**STARTUP CONTRACTS**

**Contract CO1: register**

| | |
|---|---|
| **Operation:** | register(name, info) |
| **Cross References:** | Use Cases: 5.1 Startup |
| **Preconditions:** | ● Application is running |
| **Postconditions:** | ● A user instance has been instantiated<br>● User attributes have been initialized (name, opt. weight, gender)<br>● User data has been saved to the database |

**PANTRY CONTRACTS**

**Contract CO2: getPantryItems**

| | |
|---|---|
| **Operation:** | getPantryItems() |
| **Cross References:** | Use Cases: 5.1 Startup, 1.1 Manage Pantry, 1.2 Search Pantry, 1.3 Consume Pantry Item |
| **Preconditions:** | ● User has been registered within the system and initialized<br>● Data has been loaded from the database |
| **Postconditions:** | ● System provides pantry data to user |

**Contract CO3: addPantryItem**

| | |
|---|---|
| **Operation:** | addPantryItem(foodId, quantity) |
| **Cross References:** | Use Cases: 5.1 Startup, 1.1 Manage Pantry |
| **Preconditions:** | ● Data has been loaded from the database<br>● Food already exists within the food database |
| **Postconditions:** | ● Added food item exists within the user's pantry<br>● Pantry quantity of food item has been incremented to quantity + old quantity<br>● Pantry database updated |

**Contract CO4: editPantryItem**

| | |
|---|---|
| **Operation:** | editPantryItem(foodId, quantity) |
| **Cross References:** | Use Cases: 1.1 Manage Pantry |
| **Preconditions:** | ● Data has been loaded from the database<br>● Food exists within the food database |
| **Postconditions:** | ● Food item is updated with new quantity<br>● Pantry database is updated |

**Contract CO5: removePantryItem**

| | |
|---|---|
| **Operation:** | removePantryItem(foodId) |
| **Cross References:** | Use Cases: 5.1 Startup, 1.1 Manage Pantry |
| **Preconditions:** | ● Data has been loaded from the database<br>● An item of the given food type exists within the pantry |
| **Postconditions:** | ● Food item with given name has its instance removed from pantry<br>● Pantry database is updated |

**Contract CO6: searchPantryByFoodName**

| | |
|---|---|
| **Operation:** | searchPantryByFoodName(query) |
| **Cross References:** | Use Cases: 1.2 Search Pantry |
| **Preconditions:** | ● Data has been loaded from the database |
| **Postconditions:** | ● List of food items starting with item name are returned or an empty list if no pantry items match the itemName |

**Contract CO7: consume**

| | |
|---|---|
| **Operation:** | consume(foodId, quantity) |
| **Cross References:** | Use Cases: 1.3 Consume Pantry Item |
| **Preconditions:** | ● Data has been loaded from the database<br>● FoodItem passed in exists in the pantry |
| **Postconditions:** | ● Food in pantry with type foodItem has its quantity decremented<br>● If new quantity is 0, foodItem is removed from pantry<br>● Database is updated |

**SHOPPING LIST CONTRACTS**

**Contract CO8: getShoppingItems**

| | |
|---|---|
| **Operation:** | getShoppingItems() |
| **Cross References:** | Use Cases: 4.1 Manage Shopping List, 4.3 Mark Purchased Items |
| **Preconditions:** | • Data has been loaded from the database |
| **Postconditions:** | • System provides user with their shopping list |

**Contract CO9: addShoppingItem**

| | |
|---|---|
| **Operation:** | addShoppingItem(foodId, quantity) |
| **Cross References:** | Use Cases: 4.1 Manage Shopping List |
| **Preconditions:** | • Data has been loaded from the database<br>• The food type exists within the list of registered food types<br>• The quantity is valid (non-negative, etc) |
| **Postconditions:** | • Food item with given quantity added to the shopping list<br>• Database is updated |

**Contract CO10: editShoppingItem**

| | |
|---|---|
| **Operation:** | editShoppingItem(foodId, quantity) |
| **Cross References:** | Use Cases: 4.1 Manage Shopping List |
| **Preconditions:** | • Data has been loaded from the database<br>• The food type exists within the shopping list<br>• The quantity is valid (non-negative, etc) |
| **Postconditions:** | • Quantity of given food item updated within the shopping list<br>• Database is updated |

**Contract CO11: removeShoppingItem**

| | |
|---|---|
| **Operation:** | removeShoppingItem(foodId) |
| **Cross References:** | Use Cases: 4.1 Manage Shopping List |
| **Preconditions:** | • Data has been loaded from the database<br>• The food type exists within the shopping list<br>• The quantity is valid (non-negative, etc) |
| **Postconditions:** | • Food item removed from shopping list<br>• Database is updated |

**Contract CO12: export**

| | |
|---|---|
| **Operation:** | export(fileFormat, destination) |
| **Cross References:** | Use Cases: 4.1 Manage Shopping List, 4.2 Export Shopping List |
| **Preconditions:** | • Data has been loaded from the database<br>• User has access to the given destination<br>• Given destination is valid<br>• File format is valid |
| **Postconditions:** | • Shopping list has been exported to desired format |

**Contract CO13: purchaseItems**

| | |
|---|---|
| **Operation:** | purchaseItems(foodIds) |
| **Cross References:** | Use Cases: 4.3 Mark Purchased Items |
| **Preconditions:** | • Data has been loaded from the database<br>• Given foods are registered in the list of foods<br>• User purchased given items from the shopping list |
| **Postconditions:** | • System moves given items from shopping list to pantry<br>• Database is updated |

**RECIPE CONTRACTS**

**Contract CO14: getRecipes**

| | |
|---|---|
| **Operation:** | getRecipes() |
| **Cross References:** | Use Cases: 3.1 Create Recipe, 3.2 View Recipe, 3.3 Add Recipe to Shopping List, 3.4 Modify Recipe, 3.5 Recommend Recipes |
| **Preconditions:** | ● Data has been loaded from the database |
| **Postconditions:** | ● System provides the user's list of recipes |

**Contract CO15: addRecipe**

| | |
|---|---|
| **Operation:** | addRecipe(name, ingredients, instructions, servings) |
| **Cross References:** | Use Cases: 5.1 Startup, 3.4 Modify Recipe |
| **Preconditions:** | ● Data has been loaded from the database |
| **Postconditions:** | ● Ingredients have been stored has food instances within the database<br>● The recipe represents a new recipe instance within the recipe system<br>● Recipe appears within the recipe list |

**Contract CO16: produceCookedRecipe**

| | |
|---|---|
| **Operation:** | produceCookedRecipe(recipeId, isUsePantry, consumedServings, leftoverServings) |
| **Cross References:** | Use Cases: 3.6 Cook Recipe |
| **Preconditions:** | ● Recipe exists within the recipe system<br>● Ingredients exist within the food database<br>● Data has been loaded from the database |
| **Postconditions:** | ● User is notified of any errors preventing cooking<br>● Consumed portion is logged to food log<br>● Pantry is updated to contain leftovers<br>● Database is updated |

**Contract CO17: editRecipe**

| | |
|---|---|
| **Operation:** | editRecipe(recipeId, recipe) |
| **Cross References:** | Use Cases: 3.4 Modify Recipe |
| **Preconditions:** | ● Data has been loaded from the database<br>● recipeId is valid |
| **Postconditions:** | ● User is notified of any errors preventing modification<br>● Recipe is saved to Recipe List<br>● Database is updated |

**Contract CO18: getRecipe**

| | |
|---|---|
| **Operation:** | getRecipe(recipeId) |
| **Cross References:** | Use Cases: 3.5 Recommend Recipes |
| **Preconditions:** | ● Data has been loaded from the database<br>● Recipe exists within the recipe system |
| **Postconditions:** | ● Recipe is displayed to the user |

**Contract CO19: getRecommendRecipes**

| | |
|---|---|
| **Operation:** | getRecommendRecipes() |
| **Cross References:** | Use Cases: 3.5 Recommend Recipe |
| **Preconditions:** | ● Data has been loaded from the database |
| **Postconditions:** | ● The user is presented with all recommended recipes from the Recipe List based off of items in their pantry |

**Contract CO20: searchByRecipeName**

| | |
|---|---|
| **Operation:** | searchByRecipeName(query) |
| **Cross References:** | Use Cases: 3.2 View Recipes |
| **Preconditions:** | ● Data has been loaded from the database |
| **Postconditions:** | ● List of matching recipes are displayed to the user |

**Contract CO21: addIngredientsToCart**

| | |
|---|---|
| **Operation:** | addIngredientsToCart(recipeId) |
| **Cross References:** | Use Cases: 3.3 Add Recipe to Shopping Cart |
| **Preconditions:** | ● Data has been loaded from the database |
| **Postconditions:** | ● Recipe ingredients added to shopping cart<br>● Database updated |

**Contract CO22: addMissingIngredientsToCart**

| | |
|---|---|
| **Operation:** | addMissingIngredientsToCart(recipeId) |
| **Cross References:** | Use Cases: 3.3 Add Recipe to Shopping Cart |
| **Preconditions:** | ● Data has been loaded from the database |
| **Postconditions:** | ● Recipe ingredients not in pantry added to shopping cart<br>● Database updated |

**NUTRITION LOG CONTRACTS**

**Contract CO23: addNutritionItem**

| | |
|---|---|
| **Operation:** | addNutritionItem(foodId, quantity, consumedAt) |
| **Cross References:** | Use Cases: 3.6 Cook Recipe, 2.1 Manage Nutrition Log |
| **Preconditions:** | ● Data has been loaded from the database<br>● Food exists within the recipe system<br>● Quantity is valid (non-negative, valid unit, etc.) |
| **Postconditions:** | ● Food of given quantity logged to nutrition log at given time<br>● Nutrition log is updated with new nutrition instance<br>● Database is updated |

**Contract CO24: getNutritionItems**

| | |
|---|---|
| **Operation:** | getNutritionItems(startDate, endDate) |
| **Cross References:** | Use Cases: 2.1 Manage Nutrition Log |
| **Preconditions:** | ● Data has been loaded from the database |
| **Postconditions:** | ● System provides the list of nutrition instances from the given time window |

**Contract CO25: editNutritionItem**

| | |
|---|---|
| **Operation:** | editNutritionItem(foodId, quantity, consumedAt) |
| **Cross References:** | Use Cases: 2.1 Manage Nutrition Log |
| **Preconditions:** | ● Data has been loaded from the database<br>● Quantity is valid (non-negative, valid unit, etc.) |
| **Postconditions:** | ● The selected item has had its values changed as per user request<br>● Database is updated |

**Contract CO26: removeNutritionItem**

| | |
|---|---|
| **Operation:** | removeNutritionItem(foodId, consumedAt) |
| **Cross References:** | Use Cases: 2.1 Manage Nutrition Log |
| **Preconditions:** | ● Data has been loaded from the database<br>● The given nutrition item is already in the food log |
| **Postconditions:** | ● The selected item has been removed from the log<br>● Database is updated |

**Contract CO27: addGoal**

| | |
|---|---|
| **Operation:** | addGoal(nutrient, quantity, goalType) |
| **Cross References:** | Use Cases: 2.2 Nutrition Goals |
| **Preconditions:** | ● Data has been loaded from the database<br>● The input values are valid and logical |
| **Postconditions:** | ● A nutrition goal has been created using the given fields<br>● Database is updated |

**Contract CO28: editGoal**

| | |
|---|---|
| **Operation:** | editGoal(id, nutrient, quantity, goalType) |
| **Cross References:** | Use Cases: 2.2 Nutrition Goals |
| **Preconditions:** | ● Data has been loaded from the database<br>● The input values are valid and logical |
| **Postconditions:** | ● The selected goal has had its values changed as per user request<br>● Database is updated |

**Contract CO29: removeGoal**

| | |
|---|---|
| **Operation:** | removeGoal(id) |
| **Cross References:** | Use Cases: 2.2 Nutrition Goals |
| **Preconditions:** | ● Data has been loaded from the database<br>● The given goal exists within the user's list of goals |
| **Postconditions:** | ● The selected goal has been removed<br>● Database is updated |

**Contract CO30: addReport**

| | |
|---|---|
| **Operation:** | addReport(startDate, endDate) |
| **Cross References:** | Use Cases: 2.3 Manage Nutrition Report |
| **Preconditions:** | ● The user is in the nutrition reports menu<br>● The input values are valid and logical |
| **Postconditions:** | ● A nutrition report has been created using the given fields |

**Contract CO31: editReport**

| | |
|---|---|
| **Operation:** | editReport(id, startDate, endDate) |
| **Cross References:** | Use Cases: 2.3 Manage Nutrition Report |
| **Preconditions:** | ● The user has selected a report to be edited<br>● The input values are valid and logical |
| **Postconditions:** | ● A nutrition report has been edited |

**Contract CO32: removeReport**

| | |
|---|---|
| **Operation:** | deleteReport(id) |
| **Cross References:** | Use Cases: 2.3 Manage Nutrition Report |
| **Preconditions:** | ● The user is in the nutrition report menu<br>● At least one report exists within the menu which the user wants to delete |
| **Postconditions:** | ● The selected report has been removed from the report menu |

**FOOD CONTRACTS**

**Contract CO33: getFood**

| | |
|---|---|
| **Operation:** | getFood(id) |
| **Cross References:** | Use Cases: 6.1 Manage Food Types, 6.2 Create New Food Types, 2.1 Manage Nutrition Log |
| **Preconditions:** | ● Data has been loaded from the database |
| **Postconditions:** | ● System provides user with the food of the given id |

**Contract CO34: addFood**

| | |
|---|---|
| **Operation:** | addFood(name, category, nutrition), 6.2 Create New Food Types |
| **Cross References:** | Use Cases: 6.1 Manage Food Types, 4.1 Manage Shopping List, 1.1 Manage Pantry, 1.2 Search Pantry, 2.1 Manage Nutrition Log |
| **Preconditions:** | ● Data has been loaded from the database<br>● The name is not already in use by another food<br>● The name is not empty<br>● The nutrition info is valid (non-negative macros, etc) |
| **Postconditions:** | ● System adds food to list of foods<br>● Database is updated |

**Contract CO35: editFood**

| | |
|---|---|
| **Operation:** | editFood(id, name, category, nutrition) |
| **Cross References:** | Use Cases: 6.1 Manage Food Types |
| **Preconditions:** | ● Data has been loaded from the database<br>● The info is valid<br>● The id refers to an already registered food |
| **Postconditions:** | ● System updates food in list of foods<br>● Database is updated |

**CO36: removeFood**

| | |
|---|---|
| **Operation:** | removeFood(id) |
| **Cross References:** | Use Cases: 6.1 Manage Food Types |
| **Preconditions:** | ● Data has been loaded from the database<br>● The id refers to an already registered food |
| **Postconditions:** | ● System removes given food from the list of food<br>● Database is updated |

**CO37: searchFoodsByName**

| | |
|---|---|
| **Operation:** | searchFoodsByName(query) |
| **Cross References:** | Use Cases: 4.1 Manage Shopping List |
| **Preconditions:** | ● Data has been loaded from the database |
| **Postconditions:** | ● List of food items starting with item name are returned or an empty list if no pantry items match the itemName |