

Introduction to Modelling and Animation

Mobina Sharafi (33777254)

Final Lab instructions report:

Outline

This application is a full-stack health tracking system designed to give users a structured system with a calm interface, that allows them to gain an understanding on the patterns regarding their health over time. This application allows users to log and monitor their activities and reflect on their long-term goals.

New users can create an account by entering their personal details including age, height, weight and a chosen health goal, once the user is registered, the application calculates their BMI. The BMI value calculated is paired with its own suited classification such as severely underweight, normal, obesity class II, giving user an immediate understanding of their current status.

After logging in, users can add and track health activities, which include specifications such as activity type, duration, caloric expenditure and intensity. There is a built-in link to an External AI system (ChatGPT), that will help users who need to specify their calorie estimates. These records can be searched, sorted (by their date, alphabetically, duration, and calories burnt), edited and deleted, enabling useful interactions with the stored data. The application also includes a personal vision board where users can upload motivational images and articulate their health intentions. Another feature is the diet feature which suggests 3 diets that suit the user the most in order, according to their goal and eating preferences. Diet approaches will not be given if an individual's BMI is categorised as overweight and their goal is to gain weight, same goes for an underweight individual whose goal is to lose weight.

there's also a quick wellness check feature which estimates the likelihood of certain health deficiencies by taking 5-question tests.

Architecture

This web-application uses a three-layered structure:

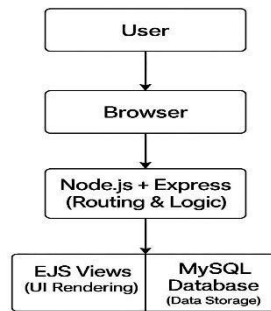
Express runs on Node.js which handles all routing and application logic.

EJS renders the user interface.

MySQL stores the data.

The server is organized into different route files for user account actions, health record management, and the vision board features.

Each route processes requests and performs validations when necessary and interacts with the database through SQL queries. Dynamic pages are generated using EJS templates in the views directory, while static assets and uploaded images are served from public. This clear separation of routing, interface rendering and data storage keeps this application organized and well maintained.



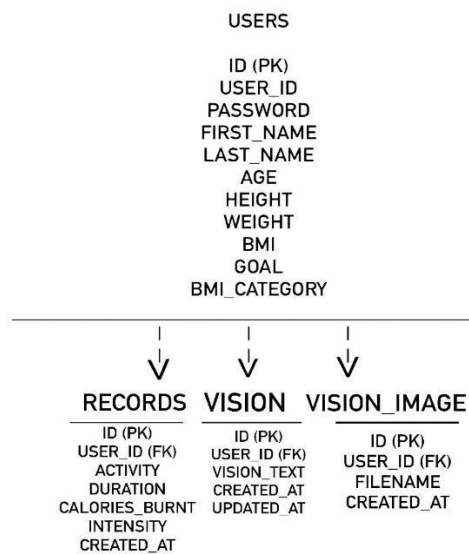
Data Model

This application uses MySQL database with four tables: users, records, vision, and vision_images.

The users table stores account and profile information including personal details (age, height, weight, first name, last name) BMI, goal and BMI category.

The records table stores individual health activity logs linked to a user through user_id. The vision table stores each user's written health vision text with created and updated timestamps.

The vision_images table stores filenames of images uploaded to each user's vision board. All three tables reference users.user_id, creating many relationships from users to their records, vision and images.



User Functionality:

This web application provides a clean and clear interface that supports users manage their health data, track their activities and maintain a personal vision board of their long-term goals. When visiting the site without having logged in, users see a navigation bar with links to Home, About, Login and Register. Once logged in, the navigation bar updates to to Home, About, View My Records, Profile, Vision Board and Logout.

The Home Page welcomes users with a message describing the purpose of the app, followed by inspirational health and fitness images.

[Home](#) [About](#) [View My Records](#) [Profile](#) [Vision Board](#) [Logout](#)

Welcome to the Health App

This application offers a calm, organised way to follow your health over time. As you record your activities and personal details, you create a clearer picture of how your habits shape your overall wellbeing.

Your profile also includes a personal health vision board, a place to gather images and intentions that reflect the life you are working toward. It serves as a small reminder of what inspires you and the direction you want to grow in.



The About Page explains that the project was built for the DWA final assignment and outlines the goals of the application and the technologies used.

[Home](#) [About](#) [Login](#) [Register](#)

About Page

This project was created for the Dynamic Web Applications final assignment, as a way to demonstrate practical skills in building a full-stack web application using Node.js, Express, MySQL, and EJS.

The app helps users track their health habits, understand their personal data more clearly, and organise their goals through features like activity logging, profile management, and a personal Health Vision Board.

Created by Mobina Sharafi, BSc Computer Science, Goldsmiths,
University of London.

Users can create an account through Register Page, where they can enter their personal details such as name, last name, age, height, weight and their health goal (selected from a dropdown or entered manually when "Other" is chosen). Their BMI will be automatically calculated and shown once they are registered as a part of their profile, and they will also be assigned a BMI category. They also create their username and password (the password must be minimum 8 characters long and have at least one alphabetical word and one number). After this, they are shown a welcome message and can now log in through the login page.

Registration form fields:

- First name:
- Last name:
- Age:
- Height (cm):
- Weight (kg):
- Health goal: Choose a
- username:
- Create a password:
-
- [Already have an account? Login](#)

The View My Records page displays all logged health activities. Each entry includes its own data such as activity name, duration, calories burnt, intensity level (0-5), and timestamps. Users can search for specific activities, sort their records (by the following orders: newest, oldest, duration, calories burned, activity A-Z), delete or edit their current records and add new ones. The calorie burned section is optional, however there is a link that allows the users to consult AI to get an estimation of it.

Page: [Home](#) [About](#) [View My Records](#) [Profile](#) [Vision Board](#) [Logout](#)

Your Personal Health Records

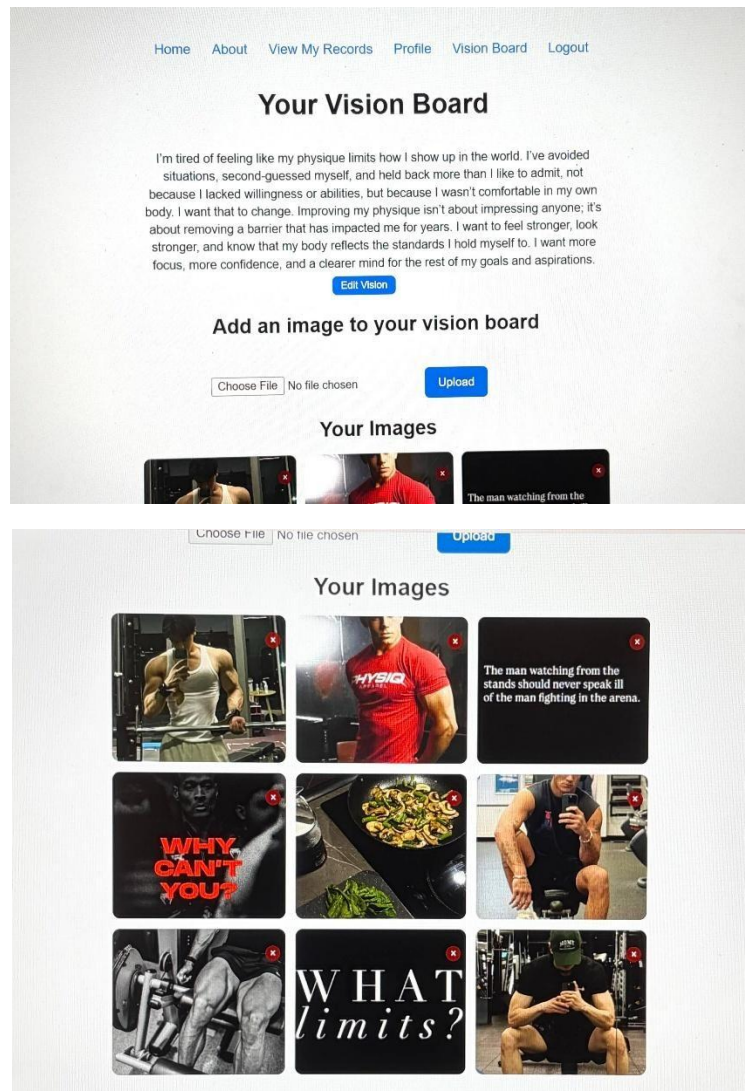
[Add New Record](#)

Search activity:

Sort by:

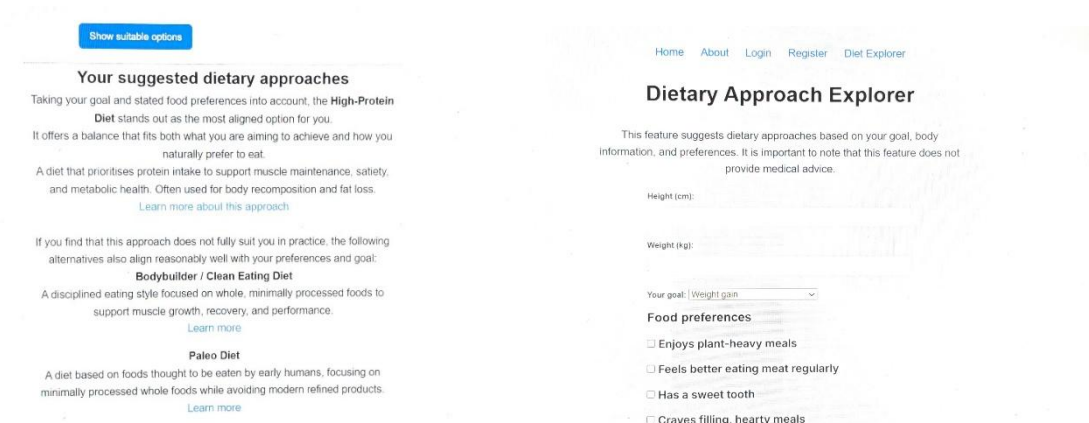
jogging — 30 min (on 10/12/2025, 22:16:57) Edit Delete	Calories burnt: 500 Intensity: 5/5
yoga — 40 minutes — 220 calories burnt — Intensity: 3/5 (on 10/12/2025, 20:24:25) Edit Delete	
rock climbing — 40 minutes — 400 calories burnt — Intensity: 3/5 (on 07/12/2025, 02:13:43) Edit Delete	
cycling — 15 minutes — calories unknown — Intensity: 4/5 (on 06/12/2025, 20:45:25) Edit Delete	

The Vision Board page allows users to write and update their health visions, describing their long-term goals and motivations. They can also add images which will appear beneath the text as a part of their vision board, each of these images can be removed, giving the users full access to add, delete or change any material within their vision board.



And at last, the log out function securely ends the session and returns users to the public log in page.

The Diet Feature suggests 3 different diet approaches according to the user's preferences and health goal, which is demonstrated in screenshots below. logged in users won't need to enter their weight/height and health goal since their BMI and health goal will be saved in the database after registration.)



The wellness feature gives the users 10 different questionnaires for them to assess their likelihood of deficiencies and risks with regard to certain diseases which have been included with the disclaimer that the results shouldn't be taken as medical advice or diagnosis.

Quick Wellness Check

These questionnaires estimate likelihood only and are not medical advice.

Iron deficiency questionnaire

How often have you felt unusually fatigued?

☐ Never ☐ Sometimes ☒ Often ☐ Almost every day

How often have you felt dizzy or light-headed?

☐ Never ☐ Sometimes ☒ Often ☐ Almost every day

How often have you noticed pale skin or darker under-eyes?

☐ Never ☐ Sometimes ☒ Often ☐ Almost every day

How often have you felt short of breath during normal activity?

☐ Never ☐ Sometimes ☒ Often ☐ Almost every day

How often have you felt colder than others around you?

☐ Never ☐ Sometimes ☒ Often ☐ Almost every day

Get iron deficiency likelihood

Your iron deficiency likelihood is 67%.

Vitamin D deficiency questionnaire

How often do you get very little sunlight exposure?

☐ Never ☐ Sometimes ☐ Often ☐ Almost every day

How often have you felt low or flat in mood?

[Home](#) [About](#) [Login](#) [Register](#) [Diet Explorer](#)

Quick Wellness Check

These questionnaires estimate likelihood only and are not medical advice.

Overall likelihood summary

Note: Some questionnaires are incomplete.

Your iron deficiency likelihood is 73%.

Your vitamin D deficiency likelihood is not available (incomplete).

Your vitamin B12 deficiency likelihood is not available (incomplete).

Your dehydration likelihood is not available (incomplete).

Your sleep deprivation likelihood is not available (incomplete).

Your stress / burnout likelihood is 33%.

Your magnesium deficiency likelihood is not available (incomplete).

Your thyroid dysfunction likelihood is 100%.

Your blood sugar instability likelihood is not available (incomplete).

Your immune weakness likelihood is 73%.

Iron deficiency questionnaire

Advanced Techniques:

- 1- Image upload handling with Multer: this application uses Multer with a custom disk storage configuration to handle user image upload for the vision board, images will be saved and filenames will be generated.
Routes/vision.js


```
// Storage setup for vision board images
const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, path.join(__dirname, '../public/uploads'));
  },
  filename: function (req, file, cb) {
    const uniqueName = Date.now() + '-' + file.originalname;
    cb(null, uniqueName);
  }
});

const fileFilter = (req, file, cb) => {
  const allowed = ['image/jpeg', 'image/jpg', 'image/png'];
  cb(null, allowed.includes(file.mimetype));
};

const upload = multer({ storage: storage, fileFilter: fileFilter });
```

- 2- Password hashing with bcrypt: User passwords are securely hashed before being stored, ensuring the database never contains plain-text passwords. Routes/auth.js

```
// Turn the password into a secure hash
const hashedPassword = await bcrypt.hash(password, 10);
```

- 3- Dynamic SQL sorting and searching: The records page supports multiple sorting options and keyword search using dynamically constructed SQL queries. Routes/records.js

```
// SQL ORDER BY logic depending on selection
let orderBy = "created_at DESC"; // default

if (sortOption === "date_asc") {
  orderBy = "created_at ASC";
} else if (sortOption === "duration") {
  orderBy = "duration DESC";
} else if (sortOption === "calories") {
  orderBy = "calories_burnt DESC";
} else if (sortOption === "activity") {
  orderBy = "activity ASC";
}

// Select all records that belong to the logged-in user, sorted properly
const [rows] = await db.query(
  `SELECT * FROM records WHERE user_id = ? ORDER BY ${orderBy}`,
  [req.session.userId]
);
```

- 4- Session-based authentication and protected routes: Only logged-in users can access sensitive/personal features such as records, the vision board, and the profile page.

Files: routes/records.js, routes/vision.js, routes/auth.js

```
if (!req.session.userId) {
  return res.redirect('/login');
}
```

- 5- RESTful PUT and DELETE request support using method-override: The application supports editing and deleting records through PUT and DELETE HTTP methods, even though HTML forms normally only support GET and POST.

Index.js routes/records.js

```

1
2 // this allows us to use HTTP verbs such as PUT or DELETE in
3 app.use(methodOverride('_method'));
4
5 // This line tells the app where to find the database
6
7 // Delete a record
8 router.delete('/records/delete/:id', async (req, res) => {
9   try {
10     if (!req.session.userId) {
11       return res.redirect('/login');
12     }
13
14     const recordId = req.params.id;
15
16     await db.query(
17       "DELETE FROM records WHERE id = ? AND user_id = ?",
18       [recordId, req.session.userId]
19     );
20
21     res.redirect('/records');
22   } catch (error) {
23     console.error(error);
24     res.send("Something went wrong");
25   }
26 });

```

- 6- Automatic BMI calculation/classification: (Not Necessarily considered as advanced, however inventory and worth a mention.) When users register or update their profile credentials, their MBTI is calculated and categorised based on medical BMI criteria.

```

8
9 // Simple helper that returns a BMI classification based on the BMI value
10 function classifyBMI(bmi) {
11   if (!bmi) return null;
12
13   if (bmi < 16) return "Severely Underweight";
14   if (bmi < 18.5) return "Underweight";
15   if (bmi < 25) return "Normal";
16   if (bmi < 30) return "Overweight";
17   if (bmi < 35) return "Obesity Class I";
18   if (bmi < 40) return "Obesity Class II";
19   return "Obesity Class III";
20 }
21

```

- 7- Autocomplete suggestion endpoint (JSON API): this web application provides an endpoint that returns activity suggestions according to what the user types and enables autocomplete functionality.

You can see the code snippet in the image attached below:

```

// Suggest activities for autocomplete
router.get('/records/suggest', async (req, res) => {
  if (!req.session.userId) return res.json([]);

  const q = req.query.q || '';
  if (q.length < 1) return res.json([]);

  const [rows] = await db.query(
    "SELECT DISTINCT activity FROM records WHERE user_id = ? AND activity L"
    [req.session.userId, q]
  );

  res.json(rows.map(r => r.activity));
});

```

- 8- The DietScore function implements a multi-criteria heuristic decision model that combines multiple user preferences using positive and negative weights.

Instead of simple filtering, it applies deterministic, rule-based inference to rank diet options in a transparent manner.

```

// Scores how well a diet matches the user's goal and preferences
function scoreDiet(diet, goal, prefs) {
  let score = 0;

  // Core goal match (still important, but no longer overwhelms everything)
  if (diet.goals.includes(goal)) score += 4;

  // Enjoys plant-heavy meals
  if (prefs.vegetarian) {
    if (["plant_based", "mediterranean", "dash"].includes(diet.key)) score += 3;
    if (["ketogenic", "paleo", "bulking", "mass_gainer"].includes(diet.key)) score -= 2;
  }

  // Feels better eating meat regularly
  if (prefs.meat) {
    if (["high_protein", "paleo", "clean_eating", "bulking"].includes(diet.key)) score += 3;
    if (diet.key === "plant_based") score -= 2;
  }

  // Has a sweet tooth
  if (prefs.sweet) {
    if (["mediterranean", "zone"].includes(diet.key)) score += 2;
    if (["ketogenic", "cutting"].includes(diet.key)) score -= 2;
  }

  // Craves filling, hearty meals
  if (prefs.hearty) {
    if (["high_protein", "bulking", "mass_gainer", "paleo"].includes(diet.key)) score += 3;
    if (["intermittent_fasting", "plant_based"].includes(diet.key)) score -= 1;
  }
}

```

9- The POST /wellness handler, specifically the loop over groups that calls `getAnswers()` and `calculateLikelihood()` performs iterative aggregation with input validation. It computes results and `overallComplete` in a single pass while handling missing answers safely. While this isn't particularly advanced, it demonstrates clear control flow, correctness under partial input, and good software design practice.

```

// Calculate all results
for (const key in groups) {
  const values = getAnswers(req.body, groups[key]);
  if (values === null) {
    results[key] = null;
    overallComplete = false;
  } else {
    results[key] = calculateLikelihood(values);
  }
}

```

AI Declaration:

AI was used as a supportive tool in a limited way during the development of this application for. For example, I used AI to help interpret a MySQL "access denied" error, which turned out

to be caused by incorrect database user permissions rather than any issue with my code. With that explanation, I was able to configure a dedicated database user with the correct privileges. I also used AI to clarify how to set up Multer's storage engine for image uploads, including MIME-type filtering and creating unique filenames. In addition, I referred to AI to better understand asynchronous MySQL queries using `async/await` and how the `mysql2` promise-based connection pool works, along with confirming best practices for using environment variables in database configuration. AI was used to come up with short explanations of each diet and the ideas for different diets according to the goals and preferences. The AI suggestions built into my code editor also occasionally helped with completing or rephrasing comments, but weren't used to generate the application logic. Overall, whenever I used AI, it was not solely to solve an issue, but also for to deepen my understanding of the underlying concepts.

Thank you for taking the time to read my report.
Mobina Sharafi 33777254