

LAPORAN PRAKTIKUM
PEMROGRAMAN WEB



RIYADI TRI WALUYA BUDI

2308065

SIK A3

I.PENDAHULUAN

Session adalah mekanisme yang digunakan dalam aplikasi web untuk menyimpan informasi Pengguna secara sementara saat mereka berinteraksi dengan aplikasi tersebut. Dalam konteks Node.js, **session** digunakan untuk melacak dan mengingat status pengguna di antara permintaan HTTP (request) yang berbeda.

Berikut adalah beberapa poin penting tentang Sesi (Session):

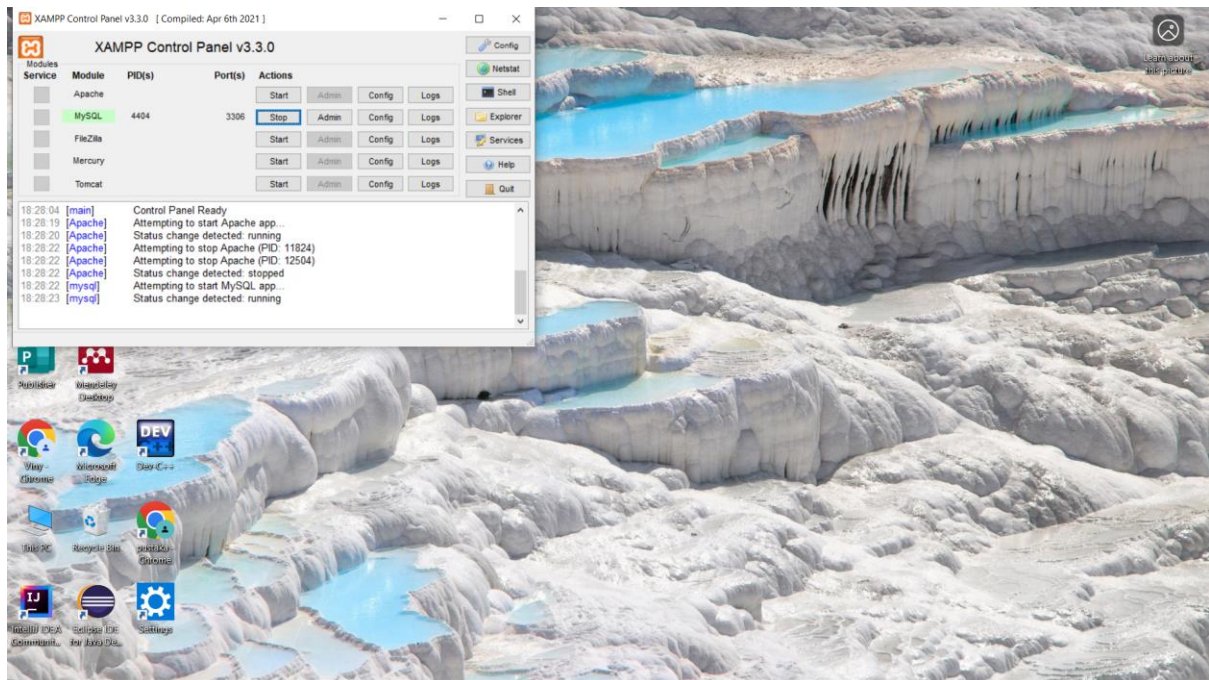
1. Penyimpanan Informasi Sementara: Saat pengguna melakukan login, informasi seperti ID pengguna, username, atau token disimpan dalam sesi, memungkinkan server untuk mengenali pengguna selama sesi tersebut berlangsung.
2. Keterikatan pada Pengguna: Setiap pengguna yang terhubung ke aplikasi memiliki sesi yang unik. Saat pengguna melakukan permintaan baru (misalnya berpindah halaman), sesi yang relevan akan dikenali dan diakses.
3. Berdasarkan Cookie: Sesi biasanya disimpan di server, dan untuk melacak sesi tersebut, browser pengguna menyimpan cookie yang berisi ID sesi. Server akan memeriksa cookie ini untuk mengenali sesi pengguna.
4. Contoh Penggunaan:
 - Ketika pengguna melakukan login, aplikasi menyimpan informasi login dalam sesi.
 - Pada permintaan berikutnya, aplikasi dapat memeriksa sesi untuk memastikan pengguna sudah login.
 - Jika pengguna melakukan logout, sesi akan dihapus, sehingga pengguna harus melakukan login lagi.

II.ALAT DAN BAHAN

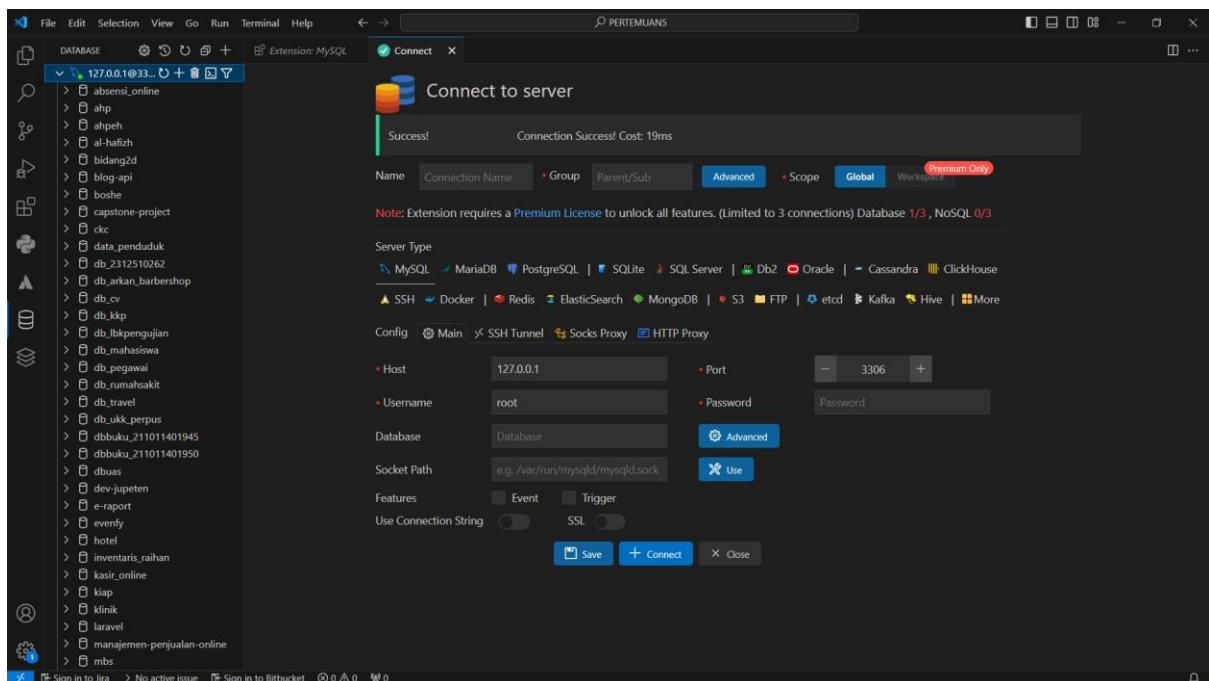
- VScode
- Node.js
- MySQL

III.LANGKAH – LANGKAH

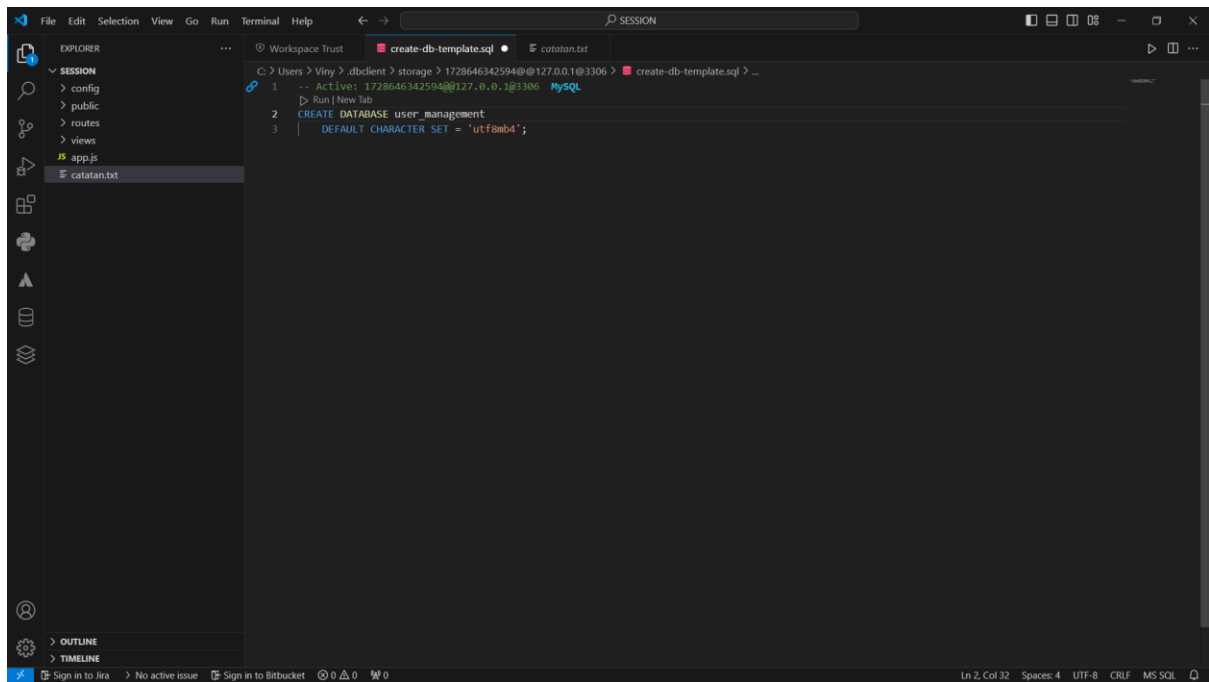
1.Pertama, Buka Aplikasi XAMPP Kemudian Aktifkan MySQL nya



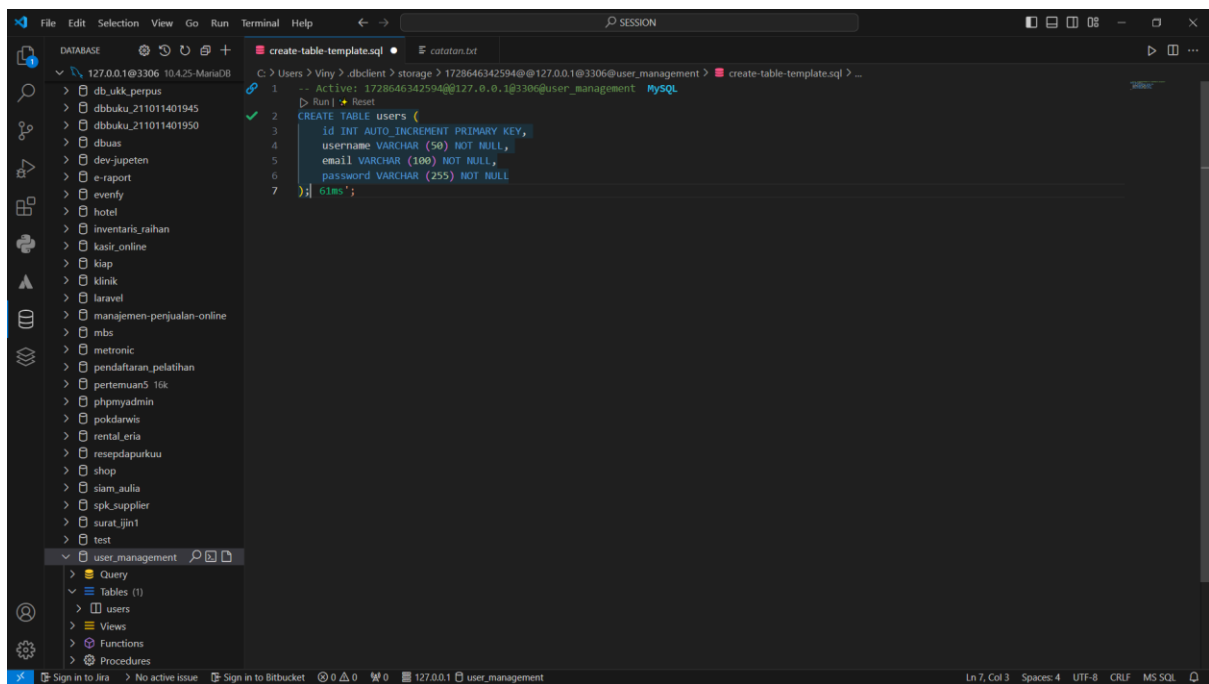
2. Kemudian buka ekstensi MySQL pada VSCode, kemudian connect pada database.



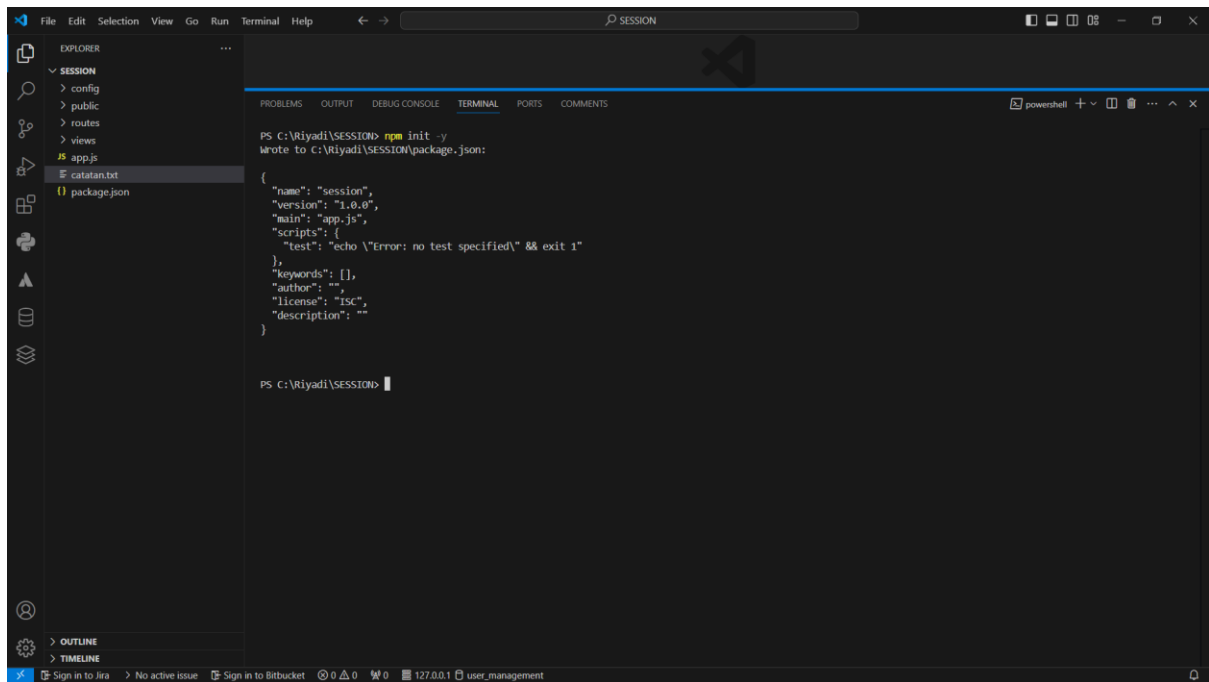
3. Kemudian tambahkan database dengan nama user_management



4. Tambahkan tabel users



5. Buka terminal, dan inisiasi npm untuk node.js

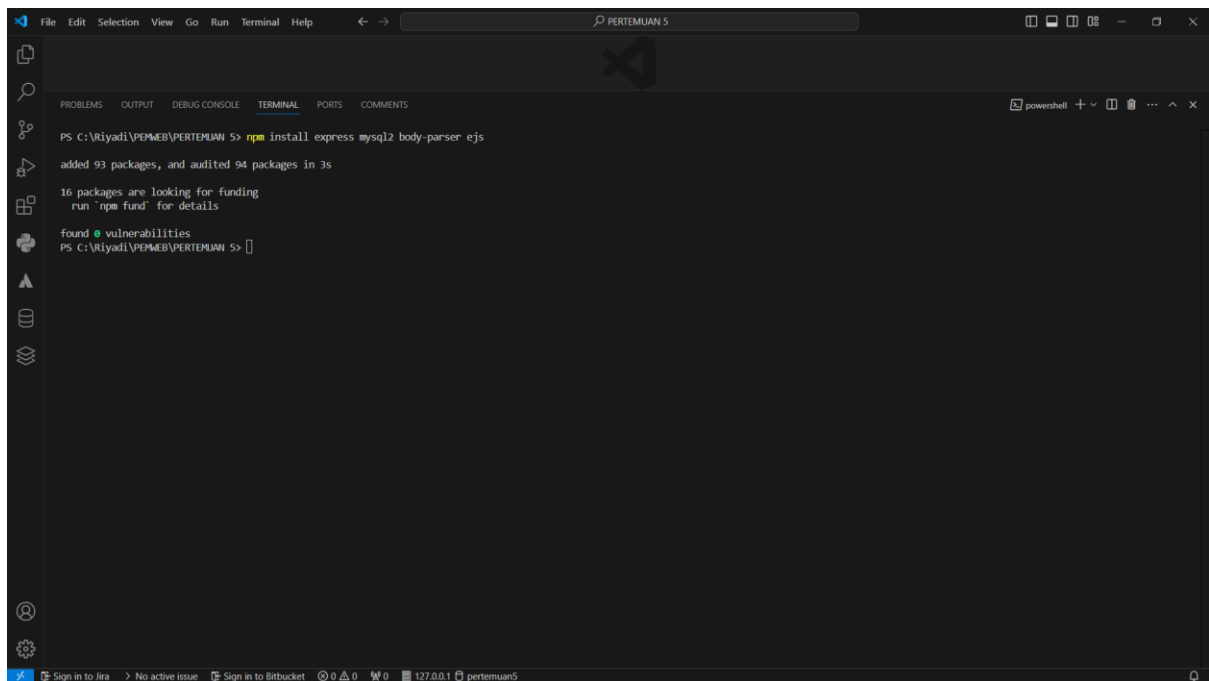


The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal is running a PowerShell session in the directory C:\Riyadi\SESSION. The command `npm init -y` has been executed, and the output shows the creation of a `package.json` file. The `package.json` file is displayed in the Explorer view on the left, showing the following content:

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

The terminal output also shows the command `npm init -y` and the resulting `package.json` file content.

6. Install express.js, mysql, bcryptjs, body-parser, express-session, dan ejs

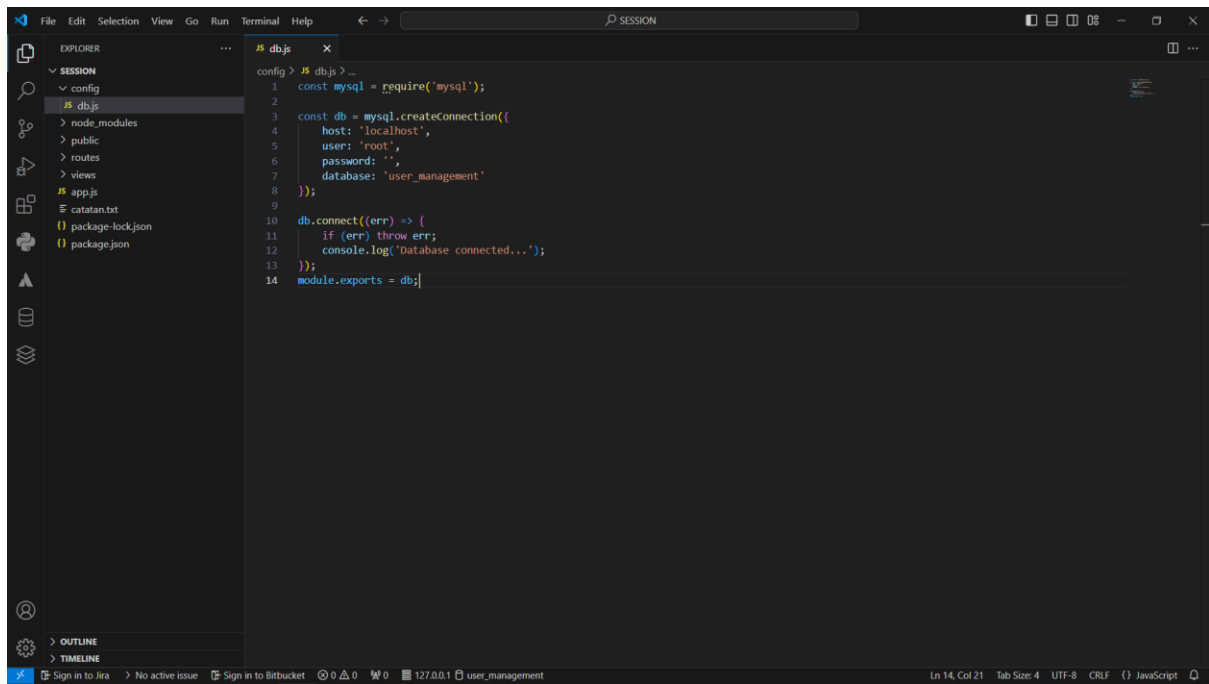


The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal is running a PowerShell session in the directory C:\Riyadi\PEMBER\PERTEMUAN 5. The command `npm install express mysql2 body-parser ejs` has been executed, and the output shows the installation of the packages. The output includes the following information:

- added 93 packages, and audited 94 packages in 3s
- 16 packages are looking for funding
- run `npm fund` for details
- found 0 vulnerabilities

The terminal output also shows the command `npm install express mysql2 body-parser ejs` and the resulting output.

7. Konfigurasi koneksi database

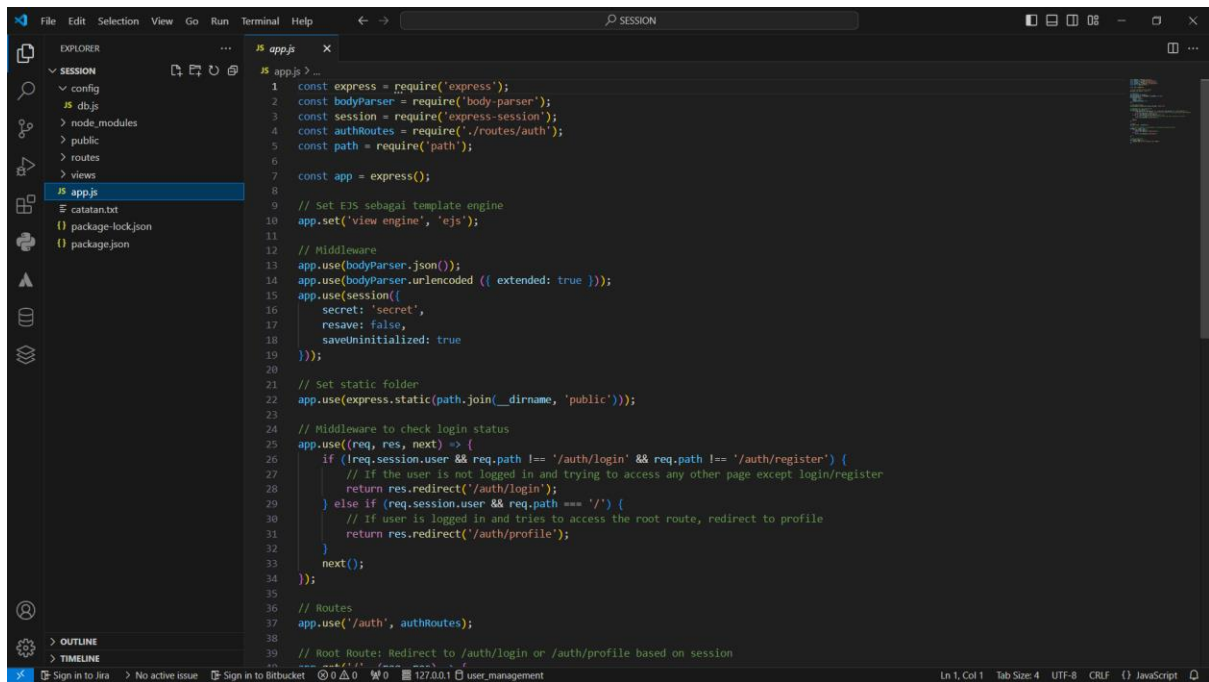


The image shows a Visual Studio Code editor window with a dark theme. The Explorer sidebar on the left displays a project structure with folders like 'config', 'node_modules', 'public', 'routes', and 'views', and files like 'app.js', 'catatan.txt', 'package-lock.json', and 'package.json'. The 'db.js' file in the 'config' folder is selected and open in the main editor. The code in 'db.js' is as follows:

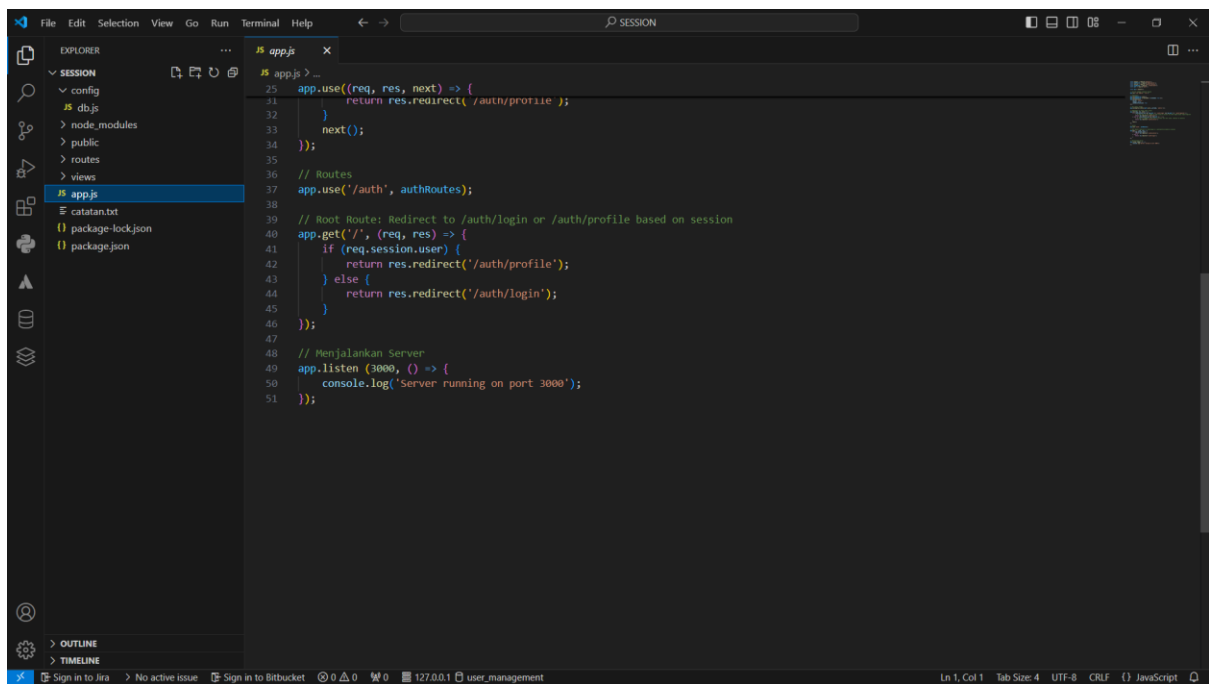
```
1 const mysql = require('mysql');
2
3 const db = mysql.createConnection({
4   host: 'localhost',
5   user: 'root',
6   password: '',
7   database: 'user_management'
8 });
9
10 db.connect((err) => {
11   if (err) throw err;
12   console.log('Database connected...');
13 });
14 module.exports = db;
```

The status bar at the bottom indicates the current file is 'db.js' at line 14, column 21, with a tab size of 4, UTF-8 encoding, CRLF line endings, and JavaScript syntax.

8. Server setup pada app.js

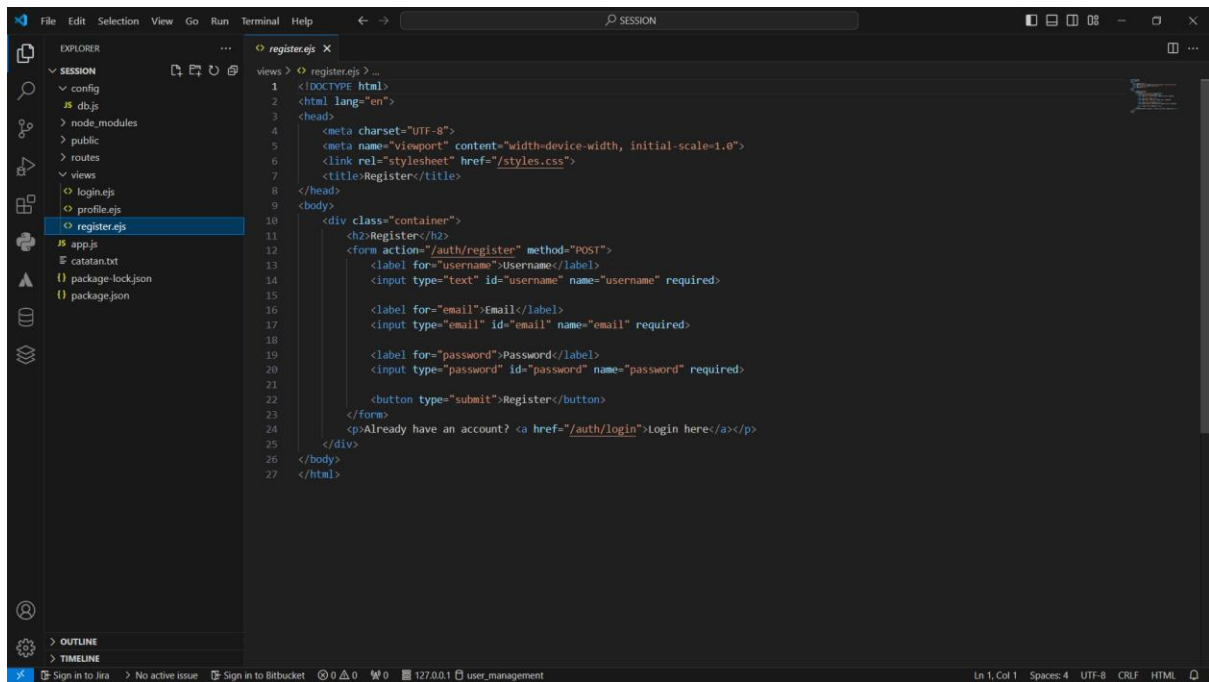


```
1 const express = require('express');
2 const bodyParser = require('body-parser');
3 const session = require('express-session');
4 const authRoutes = require('./routes/auth');
5 const path = require('path');
6
7 const app = express();
8
9 // Set EJS sebagai template engine
10 app.set('view engine', 'ejs');
11
12 // Middleware
13 app.use(bodyParser.json());
14 app.use(bodyParser.urlencoded({ extended: true }));
15 app.use(session({
16   secret: 'secret',
17   resave: false,
18   saveUninitialized: true
19 }));
20
21 // Set static folder
22 app.use(express.static(path.join(__dirname, 'public')));
23
24 // Middleware to check login status
25 app.use((req, res, next) => {
26   if ((req.session.user && req.path !== '/auth/login' && req.path !== '/auth/register')) {
27     // If the user is not logged in and trying to access any other page except login/register
28     return res.redirect('/auth/login');
29   } else if (req.session.user && req.path === '/') {
30     // If user is logged in and tries to access the root route, redirect to profile
31     return res.redirect('/auth/profile');
32   }
33   next();
34 });
35
36 // Routes
37 app.use('/auth', authRoutes);
38
39 // Root Route: Redirect to /auth/login or /auth/profile based on session
```



```
25 app.use((req, res, next) => {
31   return res.redirect('/auth/profile');
32 });
33 next();
34 });
35
36 // Routes
37 app.use('/auth', authRoutes);
38
39 // Root Route: Redirect to /auth/login or /auth/profile based on session
40 app.get('/', (req, res) => {
41   if (req.session.user) {
42     return res.redirect('/auth/profile');
43   } else {
44     return res.redirect('/auth/login');
45   }
46 });
47
48 // Menjalankan Server
49 app.listen(3000, () => {
50   console.log('Server running on port 3000');
51 });
```

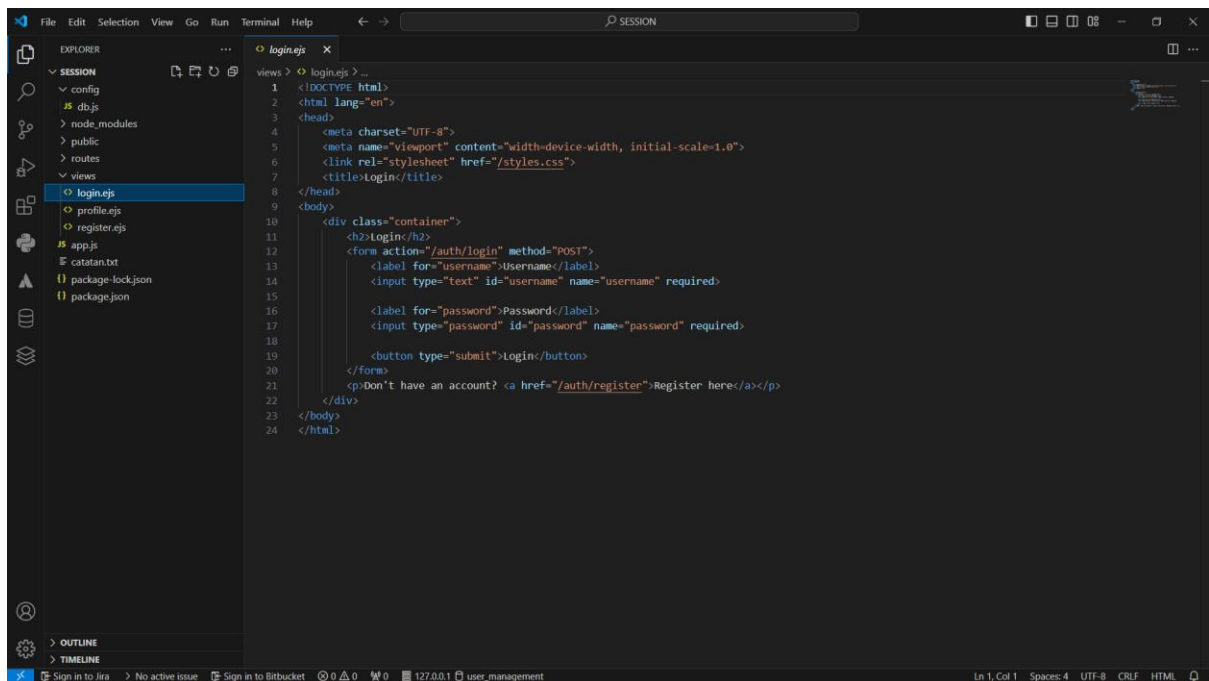
9. Membuat tampilan untuk register



The screenshot shows the Visual Studio Code editor with the 'register.ejs' file open. The Explorer sidebar on the left shows the project structure with 'register.ejs' selected under the 'views' directory. The main editor displays the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="/styles.css">
7   <title>Register</title>
8 </head>
9 <body>
10   <div class="container">
11     <h2>Register</h2>
12     <form action="/auth/register" method="POST">
13       <label for="username">Username</label>
14       <input type="text" id="username" name="username" required>
15
16       <label for="email">Email</label>
17       <input type="email" id="email" name="email" required>
18
19       <label for="password">Password</label>
20       <input type="password" id="password" name="password" required>
21
22       <button type="submit">Register</button>
23     </form>
24     <p>Already have an account? <a href="/auth/login">Login here</a></p>
25   </div>
26 </body>
27 </html>
```

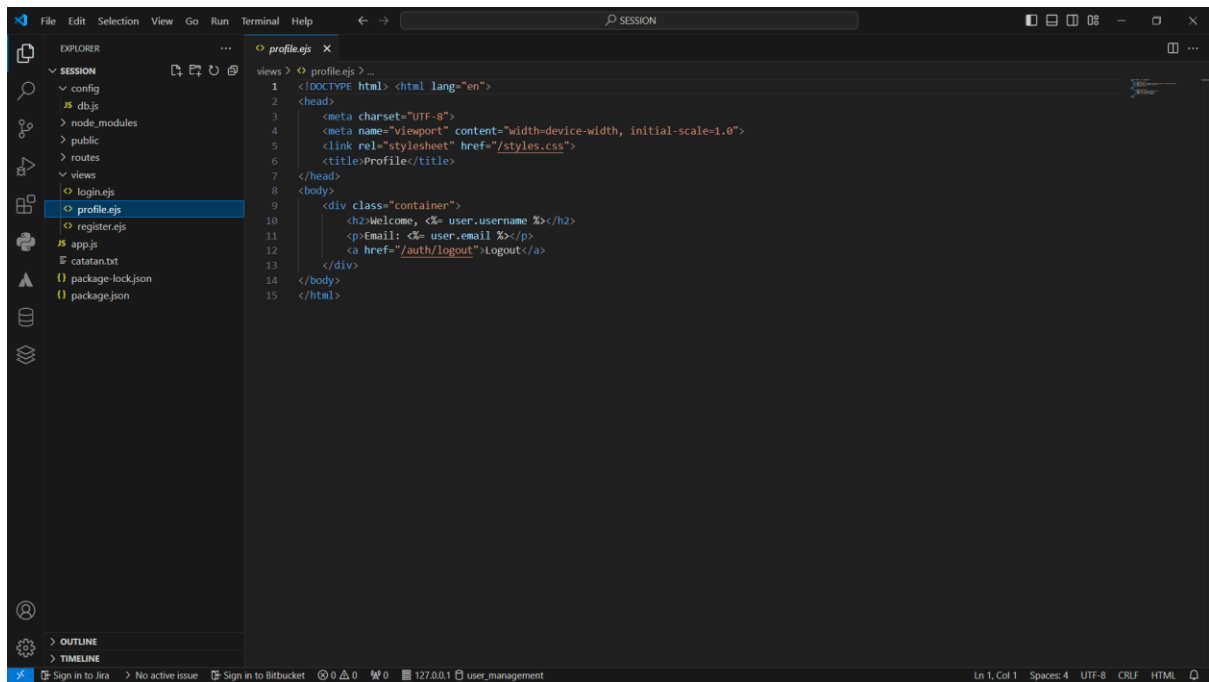
10. Membuat tampilan untuk login



The screenshot shows the Visual Studio Code editor with the 'login.ejs' file open. The Explorer sidebar on the left shows the project structure with 'login.ejs' selected under the 'views' directory. The main editor displays the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="/styles.css">
7   <title>Login</title>
8 </head>
9 <body>
10   <div class="container">
11     <h2>Login</h2>
12     <form action="/auth/login" method="POST">
13       <label for="username">Username</label>
14       <input type="text" id="username" name="username" required>
15
16       <label for="password">Password</label>
17       <input type="password" id="password" name="password" required>
18
19       <button type="submit">Login</button>
20     </form>
21     <p>Don't have an account? <a href="/auth/register">Register here</a></p>
22   </div>
23 </body>
24 </html>
```


11. Membuat tampilan untuk profile



The screenshot shows the Visual Studio Code editor with the file explorer on the left. The file explorer shows the project structure with the following files and folders:

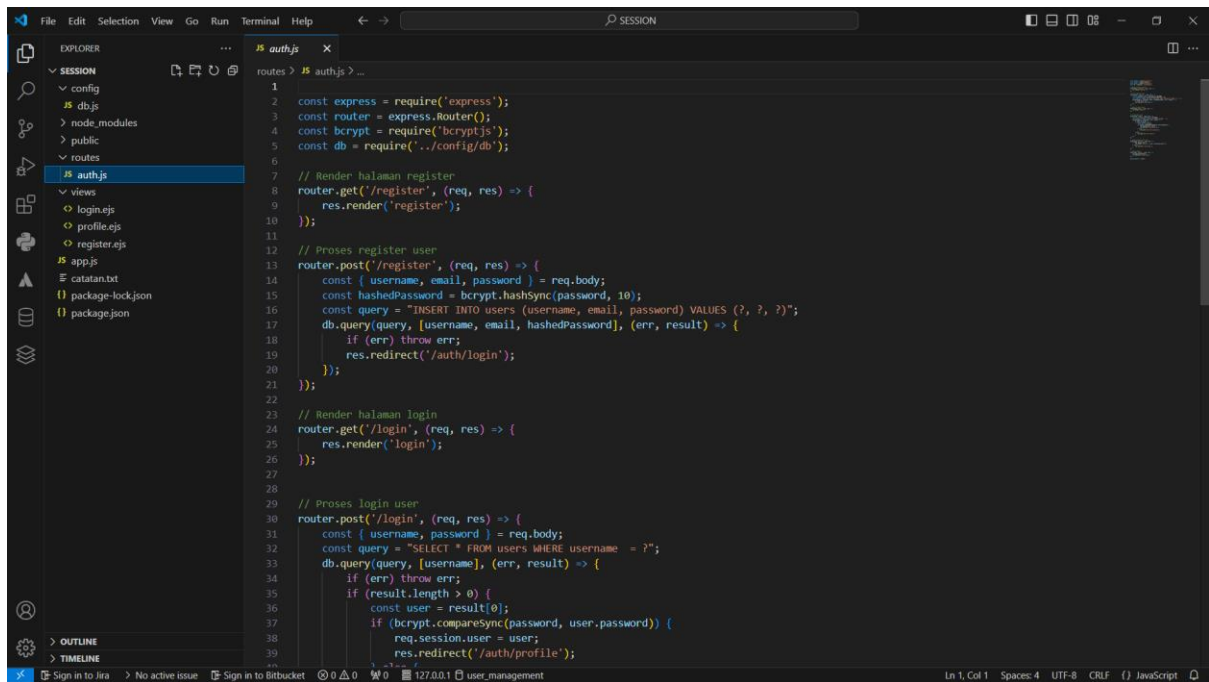
- SESSION
 - config
 - db.js
 - node_modules
 - public
 - routes
 - views
 - login.ejs
 - profile.ejs (selected)
 - register.ejs
 - app.js
 - catatan.txt
 - package-lock.json
 - package.json

The main editor area displays the content of `profile.ejs`:

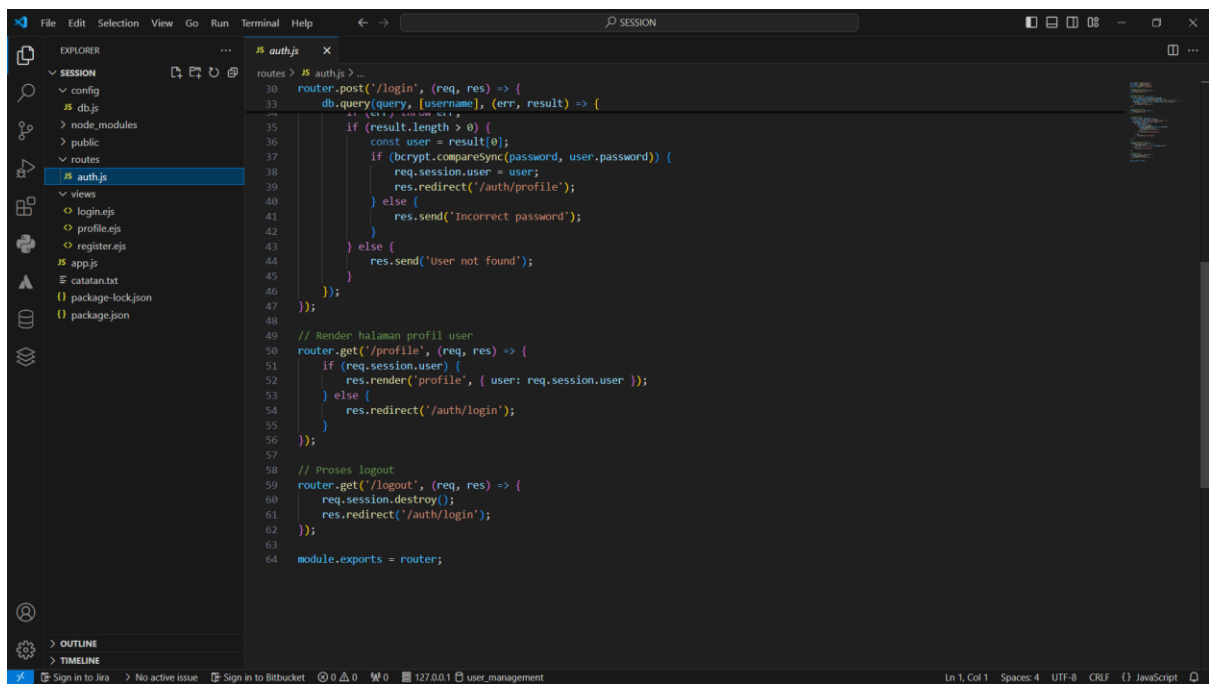
```
1 <!DOCTYPE html> <html lang="en">
2 <head>
3   <meta charset="UTF-8">
4   <meta name="viewport" content="width=device-width, initial-scale=1.0">
5   <link rel="stylesheet" href="/styles.css">
6   <title>Profile</title>
7 </head>
8 <body>
9   <div class="container">
10     <h2>Welcome, <%= user.username %></h2>
11     <p>Email: <%= user.email %></p>
12     <a href="/auth/logout">Logout</a>
13   </div>
14 </body>
15 </html>
```

The status bar at the bottom indicates the current file is `Ln 1, Col 1` with `Spaces: 4`, `UTF-8` encoding, `CRLF` line endings, and `HTML` language.

12. Buat route login, logout, dan register (routes/auth.js)

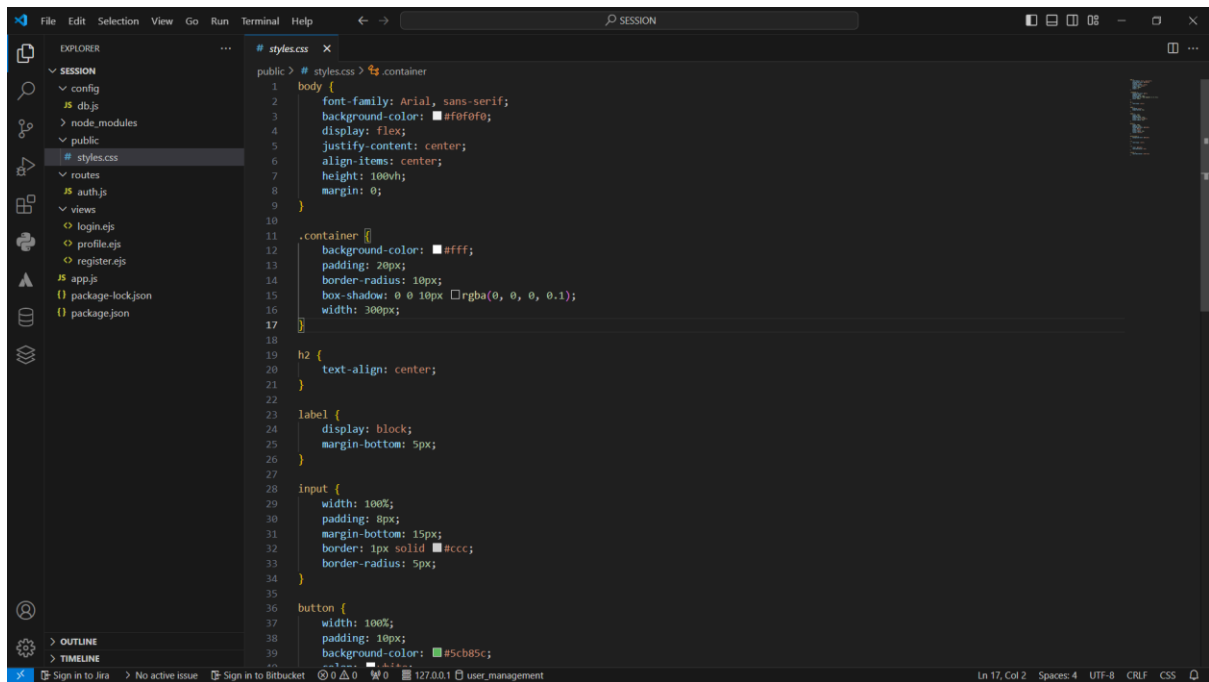


```
1
2 const express = require('express');
3 const router = express.Router();
4 const bcrypt = require('bcryptjs');
5 const db = require('../config/db');
6
7 // Render halaman register
8 router.get('/register', (req, res) => {
9   res.render('register');
10 });
11
12 // Proses register user
13 router.post('/register', (req, res) => {
14   const { username, email, password } = req.body;
15   const hashedPassword = bcrypt.hashSync(password, 10);
16   const query = "INSERT INTO users (username, email, password) VALUES (?, ?, ?)";
17   db.query(query, [username, email, hashedPassword], (err, result) => {
18     if (err) throw err;
19     res.redirect('/auth/login');
20   });
21 });
22
23 // Render halaman login
24 router.get('/login', (req, res) => {
25   res.render('login');
26 });
27
28 // Proses login user
29 router.post('/login', (req, res) => {
30   const { username, password } = req.body;
31   const query = "SELECT * FROM users WHERE username = ?";
32   db.query(query, [username], (err, result) => {
33     if (err) throw err;
34     if (result.length > 0) {
35       const user = result[0];
36       if (bcrypt.compareSync(password, user.password)) {
37         req.session.user = user;
38         res.redirect('/auth/profile');
39       } else {
40         res.send('Incorrect password');
41       }
42     } else {
43       res.send('User not found');
44     }
45   });
46 });
47
48 // Render halaman profil user
49 router.get('/profile', (req, res) => {
50   if (req.session.user) {
51     res.render('profile', { user: req.session.user });
52   } else {
53     res.redirect('/auth/login');
54   }
55 });
56
57 // Proses logout
58 router.get('/logout', (req, res) => {
59   req.session.destroy();
60   res.redirect('/auth/login');
61 });
62
63 module.exports = router;
```

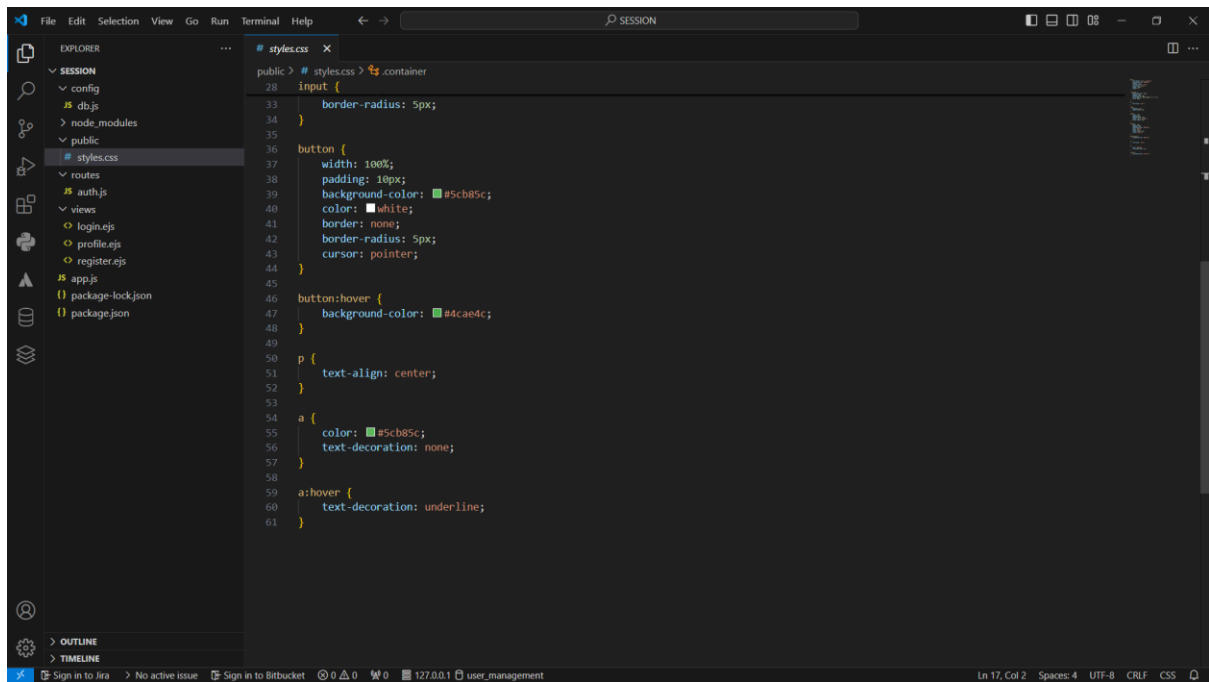


```
30 router.post('/login', (req, res) => {
31   db.query(query, [username], (err, result) => {
32     if (err) throw err;
33     if (result.length > 0) {
34       const user = result[0];
35       if (bcrypt.compareSync(password, user.password)) {
36         req.session.user = user;
37         res.redirect('/auth/profile');
38       } else {
39         res.send('Incorrect password');
40       }
41     } else {
42       res.send('User not found');
43     }
44   });
45 });
46
47 // Render halaman profil user
48 router.get('/profile', (req, res) => {
49   if (req.session.user) {
50     res.render('profile', { user: req.session.user });
51   } else {
52     res.redirect('/auth/login');
53   }
54 });
55
56 // Proses logout
57 router.get('/logout', (req, res) => {
58   req.session.destroy();
59   res.redirect('/auth/login');
60 });
61
62 module.exports = router;
```

13. Tambahkan CSSnya agar lebih menarik



```
# styles.css X
public > # styles.css > container
1 body {
2   font-family: Arial, sans-serif;
3   background-color: #f0f0f0;
4   display: flex;
5   justify-content: center;
6   align-items: center;
7   height: 100vh;
8   margin: 0;
9 }
10
11 .container {
12   background-color: #fff;
13   padding: 20px;
14   border-radius: 10px;
15   box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
16   width: 300px;
17 }
18
19 h2 {
20   text-align: center;
21 }
22
23 label {
24   display: block;
25   margin-bottom: 5px;
26 }
27
28 input {
29   width: 100%;
30   padding: 8px;
31   margin-bottom: 15px;
32   border: 1px solid #ccc;
33   border-radius: 5px;
34 }
35
36 button {
37   width: 100%;
38   padding: 10px;
39   background-color: #5cb85c;
40 }
```



```
# styles.css X
public > # styles.css > container
28 input {
29   width: 100%;
30   padding: 8px;
31   margin-bottom: 15px;
32   border: 1px solid #ccc;
33   border-radius: 5px;
34 }
35
36 button {
37   width: 100%;
38   padding: 10px;
39   background-color: #5cb85c;
40   color: #fff;
41   border: none;
42   border-radius: 5px;
43   cursor: pointer;
44 }
45
46 button:hover {
47   background-color: #4cae4c;
48 }
49
50 p {
51   text-align: center;
52 }
53
54 a {
55   color: #5cb85c;
56   text-decoration: none;
57 }
58
59 a:hover {
60   text-decoration: underline;
61 }
```

14. jalankan dengan node app.js dan buka pada localhost:3000

Browser: Login
URL: localhost:3000/auth/login

Login

Username

Password

[Login](#)

Don't have an account? [Register here](#)

15. Register akun terlebih dahulu

Browser: Register
URL: localhost:3000/auth/register

Register

Username

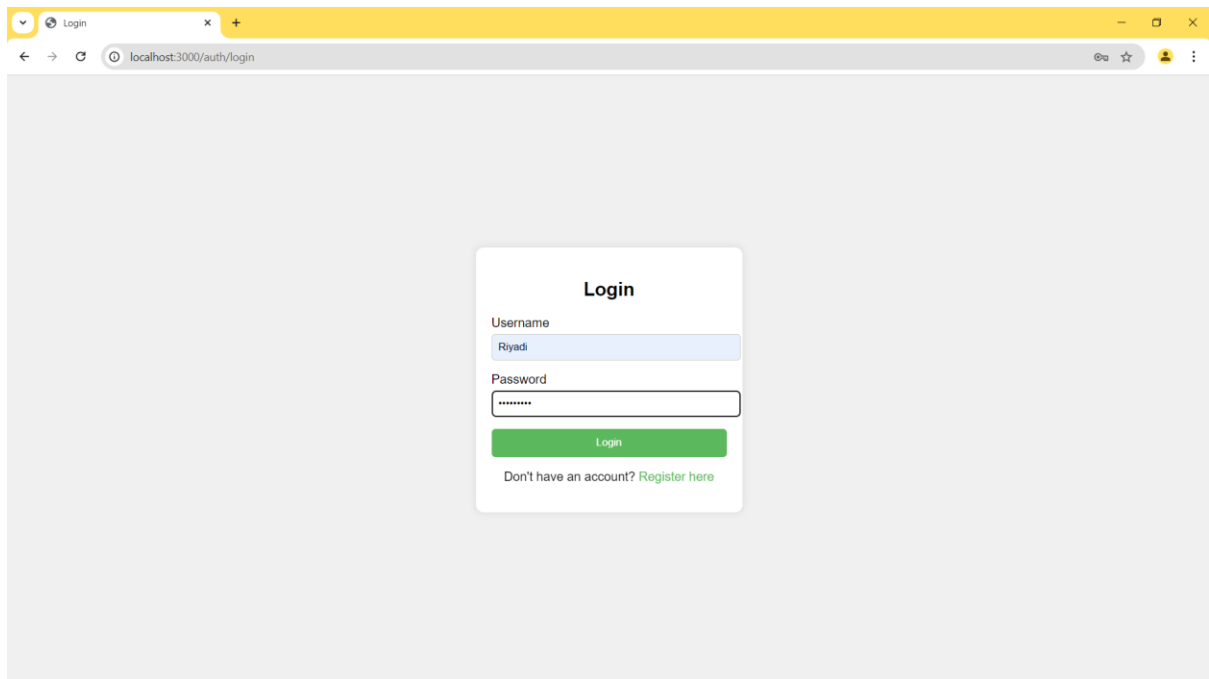
Email

Password

[Register](#)

Already have an account? [Login here](#)

16. Coba login dengan akun yang telah terdaftar



A screenshot of a web browser window with a yellow header bar. The browser's address bar shows 'localhost:3000/auth/login'. The page content is a light gray background with a white login form in the center. The form has a title 'Login', a 'Username' field with the value 'Riyadi', a 'Password' field with masked characters, a green 'Login' button, and a link 'Don't have an account? Register here'.

Browser: Login
URL: localhost:3000/auth/login

Login

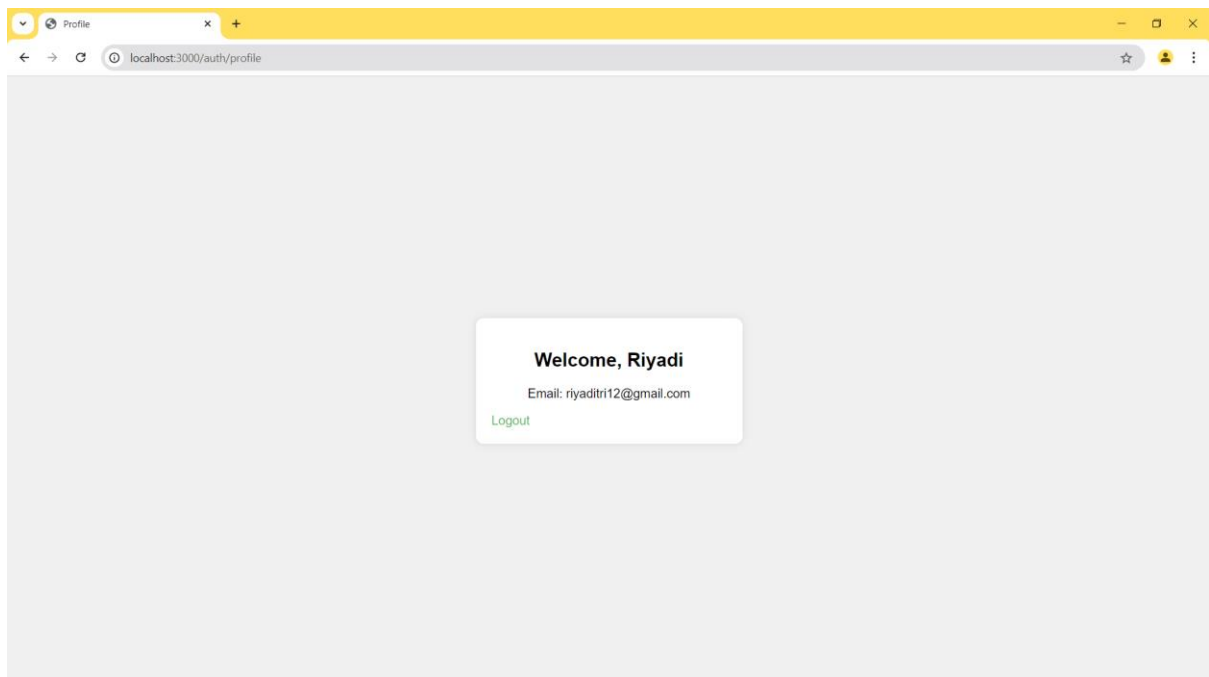
Username
Riyadi

Password

Login

Don't have an account? [Register here](#)

17. Berikut tampilan setelah login



A screenshot of a web browser window with a yellow header bar. The browser's address bar shows 'localhost:3000/auth/profile'. The page content is a light gray background with a white profile card in the center. The card displays 'Welcome, Riyadi', the email 'Email: riyaditri12@gmail.com', and a green 'Logout' link.

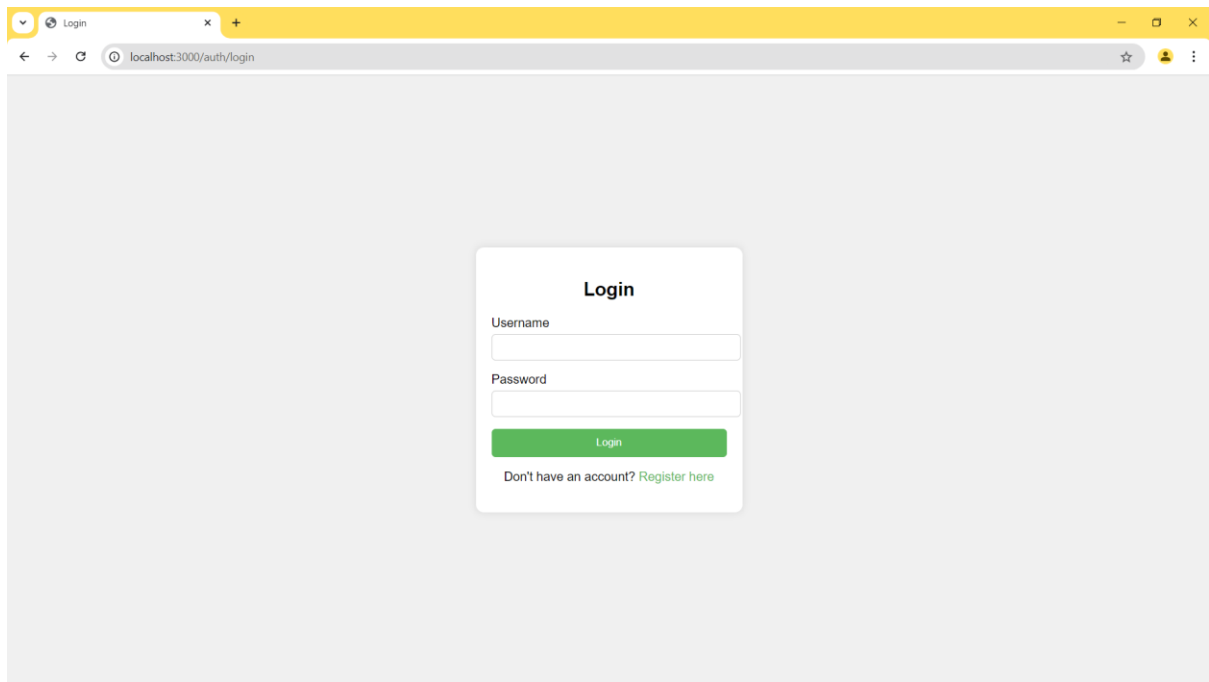
Browser: Profile
URL: localhost:3000/auth/profile

Welcome, Riyadi

Email: riyaditri12@gmail.com

[Logout](#)

18. Berikut tampilan setelah logout



The screenshot shows a web browser window with a yellow title bar. The address bar displays "localhost:3000/auth/login". The page content is a simple login form centered on a light gray background. The form has a white background and a thin gray border. It contains the following elements:

- Login**: A bold black heading at the top of the form.
- Username**: A text label above a white input field.
- Password**: A text label above a white input field.
- Login**: A green button with white text.
- Don't have an account? [Register here](#)**: A line of text with a green link.

PENJELASAN :

Session dapat menyimpan data yang diperlukan dan akan tersimpan dalam periode tertentu pada server sehingga informasi yang dibutuhkan tetap dapat diakses selama waktu tertentu. Hal ini dibutuhkan untuk login agar saat close tab atau close browser, data tetap dapat diakses. Juga saat kita berpindah halaman atau refresh juga tetap dapat mengakses data yang telah disimpan.