Test Plan Document

For

Forty and Eight Bar

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

Test Plan

# Introduction

# Business Background

# Test Objectives

# Scope

***Inclusions***

***Exclusions***

# Test types Identified

# Problems Perceived

# Architecture

With Angular’s modular-component approach to website design, we can easily modularize our unit tests to test each individual page of the website and the components and code they contain. Each webpage component in our project contains a TypeScript file containing the code to be run on that page, with the ‘.ts’ file extension, and paired with each of these is a ‘.spec.ts’ file created automatically in which we can write our unit tests for that component.

# Environment

Unit tests run in Angular website development are performed at the command line in the root folder of the project, using the ‘ng test’ command. This opens a browser window controlled by our testing software called Karma. Karma runs the unit tests on each component and gives us a readout on which tests (called “specs”) passed and which ones failed, as well as parameters telling us why certain tests may have failed. Karma also gives us a readout on how fast the tests were run, giving us insight into potential performance issues. Later on in development, we can also use the ‘ng e2e’ command similarly to run end-to-end tests in order to test the website’s functionality running as a whole.

# Assumptions

Each ‘.spec.ts’ unit test file has a ‘beforeEach’ clause in which we can set up common pre-assumed states of the website to be tested. For example, when testing the Membership Interest page, we should assume that certain required text fields are populated, while certain optional ones are not, before we run our unit tests specific to that page.

# Functionality

***Constraints and Resolutions***

Since our website is somewhat simple, being an advertising website for a Veterans’ bar, we are not likely to have any technological constraints. Our website will not use much memory, either in the online hosting of the website, or the database required for the membership list.

***Risk Identified & Mitigation Planned***

The only risks we foresee are the maintainability of the website after we are done developing it. To help with this, we will provide administrative and back-end tools for the site’s administrators to continue maintaining the website once we finish it.

***Test Strategy***

We plan to use a test-driven-development approach to developing the website. This will enable us to ensure no bugs exist as we program each component, as well as preventing future bugs from happening through component interactions.

# Security

Since our website will be connected to a database containing the members of the 40 & 8 bar’s personal information, the security features of our website will be very important to prevent this data being compromised.

In order to provide our website with some inherent security, we plan to use these best practices in Angular:

1. We will use interpolation to safely encode potentially dangerous characters and escape untrusted HTML or CSS expressions within template expressions. By default in Angular, all data is treated as unsafe so we plan to use libraries that perform output encoding by default.
2. We will not use templates generated by concatenating user input. Instead, we will use string interpolation.
3. We will not use native Document Object Model (DOM) APIs to interact directly with HTML elements, and will instead use Angular template mechanisms and Angular’s own APIs to manipulate DOMs.
4. Finally, we will regularly scan our Angular components for security vulnerabilities. This capability is directly built into the Angular CLI which we are using to develop our project.

# Performance

While our website will be relatively small and simple, we will still focus on making it as performant as possible in case it needs to be expanded at a later date. In order to do this, we will follow these best practices while developing the project:

1. We will focus on making event handlers fast. Because these event handlers are called so often by many different components, ensuring that the events take as little time as possible will ensure that change detection on the website does not take longer than 17ms, which would visibly slow down the loading of the website even on simple executions.
2. In accordance with optimizing event handlers, we will try to minimize change detections as much as possible. By default, components undergo change detection on every user interaction, but we will manually indicate to Angular if a component’s subtree is already up to date and exclude it from change detection where possible.

# Usability

***Constraints and Resolutions***

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

***Risk Identified & Mitigation Planned***

***Test Strategy***

***Automation Plans***

***Deliverables***

***Compatibility Constraints and Resolutions***

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

***Risk Identified & Mitigation Planned***

***Test Strategy***

***Automation Plans***

***Deliverables***

# Test Team Organization

# Schedule

# Defects Classification Mechanism

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type of Defects | Functionality | Performance | Security | Usability | Compatibility |
| Critical |  |  |  |  |  |
| Major |  |  |  |  |  |
| Minor |  |  |  |  |  |
| Cosmetics |  |  |  |  |  |

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

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

# Configuration Management

# Release Criteria