Lappeenrannan teknillinen yliopisto

School of Business and Management

Sofware Development Skills

<Kim Yukyeong>, <x080298>

LEARNING DIARY, <Full-Stack 2025-26> MODULE & Project

**LEARNING DIARY**

**12.8.2025**  
I read the general information of the course and set up my environment for the course including local and remote git repository for project and watched additional videos from LUT web application course (lecture videos from 1 to 6; html css and js) about 3.5 hours before starting main course.

I’ve decided to use GitHub as a version control tool and visual studio code for this course, because I am familiar with them already.

I’ve tried to set up bitbucket and SourceTree, but it went wrong during installation, I will try again at some point to get to know something new.

My access to Microsoft (via fitech) office was not granted on day of 12, I wrote this on day of 13.

**13.8.2025**

I continued watching web application course videos (from 7 to11: responsible design, json, materialize, ajax, node.js etc) for a few hours. I’ve learned/introduced a responsive CSS framework (Materialize) and how it works in general.

And I started watching course video the first module, Node.js crash course, followed his instruction and created first server from scratch successfully.

**14. 8.2025**

I’ve finished watching first module NodeJS - Node.js crash course - and have successfully performed coding exercises in tutorial and committed my work to remote git repositories (GitHub).

I’ve learned core modules of Node.js such as path, url, fs, os, process cryption and events.

**15.8.2025**

I’ve learned MongoDB.

I created MongoDB user account and created free tier cluster and set up user for database access and installed MongoDB shell to connect to cluster (database) from my local machine.

Below how i connected to the MongoDB database with Mongo shell (mongosh).



I’ve performed some of frequently used MongoDB queries such as insert, find, update and delete (CRUD) through mongosh.

insert() – Create

find() – Read

update() – Update

delete() – Delete



**16.8.2025**

Learning MongoDB continued.

I loaded sample dataset of airbnb and installed the **MongoDB Compass**, the GUI connection tool for MongoDB for aggregations (set up pipeline of stages).

Finally, I installed MongoDB for VS Code extension to use MangoDB in testing application using airbnb samples and aggregations.

I’ve performed simple app of printing out 20 airbnb results successfully, and committed to git repository.

Aggregations in our project below – copy of course materials.



**18.8.2025 (MA)**

I’ve learned Express web framework.

I’ve watched lecture video (the crash course) and successfully copied main part of the lecture (express project) and will continue ejs(embedded javascript, the templating engine) part tomorrow.

**npm install express**

**node –watch –env-file=.env server.js**

--watch is flag for auto-re-running server after editing/saving in development stage

--env-file=.env with this flag, environment variables(.env) can be used in application.

Important topics in this course are routinsg, request&response, usage of middlewares, controllers and error handling, mainly focused on backend side.

**19.8.2025 (TI)**

I’ve learned ejs and committed course work to git remote repository.

**npm install ejs**

**<%= EJS %>**

**20.8.2025 (KE)**

I’ve learned ReactJS.

Installed extensions: ***React Development Tools*** for chrome browser

***ES7+ React/Redux/React-Native snippets***for VS code (rafce)

I started a new react app using below commands

**npx create-react-app react-task-tracker**

**cd react-task-tracker**

**npm start**

This (npm start) creates a simple app with no content to start with.

Application is running on localhost:3000 by npm start, this is single page (index.html & index.js) application and various components (\*.js) are used to create diffent views.

Additional packages installed: **react-icons, json-server, react-router-dom**

package.json for json server

**”server”: ”json-server -- watch db.json -- port 5000”**

json-server is running on localhost:5000

**npm run server**

After server running, this app starts fetching (json) data from server and CRUD features are implemented. I skipped *production build* part.

useEffect( async () => { ... }) 🡺 cannot use async directly,

useEffect( () => { ... async () ... }

**21.8.2025 (TO)**

I’ve cleaned up and fixed my code (ReactJS: Task Tracker application), because lecture video content is old (v2021) and i found that some of codes are not working any more, and finally committed my work(ReactJS) to remote repository.

I started watching the last chapter: MERN project.

**22.8.2025 (PE)**

Installed packages for MERN projects

*For backend*

PS C:\Users\ugkim\lut\mern> npm i express mongoose colors jsonwebtoken bcryptjs

*For frontend*

PS C:\Users\ugkim\lut\mern> npm config set legacy-peer-deps true

PS C:\Users\ugkim\lut\mern> npx create-react-app frontend --template redux

PS C:\Users\ugkim\lut\mern\frontend> npm install ajv@latest ajv-keywords@latest

PS C:\Users\ugkim\lut\mern\frontend> npm i react-router-dom

PS C:\Users\ugkim\lut\mern\frontend> npm i react-icons

PS C:\Users\ugkim\lut\mern\frontend> npm i axios react-toastify

**frontend\package.json\scripts:**   "proxy": "http://localhost:5000",

*For running concurrently backend server and frontend app*

PS C:\Users\ugkim\lut\mern> npm i -D concurrently

**package.json\scripts**: "start": "node --watch --env-file=.env backend/server",

    "client": "npm start --prefix frontend",

    "dev": "concurrently \"npm start\" \"npm run client\""

PS C:\Users\ugkim\lut\mern>npm run dev

Access-Control-Allow-Origin: <https://yourdomain.com>

Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS

Access-Control-Allow-Headers: Content-Type, Authorization

Backend coding part like connection to database server and crud functionality using curl commands (or postman, REST client http) completed without problem.

**25-26.8.2025 (finding a bug)**

I’ve learned below:

**Backend: JsonWebToken (jwt), crypts (password)**

**Redux/toolkits: how to use global state, slice & store**

**Axios**: promise-based HTTP Client for node.js and the browser.

On the server-side it uses the native node.js http module, while on the client (browser) it uses XMLHttpRequest.

**27.8.2025**

Frontend react project with axios tool has below warning message, even though project is working as it should be. I struggled for couple of days with one tiny typo mistake and found and fixed it eventually and committed last chapter of the course.



To solve (get rid of) this warning, I’ve tried new approach with vite build tool with React plugin and have learned Vite build tool. Problem (warnings) solved.

**1.9.2025**

I started project work. I decided to create web application based on the MERN authentication project below, which was created by same lecturer of course.

*https://www.traversymedia.com/blog/mern-crash-course-part-1*

*https://github.com/bradtraversy/mern-auth*

My project is very simple memo/note app with basic CRUD functionalities on top of authentication system. Only registered and logged-in users can view own notes and create new ones, edit and delete them.

Details about the project general -how to run and build- are in readme.md in git repository.

**2.9.2025**

Learning how to use bootstrap icon.

A lot of icons are available https://icons.getbootstrap.com/

*### install bootstrap icon*

npm i bootstrap-icons

*### import library on top of xx.js*

import "bootstrap-icons/font/bootstrap-icons.css"

*### in code (icon element)*

<i

className='bi bi-star'

style= { {color: 'green'} }

/>

**3.9.2025**

I’ve learned the basic concept of react-bootstrap for front-end and decided to use it and learned how to reset/clear input fields as initial or default value after form submitting.

There are two different form control system and both approaches are used in this app. The one is fully controlled components, use React state and pass both a value and onChange event handler to trigger the inputs, the value is controlled by the state. Authentication part of the app uses this fully controlled system for example login, sign up & profile forms. In this case it’s not easy to clear/reset to placeholder value after submission. Though login or sign up process, there is no need to reset this input field.

### *form element for login/signup/profile*

<Form.Control

type='name'

placeholder='Enter name'

value={name}

onChange={(e) => setName(e.target.value)}

/>

The other approach is un-controlled components, do not use React state instead use the form element’s onSubmit handler to access the form values and reset the form. For creating note, this approach is used to clear the previous input and ready for next and return to original value after creating a note successfully.

*### form element for creating new note*

<textarea

        className="form-control" rows="5"

        name='mynote'

        type='textarea'

        placeholder='Write new note here...'

  />

*### onSubmit handler*

const submitHandler = async (e) => {

    e.preventDefault();

    const { mynote } = e.target;

.......

  e.target.reset();

  };

*https://stackoverflow.com/questions/77874051/how-to-clear-input-fields-in-react*

*https://medium.com/@laurens\_mesure/controlled-and-uncontrolled-components-in-react-js-c8c57252a1fb*

**4.9.2025**

I’ve leaned how to update view after creating/updating data in database, without reloading/refreshing page.

This can be achieved with *provides Tags* and *invalidate Tags* in Redux-toolkit

*https://redux.js.org/tutorials/essentials/part-7-rtk-query-basics*

### *notesApiSlice.js*

export const notesApiSlice = apiSlice.injectEndpoints({

  endpoints: (builder) => ({

    getNotes: builder.query({

      query: () => ({

        url: `${NOTES\_URL}`,

        method: 'GET',

      }), providesTags: ['Note']

    }),

    createNote: builder.mutation({

      query: (data) => ({

        url: `${NOTES\_URL}`,

        method: 'POST',

        body: data.content,

      }), invalidatesTags: ['Note']

    }),

    updateNote: builder.mutation({

      query: (data) => ({

        url: `${NOTES\_URL}/${data.id}`,

        method: 'PUT',

        body: data.content,

      }), invalidatesTags: ['Note']

    }),

    deleteNote: builder.mutation({

      query: (data) => ({

        url: `${NOTES\_URL}/${data.id}`,

        method: 'DELETE',

      }), invalidatesTags: ['Note']

    }),

  }),

})

**5.9.2025**

I’ve learned how to route unmatched path using react router dom.

*https://v5.reactrouter.com/web/guides/quick-start*

Path="\*" to element <></>

### *main.jsx*

*<Route path='\*' element={<>do something!</>} />*

**8.9.2025**

I’ve learned how to build project!

*### go to frontend>*

npm run build 🡺 this creates dist folder at frontend root

*### server.js*

if (process.env.NODE\_ENV === 'production') {

  const \_\_dirname = path.resolve();

  app.use(express.static(path.join(\_\_dirname, '/frontend/dist')));

  app.get('\*', (req, res) =>

    res.sendFile(path.resolve(\_\_dirname, 'frontend', 'dist', 'index.html'))

  );

} else {

  app.get('/', (req, res) => {

    res.send('API is running....');

  });

}

*### .env*

NODE\_ENV = production

*### go to main>*

npm run server