**What Do U Want Testing Document**

**Contributors: Onyekachi Onyenokwe, Wyatt Doiron-Larue, John Melloni, Ayodeji Marquis**

**1.0 INTRODUCTION & OBJECTIVES**

**1.1 Introduction**

“whatdouwant” is a chatting application that allows users to join a common network space and poll questions within the chat allowing for a random selection of outcomes. The application is web based and runs on a MySQL Database.

**1.2 Objectives**

Basically, the plan is to write test code that runs our PHP commands to and from the server observes the resulting output and compare it to a pre-specified output, shown in the test code.

This requires the use of Travis-CI. We used Travis to simultaneously run any test code in all directories of your GitHub repository.

**2.0 SCOPE**

Our website has several capabilities which include:

·         Sign Up/log-in

·         Adding a choice

·         Allowing users to create a room, and send out the room with a keyword.

·         Allowing users to generate a prompt regarding a decision to be made

·         Allowing users to respond to a prompt, and display their results in a pie chart.

·         Allowing users to change their vote via clicking on the pie chart.

·         Allowing any user to initialize the randomization.

·         Allowing users to create a new prompt following a resolved prompt.

·         Storing user information.

·         Storing room information.

**Use Case Class: Room Creation**

1.     User clicks a button to create/join a room.

2.     User enters room name and password

3.     Room consists of an interface with 3 panels; one is for a chat board, one is for a pie chart, and one is for a list of rooms the user is in.

4. User tells others the room name and password, others can use these to join the room

**Use Case Class: User Login**

1.     Prompts user for username and password

2.     Allows user to join/create a room

3.     Shows rooms the user is in

4.     Other users will see and recognize each other by their usernames in the chat board.

5.     Usernames will be used across all rooms the user is a part of

6.     If a user types in a username that is already being used, asks user to choose a different username.

**Use Case Class: Communication**

1.     The chat board panel shows all previous messages (can scroll up and down to view recent or older messages) with the username of the message sender next to the message.

2.     Includes a small section at the bottom that allows a user to input and send a message. This message will show up on the chat board.

**Use Case Class: Prompt**

1.     At any time, any user can create a prompt that all users in the room can see

2.     Users can write-in their own solutions to the query or they can select from pre-existing solutions other users wrote in.

3.     A pie chart representing the different solutions and how many users voted for each one will be displayed in the room.

4.     One vote per user. A user can change their vote by clicking on the part of the pie chart that represents their desired solution.

5.     Every time a vote is added or changed, an updated pie chart will be created and displayed after a short delay.

6.     Everyone has the option to end voting polls and to spin a randomizer that selects a solution at random based on weights that are based on the number of votes a solution has.

7.     Show the result of the spin to everyone in the room

**Security:**

The login and chatting system will have a secure database because people’s privacy is very important to us. The system will be set up to ensure that messages are sent over secure servers, preventing hackers and phishers from gaining access to passwords and messages.

**This will be measured using the following penetration assessments such:**

**·** Authentication assessment (AUTH)

·         Cryptography use assessment

·         Data Validation using user entry management assessment

**Reliability:**

· Chat History Server/Database: A reliable chat history database is important to allow users see past conversations up to a period of at least 2 months.

· Crash Free: A crash free system will allow the users have a smooth experience. What Do U Want is trying to sell based on its simplicity, so having a crash free system will increase our success rate. Unit test cases will be developed to catch bugs.

**DBMS (Database Management):**

We need to store information such as name and password. To store this user information, a database was created to house all of it. We considered using Microsoft Access but decided to go with MySQL because of its portability between operating systems. MySQL is also free and great to learn about DBMS with.

**3.0 TESTING STRATEGY**

Below we discuss various testing strategies we implemented. Our code mainly uses php to get or post files onto a database (I.e. user and client side actions/procedures.)

**3.1 Unit Testing**

**Definition:**

We have chosen a unit testing approach. This means that we will test smaller individual functions of our code and make sure that they produce the required results.

**Methodology:**

Our code mainly uses php to get or post files onto a database. Therefor we set up a Travis account to be linked to our GitHub account. We have chosen a unit testing approach. This means that we will test smaller individual functions of our code and make sure that they produce the required results.

**3.2 System and Integration Testing**

**Definition:**

We needed to make sure that our application worked correctly most times. This is where Integration testing comes into play.

**Methodology:**

Once the different bits of code have been unit tested, we then ran the website to see if it runs as required. And because we used Travis we should have been able to observe that.

**3.3 Performance and Stress Testing**

**Definition:**

We need to work on stress testing our application because our Demo showed certain flaws the app has. We did not initially perform such a test.

**Methodology:**

The best way we can solve this is to allow multiple users to log on and then to observe what crashes the system.

**4.0 HARDWARE REQUIREMENTS**

Computers

Modems (WiFi Connection)

**5.0 ENVIRONMENT REQUIREMENTS**

To set up our testing, we had to use:

·         Travis as well as an

·         Ubuntu Linux environment

·         Simple text editor

·         PHP Environment

·         Database Environment

·         Google API

**6.0 TEST SCHEDULE**

The test ran whenever a build was made. That means that whenever one of us made a commit to the GitHub repository that person would receive an email of the build status (I.e. if it failed or not.)