**Research on Test Automation**

**Comp -671 Software Verification and Testing**

**Franklinncjdjsjvsdjfdj**

**Author page**

This report is written by Akhila Chandaipeta. I'm currently attending Franklin University, Columbus, Ohio pursuing a Master of Science in Computer Science. This research paper was created as a part of an assignment for a Software Verification and Testing course in Spring, 2020 under the guidance of respected Professor James Jatau.

·

**Abstract**

As part of this research development project, I have chosen Test Automation. The test is the use of software separate from the software used for testing the software based on expected outcomes. When manual testing takes a lot of time Test Automation helps to saves time and give more accurate results. This process involves using test automation tools and by writing scripts with the tool.

Med vault is the web-application used for viewing a patient's medical record history. I am going to design this application and test on two different tools such as Cucumber and Junit.

Junit is the unit testing framework used to run a repetitive test to get the desired output. Junit is the open-source tool used for java applications.

Cucumber supports behavior-driven development (BDD). It mainly focuses on the end-user experience. It is one of the efficient testing tools.

**Introduction**

**Approaches to Test Automation**

There are many approaches to test automation. The widely used ones are below.

**Graphical user interface testing**. A testing framework that generates consumer interface events together with keystrokes and mouse click on and observe the changes that result in the front end, to validate that the observable conduct of the program is accurate.

**API driven testing**. A testing framework that makes use of a programming interface to the utility to validate the behavior beneath test. Typically, API pushed trying out bypasses' software person interface altogether. It can also be testing public (normally) interfaces to instructions, modules or libraries are tested with a spread of entering arguments to validate that the results which are back are accurate.

**Automated Testing process**

**Test tool selection**

For choosing the right tool we must analyze business requirements and the timeline of the project.

**Scope of Automation**

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

• The most essential function for commercial enterprise solutions

• When designing a scenario, we use a large number of facts and enter.

• Technical feasibility

• Business components are reusable

• The complexity of software test cases.

• Ability to apply the equal look at instances for pass-browser testing

**Planning, Design, and Development**

During this segment, you create an Automation approach & plan, which contains the following info

• Automation tool selected

• Choosing the quantity of testing wished

• Choosing the right framework based on requirements

• In-Scope and Out-of-scope items of automation

• Automation testbed coaching

• Schedule and Timeline of scripting and execution

• Deliverables of Automation Testing

• Test Report Preparation

**Test Execution**

Test execution involves writing the required scripts for the test cases and using a test automation tool to get desired results.

**Maintenance**

During the development phase if any changes or new requirements are added we can add new test cases and generate corresponding scripts. Maintenance becomes easy and manageable.

To learn automation in detail I decided to build a small web application called Med vault. I used two tools namely Junit and Cucumber to test the functionality of the application.

**Problem Statement**

The population in any country is in billions. Every day at least one lakh people go through many severe medical issues. These medical issues need immediate action. The main drawback is the unavailability of the recommended doctor. If the doctor is unavailable due to personal or professional reasons the patient has no opportunity to meet him. So, I have come up with an idea of creating a small application where a doctor can log in and view the medical history of the patients and suggest medicines based on previous records via any external source such as mail and phone.

We have chosen Test Automation to explore how it works and gain experience in test Automation. Med vault is a web application designed to view the medical record history of patients.

Doctors can log in with valid credentials it will be redirected to a patient's medical record history.

Doctors can view patient ID, Name, Age, Address, Phone number and type of disease. This web application is mainly designed to help the patient in emergencies. A doctor can view the recorded history and prescribe medicines personally or through other sources like mail, phone or fax, etc. Any user can insert details in the application. This application makes easy for doctors to know about the patient history anytime from anywhere around the world.

**Implementation / Discussion**

**Application**

I have created an application for MedVault. This Application is to store patient details and doctor details and view them when required. I have developed this application with **Rest API's** using **GET, POST and PUT** methods. I have also created a database named Hospital in **MYSQL**. IN the DB, I have created Three tables User, Doctor, and Patient. These tables will hold the basic information which helps us to fetch the information and validate it. The **Java Web Application** is deployed on **Apache Tomcat 7** server in a local system.

**REST API**

A **RESTful API** also referred to as a RESTful web service. It is based on representational state transfer (REST) technology, it is used as a source of communication between web services.

**Functionalities / Operations of APIs**: HealthCheck, SignUp, Login, InsertPatientDetails, GetPatientDetails, InsertDoctorDetails, GetDoctorDetails, UpdatePatientDetails.

**HealthCheck**: This API is to test the connectivity of the application. TO Check if the application is deployed and giving us a response saying, "Hello World".

**SignUp:** This API is a POST method that takes input Username, Password and Email in JSON format. This API will be calling to Database to store the details. Once that action is completed and there is an entry in DB, then this API will give a response saying code: Success with HTTP status 200.

**Login**: This API is a POST method that takes username and password as inputs. This API will call the Database with fetching the details from the USER table for that username and password and will validate the details. If the entry in the database match with the details entered, then the API will return as Code: Success.

**InsertPatientDetails**: This is the POST API method which takes all the patient details, Disease, and Medications. This Service will make a call to the Database and store all the information in the Patient table. Once the entry is done in the Database, then the service will return code: Success.

**GetPatientDetails**: This is a GET API method that takes the patient's name as input. With that patient name as input, this API will do the query the Database and fetch all the Patient Details. This API will return all the details in a JSON format.

**InsertDoctorDetails**: This is the POST API method which takes all the Doctor details, Disease, and Medications. This Service will make the call to the Database and store all the information in the Doctor table. Once the entry is done in the Database, then the service will return code: Success.

**GetDoctorDetails**: This is a GET API method that takes doctorName as input. With that doctorate as input, this API will do the query the Database and fetch all the Doctor Details. This API will return all the details in a JSON format.

**UpdatePatientDetails**: This is the POST API method which takes all the patient details, Disease, and Medications. This Service will make the call to the Database and Update all the information in the Patient table for that user. Once the update is done in the Database, then the service will return code: Success.

**Database**

The database is designed by using MySQL application. Coming to the Database, I have created three tables which are user, Patient, and Doctor. For these three tables, I'm performing CRUD(Create, Update, Read, Delete) operations. Also, I'm using primary keys and foreign keys to maintain the relationship among tuples.

**Cucumber Automation Testing**

A cucumber is a tool that helps Behavior Driven Development (BDD). It offers a manner to put in writing assessments that all of us can recognize, no matter their technical understanding.

In BDD, customers (business analysts, product proprietors) first write scenarios or attractiveness exams that describes the behavior of the machine from the purchaser's attitude, for evaluation and signal-off by way of the product proprietors before developers write their codes.

**Advantages of Cucumber**

* It is helpful to involve enterprise stakeholders who cannot without problems examine code
* Cucumber Testing focuses on the cease-user revel in three.
* Style of writing checks permit for simpler reuse of code in the tests
* Quick and smooth installation and execution
* Efficient device for checking out

To test the Services using automation, I have used the cucumber framework.

I have created a new Maven project with cucumber framework version of 5.4.0

In this project, I have defined some User action scenarios which cover all the Services Exposed in the MedVault web application.

**Gherkin File structure and Requirements:**

At the start of the feature film, we need to define the high-level description of the web application which we are testing. The Keyword Feature is used here. And this also helps in grouping the related scenarios.

Steps: Given, When, And, then are the executable steps in a scenario in sequential order. When cucumber is executing the steps, it will look for the matching Step definition for execution.

Given steps are used to describe the initial context of the system - the scene of the scenario. It is typically something that happened in the past.

When steps are used to describe an event or an action. This can be a person interacting with the system, or it can be an event triggered by another system.

Then steps are used to describe an expected outcome, or result.

And Steps are used to continue the successive previous steps

**Scenario Outline**

I have used a Scenario Outline keyword which helps me to run the same Scenario multiple times, with different examples or inputs.

When I'm using the scenario, I have to give the data table with the valid attributes.

For example, one of my Scenario is

Scenario Outline: User login

Given a running application

When a user enters <username> and <password>

Then login should return <result>

Examples:

| username | password | result |

| "akhila" | "akhi123" | "success" |

| "testing1" | "testing" | "success" |

Here, I'm defining username, password, and result twice. So now when I execute this scenario, it will execute two scenarios, each from the data table. One or username=Akhila and a second one for username=testing1.

Step Definition file:

Step Definition is Java class with methods invoking or links it to the steps in the feature file(Gherkins steps). When Cucumber runs/executes the Gherkin steps in the scenario, it will search for its matching step definition for execution.

In my project, I have a scenario for login as I have defined above.

For that scenario, the corresponding method in step definition file looks like

For the first step, it will look for this method shown in the image and run the script.

Similarly, it completes all the steps in the scenario sequential order.

Here as I'm using multiple examples in a data table, I need those parameters reading from the gherkins file. To achieve that, I need to give a regular expression \"([^\"]\*)\

**Example**:

When a user enters <username> and <password>

for this, the java method will look like:

Invoking Web Application to perform the actions in the scenario:

To invoke the Rest APIs, I have used the help of Spring Web Framework.

By using Rest Template, Request Entity, Response Entity, HTTP Method and Headers, I have successfully created a method, to invoke the API. Also, I have predefined the host of the running application and defined the URI as part of the step definition methods respectively.

As my web application MedVault produces and consumes JSON files in their Services, for submitting the request body and capturing the Response body I have created the model classes for that respective services and user Object Mapper to convert the Data into Json and Vice versa.

Now with the help of Spring, I'm able to access the application from my Automation code and invoke the Rest Services.

Also, I have used the Logging method provided by Java Util to print the Request and Response while calling the Application.

Structure of Cucumber project: Under the Test/resource folder, we need our feature file to be set. In that file, I have defined all the user action scenarios with Given, When, And, Then. And the Step Definition file is under com.hospital.test package in Test Folder. I have also included the model classes under the same test folder with a package as com.hospital.test.model.

**Running Cucumber Test Suite**

To run the Cucumber test suite, you need to define a test class, where it has to be defined as RunWith(Cucumber.class). This is from the cucumber-JUnit framework which is the key to invoke the automation test. In this file, we must give some predefined paths and file locations. For that, we can use @CucumberOptions annotation and inside that, we need to define the format, location of feature file, glue code(Step Definition file) and Tag.

Cucumber gives us leverage to run only the specific scenarios by using @Tags.

For example, if you have two scenarios in the feature film, you can Assign an Annotation to that scenario and define that annotation in the test file. Then only that scenario will be executed.

**Junit**

JUnit is an open-source Unit Testing Framework for JAVA. It is beneficial for Java Developers to put in writing and run repeatable checks.

Developers who are following test-driven methods need to write and execute a unit look at first before any code.

Once you are accomplished with code, you should execute all checks, and it needs to bypass. Every time any code is added, you want to re-execute all look at instances and makes

Once you are accomplished with code, you should execute all checks, and it needs to bypass. Every time any code is added, you want to re-execute all look at instances and makes

It discovers bugs from the get-go in the code, which makes our code progressively dependable.

JUnit is helpful for designers, who work in a test-driven condition.

Unit testing powers a designer to peruse code more than composing.

I have used Junit for performing unit testing for my implementation code. I have also used the PowerMock and Mockito framework for mocking the data and then used assertions to check if the mocked data entered was implemented in the right fashion and returned the expected output.

PowerMock is a structure that broadens other counterfeit libraries, for example, EasyMock with more impressive capacities. PowerMock utilizes a custom classloader and bytecode control to constructors and strategies, private techniques, static initializers.

Mockito is a java-based mocking framework, used in testing frameworks such as JUnit and TestNG. It uses the Java Reflection API and let the services create objects. A mock object returns a dummy data and avoids external dependencies.

*Here is a simple example of Junit using PowerMock and Mockito:*

Let’s say I have a Java Impl class where I validation of user login credentials are done. Let's assume that the method is log in. To test that method using Mockito and PowerMock, I need to Mock the POJO’s (Plain Old Java Object). We can use @Mock Annotation to mock a class. Now in the Test method when you are trying to feed the data into the Model classes, you can define the behavior of that method. In order to do that, We use **Mockito.when**(condition/action).**thenReturn**(DummyData). This means, if the login class has a variable username, then you can define it as **Mockito.when**(login details.getUsername).**thenReturn**(“Akhila"); In this way, you can mock the POJOs and also you can mock the services if that method is calling any internal services. This Mockito helps us in Testing all the positive and negative cases which we are expecting to throw Exceptions as well.

GitHub

The GitHub link for MedVault is given below-https://github.com/akhilaki131/Med-Vault

**Conclusion**

Considering all the application details I think Med vault is an inevitable part of Medical Emergency management systems. It makes the interaction between doctor and patient very smooth and easy. Med vault helps to deliver quicker solutions at a faster pace. It stores all the data of doctors and users and helps to manage all the medical history details. The advantages of this application are its easy to store information of patient very easily. This application is user friendly and efficient. During the development phase if any changes or new requirements are added we can add new test cases and generate corresponding scripts. Maintenance becomes easy and manageable. The main agenda for developing this application was to gain exposure to Automation Testing. Med vault is a web application designed with Rest API's using GET, POST and PUT methods. I have also created a database named Hospital in MYSQL. A cucumber is a tool that helps Behavior Driven Development (BDD). It helped to test the user-driven test case scenarios. It helped to test multiple API in a sequence manner. Junit helped to test if the functionality is performing as expected every time the code is modified using repeatable automated test cases.

References

Java Documentation - Get Started. (2019, October 10). Retrieved from <https://docs.oracle.com/en/java>

The new major version of the programmer-friendly testing framework for Java. (n.d.).

Retrieved from <https://junit.org/junit5/>.

Cucumber-Tools & techniques that elevate teams to greatness- <https://cucumber.io/>

REST API Tutorial-What is REST – Learn to create timeless REST APIs-https://restfulapi.net/

Mockito framework site-https://site.mockito.org/

MySQL-https://www.mysql.com/