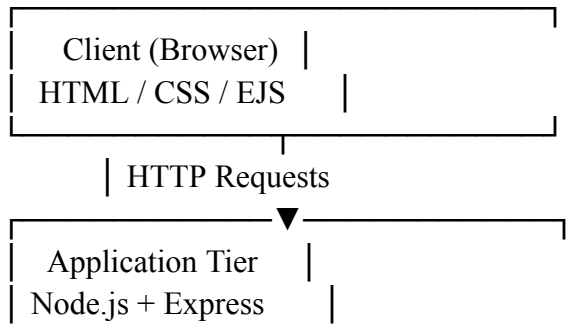
Dynamic web documentation

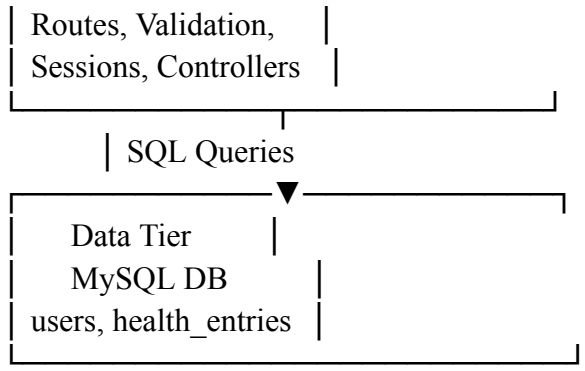
Outline:

Gold Health is a health tracking web application that is designed to help users track their daily activity. The system allows users to securely register, login, and keep a personal record for activities that they choose to record, such as meals, workouts, habits or just notes about their day. Each user can create entries and view their full history and search for past logs using the search feature. The app automatically creates a note of the date allowing the records to be accurately dated.
The application is secure and is easy to use. The user authentication keeps each entry private as users can only view their own notes. Validation ensures that only clean data is used, the system uses what we covered in Bertie's books to ensure that there is proper sanitization and structure. It is efficient to display the data in descending order so that it shows the most recent logs first.
Gold Health is built using only the allowed techniques and, overall the app provides a simple and effective way for users to reflect on their daily habits and monitor progress over time.

High level architecture:

Gold Health follows a two-tier architecture. The application tier is built using Node.js, express and ejs. It handles routing, validation, authentication and rendering user pages. The server communicates with the data tier, which is a mysql database which stores user and health entries. User sessions are managed with express-session, which allows secure login and protected routes. All validation and sanitization occur in the application layer. The system runs on both local host and a VM





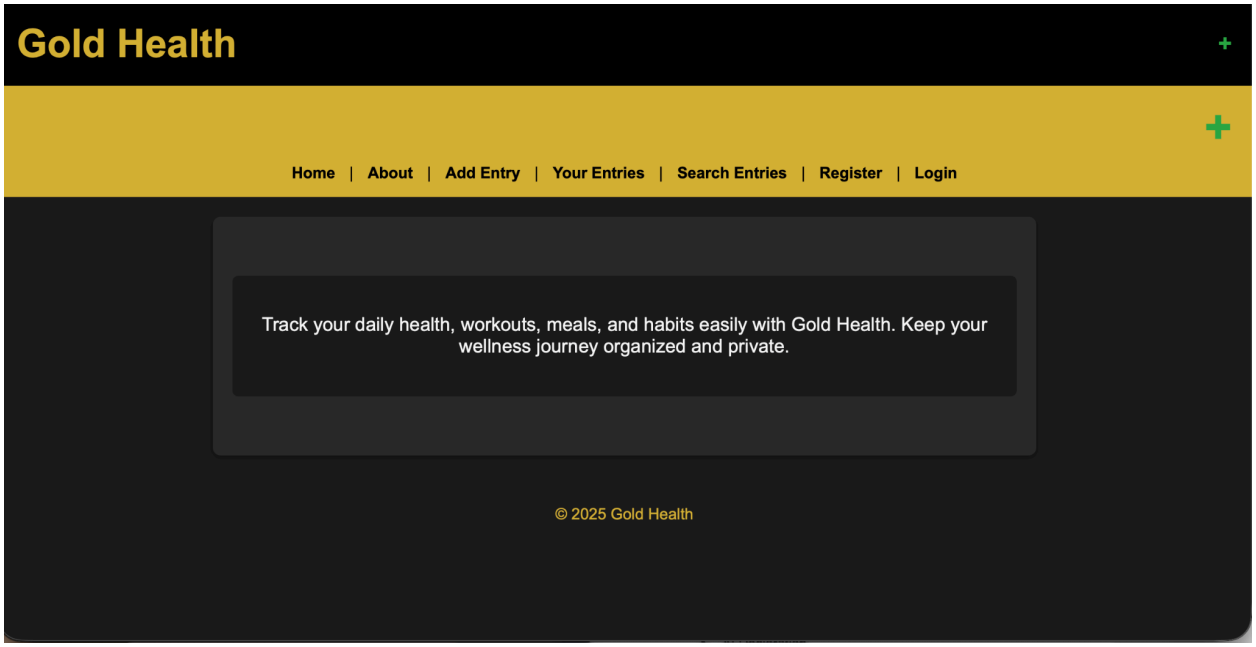
Data model:

Gold health uses two tables. Users and health_entries. The users tables stores credentials and basic information for each registered user. Health_entries tables stores activity logs, each linked to a user however audit is not implemented like it was in Bertie's books. And each user is linked to a user ID which auto increments. Queries filter entires by user_id which ensures privacy so that only users may see their own data. Sorting shows the most recent entries first.

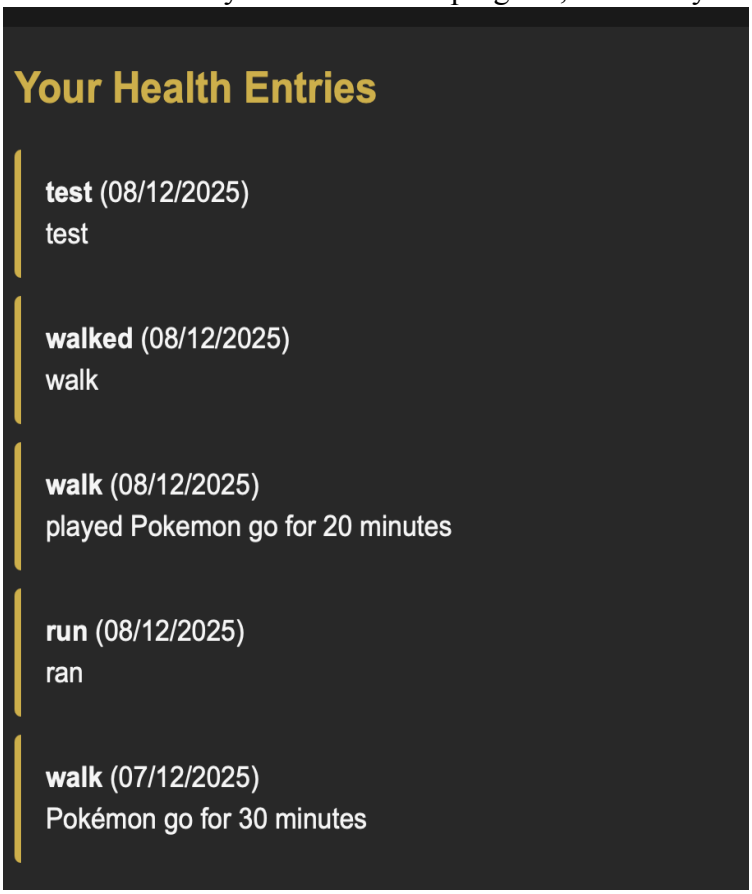
User facing functionality:

Gold health has a simple and clean user expreience that helps teh user track their daily health activities. When users arrive at the homepage , they can navigate using a menu bar that links login, about, add entry, view and search entry, and login. The entries are protected so that a user has to login to view entries.
Users can create an account through a registration form page that includes validation for fields such as username and email. Once they have registered they can login using the login page. The login process checks for calid credentials and creates a session allowing access to the protected routes. This allows access to the personalised features.
Users can add a new activity by going to the add entry page. This form requires two fields. A title and description. Validation ensures that both fields are filled in. in earlies versions the system needed manual updates to the date, but it has been updated to automatically assign the date, which removes the need for manual input. Once the information is submitted the entry is stored in the database and the user is then redirected to their list of activities.

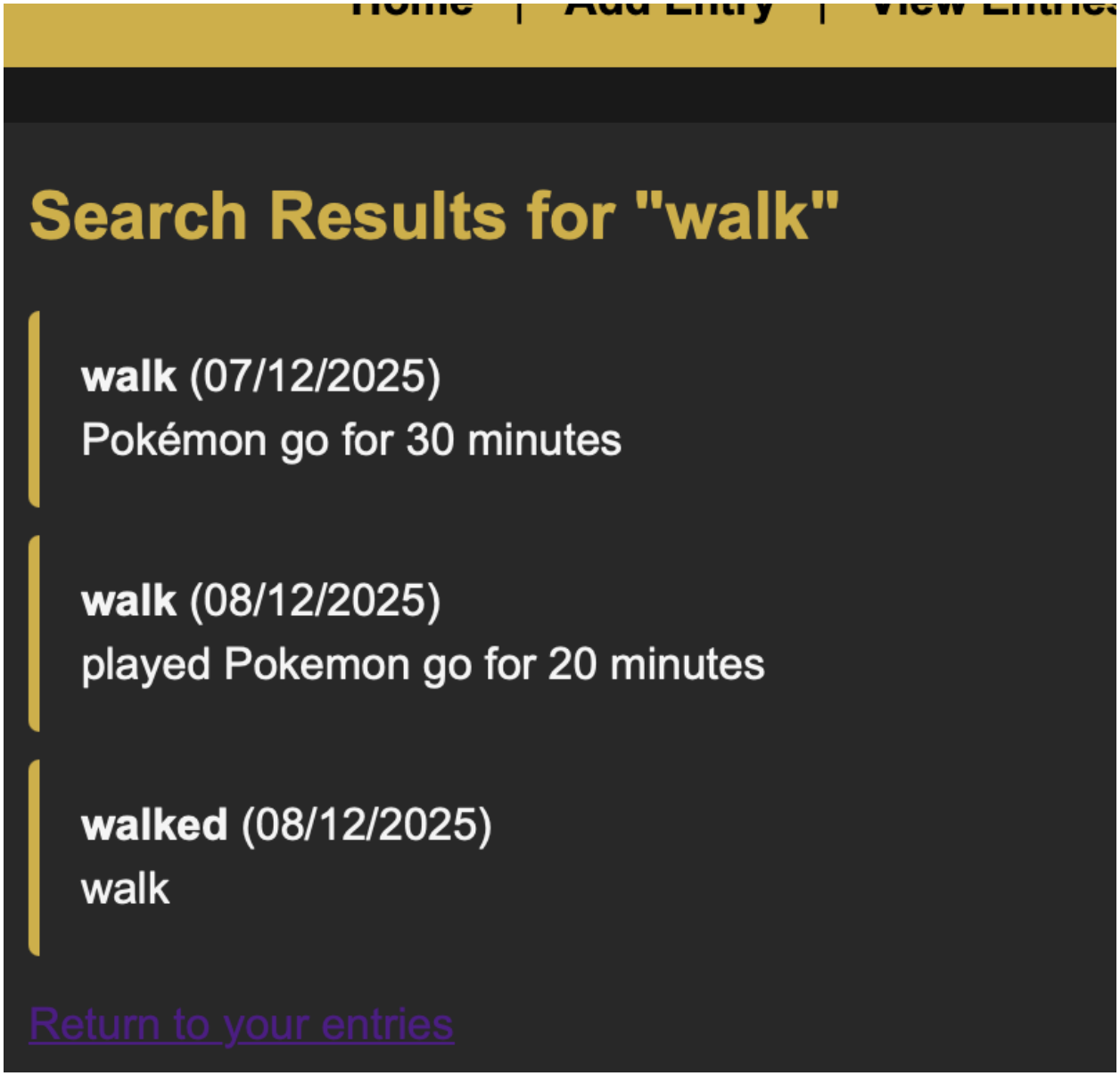
This is the screenshot of the home page.



The view entries page displays all logs created bt a user in descending order. With the newwest entry appearing first. This makes it easy to track current progress, each entry has its title, description and date.



The search entries page allows users to filter their own entries using keywords. Searches look inside teh title and detail fields this makes it easier for users to search for specific actions like workouts, meal records or notes.



A logout button terminates the users session and redirects them to the logout confirmation page. This ensures account security and prevents other people from accessing entries on shared computers.

The navigation links automatically route through the correct paths which prevent redirect issues and making the system functional on localhost and the VM.

Advances techniques:

Input validation: Gold Health uses express-validator to prevent invalid or empty form submission this ensures data quality and protects against improper input.

```
router.post(
  "/add",
  redirectLogin,
  [
    check("title").notEmpty().withMessage("Title is required"),
    check("details").notEmpty().withMessage("Details are required")
  ],
```

This is the code snippet from health.js for post/add.
Automatic date:

```
// Set date
    const date = new Date().toISOString().split('T')[0];
```

Also from health.js for the automatic date inside the add entry.

```
// Middleware to protect pages that require login
const redirectLogin = (req, res, next) => {
  if (!req.session.userId) {
    return res.redirect('../users/login');
  }
  next();
};
```

This is the middleware that protects sensitive pages which ensures only logged in users can access them. This setup allows it to work on the VM.

Ai declaration:

For this assignment, I used chatGPT to debug parts of the code, it helped me make the diagram for the data model. For the code AI helped me with the redirect logic. I was having trouble implementing the redirect, and having the same issues with Bertie's books, where it would

redirect to the root and not the proper URL. even implementing the same code as berties books did not help me with this issue so chatgpt recommended me to make a new middleware.js file. But i managed to fix the issue with the middleware by adding the period before the slash in the redirect inside health.js like so:

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

I don't think that the middleware file is actually doing anything but I have left it as is.