Wiki

This Wiki explains the uses of the functions that can be used for the GUI, allowing for the GUI programmer to quickly find what function they need and how it can be used in order to speed up development

UserManager

OVERVIEW:

This class contains code relating to the login flow

---

getUserProfileImage

OVERVIEW:

Used for getting a users image

TAKES :

1. String id

   This is the user id that you want the image for

RETURNS :

1. Image

   This is the users image if they have one if not it returns the default image

---

getUserProfiles

OVERVIEW:

Used for getting the users names, images and ids

TAKES :

RETURNS :

1. List<DataStructures.StringImageIdPair>

   This is a custom data class which contains **String** username, **String** userId, **Image** image . Variables can be retrieved via **getText(), getId(), getImage()**

---

getNumberOfUserProfiles

OVERVIEW:

Used for getting the number of users in the system

TAKES :

RETURNS :

1. Int numberOfProfiles

This is the number of profiles

---

moveNewUserProfileToUserFolder

OVERVIEW:

Used for when a new user is created to move their profile picture If added form temporary folder to the user folder

TAKES :

1. String userId

RETURNS :

---

clearTemporaryProfileImageHolder

OVERVIEW:

When 'Create New User' is selected in the login menu or at the start of the program, it should clear the temporary image if the last time the program was run the user creation process wasn't completed, but an image was uploaded.

TAKES :

RETURNS :

---

getUserJsonPath

OVERVIEW:

Used for getting the path to the specified users json

TAKES:

1. **String** userID

RETURNS:

1. **String** userJsonPath

---

changeUserPassword

OVERVIEW:

TAKES:

1. **User** user , **String** currentPassword, **String** newPassword

RETURNS:

1. Boolean

If the current passwords don't match False , else True and the password was changed

---

deleteUserFolder

OVERVIEW:

Used for deleting a specific user folder

TAKES :

1. **String** userId

RETURNS :

---

changeUserPassword

OVERVIEW:

Used for changing a users password

TAKES :

1. **User** user, **String** currentPassword, **String** newPassword

RETURNS :

1. Boolean

Changed or not based on if the currentPassword matched the one in the system

Tools

OVERVIEW:

Contains general functions that can be used throughout the program

---

getRecipeImage

OVERVIEW:

Used for getting the image for a recipe

TAKES:

1. (**String** recipeName)

This image name can be gotten by using recipe.generateImageName();

RETURNS:

1. **Image**

---

linkTagAndDietIcons

OVERVIEW;

Used for when displaying a recipe (**DO NOT USE when displaying the partial information in the recipe search function only when showing the whole recipe page**): pairs the diets and tags with their related icons

TAKES:

1. (**Recipe** recipe)

   This is the recipe that you want to display

RETURNS:

1. **List<DataStructures.StringImagePair>**

   This is a custom data class which contains **String** text (the diet/tag), **Image** image . Variables can be retrieved via **getText(), getImage()**

---

loadImage

OVERVIEW:

This function is used for loading in a user selected **image** and storing it in the given **path**

TAKES :

1. (**String** userId, **BufferedImage** image)

   This is used when using JFrame to get the image via directory

2. (**String** imagePath)

   This is used when the image is already in the users profile directory

RETURNS :

1. NULL

2. NULL

---

getNumberOfRecipesForSpecificDiet

OVERVIEW;

TAKES:

1. (**String** diet, **ArrayList<Recipe>** recipes)

   diet : (Keto, Mediterranean, Paleo, Vegan, Vegetarian, Pescatarian)

RETURNS:

1. **Int**

---

getAverageCaloriesPerMealForDiet

OVERVIEW;

TAKES:

1. (**String** diet, **ArrayList<Recipe>** recipes)

   diet : (Keto, Mediterranean, Paleo, Vegan, Vegetarian, Pescatarian)

RETURNS:

1. **Int**

---

getRangeOfCaloriesForDiet

OVERVIEW;

TAKES:

1. (**String** diet, **ArrayList<Recipe>** recipes)

   diet : (Keto, Mediterranean, Paleo, Vegan, Vegetarian, Pescatarian)

RETURNS:

1. **ArrayList<Double> range**

   min and max calories

---

getAllIngredients

OVERVIEW;

TAKES:

1. (**ArrayList<Recipe>** recipes)

RETURNS:

1. **ArrayList<DataStructures.StringPair> ingredients**

Var1 contains the name of the ingredient , **Var2** contains the measurement if it has one

Search

OVERVIEW:

Contains the two search functions

---

getRecipeSearchResults

OVERVIEW:

Used for searching via the name, level or diet of a recipe/s

TAKES:

1. (**String** query, **ArrayList<Recipe>** recipeHolder, **ArrayList<Integer>** userSavedRecipes, **String** userSavedDiet, **Boolean** showAllRecipes)

showAllRecipes is for if a user wants to see all recipes not just ones on their diet

RETURNS:

1. **ArrayList<Recipe>** returnedRecipes

---

getIngredientSearch

OVERVIEW:

Used for searching via ingredients

TAKES:

1. (**ArrayList<Recipe>** recipeHolder, **ArrayList<DataStructures.StringIntPair>** ingredientQuery, **String** userSavedDiet, **Boolean** showAllRecipes)

showAllRecipes is for if a user wants to see all recipes not just ones on their diet

RETURNS:

1. **ArrayList<Recipe>** returnedRecipes