SWE 3313

Test Document

Flock

Group 11

Ben Pallotti, Emmanuel Okafor, Victoria Miller, Alex Nguyen, Armando Ortiz

Team Members: for each member indicate

All members participated equally

a: did his/her share

b: did more than his/her share

c: did less than his/her share (explain if necessary)

d: did nothing

SWE 3313

Version: (n)

Date: 11/13/2022

TABLE OF CONTENTS

1. Testing Strategy

1.1 Overall strategy

Will will start our testing with Unit Testing. For this process, we will perform Static Testing. All code will be cross-reviewed after being written to ensure minimal rewriting and that all program requirements and design are met. This testing will be performed by the programmers.

Next, we will continue with Integration testing, at which time we will perform Specification Testing. Each class will have an individual test case to ensure code will not break. This testing will be performed by the testers.

Third, we will perform System Testing, for which we will utilize Component Testing. Each component of code will be tested, then linked, and retested, until all components are linked, debugged, and the system runs successfully. This testing will be performed by the programmers.

Fourth, we will perform Regression Testing, where all previously created test cases will be rerun to ensure that any coding changes from previous testing instances will allow the code to work correctly. This testing will be performed by the testers.

Finally, we will perform an Alpha test. This will comprise of User-Interface Testing as well as Acceptance testing. This testing will be performed by the Users.

1.2 Test Selection

We plan on using black box techniques and testing since it allows us to see how the application is working without having to see the internal code structure. We would like to use non functional testing and regression testing. Non functional testing allows us to test the performance and usability. Regression testing is also another black box technique we plan on using. This test will allow us to check if any new code that we add to the program works and if it will not affect the existing code. These techniques will help our application be usable and will allow us to see where we can improve our code in order for it to work efficiently.

1.3 Adequacy Criterion

We will assess the quality of our test cases by utilizing code coverage to ensure that all code is tested. We will complete this by utilizing a tool which works with JUnit to highlight code that has not been tested. Then, if needed, test cases will be rewritten or new test cases will be added to certify that all code will be tested.

1.4 Bug Tracking

Bugs are located through the debugging process. This process entails locating errors in the code which can come from testing, regular use, or other means. There are 4 phases, not including locating the bug, that are involved in debugging. The first is stabilization which is reproducing the error by finding the conditions that are involved in the mentioned error. The second, localization is finding the section of code that leads to this error. Next is Correction which is changing the code to fix the error. Last is verification. This involves making sure the targeted error is fixed.

Enhancement requests, assuming they are small changes, are typically included in the customer's contract. A small change would include something such as including the availability of a new device. Therefore, these enhancements are something to be done upon request.

1.5 Technology and Tools

If we're going to use an automated testing tool, it most likely would be JUnit, because Flock will be utilizing Java. JUnit is a unit testing framework found in Java. Once a class inherits from junit.framework.TestCase, JUnit can be used to test methods that start with the word "test". If the test is successful, its user interface will display a green bar. However, if any run test is unsuccessful it will display a red bar. The skeleton to create a main method that runs all tests is as follows:

```
import junit.framework.TestCase;

public class TestTriangle extends TestCase {

public static void main(String args[]) {

junit.swingui.TestRunner.run(TestTriangle.class);

 }

}
```

2. Test Cases and Test results

Purpose	*Steps*	*E.Result*	*A.Result*	*Pass /Fail*	*Additional*
Flock can run on latest Android mobile device and its two previous generations	Open Flock App on Samsung S22, S21, and S20	Flock does not crash when opened	Flock opens and goes to login screen	*Pass*	
Flock must be able to track user's likes, tweets, retweets, and follows	Give Flock access to Twitter account then check database	Flock will store the Twitter information inside the database	Database tables Twit_User, Twit_Follows, Twit_Retweets, and Twit_Likes updated with relevant information	Pass	
Flock must be able to search for Twitter users with similar likes, tweets (keywords), follows, or retweets	Use the search function and select designated search option (love interest, friends, etc.) then allow Flock to search for match	Flock finds a match for the user using the selected search option OR Matches person with no one (if not enough Twit activity)	Flock matches the user with a Twitter user that has the same likes, follows, retweets, tweets, and likes or retweets	Pass (?)	Is not able to necessarily find users with "similar" similarities just yet, only exact
Flock must be able to customize its searches by adding multiple search criteria, or removing them	Use search and configure its settings by adding or removing options such as searching by likes, tweets, retweets, and follows	Flock should be able to match users with only one or multiple search options	Flock has no problem searching for matches with only one criteria or multiple	Pass	Searching by Tweet is not an option selected by default as opposed to likes, follows, & retweets
Flock must be able to prioritize matches with other Flock users	Create two Twitter accounts near identical in Follows, Likes, and Retweets. Then create a Twitter account that is less similar to the two. Make the less similar	The Twitter user with the Flock account will get picked rather than the Twitter account with more matches	The Twitter account that had less similarities but had a Flock account was prioritized rather than the Twitter	Pass	Will display multiple matches, but the one with the Flock account will show up first

	Twitter account into a Flock account as well as one of the identical ones. Put the identical Twitter account you chose through the match process		account with more similarities		
Flock must be able to generate a chatroom for matches	Match with a user that also has a Flock account then go to matches and "Chat Room"	An interface that allows the user to chat with the match they made will pop up	A chatroom is created with the person the user matched	Pass	