Name: Matheus Magalhães

Course: Advanced Web App

Kanban Board – Track and Plan

Github - https://github.com/theusmagal/Track-and-Plan-WebApp

Youtube - https://www.youtube.com/watch?v=vw8F2Hl773o

Project Overview

The idea of this project was to implement a Kanban board webpage. A Kanban board system, users can register/login and create personal boards to manage and organize better daily/work/life tasks and plans. Users can have multiple boards, containing columns. Each column can have movable and editable cards. For example, the user can have a board for work and another board for personal life. This way, it is easy for him/her to track information and plan the future!

Techologies Used

- **Front-End:**

React and TypeScript: Used to build a dynamic page.

Tailwind CSS: For fast, precise and mobile first styling without writing custom CSS.

@hello-pangea/dnd: A React library for state management and drag and drop. It takes care of the movement between cards and columns for a smooth control.

- **Back-End:**

Node.js, Express and TypeScript: A server environment with Express to create REST API routes. TypeScript was used to ensure safety in routes logics and middleware.

PostgreSQL / Prisma ORM (hosted on Railway cloud platform): Realiable SQL based relational database. It is ideal for applications like a Kanban board where data needs to be consistently maintained.

Prisma ORM was chosen for easy integration with TypeScript, making it good to work with database. Data is sent as JSON from the frontend and Prisma handles all the reading and writing in database converting it to SQL queries for good communication.

JSON Web Tokens JWT: Secure authentication, sending signed tokens to ensure safety.

Dotenv: Used for environment/variables configuration.

Key Features

- User registration and login – hashed password storage
- JWT – authentication with protected routes
- User can create, edit and delete multiple boards
- Add, edit, delete and reorder columns and cards
- Drag and drop support for columns and cards
- Inline editing of columns titles and boards.
- Timestemps display for cards
- Each card can be commented. Comments can be edited and deleted
- Color customization for cards and columns
- Responsive UI for mobile and desktop

Installation and Setup (local)

1. **Backend**
   cd server
   npm install

Main Packages installed:

- express – Rest Api framework
- Typescript – Static typing
- Ts-node-dev – dev server with live reload
- Prisma – ORM for PosgreSQL
- @prisma/client – Prisma client runtime
- Jsonwebtoken – Creating and verifying tokens
- Dotenv – For loading env variables from .env

Prisma installation and setup:

npx prisma init

Update .env with Railway database URL:

```
DATABASE_URL=your_railway_postgres_url
JWT_SECRET=your_secret_key
```

Check database in Railway or command “npx prisma studio” (opens in the browser).

Run migrations:

npx prisma migrate dev --name init

Start development:

npm run dev (configure scripts in package.json).

2. **Frontend**

   cd client

   npm install

   npm run dev

Main packages installed:

- React, react-dom and vite – Frontend framework and tools
- TypeScript – For static typing
- @hello-pangea/dnd – Drag and drop itens
- Tailwindcss, postcss and autoprefixer – For styling and responsive webpage

User Manual

1. **Register:** Create account using name, email and password
2. **Login:** Authenticate to have access to personal dashboard
3. **Dashboard:** User can create different boards for different purposes
4. **Boards:** Creating, editing and delete boards
5. **Cards:** Drag and drop between columns. Edit titles and also delete cards
6. **Comments:** Add, edit and delete comments on any card
7. **Colors:** Customize card colors for easy control visually

All data will be saved securely in PostgreSQL database using Prisma

Deployed database in Railway platform.

Challenges faced during development

One of the challenges I faced during this project was configuring PostgreSQL database with Railway, understanding the use of environments variables being used and the actual connection string to connect locally. I also faced multiple issues with Cypress trying to install it and creating my own tests for the project. This way I could test each feature developed one by one. I also deployed the frontend in Vercel but I was not able to use CORS, which enables web applications to request information

from different domains than the one that served the webpage. Due to this, I kept deployed only the database and backend in Railway host.

Declaration of AI usage

I declared that I did not use AI to create this document.