

**Laporan Praktikum**  
**Mata Kuliah Pemrograman Berorientasi Objek**



**AUTHENTICATION (Login Multi-Level)**

Dosen Pengampu :  
Willdan Aprizal Arifin S.pd M.kom

Disusun Oleh :  
RIYADI TRI WALUYA BUDI  
2308065

**PROGRAM STUDI SISTEM INFORMASI KELAUTAN**  
**UNIVERSITAS PENDIDIKAN INDONESIA**  
**2024**

## I. PENDAHULUAN

Authentication adalah proses penting dalam aplikasi web untuk memastikan keamanan akses ke data dan fitur aplikasi. Dalam praktik ini, akan dibuat sistem autentikasi login berbasis peran menggunakan Node.js, Express, MySQL, dan React.js. Sistem ini memungkinkan pengguna dengan berbagai peran, seperti Admin, User, atau Manager, untuk mengakses fitur atau halaman yang sesuai dengan hak akses mereka.

Tujuan Praktikum:

1. Memahami Autentikasi dan Otorisasi:
  - Memahami perbedaan antara autentikasi (proses memverifikasi identitas pengguna) dan otorisasi (proses memberikan hak akses berdasarkan peran pengguna).
  - Mengimplementasikan autentikasi berbasis peran.
2. Integrasi Teknologi Full-Stack:
  - Backend: Menggunakan Node.js dan Express untuk menangani login serta manajemen pengguna.
  - Database: Menggunakan MySQL untuk menyimpan data pengguna dan peran.
  - Frontend: Menggunakan React.js untuk antarmuka login dan tampilan berbasis peran.
3. Penerapan Keamanan:
  - Mengenkripsi kata sandi menggunakan hashing (contoh: bcrypt).
  - Menggunakan JSON Web Token (JWT) untuk mengelola sesi pengguna secara aman.

Pendekatan ini menekankan praktik pengembangan aman dan modern untuk membangun sistem autentikasi yang andal dan berbasis peran.

## II. ALAT DAN BAHAN

- Node.js: Untuk membangun backend server.
- Express.js: Framework backend untuk menangani API dan middleware.
- MySQL: Database relasional untuk menyimpan data pengguna.
- React.js: Library frontend untuk membangun antarmuka pengguna.
- JWT (JSON Web Token): Untuk mengelola otorisasi berbasis token.
- bcrypt: Untuk mengenkripsi kata sandi.

### III. LANGKAH KERJA

1.Pertama ,Buka Visual Studio Code setelah itu install package.json dan hasilnya seperti di bawah ini

#### -Backend

The screenshot shows the Visual Studio Code interface with the package.json file open in the center. The file content is as follows:

```
1  {
2    "name": "backend",
3    "version": "1.0.0",
4    "description": "",
5    "type": "module",
6    "main": "index.js",
7    "scripts": [
8      "test": "echo \\"error: no test specified\\\" && exit 1"
9    ],
10   "keywords": [],
11   "author": "",
12   "license": "ISC",
13   "dependencies": {
14     "argon2": "^0.41.1",
15     "connect-session-sequelize": "^7.1.7",
16     "dotenv": "^16.4.3",
17     "express": "^4.21.1",
18     "express-session": "^1.18.1",
19     "mysql": "^2.11.14",
20     "sequelize": "^6.37.2"
21   }
22 }
```

The Explorer sidebar on the left shows the project structure, including Login-Multi-Level, Login-Multi-Role-Backend-main, and Login-Multi-Role-Frontend-main. The package.json file is selected in the Explorer.

#### -Frontend

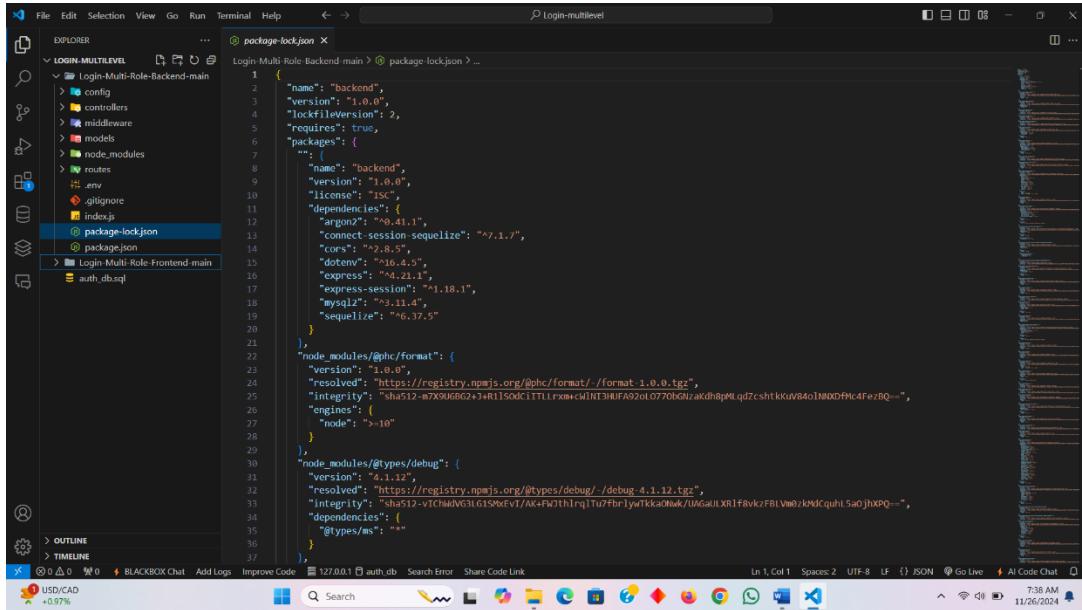
The screenshot shows the Visual Studio Code interface with the package.json file open in the center. The file content is as follows:

```
1  {
2    "name": "frontend",
3    "version": "0.1.0",
4    "private": true,
5    "dependencies": {
6      "@reduxjs/toolkit": "^2.3.0",
7      "@testing-library/jest-dom": "^5.6.3",
8      "@testing-library/react": "^16.0.1",
9      "@testing-library/user-event": "^14.5.2",
10     "axios": "1.7.2",
11     "bulma": "0.7.5",
12     "react": "18.2.0",
13     "react-dom": "18.2.0",
14     "react-icons": "5.3.0",
15     "react-redux": "8.0.1",
16     "react-router-dom": "6.0.1",
17     "react-scripts": "5.0.1",
18     "web-vitals": "4.2.4"
19   },
20   "scripts": {
21     "start": "react-scripts start",
22     "build": "react-scripts build",
23     "test": "react-scripts test",
24     "eject": "react-scripts eject"
25   },
26   "eslintConfig": {
27     "extends": [
28       "react-app",
29       "react-app/jest"
30     ],
31     "browserslist": {
32       "production": [
33         ">=24",
34         "not dead",
35         "not op_mini all"
36       ]
37     }
38   }
39 }
```

The Explorer sidebar on the left shows the project structure, including Login-Multi-Level, Login-Multi-Role-Backend-main, and Login-Multi-Role-Frontend-main. The package.json file is selected in the Explorer.

2. Kedua, Install package-lock.json untuk backend dan frontend dan hasilnya seperti di bawah ini

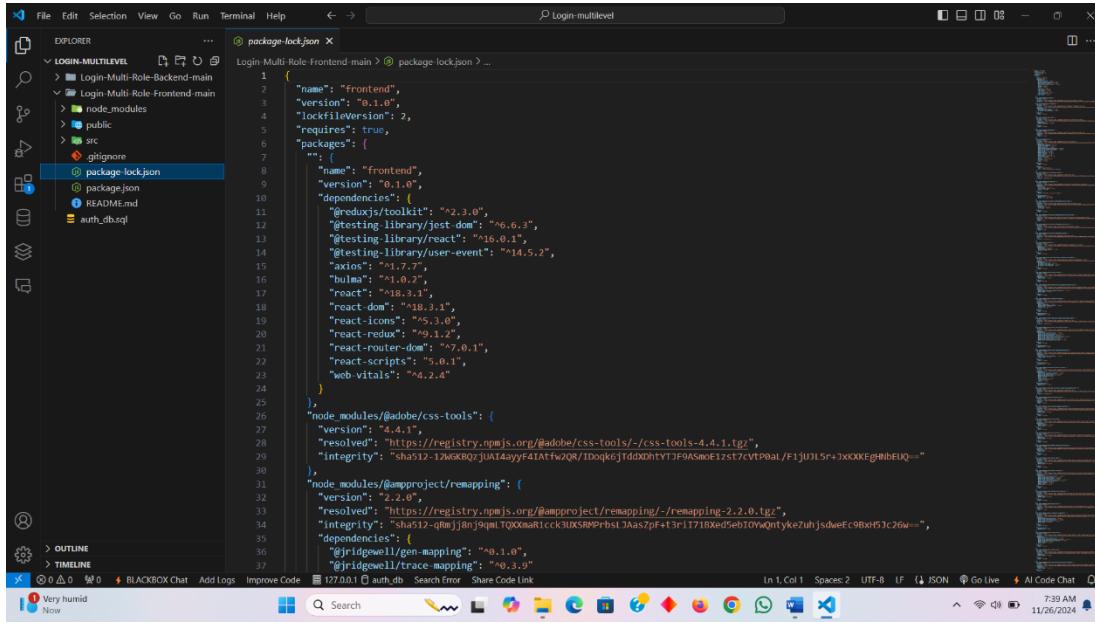
### -Backend



The screenshot shows the VS Code interface with the 'package-lock.json' file open in the center. The file content is as follows:

```
1 {
  "name": "backend",
  "version": "1.0.0",
  "lockfileVersion": 2,
  "requires": true,
  "packages": [
    {
      "name": "backend",
      "version": "1.0.0",
      "license": "ISC",
      "dependencies": {
        "argon2": "^0.41.1",
        "connect-session-sequelize": "^7.1.7",
        "cores": "^0.8.5",
        "dotenv": "^0.4.5",
        "express": "^4.21.1",
        "express-session": "^1.18.1",
        "mysql2": "^3.11.4",
        "sequelize": "^6.37.5"
      }
    },
    "node_modules/@phc/format": {
      "version": "1.0.0",
      "resolved": "https://registry.npmjs.org/@phc/format/-/format-1.0.0.tgz",
      "integrity": "sha512-m7X9066C2J+R1150dC11Lrxm+cW1NT3HfA02l077060zakdh8pMqdzshtkuV84o1N0xFMc4FezBQ==",
      "engines": {
        "node": ">10"
      }
    },
    "node_modules/@types/debug": {
      "version": "0.1.12",
      "resolved": "https://registry.npmjs.org/@types/debug/-/debug-4.1.12.tgz",
      "integrity": "sha512-vTCIhdVG3LG1SMxEv7/Ak+Fw0ThInqTu7fbrywtkka0lk/UdGaiLXR1F8vkZFLVn0zkmQquhL5a0jhXPQ==",
      "dependencies": {
        "@types/ms": "*"
      }
    }
  ]
}
```

### -Frontend



The screenshot shows the VS Code interface with the 'package-lock.json' file open in the center. The file content is as follows:

```
1 {
  "name": "frontend",
  "version": "0.1.0",
  "lockfileVersion": 2,
  "requires": true,
  "packages": [
    {
      "name": "frontend",
      "version": "0.1.0",
      "dependencies": {
        "@reduxjs/toolkit": "^2.3.0",
        "@testing-library/jest-dom": "^6.6.3",
        "@testing-library/react": "^16.0.1",
        "@testing-library/user-event": "^14.5.2",
        "axios": "^1.7.7",
        "bulma": "^1.0.2",
        "react": "^18.3.1",
        "react-dom": "^18.3.1",
        "react-icons": "^5.3.0",
        "react-redux": "^9.1.2",
        "react-router-dom": "^7.0.1",
        "react-scripts": "5.0.1",
        "web-vitals": "^4.2.4"
      }
    },
    "node_modules/@adobe/css-tools": {
      "version": "4.4.1",
      "resolved": "https://registry.npmjs.org/@adobe/css-tools/-/css-tools-4.4.1.tgz",
      "integrity": "sha512-LxGGRQjU1Aay41ATfw2Q/100qkeJdD0hYTDF94sm0E1zst/CvTPoal/F1ju3L5r+YxKKEgBbHEU=="
    },
    "node_modules/@amp-project/remapping": {
      "version": "2.2.0",
      "resolved": "https://registry.npmjs.org/@amp-project/remapping/-/remapping-2.2.0.tgz",
      "integrity": "sha512-qBwjj8njqmlTQXmaRicck3UXSRMPbSLJaas7pF+13r177IBxEd5ebTOyQntyke2uhjsdweEc9BxH53c26w==",
      "dependencies": {
        "@ridgewell/gen-mapping": "0.1.0",
        "@ridgewell/trace-mapping": "0.3.9"
      }
    }
  ]
}
```

3.Ketiga, membuat file baru “index.js” untuk backend dan frontend dan isinya seperti gambar di bawah ini

-Backend

```
File Edit Selection View Go Run Terminal Help ← → Login-Multi-level  
EXPLORER index.js  
LOGIN-MULTILEVEL Login-Multi-Role-Backend-main index.js > ...  
config controllers middleware models node_modules routes env .gitignore  
index.js package-lock.json package.json  
Login-Multi-Role-Frontend-main auth_db.sql  
1 import express from "express";  
2 import cors from "cors";  
3 import session from "express-session";  
4 import dotenv from "dotenv";  
5 import db from "./config/Database.js";  
6 import SequelizeStore from "connect-session-sequelize";  
7 import UserRoute from "./routes/UserRoutes.js";  
8 import ProductRoute from "./routes/ProductRoute.js";  
9 import AuthRoute from "./routes/AuthRoute.js";  
10 dotenv.config();  
11  
12 const app = express();  
13  
14 const sessionStore = SequelizeStore(session.Store);  
15  
16 const store = new sessionStore({  
17   db: db  
18 });  
19  
20 // (async()=>{  
21 //   await db.sync();  
22 // })();  
23  
24 app.use(session({  
25   secret: process.env.SESSION_SECRET,  
26   resave: false,  
27   saveUninitialized: true,  
28   store: store,  
29   cookie: {  
30     secure: 'auto'  
31   }  
32 }));  
33  
34 app.use(cors({  
35   credentials: true,  
36   origin: 'http://localhost:3000'  
37 }));  
Ln 15, Col 1 Spaces: 4 UTF-8 LF {} JavaScript Go Live AI Code Chat  
83°F Mostly cloudy
```

-Frontend

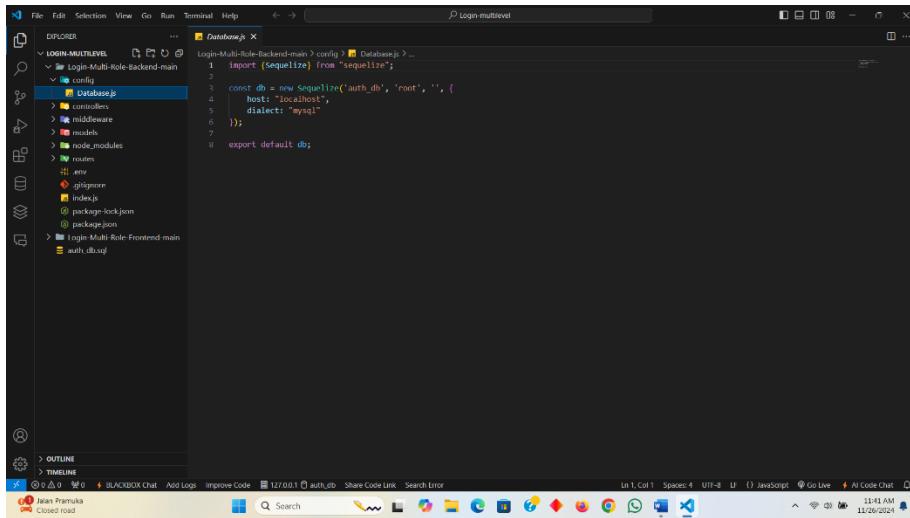


```
1 import React from 'react';
2 import { createRoot } from 'react-dom/client';
3 import { Provider } from 'react-redux';
4 import { store } from './app/store';
5 import App from './App';
6 import 'bulma/css/bulma.css';
7 import axios from "axios";
8
9 axios.defaults.withCredentials = true;
10
11 const container = document.getElementById('root');
12 const root = createRoot(container);
13
14 root.render(
15   <React.StrictMode>
16     <Provider store={store}>
17       <App />
18     </Provider>
19   </React.StrictMode>
20 );
21
22 );
```

4. Keempat, Di dalam folder “Login-Multi-Role-Backend-main” terdapat folder config, controllers, middleware, models, node\_modules, routes, env dan gitignore yang isinya seperti gambar di bawah ini.

### \*Config

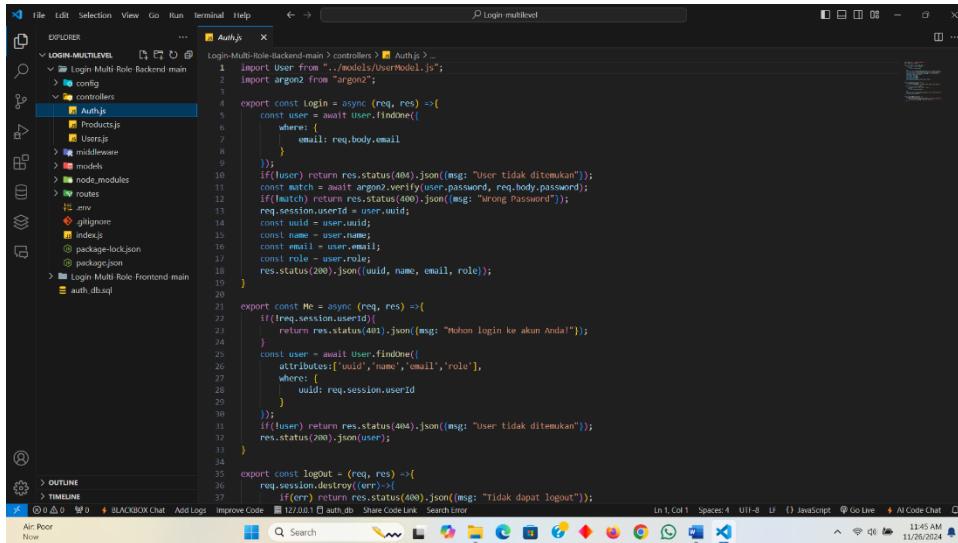
#### -Database.js



```
File Edit Selection View Go Run Terminal Help ⏴ Login multirole
EXPLORER Login-Multi-Role-Backend-main > config > Database.js
Database.js
1 import { Sequelize } from "sequelize";
2
3 const db = new Sequelize('auth_db', 'root', '', {
4   host: 'localhost',
5   dialect: 'mysql'
6 });
7
8 export default db;
```

### \*Controllers

#### -Auth.js



```
File Edit Selection View Go Run Terminal Help ⏴ Login multirole
EXPLORER Login-Multi-Role-Backend-main > controllers > Auth.js
Auth.js
1 import User from "../models/UserModel.js";
2 import argon2 from "argon2";
3
4 export const login = async (req, res) =>{
5   const user = await User.findOne({
6     where: {
7       email: req.body.email
8     }
9   );
10  if(!user) return res.status(404).json({msg: "User tidak ditemukan"});
11  const match = await argon2.verify(user.password, req.body.password);
12  if(!match) return res.status(400).json({msg: "Wrong Password"});
13  const uuid = user.uuid;
14  req.session.userId = user.uuid;
15  const name = user.name;
16  const email = user.email;
17  const role = user.role;
18  res.status(200).json({uuid, name, email, role});
19
20}
21 export const me = async (req, res) =>{
22  if(!req.session.userId){
23    return res.status(401).json({msg: "Mohon login ke akun Anda!"});
24  }
25  const user = await User.findOne({
26    attributes:['uuid','name','email','role'],
27    where: {
28      | uuid: req.session.userId
29    }
30  );
31  if(!user) return res.status(404).json({msg: "User tidak ditemukan"});
32  res.status(200).json(user);
33}
34
35 export const logout = (req, res) =>{
36  req.session.destroy((err) =>{
37    if(err) return res.status(400).json({msg: "Tidak dapat logout"});
38  })
39}
```

## -Products.js

The screenshot shows the VS Code interface with the 'Products.js' file open in the center editor pane. The file is located in the 'controllers' directory of the 'Login-Multi-Level' project. The code implements two main functions: `getProducts` and `getProductById`. The `getProducts` function uses Sequelize's `findAll` method to retrieve products. It includes a conditional check for the user role ('admin'). If the role is 'admin', it retrieves all products with attributes: 'uuid', 'name', and 'price'. If the role is not 'admin', it retrieves products for the current user with attributes: 'name' and 'email'. The `getProductById` function uses Sequelize's `findOne` method to find a product by its ID. Both functions handle errors and return JSON responses.

```
1 import Product from "../models/ProductModel.js";
2 import User from "../models/UserModel.js";
3 import {Op} from "sequelize";
4
5 export const getProducts = async (req, res) => {
6   try {
7     let response;
8     if(req.role === "admin"){
9       response = await Product.findAll({
10       attributes:[ 'uuid', 'name', 'price'],
11       include:[
12         {model: User,
13          attributes:[ 'name', 'email']
14        }
15      ]);
16    }else{
17      response = await Product.findAll({
18        attributes:[ 'uuid', 'name', 'price'],
19        where:[
20          {userId: req.userId
21        },
22        include:[
23          {model: User,
24            attributes:[ 'name', 'email'
25          ]
26        }
27      }
28    }
29    res.status(200).json(response);
30  } catch (error) {
31    res.status(500).json({msg: error.message});
32  }
33}
34
35 export const getProductById = async (req, res) =>{
36   try {
37     const product = await Product.findOne({
38       where:
39     })
40     res.status(200).json(product);
41   } catch (error) {
42     res.status(500).json({msg: error.message});
43   }
44}
```

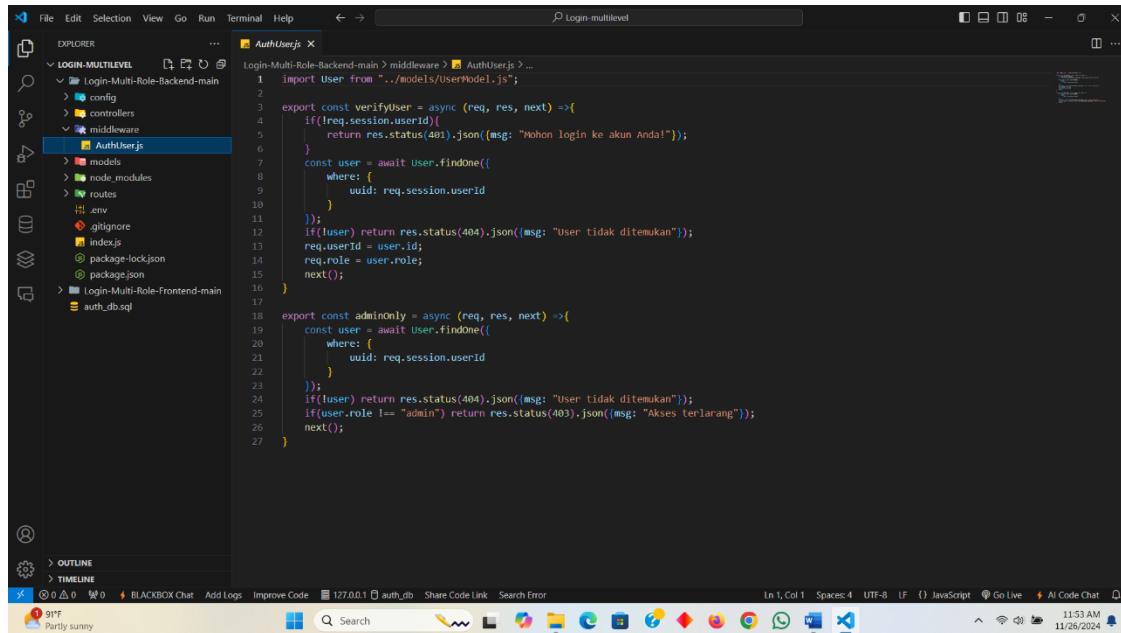
## -Users.js

The screenshot shows the VS Code interface with the 'Users.js' file open in the center editor pane. The file is located in the 'controllers' directory of the 'Login-Multi-Level' project. The code implements three main functions: `getUsers`, `getUserById`, and `createUser`. The `getUsers` function uses Sequelize's `findAll` method to retrieve users with attributes: 'uuid', 'name', 'email', and 'role'. The `getUserById` function uses Sequelize's `findOne` method to find a user by their ID with attributes: 'uuid', 'name', 'email', and 'role'. The `createUser` function handles user creation logic, including password hashing using Argon2. It checks if the provided password matches the confirmation password. If they don't match, it returns a 400 status with an error message. If the password is valid, it creates a new user in the database.

```
1 import User from "../models/UserModel.js";
2 import argon2 from "argon2";
3
4 export const getUsers = async (req, res) =>{
5   try {
6     const response = await User.findAll({
7       attributes:[ 'uuid', 'name', 'email', 'role'
7      });
8     res.status(200).json(response);
9   } catch (error) {
10    res.status(500).json({msg: error.message});
11  }
12}
13
14 export const getUserById = async (req, res) =>{
15   try {
16     const response = await User.findOne({
17       attributes:[ 'uuid', 'name', 'email', 'role'],
18       where: {
19         id: req.params.id
20       }
21     });
22     res.status(200).json(response);
23   } catch (error) {
24    res.status(500).json({msg: error.message});
25  }
26}
27
28 export const createUser = async (req, res) =>{
29   const {name, email, password, confirmPassword, role} = req.body;
30   if(password !== confirmPassword) return res.status(400).json({msg: "Password dan Confirm Password tidak cocok"});
31   const hashPassword = await argon2.hash(password);
32   try {
33     await User.create({
34       name: name,
35       email: email,
36       password: hashPassword,
37     });
38     res.status(201).json("User created successfully");
39   } catch (error) {
40     res.status(500).json({msg: error.message});
41   }
42}
```

## \*Middleware

### -AuthUser.js

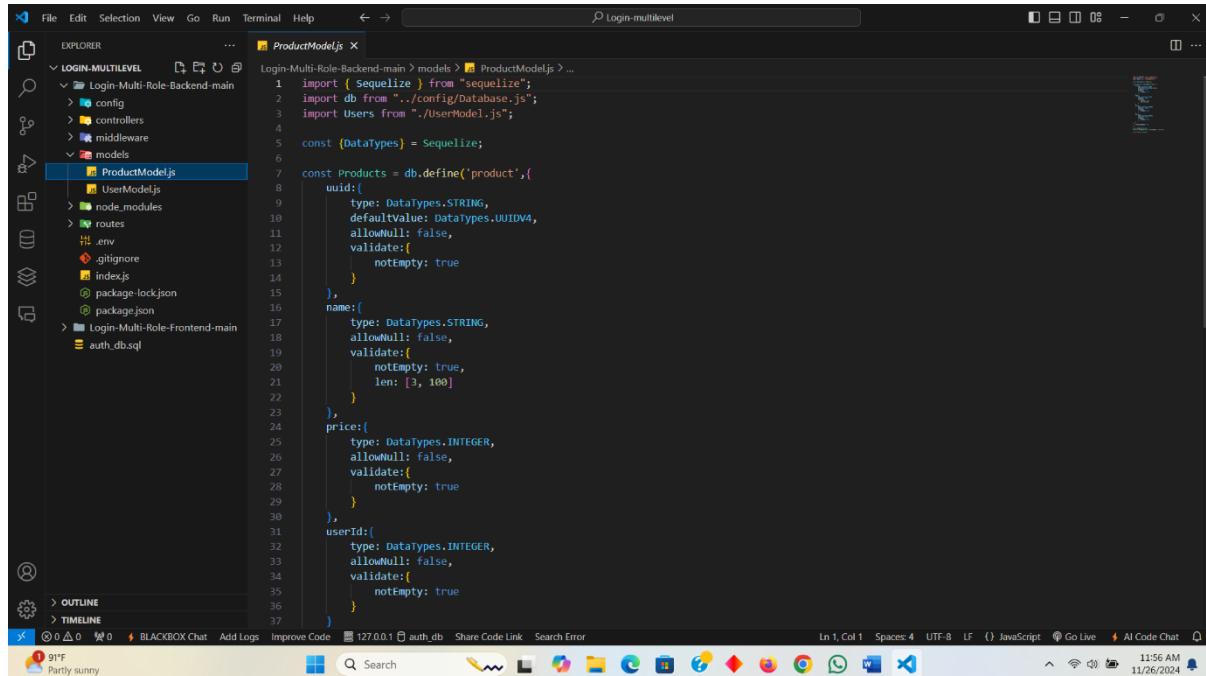


```
File Edit Selection View Go Run Terminal Help < - > Login-multilevel
EXPLORER LOGIN-MULTILEVEL
  Login-Multi-Role-Backend-main
    config
    controllers
    middleware
      AuthUser.js
    models
    node_modules
    routes
    .env
    .gitignore
    index.js
    package-lock.json
    package.json
  Login-Multi-Role-Frontend-main
    auth_db.sql

AuthUser.js
1 import User from '../models/UserModel.js';
2
3 export const verifyUser = async (req, res, next) => {
4   if(!req.session.userId){
5     return res.status(401).json({msg: "Mohon login ke akun Anda!"});
6   }
7   const user = await User.findOne({
8     where: {
9       uuid: req.session.userId
10    }
11 });
12   if(!user) return res.status(404).json({msg: "User tidak ditemukan"});
13   req.userId = user.id;
14   req.role = user.role;
15   next();
16 }
17
18 export const adminOnly = async (req, res, next) =>{
19   const user = await User.findOne({
20     where: {
21       uuid: req.session.userId
22    }
23 });
24   if(!user) return res.status(404).json({msg: "User tidak ditemukan"});
25   if(user.role != "admin") return res.status(403).json({msg: "Akses terlarang"});
26   next();
27 }
```

## \*Models

### -ProductModel.js



```
File Edit Selection View Go Run Terminal Help < - > Login-multilevel
EXPLORER LOGIN-MULTILEVEL
  Login-Multi-Role-Backend-main
    config
    controllers
    middleware
    models
      ProductModel.js
    node_modules
    routes
    .env
    .gitignore
    index.js
    package-lock.json
    package.json
  Login-Multi-Role-Frontend-main
    auth_db.sql

ProductModel.js
1 import { Sequelize } from 'sequelize';
2 import db from './config/database.js';
3 import Users from './UserModel.js';
4
5 const DataTypes = Sequelize;
6
7 const Products = db.define('product',{
8   uuid:{
9     type: DataTypes.STRING,
10    defaultValue: DataTypes.UUIDV4,
11    allowNull: false,
12    validate:{ 
13      notEmpty: true
14    }
15  },
16  name:{
17    type: DataTypes.STRING,
18    allowNull: false,
19    validate:{ 
20      notEmpty: true,
21      len: [3, 100]
22    }
23  },
24  price:{
25    type: DataTypes.INTEGER,
26    allowNull: false,
27    validate:{ 
28      notEmpty: true
29    }
30  },
31  userId:{
32    type: DataTypes.INTEGER,
33    allowNull: false,
34    validate:{ 
35      notEmpty: true
36    }
37  }
38 });


```

## -UserModel.js

The screenshot shows the VS Code interface with the file `UserModel.js` open in the editor. The code defines a Sequelize model for users:

```
1 import { Sequelize } from "sequelize";
2 import db from "../config/database.js";
3
4 const {DataTypes} = Sequelize;
5
6 const Users = db.define('users',{
7   uuid:{
8     type: DataTypes.STRING,
9     defaultValue: DataTypes.UUIDV4,
10    allowNull: false,
11    validate:{
12      notEmpty: true
13    }
14  },
15  name:{
16    type: DataTypes.STRING,
17    allowNull: false,
18    validate:{
19      notEmpty: true,
20      len: [3, 100]
21    }
22  },
23  email:{
24    type: DataTypes.STRING,
25    allowNull: false,
26    validate:{
27      notEmpty: true,
28      isEmail: true
29    }
30  },
31  password:{
32    type: DataTypes.STRING,
33    allowNull: false,
34    validate:{
35      notEmpty: true
36    }
37 },
```

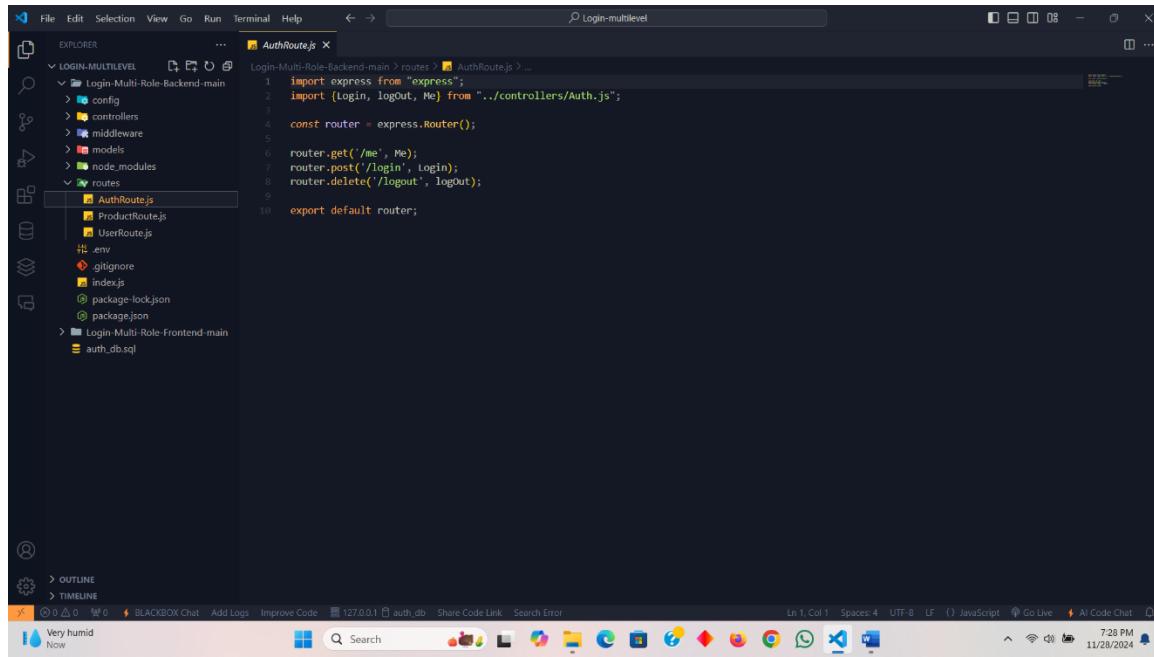
The Explorer sidebar shows the project structure, including `ProductModel.js` and `auth_db.sql`. The status bar at the bottom shows the code is 1159 AM on 11/26/2024.

## \*Node\_modules

The screenshot shows the VS Code interface with the `node_modules` folder selected in the Explorer sidebar. The folder contains numerous packages such as `argon2`, `array-flatten`, `aws-ssl-profiles`, `body-parser`, `bytes`, `call-bind`, `connect-session-sequelize`, `content-disposition`, `content-type`, `cookie`, `cookie-signature`, `cors`, `debug`, `define-data-property`, `deque`, `depd`, `destroy`, `dotenv`, `dottie`, and `ee-first`. The status bar at the bottom shows the code is 12:01 PM on 11/26/2024.

## \*Routes

### -AuthRoute.js



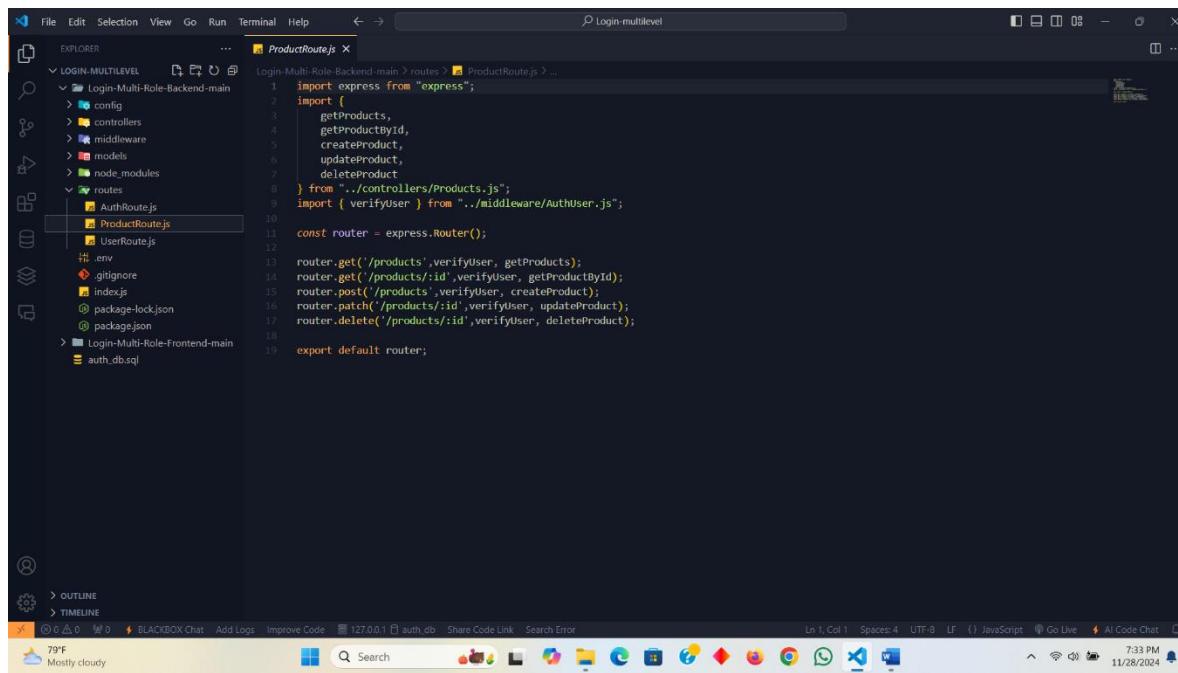
```
File Edit Selection View Go Run Terminal Help < -> Login-multilevel

EXPLORER ... AuthRoute.js
LOGIN-MULTILEVEL Login-Multi-Role-Backend-main > routes > AuthRoute.js > ...
config controllers middleware models node_modules routes
AuthRoute.js ProductRoute.js UserRoute.js
env gitignore index.js package-lock.json package.json
Login-Multi-Role-Frontend-main auth_db.sql

1 import express from "express";
2 import {login, logout, Me} from "../controllers/Auth.js";
3
4 const router = express.Router();
5
6 router.get('/me', Me);
7 router.post('/login', login);
8 router.delete('/logout', logout);
9
10 export default router;
```

The screenshot shows the VS Code interface with the file 'AuthRoute.js' open in the editor. The code defines an Express router for authentication endpoints, including '/me' (Me), '/login' (login), and '/logout' (logout). The file is part of a larger project structure under 'LOGIN-MULTILEVEL'.

### -ProductRoute.js



```
File Edit Selection View Go Run Terminal Help < -> Login-multilevel

EXPLORER ... ProductRoute.js
LOGIN-MULTILEVEL Login-Multi-Role-Backend-main > routes > ProductRoute.js > ...
config controllers middleware models node_modules routes
AuthRoute.js ProductRoute.js UserRoute.js
env gitignore index.js package-lock.json package.json
Login-Multi-Role-Frontend-main auth_db.sql

1 import express from "express";
2 import {
3   getProducts,
4   getProductById,
5   createProduct,
6   updateProduct,
7   deleteProduct
8 } from "../controllers/Products.js";
9 import { verifyUser } from "../middleware/AuthUser.js";
10
11 const router = express.Router();
12
13 router.get('/products', verifyUser, getProducts);
14 router.get('/products/:id', verifyUser, getProductById);
15 router.post('/products', verifyUser, createProduct);
16 router.patch('/products/:id', verifyUser, updateProduct);
17 router.delete('/products/:id', verifyUser, deleteProduct);
18
19 export default router;
```

The screenshot shows the VS Code interface with the file 'ProductRoute.js' open in the editor. The code defines an Express router for product management endpoints, including '/products' (get products), '/products/:id' (get product by ID), and various methods for creating, updating, and deleting products. It uses middleware to verify user authentication before performing certain operations. The file is part of a larger project structure under 'LOGIN-MULTILEVEL'.

## -UserRoute.js

```
File Edit Selection View Go Run Terminal Help < > Login-Multi-level
EXPLORER UserRoute.js
1 import express from "express";
2 import {
3   getUsers,
4   getUserById,
5   createUser,
6   updateUser,
7   deleteUser
8 } from "../controllers/Users.js";
9 import { verifyUser, adminOnly } from "../middleware/AuthUser.js";
10
11 const router = express.Router();
12
13 router.get('/users', verifyUser, adminOnly, getUsers);
14 router.get('/users/:id', verifyUser, adminOnly, getUserById);
15 router.post('/users', verifyUser, adminOnly, createUser);
16 router.patch('/users/:id', verifyUser, adminOnly, updateUser);
17 router.delete('/users/:id', verifyUser, adminOnly, deleteUser);
18
19 export default router;
```

The screenshot shows the VS Code interface with the UserRoute.js file open in the editor. The code handles various HTTP methods for users, including getting all users, getting a user by ID, creating a user, updating a user, and deleting a user. It uses express.Router() and imports functions from controllers/Users.js and middleware/AuthUser.js. The file is part of a larger project structure with multiple main and front-end components.

## \*Public

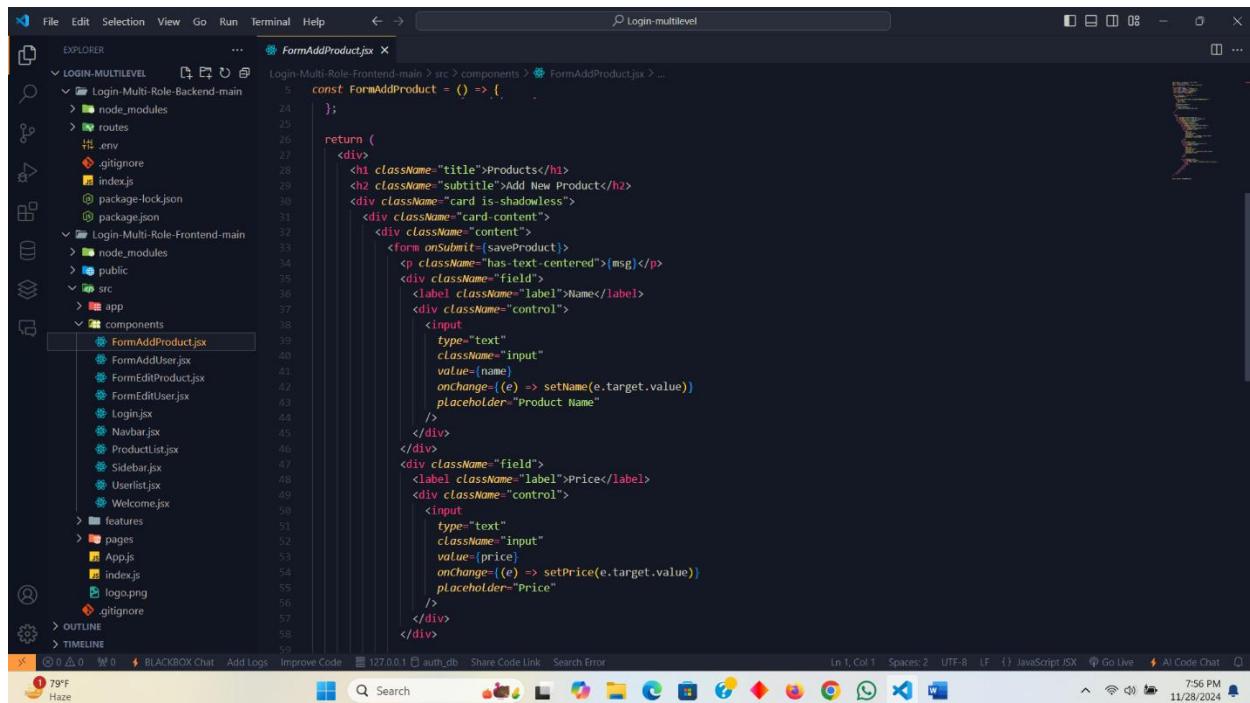
### -index.html

```
File Edit Selection View Go Run Terminal Help < > Login-mutlilevel
EXPLORER index.html
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8" />
5     <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
6     <meta name="viewport" content="width=device-width, initial-scale=1" />
7     <meta name="theme-color" content="#000000" />
8     <meta
9       name="description"
10      content="Web site created using create-react-app" />
11   </head>
12   <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
13   <!--
14     manifest.json provides metadata used when your web app is installed on a
15     user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
16   -->
17   <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
18   <!--
19     Notice the use of %PUBLIC_URL% in the tags above.
20     It will be replaced with the URL of the "public" folder during the build.
21     Only files inside the "public" folder can be referenced from the HTML.
22
23     Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
24     work correctly both with client-side routing and a non-root public URL.
25     Learn how to configure a non-root public URL by running `npm run build`.
26   -->
27   <title>React Redux App</title>
28   </head>
29   <body>
30     <noscript>You need to enable Javascript to run this app.</noscript>
31     <div id="root"></div>
32   </body>
33   <!--
34     This HTML file is a template.
35     If you open it directly in the browser, you will see an empty page.
36
37     You can add webfonts, meta tags, or analytics to this file.
38     The build step will place the bundled scripts into the <body> tag.
-->
```

The screenshot shows the VS Code interface with the index.html file open in the editor. The file is a standard HTML template for a React application, including meta tags for SEO, a link to the manifest.json file for PWA support, and a noscript block. It also includes a root div for the React component tree. The file is part of a larger project structure with multiple main and front-end components.

## \*Components

### -FormAddProduct.jsx

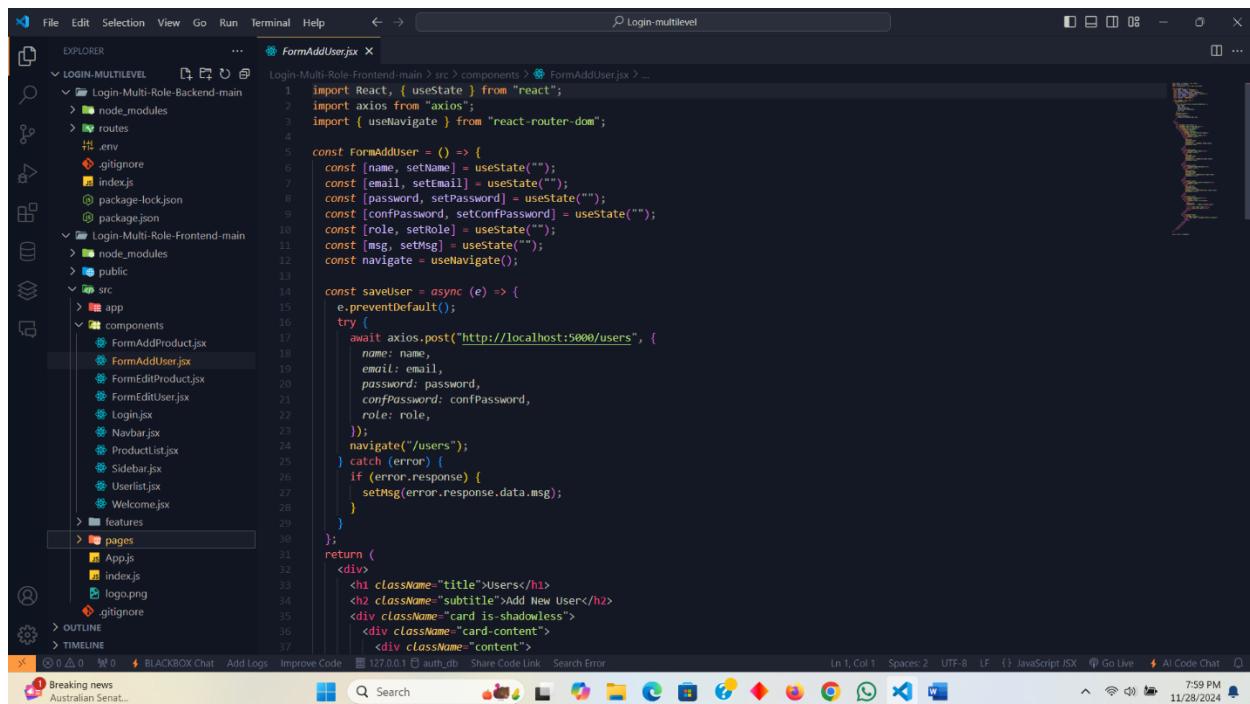


The screenshot shows the VS Code interface with the 'FormAddProduct.jsx' file open in the center editor pane. The code is a functional component for adding a product, using JSX and props to render a form with two text input fields for name and price.

```
const FormAddProduct = () => {
  return (
    <div>
      <h1>Products</h1>
      <h2>Add New Product</h2>
      <div>
        <div>
          <form onSubmit={saveProduct}>
            <p>Name</p>
            <div>
              <label>Name</label>
              <div>
                <input type="text" value={name} onChange={(e) => setName(e.target.value)} placeholder="Product Name" />
              </div>
            </div>
            <div>
              <label>Price</label>
              <div>
                <input type="text" value={price} onChange={(e) => setPrice(e.target.value)} placeholder="Price" />
              </div>
            </div>
          </form>
        </div>
      </div>
    </div>
  );
}

function saveProduct(e) {
  e.preventDefault();
  const { name, price } = e.target.elements;
  const user = { name, price };
  axios.post('http://localhost:5000/users', user);
  navigate('/users');
}
```

### -FormAddUser.jsx



The screenshot shows the VS Code interface with the 'FormAddUser.jsx' file open in the center editor pane. This component handles user addition via an asynchronous POST request to 'http://localhost:5000/users'. It uses useState for state management and react-router-dom for navigation.

```
const FormAddUser = () => {
  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [confPassword, setConfPassword] = useState("");
  const [role, setRole] = useState("");
  const [msg, setMsg] = useState("");
  const navigate = useNavigate();

  const saveUser = async (e) => {
    e.preventDefault();
    try {
      await axios.post("http://localhost:5000/users", {
        name,
        email,
        password,
        confPassword,
        role,
      });
      navigate("/users");
    } catch (error) {
      if (error.response) {
        setMsg(error.response.data.msg);
      }
    }
  };
}

return (
  <div>
    <h1>Users</h1>
    <h2>Add New User</h2>
    <div>
      <div>
        <form>
          <input type="text" value={name} onChange={(e) => setName(e.target.value)} placeholder="Name" />
          <input type="text" value={email} onChange={(e) => setEmail(e.target.value)} placeholder="Email" />
          <input type="password" value={password} onChange={(e) => setPassword(e.target.value)} placeholder="Password" />
          <input type="password" value={confPassword} onChange={(e) => setConfPassword(e.target.value)} placeholder="Confirm Password" />
          <select value={role} onChange={(e) => setRole(e.target.value)} placeholder="Role">
            <option value="admin">Admin</option>
            <option value="user">User</option>
          </select>
          <button type="submit" onClick={saveUser}>Submit</button>
        </form>
      </div>
    </div>
  </div>
);
```

## -FormEditProduct.jsx

The screenshot shows the VS Code interface with the file 'FormEditProduct.jsx' open in the center. The code handles product editing logic, including fetching a product by ID and updating it via a patch request. The code uses React hooks like useState and useEffect.

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import { useNavigate, useParams } from "react-router-dom";

const FormEditProduct = () => {
  const [name, setName] = useState("");
  const [price, setPrice] = useState("");
  const [msg, setMsg] = useState("");
  const navigate = useNavigate();
  const { id } = useParams();

  useEffect(() => {
    const getProductById = async () => {
      try {
        const response = await axios.get(`http://localhost:5000/products/${id}`);
        setName(response.data.name);
        setPrice(response.data.price);
      } catch (error) {
        if (error.response) {
          setMsg(error.response.data.msg);
        }
      }
    };
    getProductById();
  }, [id]);

  const updateProduct = async (e) => {
    e.preventDefault();
    try {
      await axios.patch(`http://localhost:5000/products/${id}`, {
        name: name,
        price: price,
      });
      navigate("/products");
    } catch (error) {
    }
  };
}

export default FormEditProduct;
```

## -FormEditUser.jsx

The screenshot shows the VS Code interface with the file 'FormEditUser.jsx' open in the center. The code handles user editing logic, including fetching a user by ID and updating it via a patch request. The code uses React hooks like useState and useEffect.

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import { useNavigate, useParams } from "react-router-dom";

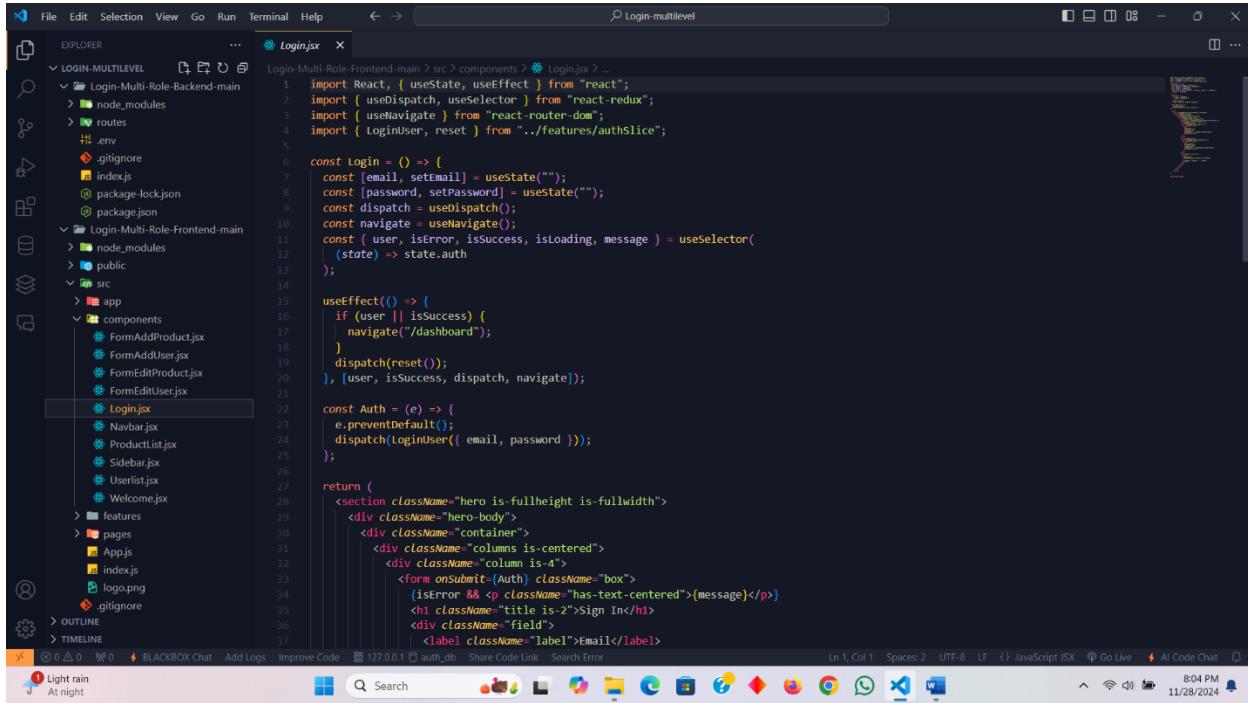
const FormEditUser = () => {
  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [confirmPassword, setConfirmPassword] = useState("");
  const [role, setRole] = useState("");
  const [msg, setMsg] = useState("");
  const navigate = useNavigate();
  const { id } = useParams();

  useEffect(() => {
    const getUserById = async () => {
      try {
        const response = await axios.get(`http://localhost:5000/users/${id}`);
        setName(response.data.name);
        setEmail(response.data.email);
        setRole(response.data.role);
      } catch (error) {
        if (error.response) {
          setMsg(error.response.data.msg);
        }
      }
    };
    getUserById();
  }, [id]);

  const updateUser = async (e) => {
    e.preventDefault();
    try {
      await axios.patch(`http://localhost:5000/users/${id}`, {
        name: name,
        email: email,
        password: password,
      });
      navigate("/users");
    } catch (error) {
    }
  };
}

export default FormEditUser;
```

## -Login.jsx



```
import React, { useState, useEffect } from "react";
import { useDispatch, useSelector } from "react-redux";
import { useNavigate } from "react-router-dom";
import { loginUser, reset } from "../features/authSlice";

const Login = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const [user, isError, isSuccess, isLoading, message] = useSelector(
    (state) => state.auth
  );

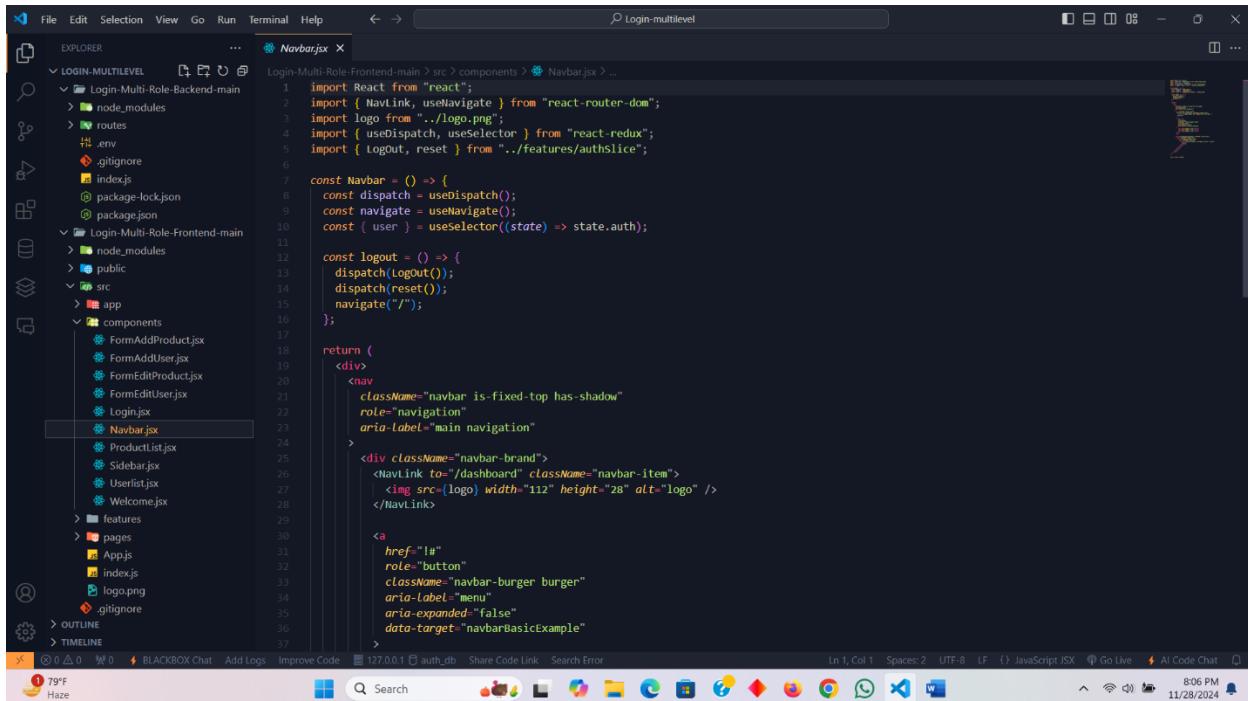
  useEffect(() => {
    if (user || isSuccess) {
      navigate("/dashboard");
    }
    dispatch(reset());
  }, [user, isSuccess, dispatch, navigate]);

  const Auth = (e) => {
    e.preventDefault();
    dispatch(loginUser({ email, password }));
  };

  return (
    <section className="hero is-fullheight is-fullwidth">
      <div className="hero-body">
        <div className="container">
          <div className="columns is-centered">
            <div className="column is-4">
              <form onSubmit={Auth} className="box">
                {isError && <p className="has-text-centered">{message}</p>}
                <h1 className="title is-2">Sign In</h1>
                <div className="field">
                  <label className="label">Email:</label>
                  <input type="text" value={email} onChange={(e) => setEmail(e.target.value)} />
                </div>
                <div className="field">
                  <label className="label">Password:</label>
                  <input type="password" value={password} onChange={(e) => setPassword(e.target.value)} />
                </div>
                <button type="submit" className="button is-primary">Login</button>
              </form>
            </div>
          </div>
        </div>
      </div>
    </section>
  );
}

export default Login;
```

## -Navbar.jsx



```
import React from "react";
import { NavLink, useNavigate } from "react-router-dom";
import logo from "../logo.png";
import { useDispatch, useSelector } from "react-redux";
import { logout, reset } from "../features/authSlice";

const Navbar = () => {
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const [user] = useSelector((state) => state.auth);

  const logout = () => {
    dispatch(logout());
    dispatch(reset());
    navigate("/");
  };

  return (
    <div>
      <nav className="navbar is-fixed-top has-shadow" role="navigation" aria-label="main navigation">
        <div className="navbar-brand">
          <NavLink to="/dashboard" className="navbar-item">
            <img alt="Logo" src={logo} width="112" height="28" />
          </NavLink>
        </div>
        <a href="#" role="button" className="nav-burger burger" aria-label="menu" aria-expanded="false" data-target="navbarBasicExample">
          <span></span>
        </a>
      </nav>
    </div>
  );
}

export default Navbar;
```

## -ProductList.jsx

The screenshot shows the VS Code interface with the ProductList.jsx file open in the editor. The code is a functional component that fetches products from a local host API and displays them in a table. It includes imports for React, useState, useEffect, axios, and react-router-dom. The code uses useState to manage the product list and useEffect to fetch data from the API. The table has columns for No, Product Name, Price, Created By, and Actions.

```
const ProductList = () => {
  const [products, setProducts] = useState([]);
  useEffect(() => {
    getProducts();
  }, []);
  const getProducts = async () => {
    const response = await axios.get("http://localhost:5000/products");
    setProducts(response.data);
  };
  return (
    <div>
      <h1>Products</h1>
      <h2>List of Products</h2>
      <a href="/products/add" class="button is-primary mb-2">
        Add New
      </a>
      <table class="table is-striped is-fullwidth">
        <thead>
          <tr>
            <th>No</th>
            <th>Product Name</th>
            <th>Price</th>
            <th>Created By</th>
            <th>Actions</th>
          </tr>
        </thead>
        <tbody>
          {products.map((product) => (
            <tr>
              <td>{product.id}</td>
              <td>{product.name}</td>
              <td>{product.price}</td>
              <td>{product.created_by}</td>
              <td>
                <a href="#">Edit</a>
                <a href="#">Delete</a>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
}
```

## -Sidebar.jsx

The screenshot shows the VS Code interface with the Sidebar.jsx file open in the editor. The code defines a sidebar component that contains a menu with links to dashboard, products, and home pages. It uses NavLink, useNavigate, and useDispatch hooks from react-router-dom, along with react-icons and react-redux. The sidebar also handles logout functionality by dispatching logOut and reset actions.

```
const sidebar = () => {
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const { user } = useSelector((state) => state.auth);

  const logout = () => {
    dispatch(logOut());
    dispatch(reset());
    navigate("/");
  };

  return (
    <div>
      <aside className="menu pl-2 has-shadow">
        <p className="menu-label">general</p>
        <ul className="menu-list">
          <li>
            <NavLink to="/dashboard">
              <IoHome /> Dashboard
            </NavLink>
          </li>
          <li>
            <NavLink to="/products">
              <IoPricetag /> Products
            </NavLink>
          </li>
        </ul>
        {user && user.role === "admin" && (
          <div>
            <p>Admin</p>
            <ul className="menu-list">
              <li>
                <NavLink to="/users">
                  <IoUser /> Users
                </NavLink>
              </li>
            </ul>
          </div>
        )}
      </aside>
    </div>
  );
}
```

## -Userlist.jsx

The screenshot shows the VS Code interface with the Userlist.jsx file open in the editor. The code handles user retrieval and deletion via an axios call to 'localhost:5000/users'.

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import { Link } from "react-router-dom";

const Userlist = () => {
  const [users, setusers] = useState([]);

  useEffect(() => {
    getusers();
  }, [ ]);

  const getusers = async () => {
    const response = await axios.get("http://localhost:5000/users");
    setusers(response.data);
  };

  const deleteUser = async (userId) => {
    await axios.delete(`http://localhost:5000/users/${userId}`);
    getusers();
  };

  return (
    <div>
      <h1>Users</h1>
      <h2>List of Users</h2>
      <a href="/users/add" type="button" class="is-primary mb-2">
        Add New
      </a>
      <table class="table is-striped is-fullwidth">
        <thead>
          <tr>
            <th>No</th>
            <th>Name</th>
            <th>Email</th>
            <th>Role</th>
            <th>Actions</th>
          </tr>
        </thead>
        <tbody>
          <tr><td>1</td><td>John Doe</td><td>john.doe@example.com</td><td>Admin</td><td><button type="button" class="is-primary">Edit</button> <button type="button" class="is-danger">Delete</button></td></tr>
        </tbody>
      </table>
    </div>
  );
}

export default Userlist;
```

## -Welcome.jsx

The screenshot shows the VS Code interface with the Welcome.jsx file open in the editor. It uses react-redux to display a welcome message based on the user state.

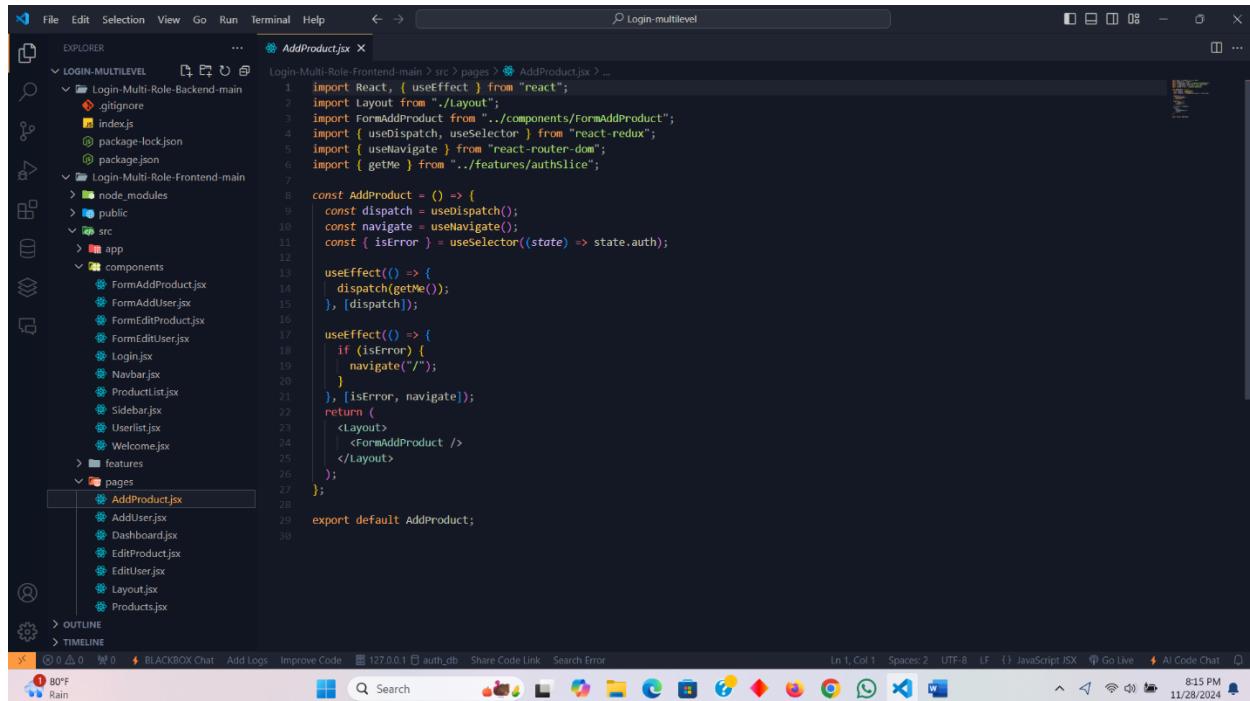
```
import React from "react";
import { useSelector } from "react-redux";

const Welcome = () => {
  const { user } = useSelector((state) => state.auth);
  return (
    <div>
      <h1>Dashboard</h1>
      <h2>Subtitle</h2>
      <p>Welcome Back <strong>{user && user.name}</strong></p>
    </div>
  );
};

export default Welcome;
```

## \*Pages

### -AddProduct.jsx



The screenshot shows the VS Code interface with the file `AddProduct.jsx` open in the editor. The code implements a functional component `AddProduct` using React hooks like `useEffect` and `useDispatch`. It handles navigation and dispatching actions related to product addition. The code is well-formatted with line numbers and syntax highlighting.

```
import React, { useEffect } from "react";
import Layout from "./Layout";
import FormAddProduct from "../components/FormAddProduct";
import { useDispatch, useSelector } from "react-redux";
import { useNavigate } from "react-router-dom";
import { getMe } from "../features/authSlice";

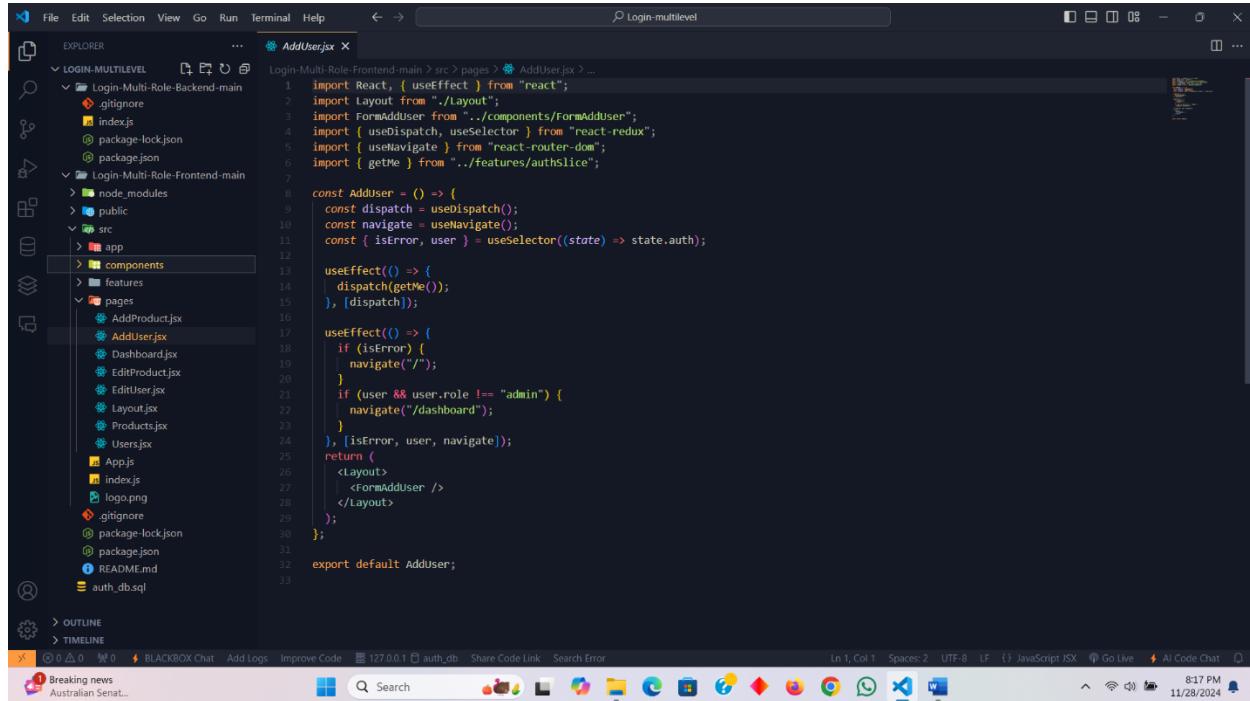
const AddProduct = () => {
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const [isError] = useSelector((state) => state.auth);

  useEffect(() => {
    dispatch(getMe());
  }, [dispatch]);

  useEffect(() => {
    if (isError) {
      navigate("/");
    }
  }, [isError, navigate]);
  return (
    <Layout>
      <FormAddProduct />
    </Layout>
  );
};

export default AddProduct;
```

### -AddUser.jsx



The screenshot shows the VS Code interface with the file `AddUser.jsx` open in the editor. The code defines a functional component `AddUser` that uses `useEffect`, `useDispatch`, and `useNavigate` hooks. It checks for errors and user roles to perform specific actions like navigating to a dashboard or displaying an error message. The code is annotated with line numbers and includes imports for components and the auth slice.

```
import React, { useEffect } from "react";
import Layout from "./Layout";
import FormAddUser from "../components/FormAddUser";
import { useDispatch, useSelector } from "react-redux";
import { useNavigate } from "react-router-dom";
import { getMe } from "../features/authSlice";

const AddUser = () => {
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const [isError, user] = useSelector((state) => state.auth);

  useEffect(() => {
    dispatch(getMe());
  }, [dispatch]);

  useEffect(() => {
    if (isError) {
      navigate("/");
    }
    if (user && user.role !== "admin") {
      navigate("/dashboard");
    }
  }, [isError, user, navigate]);
  return (
    <Layout>
      <FormAddUser />
    </Layout>
  );
};

export default AddUser;
```

## -Dashboard.jsx

The screenshot shows the VS Code interface with the 'Dashboard.jsx' file open in the center editor pane. The code implements a dashboard component using React hooks like useEffect and useSelector. It handles navigation and state management for a user session. The left sidebar displays the project structure, and the bottom status bar shows system information.

```
1 import React, { useEffect } from "react";
2 import Layout from "./Layout";
3 import Welcome from "./components/Welcome";
4 import { useDispatch, useSelector } from "react-redux";
5 import { useNavigate } from "react-router-dom";
6 import { getMe } from "../features/authSlice";
7
8 const Dashboard = () => {
9   const dispatch = useDispatch();
10  const navigate = useNavigate();
11  const { isError } = useSelector((state) => state.auth);
12
13  useEffect(() => {
14    if (isError) {
15      dispatch(getMe());
16    }
17  }, [dispatch]);
18
19  useEffect(() => {
20    if (!isError) {
21      navigate("/");
22    }
23  }, [isError, navigate]);
24
25  return (
26    <Layout>
27      <Welcome />
28    </Layout>
29  );
30
31  export default dashboard;
```

## -EditProduct.jsx

The screenshot shows the VS Code interface with the 'EditProduct.jsx' file open in the center editor pane. This component follows a similar structure to the Dashboard, using React hooks to manage state and perform actions like fetching user data. The left sidebar shows the project structure, and the bottom status bar indicates the file is 127.0.0.1 and 11/28/2024.

```
1 import React, { useEffect } from "react";
2 import Layout from "./Layout";
3 import FormEditProduct from "./components/FormEditProduct";
4 import { useDispatch, useSelector } from "react-redux";
5 import { useNavigate } from "react-router-dom";
6 import { getMe } from "../features/authSlice";
7
8 const Editproduct = () => {
9   const dispatch = useDispatch();
10  const navigate = useNavigate();
11  const { isError } = useSelector((state) => state.auth);
12
13  useEffect(() => {
14    dispatch(getMe());
15  }, [dispatch]);
16
17  useEffect(() => {
18    if (isError) {
19      navigate("/");
20    }
21  }, [isError, navigate]);
22
23  return (
24    <Layout>
25      <FormEditProduct />
26    </Layout>
27  );
28
29  export default EditProduct;
```

## -EditUser.jsx

The screenshot shows the VS Code interface with the file 'EditUser.jsx' open in the center editor pane. The code implements a function that handles user edits, using React hooks like useEffect and useState, along with Redux's useDispatch and useSelector. It checks if the user is an admin and navigates them to the dashboard if not. The code is well-formatted with proper indentation and comments.

```
1 import React, { useEffect } from "react";
2 import Layout from "./Layout";
3 import FormEditUser from "../components/FormEditUser";
4 import { useDispatch, useSelector } from "react-redux";
5 import { useNavigate } from "react-router-dom";
6 import { getMe } from "../features/authSlice";
7
8 const EditUser = () => {
9   const dispatch = useDispatch();
10  const navigate = useNavigate();
11  const { isError, user } = useSelector(({ state }) => state.auth);
12
13  useEffect(() => {
14    if (!isError) {
15      dispatch(getMe());
16    }
17  }, [dispatch]);
18
19  useEffect(() => {
20    if (isError) {
21      navigate("/");
22    }
23  }, [isError, user, navigate]);
24
25  if (user && user.role !== "admin") {
26    navigate("/dashboard");
27  }
28}, [isError, user, navigate]);
29
30 return (
31   <Layout>
32     <FormEditUser />
33   </Layout>
34 );
35
36 export default EditUser;
```

## -Layout.jsx

The screenshot shows the VS Code interface with the file 'Layout.jsx' open in the center editor pane. This component is a higher-order component that wraps other content in a React.Fragment. It includes a Navbar, a Sidebar, and a Main area. The Main area has a minimum height of 100vh and contains the children passed to the layout. The code uses CSS-in-JS for styling the main container.

```
1 import React from "react";
2 import Navbar from "../components/Navbar";
3 import Sidebar from "../components/sidebar";
4
5 const Layout = ({ children }) => {
6   return (
7     <React.Fragment>
8       <Navbar />
9       <div className="columns mt-6" style={{ minHeight: "100vh" }}>
10         <div className="column is-2">
11           <Sidebar />
12         </div>
13         <div className="column has-background-light">
14           <main>{children}</main>
15         </div>
16       </div>
17     </React.Fragment>
18   );
19 }
20
21 export default Layout;
```

## -Product.jsx

```
File Edit Selection View Go Run Terminal Help <- > Login-Multilevel
EXPLORER Products.jsx
1 import React, { useEffect } from "react";
2 import Layout from "./Layout";
3 import ProductList from "../components/ProductList";
4 import { useDispatch, useSelector } from "react-redux";
5 import { useNavigate } from "react-router-dom";
6 import { getMe } from "../features/authSlice";
7
8 const Products = () => {
9   const dispatch = useDispatch();
10  const navigate = useNavigate();
11  const { isError } = useSelector((state) => state.auth);
12
13  useEffect(() => {
14    dispatch(getMe());
15  }, [dispatch]);
16
17  useEffect(() => {
18    if (isError) {
19      navigate("/");
20    }
21  }, [isError, navigate]);
22
23  return (
24    <Products>
25      <ProductList />
26    </Products>
27  );
28
29 };
30
31 export default Products;
```

CHI - DET  
In 4 hours

File Edit Selection View Go Run Terminal Help <- > Login-Multilevel

OUTLINE TIMELINE

BLACKBOX Chat Add Logs Improve Code 127.0.0.1 auth\_db Share Code Link Search Error

Ln 1, Col 1 Spaces: 2 UTF-8 LF JavaScript JSX Go Live AI Code Chat

8:19 PM 11/28/2024

## -Users.jsx

```
File Edit Selection View Go Run Terminal Help <- > Login-Multilevel
EXPLORER Users.jsx
1 import React, { useEffect } from "react";
2 import Layout from "./Layout";
3 import UserList from "../components/UserList";
4 import { useDispatch, useSelector } from "react-redux";
5 import { useNavigate } from "react-router-dom";
6 import { getMe } from "../features/authSlice";
7
8 const Users = () => {
9   const dispatch = useDispatch();
10  const navigate = useNavigate();
11  const { isError, user } = useSelector((state) => state.auth);
12
13  useEffect(() => {
14    dispatch(getMe());
15  }, [dispatch]);
16
17  useEffect(() => {
18    if (isError) {
19      navigate("/");
20    }
21
22    if (user && user.role !== "admin") {
23      navigate("/dashboard");
24    }
25  }, [isError, user, navigate]);
26
27  return (
28    <Users>
29      <UserList />
30    </Users>
31  );
32
33 export default users;
```

CHI - DET  
In 4 hours

File Edit Selection View Go Run Terminal Help <- > Login-Multilevel

OUTLINE TIMELINE

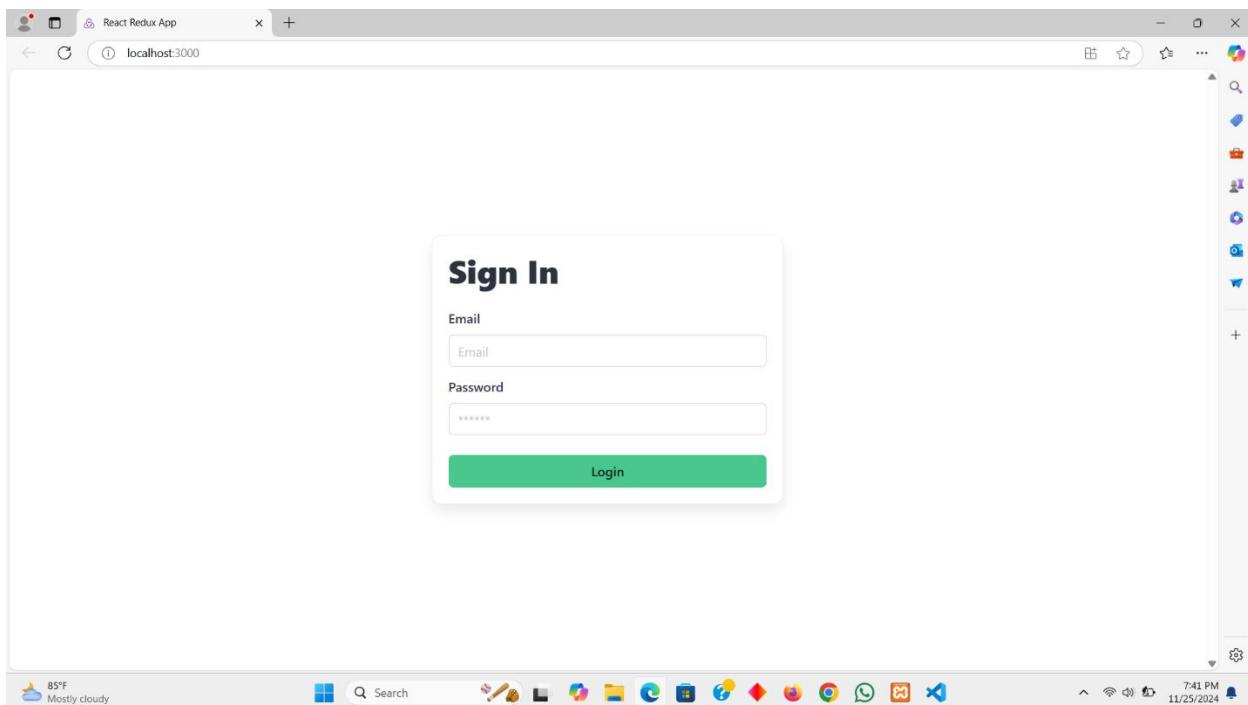
BLACKBOX Chat Add Logs Improve Code 127.0.0.1 auth\_db Share Code Link Search Error

Ln 1, Col 1 Spaces: 2 UTF-8 LF JavaScript JSX Go Live AI Code Chat

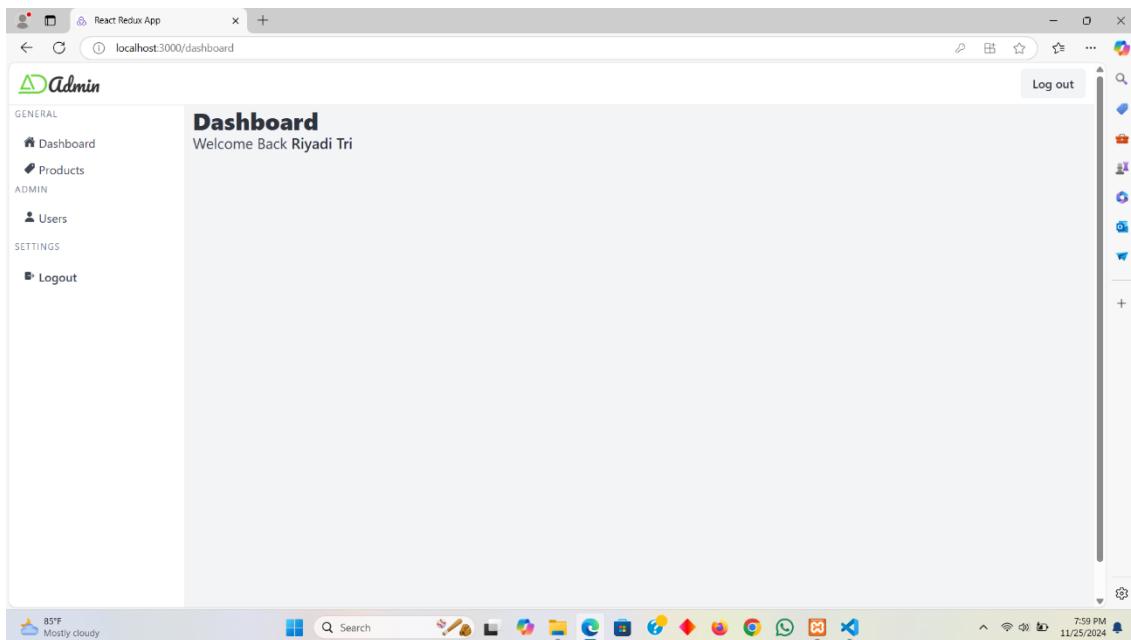
8:19 PM 11/28/2024

## \*Tampilan Web

-Sign in



-Tampilan Dashboard Sebagai Admin



Bedanya kalau Admin kita bisa mengedit siapa saja yang mau menjadi admin ataupun users sedangkan kalau login sebagai users tidak bisa

## -Tampilan Dashboard Sebagai Users

The screenshot shows a web application interface for an administrator. The top navigation bar includes a user icon, a refresh button, and a search icon. The address bar displays "localhost:3000/dashboard". The left sidebar, titled "Admin", has two main sections: "GENERAL" and "SETTINGS". Under "GENERAL", there are links for "Dashboard", "Products", and "Logout". Under "SETTINGS", there is a "Logout" link. The main content area is titled "Dashboard" and welcomes "John Doe". On the right side, there is a vertical sidebar with various icons. At the bottom, the Windows taskbar shows the date and time as "8:05 PM 11/25/2024".

## -Tampilan List Products Admin

The screenshot shows a web application interface for managing products. The top navigation bar includes a user icon, a refresh button, and a search icon. The address bar displays "localhost:3000/products". The left sidebar, titled "Admin", has sections for "GENERAL", "ADMIN" (with "Users"), and "SETTINGS". Under "GENERAL", there are links for "Dashboard", "Products", and "Logout". Under "ADMIN", there is a "Users" link. Under "SETTINGS", there is a "Logout" link. The main content area is titled "Products" and shows a "List of Products". It includes a green "Add New" button and a table with three rows of data. The table columns are "No", "Product Name", "Price", "Created By", and "Actions" (with "Edit" and "Delete" buttons). A vertical sidebar on the right contains various icons. At the bottom, the Windows taskbar shows the date and time as "8:05 PM 11/25/2024".

## -Tampilan List Products Users

React Redux App

localhost:3000/products

Admin

GENERAL

Dashboard

Products

SETTINGS

Logout

Products

List of Products

Add New

No	Product Name	Price	Created By	Actions
1	Jaket Gelembung	300000	John Doe	Edit Delete
2	Mobil Tamiya	125000	John Doe	Edit Delete
3	Tv LCD Merk Polytron	1849000	John Doe	Edit Delete

85°F Mostly cloudy

8:05 PM 11/25/2024

## -Menambahkan Barang Sebagai Admin

React Redux App

localhost:3000/products/edit/51b63e69-dda9-4af3-81f3-4a35b91abc05

Admin

GENERAL

Dashboard

Products

ADMIN

Users

SETTINGS

Logout

Products

Edit Product

Name

Hoodie Solitude

Price

250000

Update

85°F Mostly cloudy

7:55 PM 11/25/2024

## -Menambahkan Barang Sebagai Users

The screenshot shows a web application titled "React Redux App" running on "localhost:3000/products/add". The interface includes a sidebar with "GENERAL" and "SETTINGS" sections, and a main content area for "Add New Product". In the "Name" field, "Mobil Tamiya" is entered. In the "Price" field, "125000" is entered. A green "Save" button is present at the bottom of the form.

## -Logout

The screenshot shows a web application titled "React Redux App" running on "localhost:3000/dashboard". The interface includes a sidebar with "GENERAL", "ADMIN", and "SETTINGS" sections, and a main content area for "Welcome Back Riyadi Tri". A green "Logout" button is visible at the top right of the main area.

### **\*Kesimpulan**

Autentikasi website adalah langkah krusial untuk melindungi data pengguna dan mencegah akses tidak sah. Praktikum ini menekankan pentingnya penggunaan hashing password, validasi input, token autentikasi (seperti JWT), dan penerapan protokol keamanan seperti HTTPS. Pengujian keamanan dan penerapan fitur seperti logout aman dan multi-factor authentication (MFA) sangat diperlukan untuk menjaga integritas sistem