job-portal/

├── client/ # Front-end React app

│ ├── src/

│ │ ├── components/

│ │ │ ├── Login.js

│ │ │ ├── Signup.js

│ │ │ ├── JobList.js

│ │ │ ├── JobDetails.js

│ │ │ ├── ApplyJob.js

│ │ │ ├── ApplicationTracking.js

│ │ │ └── ...other components

│ │ ├── App.js

│ │ └── index.js

│ └── package.json

└── server/ # Back-end Node.js app

├── controllers/

├── models/

├── routes/

├── db.js

├── server.js

└── package.json

Server code

**// server/db.js**

const { Pool } = require('pg');

const pool = new Pool({

user: 'your\_db\_user', // PostgreSQL username

host: 'localhost',

database: 'job\_portal\_db', // PostgreSQL database name

password: 'your\_password', // PostgreSQL password

port: 5432,

});

module.exports = pool;

**// server/controllers/authController.js**

const db = require('../db');

const bcrypt = require('bcrypt');

const jwt = require('jsonwebtoken');

const register = async (req, res) => {

const { username, email, password, isEmployer } = req.body;

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

try {

const result = await db.query(

'INSERT INTO users (username, email, password, is\_employer) VALUES ($1, $2, $3, $4) RETURNING \*',

[username, email, hashedPassword, isEmployer]

);

res.status(201).json(result.rows[0]);

} catch (error) {

res.status(500).json({ error: 'User registration failed' });

}

};

const login = async (req, res) => {

const { email, password } = req.body;

try {

const result = await db.query('SELECT \* FROM users WHERE email = $1', [email]);

const user = result.rows[0];

if (user && await bcrypt.compare(password, user.password)) {

const token = jwt.sign({ id: user.id, isEmployer: user.is\_employer }, 'your\_jwt\_secret', { expiresIn: '1h' });

res.json({ token });

} else {

res.status(401).json({ error: 'Invalid credentials' });

}

} catch (error) {

res.status(500).json({ error: 'Login failed' });

}

};

module.exports = { register, login };

**// server/controllers/jobController.js**

const db = require('../db');

const postJob = async (req, res) => {

const { title, description, location, company } = req.body;

const employerId = req.user.id;

try {

const result = await db.query(

'INSERT INTO jobs (title, description, location, company, employer\_id) VALUES ($1, $2, $3, $4, $5) RETURNING \*',

[title, description, location, company, employerId]

);

res.status(201).json(result.rows[0]);

} catch (error) {

res.status(500).json({ error: 'Job posting failed' });

}

};

const applyForJob = async (req, res) => {

const jobId = req.params.jobId;

const userId = req.user.id;

try {

const result = await db.query(

'INSERT INTO applications (job\_id, user\_id) VALUES ($1, $2) RETURNING \*',

[jobId, userId]

);

res.status(201).json(result.rows[0]);

} catch (error) {

res.status(500).json({ error: 'Job application failed' });

}

};

const trackApplications = async (req, res) => {

const userId = req.user.id;

try {

const result = await db.query(

'SELECT \* FROM applications WHERE user\_id = $1',

[userId]

);

res.json(result.rows);

} catch (error) {

res.status(500).json({ error: 'Failed to track applications' });

}

};

module.exports = { postJob, applyForJob, trackApplications };

**// server/routes/auth.js**

const express = require('express');

const { register, login } = require('../controllers/authController');

const router = express.Router();

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

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

module.exports = router;

**// server/routes/jobs.js**

const express = require('express');

const { postJob, applyForJob, trackApplications } = require('../controllers/jobController');

const router = express.Router();

router.post('/post', postJob);

router.post('/:jobId/apply', applyForJob);

router.get('/applications', trackApplications);

module.exports = router;

**// server/index.js**

const express = require('express');

const cors = require('cors');

const authRoutes = require('./routes/auth');

const jobRoutes = require('./routes/jobs');

const app = express();

const PORT = 5000;

app.use(cors());

app.use(express.json());

app.use('/api/auth', authRoutes);

app.use('/api/jobs', jobRoutes);

app.listen(PORT, () => {

console.log(`Server is running on http://localhost:${PORT}`);

});

## DATABASE

CREATE TABLE users (

id SERIAL PRIMARY KEY,

username VARCHAR(255) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL,

is\_employer BOOLEAN DEFAULT FALSE

);

CREATE TABLE jobs (

id SERIAL PRIMARY KEY,

title VARCHAR(255) NOT NULL,

description TEXT,

location VARCHAR(255),

company VARCHAR(255),

employer\_id INTEGER REFERENCES users(id) ON DELETE CASCADE

);

CREATE TABLE applications (

id SERIAL PRIMARY KEY,

job\_id INTEGER REFERENCES jobs(id) ON DELETE CASCADE,

user\_id INTEGER REFERENCES users(id) ON DELETE CASCADE,

status VARCHAR(50) DEFAULT 'Pending',

applied\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CLIENT

npx create-react-app client

cd client

npm install axios react-router-dom

**// client/src/api/auth.js**

import axios from 'axios';

export const registerUser = async (data) => axios.post('/api/auth/register', data);

export const loginUser = async (data) => axios.post('/api/auth/login', data);

**// client/src/api/jobs.js**

import axios from 'axios';

export const postJob = async (data, token) => axios.post('/api/jobs/post', data, {

headers: { Authorization: `Bearer ${token}` }

});

export const applyForJob = async (jobId, token) => axios.post(`/api/jobs/${jobId}/apply`, {}, {

headers: { Authorization: `Bearer ${token}` }

});

export const getApplications = async (token) => axios.get('/api/jobs/applications', {

headers: { Authorization: `Bearer ${token}` }

});

**// client/src/components/Login.js**

import React, { useState } from 'react';

import { loginUser } from '../api/auth';

function Login() {

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const handleSubmit = async (e) => {

e.preventDefault();

const res = await loginUser({ email, password });

localStorage.setItem('token', res.data.token);

};

return (

<form onSubmit={handleSubmit}>

<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>

);

}

export default Login;

**// client/src/components/PostJob.js**

import React, { useState } from 'react';

import { postJob } from '../api/jobs';

function PostJob() {

const [title, setTitle] = useState('');

const [description, setDescription] = useState('');

const [location, setLocation] = useState('');

const [company, setCompany] = useState('');

const handleSubmit = async (e) => {

e.preventDefault();

const token = localStorage.getItem('token'); // Retrieve JWT token from localStorage

await postJob({ title, description, location, company }, token);

alert('Job posted successfully!');

};

return (

<form onSubmit={handleSubmit}>

<h2>Post a Job</h2>

<input

type="text"

placeholder="Job Title"

onChange={(e) => setTitle(e.target.value)}

required

/>

<textarea

placeholder="Job Description"

onChange={(e) => setDescription(e.target.value)}

required

/>

<input

type="text"

placeholder="Location"

onChange={(e) => setLocation(e.target.value)}

required

/>

<input

type="text"

placeholder="Company"

onChange={(e) => setCompany(e.target.value)}

required

/>

<button type="submit">Post Job</button>

</form>

);

}

export default PostJob;

**// client/src/components/ApplyJob.js**

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

import { getJobs } from '../api/jobs'; // You need to create this API

import { applyForJob } from '../api/jobs';

function ApplyJob() {

const [jobs, setJobs] = useState([]);

useEffect(() => {

const fetchJobs = async () => {

// Fetch the job listings from the server

const response = await getJobs(); // Implement this API

setJobs(response.data);

};

fetchJobs();

}, []);

const handleApply = async (jobId) => {

const token = localStorage.getItem('token');

await applyForJob(jobId, token);

alert('Application submitted successfully!');

};

return (

<div>

<h2>Available Jobs</h2>

<ul>

{jobs.map(job => (

<li key={job.id}>

<h3>{job.title}</h3>

<p>{job.description}</p>

<button onClick={() => handleApply(job.id)}>Apply</button>

</li>

))}

</ul>

</div>

);

}

export default ApplyJob;

**// client/src/components/ApplicationStatus.js**

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

import { getApplications } from '../api/jobs';

function ApplicationStatus() {

const [applications, setApplications] = useState([]);

useEffect(() => {

const fetchApplications = async () => {

const token = localStorage.getItem('token');

const response = await getApplications(token);

setApplications(response.data);

};

fetchApplications();

}, []);

return (

<div>

<h2>Application Status</h2>

<ul>

{applications.map(application => (

<li key={application.id}>

<p>Job ID: {application.job\_id}</p>

<p>Status: {application.status}</p>

<p>Applied At: {new Date(application.applied\_at).toLocaleString()}</p>

</li>

))}

</ul>

</div>

);

}

export default ApplicationStatus;

**4. Job Listing API (Fetch Job Listings)**

You'll need to create an API function to fetch job listings from the backend. Add the following function to your api/jobs.js file

**// client/src/api/jobs.js**

import axios from 'axios';

export const getJobs = async () => axios.get('/api/jobs'); // New API to fetch jobs

**// client/src/App.js**

import React from 'react';

import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';

import Login from './components/Login';

import PostJob from './components/PostJob';

import ApplyJob from './components/ApplyJob';

import ApplicationStatus from './components/ApplicationStatus';

function App() {

return (

<Router>

<div>

<h1>Job Portal</h1>

<Switch>

<Route path="/login" component={Login} />

<Route path="/post-job" component={PostJob} />

<Route path="/apply-job" component={ApplyJob} />

<Route path="/application-status" component={ApplicationStatus} />

<Route path="/" exact>

<h2>Welcome to the Job Portal</h2>

</Route>

</Switch>

</div>

</Router>

);

}

export default App;