

STARTUP CONTRACTS

Contract CO1: register

Operation:	register(name, info)
Cross References:	Use Cases: 5.1 Startup
Preconditions:	<ul style="list-style-type: none">● Application is running
Postconditions:	<ul style="list-style-type: none">● A user instance has been instantiated● User attributes have been initialized (name, opt. weight, gender)● 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:	<ul style="list-style-type: none">● User has been registered within the system and initialized● Data has been loaded from the database
Postconditions:	<ul style="list-style-type: none">● 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:	<ul style="list-style-type: none">● Data has been loaded from the database● Food already exists within the food database
Postconditions:	<ul style="list-style-type: none">● Added food item exists within the user's pantry● Pantry quantity of food item has been incremented to quantity + old quantity● Pantry database updated

Contract CO4: editPantryItem

Operation:	editPantryItem(foodId, quantity)
Cross References:	Use Cases: 1.1 Manage Pantry
Preconditions:	<ul style="list-style-type: none">• Data has been loaded from the database• Food exists within the food database
Postconditions:	<ul style="list-style-type: none">• Food item is updated with new quantity• Pantry database is updated

Contract CO5: removePantryItem

Operation:	removePantryItem(foodId)
Cross References:	Use Cases: 5.1 Startup, 1.1 Manage Pantry
Preconditions:	<ul style="list-style-type: none">• Data has been loaded from the database• An item of the given food type exists within the pantry
Postconditions:	<ul style="list-style-type: none">• Food item with given name has its instance removed from pantry• Pantry database is updated

Contract CO6: searchPantryByFoodName

Operation:	searchPantryByFoodName(query)
Cross References:	Use Cases: 1.2 Search Pantry
Preconditions:	<ul style="list-style-type: none">• Data has been loaded from the database
Postconditions:	<ul style="list-style-type: none">• 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:	<ul style="list-style-type: none">• Data has been loaded from the database• FoodItem passed in exists in the pantry
Postconditions:	<ul style="list-style-type: none">• Food in pantry with type foodItem has its quantity decremented• If new quantity is 0, foodItem is removed from pantry• 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:	<ul style="list-style-type: none">• Data has been loaded from the database
Postconditions:	<ul style="list-style-type: none">• System provides user with their shopping list

Contract CO9: addShoppingItem

Operation:	addShoppingItem(foodId, quantity)
Cross References:	Use Cases: 4.1 Manage Shopping List
Preconditions:	<ul style="list-style-type: none">• Data has been loaded from the database• The food type exists within the list of registered food types• The quantity is valid (non-negative, etc)
Postconditions:	<ul style="list-style-type: none">• Food item with given quantity added to the shopping list• Database is updated

Contract CO10: editShoppingItem

Operation:	editShoppingItem(foodId, quantity)
Cross References:	Use Cases: 4.1 Manage Shopping List
Preconditions:	<ul style="list-style-type: none">• Data has been loaded from the database• The food type exists within the shopping list• The quantity is valid (non-negative, etc)
Postconditions:	<ul style="list-style-type: none">• Quantity of given food item updated within the shopping list• Database is updated

Contract CO11: removeShoppingItem

Operation:	removeShoppingItem(foodId)
Cross References:	Use Cases: 4.1 Manage Shopping List
Preconditions:	<ul style="list-style-type: none">• Data has been loaded from the database• The food type exists within the shopping list• The quantity is valid (non-negative, etc)
Postconditions:	<ul style="list-style-type: none">• Food item removed from shopping list• 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:	<ul style="list-style-type: none">• Data has been loaded from the database• User has access to the given destination• Given destination is valid• File format is valid
Postconditions:	<ul style="list-style-type: none">• Shopping list has been exported to desired format

Contract CO13: purchaseItems

Operation:	purchaseItems(foodIds)
Cross References:	Use Cases: 4.3 Mark Purchased Items
Preconditions:	<ul style="list-style-type: none">• Data has been loaded from the database• Given foods are registered in the list of foods• User purchased given items from the shopping list
Postconditions:	<ul style="list-style-type: none">• System moves given items from shopping list to pantry• 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:	<ul style="list-style-type: none">• Data has been loaded from the database
Postconditions:	<ul style="list-style-type: none">• 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:	<ul style="list-style-type: none">• Data has been loaded from the database
Postconditions:	<ul style="list-style-type: none">• Ingredients have been stored as food instances within the database• The recipe represents a new recipe instance within the recipe system• Recipe appears within the recipe list

Contract CO16: produceCookedRecipe

Operation:	produceCookedRecipe(recipeId, isUsePantry, consumedServings, leftoverServings)
Cross References:	Use Cases: 3.6 Cook Recipe
Preconditions:	<ul style="list-style-type: none">• Recipe exists within the recipe system• Ingredients exist within the food database• Data has been loaded from the database
Postconditions:	<ul style="list-style-type: none">• User is notified of any errors preventing cooking• Consumed portion is logged to food log• Pantry is updated to contain leftovers• Database is updated

Contract CO17: editRecipe

Operation:	editRecipe(recipeId, recipe)
Cross References:	Use Cases: 3.4 Modify Recipe
Preconditions:	<ul style="list-style-type: none">• Data has been loaded from the database• recipeId is valid
Postconditions:	<ul style="list-style-type: none">• User is notified of any errors preventing modification• Recipe is saved to Recipe List• Database is updated

Contract CO18: getRecipe

Operation:	getRecipe(recipeId)
Cross References:	Use Cases: 3.5 Recommend Recipes
Preconditions:	<ul style="list-style-type: none">• Data has been loaded from the database• Recipe exists within the recipe system
Postconditions:	<ul style="list-style-type: none">• Recipe is displayed to the user

Contract CO19: getRecommendRecipes

Operation:	getRecommendRecipes()
Cross References:	Use Cases: 3.5 Recommend Recipe
Preconditions:	<ul style="list-style-type: none">• Data has been loaded from the database
Postconditions:	<ul style="list-style-type: none">• 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:	<ul style="list-style-type: none">• Data has been loaded from the database
Postconditions:	<ul style="list-style-type: none">• 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:	<ul style="list-style-type: none">• Data has been loaded from the database
Postconditions:	<ul style="list-style-type: none">• Recipe ingredients added to shopping cart• Database updated

Contract CO22: addMissingIngredientsToCart

Operation:	addMissingIngredientsToCart(recipeId)
Cross References:	Use Cases: 3.3 Add Recipe to Shopping Cart
Preconditions:	<ul style="list-style-type: none">• Data has been loaded from the database
Postconditions:	<ul style="list-style-type: none">• Recipe ingredients not in pantry added to shopping cart• Database updated

NUTRITION LOG CONTRACTS**Contract CO23: addItem**

Operation:	addItem(foodId, quantity, consumedAt)
Cross References:	Use Cases: 3.6 Cook Recipe, 2.1 Manage Nutrition Log
Preconditions:	<ul style="list-style-type: none">• Data has been loaded from the database• Food exists within the recipe system• Quantity is valid (non-negative, valid unit, etc.)
Postconditions:	<ul style="list-style-type: none">• Food of given quantity logged to nutrition log at given time• Nutrition log is updated with new nutrition instance• Database is updated

Contract CO24: getNutritionItems

Operation:	getNutritionItems(startDate, endDate)
Cross References:	Use Cases: 2.1 Manage Nutrition Log
Preconditions:	<ul style="list-style-type: none">• Data has been loaded from the database
Postconditions:	<ul style="list-style-type: none">• 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:	<ul style="list-style-type: none">• Data has been loaded from the database• Quantity is valid (non-negative, valid unit, etc.)
Postconditions:	<ul style="list-style-type: none">• The selected item has had its values changed as per user request• Database is updated

Contract CO26: removeNutritionItem

Operation:	removeNutritionItem(foodId, consumedAt)
Cross References:	Use Cases: 2.1 Manage Nutrition Log
Preconditions:	<ul style="list-style-type: none">• Data has been loaded from the database• The given nutrition item is already in the food log
Postconditions:	<ul style="list-style-type: none">• The selected item has been removed from the log• Database is updated

Contract CO27: addGoal

Operation:	addGoal(nutrient, quantity, goalType)
Cross References:	Use Cases: 2.2 Nutrition Goals
Preconditions:	<ul style="list-style-type: none">• Data has been loaded from the database• The input values are valid and logical
Postconditions:	<ul style="list-style-type: none">• A nutrition goal has been created using the given fields• Database is updated

Contract CO28: editGoal

Operation:	editGoal(id, nutrient, quantity, goalType)
Cross References:	Use Cases: 2.2 Nutrition Goals
Preconditions:	<ul style="list-style-type: none"> • Data has been loaded from the database • The input values are valid and logical
Postconditions:	<ul style="list-style-type: none"> • The selected goal has had its values changed as per user request • Database is updated

Contract CO29: removeGoal

Operation:	removeGoal(id)
Cross References:	Use Cases: 2.2 Nutrition Goals
Preconditions:	<ul style="list-style-type: none"> • Data has been loaded from the database • The given goal exists within the user's list of goals
Postconditions:	<ul style="list-style-type: none"> • The selected goal has been removed • Database is updated

Contract CO30: addReport

Operation:	addReport(startDate, endDate)
Cross References:	Use Cases: 2.3 Manage Nutrition Report
Preconditions:	<ul style="list-style-type: none"> • The user is in the nutrition reports menu • The input values are valid and logical
Postconditions:	<ul style="list-style-type: none"> • A nutrition report has been created using the given fields

Contract CO37: editReport

Operation:	editReport(id, startDate, endDate)
Cross References:	Use Cases: 2.3 Manage Nutrition Report
Preconditions:	<ul style="list-style-type: none"> • The user has selected a report to be edited • The input values are valid and logical
Postconditions:	<ul style="list-style-type: none"> • A nutrition report has been edited

Contract CO31: removeReport

Operation:	deleteReport(id)
Cross References:	Use Cases: 2.3 Manage Nutrition Report
Preconditions:	<ul style="list-style-type: none"> • The user is in the nutrition report menu • At least one report exists within the menu which the user wants to delete
Postconditions:	<ul style="list-style-type: none"> • The selected report has been removed from the report menu

FOOD CONTRACTS

Contract CO32: 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:	<ul style="list-style-type: none">• Data has been loaded from the database
Postconditions:	<ul style="list-style-type: none">• System provides user with the food of the given id

Contract CO33: 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:	<ul style="list-style-type: none">• Data has been loaded from the database• The name is not already in use by another food• The name is not empty• The nutrition info is valid (non-negative macros, etc)
Postconditions:	<ul style="list-style-type: none">• System adds food to list of foods• Database is updated

Contract CO34: editFood

Operation:	editFood(id, name, category, nutrition)
Cross References:	Use Cases: 6.1 Manage Food Types
Preconditions:	<ul style="list-style-type: none">• Data has been loaded from the database• The info is valid• The id refers to an already registered food
Postconditions:	<ul style="list-style-type: none">• System updates food in list of foods• Database is updated

CO35: removeFood

Operation:	removeFood(id)
Cross References:	Use Cases: 6.1 Manage Food Types
Preconditions:	<ul style="list-style-type: none">• Data has been loaded from the database• The id refers to an already registered food
Postconditions:	<ul style="list-style-type: none">• System removes given food from the list of food• Database is updated

CO36: searchFoodsByName

Operation:	searchFoodsByName(query)
Cross References:	Use Cases: 4.1 Manage Shopping List
Preconditions:	<ul style="list-style-type: none">• Data has been loaded from the database
Postconditions:	<ul style="list-style-type: none">• List of food items starting with item name are returned or an empty list if no pantry items match the itemName