Test Report

Virtual Letter of Life (VLOL) Application

Version 1.0

Heather Barnes

Andrew Coleman

Panhavorn Hok

Michael Marcucci

Michael Shaw

Kimberly Van Allen

SWEN 670 Software Engineering Project

University of Maryland Global Campus

Date:11/3/2020

**REVISION HISTORY**

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Description** |
| 10/31/2020 | 1.0 | Initial Test Report |

**Table of Contents**

[1. Introduction 5](#_Toc55303649)

[1.1 Purpose 5](#_Toc55303650)

[1.2 Application Overview 5](#_Toc55303651)

[2. Types of testing performed 5](#_Toc55303652)

[2.1. Unit 5](#_Toc55303653)

[2.2. Integration & System 5](#_Toc55303654)

[2.3. User Acceptance Testing 6](#_Toc55303655)

[3. Items not tested 6](#_Toc55303656)

[4. Metrics and Testing 6](#_Toc55303657)

[4.1. JUnit Test Cases 6](#_Toc55303658)

[4.2 SonarQube 10](#_Toc55303659)

[4.3 Selenium 10](#_Toc55303660)

[4.3.1 Administrator Test Suite 11](#_Toc55303661)

[4.3.2 Participant Test Suite 17](#_Toc55303662)

[4.3.3 Provider Test Suite 23](#_Toc55303663)

[4.3.4 Unauthorized User Test Suite 26](#_Toc55303664)

[4.4 Manual 31](#_Toc55303665)

[4.4.1 Manual Test Suite 32](#_Toc55303670)

[4.4.2 Testing the QR Code 34](#_Toc55303671)

[5 Defects Identified 35](#_Toc55303672)

[6 Exit Criteria 36](#_Toc55303673)

[7 Conclusion/Sign Off 37](#_Toc55303674)

**Table of Tables**

[Table 1: JUnit Final Execution Details 6](#_Toc55303677)

[Table 2: Selenium Final Execution 11](#_Toc55303678)

[Table 3: Test Suites and Requirements Tested 11](#_Toc55303679)

[Table 4: Manal Test Final Execution 31](#_Toc55303680)

[Table 5: Defect Matrix 35](#_Toc55303681)

[Table 6: Defect Description and Status 36](#_Toc55303682)

[Table 7: Sign Off 37](#_Toc55303683)

**Table of Figures**

[Figure 1: SonarQube Scan Results 10](#_Toc55226391)

[Figure 2: Defects Identified based on severity 35](#_Toc55226392)

# Introduction

## Purpose

The Test Report Summary is intended to outline the testing activities of the Virtual Letter of Life (VLOL) application. The implementation of testing strategies that were used to conduct this testing as outlined in the Master Test Plan version 1.2. In addition to functional requirements stated in the test plan, some additional items were tested that were suggested as improvements to the system. These items came from feedback from the VLOL team and stakeholders. In this Test Report, the following will be provided:

* Types of testing performed
* Items not tested
* Testing Metrics
* Test cases and results
* Defects identified

## 1.2 Application Overview

The VLOL application enables users to fill out a Virtual Letter of Life form, which emergency personnel and medical providers can view and update. The VLOL system provides demographic information along with patient medical conditions, allergies, medications, vaccinations, insurance information, and primary care physician.

# Types of testing performed

## Unit

The back-end portion of the VLOL application was written in the Java programming language. Testers have implemented the White-Box strategies technique. Testers have utilized Junit via the Bottom-Up approach, which is the conventional testing framework that most developers use when building an application in Java. The JUnit framework uses assert methods to perform proper back-end Unit Testing.

## Integration & System

For the VLOL application, developers executed the Integration and System Testing in sequential phases. For this instance, testers implemented the Big Bang approach where internal unit tests were conducted, and then the system was tested. The joint usage of JUnit and SonarQube helped handle the integration and system testing. The test classes and test suites created through JUnit were uploaded and analyzed by the SonarQube. This on-going transition provided a continuous integration and continuous development phase (CI/CD). In return, SonarQube generated a detailed report, which informed developers and testers of the existence of bugs, security vulnerabilities, potential duplicates, and code coverage analysis & percentage.

## User Acceptance Testing

The acceptance testing was completed using the Black-Box technique. VLOL developers followed the Black-Box technique using the front-end automated testing tool Selenium. All completed functional requirements stated in the SRS were validated using the Selenium software tool if possible. Some requirements were not able to be verified with the Selenium tool so manual testing was required.

# Items not tested

The following items are not in scope to be testing by the VLOL team.

* Network Security
* System deployment
* Load and Stress Testing

# Metrics and Testing

## JUnit Test Cases

The JUnit test cases were conducted to test the backend portion of the application. Developers used the JUnit framework and assert methods for each of the model classes in the application. The VLOL team ran 278 JUnit tests. Any defects identified were fixed prior to the code being committed to the database.

The final execution details of the JUnit tests are as follows:

| **JUnit Final Execution Details** | |
| --- | --- |
| Test Started | 11/02/2020 at 4:41 pm |
| Test End | 11/02//2020 at 4:45 pm |
| Test Conductor | Kimberly Van Allen |
| Number of Test Cases | 278 |
| Number of failures | 0 |

Table 1: JUnit Final Execution Details

Due to the number of JUnit tests run (278), a representative VLOL JUnit test is provided below. Inclusion of all 278 tests would require the report to exceed ~115 pages. All JUnit tests run by the VLOL team are embedded in the word document below.



**AdvanceDirectiveTest**

public class AdvanceDirectiveTest {

private Validator validator;

private final AdvanceDirective advanceDirective;

private final User user = new User();

private final byte[] bytes = new byte[] {1, 2};

public AdvanceDirectiveTest() {

// Instantiate the Allergy object

this.advanceDirective = new AdvanceDirective();

this.advanceDirective.setAdvanceDirectiveContentType("application/pdf");

this.advanceDirective.setAdvanceDirectiveFile(bytes);

this.advanceDirective.setAdvanceDirectiveFilename("test.pdf");

this.advanceDirective.setAdvanceDirectiveId(1L);

this.advanceDirective.setAdvanceDirectiveType("MOSLT");

this.advanceDirective.setUser(user);

}

@BeforeAll

public static void setUpClass() {}

@AfterAll

public static void tearDownClass() {}

@BeforeEach

public void setUp() {

// Setup validation of each method's validation annotations

ValidatorFactory factory = Validation.buildDefaultValidatorFactory();

validator = factory.getValidator();

}

/\*\* Test of getAdvanceDirectiveId method, of class AdvanceDirective. \*/

@Test

public void testGetAdvanceDirectiveId() {

System.out.println("getAdvanceDirectiveId");

Long expResult = 1L;

Long result = advanceDirective.getAdvanceDirectiveId();

assertEquals(expResult, result);

}

/\*\* Test of setAdvanceDirectiveId method, of class AdvanceDirective. \*/

@Test

public void testSetAdvanceDirectiveId() {

System.out.println("setAdvanceDirectiveId");

Long advanceDirectiveId = 2L;

advanceDirective.setAdvanceDirectiveId(advanceDirectiveId);

// Check for and print any violations of validation annotations

Set<ConstraintViolation<AdvanceDirective>> violations = validator.validate(advanceDirective);

String message =

violations.iterator().hasNext() ? violations.iterator().next().getMessage() : "";

if (!violations.isEmpty()) {

System.out.println("Violation caught: " + message);

}

// Test method

assertTrue(violations.isEmpty());

}

/\*\* Test of getAdvanceDirectiveFile method, of class AdvanceDirective. \*/

@Test

public void testGetAdvanceDirectiveFile() {

System.out.println("getAdvanceDirectiveFile");

byte[] expResult = bytes;

byte[] result = advanceDirective.getAdvanceDirectiveFile();

assertArrayEquals(expResult, result);

}

/\*\* Test of setAdvanceDirectiveFile method, of class AdvanceDirective. \*/

@Test

public void testSetAdvanceDirectiveFile() {

System.out.println("setAdvanceDirectiveFile");

byte[] advanceDirectiveFile = new byte[] {3, 4};

advanceDirective.setAdvanceDirectiveFile(advanceDirectiveFile);

// Check for and print any violations of validation annotations

Set<ConstraintViolation<AdvanceDirective>> violations = validator.validate(advanceDirective);

String message =

violations.iterator().hasNext() ? violations.iterator().next().getMessage() : "";

if (!violations.isEmpty()) {

System.out.println("Violation caught: " + message);

}

// Test method

assertTrue(violations.isEmpty());

}

/\*\* Test of getAdvanceDirectiveType method, of class AdvanceDirective. \*/

@Test

public void testGetAdvanceDirectiveType() {

System.out.println("getAdvanceDirectiveType");

String expResult = "MOSLT";

String result = advanceDirective.getAdvanceDirectiveType();

assertEquals(expResult, result);

}

/\*\* Test of setAdvanceDirectiveType method, of class AdvanceDirective. \*/

@Test

public void testSetAdvanceDirectiveType() {

System.out.println("setAdvanceDirectiveType");

String advanceDirectiveType = "TEST";

advanceDirective.setAdvanceDirectiveType(advanceDirectiveType);

// Check for and print any violations of validation annotations

Set<ConstraintViolation<AdvanceDirective>> violations = validator.validate(advanceDirective);

String message =

violations.iterator().hasNext() ? violations.iterator().next().getMessage() : "";

if (!violations.isEmpty()) {

System.out.println("Violation caught: " + message);

}

// Test method

assertTrue(violations.isEmpty());

}

/\*\* Test of getAdvanceDirectiveContentType method, of class AdvanceDirective. \*/

@Test

public void testGetAdvanceDirectiveContentType() {

System.out.println("getAdvanceDirectiveContentType");

String expResult = "application/pdf";

String result = advanceDirective.getAdvanceDirectiveContentType();

assertEquals(expResult, result);

}

/\*\* Test of setAdvanceDirectiveContentType method, of class AdvanceDirective. \*/

@Test

public void testSetAdvanceDirectiveContentType() {

System.out.println("setAdvanceDirectiveContentType");

String advanceDirectiveContentType = "application/doc";

advanceDirective.setAdvanceDirectiveContentType(advanceDirectiveContentType);

// Check for and print any violations of validation annotations

Set<ConstraintViolation<AdvanceDirective>> violations = validator.validate(advanceDirective);

String message =

violations.iterator().hasNext() ? violations.iterator().next().getMessage() : "";

if (!violations.isEmpty()) {

System.out.println("Violation caught: " + message);

}

// Test method

assertTrue(violations.isEmpty());

}

/\*\* Test of getAdvanceDirectiveFilename method, of class AdvanceDirective. \*/

@Test

public void testGetAdvanceDirectiveFilename() {

System.out.println("getAdvanceDirectiveFilename");

String expResult = "test.pdf";

String result = advanceDirective.getAdvanceDirectiveFilename();

assertEquals(expResult, result);

}

/\*\* Test of setAdvanceDirectiveFilename method, of class AdvanceDirective. \*/

@Test

public void testSetAdvanceDirectiveFilename() {

System.out.println("setAdvanceDirectiveFilename");

String advanceDirectiveFilename = "test2.pdf";

advanceDirective.setAdvanceDirectiveFilename(advanceDirectiveFilename);

// Check for and print any violations of validation annotations

Set<ConstraintViolation<AdvanceDirective>> violations = validator.validate(advanceDirective);

String message =

violations.iterator().hasNext() ? violations.iterator().next().getMessage() : "";

if (!violations.isEmpty()) {

System.out.println("Violation caught: " + message);

}

// Test method

assertTrue(violations.isEmpty());

}

/\*\* Test of getUser method, of class AdvanceDirective. \*/

@Test

public void testGetUser() {

System.out.println("getUser");

User expResult = user;

User result = advanceDirective.getUser();

assertEquals(expResult, result);

}

/\*\* Test of setUser method, of class AdvanceDirective. \*/

@Test

public void testSetUser() {

System.out.println("setUser");

User user = new User();

advanceDirective.setUser(user);

// Check for and print any violations of validation annotations

Set<ConstraintViolation<AdvanceDirective>> violations = validator.validate(advanceDirective);

String message =

violations.iterator().hasNext() ? violations.iterator().next().getMessage() : "";

if (!violations.isEmpty()) {

System.out.println("Violation caught: " + message);

}

// Test method

assertTrue(violations.isEmpty());

}

}

## 4.2 SonarQube

Below are the results from the SonarQube scan:

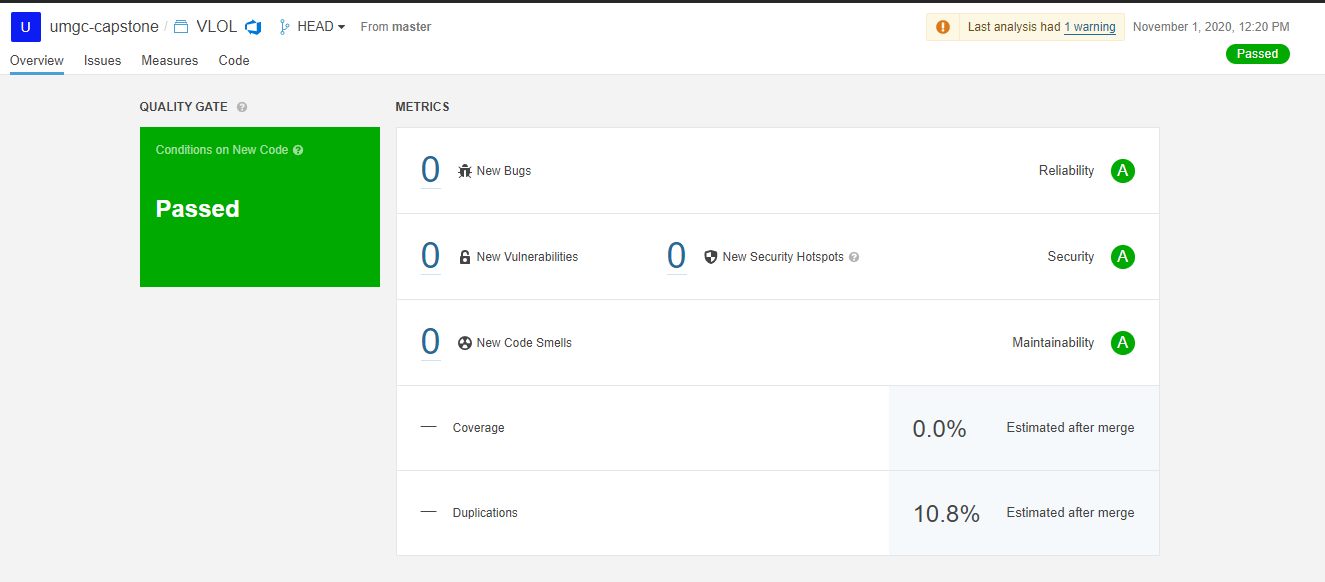


Figure 1: SonarQube Scan Results

## 4.3 Selenium

The test cases in this group focus on the functional requirements in which authenticated and unauthenticated user interacts with various web components of the VLOL application. The Selenium software recorded all tester interaction with the application (test transaction) to ensure that the UI screens projected are displaying the correct content. The table below provides a breakdown of the number of selenium tests planned and executed with the results.

| **Selenium Final Execution Details** | |
| --- | --- |
| Test Started | 11/02/2020 at 4:50 pm |
| Test End | 11/02//2020 at 5:30 pm |
| Test Conductor | Kimberly Van Allen |
| Number of Test Cases | 24 |
| Number of failures | 0 |

Table 2: Selenium Final Execution

The test cases that were conducted with the Selenium tool have been divided into three test suites. The suites are based on user roles and function according to the SRS document. To ensure all functionality is properly working the requirements completed by the previous team were tested as well. Below are the suits name and the requirements that they tested:

| **Suite Name** | **Number of Test Cases** | **New Requirements** | **Previously Completed Requirements** |
| --- | --- | --- | --- |
| Admin | 7 | none | CR1.17 – CR1.28 |
| Participant | 8 | Req 1.4 – Req 1.6 | CR1.3 – CR1.6 |
| Provider | 3 | Req 1.7 – Req 1.12 | CR 1.7 – CR1.11 |
| Unauthorized User | 6 | Req. 1.1 – Req1.3 | CR1.1 – CR1.2 |

Table 3: Test Suites and Requirements Tested

The Selenium test cases and results are below.

### **4.3.1 Administrator Test Suite**

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 1 |
| Test Name | Accounts without the active flag should not be able to log in |
| Description | The functional requirements were to ensure users an account verified, email verified or who were locked out were not able to login to the VLOL. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 2 |
| Test Name | Login view/edit my account |
| Description | The functional requirements were to enable an admin to login, view, and edit their own account information. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 3 |
| Test Name | View system activity log |
| Description | The functional requirement was for system administrators to view a log of when users last logged into their account. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 4 |
| Test Name | Add user accounts |
| Description | The functional requirement was for administrators to have the ability to create user accounts. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 5 |
| Test Name | Delete user accounts |
| Description | The functional requirement was for administrators to have the ability to delete user accounts. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 6 |
| Test Name | Search users, open user account, verify NPI for providers, verify 3 user roles |
| Description | The functional requirements were for the administrators to have the ability to search and open participants, administrators, and provider's user accounts. For provider accounts, administrators should be able to view/edit the provider NPI. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 7 |
| Test Name | Verify remember me is removed from login, view/insert API Keys, access QR Code |
| Description | The functional requirements were to verify the “remember me” is removed from the login page and verify that administrators could view, edit, or insert API Keys and access User QR Codes. |
| Picture |  |
| Results | Passed |

### **4.3.2 Participant Test Suite**

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 8 |
| Test Name | TimeOut |
| Description | The functional requirement was for the system to timeout and log a user out when there was 30 minutes of inactivity. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 9 |
| Test Name | Allow user comments over 300 characters |
| Description | The functional requirement was to allow users to type in comments over 300 characters. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 10 |
| Test Name | Valdation of phone number field |
| Description | The functional requirement was to restrict the phone number field to 10 digits. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 11 |
| Test Name | Delete my account |
| Description | The functional requirement was to allow participants to delete their accounts. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 12 |
| Test Name | Hide deceased |
| Description | The functional requirement was to remove the deceased checkbox from the participant account. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 13 |
| Test Name | Invalid Logins |
| Description | The functional requirement was to limit the number of invalid login attempts. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 14 |
| Test Name | User login view other users and edit profile |
| Description | The functional requirement was for a participant that is an authorized user to be able to login to their account and view and edit user accounts they manage. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 15 |
| Test Name | Login, edit/verify the account, profile, allergies, medications, vaccines, conditions, documents, Advance Directives, Authorized Users, Regenerate QR code. |
| Description | The functional requirements were for a participant to login to the VLOL system, access their account information, and save changes to all areas of their profile. |
| Picture |  |
| Results | Passed |

### **4.3.3 Provider Test Suite**

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 16 |
| Test Name | Registration |
| Description | The functional requirement was to have a separate registration page for providers. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 17 |
| Test Name | Remove comments section on view user |
| Description | The functional requirement was to remove the comments section from the participant view for providers. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 18 |
| Test Name | Search for users, view user information, view deceased flag |
| Description | The functional requirements were to allow providers to search for participants, view their profile information to include a deceased flag. |
| Picture |  |
| Results | Passed |

### **4.3.4 Unauthorized User Test Suite**

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 19 |
| Test Name | Contact Admin |
| Description | The functional requirement was for an unauthorized user to have the ability to contact the VLOL system admin. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 20 |
| Test Name | Registration Page |
| Description | The functional requirement was to split the registration page to make it easier for users. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 21 |
| Test Name | Tabbing order of links |
| Description | The functional requirement was to have the tabbing order follow a logical flow for users that are not able to or choose not to use a mouse. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 22 |
| Test Name | Captcha on public-facing pages |
| Description | The functional requirement was to not allow computers to access the VLOL site or submit information via the contact page. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 23 |
| Test Name | Enroll in the LOL program |
| Description | The functional requirement was to allow unauthorized users to enroll in the VLOL system. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 24 |
| Test Name | View page that introduces VLOL |
| Description | The functional requirement was to allow unauthorized users to view information to understand what the VLOL system is. |
| Picture |  |
| Results | Passed |

## 4.4 Manual

Not all the functional requirements were able to be validated with automated tests in the Selenium application. For the tests that could not be executed with Selenium manual test cases were created and executed. Items that were tested with manual testing are:

* Uploading documents
* Viewing uploaded documents
* Scanning QR codes

| **Selenium Final Execution Details** | |
| --- | --- |
| Test Started | 11/02/2020 at 6:00 pm |
| Test End | 11/02//2020 at 6:30 pm |
| Test Conductor | Kimberly Van Allen |
| Number of Test Cases | 3 |
| Number of failures | 0 |

Table 4: Manal Test Final Execution



### **Manual Test Suite**

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 1 |
| Test Name | Uploading Documents |
| Description | The functional requirement was for users to have the ability to upload documents such as Advance Directives or medical POAs and then have the ability to view the documents uploaded. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 2 |
| Test Name | Printing QR Codes |
| Description | The functional requirement was for participants to have the ability to print QR Codes. |
| Picture |  |
| Results | Passed |

| **Test Case Details** | |
| --- | --- |
| Test Case ID | 3 |
| Test Name | Scanning QR Codes |
| Description | The functional requirement was for providers to be able to scan QR codes and view the participant’s information. |
| Picture |  |
| Results | Passed |

### **Testing the QR Code**

The QR Code is a vital part of the VLOL application so ensuring users could print the code as well as scan a printed code was a vital part of the VLOL Team’s testing. To ensure the QR Code was functional the following criteria needed to be verified:

* Printed QR code was as big or bigger than the minimum size of 2 x 2 cm
* QR can be scanned by multiple devices
* A quiet zone four times larger than one module is met on all sides
* Scanning QR Code brings authenticated user to the correct participant profile

Attach is an example of a printed QR code from the VLOL application.



# Defects Identified

The table below lists all defects by severity that were found during testing of the VLOL application.

|  | **Critical** | **Major** | **Medium** | **Low** | **Total** |
| --- | --- | --- | --- | --- | --- |
| **Closed** | 0 | 3 | 3 | 8 | 13 |
| **Opened** | 0 | 0 | 0 | 0 | 1 |
| **Total** | 0 | 3 | 3 | 8 | 14 |

Table 5: Defect Matrix

Figure 2: Defects Identified based on the severity

Table 5 provides the description of each defect identified as well as the current status of the defect.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Severity** | **Description** | **Current Status** |
| 1 | Major | The provider is logged out when attempting to edit a Participant’s profile | Closed |
| 2 | Major | Admin is unable to add accounts | Closed |
| 3 | Major | Manage other user page encounters a server-side error when loading. | Closed |
| 4 | Medium | Admin must set a new user account to locked when creating the account. | Closed |
| 5 | Medium | The user does not receive notification that an email will be sent to them to complete the registration process. | Closed |
| 6 | Medium | The comments section is not saving comments. | Closed |
| 7 | Low | When a user changes their password, they are not required to complete the verify password field. | Closed |
| 8 | Low | Not all blood thinners are being identified. | Closed |
| 9 | Low | Documents are not shown on the view Participant page. | Closed |
| 10 | Low | The user gets an unclear error when entering comments over 300 characters. | Closed |
| 11 | Low | Client-side validation does not work when editing a user profile. | Closed |
| 12 | Low | Return to the menu link on the API key page does not work. | Closed |
| 13 | Low | Return to the menu on the edit Authorized User does not work. | Closed |
| 14 | Low | Password lengths are not enforced when an Admin creates an account for other users. | Closed |

Table 6: Defect Description and Status

# Exit Criteria

* Testing of all items in scope has been completed
* All test cases have been executed
* All major defects have been fixed and retested
* All minor defects have been documented

# Conclusion/Sign Off

| **Name** | **Signature** | **Date** |
| --- | --- | --- |
| Sponsor  Avital Graves |  |  |
| Sponsor  Robert Garcia |  |  |
| Professor  Dr. Mir Assadullah |  |  |
| VLOL Project Manager  Michael Shaw | Michael Shaw | 11/3/20 |
| VLOL Project Manager  Heather Barnes |  |  |

Table 7: Sign Off