**Indiana University Southeast**

**2021-22 CSCI Capstone Project**

**Workshop Management Web Application**

**Sponsor: Dr. Suranga Hettiarachchi**

**Software Test Plan**

**James Schlesener**

1 Introduction 3

2 Business Background 3

3 Test Objectives 3

4 Scope 3

5 Test types Identified 3

6 Problems Perceived 3

7 Architecture 3

8 Environment 3

9 Assumptions 3

10 Functionality 3

11 Security 4

12 Performance 4

13 Usability 5

14 Test Team Organization 6

15 Schedule 6

16 Defects Classification Mechanism 6

17 Configuration Management 6

18 Release Criteria 6

# Introduction

Jasmine is a behavior-driven development framework for test JavaScript code. It will be used to test both the frontend application as well as the backend REST APIs.

# Business Background

The Workshop Management Web Application is designed to track workshops, their participants, and the equipment used within them.

# Test Objectives

The Jasmine tests will ensure that all Angular components are fully functioning along with the server routes and pipe files and that all Express REST APIs are responding correctly.

# Scope

***Inclusions***

Front End: Angular

Back End: Express

***Exclusions***

None

# Test types Identified

Jasmine unit testing and manual end to end testing.

# Problems Perceived

There aren’t any problems perceived at this time. The unit tests along with the end to end tests should ensure that the system is working properly.

# Architecture

The Jasmine tests for the front end will be written inside the Angular project. Each component has a file which contains the unit tests for that component. There is also a file for the server routes and for each of the pipes. These tests can be run using the “ng test” command from the command line. The Jasmine tests for the back end will be stored in a file and executed from the command line. The backend tests will test all the http calls to the database. Manual end to end tests will be performed to confirm that the system is operating as expected.

# Environment

Jasmine unit tests executed from the Windows command line. The server and client will be run locally for the end to end testing.

# Assumptions

If the unit tests all pass and the software operates as expected through the end-to-end tests, then the software is functioning correctly.

# Functionality

***Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Infosys Limitations** |
| Upload Image | User must be able to upload an image equipment, if desired. | Web app uploads an image into a folder on the server. |
| Add Workshop | User must be able to add a new workshop. | Web app adds a new workshop from the View Workshops screen. |
| Edit Workshop | User must be able to edit a workshop. | Web app allows a user to edit the workshop data from the View Workshop screen. |
| Delete Workshop | User must be able to delete a workshop. | Web app allows a user to delete a workshop from the View Workshop screen and all participants assigned to that workshop become unassigned. |
| Add Participant | User must be able to add a participant. | Web app allows a user to add a participant from the View Participants screen and the View Workshop screen. |
| Edit Participant | User must be able to edit a participant. | Web app allows a user to edit a participant from the View Participant screen. |
| Delete Participant | User must be able to delete a participant. | Web app allows a user to delete a participant from the View Participant screen if the user doesn’t have any equipment checked out. |
| Add Equipment | User must be able to add equipment. | Web app allows a user to add equipment from the View Equipment screen and the View an Equipment screen. |
| Edit Equipment | User must be able to edit equipment. | Web app allows a user to edit equipment from the View an Equipment screen. |
| Delete Equipment | User must be able to delete equipment. | Web app allows a user to delete equipment from the View an Equipment screen. |

***Risk Identified & Mitigation Planned***

If Add and Delete parameters are not function, then the software will not meet functional requirements. The Edit and Upload parameters will not keep the software from functioning. The user would just have to delete and add again instead of edit.

***Test Strategy***

Unit tests will be performed to test the components and their functionality.

***Automation Plans***

The unit tests will be coded in Jasmine and be executed after updates to verify that they are functioning properly.

***Deliverables***

The software will be fully functional.

# Security

***Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Infosys Limitations** |
| Registration | User must be able to create an account | Web app creates a new user in MySQL database table. |
| Login | User must be able to login | Web app locates user with valid password and issues a JSON Web Token (JWT) which is required to access the other user screens. |
| Logout | User must be able to logout | Web app clears JWT and goes to the login screen. |

***Risk Identified & Mitigation Planned***

If the user isn’t able to register/login/logout, then the software can still function. There just won’t be any tracking of who made updates to the database.

***Test Strategy***

Unit tests will be performed to test the components and their functionality.

***Automation Plans***

The unit tests will be coded in Jasmine and be executed after updates to verify that they are functioning properly.

***Deliverables***

The user will be able to register, login, and logout and the user will be assigned a JWT so they can access the web application screens.

# Performance

***Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Infosys Limitations** |
| MySQL database | The user needs to be able to store and retrieve data. | MySQL database is available to the server and responds to SQL queries from the web server. |
| Web server | The user needs to be able to retrieve and store data. | The web server is available and responds to MySQL queries from the client. |
| Client | The user needs to be able to use the web application. | The client is available, and the user can access the web application. |

***Risk Identified & Mitigation Planned***

If the database, server, or client are unavailable, the web application will not be functional. Steps will need to be taken to get the unavailable system back up and running.

***Test Strategy***

The unit tests will test both the front end and back-end functionality.

***Automation Plans***

The unit tests will test both the front end and back-end functionality.

***Deliverables***

The unit tests will test both the front end and back-end functionality.

# Usability

***Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Infosys Limitations** |
| Constraint 1 |  |  |
| Constraint 2 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

***Risk Identified & Mitigation Planned***

The unit tests will test both the front end and back-end functionality.

***Test Strategy***

The unit tests will test both the front end and back-end functionality.

***Automation Plans***

The unit tests will test both the front end and back-end functionality.

***Deliverables***

The unit tests will test both the front end and back-end functionality.

***Compatibility***

***Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Infosys Limitations** |
| Constraint 1 |  |  |
| Constraint 2 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

***Risk Identified & Mitigation Planned***

The unit tests will test both the front end and back-end functionality.

***Test Strategy***

The unit tests will test both the front end and back-end functionality.

***Automation Plans***

The unit tests will test both the front end and back-end functionality.

***Deliverables***

The unit tests will test both the front end and back-end functionality.

# Test Team Organization

The test team consists of just one person, James Schlesener.

# Schedule

Unit tests will be run upon any updates made to the components in Angular or the REST APIs in Express to ensure that they pass and are working correctly. At the end of the software development, the manual end to end tests will be performed to ensure the system operates as expected.

# Defects Classification Mechanism

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type of Defects | Functionality | Performance | Security | Usability | Compatibility |
| Critical | Data is not saved in the database |  | User unable to register or login |  | Software doesn’t run on user’s browser |
| Major |  |  |  |  |  |
| Minor |  |  |  |  |  |
| Cosmetics |  |  |  |  |  |

***Defects Logging and Status Changing Mechanism***

The unit tests will test both the front end and back-end functionality.

***Turn Around Time for defect fixes***

The unit tests will test both the front end and back-end functionality.

# Configuration Management

The unit tests will be run whenever new code is added to ensure that it is functioning properly. The end to end tests will be completed once the software has been fully developed.

# Release Criteria

The software can be released if the tests all pass.