Lappeenrannan teknillinen yliopisto
Software Development Skills Full-stack, Online course

Leanne Llena, 001832183
LEARNING DIARY, <CHOSEN MODULE NAME> MODULE

Date : 16.11.2025
Activity : Week 1
This week I started the course by going through the general information and instructions on Moodle so I could understand the expectations. I skimmed through the tutorial videos pretty quickly just to get an overview of the workflow. After that, I jumped straight into brainstorming what kind of app I wanted to build. Because I already took the "Advanced Web Programming" course, I'm familiar with the MERN stack and the typical setup, so getting started felt comfortable. The hardest part was honestly just deciding which project idea I wanted to commit to. Once I settled on building a Calorie Tracker, things became clearer.

Learning outcome:
My previous MERN experience helped a lot. I didn't get stuck on the setup or folder structure, and I was able to start coding right away once I decided on the idea.

Date: 19.11.2025
Activity : Week 2
This week I focused on laying the foundation for the actual Calorie Tracker project. I started by creating a clean MERN structure: setting up the Express backend, connecting MongoDB, preparing React with Vite, and wiring up the basic routes. Once the boilerplate was done, I moved on to implementing core features.

I got the authentication system working — sign up, login, logout — using JWT access tokens and HTTP-only refresh cookies. Getting the refresh logic right took a bit of experimenting, but once it clicked, the whole flow felt much more easy now. On the frontend, I built the main Diary page where the user can log foods into different meals. I didn't have the full UI yet, but I created the basic components, state handling, and the form for adding entries. I also connected the USDA FoodData Central API through the backend, which meant building a search route that fetches food data and normalizes nutrients.

Learning outcome:

I learned how important it is to build reusable logic early (like unified API helpers and backend services). Also, combining external API results with my own database taught me a lot about structuring clean data models. Overall, Week 2 gave me a strong foundation to build the rest of the features.

Date: 24.11.2025

Activity: Week 3

This week was mostly about improving the user experience and making the Diary feel smooth and "alive." I focused a lot on the UI because once the basic features were working, it became obvious which parts felt clunky. I redesigned the Diary page using a cleaner layout with four meal sections (Breakfast, Lunch, Dinner, Snacks). I added instant updates so when you add, edit, or delete a food item, the totals update immediately without refreshing the whole page. That alone made the app feel way more modern. I also started building the Weekly Progress section. My first attempt at a calorie chart looked pretty rough, but I kept iterating. I introduced the KcalBars component to visualize calories per day with a target line, and I added color-coding and hover details to help the user understand their week at a glance. It took a few tries to get the chart to scale properly, but once it worked, it was really satisfying.

On the backend, I added food caching so repeated searches don't always hit the external API. This made everything much faster and more reliable.

Learning outcome:

I learned how much UI/UX matters. Even when the logic works, the app doesn't feel good to use unless the interface is clean and responsive. Rebuilding components instead of patching them is sometimes the better choice.

Date : 29.11.2025

Activity : Week 4

This week I focused on the deployment side and fixing issues that only appear once everything is hosted online. Locally, the app ran perfectly, but in production I ran into two big problems: First problem was refresh tokens weren't being sent because the client and server

were on different domains and. Second problem was Refreshing pages like /diary or /history caused a Vercel 404 instead of showing the React app.

I solved both by adding a vercel.json file that forwards all /api requests through the frontend so the browser sends the cookie and applies a fallback rewrite so every non-API path loads index.html and React Router can take over.

I also polished up the Settings page: added the TDEE calculator, goal options, and the ability to apply the calculated targets directly. This made the app feel more complete and personal — not just a food logger, but something that adapts to the user.

Learning outcome:
I learned a lot about real-world deployment problems, especially with cookies, CORS, and routing when using SPAs. Debugging these issues made me more confident handling production environments and understanding how browsers treat cross-domain requests.

Final week
This week was about polishing the app and bringing all the features together into a smooth final product. Most of the core functionality was already working, so I focused on small improvements and fixing bugs that showed up during everyday use. One of the biggest tasks was refining the Weekly Progress page. I added the MacroTiles component, which shows whether each day's macros were over, under, or within range. It took a bit of tweaking to get the layout readable on both desktop and mobile, but once it clicked, the page felt much clearer. I also added tiny quality-of-life things like remembering the user's last selected week and adding tooltips to explain the colors.

I spent time improving the Settings and History integration. Updating targets now triggers recalculations in the Diary and Progress views without needing a manual refresh. I cleaned up the API routes as well, making them more consistent and easier to maintain. A lot of this wasn't "flashy," but it made the app feel stable and professional**.**

Final reflection

This project ended up being one of the most complete MERN applications I've ever built. I didn't just write code — I dealt with real problems like authentication, API performance, user experience, deployment issues, and data consistency.

Some things felt challenging at first (especially the refresh token behavior across domains), but solving them helped me understand the "why" behind web development instead of just the "how." I also realized that good UX comes from iteration: my early designs were messy, but improving spacing, colors, and components made a huge difference.

Overall, I'm proud of how everything turned out. The app is secure, responsive, good-looking, and actually useful. I feel more confident now in building full-stack applications from scratch, and I know how to ship them into production without things breaking.