Auto Key Tester: Action-Word Testing with a General Purpose Keyword-Driven Test Automation Framework

Sonu Mittal, Neeraj Gupta
Hindu College of Engineering, Sonepat, India

***Abstract*— In this paper, we present a tool named 'AutoKeyTester' that performs Action-Word Testing based on Keyword Driven Framework. This tool is developed to provide an easier approach to Functional Testing and Business Process testing by passing appropriate Parameters for particular Action words in the framework. This framework allows reasonably intuitive tests to be developed and executed without modifying the test scripts.**

Action-Word Testing is a functional testing in which reusable test components that are assembled into test scripts. It provides the reusability of test scripts through parameterization and multiple actions. In AutoKeyTester, the test components are identified through their corresponding labels, which make tool more users friendly.

***Keywords*— Automation, Keyword, Parameterization, Playback, Testing, Action, Record, Script.**

I. INTRODUCTION

Automation testing means execution of test cases in an automated way without manual intervention.[2] It reduces the testing efforts, costs, setup complexities, risk of non coverage and achieves efficient test execution through reusability, global visibility and reliability. Automated testing also act as documentation and has a great training value.

Test Automation has evolved from simple record and play back to Keyword Driven Testing. Keyword driven testing is a software testing technique that separates much of the programming work of test automation from the actual test design. This allows tests to be developed earlier and makes the tests easier to maintain. Some key concepts in keyword driven testing include:

- Keywords, which are typically base level and describe generalized UI operations such as "click", "enter", "select"
- Business templates which are typically high level such as "login", "enter transaction"
- Action Words, or short "Actions", which can be both base level and high level and in their most general form allow earlier defined key words to be used to define higher level action words

This tool records all user actions and stored them in a tabular format as well as test script. The test scripts are played back to check whether test pass or fails. These test script can also be reused by making them parameterized. In traditional tools, the objects are identified using 'Id' attribute of the objects. In AutoKeyTester, the objects are identified by the labels related to objects. This increases the usability of test scripts and easy identification of objects.

II. KEYWORD DRIVEN TESTING

The test automation frameworks have evolved over the time. They have evolved into three generations. Figure 1 shows evolution of Test Automation. In the beginning, there was record and playback script creation. In this, there were only stand-alone test scripts which don't require a lot of scripting and technical expertise. However, these test scripts are difficult to maintain, no reusability and no extendibility. After this, comes the Functional Decomposition, which consists of reusable functional i.e. test modules. This is a modular approach which provides flexibility, maintainability and reduces redundancy. But, Functional decomposition has data within the test scripts which has limited reusability, limited ease of maintenance and is largely dependent on technical expertise. To overcome these shortcomings, the data- driven testing concept comes, in which the test data is taken out of the test scripts and maintained in external files. This reduces the size of the test pack greatly and makes the test data variation easy and similar test cases can be created quickly. Still, this technique depends upon the technical expertise of test team and maintenance and perpetuation are issues.

Today, keyword-Driven testing is getting more popular. It is a technique that separates much of the programming work from the actual test steps so that the test steps can be developed earlier and can be maintained with only minor updates. It is a robust application, in which independent reusable keyword libraries are built. Test scripts are easy to maintain, highly scalable and not dependent on application availability. However, it is also time consuming and requires great deal of efforts.

In keyword-driven testing, each keyword corresponds to an individual testing action like a mouse click, selection of a menu item, keystrokes, opening or closing a window or other actions. A keyword-driven test is a sequence of operations, in a keyword format, that simulate user actions on the tested application. Basically, to perform any testing actions, testers simply drag and drop the keyword that corresponds to the desired operation or they can just record their actions and the keyword-driven test is built for them. Keywords are organized into tables that represent a test to be executed. You can create keyword-driven tests visually by adding and deleting operations and edit them directly by changing an operation's parameters and position. The easiest way to create keyword-driven tests is to record them. This technique does not require testers to know the application's internal objects, just a test plan of what they want to test. After recording, you can modify the keyword-driven test and customize it to fit your needs.

III. Reusability

Some software development tools automate the testing by recording tests that are run, allowing "playback" of the recorded test scripts. However, an entire test script is rarely, if ever, applicable to more than one release of one application. Data-driven testing provides some modularity by keeping test input and output values separate from the test procedure, but the procedure itself is still in a monolithic script. Keyword-Driven testing breaks the test procedure into logical components that can then be used repeatedly in the assembly of new test scripts. These logical components form the basis for the reusability in the Keyword Testing. The reusability can carried in two ways i.e. by making test script parameterized or by dividing the test script into multiple business logics called actions.

Parameterization allows us to use various values for the parameter at run time. It reduces time, effort and cost by eliminating the several runs of recording modules. While testing a web application, it may be required to check how the web application performs the same operations with multiple sets of data. For example, how a Web application responds to twenty separate sets of data is to be checked. There are two ways to do this test for twenty different sets of data. Twenty separate tests are recorded, each with its own set of data. Alternatively, a test is recorded, and the values of the objects in this test are made variable for parameterization. This saves the nineteen runs of the recording process. This single test with the help of parameterization can be played twenty times using a different set of data each time. Each test run is called iteration. All iterations are numbered. Later is an efficient approach and involves the reusability of Test Script. For example, the authentication page is signed in with "sonia.123" as user ID and "passXXXX" as password. The "sonia.123" is a constant value, which means that "sonia.123" is the user ID each time the test is run. The user ID can be changed into a variable, so that a different user id can be used for each test run. In the parameterized Keyword user ID, two different user ids can be added for example "son_mittal" and "neeraj_gupta". The tool can be test the application with this different data without the need of recording with these data.

Actions divide the whole test script into various business logical sections. Actions can be reusable action or nested action. In Reusable action, the same test is run multiple times. The nested action is the action within an action, which helps in achieving modularity in test. When a new test is created, it is represented as a single action. By dividing the tests into multiple actions, more modular and efficient tests can be designed. This is another feature of reusability of keywords that makes Keyword Driven Testing more efficient and modular than the Data Driven Testing. To explain this we take the whole example of the below online shopping web application. This can be divided into several distinct processes or actions which are as follows:

The online shopping web application is logged in.

The products are added to cart.

Another action for logged out from the web application.

The above test can also be parameterized for ten different product adding to cart. This parameterized test now can be run ten times using ten different sets of data. With the help of Multiple Action, the test can also be organized so that only the second procedure runs ten times, simulating a single user logging in, adding ten products to cart, and logging out. This can be done by dividing the test into different actions. This saves the nine runs of Logged-in and Logged-out process.

IV. Keyword Driven Advantages

Test Script more lucid – Keyword tables are often more lucid than regular test scripts, because the Keyword tables have manual test procedures. The keywords are typically test object components that make reading a keyword data table similar to reading a collection of sentences, which is easier than reading code statements that don't mirror natural language.

Reusability – Reusability is even further increased with a Keyword Driven framework, because most of the functions are created in such a way to not only be reusable for multiple tests within a single application, but also for tests across multiple applications.

More standardization – The increase of reusable framework components is followed by increased standardization. The added advantage that the Keyword Framework has over Functional Decomposition, however, is that standards are by default imposed through the implementation of a Keyword framework. Provided that the framework components are created with appropriate standards, they will be invoked in the keyword data tables with every use of the framework's keywords. Standardization helps with script maintenance, because it decreases guess work involved in figuring out what a script does and how best to fix it.

Non technical nature– Working with keyword data tables for everyday application automation is a lot less technical than working with code statements. Therefore individuals that are not as technical can be brought onto the team to help create automated tests.

Test script development early – This framework increases the ability for automation to begin before the application is delivered. Using information gathered from Requirements or other documentation, keyword data tables can be created that mirror corresponding manual test procedures.

More traceability– Given the fact that keyword data tables so closely resemble a manual test procedure, it becomes simpler to trace actions in automated tests to actions in manual tests. In addition, there will be greater reuse of manual test procedures.

Easy Error handling– Patch work error handling solutions on a script by script basis are difficult to introduce and maintain. With reusable Keyword framework components, error handling can be introduced that reaches multiple scripts across multiple applications. This will ultimately improve the effectiveness of unattended execution of the test scripts.

V. KEYWORD DRIVEN CHALLENGES

Need technical expertise – While the technical nature of creating automated tests within the framework is decreased, the technical skills required to create and maintain the framework itself is greatly increased from that required for Functional Decomposition frameworks. There are numerous dependencies and relationships that must be understood and maintained, as well as advanced tool components and structures.

No Intuition – In order to effectively implement a Keyword framework, reliance on intuition must be reduced, while reliance on standards must be increased. While some standards are automatically imposed, with this type of framework, many standards are not, so there's an ongoing effort to ensure resources are aware of standards, understand them, and are able to effectively implement them. Increased documentation will probably be required to identify framework features, particularly documentation that chronicles the keywords that exist as part of the framework that may be used.

Requires more management – Management support is a challenge for any automation effort, but it is particularly difficult with Keyword Driven frameworks. There must be strong management support, however, for the time and resources necessary for creating and maintaining the structures, documentation, and personnel (both technical and non-technical).

Restrictive – For technically adept resources that are tasked with day-to-day automation of a software application, Keyword frameworks may be overly restrictive. They may be perceived as an entity that "ties their hands" into automating in a "standard" way at the expense of automating in the most efficient way for a particular application, or particular feature within an application. Keyword applications typically require increased "public relations" work to sale the approach to both resources and management.

6. AutoKeyTester

AutoKeyTester is an automated testing tool that performs Keyword Driven Testing for any desired web application like authentication page, online shopping web application, online reservation web application, etc. Figure 1 shows the main window of AutoKeyTester It will perform the functional testing on a complete, integrated web application to evaluate the web application's compliance with its specified requirements. AutoKeyTester performs recording, playback, test result reporting, parameterization, creation of multiple actions, and maintenance of Object Repository.

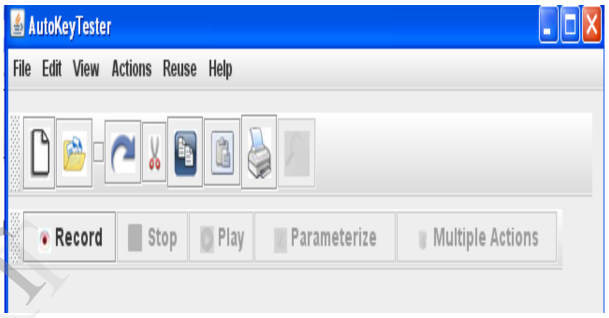
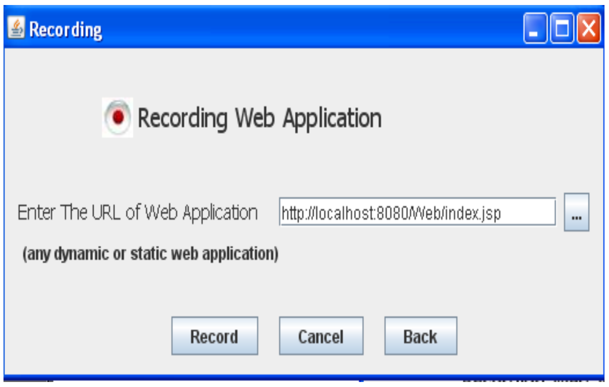


Figure 1: AutoKeyTester

First, a web application is taken which is needed to be tested. To start the process of recording, enter the URL of the desired web application (Figure 2). An online shopping web application is taken here, as an example. As user navigates the Web application, the AutoKeyTester records all user actions (Figure3). User logins the application and provides the Personal details, the Click Button is clicked and Details entered successfully web page opens. These operations form the basis of the test. The tool records all the operations performed in the Web browser until the recording is stopped.

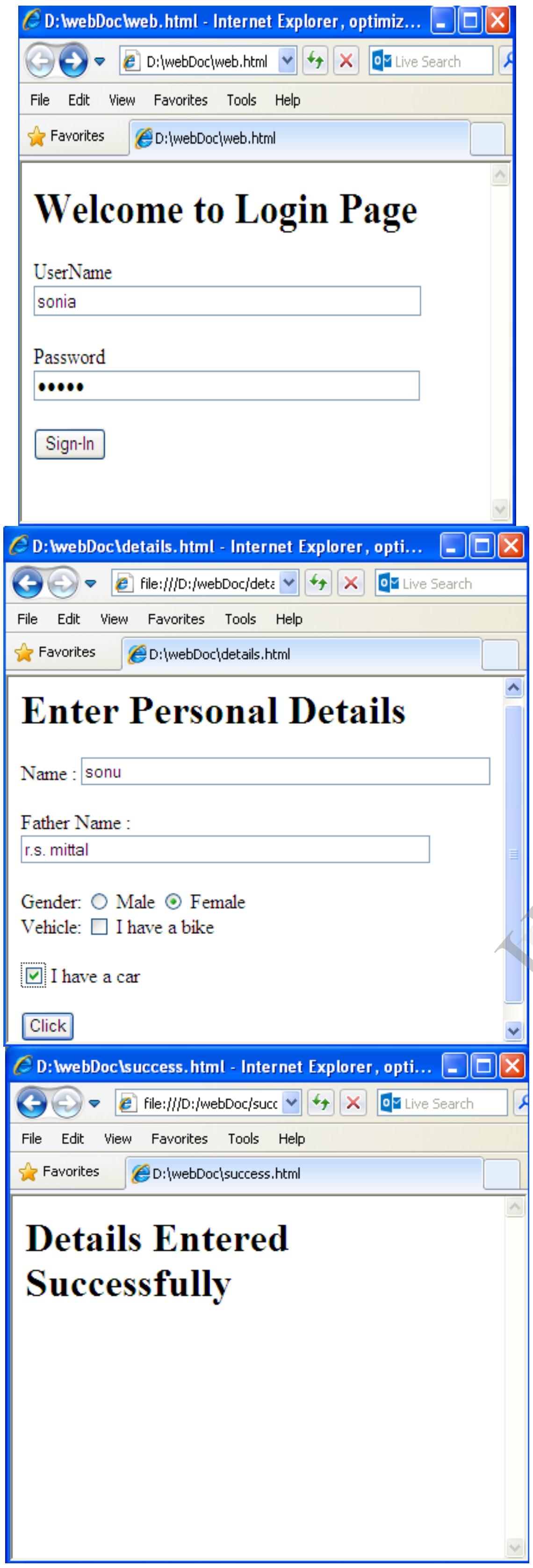

Recording

Recording Web Application

Enter The URL of Web Application: http://localhost:8080/Web/index.jsp

(any dynamic or static web application)

Figure 2: Recording Page


D:\webDoc\web.html - Internet Explorer, optimiz...
File Edit View Favorites Tools Help
Favorites
D:\webDoc\web.html
D:\webDoc\details.html - Internet Explorer, opti...
file:///D:/webDoc/deta
D:\webDoc\details.html
D:\webDoc\success.html - Internet Explorer, opti...
file:///D:/webDoc/succ
D:\webDoc\success.html
Live Search

Welcome to Login Page

UserName: sonia

Password: •••••

Sign-In

Enter Personal Details

Name : sonu

Father Name : r.s. mittal

Gender: ( ) Male [✓ Female]

Vehicle: ☐ I have a bike

☑ I have a car

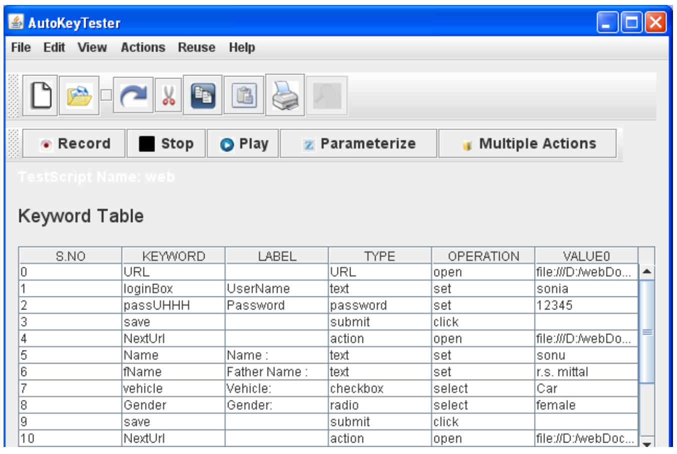
Click

Details Entered Successfully

Figure 3: User Actions

When the recording is stopped, a test script file is generated. This generated test script file containing all user actions, is saved. The user actions will comprise of items clicked, items selected, value typed etc, during the recording procedure. The tool will also generate steps in the table format (Figure 4), representing each operation performed in the form of Keyword, Label, Type, Value and Operation. For example, User name, Vehicle checkbox and Submit button are the keywords. The Labels are the label tags or any text before the keyword. It helps in making the test script more users friendly because user identifies the keyword in a web page through its label, not with its name or id. The Type shows the type of the Keyword like text, radio button, checkbox, submit, URL, etc. The Value represents values entered by the user in the items. The Operation includes the click, select, set, open, etc. The test Script will be useful in the play back of a test and reusability of the test script i.e. parameterization and multiple actions.

When the test is play backed, the tool runs the saved test script file. The recorded web application opens in the web browser and all steps are performed automatically, as it was originally recorded in the test. When the test run is completed, it displays the results of the run (whether a test is passed or failed) in the test result page. The Test Results window opens, which contains the result summary of the test execution. Object Repository is a centralized place for storing the properties of objects available in AUT (Application under Test). The keywords can be added in the object repository, either manually or at the time of recording. All software applications and websites are developed using many different components or small units are known as Objects. Each object is identified on the basis of the object type. Each object has properties (for example *name*, *title*, *caption*, *color and size*) and specific set of method, which help in identification of an object.

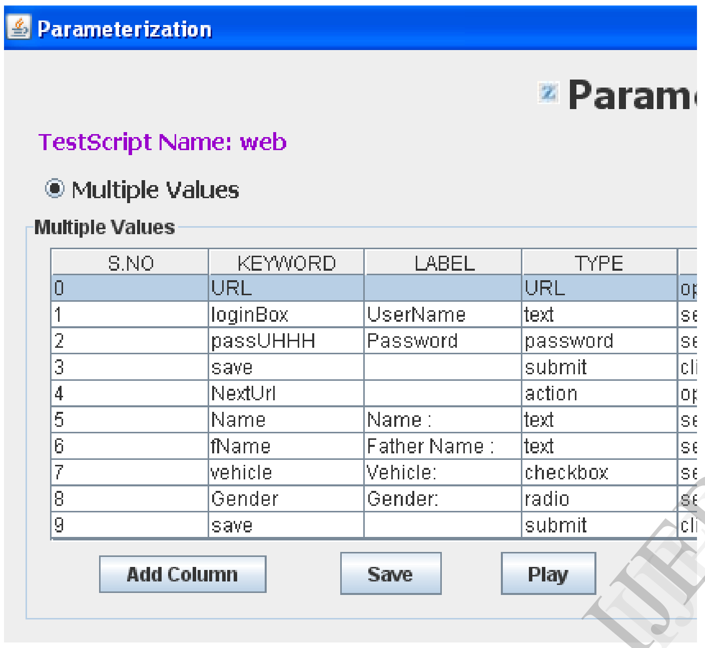


Keyword Table

S.NO	KEYWORD	LABEL	TYPE	OPERATION	VALUE0
0	URL		URL	open	file:///D:/webDo...
1	loginBox	UserName	text	set	sonia
2	passUHHH	Password	password	set	12345
3	save		submit	click	
4	NextUrl		action	open	file:///D:/webDo...
5	Name	Name :	text	set	sonu
6	fName	Father Name :	text	set	r.s. mittal
7	vehicle	Vehicle:	checkbox	select	Car
8	Gender	Gender:	radio	select	female
9	save		submit	click	
10	NextUrl		action	open	file://D:/webDoc...

Figure 4: Keyword Tabular View

The Test scripts can be added, deleted and modified. The test script modification helps in the parameterization of the test as well as during the division of a test into multiple actions. While testing a web application, there may be a need to check how the web application performs the same operations with multiple sets of data. A test is recorded, and the values of the objects in this test are made variable for parameterization. This saves the nineteen runs of the recording process. This single test with the help of parameterization can be played twenty times using a different set of data each time. In the above example, an excel containing different values for the keywords are uploaded in this tool. The tool will first check for that number of keywords in the excel sheet should not be greater than the number of keywords in original recorded test script. After that, tool will check for correct keywords and generate the different test scripts for each set of values. The tool can be test the desired application with this different data without the need of recording with these data. Figure 5 shows the parameterization and Figure 6-7 shows the automated play back of parameterization results without doing recording process.



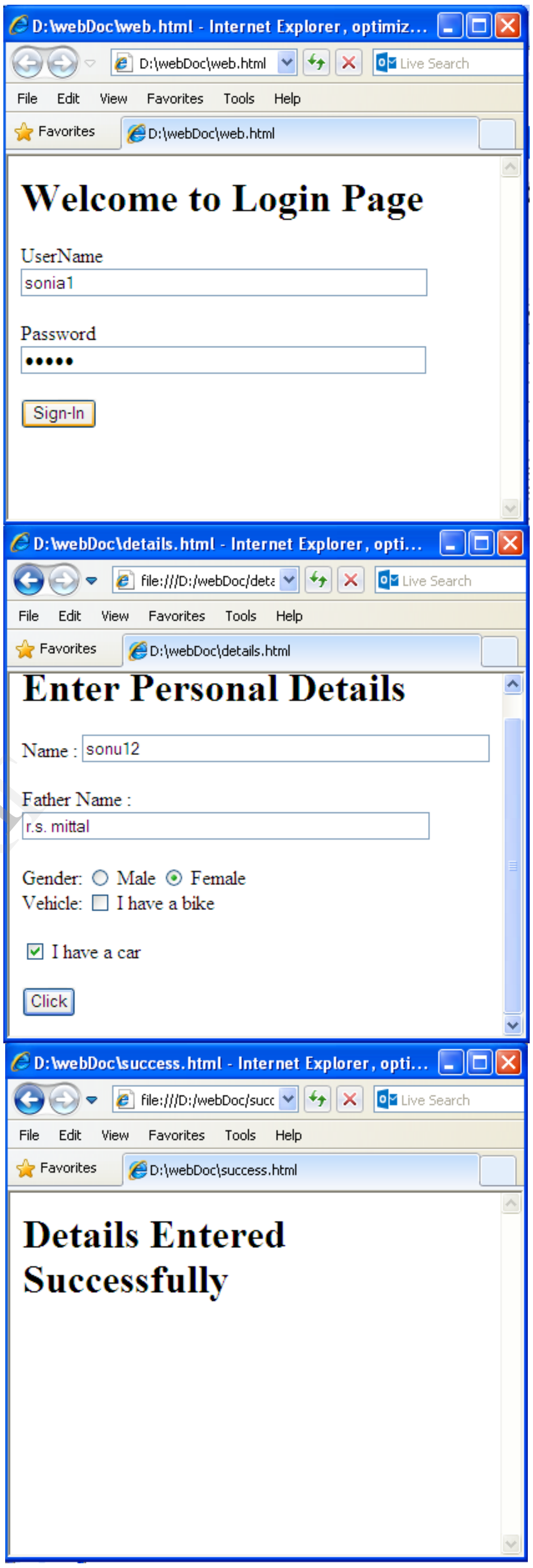
TestScript Name: web

● Multiple Values

Multiple Values

S.NO	KEYWORD	LABEL	TYPE	
0	URL		URL	op
1	loginBox	UserName	text	se
2	passUHHH	Password	password	se
3	save		submit	cl
4	NextUrl		action	op
5	Name	Name :	text	se
6	fName	Father Name :	text	se
7	vehicle	Vehicle:	checkbox	se
8	Gender	Gender:	radio	se
9	save		submit	cl

Figure 5: Parameterization


D:\webDoc\web.html - Internet Explorer, optimiz...
D:\webDoc\web.html
Live Search
File Edit View Favorites Tools Help
Favorites
D:\webDoc\web.html
Sign-In
D:\webDoc\details.html - Internet Explorer, opti...
file:///D:/webDoc/det
Live Search
File Edit View Favorites Tools Help
Favorites
D:\webDoc\details.html
Click
D:\webDoc\success.html - Internet Explorer, opti...
file:///D:/webDoc/succ
Live Search
File Edit View Favorites Tools Help
Favorites
D:\webDoc\success.html

Welcome to Login Page

UserName: sonia1

Password: •••••

Enter Personal Details

Name : sonu12

Father Name : r.s. mittal

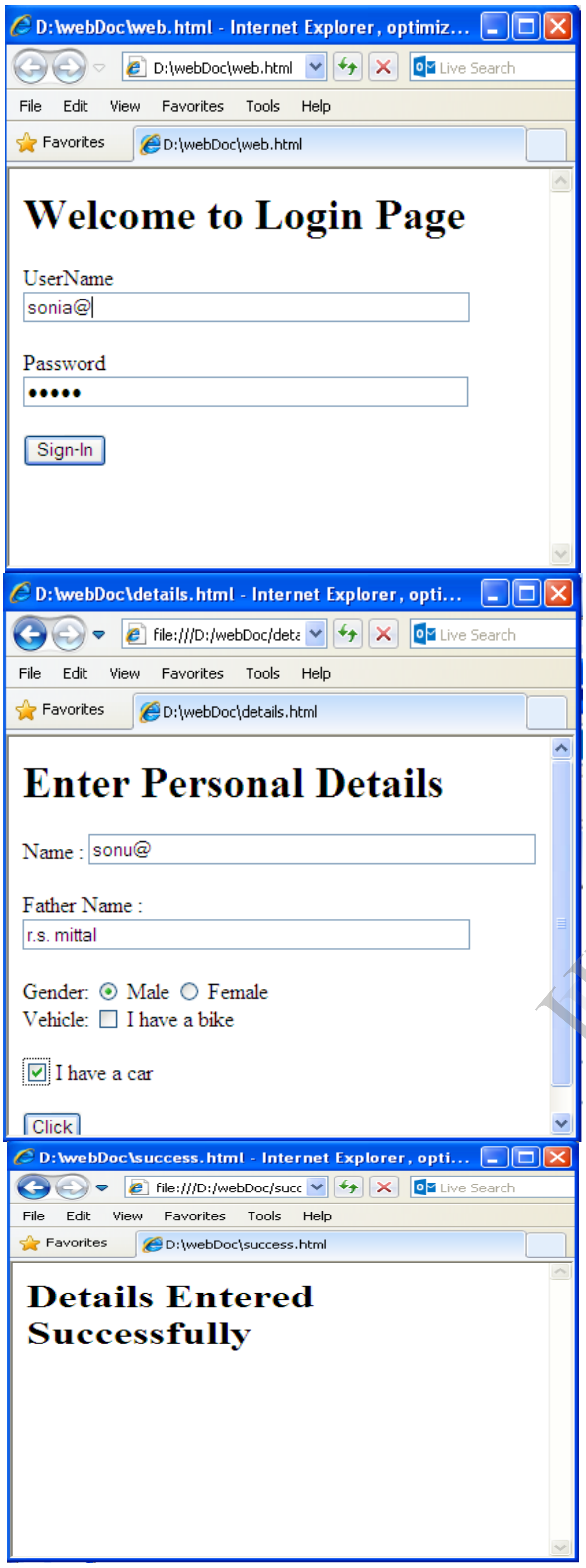
Gender: ○ Male [✓ ◉ Female]

Vehicle: ☐ I have a bike

☑ I have a car

Details Entered Successfully

Figure 6: Automated Playback of Parameterization Result for Value1


D:\webDoc\web.html - Internet Explorer, optimiz...
D:\webDoc\web.html
Live Search
File Edit View Favorites Tools Help
Favorites
D:\webDoc\web.html

Welcome to Login Page

UserName
sonia@

Password
•••••

Sign-In

D:\webDoc\details.html - Internet Explorer, opti...
file:///D:/webDoc/deta
Live Search
File Edit View Favorites Tools Help
Favorites
D:\webDoc\details.html

Enter Personal Details

Name : sonu@

Father Name :
r.s. mittal

Gender: [✓ Male] Female
Vehicle: I have a bike

[✓ I have a car]

Click

D:\webDoc\success.html - Internet Explorer, opti...
file:///D:/webDoc/succ
Live Search
File Edit View Favorites Tools Help
Favorites
D:\webDoc\success.html

Details Entered Successfully

Figure 7: Automated Playback of Parameterization Result for Value2

VI. Conclusion

In this paper, we investigate the concept, benefit, challenges and various tools for the Keyword Driven Testing. We design a tool named “AutoKeyTester” which performs Keyword Driven Testing to test all functional aspects of any web application. AutoKeyTester is not a simple recording and play back tool, but it also provides other features like test reporting, parameterization, Keyword extraction, etc. In this, labels are also mapped with keywords to make test scripts more users friendly. Our future work will focus on the creation of module for multiple action and optimization of whole process.

References

[1] http://en.wikipedia.org/wiki/Software_testing
[2] Jingfan Tang, Xiaohua Cao,Albert Ma, “Towards Adaptive Framework of Keyword Driven Automation Testing”, International Conference on Automation and Logistics Qingdao, China September 2008, pp-1631-1637.
[3] Juha Rantanen,”Acceptance Test-Driven Development with Keyword-Driven Test Automation Framework in an Agile Software Project” Helsinki University of Technology, Department of Computer Science and Engineering, Software Business and Engineering Institute. 2007, pp. 1-102 .
[4] Ayal Zylberman and Aviram Shotten, “Test Language: Introduction to Keyword Driven Testing”. 2010, pp 1-7
[5] http://www.forrester.com
[6] Jie Hui, Lan Yuqing, Luo Pei, Gao Jing, Guo Shuhang, “LKDT: A Keyword-Driven Based Distributed Test Framework”, International Conference on Computer Science and Software Engineering, 2008, pp. 719-722
[7] Pekka Laukkanen, “Data-Driven and Keyword-Driven Test Automation Frameworks”, Helsinki University of Technology, Department of Computer Science and Engineering Software Business and Engineering Institute. 2007, pp. 1-102
[8] Tommi Takala, Mika Maunumaa, and Mika Katara. ”An Adapter Framework for Keyword-Driven Testing”, Department of Software Systems, Tampere University of Technology, Finland. Ninth International Conference on Quality Software. 2009, pp. 201-210
[9] Bharath Anand R., Harish Krishnankutty, kaushik Ramakrishnan, Venkatesh V.C.,” Business Rules- Based Test Automation- A novel Approach for accelerated testing”. 2007, pp. 1-12