Kanban full-stack project report (AWA course project)

Antti Kuru – 061431

Introduction

This document contains an overview of the Kanban full-stack project which was done in the Advanced Web Application course. The project report provides details on what was done, what tools were used and why, AI declaration, and how many points should be gotten from the project.

Description

The project is a full stack application that provides registered and logged in users ability to post cards to a kanban board. Logged in user can create as many columns as possible and add cards into the columns. User can set the following attributes to cards: header, content, estimated time to finish, colour. User can make as many cards he/she wants. Time spent information is displayed automatically at 0h. User can register the time he/she has spent on the task by clicking the "Time spent: X h" field. All other card attributes are editable as well by clicking the attribute e.g. header or content. User is able to save the edit which will handle the edit also in the database. All edits will display instantly. User is able to drag cards between different columns by clicking the card. User can also reorder the cards within a column by clicking "Drag here to change order" text and after a reorder pressing "Save edits" button that will display on the bottom of the column.

User can also create comments to cards. Comments will display in the bottom of the card and also the content in comments is editable. Cards and comments have created/modified at time when the item is created or modified. The time will update correctly after editing some attribute. User can delete columns and cards in his/her own board.

Technologies used

The project uses Node.js and Express.js on the backend, MongoDB as a database, and React with TypeScript in the frontend. Node is very popular JavaScript runtime environment and React is very popular framework in today's web development which makes utilizing them valuable. MongoDB is easy to use document-oriented (non-relational) database program. All used packages are listed in the following list with the explanation why those were used.

Backend packages used:

- express (for server)
- mongoose (for database connection)
- morgan (for easing development with logs)
- cors (for making API calls easier in frontend)
- path (to connect frontend to backend in production mode)
- dotenv (for managing environment variable SECRET in .env file)
- jsonwebtoken (for token-based authorization)
- express-validator (for errors in registering/logging in)

Frontend packages used:

- react-router-dom (for routing in React)
- materialize (for styling the UI)

- framer-motion (for handling reordering)

Root folder packages used:

- concurrently (to start frontend and backend on same command)

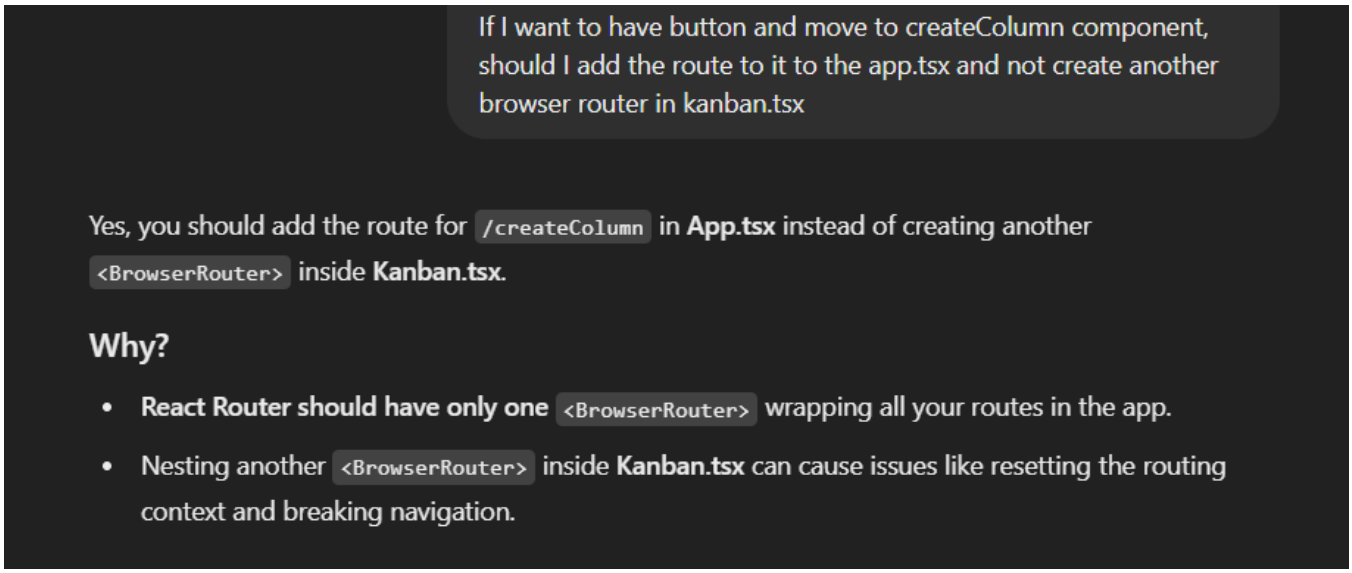
Declaration of AI

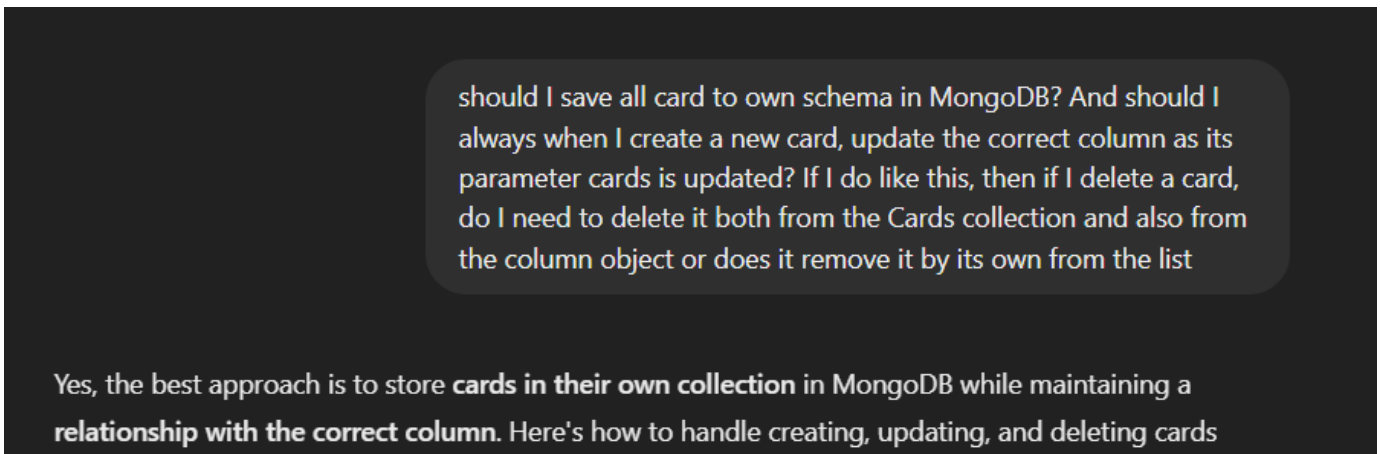
ChatGPT was used during the project development to get ideas for solving some problems that I got stuck in myself, getting confirmation for my own strategy (architectural concept), and explaining errors I didn't understand. ChatGPT also gave me inspiration for styling the cards. I believe I learned a lot by seeing how AI thought on certain things including how to pass props to components and then extending that way on my own.

One key function that ChatGPT gave me was the single function that could handle different attribute edits in the card. I knew it would be inefficient to make own function for every attribute, but I didn't understand how I could have single function that handled different types of edits (text/number).

Largest problem I had was related on dropping card to another column. I got everything else working by myself but deleting the moved card from source column in React. I managed to delete it correctly from the source column's cards list in MongoDB. However, I didn't manage to find out even with using ChatGPT how could I have remove it instantly from the source column in React as cards state parameter always gave the target column's cards. ChatGPT was able to make a cheeky way to handle it by using querySelector and deleting it. However, a warning will raise if you drag card to another column and then back to the original column stating that unique key was found already in the column. Nevertheless, the dragging between columns works flawlessly.

Below are some discussions with ChatGPT I had during the development process.





it says it cant find fetchCards when I put it to the onDelete={}

The error you're encountering happens because `fetchCards` is defined inside the `Column` component, and it's not available in the scope where you're trying to use it for the `onDelete` prop of the `Card` component.

To fix this, you should pass a function to `Card` that allows you to re-fetch the cards from the `Column` component. This can be done by defining `fetchCards` inside `Column` and passing it down to `Card` as a prop.

Solution:

1. **Define `fetchCards` inside the `Column` component.**
2. **Pass `fetchCards` as a prop to `Card`** so that it can be called when a card is deleted.

Here's how you can modify your code:

Updated `Column` component

tsx Kopioi Muokkaa

```tsx
import { useState, useEffect } from 'react'
import Card, { ICard } from './Card'
import { useNavigate } from 'react-router-dom'
```

The target column is updated correctly, how can I delete the moved card from source column just in frontend as it does the deletion from the database but fetchCards doesn't update the UI as it is async, so I need some cheeky way just to remove the dragged item from source column

You're right—since `fetchCards` is async, it doesn't immediately update the UI. To handle this smoothly, you can **optimistically update** the state by removing the dragged card from the source column before the backend operation completes.

Kopioi Muokkaa

```tsx
const handleOnDrop = async (e: React.DragEvent) => {
    e.preventDefault();

    const cardId = e.dataTransfer.getData("CardId");
    const sourceColumnId = e.dataTransfer.getData("OriginalColumnId");
    const cardString = e.dataTransfer.getData("Card");

    if (!cardId || !sourceColumnId || !cardString) return;

    const card: ICard = JSON.parse(cardString);

    if (sourceColumnId !== column._id) {
        console.log("Removing from source column immediately...");

        // Remove the card from the source column in the frontend
        const sourceColumnElement = document.querySelector(`[data-column-id="${sourceColumnId
        if (sourceColumnElement) {
            const cardElement = sourceColumnElement.querySelector(`[data-card-id="${cardId}"]
            if (cardElement) {
                cardElement.remove(); // Instantly removes the card from the DOM
            }
```

Installation guide and user manual

Installation guide

1. Open IDE (Visual Studio Code etc.)
2. Create empty folder
3. Open created folder and open terminal
4. type *git clone* *https://github.com/antti-kuru/Kanban_full-stack.git*
5. move to kanban_full-stack folder by *cd kanban_full-stack/*
6. install all packages by typing *npm install* in the kanban_full-stack/ folder
7. start application by typing *npm start dev* (this starts both frontend and backend)
8. click the http://localhost:3000/ link in the terminal
9. you are good to go

User manual

1. First you have option to login or register. If you open the app for the first time, you don't have existing account so **at the first-time press Register**
2. Create an account and be sure to follow the criteria for username and password. If one of the criteria is not met, you are unable to register.
3. After successful register you are directed to login page where you log in with you created account
4. After successful login you are directed to Kanban Board page
5. Start adding columns from "Add column" button
6. Start adding cards to columns from "Add card" button / delete created columns from "Delete column" button
7. To reorder
    a. press "Drag here to change order" text
    b. Reorder cards by your liking
    c. press "**Save edits"** button that will display below the cards in the column
8. To drag cards between columns
    a. press some portion of the card and drag it to another column
9. Delete card
    a. press "Delete card" Button
10. Add comment
    a. press "Add comment" button
    b. Type content of the comment and submit
11. Rename any attribute
    a. press any attribute you want to edit and press "Save"
12. Logout
    a. press "Logout" in the navigation bar

Accessibility testing

I tested accessibility of the app with keyboard and voice command. With keyboard, I was able to register and login by pressing Tab to get to right place. After that I couldn't get to "Add column" button so it failed here. On voice command test I used Microsoft Speech Recognition. I was able to open google chrome and navigate to localhost:3000/ but after that I couldn't make the recognition software to click any button in the web page.

Points table

Feature	**Requested points / Max points**
Basic features with well written documentation	25 / 25
Utilization of a frontside framework (React)	3 / 3
Cards can be reordered with drag and drop	2 / 2
User can set the colour of a card	1 / 1
Test software for accessibility: can it be used only with keyboard / voice command? Can screen readers work with your application?	2 / 3
User has the option just to double click any edible content (like header or card description) and edit it	4 / 4
Cards can have comments in them, one or many	3 / 3
Cards and comments have visible timestamps when they have been created and updated	4 / 3
Cards have estimated time, when the work is done	1 / 1
User is able to register the time she has spended with the task/card	1 / 1
All other features	0 / 3
Total	**46 / 50**