MERN-7: Proejct Part 3 - Building Protected Route

Agenda

Bearer Token Validation with AuthMiddleware
axios Instance to get current user
Protected Component (Route)
Setting up Redux for State management

title: Bearer Token Validation with AuthMiddleware

In this code, the authorization field within the headers is being set up for an Axios instance. Here's a detailed explanation of what's happening:

```
import axios from "axios";

export const axiosInstance = axios.create({
 headers: {
   "Content-Type": "application/json",
   'authorization' : `Bearer ${localStorage.getItem('token')}`

 },
});
```

This code creates a new instance of Axios with custom default settings, specifically setting the headers that will be included with every request made by this instance.

Detailed Explanation of the authorization Field

```
headers: {
    'Content-Type': 'application/json',
    'authorization': `Bearer ${localStorage.getItem('token')}`
}
```

The headers object contains key-value pairs that specify HTTP headers to be included in every request made by this Axios instance.

Bearer Token:

The Bearer part in the Authorization header is used to specify the authentication scheme that is being used. In this case, it indicates that the client is using a Bearer Token for authentication.

A Bearer Token is a type of token that is used for authentication. The term "bearer" means that whoever "bears" (holds) the token can use it to access protected resources. The token itself is proof of the bearer's right to access the resource

Example Scenario

Suppose a user logs in to an application and the server responds with a JWT (JSON Web Token). This token is stored in the browser's local storage:

localStorage.setItem('token',
'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...');

When making API requests using the axiosInstance, the authorization header will include this token:

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

Importance of the authorization Header

Authentication:
The server uses the token to verify the identity of the client making the request. This is essential for accessing protected resources or endpoints that require authentication.

Security:
By including the token in the header, sensitive information is kept secure during transmission. The server can validate the token and ensure that the request is from an authenticated user.

In summary, the authorization field in the headers object of the Axios instance is crucial for making authenticated requests to a server, ensuring that only authorized users can access protected resources.

Building the Auth Middleware Get Current user Route

First of all create a middlewares names directory and create the authMiddleware

```
const jwt = require("jsonwebtoken");

const auth = (req, res, next) => {
 try {
   console.log("req.headers", req.headers.authorization);
   const token = req.headers.authorization.split(" ")[1];
   console.log("token", token);
   const verifiedtoken = jwt.verify(token, process.env.JWT_SECRET);
   console.log("verifiedtoken", verifiedtoken);
   req.body.userId = verifiedtoken.userId;
   next();
 } catch (error) {
   res.status(401).send({ success: false, message: "Token Invalid" });
 }
};

module.exports = auth;
```

Explanation

1. We are extracting the token from the header
2. Verifying the token is valid or not
3. The token is verified using a secret key (process.env.secret_key_jwt). If the token is valid, it returns the decoded token.
4. Next a very important part is that we are pulling out the data (user id) from the token and making it available to the request object

5. Now the same request object is shared by any following middleware or controller
6. The userId from the verified token is attached to the req.body, making it accessible in subsequent middleware or route handlers.

Code for getCurrent user Route

1. Create a route in userRoute

```
userRouter.get('/get-current-user', getCurrentUser)
```

2. Define the method in the controller

```
const getCurrentUser = async (req, res) => {
 const user = await User.findById(req.body.userId).select("-password");

 res.send({
   success: true,
   message: "You are authorized to go to the protected route!",
   data: user,
 });
};

module.exports = {
 register,
 login,
 getCurrentUser,
};
```

Explanation

1. Here the userId in the req.body is available because of our authMiddleware

2. Now we find that user by the id and return attributes without the password

Code in userRouter

```
const express = require("express");
const User = require("../models/userModel");
const { register, login, getCurrentUser } = require("../controllers/userController");

const userRouter = express.Router();
// Register a user
userRouter.post("/register", register);

userRouter.post("/login", login);

userRouter.get('/get-current-user', getCurrentUser)

module.exports = userRouter;
```

Chaining

Now the point is how we ensure that the authMiddleware is called when someone tries to access this get-current-user route ?

Now comes the idea of chaining where we will chain our middlewares with handlers

```
const express = require("express");
const User = require("../models/userModel");
const { register, login, getCurrentUser } = require("../controllers/userController");
const auth = require("../middlewares/authMiddleware");

const userRouter = express.Router();
// Register a user
userRouter.post("/register", register);
```

```
userRouter.post("/login", login);

userRouter.get('/get-current-user',auth, getCurrentUser)

module.exports = userRouter;
```

Relation to Axios Instance

The Axios instance you created includes an authorization header with a Bearer token:

Token Inclusion:
When a request is made using this Axios instance, it automatically includes the authorization header with the JWT token retrieved from localStorage.

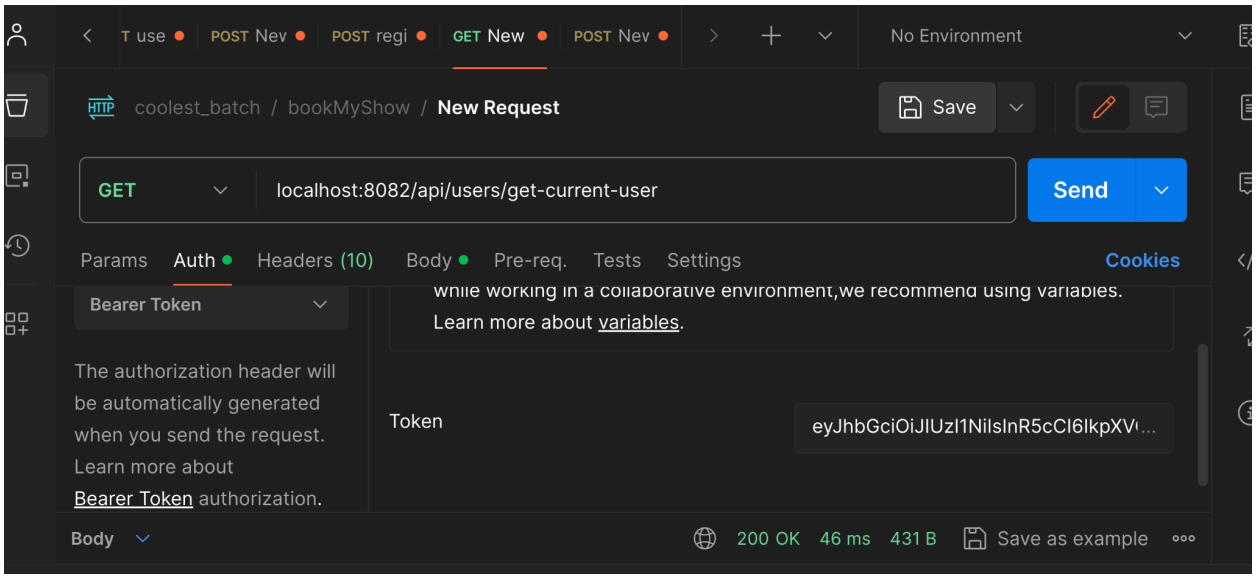
Middleware Verification:
The authMiddleware extracts this token from the authorization header of the incoming request and verifies it using the secret key. If the token is valid, it allows the request to proceed to the protected route.

Route Handling:
Once the middleware verifies the token and attaches the userId to the request body, the /get-current-user route handler uses this userId to fetch and return the authenticated user's data.

In summary, the Axios instance ensures that each request includes the necessary token for authentication. The middleware then verifies this

token to authorize access to protected routes, and the route handler fetches and returns user-specific data based on the verified token.

IN POSTMAN, in the Auth tab, add the token and check



title: Axios Instance to get Current user

Inside your api folder under client in the file users.js add one more function like this

```
export const GetCurrentUser = async () => {
 try {
   const response = await axiosInstance.get("api/users/get-current-user");
   return response.data;
 } catch (error) {
   console.log(error);
 }
};
```

This function will make an authenticated request to the /get-current-user endpoint using the axiosInstance we discussed earlier

title: Protected Route

Create a Home page , A bare minimum compoenet for now inside the pages folder

```
const Home = () => {
   return (
     <div>
       This is my home page
     </div>
   )
 }
 export default Home;
```

Now Go to Client folder and create a directory by the name Components and add your first component as ProtectedRoute.js

Let's now build this protected component which will check if the token exists and will validate it and can get the current user

```
import React from 'react'

const ProtectedRoute = ({childern}) => {
   return (
       <div>{children}</div>
   )
}
export default ProtectedRoute
```

Here,
childern (home page): We want to make our Home PAge Protected

In our app.js add this with login and register routes

```
        <Route path="/" element={<ProtectedRoute><Home /></ProtectedRoute>} />
```

Redux

As we move further our application state will grow which is why we'll be using redux and then we will continue building the Protected Route

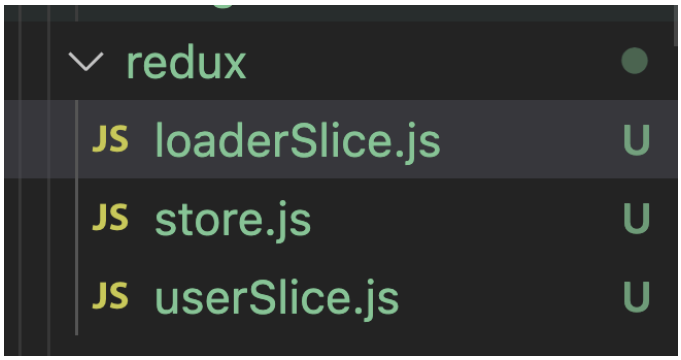
npm install @reduxjs/toolkit
npm install react-redux

We're using state management here to basically use a loader whenever a user logs in

Three files can be created as following under redux folder

/redux
- loaderSlice.js
- store.js

- userSlice.js



\

Creating a slice in Redux Toolkit serves the purpose of organizing your Redux state and its related actions and reducers in a concise and manageable way.

A slice is essentially a collection of Redux reducer logic and actions for a single feature of your application. By using createSlice, you streamline the process of defining action creators and the corresponding reducer logic, reducing boilerplate code and making your Redux code more maintainable.

```
import {createSlice} from '@reduxjs/toolkit';

const loaderSlice = createSlice({
   name: "loaders",// Name of the slice, used for action type prefixes
   initialState: {
       loading: false,// Initial state
   },
   reducers: {
       // Reducer to show loading spinner
       ShowLoading : (state) => {
           state.loading = true;
       },
       HideLoading : (state) => {
```

```
            state.loading = false;
        }
    }
});
export const {ShowLoading, HideLoading} = loaderSlice.actions;
export default loaderSlice.reducer;
```

In Redux Toolkit, the createSlice function simplifies the process of creating reducers and action creators.

When you define reducers within a slice, Redux Toolkit automatically generates corresponding action creators for each reducer function.

These action creators are then made available as part of the slice's actions object. This automation helps reduce boilerplate code and ensures consistency in your Redux setup.

Example action creator (don't add in the code)

const showLoadingAction = ShowLoading();
console.log(showLoadingAction);
// Output: { type: 'loaders/ShowLoading' }

userSlice.js

```
import { createSlice } from "@reduxjs/toolkit";

const usersSlice = createSlice({
```

```
    name: "users",
    initialState: {
        user : null,
    },
    reducers: {
        SetUser : (state, action) => {
            state.user = action.payload;
        }
    }
});

export const { SetUser } = usersSlice.actions;

export default usersSlice.reducer;
```

Update store.js

```
import { configureStore } from "@reduxjs/toolkit";
import loaderReducer from "./loaderSlice";
import usersReducer from "./userSlice";

const store = configureStore({
 reducer: {
   loaders: loaderReducer,
   users: usersReducer,
 },
});

export default store;
```

Update App.js with Provider

```
import Home from "./pages/Home";
import Login from "./pages/Login";
import Register from "./pages/Register";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import "./App.css";
import ProtectedRoute from "./components/ProtectedRoute";
import { Provider } from "react-redux";
```

```
import store from "./redux/store";

function App() {
 return (
   <div>
     <Provider store={store}>
       <BrowserRouter>
         <Routes>
           <Route path="/login" element={<Login />} />
           <Route path="/register" element={<Register />} />
           <Route path="/" element={<ProtectedRoute><Home /></ProtectedRoute>} />
         </Routes>
       </BrowserRouter>
   </Provider>
   </div>
 );
}

export default App;
```

Our second step would be to show details if the existing user tries to login again, so the question here is how to get the details of a particular user

We will use some icons from antd

npm i @ant-design/icons

Building ProtectRoute

1. Selecting users from the store

```
import React from 'react'
import { useDispatch, useSelector } from "react-redux";
```

```
const ProtectedRoute = ({children}) => {
     const {user} = useSelector(state => state.users)
    return (
        <div>{children}</div>
    )
}
export default ProtectedRoute
```

1. state.users is because of the slice defined in store.js which is users
2. userSlice has state variable by the name user

2. Dispatch and navigate

```
import React from 'react'
import { useDispatch, useSelector } from "react-redux";
import { useNavigate } from "react-router-dom";

const ProtectedRoute = ({children}) => {
    const {user} = useSelector(state => state.users)
    const dispatch = useDispatch();
     const navigate = useNavigate();

    return (
        <div>{children}</div>
    )
}
export default ProtectedRoute
```

3. Creating nav items

```
import React from 'react'
import { useDispatch, useSelector } from "react-redux";
```

```
import { useNavigate } from "react-router-dom";
import {
    HomeOutlined,
    LogoutOutlined,
    ProfileOutlined,
    UserOutlined,
  } from "@ant-design/icons";


const ProtectedRoute = ({children}) => {
    const {user} = useSelector(state => state.users)
    const dispatch = useDispatch();
    const navigate = useNavigate();
    const navItems = [
        {
          label: "Home",
          icon: <HomeOutlined />,
        },

        {
          label: `${user ? user.name : ""}`,
          icon: <UserOutlined />,
          children: [],
        },
      ];

    return (
        <div>{children}</div>
    )
}
export default ProtectedRoute
```

4. Add children option of My Profile and logout for the profile option in the menu

```
const navItems = [
        {
          label: "Home",
```

```
            icon: <HomeOutlined />,
          },

          {
            label: `${user ? user.name : ""}`,
            icon: <UserOutlined />,
            children: [
              {
                    label: (
                      <span>
                        My Profile
                      </span>
                    ),
                    icon: <ProfileOutlined />,
                  },

                  {
                    label: (
                      <span>
                        Log Out
                      </span>
                    ),
                    icon: <LogoutOutlined />,
                  },
            ],
          },
        ];
```

5. Update MyProfile and Logout with handlers and conditions

```
import React from 'react'
import { useDispatch, useSelector } from "react-redux";
import { useNavigate, Link } from "react-router-dom";
import {
    HomeOutlined,
    LogoutOutlined,
    ProfileOutlined,
    UserOutlined,
```

```
  } from "@ant-design/icons";

const ProtectedRoute = ({children}) => {
    const {user} = useSelector(state => state.users)
    const dispatch = useDispatch();
    const navigate = useNavigate();
    const navItems = [
        {
          label: "Home",
          icon: <HomeOutlined />,
        },

        {
          label: `${user ? user.name : ""}`,
          icon: <UserOutlined />,
          children: [
            {
                label: (
                    <span
                      onClick={() => {
                        if (user.role === 'admin') {
                          navigate("/admin");
                        } else if (user.role === 'partner') {
                          navigate("/partner");
                        } else {
                          navigate("/profile");
                        }
                      }}
                      >
                        My Profile
                      </span>
                ),
                icon: <ProfileOutlined />,
              },

              {
                label: (
                    <Link
                      to="/login"
                      onClick={() => {
```

```
                        localStorage.removeItem("token");
                      }}
                    >
                      Log Out
                    </Link>
                  ),
                  icon: <LogoutOutlined />,
                },
              ],
            },
          ];

      return (
          <div>{children}</div>
      )
  }
  export default ProtectedRoute
```

6. Add an useEffect to get the valid user

```
useEffect(() => {
      if (localStorage.getItem("token")) {
          getValidUser();
      } else {
          navigate("/login");
      }
      }, []);
```

7. Define the validUser function

```
  import React,{useEffect} from 'react'
  import { useDispatch, useSelector } from "react-redux";
  import { useNavigate, Link } from "react-router-dom";
  import {
      HomeOutlined,
      LogoutOutlined,
```

```
    ProfileOutlined,
    UserOutlined,
  } from "@ant-design/icons";
import { GetCurrentUser } from "../api/user";
import { SetUser } from "../redux/userSlice";
import { message } from "antd";
import { ShowLoading, HideLoading } from "../redux/loaderSlice";


const ProtectedRoute = ({children}) => {
    const {user} = useSelector(state => state.users)
    const dispatch = useDispatch();
    const navigate = useNavigate();
    const navItems = [
        {
          label: "Home",
          icon: <HomeOutlined />,
        },

        {
          label: `${user ? user.name : ""}`,
          icon: <UserOutlined />,
          children: [
            {
                label: (
                    <span
                    onClick={() => {
                      if (user.role === 'admin') {
                        navigate("/admin");
                      } else if (user.role === 'partner') {
                        navigate("/partner");
                      } else {
                        navigate("/profile");
                      }
                    }}
                    >
                      My Profile
                    </span>
                ),
                icon: <ProfileOutlined />,
            },
```

```
            {
              label: (
                  <Link
                  to="/login"
                  onClick={() => {
                    localStorage.removeItem("token");
                  }}
                >
                  Log Out
                </Link>
              ),
              icon: <LogoutOutlined />,
            },
          ],
        },
      ];
      const getValidUser = async () => {
        try {
          dispatch(ShowLoading());
          const response = await GetCurrentUser();
          console.log(response)
          dispatch(SetUser(response.data));
          dispatch(HideLoading());
          // Hide Loader
        } catch (error) {
          dispatch(SetUser(null));
          message.error(error.message);
        }
      };
    useEffect(() => {
    if (localStorage.getItem("token")) {
        getValidUser();
    } else {
        navigate("/login");
    }
    }, []);


    return (
        <div>{children}</div>
    )
```

```
  }
  export default ProtectedRoute
```

8. Now the markup to be returned

```
import React, { useEffect } from "react";
import { useDispatch, useSelector } from "react-redux";
import { useNavigate, Link } from "react-router-dom";
import {
 HomeOutlined,
 LogoutOutlined,
 ProfileOutlined,
 UserOutlined,
} from "@ant-design/icons";
import { GetCurrentUser } from "../api/user";
import { SetUser } from "../redux/userSlice";
import { message, Layout, Menu } from "antd";
import { ShowLoading, HideLoading } from "../redux/loaderSlice";

const ProtectedRoute = ({ children }) => {
 const { user } = useSelector((state) => state.users);
 const dispatch = useDispatch();
 const navigate = useNavigate();
 const { Header, Content, Footer, Sider } = Layout;
 const navItems = [
   {
     label: "Home",
     icon: <HomeOutlined />,
   },

   {
     label: `${user ? user.name : ""}`,
     icon: <UserOutlined />,
     children: [
       {
         label: (
           <span
             onClick={() => {
               if (user.role === "admin") {
                 navigate("/admin");
```

```
              } else if (user.role === "partner") {
                navigate("/partner");
              } else {
                navigate("/profile");
              }
            }}
          >
            My Profile
          </span>
        ),
        icon: <ProfileOutlined />,
      },

      {
        label: (
          <Link
            to="/login"
            onClick={() => {
              localStorage.removeItem("token");
            }}
          >
            Log Out
          </Link>
        ),
        icon: <LogoutOutlined />,
      },
    ],
  },
];
const getValidUser = async () => {
  try {
    dispatch(ShowLoading());
    const response = await GetCurrentUser();
    console.log(response);
    dispatch(SetUser(response.data));
    dispatch(HideLoading());
    // Hide Loader
  } catch (error) {
    dispatch(SetUser(null));
    message.error(error.message);
  }
};
```

```
 useEffect(() => {
   if (localStorage.getItem("token")) {
     getValidUser();
   } else {
     navigate("/login");
   }
 }, []);

 return (
   user && (
     <>
       <Layout>
         <Header
           className="d-flex justify-content-between"
           style={{
             position: "sticky",
             top: 0,
             zIndex: 1,
             width: "100%",
             display: "flex",
             alignItems: "center",
           }}
         >
           <h3 className="demo-logo text-white m-0" style={{ color:
"white" }}>
             Book My Show
           </h3>
           <Menu theme="dark" mode="horizontal" items={navItems} />
         </Header>
         <div style={{ padding: 24, minHeight: 380, background: "#fff"
}}>
           {children}
         </div>
       </Layout>
     </>
   )
 );
};
export default ProtectedRoute;
```

Explanation

1. ProtectedRoute Component:
    a. This is a React functional component that ensures only authenticated users can access its children components.
2. useSelector Hook:
    a. const { user } = useSelector((state) => state.user);
    b. This hook extracts the user state from the Redux store. It ensures that the component reacts to any changes in the user state.
3. navItems:
    a. This array defines the items to be displayed in the navigation menu. Each item has a label, icon, and potentially children for nested menu items.
4. getValidUser Function:
    a. This asynchronous function retrieves the current user's data by making an API call via GetCurrentUser.
    b. dispatch(showLoading()) shows a loading indicator.
    c. If the API call is successful, it dispatches the setUser action to update the user state in the Redux store and hides the loading indicator.

d. If the API call fails, it dispatches the setUser action with null to reset the user state and displays an error message.

UseEffect in Login for logged in User

In order to prevent the user from going back to login page after logging in we can do the following

```
useEffect(() => {
  if (localStorage.getItem("token")) {
    navigate("/");
  }
}, []);
```