

LABORATORIUM TEKNIK INFORMATIKA



PRAKTIKUM PEMROGRAMAN WEB

MANUAL BOOK

“LOGIN & CRUD”

Nama	:	Vierza Vandifa
NPM	:	51420267
Kelas	:	3IA16
Fakultas	:	Teknologi Industri
Jurusan	:	Informatika
Ketua Asisten	:	Robby Nugraha
Jumlah Lembar	:	19

UNIVERSITAS GUNADARMA

2023

MEMBUAT FRONTEND

- INDEX.JS

```
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import App from './App';
4 import "bulma/css/bulma.css";
5 import axios from "axios";
6 axios.defaults.withCredentials = true;
7
8 ReactDOM.render(
9   <React.StrictMode>
10     <App />
11   </React.StrictMode>,
12   document.getElementById('root')
13 );
```

Syntax diatas merupakan index.js yang berfungsi untuk mengimport react, react-dom, app, bulma dan axios. Syntax diatas akan merender app.js.

- APP.JS

```
1 import { BrowserRouter, Route, Switch } from "react-router-dom";
2 import AddUser from "./components/AddUser";
3 import EditUser from "./components/EditUser";
4 import Dashboard from "./components/Dashboard";
5 import Login from "./components/Login";
6 import Register from "./components/Register";
7
8
9 function App() {
10   return (
11     <BrowserRouter>
12       <Switch>
13         <Route exact path="/">
14           <Login/>
15         </Route>
16         <Route path="/register">
17           <Register/>
18         </Route>
19         <Route path="/dashboard">
20           <Dashboard/>
21         </Route>
22         <Route path="/add">
23           <AddUser/>
24         </Route>
25         <Route path="/edit/:id">
26           <EditUser/>
27         </Route>
28       </Switch>
29     </BrowserRouter>
30   );
31 }
32
33 export default App;
```

Syntax diatas berfungsi untuk mengatur route. Untuk main page akan menampilkan login.js. Untuk /register akan menampilkan register.js. Untuk /dashboard

akan menampilkan dashboard.js. Untuk /add akan menampilkan adduser.js. Untuk /edit/:id akan menampilkan edituser.js

- **ADDUSER.JS**

```

1 import React, { useState } from 'react'
2 import axios from 'axios';
3 import { useHistory } from "react-router-dom";
4
5 const Register = () => {
6   const [name, setName] = useState('');
7   const [email, setEmail] = useState('');
8   const [gender, setGender] = useState('Male');
9   const [password, setPassword] = useState('');
10  const [confPassword, setConfPassword] = useState('');
11  const [msg, setMsg] = useState('');
12  const history = useHistory();
13
14  const Register = async (e) => {
15    e.preventDefault();
16    try {
17      await axios.post('http://localhost:5000/users', {
18        name: name,
19        email: email,
20        gender: gender,
21        password: password,
22        confPassword: confPassword
23      });
24      history.push("/dash-board");
25    } catch (error) {
26      if (error.response) {
27        setMsg(error.response.data.msg);
28      }
29    }
30  }
31
32  return (
33    /
34    CREATED BY :
35
36    <div class="hero">
37      <div class="hero-body">
38        <div class="container">
39          <div class="columns is-centered">
40            <div class="column is-4-desktop">
41              <form onSubmit={Register} class="box">
42                <div class="has-text-centered"><h3>Sign Up</h3></div>
43                <div class="field mt-5">
44                  <label class="label">Name</label>
45                  <input type="text" class="input" placeholder="Name"
46                    value={name} onChange={e => setName(e.target.value)} />
47                </div>
48                <div class="field mt-5">
49                  <label class="label">Email</label>
50                  <input type="text" class="input" placeholder="Email" value={email} onChange={e => setEmail(e.target.value)} />
51                </div>
52                <div class="field">
53                  <label class="label">Gender</label>
54                  <div class="control">
55                    <select
56                      value={gender} => setGender(e.target.value)
57                      >
58                      <option value="Male">Male</option>
59                      <option value="Female">Female</option>
60                    </select>
61                  </div>
62                </div>
63                <div class="field mt-5">
64                  <label class="label">Password</label>
65                  <input type="password" class="input" placeholder="*****" value={password} onChange={e => setPassword(e.target.value)} />
66                </div>
67                <div class="field mt-5">
68                  <label class="label">Confirm Password</label>
69                  <div class="control">
70                    <input type="password" class="input" placeholder="*****" value={confPassword} onChange={e => setConfPassword(e.target.value)} />
71                  </div>
72                </div>
73                <div class="field mt-5">
74                  <button class="button is-success is-fullwidth">Add User</button>
75                </div>
76              </form>
77            </div>
78          </div>
79        </div>
80      </div>
81    </div>
82  )
83
84  export default Register
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101

```

Syntax diatas berfungsi untuk melakukan penambahan user pada tampilan /adduser. Syntax diatas akan menyimpan nama, email, gender, dan password. Jika prosedur penyimpanan berhasil maka user akan dilempar ke tampilan /dashboard.

- DASHBOARD.JS

```

1  // eslint-disable react-hooks/exhaustive-deps
2  import React, { useState, useEffect } from 'react'
3  import axios from 'axios'
4  import jwt_decode from 'jwt-decode'
5  import { useHistory } from 'react-router-dom'
6  import { Link } from 'react-router-dom'
7
8  const Dashboard = () => {
9    const [name, setName] = useState('')
10   const [token, setToken] = useState('')
11   const [expire, setExpire] = useState('')
12   const [users, setUsers] = useState([])
13   const history = useHistory()
14
15   useEffect(() => {
16     refreshToken()
17     getUsers()
18   }, [])
19
20   const refreshToken = async () => {
21     try {
22       const response = await axios.get('http://localhost:5000/token')
23       setToken(response.data.accessToken)
24       const decoded = jwt_decode(response.data.accessToken)
25       setName(decoded.name)
26       setExpire(decoded.exp)
27     } catch (error) {
28       if (error.response) {
29         history.push("/")
30       }
31     }
32   }
33
34   const getUsers = async () => {
35     const response = await axios.get('http://localhost:5000/users', {
36       headers: {
37         Authorization: 'Bearer ' + token
38       }
39     })
40     setUsers(response.data)
41   }
42
43   const deleteUser = async (id) => {
44     try {
45       await axios.delete('http://localhost:5000/users/' + id)
46       getUsers()
47     } catch (error) {
48       console.log(error)
49     }
50   }
51
52   const axiosJWT = axios.create()
53
54   axiosJWT.interceptors.request.use(async (config) => {
55     const currentDate = new Date()
56     if (expire + 1000 < currentDate.getTime()) {
57       const response = await axios.get('http://localhost:5000/token')
58       config.headers.Authorization = 'Bearer ' + response.data.accessToken
59       setToken(response.data.accessToken)
60       const decoded = jwt_decode(response.data.accessToken)
61       setName(decoded.name)
62       setExpire(decoded.exp)
63     }
64     return config
65   }, (error) => {
66     return Promise.reject(error)
67   })
68
69   return (
70     //
71     CREATED BY :
72     
73     //
74     <div className="container mt-5">
75       <h1>Welcome Back: {name}</h1>
76       <div>
77         <Link className="add" to="/add" /> Add New
78       </div>
79       <table className="table is-striped is-fullwidth">
80         <thead>
81           <tr>
82             <th>No</th>
83             <th>Name</th>
84             <th>Email</th>
85             <th>Gender</th>
86             <th>Actions</th>
87           </tr>
88         </thead>
89         <tbody>
90           {users.map((user, index) => (
91             <tr key={user.id}>
92               <td>{index + 1}</td>
93               <td>{user.name}</td>
94               <td>{user.email}</td>
95               <td>{user.gender}</td>
96               <div>
97                 <Link
98                   className="edit"
99                   to={`/edit/${user.id}`}
100                 > Edit
101               </Link>
102               <button
103                 className="button is-small is-info mr-2"
104                 onClick={() => deleteUser(user.id)}
105               > Delete
106             </button>
107           </div>
108           </td>
109         </tr>
110       </tbody>
111     </table>
112   </div>
113 )
114 }
115
116 export default Dashboard

```

Syntax diatas berfungsi untuk menampilkan /dashboard. Syntax diatas juga berfungsi untuk mengatur nama user yang sudah login, token, expire, keadaan users

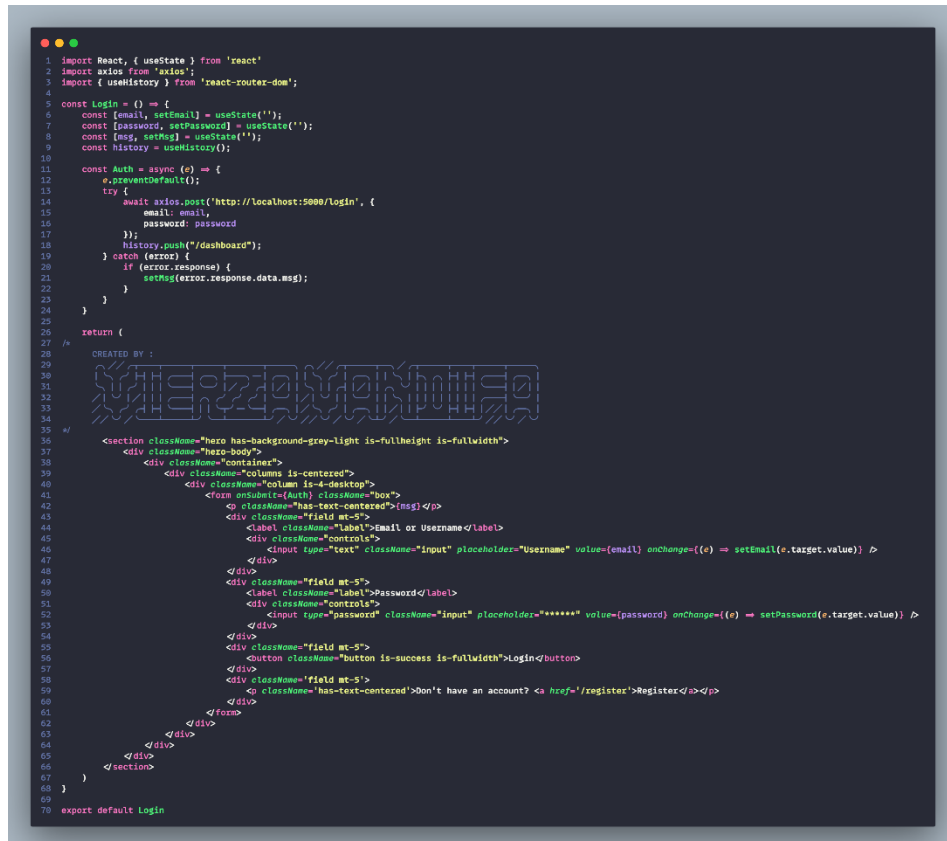
dan penghapusan user. Pada syntax ini juga menampilkan data – data user yang sudah register. User yang telah login bisa menambahkan, mengedit dan menghapus user yang sudah teregister.

- **EDITUSER.JS**

```
1 import React, { useState, useEffect } from "react";
2 import axios from "axios";
3 import { useHistory, useParams } from "react-router-dom";
4
5 const EditUser = () => {
6   const [name, setName] = useState("");
7   const [email, setEmail] = useState("");
8   const [gender, setGender] = useState("Male");
9   const navigate = useHistory();
10  const { id } = useParams();
11
12  useEffect(() => {
13    // eslint-disable-next-line
14    getUserById();
15    // eslint-disable-next-line
16  }, []);
17
18  const updateUser = async (e) => {
19    e.preventDefault();
20    try {
21      await axios.patch('http://localhost:5000/users/${id}', {
22        name,
23        email,
24        gender,
25      });
26      navigate.push("/dashboard");
27    } catch (error) {
28      console.log(error);
29    }
30  };
31
32  const getUserById = async () => {
33    const response = await axios.get('http://localhost:5000/users/${id}');
34    setName(response.data.name);
35    setEmail(response.data.email);
36    setGender(response.data.gender);
37  };
38
39  return (
40    /*
41     * CREATED BY :
42     *
43     *
44     *
45     *
46     *
47     *
48     */
49    <div className="columns mt-5 is-centered">
50      <div className="column is-half">
51        <form onSubmit={updateUser}>
52          <div className="field">
53            <label className="label">Name</label>
54            <div className="control">
55              <input
56                type="text"
57                className="input"
58                value={name}
59                onChange={(e) => setName(e.target.value)}
60                placeholder="Name"
61              />
62            </div>
63          </div>
64          <div className="field">
65            <label className="label">Email</label>
66            <div className="control">
67              <input
68                type="text"
69                className="input"
70                value={email}
71                onChange={(e) => setEmail(e.target.value)}
72                placeholder="Email"
73              />
74            </div>
75          </div>
76          <div className="field">
77            <label className="label">Gender</label>
78            <div className="control">
79              <div className="select is-fullwidth">
80                <select
81                  value={gender}
82                  onChange={(e) => setGender(e.target.value)}>
83                  <option value="Male">Male</option>
84                  <option value="Female">Female</option>
85                </select>
86              </div>
87            </div>
88          </div>
89          <div className="field">
90            <button type="submit" className="button is-success">
91              Update
92            </button>
93          </div>
94        </form>
95      </div>
96    </div>
97  );
98 };
99
100 export default EditUser;
101
```

Syntax diatas berfungsi untuk menampilkan /edituser yang berisi 3 kolom seperti nama, email dan gender yang dimana kolom control yang berisi male dan female. Syntax ini mengatur nama, email, gender. Pada syntax ini user dapat mengedit user yang sudah register, tetapi hanya bisa mengedit nama email dan gender.

- **LOGIN.JS**



Syntax diatas berfungsi untuk menampilkan /login yang terdapat 2 kolom email dan password dan juga href untuk register. Syntax ini mengatur email, password dan message. Syntax ini akan melakukan autentifikasi ke database jika sesuai dengan data yang terdapat pada database maka user akan dikirim ke /dashboard jika tidak ada maka akan menampilkan error response.

- **REGISTER.JS**

Syntax diatas berfungsi untuk menampilkan /register yang berisi 5 kolom nama, email, gender, password, dan confirm password. Syntax diatas mengatur nama, email, gender, password, dan confirm password. Disini user setelah memasukkan nama, email, gender, dan password. Akan dimasukkan kedalam database jika success maka user akan dipindahkan ke menu utama atau /.

MEMBUAT BACKEND

- DATABASE.JS



```
1 import {Sequelize} from "sequelize";
2
3 const db = new Sequelize('vierzaujian_db','root','123456',{
4   host: "localhost",
5   dialect: "mysql"
6 });
7
8 export default db;
```

Syntax berfungsi untuk mengimport database yang Bernama vierzaujian_db dengan username root dan password 123456 pada localhost.

- REFRESHTOKEN.JS



```
1 import Users from "../models/UserModel.js";
2 import jwt from "jsonwebtoken";
3
4 export const refreshToken = async(req, res) => {
5   try {
6     const refreshToken = req.cookies.refreshToken;
7     if(!refreshToken) return res.sendStatus(401);
8     const user = await Users.findAll({
9       where:{
10         refresh_token: refreshToken
11       }
12     });
13     if(!user[0]) return res.sendStatus(403);
14     jwt.verify(refreshToken, process.env.REFRESH_TOKEN_SECRET, (err, decoded) => {
15       if(err) return res.sendStatus(403);
16       const userId = user[0].id;
17       const name = user[0].name;
18       const email = user[0].email;
19       const accessToken = jwt.sign({userId, name, email}, process.env.ACCESS_TOKEN_SECRET,{
20         expiresIn: '15s'
21       });
22       res.json({ accessToken });
23     });
24   } catch (error) {
25     console.log(error);
26   }
27 }
```

Syntax diatas berfungsi untuk mengatur refresh roken yang berfungsi untuk mengidentifikasi jika user dalam keadaan login atau tidak. Jika tidak maka akan menampilkan error.

- **USERS.JS**

```


1 import Users from "../models/UserModel.js";
2 import bcrypt from "bcrypt";
3 import jwt from "jsonwebtoken";
4
5 export const getUsers = async (req, res) => {
6   try {
7     const users = await Users.findAll({
8       attributes: ['id', 'name', 'email', 'gender']
9     });
10    res.status(200).json(users);
11  } catch (error) {
12    console.log(error);
13  }
14 }
15
16 export const getUserId = async (req, res) => {
17   try {
18     const response = await Users.findOne({
19       where: {
20         id: req.params.id
21       }
22     });
23    res.status(200).json(response);
24  } catch (error) {
25    console.log(error.message);
26  }
27 }
28
29 export const Register = async (req, res) => {
30   const { name, email, gender, password, confirmPassword } = req.body;
31   if (password !== confirmPassword) return res.status(400).json({msg: "Password and Confirm Password do not match"});
32   const salt = await bcrypt.genSalt(10);
33   const hashPassword = await bcrypt.hash(password, salt);
34   try {
35     await Users.create({
36       name: name,
37       email: email,
38       gender: gender,
39       password: hashPassword
40     });
41     res.json({msg: "Registration Successful"});
42   } catch (error) {
43     console.log(error);
44   }
45 }
46
47 export const Login = async (req, res) => {
48   try {
49     const user = await Users.findOne({
50       where: {
51         email: req.body.email
52       }
53     });
54     const match = await bcrypt.compare(req.body.password, user[0].password);
55     if (!match) return res.status(400).json({msg: "Wrong Password"});
56     const userId = user[0].id;
57     const name = user[0].name;
58     const email = user[0].email;
59     const accessToken = jwt.sign({userId, name, email}, process.env.ACCESS_TOKEN_SECRET, {
60       expiresIn: '15s'
61     });
62     const refreshToken = jwt.sign({userId, name, email}, process.env.REFRESH_TOKEN_SECRET, {
63       expiresIn: '1d'
64     });
65     await Users.update({refresh_token: refreshToken}, {
66       where: {
67         id: userId
68       }
69     });
70     res.cookie('refreshToken', refreshToken, {
71       httpOnly: true,
72       maxAge: 24 * 60 * 60 * 1000
73     });
74     res.json({ accessToken });
75   } catch (error) {
76     res.status(404).json({msg: "Email not found"});
77   }
78 }
79
80 export const Logout = async (req, res) => {
81   const refreshToken = req.cookies.refreshToken;
82   if (!refreshToken) return res.status(204);
83   const user = await Users.findOne({
84     where: {
85       refresh_token: refreshToken
86     }
87   });
88   if (!user[0]) return res.status(204);
89   const userId = user[0].id;
90   await Users.update({refresh_token: null}, {
91     where: {
92       id: userId
93     }
94   });
95   res.clearCookie('refreshToken');
96   return res.status(200);
97 }
98
99 export const createUser = async (req, res) => {
100   try {
101     await Users.create(req.body);
102     res.status(201).json({msg: "User Created"});
103   } catch (error) {
104     console.log(error.message);
105   }
106 }
107
108 export const updateUser = async (req, res) => {
109   try {
110     await Users.update(req.body, {
111       where: {
112         id: req.params.id
113       }
114     });
115     res.status(200).json({msg: "User Updated"});
116   } catch (error) {
117     console.log(error.message);
118   }
119 }
120
121 export const deleteUser = async (req, res) => {
122   try {
123     await Users.destroy({
124       where: {
125         id: req.params.id
126       }
127     });
128     res.status(200).json({msg: "User Deleted"});
129   } catch (error) {
130     console.log(error.message);
131   }
132 }

```

Syntax diatas berfungsi untuk mengatur semua control yang terjadi pada backend seperti mengambil userid yang sudah diatur dari database secara autoincrement. Register yang akan mengambil nama, email, gender dan password

kemudian jika password dan confirm password tidak sama maka akan menampilkan message password tidak sesuai. Jika sesuai password akan di hash dan disimpan kedalam database. Login akan mengambil username dan password jika user memasukan username dan password sesuai maka akan mendapatkan access token dan refresh token. createUser berfungsi untuk menambahkan user yang dibuat pada /adduser kedalam database. updateUser berfungsi untuk mengedit data user pada /edituser dan menyimpannya dalam database. deleteUser berfungsi untuk mendestroy data user berdasarkan id yang di klik.

- **VERIFYTOKEN.JS**



```
1 import jwt from "jsonwebtoken";
2
3 export const verifyToken = (req, res, next) => {
4   const authHeader = req.headers['authorization'];
5   const token = authHeader && authHeader.split(' ')[1];
6   if(token == null) return res.sendStatus(401);
7   jwt.verify(token, process.env.ACCESS_TOKEN_SECRET, (err, decoded) => {
8     if(err) return res.sendStatus(403);
9     req.email = decoded.email;
10    next();
11  })
12 }
```

Syntax diatas berfungsi untuk memverifikasi token yang didapat dari login. Jika token kosong maka akan menampilkan status error 401, jika error akan menampilkan status error 403.

- **USERMODEL.JS**

```
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   name:{
8     type: DataTypes.STRING
9   },
10  email:{
11    type: DataTypes.STRING
12  },
13  password:{
14    type: DataTypes.STRING
15  },
16  refresh_token:{
17    type: DataTypes.TEXT
18  },
19  gender:{
20    type: DataTypes.STRING
21  }
22 },{
23   freezeTableName:true
24 });
25
26 export default Users;
27
28 (async()=>{
29   await db.sync();
30 })();
```

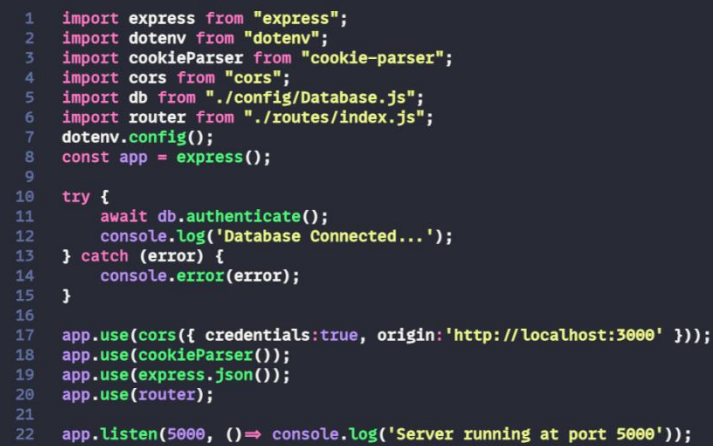
Syntax diatas berfungsi untuk mendefine users dalam database menjadi nama yang bertipe data string, email yang bertipe data string, password yang bertipe data string, refresh token yang bertipe data text, gender yang bertipe data string. Kemudian syntax ini juga mensinkronasi ke database.

- **INDEX.JS (ROUTES)**

```
1 import express from "express";
2 import { getUsers, getUserById, Register, Login, Logout, createUser, updateUser, deleteUser } from "../controllers/Users.js";
3 import { verifyToken } from "../middleware/VerifyToken.js";
4 import { refreshToken } from "../controllers/RefreshToken.js";
5
6 const router = express.Router();
7
8 router.get('/users', verifyToken, getUsers);
9 router.post('/users', Register);
10 router.post('/login', Login);
11 router.get('/token', refreshToken);
12 router.delete('/logout', Logout);
13 router.get('/users/:id', getUserById);
14 router.post('/users', createUser);
15 router.patch('/users/:id', updateUser);
16 router.delete('/users/:id', deleteUser);
17
18 export default router;
```

Syntax diatas berfungsi mengatur method method yang digunakan pada setiap /. Seperti /users verifytoken dan getuser menggunakan method get, jika /users register menggunakan post. /login untuk login menggunakan method post. /token untuk refreshtoken menggunakan method get. /logout menggunakan method delete. /users/:id menggunakan method get. /users untuk create user menggunakan method post. /users/:id pada update user menggunakan method patch pada delete user menggunakan method delete.

- **INDEX.JS**

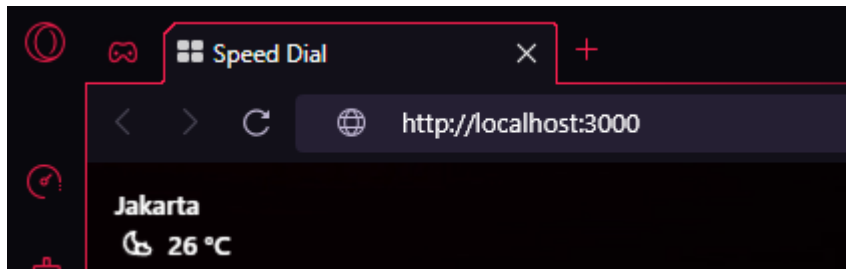


```
1 import express from "express";
2 import dotenv from "dotenv";
3 import cookieParser from "cookie-parser";
4 import cors from "cors";
5 import db from "../config/Database.js";
6 import router from "../routes/index.js";
7 dotenv.config();
8 const app = express();
9
10 try {
11   await db.authenticate();
12   console.log('Database Connected...');
13 } catch (error) {
14   console.error(error);
15 }
16
17 app.use(cors({ credentials:true, origin:'http://localhost:3000' }));
18 app.use(cookieParser());
19 app.use(express.json());
20 app.use(router);
21
22 app.listen(5000, () => console.log('Server running at port 5000'));
```

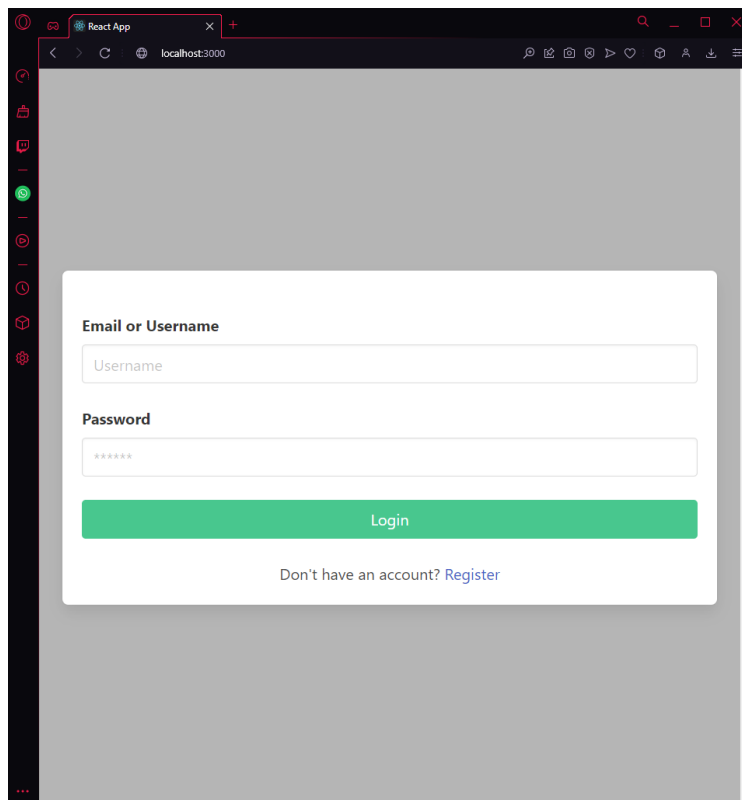
Syntax diatas berfungsi untuk mengautentifikasi database jika berhasil maka akan menampilkan database connected. Jika error akan menampilkan error pada console. Pada syntax ini juga mengatur port yang digunakan.

LANGKAH LAHKAH PENGGUNAAN WEB

1. Ketiklah URL : <http://localhost:3000>.



2. Maka akan tampil halaman Login. User yang sudah terdaftar dapat langsung memasukkan username dan password, jika belum terdaftar dapat melakukan pendaftaran dengan mengklik tombol register.



3. Setelah berada dalam halaman register user dapat langsung mengisi data data yang diperlukan.

React App

localhost:3000/register

Name

NamaUser

Email

emailuser@email.com

Gender

Male

Password

.....

Confirm Password

.....

Register

Jika terdapat ketidaksesuaian pada Confirm Password web akan menampilkan

React App

localhost:3000/register

Password and Confirm Password do not match

Name

NamaUser

Email

emailuser@email.com

Gender

Male

Password

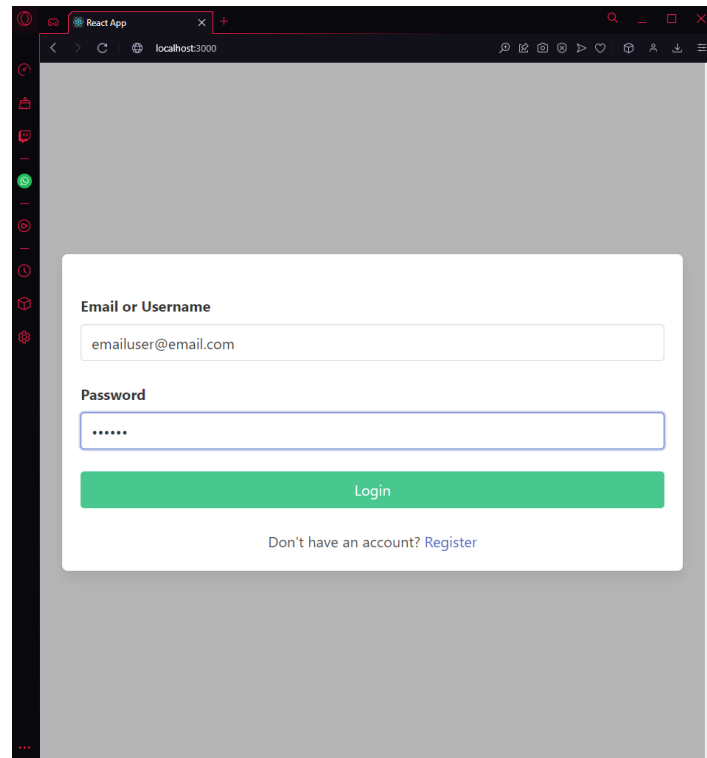
.....

Confirm Password

.....

Register

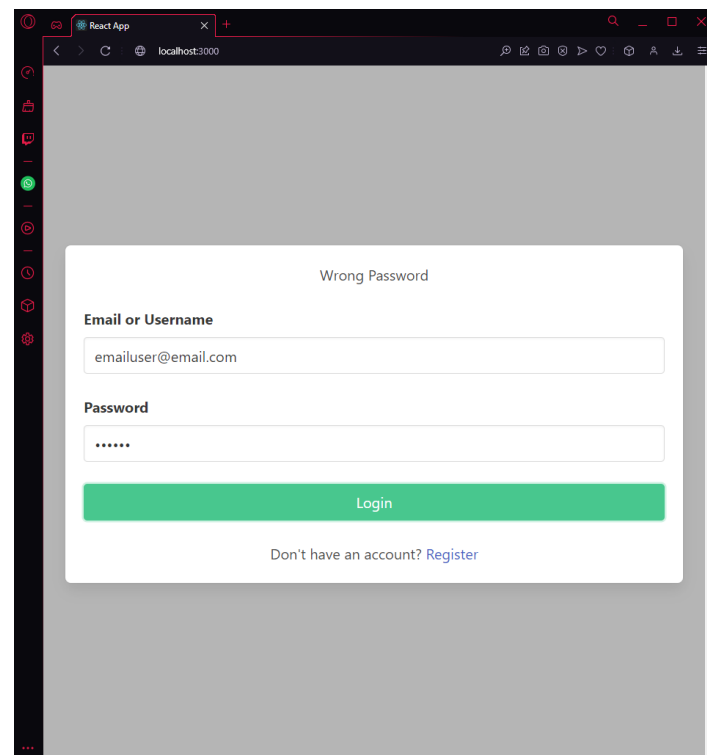
4. Kemudian user dapat langsung melakukan login dengan email dan password yang sudah dibuat sebelumnya.



The screenshot shows a web browser window with the address bar displaying 'localhost:3000'. The page contains a login form with the following elements:

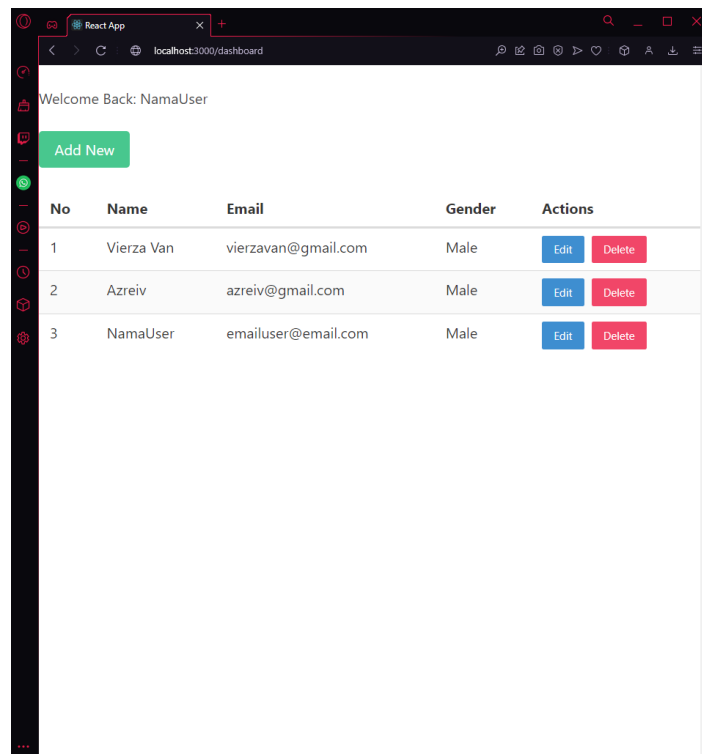
- Email or Username:** A text input field containing 'emailuser@email.com'.
- Password:** A text input field containing six dots (password masked).
- Login:** A green button.
- Register:** A link labeled 'Don't have an account? Register'.

Jika terdapat ketidaksesuaian pada password maka web akan menampilkan



The screenshot shows the same web browser window, but now an error message 'Wrong Password' is displayed above the login form. The form fields and buttons remain the same as in the previous screenshot.

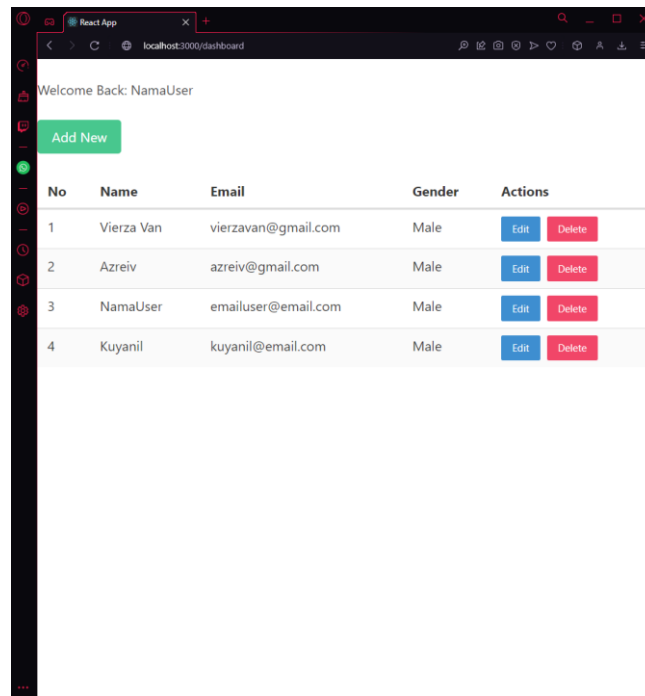
5. Jika user berhasil melakukan login maka akan diarahkan ke halaman dashboard, yang dimana user dapat melakukan create, read, update, dan delete.



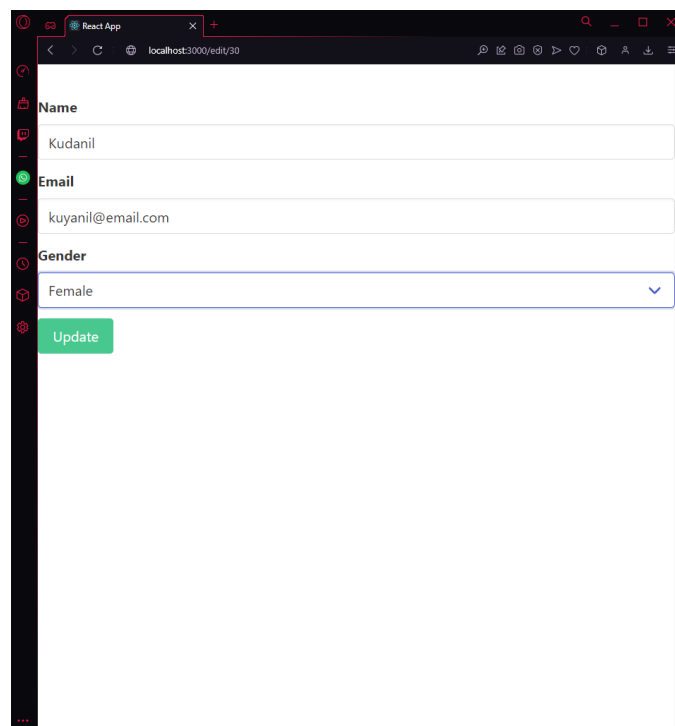
6. User dapat menambahkan data dengan mengklik Add New, jika sudah diklik maka akan diarahkan ke halaman add. Pada halaman add user dapat memasukkan nama, email, gender, password dan confirm password. Jika dirasa sudah tepat maka user dapat mengklik tombol add user.

The screenshot shows the "Add User" form. It contains input fields for "Name" (filled with "Kuyanil"), "Email" (filled with "kuyanil@email.com"), "Gender" (a dropdown menu with "Male" selected), "Password" (filled with "*****"), and "Confirm Password" (filled with "*****"). A green "Add User" button is at the bottom.

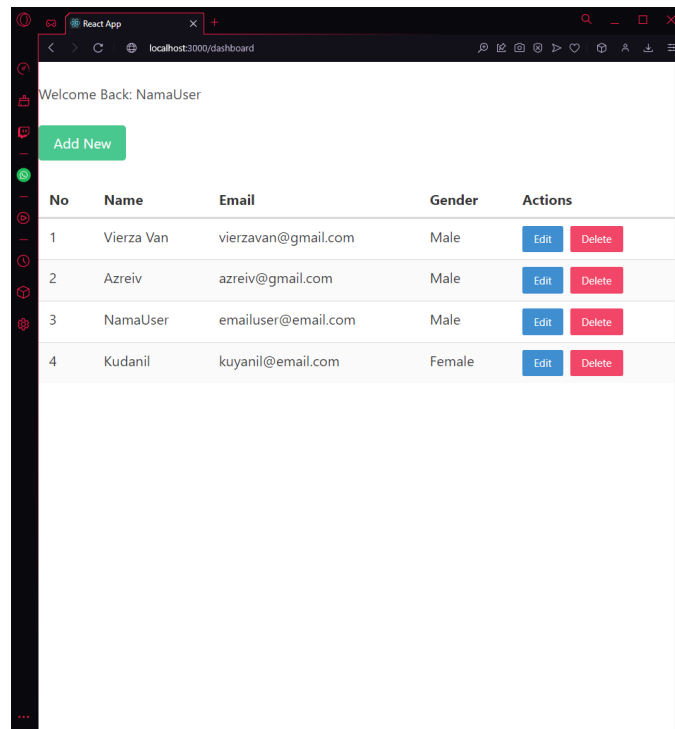
7. Jika berhasil ditambahkan maka data yang telah dimasukkan maka user akan diarahkan ke dashboard.



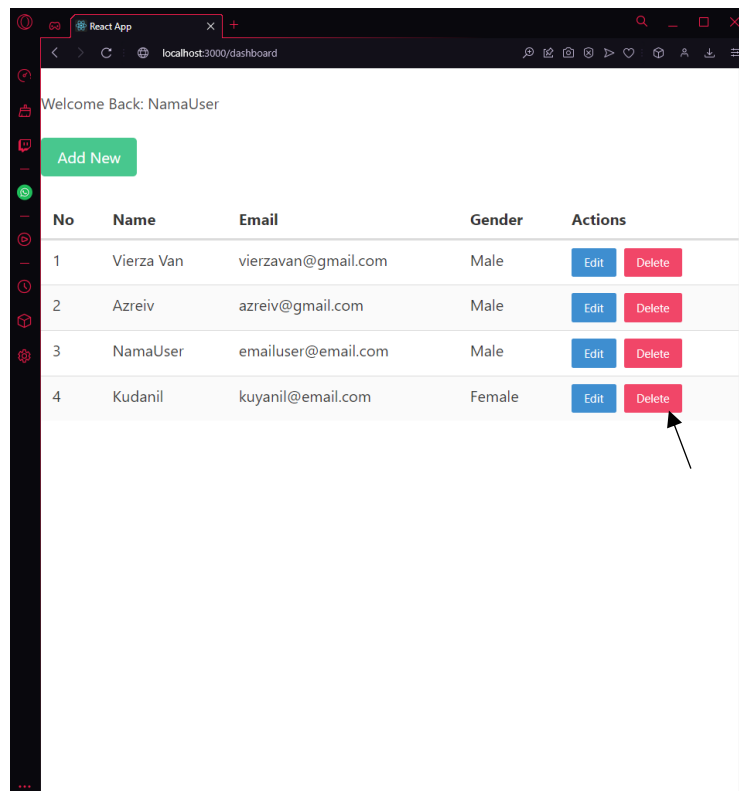
8. User juga dapat melakukan pengeditan data hanya dengan mengklik tombol edit. Jika user sudah mengklik maka akan diarahkan ke halaman edit/30 (30 merupakan id user yang diedit). Disini user hanya dapat mengubah nama, email, dan gender.



9. Jika sudah user dapat mengklik tombol update, kemudian user akan diarahkan ke halaman dashboard dengan data yang sudah berubah.



10. Lalu user juga dapat melakukan penghapusan data dengan mengklik tombol delete.



Sesudah didelete

