Lappeenrannan teknillinen yliopisto

School of Business and Management

Sofware Development Skills

Yuliya Chsherbakova, 003353987

LEARNING DIARY, FULL-STACK MODULE

LEARNING DIARY

23.09.2025

I reviewed the general course information and understood the main objective: to discover my passion as a software developer and create a unique project to showcase my skills. I selected the full-stack module as it offered the most interesting project opportunity. I also began setting up my development environment, choosing VS Code as my primary code editor for this course.

27.09.2025

I started watching the recommended lectures in sequence. I decided to focus on updated video versions and began with Node.js, following along while practicing on my personal laptop.

Node.js
After installation, I encountered environment configuration issues related to working on Windows OS. With AI assistance, I resolved the errors and switched to using the Ubuntu terminal as my main development environment.

MongoDB
After completing the MongoDB tutorial, I created my account and successfully connected to the server through the Ubuntu terminal.

28.09.2025

I watched the first of four videos about the complete MERN project. I wanted to gain a preliminary understanding of how I would apply the knowledge from previous topics in practice.

29.09.2025

ExpressJS

After watching the video, I identified the key concept: If Node.js is the "bare engine" of a car, then ExpressJS is the complete car body with steering wheel, seats, and pedals.

30.09.2025

React learning began with framework fundamentals - understanding virtual DOM and component-based architecture, where I learned to break down interfaces into reusable components. I then delved into components, state, and hooks - mastering functional components, useState for state management, and useEffect for side effects, enabling me to build interactive UIs. Finally, I progressed to events, JSON Server, and routing - implementing user event handling, backend communication through REST API, and configuring client-side routing with React Router, completing the development of a fully-functional SPA application.

15.10.2025

I completed watching the entire MERN project tutorial and began outlining topics for my new application while maintaining the original project foundation.

13.11.2025

I created accounts in Bitbucket and GitHub, then downloaded Sourcetree for version control management.

<u>Backend Infrastructure Development</u>

At this stage, I completed setting up the fundamental architecture for the budget tracker's server-side. Unlike the original goal-based tutorial, I adapted the code for financial management by replacing the "Goal" entity with "Transaction". I built a complete REST API structure with controllers, data models, and routes for transaction management.

<u>Core Mechanism Implementation</u>

I organized error handling through middleware and integrated asyncHandler to simplify asynchronous operations. The server now successfully runs on port 5000 and is ready for database integration. I paid special attention to validation - controllers check required fields to ensure financial data integrity. Currently, the project represents a working API core that can be extended with additional business logic.

The structure is clearly divided into components: routing, business logic, data models, and configuration - this ensures code maintainability as functionality grows.

14.11.2025

<u>Basic Authentication System Implementation</u>

This time I focused on application security. I implemented the protect middleware function that provides user authentication through JWT tokens. Now all financial endpoints will be secured - the system checks Bearer tokens in request headers, verifies their authenticity, and adds user information to the request object.

I paid special attention to error handling: when tokens are missing or invalid, the API returns correct HTTP 401 statuses with clear messages. This creates a reliable foundation for the future user system, where each user will only have access to their own financial data. The middleware is ready for integration with transaction routes after User model implementation.

<u>Complete Authentication System Implementation</u>

I have now completed the development of the authentication system in the budget tracker. I implemented the full interaction chain between frontend and backend: from registration and login forms to user state management through Redux.

On the frontend, I created specialized Register, Login, and Dashboard components, integrated with Redux Store through authSlice. The system uses asynchronous thunk-actions to handle API requests, providing a smooth user experience with loading states and error handling. JWT tokens are stored in localStorage to maintain user sessions.

On the backend, I configured protected routes with authentication middleware that verify token validity before granting access to financial data. Special attention was paid to security - passwords are hashed with bcrypt before saving to the database.

The system ensures reliable access control: each user will only have access to their own transactions, which is critically important for a financial application. The basic infrastructure for personalized budget management experience is ready.

15.11.2025

<u>Complete Transaction Management System</u>

I successfully implemented the full transaction management functionality in the budget tracker application. The system now supports complete CRUD operations for financial transactions, all protected behind authentication middleware.

Implemented Features:
- Full CRUD for transactions
- Protected routes - transactions accessible only to authorized users
- Dashboard with transaction list displaying all user transactions
- Form for adding new transactions with proper validation
- Visual transaction display with correct timestamps and dates
- Seamless frontend-backend integration for real-time data synchronization

Technical Improvements:
- Redux state management for efficient transaction handling
- API service layer (transactionService) for clean separation of concerns
- Comprehensive error handling and loading state management
- Proper component structure following React best practices
- Secure API endpoints with JWT token verification

The application now provides a solid foundation for personal financial management, with each user having their own isolated transaction history and secure access to their financial data.

18.11.2025

<u>Application Core Functionality Implementation</u>

I developed the fundamental application structure based on the tutorial, establishing the basic transaction management system. Configured the initial integration between frontend and backend components, ensuring seamless data flow between the React interface and Node.js server.

Implemented core CRUD operations for comprehensive transaction management, creating a functional prototype that handles the complete lifecycle of financial records from creation to deletion.

Results:
- Working prototype application with transaction addition and display capabilities
- Basic component structure and state management foundation
- Prepared groundwork for future functionality expansion
- Established reliable communication between client and server

19.11.2025

<u>User Interface Enhancement and Documentation</u>

I designed and implemented an attractive welcome page (Welcome page) featuring modern gradient design and responsive layout. Enhanced the overall project visualization with improved UI/UX principles, creating a more engaging user experience.

Completely updated project documentation with comprehensive installation and running instructions. Added video demonstration link to showcase application functionality and provide visual guidance for users.

Results:
- Landing page with contemporary gradient design aesthetic

- Fully formatted README with detailed setup and execution guidelines
- Completed video demonstration showcasing application capabilities