**System and Unit Test:**

**Heading:** Document name ("System and Unit Test Report"), product name, team name, date.

**System Test scenarios (25 points per sprint – total 75 points):** You have identified user stories (that map to user requirements/functionality for your system) completed for each sprint (in that sprint’s report). You will be using scenario-based testing. Scenario-based testing focuses on how the user uses the system and allows for multiple user roles as well as multiple functions provided for each user role.

For each sprint, list the user story or stories and the scenario or scenarios that show ‘coverage’ of those user stories. A scenario is a list of system level actions (including precise input and output) a user would follow to determine that each user story has been completed.

Example:

A. User story 1 from sprint 1: As a user I want to create an account so that I can use the toilet location system.

B. User story 2 from sprint 1: As a registered user I want to view a map showing locations of all toilets in 5 mile radius of current location so that I can visually choose more information about toilets of interest.

Scenario:

1. start Toilet app; select ‘new user’; type

• name = <Linda Werner>

• password = <LLWPass9>

• password confirmation = <LLWPass9>

• press Enter Key

• user should see verification message that account is now active

2. select ‘view toilets in 5 mile radius’;

3. User should see map of 5 miles radius of current location with all toilets within the toilet database marked.

**Unit tests (25 points):** Include a file/directory named ‘Testing’ in your Git Repository. There should be details (can be in a separate file in the directory) provided by each team member about the module and the functional testing they have done. Each team member picks a module or module and lists the equivalence classes and the test cases selected to cover all equivalence classes

**System and Unit Test Report:**

**----------------------------------------------------------------------------------------------------------------------------**

**Product Name: SandwichStory**

**Team Name: The Sandwich Guys**

**Date: 7/22/17**

**----------------------------------------------------------------------------------------------------------------------------**

**System Test Scenarios**

* **Sprint 1 Scenarios**
  + **Scenario 1**
* User Story 1: As a user, I would like to access a list of my saved recipes so I can recall my desired recipes at any time.
* User Story 2: As a user, I would like to be able to create new recipes with my own ingredients so that I can save recipes I have made myself.
* User Story 5: As a user, I would like to be able to share my recipes so I can show my creativity with other users.

*Instructions:*

1. Open the SandwichStory application
2. Select the “Create” icon in the top navigation bar; this opens up the interface for creating a sandwich

* Name = “test”
* Qty = “1”
* Ingredient name = ”tomato”
* Press the “+” button to add the ingredient to the list
* Take a picture or select from gallery
* Description = “test sandwich”
* Click the “save sandwich” button and select yes on the pop-up window

3. Select “My List” icon in the top navigation bar; click on the test sandwich

and verify all the entries you have just entered are correct

1. Select the global icon and verify that your sandwich has been added to the database for other users to see
   * **Scenario 2**

* User Story 3: As a user, I would like to able to edit my recipes so that I can make modifications to recipe if needed.
* User Story 4: As a user, I would like to be able to delete recipes so that I may remove recipes I no longer have interest in.

*Instructions:*

1. Click the “My List” icon and click on the test sandwich

* On the interface that appears, select the edit button at the bottom left of the screen
* In the name text box, delete the name “test” and enter “test2”
* Delete the “tomato” ingredient by hitting the “x” to the right of the ingredient name, and add a new ingredient called “olive”
* Take a different photo and add it as the picture
* Replace the previous description with “editing sandwich”

2. Click the “Save Sandwich” button and select “no”

3. Select “My List” icon in the top navigation bar; click on the test2 sandwich and verify all the entries you have just entered

4. Select “My List” icon and select test2 sandwich

5. Click the delete button at the bottom right of the screen

6. Select yes on the popup menu

7. Navigate back to “My List” and confirm that test2 sandwich is no longer there

* **Sprint 2 Scenarios**
  + **Scenario 1**
* User Story 1: As a user, I would like to be able to add other user created recipes to my list so that I have quick access to good recipes besides the ones I have created.

*Instructions:*

1. Open the SandwichStory application

2. Select the “Global” icon in the top navigation bar; this opens up the

interface for the database that contains 15 random sandwiches

3. Select any sandwich you would like to add, it will bring you to the details

page

4. If you like the sandwich, scroll to the bottom of the info page and click the save button at the bottom

5. Navigate back to “My List” and confirm this sandwich is now in your list

* + **Scenario 2**
* User Story 2: As a user, I would like to view traditional recipes so that I have easy access to simple recipes

*Instructions:*

1. Open the SandwichStory application

2. Select the “Classic” icon in the top navbar; this opens up the

traditional/developers sandwiches list

3. Clicking any sandwich will display the information of that particular

sandwich

* **Sprint 3 Scenarios**
  + **Scenario 1**
* User Story 1: As a user, I would like to be able to have a stable database so that I can be connected with the users of the Sandwich Story application
* User Story 2: As a user, I’d like to be able to search for recipes based on keywords so I can easily find recipes I’m looking for.

*Instructions:*

1. Open the SandwichStory application

2. Select the “Global” icon in the top navigation bar; this opens up the

interface for the database that contains 15 random sandwiches

3. Click on the search button and enter the name of the sandwich or

ingredient that you would like to find

4. The list will now display sandwiches according to the search parameters with the most relevant options, sorted in descending order

5. Clicking the sandwich will display the details of the sandwich searched for

**Unit Tests**

**Choosing/Taking Picture Module: - Davie**

***Details of the Module:***

This module is used to assign an image to the corresponding sandwich that is being created or edited. If the user does this the first time permissions will be asked by the “requestPermissions()” function so that the app can have access to the camera and local storage. If the user denies the permissions the “onRequestPermissionsResult(...)” will continue to ask the user for permissions or until they close the dialog. If the user closes the dialog, the image function will not work and the user will be unable to edit or create a new sandwich. If the user accepts the permissions than the “onRequestPermissionsResult(...)” will run the “showPictureDialog()” function which will give the user the choice to choose a picture from their gallery which will run the “choosePhotoFromGallery()” function or the option to take a picture which will run the “takePhotoFromCamera()” function. After Choosing one of two options, “onActivityResult(...)” function will run determining which method the user choose. In the function, it will grab the image that the user choose or took a picture of and attach it to the imageView. If the user took a picture, the “saveImage(...)” function will run to store the image to the phone’s local storage. The “turnBitmapToEncodedString(...)” is ran to convert the bitmap into a string which allows it to be stored to appinfo and sharedPreferences. The storage allows the image to be loaded throughout the app.

***Test Cases That Cover The Module:***

Denying permissions when prompted by the app.

Accepting permissions when prompted by the app.

Choosing Gallery/Camera

Selecting an image/Taking a picture

Ignoring the Add Photo Button

Pushing Back Button during any of the steps above

***Equivalence Classes:***

showPictureDialog(), choosePhotoFromGallery(), takePhotoFromCamera(), onActivityResult(int requestCode, int resultCode, Intent data), turnBitmapToEncodedString(Bitmap image), saveImage(Bitmap imageToSave), hasPermissions(), requestPermissions(), onRequestPermissionsResult(int requestCode, @NonNull String [] permissions, @NonNull int [] grantResults)

**Loading Sandwiches from the Shared Preferences Module: - Matthew**

*Details:*

The module is used when the application is first opened. It allows for the application to get stored data in the internal memory of the phone and puts it into the RAM for easy access for the application. This is done by calling the application’s system preferences which are stored on the phone’s internal memory and putting all the information into the application’s AppInfo object for later use.

*Equivalent Classes:*

loadRecipes(), getSharedPreferences(), AppInfo getInstance(), JSONObj.getString(), ArrayList.add()

*Test Cases that Covers the Classes:*

In order to test this we would add sandwiches to the application and after making sure that they would save to the appInfo correctly by outputting the JSON.toString() into Android Monitor using logtags. After doing this, we could commit the data into the System Preferences of the device. After making sure all this is to work we would add the sandwich, and close the application and open it again to make sure that the sandwich would still be there. Once we have finish this test to make sure the sandwich is still there we would delete the sandwich and commit it again, which would delete the sandwich off the System Preferences. Once this is done we would close the application again and open it to check if the sandwich is still there. If the sandwich is no longer there then our delete and our loadRecipes would work.

**Deleting a sandwich from appInfo: - Chris**

*Details:*

The user selects a sandwich from the grid list and is given the option to either delete it or not to delete it from the current grid list. If the user does chose to delete the sandwich the sandwich information is found and all information is then deleted from the local memory location.

*Equivalent Classes:*

startActivity(), saveAsJSON(),addFlag(), Intent addFlags, AppInfo savedSandwichremove(), dialog dismiss()

*Test Cases that covers the Classes:*

Test the function though the users selecting a sandwich and using the button to delete the sandwich. So to test this function we must access the function by it self and using the assumption that those features have been selected prehand by the user. Next the function then delete it from the list. Note the function will crash if more than one of the same sandwich names are used because it will not be able to distinguish with sandwich to delete.

**Adding sandwich on the back end - David**

*Details of the Module:*

The function in the backend code, “add\_sandwich”, is used to hold the front-end info for sandwich recipes. This is done via passing a json request with the recipe object to the backend function itself. The backend function parses the info via dictionary look-up of request object variables and stores them in the SandwichStorage class defined there. As a developer, you can that the contents are saved correctly via calling the backend function “get\_user\_recipes” via url call.

*Functional Testing:*

We entered test input by calling the “add\_sandwich” database url and initializing the necessary request variables by passing arguments through the url. We verified that this function stores the information correctly by calling the url “get\_user\_recipes”, which will output what is stored in the SandwichStorage class. This has worked consistently with the url, and even when the information is sent from the app itself.

*Equivalence Classes and Test Cases:*

There are no true equivalence classes for this module since the execution of adding a sandwich is essentially the same for any variation of values passed to “add\_sandwich”. Thus, there is only one main test case (a described above in the “Functional Testing” tab).

**Search sandwiches on subqueries - Alex**

*Details of the Module:*

The function in the backend, “search”, uses a search keyword passed from the front-end

for finding user sandwich names and/or ingredients. It does this by stripping the keyword passed by the front-end of white space and transforming it to lowercase, in order to make sure the correct keyword is targeted by the query (as noted by a stack-overflow post we referenced from). We also define a limit variable to get targeted search results within a certain margin (example: ‘dog’ to ‘doh’). This means that the query can return substrings of anything the user inputs for searching (‘doggy’ would be found upon searching ‘dog’). The query we designed on the next line selects the sandwiches that have names and/or ingredient strings corresponding to these specific ranges, using a logical OR operator in the database call. The result is passed to the function “get\_result\_list”, where the sandwiches returned by the query are put into a json dictionary object to be returned to the front-end for parsing.

*Functional Testing:*

To test the functionality, after adding a select number of sandwiches to the database, we started inputting names in the search bar referencing the sandwich names that we added. On paper, we wrote down the sandwich results which should appear after entering specific name queries, and cross checked the sandwiches the search function displayed. We repeated this process for ingredient names, followed by cases where names and ingredients are identical. Finally, we teste substrings following the same technique described above.

*Equivalence Classes and Test Cases:*

There are no true equivalence classes for this module since the execution of searching

for a sandwich by name or ingredient name is essentially the same for any variation of values passed to “search”. Thus, there is only one main test case (a described above in the “Functional Testing”).

Saving sandwiches to the appInfo arrayList

Making sure the correct position is received when pressing a sandwich on the gridview