool will include several other enhanced features such as, generation of test reports, and ability to add the executing conditions to the script level etc.

Further, the new test automation tool will make use of the API of 'Sikuli' automation tool in component identification.

This project involves the following set of objectives to be achieved during the project.

- Literature survey being conducted to get a solid idea on how the test case automation is being done at the moment. This will involve deeply analyzing the existing test automation tools in the market. Literature survey will also cover the understanding of present research work being carried out in line with this purpose.
- Requirement analysis on the topic being selected for the project. The drawbacks of the present solutions and the exact expectation of the end user is taken into account in this case.
- The design and the framework of specifications to implement the project. This design will be the foundation to carry out the development of the new test automation tool.
- Considering the complexity, features of the design and the specifications, define the set of development tools that will be used for the project.
- Development of a fully functioning prototype to demonstrate the goal of the project being proposed.
- Testing of the prototype to ensure that it adheres to the final expectations of the project.
- Critical evaluation of the parts of the tools being developed to check whether the expected outcome is achieved and the expected performance standards are being met.
- Clear and descriptive documentation of the entire process of developing the test automation tool.

1.4 Structure of the thesis

Chapter 1 – Introduction

This will provide the overview of this research project including problem identification, prior work, project aim, objectives, prototype features, deliverables, resource requirements and the final documentation structure.

Chapter 2 – Literature Review

The purpose of this chapter is to provide details on related work for the research topic. This chapter will cover the information extracted from the relevant documented literature available. This chapter covers the importance of software testing, already available software test automation tools and their novelties and limitations, and technical details on how the image processing based GUI component identification is done etc.

Chapter 3 - Project Management

The project management chapter will focus on the project plan of this research including the work schedule, risk mitigation plan, the most feasible software development methodology, and the research methodology proposed for the project.

Chapter 4 - Requirements Analysis

This chapter will cover the process of requirement elicitation which included the process of information gathered via various sources such as literature review, questionnaires and brainstorming sessions. It also includes the identifying the stakeholders of the system, use case model, and finalized set of functional and non-functional requirements.

Chapter 5 - Architecture and Design

Chapter 5 intends to cover the design considerations, design goals, high level design in terms of rich picture and high level architecture, and system design in terms of class diagram, sequence diagram, and entity relationship diagram.

Chapter 6 – Implementation

The chapter on Implementation provides a detailed explanation as to how the proposed design was implemented to develop the prototype. Technology selection and the core functionalities description with the aid of code snippets. Finally, the outcome of the prototype and sample results are presented.

Chapter 7 – Testing

This chapter will focus on describing how the testing is carried for the implemented prototype. The testing criteria will be defined at the beginning. Then the testing results of functional requirements (elicited during requirement analysis stage and during the implementation stage as well) and the nonfunctional requirements are presented. The chapter closes with stating the limitations of the testing process conducted.

Chapter 8 – Evaluation

The purpose of this chapter is to present the evaluation carried out by external parties such as QA Leads, QA engineers and software engineers etc. Their feedback was gathered under 6 aspects and for each aspect the summary of the feedback and the review is presented. This chapter mainly focuses on providing a qualitative evaluation for the system developed. Finally a self-evaluation is carried out with the purpose of identifying the strengths and weaknesses of the project.

Chapter 9 – Conclusion

This chapter will provide the closing remarks of the entire project through a brief analysis of the achievement of the aim and objectives defined in chapter 1, how the modules followed in the course helped to carry on the project, utilization of existing knowledge, the learning outcomes, challenges faced and the future improvements.

Chapter 2: Literature Review

Chapter 2. Literature Review

2.1 Chapter Overview

This chapter is prepared to give the reader an overview of the software testing process in today's context. This will provide the background of the problem being addressed through this research project and will also will testify the need to incorporate automated software testing tools. This also provides detailed descriptions as to how the existing methodologies will function, their drawbacks and related technologies and tools available to overcome the said issues. The knowledge gathered from this literature review will be thoroughly used to develop the new approach introduced in this research to improve the automated software testing procedures.

2.2 Software Testing

To identify the completeness, quality and correctness of a developed product or application is known as Software testing. This includes set of activities conducted, to identify errors in software so that it can be corrected before the product is released to the customer.

Software testing has become an integral part of the software engineering process in today's context. The customer's expectation on obtaining the highest possible quality product for the price being paid has resulted in increasing attention towards ensuring the quality aspect of the final software product. Throughout the entire development process of the software, testing is involved in various phases.

Moreover, unattended software defects can end up dangerous and even expensive [7]. Following are few of the instances which provide evidences as to the extent of damage those could cause.

- Chinook [8] helicopter crashed due to a software flaw in the FADEC system in 1994 in Scotland, killing 29 passengers on board.
- A software bug in Canada's Therac-25 [9] radiation therapy machine caused it to severely malfunction in 1985 which caused to deliver lethal radiation doses to patients, leaving 3 people dead and critically injuring 3 others.
- A failure worth of USD 1.2 billion was caused during a military [10] satellite launch as a result of a software bug in April 1999. This considered as one of the costliest accidents in the history of Cape Canaveral launches.

2.3 Process of Software Testing

As stated in section 2.2, Software Testing [11] is involved throughout the various phases involved in a Software project. At the same time, the process does not include just a single activity, but a series of activities carried out in a well-planned manner which will ultimately guarantee the quality of the software product. This series of activities are technically known as the Software Testing Life Cycle (STLC). STLC has the following steps.

- Requirement analysis
- Test planning
- Test case development
- Test environment setup
- Test execution
- Test cycle closure

2.4 Why Automation of Software testing is Important

As the amount of testing being done is increasing at a rapid pace, it is very common to see that various types of automation tools are being introduced to the market, to automate the test cases being performed on the product. This has drastically resulted in enhanced accuracy and the improved efficiency in the software testing process.

Basic functioning of automation testing would be, using a special tool which could execute test scripts automatically to run a software application. Cost of automated testing is evaluated based on the effort and time required to create the tests. Thus, based on the cost-benefit analysis, a particular test can be decided to automate [12].

The requirement of test automation is further justified by following reasons.

- Need to do regression testing – This involves reusing existing test cases repeatedly to check whether new additions to a particular product does not create a negative impact or a defect in the software product. This approach is useful to provide efficient results.
- When there is a need to run test cases multiple times iteratively using different datasets.
- These days it is a very common practice to use agile approach to perform test cases. Test automation eases this process as it reduces the effort required.
- Different stakeholders require different types of test reports on a test being done. Automation can helps to generate customized reports as per the requirement.

2.5 Existing tools in the Market for Automated Software Testing

2.5.1 MarathonITE

Marathon Integrated Test Environment (MarathonITE) [2] is a cross platform test automation framework for Java/Swing applications. This has an inbuilt script recorder, which can create test scripts, and then run later separately or as a batch. Refactoring tools used in MarathonITE helps to have a better reusability of test scripts. Its ability to continuously monitor the actions of the test performer, helps to conduct exploratory testings more efficiently. Semi-automatic tests can be performed as well in the case where manual handling is required.

However performance of MarathonITE is limited as it sometimes doesn’t function properly with the subjected software product due to certain incompatibilities. In such cases, the company using this tool will develop their own framework to enable these functionalities.

2.5.2 Selenium

Selenium, is an automated web testing tool [1] which is quite popular around the world. It is a set of different test tools (tool suite), in which the user can choose the tools they prefer. Each tool has a different set of capabilities and options, so it is easier to build a customized test tool according to the requirement.

Tool suite of Selenium includes the following.

Selenium 1 - This is the very first tool came under Selenium. Introduction of this was the first time to witness an automated test tool which allowed to control browser actions from any language. However the benefits were limited of this version due to Javascript based automation engine, and security limitations imposed by the browsers on Javascript.

Selenium 2 - This is also known as the Selenium WebDriver which provides an advanced object oriented API to overcome the issues in the previous version.

Selenium IDE - This prototype tool serves the purpose of writing test scripts. Its recording feature helps to record the steps as they are executed and then allows to export them as reusable scripts. However features like iteration and conditional statements of test scripts do not include in this.

Selenium Grid - This is an extended version of Selenium 1 which has the capability to run very large test cases, and to run in multiple test environments making use of the parallel processing techniques. Limitations [13] of Selenium tools include not supporting non-web based applications, and no in-built reporting tools (thus need plugins such as JUnit).

2.5.3 Sikuli

This open source software test automation tool [3] identifies and controls GUI components of the software product through image recognition. In contrast to the other tools in the market, this doesn't require code level access making it less complex and more user friendly for the user. More information could be found in 2.6.2.

Yet, the performance is limited as the test scripts can be run only one at a time

2.6 Literature on Related Techniques used in Software Test Automation

This section covers the findings on the most relevant literature for the proposed solution.

2.6.1 Types of Tools Available for Visual Workflow Automation

Visual Workflow of a particular task can be defined as the set of steps being conducted to perform that task. If that is being performed repeatedly, it is easy to use an automatic approach. There are 3 types of techniques been used in tools available for visual workflow automation.

Recorder – This is when the tool tracks the action of the mouse pointer and keyboard and record those actions and then play back when needed. These recordings are usually flexible to amend with any additional features. [14]

GUI aware – This is where the code level of GUI components being accessed and program the task to happen automatically.

Visually – This approach will observe the screen as images, and the mouse and keyboard can perform simulations on these images to depict the actual task being done. Most of the times this will not require the code level access which makes this category more suitable for the proposed solution.

2.6.2 Image Processing based GUI Component Identification Method used in Sikuli

Sikuli's approach [14] for visual workflow is the 3rd category mentioned in 2.6.1. This section will briefly describe the major aspects involved in this process which are specific to Sikuli

2.6.2.1 Template Matching Technique to Find Images on the Screen

This tool [15] utilizes the OpenCV package which uses template matching technique. Basically this will find areas of an image that will match to a template image. Given that, there are 2 primary components namely source image (the image subjected to find matches with a template image) and template image (which will act as the reference to find matches). The area which has the highest matching will be will be identified through this process. To find the matching area for two images, the template image will be slid across the source image.

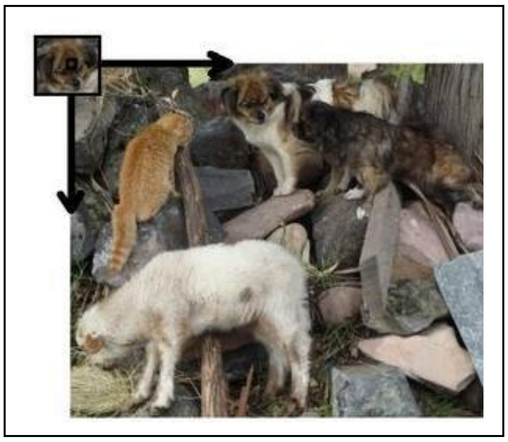


Figure 2.6.1: Sliding of Template Image over the Source Image. Image downloaded from OpenCV 2.4.13.7 documentation

As shown in Figure 2.6.1, the template image will be moved one pixel at a time from left to right and up to down. At each point, a metric as to the extent of matching will be calculated and these values will be stored in a matrix. I.e. This matrix will have match metrics for each location with the corresponding {x, y} coordinates. Based on the 6 types of different matching methods available [15], the respective location coordinates of the highest (or lowest) metric value, will be taken as the most matching location. The Figure 2.6.2 depicts the steps involved in the OpenCV code for template matching process.

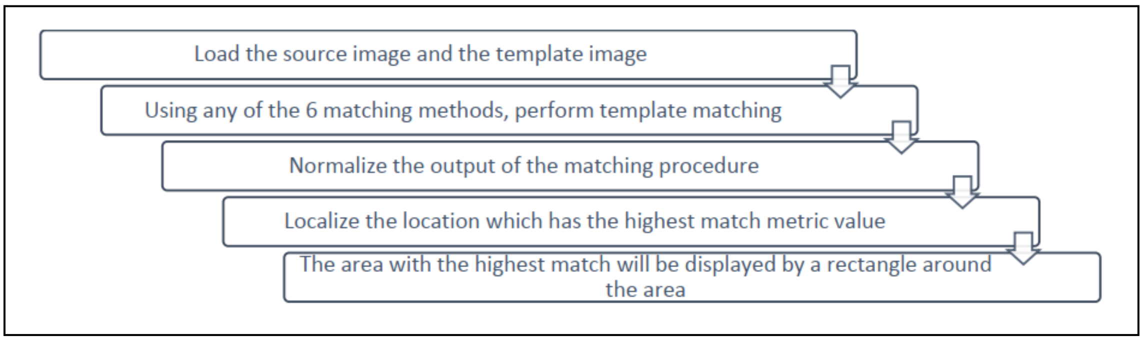

Load the source image and the template image
Using any of the 6 matching methods, perform template matching
Normalize the output of the matching procedure
Localize the location which has the highest match metric value
The area with the highest match will be displayed by a rectangle around the area

Figure 2.6.2: OpenCV Algorithm for Template Matching

2.7 Literature on Alternate Approaches for Software Test Automation

2.7.1 Component Identification Techniques

GUI Ripping: Reverse Engineering of Graphical User Interfaces for Testing [16]

As Graphical User Interfaces (GUIs) plays a vital role in ensuring the expected behavior of the entire software, their correct execution is quite critical in today's context. This research has identified this requirement has proposed a new mechanism for automated GUI testing, which can identify the components of the GUI and store them in the manner of test cases.

The researchers have used a reverse engineering approach of a model represented by structures which are known as GUI forest, event-flow graphs and an integration tree directly from the executable

GUI. The term “GUI Ripping” has been used to describe a dynamic process where the particular software’s GUI is automatically travelled across by opening all windows of the software. Through this way, all their widgets which are also known as GUI objects, properties and values are extracted. This information will then be verified by the test designer and then will be used to generate the test cases automatically. The outcome of this research was to build algorithms to facilitate this ripping process and those have been implemented in a tool suite which operates on Java and Microsoft Window’s GUI. The results of a set of case studies prove that this approach can be handled with very little human engagement. Thus, it is obvious that this will be very useful in regression testing of software which is modified quite frequently. Through this concept being tested in several large experiments, now it is available as a downloadable tool.

GUI Forest

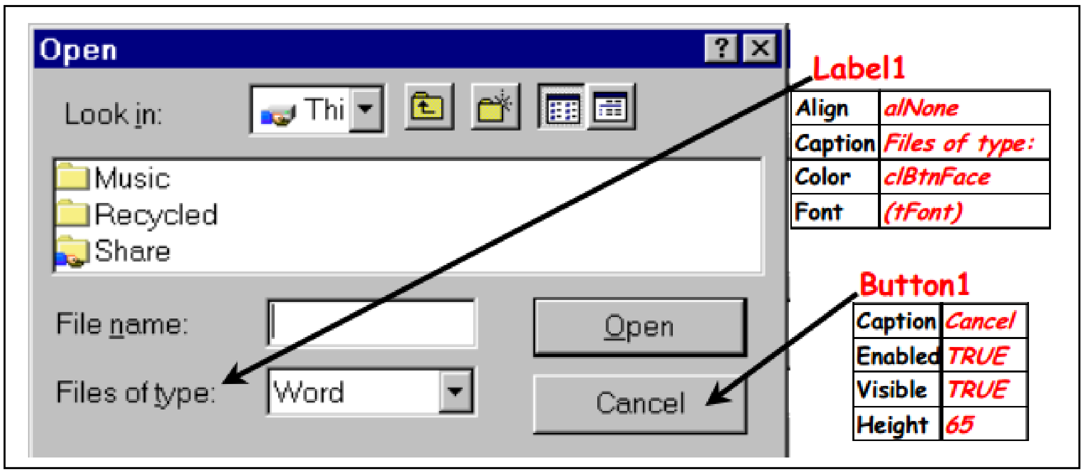
For the purpose of this proposed research topic, having an insight to the concept of “GUI Forest” is quite worthwhile. This is very first GUI portrayal obtained during the aforementioned ripping process. As the name implies, GUI forests will represent the structure of the windows of a GUI as nodes of the forest and the hierarchical relationship between each of the windows as edges. Enclosed in each node is the state of a window which comprises of the window’s widgets, their properties and the values. In other words, a GUI window is modeled as a set of widgets (e.g., buttons, labels, text fields) that creates the window, a set of properties (e.g., background color, size, and font) of these widgets, and a set of values (e.g., red, bold, 16pt) associated with the properties as each window will contain certain types of widgets with different properties. As a result of this, the window could be described in terms of specific widgets it possess at the moment of its execution along with the values of their various properties.

Symbolically, a window at time t can be modeled in terms of

- Widgets W = {w1,w2,...,wl}, i.e., the widgets that the window currently contains,
- Properties P = {p1,p2...,pm} of the widgets, and
- Values V = {v1,v2...,vn} of the properties.

For example, consider the Open window shown in Figure 1. Though this window contain several widgets, it has specifically named two widgets as Button1 and Label1. A small subset of properties for these two widgets were also shown. It is important to note that have a predefined set of properties where each of this property can take values in a predefined set again. The set of widgets and their properties will be used to create a model for the state of the window.

Thus, the state of a window at a particular time t is the set S of triples {(wi,pj,vk)}, where wi ∈ W , pj ∈ P ,and vk ∈ V .



Align	alNone
Caption	Files of type:
Color	clBtnFace
Font	(tFont)

Caption	Cancel
Enabled	TRUE
Visible	TRUE
Height	65

Figure 2.7.1 Open Window with Some Identified Widgets

Ultimately, to describe the complete state of a window, it has to contain information of the types of all the widgets that are currently in the window along with the values of the properties of each widget.

GUI's windows form a hierarchical structure where the user will be prompted with a top level window (or a set of windows) at the time of being invoked. Other windows of the GUI will be called via one of these top level windows or their descendants. Usually, these relationships among the windows can be illustrated through a set of Directed Acyclic Graphs (DAGs) as multiple windows can be used to invoke a single window.

Though a direct representation of the relationships can be obtained from DAGs, a tree model is quite simple to build algorithms which requires traversing across the nodes. This, each DAG will be converted to a tree by copying nodes. As most GUIs have only a single top level window, the forest can be simplified to a single tree.

The above process will be useful in identifying the components of GUI application, which I will store in separate location, so that it can be easily accessible, without digging into the code.

2.7.2 Techniques Available for Training Process of Component Identification

A Neural Network Based Approach for Modeling of Severity of Defects in Function Based Software Systems [17]

To predict the robustness of a software systems towards the possible faults, a lot of technical work is conducted. Yet, what matters most is the severity of the faults that could incur rather than the number of faults which exists in the developed system. This is because, major faults requires the immediate attention of the developers. Neural Networks (NN) have become a popular concept which is being used in the software engineering applications where those could be deployed to build reliability growth models.

NNs are non-linear modeling techniques which has the capability to model complex functions. At the times where the exact number of inputs and outputs cannot be determined, NN techniques can be used. A training process is used to learn the relationship between the input parameters and the output. This research has focused on exploring five key NN based techniques along with a comparative analysis as to the modeling of severity of faults which are present in function based software systems.

A NN can be considered as a network consisting of connected neurons. Similar to the behavior of neurons in human nerves system, information in NN will be propagated by firing electric pulses through its connections. NNs use the fact that the connection changes throughout the lifetime of a neuron and the amount of incoming pulses needed to activate a neuron also changes to perform the learning process. This way a NN can be trained to perform a certain function by adjusting the values of the connections between the neurons.

The researchers have used NASA's public domain defect dataset for the modeling purposes. Parameters such as Mean Absolute Error, Root Mean Square Error and Accuracy Values are used to perform the quantitative comparison among different algorithms.

The authors have tried to find a suitable algorithm to model software components into different levels of fault severity in software systems. For this, the following five Neural Network algorithms are experimented:

- Batch Gradient Descent
- Batch Gradient Descent with momentum
- Variable Learning Rate
- Variable Learning Rate training with momentum
- Resilient Backpropagation

The Resilient Backpropagation algorithm based NN has proven to be the best for the purpose of modeling the components of the software into different levels of severity. Through that way, the authors are suggesting that it can be used to identify the modules which contains most.

This research could be considered as a guideline to identify the characteristics of each algorithm which has the possibility to be deployed in the training mechanism of the proposed new research. Depending on the exact requirement, the algorithm to be used can be finalized.

2.7.3 Techniques to Add Enhanced Features to Test Automation Tools

A Study on an Intelligent General-Purpose Automated Software Testing Suit [4] shows that this manual software testing can be automated with the aid of an automated software testing scheduler and a log file error analyzer. This has enabled the testing engineer to keep track of the test process even from an outside location through a report generated during the process.

In this article, they have built an Automation Software tool, which will overcome the difficulties in manual testing. Also the time has been saved due to automating, repeated testing scenarios. So, that's the tester only will focus on the fail scenarios.

The process in Figure 2.7.2 will be executed in this approach.

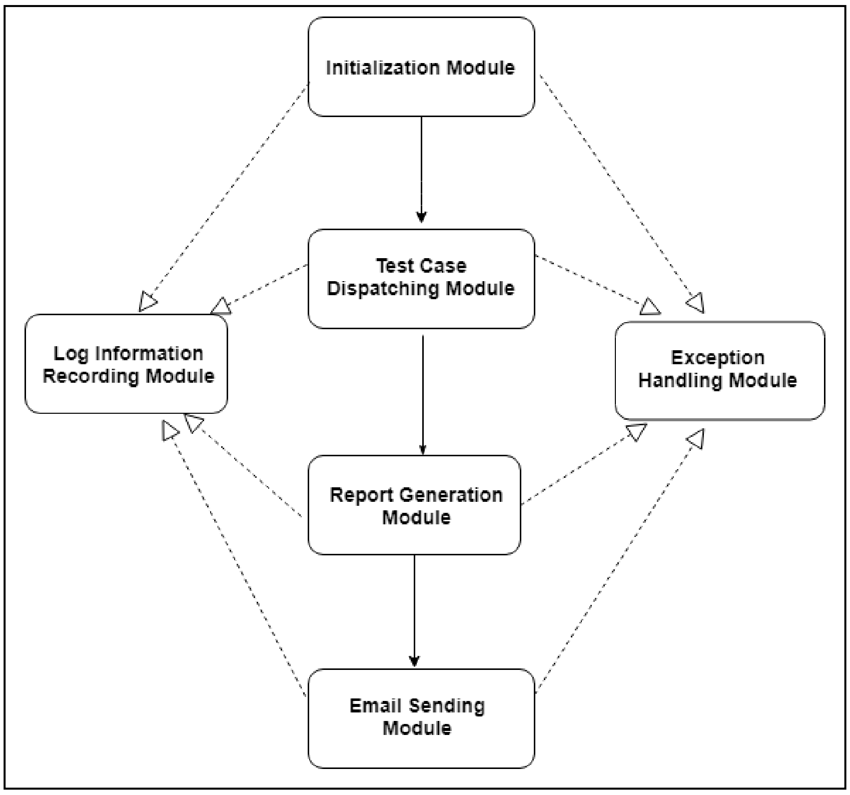

Initialization Module
Test Case Dispatching Module
Log Information Recording Module
Exception Handling Module
Report Generation Module
Email Sending Module

Figure 2.7.2 Process Involved

By using above methods, I can use to create the scripts, handle exceptions, log reports and organize the test case structure.

All these functionality and features help to increase the testing efficiency, reduce the labor intensity of software testing, which in turn, leads to the decrease in testing cost.

Another software testing automation framework [5] being published by IEEE suggests a multiplatform, multi-language approach for the reuse the services that can be used to automate major activities in the testing process. Here reusing has been identified as a potential solution to efficiently automate resource-intensive test suites and was actually tested in IBM.

Analysis of a software automation test protocol [6] also covers a deep analysis of software automation testing, and the definition, characteristics and functions which will be very useful in coming up with a suitable solution to the problem being addressed through this project.

2.8 Summary

This literature survey reviewed the available related work for the chosen research topic which was to develop an automated test script executing tool. It can be seen that there are several resources available which has incorporated things such as creating test scripts in their automated software testing tools, however, there are not any techniques to run a bundle of test scripts at one run and also to generate customized test reports. Even though these features are separately available in different tools, there is not a single tool which can achieve both the tasks. The literature review helped to identify the limitations of each tool available in the market which helped to lay the foundation as to what aspects should be covered in this new tool being developed.

Chapter 3: Analysis

Chapter 3. Analysis

3.1 Chapter Overview

This chapter will cover the in-depth analysis of the requirements for the proposed solution. The exact set of requirements were derived through the date gathered from the literature reviews, questionnaires and brainstorming sessions with the peers. This data was analyzed to get quantitative and qualitative information to build the functional and non-functional requirements. This chapter also presents a context diagram of the solution, and use case specifications as well.

3.2 Requirement Elicitation

When the process of software testing is considered, the test environment varies from one software product to another. As stated in 2.3.4, factors such as software, hardware and network configurations act as building blocks of the test environments. Thus, it is obvious that the scope of testing will differ from one test environment to another.

When building a new test automation tool, it is necessary, that the tool will be compatible with all the possible types of test environments. The performance of the tool should not be limited to one type of a test environment.

Requirement elicitation was carried out in order to understand the common aspects required for the tool to operate without any restrictions. Two main approaches were undertaken.

- Questionnaire – A standard of set of questions were developed to understand the aspects of the new automation tool being developed. This was shared among different people from different organizations so that an unbiased sample is taken into consideration.
- Brainstorming sessions – These were carried out with my co-workers in order to grasp any special requirements or customizations that should be considered in developing the new tool.

3.2.1 Literature Review

The purpose of the literature survey was to investigate the existing mechanisms available for software test automation. The pros and cons of each method were analyzed extensively through this review. This was helpful to understand the drawback of the existing mechanisms which should be taken into account when developing the new tool

Table 3.2.1: Pros and Cons of Literature Review for Requirement Elicitation

Advantages	• Available literature on the research area helps to understand the aspects which requires improvement or needs revision. • Proper documentation is available which is more reliable. • As a result, it provides the foundation for requirement elicitation and provide the path to understand what areas should be covered from other requirement elicitation techniques.
Disadvantages	• Very time consuming, as this requires searching for all the available resources such as online documents, journal papers, conference proceedings etc. • The effort required to review, analyze and extract necessary information is comparatively high as the required facts are not readily available.

3.2.2 Questionnaire

An online form with the questions mentioned in Table 3.2.2 was circulated among various members in the company. The purpose of this was to obtain a feedback about the functionality, performance and expectations related to test automation tools.

The questions were categorized in to three main segments.

- Basic questions – which are common to everyone despite the difference of roles
- Questions specific to each type of role
- Final common questions to cover other useful information which are not being directly asked through the questionnaire.

Table 3.2.2: Structure of the Questionnaire Prepared for the Research

Basic Questions	• What are you? (E.g: QA Lead, QA Engineer, Developer, Intern) • Do you work with only local clients or both local and foreign clients? • What is the tool you are using as the test automation tool? (Eg: MarathonITE, Sikuli, Selenium, Other or None) • If you are not using any of the automation tools, what is the main reason? (Eg: The software product is complex to undergo an automated testing process, product development frameworks and technologies do not support automation tools, Lack of knowledge on software automation tools and other (specify))
As a QA Lead	• Does your automation tool automatically generate reports based on the test outcomes? (Yes or No) • If yes, - What is the mechanism used to automatically generate the report? (Readily available in the automation tool, a customized plugin was developed by the company itself) - What are the types of reports being generated by the tool? (Eg: Test execution status, Error reports, Reports of time elapsed, History reports, and Other) - Rate your preference (how important it is) for each type of report from 1-5 (where 5 being the highest preference)
As a QA Engineer	• What is the average time do you spend on writing a long test script (scenario wise)? (Less than 20 mins, 20-40mins, more than 40 mins) • What is the average time do you spend on writing an atomic test script? (Less than 10 mins, 10-20mins, more than 20 mins) • How do you rate the convenience of writing long test scripts from the automation tool?(E.g:1-5 where 5 is the most convenient) • How do you rate the convenience of writing atomic test scripts from the automation tool?(E.g:1-5 where 5 is the most convenient)
As a Developer	• Out of the following test mechanisms what approach do you prefer or more convenient? - Writing unit tests (coding required) - Image based identification of test components for testing (code level access not required)

As an Intern	• How long did it take to familiarize with the test automation tool you are using? (Less than 1 week, 1-4 weeks, more than 4 weeks)
Common question (Last Comments)	• What are the drawbacks you notice in the current automation tool? (Which is not highlighted through this questionnaire) • What are the additional features you would like to see in a new automation tool?

Table 3.2.3: Pros and Cons of Questionnaires for Requirement Elicitation

Advantages	• Can obtain different perspectives of the same scenario in accordance with the role undertaken by each of the participant. • Relatively less time consuming. • Data can be easily processed to obtain both quantitative and qualitative information.
Disadvantages	• The reliability of the answers is not guaranteed as the responses can be more personalized than the general context. • When the answers are open ended, analysis becomes complex.

3.2.3 Brainstorming Sessions

Several brainstorming sessions were conducted with the participation of my co-workers at the work place. For each session about 3-4 people participated and the discussions on the requirement of more improved versions of test automation tools took place. The discussions also focused on more desired expectations of a QA engineer in contrast to the existing software test automation tools.

General questions covered in there sessions are as follows.

- What are the advantages and disadvantages of each software automation tool available in the industry from the perspective of a QA engineer?
- Are there any customization tools available and what are the restrictions?
- How can we improve the existing mechanisms in test automation?
- What are the technologies which will be more suitable to develop a new test automation tool?
- What could be different types of approaches to develop a new test automation tool?
- What could be the added benefits of a new test automation tool?
- What challenges could incur when developing the new test automation tool?

Table 3.2.4: Pros and Cons of Brainstorming Sessions for Requirement Elicitation

Advantages	• As the discussion happens in real time, it is easy to clarify any doubts on the answers. i.e. Ambiguity of answers can be eliminated. • Validity of the answers can be higher as the discussion points are subjected to further analysis. • Limitations and restrictions are easily understood.
Disadvantages	• Very time consuming, as the discussions are open ended. • As the number of participant increases, there is a possibility to compromise with the productivity. • As the audience requires to prepare in advance to give a more fruitful outcome, they might be reluctant to engage sometimes.

3.3 Feedback Response Analysis

3.3.1 Understanding the Background of the Job Role of Participants

Table 3.3.1: Number of Questionnaire Participants

Job Role	Count
QA Lead	11
QA Engineer	38
Developer	19
QA Intern	13
Total	81

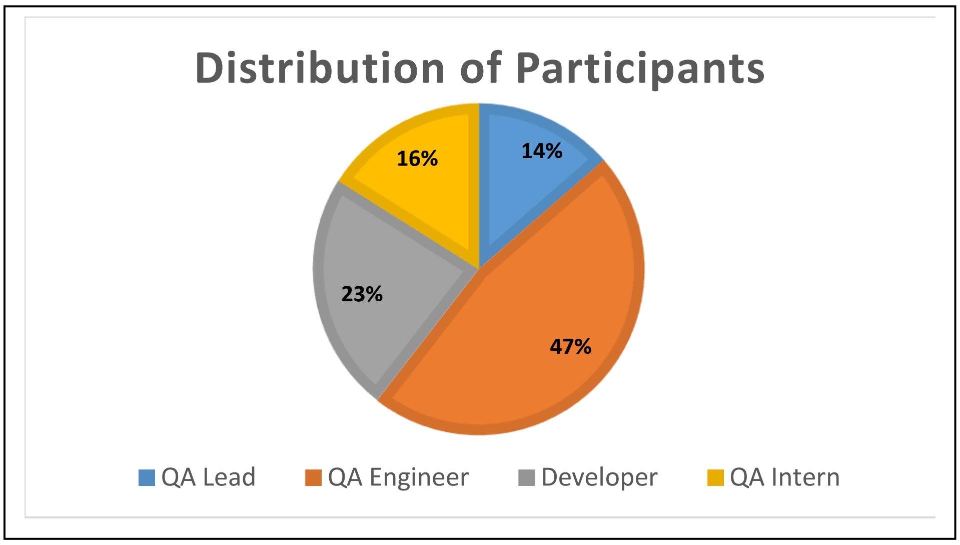


Figure 3.3.1: Pie Chart - Distribution of Questionnaire Participants

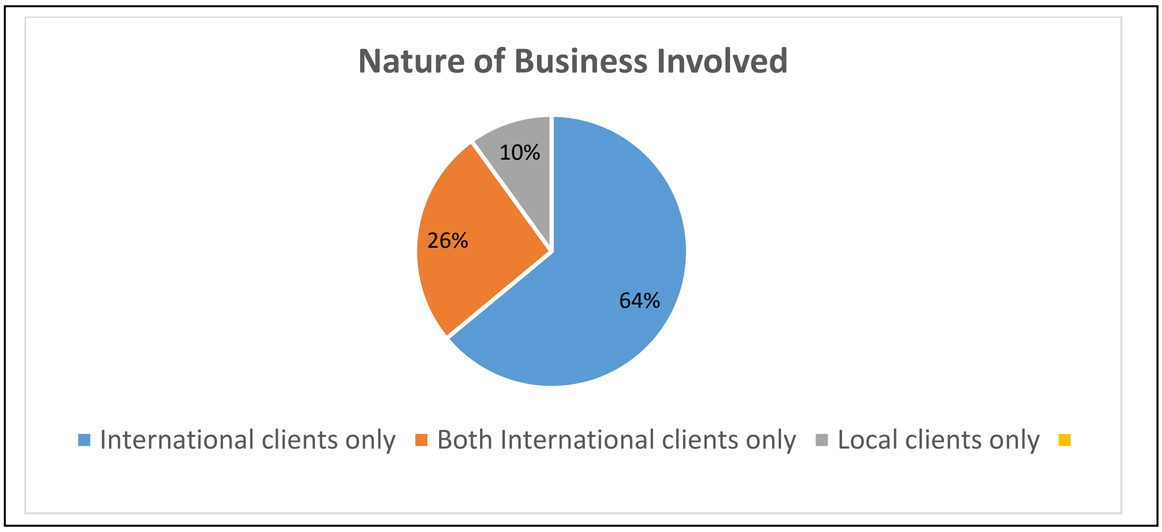


Figure 3.3.2: Pie Chart - Nature of Business Involved by the Participants

3.3.2 Analysis of Types of Automation Tools Used

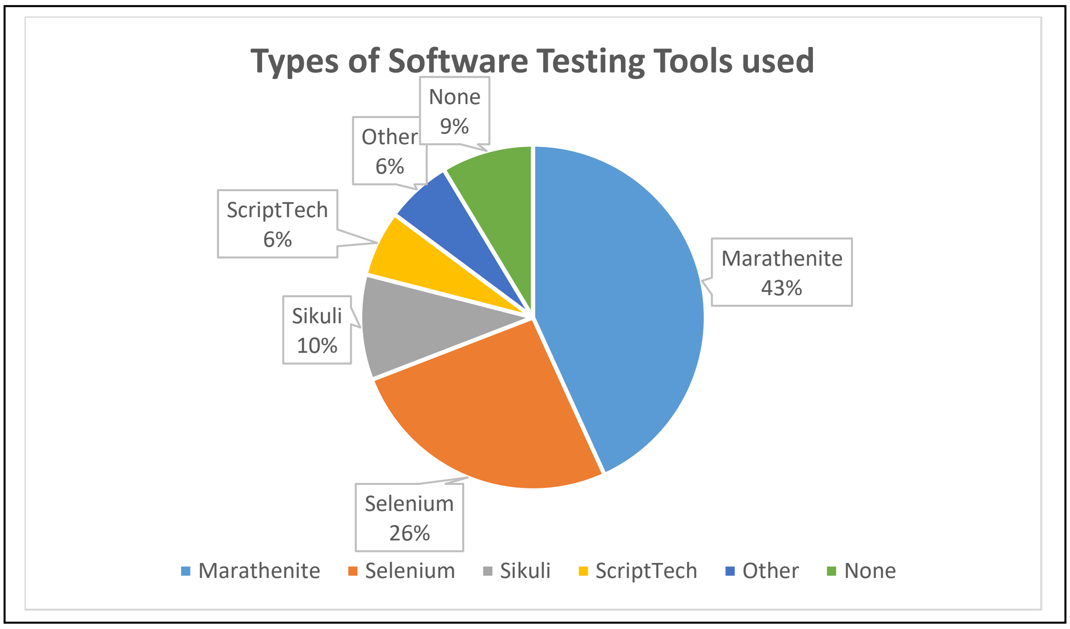


Figure 3.3.3: Pie Chart - Types of Software Testing Tools Used

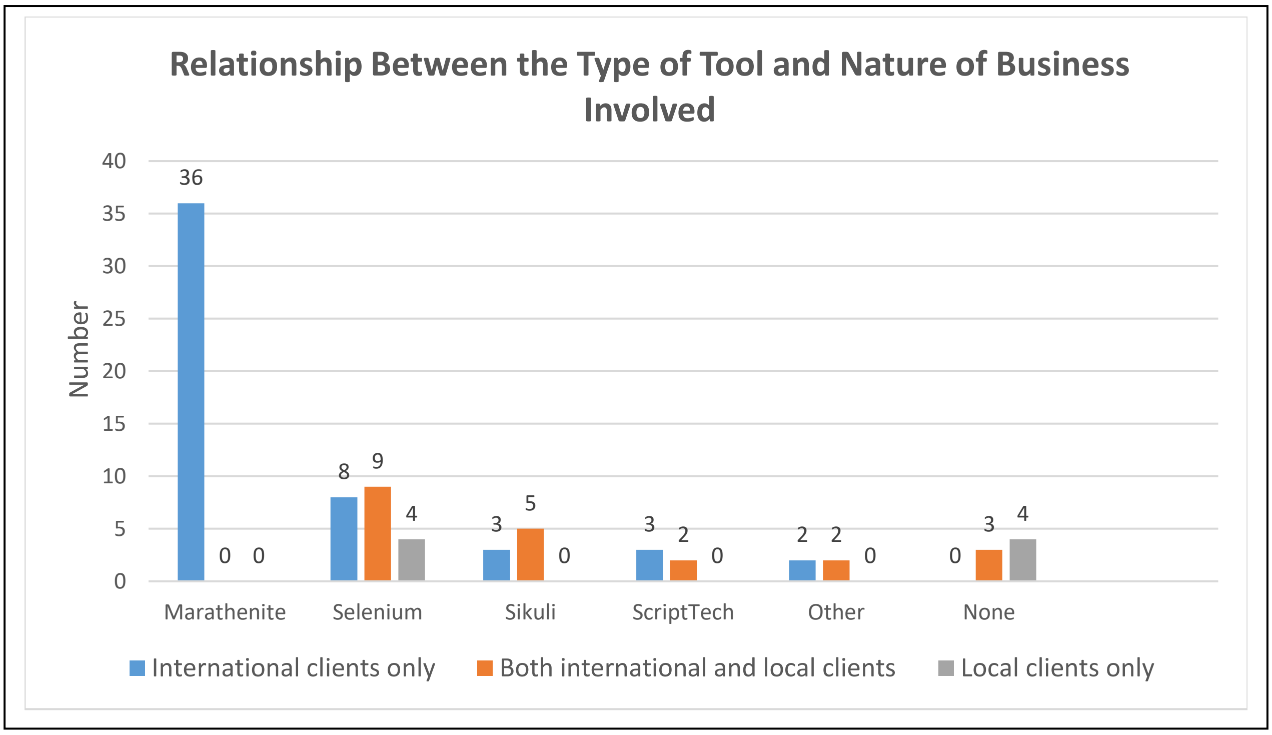


Figure 3.3.4: Bar Chart - Relationship between the Type of Tool and Nature of Business Involved

The purpose of Figure 3.3.4 was to understand the most common features a software testing tool should have for each type of business operation.

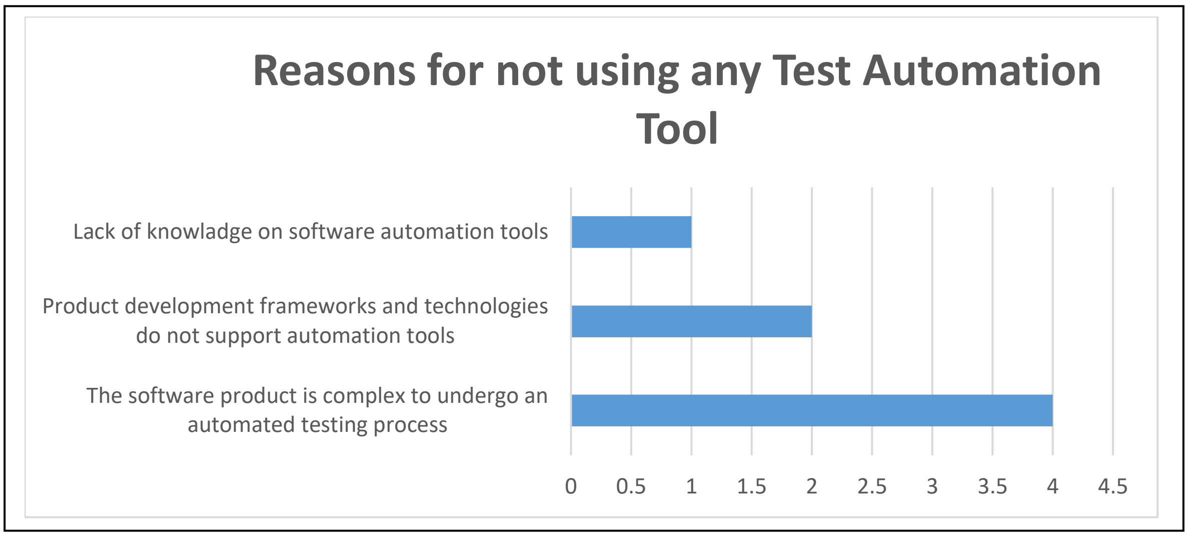


Figure 3.3.5: Bar Chart - Quantitative Representation of Reasons for not using any Test Automation Tool

Through Figure 3.3.5 it was easy to understand what improvements were expected in the new software test automation tool being developed through this project.

3.3.3 Understanding Different Requirements of Different Job Roles

This section summarizes the responses obtained from each of the job role in separately aggregated levels.

3.3.3.1 QA Lead

All the participated QA leads have indicated that there is a mechanism to generate the reports automatically in their testing tool being used.

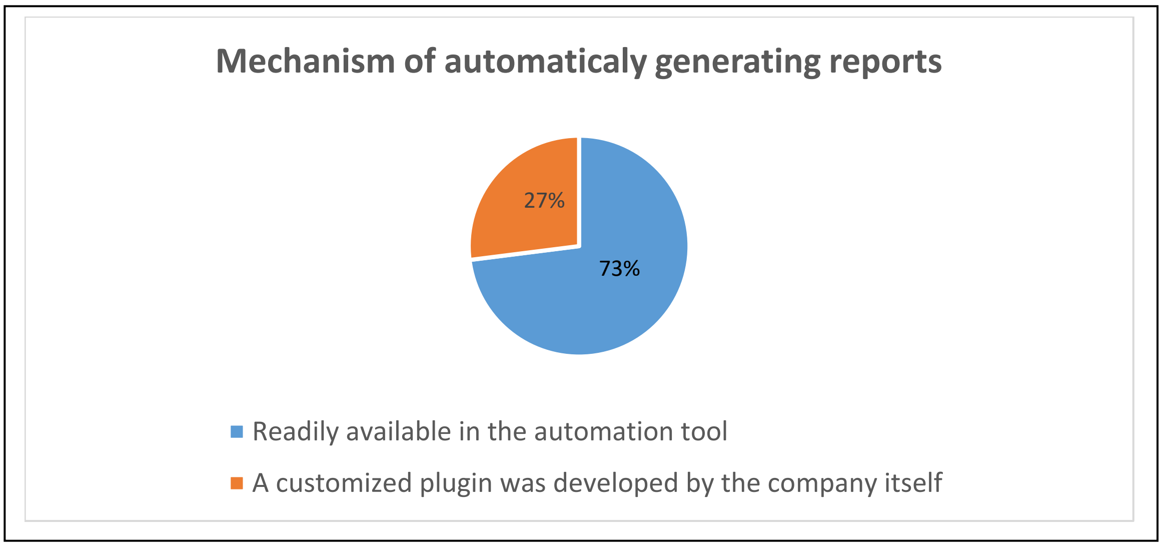


Figure 3.3.6: Pie Chart - Distribution of Mechanisms Used to Automatically Generate Reports

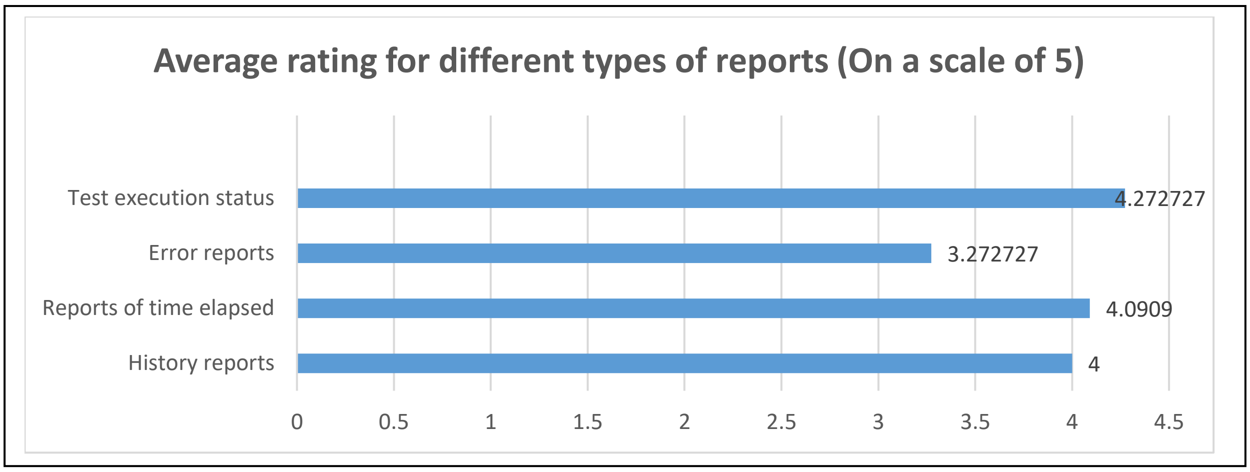


Figure 3.3.7: Bar Chart - Average Rating for Different Types of Reports Required by QA Leads

It can be seen that there is not much of a difference in the ratings for different types of reports. It can be concluded that every type of report is essential in general for a QA Lead.

3.3.3.2 QA Engineer

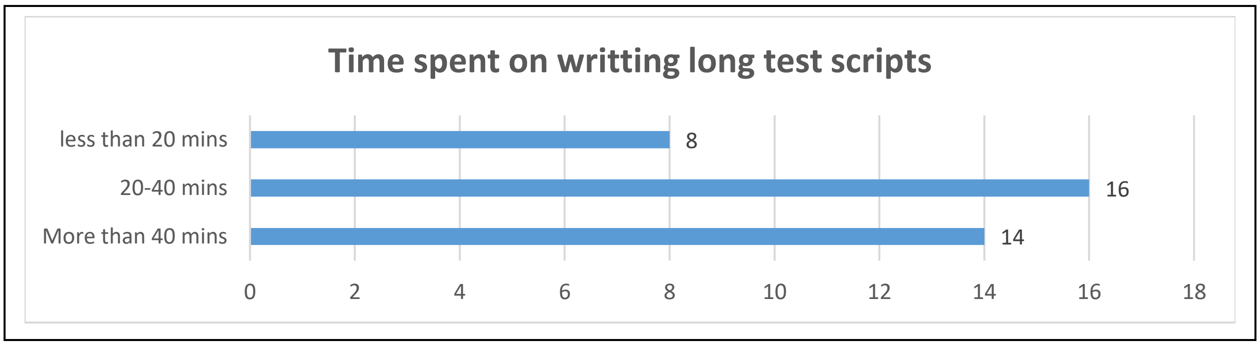


Figure 3.3.8: Bar Chart - Distribution of Time Spent on Writing Long Test Scripts

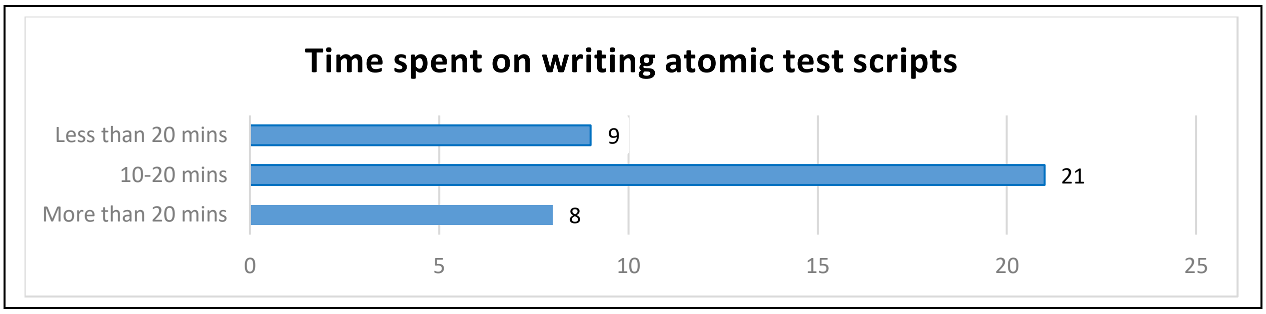


Figure 3.3.9: Bar Chart - Distribution of Time Spent on Writing Atomic Test Scripts

Table 3.3.2: Average Ratings for Convenience for Writing Different Test Scripts

Convenience for writing long test scripts	4.12/5.00
Convenience for writing atomic test scripts	3.65/5.00

From the above figures, it can be seen that on average, the time being spent to write a long test script will be 20-40 minutes, which is relatively high. Thus the new tool needed a feature to compile a long test script within a less time duration. Also according to Table 4.3.2, participants have indicated that they find it more convenient to write a long test script rather than an atomic test script. This could be incorporated to the new tool by the feature to create test flows.

3.3.3.3 Developer

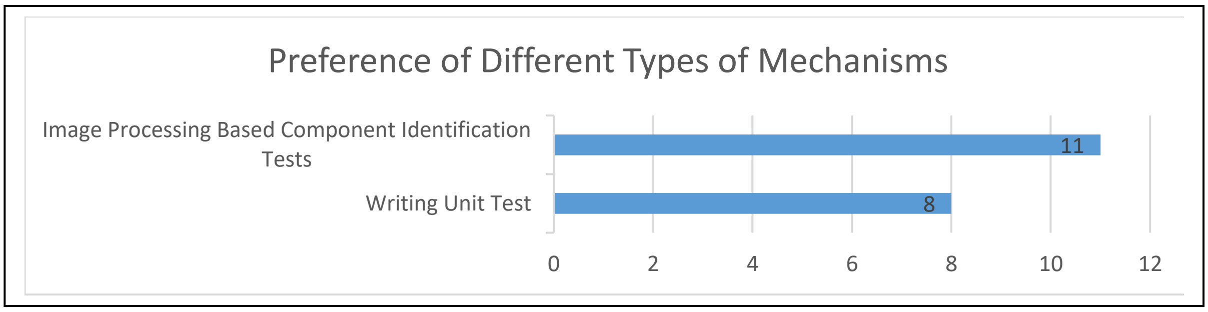


Figure 3.3.10: Bar Chart - Preference for Different Types of Test Mechanisms

According to Figure 3.3.10, a higher number of developers prefer an approach to eliminate the code level access requirement in the test mechanisms.

3.3.3.4 QA Intern

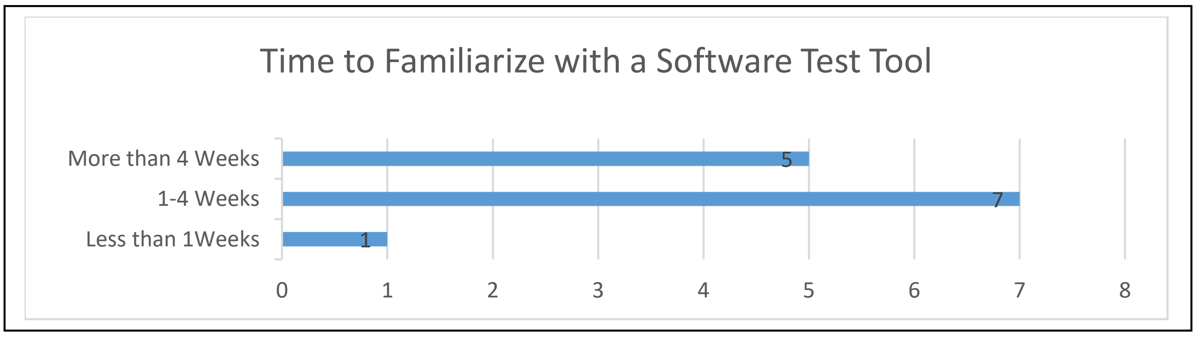


Figure 3.3.11: Time to Familiarize with a Software Test Tool

Figure 4.3.11 shows that average time for an intern to familiarize with a software automation tool is from one week to four weeks which is a considerable time period when compared to the total period of time he/she works in the particular company. Thus, the new tool should have a shorter learning period which also needs the feature of easiness to handle.

3.3.4. Feedback Responses Analysis for Additional Comments

Mostly stated drawbacks of currently used software test automation tools were,

- Need code level access to identify the test components
- Most tools required the company to write a framework to enable the compatibility between the software product and the test tool
- Complexity of the testing tool in terms of writing test scripts

3.3.5 Analyzing Brainstorming Feedback Responses

Brainstorming session was carried out with 14 QA Engineers at my workplace. The summary of the responses are as follows.

• What are the drawbacks of software automation tools in general from the perspective of a QA engineer?

- Incompatibilities between the software product and the test tool platform. QA teams may sometimes have to implement a framework on their own to fully support the system.

- Occurrence of complex test scenarios which cannot be automated easily using the existing tools.

• How can we improve the existing mechanisms in test automation?

- Most needed requirement was suggested as to come up with a tool which doesn't require code level access to identify GUI components.
- The new tool have to work well in multiple platforms (Eg: Windows, Linux, Mac, Android, etc.)

• What are the technologies which will be more suitable to develop a new test automation tool?

- Image processing, neural networks, Data mining

• What could be the added benefits of a new test automation tool?

- Ability to generate test reports in a wide range

• What challenges could incur when developing the new test automation tool?

- Making the new tool compatible with all the platforms (i.e. ensure that there is no need of a tailor-made framework).
- Not compromising the operating speed of the tool when the number of features added is increased.

3.4 Context Diagram

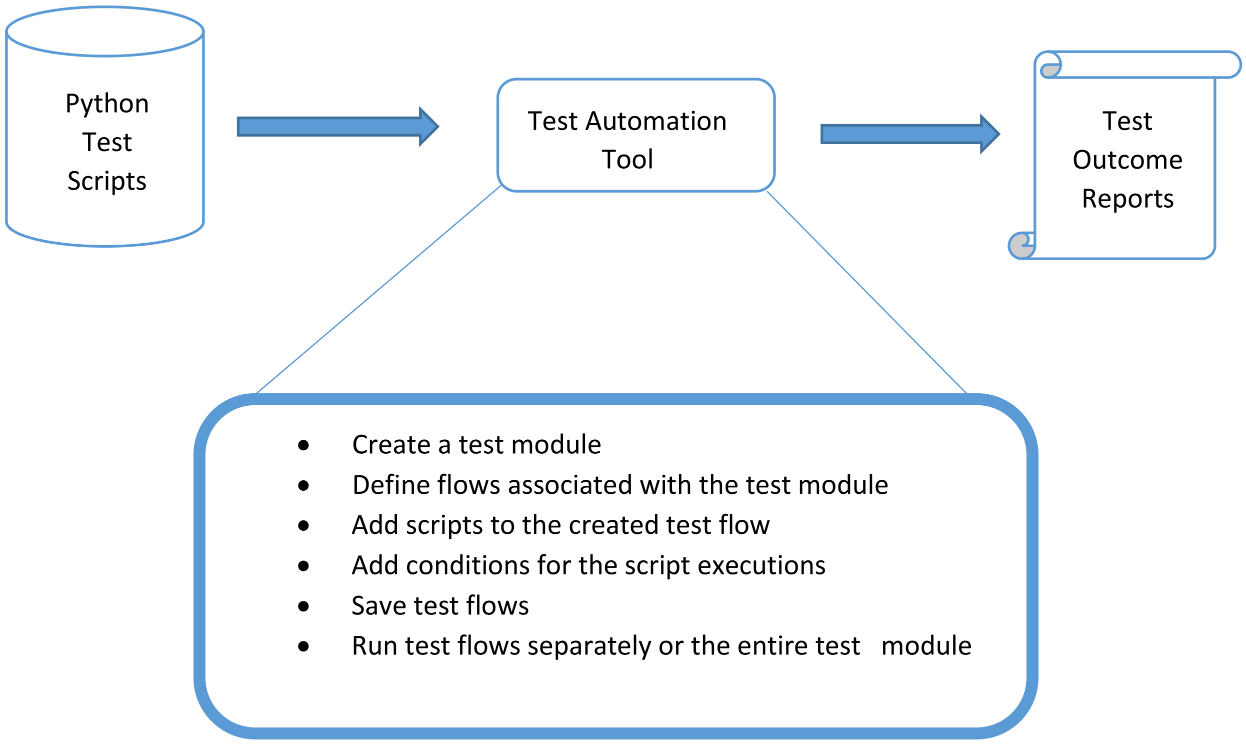

Python Test Scripts
Test Automation Tool
Test Outcome Reports
Create a test module
Define flows associated with the test module
Add scripts to the created test flow
Add conditions for the script executions
Save test flows
Run test flows separately or the entire test module

Figure 3.4.1: High Level Context Diagram of the Proposed Solution

A high level context diagram (Figure 4.4.1) could give a bird's eye view about the proposed test automation tool. Here, the proposed test automation tool will load the python test scripts generated by the image processing based component identification tool (Eg: Sikuli) and will perform the tasks mentioned in the diagram. More understanding of the test modules and test flows can be obtained from Figure 4.4.2. At the end of the testing process, necessary reports will be generated automatically based on the test outcome.

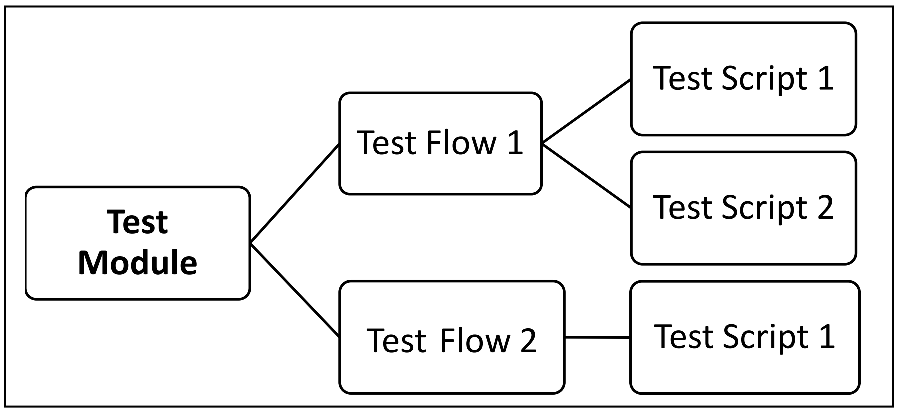


Figure 3.4.2: Structure of a Test Module

3.5 Use Case Model

3.5.1 Use Case Diagram

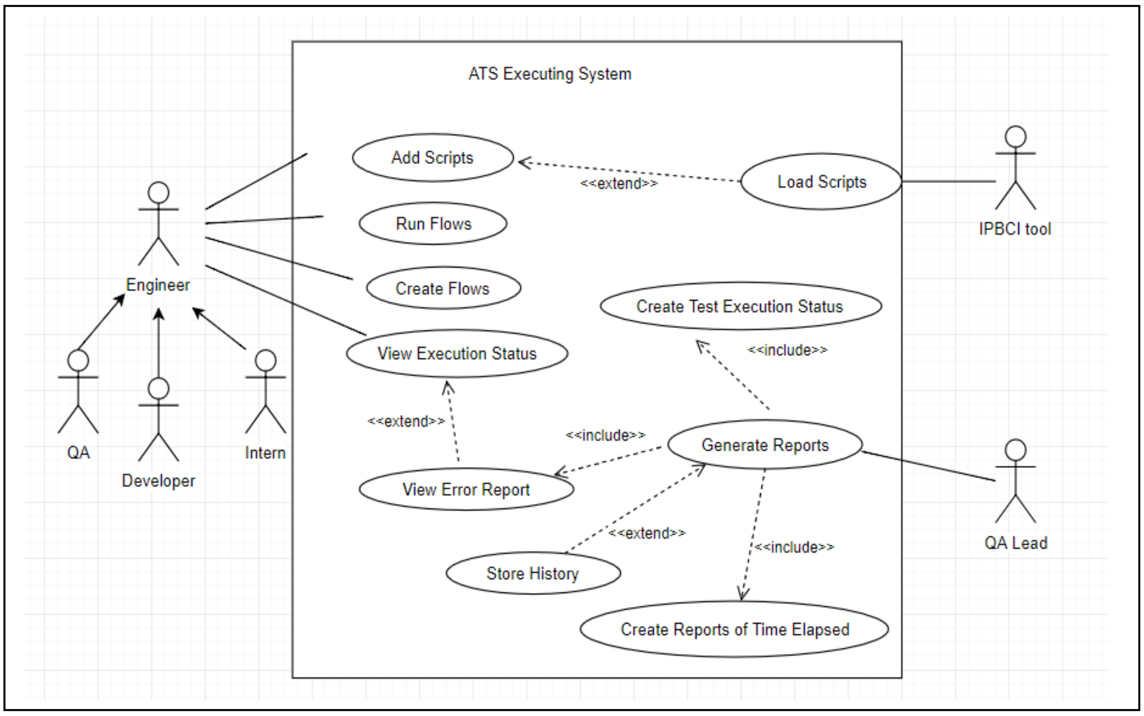


Figure 3.5.1: Use Case Diagram for the Automated Test Script Executing Tool

Figure 4.5.1 illustrates different types of users which could interact with the proposed automated test script executing tool. There are five actors involved in this use case diagram. They are, QA Engineer, Developer, the Intern (All three inherits from class Engineer), QA Lead, and the IPBCI tool.

Initially IPBCI tool will load the predefined test scripts to the system. Then the three types of engineers can add those scripts to test flows and execute these flows using the tool. They can also view the execution status of each test script being run. The QA lead, whose interest lies much on the reports generated by test outcomes for further analysis. Using these reports he can obtain information on test statuses, errors incurred, time elapsed for each test case and history details. Based on these information he can prepare other analytical documents to provide in-depth details on the testing process.

3.5.2 Use Case Specifications

Table 3.5.1: Use Case Specifications for 'Add Script' Task

Use Case	Add Script
Description	Tester load created test scripts
Actor	QA Engineer, Developer, Intern
Pre-condition	Created test scripts should be available
Extend Use case	Load Scripts
Main Flows	1. Create a module 2. Create a test flow to the module 3. Load test scripts 4. Use case terminate
Post Condition	Scripts are available to add to the flows

Table 3.5.2: Use Case Specifications for 'Create Flow' Task

Use Case	Create Flow
Description	Tester create flows by adding test scripts
Actor	QA Engineer, Developer, Intern
Pre-condition	Test scripts and module name should be available
Main Flows	1. Create a module 2. Create a test flow to the module 3. Load test scripts 4. Add test scripts to the test flow 5. Save test flow 6. Use case terminate
Exceptional flows	User try to load test scripts – Error message will pop-up if there are no test scripts available
Post Condition	Created test flows will be available

Table 3.5.3: Use Case Specifications for 'Run Flow' Task

Use Case	Run Flow
Description	Tester can execute the created flows
Actor	QA Engineer, Developer, Intern
Pre-condition	Already created test flows should be available for particular module
Main Flows	1. Select the module 2. Select the test flows, which needs to run a particular module

	3. Run the module 4. Use case terminate
Exceptional flows	User try to run the module – Error message will pop-up if there are no test flows available for the selected module, Error message will pop-up if there is a corrupted test scripts file
Post Condition	Execution results will be available for the tested module

Table 3.5.4: Use Case Specifications for 'View Execution Status' Task

Use Case	View Execution Status
Description	Tester can examine the test results
Actor	QA Engineer, Developer, Intern
Pre-condition	At least on module must have completed the test execution
Extend Use case	View Error Reports
Main Flows	1. Select the module 2. Select View Execution Status 3. Check the pass and fail test flows 4. Use case terminate
Exceptional flows	User try to run the module – Error message will pop-up if there was no test execution done for a particular module

Table 3.5.5: Use Case Specifications for 'Load Script' Task

Use Case	Load Scripts
Description	Load the test scripts, which was created from the IPBCI tool
Actor	QA Engineer, Developer, Intern
Pre-condition	Test Scripts should be available, which was created from IPBCI tool
Main Flows	1. Create test scripts from IPBCI tool 2. Store them on shared location which will be visible for the ATS System 3. Use case terminate
Post Condition	Test scripts will be available

Table 3.5.6: Use Case Specifications for 'Generate Report' Task

Use Case	Generate Reports
Description	Here the reports can be generated to the particular tested module
Actor	IPBCI tool
Pre-condition	There should be a tested module or modules available for particular environment
Include Use case	Create Test Execution Status, View Error Reports, Create Reports of Time Elapsed
Extend Use case	Store History

Main Flows	1. Select the module 2. Select which report is required 3. Use case terminate
Exceptional flows	User try to run the module – Error message will pop-up If there is no tested module or modules
Post Condition	Execution reports will be available

Table 3.5.7: Use Case Specifications for 'Create Test Execution Status' Task

Use Case	Create Test Execution Status
Description	Test execution status reports creator
Actor	QA Lead
Pre-condition	There should be module or module which has completed the test execution
Include Use case	Generate Reports
Main Flows	1. Select the module or modules 2. Select Create test execution report 3. Then select which statue you want, pass/fail/block/did not execute 4. Use case terminate
Exceptional flows	User try to run the module – Error message will pop-up If there is no tested module or modules
Post Condition	Execution results will be available for the tested module or modules

Table 3.5.8: Use Case Specifications for 'View Error Report' Task

Use Case	View Error Reports
Description	Test execution Error report state
Actor	QA Lead
Pre-condition	There should be module or modules which has completed the test execution
Include Use case	Generate Reports
Main Flows	1. Select the module or modules 2. Select View Error report 3. Then select which error report need to view 4. Use case terminate
Exceptional flows	User try to run the module – Error message will pop-up If there is no tested module or modules
Post Condition	Execution error results will be available for the tested module or modules

Table 3.5.9: Use Case Specifications for 'Create Reports for Time Elapsed' Task

Use Case	Create Reports of Time Elapsed
Description	Test execution Time elapsed state
Actor	QA Lead
Pre-condition	There should be module or modules which has completed the test execution

Include Use case	Generate Reports
Main Flows	1. Select the module or modules 2. Select Create Reports of Time Elapsed 3. Use case terminate
Exceptional flows	User try to run the module – Error message will pop-up If there is no tested module or modules
Post Condition	Execution Time Elapsed will be available for the tested module or modules

Table 3.5.10: Use Case Specifications for 'Store History' Task

Use Case	Store History
Description	Past test execution History can be viewed from here
Actor	QA Lead
Pre-condition	There should be module or modules which has completed the test execution
Include Use case	Generate Reports
Main Flows	1. Select the module or modules 2. Select History 3. Use case terminate
Exceptional flows	User try to run the module – Error message will pop-up If there is no tested module or modules
Post Condition	Execution History will be available for the tested module or modules

3.6 Functional Requirements

Time and resource constraints make it essential to prioritize the identified requirements found via the requirement elicitation process, in order to ensure the most critical requirements are given the highest attention. Table 4.6.1 depicts the definition of priority levels which will be used to categorize both functional and non-functional requirements.

Table 3.6.1: Priority Levels Definition

Priority level	Description
Critical	Mandatory to be implement and it's the core functionality of the system
Important	Considered as necessary even though it's not essential
Desirable	Need further development and out of scope

Table 3.6.2 shows the identified functional requirements along with their corresponding priority levels and the use case.

Table 3.6.2: Functional Requirements

Requirement description	Priority Level	Use case mapping
User should be able to load scripts to the ATS System	Critical	Add Script
User should be able to add scripts to a flow	Critical	Add Script
User should be able to create a new flow	Critical	Create Flow
User should able to run the created test flow	Critical	Run Flow
User should be able to run the created test module	Critical	Run Flow
User should be able to view the execution status	Critical	View Execution Status

User should be able to create reports related to Execution status of an environment	Critical	Create Test Execution
User should be able to view error reports which are reported in test execution	Important	View Error Reports
User should be able to keep a History record on tested environment	Important	Store History
User should be able to create reports for the time elapsed	Important	Create Reports of Time Elapsed
Integrate Sikuli IDE with ATS system	Desirable	Add Scripts

3.7 Non Functional Requirements

With the purpose of providing better user satisfaction, enhancing performance and maintainability, a set of non-functional requirements are defined for the project and they are also prioritized according to the importance of each.

Table 3.7.1: Non-Functional Requirements

Requirement description	Priority Level
Test execution should not be slow due to many test flow executions	Critical
This tool should be user friendly	Critical
Should work on any windows operating system above Windows XP	Important
Need to verify whether there are unnecessarily scripts getting load into the system	Desirable

3.8 Summary

This chapter presented about how the project scope would be redefined and restructured to incorporate the requirements of various stakeholders, namely, QA Lead, QA Engineer, Developer and Intern. Their expectations and information gathered from the literature review was very useful to define the final set of functional and non-functional requirements.

As per Chapter 2, it was identified that software test automation tools which could execute the test scripts in batches, and also generate customized reports with various test statistics was not available in the market. Even there are tools which could perform these tasks to a certain extent, it was still questionable, as the expected outcomes are limited due to various reasons.

This need to come up with a novel solution was further justified by the information processed via other requirement elicitation processes such as questionnaires and brainstorming sessions. The final model of the system is depicted in the form of a context diagram and the related users were modeled in the use case diagram.

Finally, with all the information gathered, functional and non-functional requirements were defined and they were prioritized based on the importance and the impact.

Chapter 4: Design

Chapter 4. Architecture and Design

4.1 Chapter Overview

Chapter 5 intends to provide an in-depth analysis over the proposed system architecture for the ATS tool. It starts with the presentation of the high level design of the solution via a rich picture and the high level architecture diagram. The purpose of the high level design would be to provide a bird's eye view to the system. Later on, the system design will be presented through a class diagram and sequence diagram with the descriptions and finally the overall design goals are stated.

4.2 High Level Design

The purpose of high level design is to provide the big picture of the proposed solution also incorporating the requirements gathered via requirement elicitation process. Here the overall solution is provided with more precise details. Later on, these designs can be improved with finer details in order to obtain the detailed steps involved to develop the project.

4.2.1 Prototype features

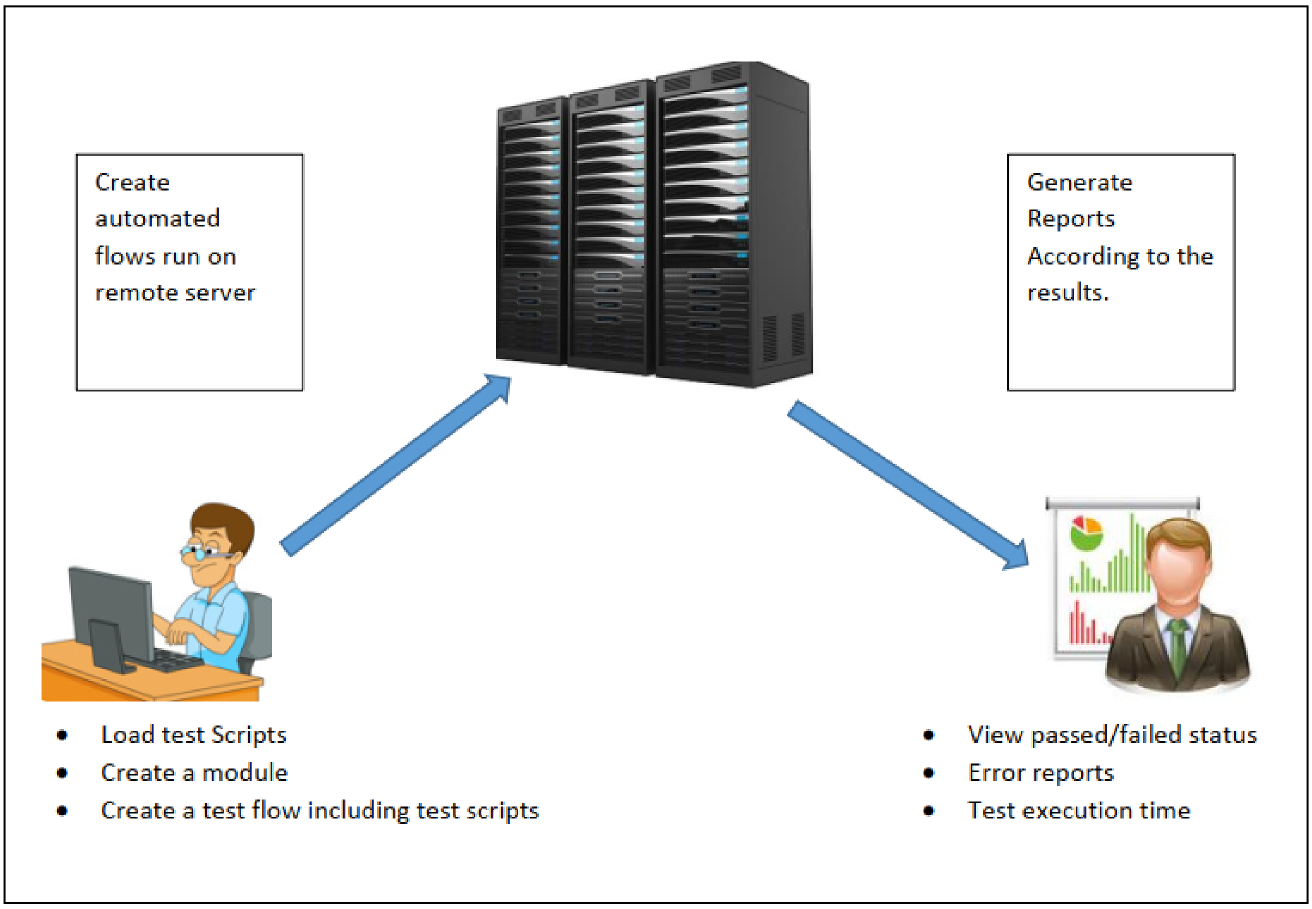

Create automated flows run on remote server
Generate Reports According to the results.
Load test Scripts
Create a module
Create a test flow including test scripts
View passed/failed status
Error reports
Test execution time

Figure 4.2.1 Porotype features

4.2.2 Rich Picture of the ATS Executing tool

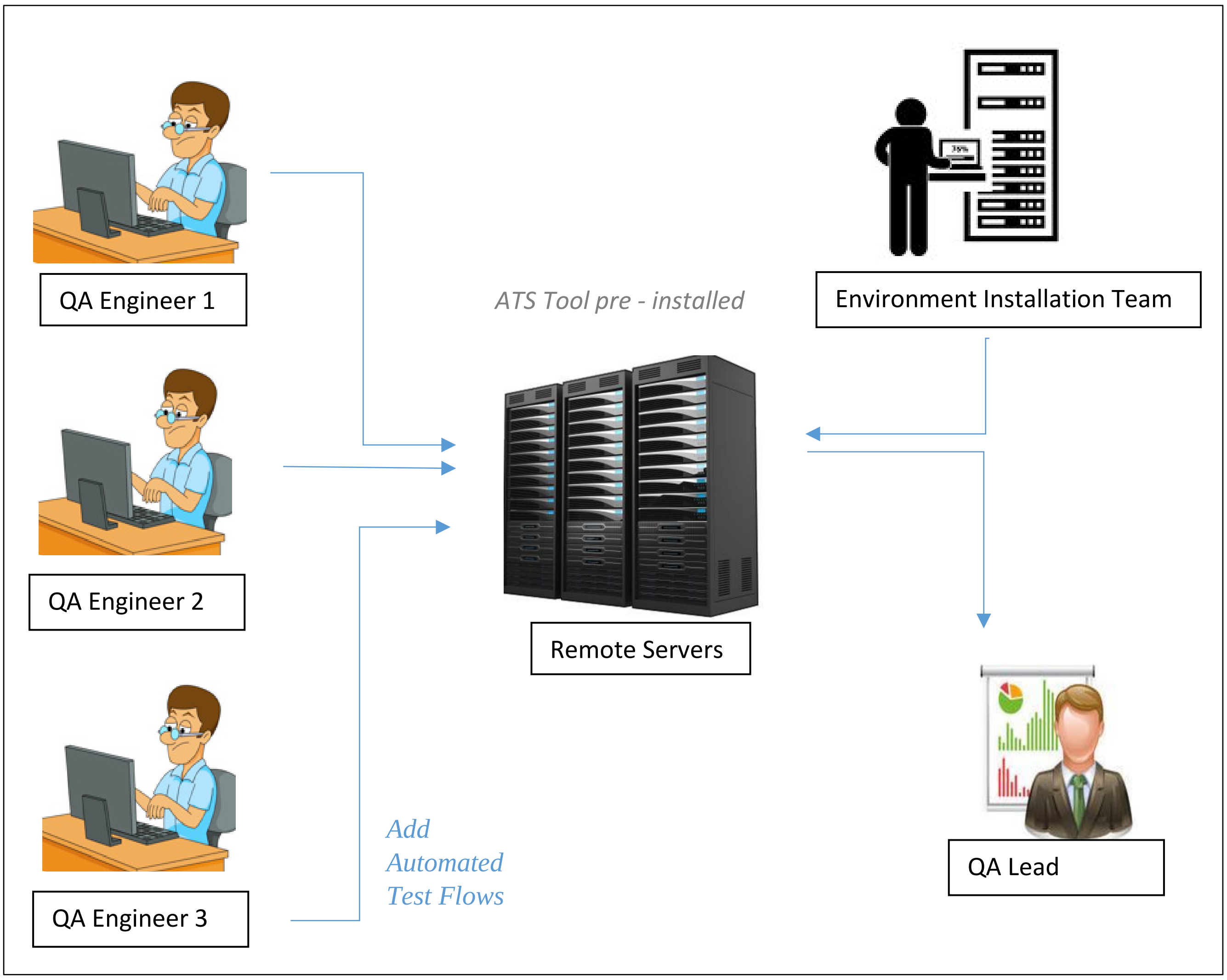


Figure 4.2.2: Rich Picture of the ATS System

Figure 4.2.2 depicts the rich picture of the proposed automated test script executing system. It displays how the tool is connected to the users and how the overall system would function. The rich picture will define the high level idea through a graphical representation, creating a very basic design model. Using a suitable design methodology, this could be expanded.

In brief, the new ATS tool will be initially installed to a centralized remote server, in which the environment installation team will set up the test environments to execute the test scenarios. Then the tool will create the test module and corresponding test flows and QA engineers could add test scripts accordingly. Several QA engineers could perform this task simultaneously. Once the test is over, test statistics will be presented in the form of test reports to the QA Leads.

4.2.3 High Level Architecture

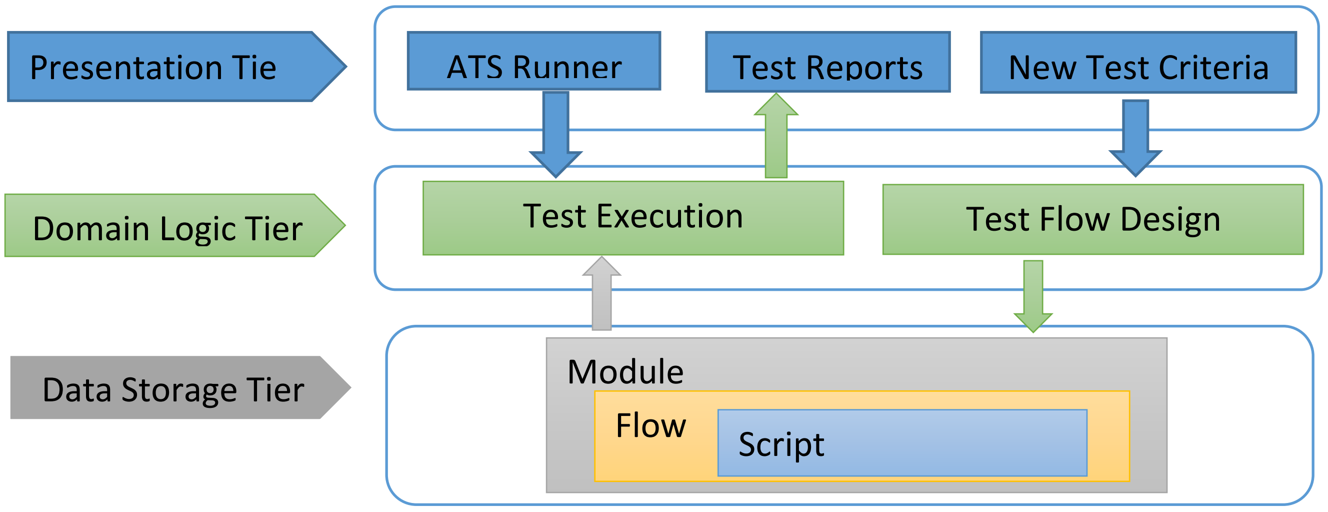


Figure 4.2.3: High Level Architecture Diagram

Three - Tier architecture is an industry certified software architecture model which support modelling of enterprise-level client/server applications. Main advantage of the three-tier architecture is to be upgraded or replaced independently by allowing any of the three tiers. Scalability, security and fault tolerance are capable in three-tier architecture. The high level architecture of the proposed system was modelled using the three-tier architecture due to the above mentioned reasons.

Presentation tire contains three modules that would be exposed to the outside world.

- **ATS Runner:** This module has the potential to select the modules for the test execution.
- **Test Reports:** This generates the test reports for the test execution.
- **New Test Criteria:** This module can define a new testing scenario/area.

The business logic of the system was carried out in the domain logic tier.

- **Test Execution Engine:** This module will load and execute the test scripts which are corresponding to the given test flow.
- **Test Flow Design:** This will design the flows by adding test scripts.

Data storage tier contains the storage procedure of the modules, flows and Scripts.

4.3 System Design

The set of tasks being followed from the beginning till the completion of a software development process if known as the design methodology. The chosen approach is chosen based on several factors like requirements of various stakeholders, type of software being developed (i.e. desktop application or mobile application etc.), the development environment, and time period of the project etc. In this project, the system design could be elaborated in two ways namely, class diagram, and the sequence diagram.

4.3.1 Class Diagram

The class diagram [21] is a static representation of the software application being developed. Apart from the visualization and documentation of the system, this class diagram could be used as the foundation to develop executable code of the software system. As class diagrams could be directly mapped with object oriented languages, it is widely being used in practice. It depicts the attributes and operations of a class as well as the different types of constraints, associations and collaborations of the system. Unlike to other UML diagrams which only shows the sequential flow of an application, class diagram serves a multiple number of purposes including describing the functionalities of a system, analyzing the static view of an application and also to be used as the underlying path in creating component and deployment diagrams.

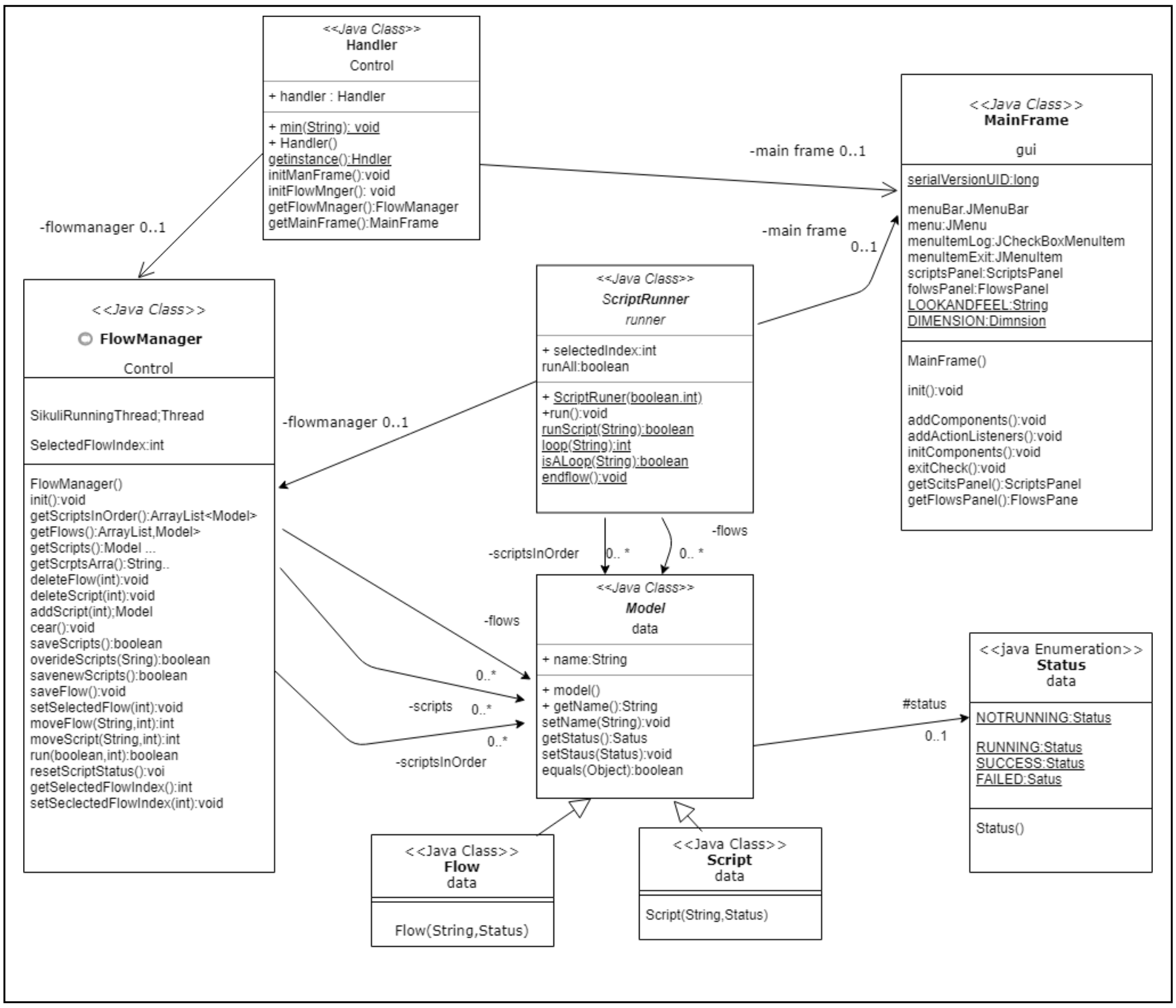


Figure 4.3.1: Class Diagram of ATS System

Classes shown in Figure 4.3.1 are the core classes of the ATS system which handle the main work flow of the system. Table 4.3.1 describes the purpose of each Class.

Table 4.3.1: Class Descriptions

Class	Description
FlowManager	This will focus on saving test scripts to a flow and the file storage functionality.

Handler	This will handle the object instance creating part of each class including threads.
ScriptRunner	This will focus on the running of a particular test flow or set of test flows (Module).
Model	This will create the structure of the data store of a test flow and load the scripts.
Main Frame	Main GUI (Eg: test flows list and test script list).
Status	This will give the execution result of a test flow or a set of test flows (Module).

4.3.2 Sequence Diagram

The sequence diagram (Figure 4.3.2) shows the sequence of actions taking place when a certain action is performed.

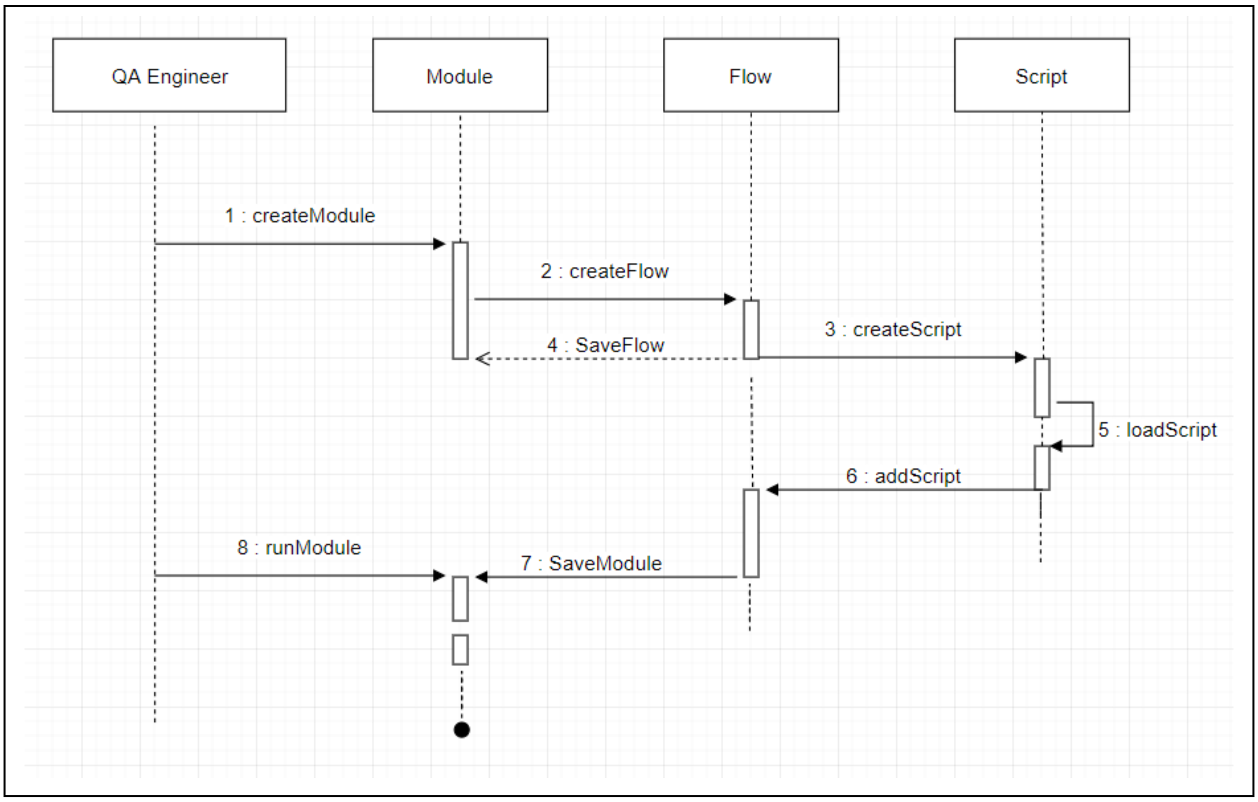


Figure 4.3.2: Sequence Diagram of ATS System

4.3.3 Script file and Flow file storage procedure

Entity Relationship diagram for module, flow and script file storage is as in the Figure 4.3.3.

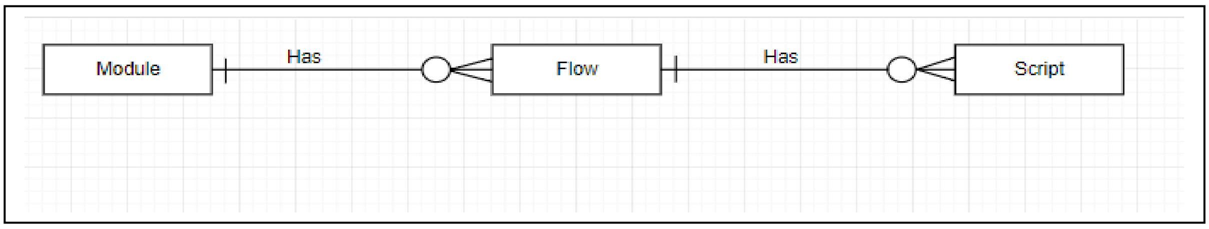


Figure 4.3.3 ER Diagram for file storage

Module to flow and flow to module has one to many entity relationship.

4.4 Design Goals for Overall Solution

Throughout the design phase, following aspects were paid keen attention as the design goal of the overall solution.

- Accuracy
  Accuracy of the overall solution is considered to be a very important design goal as this tool is being directly deployed for the QA activities in a company. As QA activities being very critical to ensure the correct functionality without any bugs, accuracy of the solution is considered crucial. Highest possible accuracy of all the related modules in the proposed architecture will ensure the accuracy of the outcome of the tool.

- Scalability
  Scalability in the design means that its ability to support expansions. As the scalability of the design is increased, it is easier to incorporate various changes and increase the number of users without compromising the performance and the functionality of the system. It gives more flexibility to the system.
  - Neatness
  The non-technical users. The neatness of the design also contributes towards the ability to easily debug the source code as well as the understandability.

- Portability

  System can be easily installed in several servers to run the automated scripts and could be executed in any test environment.

4.5 Chapter Summary

The high level design of the proposed solution is presented via a rich picture and a high level architecture which highlighted the basic functionality the ATS executing system. The system design was described using Class diagram, Entity Relationship diagram and Sequence diagram. The set of defined classes include FlowManager, Handler, ScriptRunner, Model, Main Frame, and Status. The actions happening in the system from creating test module to executing the module, is presented in the sequence diagram. The relationship between the three main entities related to the system; module, flow, and script is presented in the entity relationship diagram. Finally, the overall design goals such as accuracy, scalability, neatness, and portability were discussed with reference to the ATS tool.

Chapter 5: Implementation

Chapter 5. Implementation

5.1 Chapter Overview

Following the design phase, implementation will come as the next stage of the software development life cycle. This chapter discusses about the implementation of the proposed conceptual design of the solution. The chapter starts with the technology selection for the prototype development and then goes to describe the implementation process in detail. The corresponding code snippets and screenshots are included in order to describe the actual implementation details more clearly.

5.2 Technology Selection

There are so many different technologies that has been coincided and used to develop various types of automation tools. For this project I have selected the following list of technologies (Table 5.2.1).

Table 5.2.1: Technologies used

Technology	Description
Java	Java provides a system for develop software and deploying it in a cross platform computing environment.
IntelliJ IDEA	This IDE is used for development of this project. It is a java integrated development environment.
Python	Python is the scripting language used to write test steps
OpenCV	Template matching technology is used from OpenCV to identify the components of a given screenshot.
Sikuli API	Sikuli API was used to run the Python scripts.

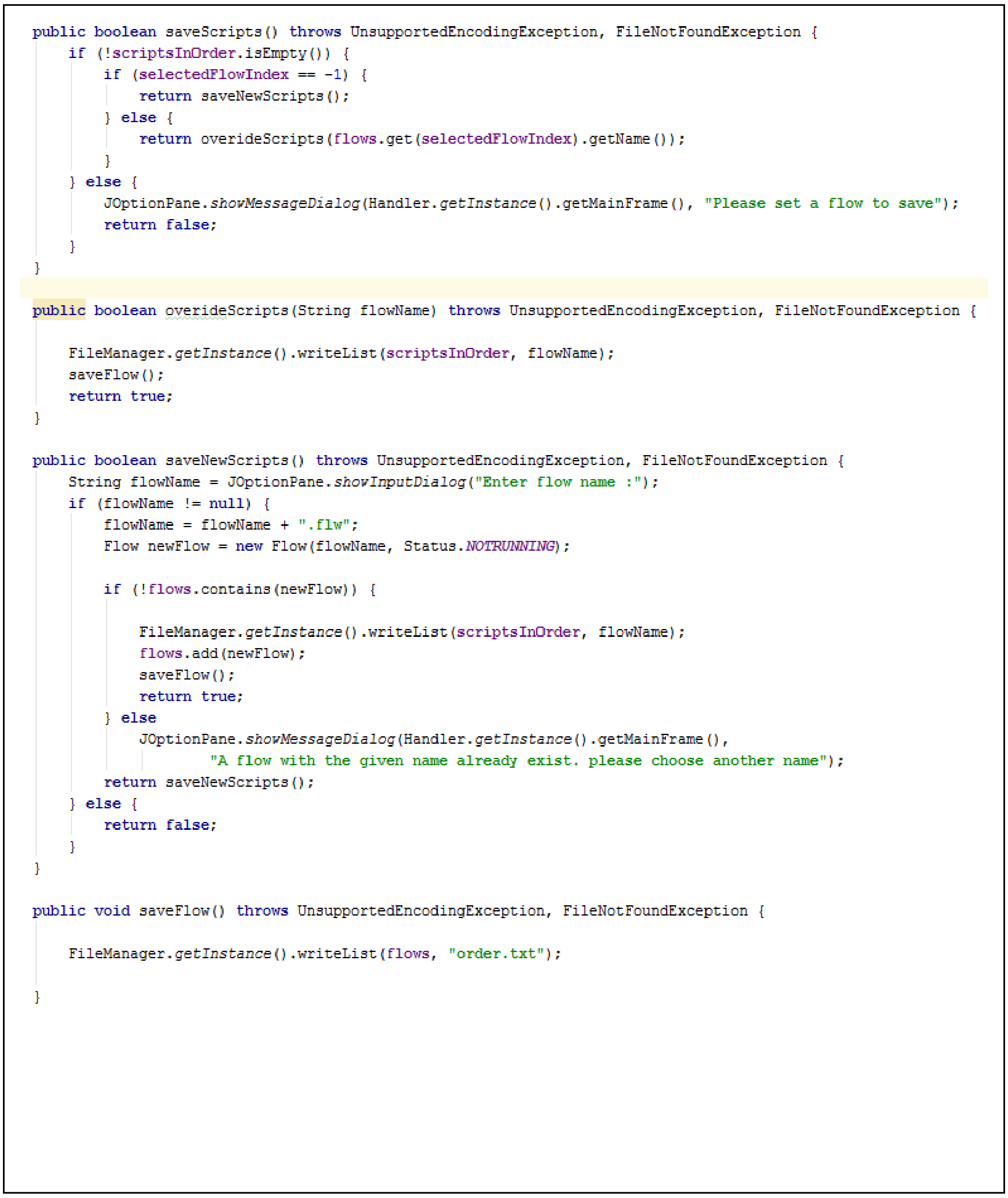
5.3 Core Implementation

This section outlines several important core functionalities of the proposed system.

5.3.1 Saving a New Test Script to a Test Flow

In the below Figure 5.3.1, the code implementation done for the execution of a test flow is shown. This is one of the most important areas where the already created test script file names are stored in an execution order so that this file can be later read for the execution purpose.

This saved test flow file also contains the conditions which have been set for a particular script (Eg: Repeated conditions like for loop, for, while and also If, else are define in this saved test flow).

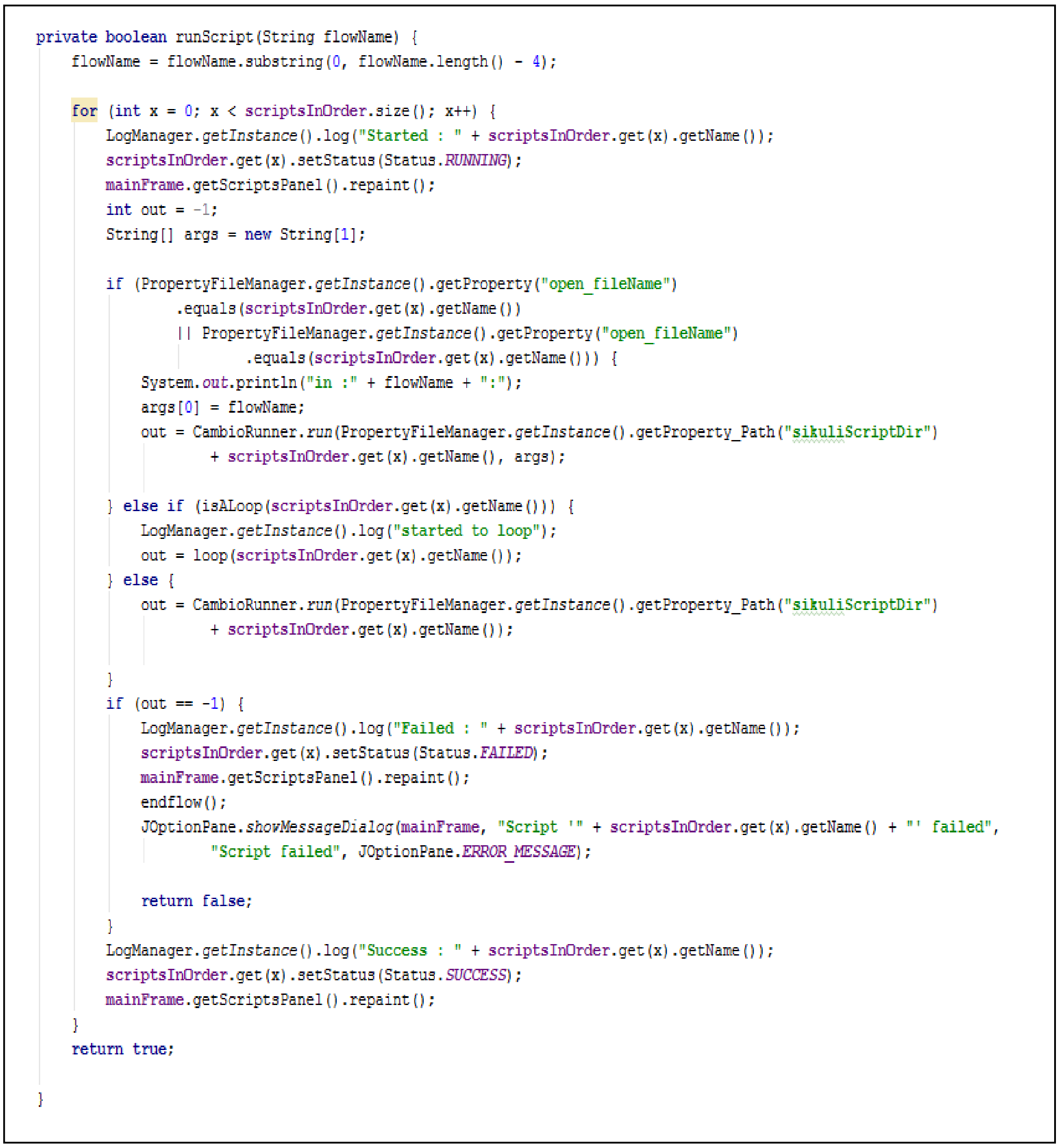


```java
public boolean saveScripts() throws UnsupportedEncodingException, FileNotFoundException {
    if (!scriptsInOrder.isEmpty()) {
        if (selectedFlowIndex == -1) {
            return saveNewScripts();
        } else {
            return overideScripts(flows.get(selectedFlowIndex).getName());
        }
    } else {
        JOptionPane.showMessageDialog(Handler.getInstance().getMainFrame(), "Please set a flow to save");
        return false;
    }
}

public boolean overideScripts(String flowName) throws UnsupportedEncodingException, FileNotFoundException {

    FileManager.getInstance().writeList(scriptsInOrder, flowName);
    saveFlow();
    return true;
}

public boolean saveNewScripts() throws UnsupportedEncodingException, FileNotFoundException {
    String flowName = JOptionPane.showInputDialog("Enter flow name :");
    if (flowName != null) {
        flowName = flowName + ".flw";
        Flow newFlow = new Flow(flowName, Status.NOTRUNNING);

        if (!flows.contains(newFlow)) {

            FileManager.getInstance().writeList(scriptsInOrder, flowName);
            flows.add(newFlow);
            saveFlow();
            return true;
        } else
            JOptionPane.showMessageDialog(Handler.getInstance().getMainFrame(),
                    "A flow with the given name already exist. please choose another name");
        return saveNewScripts();
    } else {
        return false;
    }
}

public void saveFlow() throws UnsupportedEncodingException, FileNotFoundException {

    FileManager.getInstance().writeList(flows, "order.txt");

}
```

Figure 5.3.1: Code - Saving a new test script to a test flow

5.3.2 Executing a Test Flow

Below Figure 6.3.2 shows the code implementation done for test flow creation. This is one of the most import areas where the already created test flow file is identified and execute the test scripts in an order which is stored in the test flow file. When a module is executed also, they will go through the flow names which are assigned to the module and execute the below code for the each test flow file by taking test flow file as an input.

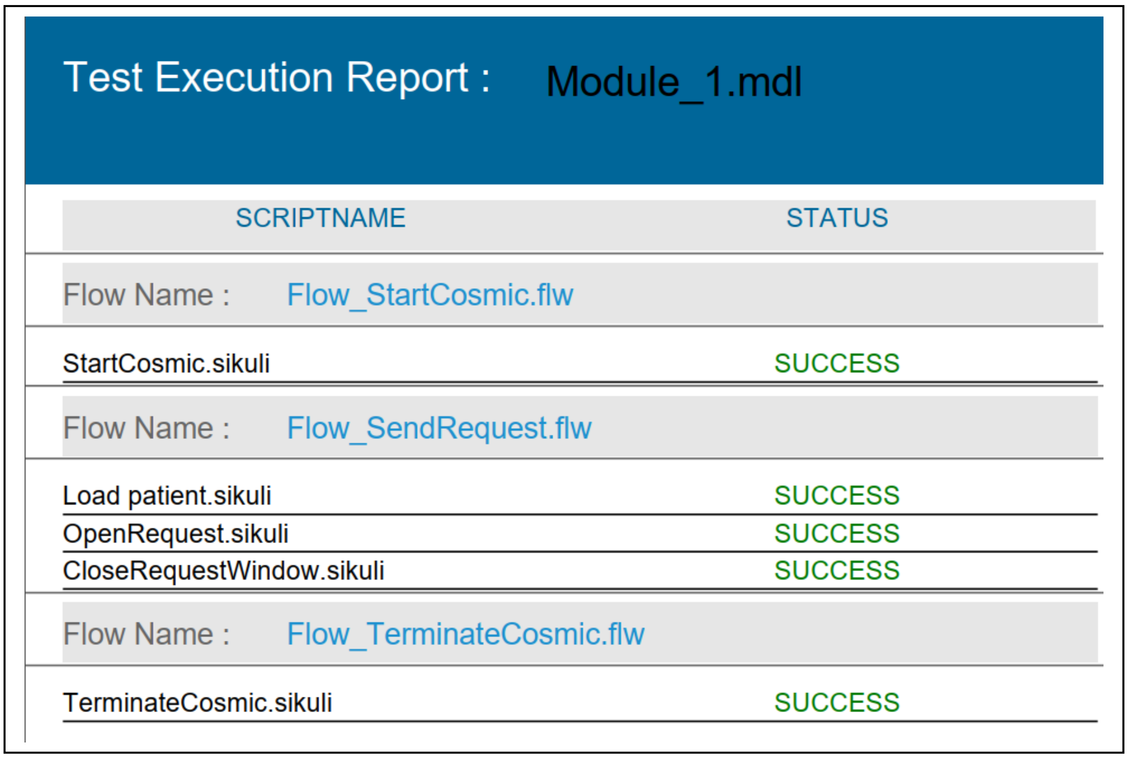


```java
private boolean runScript(String flowName) {
    flowName = flowName.substring(0, flowName.length() - 4);

    for (int x = 0; x < scriptsInOrder.size(); x++) {
        LogManager.getInstance().log("Started : " + scriptsInOrder.get(x).getName());
        scriptsInOrder.get(x).setStatus(Status.RUNNING);
        mainFrame.getScriptsPanel().repaint();
        int out = -1;
        String[] args = new String[1];

        if (PropertyFileManager.getInstance().getProperty("open_fileName")
                .equals(scriptsInOrder.get(x).getName())
                || PropertyFileManager.getInstance().getProperty("open_fileName")
                        .equals(scriptsInOrder.get(x).getName())) {
            System.out.println("in :" + flowName + ":");
            args[0] = flowName;
            out = CambioRunner.run(PropertyFileManager.getInstance().getProperty_Path("sikuliScriptDir")
                    + scriptsInOrder.get(x).getName(), args);

        } else if (isALoop(scriptsInOrder.get(x).getName())) {
            LogManager.getInstance().log("started to loop");
            out = loop(scriptsInOrder.get(x).getName());
        } else {
            out = CambioRunner.run(PropertyFileManager.getInstance().getProperty_Path("sikuliScriptDir")
                    + scriptsInOrder.get(x).getName());

        }
        if (out == -1) {
            LogManager.getInstance().log("Failed : " + scriptsInOrder.get(x).getName());
            scriptsInOrder.get(x).setStatus(Status.FAILED);
            mainFrame.getScriptsPanel().repaint();
            endflow();
            JOptionPane.showMessageDialog(mainFrame, "Script '" + scriptsInOrder.get(x).getName() + "' failed",
                    "Script failed", JOptionPane.ERROR_MESSAGE);

            return false;
        }
        LogManager.getInstance().log("Success : " + scriptsInOrder.get(x).getName());
        scriptsInOrder.get(x).setStatus(Status.SUCCESS);
        mainFrame.getScriptsPanel().repaint();
    }
    return true;

}
```

Figure 5.3.2: Code - Executing a test flow

5.3.3 Generate Log Reports of the Executions Status

In the implementation of the log files generator, first the system will generate xml file with the execution details. After that the system will generate a user friendly jasper report by reading the xml file content. This area is very important as it is the case where the status of the execution and the error logs are generated. Generated sample log report is shown below.

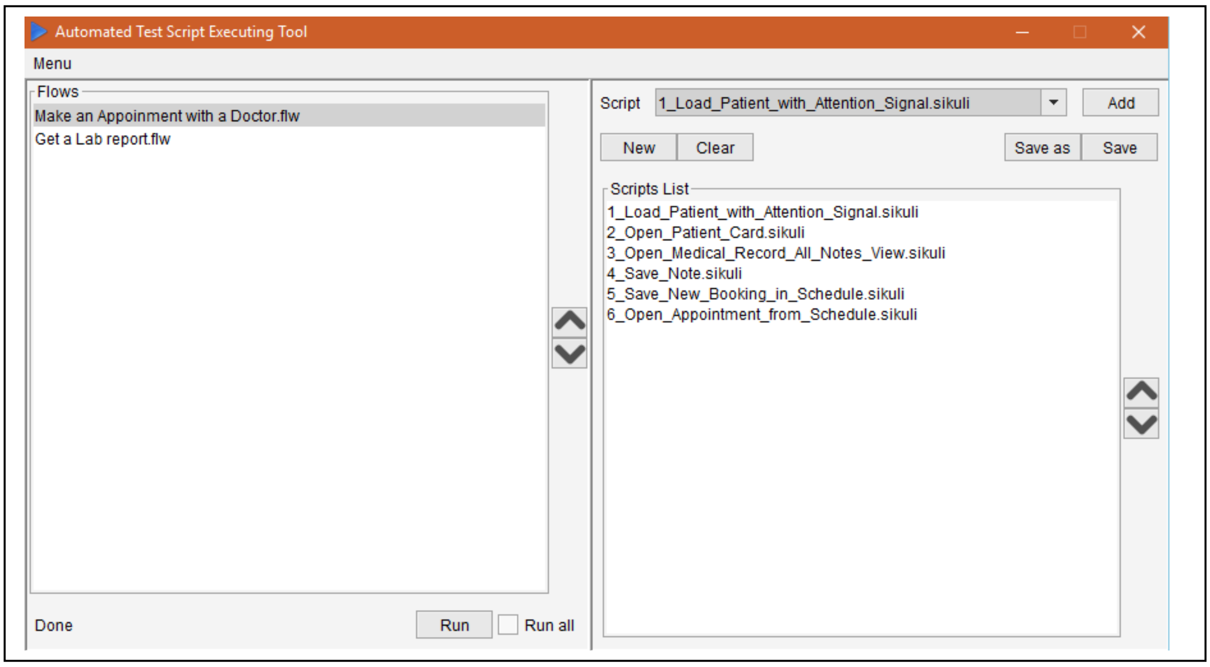


Test Execution Report : Module_1.mdl

SCRIPTNAME	STATUS
Flow Name : Flow_StartCosmic.flw	
StartCosmic.sikuli	SUCCESS
Flow Name : Flow_SendRequest.flw	
Load patient.sikuli	SUCCESS
OpenRequest.sikuli	SUCCESS
CloseRequestWindow.sikuli	SUCCESS
Flow Name : Flow_TerminateCosmic.flw	
TerminateCosmic.sikuli	SUCCESS

Figure 5.3.3: Generated log report for some sample test execution flows

5.4 Look and feel of the proposed ATS System



Automated Test Script Executing Tool

Menu

Flows
- Make an Appoinment with a Doctor.flw
- Get a Lab report.flw

Script: 1_Load_Patient_with_Attention_Signal.sikuli

Scripts List
1_Load_Patient_with_Attention_Signal.sikuli
2_Open_Patient_Card.sikuli
3_Open_Medical_Record_All_Notes_View.sikuli
4_Save_Note.sikuli
5_Save_New_Booking_in_Schedule.sikuli
6_Open_Appointment_from_Schedule.sikuli

Done

Figure 5.4.1 Look and feel of the proposed ATS system

5.5 Chapter Summary

This chapter discussed about the implementation details of the proposed ATS tool. First it begins with the technologies and tools selection with reasoning. JAVA, IntelliJ IDEA, Python, OpenCV, and Sikuli API are the selected set of development tools for this project. Implementation details of saving test scripts to a test flow, executing a test flow, and generating log reports based on the test statistics are given through snapshots of the codes. The chapter ends with the look and feel of the implemented system.

Chapter 6: Testing and Evaluation

Chapter 6. Testing and Evaluation

6.1 Chapter Overview

After the implementation stage of the proposed solution, the most important task is to perform the testing of both functional and non-functional requirements against the expected standard of level. The chapter will outline the testing criteria, testing methods, and finally, a thorough evaluation of the testing results in order to identify the remedies required in the future (if any).

In the testing process, to evaluate the performance of the system against the functional and the non-functional requirements was presented. Through this chapter, testing and the evaluation carried out by the external reviewers (mostly the different types of stakeholders) is presented along with their views and feedback given to the new ATS system. This also includes the self-evaluation carried out in order to identify the strengths and weaknesses of the project.

6.2 Objectives and Goals of Testing

In order to verify that the developed software system performs according to the defined requirements and produces the expected outcomes, software testing procedure is carried out. The key objectives of this process could be listed as:

- To verify and validate the functional and non-functional requirements of the system.
- To isolate the errors and defects of the system on order to revise those to get eliminated from.
- To ensure that final products contains no or negligible number of errors and bug.
- To keep enhancing and improving the product based on the test results.

6.3 Testing Criteria

When the implemented software system undergoes the testing process, it is the rigorous examination of the system to ensure that the pre-defined functional and non-functional requirements are satisfied and also, the quality expectations are met. To measure the software quality, there are two aspects being considered:

- Functional quality – This focuses on both the product development features and the technical requirements based on the pre-defined functional requirements.
- Non-functional quality – This is where the product is assessed against the fulfillment of the nonfunctional requirements defined for the product.

6.4 Functional Testing

The chosen software development methodology in <**referance**>**.** the Spiral Methodology, has the flexibility of conducting the testing tasks of the system being built in parallel with the product implementation phase. Thus, for this project, functional requirements of the Product were tested in the implementation stage itself which allowed more freedom to do necessary changes to eliminate the undesired results.

6.4.1 Testing Elicited Functional Requirements during Requirement Analysis

Table 6.4.1: Testing Elicited Functional Requirements during Requirement Analysis

Requirement	**Status**
Load scripts from IPBCI to the ATS System	Passed
Add test Script to a test flow	Passed
Create new test flow	Passed
Run created test flow	Passed

Run created test module	Passed
View execution history	Passed
Report creation for the execution history	Passed
Generate error reports	Passed
View history record	Passed
Generate report for the time elapsed	Passed

6.4.2 Testing Derived Functional Requirements during Implementation

It's not only the functional requirements which were finalized at the requirement analysis stage which is sufficient to guarantee the proper functionality of the developed solution. There are certain test cases which will be required for further evaluations with regard to the testing performance of the system and they are mentioned in Table 6.4.2.

Table 6.4.2: Testing Derived Functional Requirements during Implementation

Requirement	**Status**
All field during data entering has to be validated	Passed
Navigation path using keyboard(Tab navigations) should work fine	Passed

6.5 Integration Testing

As the name implies, integration testing involves integrating each module of the system to test as a complete unit. In this ATS system there are several modules which perform tasks like adding scripts to the test flows, executing test flows, and generating customized test reports etc. It is important that a testing mechanism is there to test the entire product at one run.

The system was tested as to whether it could navigate from one function to another without any errors. No error was found with regard to that. Also, it was tested whether the connection with the file storage works properly and it is also verified as correct. Thus, it can be concluded that the system successfully passed the integration testing as well.

6.6 Non Functional Testing

The Spiral methodology not only allows the functional requirements to be tested, but also the nonfunctional requirements as well. The following sub sections describes about the testing results on the non-functional requirements defined earlier.

6.6.1 User Friendliness

This system was tested with 15 different users (QA Engineers, QA Leads and Developers) to check how much ratings they will give on user friendliness.

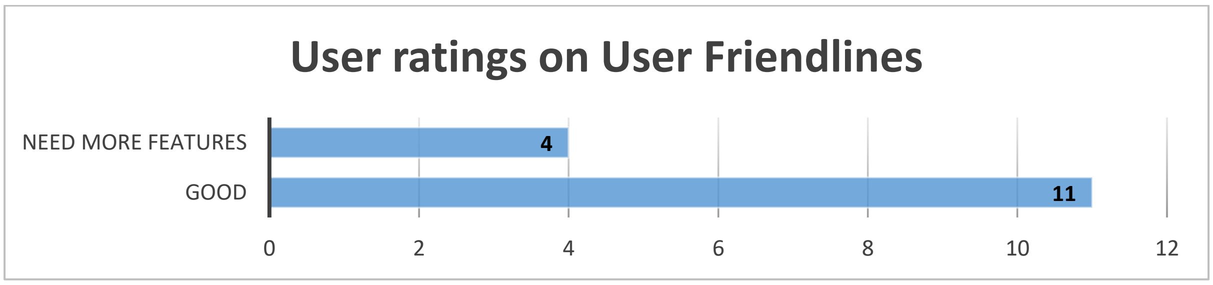


Figure 6.6.1: User Friendliness Test Results

From those 15 participants, some suggested that it will be easier for the users, if there is an option to drag and drop scripts and flows. And they also noted that it will look more supportive if there is more customization options for the reports like charts, fonts etc.

6.6.2 Performance

According to Table 7.6.1, the elapsed time to perform the tasks of loading the ATS tool, loading 50 scripts to the system, running a single test script of 5 steps and generating a module report was calculated. It can be seen that, the actual time spent on the relevant tasks are reasonable with the weight of workload.

Table 6.6.1: Performance Test Results

Tested Areas	**Actual Time**
Loading the ATS Tool	8.24 Sec
Loading test scripts to the ATS System(50 Scripts)	2.5 Sec
To run a single test script file(5 Steps)	1.3 Sec (Excluding the functional time)
To generate a module report	1.5 Sec

6.6.3 Load and Scalability Testing

To test the quality and the efficiency of a system in handling the load and the resources allocated is used to test the load and scalability of the system. There are many tools available in the market to do the load and scalability testing. The most popular tool is ApacheBenchmark [22] as it is much easier to present the test results. This tool was used to test the load and scalability of the ATS system. Core i7 processor of 3 GHz and 8GB of Ram and Windows 10 64bit version as the operating system.

Below are the results got from the ApacheBenchmark tool.

- Total data transferred is 871000 bytes for 1500 requests. It is around 50 test flows which has around 200 test scripts
- Test completed in 50.270s.
- Time per request was 14269.823 ms.
- Transfer rate is received as: 48.45 (Kbytes/sec).
- The conclusion of the load testing of the ATS system has the capability of handling at least 100 or more.

6.7 Limitations of testing process

6.7.1 Difficulty in Testing the System for Various Resolutions of Display

As this tool uses image processing techniques, the resolution of the display, plays a vital role in correct functioning of the tasks. However, from the testing perspective, it is quite difficult to test the system for different types of resolutions of the displays. Thus, the validity of the test results is limited by the different types of resolutions that underwent the testing process.

6.7.2 Not Tested on Enterprise Level Environment

As the testing could not be done in a company-level where there are other factors which could affect the performance of the ATS tool, there is a possibility that the testing process did not cover all the aspects of an enterprise level environment. There is room for several complications when it is deployed and tested in an enterprise level.

6.8 Evaluation Criteria

The criteria mentioned in Table 6.8.1 will be used as the base for the evaluation process. This will cover the main aspects of the ATS tool.

Table 6.8.1: Evaluation Criteria

Evaluator Category	Description
QA Engineers	QA engineers who are doing manual testing and automation testing
QA Lead	QA Leads who are reviewing the test flows and the execution status
Software Developer	Developers who are responsible of developing the tested product

6.9 Evaluation Methodology

Through this evaluation process, the extent to which the success has been achieved in the project could be understood from the feedback given by the evaluators in various domains. Feedback is provided to all the stages of the software product life cycle such as problem identification, requirement analysis, design and implementation etc. This project was evaluated in both quantitative and qualitative aspects.

The quantitative analysis was completed to a greater extent in the software testing phase of the project. Thus, in this section, qualitative aspect of the project is evaluated a lot. Evaluation of the ATS tool was carried out via a questionnaire (Table 8.4.1) and one-on-one interviews.

Table 6.9.1: Evaluation Survey Questions

No	Question
	User related Questions
1	How many years of experience do you have in testing and automation tools?(QA)
2	How do you see this ATS tool and Unit testing? (Developer)
	Overall comment about the concept and the complete project
3	What is you general idea about the ATS tool?
4	What would be the benefit of this solution would have on the selected user groups?
	Scope and depth of the project
5	Do you think the scope of the project is accepted for postgraduate level?
6	To which extend has this ATS tool addressed the problems identified?
	System design architecture and implementation
7	What is your feedback on design and architecture with regards to the project idea?
8	What are you suggestions on the design and the implementation?
	Solution and prototype
9	How do you rate the solution and prototype?
10	Do you think the system provides a solution to the identified problem?
	User friendliness and performance
11	How do you rate the user-friendliness and performance?
	Limitations of the Systems and future enhancements
12	What are the limitations of the proposed solution?

13	What are the possible improvement can be added to the ATS tool?

6.10 Evaluator Feedback

The information gathered through the evaluation questionnaire and other interviews are summarized and presented in this section. User related questions helped to support the credibility of the opinions provided.

6.10.1 Overall Concept

"***Though the idea seemed quite simple, the problem it has addressed is very critical and a very timely issue for most of the IT organizations. The solution has introduced a novel and feasible approach to overcome the issues."***

QA Lead – Cambio Software Engineering

"Out of box thinking to address the issues in Software test automation through techniques such as image processing. These techniques will definitely improve the performance of the test tools which will help to produce a more fruitful outcome for the testing processes."

Senior Software Engineer – Cambio Software Engineering

Table 6.10.1: Feedback and Review on Overall Concept

Feedback	The feedback suggested that the developed solution is a good approach in increasing the efficiency of the software test automation.
Review	Strongly agree with the feedback as this solution is intended to overcome the drawbacks of current tools in the market. The features of the solution helps to perform the tests much faster.

6.10.2 Scope and Depth

"The domain of test automation has a huge scope depending on its applications. It is clearly evident that this project has covered a significant portion of this problem domain with some advanced techniques. It would have been better, if the functionalities used in the supporting tool is integrated to the ATS tool as well."

QA Lead – Cambio Software Engineering

"There is a lot of room for improvement when it comes to test automation. The project has addressed some crucial and advanced problems in automated testing with aspects such as elimination of code level access, and not requiring a customized framework to support the tool."

QA Lead – Cambio Software Engineering

Table 6.10.2: Feedback and Review on Scope and Depth

Feedback	The feedback has recognized test automation as a very wide domain with lots of possible avenues for improvements. It is said that the aspects covered by this ATS tool are very significant. A suggestion to integrate the features of supporting tool to the ATS tool has been mentioned.
Review	The comments about the scope of test automation are very valid as there is no common tool to address all the expectations of QA engineers. The suggestion on integrating the functionalities of supporting tool to the ATS tool is a very good suggestion for future improvements.

6.10.3 Design and Implementation

"The decision to develop this project using JAVA technologies is a very wise decision as it supports cross-platform functionality as well as the ease of maintenance. Future improvement of the design could be that, this tool works fine for any resolution of the displays."

QA Lead – Cambio Software Engineering

"The approach to use OpenCV technologies seems quite visionary, as this project could be extended as an open source tool. Design approach to use 3-Tier architecture has increased the maintainability of the product."

Senior Software Engineer – Cambio Software Engineering

Table 6.10.3: Feedback and Review on Design and Implementation

Feedback	Use of Java and OpenCV technologies is highly appreciated due to various added benefits. 3-Tier architecture is recognized as a suitable approach. Suggestion to further improve the system to support any resolution is mentioned.
Review	As a result of 3-tier architecture, the design has the flexibility to change a part of the design without much hassle. Using JAVA is helpful as it can accommodate technology upgrades. Further research would help to improve the system to support different resolutions.

6.10.4 Solution and Prototype

"It is very interesting to see how this solution has integrated various useful functionalities of different QA tools into a single system."

Associate QA Lead – Cambio Software Engineering

"The prototype depicts that this solution will help to make the QA tasks more productive."

QA Engineer – Cambio Software Engineering

Table 6.10.4: Feedback and Review on Solution and Prototype

Feedback	The solution has included various desired aspects of available QA tools in the market to a single system which will help to perform software automations more effectively.
Review	Features such as no code level access needed to identify GUI components, generation of customized reports, ability to execute multiple test scripts in a single flow etc. contribute towards the enhanced efficiency of the testing process using this tool.

6.10.5 User-Friendliness and Performance

"Performance wise this tool is a big leap for test automation. Ability to execute test flows is very useful and the test reports which could be generated via this tool helps to analyze the test outcomes more rigorously."

Automation Tech Lead – Cambio Software Engineering

"The tool is easily understandable and very user-friendly with a simple GUI."

QA Engineer – ISM APAC

"Though it is quite easy to understand the functionalities of the tool, it would have been better if

there was a user manual and a help guide with all the instructions."

Intern-Cambio Software Engineer

Table 6.10.5: Feedback and Review on User-Friendliness and Performance

Feedback	GUI of the tool is very understandable which makes this more user-friendly. Performance of the ATS tool is highly appreciated in terms of time being taken to perform various tasks and accuracy of the results produced. Suggestion to produce a user manual is mentioned.
Review	The feedback could be justified because of the code reusability which helps to increase the performance and simple structure of GUI which makes the tool more user-friendly. If the product is commercialized, a user-manual can be created.

6.10.6 Limitations and Future Enhancements

"When executing the test flows, the execution server needs to have the same resolution all the time when it is writing the test flows."

Senior Software Engineer – Mubasher

"Currently the tool has the support for executing a single module. When it comes to the domain of team level, there can be multiple modules within a team CI environment. It could be better to have the support of run multiple modules at a time."

Senior QA Engineer – Cambio Software Engineering

Table 6.5.6: Feedback and Review on Limitations and Future Enhancements

Feedback	Limitation of need to have the same resolution is highlighted in several feedbacks. It was also suggested as a future improvement to extend this tool to run multiple modules at a time.
Review	Since the tool is still in the prototype level, this could be easily integrate with supporting of multiple modules. The tool's applicability would have been increased if it can support multiple modules as well which could be incorporated as a future enhancement.

6.11 Self-Evaluation

Table 8.6.1 presents the self-evaluation by myself about the ATS tool.

Table 6.11.1: Self-Evaluation

Criteria	**Description**
Overall concept and whole project	The concept of this project can be used to develop a promising business model because it has identified the problem areas in the current test automation domain and has provided a solution to overcome them.
Scope and depth of the project	Satisfied about the scope covered through this project as the field of test automation is a very wide subject area which keeps on expanding day by day.

System design and implementation	The design of the system was done with very much consideration to its adaptability to any system condition. In that case, the design supports different test environments. With the use of 3-tier architecture, it is easier to incorporate any changes without affecting the entire design. Elimination of the need to have code level access to identify GUI components is a significant achievement.
Solution and prototype	The solution has addressed several critical drawbacks of the existing test automation tools such as need of tailor-made frameworks to support the software product, inability to execute test flows etc. This solution provides a robust platform to make it compatible with any software product without any bridging framework.
User-friendliness and performance	Simple structure of GUI makes it easier for a user to grasp the basics of using the tool very easily. Code reusability significantly affects the increased performance of the ATS tool.
Limitations of the Systems and future enhancements	The issue of this tool not supporting all the resolutions due to the fact that it uses image processing techniques is a considerable limitation in this new solution. The ability to make the system independent of resolutions should be incorporated as a future enhancement. Furthermore, in the future, the functionalities of the supporting tool could be integrated to the ATS tool itself.

6.12 Evaluation of Functional Requirements

The functional requirements gathered during requirement analysis stage was presented to the evaluators in order to review on the level of completion (Table 6.7.1).

Table 6.12.1: Evaluation of Functional Requirements

Requirement	Status	Evaluator Feedback
Load scripts from IPBCI to the ATS System	Passed	Fully Achieved
Add test Script to a test flow	Passed	Fully Achieved
Create new test flow	Passed	Fully Achieved
Run created test flow	Passed	Fully Achieved
Run created test module	Passed	Fully Achieved
View execution history	Passed	Fully Achieved
Report creation for the execution history	Passed	Fully Achieved
Generate error reports	Passed	Fully Achieved
View history record	Passed	Fully Achieved
Generate report for the time elapsed	Passed	Fully Achieved
Being independent of screen resolution	-	Not Implemented
Integration of supporting tool functionalities	-	Not Implemented
Support web-based applications	Passed	Not Implemented

6.13 Chapter Summary

This chapter presented about the rigorous tastings that were conducted to the ATS tool in order to verify the correct functionality of the ATS system. After stating the objectives of this testing phase, the testing criteria was defined. This included the testing being conducted on both functional and nonfunctional aspects. Under functional testing, the functional requirements elicited during the requirement analysis phase and the functional requirements derived in the implementation phase were both tested. Then the non-functional testing was performed based on the performance, user-friendliness, and scalability. It was identified that all the tests passed the expected outcome and thus it can be concluded that the system is up to the standard and overall testing was concluded on a satisfactory level. The limitations of this testing mechanism was discussed after that under the testing mechanism, which included points of testing for various resolutions and not being able to test the system in the enterprise level. Next, briefly described about the evaluation methodology. Under that, how this tool was evaluated using external evaluators is presented. First, the evaluation criteria was defined with 6 aspects in consideration. The collection of feedback was conducted via questionnaires and face-to-face interviews. In brief, the project received a lot of positive responses where most of them appreciated the effort put on to integrate essential functionalities of the existing systems into a single system and making it more user-friendly and efficient to handle. At the same time, suggestions to extend this project into web-based applications and to integrate the supporting tool functionalities to the ATS tool were given as main possible improvements in the future. Finally, a self-evaluation of the project is presented.

Chapter 7: Conclusion

Chapter 7. Conclusion

7.1 Chapter Overview

The purpose of this chapter is to denote the closing remarks of the project where there is an analysis carried on to understand if the project aim and objectives were successfully achieved, a justification of learning outcomes, a highlight of the challenges faced and a set of future enhancements which will add value to the novelty of the ATS tool.

7.2 Achievement of Aims and Objectives

7.2.1 Aim

To design, develop and evaluate a software tool which can automate the execution of multiple software test scripts during one test cycle, using a readily available image processing based component identification tool. The new test tool will include several other enhanced features such as, generation of test reports, and ability to add the executing conditions to the script level etc.

It can be stated that the project aim was achieved successfully within the dedicated time frame and the developed prototype is critically evaluated in both quantitative and qualitative aspects via external evaluations of stakeholders and a self-evaluation process as well.

7.2.2 Objectives

This project involves the following set of objectives to be achieved during the project.

- Literature Survey
  Thoroughly analyzing the existing knowledge about the software test automation tools was very helpful in understanding the constraints in the existing tools and possible avenues for improvement. These sources helped to obtain a clear idea about the basics of test automation as well as how novel technologies could be utilized to enhance the performance of the existing test automation tools. The sources include, conference proceedings, online articles, and technical sessions. Chapter 2 of the report contains the complete literature survey conducted with related to this project.
- Requirement Analysis
  As a result of this, I was able to clearly understand, the exact requirements which should be addressed through this project. Questionnaire was completed with the participation of 81 people in the related domain, but with different job roles. Apart from that several brainstorming sessions were conducted with the peers. These activities gave an insight to the real world expectations in test automation. A proper set of functional and non-functional requirements were drafted at the end of this stage so that it will act as a set of guidelines in the next phases. Please refer Chapter 4 for the detailed presentation of requirement analysis.
- Design and Framework
  A successful design framework was established with the use of 3-tier architecture model, class diagram, sequence diagram, and entity relationship diagram etc. A foundation was laid through this to implement the prototype in real time. Chapter 5 will elaborate more about how the design of the entire system was created from simple illustrations to detailed designs.
- Selection of Development Tools
  Based on the design, complexity, and features of the proposed ATS tool, a suitable and feasible set of development tools was chosen (as in Chapter 6). The proper selection of development

tools and technologies was very helpful in optimizing the performance of the prototype and ensuring its functionality.

- Prototype Development
  The level of completion of the prototype is in a satisfactory level as it is capable of performing the tasks defined in the design phase. It was able to understand that the ATS tool prototype has overcome the said issues identified in the existing systems. Please refer Chapter 6 to get more insight to the implementation details.
- Testing of the prototype
  This testing is performed to get more of a quantitative perspective of the quality of the prototype developed. A certain amount of testing was carried out in the development stages as well due to the adopted development methodology (Spiral). Testing described in Chapter 7 describes about how each functional and non-function requirement (pre-defined and popped up during the development stage) is verified for its functionality.
- Critical Evaluation
  This critical evaluation of the prototype was carried out to obtain a qualitative analysis as to whether the functionalities of the prototype is achieved successfully. The feedback from QA engineers, QA leads, and software engineers was helpful to understand that this project has catered to an untouched area of test automation domain. This was further analyzed with a self-review of the project outcomes.
- Documentation
  Proper, clear, and descriptive documentation of each phase was carefully carried out, so that in the end every aspect the project is well documented. This also includes proper referencing to the original sources as well.

7.3 Utilization of Existing Knowledge

The knowledge I currently possess on design skills such as context diagrams and flow charts were very useful in converting the conceptual idea into the implementable software design during the design phase of this project. At the same time, my previous experience and knowledge in programming techniques and tools such as object oriented programming and JAVA technologies. My hands – on experience with software test automation was very instrumental in adding novelty to the ATS tool being developed.

7.4 Learning Outcomes

- Even though the knowledge gained through the MSc program with related to different aspects of software engineering was very useful, it was not very specific to the software test automation domain. Keeping the basic knowledge as a guide, I had to Self-learn most of the knowledge required to build this tool via online tutorials and documents etc.
- Several technologies used in the project such as, Python, OpenCV, Jasper Reports were not much familiar to me at the beginning of this project. However, with the base knowledge. I was able to polish the knowledge and expertise in these technologies which were extensively used throughout the research project.
- Essential soft skills for a technical career such as formal documentation skills and creativity were developed to a great extent as a result of hands on experience and learning.

It can be justified that the success of the project had contributions from all three segments discussed above; knowledge from the MSc curriculum, existing knowledge areas and self-learned knowledge and skills.

7.5 Problems and Challenges Faced

7.5.1 Lack of initial knowledge on key technologies used

As stated in 9.5, several technologies such as *Python* and *OpenCV* had to be learnt at the initial stages of the project. As this required additional work, there were certain instances where the original objective of the research project which was to build the ATS tool was diverted.

7.5.2 Lack of academic research publications on Software Test Automation

Though test automation is a vast knowledge area academic publications related to test automation was quite limited. This may be as a result of this topic is more prominent in the corporate level. However through rigorous search in other information sources such as online articles videos, I was able to overcome this challenge.

7.5.3 Wide scope of the research domain

Extensive scope of the domain of test automation became challenging when choosing the research topic. I had to analyze many possible research topics in the domain before finalizing what is most suitable as the project. The factors like importance and the significance of the problem being addressed, ability of maintaining the compatibility with existing tools and the time frame etc. were taken into consideration.

7.5.4 Time Constraint

This research project being subjected to a limited time frame, there was always a challenge to stick with the proposed timeline in order to ensure the timely completion of the project. Even with small deviations from the original timeline, due to facts such as revision of requirements at certain stages and the need to obtain knowledge on technologies being used, the project was completed on time.

7.6 Future Enhancements

7.6.1 Being independent of screen resolution

At the moment, when executing the automation scripts, resolution of the system should be as same as that was there during the creation of that automation script. This limitation could be overcome by making the tool independent of screen resolution. This requires further extensive research to be carried out in this domain.

7.6.2 Integration of supporting tool functionalities

As a further improvement, the supporting tool which creates the test scripts at the moment could be integrated to the ATS tool. It will make the test automation process easier.

7.6.3 Support web-based applications

As this tool was developed using Java swinging, this need to convert in to a web based tool, so that accessing the tool becomes more convenient.

These points are identified through the critical evaluation stated in Chapter 8. These feature could be implemented to the solution in the future as further enhancements. As much as this new ATS tool increase the productivity of the current tools, all these features will make the tool more suitable to be implemented in the corporate level as they will contribute towards overcoming more and more limitations in the existing test automation tools.

7.7 Contributions

This research project will help the Software development companies to save the time of the manual testing work and also this tool automates an application without going through the code to find component names. From the perspectives of increasing efficiency via time saving, reduction of complexity by no need of code-level access, and being compatible with present frameworks, this tool stands out from the existing software test automation tools. Thus it can be said that given the fact that this ATS tool is extended into the enterprise level, it will be a very popular solution in the test automation domain.

7.8 Closing Remarks

This time saving and user friendly test script automation tool opens up a new pathway for software test automation processes as it is addressing several crucial requirements such as elimination of code level access for GUI component identification, ability to execute a batch of test scripts at one run, generation of customized test reports, and elimination of the need to write a customized framework to enable the compatibility etc. This image processing based test automation tool with enhanced features can said to be a very good match to the corporate level expectations.

References

[1] S. Badle, J. Bakken and A. Barantsev, "Selenium," 2008-2012. [Online]. Available: http://www.seleniumhq.org/docs/02_selenium_ide.jsp#introduction.

[2] Jalian Systems Pvt. Ltd, "marathon," [Online]. Available: https://marathontesting.com/.

[3] R. Hocke, "sikuli," [Online]. Available: http://www.sikuli.org/.

[4] X. Wu and J. S. in, "The Study on an Intelligent General-Purpose Automated Software Testing Suite," in *International Conference on Intelligent Computation Technology and Automation (ICICTA)*, 2010.

[5] C. Rankin, "The Software Testing Automation Framework," *IBM Systems Journal,* pp. 126-139.

[6] X. Meng, "Analysis of software automation test protocol," in *International Conference on Electronic and Mechanical Engineering and Information Technology (EMEIT)*, 2011.

[7] TECH BLOG, "Royal pingdom," 19 03 2009. [Online]. Available: http://royal.pingdom.com/2009/03/19/10-historical-software-bugs-with-extreme-consequences/.

[8] P. S. Rogerson, "The Chinook Helicopter," *ETHIcol in the IMIS,* vol. 12, p. 4, 2002.

[9] B. W. Rose, "ccnr," 06 1994. [Online]. Available: http://www.ccnr.org/fatal_dose.html. [Accessed 06 02 2016].

[10] R. Hower, "softwareqatest," 1996-2016. [Online]. Available: http://www.softwareqatest.com/qatfaq1.html.

[11] Guru, "guru99," 2016. [Online]. Available: http://www.guru99.com/software-testing-life-cycle.html.

[12] G. Sohoni, "dotnetcurry," *Comparison of Automated Testing Tools: Coded UI Test, Selenium and QTP,* no. DotNetCurry .NET Magazine , 2014.

[13] M. Verma, "softwaretestingmentor," 27 01 2013. [Online]. Available: http://www.softwaretestingmentor.com/selenium-tutorials/limitations-of-selenium/.

[14] R. H. a. RaiMan, "readthedocs," 2014. [Online]. Available: http://sikulix-2014.readthedocs.io/en/latest/basicinfo.html.

[15] OpenCV, "doc.opencv," [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html#template-matching.

[16] A. Memon, I. Banerjee and A. Nagarajan, "GUI ripping: reverse engineering of graphical user interfaces for testing," in *10th Working Conference on Reverse Engineering*, 2003.

[17] Z. Jianhong, P. S. Sandhu and S. Rani, "A Neural Network Based Approach for Modeling of Severity of Defects in Function Based Software Systems," in *IEEE International Conference on Electronics and Information Engineering*, 2010.

[18] ILX Group, "prince2," 2016. [Online]. Available: https://www.prince2.com/uk/what-is-prince2.

[19] J. Wilson, "Essentials of Business Research: A Guide to Doing Your Research Project," SAGE Publications, 2010.

[20] W. &. M. S. Goddard, Research Methodology: An Introduction, 2nd ed., Blackwell, 2004.

[21] M. M, "tutorialspoint," 2016. [Online]. Available:
http://www.tutorialspoint.com/uml/uml_class_diagram.htm.

[22] wordpress, "devside," 19 09 2014. [Online]. Available:
https://www.devside.net/wampserver/load-testing-apache-with-ab-apache-bench.