SE 3XA3: Test Plan
Node Messenger

Team #24, Node Messenger
Tasin Ahmed - ahmedm31
Shardool Patel - pates25
Omar Elemary - elemaryo

October 24, 2018

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

This document ...

1 General Information

1.1 Purpose

This purpose of this document is to outline areas of testing from tools to methodology and plan that will verify both functional and non-functional requirements of the product and ensure that we are delivering a valid product to our clients designed in accordance to the outlined requirements.

1.2 Scope

As this is web-messenger product, testing will be focused on elements that are integral for mobile communication. Test plan shall cover both front-end and back-end implementations of functions to ensure validity. Front-end testing will consist of user input and output correction as well and proper rendering. Back-end testing will ensure that proper input and output collection is correct as well as user authentication, data transfer and stress testing firebase database system. If the product passes all the tests outlined in this document, it will be considered a valid implementation of the requirements outlined in the SRS document.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
Abbreviation1	Definition1
Abbreviation2	Definition2

1.4 Overview of Document

This document will outline a proper test plan for functional and non-functional requirement as well as tests for our current proof of concept demonstration.

Table 3: **Table of Definitions**

Term	**Definition**
Term1	Definition1
Term2	Definition2

The document will also break down the testing process into multiple divided section and elaborating on them in great detail to provide context to our testing methodologies. Finally this implementation of the product will be compared to a successful one based on a list of usability questions and through this an outlined unit test plan will be formed. Any abstract symbols can be explored in the attached appendix for clarification on any unclear document elements.

2 Plan

2.1 Software Description

2.2 Test Team

2.3 Automated Testing Approach

2.4 Testing Tools

We will be utilizing Jest in order to test React our code, and Mocha to test Javascript. This will be done in order to keep bugs to a minimum, and to better the user experience. There will be a sections on our website where users can leave their reviews, and suggestions to improve our product.

2.5 Testing Schedule

See Gantt Chart at the following url ...

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Area of Testing1

Title for Test

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

3.1.2 Area of Testing2

...

3.2 Tests for Nonfunctional Requirements

3.2.1 Area of Testing1

Title for Test

1. test-id1

   Type:

Initial State:

Input/Condition:

Output/Result:

How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

3.2.2 Area of Testing2

...

3.3 Traceability Between Test Cases and Requirements

4 Tests for Proof of Concept

4.1 Area of Testing1

Title for Test

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

4.2 Area of Testing2

...

5 Comparison to Existing Implementation

Node Messenger is based upon Tinode, an existing implantation of a web messaging app. Tinode implements a successful messaging app with the inclusion of 3 main functions: Correct display of messages being sent and received, displays a list of past messages with other users, and stores the messages of a user, allowing them to logout or login with a different account. We will be testing to see if the messages being sent, and the messages being received are formatted properly, and they display on the correct side of the screen. Node Messenger will also be tested to ensure it can create different messaging rooms to let the user chat with other users flawlessly. We will be creating multiple accounts to test if Node Messenger is able to store, access, and show data specific to each user.

6 Unit Testing Plan

We will be using Jest, and Mocha to test the functions governing Node Messenger. We will be ensure that the functions carry out their task properly and return their expected output. Messages should display properly if it is entered correctly, e.g. if a blank message is entered, it must not send an empty message to the message box. Our group will try to cover as much ground as possible in order to ensure a better user experience.

6.1 Unit testing of internal functions

Unit testing using Jest will be mainly used for testing internal functions in Node Messenger. We will be testing the functions above, as well as other key internal functions to ensure Node Messenger functions as expected. Unit testing these internal functions will help us to determine hidden errors, or loopholes in the code that cannot be easily spotted.

6.2 Unit testing of output files

Not Applicable.

References

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

7.2 Usability Survey Questions?

This is a section that would be appropriate for some teams.