I have generated ER Diagram on MySQL Workbench and I have added cardinality to it.

It is the one and only one on the left side because food items can have only one author – the user, on the right side is zero or many because the user can have as many food items as the user wants but also the user doesn't need to have any.

Diagram

Description automatically generated

Below I wrote down everything I did from the tasks in each page and in which lines they are located. More specifically what I did step by step is explained in the comments in main.js.

//////////HOME PAGE////////////

The home page does its purpose by giving links to other pages and displaying the name of the application when the user can explore every page of the website

R1A: The name of the web application is displayed in the top left corner – index.ejs line 8

R1B:  Links to other pages are displayed on the main page – index.ejs lines from 9 to 16.

////////////ABOUT PAGE//////////////

R2A: This page displays the information about the web application including my name as the developer and link to the home page – about.ejs lines 9-11

///////////REGISTER PAGE/////////////////

R3A: This page displays a form to the user where the user can add their first name, last name, e-mail address, username, and password – register.ejs lines 9-16 and link to the home page – register.ejs line 17

R3B:   To provide security of data in storage, only a hashed password is saved in the database, not a plain password. – main.js lines 96-100. Collecting form data to be passed to the database and storing user data in the database – main.js lines 85-117

R3C: Displaying a message indicating that add operation has been done – main.js lines 111-114

///////////LOGIN PAGE///////////////////

R4A: Displaying a form for users to log in to the dynamic web application. The form consists of username and password. - login.ejs lines 9-13 Displaying a link to the home page – login.ejs line 14

R4B: Collecting form data to be checked against data stored for each registered user in the database. Users are logged in if and only if both username and password are correct – main.js lines 34-50

R4C: Display a message indicating whether login is successful – main.js line 46 or not and why not successful – main.js line 49.

/////////////////LOGOUT PAGE////////////

R5: When the user clicked the log-out link on the main page, the user is sent to the home page, and the message is shown to confirm that the user has logged out - main.js lines 70-78

//////////////////ADD FOOD PAGE/////////////////

**R6A**: Displaying a form to users to add a new food item and all the nutrition information to the database. – addfood.ejs lines 9-19.

I also added back-end form validation to be sure that the input from the user is correct such as carbs, fat, sugar, salt, protein, and the value contains only numbers and names only the letters – main.js lines 127-134

Displaying a link to the home page – addfood.ejs line 20

**R6B**:  Collecting form data to be passed to the database and storing food items in the database – main.js lines 125-159 Saving the username of the user who has added this food item to the database – inserting author as req.session.userId. – main.js lines 140 and 151.

**R6C**: Display a message indicating that add operation has been done. – main.js line 157

//////////////////SEARCH FOOD PAGE////////////////////

**R7A**: Displaying a form (contains only a field to input the food’s name) to users to search for a food item in the database. – search.ejs lines 9-13.

I also added back-end form validation to be sure that the input from the user is correct - name contains only the letters – main.js line 168

Display a link to the home page search.ejs line 14.

**R7B**:  Collecting form data to be passed to the back end and searching the database based on the food name collected from the form. – main.js line 172.

Displaying a list.ejs including data related to the food found in the database to users if a food item is found – main.js line 184.

Displaying a message to the user, if not found. – main.js line 182

***R7C***: Searching food items - it is searching for a keyword (input from the user) in the database and if the user searching for bread all related data will be displayed such as wholemeal bread, pitta bread, etc. – main.js lines 172 - 185

////////////////////UPDATE FOOD PAGE//////////////////////

 only available to logged-in users – redirectLogin – main.js lines 206

**R8A**: Displaying search food form (same as on the search page) – search-update.ejs lines 9-13

Displaying a link to the home page – search-update.ejs lines 9-13, updatefood.ejs line 41

**R8B**: Displaying all data related to the food found in the database to users in forms so users can update each field. –updatefood.ejs lines 17-30 and backend main.js line 216

Displaying a message to the user if the food item is not found. – main.js lines 226

Collecting form data to be passed to the database and storing updated food items in the database. – main.js lines 270-289

Display a message indicating the update operation has been done – main.js lines 285

Letting ONLY the user who created the same food item update it – main.js lines 258-268

**R8C**: Implementing a delete button to delete the whole record – updatefood.ejs line 33

when the delete button is pressed, then 'Are you sure?' question appears -updatefood.ejs script – lines 44-55

Deleting the food item from the database – updatefood.ejs lines 34-37 and back-end main.js line 312

Display a message indicating the delete has been done – main.js lines 320

Letting ONLY the user who created the same food item delete it – main.js lines 300-310

/////////////////////////LIST FOOD PAGE/////////////////////

**R9A**: Displaying all fields for all foods stored in the database. – main.js line 332 and listfood.ejs lines 29-44

Displaying a link to the home page -listfood.ejs line 97

**R8B**: The list is organised in a tabular format – implemented in listfood.ejs lines 29-44

***R9C:*** Displaying a checkbox next to each food item – listfood.ejs line 40

Letting the user input the amount of each food item in the recipe – listfood line 41

Collecting the name of all selected foods and calculating the sum of the nutritional information related to all selected food items for a recipe or a meal – listfood.ejs script lines 59-95

Displaying calculated information as ‘nutritional information of a recipe or a meal’ – listfood.ejs lines 50-55

/////////////////////////API/////////////////////////////////

GET

/api There is a basic API displayed on '/api' route listing all foods stored in the database in JSON format.

/api/id/:id

There is an API listing food with the exact same id provided from the input

/api/name/:name

Listing food with the exact same name provided may list food with the same name for example for bread – whole bread, white bread, etc.

/api/author/: author

Listing food with the exact same author provided may list all food items with the same author

POST -There is an API displayed on ‘’/api/post’’ where the user can add a new food item

DELETE - There is an API displayed on ‘’/api/id/:id where the user can delete food item

PATCH - There is an API displayed on ‘’/api/id/:id where the user can update food items. I used COALESCE to allow the user to change only one of the nutritional information, the user does not have to enter all of them, just the one which wants to change.