Milestone 4 | CIS 550

Pastry Dataset

Route 1: /getPastry

Description: Returns information about a recipe, specified by recipeName

Route Parameter(s): None

Query Parameter(s): recipeName (string)

Route Handler: getPastry(req, res)

Return Type: JSON

Return Parameters : {results

(JSON array of { RecipeName (string), Ingredients (string), Directions (string), RecipePhoto (string), TotalTime(int) }

Expected (Output) Behavior:

- If the RecipeName(full name) is found return the singleton array of all the attributes available, but if the RecipeName is a string but is not found, return an empty array as 'results' without causing an error
- Values like TotalTime might be NULL for some entries - return them as is.

Route 2: /search/pastry

Description: Returns an array of selected attributes for recipes that match the search query

Route Parameter(s): None

Query Parameter(s):

- recipeName(string)*

- ingredient(string)*
- totalTimeLow(int)*(default: 0)
- totalTimeHigh(int)*(default: 1000)
- ratingLow(int)*(default: 0)
- ratingHigh(int)*(default: 5)
- page (int)*
- pagesize (int)(default: 10)*

Route Handler: searchPastry(req, res)

Return Type: JSON

Return Parameters: {results (JSON array of { recipeName(string), Author (string), PrepareTime

(int), CookTime(int), TotalTime(int), AVGRating(float), Total_Review(int) }) }

Expected (Output) Behavior:

- Return an array with all recipes that match the constraints. If no recipe satisfies the constraints, return an empty array as ‘results’ without causing an error
- The behavior of the route with regard to page and, pagesize is:
    - Case 1: If the page parameter (page) is defined
        - Return match entries with all the above return parameters for that page number by considering the page and pagesize parameters. For example, page 1 and page 7 for a page size 10 should have entries 1 through 10 and 61 through 70 respectively.
    - Case 2: If the page parameter (page) is not defined
        - Return all match entries with all the above return parameters.
- xHigh and xLow are the upper and lower bound filters for an attribute x. Entries that match the ends of the bounds should be included in the match. For example, if ratingLowwere 1 and ratingHigh were 5, then all recipes whose AVGRatingwas >=1 and <=5 would be included.
- Values like TotalTime might be NULL for some entries - return them as is.
- Alphabetically sort the results by the recipeName attribute

---

Route 3: /getPastryReview

Description: Returns reviews about a recipe, specified by recipeName

Route Parameter(s): None

Query Parameter(s): recipeName (string)

Route Handler: getPastryReview(req, res)

Return Type: JSON

Return Parameters : {results(JSON array of { Comment(string), Rate(int) }

Expected (Output) Behavior:

- If the RecipeName(full name) is found return the singleton array of all the attributes available, but if the RecipeName is a string but is not found, return an empty array as 'results' without causing an error.

Route 4: /clearFridge

Description: Returns recipes that consume largest number of ingredients that the user wants to use. Possibly we can return recipes that consume the top 5 largest number of ingredients

Route Parameter(s): None

Query Parameter(s): ingredients (array of string)

Route Handler: clearFridge (req, res)

Return Type: JSON

Return Parameters : {results

(JSON array of { RecipeName (string), Ingredients (string), Directions (string), RecipePhoto (string), TotalTime(int) }

Expected (Output) Behavior:

- If the recipes which are satisfied the conditions are found, return the singleton array of all the attributes available, but if not recipes are fond, return an empty array as 'results' without causing an error

- Values like TotalTime might be NULL for some entries - return them as is.

MainDishes

Route 5: /getDish

Description: Returns cook information about a recipe, specified by recipe name.

Route Parameter(s): None

Query Parameter(s):

- Name (string)

Route Handler: getPastry(req, res)

Return Type: JSON

Return Parameters: {results (JSON array of { recipeName(string), Author (string), IngredientName(string), CookSteps(string),Cookminutes(int),Calorie(float),Carbohydrates(float),Protein(float),SaturatedFat(float),Sodium(float),Sugar(float),TotalFat(float) }) }

Expected (Output) Behavior:

- If the RecipeName(full name) is found return the singleton array of all the attributes available, but if the RecipeName is a string but is not found, return an empty array as 'results' without causing an error
- The behavior of the route with regard to page and, pagesize is:
    - Case 1: If the page parameter (page) is defined
        - Return match entries with all the above return parameters for that page number by considering the page and pagesize parameters. For example, page 1 and page 7 for a page size 10 should have entries 1 through 10 and 61 through 70 respectively.
    - Case 2: If the page parameter (page) is not defined
        - Return all match entries with all the above return parameters.
- Values like TotalTime might be NULL for some entries - return them as is.
- Alphabetically sort the results by the recipeName attribute

---

Route 6: /searchDish

Description: Returns an array of selected attributes for recipes that match the search query.

Route Parameter(s): None

Query Parameter(s):

- recipeName(string)
- page (int)
- pagesize (int)(default: 10)

Route Handler: searchDishes(req, res)

Return Type: JSON

Return Parameters: {results (JSON array of { RecipeId(int), Name (string), Rate(float)}) }

Expected (Output) Behavior:

- Return an array with all recipes that match the constraints. If no recipe satisfies the constraints, return an empty array as 'results' without causing an error

---

Route 7: /review

Description: Returns an array of reviews for recipes that match the search query according to the recipe name(or recipe ID) and can filter certain reviews according to the given keywords.

Route Parameter(s): None

Query Parameter(s):

- recipeName(string)
- keyword(string) by default null
- page (int)
- pagesize (int)(default: 10)

Route Handler: searchReview(req, res)

Return Type: JSON

Return Parameters: {results (JSON array of { Name (string), ReviewContent(string)}) }

Expected (Output) Behavior:

- Return an array with all recipes that match the constraints. If no recipe satisfies the constraints, return an empty array as 'results' without causing an error

- Reviewcontent might be NULL for some entries - return them as is.

- Alphabetically sort the results by the recipeName attribute.

Route 8: /user

Description: Returns an array of recipes that have been reviewed by this user, associated with recipeId, recipeName, rate(he gave), AVGRating(total average rating), reviewContent. Also provide the favorite ingredients of this user as well as a distribution for the rate he has given(review behavior).

Route Parameter(s): userId

Query Parameter(s): None

Route Handler: getUser(req, res)
Return Type: JSON
Return Parameters: {results (JSON array of {
RecipeId(int), RecipeName (string), Rating(int), AVGRating(float), ReviewContent(string), FavoriteIngredients(Array(string)), Rating0(int), Rating1(int),Rating3(int), Rating4(int),Rating5(int)
})}

Expected (Output) Behavior:
• Return an array with all recipes that have been reviewed by the user. If no recipe satisfies the constraints, return an empty array as 'results' without causing an error
• As for FavoriteIngredients(Array(string)), it counts the number of times per ingredient which have appeared in those recipes that have been rated 4 and above by

this user and at the same time have AVGRating higher than 3.5. By default, return 10 of them.

  • As for Rating0(int), Rating1(int), Rating3(int), Rating4(int), Rating5(int), they stand for how many rates the user gives as 0, 1, 2… respectively.

Route 9: /recommendationByUser

Case1: given user, recommend similar users who have most similarities, sorting by the count decreasingly. Define the similarity as the weighted sum of abs(rating_by_user1-rating_by_user2) among the common recipes shared by user1 and user2.

Route Parameter(s): userId

Query Parameter(s): None

Route Handler: getRecommendationByUser (req, res)
Return Type: JSON
Return Parameters: {results (JSON array of {
  UserId(int), Username (string), Similarity(float)
})}

Expected (Output) Behavior:
  • Return an array with all users that have been behaved similarly as the given user. If no user satisfies the constraints, return an empty array as ‘results’ without causing an error. The result will be sorted by similarity decreasingly.

Case2: given user, recommend an array of similar recipes which consist of as many as his favorite ingredients.

Route Parameter(s): userId

Query Parameter(s): None

Route Handler: getRecommendationByUser (req, res)
Return Type: JSON
Return Parameters: {results (JSON array of {
    RecipeId(int), RecipeName(string), CommonIngredientsCount(int)
})}

Expected (Output) Behavior:
    • Return an array with all recipes that have as many as favorite ingredients of the given user. If no user satisfies the constraints, return an empty array as ‘results’ without causing an error. The result will be sorted by CommonIngredientsCount decreasingly.

Route 10: /recommendationByRecipe

Case1: Given recipe, recommend similar recipes who have the most similarities, sorting by the count decreasingly. Define the similarity as the weighted sum of the same ingredient among recipe (count of the total number of same ingredients among two recipes/(max(the total amount of the ingredients in the recipe)))

Route Parameter(s): userId

Query Parameter(s): None

Route Handler: getRecommendationByRecipe (req, res)
Return Type: JSON
Return Parameters: {results (JSON array of {
    RecipeId(int), Recipename (string), Similarity(float)
})}

Expected (Output) Behavior:
    • Return an array with all recipes that contains high collection of the similar ingredients as the given user. If no recipe satisfies the constraints, return an empty array as ‘results’ without causing an error. The result will be sorted by similarity decreasingly.

Case2: given recipe, recommend an array of similar recipes which consist of as many as this recipes ingredients.

Route Parameter(s): reciperId

Query Parameter(s): None

Route Handler: getRecommendationByRecipe (req, res)
Return Type: JSON
Return Parameters: {results (JSON array of {
RecipeId(int), RecipeName(string), CommonIngredientsCount(int)
})}

Expected (Output) Behavior:
• Return an array with all recipes that have as many as favorite ingredients of the given user. If no user satisfies the constraints, return an empty array as ‘results’ without causing an error. The result will be sorted by CommonIngredientsCount decreasingly.

Route 11: /rankSimilarRecipe

Description: Rank given recipes (up to 5) by similarity in nutritious with a given recipe, specified by recipeName

Route Parameter(s): None

Query Parameter(s): recipeName (string), recipeList (array of string)

Route Handler: rankSimilarRecipe(req, res)

Return Type: JSON

Return Parameters : {results

(JSON array of { RecipeName (string), Author(string), Ingredients (string), Directions (string), TotalTime(int)})

Expected (Output) Behavior:

- If the given recipeList (second parameter) is not empty, return these recipes in the order of similarity in nutritious with a given recipe (first parameter)
- Values like TotalTime might be NULL for some entries - return them as is.