Outline

I built a Fitness Progress Tracker that lets users record, review, and manage their daily exercise activity. After creating an account, users can log details such as activity type, duration, calories burned, notes, and date. The application includes search tools, a statistics page that visualises calorie trends, and full editing and deletion of entries. Passwords are hashed using bcrypt, and user sessions ensure that only the authenticated user can access their data. The project runs both locally and on the Goldsmiths server through automatic base-path detection. Overall, the work demonstrates secure authentication, CRUD operations, database interaction, form validation, and multi-environment deployment.

Architecture

The system is built with Node.js and Express.js, which handle routing, session management, and form validation. EJS templates render the interface. A MySQL database stores users and activities, accessed with the MySQL2 driver. All credentials are hashed with bcrypt. Chart.js is used on the client side for visual statistics. Base-path detection allows the same codebase to run in different environments without modification.

```
   +-------------+
   |   Browser   |
   +-------------+
         |
         v
 +-------------------+
 |   Express Server   |
 | Routes + Sessions  |
 +-------------------+
         |
         v
  +-----------------+
  |   MySQL DB       |
  | users + activity |
  +------------------+
```

Data Model

The data model contains two tables: **users** and **activities**.
The users table stores usernames and bcrypt-hashed passwords.
The activities table stores each entry, including type, duration, calories, notes, and date. A foreign key links each activity to the user who created it. Cascading deletes ensure that removing a user also removes their activities.

Data Model Diagram
Users (id PK → username, password_hash)
Activities (id PK → user_id FK, type, duration, calories, notes, date)

User Functionality

Users begin by signing up with a password that must meet strength requirements. After logging in, they are taken to a dashboard showing all their activities in date order. Each entry can be edited or removed, and new activities can be added through a simple form.

A search page allows users to filter activities by keywords found in the type or notes field. The statistics page presents a line graph that plots calorie changes over time, allowing users to see trends at a glance. The application also adjusts links automatically depending on whether it runs locally or on the Goldsmiths server. All private pages require an active session, and the login system redirects users back to the page they attempted to reach before signing in.

Advanced Techniques

Several features were implemented to strengthen the application and make it easier to maintain.

The first is dynamic base-path detection, which determines at runtime whether the application is running locally or on the Goldsmiths server. This allows all routes and redirects to adapt automatically:

```
function getBase(req) {
    return req.headers.host.includes("doc.gold.ac.uk") ? "/usr/441" : "";
}
```

The second technique is secure password storage using bcrypt. Passwords are hashed when users register and safely compared during login:

```
const hash = await bcrypt.hash(password, 10);
```

A related improvement is the normalisation of MySQL date values so that they display correctly in HTML date inputs:

```
function formatDate(dateObj) {
    return dateObj.toISOString().substring(0, 10);
}
```

Finally, the statistics page uses Chart.js to produce a clear visual summary of calorie trends. It renders data retrieved from the database into a line chart:

```
new Chart(ctx, {
    type: "line",
    data: { labels, datasets: [{ data: values }] }
});
```

Together, these techniques contribute to a smoother user experience and a cleaner backend design.

AI Declaration

I used ChatGPT during this project to support my planning and understanding of key concepts. It helped me improve sentence clarity in this documentation, understand and resolve errors, and restructure parts of my code when I was unsure how to organise them properly. All development work, debugging, and final implementation choices were carried out by me. AI was used only as a support tool.