

# SE 3XA3: Test Report Question Chat

Team #14, QChat  
Adit Patel - patela14  
Harsh Patel - patelh11  
Vrushesh Patel - patelv12

December 06, 2017

# Contents

<b>1</b>	<b>Functional Requirements Evaluation</b>	<b>1</b>
<b>2</b>	<b>Nonfunctional Requirements Evaluation</b>	<b>2</b>
2.1	Usability and Look and Feel . . . . .	2
2.2	Performance . . . . .	3
2.3	Operational and Portability . . . . .	4
2.4	Maintainability . . . . .	4
2.5	Security . . . . .	4
2.6	Cultural . . . . .	4
2.7	Robustness . . . . .	4
<b>3</b>	<b>Comparison to Existing Implementation</b>	<b>5</b>
<b>4</b>	<b>Unit Testing</b>	<b>5</b>
<b>5</b>	<b>Changes Due to Testing</b>	<b>5</b>
<b>6</b>	<b>Automated Testing</b>	<b>6</b>
<b>7</b>	<b>Trace to Requirements</b>	<b>6</b>
7.1	Test trace to Functional Requirements . . . . .	6
7.2	Test trace to Non-Functional Requirements . . . . .	7
<b>8</b>	<b>Trace to Modules</b>	<b>8</b>
<b>9</b>	<b>Code Coverage Metrics</b>	<b>8</b>

# List of Tables

<b>1</b>	<b>Revision History</b> . . . . .	<b>ii</b>
<b>2</b>	<b>Trace Between Requirements and Tests</b> . . . . .	<b>6</b>
<b>3</b>	<b>Trace Between Requirements and Tests</b> . . . . .	<b>7</b>
<b>4</b>	<b>Trace Between Tests and Modules</b> . . . . .	<b>8</b>

Table 1: **Revision History**

<b>Date</b>	<b>Version</b>	<b>Notes</b>
12/05/2017	0.1	Initial Version
12/06/2017	Rev 1.0	Final Revision

This document provides the results of the tests conducted for QChat. It contains the test cases and the results for the functional and non-functional requirements.

## 1 Functional Requirements Evaluation

The following tests were performed to ensure that the software runs correctly and the user is able to use the software as per the requirements. This portion will define each test and what the domain of the results.

### **Test-F1: Application runs on all devices**

**Results:** The user is able to access QChat ([www.qchat.online](http://www.qchat.online)) on all browsers and all devices that can connect to the internet.

**Input:** Enter URL into web browser.

**Output:** Website loads and home page shown.

### **Test-F2: Connect to class chatroom**

**Results:** The user is able to enter the class chatroom using the given code.

**Input:** User enters given string into textbox, presses submit key to continue.

**Output:** Display changes and now shows the class chatroom.

### **Test-F3: Ask Question/Answer Question**

**Results:** The user is able to submit a question or answer and have it show up in the class chatroom.

**Input:** User enters given string into the ask question or answer textbox, presses submit key to continue.

**Output:** Display changes and now the class chatroom shows the question the user asked or updates the number of solutions for the question answered.

### **Test-F4: Post anonymously**

**Results:** There is no trace between a submission on QChat and the user who posted it.

**Input:** None

**Output:** System does not require any form of authentication or identification to allow a user to post, anybody can visit the site from anywhere and post a comment.

#### Test-F4: Upvote and Downvote Questions

**Results:** The user is able to upvote and down questions and the new vote count reflects this.

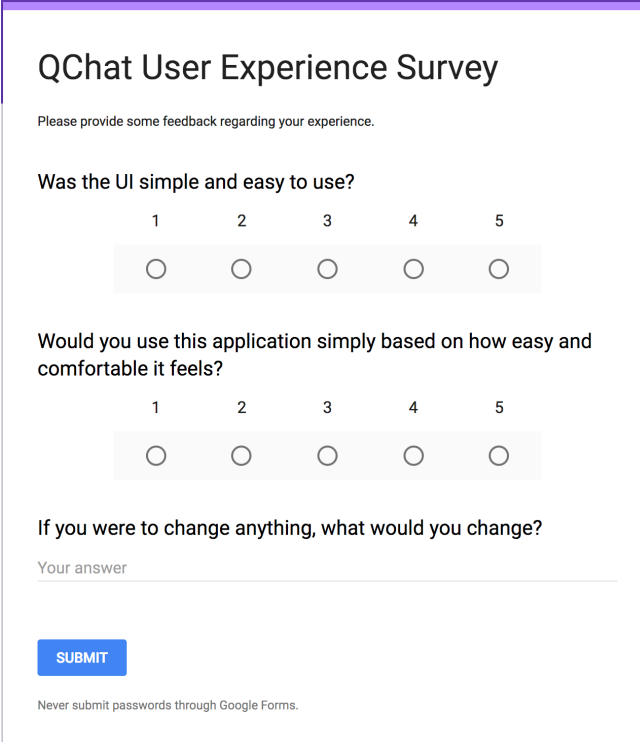
**Input:** User clicks on the upvote button to upvote, user clicks on the downvote button to downvote.

**Output:** The vote count updates and increases by 1 if upvoted or decreases by 1 if downvoted.

## 2 Nonfunctional Requirements Evaluation

### 2.1 Usability and Look and Feel

The user survey conducted:



The image shows a Google Form titled "QChat User Experience Survey". The form is enclosed in a purple border. It contains three questions, each with a 5-point Likert scale. The first question is "Was the UI simple and easy to use?". The second question is "Would you use this application simply based on how easy and comfortable it feels?". The third question is "If you were to change anything, what would you change?". Below the third question is a text input field labeled "Your answer". At the bottom of the form is a blue "SUBMIT" button. A small note at the very bottom says "Never submit passwords through Google Forms."

**QChat User Experience Survey**

Please provide some feedback regarding your experience.

Was the UI simple and easy to use?

1 2 3 4 5

☐ ☐ ☐ ☐ ☐

Would you use this application simply based on how easy and comfortable it feels?

1 2 3 4 5

☐ ☐ ☐ ☐ ☐

If you were to change anything, what would you change?

Your answer

**SUBMIT**

Never submit passwords through Google Forms.

Figure 1: Use hierarchy among modules

**Description of Tests:** Usability and Look and Feel was tested by our colleagues at McMaster University that were from different backgrounds like Engineering, Health Science, etc. to get a better opinion on the technological experience for QChat.

**Test Name:** TestNF-1

**Results:** All the participants rated the look and the feel of QChat. The average was 3.9. The average was below the expected average and some feedback was received that QChat can be made to make it look like a messaging app (Ex. iMessage).

**Test Name:** TestNF-2

**Results:** The average for the responsiveness of QChat was 3.9 The average was below 4. Feedback was received that QChat can have different UI for cell phones devices.

**Test Name:** TestNF-3

**Results:** All the participants were able to use QChat without any prior instruction given. The average for the look and feel test was 4.2. The average rating was above 4 as expected which means that QChat is easy to use.

## 2.2 Performance

**Description of Tests:** The performance of the QChat was measured by timing the actions. For example, accessing the QChat or posting a question or an answer.

**Test Name:** TestNF-4

**Results:** The average time that the QChat take to load was 2.3 seconds. Sometimes it did take longer than 3 seconds, but mainly it was because of slow internet connection.

**Test Name:** TestNF-5

**Results:** The average elapsed time between right after user hits send and text (question or answer) being updated on all the end users connected to same session was 1.6 seconds. The update on end users screen was fast, but it did take longer for sometimes when the internet connection was slow.

## 2.3 Operational and Portability

**Test Name:** TestNF-6

**Results:** QChat was accessible and functional on cell phones, tablets, laptops and desktops with all different OS like android, IOS, Windows, Linux and MacOS.

## 2.4 Maintainability

**Test Name:** TestNF-7

**Results:** The developers were able to communicate, understand and add to the code.

## 2.5 Security

**Test Name:** TestNF-8

**Results:** QChat stores no personal data of users. Therefore, no user data was or can be compromised.

## 2.6 Cultural

**Test Name:** TestNF-9

**Results:** The participants did not find any vulgar text. The average was 1.

## 2.7 Robustness

We didnt explicitly test robustness. Robustness was not a quality that we put within the scope of this project, however it is an important quality and it was kept in mind as we implemented and tested. The main reason robustness was not explicitly tested is because we have put adding profanity filters as future goals and the robustness of how we handle inputs can only truly be tested once these filters are in place. Otherwise our project handles any

input and would indicate it is robust without even having restrictions.

### 3 Comparison to Existing Implementation

This project did not have an open source existing implementation. We were however able to compare to [sli.do](#). This product is a private audience interaction application which in a sense is a much more scaled product than our project. Our project does provide the same functionality at a lower scope; our project is to be used for lectures and providing a means for communication amongst students as well as with the professor.

### 4 Unit Testing

Completed unit testing using jasmine.js. Our group ran multiple tests that would test most of the methods defined in our JavaScript file.

Our unit testing covered a subset of what our manual testing covered to ensure we could verify the expected results. This lead to us unit testing each of our methods. There were some situations where we saw that unit testing certain portions of our code in smaller sections would be better. For example, testing that the integration of Firebase works was broken down and required us to manually check on Firebase console as well.

Our unit testing verified the results that are mentioned in Functional Requirement Evaluation Section. To be specific, the tests were conducted for Test-F1 to Test-F4 only. Test-F5 was merely a visual change and could not be unit tested.

### 5 Changes Due to Testing

After conducting usability testing surveys we realized that QChats initial user interface was not an intuitive to follow as we thought. Using the feedback we collected we proceeded to update the interface in a manner that everyone would understand, using the scroll view. The scroll view allowed for all our question to be organized top to bottom in a certain ordering. We also



went on to hide the answers to each question instead of showing them right away as this saved space and made the class chatroom look clearer and more organized. To view answer we added a button called show answer to each question and once a user was done looking at the answer the answer could be hidden again.

## 6 Automated Testing

As mentioned in our test plan, the unit testing is one of the only ways we could automate our testing. We were able to carry out these tests and successfully pass them.

In our test plan, we mentioned that we could look into writing a script that accesses the Firebase database and reads/writes to verify that it up and working automatically. We realized that did not fit the scope of our project and decided we would allocate our resources better.

## 7 Trace to Requirements

### 7.1 Test trace to Functional Requirements

**NOTE:** Refer to SRS to see the Requirements

Test	Requirements
Test-F1:	Req-F2
Test-F2:	Req-F3
Test-F3:	Req-F1
Test-F4:	Req-F1
Test-F5:	Req-F4

Table 2: Trace Between Requirements and Tests

## 7.2 Test trace to Non-Functional Requirements

**NOTE:** Refer to SRS to see the Requirements

Test	Requirements
TestNF-1:	Usability and Look and Feel
TestNF-2:	Usability and Look and Feel
TestNF-3:	Usability and Look and Feel
TestNF-4:	Performance
TestNF-5:	Performance
TestNF-6:	Operational and Portability
TestNF-7:	Maintainability
TestNF-8:	Security
TestNF-9:	Cultural

Table 3: Trace Between Requirements and Tests

## 8 Trace to Modules

Test	Modules
Test-F1:	M1, M2, M3, M4, M5, M6
Test-F2:	M3, M4, M5, M6
Test-F3:	M2, M4, M5, M6
Test-F4:	M2, M4, M5, M6
Test-F5:	M2, M4, M5, M6
TestNF-1:	M4
TestNF-2:	M2, M3, M5
TestNF-3:	M4
TestNF-4:	M5
TestNF-5:	M6
TestNF-6:	M1, M2, M3, M4, M5, M6
TestNF-7:	M2, M3, M4, M5, M6
TestNF-8:	M3, M5
TestNF-9:	M6

Table 4: Trace Between Tests and Modules

## 9 Code Coverage Metrics

For Qchat, the code covered in the testing is approximately 85%. This number is based off the unit testing covering the functionality as well as manual testing which confirms the unit testing coverage as well as accesses additional code on top of it. This is evident in our trace to modules section. Along with this, we used JSLint to ensure that our all of our code followed syntax and coding guidelines for JavaScript.