FullStack Intern Coding Challenge

Tech Stack

- Backend: Any one of the following frameworks - ExpressJs, Loopback, NestJs
- Database: PostgreSQL or MySQL
- Frontend: ReactJs

Project Overview

We need a web application that allows users to submit ratings for stores registered on the platform. The ratings should range from 1 to 5. A single login system should be implemented for all users. Based on their roles, users will have access to different functionalities upon logging in.

Extracted Code Summary

Frontend Components

- Login.js: Handles user authentication with email and password.
- Register.js: Allows users to sign up with validation for name, email, address, and password strength.
- StoreList.js: Fetches and displays a list of stores from the backend.

Backend Controllers & Models

- authController.js: Handles user authentication with JWT.
- Store.js: Defines the Store model.
- User.js: Defines the User model.
- authRoutes.js: Defines Express routes for authentication.

Complete Code Files

Login Component

```
import React, { useState } from 'react';
import axios from 'axios';
import { useHistory } from 'react-router-dom';

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

    const handleLogin = async (e) => {
        e.preventDefault();
        try {
            const response = await axios.post('http://localhost:5000/api/auth/login', { email, password });
            localStorage.setItem('token', response.data.token);
            history.push('/dashboard');
        } catch (error) {
            console.error('Login failed', error);
        }
    };

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

export default Login;
```

Register Component

```
import React, { useState } from "react";
import axios from "axios";

const Register = () => {
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    address: "",
    password: "",
    role: "normal_user", // Default role
  });

  const [error, setError] = useState("");

  // Handle input change
  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  // Validate form
  const validateForm = () => {
    const { name, email, address, password } = formData;

    if (name.length < 20 || name.length > 60) {
      setError("Name must be between 20 and 60 characters.");
```

```
      return false;
    }
    if (address.length > 400) {
      setError("Address must not exceed 400 characters.");
      return false;
    }
    if (!/^(?=.*[A-Z])(?=.*[^A-Za-z0-9]).{8,16}$/.test(password)) {
      setError(
           "Password must be 8-16 characters long, contain at least one uppercase letter and one special
character."
      );
      return false;
    }
    if (!/^\S+@\S+\.\S+$/.test(email)) {
      setError("Invalid email format.");
      return false;
    }
    setError("");
    return true;
  };

  // Handle form submission
  const handleSubmit = async (e) => {
    e.preventDefault();
    if (!validateForm()) return;

    try {
      const response = await axios.post("http://localhost:5000/api/register", formData);
      alert("Registration Successful!");
      console.log(response.data);
    } catch (error) {
      setError("Registration failed. Please try again.");
      console.error(error);
    }
  };

  return (
    <div className="flex justify-center items-center h-screen bg-gray-100">
      <div className="bg-white p-6 rounded-lg shadow-lg w-96">
        <h2 className="text-2xl font-semibold text-center mb-4">Register</h2>
        {error && <p className="text-red-500 text-sm">{error}</p>}
        <form onSubmit={handleSubmit}>
          <label className="block mb-2">Name:</label>
          <input
            type="text"
            name="name"
            value={formData.name}
            onChange={handleChange}
            className="w-full p-2 border rounded mb-3"
            required
          />

          <label className="block mb-2">Email:</label>
          <input
            type="email"
            name="email"
            value={formData.email}
            onChange={handleChange}
            className="w-full p-2 border rounded mb-3"
            required
          />

          <label className="block mb-2">Address:</label>
          <textarea
            name="address"
            value={formData.address}
            onChange={handleChange}
```

```
            className="w-full p-2 border rounded mb-3"
            required
          />

          <label className="block mb-2">Password:</label>
          <input
            type="password"
            name="password"
            value={formData.password}
            onChange={handleChange}
            className="w-full p-2 border rounded mb-3"
            required
          />

          <label className="block mb-2">Role:</label>
          <select
            name="role"
            value={formData.role}
            onChange={handleChange}
            className="w-full p-2 border rounded mb-3"
          >
            <option value="normal_user">Normal User</option>
            <option value="store_owner">Store Owner</option>
            <option value="admin">Admin</option>
          </select>

          <button
            type="submit"
            className="w-full bg-blue-500 text-white py-2 rounded hover:bg-blue-600"
          >
            Register
          </button>
        </form>
      </div>
    </div>
  );
};

export default Register;
```

Store List Component

```
import React, { useEffect, useState } from 'react';
import axios from 'axios';

const StoreList = () => {
    const [stores, setStores] = useState([]);

    useEffect(() => {
        const fetchStores = async () => {
            const response = await axios.get('http://localhost:5000/api/stores');
            setStores(response.data);
        };
        fetchStores();
    }, []);

    return (
        <div>
            <h1>Stores</h1>
            <ul>
                {stores.map(store => (
                    <li key={store.id}>{store.name} - {store.address} - Rating: {store.rating}</li>
                ))}
            </ul>
        </div>
    );
};
```

```
export default StoreList;
```

Authentication Controller

```
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const User = require('/models/User');

exports.register = async (req, res) => {
    try {
        const { name, email, password, address, role } = req.body;

        // Input validation
        if (!name || !email || !password || !address || !role) {
            return res.status(400).json({ message: 'All fields are required' });
        }

        // Check if user already exists
        const existingUser = await User.findOne({ where: { email } });
        if (existingUser) {
            return res.status(400).json({ message: 'User already exists' });
        }

        // Hash the password
        const hashedPassword = await bcrypt.hash(password, 10);

        // Create the user
        const user = await User.create({ name, email, password: hashedPassword, address, role });

        // Return success response
        res.status(201).json({ message: 'User registered successfully', user });
    } catch (error) {
        console.error('Error during registration:', error);
        res.status(500).json({ message: 'Internal server error' });
    }
};

exports.login = async (req, res) => {
    try {
        const { email, password } = req.body;

        // Input validation
        if (!email || !password) {
            return res.status(400).json({ message: 'Email and password are required' });
        }

        // Find the user by email
        const user = await User.findOne({ where: { email } });
        if (!user) {
            return res.status(401).json({ message: 'Invalid credentials' });
        }

        // Compare the password
        const isPasswordValid = await bcrypt.compare(password, user.password);
        if (!isPasswordValid) {
            return res.status(401).json({ message: 'Invalid credentials' });
        }

        // Generate JWT token
        const token = jwt.sign({ id: user.id, role: user.role }, process.env.JWT_SECRET, { expiresIn: '1h' });

        // Return success response with token
        res.status(200).json({ token });
    } catch (error) {
        console.error('Error during login:', error);
        res.status(500).json({ message: 'Internal server error' });
```

```
    }
};
```

Store Model

```
const { DataTypes } = require('sequelize');
const sequelize = require('../config/db');

const Store = sequelize.define('Store', {
    id: {
        type: DataTypes.INTEGER,
        primaryKey: true,
        autoIncrement: true,
    },
    name: {
        type: DataTypes.STRING,
        allowNull: false,
    },
    email: {
        type: DataTypes.STRING,
        allowNull: false,
        unique: true,
    },
    address: {
        type: DataTypes.STRING(400),
        allowNull: false,
    },
    rating: {
        type: DataTypes.FLOAT,
        defaultValue: 0,
    },
});

module.exports = Store;
```

User Model

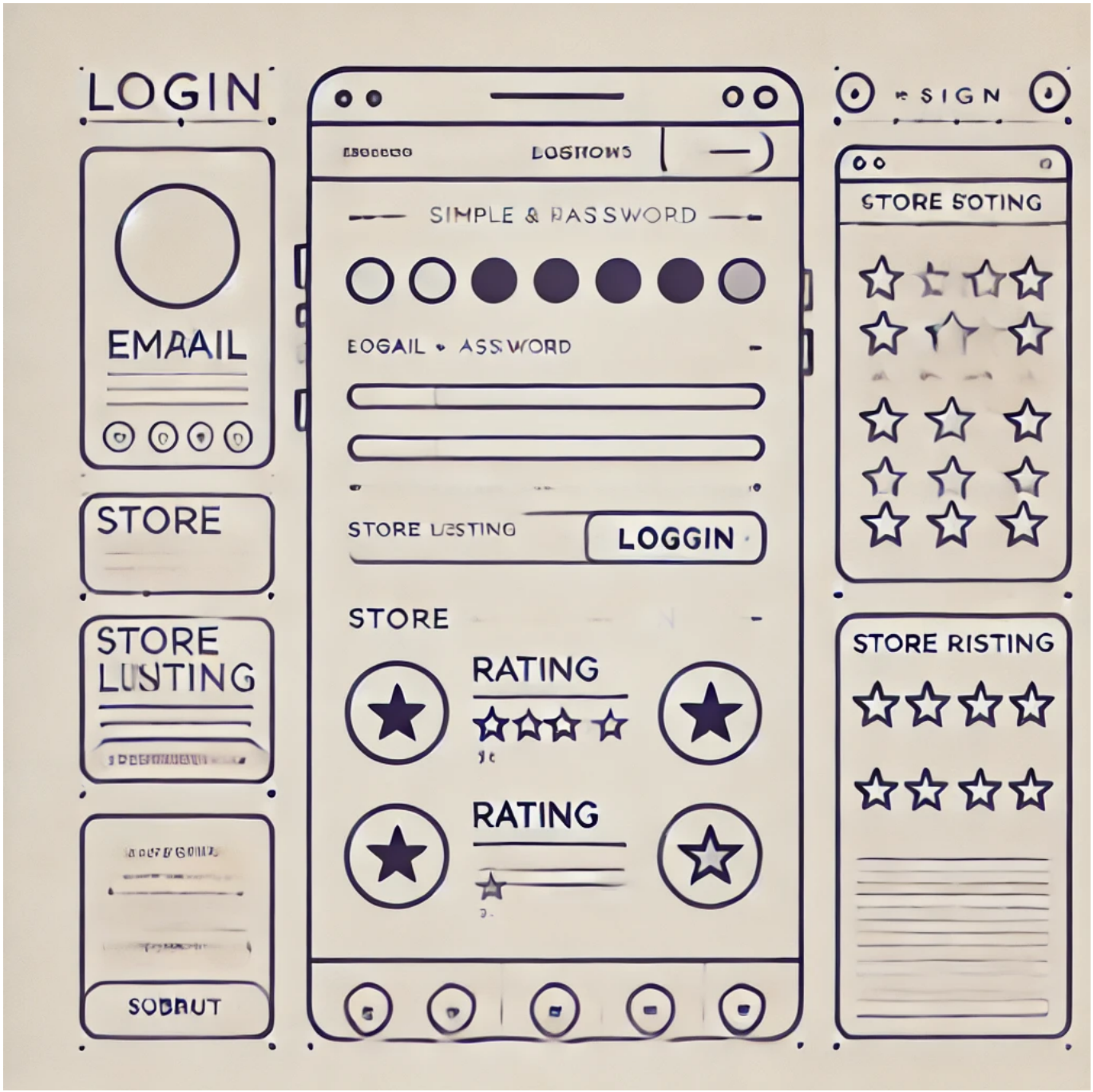
```
const { DataTypes } = require('sequelize');
const sequelize = require('../config/db');

const User = sequelize.define('User', {
    id: {
        type: DataTypes.INTEGER,
        primaryKey: true,
        autoIncrement: true,
    },
    name: {
        type: DataTypes.STRING(60),
        allowNull: false,
    },
    email: {
        type: DataTypes.STRING,
        allowNull: false,
        unique: true,
    },
    password: {
        type: DataTypes.STRING,
        allowNull: false,
    },
    address: {
        type: DataTypes.STRING(400),
        allowNull: false,
    },
    role: {
        type: DataTypes.ENUM('admin', 'user', 'store_owner'),
        defaultValue: 'user',
    },
```

```
});

module.exports = User;
```

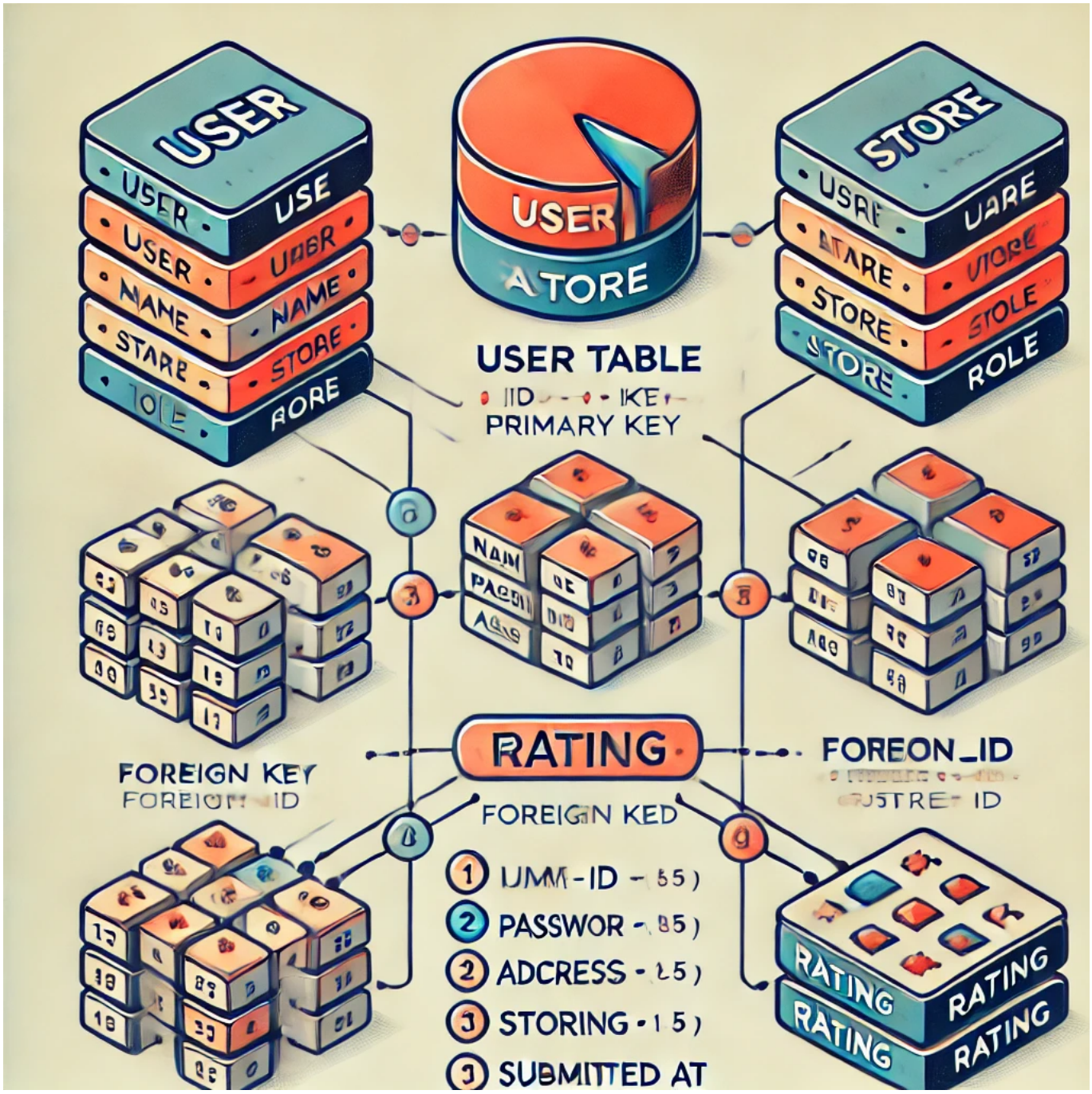
Authentication Routes

```
const express = require('express');
const authController = require('../controllers/authController');
const router = express.Router();

router.post('/register', authController.register);
router.post('/login', authController.login);

module.exports = router;
```

Wireframe Layout



Database Schema Diagram



User Flowchart

Tech Stack Logos