BoilerHungry Defect Log #2

Design Inspection:

Date: 03/01/17
Author: The BoilerHungry Team
Moderator: Matthew Ashbeck
Inspectors: Eric Aguilera, Lena Arafa, Avi Rakesh
Recorder: Jason Chen
Defects:

1)
   Problem:
   The Food module contains separate boolean variables for each of the allergens that a food might have. This is a very poor design decision, as the app would no longer work if an allergen was added or removed from the Purdue dining court API.

   Severity: Important

   Solution:
   The Food module was changed to have a dynamic hashmap of string keys that represent allergens that map to true/false values. The app will now adapt to changes in types of allergens on the Purdue dining court API.

2)
   Problem:
   The DietaryPreferences module is designed in a way such that there is a boolean variable for a finite number of dietary options. If the dining court api were to change in any way such that dietary features were added or removed, the app would no longer function.

   Severity: Important

   Solution:
   The Dietary Preferences module was completely removed. It was instead replaced by functions in the Settings module that return a collection of strings that are to be excluded by a query of the available foods. The app will now adapt to changes in the dining court API.

Code Inspection:

Date: 03/01/17
Author: The BoilerHungry Team
Moderator: Matthew Ashbeck
Inspectors: Eric Aguilera, Lena Arafa, Avi Rakesh
Recorder: Jason Chen
Defects:

1.
Description:
The current version of JWebUnit that was added to the project dependencies will not work with Maven Jetty 9.42. The program will not compile until this is fixed.

Severity: Critical

How Corrected:
Downgraded Jetty version from 9.42 to 9.38 to allow for JWebUnit to run with our program.

2.
Description:
There needs to be some way to keep the context of one Settings module so that it does not need to be reloaded and overwritten in every module.

Severity: Important

How Corrected:
Created one Settings module and set the context in the homeServlet.

3.
Description:
Notifications code didn't match food item name to search results. Using the SearchUpcoming call to the API

Severity: Important

How Corrected:

Changed the API request to look in the locations/ folder from the Purdue dining court API to get a dynamic array that will add new dining courts upon a new API request if a new dining court is added.

4.
Description:
JSON Object from searchUpcoming API call contains a value “Stations”, which may be NULL

Severity: Important

How Corrected:
Wrapped the ItemAppearance JSONObject in an Optional object and then check to see if each of the attributes of this object (i.e Stations) is null. If this attribute is null, a null pointer will not be thrown and the program will still continue to run.

Unit Testing:

We decided to use the JUnitWeb framework to automate our test cases with the application. In order to test the functionality of our application, we had to make sure the HTML responses would remain consistent no matter how many times the test cases are run. To solve this problem, we implemented a MockDiningCourtAPI to redirect Purdue dining court API calls to return data from static files located in our test case folders. By creating a mock API, we were also able to broaden the scope of our test cases by modifying the mock API response files to test each and every feature that we parse. We strived to maximize the test case coverage to test out each unique feature after it was implemented. By using this automated test strategy, we would be able to not only tell if a feature is broken but also tell whether a new feature breaks any of our previously implemented features.

Unit Tests

Description:
Before every test case, a setup was made set the post for the web page and to navigate to “http://localhost:8080”

After every test case is run, a function refreshes and tears it down, so that every test case can be run independently and one test case don't impact the other.

What was Tested?	**How it was run?**	**The results:**
testTabTitle: - Begin the url at "/home" - Verified the Title of tab using XPath	JWebUnit	Passed
testWebPageTitle: - Begin the url at "/home" - Verify the text present at the Title: "BoilerHungry"	JWebUnit	Passed
testDifferentDiningCourtTitles: - Begin the url at "/home" - Verify the presence of Earhart Title using XPath - Verify the presence of Ford Title using XPath	JWebUnit	Passed
testDiningCourtOnPage: - Begin the url at "/home" - Verify that the Dining court on the webpage has "active" class and is being displayed - Verify that other Dining court does not have "active" class and hence not being displayed	JWebUnit	Passed
testNavigationToOneDiningCourtMenu: - Begin the url at "/home" - Verify that the Dining court visible on web page is Earhart using XPath - Click on the View Menu for Earhart using XPath - Verify that the "location" label is present on the web page using XPath - Verify that the location label is present on the web page using XPath - Verify that the location of Earhart is correct using Text	JWebUnit	Passed
testNavigationToOneDiningCourtMenuAndBack: - Begin the url at "/home" - Verify the presence of "Earhart" using XPath - Click on "View Menu" for Earhart - Verify Carousel is not present - Verify that the "location" label is present on the web page using XPath - Verify that the location label is present on the web page using XPath	JWebUnit	Passed

- Verify that the location of Earhart is correct using Text - Verify “Late Lunch” Text is not present. - Click on “Home” - Verify the presence of the Carousel - Verify presence of Earhart as the displaying dining court - Click on “next” - Click on “View Menu” - Verify location of Ford on web page - Verify “Late Lunch” Text on web page		
testNotificationTicker: - Begin the url at “/home” - Verify the presence of Earhart on the web page using XPath - Verify the presence of notification ticker on the web page using XPath	JWebUnit	Passed
testPictureOnHomePage: - Begin the url at “/home” - Verify that Earhart is present using XPath - Verify that the pictures on the background are present using XPath	JWebUnit	Passed
testSettingsTab: - Begin the url at “/home” - Verify the presence of Settings tab on the web page using XPath - Verify that My Foods is not present on the web page - Verify that Dietary Preferences is not present on the web page - Click on the Settings to open the tab - Verify the presence of “My Foods” - Verify the presence of “Dietary Preferences”	JWebUnit	Passed
testDiningCourtNavigation: - Begin the url at “/home” - Verify the presence of “Earhart” on the page - Verify the presence of the image on the background - Click on “next” - Verify the absence of Earhart on the page - Verify the presence of a new image on the page - Confirming that new Dining court is visible on the page	JWebUnit	Passed