Outline

My project is a health and fitness “Workout Logger” web application. Its purpose is to help users log their workouts, review their training history, and analyse patterns over time. Built with Node.js, Express, EJS and MySQL, following the same architecture as the Bertie’s Books labs.

Users are able to register an account, login, and then create, view and search their own workouts. Each workout includes the date, category (Cardio, Strength, Mobility etc.), duration, intensity and personal notes for that workout. Workouts are stored in a MySQL database and are linked to the currently logged in user.

The logger also includes validation and sanitisation of user input, protecting against invalid data and basic XSS attacks. Login attempts and logout events are recorded in an audit log table which can be viewed through a restricted “Audit Log” page, only accessible by admin (User: gold - Pass: smiths).

Architecture

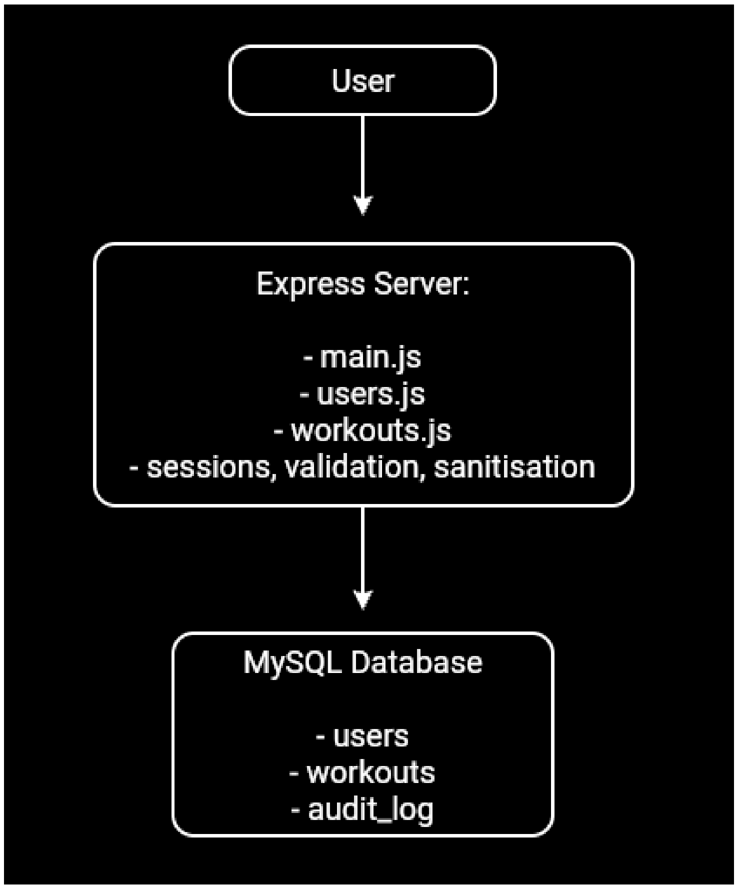
The application follows a classic three tier architecture:

Presentation tier: EJS templates rendered by Express, with HTML and CSS, and navigation that adapts based on login status.

Application tier: Node.js + Express routes organised into modules (main.js, users.js, workouts.js), with middleware for sessions, validation, sanitisation and access control.

Data tier: MySQL database health, accessed via a mysql2 connection defined in index.js, globally as db.

The app is deployed on the Goldsmiths VM and runs using forever so it stays available for marking.



Data Model

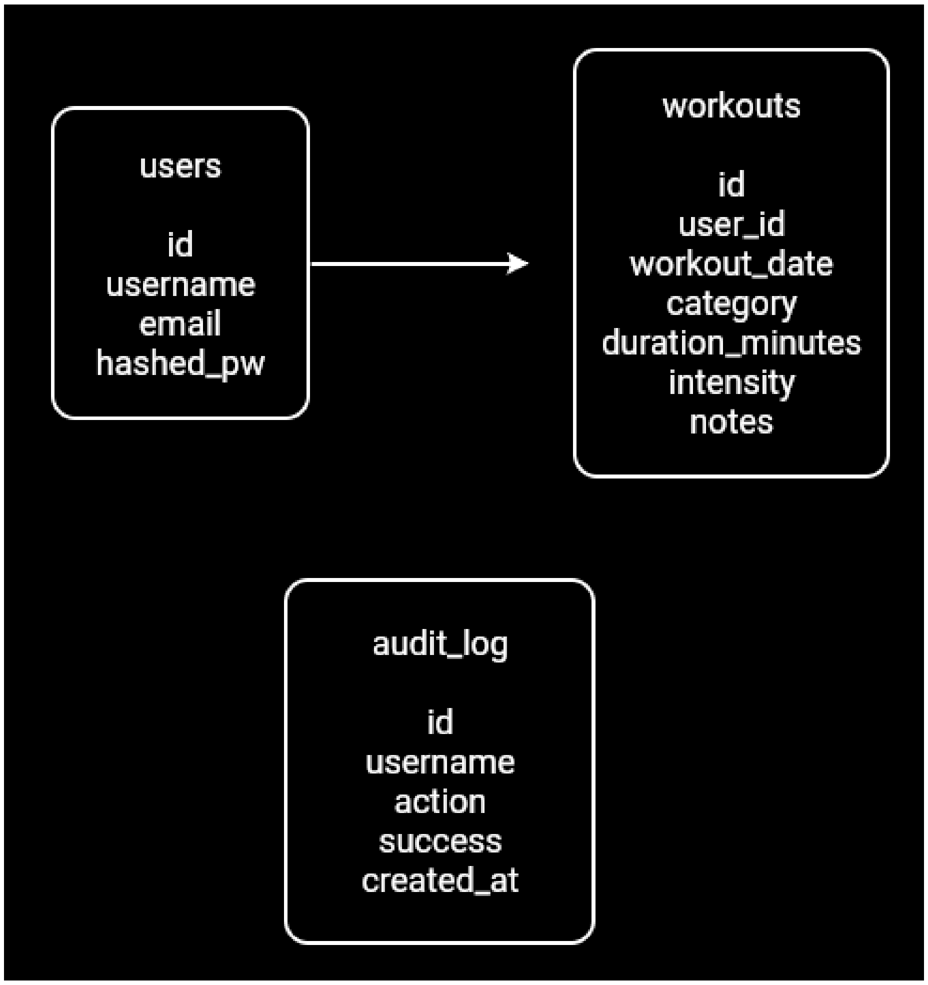
The health database contains three main tables:

users: id, username, first_name, last_name, email, hashed_password, created_at. Stores registered users with hashed passwords.

workouts: id, user_id, workout_date, category, duration_minutes, intensity, notes, created_at. Each row is a workout linked to a user via user_id.

audit_log: id, username, action, success, created_at. Records login successes, failures and logout events for security and audit purposes.

Install and test data are created via create_db.sql and insert_test_data.sql, including the required gold / smiths user.

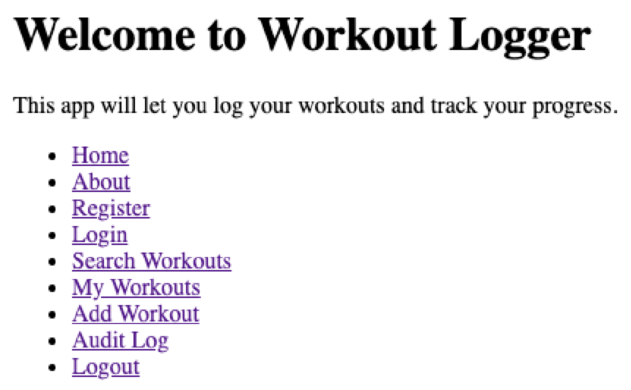


User Functionality

The application is designed for a simple, linear user journey that starts from the home page and then branches into account management and workout logging.

Home and About

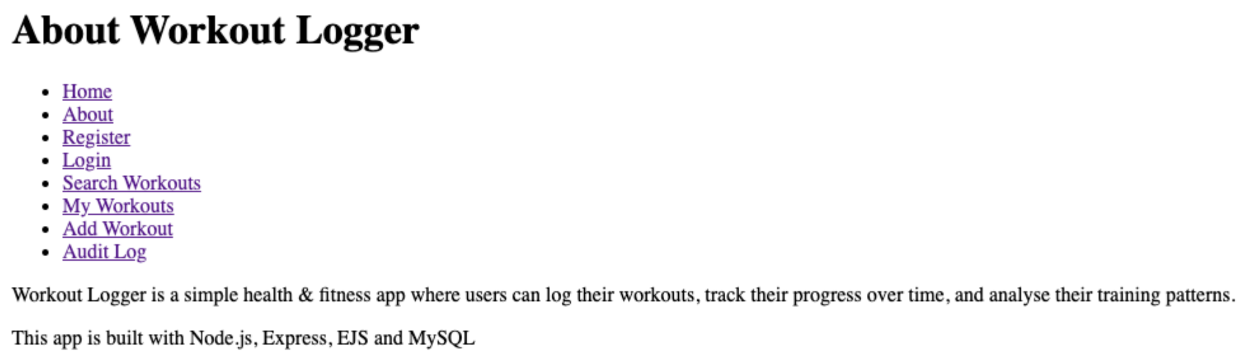
The home page introduces the Workout Logger and provides navigation links to About, Register, Login, My Workouts, Add Workout, Search Workouts and (for the admin user) the Audit Log. The navigation dynamically changes depending on whether a user is logged in. The About page briefly explains the purpose of the app and the technology stack.



Welcome to Workout Logger

This app will let you log your workouts and track your progress.

- Home
- About
- Register
- Login
- Search Workouts
- My Workouts
- Add Workout
- Audit Log
- Logout



About Workout Logger

- Home
- About
- Register
- Login
- Search Workouts
- My Workouts
- Add Workout
- Audit Log

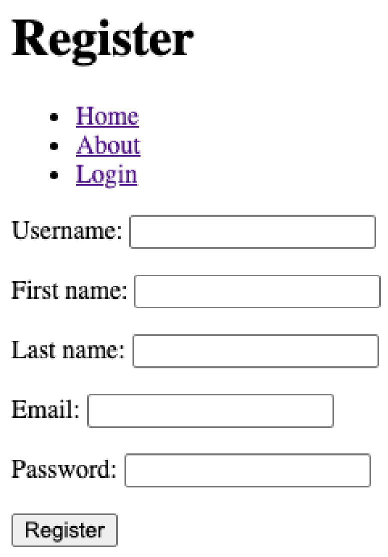
Workout Logger is a simple health & fitness app where users can log their workouts, track their progress over time, and analyse their training patterns.

This app is built with Node.js, Express, EJS and MySQL

Registration and Login

Users can create an account via the Register page. The registration form collects username, first name, last name, email and password. Server side validation checks that the username length is sensible, the email (if provided) is valid and the password is at least eight characters long. Inputs are sanitised to protect against embedded scripts before being saved.

The Login page allows existing users to login using their username and password. Passwords are stored using hashing, and login credentials are verified by comparing the submitted password with the stored hash. After logging in, a session is created and the username is stored in the session so the app can recognise the user across requests.



Register

- Home
- About
- Login

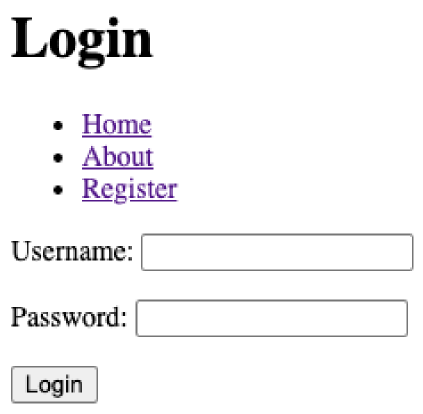
Username:

First name:

Last name:

Email:

Password:

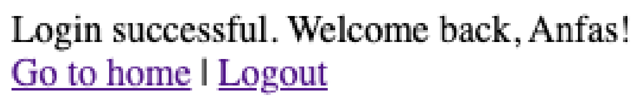


Login

- Home
- About
- Register

Username:

Password:



Login successful. Welcome back, Anfas!
Go to home | Logout

Workout Management

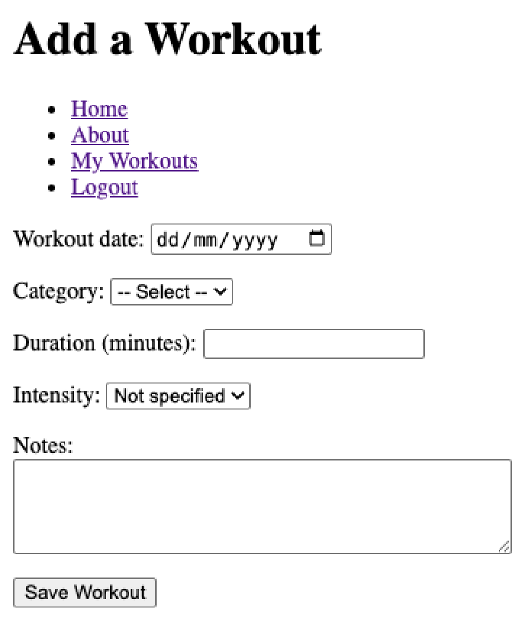
Logged in users can manage their workouts through three main features:

Add Workout: A form where users enter workout date, category (Cardio, Strength…), duration in minutes, intensity (low/medium/high) and optional notes. Validation ensures required fields and reasonable duration values; all text inputs are sanitised. The workout is stored in the workouts table, linked to the current user.

My Workouts: Displays a table of the logged in user's workouts, ordered by date. Each row shows date, category, duration, intensity and notes. This gives a quick overview of training history.

Search Workouts: Provides a search form where users can filter their workouts by keyword and optional date range. The search results page shows only the workouts that match the criteria for the current user.

All workout routes are protected by middleware so they can only be accessed when a user is logged in.



Add a Workout

- Home
- About
- My Workouts
- Logout

Workout date: dd/mm/yyyy

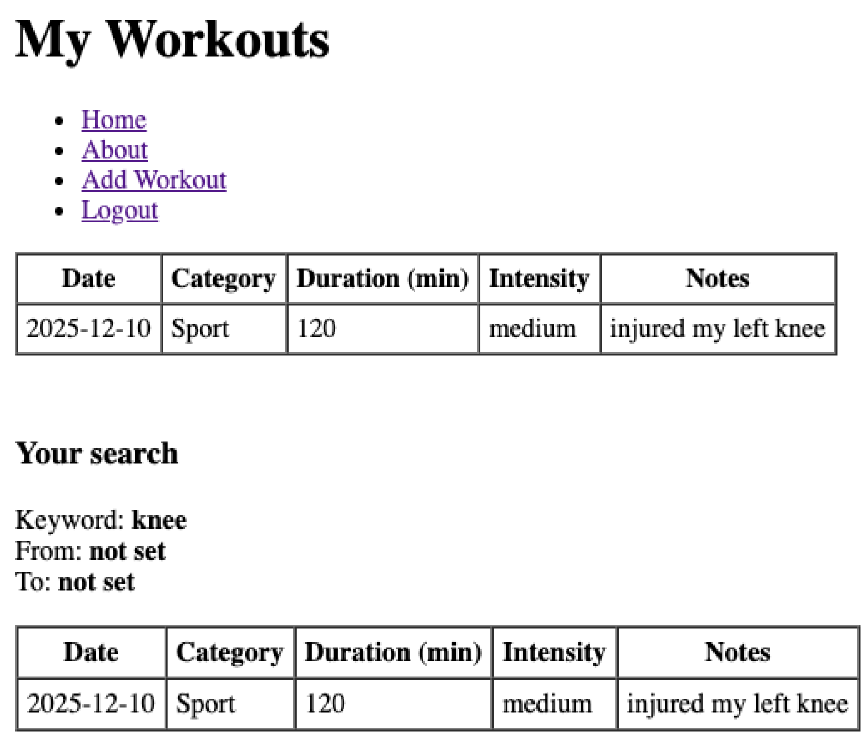
Category: -- Select --

Duration (minutes):

Intensity: Not specified

Notes:

Save Workout



My Workouts

- Home
- About
- Add Workout
- Logout

Date	Category	Duration (min)	Intensity	Notes
2025-12-10	Sport	120	medium	injured my left knee

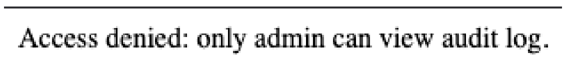
Your search

Keyword: **knee**
From: **not set**
To: **not set**

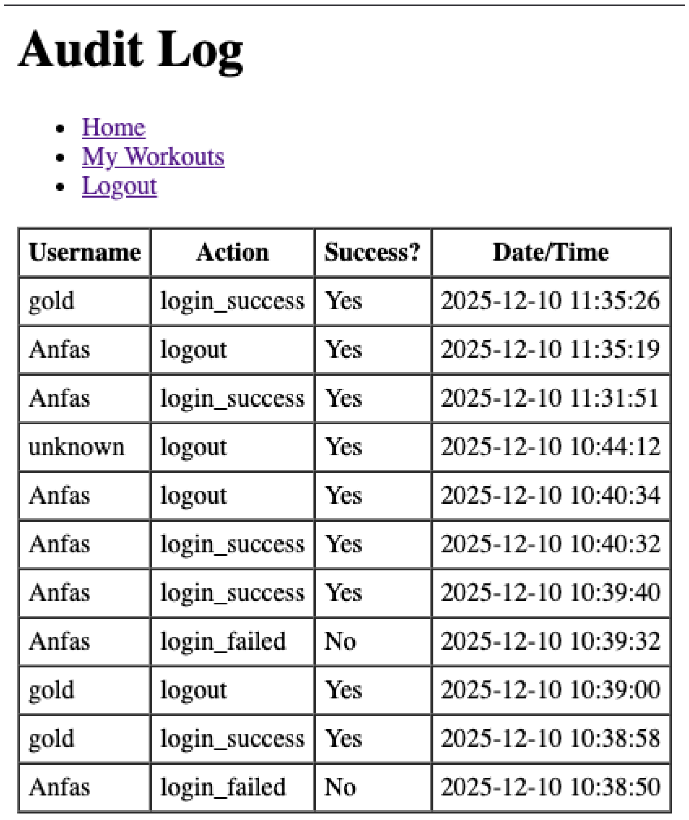
Date	Category	Duration (min)	Intensity	Notes
2025-12-10	Sport	120	medium	injured my left knee

Audit Log

For the admin user (gold), there is an Audit Log page that lists username, action, success flag and timestamp for login attempts and logout events. This demonstrates access control and basic security auditing.



Access denied: only admin can view audit log.



Audit Log

- Home
- My Workouts
- Logout

Username	Action	Success?	Date/Time
gold	login_success	Yes	2025-12-10 11:35:26
Anfas	logout	Yes	2025-12-10 11:35:19
Anfas	login_success	Yes	2025-12-10 11:31:51
unknown	logout	Yes	2025-12-10 10:44:12
Anfas	logout	Yes	2025-12-10 10:40:34
Anfas	login_success	Yes	2025-12-10 10:40:32
Anfas	login_success	Yes	2025-12-10 10:39:40
Anfas	login_failed	No	2025-12-10 10:39:32
gold	logout	Yes	2025-12-10 10:39:00
gold	login_success	Yes	2025-12-10 10:38:58
Anfas	login_failed	No	2025-12-10 10:38:50

Advanced Techniques

1. Session based Navigation and Access Control

The application uses an express session to maintain user login state. req.session.username to all EJS templates via res.locals allows the navigation bar to adapt depending on whether a user is logged in or not. redirectLogin protects sensitive routes (such as workout pages and the audit log), redirecting anonymous users to the login page.

Additionally, access to the /users/audit route is restricted so that only the admin user can view the audit log. This adds a simple form of role based access control on top of authentication.

2. Validation and Sanitisation

The app uses express validator to validate form inputs and express sanitizer to guard against XSS attacks. Registration validates username length, password length and email format, while workout creation validates required fields and duration ranges. Inputs such as first name, last name, notes and category are sanitised using req.sanitize(). This demonstrates secure handling of user input and protection against malicious HTML/JavaScript.

3. Audit Logging of Authentication

Every login attempt and logout action is recorded in the audit_log table. Successful logins are logged with login_success, failed attempts with login_failed and logouts with logout. This is implemented via a small helper function that adds a line to the audit table. The audit page displays these entries in descending order of time, a good security practice for monitoring authentication activity.

4. User Search and Filtering

The workout search feature allows users to search their workouts using multiple optional filters: text keyword and date range. The server builds the SQL query based on the filters submitted and always restricts results by user_id. Ensuring that users never see someone else's data.

These techniques, combined with proper deployment and database scripting, I have integrated multiple module topics into a secure, user focused application.

AI Declaration

AI (ChatGPT) was used only for debugging purposes when my own attempts were unsuccessful. Near the end of the project, I also relied on AI to help solve an issue involving two forever loops running simultaneously that I was unable to stop.