TESTING STRATEGIES DOCUMENT

Team Members: Izzy Benavente - Albert Yue - Sydney Liu - Rebecca Wu – Mingzhe (Elliscope) Fang

Stress Testing

One of the biggest complaints (and most frustrating experiences for users) is when an app crashes on a user. We will be stress testing to ensure that regardless of what device, you will have a great experience on the app.

For stress testing, we will use Exerciser Monkey (http://developer.android.com/tools/help/monkey.html), which generates random touches, clicks, and more. This will help us recognize when the app freezes, receives any un-handled exceptions, and if the app crashes with the intense and random usage from Exerciser Monkey. It will also try many other cases such as turning off wifi, cell service, muting, and more to ensure that that all inputs are tested. The program will run thousands of outputs quickly. There is a really nice tutorial about how to use it here: http://adventuresinqa.com/2014/04/01/how-to-stress-test-your-android-app-with-monkey/

Monkey is able to put the computational load necessary to test the app.

We may consider options of load testing as well as a side-test of the stress testing. While we do not anticipate loads to be a huge issue, we may take a look at this. Load testing seems to be primarily done with other services that connect through your app and emulate heavy loads and connections while tracking usability.

Regression Testing

As we code and make changes, we will keep track of any major bugs and create fixes for them. When large changes are made to the program, we will test to ensure that the documented major bugs do not reappear in our newer versions. If for any reason we need to switch to a newer version of Android, we will perform configuration testing to see if our program still works on the targeted devices that run our targeted version of Android and newer. We will perform this test by running functionality tests on different, emulated versions of Android devices and record the results, fixing any unexpected bugs.

To ensure optimal performance of our app, we will monitor the execution time of certain functions such as Food Feed population and searching and test them at each major update to the app to maintain a certain level of quality and speed. As mentioned in the Stress Test section, we will use Exerciser Monkey to see how the app handles random input. We will also perform this test as we update the app to check for any new bugs.

Black Box Testing

We will assign testers to parts of the program which other members have coded in order to avoid any testing bias. Because the programmers of a certain section will expect it to perform the functionality they coded it for, we need to test code with which we are unfamiliar. Our team will perform black box testing and record any complications or bugs discovered along the way. We will then pass our findings to the person that coded that portion so the bug can be fixed.

We will be using black box testing to ensure we are meeting the project requirements our team has put together. This will help us validate that we are creating the program with the functionality we planned.

White Box Testing

The benefit of white box testing is that the tester will have knowledge of the code as they run the tests. We will have the coders of each section run white-box tests on their own code. Because they know what to expect from the results, they will know what tests to run against their code's functionalities.

White box testing will help us verify that the program performs its functionalities correctly and produces the proper, expected output.

Planned Unit Tests

At the moment, we do not have a functional application that we can test. As a result, we will be creating test cases and expected results, but we will add the actual results once we have a working application. We will follow the the four-item test case template (ID, Description, Expected Results, Actual Results) when planning your test cases. However,we will add Coder, Tester, and Actual Results columns when we implement the actual tests. This will

allow us to effectively test the functionality of our code and ensure that we achieve the results that we want.

Test ID	Description	Expected Results
1	User inputs a valid email address and password when creating account.	User account creation is successful and he or she moves on to the login screen.
2	Upon startup, (guest) user is taken to the feed showing most popular Noms. User then clicks on a registered user button, such as Nom, upvote, or toolbar item.	User is taken to Sign Up page.
3	User makes a login attempt with an invalid email address or password.	App should notify user that the email address was not valid.
4	User uploads non-image file to Nom.	Error toast message pops up explaining the file was unable to be uploaded.
5	User makes a successful login attempt.	User should be taken to the Food Feed page.
6	User searches for a food dish.	The app should display relevant noms that match the keywords used in the search. If no such noms exist, prompt the user to try different keywords.
7	Successfully logged-in user upvotes a Nom.	A notification is sent to the Nom's owner (in the app and on the phone's notification bar if allowed), the upvote button on the user's screen turns red, and the upvote count increases by one.
8	User comments on a nom.	The page should refresh to show the new

		comment as the most recently made comment on the nom. It should should the name of the commenter and the time posted.
9	User chooses one of the pre-made categories: Vegan, Vegetarian, Breakfast, Lunch, Dinner	The app should take the user to a page that has the most popular dishes of that category.
10	User tries to change password.	Text boxes prompt for old password, new password, and new password again. If the new passwords don't match, the text boxes are cleared, and a toast message appears saying the new passwords do not match. If the new password is the same as the old password, the new password is cleared from the two boxes, and a toast message appears saying the user must enter a different password.
11	User B comments on a User A's nom.	User A should see a notification indicator on the bottom toolbar.
12	User has a notification indicator and presses the notification button on the toolbar.	The notification indicator on the toolbar should disappear and the notifications page should show the user the most recent notifications they've received.
13	User B comments on User A's nom while User A's NomNet app is not open..	User A should receive a push notification that tells them about User B's comment.
14	User drags down on a food feed	Feed is refreshed with updated information.
15	User switches posts to private.	None of this user's posts are visible to other

		users.
16	User taps camera icon	Opens a menu with 2 tabs: Gallery and camera. Users can choose to upload pictures from the gallery or take a new picture here.
17	User inputs an invalid email address and password when creating account.	User is asked to please input correct information.
18	User types in part of a Nom	The search bar should attempt to complete the Nom and display suggestions based on the partial string.
19	User is disconnected from wifi and tries to log in.	Toast will appear saying the app cannot authenticate the user.
20	User is disconnected from wifi and tries to interact with the app functionalities.	Already loaded Noms from when user was connected to wifi will appear, but no new ones will be able to load and toast message will appear saying the items cannot be loaded.