System and Unit Test Report:

Product Name: SandwichStory
Team Name: The Sandwich Guys
Date: 7/25/17

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
    4. 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 Truong

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 for 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, "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 chose or took a picture of and attach it to the imageView on the screen. 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)

Deleting a Recipe from AppInfo Module
Christopher Hahn

Details of the Module:
When the user selects a sandwich from the GridView list (only on “My List”), a “delete” button provides the option to delete the sandwich from the list. If the user does choose to delete the sandwich, the object is found in local memory and deleted.

Test Cases That Cover The Module:
We tested the delete functionality by first selecting a sandwich from the list. From there, we selected the “delete” button. This button then called a function that finds the object in AppInfo based on the position passed by the GridView and deletes it. To ensure it is actually deleted, we navigated back to “My List” and confirmed that the sandwich was no longer in the list.

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

Loading Sandwiches from the Shared Preferences Module
Matthew Diep

Details of the Module:
The module is used when the application is opened. It allows 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.

Test Cases That Cover The Module:
In order to test this, we added sandwiches to the application. To make sure that they saved to the AppInfo correctly, we outputted the recipe in JSON format printed as a string to Android Monitor. After this, we committed the data into the System Preferences of the device, which stored the data into the internal storage of the phone. To make sure the data was actually saved in System Preferences, we closed the application and reopened it, ensuring that the sandwich was still in our list. To test deleting sandwiches, we clicked the delete button on a specific recipe which updated System Preferences. The deletion should be immediately apparent, seeing that the sandwich is no longer in your list but to make sure, we closed the application again and reopened, ensuring that the sandwich was indeed gone from our list.

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

Adding Sandwich to the Back-End
David Stewart

Details of the Module:

The function in the back-end 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.

Test Cases That Cover The Module:
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:
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 Williamson

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.

Test Cases That Cover The Module:
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:
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”).

Loading Sandwiches from a Text File
Celine Peña

Details of the Module:
This is called when the application is opened, specifically by the file “presetRecipes.java” This file corresponds to the second tab of the application which displays classic recipes parsed straight from the text file, made into a sandwich object, loaded into AppInfo and subsequently loaded into the GridView.

Test Cases That Cover The Module:
To test the functionality of parsing a text file and loading the data into the classic recipes page, we included many variations on recipes as input in the text file. First, including recipes missing one or more required components messed up the parsing for the rest of the file since it reads it in in a rather delicate and specific way. But, since the user is not allowed to create a recipe without any ingredients, name or description, we did not include any recipes like this in our final input file. The basis for the rest of the testing relied on checking each recipe on the page and ensuring the details for each recipe matched the inputted text in the file.

Equivalence Classes:
1) Sandwiches missing one or more components (name, ingredients, description or image ID), these all caused the file to be parsed incorrectly following this specific recipe since the function parses in a very particular way.

2) Sandwiches with one or more missing components in any ingredient. When parsing, they are read as empty strings and subsequently displayed as empty strings. (commas between each component are required though. Without these, the ingredient is not correctly parsed)