Department of Veterans Affairs (VA)

Benefits Claims Decision Support System (BCDSS)

**Testing Manual**

****

**January 2017**

**Version 1.11**

**Revision History**

| **Date** | **Revision** | **Description** | **Project Manager / Author** |
| --- | --- | --- | --- |
| January 21, 2017 | 1.11 | Updated Back End Testing as current and last sprint involved user to validate front end values | Bhupinder Singh |
| December 21, 2016 | 1.10 | Updated [Regression Testing](#_Regression_Testing) with suggested cosmetic changes. | Bhupinder Singh |
| November 21, 2016 | 1.9 | No Change except Month label update | Bhupinder Singh |
| October 20, 2016 | 1.8 | No Change except Month label update | Bhupinder Singh |
| September 26, 2016 | 1.7 | Editorial Changes | Bhupinder Singh |
| September 19, 2016 | 1.6 | No Change except Month label update | Bhupinder Singh |
| August 18,2016 | 1.5 | No Change except Month label update | Bhupinder Singh |
| June 28, 2016 | 1.4 | Updated General formatting | Bhupinder Singh |
| June 20, 2016 | 1.3 | Updated General formatting | Bhupinder Singh |
| May 23, 2016 | 1.2 | Updated Performance, stress and load test areas. | Bhupinder Singh  Dominic Yeh  Erik Rothwell |
| May 5, 2016 | 1.0 | Initial Version | Bhupinder Singh  Dominic Yeh  Erik Rothwell |

Table of Contents

[1. Introduction 5](#_Toc472085557)

[1.1. Purpose 5](#_Toc472085558)

[1.2. Scope 5](#_Toc472085559)

[2. Functional Testing 5](#_Toc472085560)

[2.1. Unit Testing 5](#_Toc472085561)

[2.1.1. Limitations of Unit Testing 6](#_Toc472085562)

[2.2. Integration Testing 6](#_Toc472085563)

[2.3. System Testing 6](#_Toc472085564)

[2.4. Regression Testing 7](#_Toc472085565)

[2.5. Acceptance Testing 7](#_Toc472085566)

[2.6. Back End Testing 7](#_Toc472085567)

[3. Non-Functional Testing 8](#_Toc472085568)

[3.1. Performance Testing 8](#_Toc472085569)

[3.1.1. Load Testing 8](#_Toc472085570)

[3.1.2. Stress Testing 8](#_Toc472085571)

[3.2. Security Testing 8](#_Toc472085572)

[3.3. 508 Compliance Testing 9](#_Toc472085573)

[4. Test Summary Report 9](#_Toc472085574)

[5. Approval Signatures 11](#_Toc472085575)

# Introduction

## Purpose

The purpose of this document is to describe a standardized step-by-step testing procedure to follow and evaluate the Benefits Claims Decision Support System (BCDSS) application and its components with the intent to find that whether it satisfies the all specified requirements or not. These standards and procedures state the general standards and procedures to follow to plan and conduct software testing and validation.

## Scope

The Testing Manual will outline the strategy that will be used to test the BCDSS Pilot Project Module. The BCDSS Pilot project will develop a system that predicts the rating of claims and their levels based upon historical facts/data.

# Functional Testing

This is a type of black box testing that is based on the specifications of the software that is to be tested. The BCDSS application will be tested by providing input to the specific test and then, analyzing the results based on the functionality it was intended for. Functional Testing of the software will be conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. Functional Testing will be carried out as described in the contract.

There will be five steps involved when testing the BCDSS application for functionality:

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | The determination of the functionality that the intended application is meant to perform. |
| 2 | The creation of test data based on the application specifications. |
| 3 | The output based on the test data and the application specifications. |
| 4 | The writing of Test Scenarios and the execution of test cases. |
| 5 | The comparison of actual and expected results based on the executed test cases. |

As an effective testing practice, the Testing Team will apply the above steps during the testing activities of the BCDSS application to make sure that the Team maintains the highest standards when it comes to software quality.

## Unit Testing

This type of testing will be performed by the developers before the setup is handed over to the Testing Team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of the source code assigned areas. The developers use test data that is separate from the test data of the Pilot Team.

The goal of Unit Testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

### Limitations of Unit Testing

Unit Testing cannot catch each bug in an application. It is impossible to evaluate every execution path in every software application. The same is the case with unit testing.

There is a limit to the number of scenarios and test data that the developer can use to verify the source code. Therefore, after the developer has exhausted all options there is no choice but to stop unit testing and merge the code segment with other units.

## Integration Testing

Integration Testing determines that the combined parts of an application function correctly together when they are integrated. There are two methods of doing Integration Testing: Bottom-up Integration testing and Top-down Integration testing.

|  |  |
| --- | --- |
| **S.N.** | **Integration Testing Method** |
| 1 | **Bottom-up integration**  This testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds. |
| 2 | **Top-Down integration**  In this testing, the highest-level modules are tested first and progressively lower-level modules are tested after that. |

In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic those it will encounter in customers' computers, systems, and network.

## System Testing

This is the next level of testing to be used in the BCDSS. The application as a whole is tested rigorously to meets the Quality Standards. The System Testing is important for the following reasons:

* The System Testing is the first step in the Software Development Life Cycle, where the application is tested as a whole.
* The application is tested thoroughly to verify that it meets both the functional and technical specifications.
* The application meets all the acceptance criteria.
* The application is tested in an environment which is very close to the production environment where the application will be deployed.
* System Testing enables us to test, verify, and validate both the business requirements as well as the application architecture.

## Regression Testing

There is a possibility that other functionalities within the application will get affected with a change in a specific area of the software. To verify that a bug fix didn’t result in another functionality or business rule violation, Regression Testing happens after implementing the changes. The intent of Regression Testing is to ensure that a change, such as a bug fix or suggested cosmetics changes did not result in another fault being uncovered in the application.

Regression Testing will occur as many times as needed during the BCDSS software development lifecycle to:

* Minimize the gaps in testing when there is an application with new changes to test.
* Verify that the change made did not affect any other area of the application.
* Mitigate risks when conducting regression testing on the application.
* Increase test coverage without compromising timelines.
* Increase speed to market the product.
* Ensure that defects found during testing are either remediated and retested or are carried over to the next sprint as a new user story.

## Acceptance Testing

Acceptance Testing is probably the most important type of testing as it will be conducted by the “Pilot Team”, who will measure whether the application meets the intended specifications and satisfies the client’s requirements. The Quality Assurance (QA) Team will have a set of pre-written BCDSS scenarios and test cases that will be used to test the application.

More ideas will be shared about the application and more tests can be performed on it to gauge its accuracy and the reasons why the project was initiated. Acceptance tests are not only intended to point out simple spelling mistakes, cosmetic errors or Interface gaps, but also to point out any bugs in the application that will result in system crashers or major errors in the application.

## Backend Testing

Backend testing is very significant particularly when measuring if the frontend interface shows the correct data where calculations are involved. This involves communicating to the user setting up the database the tables schema configuration. The user has to have appropriate login credentials and be able to execute SQL commands to verify database elements. Then the user has to validate that the frontend interface is mapped to the correct values.

# Non-Functional Testing

This section is based upon the testing of the BCDSS application non-functional attributes. Non-Functional Testing of software involves testing the software from the requirements which are non-functional in system, but important as well, such as, performance, security, and user interface, etc. Non-Functional Testing will be carried out if required by the client as described in the contract.

Some of the important and commonly used non-functional testing types are discussed below.

## Performance Testing

Performance Testing is not currently within the scope of testing.

### Load Testing

Load Testing is a process of testing the behavior of the System by applying maximum load in terms of System accessing and manipulating large amounts of input data. It can be done at both normal and peak-load conditions. This type of testing identifies the maximum capacity of the System and its behavior at peak time.

Most of the time, Load Testing is performed using automated tools such as Load Runner, IBM Rational Performance Tester, Apache, Silk Performer, Visual Studio Load Test etc.

Virtual Users (VUsers) are defined in the automated testing tool and the script is executed to verify the Load testing for the Software. The quantity of users can be increased or decreased concurrently or incrementally based upon the requirements

### Stress Testing

This testing type includes the testing of System behavior under abnormal conditions. Taking away the resources and applying load beyond the actual load-limit is Stress testing.

The main intent is to test the System by applying the load to the system and taking over the resources used by the System to identify the breaking point where the component crashes. This testing can be performed by testing different scenarios such as:

* Shutdown or restart of network ports randomly
* Turning the database on or off
* Running different processes that consume resources such as CPU, Memory, server, etc.
* Monitoring the impact of network traffic

## Security Testing

Security Testing involves the testing of Software in order to identify any flaws and gaps from the security and vulnerability point of views. Following are the main aspects which Security Testing should ensure:

* Confidentiality.
* Integrity.
* Authentication.
* Availability.
* Authorization.
* Non-repudiation.
* Software is secure against known and unknown vulnerabilities.
* Software data is secure.
* Software is according to all security regulations.
* Input checking and validation.
* SQL insertion attacks.
* Injection flaws.
* Session management issues.
* Cross-site scripting attacks.
* Buffer overflows vulnerabilities.
* Directory traversal attacks.

## 508 Compliance Testing

Section 508 requires that all Web site content be equally accessible to people with disabilities. This applies to Web applications, Web pages and all attached files on the intranet, as well as, internet. Please refer to following URL for resources for automated test tools for Section508. <http://www.section508.va.gov/section508/Resources.asp>.

508 Compliance Testing will be performed as specified in the BCDSS contract Performance Work Statement (PWS) using JAWS tool.

# Test Summary Report

As the last step of this Testing Manual, the Test Summary Report of BCDSS application will be created to analyze the software testing, unit/module, subsystem integration, system, user acceptance, and security - as defined in the test plan. The Test Summary Report summarizes the results of the tests, evaluation, and corrective action plan.

This manual will be updated throughout the BCDSS platform development.

# Approval Signatures

This section is used to document the approval of the BCDSS Software Testing Manual during the Formal Review. The review should be conducted face to face, where signatures can be obtained “live” during the review.

If unable to conduct a face-to-face meeting, then it should be held via LiveMeeting and concurrence captured during the meeting. The Scribe should add /estimated/name by each position cited.

The Business Sponsor and Project Manager are required to sign.

REVIEW DATE:

SCRIBE:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

< Business Sponsor >

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

< Project Manager >