Test Plan

SERVUS

ChangeLog

Version	Change Date	By	Description
v0.0.0	2021/02/24	Andy	Create an initial test plan.
v0.0.1	2021/02/25	Arvind & Andy	Expanding on test plan

1 Introduction

Brief introduction of the test strategies, process, workflow and methodologies used for the project

1.1 Scope

1.1.1 In-Scope

Features that will be tested in SERVUS:

Creation and deletion of database items;

- Creating, deleting, and customizing services.
- Booking appointments and cancelling existing appointments.

Querying database items given specifications;

- Fetching nearby services.
- Fetching calendar availability for services.

Creation of React components

- Verifying the creation of UI components

Interaction with React components

- Interacting with various UI components and checking the result.

Verify React routing

- Verify the react routing components route to the correct page

Test various Node JS API endpoints

- GET, PUT, POST, DELETE User
- GET, PUT, POST, DELETE Services
- GET, POST, DELETE Appointments

1.1.2 Out-of-Scope

Features that will not be tested in SERVUS:

- Google Maps functionality
- Material UI components

1.2 Quality Objective

- Identify all defects and bugs and fix them before release
- Conform the Application Under Test (AUT) to all specified requirements
- Ensure the AUT meets the quality specifications defined by the client

1.3 Roles and Responsibilities

Name	Net ID	GitHub username	Role
Risto Zimbakov	zimbakor	rikizimbakov	Full Stack Developer
Andy Tan	tana	andy-tan7	Project Manager
Arvind Maan	maana3	arvind-maan	System Architect
Caden Chabot	chabotc	cadenchabot	Full Stack Developer

2 Test Methodology

2.1 Overview

We have set up the framework for **continuous integration** tests as of Sprint 2.
In Sprint 3, we plan to start creating **unit tests** that test the individual components and pieces of software in the frontend and backend.
In Sprint 4, we plan to create **end-to-end testing**.

2.2 Test Levels

We are using continuous integration with the Application Under Test (AUT). We will be creating unit tests in the next sprints.

The goal of our testing is to create a robust and well-tested system.

2.3 Test Completeness

The following criteria will be used to check Test Completeness:

- >85% test coverage
- All Manual & Automated Test cases executed
- All open bugs are fixed or will be fixed before the next release

3 Test Deliverables

- Test Plan
- Test Cases
- Requirement Traceability Matrix
- Bug Reports

4 Resource & Environment Needs

4.1 Testing Tools

- Our frontend and backend are tested with Jest.
- CI is set up with Github Actions and integrated with our git repo.
- Our issues are tracked within Github and project boards are set up for each sprint.
- A test database will be set up to make interactions with the database easier.
- We'd like to display code coverage on our GHA CI using CodeCov, if time permits.

4.2 Test Environment

- Continuous integration tests are done on GitHub when a pull request is made to main and when a push occurs in main. No specific hardware/software is currently required on a developer's machine for tests.

5 Terms/Acronyms

Make a mention of any terms or acronyms used in the project

TERM/ACRONYM	DEFINITION
AUT	Application Under Test
GHA	GitHub Actions
CI	Continuous Integration
UI	User Interface