SE 3XA3: Test Plan
Title of Project

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

October 27, 2017

Contents

1 General Information **1**
- 1.1 Purpose . . . 1
- 1.2 Scope . . . 1
- 1.3 Acronyms, Abbreviations, and Symbols . . . 1
- 1.4 Overview of Document . . . 1

2 Plan **2**
- 2.1 Software Description . . . 2
- 2.2 Test Team . . . 2
- 2.3 Testing Tools . . . 2
- 2.4 Automated Testing . . . 2
- 2.5 Testing Schedule . . . 3

3 System Test Description **3**
- 3.1 Tests for Functional Requirements . . . 3
  - 3.1.1 Application running on all devices . . . 3
  - 3.1.2 Connect to chat rooms . . . 3
  - 3.1.3 Post Questions and Answers to chat room anonymously 4
  - 3.1.4 Upvote/Downvote Questions . . . 5
- 3.2 Tests for Nonfunctional Requirements . . . 5
  - 3.2.1 Look and Feel Requirement . . . 5
  - 3.2.2 Usability and Humanity Requirement . . . 6
  - 3.2.3 Performance Requirement . . . 7
  - 3.2.4 Operational and Environmental Requirements . . . 7
  - 3.2.5 Maintainability and Support Requirements . . . 8
  - 3.2.6 Security Requirements . . . 8
  - 3.2.7 Cultural Requirements . . . 9
- 3.3 Traceability Between Test Cases and Requirements . . . 9

4 Tests for Proof of Concept **9**
- 4.1 User can input text . . . 9
- 4.2 Data stored in Firebase Database . . . 10

5 Comparison to Existing Implementation **10**

6 Unit Testing Plan **10**
6.1 Unit testing of internal functions 10
6.2 Unit testing of output files 11

7 Appendix **12**
7.1 Symbolic Parameters 12
7.2 Usability Survey Questions? 12

List of Tables

1 **Revision History** ii
2 **Table of Abbreviations** 1

List of Figures

Table 1: **Revision History**

Date	Version	Notes
27/10/2017	1.0	Inital Version
23/11/2017	2.0	Updated portions of output testing plan
25/11/2017	3.0	Updated Acronyms, Abbrevations, and Symbols and Upvote/Downvote Questions

1 General Information

1.1 Purpose

The purpose of this document is to layout Qchats Testing Plan. This document details tools being used such as the unit testing framework, types of tests, how testing will be performed and plans for incorporating multiple different types of tests.

1.2 Scope

Scope of our testing plan will be to check correctness and functionality of our QChat project, it is known that nothing can be fully tested for correctness and so we will need to test robustness and reliability as well. This will ensure that the product is functioning as expected and allows thorough testing before being released.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
QChat	Our software name of the project we are building
App/Application	This term refers to the software project the team is building
Jasmine.js	Testing framework to test JavaScript Code
JSLint	Static code analysis too for JavaScript

1.4 Overview of Document

The QChat project will implement an anonymous Q and A service. The software will allow users to ask and answers questions in class and enhance the student experience. This document will outline how group 14 plan on testing our software and tools we will use.

Our software will have a front-end user interface that handles all interactions with the user. Along with the front-end component, we also will have a back-end component which is the database that stores all the data being fed into the system.

To test these components we will require the use of Jasmine.js - a unit testing framework for JavaScript. JSLint is a static code analysis tool which is used for checking if JavaScript source code has proper syntax.

2 Plan

2.1 Software Description

The QChat project will implement an anonymous Q and A service. The software will allow users to ask and answers questions in class and enhance the student experience. Along with asking and answering, users will be able to upvote and downvote questions to indicate a preference to which one they want answered.

2.2 Test Team

The test team will include all the members of group 14 and a few colleagues from 3XA3 lab section 1 to ensure that the testing is verifiable. By allowing multiple people to conduct tests, we can verify that the results were not just specific cases for which the system works.

2.3 Testing Tools

We will be using Jasmine.js for unit testing and JSLint for static code analysis.

2.4 Automated Testing

For our project, the code coverage for unit-testing is about 80 % which makes it a candidate for automated testing of the units. As far as other automation's go, we are currently in the researching phase, but one possibility we

can attempt is writing a script that stores fresh new data in the database, then queries for it and then deletes it on a daily basis. This would help ensure that the back-end is up and running. Automation testing for a system that is heavily based on user interaction is quite hard, as HTML and CSS are not behavioral, only the underlying JavaScript is.

Another possibility is UI-Regression testing but that requires learning and time efforts that do not prove to be cost-effective for this project. This is definitely a future possibility when the project is scaled and potentially published.

2.5 Testing Schedule

Please click here to open GanttChart.

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Application running on all devices

1. Test-F1

   Type: Functional, Dynamic, Manual

   Initial State: application is available to all users

   Input: application will be accessed on a web browser

   Output: application loads on the web browser

   How test will be performed: the application will be accessed using the link on all different devices and browsers. The expected output will be tested against the actual output. It is assumed that all the devices will have internet access.

3.1.2 Connect to chat rooms

1. Test-F2

Type: Functional, Dynamic, Manual

Initial State: Home page

Input: Chat room id

Output: Inside the chat room, where all the questions and answers are being asked

How test will be performed: multiple chat rooms will be created each with unique id. Different users from the test group will try to join chat room, they should be able to see question and answers in that specific chat room once they enter the chat id and join. The expected output will be compared against the actual output.

2. Test-F3

Type: Functional, Dynamic, Manual

Initial State: Home page

Input: Chat room id that does not exists

Output: On home page where chat id is entered to enter a chat room

How the test will be Performed: chat room id that does not exists will be entered. The expected output will be compared with the actual output.

3.1.3 Post Questions and Answers to chat room anonymously

1. Test-F4

Type: Functional, Dynamic, Manual

Initial State: Inside chat room

Input: User posts a question through web app on one specific chat room

Output: All the users connected to that chat room should be able to see that question

How test will be performed: Have multiple devices and connect multiple users. Ask all user to connect to same chat room. Then one of the user will ask a question, that question should appear on every users device.

If the expected output does not match the actual output then the test case will be considered failed.

3.1.4 Upvote/Downvote Questions

1. Test-F5

   Type: Functional, Dynamic, Manual

   Initial State: Inside chat room

   Input: User upvotes or downvotes a question

   Output: Number of vote on that question should go up by 1 for each user that upvotes a question else it goes down if the user downvotes.

   How the test will be Performed: 5 users will upvote a question in a chat room, number of votes on that question should go up on all the users devices that are in the same chat room to 5. The expected output (of 5 votes on that question) will be compared against the actual output. After testing upvote, downvote will be tested, the number of votes should go down to 0 if 5 users downvotes.

3.2 Tests for Nonfunctional Requirements

3.2.1 Look and Feel Requirement

1. TestNF-1

   Type: Structural, Static, Manual

   Initial State: Main Page

   Input/Condition: Users in the test group are asked to rate the application on a scale of 0 to 10 where 0 being poor, 6 being above average and 10 being excellent by taking a look at QChats UI

   Output/Result: the average rating from the test group and their reviews/notes

   How test will be performed: Users in the test group will rate the QChats UI on scale of 0 to 10 where 0 being poor, 6 being above average and 10 being excellent. There reivews (notes) about the QChats UI will also

be recorded. The average of their rating and the similarities in their reviews will be taken to improvise the UI. The average must be 8 or above.

2. TestNF-2

   Type: Structural, Static, Manual

   Initial State: application is available to all users

   Input: application will be accessed on a web browser and the browser window will be resized

   Output: application loads on the web browser and on resizing the browser window the application should resize itself till it hits its minimum size. the average rating from the test group

   How test will be performed: users in the test group will access the application on different devices with different sizes and will try to resize the browser window. The users will then rate the applications responsiveness based on a scale of 0 to 10 where 0 is poor and 10 is excellent. The average must be 9 or above.

3.2.2 Usability and Humanity Requirement

1. TestNF-3

   Type: Structural, Static, Manual

   Initial State: The look and feel of the application is already tested

   Input: Users in the test group are asked to rate the application based on the criteria of ease of access, ease of use, understandability of texts and symbols and overall satisfaction on a scale of 0 to 10

   Output: the average rating from the test group on each category and their review (note)

   How the test will be Performed: Each user in the test group will be provided with the questionnaire that provides the list of criteria:- ease of access, ease of use, understandability of texts and symbols and overall satisfaction, each with a scale that goes from 0 to 10 where 0 is poor, 6 is above average and 10 is excellent. The average must be 8 or above.

3.2.3 Performance Requirement

1. TestNF-4

   Type: Structural, Dynamic, Manual

   Initial State: Web browser is open and the link is entered

   Input/Condition: Users in the test group will hit enter (or click search)

   Output/Result: the application should load

   How test will be performed: the application will be tested on the web browser 10 times. The time elapsed between right after the user has hit enter and the application has loaded completely will be recorded. The average of those 10 recorded times will be taken. The average must be below 3 seconds.

2. TestNF-5

   Type: Structural, Dynamic, Manual

   Initial State: the application has loaded

   Input: user inputs text in the input field and hits send

   Output: all the users in the test group connected to the same session should be able to see the text entered by one of the user

   How the test will be Performed: the user in the test group will enter the text in the input field on the application. The time elapsed between right after the user hits send and that text being updated on all the users screen that are connected to the same session will be recorded. Same thing will be done 10 times and the average of those 10 recorded times will be taken. The average must be below 2 seconds.

3.2.4 Operational and Environmental Requirements

1. TestNF-6

   Type: Structural, Static, Manual

   Initial State: application is available to all users of the test group

Input: users of the test group will access the application on devices with web browsers

Output: the application will work on all the devices with web browser, meaning it will load and will be usable How the test will be Performed: Each user in the test group will access the application on different devices like cellphones, tablets, laptops, desktops etc. with all different OS like android, IOS, Windows, Linux etc.

3.2.5 Maintainability and Support Requirements

1. TestNF-7

   Type: Structural, Static, Manual

   Initial State: Application is available to all users

   Input: Inspecting code and walking through code to make it efficient or to add features

   Output: N/A

   How the test will be Performed: the testers (team 14) will test to make the code more efficient and to maintain structural integrity of the backend code

3.2.6 Security Requirements

1. TestNF-7

   Type: Structural, Static, Manual

   Initial State: Application is available to all users

   Input: Inspecting code and walking through code to make it efficient or to add features

   Output: N/A

   How the test will be Performed: the testers (team 14) will test to make the code more efficient and to maintain structural integrity of the backend code.

3.2.7 Cultural Requirements

1. TestNF-9

   Type: Structural, Static, Manual

   Initial State: Usability and Humanity Requirements are tested

   Input: users of the test group will use the application

   Output: the average rating from the test group and their reviews/notes

   How the test will be Performed: Each user in the test group will be provided with the questionnaire that provides the list of criteria:- Texts and symbols, and each with a scale that goes from 1 to 5 where 1 is not offensive and 5 being extremely offensive. Their reviews will also be recorded which will help improvise the text and symbols on our application. The average must be 1.

3.3 Traceability Between Test Cases and Requirements

It is really important to be able to trace your test cases with your requirements to ensure your requirements are completely met. Traceability must be maintained across the lifecycle, instead of a traceability matrix, in each of our test cases we have defined which requirement and in which domain of use case it will be tested in. This will allow us to trace our test cases with the requirements and the use case for those requirements.

4 Tests for Proof of Concept

4.1 User can input text

1. TestPOC-1

   Type: Functional, Static, Manual

   Initial State: Main page

   Input: String that lets you enter the chat room

   Output: If successful, the page will go to a new screen which represents the chat room they desired to enter

How the test will be Performed: Test user will be asked to input a specific text on screen, as the user types the string, if the letters appear on the screen we know that test passes.

4.2 Data stored in Firebase Database

1. TestPOC-2

   Type: Functional, Static, Manual

   Initial State: Main page

   Input: User inputs a question

   Output: Question is stored in database

   How the test will be Performed: Once the user enters a question (data), it will be stored in Firebase Database and to test if the question (data) was stored we will check Firebase console where the stored data can be seen.

5 Comparison to Existing Implementation

There are may features of our implementation that will overlap current existing implementations. Posting a question, upvote feature, answering or commenting on a current post will all be similar to the implementation done by the anonymous social media app YikYak. The key difference between the two implementation is QChat is aimed towards being a real time question and answers forum. Second major difference is that instead one massive anonymous chat room for everyone like in YikYak, QChat will have multiple smaller chatrooms the people will enter using specific codes. Each chat room will represent a different group such as a club or a class room.

6 Unit Testing Plan

6.1 Unit testing of internal functions

We will be using Jasmine.js for unit testing as it is a JavaScript Unit Testing Framework.

Unit testing for internal functions will include testing for correctness of results by means of assertions on the actions that the javascript methods may make. To elaborate, we will provide it simulated dependencies and test if the inputs provide the expected output. Jasmine.js also allows testing synchronous and asynchronous JavaScript code which can be advantage for us as we scale the project to a multi-user program.

Code coverage is expected to be 80% of the JavaScript source code our project will have. The 20% that have been excluded will cover variables and/or methods that only concern the front-end of the program. For example, methods such as show() or hide() which will hide a specific div in the HTML file, does not require unit testing. Simple manual, dynamic testing will suffice.

At this point in time, the discussion and analysis required to determine if the project will require stubs and drivers is undecided. A meeting with the stakeholders will help us determine the proper usage and testing capabilities of it.

6.2 Unit testing of output files

Our project will not have any output files and therefore will not require unit testing. The output will be in the format of UI for our website web app.

7 Appendix

7.1 Symbolic Parameters

We have not used any symbolic parameters in this document.

7.2 Usability Survey Questions?

1. Is the user interface understandable and easy to follow? Rate from 1 to 10 where 1 mean not easy to follow to 10 being very easy to follow.

2. Were you able to complete the desired tasks quickly using QChat?

3. Did you run into any unexpected things during your process of using QChat?

4. Would you recommend QChat to a friend?

5. If you could change one thing about QChat what would it be and why?

6. What features could you not live without?

7. What features do you think are not necessary?

8. What do you like best about QChat?

9. How can we improve QChat? Share your ideas and suggestions?