Laporan Praktikum

Mata Kuliah

PEMROGRAMAN BERORIENTASI OBJEK

Praktikum

Dosen Pengampu :
Willdan Aprizal Arifin, S.Pd., M.Kom.

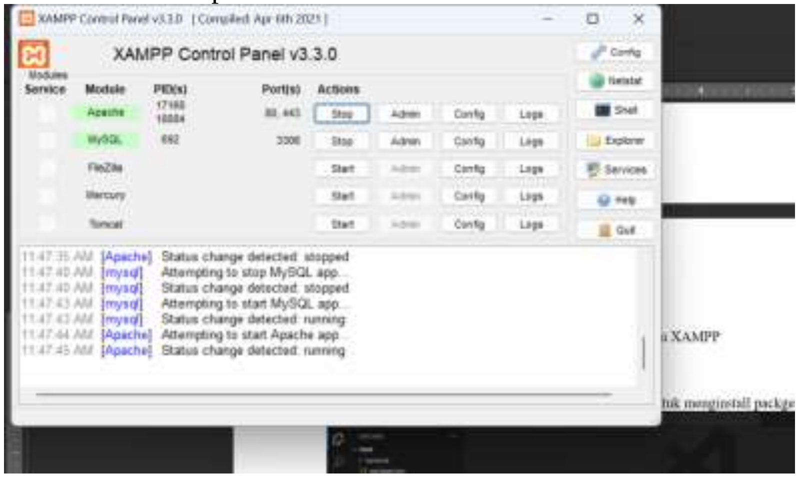
Disusun Oleh:
Serliya
2306837

PROGRAM STUDI SISTEM INFORMASI KELAUTANUNIVERSITAS PENDIDIKAN INDONESIA

2024

Langkah-Langkah

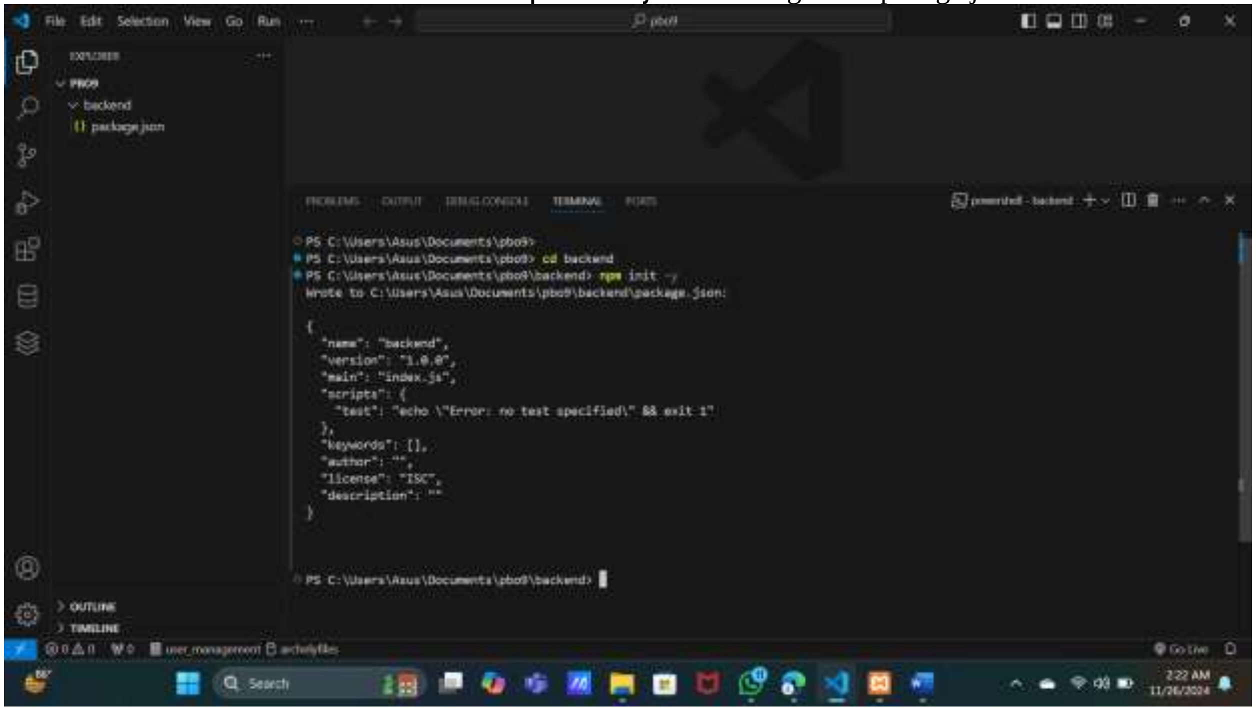
Sebelum masuk aplikasi VSCODE aktifkan terlebih dahulu XAMPP


XAMPP Control Panel v3.3.0 [Compiled: Apr 6th 2021]
XAMPP Control Panel v3.3.0
Config
Netstat
Shell
Explorer
Services
Help
Quit

Service	Module	PID(s)	Port(s)	Actions			
	Apache	17168 16884	80, 443	Stop	Admin	Config	Logs
	MySQL	692	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

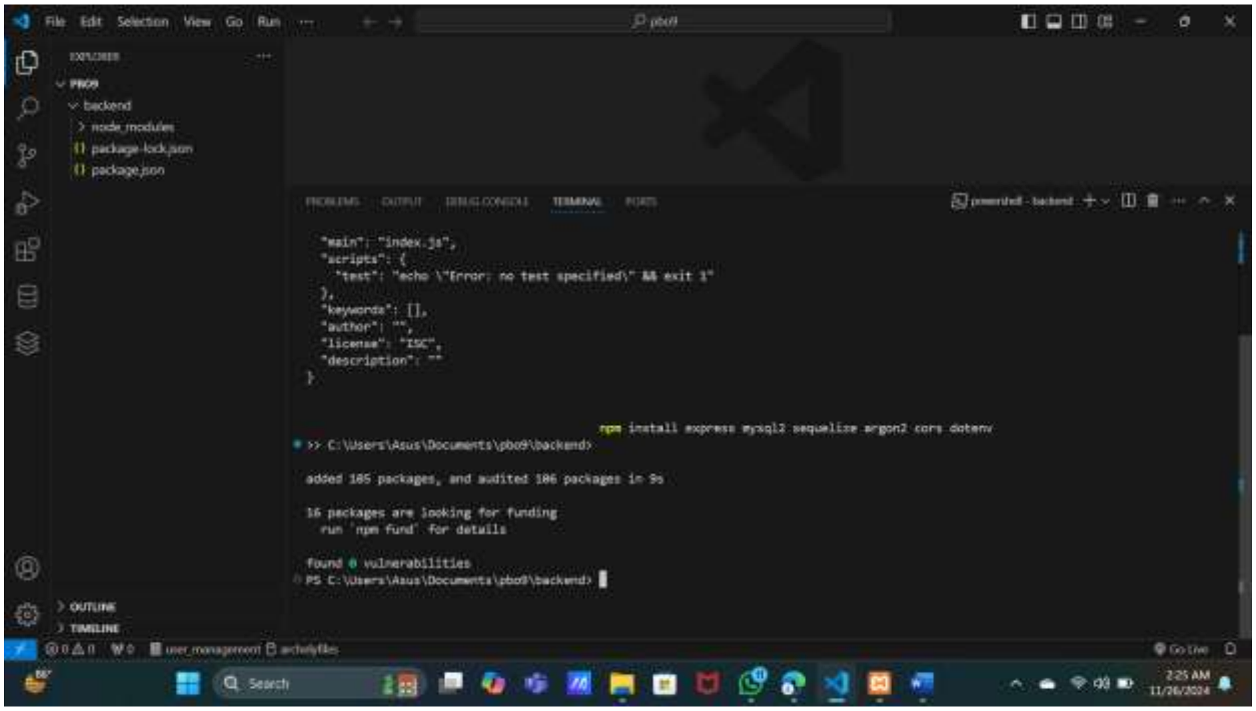
```
11:47:35 AM  [Apache]  Status change detected: stopped
11:47:40 AM  [mysql]   Attempting to stop MySQL app...
11:47:40 AM  [mysql]   Status change detected: stopped
11:47:43 AM  [mysql]   Attempting to start MySQL app...
11:47:43 AM  [mysql]   Status change detected: running
11:47:44 AM  [Apache]  Attempting to start Apache app...
11:47:45 AM  [Apache]  Status change detected: running
```

Buat folder baru, buka aplikasi vscode pilih “open folder”
buat sub folder bernama “backend”
Pada terminal ketik **cd backend** lalu ketik **npm init -y** untuk menginstall packge.json



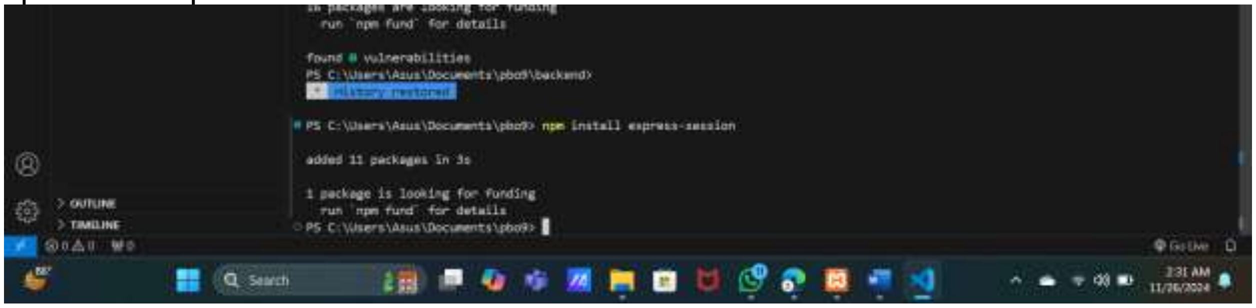
```
PS C:\Users\Asus\Documents\pbo9>
PS C:\Users\Asus\Documents\pbo9> cd backend
PS C:\Users\Asus\Documents\pbo9\backend> npm init -y
Wrote to C:\Users\Asus\Documents\pbo9\backend\package.json:

{
  "name": "backend",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}


PS C:\Users\Asus\Documents\pbo9\backend>
```

npm install express mysql2 sequelize argon2 cors dotenv



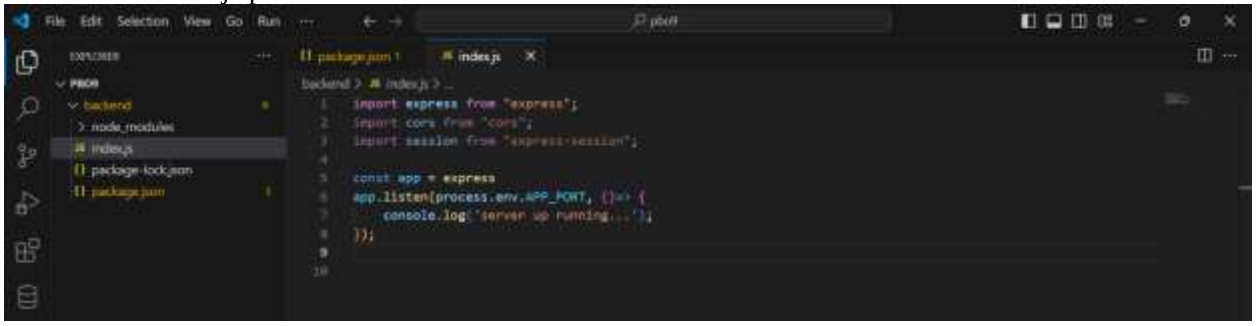
```
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}

                                          npm install express mysql2 sequelize argon2 cors dotenv
>> C:\Users\Asus\Documents\pbo9\backend>

added 105 packages, and audited 106 packages in 9s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\Asus\Documents\pbo9\backend>
```

npm install express-session



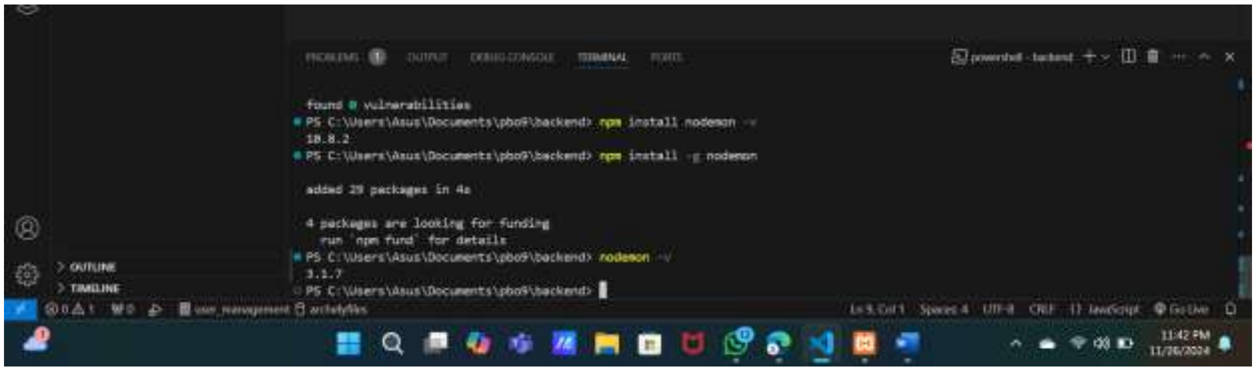
```
16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\Asus\Documents\pbo9\backend>
 * History restored

PS C:\Users\Asus\Documents\pbo9> npm install express-session

added 11 packages in 3s

1 package is looking for funding
  run `npm fund` for details
PS C:\Users\Asus\Documents\pbo9>
```

Buatlah file index.js pada folder backend



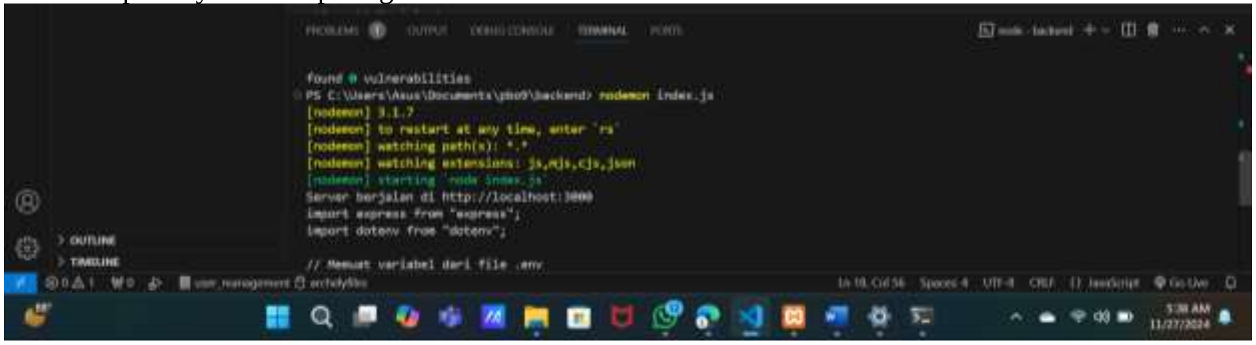
```
import express from "express";
import cors from "cors";
import session from "express-session";

const app = express
app.listen(process.env.APP_PORT, ()=> {
    console.log('server up running...');
});
```

Install nodemon -v secara global

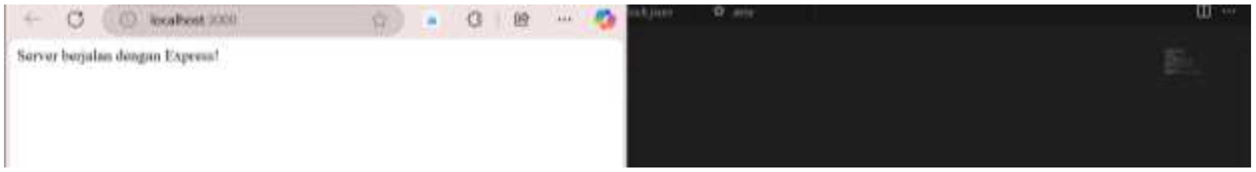


```
found 0 vulnerabilities
PS C:\Users\Asus\Documents\pbo9\backend> npm install nodemon -v
10.8.2
PS C:\Users\Asus\Documents\pbo9\backend> npm install -g nodemon

added 29 packages in 4s

4 packages are looking for funding
  run `npm fund` for details
PS C:\Users\Asus\Documents\pbo9\backend> nodemon -v
3.1.7
PS C:\Users\Asus\Documents\pbo9\backend> 
```

Maka tampilannya akan seperti gambar dibawah

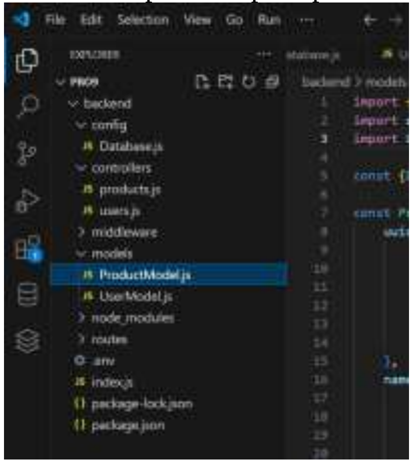


```
found 0 vulnerabilities
PS C:\Users\Asus\Documents\pbo9\backend> nodemon index.js
[nodemon] 3.1.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
Server berjalan di http://localhost:3000
import express from "express";
import dotenv from "dotenv";

// Memuat variabel dari file .env
```



Server berjalan dengan Express!

Buat beberapa file seperti pada video



Pada folder models, buatlah file UserModels.js dan ProductsModel.js kodenya sebagai berikut;

UsersModel.js	ProductsModel.js
`import {Sequelize} from "sequelize";`	`import {Sequelize} from "sequelize";`

```
import db from "/config/Database.js";

const {DataTypes} = Sequelize;

const Users = db.define('users',{}, {
    uuid: {
        type: DataTypes.STRING,
        defaultValue:
DataTypes.UUIDV4,
        allowNull: false,
        validate: {
            notEmpty: true
        }
    },
    name:{
        type: DataTypes.STRING,
        allowNull: false,
        validate: {
            notEmpty: true,
            len: [3, 100]
        }
    },
    email:{
        type: DataTypes.STRING,
        allowNull: false,
        validate: {
            notEmpty: true,
            isEmail: true
        }
    },
    password:{
        type: DataTypes.STRING,
        allowNull: false,
        validate: {
            notEmpty: true
        }
    },
    role:{
        type: DataTypes.STRING,
        allowNull: false,
        validate: {
            notEmpty: true
        }
    }
},{
```

```
import db from "/config/Database.js";
import Users from "/UserModel.js";

const {DataTypes} = Sequelize;

const Products = db.define('products',
{
    uuid: {
        type: DataTypes.STRING,
        defaultValue:
DataTypes.UUIDV4,
        allowNull: false,
        validate: {
            notEmpty: true
        }
    },
    name:{
        type: DataTypes.STRING,
        allowNull: false,
        validate: {
            notEmpty: true,
            len: [3, 100]
        }
    },
    email:{
        type: DataTypes.STRING,
        allowNull: false,
        validate: {
            notEmpty: true,
            isEmail: true
        }
    },
    price:{
        type: DataTypes.INTEGER,
        allowNull: false,
        validate: {
            notEmpty: true
        }
    },
    userId:{
        type: DataTypes.INTEGER,
        allowNull: false,
        validate: {
            notEmpty: true
        }
    freezeTableName: true
});

export default Users;
```

```
    }
},{
    freezeTableName: true
});

Users.hasMany(Products);
Products.belongsTo(Users,
{foreigenKey: 'userId'});

export default Products;
```

Pada folder Controller, buatlah file UserModels.js dan ProductModel.js kodenya sebagai berikut;

UsersModel.js

```
import User from
"../models/UsersModel.js";
import argon2 from "argon2";

export const getUsers = async(req, res)
=>{
    try {
        const response = await
User.findAll({
            attributes:['uuid',
'email', 'role']
        });
        res.status(200).json(response);
    } catch (error) {
        res.status(500).json({msg:
error.message});
    }
}

export const getUserById = async(req,
res) =>{
    try {
        const response = await
User.findOne({
            where: {
                uuid: req.params.id
        }
        });
        res.status(200).json(response);
    } catch (error) {
```

ProductsModel.js

```
import Product from
"../models/ProductModel.js";
import argon2 from "argon2";

export const getProducts = (req, res)
=>{
    }
export const getProductById =
async(req, res) =>{
}

export const createProduct = (req,
res) =>{

}
export const updateProduct = (req,
res) =>{

}
export const deleteProduct = (req,
res) =>{

}
```

```
        res.status(500).json({msg:
error.message});
    }
}

export const createUser = async(req,
res) =>{
    const {name, email, password,
confPassword, role} = req.body;
    if(password !== confPassword)
return res.status(400).json({msg:
"Password dan Confirm Password tidak
cocok"});
    const hashPassword = await
argon2.hash(password);
    try {
        await User.create({
            name: name,
            email: email,
            password: hashPassword,
            role: role
        });
        res.status(201).json({msg:
"Register Berhasil"});
    }  catch (error) {
        res.status(400).json({msg:
error.message});
    }
}
export const updateUser = async(req,
res) =>{}
    const user = await User.findOne({
        where: {
            uuid: req.params.id
        }
    });
    if(!user) return
res.status(404).json({msg: "User tidak
ditemukan"});
    const {name, email, password,
confPassword, role} = req.body;
    let hashPassword;
    if(password === "" || password ===
null){
        hashPassword = user.password
    }else{
        hashPassword = await
argon2.hash(password);
    }
    if(password !== confPassword)
return res.status(400).json({msg:
"password dan Confirm Password tidak
cocok"});
    try {
        await User.create({
            name: name,
            email: email,
            password: hashPassword,
            role: role
        },{
            where:{
                id: user.id
            }
        });
        res.status(200).json({msg:
"User Updated"});
    }  catch (error) {
        res.status(400).json({msg:
error.message});
    }

export const deleteUser = async(req,
res) =>{
    const user = await User.findOne({
        where: {
            uuid: req.params.id
        }
    });
    if(!user) return
res.status(404).json({msg: "User tidak
ditemukan"});
    try {
        await User.destroy({
           where:{
                id: user.id
           }
        });
        res.status(200).json({msg:
"User Deleted"});
    } catch (error) {
        res.status(400).json({msg:
error.message});
    }
}
```

Pada folder routes, buatlah file UserModels.js dan ProductsModel.js kodenya sebagai berikut;

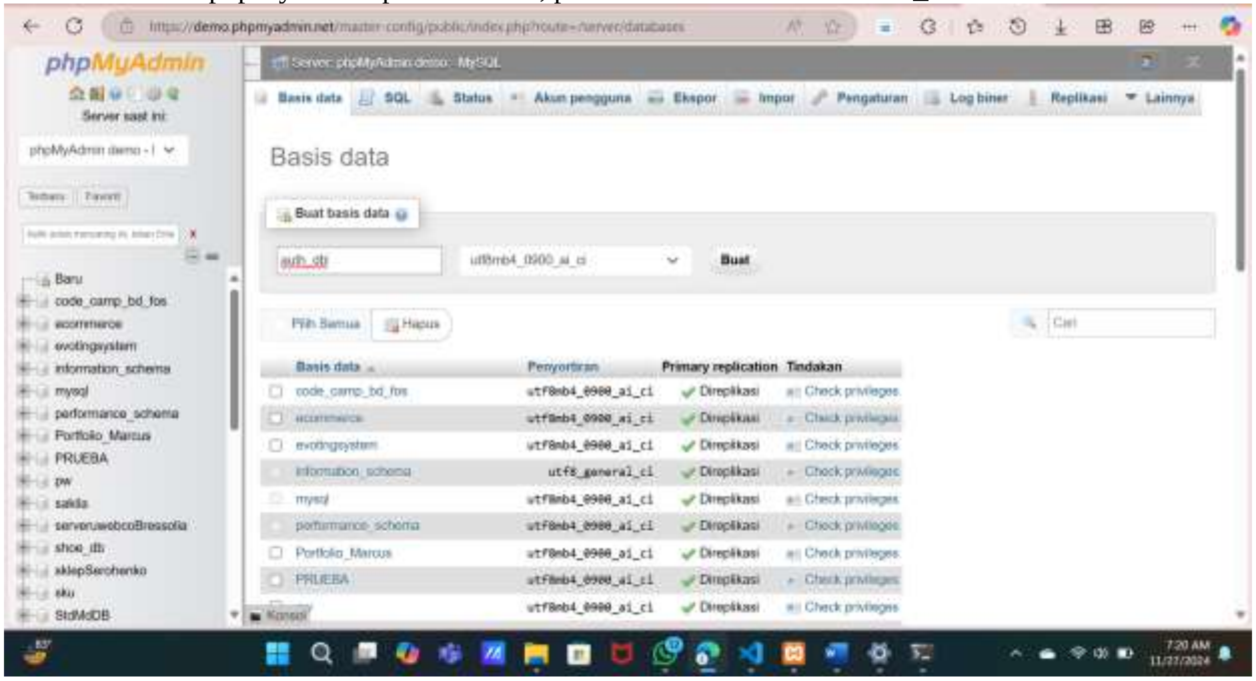
UserRoute.js

```
import express from "express";
import {

    getUsers,
    getUserById,
    createUser,
    UpdateUser,
    deleteUser
} from "/controllers/Users.js";


const router = express.Router();

router.get ('/users', getUsers);
router.get ('/users/:id',
getUserById);
router.post ('/users', createUsers);
router.patch ('/users/:id',
updateUsers);
router.delete ('/users/:id',
deleteUsers);
export default router;
```

ProductsRoute.js

```
import express from "express"
import {
    getProducts,
    getProductById,
    createProduct,
    UpdateProduct,
    deleteProduct
} from "../controllers/products.js";

const router = express.Router();

router.get ('/products', getProducts);
router.get ('/products/:id',
getProductById);
router.post ('/products',
createProduct);
router.patch ('/products/:id',
UpdateProduct);
router.delete ('/products/:id',
deleteProduct);
export default router;
```

search localhost//phpMyAdmin pada browser, pada basis data beri nama auth_db klik buat



https://demo.phpmyadmin.net/master-config/public/index.php?route=/server/databases

Server: phpMyAdmin demo · MySQL

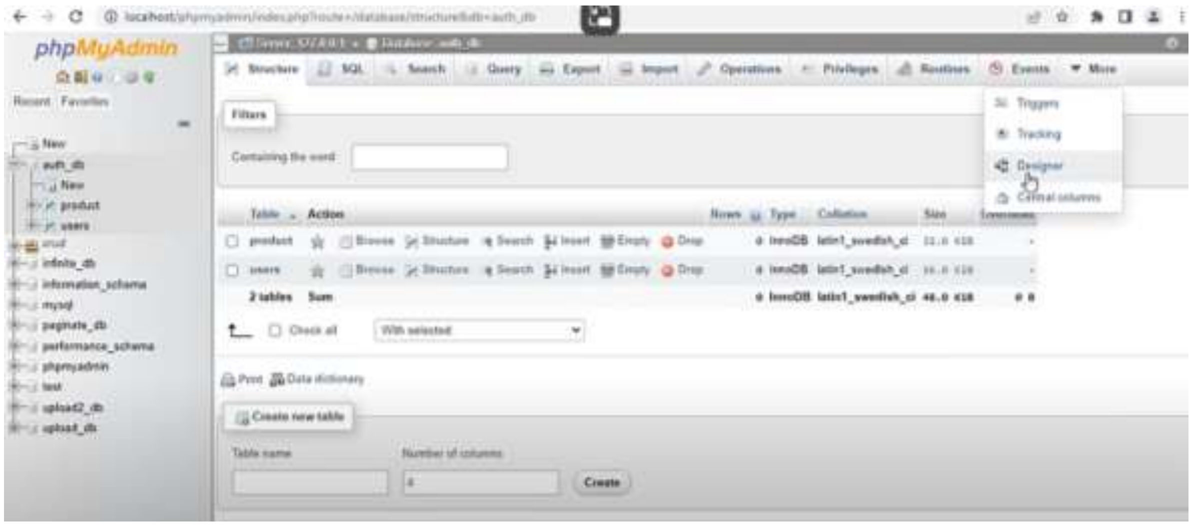
Basis data

Buat basis data

auth_db · utf8mb4_0900_ai_ci · Buat

	Basis data	Penyortiran	Primary replication	Tindakan
☐	code_camp_bd_fos	utf8mb4_0900_ai_ci	✓ Direplikasi	Check privileges
☐	ecommerce	utf8mb4_0900_ai_ci	✓ Direplikasi	Check privileges
☐	evotingsystem	utf8mb4_0900_ai_ci	✓ Direplikasi	Check privileges
	information_schema	utf8_general_ci	✓ Direplikasi	Check privileges
	mysql	utf8mb4_0900_ai_ci	✓ Direplikasi	Check privileges
	performance_schema	utf8mb4_0900_ai_ci	✓ Direplikasi	Check privileges
☐	Portfolio_Marcus	utf8mb4_0900_ai_ci	✓ Direplikasi	Check privileges
☐	PRUEBA	utf8mb4_0900_ai_ci	✓ Direplikasi	Check privileges
☐	pw	utf8mb4_0900_ai_ci	✓ Direplikasi	Check privileges

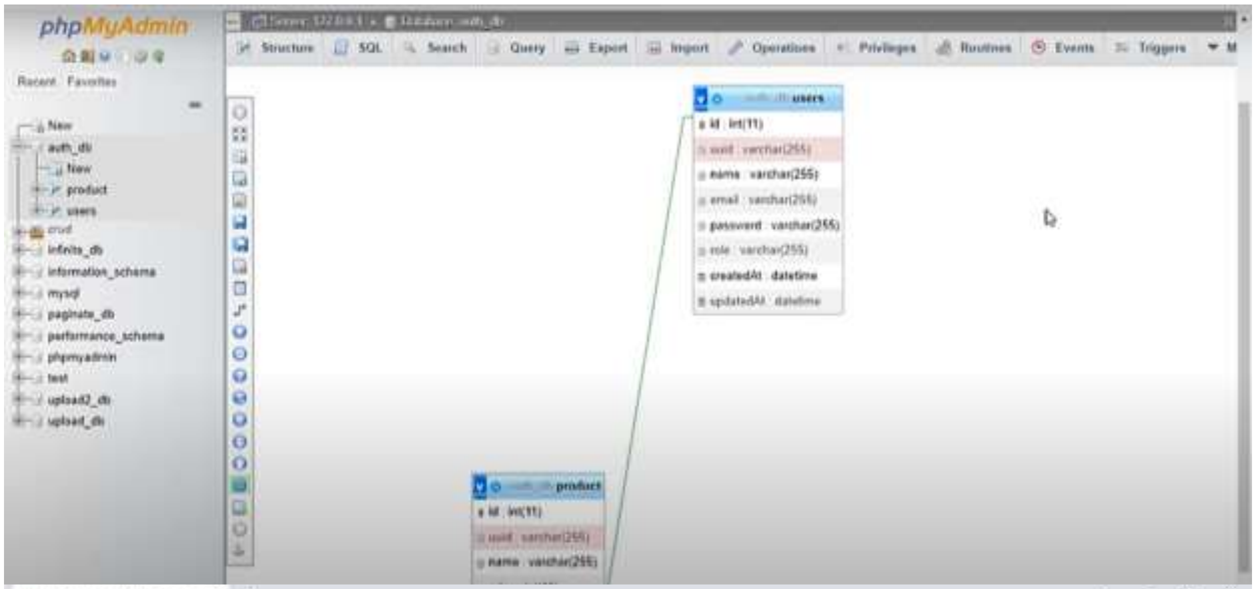
buka phpmyAdmin, maka databasenya akan terlihat



localhost/phpmyadmin/index.php?route=/database/structure&db=auth_db

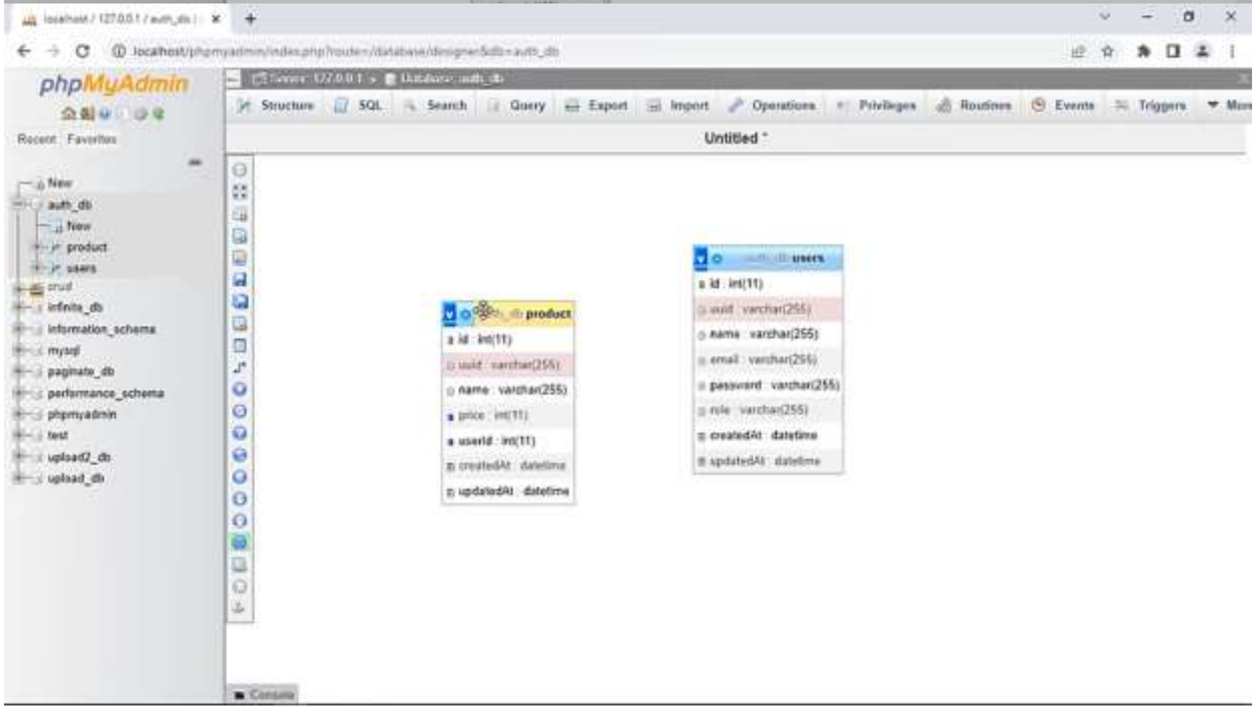
Server: 127.0.0.1 » Database: auth_db

	Table	Action							Rows	Type	Collation	Size	Overhead
☐	product	☆	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	latin1_swedish_ci	32.0 KiB	-
☐	users	☆	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	latin1_swedish_ci	16.0 KiB	-
	2 tables	Sum							0	InnoDB	latin1_swedish_ci	48.0 KiB	0 B



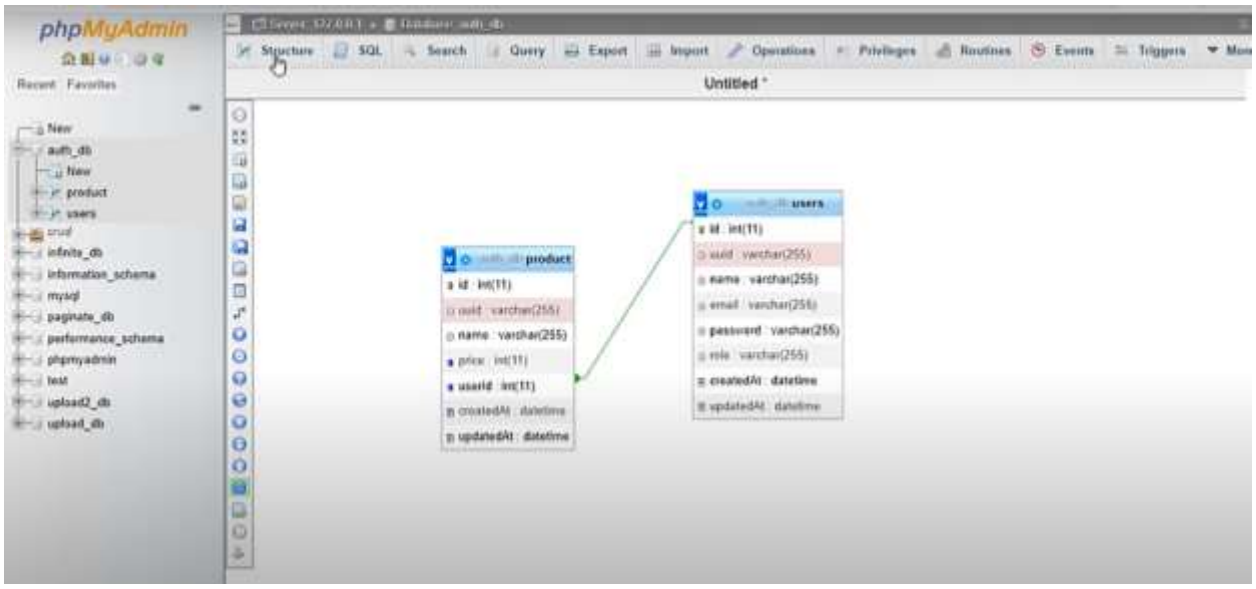
auth_db users
id : int(11)
uuid : varchar(255)
name : varchar(255)
email : varchar(255)
password : varchar(255)
role : varchar(255)
createdAt : datetime
updatedAt : datetime

auth_db product
id : int(11)
uuid : varchar(255)
name : varchar(255)



auth_db product
id : int(11)
uuid : varchar(255)
name : varchar(255)
price : int(11)
userId : int(11)
createdAt : datetime
updatedAt : datetime

auth_db users
id : int(11)
uuid : varchar(255)
name : varchar(255)
email : varchar(255)
password : varchar(255)
role : varchar(255)
createdAt : datetime
updatedAt : datetime



kode folder controller yang berisi;
auth.js

```
import User from "../moduls/UserModel.js";
import argon2 from "argon2";

export const Login = async (req, res) =>{
    const user = await User.findOne({
        where: {
            email: req.body.email
        }
    });
    if(!user) return res.status(404).json({msg: "User tidak ditemukan"});
    const match = await argon2.verify(user.password, req.body.password);
    if(!match) return res.status(400).json({msg: "Wrong Password"});
    req.session.userId = user.uuid;
    const uuid = user.uuid;
    const name = user.name;
    const email = user.email;
    const role = user.role;
    res.status(200).json({uuid, name, email, role});
}
export const Me = async (req, res) =>{
    if(!req.session.userId){
        return res.status(401).json({msg: "Mohon login ke akun Anda!"});
    }
    const user = await User.findOne({
        attributes:['uuid', 'name', 'email', 'password', 'role'],
        where: {
            uuid: req.session.userId
        }
    });
    if(!user) return res.status(404).json({msg: "User tidak ditemukan"});
    res.status(200).json(user);
}

export const logOut = (req, res) =>{
    req.session.destroy((err)=>{
        if(err) return res.status(400).json({msg: "Tidak dapat logout"});
        res.status(200).json({msg: "Anda telah logout "});

    });
}
```

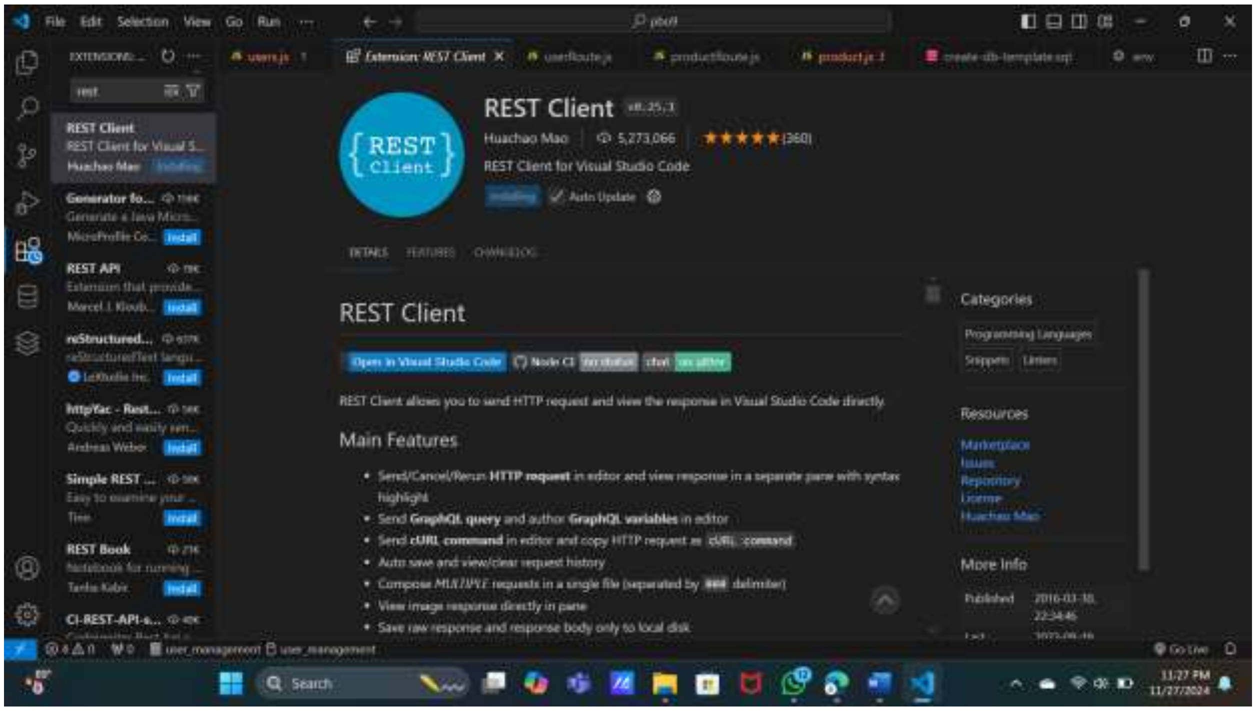
Product.js

```
import Product from "../models/ProductModel.js";
import argon2 from "argon2";

export const getProducts = (req, res) =>{
    }
export const getProductById = async(req, res) =>{
}

export const createProduct = (req, res) =>{

}
export const updateProduct = (req, res) =>{

}
export const deleteProduct = (req, res) =>{

}
```

User.js

```
import User from "../models/UsersModel.js";
import argon2 from "argon2";

export const getUsers = async(req, res) =>{
    try {
        const response = await User.findAll({
            attributes:['uuid', 'email', 'role']
        });
        res.status(200).json(response);
    } catch (error) {
        res.status(500).json({msg: error.message});
    }
}

export const getUserById = async(req, res) =>{
    try {
        const response = await User.findOne({
            where: {
                uuid: req.params.id
        }
        });
        res.status(200).json(response);
    } catch (error) {
        res.status(500).json({msg: error.message});
    }
}

export const createUser = async(req, res) =>{
    const {name, email, password, confPassword, role} = req.body;
    if(password !== confPassword) return res.status(400).json({msg: "Password dan
Confirm Password tidak cocok"});
    const hashPassword = await argon2.hash(password);
    try {
        await User.create({
            name: name,
            email: email,
            password: hashPassword,
            role: role
        });
        res.status(201).json({msg: "Register Berhasil"});
    }  catch (error) {
        res.status(400).json({msg: error.message});
    }
}
export const updateUser = async(req, res) =>{}
    const user = await User.findOne({
        where: {
            uuid: req.params.id
        }
    });
    if(!user) return res.status(404).json({msg: "User tidak ditemukan"});
    const {name, email, password, confPassword, role} = req.body;
    let hashPassword;
    if(password === "" || password === null){
        hashPassword = user.password
    }else{
        hashPassword = await argon2.hash(password);
    }
    if(password !== confPassword) return res.status(400).json({msg: "password dan Confirm Password tidak cocok"});
    try {
        await User.create({
            name: name,
            email: email,
            password: hashPassword,
            role: role
        },{
            where:{
                id: user.id
            }
        });
        res.status(200).json({msg: "User Updated"});
    }  catch (error) {
        res.status(400).json({msg: error.message});
    }

export const deleteUser = async(req, res) =>{
    const user = await User.findOne({
        where: {
            uuid: req.params.id
        }
    });
    if(!user) return res.status(404).json({msg: "User tidak ditemukan"});
    try {
        await User.destroy({
           where:{
                id: user.id
           }
        });
        res.status(200).json({msg: "User Deleted"});
    } catch (error) {
        res.status(400).json({msg: error.message});
    }
}
```

install REST Client



REST Client v0.25.1

Huachao Mao | 5,273,066 | ★★★★★ (360)

REST Client for Visual Studio Code

REST Client

REST Client allows you to send HTTP request and view the response in Visual Studio Code directly.

Main Features

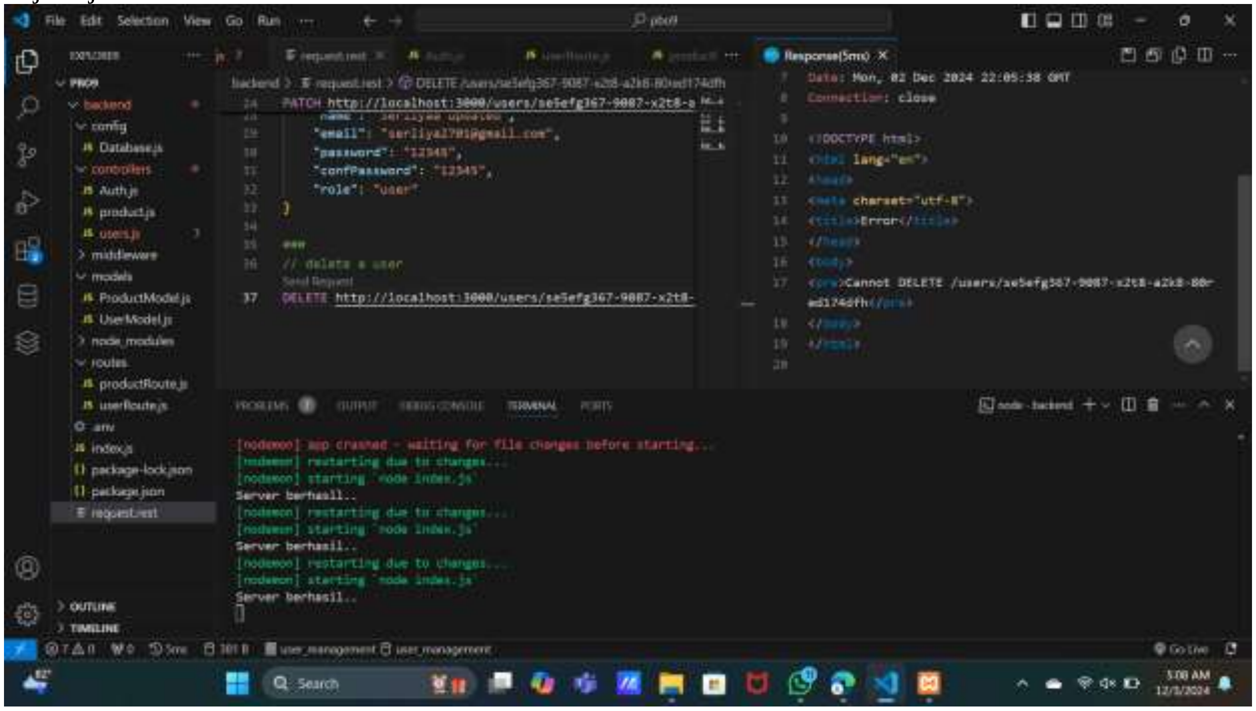
- Send/Cancel/Rerun **HTTP request** in editor and view response in a separate pane with syntax highlight
- Send **GraphQL query** and author **GraphQL variables** in editor
- Send **cURL command** in editor and copy HTTP request as `cURL command`
- Auto save and view/clear request history
- Compose *MULTIPLE* requests in a single file (separated by `###` delimiter)
- View image response directly in pane
- Save raw response and response body only to local disk

Categories: Programming Languages, Snippets, Linters

Resources: Marketplace, Issues, Repository, License, Huachao Mao

More Info: Published 2016-03-30, 22:34:46

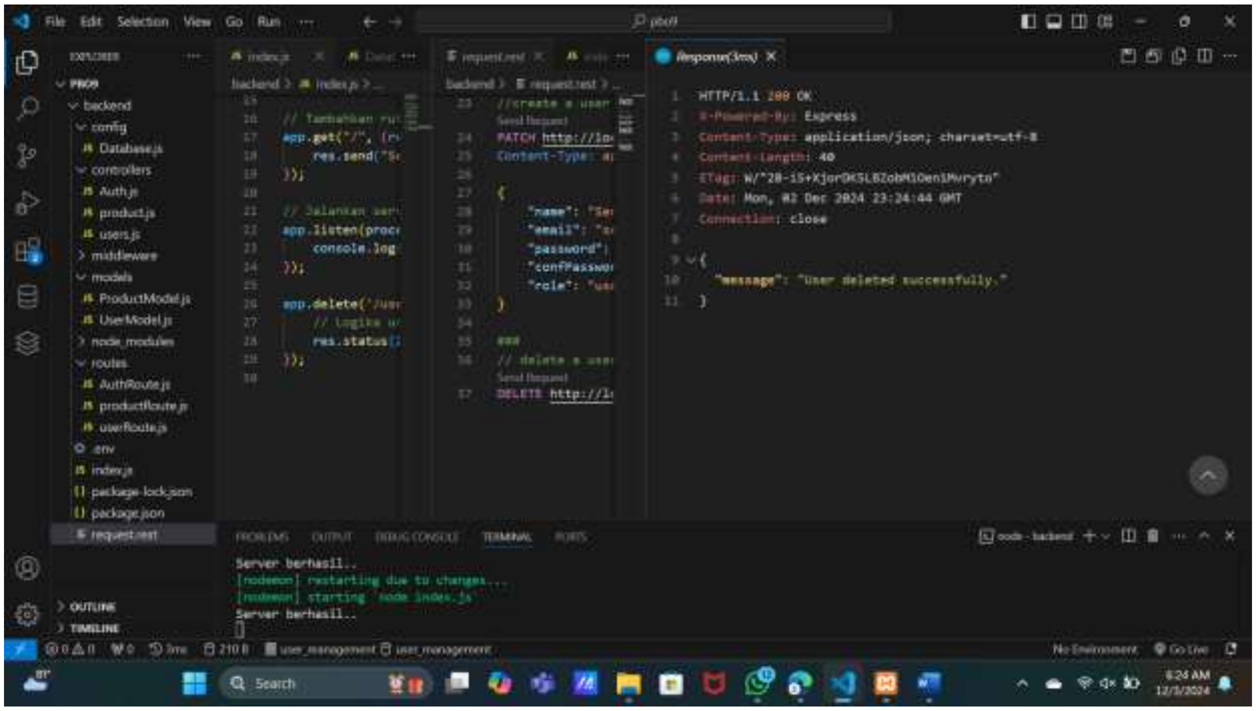
bhjXxsj



```
PATCH http://localhost:3000/users/se5efg367-9087-x2t8-a
    "name": "serliyaa updated",
    "email": "serliya2701@gmail.com",
    "password": "12345",
    "confPassword": "12345",
    "role": "user"
}

###
// delete a user
Send Request
DELETE http://localhost:3000/users/se5efg367-9087-x2t8-
```

```
Date: Mon, 02 Dec 2024 22:05:38 GMT
Connection: close

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Cannot DELETE /users/se5efg367-9087-x2t8-a2k8-80red174dfh</pre>
</body>
</html>
```

```
[nodemon] app crashed - waiting for file changes before starting...
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Server berhasil..
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Server berhasil..
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Server berhasil..
```

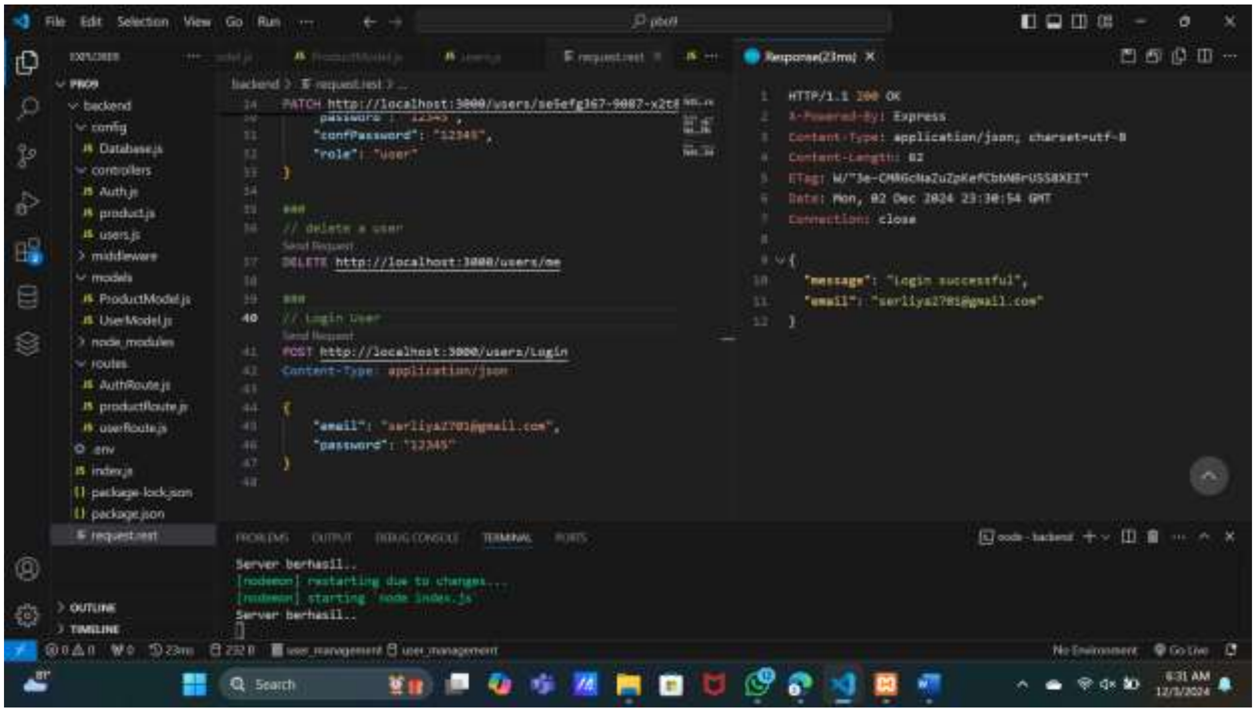


```
// Tambahkan ru
app.get("/", (r
    res.send("S
});

// Jalankan ser
app.listen(proc
    console.log
});

app.delete('/use
    // Logika u
    res.status(
});
```

```
//create a user
Send Request
PATCH http://lo
Content-Type: a

{
    "name": "Se
    "email": "s
    "password":
    "confPasswor
    "role": "use
}

###
// delete a user
Send Request
DELETE http://l
```

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 40
ETag: W/"28-15+KjorDKSLBZobM1Oen1Mvryto"
Date: Mon, 02 Dec 2024 23:24:44 GMT
Connection: close

{
  "message": "User deleted successfully."
}
```

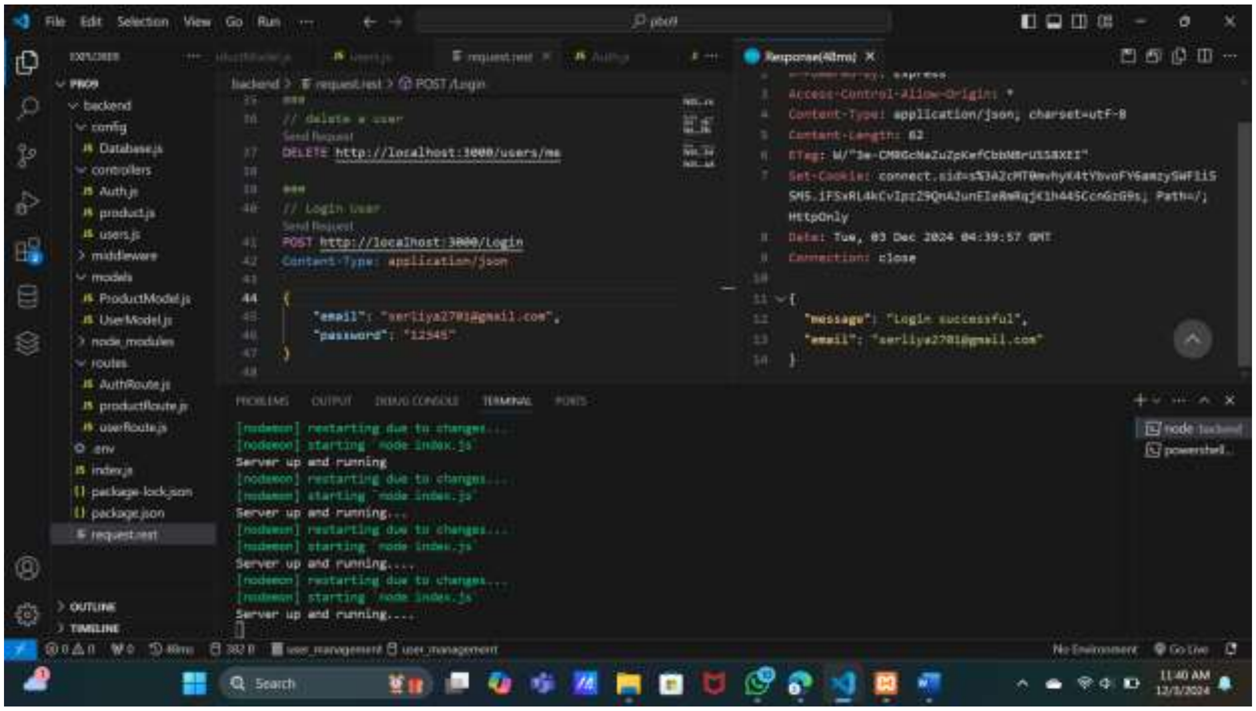
```
Server berhasil..
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Server berhasil..
```



```
PATCH http://localhost:3000/users/se5efg367-9087-x2t8
    "password": "12345",
    "confPassword": "12345",
    "role": "user"
}

###
// delete a user
Send Request
DELETE http://localhost:3000/users/me

###
// Login User
Send Request
POST http://localhost:3000/users/Login
Content-Type: application/json

{
    "email": "serliya2701@gmail.com",
    "password": "12345"
}
```

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 82
ETag: W/"3e-CMRGcNaZuZpKefCbbN8rUSS8XEI"
Date: Mon, 02 Dec 2024 23:30:54 GMT
Connection: close

{
  "message": "Login successful",
  "email": "serliya2701@gmail.com"
}
```

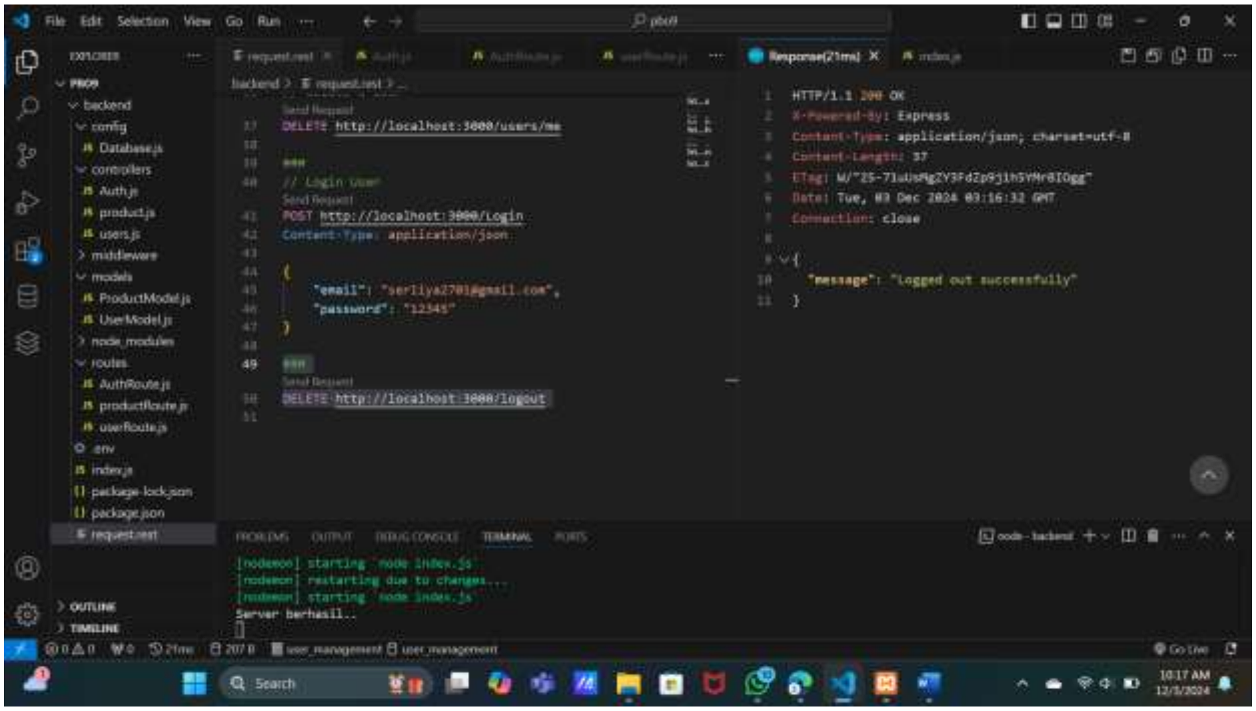
```
Server berhasil..
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Server berhasil..
```



```
###
// delete a user
DELETE http://localhost:3000/users/me

###
// Login User
POST http://localhost:3000/Login
Content-Type: application/json

{
    "email": "serliya2701@gmail.com",
    "password": "12345"
}
```

```
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Content-Length: 82
ETag: W/"3e-CMRGcNaZuZpKefCbbNBrUSS8XEI"
Set-Cookie: connect.sid=s%3A2cMT0mvhyX4tYbvoFY6amzySWF1iS
SM5.1FSxRL4kCvIpz29QnA2unEIeRmRqjK1h44SCcnGrG9s; Path=/;
HttpOnly
Date: Tue, 03 Dec 2024 04:39:57 GMT
Connection: close

{
  "message": "Login successful",
  "email": "serliya2701@gmail.com"
}
```

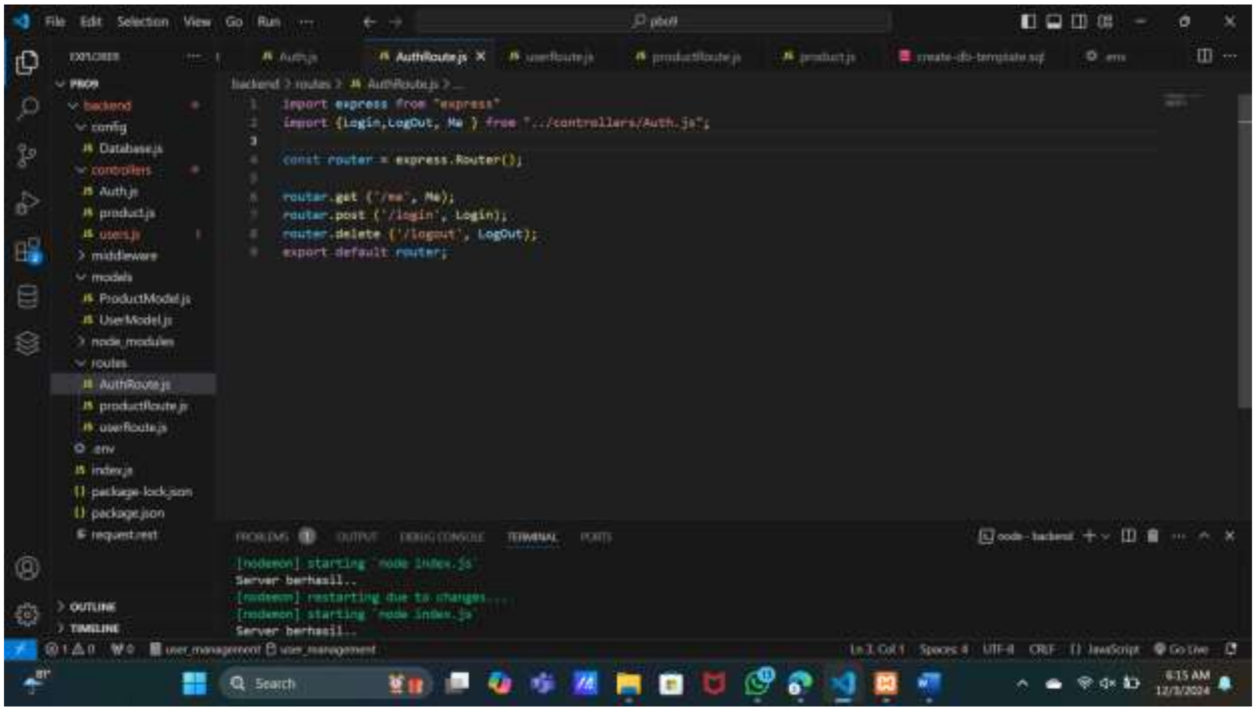
```
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Server up and running
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Server up and running...
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Server up and running....
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Server up and running....
```



```
DELETE http://localhost:3000/users/me

###
// Login User
POST http://localhost:3000/Login
Content-Type: application/json

{
    "email": "serliya2701@gmail.com",
    "password": "12345"
}

###
DELETE http://localhost:3000/logout
```

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 37
ETag: W/"25-7luUsMgZY3FdZp9j1h5YMr8IOgg"
Date: Tue, 03 Dec 2024 03:16:32 GMT
Connection: close

{
  "message": "Logged out successfully"
}
```

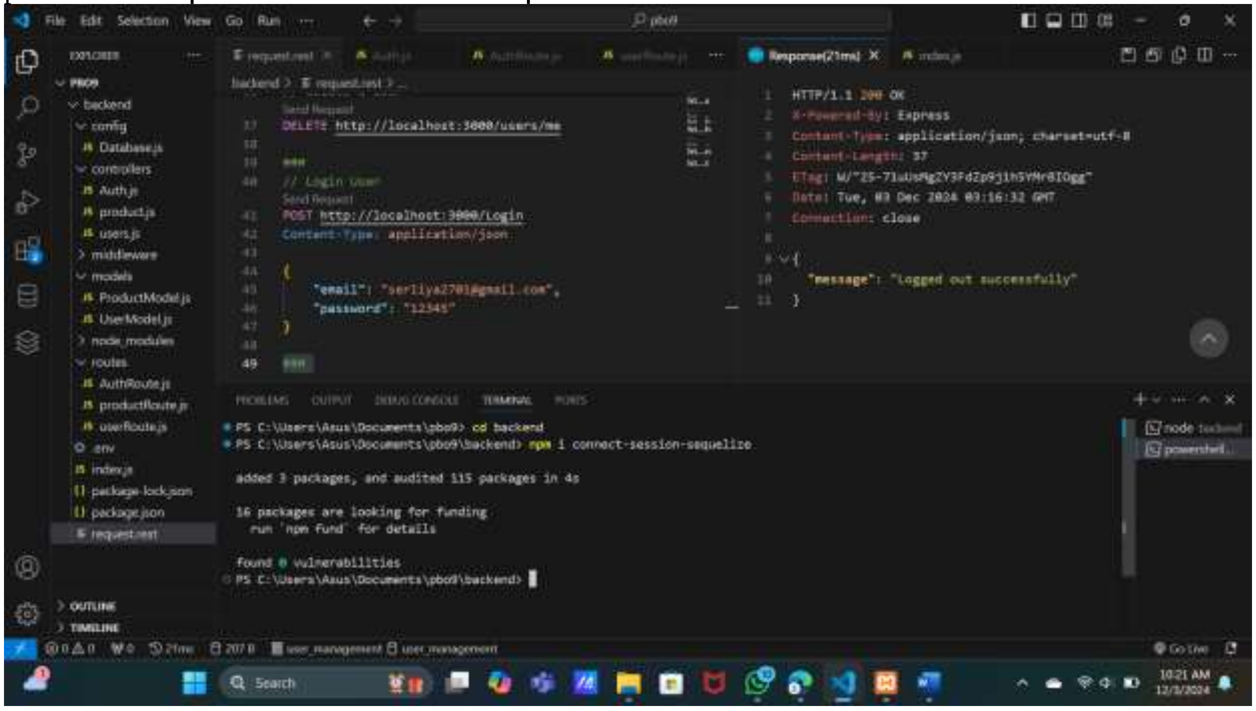
```
[nodemon] starting `node index.js`
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Server berhasil..
```



```
import express from "express"
import {Login,LogOut, Me } from "../controllers/Auth.js";

const router = express.Router();

router.get ('/me', Me);
router.post ('/login', Login);
router.delete ('/logout', LogOut);
export default router;
```

```
[nodemon] starting `node index.js`
Server berhasil..
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Server berhasil..
```

Install npm
pada terminal **npm i connect-session-sequelize**

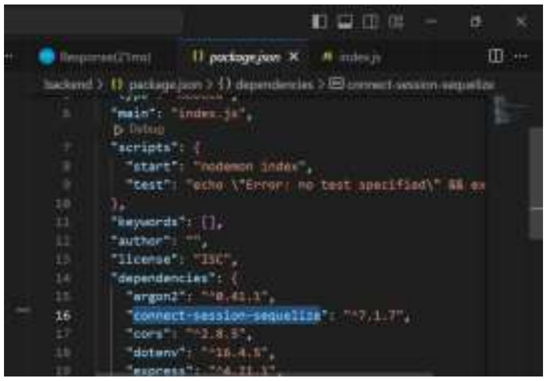


```
Send Request
DELETE http://localhost:3000/users/me

###
// Login User
Send Request
POST http://localhost:3000/Login
Content-Type: application/json

{
    "email": "serliya2701@gmail.com",
    "password": "12345"
}

###
```

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 37
ETag: W/"25-71uUsMgZY3FdZp9j1h5YMr8IOgg"
Date: Tue, 03 Dec 2024 03:16:32 GMT
Connection: close

{
  "message": "Logged out successfully"
}
```

```
PS C:\Users\Asus\Documents\pbo9> cd backend
PS C:\Users\Asus\Documents\pbo9\backend> npm i connect-session-sequelize

added 3 packages, and audited 115 packages in 4s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\Asus\Documents\pbo9\backend> 
```

Jika sudah terinstall tertera pada package.json



```json
"main": "index.js",
"scripts": {
  "start": "nodemon index",
  "test": "echo \"Error: no test specified\" && ex
},
"keywords": [],
"author": "",
"license": "ISC",
"dependencies": {
  "argon2": "^0.41.1",
  "connect-session-sequelize": "^7.1.7",
  "cors": "^2.8.5",
  "dotenv": "^16.4.5",
  "express": "^4.21.1",
```

Berikut merupakan code index.js

```javascript
import express from "express"; // Pastikan Express diimpor dengan benar
import cors from "cors";
import session from "express-session";
import dotenv from 'dotenv';
import db from "/config/Database.js";
import Sequelize from "./connect-session-sequelize";
import session from "express-session";
import UserRoute from "/routes/ProductRoute.js";
import ProductRoute from "/routes/ProductRoute.js";
import AuthRoute from "/routes/ProductRoute.js";

dotenv.config(); // Memuat konfigurasi dari .env jika ada

const app = express();
const sessionStore = Sequelize(session.Store);

const store = new sessionStore({
    db:db
});
const PORT = process.env.APP_PORT || 3000; // Gunakan port dari .env atau default
3000

app.use(session({
    secret: process.env.SESS_SECRET,
    resave: false,
    saveUninitialized: true,
    Store: store,
    cookie: {
        secure: "auto"
    }
});

app.use(cors({
    credentials: true,
    origin: 'http://localhost:3000'
}));
app.use(express.json());
app.use(UserRoute);
app.use(ProductRoute);
app.use(AuthRoute);

store.async();

app.listen(process.env.APP_PORT, ()=> {
    console.log('Server up and Running...');
});
```

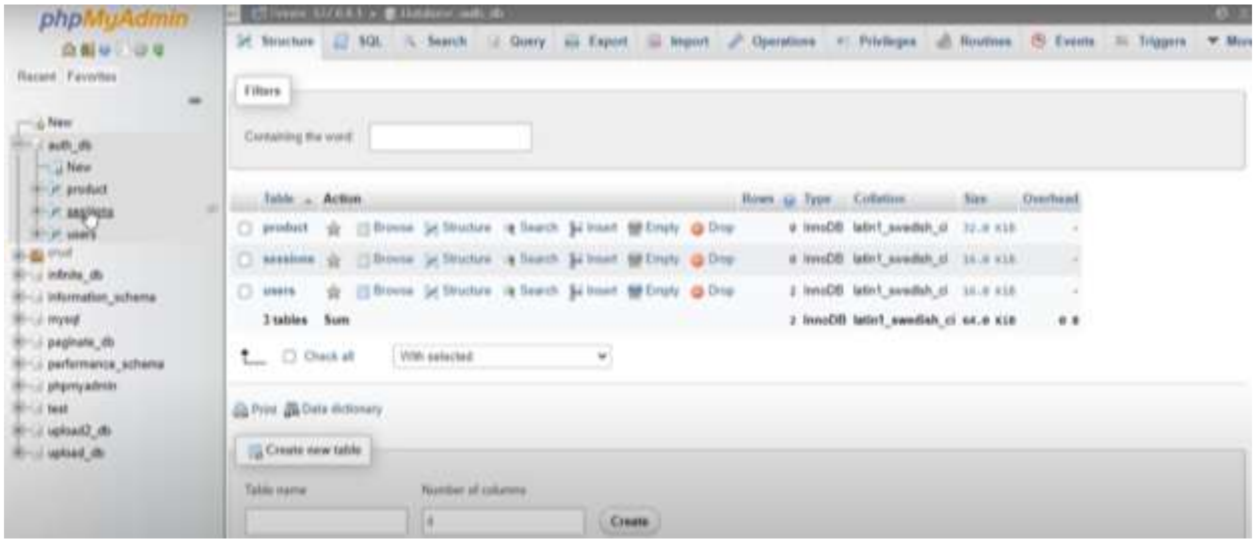


Table	Action							Rows	Type	Collation	Size	Overhead
product	Browse	Structure	Search	Insert	Empty	Drop		0	InnoDB	latin1_swedish_ci	32.0 KiB	-
sessions	Browse	Structure	Search	Insert	Empty	Drop		0	InnoDB	latin1_swedish_ci	16.0 KiB	-
users	Browse	Structure	Search	Insert	Empty	Drop		2	InnoDB	latin1_swedish_ci	16.0 KiB	-
3 tables	Sum							2	InnoDB	latin1_swedish_ci	64.0 KiB	0 B

Buatlah kode authroute.js pada folder route

```
import User from "/models/UserModel.js";

export const verifyUser = async (req, res, next)=>{
if(!req.session.userId){
    return res.status(401).json({msg: "Mohon login ke akun Anda!"});
}
const user = await User.findOne({
    attributes:['uuid', 'name', 'email', 'password', 'role'],
    where: {
        uuid: req.session.userId
    }
});
if(!user) return res.status(404).json({msg: "User tidak ditemukan"});
res.status(200).json(user);
}
```