COS420 : Test Plan Document
Group: Cooking Code-panions
Date: 04/21/23

Below is a document detailing six use case tests and their associated unit test cases. Five of the unit tests are implemented in our code and can be viewed in our GitHub

USE CASE 1

Title	Actors	Requirements
Login	User	Navigated to login page

Main Scenario
1. The user enters email and password then presses enter 2. System verifies credentials 3. System notifies the user of the result.

Alternate Scenarios
2a. System detects invalid credentials 2a1. System informs user of invalid login 2a1. System returns user to empty login page

Test Situations
a. Logging in with valid email and valid password b. Logging in with valid email and invalid password c. Logging in with invalid email

Test Coverage
Base: number of main and alternative scenario = 3 Number of Test Cases: 3 Test Coverage: 100%

USE CASE 2

Title	Actors	Requirements
Create Account	User	Navigated to create account page

Main Scenario
1. The user presses the create account button. 2. The system shows a new page where users can enter their account information and login credentials. 3. The user enters the data and presses the create account button 4. The system checks for pre-existing account associated with user provide email address 5. The system creates the new account on the backend 6. System displays a message reporting whether the operation was successful or not. 7. System begins an authorized user session.

Alternate Scenarios
4a. User provided email address already associated with a pre-existing account 4a1. System reports failed account creation, citing reason as pre-existing account 4b. User provided email address is invalid (improper syntax) 4b1. System reports failed account creation, citing reason as invalid email address

Test Situations
a. Creating a new account with novel email b. Creating a new account with preexisting email

Test Coverage
Base: number of main and alternative scenarios = 2 Number of Test Cases: 2 Test Coverage: 100%

USE CASE 3

Title	Actors	Requirements
Search recipe by name	User	User is on a page with search bar

Main Scenario
1. User enters recipe name into text box and clicks the “search” button 2. System searches database for list of recipes 3. System returns and presents results

Alternate Scenarios
2a. System finds no recipes matching search terms. 2a1. System returns an empty list. 2a2. System displays a failed search message

Test Situations
a. Searching for a recipe known to exist in database* b. Search full text for term not in title* c. Search with terms known not to exist

Test Coverage
Base: number of main and alternative scenario = 2 Number of Test Cases: 2 Test Coverage: 100%

USE CASE 4

Title	Actors	Requirements
Change Password from Profile settings	User	User navigates to profile settings

Main Scenario

1. User presses button to change password
2. System prompts user to re-enter credentials
3. User enters old credential
4. System verifies old credentials
5. System prompts user to enter new password
6. User enters new password
7. System saves new password

Alternate Scenarios

2a. User enters incorrect credentials
2a1. System informs user incorrect credentials, prompts user to enter credentials
4a. User enters invalid new password
4a1. System informs user incorrect password, prompts user to enter new password

Test Situations

a. User changes password to a valid alternative
b. User changes password to Invalid alternative
    i. Password too short
    ii. Password starts with number/special characters
    iii. Password doesn't contain capital letters
    iv. Password doesn't contain numbers

Test Coverage

Base: number of main and alternative scenario = 2
Number of Test Cases: 2
Test Coverage: 100%

USE CASE 5

Title	Actors	Requirements
View Recipe	User	

Main Scenario
1. The user clicks the link to a specific recipe. 2. The system takes the user to a recipe viewing page. 3. The system displays the ingredients and instructions for the specific recipe.

Alternate Scenarios
1. The user clicks the link to a specific recipe.

Test Situations
a. Retrieve an arbitrary recipe from database* b. Retrieve a public recipe by id* c. Retrieve public recipe known not to exist* d. View a private recipe from owner's account e. View a private recipe from unauthorized account f. Attempt to view an recipe w/ invalid URL

Test Coverage
Base: number of main and alternative scenario = 4 Number of Test Cases: 4 Test Coverage: 100%

USE CASE 6

Title	Actors	Requirements
Filter Search with Tags	User	User is viewing search results

Main Scenario
1. User clicks the “filter” button 2. System displays a list of filter options and tags 3. User selects a list of filter criteria 4. User confirms choices 5. System removes any recipes that do not meet the criteria from the list of recipes to display 6. System displays the adjusted list of recipes in the database view

Alternate Scenarios
3a. User selects an unrecognized filter option 3a1. System disregards filter option and displays message to user that filter is unrecognized 4a. User has input no filter criteria 4a1. System denies attempt to confirm and adds message that filter 6a. No recipes are left in the list. 6a1. System displays a “No recipes found” message on the blank search display page.

Test Situations
a. Search by a valid tag b. Search by an invalid tag

Test Coverage
Base: number of main and alternative scenario = 2 Number of Test Cases: 2 Test Coverage: 100%

Unit Testing

1.) Login
   a. Logging in with valid email and valid password
   b. Logging in with valid email and invalid password
   c. Logging in with invalid email

2.) Create Account
   a. Creating a new account with novel email
   b. Creating a new account with preexisting email

3.) Search recipe by name
   a. Searching for a recipe known to exist in database*
   b. Search full text for term not in title*
   c. Search with terms known not to exist

4.) Change Password from Profile settings
   a. User changes password to a valid alternative
   b. User changes password to Invalid alternative
      i. Password too short
      ii. Password starts with number/special characters
      iii. Password doesn't contain capital letters
      iv. Password doesn't contain numbers

5.) **View Recipe**
   a. Retrieve an arbitrary recipe from database*
   b. Retrieve a public recipe by id*
   c. Retrieve public recipe known not to exist*
   d. View a private recipe from owner's account
   e. View a private recipe from unauthorized account
   f. Attempt to view an recipe w/ invalid URL

6.) Filter Search with Tags
   a. Search by a valid tag
   b. Search by an invalid tag

*Unit test is implemented in code on GitHub

Tests Implemented in Code & Screenshots:

Test: Getting a recipe from an existing ID

```
def test_get_recipe_by_existing_id(self):
    """
    Retrieve recipe that is known to exist
    """
    identifier = "6439ee7a028ed862258322b8"
    result = self.db.get_recipe_by_id(identifier)
    self.assertEqual(result["_id"], ObjectId(identifier), f"Should be {identifier}")
```

Test: Getting a recipe from an non-existing ID

```
def test_get_recipe_by_nonexisting_id(self):
    """
    Retrieve recipe that is known not to exist
    """
    identifier = "6439ee7a028ed882258322c8"
    result = self.db.get_recipe_by_id(identifier)
    self.assertEqual(type(result), type(None), f"Should be {type(None)}")
```

Test: Tests full text search capability

```
def test_full_search_(self):
    """
    Verify that full_search returns  a list of recipe objects
    """
    known_recipe = self.db.get_recipe_by_id("6439ee7a028ed862258322b8")
    search_results = self.db.full_search(known_recipe["title"])

    for recipe in search_results:
        self.assertEqual(type(recipe), type(known_recipe), f"Should be {type(known_recipe)}")
```

Test: Getting a recipe from a title

```
def test_get_with_title(self):
    """
    Searches for a recipe with a specific title and checks that
    the recipe that is returned has that title
    """
    title = "Pumpkin Pancakes"
    recipe = list(self.db.get_with_title(title))[0]
    self.assertEqual(title, recipe["title"], "The titles must match")
```

Test: Getting first recipe in database

```
def test_get_first_recipe(self):
    """
    Checks to make sure that get_first_recipe returns a recipe and it has a title
    """

    recipe = self.db.get_first_recipe()
    self.assertIsInstance(recipe, dict, "The recipe must be a dictionary")
```

Test: Get recipe by title with full search

```
def test_full_search_by_title(self):
    """
        Verify that full_search returns known recipe that you searched for.
    """
    known_recipe = self.db.get_recipe_by_id("6439ee7a028ed862258322b8")
    search_results = self.db.full_search(known_recipe["title"])

    containsKnown = False
    for recipe in search_results:
        if recipe["title"] == known_recipe["title"]:
            containsKnown = True

    self.assertTrue(containsKnown)
```

TESTING CODE OUTPUT

```
----------------------------------------------------------------------
Ran 5 tests in 1.675s

OK
```