SDS: TEST PLAN

Product: FindMyMeal

Version: 1.0

Date: 2023-10-27

1. **Introduction**
   This Software Design Specification Test Plan outlines the testing activities to ensure that the software developed for FindMyMeal follows the specifications previously stated. This plan serves as a guideline for the testing process, including the test objectives, scope, strategy, environment, and schedule.

2. **Objectives**
   - **a.** Verify that the User features: Craftable Recipes, Shopping Lists, Inventory Tracking function as specified
   - **b.** Test the Non-Volatile Storage of objects (recipe, prepopulated pantry)
   - **c.** Testing the Reminder Feature
   - **d.** Testing the Budget Feature\
   - **e.** Testing the Login Feature and the pages within the software
3. **Test Strategy**
   - **a.** Test Levels
     i. Unit Testing
     ii. Integration Testing
     iii. System Testing
     iv. Acceptance Testing
   - **b.** Test Types
     i. Unit Testing
     ii. Functional Testing
     iii. System Testing (Performance, Security)
     iv. Usability Testing
   - **c.** Test Approach
     i. Test cases will be derived from the software design specifications.
     ii. Test data will be generated to cover a wide range of scenarios.
4. **Test Cases**
   a. Detailed test Cases are documented on the next pages.

Thursday, October 26, 2023

Project 2 CS 250

Hong Nhung Nguyen

1. Unit Test & Functional Test

- Def: Unit test maybe an individual function, method, procedure, module or object in the project
- Unit Test example: UserAccount Class. This class detailed basic users's account setting.
  - Tested Components: Username; Password
  - Tested Methods: setUserName(); setPassword(); addPlanner(); removePlanner(); addRecipe(); removeRecipe(); setBudget();getBudget()

Unit Testing for Username & Password			
Test Case	**Type**	**Description**	**Objectives**
One piece	String	Accepted Input	Verify that an user could enter their user name and password
" "	null	Error: Invalid Input. Username and password cannot be null	Verify that an user cannot leave their username and password blank
12578	Int	Accepted Input	Verify that an user could enter either string or int datatype
2200483647	Int	Error: out of limit of int data type	Verify what happen if input received out of upper bound

Functional Test Plan for methods in UserAccount Class			
setUserName()	setPassword()	addRecipe()	removeRecipe()
Enter valid user name: "Jenny123"	Enter valid password: "Pass123"	Enter content of recipe	Choosing an existing recipe
Expected output: Pass. when user enter valid input, setUserName() should add new user name to the database	Expected output: Pass. when user enter valid password , setPassword() should save password associated with the user account to the database	Expected output: Pass. when user choose add recipe option; addRecipe() should allow new recipe to be added to the database	Expected output: Pass. when user choose to remove an existing recipe in their account, removeRecipe() will enable a recipe to be removed from the database.

2. System Testing:

Testing the integrated system as a whole

- *Login Page:*
  - Test case 1: verify user enter valid user name
  - Test case 2: if user enter valid user name -> ask to enter password -> verify user enter valid password
- *Recipe page:*
  - Test case 1: once user login successfully. Verify that user can add recipe in recipe page.
  - Test case 2: once user login successfully. Verify that user can search for recipe under recipe page.
  - Test case 3: once user login successfully. Verify that user can modify their recipe.
- *Data manangement:*
  - Test case 1: verify that database is updated frequent once any changes happens
  - Test case 2: verify that user can see their history on their account.
- *App UI:*

- Test case 1: verify that all app element are displayed correctly and user friendly. For example, when opening app user should see login page pop up first with sign-in button and “create account” button next to each other.
- Test case 2: verify that each app options is correctly interacted to the other. For example: after login successfully, user will able to see their homepage layout different app options.

• *Performance testing:*

- Test case 1: measure app performance running on OS/Android mobile platform
- Test case 2: measure website performance such as loading page speech when running on desktop.

Reminder Class
Reminders feature allows the user to set a reminder using time and date features when selecting ingredients to add for purchase. The user would use the reminder feature to create a list of ingredients and plan for when to purchase or prepare ingredients to use in recipes for meal preparation.

Unit Testing - Test Case for Reminder Feature

Test Case Description	The users should be able to set a reminder using time and date features and add ingredients needed to purchase.				
Test Scenario	Checking that after entering, the reminder is set at the correct time and day with ingredients needed for purchase.				
Test Case ID	Test Steps	Test Input	Expected Results	Actual Results	Status
1	1. Enter valid date (mm/dd/yyyy)	Date:10/30/2023	Reminder: 10/30/2023 at 05:03 PM	Reminder: 10/30/2023 at 05:03 PM	Valid
	2. Enter Time (hh:mm)	Time: 05:30			
	3. Select Am or PM	PM			
2	1. Enter past date (mm/dd/yyyy)	Date:10/23/2022	System should display an error message indicating that a reminder cannot be set for a past date and time.	[Error Message]: "Cannot set a reminder for a past date and time."	Valid
	2. Enter Time (hh:mm)	Time: 03:45			
	3. Select Am or PM	PM			
3	1. Enter future date (mm/dd/yyyy)	Date:11/25/2023	Reminder: 11/25/2023 at 08:15 AM	Reminder: 11/25/2023 at 08:15 AM	Valid
	2. Enter Time (hh:mm)	Time: 08:15			
	3. Select Am or PM	PM			

Unit Testing (Budget Class)

Testing the Operations for the Budget Class

Budget.increaseBudget()

Test for Blank Input, Invalid Input, Valid Input. (Any positive double between 0 and 9,999,999)

- Empty Input: increaseBudget(“ “)
- Invalid Input: increaseBudget(“abcd123”)
- Valid Input: increaseBudget(“100.99”)

Budget.decreaseBudget()

Test for Blank Input, Invalid Input, Valid Input. (Any negative double between the Budget and 0)

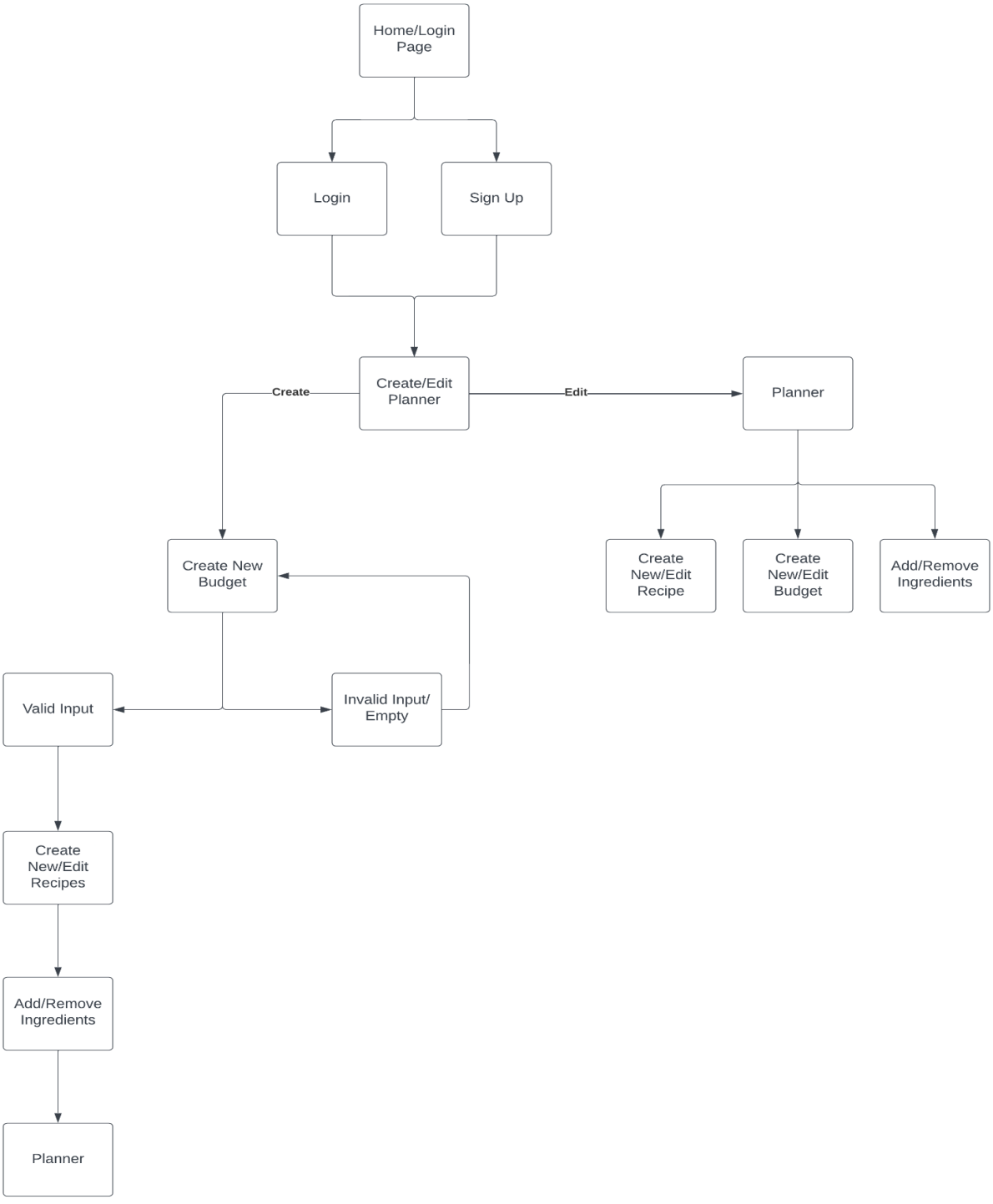
- Empty Input: decreaseBudget(“ “)
- Invalid Input: decreaseBudgetN(“-catmonkey”)
- Valid Input: increaseBudget(“-99.99”)

Functional Testing (Adding a Planner)

Adding a Planner that has a set budget and directly compares it to the Recipe price

As a user, I want to be able to create a planner that has a set budget that restricts me from going over as I shop and create recipes. The software will accumulate the price of every Ingredient entered and the Recipe price should be equal to the sum of all ingredients Recipe price will be compared to the Budget Object and check if the Recipe is below Budget.

System Testing (Budget)



4	1. Enter invalid date (dd/mm/yyyy)	Date: 25-10-2023	System should display an error message indicating that a reminder cannot be set for a past date and time.	[Error Message]: "Cannot set a reminder for a past date and time."	Valid
	2. Enter Time (hh:mm)	Time: 12:30			
	3. Select Am or PM				
5	1. Enter valid date (mm/dd/yyyy)	Date: 10/26/2023			
	2. Enter an invalid time (99:99 AM).	Time: 99:99 AM	System should display an error message indicating that the time format is incorrect.	[Error Message]: "Invalid time format. Please use hh:mm AM/PM."	Valid
	3. Select Am or PM				

Test Set: Ingredient Selection in Reminder Feature

Test Case Description	The users should be able to select ingredients within the Reminder feature. This set of test cases ensures that the system handles ingredient selection correctly. These test cases ensure the user has full functionality in the Reminder feature, like adding, editing, or removing ingredients.				
Test Scenario	Checking that after selecting the ingredient, the reminder has the full ingredient list as well as the correct date and time.				
Test Case ID	Test Steps	Test Input	Expected Results	Actual Results	Status
1	1. Create a new reminder	Reminder Date: 10/27/2023	Reminder displays: 10/27/2023 at 03:00 PM.	Reminder displays: 10/27/2023 at 03:00 PM.	Valid
	2. Select the "Add Ingredient" option.	Time: 03:00 PM			

	3. Search for an ingredient	Ingredient: Tomatoes			
	4. Add the ingredient to the reminder		Ingredient List: Tomatoes	Ingredient List: Tomatoes	
2	1. Create a new reminder	Date:10/23/2022	System should display an error message indicating that a reminder cannot be set for a past date and time.	[Error Message]: "Cannot set a reminder for a past date and time."	Valid
	2. Select the "Add Ingredient" option.	Time: 03:45			
	3. Attempt to add an ingredient with an invalid name (e.g., "Invalid!@#").	Invalid Ingredient: Invalid!@#			
	4. Verify if the system rejects the invalid ingredient.		System should display an error message indicating that the ingredient name is invalid and should not add the ingredient to the list.	[Error Message]: "Invalid ingredient name.	
3	1. Create a new reminder	Date:11/25/2023 at 08:15 AM	Reminder: 11/25/2023 at 08:15 AM	Reminder: 11/25/2023 at 08:15 AM	Valid
	2. Select the "Add Ingredient" option.				
	3. Search for an ingredient (e.g., "Onions")		Initial Ingredient: Onions		

	4. Add the ingredient to the reminder.				
	5. Edit the ingredient name to “Green Onions."		Edited Ingredient: Red Onions	Ingredient List: Red Onions	
4	1. Create a new reminder	Date:11/25/2023 at 08:15 AM	Reminder: 11/25/2023 at 08:15 AM	Reminder: 11/25/2023 at 08:15 AM	Valid
	2. Select the "Add Ingredient" option.		Ingredient List: Carrots		
	3. Search for an ingredient (e.g., “Squash")				
	4. Add the ingredient to the reminder.				
	5. Delete the added ingredient from the reminder			Ingredient List is empty (no ingredients).	

Functional Testing:

Case 1: Creating a Reminder

- Test the end-to-end process of creating a reminder with a specific date, time, and a list of ingredients.
- Verify that the reminder is successfully created and saved.

Case 2: Editing a Reminder

- Create a reminder.
- Test the ability to edit the reminder's date, time, and ingredients.
- Verify that the edited information is saved correctly.

Case 3: Deleting a Reminder

- Create a reminder.
- Test the ability to delete a reminder.
- Verify that the reminder is removed from the system.

Case 4: Ingredient List Management

- Create a reminder with a list of ingredients.
- Test the ability to add, edit, and delete ingredients from the list.
- Verify that the ingredient list is correctly updated.

System Testing:
Time (hour) - accepts values between 1 - 12
Time (minutes) - accepts values between 0 - 60
Date (mm/dd/yy) -
Ingredient- accepts a string for the ingredient needed for

System Testing - Equivalence classes for the input data

Test Case	Valid	Invalid	Empty
Time - Hour	>= 1 & <= 12	< 1 & >12	“ “
Time - Mintes	>= 0 & <= 60	< 0 & > 60	“ "
Date - Month	>= 1 & <= 12	< 1 & >12	“ "
Date - Day	>= 1 & <= 31	< 1 & >31	“ "
Date - Year	>= current year & <= 2100	< current year & > 2100	“ "
Ingredient	String of valid ingredient	!@#$	“ "

Blackbox Test Case for Reminder

Takes int input from the user for setting a reminder					
TC ID	**Test Name**	**Test Condition**	**Test Steps**	**Test Input**	**Expected Result**
1	Time - Hour	Takes integer input from the user for setting a reminder	1. Input int > 00:00	05:00	Reminder Set for 5:00
2	Time - Hour	Takes integer input from the user for setting a reminder	1. Input int < 01:00	00:00	Invalid
3	Time - Minute	Takes integer input from the user for setting a reminder	1. input int > 00:00	00:30	Reminder set
4	Time - Minute	Takes integer input from the user for setting a reminder	1. input int < 00:00	- 00:00	Invalid
5	Ingredient	Accepts a string input from the user for the ingredient needed for the reminder	1. input a valid string: “Pasta”	Pasta	Ingredient added to the reminder

Takes int input from the user for setting a reminder					
TC ID	**Test Name**	**Test Condition**	**Test Steps**	**Test Input**	**Expected Result**
6	Ingredient	Accepts a string input from the user for the ingredient needed for the reminder	1. Attempt to input an invalid ingredient name	Oranges!	Invalid

System Testing

User Features

1. Craftable Recipes

As a user, I want to know what recipes I can make with the ingredients I have on hand so I can better decide what meal I want to prepare.

Given: a predefined recipe book and populated pantry containing a set of ingredients **When**: the user navigates to the craftable recipes feature **Then**: they should see a list of recipes able to be made with the available ingredients

2. Shopping Lists

As a user, I want to know what ingredients (if any) I'm missing in order to create a given recipe or collection of recipes so I know what groceries I need to buy.

Given: the user is viewing the details of a recipe or a list of recipes in the recipe book **When**: the user selects one or more recipes from a list and presses a button to plan these recipes, or presses a corresponding button on an individual recipe page **Then**: the recipe(s) are added to the meal plan

Given: the user is viewing the meal plan **When**: the user presses a button to generate a shopping list **Then**: the user will see a list of ingredients and quantities needed to create the planned recipes, less the amounts already on hand

3. Inventory Tracking

As a user, I want the software to keep track of when I use ingredients that are part of known recipes so I don't have to update the on-hand quantities manually every time I create a meal.

Given: the user is viewing the details of a recipe **When**: the user presses a button to create this recipe **Then**: the quantity of each ingredient in the pantry are reduced by the amount defined in the recipe, or a multiple (of servings) thereof AND removed from the panty completely if the new quantity is zero

Given: the user is viewing the shopping list **When**: the user presses a button to "do shopping" **Then**: the user is able to change the quantities of each ingredient in the list and unselect certain ingredients (to account for the fact that ingredients might not be bought - or sold - in the exact amount needed) **When**: the user presses a button to "add shopping to pantry" **Then**: the quantity of each ingredient in the pantry is increased by the corresponding amount in the shopping list

Integration Testing

Non-volatile Storage

Given: a prepopulated pantry or recipe book object and an injectable storage provider **When**: the object is saved and a new object of the same class is instantiated using the same storage provider **Then**: the two objects should be deeply equal

Unit Testing

Recipe.isCraftable(Pantry p)

Given: a pantry object containing a list of available ingredients **When**: the pantry is empty **Then**: the recipe is not craftable (return false) **When**: the pantry contains all required ingredients in one less than the required quantity **Then**: the recipe is not craftable **When**: the recipe contains all but one of the required ingredients in sufficient quantity, and one ingredient is absent **Then**: the recipe is not craftable **When**: the pantry contains all required ingredients in the exact quantities required **Then**: the recipe is craftable **When**: the pantry contains all required ingredients in the maximum storable value of their datatype **Then**: the recipe is craftable

RecipeBook.getCraftableRecipes(Pantry p)

Given: a pantry object containing a list of available ingredients **When**: the pantry is empty **Then**: method returns an empty set **Given**: the pantry contains sufficient ingredients to craft some but not all recipes in RecipeBook **Then**: method returns a list of only the craftable recipes **Given**: the pantry contains sufficient ingredients to craft any recipe in RecipeBook **Then**: method returns a list contains all recipes in RecipeBook