Databases, Network and the Web Coursework for Midterm:

CalorieBuddy

Introduction

In the course, we developed a small dynamic web application including routes, forms, and database access. In this assignment you are tasked with creating another dynamic web application that will function as a digital calorie counter to help users manage their diet. Essentially the dynamic web application interacts with users to calculate and display nutritional facts including calories for their recipes or meals based on food ingredients in the recipe or the meal.
The user should be given the choice to add nutritional facts for food ingredients and to store them in the database. The web application should provide users with the sum of the calories for a recipe or a meal based on the food ingredients selected by the user to be included in the recipe or the meal. As an example, the user should be able to select the food ingredients for the recipe of 'apple pie' by choosing 'flour', 'egg', 'butter', 'brown sugar', and 'apple' and then the web application retrieves the nutritional facts for each food ingredient and calculates and displays the nutritional facts including calorie count of the 'apple pie' recipe. If a food ingredient is not in the database the user should be given the choice to add it to the database. A full list of the functionalities of the web application is explained in the next sections of this document. Some requirements are 'base' requirements to pass the assignment and some requirements are 'stretch goals' indicated as 'going beyond' and are designed for students who would like to achieve the full mark.

Working Environment

For the purpose of this assignment, we have setup a working directory called "mid-term" inside the "topic7" folder of your labs. At the moment the folder contains a dummy "index.js" file which creates a web server on PORT 8089. Open this lab and take a look at the folder structure.
You can see the root route of your application by visiting the following URL: "<Your lab url>" + "/topic7/mid-term/"
For the application to work correctly, you should not move the "index.js" file from its current directory. Furthermore, the application must run on PORT 8089.

Requirements

The purpose of the web application is to help people manage their diet by displaying nutritional facts including calories, carbs, fat, protein, salt and sugar in a recipe based on food ingredients in the recipe. The web application provides several functionalities that should meet the following requirements:

- R1: Home page:

    - R1A: Display the name of the web application.
    - R1B: Display links to other pages or a navigation bar that contains links to other pages.
    - Students can go beyond this requirement by implementing R2
- R2: About page: (only for students who would like to go beyond)
    - R2A: Display information about the web application including your name as the developer. Display a link to the home page or a navigation bar that contains links to other pages.
- R3: Add food page:
    - R3A: Display a form to users to add a new food item to the database. The form should consist of the following items: name, typical values, unit of the typical value, calories, carbs, fat, protein, salt, and sugar. Display a link to the home page or a navigation bar that contains links to other pages.
    - R3B: Collect form data to be passed to the back-end (database) and store food items in the database. Each food item consists of the following fields: name, typical values, unit of the typical value, calories, carbs, fat, protein, salt, and sugar. Here is an example of a food item:

      name: flour, typical values per:100, unit of the typical value: gram, calories: 381 kilocalories, carbs: 81 g, Fat: 1.4 g, Protein: 9.1 g, salt: 0.01 g, and sugar: 0.6 g. The unit of the typical value may have values such as gram, litre, tablespoon, cup, etc.
    - R3C: Display a message indicating that add operation has been done.
- R4: Search food page
    - R4A: Display a form to users to search for a food item in the database. 'The form should contain just one field - to input the name of the food item'. Display a link to the home page or a navigation bar that contains links to other pages.
    - R4B: Collect form data to be passed to the back-end (database) and search the database based on the food name collected from the form. If food found, display a template file (ejs, pug, etc) including data related to the food found in the database to users; name, typical values, unit of the typical value, calories, carbs, fat, protein, salt, and sugar. Display a message to the user, if not found.
    - R4C: Going beyond, search food items containing part of the food name as well as the whole food name. As an example, when searching for ‘bread’ display data related to ‘pitta bread’, ‘white bread’, ‘wholemeal bread’, and so on.
- R5: Update food page
    - R5A: Display search food form. Display a link to the home page or a navigation bar that contains links to other pages.
    - R5B: If food found, display data related to the food found in the database to users including name, typical values, unit of the typical value, calories, carbs, fat, protein, salt, and sugar in forms so users can update each field. Display a message to the user if not

found. Collect form data to be passed to the back-end (database) and store updated food items in the database. Display a message indicating the update operation has been done.

  - R5C: going beyond by implementing a delete button to delete the whole record, when the delete button is pressed, it is good practice to ask 'Are you sure?' and then delete the food item from the database, and display a message indicating the delete has been done.

- R6: List foods page
  - R6A: Display all foods stored in the database including name, typical values, unit of the typical value, calories, carbs, fat, protein, salt, and sugar, sorted by name.
  - R6B: Display a link to the home page or a navigation bar that contains links to other pages.
  - R6C: going beyond by letting users select some food items (e.g. by displaying a checkbox next to each food item and letting the user input the amount of each food item in the recipe e.g. 2x100 g flour). Then collect the name of all selected foods and calculate the sum of the nutritional information (calories, carbs, fat, protein, salt, and sugar) related to all selected food items for a recipe or a meal and display them as 'nutritional information and calorie count of a recipe or a meal'. Please note, it is not necessary to store recipes or meals in the database.

Code style and technique

Your code should be written according to the following style and technique guidelines:

- C1: Code is organised into java script (.js) files and template files (.html or .ejs or .pug). Java script files contain web server code (index.js) and middleware (main.js in routes folder). Template files (.html and .ejs and .pug) are stored in views folder.
- C2: Each route in main.js has comments describing purpose, inputs, and outputs
- C3: Code is laid out clearly with consistent indenting
- C4: Each database interaction has comments describing the purpose, inputs, and outputs
- C5: Functions and variables have meaningful names, with a consistent naming style

Documentation

You should write a documentation report and include your source code. The submission should contain the following items and information:

- D1: Source code in standard ZIP format
- D2: documentation report is in PDF format
- D3: List of requirements: for each sub-requirement (R1A → R6C) state how this was achieved or if it was not achieved. Explain where it can be found in the code. Use focused, short code extracts if they make your explanation clearer.
- D4: Database structure: tables including purpose, field names, and data types for each table.

Marking criteria

We will mark your work according to the set of criteria shown below, which consider the requirements, your programming technique and style, and the documentation you have provided:

a) Code style and technique:

Category	Criterion	Not addressed [0]	Attempted but did not meet requirements completely [1]	Met requirements completely. [2]	Weight
Code style and technique	C1: Code is organised into java script (.js) files and template files (.html or .ejs or .pug). Java script files contain web server code (index.js) and middleware (main.js in routes folder), template files (.html and .ejs and .pug) are stored in views folder.				2
Code style and technique	C2: Each route in main.js has comments describing purpose, inputs, and outputs				2
Code style and technique	C3: Code is laid out clearly with consistent indenting				2
Code style and technique	C4: Each database interaction has comments describing the purpose, inputs, and outputs				2
Code style and technique	C5: Functions and variables have meaningful names, with a consistent naming style				2
					Total:10

b) Documentation report [30 marks in total]

- D1: Source code in standard ZIP format [2]
- D2: Report is in PDF format [2]
- D3: List of requirements: for each sub-requirement (R1A → R6C) state how this was achieved or if it was not achieved. Explain where it can be found in the code. Use focused, short code extracts if they make your explanation clearer. [each requirement 1 mark, 15 marks in total for D3]
- D4: Database structure: tables including purpose [2 marks], field names [2 marks], and data types for each table [7 marks]. [11 marks in total for D4]

c) Requirements [60 marks in total]

Requirement	Parts of the requirement	Not addressed [0]	One part of the requirement is addressed [4]	Two parts of the requirement are addressed [8]	All three parts of the requirement (A, B and C) are addressed [12]
R1: Home page R2: About page	R1A, R1B R2A (this is for students who would like to go beyond)				
R3: Add food page	R3A, R3B, R3C				
R4: Search food page	R4A, R4B, R4C (going beyond)				
R5: Search food page	R5A, R5B, R5C (going beyond)				
R6: List foods page	R6A, R6B, R6C (going beyond)				