Visvesvaraya Technological University,

Jnana Sangama, Belgaum - 590014

A Project Report on

"Automation of Testing Scenarios of HPE Verity"

Submitted in partial fulfilment of the requirements for the award of degree of

Computer Science & Engineering

Submitted by:

Sharath K P 1PI13CS141

Under the guidance of

Internal Guide	External Guide
Prof Phalachandra H L	**Mr. Anil Maiya**
Associate Professor, PES University	Manager, Hewlett-Packard Enterprise

Jan – May 2017

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

PES INSTITUTE OF TECHNOLOGY,

(AN AUTONOMOUS INSTITUTE UNDER VTU, BELGAUM AND UGC, NEW DELHI)

100FT RING ROAD, BSK 3RD STAGE, BENGALURU - 560085

PES INSTITUTE OF TECHNOLOGY

(An Autonomous Institute under VTU, Belgaum)

100 Feet Ring Road, BSK- III Stage, Bangalore – 560 085

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

Certified that the eighth semester project work titled **"Automation of Testing Scenarios of HPE Verity"** is a bonafide work carried out by

Sharath K P **1PI13CS141**

in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belgaum during the academic semester January 2017 – May 2017. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said Bachelor of Engineering.

Signature of the Guide	Signature of the HOD	Signature of the Principal,
Prof. Phalachandra H L	**Prof. Nitin V. Pujari**	**Dr. K S Sridhar**

External Viva

Name of Examiners	Signature with Date
______________________________	______________________________
______________________________	______________________________

ACKNOWLEDGEMENT

This project has resulted from suggestions, guidance and encouragement of numerous individuals. We deeply express our heartfelt gratitude to all those who have helped us in its completion.

To begin with, we are grateful to our Internal Guide, **Prof. Phalachandra H L**, Associate Professor, for his valuable guidance, novel ideas and excellent support during the course of our study and project work. He has always been a source of constant help and encouragement. We would also want to thank him for his continuous inputs and ideas for this project.

We are also grateful to our External Guide, **Mr. Anil Maiya**, Manager, for his valuable guidance, ideas and excellent support during the course of our study and project work. He has always been a source of constant help and encouragement. We would also want to thank him for allowing us to take up this project.

We acknowledge the support, the encouragement, extended for this study by our Project Co-ordinator, **Prof. V R Badri Prasad**, Associate Professor. We heartily thank him for facilitating the whole process and guiding us at every step.

We would like to thank **Prof. Nitin V. Pujari,** Head of the Department CSE for his constant support and encouragement.

We are grateful to our Principal, **Dr. K S Sridhar** who provided us adequate time and sufficient facilities to carry out the project. We express our gratitude for having given us an opportunity to learn and to experience the essence of teamwork in the due course of the project.

Last but not the least, we would also like to thank our parents and other family members for their continuous support to us, without which, we would not have been able to achieve whatever we have. We are also thankful to our friends who directly or indirectly helped us to complete our project successfully.

ABSTRACT

In the realization or implementation of any software application, Test automation plays an integral role. For any organization, budget and the timeline are the most essential and fundamental aspects. Development and testing must be accommodated within the stipulated economic and timeframe of any organization. This is one of the driving forces for accelerating the process of testing. The result obtained as a consequence is the Test Automation.

Automated Testing is an essential procedure which makes the testing faster and more reliable. The time to arrive to the market must be reduced as there is huge competition in the business ecosystem. The quality of the product, which is of primary significance, can be maintained and improved by making this well-structured automated testing. Another advantage of automating the testing process is that new builds can be verified autonomously by re-deploying appropriate code, thus ensuring functionality is not broken in present and future releases.

An application functioning as desired relies on it working with accurate and complete data in the back-end. Therein lies the importance of thoroughly verifying and validating that the data that goes into the back-end is accurate. This also shines light on the limits of the application and sets specific bounds on the input, while also exposing flaws in the logic, and presents areas of improvement in the same.

Therefore, the scope and purpose of Test Automation and Validation is far-reaching and hence, there is ample room for enhancements, improvements and learning. This project is carried out in order to understand the benefits of the automated testing process and develop expertise in various industry-standard tools for the same.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1 Introduction

Every software product needs to meet the expectations from the product in terms of functionality and quality. So quality related activities like testing is part of the Software development life cycle.

On the other hand, it is also necessary to release the product within short, well-defined timelines. In order to achieve that, development and testing must be done rapidly. If we see the effort estimation of each phase in SDLC, on an average 50% of the total effort of development life cycle is put on the various types of testing. It is also often seen that a significant part of testing happens manually. Hence, we need to find ways to reduce the resources consumed by the manual testing. In order to achieve that, we need to automate the test scenarios so that they, once coded, perform the required tests consuming reasonable amounts of time, and can also be reused to perform regression tests.

1.1 Problem Definition

Since, automation testing acts as a great work around for manual testing, it is also important to select which tools to use for the required requirements. Hence, following are the required goals/tasks which needs to be performed for automated testing.

1. Studying the various types of testing frameworks and testing tools:

As mentioned, it is very important to pick the proper testing tool of automation. The many testing tools are available in the market for various uses. The tool may be developer oriented, functional oriented or load testing tools. Along with this developing the automation software itself acts as a development process. Hence, study on different test automation frameworks needs to be done.

2. Automation of GUI Testing

After selecting appropriate testing tool and test framework, instead manual testing, automate the GUI Testing of HPE Verity product.

3. Automation of Back end validation

Instead of doing the manual testing for validating the back end. Automate the validation of back end involved in HPE Verity Product.

1.2 <u>Generic Proposed Solution</u>

Understanding the importance and realizing the significance of automated testing is the primary step of this project. Also the aim is to divide the automated testing into various modules

Since HPE Verity is tested manually, Instead of manual testing, Automation software needs to be written for the following modules.

1. Automation of Users section

The user's section contains user's module, roles module and group's module. Hence, objective is to complete the automation for complete section

2. Automation of Graph validation

In HPE Verity, many graphs are used to perform the analysis on uploaded data. Data flow graph is one of important graph comes in Information Archiving page. Automation of data flow graph validation is another solution for manual testing.

3. Automation of back end validation

Plenty of testing is done in Verity back end validation. In these, validation of language detection for documents needs to be automated. By automating this validation, testing engineer need not waste too much time in validating the detected language.

Another back end validation is speech to text validation. Since it is very difficult is check the accuracy of speech to text engine involved in back end of HPE Verity. Objective is to study the various tools and write automation software for finding the accuracy of speech engine for different language in the world and to make the comparative study w.r.t other speech engines available.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

PES INSTITUTE OF TECHNOLOGY

(An Autonomous Institute under VTU, Belgaum)

100 Feet Ring Road, BSK- III Stage, Bangalore – 560 085

Project ID: 2017/13CS495/PW085

Project Title: Automation of Testing Scenarios of HPE Verity

Project Team:

1PI13CS141	Sharath K P

This project report was submitted for review on ____________________________. I acknowledge that the project team has implemented all recommended changes in the project report.

Guide signature with date:

Guide Name: Prof. Phalachandra H L

2 Literature Survey

There are many automation frameworks that exist for automated testing purposes. Each frameworks has its own advantages and disadvantages. It is very important to select the proper test automation framework so that it will help in the software development life cycle.

A software framework is a combination of protocols, rules, standards, regulations and guidelines that can be implemented or followed so as to obtain the advantages of the blueprint provided by the framework. Hence, we use several generic functionalities provided by the framework and also add our own code to get selective functionalities from the software product. Since this project mainly deals with test automation frameworks, let us briefly describe them as follows. A "Test Automation Framework" is a blueprint that is used to provide the environment to execute the automation test scripts. The framework helps the user to write and execute the test scripts. And also it helps a developer generate reports based on the results of test cases. Hence, framework acts like a system to automate test cases.

It is very important for the framework to be independent of software application being developed. That is, it should be amenable for use with any application without considering the internal implementation of application under test. Some popular test automation frameworks are:

Types of test automation frameworks.

a) Module based testing frameworks

In module-based frameworks, the framework divides the whole application into small modules. After doing so, test scripts are coded for each of those individual modules. After combining all the test scripts, the net entity will represent the entire module. The application is divided into small modules based on one of the most popular OOPs concepts – Abstraction.

Advantages

- Module based automation framework has its own advantages. Since it divides the whole application into several independent modules, there is a high level of modularization of the application. It leads to easier and cost-efficient maintenance.
- Since framework divides the application into individual modules, whenever new features are added to the application, it is easy to test them. Also whenever some changes are made to the application itself, there is no need to change all the test script files. Remaining test scripts can be untouched.

Disadvantages

- Since test data is embedded in the script files, whenever there is a need to test the application with other set of testing data, script files need to be changed manually.

b) Library architecture testing framework

It is built on the module-based framework. Instead of dividing the application into test scripts, the application is divided into common functions. Hence, we create a common library which consists of common functions, so that these functions can be called by other test scripts whenever required.

Advantages

- Like module-based framework, this also introduces a high level of modularity. This enhances ease of maintenance, cost-efficiency and scalability.

  Since common libraries which contains all the common functions are created, these can be used by other test scripts which adds a great deal of reusability.

Disadvantages

- Because of libraries, framework becomes little complicated.
- As with the module based framework, test data is embedded into the script files itself, hence these files need to be changed whenever there is a need to test with other test data.

c) Data driven testing framework

The main aim of the data driven testing framework is to remove the core disadvantage of the module-based and library-based testing framework. Here we don’t embed test data inside the test script files. Instead, test data is separated from the script files and are stored in external storage. This may be a database, XML, CSV or even text files. Along with the test input, expected output data can also be stored in external storage.

Advantages

- Since test input data is separated from script files, very less scripting is required to cover all scenarios. This feature helps in maintainability of code.
- If some changes needs to be done to the input data or script files, there is no need to change the rest, which adds flexibility.

Disadvantages

- Since a separate input data file needs to be maintained, there is a need to come up with input source(s) for data collection.

d) Keyword driven testing framework

This framework is considered as extension to the Data Driven Framework. Along with the test data it also segregates the test scripts. These test scripts are considered as the Keywords. Hence, keyword consists of script file and the input data which is stored in external storage for these scripts files. Keywords specify what need to be done in the application. Hence these keywords specify the actions needs to be done. Usually these keywords and input data is stored in tables. Hence, this type of automation is also known as the Table Driven Framework. Also it important to note that keywords and test data is independent of automation tool being used. These keywords are used by the other test cases whenever it is required. For example, Login can be considered as the keyword in certain application and this Login keyword is used in the other test cases.

Advantages

- All the advantages of Data Driven Framework applies and also, user need not know the knowledge in certain cases. Keywords can be used in such cases which avoids the reinventing the wheel. Hence these keywords can be used in various test cases.

Disadvantages

- For large application this framework becomes complicated.
- Developer should know the clear insight about Keyword Driven Framework to use the advantages of the framework.

e) Hybrid driven testing framework

As the name suggests, this framework combines one or more automation frameworks like Data Driven Framework, Keyword Driven Framework etc.

f) Behavior driven testing framework

Behavior driven framework allow us to validate the functionalities of the application. Hence, from this framework it will be very easy to understand the application. User need not know the programming knowledge. Hence in this type of automation, we verify the certain functionality in-order to check the remaining functionalities.

By studying the various types of automation framework, I will be using the Hybrid Test Automation framework. In the Hybrid Automation Framework, I am combining the Data Driven Framework and Behavior Driven Framework. The reason for choosing this framework is, automation needs to be done with various test data hence we can store and pass these data from the XML files. And reason for using Behavior driven framework because certain functionalities needs to be validated prior to validate any other functionalities. For example, we need to verify the login functionality before checking the other functionality.

Test Automation Tools:

After understanding about the various frameworks, it is important to select the proper tool for Automation. The decision is taken mainly based on the goal need to be achieved like based on type of testing. Below is the generalized categories of types of test and the tools associated with them.

a. Developer oriented tools
b. Functional testing tools
c. Load testing tools
d. Performance monitoring and maintenance tools

a) Developer oriented tools

This basically involves the component or unit testing tools. This type of testing is done in the early stages of SDLC. Testing is done to the individual components or unit of the software product.

Tools: Junit, TestNG.

b) Functional testing tools

These tools basically helps to check the functionalities of the software product. Hence script codes are modifies based on the changes done to the product in the upcoming releases. These tools also help to conduct regression test on the product.

Tools: Hewlett Packard Enterprise UFT (formerly Quick Test Pro), Selenium.

c) Load testing tools

After the product has been developed, it is very important to check how the application works in the various scenarios. Hence, performance of the application need to be checked. For the reasons, performance and stress testing tools are used.

Tools: LoadRunner, Jmeter.

d) Performance monitoring and maintenance tools

These tools are used after the application already in production. Hence, product needs to be monitor and verified how the application performs in the real world. Hence, tools monitors the environment required for the applications within the certain thresholds.

Following are the some of the tools studied

a) Junit – Java Unit Testing Tool

Junit is unit testing tool which is used for java applications. Unit testing is mainly done by using various annotations; an annotation always starts with @.

Some of the annotations are as follows

- @BeforeClass: Executes prior running any tests in the class. Annotated method will be executed only once.
- @AfterClass: Executes after all the tests has been completed. Annotated method will be executed only once
- @Before: Method will execute before running any test method
- @After: Runs after the completion of any test method
- @Test: Used to declare the test method.

Some of the features of Junit is as follows

i. Fixtures: Fixtures are something like methods which is used to maintain the environment to run test cases. setUp() function is used to set-up the environment and tearDown() is used clear the environment which has been set.

ii. Test Suites: Suits are mainly used to combine/bundle the many test cases so that we can run all of them at a time. Using suits we can also run other suites along with the tests. TestSuite class is used for this purpose.

iii. Test Runners: Used to run the Junit tests. While running the Junit tests, we need to store the result of the tests in the TestResult object.

iv. Junit Classes: Junit comes with many classes which is mainly used to create the test case.

Some of the classes are as follows

- Assert - used to verify whether test cases are passed or not. Assert class comes with various functions which can be used to get the result. Assertions are used in test method.
- TestCase - Used to define the Fixtures to run the test cases. Fixtures are mainly used to set-up the environment required to run tests cases.
- TestResult -collects results of all the test cases.
- TestSuite – Used to create suits and run them.

Since Junit unit doesn’t support GUI testing, another automation framework named Abbot can be used for this purpose. However, this automation framework is restricted to only Java’s SWING GUI.

NUnit is the Unit testing automation framework used to perform unit testing in .NET applications.

b) <u>TestNG – Unit and Integration testing tool</u>

TestNG is another tool which is mainly used for the unit testing. The tool can also be used to perform integration testing. TestNG comes which several advantages over Junit.

Features of the TestNG:

i. Comes with more number of annotations as compared to the Junit. Following are the annotations which can be used to perform unit testing

- @BeforeSuite – Runs only once for a single suite. Before running any tests in the suits, this method will be executed.
- @AfterSuite – Same as the @BeforeSuite. However, it will be executed after all the tests in the suite executed.
- @BeforeClass – Method will be executed once before running any of the test method inside the class.
- @AfterClass – Same as @BeforeClass, However, method will be executed once after all the test method in the class executed.
- @BeforeMethod – Same as @Before in Junit. Method will execute before running any test method.
- @AfterMethod – Same as @after in Junit. Method will execute after running any test method.
- @BeforeGroups – In TestNG, test cases can be grouped, so that we can run all those tests belonging to a group at a time. This annotated method will be executed before running any of the test method in the group.
- @AfterGroups – Same as @BeforeGroups. However, method will be executed once after all the tests inside the group completes the execution.
- @BeforeTest – Method runs before any test methods which belong to the class inside the <Test> tag. This tag is mentioned in the xml file.
- @AfterTest – Same as the @BeforeTest. However, it will run after all the test methods

execution got completed.

- @DataProvider –used to pass the test data to test method. In-order to use this method, test method needs to specify this annotation name. This method always returns the 2D array. Each row is passed as the parameter set to the test method.
- @Test – Used to declare a test method.
- @Parameters – Used to pass the parameters to the test method.

ii. Since TestNG comes with the various annotations, we can have more control over the order of execution of test cases.

iii. We can also run the single test cases. If test method depends on any other test, those tests will be executed first.

iv. Since test cases can be run from XML file, unlike Junit, test cases belonging to a particular group, single test case which belong to the group or class can be run separately. Test cases can also run in parallel, so that it will take less time to complete the execution. Parameters for test cases can be injected from xml file using the parameter tag. @Parameter is used to fetch the parameter value in test case. Default value can also be set in test case, so that default value is used when data is not passed from XML file.

v. Annotations can take various parameters. Some of the parameters are: dependsOnMethods, alwaysRun, priority, groups, dataProvider, dataProvoderClass (Used when we move the dataProvider to separate class).

vi. TestNG supports various listeners which can be used for outcomes of the test case. Usual outcomes of the test case is success, failure or skipped. These listeners listens for every test method before and after the test method.

c) <u>LoadRunner – Performance Testing Tool</u>

LoadRunner is the Performance Testing Tool which is developed by the Hewlett-Packard. LoadRunner is not an open source tool, however, one can use the trial version for testing, which provides to limited functionalities.

Components of the LoadRunner as follows

i. VUGen

VUGen is basically a Virtual User Generator. It is used to generate virtual users who uses the system under test (SUT). VUGen also generates a scripts which access the various features of the SUT by the Virtual User. Here, VUGen (Scripts) mainly runs on the client, connects with the server and sends the response sent by the server to the client.

ii. Controller

Controller basically controls and monitors the virtual users. Controller set-up the performance test and generates the load on the application in-order to analyze the behavior of the system.

iii. Analyzer

It is used to view and analysis the performance test results. Analyzer collects and consolidates all the test logs from the performance test. We can also generate the report and various graphs to understand the system behavior.

In-order to run the performance test, script file needs to be written. Script file can also generated by recording the actions performed manually. However, recording feature is available in Windows Operating System. Recorded script could be run in any Operating System. Script will be generated in C Programming. Once the script is generated, modify It by adding various inbuilt functions provided by the LoadRunner. Some featured are:

i. LoadRunner supports for running the test from command line.
ii. Instead of hard coding the test data in the script, LoadRunner supports for the data driven test by parameterizing it. Hence we can run the test for user given data.
iii. LoadRunner supports various checks on the response data. Hence, we can do the content search like text check and image check on the data received from the server. These checks are very helpful when there is more loads on the server.
iv. Can set up the transactions. Using this we can set the maximum time for any step to perform. Hence, we can specify maximum response time and compare with the actual response time.
v. LoadRunner supports to set-up the Manual scenario and Goal oriented scenario. Performance load test is called Scenario. In Manual Scenario, we specify the number of Virtual Users. Where as in Goal Oriented Scenario, we specify the actual goal and controller can increase the number of virtual users in-order to reach the goal.
vi. Unit test scripts can be executed in LoadRunner.
vii. When multiple virtual users are running, resources consumed by these users dynamically can be seen.
viii. Analyzer helps to verify performance requirements of a system which is specified in the service level agreement.
ix. Using the analyzer, summary report can be generated.
x. Various graph provided by the Analyzer can be used to gather the information. These graph are updated dynamically while running the performance test.

d) JMeter

JMeter is an Open Source testing software. It is 100% pure Java application for load and performance testing. JMeter is designed to cover categories of tests like load, functional, performance, regression, etc., and it requires JDK 5 or higher.

It is a platform-independent tool. On Linux/Unix, JMeter can be invoked by clicking on JMeter shell script. On Windows, it can be invoked by starting the jmeter.bat file.

Basically, the core working functionality of Jmeter is that JMeter simulates a group of users sending requests to a target server, and returns statistics that show the Performance/functionality of the target server/application via tables, graphs, etc. It achieves this by various components in it.

The major parts in Jmeter which play a crucial role are:

i. Test plan - It's like a container which contains all the elements required to run the test.
ii. Thread group - It's a group of users which are used to run the test. Set up an environment like no. of users/threads, ramp up period, scheduler which takes start and end time based on the requirement.
iii. Work bench - It contains temporary elements and is useful for complex test.
iv. Assertions - This checks on the response. Different types of assertion are present like
    - Response assertion: checks the code, text response, URL Sample
    - Duration assertion: can set up response time
    - Size assertion: check on the number of bytes it received in the response
    - HTML Assertion: check on format of response
    - XML Assertion: check on XML format
    - XPath Assertion: check for the particular node or the value of that node
v. Listeners- are elements of JMeter which gathers information about the performance test being executed. It is used to view the results. Results can be viewed in different formats such as:
    - View results in table listener - It helps us to see the results in the table - like latency, size of the request, Connect time, Bytes, Sample.
    - View results in tree - It gives in tree representation and consumes a lot of memory. Hence advisable to use it less in test plans running heavily.
    - Aggregate report - gives all the average values of view results in table form. It also shows throughput.

- Simple Data Writer - log our results to a file –can be configured for what to and what not to log.

vi. Graph results - It shows graphical representation of the test output. It is generated in real time and consumes high memory. So it is better not to use.

vii. Summary report -It shows general summary like Label (Name of the test), Samples, Average, Min, Max, Standard Deviation, Error %, Throughput, KB/sec, Average Bytes.

viii. Some special features of Jmeter are:

ix. JMeter can be run in command line- It is very efficient and consumes less memory.

x. Since it is open source, custom plugins are available (jmeter-plugins.org/wiki/Start).

xi. Download Jar of plugin and add it to ext folder inside bin or we can use plugin manager to install plugin, remove old plugin, upgrade existing plugins.

xii. Instead of hard coding the test data, pass the test data by parameterization through csv files and read from csv file.
    - Use CSV Data Set Config(Config Element) to take the data from the csv file.
    - Update the variable fields by ${variable_name} in both REST and SOAP servers

xiii. Can also have the function along with the support of arguments which one can use to populate the other element of the test plan. It follows camel case and case sensitive operations.

xiv. JMeter comes with certain inbuilt functions to make our tasks easier like - __log(),__time(dd MM YYYY HH MM SS),__threadNum(),__intSum(),

xv. It can perform real world performance test - It makes use of pacing and ramp up (add certain number of users in a specified time period), think time (simulate users with timings/delays), ramp down - control time delays between iterations, achieve x iterations in y mins/sec.

xvi. It uses plugin called Stepping Thread Group - in JMeter. It is similar as load runner tool - supports all the feature of load runner in JMeter.

xvii. Correlation - This helps us to extract the dynamic value from response and uses it in subsequent request. To use these values use regular expression extractor to fetch the value from the response.

xviii. Timers -This adds time delays between user interaction using various timers like Constant timer, Constant throughput timer, Uniform Random Timer, Poisson Random Timer etc.

xix. Can also create a test plan for database

Since, Jmeter is an Open Source Tool, it provides lot of features and also features to the tool will be keep on adding unlike LoadRunner. However, The Graphs provided by the

LoadRunner are very useful because of Dynamic behavior of graphs while running. However, these graphs are not provided by the Jmeter. Hence, one can analyze in better way and get lot of feedback regarding the load generated while running from the LoadRunner.

Code Coverage Tools:

There are many tools available to do code coverage testing. Code coverage testing is basically to test the source code written on various coverage metrics. Some of the coverage metrics are: Method Coverage, Statement coverage, Line coverage, Branch Coverage, Decision Condition coverage etc. All these metrics are shown in percentages and generated based on the test run. From these coverage metrics, one can identify the critical section of the code, redundant test case, untested code segments etc. Hence, doing this, quality of code written can be known. After performing the code coverage testing, a report can be generated based on the tool being selected for testing purpose.
Following are some of the code coverage tools studied.

e) <u>Code Cover</u>

Code Cover is the Code Coverage tool for the Java Applications. It can be used by both standalone and also as a plugin which can be integrated with the Eclipse IDE. From IDE, test cases can be run with Code Runner. So that tool can generate the metrics to analyze the source code.

There are several ways to execute the System under Test (SUT). By using the system manually. In that case, measured data will be collected to a single test case. Or one can use the "Live Notification", in this each test cases can be recorded. One can generate the code cover measurements on test suites of Junit etc. After the completion of SUT, metrics can be seen or report can be generated in HTML and CSV Formats. Once the SUT is finished, tool helps us to identify the critical section by highlighting it in different colors. However, new release haven't come since 2011. Hence, it is supported for Java 1.7

f) <u>Atlassian Clover</u>

Clover is a code coverage tool released by Atlassian Company. Following are the features of Clover tool:

i. Tool measures the metrics based on source code, not on the byte code.
ii. Coverage Metrics supported by Clover are – Method coverage, Statement coverage, Branch coverage.
iii. Can generate report in various formats like HTML, PDF, XML, JSON, TEXT. Report

consist of percentages of various coverage metrics, what are the critical sections, what are the sections which is not executed etc.

iv. Supports for Java programming Language. JDK version is 1.8
v. Code coverage can be done on famous testing frameworks. Supported frameworks are Junit, TestNG.
vi. Tool can be integrated with various IDE's like IntelliJ IDEA, Eclipse.
vii. Tool can be integrated with other build tools like command line, Ant, Maven, Grails, and Gradle.

Advantages of Clover:

HTML Report generated by the Clover is highly configurable. It also shows the top risks. Tools also supports the latest Java JDK version and it is being actively developed as compared to the Code Cover.

g) Selenium

Selenium is a browser-based automation framework. It is primarily used to automate web applications for testing purposes. Web-based administration tasks can be automated as well. Selenium is open-source.

Owing to its popularity, Selenium is supported by major browsers as a native part of the application. Drivers for various browsers are available. The approach taken by Selenium Web Driver, which is the tool used in this project, is to make direct calls to the browser using its native support for automation.

A domain specific language called Selenese is used to write tests. Alternatively, majority of the popular programming languages, including C#, Perl, Java, and so on, can also be used.

Components of Selenium include:

i. Selenium IDE:

It is implemented as a Firefox add-on, and allows the developer to record, edit and debug tests. Selenese is used to record the scripts, which provides commands for various options such as clicking and selecting elements.

ii. Selenium Client API:

Test can be written in various programming languages. These interact with Selenium by calling methods in the Selenium Client API. With Selenium 2, a new Client API was introduced (with Web Driver as its central component). However, the old API (using class Selenium) is still supported.

iii. Selenium Remote Control (RC):

Selenium RC is a server that accepts commands through HTTP. RC provides the flexibility to write tests in various languages. A separate instance of RC server is needed to run html test cases, which necessitates different ports for parallel runs.

iv. Selenium Web Driver:

This is the latest version of Selenium in use. Commands can be passed through either Selenese, or through a Client API. Most browsers launch a browser application, and pass commands to it. Unlike previous version, Selenium server is not required to run the tests. Selenium 2 provides the basic foundation required for automation. Developers are free to develop their own domain specific language upon this.

Selenium Grid:

It is a server that allows for multiple instances of the browser, running in parallel on several machines. One hub is required. Tests interact with the hub to get access to the instances of the browser they will run. This allows load distribution and also speeds-up execution.

Studying the various tools available for different purposes, for GUI testing, Selenium tool is predominantly used to test the functionalities of the software application. Java JDK 1.8 is used as the programming language for writing the script files using Selenium.

Language Detection

Since Verity has customers across the globe, it is very important provide the language detection feature accurately. Based on the customer’s requirement, language like Chinese, Japanese, Thai and Arabic languages are needed to be identified. Some of the open source tool available to detect the language of a text are as follows

i. Google Language Detection

Google provides its own java library to detect the language of the text. It identifies around 53 languages globally with the more than 90% accuracy. It basically uses profiles of the various languages. Profiles is basically an N-Gram associated with each languages. These N-Grams are taken from the Wikipedia. Since Wikipedia is multilingual encyclopedia, users contribute to it worldwide. Hence, detector instance use this N-Gram language corpus to detect the language of particular type. In order to use these language profiles, we need to load these into the detector instance. After loading the profiles, text is appended to the detector instance, so that language can be detected for the appended text. If the text appended contains multiple languages, one can get the probabilities of each languages identified through detector instance.

Since the last update was in 2011, which was almost 6 years ago, it has been discontinued. Hence proper results cannot get for current time. Also, library returns only the language code. In order to get the language name, again we have to some different work around, hence it is not feasible.

ii. Google Translate text to speech

Similar to the Google's language detection library, this library is written in Java with more functionalities like creation of text from speech, translating a text from one language to other, etc. however, unlike Google's language detection, one can get the language name using the language code in this library. The main disadvantage is that, last update was in 2011, supports only around 50 languages.

iii. Relativity – Structured Analytics

Unlike Google's libraries, it is desktop application. It will be very difficult to use this since automating a desktop application is beyond the scope. Also there is no support for detecting the multiple languages of the document.

iv. Polyglot

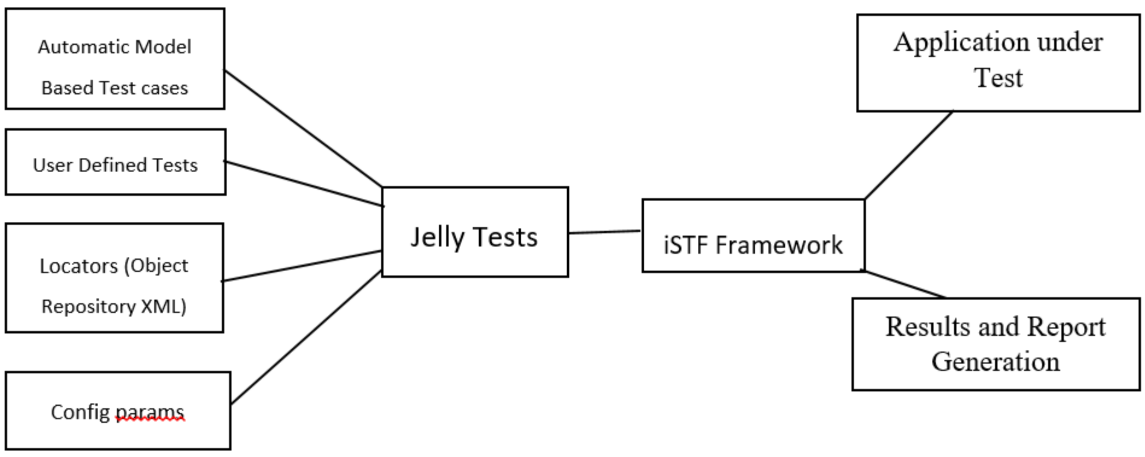
Polyglot is natural language pipeline available for the multilingual applications. We can use this in many ways like command line, as a python modules etc. since it is used for many natural language applications, It provides functionalities like Language Detection, Part of Speech tagging, Transliteration, Sentiment Analysis, etc. out of these featured language detection functionality is used in our project to detect the language of the document. Polyglot internally uses pycld2 module. However, pycld2 again uses cld2. Cld2 is basically a compact language detector which has been developed by Google. Cld2 probabilistically detect around 83 languages. Also if the given text contains mixed languages, it gives the top three languages along with the percentage coverage of each of the language w.r.t to the total text bytes. Since pycld2 uses cld2, further enhancement is done in order to cover the wide range of languages. Hence, as of now, pycld2 detects over 165 languages. One can detect the language of the document either for plain text, HTML or XML documents which are encoded in UTF-8. Over all, due to dependency of polyglot on pycld2, in resent updates, polyglot can detect the languages of around 195 languages. If we don't want to use polyglot as a python module, we can also use polyglot directly from the command line.

Since, polyglot module can detect the multiple languages in the given document, it provides use the required functionality. Hence, we are using this module in order to detect the language of the document.

3 System Requirements Specification

3.1 High level block diagram of the solution

Before describing in detail the project structure, it is necessary to know the Software Requirement Specification (SRS) for the project. The **I**ntelligent **S**olutions **T**esting **F**ramework (iSTF) is used for automated testing. High-level block diagram of the solution is shown below. Internally, the framework itself uses various software tools to make for smoother automation.



Automatic Model Based Test cases
User Defined Tests
Locators (Object Repository XML)
Config params
Jelly Tests
iSTF Framework
Application under Test
Results and Report Generation

Figure 3.1: High Level Block Diagram

i. Automated Model Based Test cases: Contains inbuilt functions like clicking the button, verify test in the text box etc.
ii. User Defined Tests: Customized tags are created for specific test cases. For this, Java classes need to be written to handle the functionality. These classes are then registered to the tag library. After that, these tags are used in Jelly Files to use this new functionality.
iii. Locators: This contain all information related to the GUI. Locator file contains the UI element information. By storing here, these values can be referred from the User Defined Tests.
iv. Configuration Parameters: This contains all the static data like jelly test location, admin information, database location etc. These are referred from the user defined tests.

Bringing these components together, a jelly file is written which specifies the step by step actions for automation of a single test scenario. After executing these test scripts, a report is generated using the ATU Reporting tool.

3.2 Environment used in the project

3.2.1 Hardware Requirements

Minimum Hardware required for the project are

i. Minimum of 4GB of RAM
ii. Minimum of 1 CPU
iii. Monitor or Screen of resolution 1024 x 768

3.2.2 Software Requirements

Below are the Software/Tools required for the project

i. These are the tools which are integrated with iSTF Framework
   - Selenium Web Driver – Tool for automating the Web Applications.
   - ATU Reporter
   - Protractor – Used to automate the AngularJS Web Application with selenium
   - TestNG – Used to control the various test suites which are need to be run.
   - Ant – To create the process as specified in Build file
   - Maven – Run with Ant help for reporting, documentation purpose.

ii. Eclipse and Notepad++ is used for programming purpose
iii. Driver for Internet Explorer 11, Chrome Web Driver of version 48 or later and Firefox Web Driver of version 43 or later are required to run the automation
iv. Java Runtime Environment (JRE) 1.8 and Java Development Kit (JDK) 1.8 is used for development.
v. XML is used to create the Jelly files which specifies the actions needs to carry out sequentially.
vi. Sense plugin for Chrome. To make API calls to Elastic Search database.
vii. Postman plugin for chrome. To send HTTP queries to external applications.
viii. Interceptor tool to add headers to the HTTP requests.

3.2.3 Requirements for the Project

3.2.3.1 Functional Requirements

The main aim of the project is to automate the users' module in the Verity product of the Hewlett-Packard Enterprise. Along with automation of users' module, automation of back-end validation needs to be performed.
The users' module contains the following functionalities

[F1] Automation of Users module:

Once the new user is on boarded, he can create internal users who belong to the organization. Hence, one of the requirement is to automate the complete user's module. Various requirements involved in the automation are specified below.
Requirements:
[F1.1] Create User
Once on boarded user logged in, he can create a new user with a certain privileges. One of the requirement is to automate the creation of user's with various combination involved with the internal user. After creating the user, also we need to validate whether the user has been created successfully in UI.
[F1.2] Edit a User
Once the new user is created, we can edit this user. Hence, automate the procedure to edit the user. As with the create user, validate whether the user details has been edited successfully or not in UI.
[F1.3] Delete a User
Automate the process of deleting a user after editing the user.

[F2] Automation of Roles module

The Verity product has a set of predefined roles that are to be followed by the users. After the data has been archived, it needs to be managed by a certain set of rules. These rules and permissions are given to specific people working in the team or group. Hence, one of the functionality is to automate the complete roles module.

Requirements:

[F2.1] Create a Role:

Automate the procedure of creating a role. In the automation, create role with various permissions and validate whether the role is created in UI successfully.

[F2.2] Edit a Role:

Once the role is created, automate the procedure to edit the role. After creating a user, check whether the edition of the role is successful or not from UI.

[F2.3] Delete a Role:

Automate the process of deleting a role. After editing a role, delete this role and verify the role has been deleted or not.

[F3] Automation of Groups module:

Different users can be combined together to form groups. This makes the workflow easier and manageable. Groups are created with specific roles for each group. Hence, another functionality is to automate the complete group's module.

Requirements:

[F3.1] Create a group

Automate the process of creating a group and verify the group has been created successfully or not in UI.

[F3.2] Edit a group

Automate the process of editing a group. Once the group has been edited, verify whether group has been created successfully in UI.

[F4] Validation of Data Flow Graph in Information Archiving

Once the data has been uploaded, the data flow graph categorize the uploaded data based on type of data. Graph will also shows the total size of the categorized data, and total number of these data. The functionality is to automate the process of validation of data flow graph against the data stored in back end database.

[F5] Automating the Validation of Language Detection of Documents

Once the data is processed, lot of information related to the data is stored in Meta data. LANGUAGE_NAME is one of the Meta data field is used to tell the language of the uploaded document. Hence, one can also make query on the archived data specific to the languages. Though, Elastic search detects the language of the document, we need to validate whether the detected language using the open source tool.

[F6] Automation of Speech to Text Validation

When audio file is uploaded, elastic search extracts the text from the audio file and store it. Validate this transcripts against original transcripts. Also, compare the elastic search transcript against other tools and make a comparative study on the various tools.

3.2.3.2 Non-Functional Requirements

[NF1] Portability

Need to carry out the automation in different environments like various browsers. Hence we need to verify the functionalities across browsers

[NF2] Integrity

Need to automate the working of whole application testing after new builds are deployed for every timeframe.

[F3] Reusability

May need to write the scripts which has common functionalities for different test cases.

[NF4] Robustness

Need to check whether the functionalities works fine for different input data

3.2.4 Constraints and Dependencies

i. Automation can be done only for Web Applications and not for desktop based applications
ii. UI automation is carried out only for those which uses AngularJS as the front end framework
iii. Automation is done by taking care of only Chrome 48, Firefox 43 and Internet Explorer 11 browsers or any higher versions.

3.2.5 Assumptions

i. All the test input data is passed only through the Jelly (XML) files.

ii. All the static information like IP address, Database location, whether the automation should run in local machine or in remote machine and in which browser to automate, those information are specified in Properties file and these values are used throughout project. Hence, it will be easy to maintain the code.

iii. All the object locators like button, test box of the UI which is being automated is stored in object properties file. These values are then used in the script files. Hence, whenever any object field in UI is changed, we can easily make the changes in the code.

3.2.6 Use Case diagram of the System

3.2.6.1 Create/Edit/Delete the users

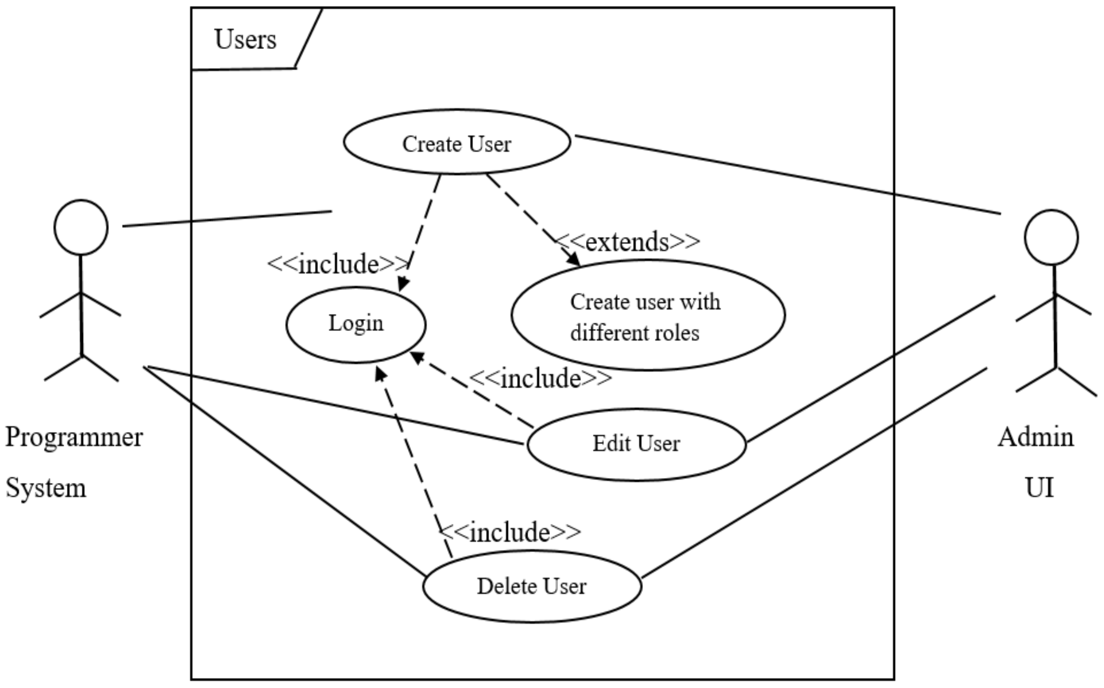


Figure 3.2.1: Use Case Diagram of Users module

3.2.6.2 Create/Edit/Delete the Roles

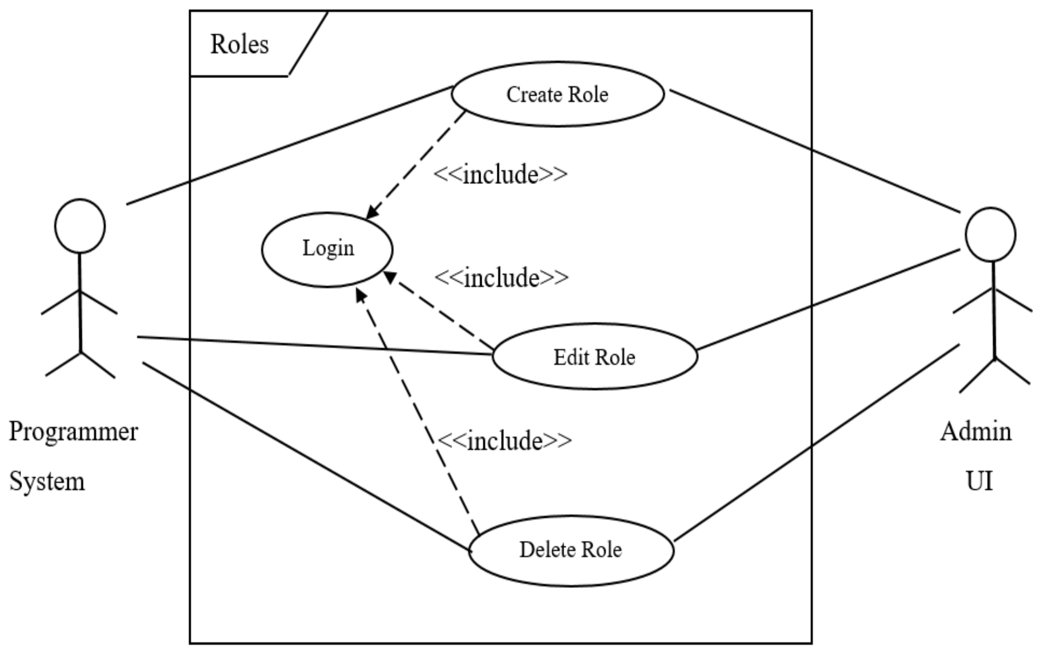


Figure 3.2.2: Use Case Diagram of Roles module

3.2.6.3 Create/Edit the Groups

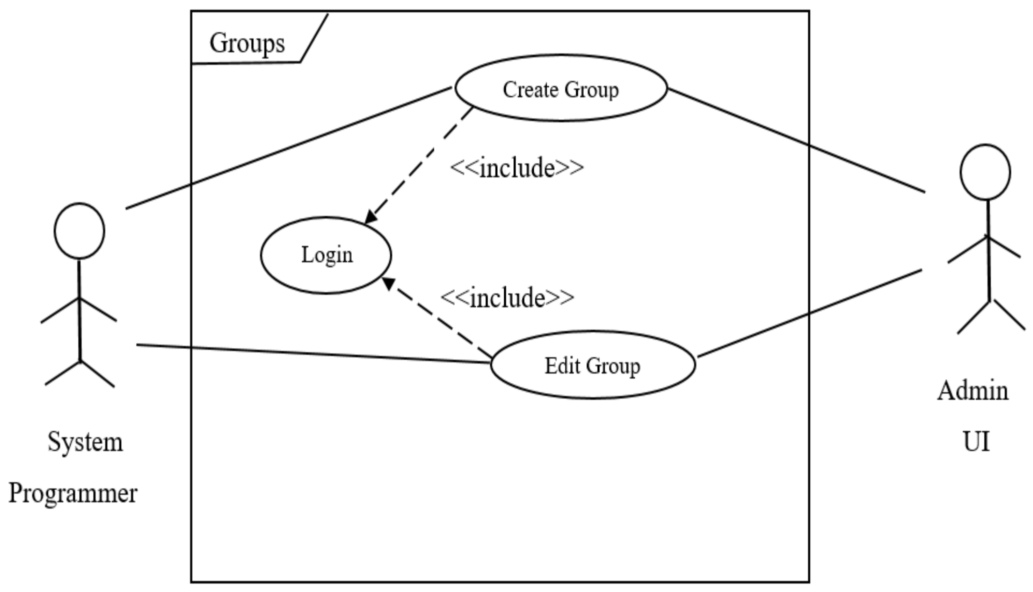


Figure 3.2.3: Use Case Diagram of Groups module

3.2.6.4 Automation of Back-End Validation

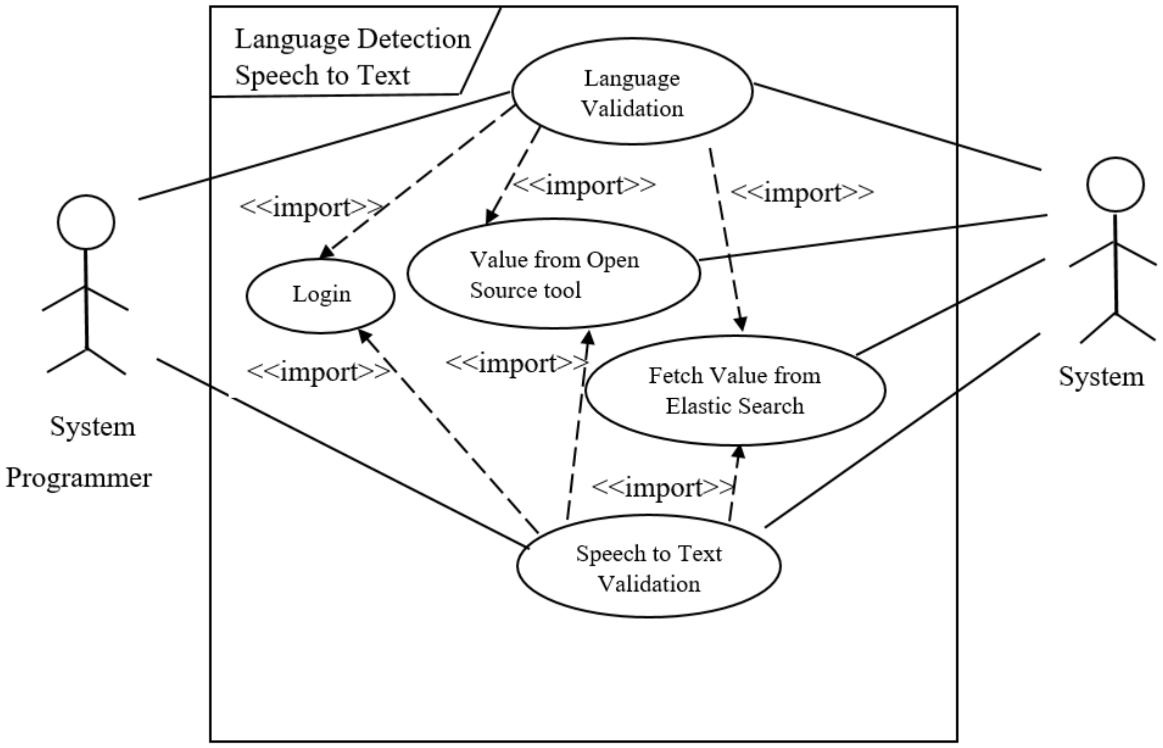


Figure 3.2.4: Use Case Diagram of Back-end validation

3.2.7 Requirement Traceability Matrix

Req. No.	Req. Description	System Design (Architecture)	Detailed Design	Implementation	Testing
F1.1	Create User				
F1.2	Edit User				
F1.3	Delete User				
F2.1	Create Role				
F2.2	Edit Role				
F2.3	Delete Role				
F3.1	Create Group				
F3.2	Edit Group				
F4	Validation of Data Flow Graph				
F5	Validation of Language Detection of Documents				
F6	Speech to Text Validation				

Table 3.1: Initial RTM

4 Schedule

Milestones	**Start Date**	**End Date**
Literature Survey	20/1/2017	10/3/2017
Understanding of Verity Product	16/1/2017	5/2/2017
Understanding about the iSTF Framework	23/1/2017	30/1/2017
SRS	30/1/2017	8/2/2017
Functionalities of Users Section in Admin UI of Verity	6/2/2017	7/2/2017
Automation of Users Module	8/2/2017	28/2/2017
Validation of Data Flow Graph	1/3/2017	6/3/2017
Automation of Groups Module	7/3/2017	10/3/2017
Validation of Language detection	13/3/2017	31/3/2017
Automation of Roles Module	3/4/2017	9/4/2017
Validation of Speech To Text	9/3/2017	19/3/2017

Table 4.1: Schedule

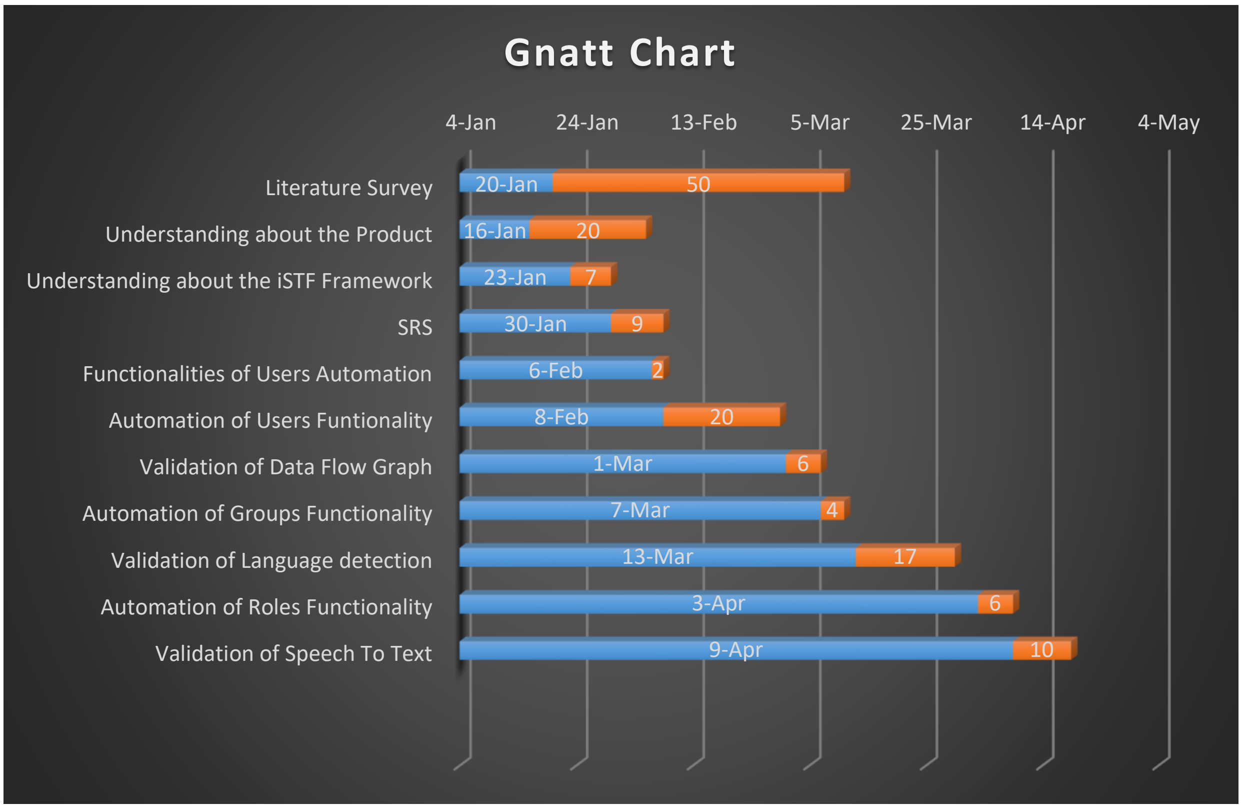


Figure 4.1: Gnatt Chart

5 System Design or High Level Design

5.1 Architectural Diagram

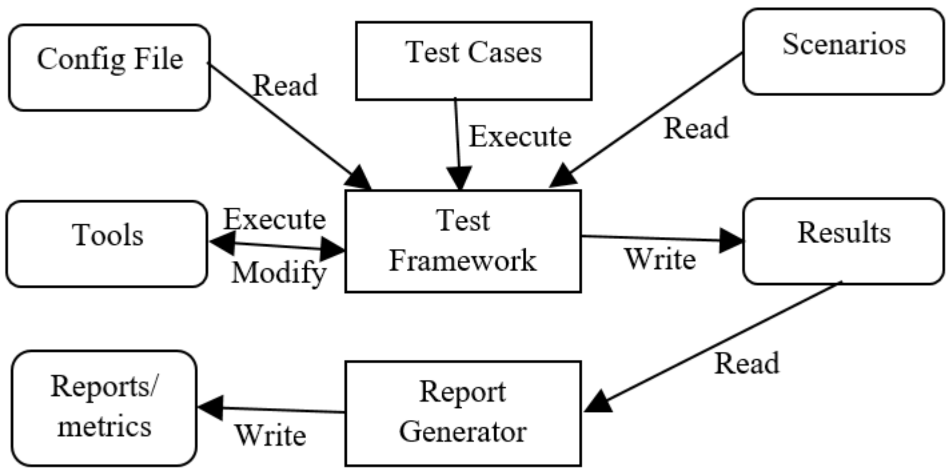


Figure 5.1.1: Architecture Diagram

Description of modules

5.1.1 Config file

This file contains all the constant parameters required for the automation. Hence, by keeping this constant values separate, we don’t have to touch the source code whenever something changes. And also, person with less technical knowledge can also run the automation by setting these parameters. In the project, all the configuration files are stored in conf folder.

5.1.2 Test Cases

This contains all the test cases which need to be executed. Test cases are created using Jelly files in XML format. All the test input data required for the test case is passed from here. Here, once the test case is failed, still remaining test cases are executed. Jelly files are stored in the AdminJellyCases in the root directory.

5.1.3 Scenarios

Scenario is something which explains how each test case need to be executed. Hence, in out project, each step required involved during automation is specified in Jelly files. Hence, all the scenarios are stored in the form of Jelly files. Here, each Jelly file is considered as one scenario.

5.1.4 <u>Test Framework</u>

Test framework being used for the automation is iSTF (Intelligent Solutions Testing Framework). Since some organization develop their own framework for automation purpose, this framework is developed by the HPE. The framework belongs to Hybrid automation framework. Framework consists of both data driven and behavior driven framework.

5.1.5 <u>Tools</u>

Since framework just provides the blueprint, iSTF itself uses many open source tools for various purpose. Some of the tools integrated with the framework are Java 1.8, Selenium, TestNG and ATUReports.

5.1.6 <u>Results</u>

Once the test case execution is completed, the result of these test cases can be identified by TestNG. From these results, report can then generated by using the tool ATUReports. All the generated reports are stored in ATUReports folder inside the root directory.

5.1.7 <u>Report Generator</u>

Once these reports are written, some of the summary reports can be generated by using ATUReports. Here, these reports can also be sent to the email id mentioned in the configuration file.

5.1.8 <u>Reports/metrics</u>

As explained above. Using the individual reports, summary reports is generated and sent to the email. In order to generate these reports, ATUReports is being used.

5.2 Sequence Diagram

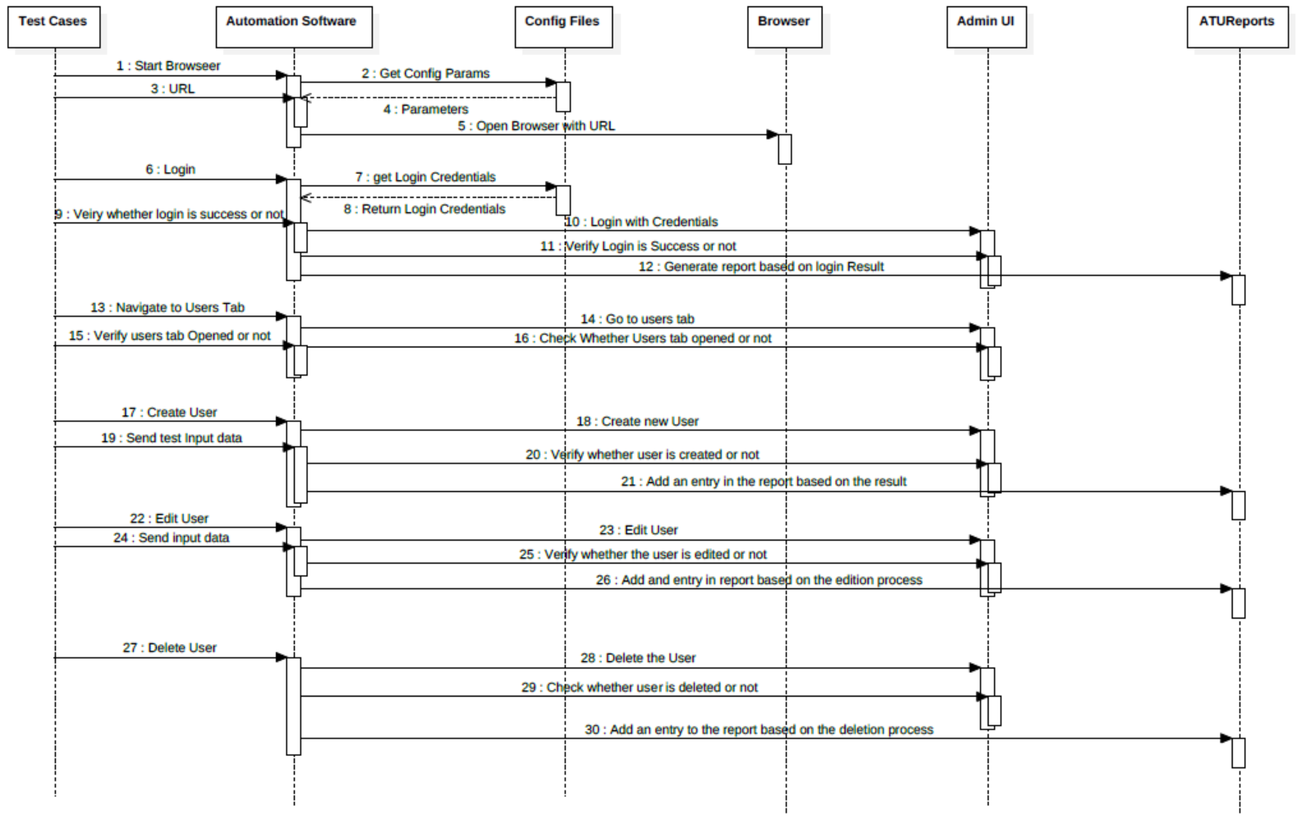

Test Cases
Automation Software
Config Files
Browser
Admin UI
ATUReports
1 : Start Browseer
2 : Get Config Params
3 : URL
4 : Parameters
5 : Open Browser with URL
6 : Login
7 : get Login Credentials
8 : Return Login Credentials
9 : Veiry whether login is success or not
10 : Login with Credentials
11 : Verify Login is Success or not
12 : Generate report based on login Result
13 : Navigate to Users Tab
14 : Go to users tab
15 : Verify users tab Opened or not
16 : Check Whether Users tab opened or not
17 : Create User
18 : Create new User
19 : Send test Input data
20 : Verify whether user is created or not
21 : Add an entry in the report based on the result
22 : Edit User
23 : Edit User
24 : Send input data
25 : Verify whether the user is edited or not
26 : Add and entry in report based on the edition process
27 : Delete User
28 : Delete the User
29 : Check whether user is deleted or not
30 : Add an entry to the report based on the deletion process

Figure 5.2.1: Users Sequence Diagram

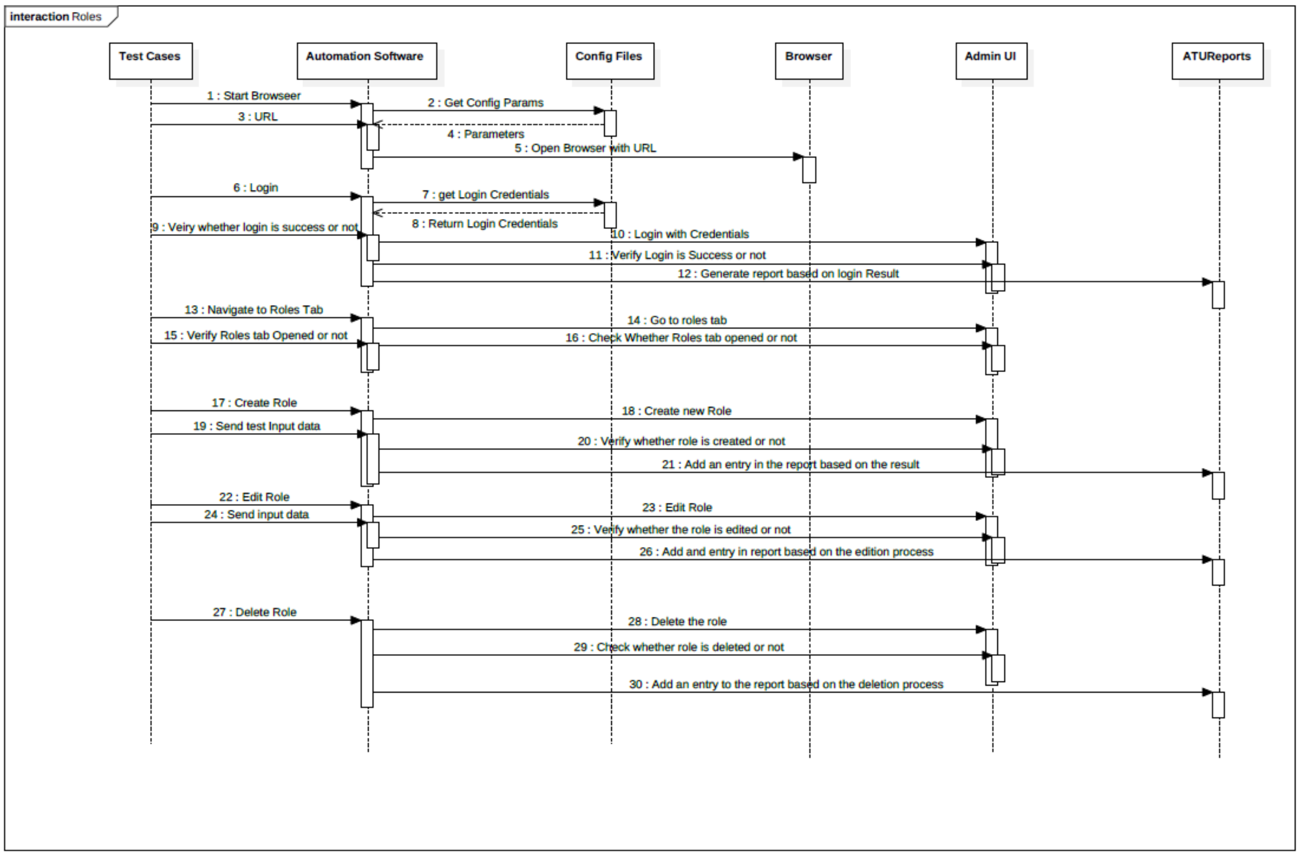

interaction Roles
Test Cases
Automation Software
Config Files
Browser
Admin UI
ATUReports
1 : Start Browseer
2 : Get Config Params
3 : URL
4 : Parameters
5 : Open Browser with URL
6 : Login
7 : get Login Credentials
8 : Return Login Credentials
9 : Veiry whether login is success or not
10 : Login with Credentials
11 : Verify Login is Success or not
12 : Generate report based on login Result
13 : Navigate to Roles Tab
14 : Go to roles tab
15 : Verify Roles tab Opened or not
16 : Check Whether Roles tab opened or not
17 : Create Role
18 : Create new Role
19 : Send test Input data
20 : Verify whether role is created or not
21 : Add an entry in the report based on the result
22 : Edit Role
23 : Edit Role
24 : Send input data
25 : Verify whether the role is edited or not
26 : Add and entry in report based on the edition process
27 : Delete Role
28 : Delete the role
29 : Check whether role is deleted or not
30 : Add an entry to the report based on the deletion process

Figure 5.2.2: Roles Sequence Diagram

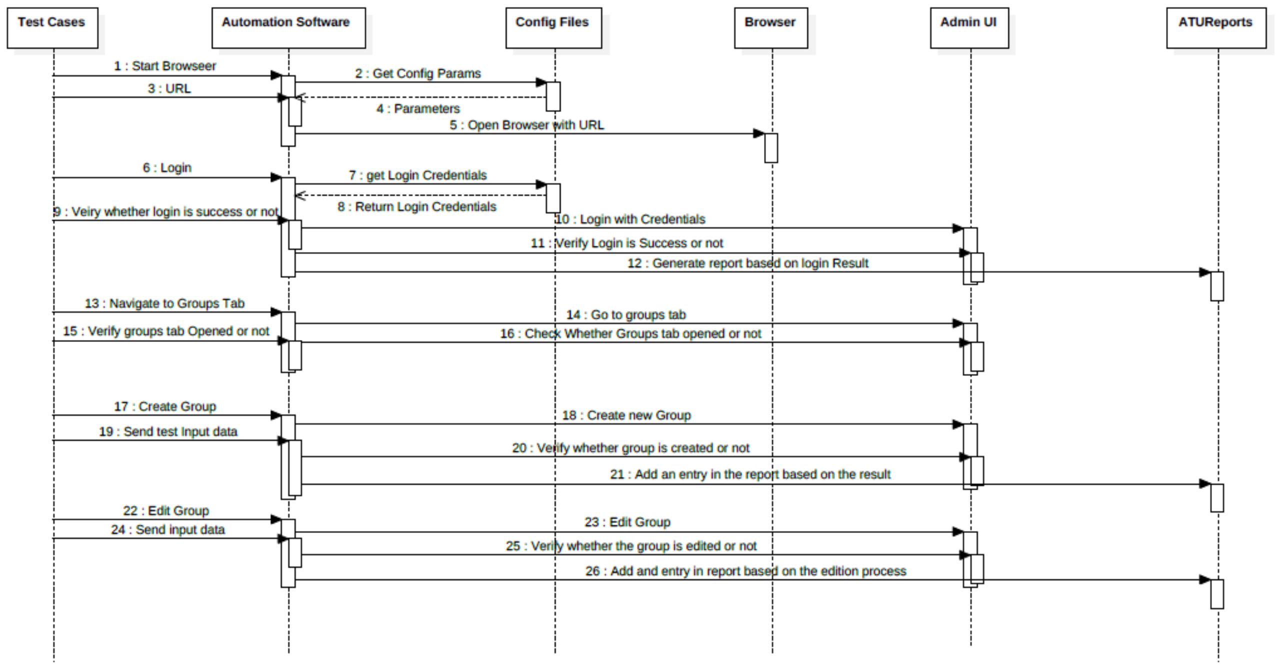

Test Cases
Automation Software
Config Files
Browser
Admin UI
ATUReports
1 : Start Browseer
2 : Get Config Params
3 : URL
4 : Parameters
5 : Open Browser with URL
6 : Login
7 : get Login Credentials
8 : Return Login Credentials
9 : Veiry whether login is success or not
10 : Login with Credentials
11 : Verify Login is Success or not
12 : Generate report based on login Result
13 : Navigate to Groups Tab
14 : Go to groups tab
15 : Verify groups tab Opened or not
16 : Check Whether Groups tab opened or not
17 : Create Group
18 : Create new Group
19 : Send test Input data
20 : Verify whether group is created or not
21 : Add an entry in the report based on the result
22 : Edit Group
23 : Edit Group
24 : Send input data
25 : Verify whether the group is edited or not
26 : Add and entry in report based on the edition process

Figure 5.2.3: Groups Sequence Diagram

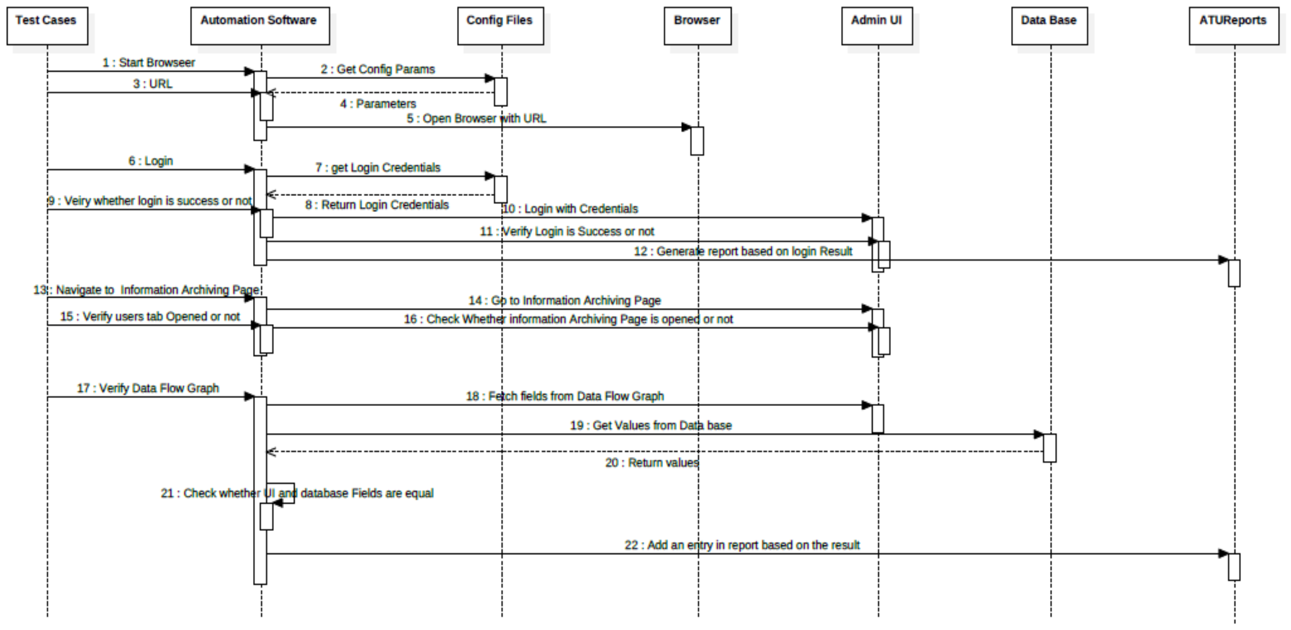

Test Cases
Automation Software
Config Files
Browser
Admin UI
Data Base
ATUReports
1 : Start Browseer
2 : Get Config Params
3 : URL
4 : Parameters
5 : Open Browser with URL
6 : Login
7 : get Login Credentials
8 : Return Login Credentials
9 : Veiry whether login is success or not
10 : Login with Credentials
11 : Verify Login is Success or not
12 : Generate report based on login Result
13 : Navigate to Information Archiving Page
14 : Go to Information Archiving Page
15 : Verify users tab Opened or not
16 : Check Whether information Archiving Page is opened or not
17 : Verify Data Flow Graph
18 : Fetch fields from Data Flow Graph
19 : Get Values from Data base
20 : Return values
21 : Check whether UI and database Fields are equal
22 : Add an entry in report based on the result

Figure 5.2.4: Validation of Data Flow Graph

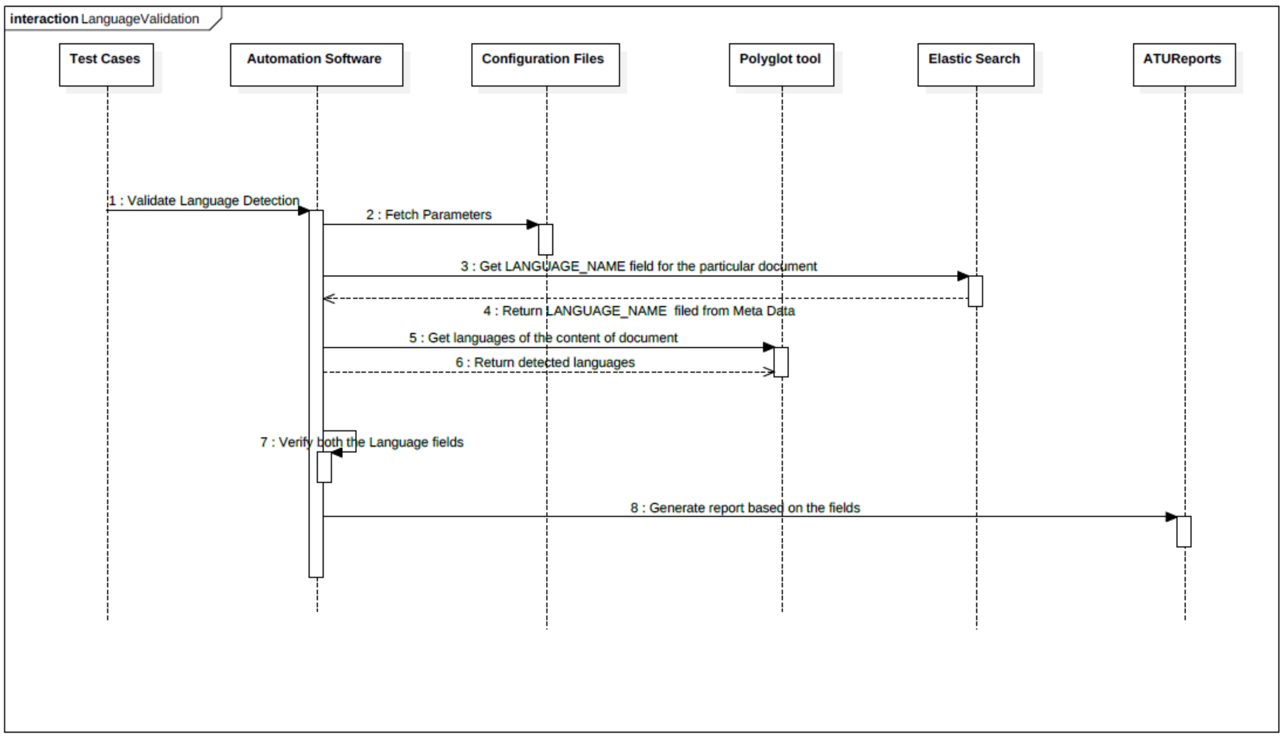

interaction LanguageValidation
Test Cases
Automation Software
Configuration Files
Polyglot tool
Elastic Search
ATUReports
1 : Validate Language Detection
2 : Fetch Parameters
3 : Get LANGUAGE_NAME field for the particular document
4 : Return LANGUAGE_NAME filed from Meta Data
5 : Get languages of the content of document
6 : Return detected languages
7 : Verify both the Language fields
8 : Generate report based on the fields

Figure 5.2.5: Automation of Language Detection Validation

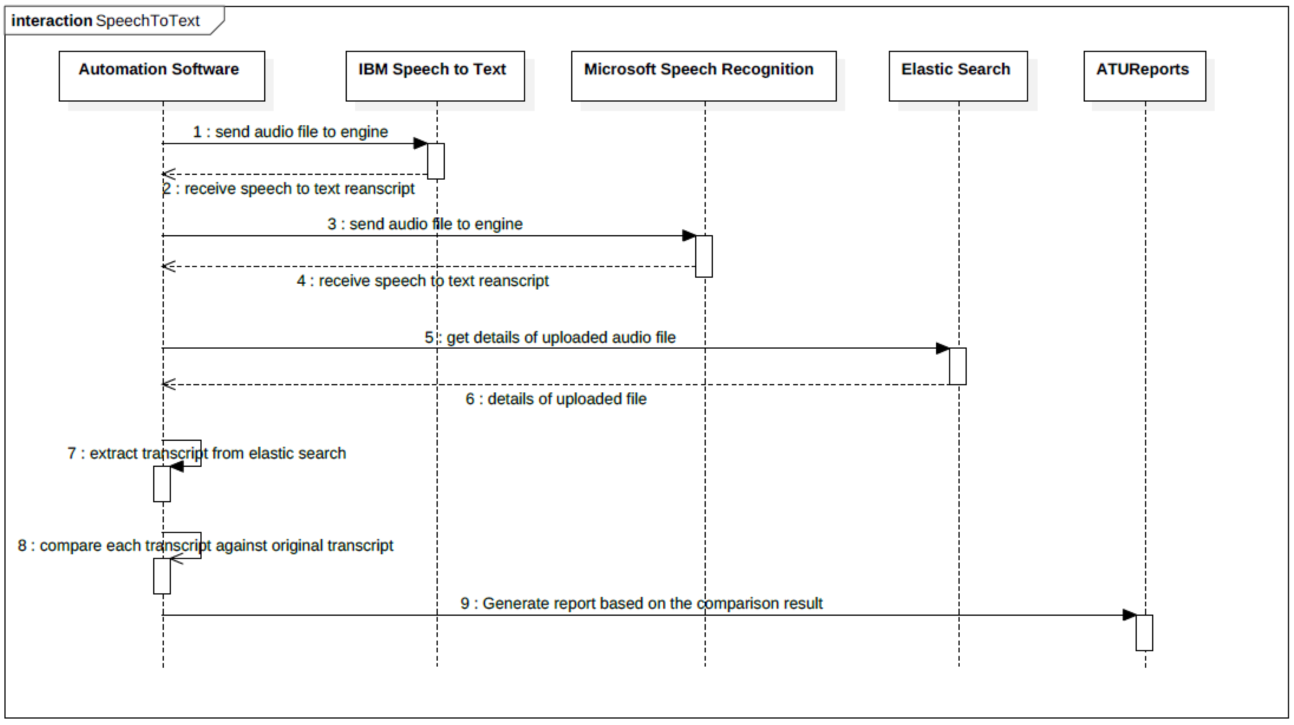

interaction SpeechToText
Automation Software
IBM Speech to Text
Microsoft Speech Recognition
Elastic Search
ATUReports
1 : send audio file to engine
2 : receive speech to text reanscript
3 : send audio file to engine
4 : receive speech to text reanscript
5 : get details of uploaded audio file
6 : details of uploaded file
7 : extract transcript from elastic search
8 : compare each transcript against original transcript
9 : Generate report based on the comparison result

Figure 5.2.6: Automation of Speech to Text Validation

5.3 UI Design

Since the project is about automation of testing scenarios, user interface is not available.

5.4 Updated RTM

Req. No.	Req. Description	System Design (Architecture)	Detailed Design	Implementation	Testing
F1.1	Create User	Section 5.1 Figure 5.2.1			
F1.2	Edit User	Section 5.1 Figure 5.2.1			
F1.3	Delete User	Section 5.1 Figure 5.2.1			
F2.1	Create Role	Section 5.1 Figure 5.2.2			
F2.2	Edit Role	Section 5.1 Figure 5.2.2			
F2.3	Delete Role	Section 5.1 Figure 5.2.2			
F3.1	Create Group	Section 5.1 Figure 5.2.3			
F3.2	Edit Group	Section 5.1 Figure 5.2.3			
F4	Validation of Data Flow Graph	Section 5.1 Figure 5.2.4			
F5	Validation of Language Detection of Documents	Section 5.1 Figure 5.2.5			
F6	Speech to Text Validation	Section 5.1.4 – 5.1.6 Figure 5.2.6			

Figure 5.1: Updated RTM

6. Design

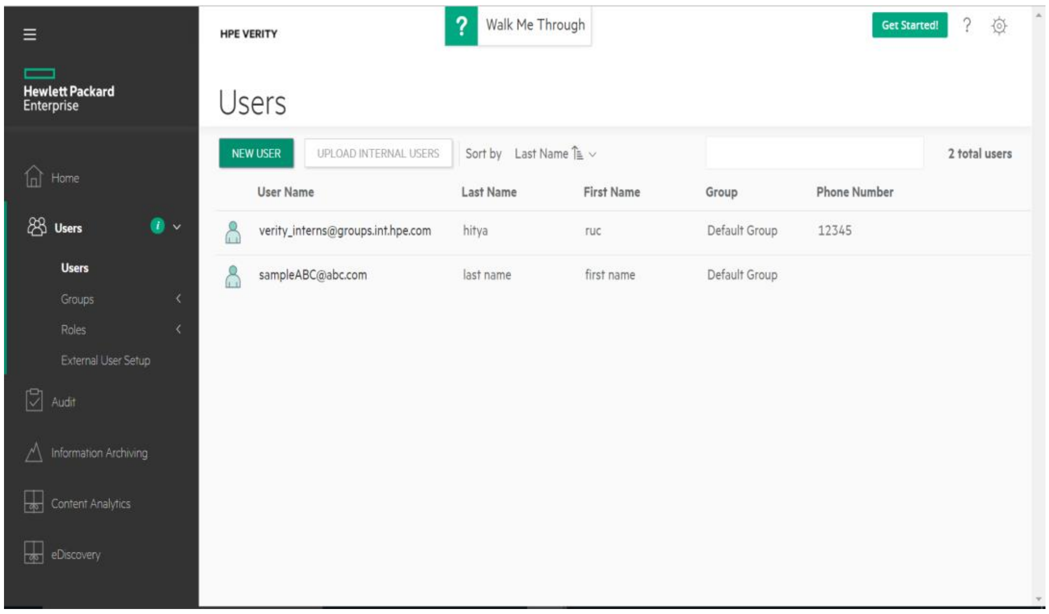
6.1 Detailed design of Modules

6.1.1 Automation of Users Module

Users plays as an important entity in the data archiving. At last users are the one who is going involve with the archiving product. Once the data is archived, users are created with certain privileges to maintain and govern the data. Hence while creating a user, one can assign them the certain roles or permissions. Generally administrator will have all the access to the archived data. Later, he is the one who can create other users and assign them the role. One can also have all the accesses.

6.1.1.1 Create a User

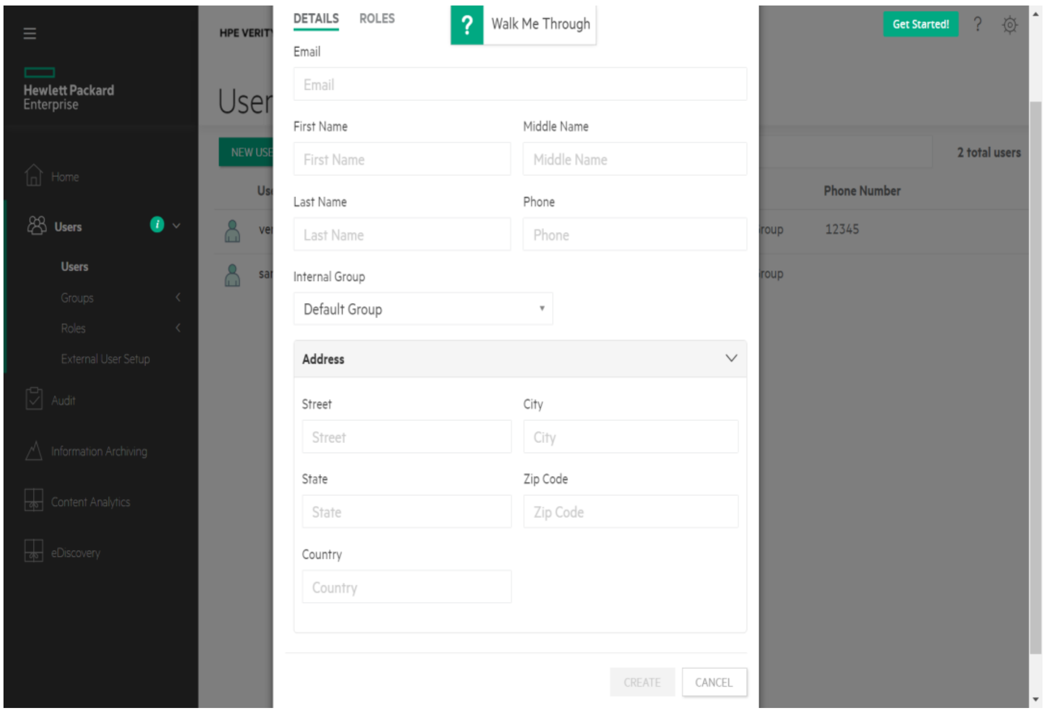
In order to create a user, one has to sign in with the administrator account. After signing in, navigate to the user's page to see the existing users. Admin can also delete, edit or create the new users. Figure 6.1 is the typical UI of the user's page.



User Name	Last Name	First Name	Group	Phone Number
verity_interns@groups.int.hpe.com	hitya	ruc	Default Group	12345
sampleABC@abc.com	last name	first name	Default Group	

Figure 6.1: Users Module

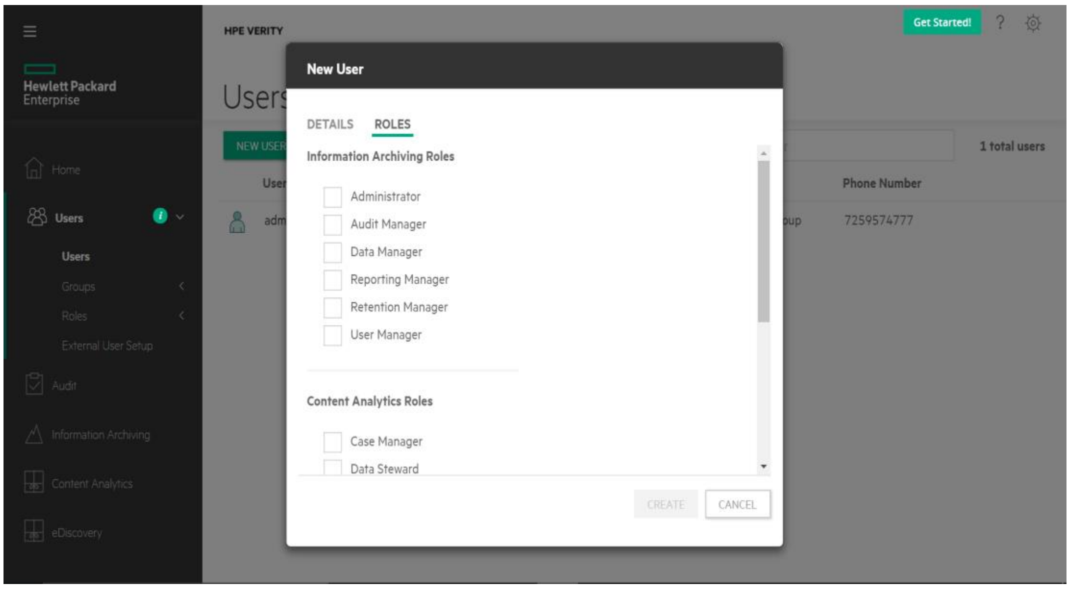
User's involves many fields, email address, first name, last name, phone number, group to which he belongs to, address and permissions. Out of these information, email, first name and last name are mandatory fields. Email address also needs to be in proper format. Hence automate for the various email formats which needs to be taken and also do the negative testing, in which UI should show an error message for the invalid email format. Email address can take 65 character length string for each part of the email address. Hence automation of this validation need to be done so that it can be run as a basic test for daily builds. Typical UI to create the user is shown in Figure 6.2 and 6.3.



Email: Email
First Name: First Name
Middle Name: Middle Name
Last Name: Last Name
Phone: Phone
Internal Group: Default Group
Address
Street: Street
City: City
State: State
Zip Code: Zip Code
Country: Country

Figure 6.2: Creating a User

All the fields can take up to 50 characters. Also, new user may be assigned to certain roles. So that administrator can provide certain privileges to this user. So, while creating a user, also automate the procedure to assign him certain roles as shown in Figure 6.3. Since users consists of various fields, many test cases is generated based on the combination of the input fields.



New User
DETAILS ROLES
Information Archiving Roles
- Administrator
- Audit Manager
- Data Manager
- Reporting Manager
- Retention Manager
- User Manager

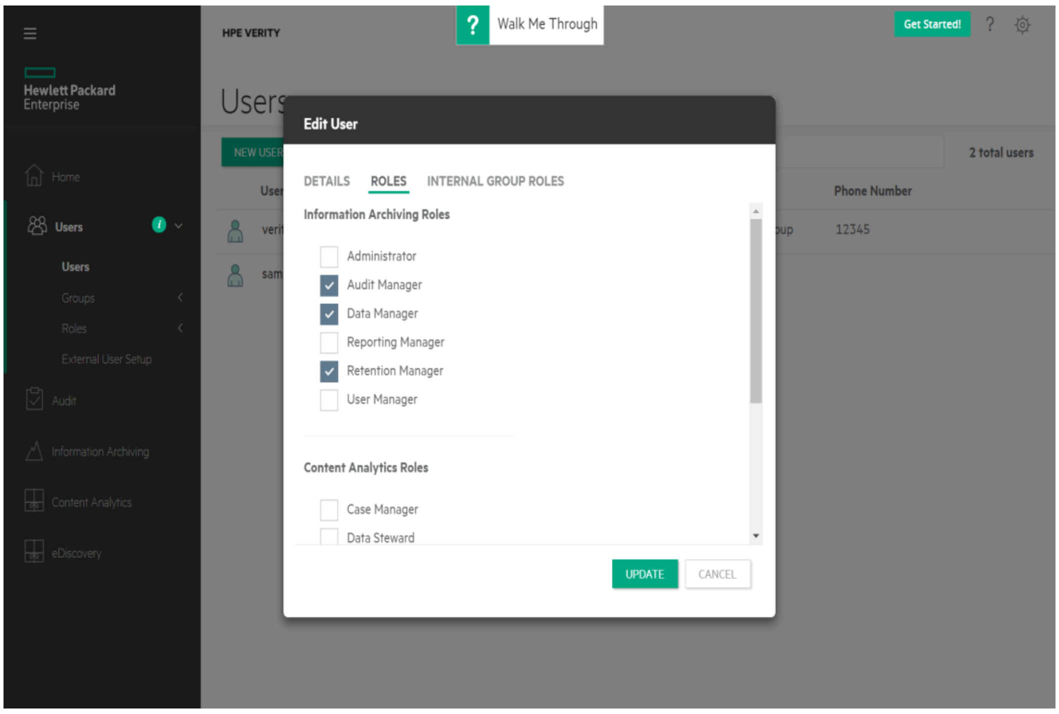
Content Analytics Roles
- Case Manager
- Data Steward

Figure 6.3: Creating a User

During the process of creation of user, total number of users are validated for the UI elements. Hence, number of users must one greater than the initial value. Along with this, we validated the created user is successfully created or not by the checking in the UI. Since email is considered as username, while creating the new user, if we give existing email as email address, it will show an error saying username is already exist and asks us to give different email. However this restriction is not applicable to first name and last name. Until we give mandatory fields, create button will be disabled. Once we give these entries with the certain constraints like uniqueness, maximum length etc., we can create the new user.

6.1.1.2 Edit a User

Once the users is created, we can also edit the created user. Edit button is given at the end of the each user row. Here, we cannot modify the details of administrator. All the previous details given can be modified, however, similar to creation of new user, modified email address should be unique. We can also change the roles assigned to the user. While automating, if we want to change the role assigned to particular user, first we need to clear the existing permissions, then only we can assign the new permission. So that, by doing this we can simplify the various complications. Similar to creation of users, until we give mandatory fields, we cannot update the user details. After updating, we verify the total number of users before and after the editing. Here, count should be same. And also we verify whether the updated username is present in the user's UI. The UI for editing the user is almost similar to the UI of create user, however, the current details of the user is shown in the Pop-up. The UI to modify the existing permission is shown in Figure 6.4.



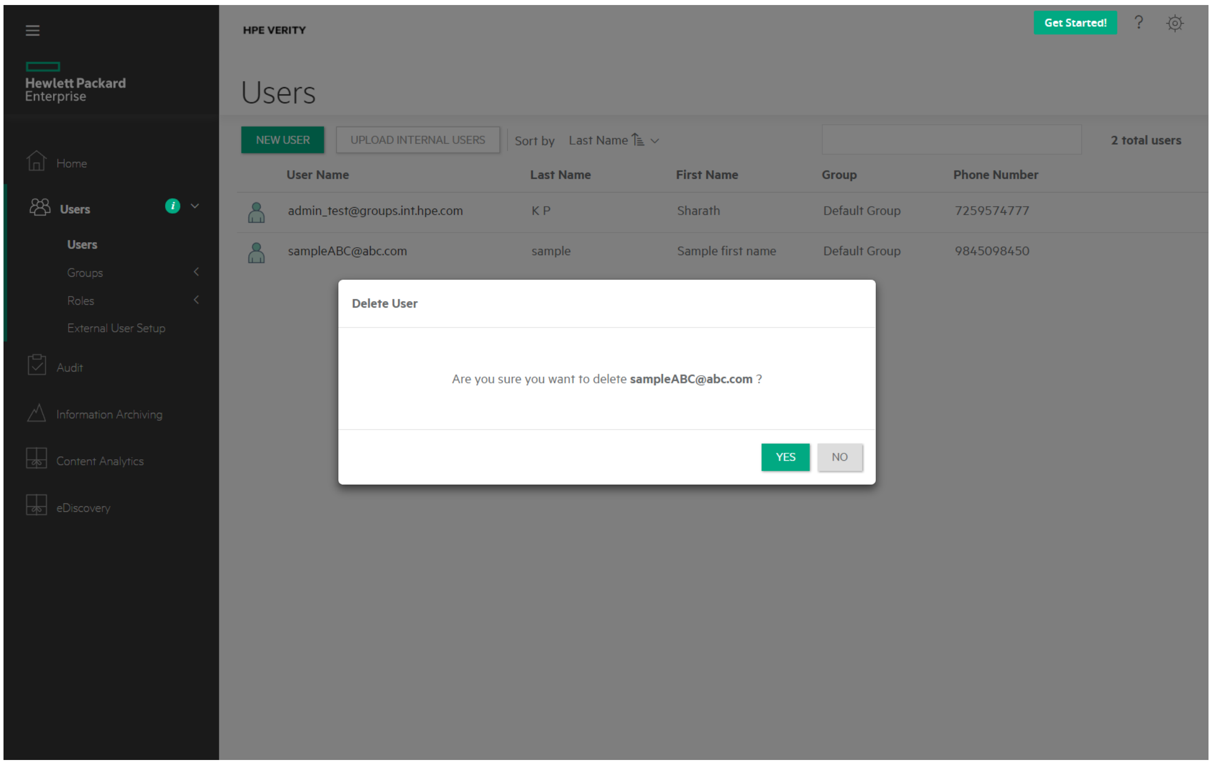
Information Archiving Roles
- [ ] Administrator
- [✓] Audit Manager
- [✓] Data Manager
- [ ] Reporting Manager
- [✓] Retention Manager
- [ ] User Manager

Content Analytics Roles
- [ ] Case Manager
- [ ] Data Steward

Figure 6.4: Edit User

6.1.1.3 <u>Delete a User</u>

Created users can also be deleted. Hence, during the process of automation, once the edition process is completed, these users can be deleted. Delete button is available in each user row, next to the edit button. However, permission to delete the administrator account is not possible. As shown in Figure 6.5, after selecting the delete option, confirmation pop-up is displayed.



User Name	Last Name	First Name	Group	Phone Number
admin_test@groups.int.hpe.com	K P	Sharath	Default Group	7259574777
sampleABC@abc.com	sample	Sample first name	Default Group	9845098450

Delete User

Are you sure you want to delete **sampleABC@abc.com** ?

Figure 6.5: Delete a User

Selected user will be deleted after agreeing to the confirmation message. Similar to creation, edition of users, and total number of user is checked. In this case, count should less than one to the initial value. Also the deleted user is not displayed in the UI. Hence, this validation is done by the automation.

6.1.2 <u>Automation of roles module</u>

One of the requirement is to automate the process of various scenarios in roles module. These roles are assigned to internal roles, so that functioning of product is well organized and well modularized.

These roles can be assigned to users as well as to a group (a group of users). Based on which role the user is assigned, he/she will see be able to access and view only those parts of the site which correspond to their permissions. For example, an Audit Manager will see only the Data

Analytics part of the tool and will not be able to view the creation of groups and users, whereas an Administrator will have access to all the services. To test the functionality of the group and to automate this testing, the following methods were followed to validate the different test cases.

6.1.2.1 Creating a new role

To create a new role, one has to navigate to the Users>Roles>Internal Roles Page. Here the button which stands for the creation of a new role is clicked and a dialog box opens in the same window. Here the details such as the name of the role and the description of the role must be entered. It is to be noted that the name should not exceed 50 characters and the description should not exceed 255 characters.

The Role name cannot contain commas but can however contain special characters and text from other languages. Test cases or the same are to be taken into consideration. After entering the role name and permissions, one must click on the next tab in the dialog box which reads Permissions. There are a set of predefined permissions relating to the roles mentioned above. The required one has to be chosen by click on a checkbox next to the string. Multiple permissions can be given for a new Role. After checking the permissions for the new role, there are two buttons on screen, these are, create and cancel button.

Click on create to successfully add this new role to the set of existing ones. In case after trying to create by clicking on the create button, the user is not able to do so and an error message pops up, the cancel button is clicked and a message is flagged into the ATU Reports stating the reason of failure of the test case. If created, the dialog box closes and a message saying that the role has been created, is displayed. There is a number on the top left hand side, which indicated the total number of roles. If a role gets created, then this number gets incremented. The test case is made to pass under the conditions that this number increments by 1 after the creation of the role, the role name is verified and seen in the list of all role names displayed on screen. To open the role (click the corresponding XPath or id) and a side panel opens up which contains the role name, role description and role permissions.

By taking the ids of all the parameters there, we verify these properties of the new role that has just been created. Other test cases for the creation of the role include, creating the role with Chinese, Japanese, Thai, Korean and other local language characters. Checking if these properties take special characters as the value and verifying the new roles being created.

6.1.2.2 <u>Editing a Role</u>

Once created a role can be edited. The role name, role description and the permissions can be changed. There is an edit button for every role in the list of roles next to the names. This button is clicked and opens a dialog box wherein you can edit the name, description and move onto the next tab and edit the permissions as well. Once you are done editing, you will have to click the Update button. Test cases have to done to make sure that the updated role name does not coincide with another existing name in the list, else it would throw an error.

6.1.2.3 <u>Deleting a Role</u>

When automating test cases, it is a good practice to delete the roles or other instances you have created. Else, when you run the suite of test cases over again, there would be an error which would say that the following roles have already been created. So after every creation, in the test cases, the new role is deleted. For deleting a role, we do not simply remove it. There are certain user and groups which may have been assigned this role. To avoid uncertainty, whenever you delete a role, you assign another role which will take on its functionality. So if you wanted to delete the audit manager role, you give the other role as Administrator so that the administrators can watch over the users and groups that were under Audit manager previously. So to delete the role, we click on the delete button that is present next to each of the role names in the list of roles. Once we click this button, the dialog box opens up and asks for a replacement role which has to be chose. A Delete and cancel button are available, and the delete button is clicked which closes the dialog box and removes that role. The total number of groups will decrement by one and this is verified by the code.

6.1.3 <u>Automation of group's module</u>

Different users can be combined together to form groups. This makes the workflow easier and manageable. Groups are created with specific roles for each group. The roles have been discussed in the previous section. Once these groups are created, the existing as well as new users can be assigned to a particular group indicating that they will have the roles as mentioned in the group. There is a default group as well which a user is automatically assigned to. The list of all created groups along with the default group is displayed in the Users>Groups Page. Following is the stapes taken in order to automate.

6.1.3.1 Creating a new Group

To create a new group, one has to navigate to the Users>Roles>Internal Groups Page. Here the button which stands for the creation of a new group is clicked and a dialog box opens in the same window. Here the details such as the name of the group and the description of the group must be entered. It is to be noted that the name should not exceed 50 characters and the description should not exceed 255 characters. The Group name cannot contain commas but can however contain special characters and text from other languages. Test cases or the same are to be taken into consideration. After entering the group name and description, one must click on the next tab in the dialog box which reads roles. Here the different roles for a group are to be selected. There are a set of predefined roles, the required one has to be chosen by clicking on a checkbox next to the text string. Multiple roles can be given for a new Group.

After selecting the roles for the new group, there are two buttons on screen which are the ‘create’ and ‘cancel’ button. Click on create to successfully add this new group to the set of existing ones. If created, the dialog box closes and a message saying that the group has been created, is displayed. There is a number on the top left hand side, which indicated the total number of groups. If a group gets created, then this number gets incremented. The test case is made to pass under the conditions that this number increments by 1 after the creation of the group, the group name is verified and seen in the list of all group names displayed on screen. And this is clicked open (by giving the respective XPath or id and a side panel opens up which contains the group name, group description and group roles. By taking the ids of all the parameters there, we verify these properties of the new group that has just been created. Other test cases for the creation of the group include, creating the group with Chinese, Japanese, Thai, Korean and other local language characters. Checking if these properties take special characters as the value and verifying the new groups being created.

6.1.3.2 Editing a Group

Once created a group can be edited. The group name, group description and the roles can be changed. There is an edit button for every group in the list of groups next to the names. This button is clicked and opens a dialog box wherein you can edit the name, description and move onto the next tab and edit the permissions as well. Once you are done editing, you will have to click the Update button. Test cases have to done to make sure that the update the edit group name does not coincide with another existing name in the list, else it would throw an error.

Delete a Group: Presently, a user cannot delete a group that has been created. To remove all the groups created by one particular user, one can remove the user and create an account again. Otherwise, there would be an error for all the test cases saying that a group with the same name already exists.

6.1.4 Information Archiving – Validation of Data flow graph

As explained in functionalities of the project in previous chapter, one of the task is to automate the validation process of Data Flow graph. Once the data is uploaded, it goes through various stages before storing. Once data is uploaded, we archive those data, after archiving, we extract the data and process those. Once the data processing is over we get lot of information related to the each file uploaded.

In Verity, uploaded data is stored in two places, Meta data related to the file is stored in HPE's very own Vertica Database. And along with the Meta data, primary content of the data is stored in Elastic Search. Elastic search is open source tool available as a search engine. Various other open source tools are integrated with the elastic search tool in order to achieve the various data analysis. Figure 6.6 shows all the information regarding the uploaded data.

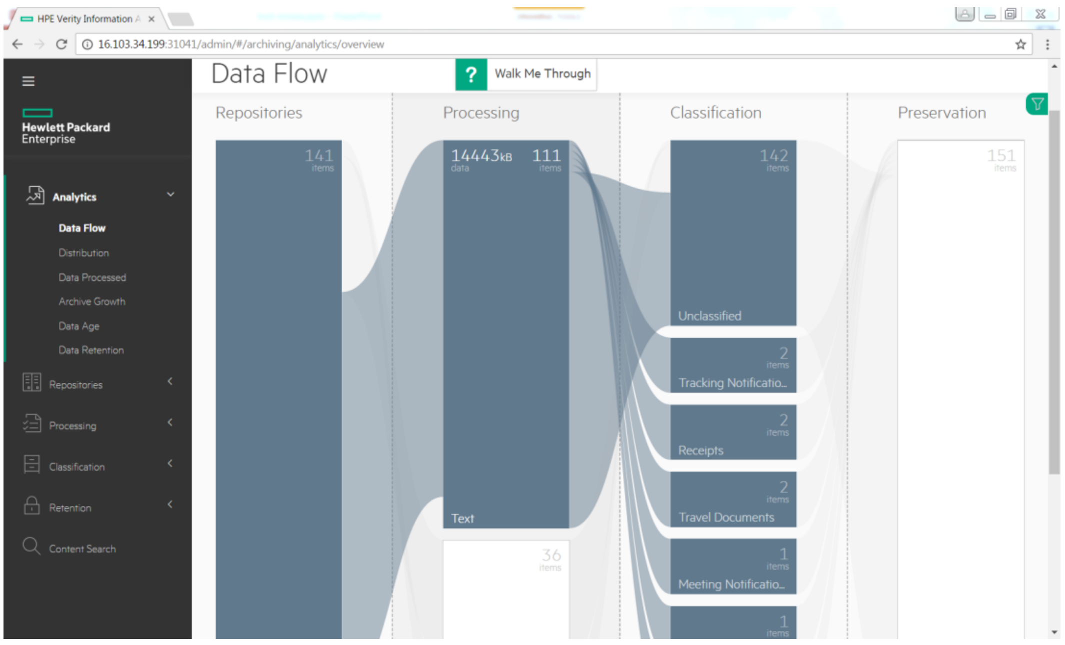


Figure 6.6: Data Flow Graph which needs to be validated

The data flow graph categorize the uploaded data based on type of data like text document, Optical Character recognized files and audio files. Also with this, it can further categorize as format of data uploaded like whether it is email, if so what type of email it is like, Social Networking Notification, Meeting Notification, Newsletters, Receipts, Travel documents etc.

along with these, graph will also shows the total size of the categorized data, and total number of these data. So, automation task is to validate these fields with respect to back end data and UI data.

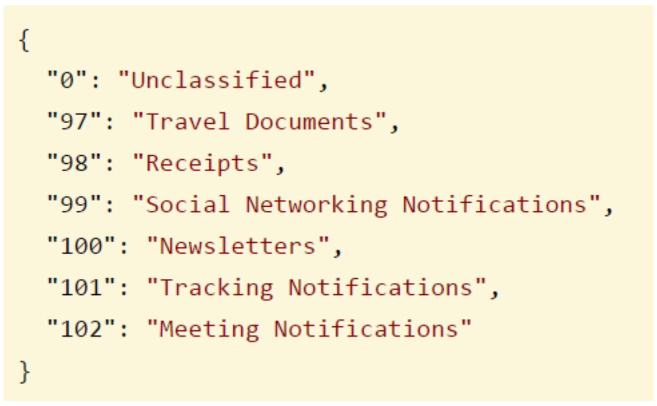
Since Meta data is stored in Vertica data base, type of the file is stored in the items table in Vertica. When the user is created, the new schema is create for the user, inside the schema items is one of the table which contains all the Meta data information of the archived data. In UI, size of the data is stored in only one unit. Either everything will be displayed in KB's or in MB's. However, in Vertica database, file size is stored in Bytes. Here, while validating, convert the aggregate value from DB into the unit which is being displayed in Information Archiving UI.

In the Data Flow graph, all the elements are displayed using SVG, hence, XPath cannot be used to locate these elements directly. Due to this limitations approach being used in XPath is modified to get the values stored in SVG elements. Here, instead of referring to the SVG elements directly, write generalized XPath locations, like //* instead of //svg in XPath. Once these UI values are available, fetch the values stored in Vertica database. Hence, query is made to the Vertica database to get the corresponding fields. Since there are around 15 Queries to get the required values from Vertica database, instead of creating separate connection, single connection is used to query the Vertica database. Typical query is as follows:

```
Select sum(file_size) from account_1856533787081451 52.item where
POLICY_MATCHED_POLICYID=44 and is_ole='false' and secondary_row is null;
```

The above query fetches the total size of the Receipts. POLICY_MATCHED_POLICYID is the number mapped to the Receipts category. account_[integer] being the schema related to the administrator. Integer is the realm ID associated with administrator. Hence, this realm ID is unique for each on-boarded user. Once aggregate bytes count is received as a query response, convert it into particular unit which is being represented in UI.

Since, text documents are classified further as Social Networking Notification, Receipts etc., various policy id's are stored in database instead of storing classification names. Hence, Rest Assured API call is made to get the corresponding mappings and these mappings are used while making query to the Vertica database.



```
{
  "0": "Unclassified",
  "97": "Travel Documents",
  "98": "Receipts",
  "99": "Social Networking Notifications",
  "100": "Newsletters",
  "101": "Tracking Notifications",
  "102": "Meeting Notifications"
}
```

Figure 6.7: Policy ID and classification mappings

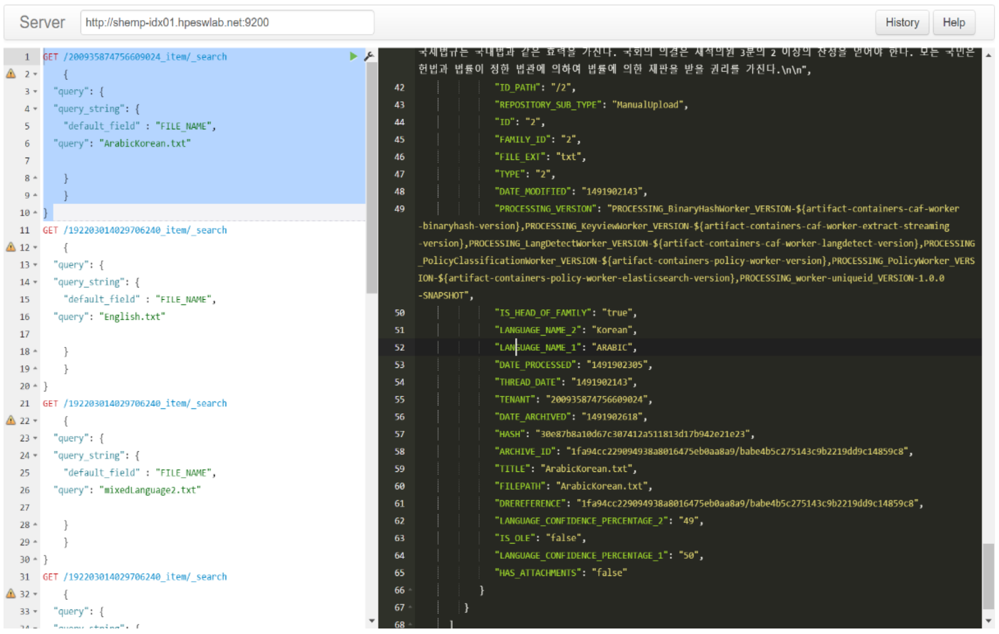
Once both values are available, these values are validated against each other and corresponding entry is made in ATUReports.

6.1.5 Automating the Validation of Language Detection

As explained in the Literature Survey, we are using the polyglot module in the python to detect the language of the document. In order to validate against the Meta data stored in Verity, fetch these Meta data from the elastic search. Since elastic search follows its own query language, syntax of JSON query language is followed to fetch these Meta data from the elastic search. Make the query to elastic by Rest Assured API calls. The result is received from the elastic is JSON object. For that, Java's JSON library is used to parse the JSON result from elastic search and get the required information. For language validation, LANGUAGE_NAME_1 field is needed. Hence extract this field from the JSON object. Incase language contains multiple language, elastic search will detect these languages and stored in LANGUAGE_NAME_1, LANGUAGE_NAME_2 and so on. In order to get these from the elastic search, realm ID of the on-boarded user is required. By using the realm ID, select the appropriate schema and inside this schema, make the query to the item table. The sample query is as follows:

```
GET /200935874756609024_item/_search{
        "query": {
                "query_string": {
                        "default_field" : "FILE_NAME",
                        "query": "ArabicKorean.txt"
                     }
        }
}
```

Here, 200935874756609024 is the realm ID, and selecting the item table as 200935874756609024_item. After this, make the required query on the item table. Here, query is made on the FILE_NAME fields and searching for the ArabicKorean.txt value in the FILE_NAME. After matching result is found, get all the Meta data related to that file. Screen shot of the sample elastic query along with the query result is shown in Figure 6.8.



```
GET /200935874756609024_item/_search
    {
  "query": {
  "query_string": {
    "default_field" : "FILE_NAME",
  "query": "ArabicKorean.txt"

    }
    }
}
GET /192203014029706240_item/_search
    {
  "query": {
  "query_string": {
    "default_field" : "FILE_NAME",
  "query": "English.txt"

    }
    }
}
GET /192203014029706240_item/_search
    {
  "query": {
  "query_string": {
    "default_field" : "FILE_NAME",
  "query": "mixedLanguage2.txt"

    }
    }
}
GET /192203014029706240_item/_search
    {
  "query": {
  "query_string": {
```

```
국제법규는 국내법과 같은 효력을 가진다. 국회의 의결은 재적의원 3분의 2 이상의 찬성을 얻어야 한다. 모든 국민은
헌법과 법률이 정한 법관에 의하여 법률에 의한 재판을 받을 권리를 가진다.\n\n",
            "ID_PATH": "/2",
            "REPOSITORY_SUB_TYPE": "ManualUpload",
            "ID": "2",
            "FAMILY_ID": "2",
            "FILE_EXT": "txt",
            "TYPE": "2",
            "DATE_MODIFIED": "1491902143",
            "PROCESSING_VERSION": "PROCESSING_BinaryHashWorker_VERSION-${artifact-containers-caf-worker
-binaryhash-version},PROCESSING_KeyviewWorker_VERSION-${artifact-containers-caf-worker-extract-streaming
-version},PROCESSING_LangDetectWorker_VERSION-${artifact-containers-caf-worker-langdetect-version},PROCESSING
_PolicyClassificationWorker_VERSION-${artifact-containers-policy-worker-version},PROCESSING_PolicyWorker_VERS
ION-${artifact-containers-policy-worker-elasticsearch-version},PROCESSING_worker-uniqueid_VERSION-1.0.0
-SNAPSHOT",
            "IS_HEAD_OF_FAMILY": "true",
            "LANGUAGE_NAME_2": "Korean",
            "LANGUAGE_NAME_1": "ARABIC",
            "DATE_PROCESSED": "1491902305",
            "THREAD_DATE": "1491902143",
            "TENANT": "200935874756609024",
            "DATE_ARCHIVED": "1491902618",
            "HASH": "30e87b8a10d67c307412a511813d17b942e21e23",
            "ARCHIVE_ID": "1fa94cc229094938a8016475eb0aa8a9/babe4b5c275143c9b2219dd9c14859c8",
            "TITLE": "ArabicKorean.txt",
            "FILEPATH": "ArabicKorean.txt",
            "DREREFERENCE": "1fa94cc229094938a8016475eb0aa8a9/babe4b5c275143c9b2219dd9c14859c8",
            "LANGUAGE_CONFIDENCE_PERCENTAGE_2": "49",
            "IS_OLE": "false",
            "LANGUAGE_CONFIDENCE_PERCENTAGE_1": "50",
            "HAS_ATTACHMENTS": "false"
          }
        }
      ]
```

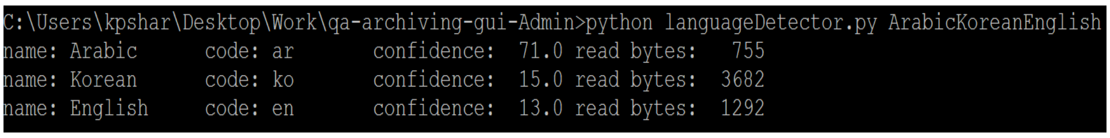
Figure 6.8: Language Meta data in Elastic Search

These fields are validated against the Polyglot library. In order to get the result from Polyglot, python script is wrote, which internally uses polyglot. The sample code is shown below:

```
import codecs
from polyglot.detect import Detector
f = codecs.open("C:\\Users\\kpshar\\Desktop\\Work\\Language Detection Dataset\\UTF-8\\ArabicKorean.txt", "r", "utf-8")
document_content = f.read()
for language in Detector(document_content).languages:
        print(language)
```

Here, detector instance of the Polyglot to detect the language of the document. codecs module is used to open the input file in a proper encoding. After this, pass the content of the file to detector instance and language/languages of document is used.

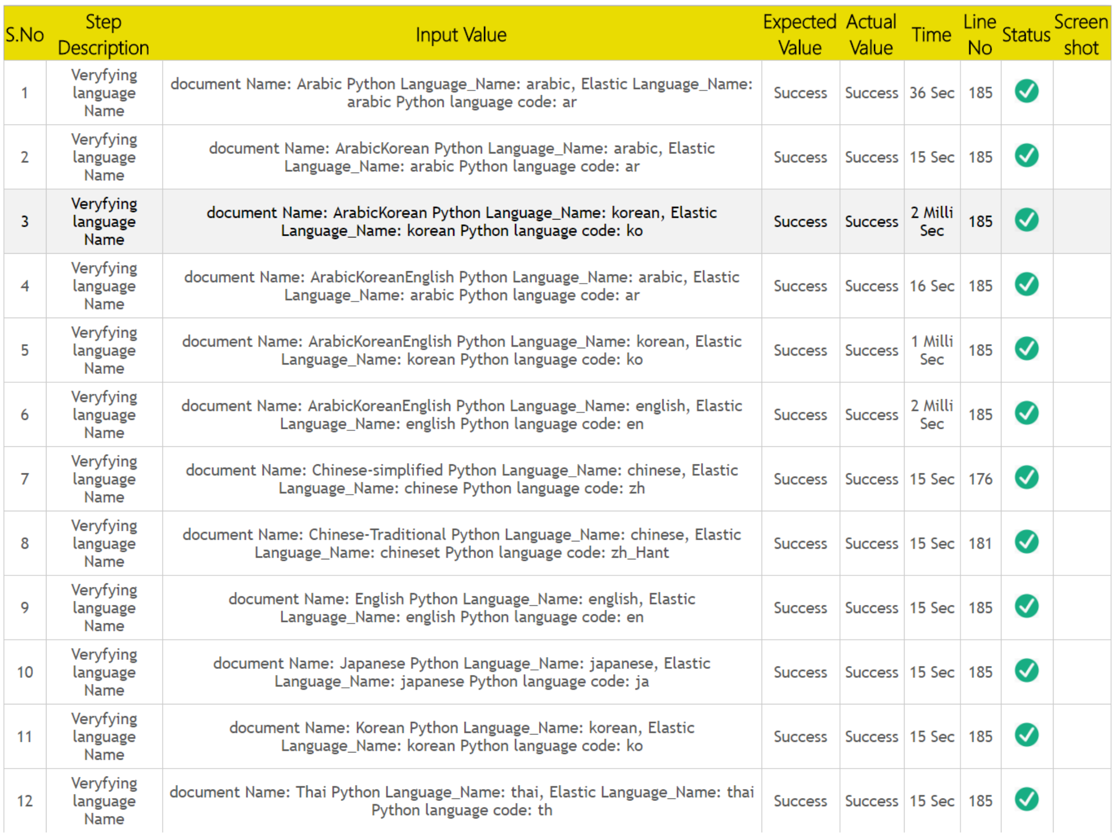
Run this python script from Java by creating a separate process. Get the result of the python script by using BufferedReader and InputStreamReader instances of Java. The typical result from the Polyglot is shown in figure 6.9.



```
C:\Users\kpshar\Desktop\Work\qa-archiving-gui-Admin>python languageDetector.py ArabicKoreanEnglish
name: Arabic      code: ar       confidence:  71.0 read bytes:   755
name: Korean      code: ko       confidence:  15.0 read bytes:  3682
name: English     code: en       confidence:  13.0 read bytes:  1292
```

Figure 6.9: Polyglot Output for ArabicKoreanEnglish.txt

In the above result, only language name is required. Hence, use regular expression to fetch this value. After having both the values from Polyglot python script and elastic search, validate these values using TestNG assertion method and based on the corresponding result, test case is noted in ATUReports as shown in Figure 6.10.



S.No	Step Description	Input Value	Expected Value	Actual Value	Time	Line No	Status	Screen shot
1	Veryfying language Name	document Name: Arabic Python Language_Name: arabic, Elastic Language_Name: arabic Python language code: ar	Success	Success	36 Sec	185	✓	
2	Veryfying language Name	document Name: ArabicKorean Python Language_Name: arabic, Elastic Language_Name: arabic Python language code: ar	Success	Success	15 Sec	185	✓	
3	Veryfying language Name	document Name: ArabicKorean Python Language_Name: korean, Elastic Language_Name: korean Python language code: ko	Success	Success	2 Milli Sec	185	✓	
4	Veryfying language Name	document Name: ArabicKoreanEnglish Python Language_Name: arabic, Elastic Language_Name: arabic Python language code: ar	Success	Success	16 Sec	185	✓	
5	Veryfying language Name	document Name: ArabicKoreanEnglish Python Language_Name: korean, Elastic Language_Name: korean Python language code: ko	Success	Success	1 Milli Sec	185	✓	
6	Veryfying language Name	document Name: ArabicKoreanEnglish Python Language_Name: english, Elastic Language_Name: english Python language code: en	Success	Success	2 Milli Sec	185	✓	
7	Veryfying language Name	document Name: Chinese-simplified Python Language_Name: chinese, Elastic Language_Name: chinese Python language code: zh	Success	Success	15 Sec	176	✓	
8	Veryfying language Name	document Name: Chinese-Traditional Python Language_Name: chinese, Elastic Language_Name: chineset Python language code: zh_Hant	Success	Success	15 Sec	181	✓	
9	Veryfying language Name	document Name: English Python Language_Name: english, Elastic Language_Name: english Python language code: en	Success	Success	15 Sec	185	✓	
10	Veryfying language Name	document Name: Japanese Python Language_Name: japanese, Elastic Language_Name: japanese Python language code: ja	Success	Success	15 Sec	185	✓	
11	Veryfying language Name	document Name: Korean Python Language_Name: korean, Elastic Language_Name: korean Python language code: ko	Success	Success	15 Sec	185	✓	
12	Veryfying language Name	document Name: Thai Python Language_Name: thai, Elastic Language_Name: thai Python language code: th	Success	Success	15 Sec	185	✓	

Figure 6.10: Language Detection Report

6.1.6 Automation of Speech to Text Validation

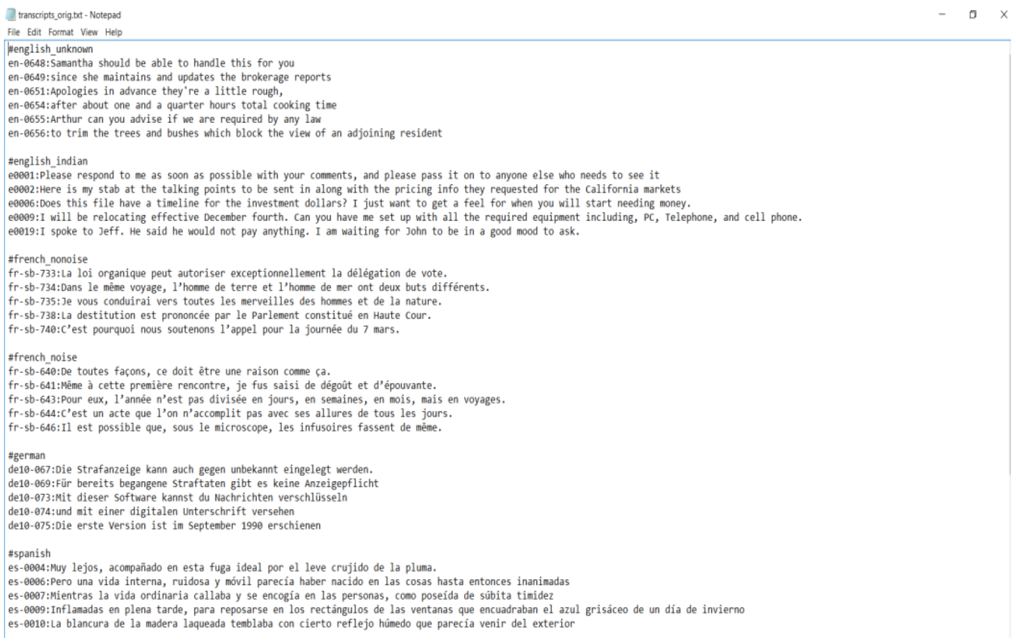
Speech to text validation proceeds in much the same way as the above approache. Elastic search is queried to get the extracted content, which comes mixed with metadata. The first step is thus to parse the JSON object returned and extract only the relevant content. Silent regions are marked with <sil> tokens, which also have to be removed to perform comparisons. Elastic search queries are made using rest assured API calls.

A file containing transcripts of all uploaded audio corpus files was created. The calls to database and writing to the files were all automated. In case new files are added, its name needs to be added to a file containing all uploaded file names. Rest of the process is handled automatically.

To validate the performance of the internal tool handling speech to text applications, there needs to be comparisons with:

1. Manual transcriptions
2. Other industry standard tools

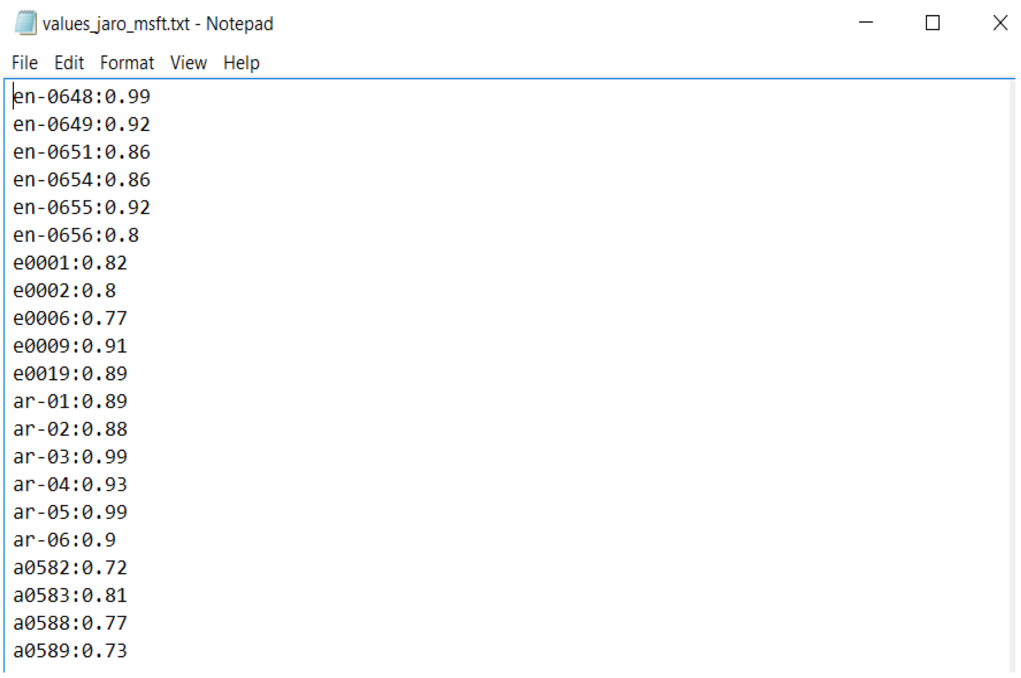
For step 1, a file containing accurate transcriptions of all files is created, as in figure 6.11. This is a manual process as no tool can guarantee accuracy all the time.



```
#english_unknown
en-0648:Samantha should be able to handle this for you
en-0649:since she maintains and updates the brokerage reports
en-0651:Apologies in advance they're a little rough,
en-0654:after about one and a quarter hours total cooking time
en-0655:Arthur can you advise if we are required by any law
en-0656:to trim the trees and bushes which block the view of an adjoining resident

#english_indian
e0001:Please respond to me as soon as possible with your comments, and please pass it on to anyone else who needs to see it
e0002:Here is my stab at the talking points to be sent in along with the pricing info they requested for the California markets
e0006:Does this file have a timeline for the investment dollars? I just want to get a feel for when you will start needing money.
e0009:I will be relocating effective December fourth. Can you have me set up with all the required equipment including, PC, Telephone, and cell phone.
e0019:I spoke to Jeff. He said he would not pay anything. I am waiting for John to be in a good mood to ask.

#french_nonoise
fr-sb-733:La loi organique peut autoriser exceptionnellement la délégation de vote.
fr-sb-734:Dans le même voyage, l’homme de terre et l’homme de mer ont deux buts différents.
fr-sb-735:Je vous conduirai vers toutes les merveilles des hommes et de la nature.
fr-sb-738:La destitution est prononcée par le Parlement constitué en Haute Cour.
fr-sb-740:C’est pourquoi nous soutenons l’appel pour la journée du 7 mars.

#french_noise
fr-sb-640:De toutes façons, ce doit être une raison comme ça.
fr-sb-641:Même à cette première rencontre, je fus saisi de dégoût et d’épouvante.
fr-sb-643:Pour eux, l’année n’est pas divisée en jours, en semaines, en mois, mais en voyages.
fr-sb-644:C’est un acte que l’on n’accomplit pas avec ses allures de tous les jours.
fr-sb-646:Il est possible que, sous le microscope, les infusoires fassent de même.

#german
de10-067:Die Strafanzeige kann auch gegen unbekannt eingelegt werden.
de10-069:Für bereits begangene Straftaten gibt es keine Anzeigepflicht
de10-073:Mit dieser Software kannst du Nachrichten verschlüsseln
de10-074:und mit einer digitalen Unterschrift versehen
de10-075:Die erste Version ist im September 1990 erschienen

#spanish
es-0004:Muy lejos, acompañado en esta fuga ideal por el leve crujido de la pluma.
es-0006:Pero una vida interna, ruidosa y móvil parecía haber nacido en las cosas hasta entonces inanimadas
es-0007:Mientras la vida ordinaria callaba y se encogía en las personas, como poseída de súbita timidez
es-0009:Inflamadas en plena tarde, para reposarse en los rectángulos de las ventanas que encuadraban el azul grisáceo de un día de invierno
es-0010:La blancura de la madera laqueada temblaba con cierto reflejo húmedo que parecía venir del exterior
```

Figure 6.11: Transcripts for the audio corpus

Next, Levenshtein distance and Jaro-Winkler distance are calculated for all files and written to a file, as shown below. The two strings are read from the manual transcript and elastic search-read transcript for each computation. Results of this process are show in figure 6.12. All such processes have been automated for ease of use in the future.



```
en-0648:0.99
en-0649:0.92
en-0651:0.86
en-0654:0.86
en-0655:0.92
en-0656:0.8
e0001:0.82
e0002:0.8
e0006:0.77
e0009:0.91
e0019:0.89
ar-01:0.89
ar-02:0.88
ar-03:0.99
ar-04:0.93
ar-05:0.99
ar-06:0.9
a0582:0.72
a0583:0.81
a0588:0.77
a0589:0.73
```

Figure 6.12: Jaro-Winkler distance For Transcripts from Bing Speech API

For step 2, the same process is repeated, but with transcripts obtained from other external tools, like Microsoft Bing Speech, IBM Watson, etc.

To make queries to these external tools, requests have to be sent with appropriate authentication, as determined by the tool's owners. For example, to use Bing Speech to text API, first an authentication token needs to be obtained. This key is valid for ten minutes. Using this authentication as a header, and adding few other required headers, API call has to be made. Query parameters such as the language used, system ID, application ID, file type, sampling rate, etc. need to be mentioned. The audio file is sent as a binary file. HTTP connection is used.

To automate this process, there is a need to monitor the authentication refresh time, store the authentication token and so on. Files are used for this process as regardless of whether the application is running or not, these values need to be refreshed once the time limit is crossed.

Transcripts are returned as JSON objects, which again have to be parsed to extract the required data, which is subsequently written to a file, as mentioned previously. Once this is done, Levenshtein and Jaro-Winkler values are again computed for these strings by comparing with the manually generated transcript.

Once transcripts are obtained from each of the external tools, a visualization comparing the performance of the internal tool with the rest is generated, as in figure 6.13. The lines in bold are the results obtained from the internal tool.

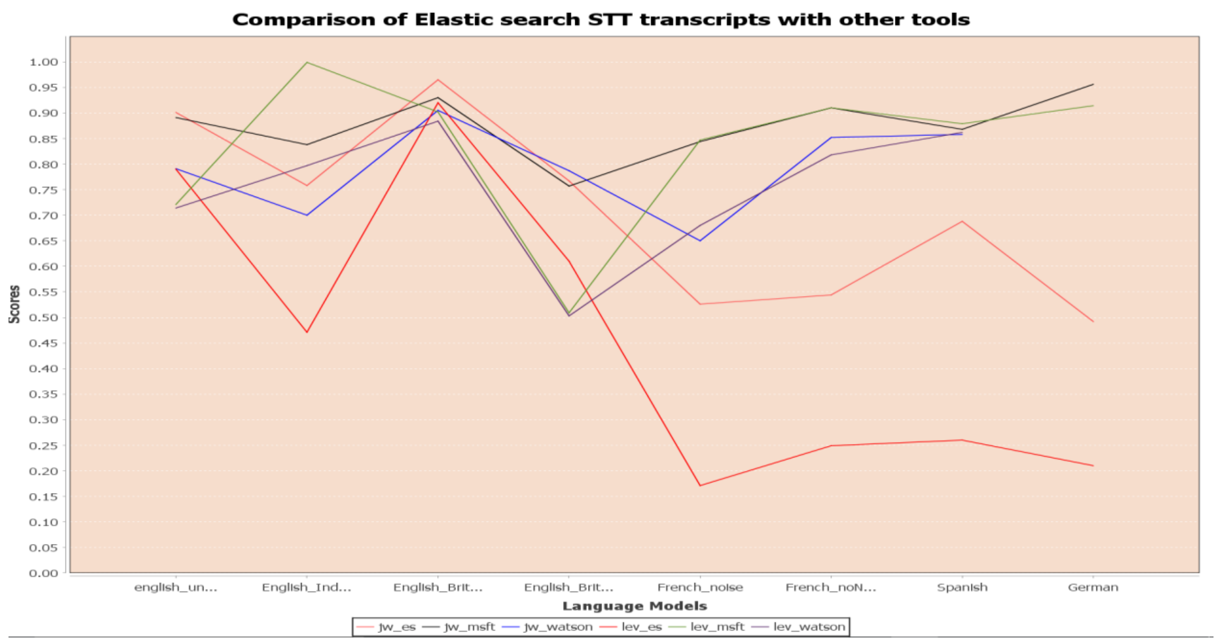


Figure 6.13: Performance of Various Tools with Different Language Sets

It is observed that while the internal tool works well with English in American and mild British accents, performance degrades quite sharply. Also, the number of languages is limited. The one advantage is that there is no constraint on the length of the input file. This was a constraint at least in free versions of the external tools. Remedies for the same were proposed. Control over the detected language was one such solution.

6.2 Data Flow Diagram

A **Data flow diagram (DFD)** is a graphical representation of the “flow” of data through an information system. DFDs can also be used for the visualization of data processing (structured design).

6.2.1 Level 0:

Level 0 specifies the basic data flow of all the six module automated in the Automation software. Data flow diagram of the same is shown below.

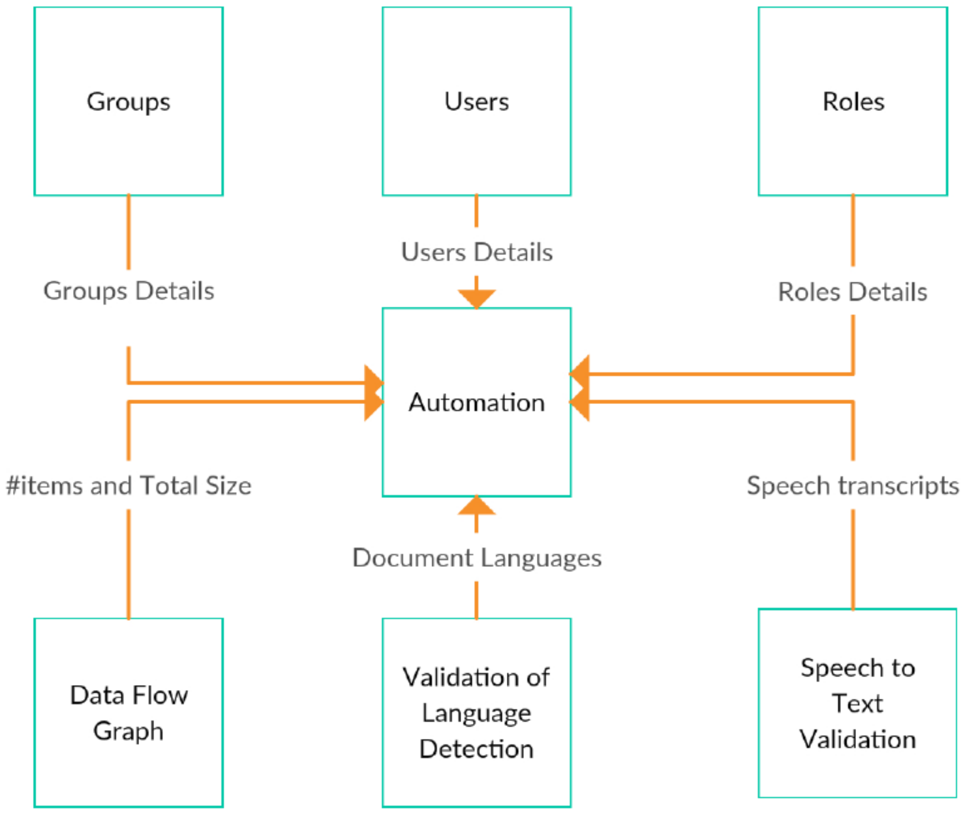


Figure 6.14: Level 0 Data flow diagram

6.2.2 Level 1:

Level 0 shows the basic movement of data in the automation software, Level 1 shown in depth view of the movement of data for each of the modules. These Data Flow Diagrams are shown below.

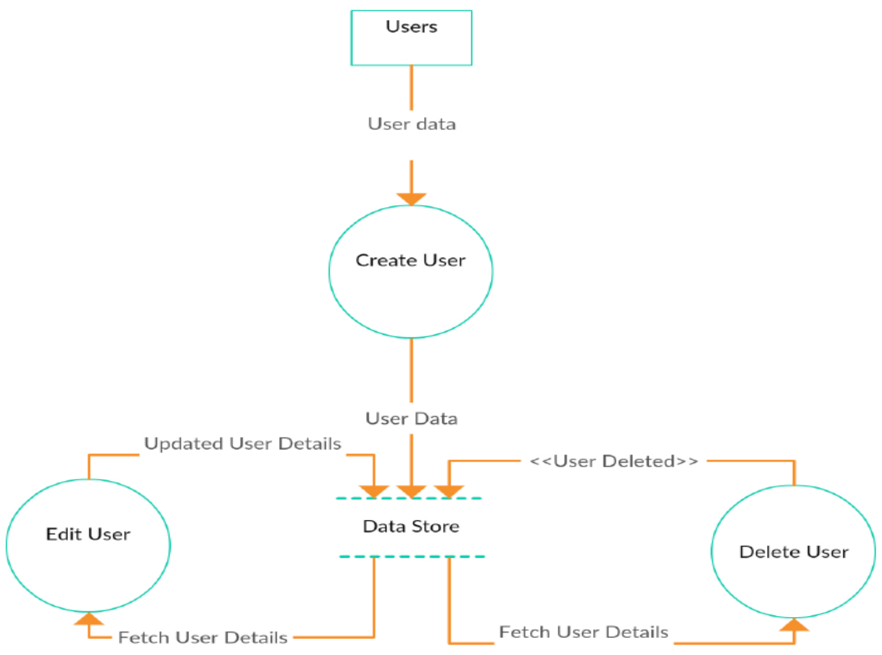


Figure 6.15: Users Module Data Flow Diagram

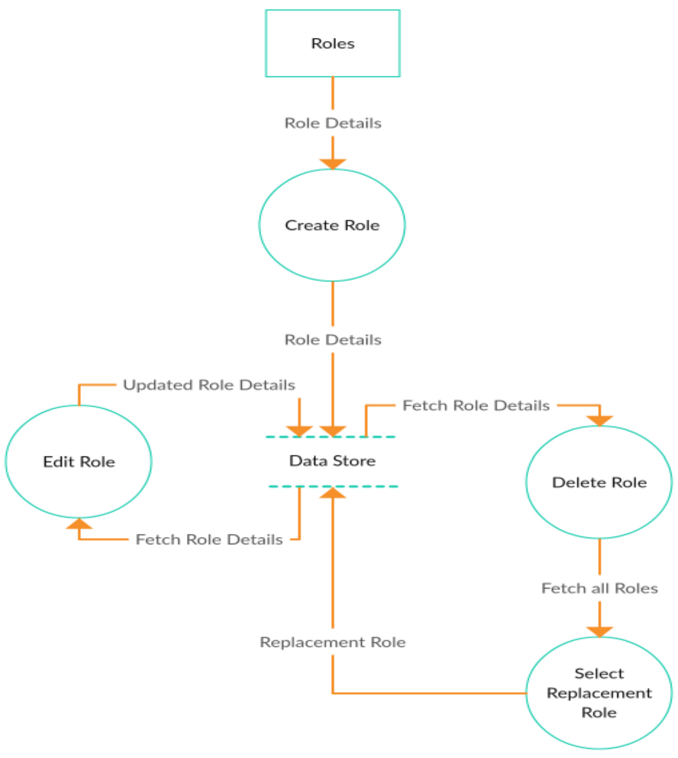


Figure 6.16: .Roles Module Data Flow Diagram

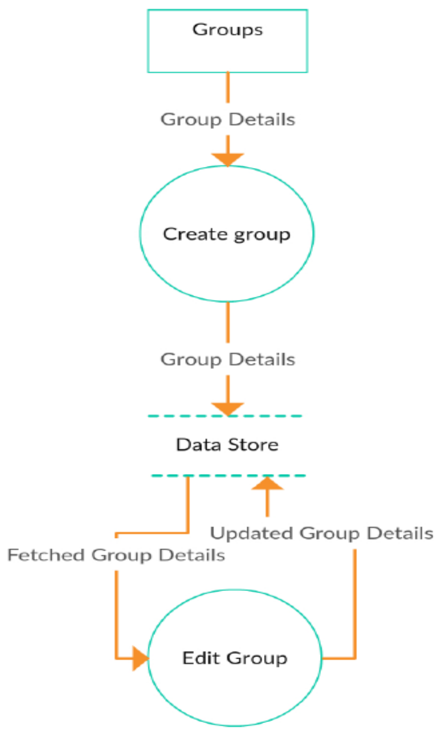


Figure 6.17: Groups Module Data Flow Diagram

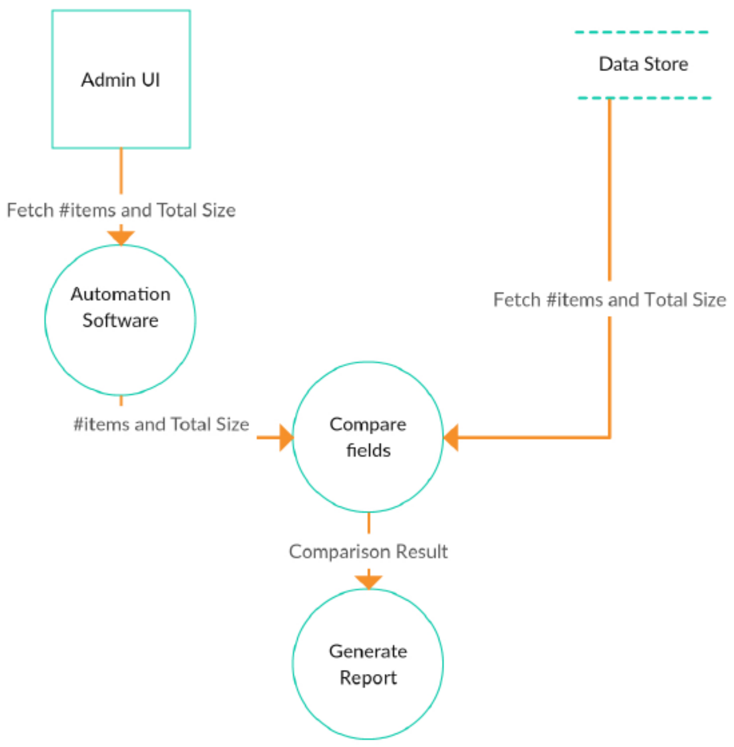


Figure 6.18: Data Flow Diagram for data flow graph validation

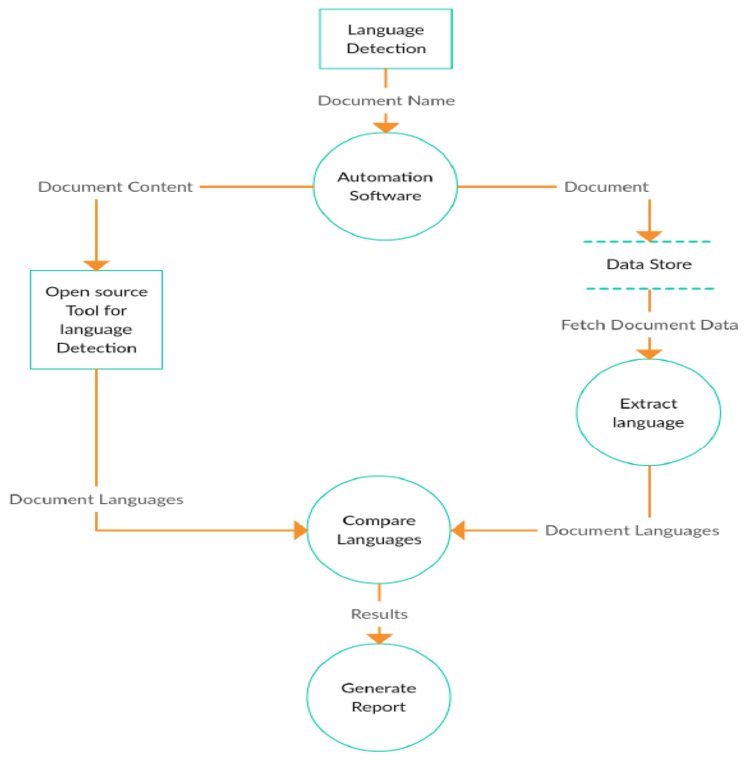


Figure 6.19: Data Flow Diagram of Validation of Language Detection

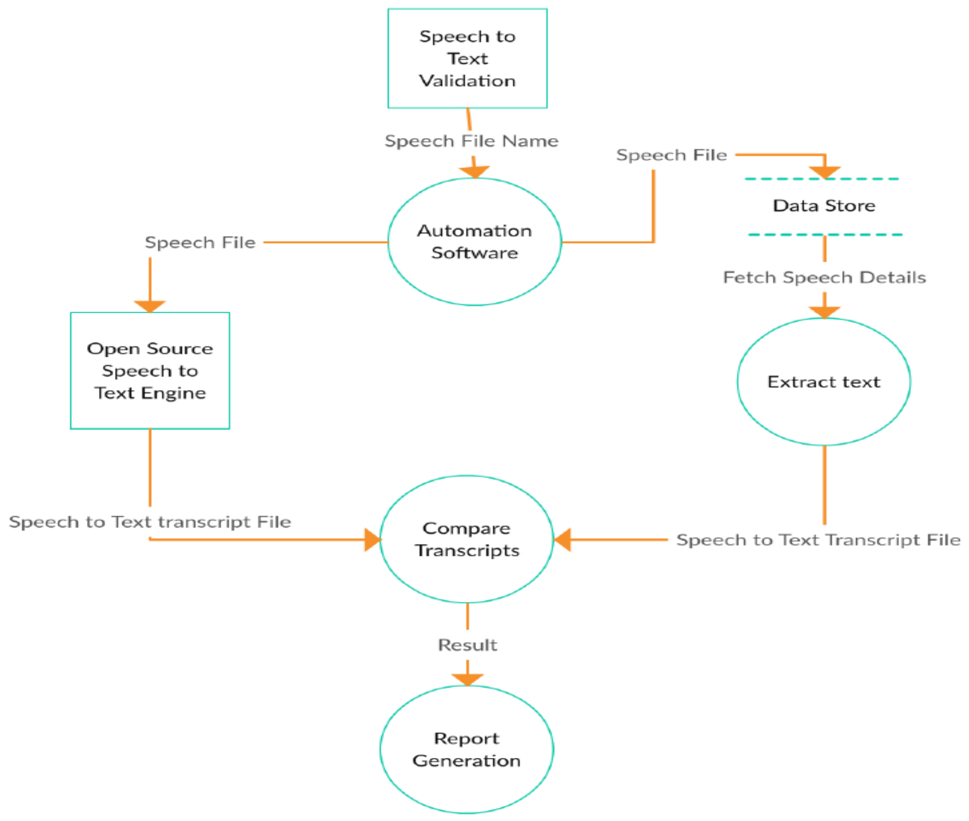


Figure 6.20: Data Flow Diagram of Speech to Text Validation

6.3 Updated RTM

Req. No.	Req. Description	System Design	Detailed Design	Implementation	Testing
F1.1	Create User	Section 5.1 Figure 5.2.1	Section 6.1.1.1 Figure 6.15		
F1.2	Edit User	Section 5.1 Figure 5.2.1	Section 6.1.1.2 Figure 6.15		
F1.3	Delete User	Section 5.1 Figure 5.2.1	Section 6.1.1.3 Figure 6.15		
F2.1	Create Role	Section 5.1 Figure 5.2.2	Section 6.1.2.1 Figure 6.16		
F2.2	Edit Role	Section 5.1 Figure 5.2.2	Section 6.1.2.2 Figure 6.16		
F2.3	Delete Role	Section 5.1 Figure 5.2.2	Section 6.1.2.3 Figure 6.16		
F3.1	Create Group	Section 5.1 Figure 5.2.3	Section 6.1.3.1 Figure 6.17		
F3.2	Edit Group	Section 5.1 Figure 5.2.3	Section 6.1.3.2 Figure 6.17		
F4	Validation of Data Flow Graph	Section 5.1 Figure 5.2.4	Section 6.1.4 Figure 6.18		

F5	Validation of language detection of documents	Section 5.1 Figure 5.2.5	Section 6.1.5 Figure 6.19		
F6	Speech to Text Validation	Section 5.1.4 – 5.1.6 Figure 5.2.6	Section 6.1.6 Figure 6.20		

Table 6.1 Updated RTM

7 Implementation

7.1 Pseudo Code / Algorithms

7.1.1 Automation of User Module

Below is the Pseudo code for automation of user’s module

Step	**Description**
1.	Open Browser
2.	Login with users credentials. Fetch these credentials from the configuration file.
3.	Navigate to the users tab in the admin user interface.
4.	Verify whether users tab is opened or not
6.	Create/Edit/Delete the user. Pass the required test data from the Jelly file.
7.	Verify whether Creation/Edition/Deletion of user is success or not from UI
8.	Generate report based on the previous step result. If the previous step is success. Print test Data in the report

7.1.2 Automation of Groups Module

Pseudo code of the automation procedure for Groups module is given below:

Step	**Description**
1.	Open Browser
2.	Login with users credentials. Fetch these credentials from the configuration file.
3.	Navigate to the Groups tab in the admin user interface.
4.	Verify whether Groups tab is opened or not
6.	Create or Edit the group. Pass the required test data from the Jelly file.
7.	Verify whether Creation and Edition of group is success or not from user interface
8.	Generate report based on the previous step result. If the previous step is success. Print test Data in the report

7.1.3 Automation of Roles Module

Pseudo code of the automation procedure for Roles module is given below:

Step	Description
1.	Open Browser
2.	Login with users credentials. Fetch the credentials from configuration file.
3.	Navigate to the Roles tab in the admin user interface.
4.	Verify whether Roles tab is opened or not
5.	Navigate to any Roles sub-category page
6.	Verify whether sub-category tab is opened or not
7.	Create/Edit/Delete the role. Pass the required test data from the Jelly file.
8.	Verify whether Creation/Edition/Deletion of role is success or not from user interface.
9.	Generate report based on the previous step result. If the previous step is success. Print test Data in the report

7.1.4 Validation of Data Flow Graph

Below is the Pseudo code to automate the validation process of data flow graph

Step	Description
1.	Open Browser
2.	Login with users credentials. Fetch these credentials from the configuration file.
3.	Navigate to the Information Archiving tab in the admin user interface.
4.	Verify whether Information Archiving tab is opened or not
5.	For each section in the data flow graph, perform step 6 and 7.
6.	Fetch Total number of items and total size of each section from UI
7.	Make query to the Vertica database and get the total number of items and total size. These two fields are basically back end data which is used to compare against UI field values.
8.	Compare these values each other. While comparing, convert the size Unit accordingly.
9.	Generate a report based on the comparison result and also pass these field values to the report for better understanding.

7.1.5 Automation of Language Detection Validation

Pseudo code of the process of validation of the language detection is given below:

Step	Description
1.	Pass the file name of which, language needs to be validated through jelly file
2.	Take this file name and run the python script which internally uses polyglot tool. While running the python script, pass the file name as the command line
3.	Read the file in python and pass the content of the file to Polyglot tool.
4.	Get the result and store
5.	Make API call to the elastic search to get all the details of the file. Store the response in JSON object. If the file is not uploaded before, upload the file manually.
6.	Process the JSON Object and extract LANGUAGE_NAME field from the JSON Object
7.	Compare both the detected language fields. If document contains multiple languages, each of language fields need to be validated accordingly.
8.	Generate report based on the comparison result.

7.1.6 Automation of Speech to Text Validation

Pseudo code of the process of validation of the language detection is given below:

Step	Description
1.	Based on the audio file name, fetch the details of the audio from elastic search. Since it returns a JSON Object. Store this JSON Object.
2.	Process the content primary field of JSON Object and extract the speech to text transcript and store the transcript in a file.
3.	Make an API call to the IBM Watson Speech to Text Engine. In the API call, send the audio file in the body of the API call. Get the response from the engine and store the corresponding transcript in a file.
4.	Make an API call to the Microsoft's Bing Speech recognition Engine. In the API call, send the audio file in the body of the API call. Get the response from the engine and store the corresponding transcript in a file.
5.	Store the audio's original transcript in a file.

6. Now, compute the Levenshtein's and Jaro-Winkler's edit distance values for each of the IBM Watson transcript and Microsoft's Bing Speech recognition transcript against original transcript.
7. Store Levenshtein's and Jaro-Winkler's edit distances in a file
8. Generate a graph based on these values which help for the analysis.

The two algorithms used were namely Levenshtein algorithm and Jaro-Winkler algorithm. Levenshtein algorithm is the most popular algorithm for such purposes and has found widespread use.

7.1.6.1 Levenshtein Algorithm:

It is a string metric for measuring the difference between two sequences. Order of complexity is O(m*n).Taking two strings as its inputs, it finds the number of insertions, deletions and substitutions necessary to change one string to the other.

Mathematically, the Levenshtein distance between two strings s1 and s2, of lengths |s1| and |s2| respectively is $\text{lev}_{s1,s2}(|s1|,|s2|)$, where

$$\text{lev}(\text{i},\text{j}) = \begin{cases} \max(i,j), & if \min(i,j) = 0 \\ min \begin{cases} \text{lev}\ (\text{i}-1,\text{j})+1 \\ lev\ (i,j-1)+1 \\ lev\ (i-1,j-1)+1 \end{cases} & ,otherwise \end{cases}$$

Where 1 is the indicator function. It is 0 when $a_i = b_j$ and 1 otherwise. $\text{lev}_{a,b}(|a|,|b|)$ is the distance between the first i characters of a, and the first j characters of b.

The Levenshtein distance has few simple upper and lower bounds. These include:

- The lower bound is the difference of the sizes of the two strings.
- The upper bound is the length of the longer string.
- It is zero only if the two strings are the same.

A dynamic programming approach can be taken to develop the code. That was the approach used in the project. Alternatively, a recursive approach which uses less memory can be implemented. The algorithm proceeds as shown below.

Step	Description
1	Set n = |s1|. Set m = |s2|. If n = 0, return m and exit. If m = 0, return n and exit. Construct a matrix containing m rows and n columns.
2	Initialize the first row to 0..n. Initialize the first column to 0..m.
3	Examine each character of s (i from 1 to n).
4	Examine each character of t (j from 1 to m).
5	If s1 [i] equals s2 [j], the cost is 0. If s1 [i] doesn't equal s2 [j], the cost is 1.
6	Set cell d [i, j] of the matrix equal to the minimum of: a. The cell immediately above plus 1: d[i-1,j] + 1. b. The cell immediately to the left plus 1: d[i,j-1] + 1. c. The cell diagonally above and to the left plus the cost: d[i-1,j-1] + cost.
7	After the iteration steps (3, 4, 5, 6) are complete, the distance is found in cell d [n, m].

7.1.6.2 Jaro-Winkler Algorithm:

The Jaro-Winkler distance is a string metric to measure the edit distance between two strings. Order of complexity is O (m*n). It is the minimum number of single character transpositions required to make the two input strings equal. Lower the returned value, more similar the two strings are.

For the sake of clarity to an uninitiated user, (1 – metric) value is returned, so that similar strings return a value closer to one, which makes intuitive sense.

First, the Jaro distance is computed, as explained below. Winkler actually added to the Jaro distance by giving weight to characters that appear at the beginning of the string, the logic being if initial characters are incorrect, it is tough to make sense of the string. As such, initially the Jaro distance is computed as shown below.

$$d_j = \begin{cases} 0, if\ m = 0 \\ \frac{1}{3}\left(\frac{m}{|s1|} + \frac{m}{|s2|} + \frac{m-t}{m}\right), otherwise \end{cases}$$

where:

s1, s2 are the two strings

|s1| and |s2| are lengths of the two strings respectively

m is the number of matching characters

t is half the number of transpositions.

Two characters are considered to be matching if they are equal and not further than $\left\lfloor\frac{\max(|s1|,|s2|)}{2}\right\rfloor - 1$. First we iterate over each character in string s1. It is then compared with each character in s2, searching for matching characters. If two characters are matching but appear in different sequence order, it is counted as a transposition. The total number of transpositions is divided by two to get the value of t. Now, to calculate Jaro-Winkler distance, the formula is

$d_w = d_j + (l*p*(1 – d_j))$

where:

d_w is Daro-Winkler distance

d_j is Jaro distance

l is length of common prefix at beginning of string, up to 4 characters

p is a scaling factor to adjust the score upwards for common prefixes.

Typical value is 0.1, up to 0.25.

7.2 Codebase Structure

The Codebase Structure of the project is described below.

```
\---qa-archiving-admin
   +--- AdminJellyTests
   |  +--- Admin
   |  |  +--- groupsJelly
   |  |  +--- informationArchivingJelly
   |  |  +--- languageValidationJelly
   |  |  +--- rolesJelly
   |  |  +--- usersJelly
   +---ATU Reports
   |  \---Results
   |      +---Run_1
   |      +---Run_10
   |          +---barChart.js
   |          +---report.html
   |          +---settings.properties
   +---conf
   |  +---app.properties
   |  +---NextGen.properties
   |  +---Reporting.properties
   |  +---modelfile.properties
   |  +---serverConfig.xml
   +---dist
   |  \---classes
   |      \---ISTFCore.jar
   +---libs
   |  +---ATUReporterlibs
   |  \---rest-assured-2.6.0-deps
   |  \---LanguageDetect
   |  \---speechToText
   +---dataset
   |  +---Audio corpus
   +---Screenshots
   +---src\java\com\test\framework
   |  +---admin
   |  |  +---users
   |  |  |   +--- CreateUserTag.java
   |  |  |   +--- EditUserTag.java
   |  |  +---roles
   |  |  |   +--- CreateRoleTag.java
   |  |  |   +--- EditRoleTag.java
   |  |  |   +--- DeleteRoleTag.java
   |  |  +---groups
   |  |  |   +--- CreateGroupTag.java
   |  |  |   +--- EditGroupTag.java
   |  |  +---informationArchiving
   |  |  |   +--- verifyDataFlowTag.java
   |  |  +---languageDetect
   |  |  |   +--- languageDetect.java
   |  |  +---speechToText
   |  |  |   +--- compare_transcripts.java
   |  |  |   +--- elasticSearchMain.java
   |  |  |   +--- jaro_winkler.java
   |  |  |   +--- Levenshtein_dist.java
   |  |  |   +--- Watson.java
   |  |  |   +--- LinceChart.java
```

The root folder is qa-archiving-admin. AdminJellyTests folder contains all the jelly files scripted to test the application. These are pushed into the repository at the end of each work cycle. The sub-folders for users, roles, groups and so on are seen. ATU Reports folder contains the test reports generated after each run of tests. They can be opened using an .html link within the folder. Visualization of the test cases passed, output values generated for specific inputs, stack trace during an exception, all these can be seen in a clear format on the webpage.

The configuration folder contains property files from which all classes read shared data such as username, password, server IP address and so on. Keeping one central store for this common data avoids having to make changes in each of the dependent files every time some property changes.

Dist folder contains the output of each compilation. A .jar file is created as the output of compilation. Contents of this folder are thus over-written during each compilation run.

Libs folder contains the various external libraries that are referenced throughout the code. These can be added in through Eclipse IDE as well.

Dataset folder contains the various audio files in different languages, emails, PDFs, messages and so on. These are selected to be a diverse collection of files in various formats and languages. These form the basis for testing as a large variety in the corpus would lead to a broad range of test cases.

Screenshots are captured and stored whenever a test case fails. This is done within the ATU Reporter. This is of course valuable more during UI automation as the state of the UI can be captured at the moment any test case fails.

Finally, the source folder contains the various java files created over this project. They are grouped by functionality for ease of maintenance.

Ant was used to manage the project initially. The functionality used does not differ significantly between Maven and Ant. By running the command ‘ant test’, the project is built. This includes compilation, jar creation, execution, running tests, and logging reports. The build.xml file for the project specifies the jellies to be run in particular. Test scripts should be written such that only one test happens within one script. This is necessary as it otherwise becomes difficult to pinpoint the source of error in case a test fails.

The jelly files have to be placed in a folder two levels down in hierarchy from the root folder, according to the structure of the project. The tags used to call the classes have to first be written into the framework tag library java file. If there is a match, then the arguments, if any, are passed to the class using setter methods. Names of the variable specified need to match.

For each class to be called through jelly files, a doTag method needs to be defined. This is the code that is run each time the particular class is called from a jelly file. Within this method, report generation also needs to be handled. The type of assertion (soft/hard) should be mentioned. In case of hard assert, execution stops at the first error encountered.

Since the application is deployed on a remote machine, the server address and login credentials also need to be mentioned within the properties file. Initially, a login method is called. This enters the credentials which have been stored for a user created manually one time. This user is used to access the application each time.

For backend validation, a different set of tools and processes are needed. Queries to the database, which is elastic search, are made in REST API format. JSON objects are returned and parsed for required data. Different approaches are taken for the both the different tests, namely language detection and speech to text.

Description of each of the source code written for achieving the requirements is explained below:

7.2.1 <u>Automation of Users module</u>

7.2.1.1 <u>Create User</u>

Input data is passed from the jelly files to CreateUserTag. CreateUserTag.java creates the new with the data passed and verifies whether user is created or not. Based on the result, appropriate report is generated.

7.2.1.2 <u>Edit User</u>

Similar to create user, test data to edit the user is passed via jelly file to EditUserTag. EditUserTag.java modifies the details of the corresponding user. Also, code verifies whether changes had been made successfully or not and report is generated.

7.2.1.3 <u>Delete User</u>

In order to delete the user, we can directly delete the user from jelly files itself. Here, common functionalities provided by the framework is used in jelly files. Also, user name which needs to be deleted is passed from the jelly file.

7.2.2 Automation of Roles module

7.2.2.1 Create Role

Details for creating the role is passed from the jelly file to CreateRoleTag. CreateRoleTag.java uses these data and creates the role. After creating, code verify whether role has created successfully or not by verifying all the details passed. Based on the result, appropriate report is generated.

7.2.2.2 Edit Role

As in creating a role, required details are passed from jelly file to EditRoleTag. Using these details in EditRoleTag.java, role is edited. After edition, modified role validated against test data passed in UI and report is generated.

7.2.2.3 Delete Role

Since replacement role needs to be selected while deleting the role, we cannot delete the role directly from jelly files. Hence, role which need to be deleted and replacement role is passed to the DeleteRoleTag.java.

7.2.3 Automation of Groups module

7.2.3.1 Create Group

Details required to create the group is passed from jelly files to CreateGroupTag.java program. The program create the group and verify whether group is created properly or not. Based on the result appropriate report is generated.

7.2.3.2 Edit Group

To edit the group, required details are passed from jelly files to EditGroupTag.java program. The program modifies the group with the passed test data. Also, program verifies whether group is successfully edited or not and report is generated accordingly.

7.2.4 Automation of Data Flow Graph validation

To validate the data flow graph, call the verifyDataFlow.java from jelly file. In verifyDataFlow.java, fetch the UI values, make the queries to Vertica database and compare the values. Based on the result, report is generated.

7.2.5 Automation of Language Detection validation

To validate the specific document, pass the document name from jelly file to languageDetect.java. Using the file name in languageDetect.java, fetch the corresponding language fields from elastic search and Polyglot. Compare these fields and generate the report.

7.2.6 Speech to Text validation

Speech to Text transcript of all the audio files is extracted from the elastic search and store it in transcripts_es.txt using elasticSearchMain.java program. Similarly, store the original transcripts in transcripts_orig.txt. Make is API calls to IBM Watson and Microsoft Bing speech engines and extract the transcripts in transcripts_watson.txt and transcripts_msft.txt files respectively. Once all the transcripts are available, calculate the difference between every transcripts from compareTranscripts.java program. In this program call jaro_winkler and Levenshtein_dist methods to find the edit distance and normalize it. Once calculating all normalize value, calculate the aggregate value for different accents and store these values in separate files. Plot the graph from LinceChart.java program.

7.3 Coding Guidelines Used

Some best coding practices exist for every programming language. These guidelines allow user to code more efficiently. Not only the quality of the code increases, but by following these guidelines, the programmer also improves his coding skills,

We in the due course of the implementation of our project wanted to implement the guidelines of Java. We strictly adhered to them to a maximum extent possible. Some deviations at some instances if any were because of unavoidable reasons.

i. Add Author, Assignment and class to the beginning of all source file.
ii. Individually and meaningfully comment member variables and class constants.
iii. Properly comment your code. A developer should able to get a general idea of what’s going on by just reading comment.
iv. Add Javadoc to all your code, including classes, methods, etc.
v. Do not use Spaces. Use tab as a unit of Indentation instead of spaces.
vi. Always use braces for code blocks, even single line of code.
vii. Import all necessary classes. Do not use wildcard imports (java.util.*) unless there are four or more classes from that package.

viii. Add any and all annotation hints to code. (e.g., @Override).
ix. Catch the most specific exception type.
x. Do not explicitly extend object.
xi. Insure Variable and method names meaningful.
xii. One declaration per line is recommended since it encourages commenting.
xiii. Blank lines improve readability by setting off sections of code that are logically related.
xiv. Don't make any instance or class variable public without good reason.
xv. Don't keep any unused object or variables
xvi. Avoid using an Object to access a Class (static) variable or method. Use class name instead.

7.4 Sample Code

Below is the sample code for automation of the user creation. Since for some of the other modules, logic follows as same, only creation of user's explained here

When we want to create the user, we need many details pertaining to the user. Here, email, first name and last name are the mandatory fields. Other fields are, phone number, middle name, address (Street name, locality, pin code, country and state), groups and roles. Roles specifies that what privileges should assign to the new user. Also, user can be assigned to the one of the group. Group contains the many users, so that we can assign the role collectively. These details are passed from the jelly files. Sample jelly file is given below

```
<?xml version="1.0"?>
<j:jelly
xmlns:j="jelly:core"
xmlns:x="jelly:xml"
xmlns:spc="jelly:com.hp.test.spc.tags.SPCTagLibrary"
xmlns:sel="jelly:com.hp.test.framework.jelly.FrameworkTagLibrary"
xmlns:admin="jelly:com.hp.test.framework.admin.AdminFrameworkTagLibrary">

<admin:createuser email="user@gmail.com" firstname="user1firstname"
middlename="user1middlename" lastname="user1lastname"  phone="1234567890"
groups="" address="3rd cross|bangalore|karnataka|767676|IN" roles="Administrator|Audit
Manager|Data Manager|Reporting Manager|User Manager|Discovery Services
Administrator|Reviewer"/>
<sel:waituntilelementvisible id="xpath=//span[text()='user@gmail.com']"/>
<admin:verifyelementsinlist id="css=span.truncateList span.text-underline"
expected="user@gmail.com"/>
</j:jelly>
```

Here, we are using the tag named createuser, this is tag which we declare in AdminTagLibrary java class. This class contains all mapping between tags and corresponding java classes. Below is the sample TagLibrary class.

```
package com.hp.test.framework.admin;
import org.apache.commons.jelly.TagLibrary;
import com.hp.test.framework.admin.users.NewUserTag;
public class AdminFrameworkTagLibrary extends TagLibrary {
        public AdminFrameworkTagLibrary() {
                        this.registerTag("createuser", NewUserTag.class);
        }
}
```

Using these mappings, all the data from the jelly files are passed to the corresponding class. Here, getters and setters need to be used in order to initialize these data to the local variables. This feature is supported by the framework. Java file extends the SeleniumTagSupport, SeleniumTagSupport internally extends the TagSupport class. We need to override the doTag method of TagSupport class. This method acts like a main method in the automation. Inside this method, we create an instance selenium driver and interact with the browser. Objmap is used to fetch the static locators of the UI. Sample code of createuser is given below.

```
public class NewUserTag extends SeleniumTagSupport {
        private String email, firstname, middlename, lastname, phone, groups, roles, address;
        String workingDir = System.getProperty("user.dir");
        ObjectMap objmap = new ObjectMap(workingDir +
        "/src/java/com/hp/test/framework/admin/objectProperty/adminObjects.properties");
        public void setemail(String email) {
                this.email = email;
        }
        public String getemail() {
                return this.email;
        }
        public void addRoles(WebDriver driver, String roles) {
                String diffRoles[] = roles.split("\\|");
                for (String role : diffRoles) {
                        WebElement el = driver.findElement(By.xpath("//label[text()=' " + role
                        + "']//ins"));
                        el.click();
                }
        }
        public void addAddress(WebDriver driver, String address) {
                String addressParts[] = address.split("\\|");
                try {
                driver.findElement(objmap.getLocator("address")).click();
                driver.findElement(objmap.getLocator("street")).sendKeys(addressParts[0]);
                driver.findElement(objmap.getLocator("city")).sendKeys(addressParts[1]);
                driver.findElement(objmap.getLocator("state")).sendKeys(addressParts[2]);
                driver.findElement(objmap.getLocator("zip")).sendKeys(addressParts[3]);
                driver.findElement(objmap.getLocator("country")).sendKeys(addressParts[4]);
                Assert.assertEquals(addressParts.length, 5);
                ATUReports.add("Address Fields",   "street :" + addressParts[0] + " city : " +
                addressParts[1] + " state : " + addressParts[2]+ " zip : " + addressParts[3] + "
                country : " + addressParts[4],"Success", "Success", false);
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
        public void doTag(XMLOutput output) throws MissingAttributeException,
JellyTagException {
        try {
        driver.findElement(objmap.getLocator("newUser")).click();
        WebElement emailElement;
        emailElement = driver.findElement(objmap.getLocator("email"));
        emailElement.sendKeys(email);
        driver.findElement(objmap.getLocator("firstName")).sendKeys(firstname);
        if (!middlename.equals(""))
        driver.findElement(objmap.getLocator("middleName")).sendKeys(middlename);
        driver.findElement(objmap.getLocator("lastName")).sendKeys(lastname);
        if (!phone.equals(""))
                driver.findElement(objmap.getLocator("phone")).sendKeys(phone);
        if (!address.equals("")) {
                addAddress(driver, address);
        }
        if (!roles.equals("")) {
                driver.findElement(objmap.getLocator("rolesTab")).click();
                addRoles(driver, roles);
        }
        driver.findElement(objmap.getLocator("createUser")).click();
        Thread.sleep(5000);
        } catch (Exception e) {
                e.printStackTrace();
        }
        ATUReports.setAuthorInfo("Core Automation Team", "DATE()", "1.0");
        ATUReports.add("New User","email : " + email + " firstName :" + firstname + "
        lastName : " + lastname + " phone : " + phone,"Success", "Success", false);
        }
}
```

Here, when the user is created successfully, we can make an entry of this in ATUReports.

7.5 Unit Test Cases

Unit test cases for all the requirements is mentioned below.
Requirement ID: F1

Input: Email address, First Name, Last Name, Middle Name, Phone, Group, address and roles. Address contains Street, City, State, Zip Code and Country fields.

Expected Output: User should be created/edited/Deleted successfully.

Req. No.	Test Case	Input Data	Expected Result	Actual Result
F1.1	Create User	Email: sample1@abc.com First Name: first name Last Name: last name	New user should be created with the input data	New User Created Successfully
F1.1	Create User	Email:sample2@abc.com First Name: first name Last Name: last name Middle Name: middle name Phone: 12345 Address: 4thMain|Bengaluru| Karnataka|560078|IN	New user should create with all the passed input data	User is created successfully
F1.1	Create User	Email: sample3@abc.com First Name: first name Last Name: last name Role: Administrator| Audit Manager	New user should create with all the passed input data	User is created successfully
F1.2	Edit User	Email:sample1_edited@abc.com First Name: edited first name Last Name: edited last name	User with username sample1@abc.com should be updated	sample1@abc.com successfully updated
F1.3	Delete User	Email: sample1_edited@abc.com	User should delete	User deleted
F1.3	Delete User	Email: sample2@abc.com	User should delete	User deleted
F1.3	Delete User	Email: sample3@abc.com	User should delete	User deleted

Table 7.1: Unit Test cases of F1

Requirement ID: F2

Input: Category, Role Name, Role Descriptions and Permissions

Expected Output: Roles should be created/edited/Deleted successfully with passed data.

Req. No.	Test Case	Input Data	Expected Result	Actual Result
F2.1	Create Role	Category: Information Archiving Role Name: sample role Role Description: sample description Permissions: Auditing	New role should be created with the input values in information archiving section	New role is Created Successfully
F2.2	Edit Role	Category: Information Archiving Role Name: sample edited role Role Description: sample edited description Permissions: Reporting	Role should be edited successfully in information archiving section	Role edited successfully
F2.3	Delete Role	Category: Information Archiving Role Name: sample edited role Replacement Role: Administrator	Role should be deleted	Role deleted successfully

Table 7.2: Unit Test Cases of F2

Requirement ID: F3

Input: Group Name, Group Descriptions and Roles.

Expected Output: Group should be created/edited successfully with passed data.

Req. No.	Test Case	Input Data	Expected Result	Actual Result
F3.1	Create Group	Group Name: sample Group Group Description: sample description Role: Reporting Manager	New Group should be created with the input data	New Group is Created Successfully

F2.2	Edit Group	Group Name: sample edited Group Group Description: sample edited description Permissions: Reporting	Group should be edited successfully	Group edited successfully

Table 7.3: Unit Test Cases of F3

Requirement ID: F4

Expected Output: Data Flow graph should be validated against the data base values and report should be generated.

Req. No.	Test Case	Expected Result	Actual Result
F4	Validate the data flow graph	Both values evaluated as same and report should be generated.	Both values evaluated as same and report for the same is generated.

Table 7.4: Unit Test Cases of F4

Requirement ID: F5

Input Data: Document name for which language fields needs to be validated.

Expected Output: Language fields of both elastic search and Polyglot should be same and report should be generated for passed document.

Req. No.	Test Case	Input Data	Expected Result	Actual Result
F5	Validate the language Detection	Document Name: ArabicKorean.txt	Language fields of both elastic search and Polyglot should be Arabic, Korean. Report should generate.	Language fields are same and report is generated successfully

Table 7.5: Unit Test Cases of F5

Requirement ID: F6

Req. No.	Test	Expected Result	Actual Result
F6	Extract the transcripts from elastic search	Elastic search transcripts should be stored in transcripts_es.txt file	Transcripts Stored in file
F6	Extract the transcripts from IBM Watson speech engine	Transcripts should be stored in transcripts_watson.txt file.	Transcripts Stored in file
F6	Extract the transcripts from Microsoft's Bing speech engine	Transcripts should be stored in transcripts_msft.txt file.	Transcripts Stored in file
F6	Calculate edit distances of each transcripts against original transcripts and store the normalized result	Normalized edit distances should be stored in a file.	Edit distance values are stored properly
F6	Plot the graph using average edit values for each language	Graph should be generated successfully	Graph Plotted successfully

Table 7.6: Unit Test Cases of F6

7.6 Metrics for Unit Test Cases

Req. No.	No. of total runs	No. of successful runs	Test Pass Rate
F1	25	23	92
F2	20	17	85
F3	12	6	50
F4	6	4	66
F5	10	10	100
F6	40	30	75

Table 7.7: Metrics for Unit Test Cases

7.7 Updated RTM

Req. No.	Req. Description	System Design (Architecture)	Detailed Design	Implementation	Testing
F1.1	Create User	Section 5.1 Figure 5.2.1	Section 6.1.1.1 Figure 6.15	Section 7.2.1.1	
F1.2	Edit User	Section 5.1 Figure 5.2.1	Section 6.1.1.2 Figure 6.15	Section 7.2.1.2	
F1.3	Delete User	Section 5.1 Figure 5.2.1	Section 6.1.1.3 Figure 6.15	Section 7.2.1.3	
F2.1	Create Role	Section 5.1 Figure 5.2.2	Section 6.1.2.1 Figure 6.16	Section 7.2.2.1	
F2.2	Edit Role	Section 5.1 Figure 5.2.2	Section 6.1.2.2 Figure 6.16	Section 7.2.2.2	
F2.3	Delete Role	Section 5.1 Figure 5.2.2	Section 6.1.2.3 Figure 6.16	Section 7.2.2.3	
F3.1	Create Group	Section 5.1 Figure 5.2.3	Section 6.1.3.1 Figure 6.17	Section 7.2.3.1	
F3.2	Edit Group	Section 5.1 Figure 5.2.3	Section 6.1.3.2 Figure 6.17	Section 7.2.3.2	
F4	Validation of Data Flow Graph	Section 5.1 Figure 5.2.4	Section 6.1.4 Figure 6.18	Section 7.2.4	

F5	Validation of Language Detection of Documents	Section 5.1 Figure 5.2.5	Section 6.1.5 Figure 6.19	Section 7.2.5	
F6	Speech to Text Validation	Section 5.1.4 – 5.1.6 Figure 5.2.6	Section 6.1.6 Figure 6.20	Section 7.2.6	

Table 7.8: Updated RTM

8. Testing

8.1 System/Function test Spec for the Project

Test Case No.	Test	Expected value to be verified
1	Create the user, edit the user and delete the user	User should be successfully able to create after that it should be edited and later the same user should be deleted
2	Create the role, edit the role and delete the role	Automation Software should create the role, later it should edit this role and delete the role after editing.
3	Create group and delete the created group	Automation Software should create the group and edit this group
4	Validate the data flow graph	Fetch the values from both UI and Vertica data base and compare the values and generate report
5	Validate the language detection	Get the language fields from elastic search and Polyglot and compare the values and generate the report
6	Validate the speech to text transcripts	Calculate the normalized Levenshtein's and Jaro-Winkler edit metrics and plot the graph of these values for all transcripts.

Table 8.1: System Test Specification

8.2 Test Environment Used

Since this project or work of Automation Of test scenarios is a process of testing by itself, an overlook if the whole process gets executed correctly must be done manually. The environment that can be used for this manual test process is the same as that used for the process i.e. automation. Since this manual testing cannot be carried out without the actual automation happening, the automation environment is most essential and is used for the given project.

8.3 Test Procedure

Manual Testing is the means by which the System is tested manually without the use of any Automation Tools. Since we are automating the test scenarios, we need to manually test the automation instead of using any automation tool for this purpose. Hence, when running the automaton, we need to verify the steps taken by the automation software is working properly. And also, we should verify the result of the automation in the ATUReports reports. This tool is integrated with iSTF Framework. Hence, in order to test the automation we verified the report generated by the tool. Also, we used JIRA as the defect repository and bug tracking tool. Hence, when we find any bug or defect, we instantly raise the issue in JIRA.

8.4 Test Result

Test Cases executed (Manual Testing)

Test Case Id	Req. No.	Test Case description	Result
TC-1	F1.1	Check whether automation software able to create the user.	**PASS**
TC-2	F1.2	Check whether automation software able to edit the user	**PASS**
TC-3	F1.3	Check whether automation software able to delete the user	**PASS**
TC-4	F2.1	Check whether automation software able to create role.	**PASS**
TC-5	F2.2	Check whether automation software able to edit the role	**PASS**
TC-6	F2.3	Check whether automation software able to delete the role	**PASS**
TC-7	F3.1	Check whether automation software able to create group and later edit the group	**PASS**
TC-8	F3.2	Check whether automation software able to edit the group	**PASS**

TC-9	F4	Validate the data flow graph by fetching values from database	**PASS**
TC-10	F5	Validate the language detection for Arabic.txt from both elastic search and Polyglot	**PASS**
TC-11	F6	Extract the transcripts from elastic search, Watson and Bing speech engines and plot the comparative graph	**PASS**

Table 8.2: Test Results

8.5 Test Metrics

No. of total runs	No. of successful runs	Test Pass Rate
113	90	79.62

8.3: Test Case Metrics of Testing

8.6 Updated RTM

Req. No.	Req. Description	System Design (Architecture)	Detailed Design	Implementation	Testing
F1.1	Create User	Section 5.1 Figure 5.2.1	Section 6.1.1.1 Figure 6.15	Section 7.2.1.1	TC-1
F1.2	Edit User	Section 5.1 Figure 5.2.1	Section 6.1.1.2 Figure 6.15	Section 7.2.1.2	TC-2
F1.3	Delete User	Section 5.1 Figure 5.2.1	Section 6.1.1.3 Figure 6.15	Section 7.2.1.3	TC-3
F2.1	Create Role	Section 5.1 Figure 5.2.2	Section 6.1.2.1 Figure 6.16	Section 7.2.2.1	TC-4
F2.2	Edit Role	Section 5.1 Figure 5.2.2	Section 6.1.2.2 Figure 6.16	Section 7.2.2.2	TC-5

F2.3	Delete Role	Section 5.1 Figure 5.2.2	Section 6.1.2.3 Figure 6.16	Section 7.2.2.3	TC-6
F3.1	Create Group	Section 5.1 Figure 5.2.3	Section 6.1.3.1 Figure 6.17	Section 7.2.3.1	TC-7
F3.2	Edit Group	Section 5.1 Figure 5.2.3	Section 6.1.3.2 Figure 6.17	Section 7.2.3.2	TC-8
F4	Validation of Data Flow Graph	Section 5.1 Figure 5.2.4	Section 6.1.4 Figure 6.18	Section 7.2.4	TC-9
F5	Validation of Language Detection of Documents	Section 5.1 Figure 5.2.5	Section 6.1.5 Figure 6.19	Section 7.2.5	TC-10
F6	Speech to Text Validation	Section 5.1.4 – 5.1.6 Figure 5.2.6	Section 6.1.6 Figure 6.20	Section 7.2.6	TC-11

Table 8.4: Updated RTM

9. Results & Discussions

Design and implementation of the various modules that make up this project have thus been discussed. Significant parts of the UI were automated. Backend validation was also performed successfully on the different types of inputs mentioned, namely OCR, audio and multilingual documents.

Flow of information and knowledge was in both directions. The learning experience over the course of the internship was significant, and valuable suggestions were also made towards enhancing the product. There was a significant gain of practical knowledge, which enriched learnings from the classroom over seven semesters in college.
Specifically, various different approaches were explored to solve each problem. This enabled us to get a perspective on how experienced folk navigate through such situations and pick the most optimal solution. For example, to perform speech to text transcription and obtain results from external tools, three different approaches were immediately available. Only when work started did shortcomings of some methods become evident. This was something that happened over the course of the project, and iteratively we managed to obtain the best approach to solve each problem.

Valuable experience was gained in handling an application at industry-scale. There were also a lot of learnings that do not directly tie into the project. We also learnt to handle project management tools, version control systems, application deployment and many other such tools.

The expectation outlined in the introduction were all met successfully.

10. Retrospective

This juncture of time is the most crucial one in any student’s life, a transition from the environment of the college to the industry is a significant milestone in one’s life. This internship project offered us -the interns and me in particular a learning for lifetime. The knowledge obtained wasn’t just limited to the sphere of technology but provided us with survival skills. However, this did not come with ease. There were difficulties in the due course of the project. We were struck at places wondering what to do.

But these hardships did not deter us. Instead it enthused and motivated to work harder for the achievement of the final goal. At places, we failed to keep up to our planned schedule. But we learnt from our mistakes. We understood the importance of deadline. We got to experience a real-time agile feel. We realized the significance of schedules, setting up milestones, low level planning and much more.

There is a famous quote “Give a man a fish and you feed him for a day; teach a man to fish and you feed him for a lifetime”. This project taught us to deal with things like these. The values and the competency it induced into us is of high significance. Thus, this project provided us with an opportunity to learn and an experience to remember.

11 References

[1] TestNG in integration Testing
https://zeroturnaround.com/rebellabs/the-correct-way-to-use-integration-tests-in-your-build-process/
[2] Diff btw Selenium web driver and selenium RC
http://docs.seleniumhq.org/docs/03_webdriver.jsp#introducing-webdriver
[3] Code coverage tools
http://www.irisa.fr/lande/lande/icse-proceedings/ast/p99.pdf
[4] Comparison of code coverage tools
https://confluence.atlassian.com/display/CLOVER/Comparison+of+code+coverage+tools
[5] Test Automation Frameworks
http://www.softwaretestinghelp.com/test-automation-frameworks-selenium-tutorial-20/
http://safsdev.sourceforge.net/FRAMESDataDrivenTestAutomationFrameworks.htm
http://www.guru99.com/quick-test-professional-qtp-tutorial-34.html
[6] Comparison of GUI testing tools -
https://en.wikipedia.org/wiki/Comparison_of_GUI_testing_tools
[7] Selenium
http://www.softwaretestinghelp.com/test-automation-frameworks-selenium-tutorial-20/
[8] Google Language Detection
https://code.google.com/archive/p/java-google-translate-text-to-speech/
https://code.google.com/archive/p/language-detection/
[9] Language Detection from Relativity
https://help.kcura.com/9.0/Content/Relativity/Analytics/Language_identification_results.htm
[10] Polyglot Language Detection
http://polyglot.readthedocs.io/en/latest/Detection.html
[11] IBM Watson Speech to Text
https://www.ibm.com/watson/developercloud/doc/speech-to-text/index.html
[12] Microsoft's Bing Speech Recognition
https://www.microsoft.com/cognitive-services/en-us/speech-api
[13] REST Assured API
https://github.com/rest-assured/rest-assured/wiki/Usage
[14] JFree Charts
http://www.jfree.org/jfreechart/api/gjdoc/org/jfree/chart/