Test Plan Document

For

Forty and Eight Bar

1 Introduction 3

2 Business Background 3

3 Test Objectives 3

4 Scope 4

5 Test types Identified 4

6 Problems Perceived 4

7 Architecture 4

8 Environment 5

9 Assumptions 5

10 Functionality 5

11 Security 6

12 Performance 6

13 Usability 6

14 Test Team Organization 7

15 Schedule 7

16 Defects Classification Mechanism 7

17 Configuration Management 7

18 Release Criteria 7

# Introduction

This documents serves the purpose to show our test cases and what we intend on testing within our program. We will go through front end and back end test cases to make sure that the website/program functions as it should to ensure there will be little to no maintenance to maintain this website. This website serves the purpose to the 40 &8 to inform its members and to also expand the community of the 40&8. The tests we will cover in this document will be related to maintaining this website to stay online and also testing the user experience will using the site. The tests will also make sure navigating the site and entering data into the site will be smooth and will allow the user to have a good experience.

# Business Background

The background of the 40&8 was that this was started as a veteran club but they did not want the club to just be vets they wanted to allow the vets to be able to invite their friends and family. So they started with just vets they invited their friends and family to become members and from their friends and family they were also able to invite other into the club. So this club is not just a veteran bar/club this is an organization that supports veterans and its community. They have a nursing scholarship and they do different charitable activities in the community.

# Test Objectives

1. Will the site work on mobile phones and tablets
2. If the url path the user uses is wrong what screen will come up?
3. The form they will fill out to try and become a member each textbox needs to have parameters to make sure those textboxes are all formatted the same way
4. If the mail server is unable to send the email what will happen to that form?
5. When an admin approves a user we need to check that that new member was added to the database and also if they were declined we need to check and make sure they were not added to the database
6. Along with that we need to make sure when they submit the form the form is being sent in an email to the admin and it is also available for the admin to see when they are logged in
7. We need to test and make sure that the admin site is locked only to the admins login and that no one else can make it in to that part of the application
8. We also need to check and make the sure the database is always live and it never gets disconnected from the website because if someone applies on the site to become a member and the database is offline then we have lost that application
9. We will also make it available that every 7 days the mail server will email the admin saying to update the schedule and we will have a form for them to fill out so that that can be posted under the calendar part of the website so we will need to test that that email comes out every 7 days and that the form is available every 7 days.
10. We will also have to test whether the Facebook links we put in there if they are valid and what happens if they click on the link and in the rarest instant Facebook is offline what will happen.

# Scope

***Inclusions:*** Functionality tests, API tests, Database testing

***Exclusions:*** hardware testing, System test, Agile Testing

# Test types Identified

The testing types we identified as necessary are Functionality tests, API tests, and Database tests. Functional testing and database testing are one in the same so we will perform functional testing to make sure that the database is working how it should. API testing will test that the API’s that we are using are giving us the output that we are looking for. The database testing is a little more complicated we will have to find documentation for that but I am sure it will be fairly simple. For the database we just need to check for new entries and to make sure those entries are accurate from what the user submitted.

# Problems Perceived

I think one of the bigger problems that comes to mind is testing the mail server as a whole. One aspect of that is testing to make sure the form was submitted and the email was sent another aspect of that is making sure that what is in the form is accurate information and not the applicant before that’s information. Another aspect of the mail server concept is when the admin accepts or denies the application the system will either add them to the database or their information will be erased.

# 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

Our plan to go about testing usability, is to allow the rest of the class to use our site. The site will be hosted on a cloud server, being able to be accessed by everyone that knows the URL. Having other people other than us testing this will give us good feedback as to how easy it is to navigate. The customer wants the site to be very simple, considering who this is being built for. There is no plan to automate this type of testing, as this type of testing is difficult to automate.

# Test Team Organization

Other groups with the P446 course.

# Schedule

**Perform Execution of Tests** - <5 mins

**Test Reporting** - <5 mins

# Defects Classification Mechanism

* **Functional Defects** – Experiencing no functional defects currently. The UI responds as it should, data is returning with the correct values.
* **Performance Defects** – Experiencing no performance defects currently. Everything is running and executing at the speed it should. Tests are not taking too long to run.
* **Usability Defects** - We experience a cosmetic defect with usability, since older people will now have to use a newer system to show interest in the club. This will not be a major issue, just may take a very short amount of time for them to figure out how it works. If the UI would need to be structured any different way, turnaround time should be around 1 to 2 days. These are minor things that can be fixed easily.
* **Compatibility Defects** – Currently, we are experiencing some minor compatibility defects. These defects are browser related, as the UI is displayed different depending on your chosen browser. This can be fixed with simple polyfills, turnaround time will be 1 to 2 days to fix this defect.
* **Security Defects** – Experience no security defects currently.

# Configuration Management

Our tests are being configured using Karma and Jasmine. These are what the angular CLI uses to test the UI. Karma is running the tests and Jasmine is the test reporter which is generated through HTML.

# Release Criteria

Tests should execute successfully and test all important features of the application. Also, ensure compatibility amongst all environments.