Web Application Project: User Rating System

By-

Mobin Raja

mobin2002raza@gmail.com

9284688056

1. Backend - Express.js Setup

This part will handle user authentication, store management, ratings, and role-based access control. We'll use PostgreSQL for the database.

Steps for Backend Setup:

1. **Initialize Node.js Project:**
   - Run `npm init -y` to initialize a new Node.js project.
   - Install required dependencies:

     ```
     bash
     CopyEdit
     npm install express pg bcryptjs jsonwebtoken dotenv cors
     npm install --save-dev nodemon
     ```

2. **Create Folder Structure:**

   ```
   bash
   CopyEdit
   /backend
   ├── /controllers
   ├── /models
   ├── /routes
   ├── /middlewares
   ├── .env
   ├── server.js
   ├── db.js
   ```

3. **Create `server.js` for Express Setup:**

```
js
CopyEdit
const express = require('express');
const cors = require('cors');
const dotenv = require('dotenv');
const db = require('./db');

dotenv.config();

const app = express();
app.use(cors());
app.use(express.json());

// Routes
const userRoutes = require('./routes/userRoutes');
const storeRoutes = require('./routes/storeRoutes');

app.use('/api/users', userRoutes);
app.use('/api/stores', storeRoutes);

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

4. **Database Connection (`db.js`):**

```
js
CopyEdit
const { Pool } = require('pg');

const pool = new Pool({
  user: process.env.DB_USER,
  host: process.env.DB_HOST,
  database: process.env.DB_NAME,
  password: process.env.DB_PASSWORD,
  port: process.env.DB_PORT,
});

module.exports = pool;
```

5. **User Authentication Controller (`controllers/userController.js`):**

```
js
CopyEdit
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const pool = require('../db');

// Sign Up
const signUp = async (req, res) => {
  const { name, email, password, address } = req.body;
  const hashedPassword = await bcrypt.hash(password, 10);
  const newUser = await pool.query(
    'INSERT INTO users (name, email, password, address) VALUES ($1,
$2, $3, $4) RETURNING *',
    [name, email, hashedPassword, address]
  );
  res.status(201).json(newUser.rows[0]);
};

// Login
const login = async (req, res) => {
  const { email, password } = req.body;
  const user = await pool.query('SELECT * FROM users WHERE email =
$1', [email]);

  if (user.rows.length === 0) {
    return res.status(400).json({ message: 'User not found' });
  }

  const isMatch = await bcrypt.compare(password,
user.rows[0].password);
  if (!isMatch) {
    return res.status(400).json({ message: 'Invalid credentials' });
  }

  const token = jwt.sign({ userId: user.rows[0].id },
process.env.JWT_SECRET, { expiresIn: '1h' });
  res.json({ token });
};

module.exports = { signUp, login };
```

6. **Routes for Users (`routes/userRoutes.js`):**

```
js
CopyEdit
const express = require('express');
const { signUp, login } = require('../controllers/userController');

const router = express.Router();

router.post('/signup', signUp);
router.post('/login', login);

module.exports = router;
```

7. **Environment Variables (`.env`):**

```
ini
CopyEdit
DB_USER=your_database_user
DB_HOST=localhost
DB_NAME=rating_app
DB_PASSWORD=your_database_password
DB_PORT=5432
JWT_SECRET=your_jwt_secret
```

---

2. Frontend - React.js Setup

1. **Create React App:**

```
bash
```

```
CopyEdit
npx create-react-app frontend
cd frontend
npm install axios react-router-dom
```

2. **Folder Structure for React:**

```
bash
CopyEdit
/frontend
├── /components
├── /pages
├── /services
├── /utils
├── App.js
├── index.js
```

3. **Create Components:**
   - `Login.js`
   - `Signup.js`
   - `StoreList.js`
   - `StoreItem.js`
4. **Example `Login.js`:**

```
js
CopyEdit
import { useState } from 'react';
import axios from 'axios';

const Login = () => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const handleLogin = async (e) => {
    e.preventDefault();
    try {
      const response = await
axios.post('http://localhost:5000/api/users/login', { email, password
});
      localStorage.setItem('token', response.data.token);
      // Redirect to dashboard or homepage
    } catch (error) {
      console.log('Error logging in:', error.response.data);
    }
  };

  return (
    <div>
      <form onSubmit={handleLogin}>
        <input type="email" placeholder="Email" onChange={(e) =>
setEmail(e.target.value)} />
        <input type="password" placeholder="Password" onChange={(e)
=> setPassword(e.target.value)} />
        <button type="submit">Login</button>
      </form>
    </div>
  );
};
```

```
export default Login;
```

5. **Main App Component (`App.js`):**

```
js
CopyEdit
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';
import Login from './components/Login';
import Signup from './components/Signup';

const App = () => {
  return (
    <Router>
      <Routes>
        <Route path="/login" element={<Login />} />
        <Route path="/signup" element={<Signup />} />
      </Routes>
    </Router>
  );
};

export default App;
```

---

3. Frontend + Backend Integration

Now connect the React frontend with your Express backend by using Axios to handle HTTP requests to the backend API (such as login, sign up, fetching stores, and submitting ratings).

---

4. Running the Application:

1. **Run Backend:**

```
bash
CopyEdit
node server.js
```

2. **Run Frontend:**

```
bash
CopyEdit
npm start
```

---

5. Deploying the Application:

You can deploy:

- **Backend** on platforms like **Heroku**, **DigitalOcean**, or **AWS**.
- **Frontend** on **Netlify** or **Vercel**.