Dynamic Web Applications Documentation by Vlad Rotari 33830924

Outline

LevelUp- Health Quests is a web application designed to turn everyday workouts into short quests that users can complete and log, through a gamified process. Finding the right workout plans is usually difficult, and the app tries to offer a better solution by providing quests such as **Abs of Steel** or **Medium Jogging Quest**, each with it's own difficulty level, target duration and a list of exercises to do.

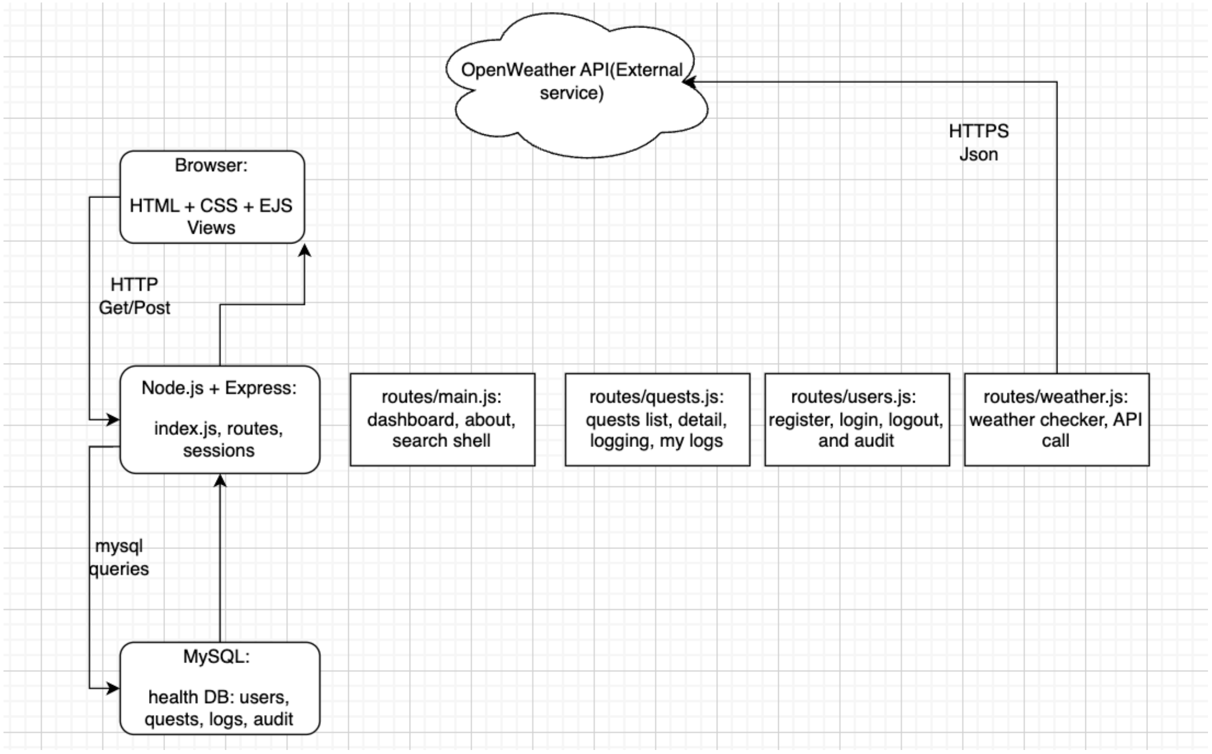
Users are able to register, log in, browse quests, and even view quest details and record completions, which are stored as personal logs under their separate tab. A dashboard highlights a daily quest and basic statistics like total quests, core quests and outdoor quests to give a quick sense of available content for that day. The app also includes a weather checker that uses the OpenWeather API to advise whether it is a good idea to attempt outdoor jogging quests. The user is able to check the weather separately as well but its meant to be used as a helping tool inside the app.

The system is built with **Node.js**, **Express**, **MySQL**, **EJS** templates and features a custom CSS. The project demonstrates a secure login feature with bcrypt, structured data storage for quests and logs, session-based authentication, and safe integration with an external API, all deployed behind Apache under a /usr/ prefix on the Goldsmith's Virtual machines.

Architecture

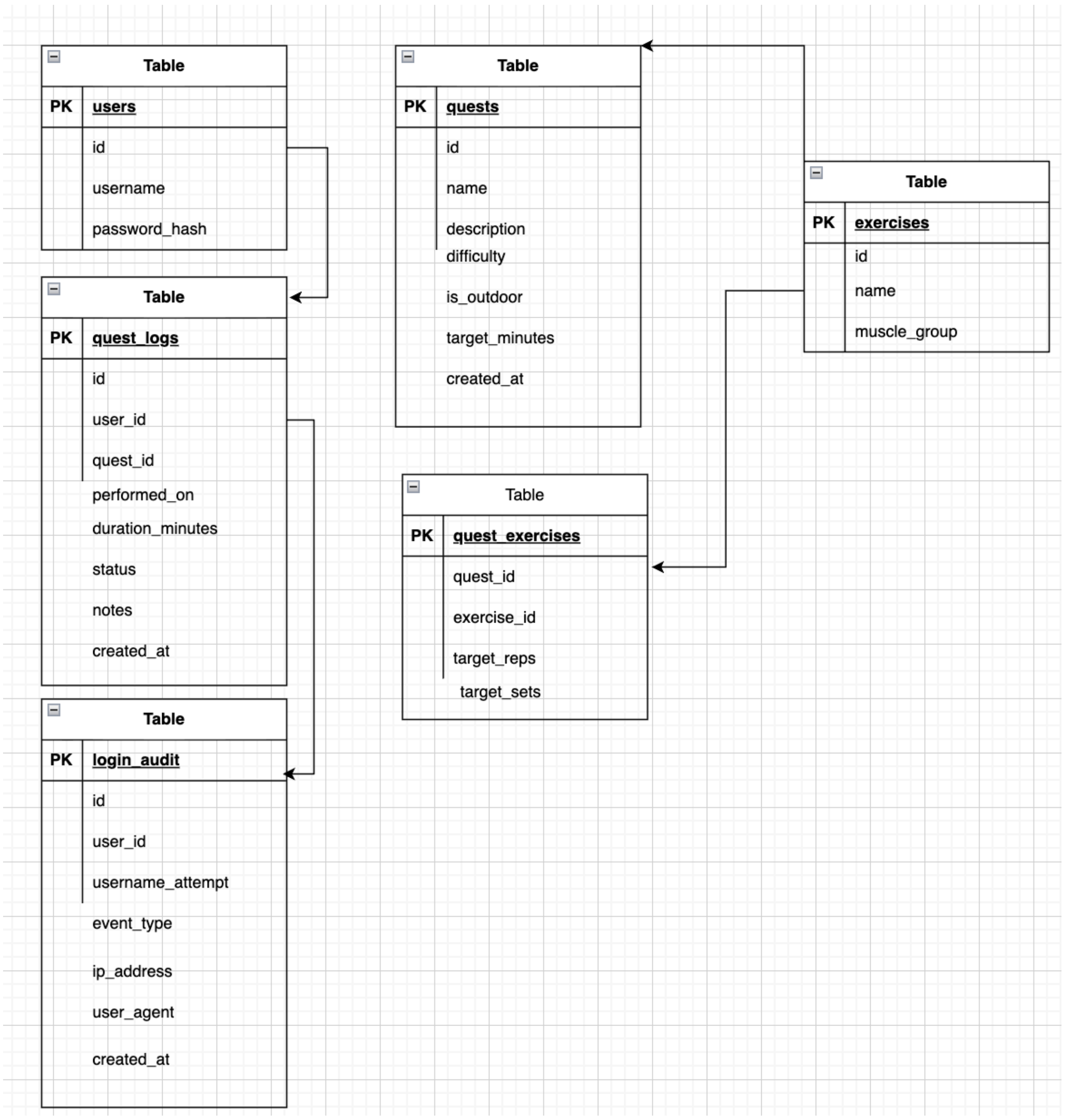
High-level architecture

The application tier features a Node.js/Express server which lives under index.js that mounts modular routers and renders EJS templates into HTML. Static assets such as public/style.css are served by Express. Configuration like database credentials, session secret, OpenWeather API key, and APP_BASE_PATH for the VM is handled via environment variables (dotenv). The data tier uses MySQL and its being accessed using mysql2 with parameterised queries.


OpenWeather API(External service)
HTTPS Json
Browser: HTML + CSS + EJS Views
HTTP Get/Post
Node.js + Express: index.js, routes, sessions
routes/main.js: dashboard, about, search shell
routes/quests.js: quests list, detail, logging, my logs
routes/users.js: register, login, logout, and audit
routes/weather.js: weather checker, API call
mysql queries
MySQL: health DB: users, quests, logs, audit

Data Model

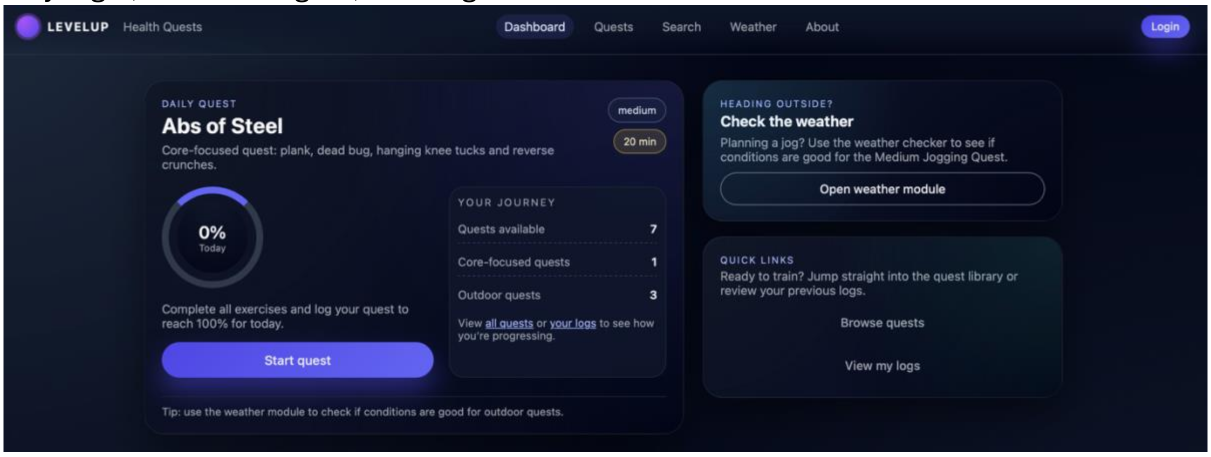
Core entities



User Functionality

<u>Dashboard (/)</u>

The dashboard element of the web app presents a “Daily Quest” card. This greets the user with a nice dashboard interface showcasing a quests name, description, difficulty and target minutes. Below that, it displays instances of all quests, core-focused quests and outdoor quests, all fetched from MySQL via routes/main.js. The app also features a right-hand sidebar which offers quick links to the weather module and quest library. The navigation bar is consistent across pages and adapts to login status showing greeting, “My logs”, “Audit” for gold, and Logout when authenticated.



DAILY QUEST
Abs of Steel
Core-focused quest: plank, dead bug, hanging knee tucks and reverse crunches.
medium
20 min
0%
Today
Complete all exercises and log your quest to reach 100% for today.
Start quest

YOUR JOURNEY	
Quests available	7
Core-focused quests	1
Outdoor quests	3

View all quests or your logs to see how you're progressing.

Tip: use the weather module to check if conditions are good for outdoor quests.

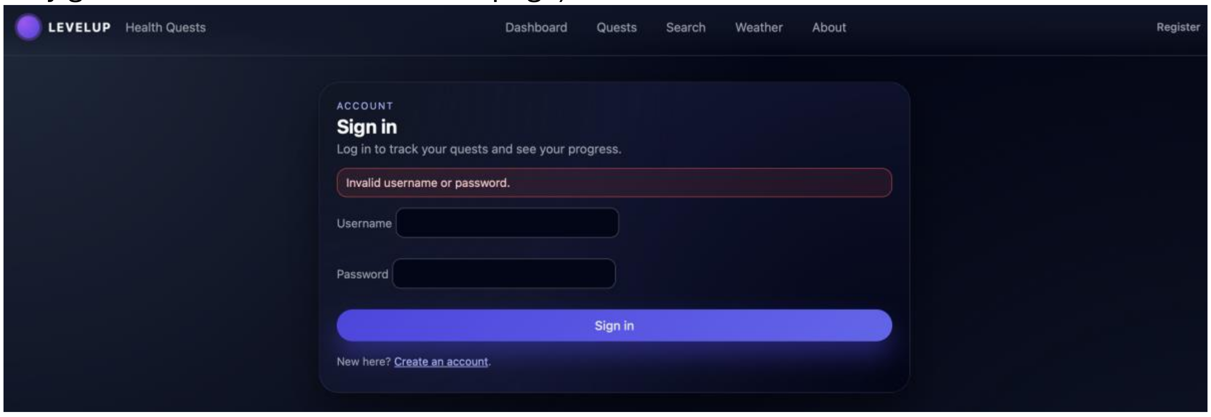
HEADING OUTSIDE?
Check the weather
Planning a jog? Use the weather checker to see if conditions are good for the Medium Jogging Quest.
Open weather module

QUICK LINKS
Ready to train? Jump straight into the quest library or review your previous logs.
Browse quests
View my logs

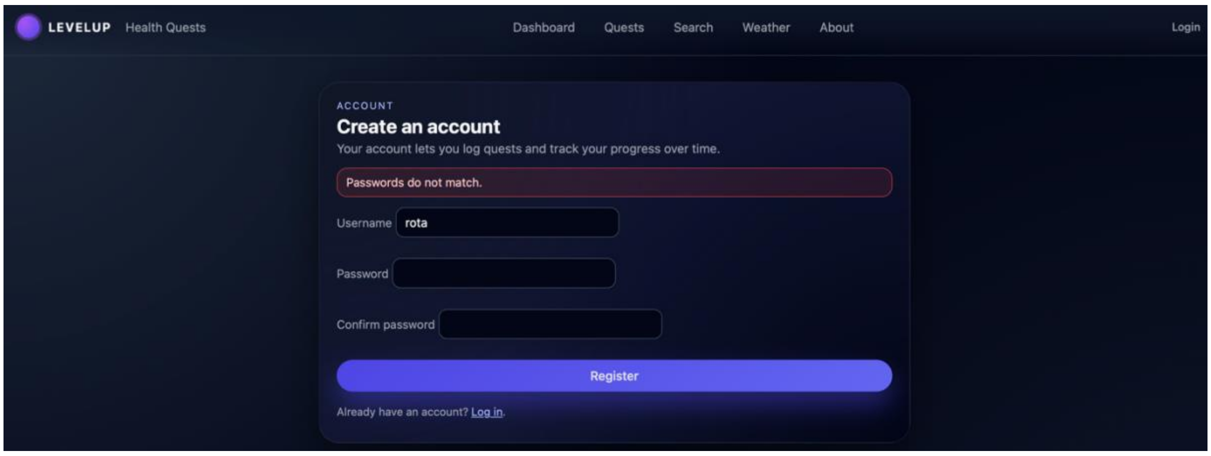
Screenshot: Dashboard showcasing daily quest card, stats and top navigation.

Accounts and user authentication

At /users/register, users are able to create an account with validation for username length, password length and password confirmation. On /users/login, credentials are checked using bcrypt. When login succeeds, the user’s id and username are stored in the session. A redirectLogin middleware in routes/users.js and routes/quests.js protects routes such as **Log completion**, **My logs**, and **Audit**, ensuring only logged-in users (and only gold as the admin for the audit page) can access them.



ACCOUNT
Sign in
Log in to track your quests and see your progress.
Invalid username or password.
Username
Password
Sign in
New here? Create an account.



ACCOUNT

Create an account

Your account lets you log quests and track your progress over time.

Passwords do not match.

Username	rota
Password	
Confirm password	

Already have an account? Log in.

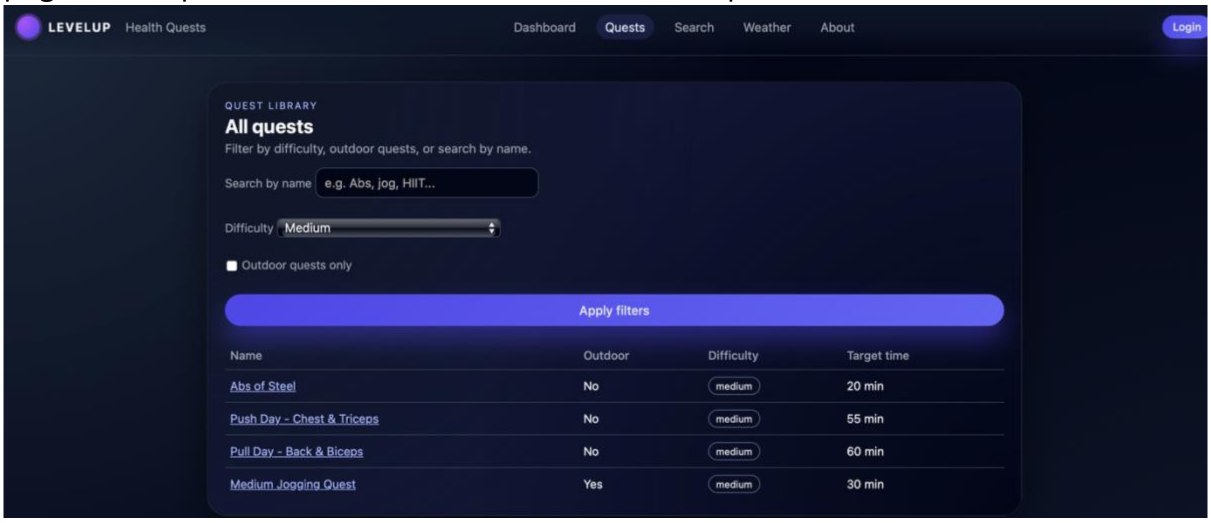
Screenshot: The Login and registration forms with an error message examples for sanity checks.

Quest library, search and quest detail

The quest library at /quests lists all quests in a table. Users can filter these by:

- Searching a term like the quest's name,
- Difficulty (easy/medium/hard),
- Outdoor-only checkbox.

These filters are applied in routes/quests.js using parameterised SELECT queries with optional WHERE clauses. Clicking upon a quest name opens /quests/:id, which displays the full details and associated exercises fetched via a join on quest_exercises and exercises. Outdoor quests include a prompt to check the weather first. The /search page is a simplified front-end that submits filters to /quests.



QUEST LIBRARY

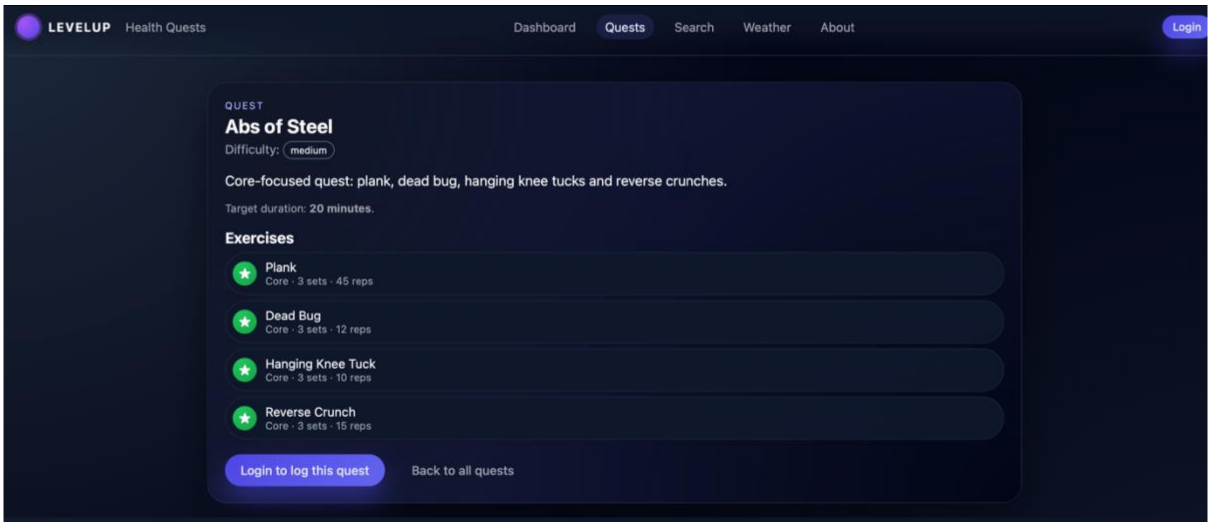
All quests

Filter by difficulty, outdoor quests, or search by name.

Search by name	e.g. Abs, jog, HIIT...
Difficulty	Medium

Outdoor quests only

Name	Outdoor	Difficulty	Target time
Abs of Steel	No	medium	20 min
Push Day - Chest & Triceps	No	medium	55 min
Pull Day - Back & Biceps	No	medium	60 min
Medium Jogging Quest	Yes	medium	30 min



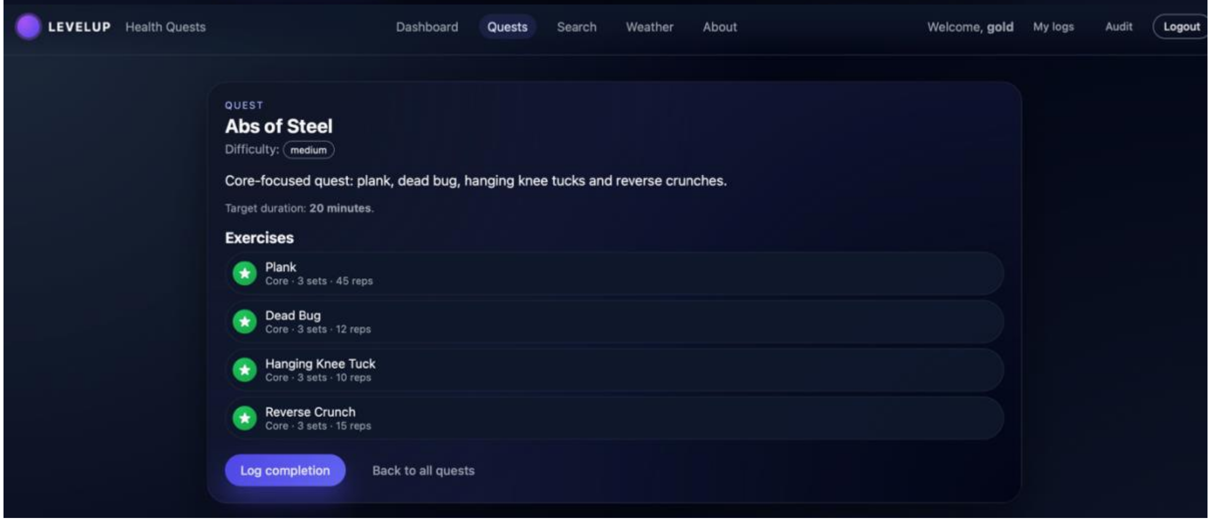
QUEST
Abs of Steel
Difficulty: medium

Core-focused quest: plank, dead bug, hanging knee tucks and reverse crunches.

Target duration: **20 minutes**.

Exercises

- Plank
  Core · 3 sets · 45 reps
- Dead Bug
  Core · 3 sets · 12 reps
- Hanging Knee Tuck
  Core · 3 sets · 10 reps
- Reverse Crunch
  Core · 3 sets · 15 reps



QUEST
Abs of Steel
Difficulty: medium

Core-focused quest: plank, dead bug, hanging knee tucks and reverse crunches.

Target duration: **20 minutes**.

Exercises

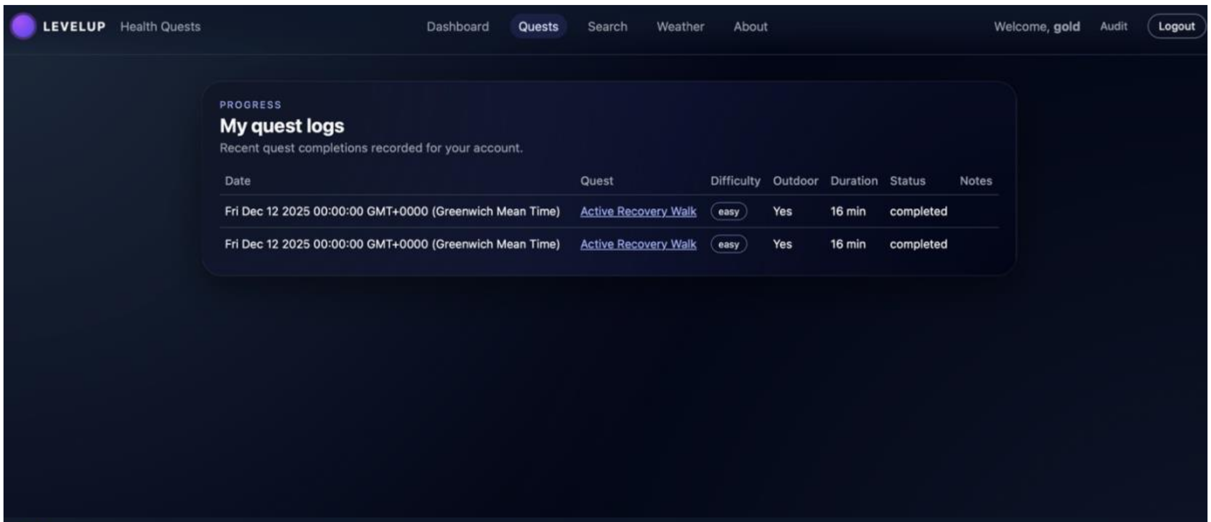
- Plank
  Core · 3 sets · 45 reps
- Dead Bug
  Core · 3 sets · 12 reps
- Hanging Knee Tuck
  Core · 3 sets · 10 reps
- Reverse Crunch
  Core · 3 sets · 15 reps

Screenshot: Quests page with a filter applied and its form and results table.
Screenshot: Quest detail page with exercises list and "Log completion" button, while logged in and one without.

Logging completions and viewing history

In order to see the logs the app opens from a quest detail page, the **Log completion** button which opens exactly quests/:id/log. This form allows the user to select the date, a duration in minutes, a status like completed/failed/abandoned, and free-text notes in order to log a quests done for that day and anything else they might've experience then. On submit, the routes/quests.js inserts a new row into quest_logs for the current user_id and quest_id, and then redirects back to the quest details.

The **My logs** page at /quests/me/logs shows a joined view of each user's quest_logs with quest metadata (name, difficulty, outdoor flag). This gives a simple history of training sessions and outcomes.

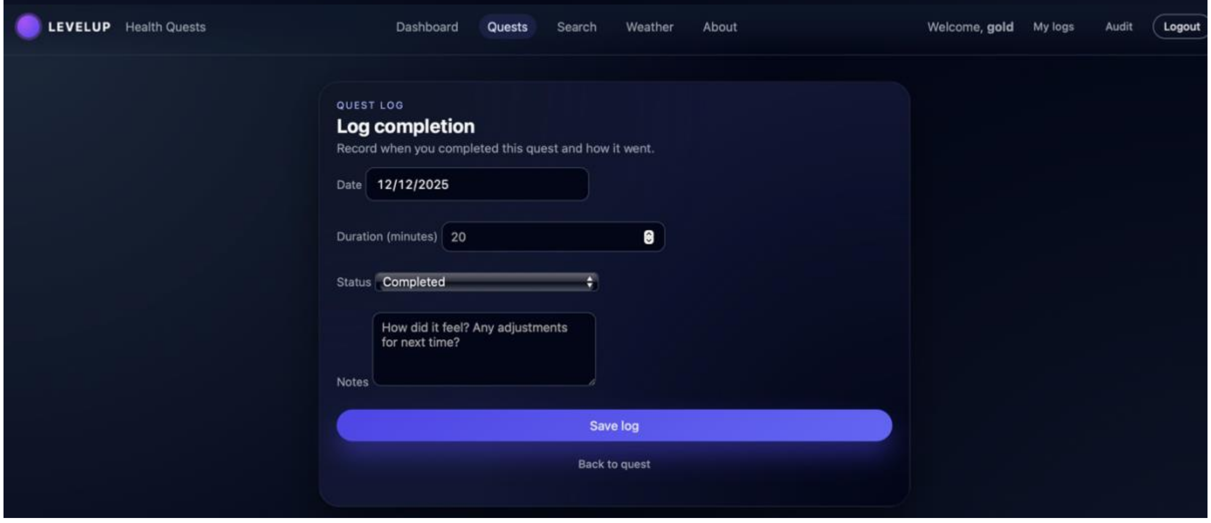


PROGRESS

My quest logs

Recent quest completions recorded for your account.

Date	Quest	Difficulty	Outdoor	Duration	Status	Notes
Fri Dec 12 2025 00:00:00 GMT+0000 (Greenwich Mean Time)	Active Recovery Walk	easy	Yes	16 min	completed	
Fri Dec 12 2025 00:00:00 GMT+0000 (Greenwich Mean Time)	Active Recovery Walk	easy	Yes	16 min	completed	



QUEST LOG

Log completion

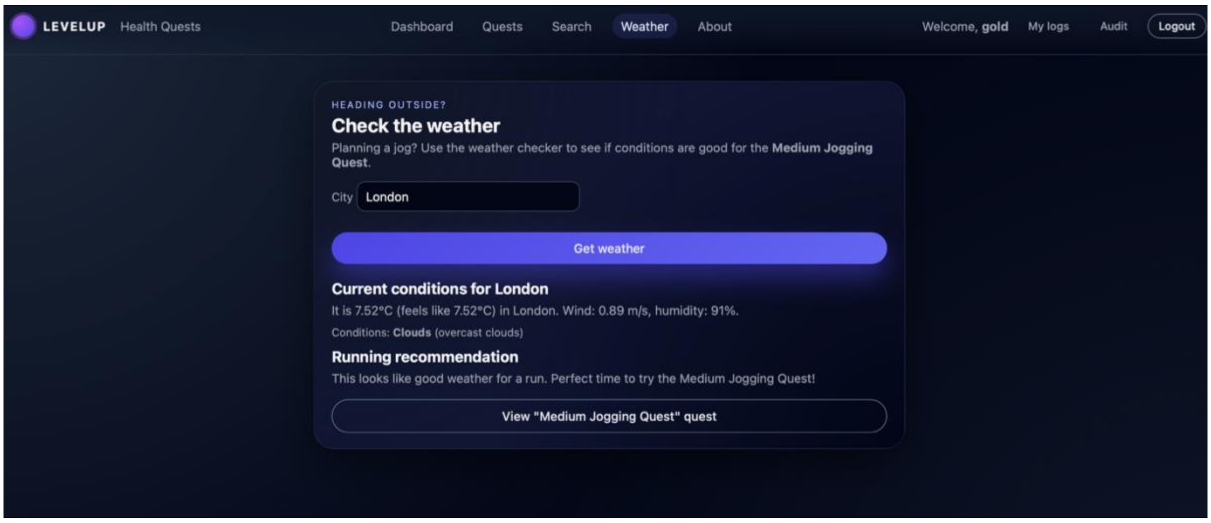
Record when you completed this quest and how it went.

Date	12/12/2025
Duration (minutes)	20
Status	Completed
Notes	How did it feel? Any adjustments for next time?

Screenshot: Log completion form.
Screenshot: My logs page showing recent quest logs.

Weather module

The weather page at /weather lets users enter any city they would like. Then the server sanitises the input, calls the OpenWeather API using the configured API key, parses the JSON and displays a summary including: temperature, feels-like temperature, wind and humidity. Basic logic behind it classifies the conditions as too cold/hot, storming, raining and then it generates a friendly running recommendation, for example suggesting indoor training or encouraging an outdoor jog. I've coded one example in which if "Medium Jogging Quest" exists in quests, the page links directly to it.


LEVELUP Health Quests

Dashboard Quests Search Weather About

Welcome, gold My logs Audit Logout

HEADING OUTSIDE?

Check the weather

Planning a jog? Use the weather checker to see if conditions are good for the **Medium Jogging Quest**.

City: London

Get weather

Current conditions for London

It is 7.52°C (feels like 7.52°C) in London. Wind: 0.89 m/s, humidity: 91%.

Conditions: **Clouds** (overcast clouds)

Running recommendation

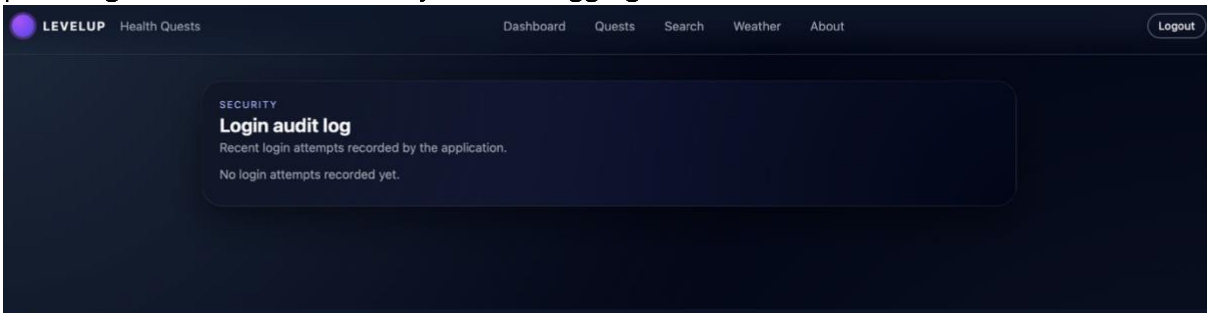
This looks like good weather for a run. Perfect time to try the Medium Jogging Quest!

View "Medium Jogging Quest" quest

Screenshot: Weather page with current conditions and jogging recommendation.

About and audit

The About page (/about) explains the purpose of the LevelUp- Health application and its tech stack. The audit page (/users/audit), is restricted only to the gold user or an admin if you will, and it lists recent login events from login_audit so people with admin priviledges can see the security behind logging in.


LEVELUP Health Quests

Dashboard Quests Search Weather About

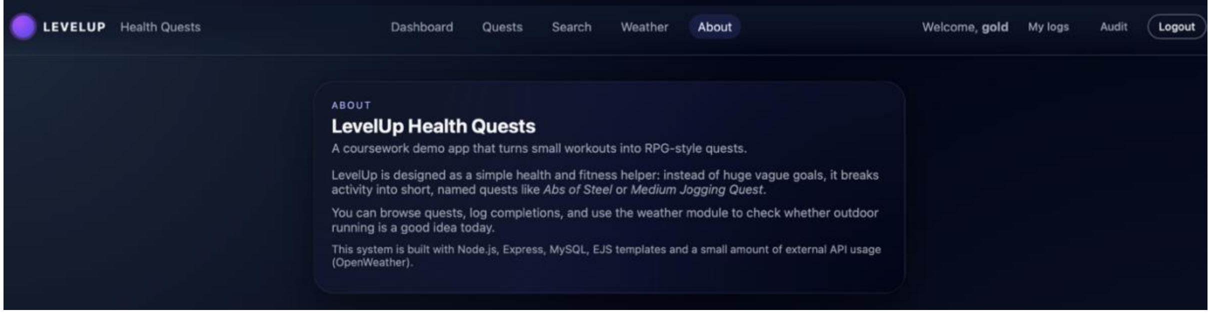
Logout

SECURITY

Login audit log

Recent login attempts recorded by the application.

No login attempts recorded yet.


LEVELUP Health Quests

Dashboard Quests Search Weather About

Welcome, gold My logs Audit Logout

ABOUT

LevelUp Health Quests

A coursework demo app that turns small workouts into RPG-style quests.

LevelUp is designed as a simple health and fitness helper: instead of huge vague goals, it breaks activity into short, named quests like *Abs of Steel* or *Medium Jogging Quest*.

You can browse quests, log completions, and use the weather module to check whether outdoor running is a good idea today.

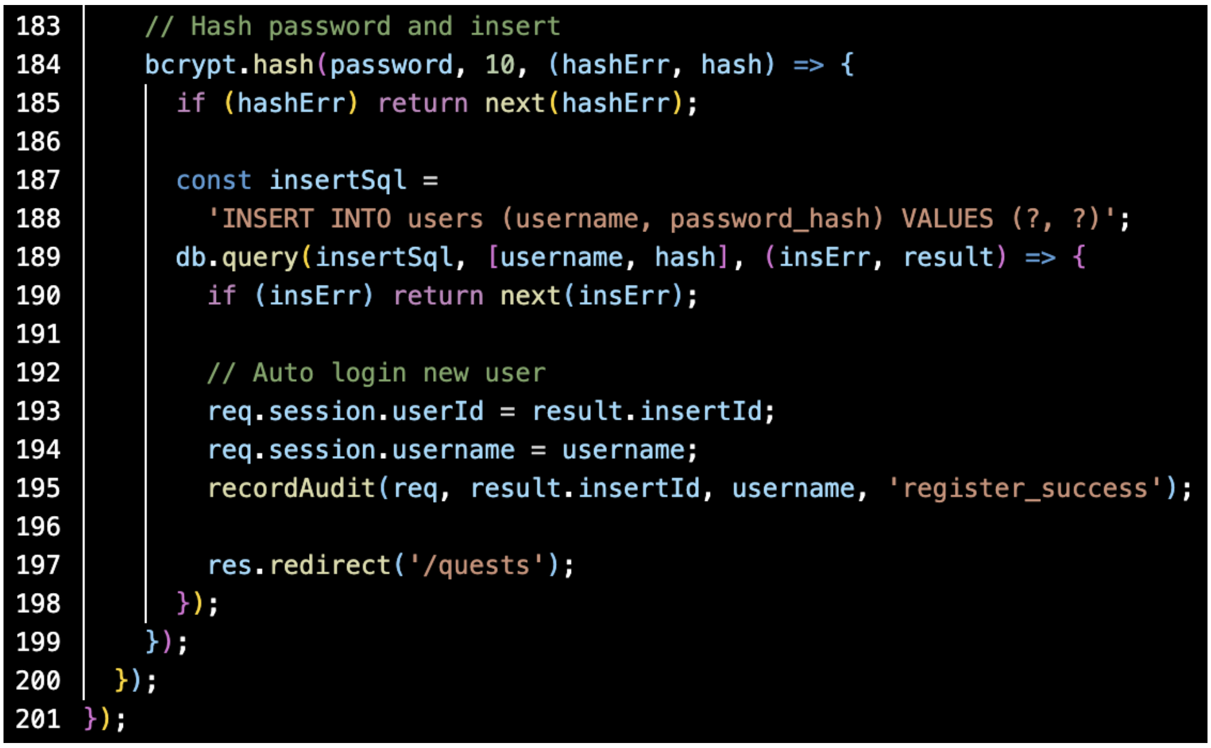
This system is built with Node.js, Express, MySQL, EJS templates and a small amount of external API usage (OpenWeather).

Screenshot: About page and Audit table.

Advanced Techniques

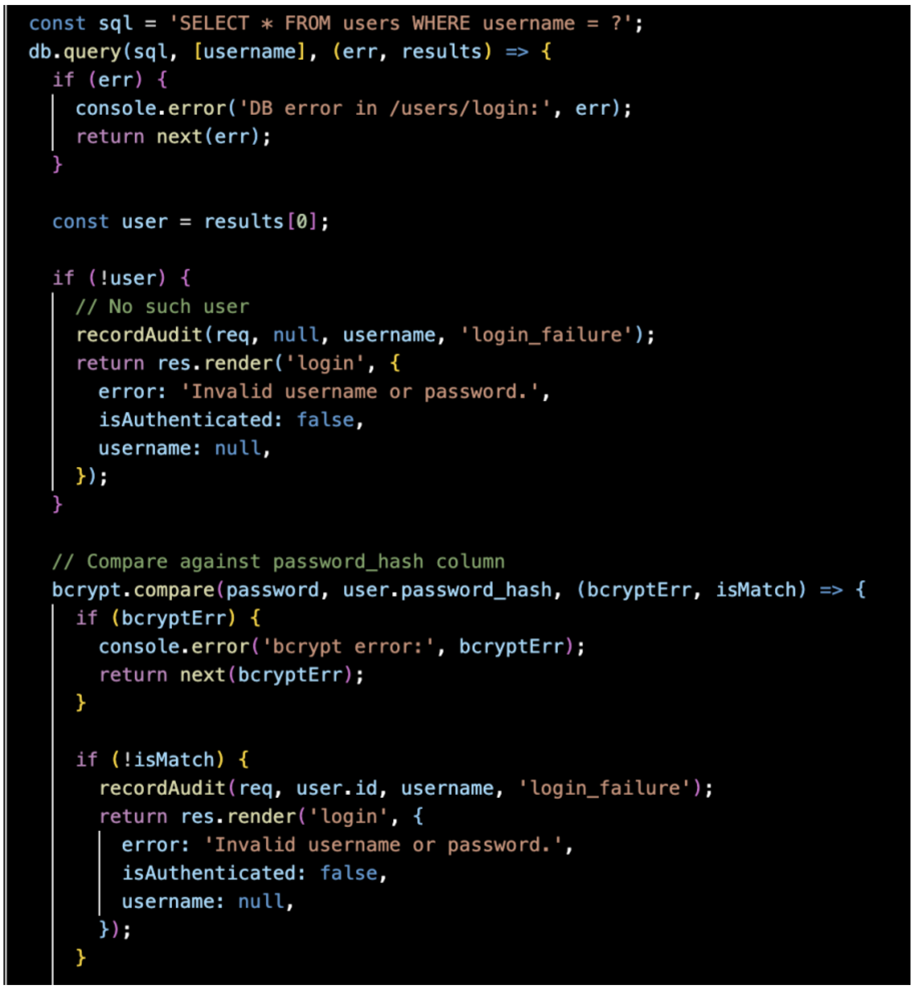
1. Secure password handling with bcrypt and sessions

User passwords are never stored in plaintext. In routes/users.js, registration hashes all passwords with bcrypt:



```
    // Hash password and insert
    bcrypt.hash(password, 10, (hashErr, hash) => {
      if (hashErr) return next(hashErr);

      const insertSql =
        'INSERT INTO users (username, password_hash) VALUES (?, ?)';
      db.query(insertSql, [username, hash], (insErr, result) => {
        if (insErr) return next(insErr);

        // Auto login new user
        req.session.userId = result.insertId;
        req.session.username = username;
        recordAudit(req, result.insertId, username, 'register_success');

        res.redirect('/quests');
      });
    });
  });
});
```

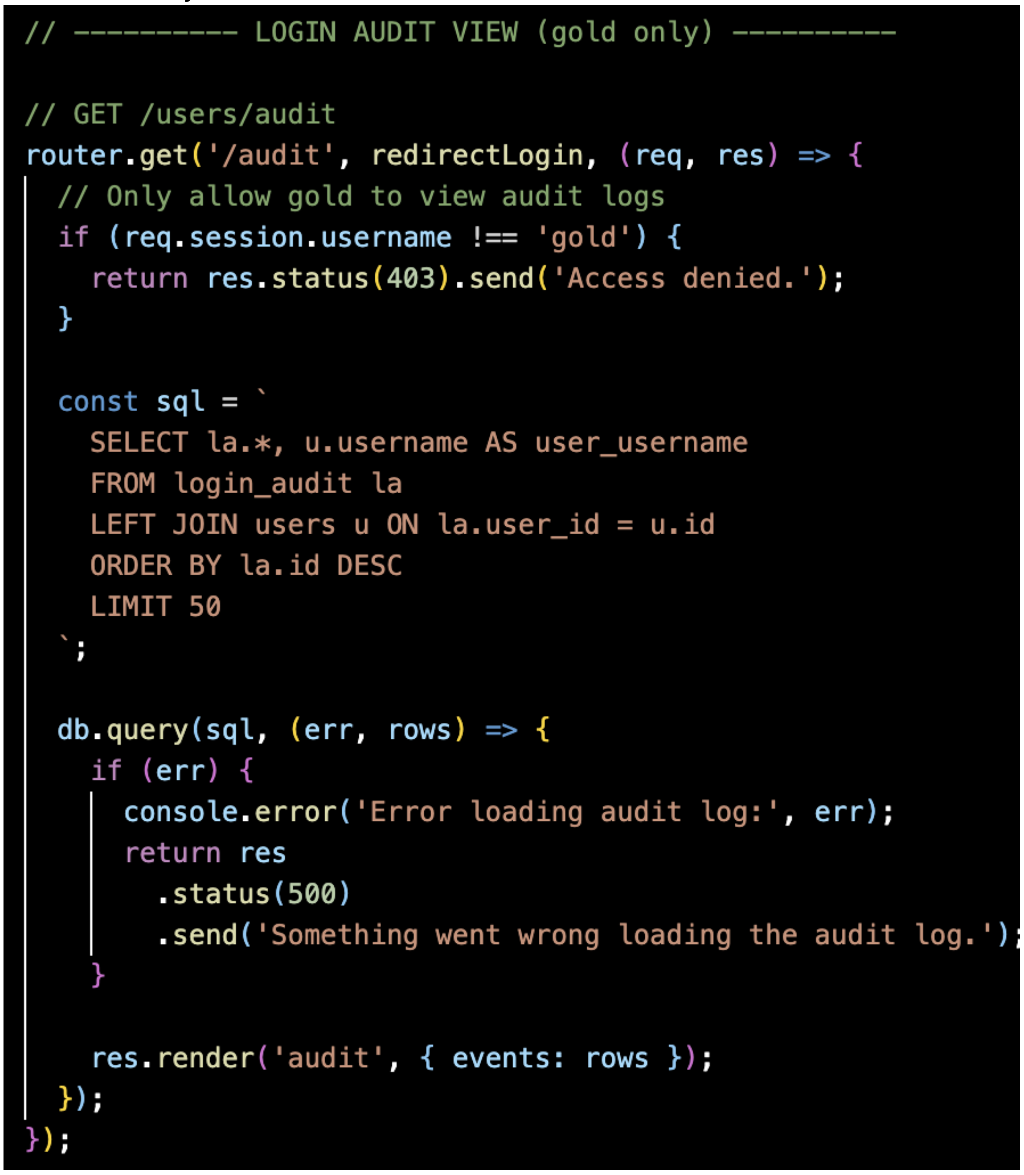
Login part then compares the supplied password with the stored hash and sets session variables:



```
const sql = 'SELECT * FROM users WHERE username = ?';
db.query(sql, [username], (err, results) => {
  if (err) {
    console.error('DB error in /users/login:', err);
    return next(err);
  }

  const user = results[0];

  if (!user) {
    // No such user
    recordAudit(req, null, username, 'login_failure');
    return res.render('login', {
      error: 'Invalid username or password.',
      isAuthenticated: false,
      username: null,
    });
  }

  // Compare against password_hash column
  bcrypt.compare(password, user.password_hash, (bcryptErr, isMatch) => {
    if (bcryptErr) {
      console.error('bcrypt error:', bcryptErr);
      return next(bcryptErr);
    }

    if (!isMatch) {
      recordAudit(req, user.id, username, 'login_failure');
      return res.render('login', {
        error: 'Invalid username or password.',
        isAuthenticated: false,
        username: null,
      });
    }
```

2. Robust login audit logging

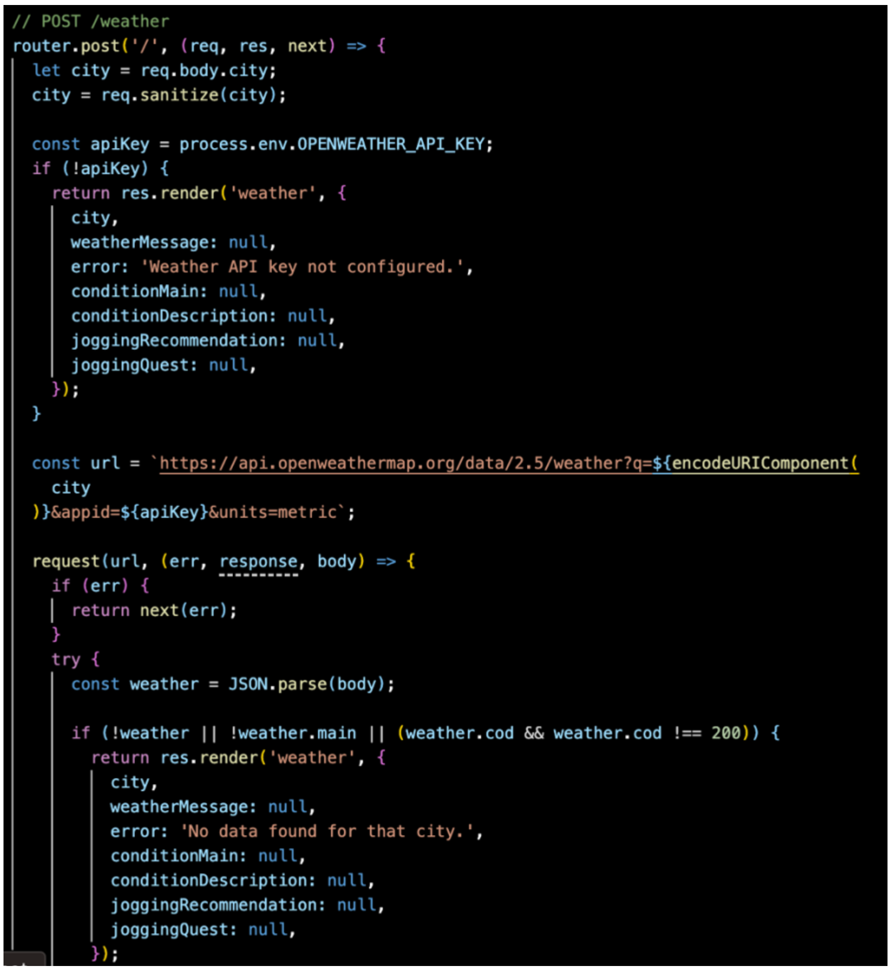
Every time someone logs in successfully, every time someone fails and logs out is logged in login_audit with timestamp, IP and user agent. A helper function in routes/users.js centralises this:



```
// ---------- LOGIN AUDIT VIEW (gold only) ----------

// GET /users/audit
router.get('/audit', redirectLogin, (req, res) => {
  // Only allow gold to view audit logs
  if (req.session.username !== 'gold') {
    return res.status(403).send('Access denied.');
  }

  const sql = `
    SELECT la.*, u.username AS user_username
    FROM login_audit la
    LEFT JOIN users u ON la.user_id = u.id
    ORDER BY la.id DESC
    LIMIT 50
  `;

  db.query(sql, (err, rows) => {
    if (err) {
      console.error('Error loading audit log:', err);
      return res
        .status(500)
        .send('Something went wrong loading the audit log.')
    }

    res.render('audit', { events: rows });
  });
});
```

3. External API integration and input sanitisation

The weather route demonstrates server-side HTTP integration and safe handling of user input, see below:



```
// POST /weather
router.post('/', (req, res, next) => {
  let city = req.body.city;
  city = req.sanitize(city);

  const apiKey = process.env.OPENWEATHER_API_KEY;
  if (!apiKey) {
    return res.render('weather', {
      city,
      weatherMessage: null,
      error: 'Weather API key not configured.',
      conditionMain: null,
      conditionDescription: null,
      joggingRecommendation: null,
      joggingQuest: null,
    });
  }

  const url = `https://api.openweathermap.org/data/2.5/weather?q=${encodeURIComponent(
    city
  )}&appid=${apiKey}&units=metric`;

  request(url, (err, response, body) => {
    if (err) {
      return next(err);
    }
    try {
      const weather = JSON.parse(body);

      if (!weather || !weather.main || (weather.cod && weather.cod !== 200)) {
        return res.render('weather', {
          city,
          weatherMessage: null,
          error: 'No data found for that city.',
          conditionMain: null,
          conditionDescription: null,
          joggingRecommendation: null,
          joggingQuest: null,
        });
```

Using req.sanitize, encodeURIComponent and environment variables demonstrates awareness of injection risks and attacks as well as configuration management.

AI Declaration

I used AI ChatGPT as a supporting tool during this assignment in the following ways:

- **Design and planning:** it helped me refine the concept of the LevelUp app, decide on the main features such as quests, quest logs, weather module, daily quest feature.
- **Debugging and refactoring:** to diagnose and fix issues such as broken EJS paths, base-path problems on the VM, and missing template variables. I asked for explanations of error messages and then applied and tested the suggested fixes afterwards. You could say it was considered as a part time tutor.
- **Documentation:** I've utilised Ai here to produce the best documentation template for an application like this.

Any content that was Ai generated was adapted, reviewed and integrated by myself. The final decisions regarding the web app, the debugging or testing process was all my doing. Finally the application runs on the Goldsmiths Virtual Server machine.