# Health & Fitness Tracker Application Documentation

## Outline

## The web application with main constituents used Node.js and Express. Core features enable users to keep track of workouts and share progress in various ways while having a platform to participate in a community-driven setting via the presence of forums. The next stage of development consisted of user registration (external two-factor TOTP verification) and protecting the web application from bot-made accounts, while additionally addressing the need for moderation by introducing roles. A more advanced feature was the implementation of a tool allowing users to manually note or upload GPX files related to workout data. The dashboard is where users could find stored information and track personal workout history while offering CSV export with the goal of enabling access at all times. A public feed and community posts drive engagement among users while also facilitating workout suggestions and boosting engagement via comments. An admin has the privileges of a hidden menu with an ability to manage user roles and posts. The data used is stored in a MySQL database with tables for users, workouts, posts, comments, attachments, and audit logs. Further strengthening our web app is security enforced via bcrypt hashing, express-session, CSRF tokens, and input sanitisation.

## Architecture

**High-level architecture diagram**

**A computer screen shot of a computer screen

AI-generated content may be incorrect.**

Implementing multi-tier system architecture: an application tier consisting of a Node.js/Express server that handles routing, logistics, and template rendering (EJS). Static assets (CSS, JavaScript) and charts. js elevates dynamic UI components on the client side, followed by the data tier (MySQL) database (accessed via the mysql2 promise pool), which is responsible for storing all entities. Middleware (helmet, csurf, express-validator) provides security. Key components: Node/Express web server, EJS views, RESTful routes (including a JSON API), and MySQL with connection pooling.   
  
**End user flowchart**

A diagram of a computer

AI-generated content may be incorrect.

## Data Model

The MySQL schema uses normalised tables. Users link to workouts and posts via foreign keys (e.g., workouts.user\_id → users.id). As required, each workout and post can have many comments (foreign key comments.workout\_id or comments.post\_id to workouts.id or posts.id). Attachments further relate to a workout or comment (attachments.workout\_id and attachments.comment\_id). Audit logs record provide a way to monitor login attempts. Hence, all relationships use foreign keys to enforce referential integrity. Furthermore, deleting a user cascades to their workouts/posts. Indexes on activity and date are crucial to optimise searches and guarantee scalability as data increases.

A screenshot of a computer

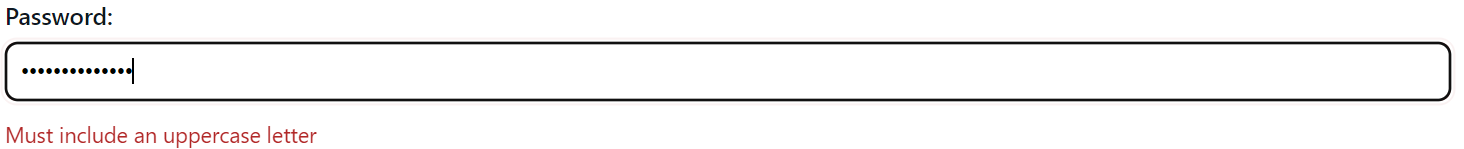
AI-generated content may be incorrect.

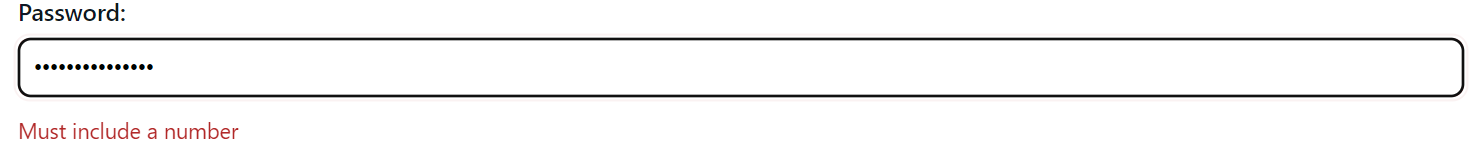
## User Functionality

**Registration & Login:** New users register with name, email, username, and password.

A screenshot of a fitness tracker

AI-generated content may be incorrect.





Upon successful registration, a TOTP secret is generated and shown as a QR code (users scan it in an authenticator app) via Speakeasy. After setup, users verify a 6-digit code to activate the account. Verified users log in with username/password; on first login they enter a TOTP code.

A screenshot of a computer

AI-generated content may be incorrect.

**Dashboard (My Workouts):** Upon successful logging, get access to an initially empty workouts table (paginated).

A screenshot of a computer

AI-generated content may be incorrect.

With each entry stored in their personal table. The table contains date, activity, duration, calories, and notes. Users can add a workout via a form (enter activity, date, duration, etc.) or upload a GPX file to automatically import one or more workouts. A “Download CSV” button exports all personal workouts as a CSV file. (Screenshot: dashboard with chart and workout list).

A screenshot of a fitness form

AI-generated content may be incorrect.

**Search & Public Feed:** A search page improves user experience by providing a tool to find workouts by keyword, date range, or user, with results shown in a table.

A black line on a white background

AI-generated content may be incorrect.

The public feed shows all workouts marked public by any user, newest first. Clicking a workout in the feed opens its detail page.

A screenshot of a computer

AI-generated content may be incorrect.

**Community Posts:** A forum at /posts lists community messages. Authenticated users can create a new post (title and content).

A close-up of a white background

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Each post page displays the content and its comments. Users can add text comments on posts; no file uploads are allowed on post comments. Only the post owner or an admin can delete a post.

A white text box with a black text box

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Admin Interface: Users with the admin role see an extra “Admin: Users” page. This lists all registered users (username, email, role, verified status, creation date). Admins can change any user’s role (user, admin, trainer, manager) via a dropdown form. Access to admin pages is restricted via role-based middleware.

A screenshot of a computer

AI-generated content may be incorrect.

## Throughout the UI, navigation links (Home, About, Workouts, Posts, etc.) and consistent styling (CSS) create a smooth user experience. All forms embed CSRF tokens and use server-side validation to prevent invalid input.

## Advanced Techniques

**Several advanced features showcase the application’s robustness**

**Two-Factor Authentication (TOTP):** In routes/users.js, the option of choice was speakeasy library to generate and verify TOTP codes. Registration is made more secure by generating a secret and QR code URL:

A screen shot of a computer code

AI-generated content may be incorrect.

Upon execution, a base32 generates; hence, a secret and an otpauth URL for a QR code appear. Following instructions, the user scans this in their authenticator app. During login, we verify the code:

A screen shot of a computer code

AI-generated content may be incorrect.

Only if the code matches (within the time window) do we mark login success. Therefore, a strong second factor validates preventing potential breaches on top of bcrypt-hashed passwords.

**GPX File Import:** The /workouts/upload route (in routes/workouts.js) uses multer to accept a GPX or XML file upload. We read the file and parse it with xml2js:

A computer screen shot of a computer code

AI-generated content may be incorrect.

A computer screen with text and symbols

AI-generated content may be incorrect.

Iteration through each track segment (trkseg and trkpt) extracts timestamps and computes the workout duration and date. Subsequently, it then inserts a new instance of the workout record with source “gpx” for each track. Minimising the time it takes to add multiple workouts by enabling bulk importing multiple workouts from one file without user effort.

**File Attachments with Transaction:** Additionally, when posting a comment on a workout, there is a feature to attach files (in routes/workouts.js). It was opted for multer setup (attachUpload.single('attachment')) then it’s handled in a database transaction:

A screen shot of a computer code

AI-generated content may be incorrect.

By wrapping the operations in beginTransaction() and commit(), the comment and its attachment are saved atomically. Meaning if anything causes unexpected behaviour, the transaction rolls back, preventing orphaned records.

**CSV Export:** The route /workouts/export streams user workout data as CSV. Server-side code constructs CSV lines and sets headers:

A screen shot of a computer code

AI-generated content may be incorrect.

**Security and validation** were inspired by previous lab work with advanced input rules (e.g., password strength, valid dates), and express-sanitiser to strip HTML from text inputs. The helmet middleware sets secure HTTP headers. CSRF protection (csurf) is applied to all non-API POST forms. Session management uses express-session with a secret. All these techniques protect against common web vulnerabilities.

A screen shot of a computer program

AI-generated content may be incorrect.

Each of these techniques is implemented in the code files (notably routes/users.js, routes/workouts.js, public/charts.js, etc.) and demonstrates advanced use of Node.js libraries and database features. As a result, they provide a secure, user-friendly fitness tracking application.