Calorie Tracker

Student: Kenny Hoang

ID: 33786189

Introduction

This is a web application developed to log foods that users have eaten, automatically retrieve calorie information using the USDA API, store the entries then allow the user to view or search logged data.

Features

Home page

- Displays today's logged foods
- Shows calories and time added
- Summarises total calories for the current day
- Contains navigation links to all other pages

About Page

Simple page describing what the application does

Add Food Page

- Contains a form where the user types a food name (e.g., "banana", "pork 200g").
- When submitted:

  The system calls the USDA FoodData Central API.
  Calories per 100g are extracted.
  If a weight is included in the query, calories are scaled appropriately (e.g., 200g → ×2).
  The entry is inserted into the MySQL database.

Search Functionality

- Search entries by partial food name
- Case-insensitive
- Queries directly against MySQL using LIKE
- Results displayed in a table with food name, calories, and timestamps

View by Date

- Displays calories for any chosen day
- A date selector sends `/day?date=YYYY-MM-DD`
- MySQL filters by DATE(created_at)

2.6 Database Storage

The app stores all logged foods in a MySQL database, including:

- Food_name
- Calories
- Created_at

Installation

- npm install installs all dependencies
- create_db.sql builds the database tables
- insert_test_data.sql inserts default users and test foods
- node index.js runs the app on port 8000

API

The USDA FoodData Central API is used to obtain real calorie values. USDA API is used because it is free and has no limits for core use along with just being accurate

Extraction Logic

- Extract “Energy (kcal)” nutrient
- If user provides weight (e.g., “200g”), calories are scaled
- If multiple results exist, the application prioritises:

    1. Exact matches

    2. Processed foods if keywords (“bread”, “chips”) appear

    3. Raw foods for single-word inputs

    4. Close matches

    5. Fallback to the first USDA result

How to Install

Clone the GitHub repository

Run:

```
npm install
```

In MYSQL run create_db.sql and insert_test_data.sql

Create a .env file using

DB_HOST=localhost
DB_USER=calorie_user
DB_PASSWORD=12345
DB_NAME=calorie_app
USDA_API_KEY=A0PIVnLsYTBERvL5Wi0v1WZoXaiBUj1FjUD2qIvS (this is my API key, and im not sure if you need it)

Create `.env` file with:

```
USDA_API_KEY=YOUR_KEY
DB_HOST=localhost
DB_USER=YOUR_USER
DB_PASSWORD=YOUR_PASSWORD
DB_NAME=calorie_app
```

To allow the application to connect to the MySQL database on the server, a dedicated database user must be created. This user is named calorie_user, and it is assigned the password 12345. After creating the user, full permissions are granted so that the application can read and write to the database.

The SQL commands used to set this up are:

Create the user:
 CREATE USER 'calorie_user'@'localhost' IDENTIFIED BY '12345';

Grant the user full privileges on the application’s database:
 GRANT ALL PRIVILEGES ON calorie_app.* TO 'calorie_user'@'localhost';

Apply the privilege changes:
 FLUSH PRIVILEGES;

Conclusion

The application successfully demonstrates the essential skills taught in the Dynamic Web Applications module:

- Building an Express.js application
- Using EJS templates
- Working with MySQL databases
- Validating forms
- Handling POST/GET requests
- Integrating external APIs
- Deploying to Goldsmiths VM
- Providing documentation and installation scripts