SE 3XA3: Test Report
Node Messenger

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

December 3, 2018

Contents

List of Tables

List of Figures

This document ...

Table 1: **Revision History**

Date	**Version**	**Notes**
Date 1	1.0	Notes
Date 2	1.1	Notes

1 Functional Requirements Evaluation

1.1 User Inputs

We tested all the input fields in our messenger app. Through testing, we determined that the login and signup features work as intended. The messenger returns an error if a field is empty while logging in or registering. It returns an error if the email entered is invalid, if the passwords do not match, and if the password is less than 6 characters. While logging in, the console returns errors correctly if the email and password entered is incorrect. While adding people to a new conversation, the messenger ensures that a email is entered and that it is valid. The messenger prevents the user from sending a message if the message box is empty. Pressing 'Enter' or clicking the 'Send' button both send messages correctly. By testing these user inputs we are able to ensure that the core functions of our messenger works as intended.

1.2 Data Storage

Testing was done to ensure that data sent from Node Messenger was being stored correctly in the firebase database. Once an user creates a new account, Node Messenger correctly stores their Name, Email, and Password in the database. After a message a sent by the user, it is sent to our database and stored. The database stores the messages, the time it was sent at, the chat it was sent to, and the user it was sent by. Our database consists of 3 collections: Messages, Users, and Conversations. By testing we made sure that the information being sent to the database gets stored into their respective collection.

1.3 Data Retrieval

Node Messenger is able to look through the firebase database, and retrieve the necessary data. This is done when a new user is signing up to the website. The messenger must look through our database and return an error if the user email already exists in our database. A similar procedure occurs during login. The messenger must take the entered email and password, and allow the user to login if the credentials match that on the database. The messenger is able to display the correct messages to the correct chat, display the name of the user it was sent by, and the time it was sent at. When adding an

user to a conversation, the messenger again looks through our database, and ensures the email address that was entered is valid.

2 Nonfunctional Requirements Evaluation

2.1 Look and Feel

2.2 Usability and Humanity

2.3 Performance

2.4 Operational and Environmental

2.5 Maintainability and Support

2.6 Security

2.7 Legal

3 Comparison to Existing Implementation

This section will not be appropriate for every project.

4 Unit Testing

5 Changes Due to Testing

6 Automated Testing

7 Trace to Requirements

8 Trace to Modules

9 Code Coverage Metrics