HEALTH APP

Dynamic Web Applications – Final Coursework

Student Name: Mercedes Anegbeh

Student ID: 33828328

Module: Dynamic Web Applications

Department: Computing

Institution: Goldsmiths, University of London

Academic Year: 2025–2026

Project Title: *Health App – A Fitness & Nutrition Tracking Web Application*

1. Outline

The Health App is a full-stack wellbeing and fitness tracker designed to help users monitor their daily physical activity and nutrition. It provides a secure login system, a workout logging tool with an integrated timer, an automatic calories-burned calculator using an external API, and a meal logging system for tracking calorie and protein intake. Users can create, edit, and delete workouts, record the duration of their exercises using a live digital timer, and automatically estimate calories burned based on activity type and duration.

The app also supports meal tracking, enabling users to add foods with calorie and protein counts for nutritional monitoring. A search feature allows users to filter workouts or meals by name or activity. All data is stored in a MySQL database and can be viewed through clean, responsive EJS-powered pages.

The application supports the full user journey: logging in, navigating the home and about pages, tracking workouts using custom tools, searching previous entries, and managing personal data securely. It is built to meet the coursework requirements while demonstrating modern web-development practices including validation, sanitisation, session management, and external API integration.

2. Architecture

The Health App follows a classic MVC-style Node.js architecture using Express for routing, EJS for templating, and MySQL for data storage. The routes/ folder contains route controllers for workouts, meals, users, and pages. Views are rendered through EJS templates in the views/ directory. The database is defined using SQL scripts (create_db.sql, insert_test_data.sql) and accessed through a MySQL connection pool. Client-side JavaScript powers dynamic behaviour such as the workout timer and API calls. User sessions are managed using express-session.

Client (Browser)
→ Express (Node.js application handling requests)
→ Route Handlers (workouts, meals, users, timer, search)
→ MySQL Database (users, workouts, meals, audit)
→ EJS Templates (views folder)
→ Response sent back to Client

3. Data Model

The application stores its data in a MySQL database named health, consisting of four primary tables:

- users: id, username, first, last, email, hashedPassword
- workouts: id, name, activity, calories, time, created_at
- meals: id, name, calories, protein

- audit: id, table_name, operation, timestamp

The users table supports authentication, while workouts and meals store structured fitness data for search and logging features. Relationships are simple and independent for ease of marking.

4. User Functionality

The Health App offers a range of user-facing features accessible after logging in. The Home page introduces the app and links to all major sections. The About page explains the purpose of the system and its health-tracking features.

1. Login & Registration

Users sign in using a secure login form. Passwords are hashed using bcrypt and stored safely in the database. Input sanitisation prevents malicious input.

2. Workouts

The Workouts List displays all logged workouts with options to edit or delete entries.

The Add Workout page allows users to manually enter a workout name, choose an activity type, and specify calories and duration.

The Workout Timer page includes a live stopwatch implemented in JavaScript. When the user stops the timer, the total duration is saved and can be submitted as part of a new workout entry.

3. Auto-Calculate Calories

The timer page includes an Auto-Calc Calories button which sends an AJAX request to /workouts/calcCalories. Based on the selected activity and the timer duration, the server fetches calorie burn rates from the API Ninjas Calories API, calculates estimated calories, and automatically fills the calories field. This allows users to log realistic workout data with minimal manual input.

4. Search

The app includes search functionality for both workouts and meals. Users can search by name or activity, with matching records retrieved from MySQL and displayed using EJS templates.

5. Meals

Users may log meals by providing a name, calories, and protein value. All meals are viewable in the Meals List, and searchable via the Search page.

6. Navigation

All pages are linked through a consistent navigation bar providing access to:

Home, About, Workouts, Timer, Meals, Search, Login, Logout.

5. Advanced Techniques

The Health App implements several advanced techniques beyond the core requirements.

External API Integration

The calorie estimation system uses the API Ninjas Calories Burned API. The server sends a request with the selected activity and receives a list of calories burn rates. The backend calculates calories burned using:

```
const calories = (caloriesPerHour / 3600) * durationSeconds;
```

File: routes/workouts.js

This demonstrates asynchronous API usage, JSON handling, and server-side computation.

Workout Timer

A custom JavaScript stopwatch provides real-time workout timing.

```
let totalSeconds = 0;
setInterval(() => {
   totalSeconds++;
   timerDisplay.textContent = format(totalSeconds);
}, 1000);
```

File: views/workouts/timer.ejs

Security Features:

- Password hashing using bcrypt
- Session management with express-session
- Input sanitisation using express-sanitizer

These prevent SQL injection, unsafe input, and plaintext password storage.

Validation

The backend ensures users cannot submit invalid workout data:

```
if (!name || !activity || isNaN(calories) || isNaN(time)) {
   return res.send("Error: Invalid workout data.");
}
```

Clean Architecture

- Separation of routes, views, and SQL scripts
- Reusable layout
- Module structure suitable for interview-level backend development