System Requirements Specification

CSE 361 - Spring 2014

**Regression Testing System**

|  |  |
| --- | --- |
| Authors:  Alex Tobias  James Duin | Supervisor:  Dr. Greg Rothermel |

February 10, 2014

# **Table of Contents**

[Table of Contents](#_xjtwv3yydf1e)

[Revision History](#_sg4un0fqju49)

[1. Introduction](#_yng1l0oi8anu)

[1.1 Purpose](#_1whu1kractnu)

[1.2 Document Conventions](#_grz7s8d8vcv0)

[1.3 Intended Audience and Reading Suggestions](#_1ezz2msvlems)

[1.4 Product Scope](#_prlo5pqes0r6)

[1.5 References](#_r1thgguc72pf)

[2. Overall Description](#_v7q48a99jza5)

[2.1 Product Perspective](#_3ifu1d4gmsa)

[2.2 Product Functions](#_5ficg7itaoi)

[2.3 User Classes and Characteristics](#_aajha6h0pvm2)

[2.4 Operating Environment](#_si9u0yuguldh)

[2.5 Design and Implementation Constraints](#_a2gplbofwgrq)

[2.6 User Documentation](#_tdmypsuzynwf)

[2.7 Assumptions and Dependencies](#_qmt6zij9pzx)

[3. External Interface Requirements](#_7gs61a4vi24j)

[3.1 User Interfaces](#_siy11hy3shvf)

[3.2 Hardware Interfaces](#_rwd2p7fqzdf7)

[3.3 Software Interfaces](#_x6vfjgblhvo1)

[3.4 Communications Interfaces](#_i75apk7xv8jo)

[4. System Features](#_g4nj63n7pqmi)

[4.1 Phase I](#_x6p79izf2x5y)

[4.1.1 Description and Priority](#_1snf0qa7j3g5)

[4.1.2 Stimulus/Response Sequences](https://docs.google.com/document/d/1vvTGE7Q-JYM-zp4KOoHr-gdo6cljRL_l9P-_RtwoIak/edit#heading=h.2ptmtaps8blo)

[4.1.3 Functional Requirements](#_vdxkqbg4vnuf)

[4.2 Phase II](#_u1yr2k8ay2ky)

[4.3 Future Release](#_f04aeqtxpw4z)

[5. Other Nonfunctional Requirements](#_90ua7k8d5vid)

[5.2 Performance Requirements](#_ru9fcsmikmyv)

[5.2 Safety Requirements](#_iw8unerzf8o2)

[5.3 Security Requirements](#_7xiqyze1kdws)

[5.4 Software Quality Attributes](#_h8cwfh53w3ly)

[5.5 Business Rules](#_bhbmsrvr2u95)

[6. Other Requirements](#_45x1sekxiun4)

# **Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Alex Tobias & James Duin | 2-10-2014 | Initial draft | 1st Draft |
|  |  |  |  |

# **1.** **Introduction**

## 1.1 Purpose

<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.>

The purpose of this document is to describe the requirement specifications for a Regression Testing System for software development teams. This document will cover the creation of the entire system and detail the initial two phases of the system along with features to be added in the future.

## 1.2 Document Conventions

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>

The requirements will be grouped into the three phases of the product’s release, each requirement within a phase has its own priority in relation to other requirements in that phase. The list of requirements will show the priority of the requirements, that is, the higher the priority of the requirement the higher it is in the list. The definitions, acronyms, and abbreviations used in this document are as follows:

**QA** - Quality Assurance

**Test Case** - A browser automation script possibly written to be run by Selenium or Sahi.

**Test Suite** - A grouping of test cases, test case groupings are not mutually exclusive.

**Web App** - Web based application.

**Environment** - The url that the tests are to be executed against. This can be either point to a live site, or a development site that contains features are currently being worked on.

**CSE** - Computer Science and Engineering department at the University of Nebraska at Lincoln.

**VM** - Virtual Machine.

**Front End** - This references the web application portion of our software system.

**Back End** - This references the part of our software system responsible for executing the configuration completed by the user in the webapp.

## 1.3 Intended Audience and Reading Suggestions

<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>

This document is intended for the developers of this product, project managers, marketing staff, users, testers. For project managers, users, and marketing staff, we recommend reading the Product Scope, Product Functions, and User Interfaces sections and then the requirements breakdown in the System Features section. The Operating Environment, Software Interfaces, and Assumptions and Dependencies sections are meant primarily for testers and developers.

## 1.4 Product Scope

<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>

For our project, we will create a regression testing software platform for Quality Assurance (QA) testers. The user base would be QA testers that have groups of browser automation test cases but do not have a way of running those test cases on a schedule or on multiple browser/os combinations. The idea is the qa tester would access our regression testing web app and would upload the code for their browser automation test cases. They would then group the uploaded test cases into test suites via configuration pages on our website. On another configuration page they would specify the testing schedules they desire. These testing schedules consist of the following: the test suite, the time schedule they want the tests run, the browser/os combination, and the environment (live or dev url) they want to test on. The tests then will be run automatically for them and they will be able to see the test result logs on the website. They also will be able to see summary report pages that the website will generate from the test results. The reports will provide QA testers with the ability to track percent failure and test time for a testing suite over successive releases.

## 1.5 References

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

The following web addresses are references for the component open source software systems that will be integrated into our Regression Testing System:

1) Jenkins. (n.d.). *About Jenkins CI*. Retrieved Feb 9, 2014, from Jenkins: http://jenkins-ci.org/content/about-jenkins-ci

2) Sahi. (n.d.). *About Us | Sahi*. Retrieved Feb 9, 2014, from Sahi: http://sahi.co.in/about-us/

3) Selenium. (n.d.). *About Selenium*. Retrieved Feb 9, 2014, from Selenium: http://docs.seleniumhq.org/about/

4) VirtualBox. (n.d.). *Screenshots – Oracle VM VirtualBox*. Retrieved Feb 9, 2014, from VirtualBox: https://www.virtualbox.org/wiki/Screenshots

# **2.** **Overall Description**

## 2.1 Product Perspective

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

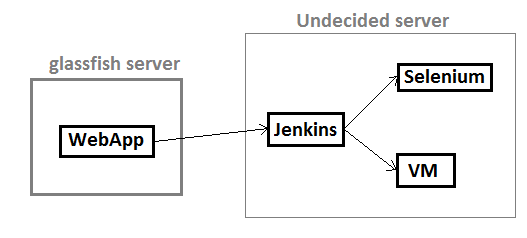
This project will be a new, self-contained product. The major components are as follows:

a) WebApp - The User Interface for our system, run on a glassfish server, communicates with our Jenkins configuration

b) Jenkins - Continuous integration tool written in Java, communicates Selenium and VM

c) Selenium - Browser automation software

d) VM - virtual machine, a software based emulation of a computer



## 2.2 Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

* The user accesses a script upload page on the webapp and specifies the script name, and what type of script it is.
* The user then sets up when the script will run and save’s the information on the page.
* The product will then display the results of the user’s script after the script is run on it’s specified time/date

## 2.3 User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

The main class of user would be QA testers that would use the system execute browser automated regression testing for a web application on a schedule. The important features are the ability to upload existing code and have it be run on a specified browser/os virtual machine. The QA tester will have access to all of the features of the web app.

## 2.4 Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

The platform involves integrating existing open source tools for browser automation, virtualization and running test jobs, with a web application that we build ourselves. The platform contains the following components:

1) Jenkins: A continuous integration server. This will be used to kick off test cases on a schedule and deploy test results to our Regression web application. This will need to be installed on the testers pc and have access to the server the that hosts the webapp. The QA tester will specify the schedule for the tests, the browser and machine that the tests will be executed on, and environment (live or dev url) that the tests will be run against. Jenkins will use shell scripts to execute the tests on the specified browser and machine. When the tests have concluded shell scripts in Jenkins will add any extra information to the test result xmls, such as machine and environment, and deploy the test results to the webapp.

2) VirtualBox: An open source virtualization software package. This will be used to create several virtual machines that will be setup with different browser and OS combinations. This will also have to be installed on the testers pc and will be limited to Windows XP and linux instances as these are freely available.

3) Selenium and Sahi: Both are browser automation and scripting software. This will be used to code and run the test cases which will be started by Jenkins. This will need to be installed on the testers pc and Virtual machine instances. It will also generate the log files that will be modified and deployed by shell scripts within Jenkins. We will have to code some dummy test cases in the Selenium script syntax just to demonstrate our application.

4) Regression testing web application: This will parse out information from the log files and store the information in a database, generate report pages. It should give QA engineers necessary metrics on regression testing over many releases of a product.

## 2.5 Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>

Since the backend operations of the web app will be on a different machine than the server that is hosting the web app, the two sides will have to communicate with XML data objects. The actual web page will be limited to using HTML, javascript, and java. The backend operations will be limited to using Jenkins. Also, since the delivered software is a public facing web app, we will maintain the delivered software.

## 2.6 User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

No user documentation components at this time, online instruction page might be incorporated in a future release.

## 2.7 Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

The assumed factor for this project is mainly the project’s dependency on Jenkins, Selenium, and Sahi to be reliable tools for our page to use. Also, we assume any automatically generated code inside the IDE we will use (Eclipse, NetBeans) will be correct and work properly.

# **3.** **External Interface Requirements**

## 3.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

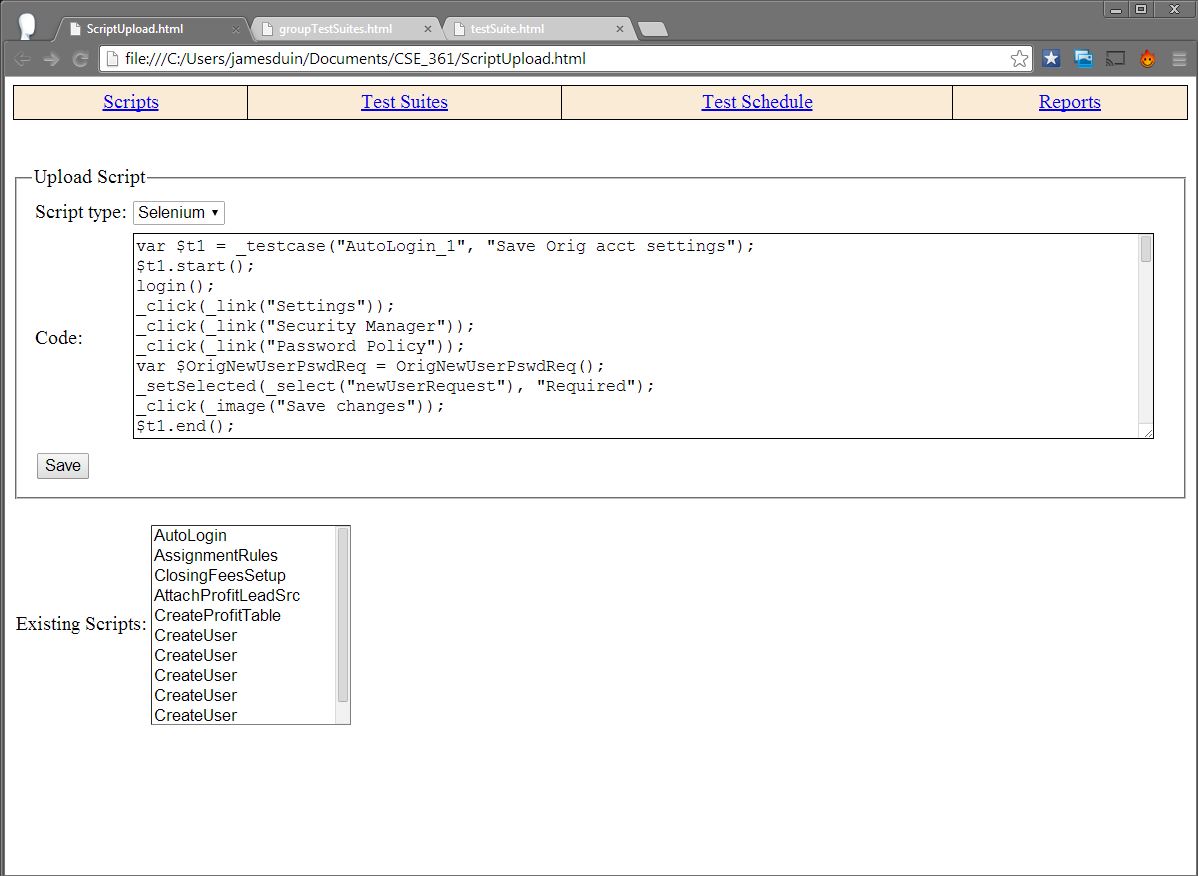


Figure 1: Page to upload a script.

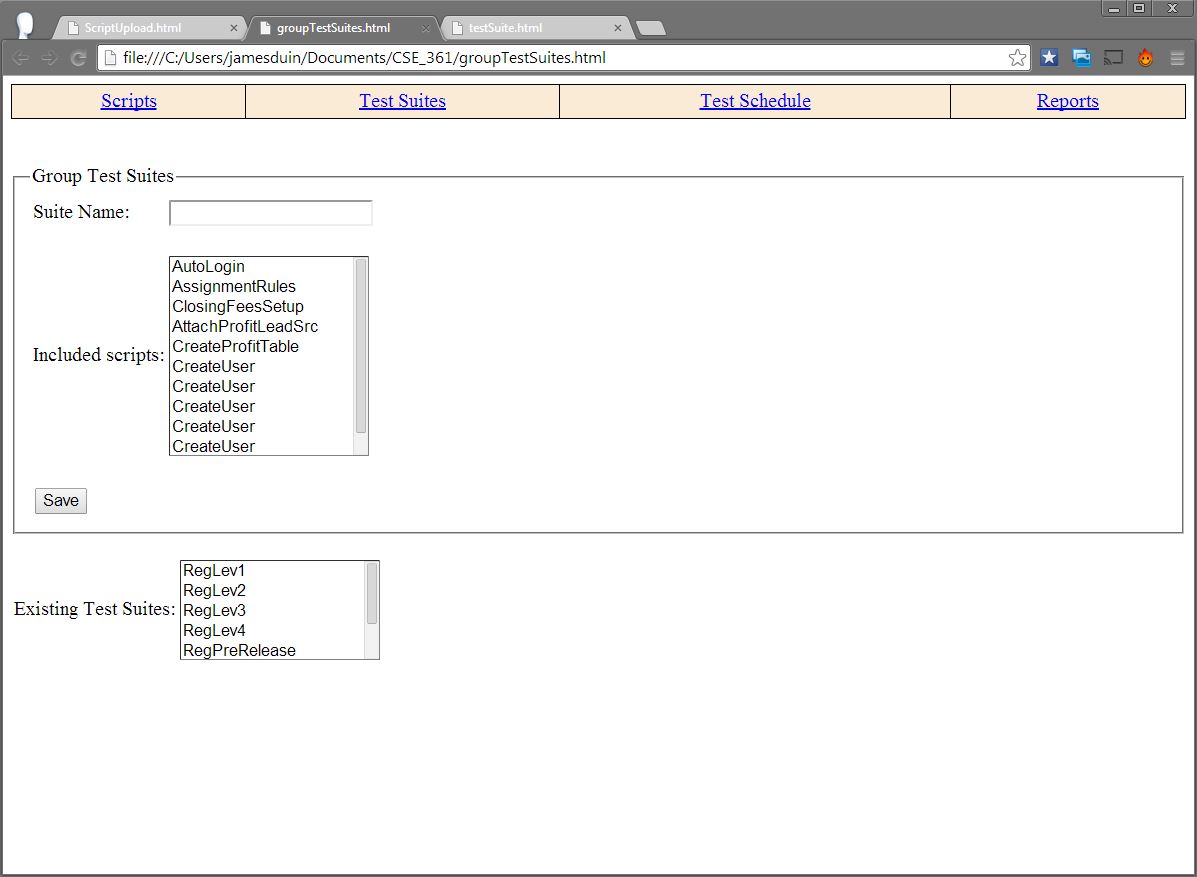


Figure 2: Page to group scripts into test suites.

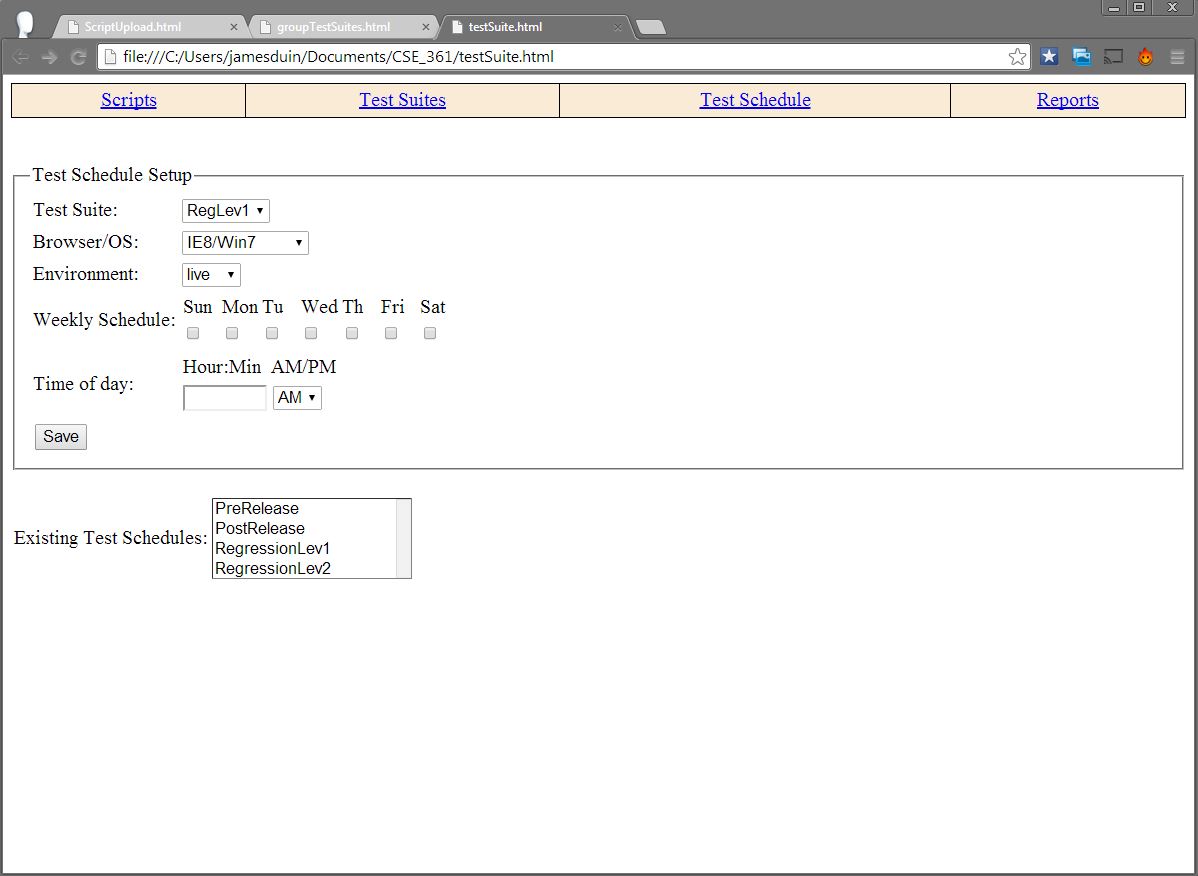


Figure 3: Page to specify a testing schedule.

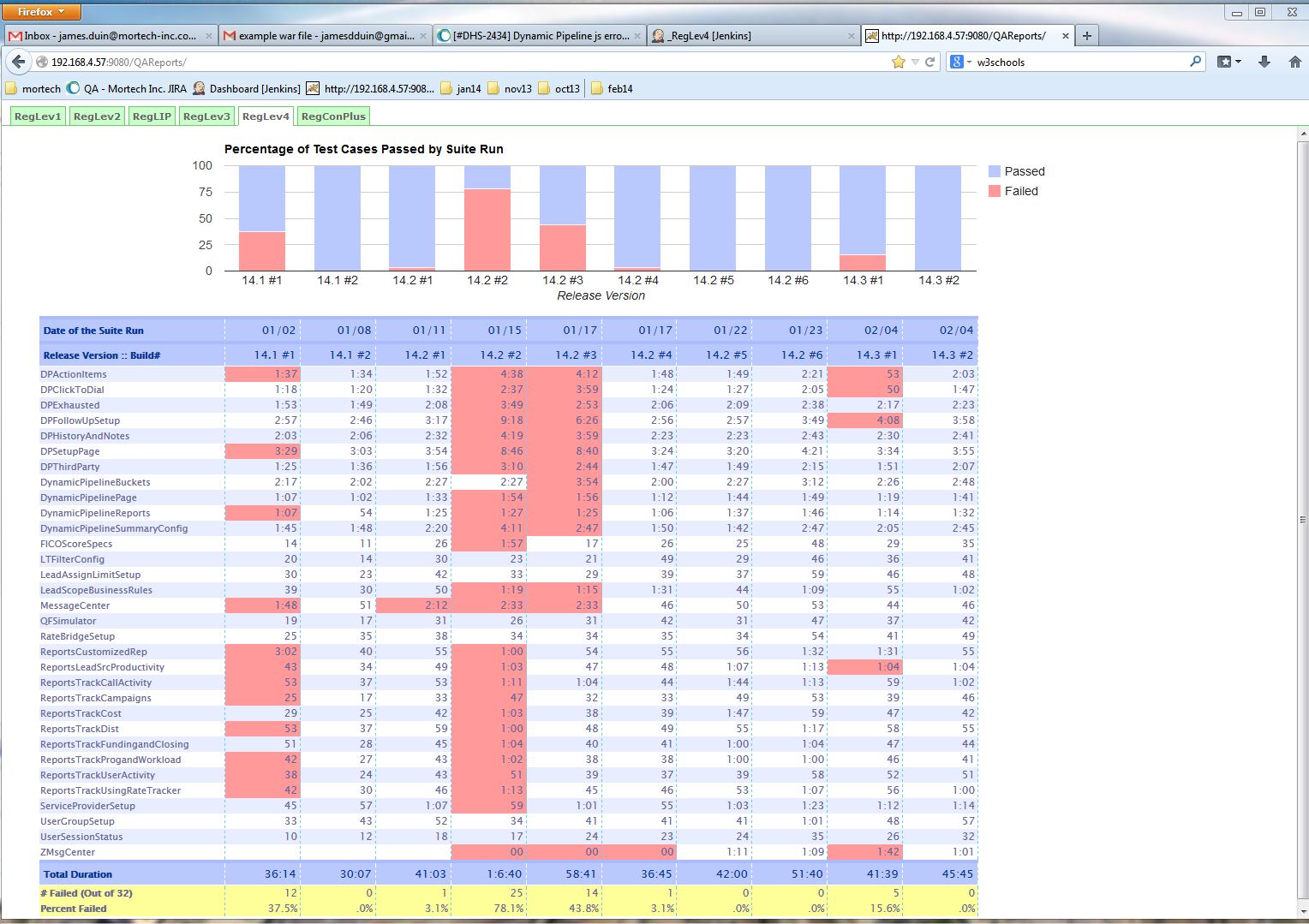


Figure 4: Report pages that show the results of regression testing from successive releases.

## 3.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

Our system will work on any device that has a functional web browser installed and a reliable internet connection.

## 3.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

## 3.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

To be able to use the web app, the user is required to have a functioning web browser along with a reliable internet connection. To send data from the web page to the Jenkins configuration, our web app will use XML objects.

# **4.** **System Features**

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

## 4.1 Phase I

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1 (Add Test Case): User must be able to upload a selenium/sahi script with test case settings, and see test results when it runs

REQ-2(Save Test Case): User must be able to save a test case, leave the page, and have the test case be there when the page returns.

### 4.1.1 Description and Priority

For the first phase of our system we will provide only the highest priority features, so as to demonstrate the most important aspects of the product functioning. The main features are the ability for the user to upload code for a test case in Sahi and have that script be executed on a virtual machine on the specified schedule and the test results be accessible to the user on the reports page.

### 4.1.2 Stimulus/Response Sequences

The user will upload the code for a test case in Sahi, and enter in all of the test configurations on the web app. When the test case is ready, the user will click ‘Save’ on the web page. This will trigger an event which grabs all of the data given by the user, formats it into an XML object, and posts it for Jenkins to see. Jenkins will be set up to check for new test case objects every 5 minutes, and will grab the XML object, and use the information to configure the test case. Then Jenkins will run the test case and produce a new XML object with the test case results, which it will then post back to the web page. Then, on the reports page of the web app, it will parse through the provided XML object and fill out the page with the test case data.

### 4.1.3 Functional Requirements

REQ-1: The system shall allow a user to upload a single script in Sahi code via a Scripts page, this script will be saved and should be accessible whenever navigating to ths Scripts page.

REQ-2: The system will allow the user to view the results of the test case after it has run to completion.

## 4.2 Phase II

### 4.2.1 Description and Priority

For the second phase of our system we will add on a new page to the web app that allows the user to group multiple test cases into a test suite, and save the created test suite. They will also

### 4.2.2 Stimulus/Response Sequences

The user will upload the code for a test case in Sahi, and enter in all of the test configurations on the web app. When the test case is ready, the user will click ‘Save’ on the web page. This will trigger an event which grabs all of the data given by the user, formats it into an XML object, and posts it for Jenkins to see. Jenkins will be set up to check for new test case objects every 5 minutes, and will grab the XML object, and use the information to configure the test case. Then Jenkins will run the test case and produce a new XML object with the test case results, which it will then post back to the web page. Then, on the reports page of the web app, it will parse through the provided XML object and fill out the page with the test case data.

### 4.2.3 Functional Requirements

REQ-1: The system will allow the user to upload multiple test case scripts, and group them into suites which can be saved and run at a later time.

REQ-2: The system will allow the user to select from a list of several VM’s and the test script will run on that particular VM.

## 4.3 Future Release

Possibilities for future release:

- Add a tutorial document of some sort that can be accessed from the web app.

-Add the ability to save past test results

# **5.** **Other Nonfunctional Requirements**

## 5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

The user does not need to be able to run a test immediately, in fact the only way they can run tests through our system is by grouping them into a test suite and specifying a schedule for that test suite to run. The

## 5.2 Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could

result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>

No safety requirements are needed at this time, no foreseeable damage or harm will result from the use of the product.

## 5.3 Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

Since this is a public facing web app that doesn’t ask for any personal information from the user, there are no security requirements needed at this time.

## 5.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

adaptability -

availability -

correctness -

flexibility -

interoperability -

maintainability -

portability -

reliability -

reusability -

robustness -

testability -

usability -

## 5.5 Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

Anyone who has access to this web app can utilize the full functionality of it.

# **6.** **Other Requirements**

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

**Jenkins** - Jenkins is an open source continuous integration tool written in Java. The project was forked from Hudson after a dispute with Oracle. Jenkins provides continuous integration services for software development.

**XML** - Extensible Markup Language

**Test Case** - A browser automation script possibly written to be run by Selenium or Sahi.

**Test Suite** - A grouping of test cases, test case groupings are not mutually exclusive.

**Web App** - Web based application.

**Environment** - The url that the tests are to be executed against. This can be either point to a live site, or a development site that contains features are currently being worked on.

**CSE** - Computer Science and Engineering department at the University of Nebraska at Lincoln.

**VM** - Virtual Machine.

**Front End** - This references the web application portion of our software system.

**Back End** - This references the part of our software system responsible for executing the configuration completed by the user in the webapp

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>