**Job Portal Project - MERN Stack**

Welcome to the Job Portal project! This application is designed to provide a platform for users to find and apply for job opportunities. It is built using the MERN (MongoDB, Express.js, React, Node.js) stack. Please keep in mind that this is the developer's first large-scale project, so the file structure might not be optimal.

**Project Overview**

The Job Portal project allows users to:

* Browse and search for job listings
* Create an account to apply for jobs
* Post job openings for employers
* Manage job applications

**Technologies Used**

* **React**: Frontend library for building user interfaces.
* **Node.js**: JavaScript runtime for building scalable network applications.
* **Express.js**: Web application framework for Node.js.
* **MongoDB**: NoSQL database for storing application data.

**Additional Packages Used**

* **bcryptjs**: Library for hashing passwords securely.
* **Formik & Yup**: Form library for React applications with validation using Yup.
* **react-bootstrap**: Bootstrap components as React components for easy UI development.
* **react-router-dom**: Declarative routing for React applications.

**Getting Started**

To run the project locally, follow these steps:

**Prerequisites**

Make sure you have the following installed:

* Node.js
* npm
* MongoDB

**Installation**

1. Clone the repository:
2. git clone https://github.com/your-username/job-portal.git
3. Change into the project directory:
4. cd job-frontend

npm install -f

cd job-backend

npm i -f

1. Create a .env file with necessary url's and variables.
2. Run the client server with:

npm start

1. Run the backend server with:

npm start

Code:

# Job Portal Project - MERN Stack

Welcome to the Job Portal project! This application is designed to provide a platform for users to find and apply for job opportunities. It is built using the MERN (MongoDB, Express.js, React, Node.js) stack. Please keep in mind that this is the developer's first large-scale project, so the file structure might not be optimal.

## Project Overview

The Job Portal project allows users to:

- Browse and search for job listings

- Create an account to apply for jobs

- Post job openings for employers

- Manage job applications

## Technologies Used

- \*\*React\*\*: Frontend library for building user interfaces.

- \*\*Node.js\*\*: JavaScript runtime for building scalable network applications.

- \*\*Express.js\*\*: Web application framework for Node.js.

- \*\*MongoDB\*\*: NoSQL database for storing application data.

### Additional Packages Used

- \*\*bcryptjs\*\*: Library for hashing passwords securely.

- \*\*Formik & Yup\*\*: Form library for React applications with validation using Yup.

- \*\*react-bootstrap\*\*: Bootstrap components as React components for easy UI development.

- \*\*react-router-dom\*\*: Declarative routing for React applications.

## Getting Started

To run the project locally, follow these steps:

### Prerequisites

Make sure you have the following installed:

- Node.js

- npm

- MongoDB

### Installation

1. Clone the repository:

```bash

git clone https://github.com/your-username/job-portal.git

2. Change into the project directory:

```bash

cd job-frontend

npm install -f

```

```bash

cd job-backend

npm i -f

```

3. Create a .env file with necessary url's and variables.

4. Run the client server with:

```bash

npm start

```

5. Run the backend server with:

```bash

npm start

```

Known Issues

As mentioned earlier, this is the developer's first large-scale project, and the file structure may not be optimal. Additionally, there might be some bugs or incomplete features. Please feel free to contribute by submitting issues or pull requests.

Future Improvements

The following enhancements are planned for future releases:

Improved file structure

Enhanced user authentication and authorization

Additional features for employers and job seekers

Thank you for checking out the Job Portal project! If you have any questions or feedback, please don't hesitate to reach out. Happy coding!

Backend:

Admin.js

const { validationResult } = require("express-validator");

const bcryptjs = require("bcryptjs");

const User = require("../models/user");

const Job = require("../models/job");

const Applicant = require("../models/applicant");

const { clearResume } = require("../util/helper");

exports.getStats = (req, res, next) => {

let providerCount;

let jobCount;

let applicantCount;

let seekerCount;

User.find({ \_id: { $ne: req.userId }, role: "Job Provider" })

.countDocuments()

.then((count) => {

providerCount = count;

return User.find({

\_id: { $ne: req.userId },

role: "User",

}).countDocuments();

})

.then((count) => {

seekerCount = count;

return Job.find().countDocuments();

})

.then((count) => {

jobCount = count;

return Applicant.find().countDocuments();

})

.then((count) => {

applicantCount = count;

res.status(200).json({

message: "Successfully fetched stats",

stats: { jobCount, providerCount, applicantCount, seekerCount },

});

});

};

exports.getRecent = (req, res, next) => {

let recentUsers = [];

let recentJobs = [];

User.find({ \_id: { $ne: req.userId } })

.lean()

.sort({ createdAt: -1 })

.limit(3)

.then((users) => {

recentUsers = users;

return Job.find().lean().sort({ createdAt: -1 }).limit(3);

})

.then((jobs) => {

recentJobs = jobs;

res.status(200).json({

message: "Successfully fetched recent stats",

recentUsers,

recentJobs,

});

});

};

exports.getUsers = (req, res, next) => {

User.find({ \_id: { $ne: req.userId } })

.lean()

.then((users) => {

res.status(200).json({

message: "Fetched the list of users",

users: users,

});

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.postUser = (req, res, next) => {

const errors = validationResult(req);

if (!errors.isEmpty()) {

const error = new Error("Validation failed");

error.statusCode = 422;

error.data = errors.array();

throw error;

}

const password = req.body.password;

bcryptjs

.hash(password, 12)

.then((hashedPw) => {

const newUser = new User({ ...req.body, password: hashedPw });

return newUser.save();

})

.then((user) => {

res.status(201).json({ message: "User Added Successfully!" });

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.getUser = (req, res, next) => {

const userId = req.params.userId;

User.findById(userId)

.lean()

.then((user) => {

if (!user) {

const error = new Error("User not found");

error.statusCode = 404;

throw error;

}

res

.status(200)

.json({ message: "Fetched the user Successfully", user: user });

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.editUser = (req, res, next) => {

const userId = req.params.userId;

const errors = validationResult(req);

if (!errors.isEmpty()) {

const error = new Error("Validation failed");

error.statusCode = 422;

error.data = errors.array();

throw error;

}

if (userId === req.userId) {

const error = new Error("Cannot edit the current User");

error.statusCode = 401;

throw error;

}

User.findByIdAndUpdate(userId, req.body, { useFindAndModify: false })

.then((data) => {

if (!data) {

res.status(404).json({

message: `Cannot update user with id=${id}. Maybe user was not found!`,

});

} else

res.status(200).json({ message: "User was updated successfully." });

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.deleteUser = (req, res, next) => {

const userId = req.params.userId;

if (userId === req.userId) {

const error = new Error("Cannot delete the current User");

error.statusCode = 401;

throw error;

}

let jobs;

let role;

let resumes = [];

let applicants = [];

User.findById(userId)

.then((user) => {

if (!user) {

const error = new Error("Cannot delete user. User not found!");

error.statusCode = 404;

throw err;

}

role = user.role;

if (role === "Job Provider") {

jobs = user.jobsPosted;

}

return User.findByIdAndDelete(userId);

})

.then((result) => {

if (role === "Job Provider") {

return Job.deleteMany({ \_id: { $in: jobs } });

}

})

.then((result) => {

if (role === "Job Provider") {

return Applicant.find({ providerId: userId }).then((docs) => {

docs.forEach((doc) => {

resumes.push(doc.resume);

applicants.push(doc.\_id);

});

});

}

if (role === "User") {

return Applicant.find({ userId: userId }).then((docs) => {

docs.forEach((doc) => {

resumes.push(doc.resume);

applicants.push(doc.\_id);

});

});

}

})

.then((result) => {

return Applicant.deleteMany({ \_id: { $in: applicants } });

})

.then((result) => {

resumes.forEach((resume) => clearResume(resume));

res.json({

message: "User record was deleted successfully!",

});

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.getJobs = (req, res, next) => {

Job.find()

.lean()

.then((jobs) => {

res.status(200).json({

message: "Fetched the list of jobs",

jobs: jobs,

});

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.addJob = (req, res, next) => {

const errors = validationResult(req);

if (!errors.isEmpty()) {

const error = new Error("Validation failed");

error.statusCode = 422;

error.data = errors.array();

throw error;

}

let jobId;

const newJob = new Job({ ...req.body, providerId: req.userId });

newJob

.save()

.then((job) => {

jobId = job.\_id;

return User.findById(req.userId);

})

.then((user) => {

user.jobsPosted.push(jobId);

return user.save();

})

.then((result) => {

res.status(201).json({ message: "Job Added Successfully" });

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.getJob = (req, res, next) => {

const jobId = req.params.jobId;

Job.findById(jobId)

.lean()

.then((job) => {

if (!job) {

const error = new Error("Job not found");

error.statusCode = 404;

throw error;

}

res

.status(200)

.json({ message: "Fetched the job Successfully", job: job });

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.editJob = (req, res, next) => {

const jobId = req.params.jobId;

const errors = validationResult(req);

if (!errors.isEmpty()) {

const error = new Error("Validation failed");

error.statusCode = 422;

error.data = errors.array();

throw error;

}

Job.findByIdAndUpdate(jobId, req.body, { useFindAndModify: false })

.then((data) => {

if (!data) {

res.status(404).json({

message: `Cannot update job with id=${id}. Maybe job was not found!`,

});

} else res.status(200).json({ message: "Job was updated successfully." });

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.deleteJob = (req, res, next) => {

const jobId = req.params.jobId;

let providerId;

let resumes = [];

let applicants = [];

Job.findById(jobId)

.then((job) => {

if (!job) {

const error = new Error("Cannot delete job. Job not found!");

error.statusCode = 404;

throw err;

}

providerId = job.providerId;

return Job.findByIdAndDelete(jobId);

})

.then((result) => {

return User.findByIdAndUpdate(

{ \_id: providerId },

{ $pull: { jobsPosted: jobId } }

);

})

.then((result) => {

return Applicant.find({ jobId: jobId }).then((docs) => {

docs.forEach((doc) => resumes.push(doc.resume));

docs.forEach((doc) => applicants.push(doc.\_id));

});

})

.then((result) => {

return Applicant.deleteMany({ \_id: { $in: applicants } });

})

.then((result) => {

resumes.forEach((resume) => clearResume(resume));

res.json({

message: "Job record was deleted successfully!",

});

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

Auth.js

const bcryptjs = require("bcryptjs");

const { validationResult } = require("express-validator");

const jwt = require("jsonwebtoken");

const User = require("../models/user");

const Job = require("../models/job");

exports.signup = (req, res, next) => {

const errors = validationResult(req);

if (!errors.isEmpty()) {

const error = new Error("Validation failed");

error.statusCode = 422;

error.data = errors.array();

throw error;

}

const password = req.body.password;

bcryptjs

.hash(password, 12)

.then((hashedPw) => {

const newUser = new User({ ...req.body, password: hashedPw });

return newUser.save();

})

.then((user) => {

res.status(201).json({ message: "Registered Successfully!" });

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.login = (req, res, next) => {

const errors = validationResult(req);

if (!errors.isEmpty()) {

const error = new Error("Validation Failed");

error.statusCode = 422;

error.data = errors.array();

throw error;

}

const email = req.body.email;

const password = req.body.password;

let loadedUser;

User.findOne({ email: email })

.then((user) => {

if (!user) {

const error = new Error("Email does not exist");

error.statusCode = 401;

throw error;

}

loadedUser = user;

return bcryptjs.compare(password, user.password);

})

.then((isEqual) => {

if (!isEqual) {

const error = new Error("Incorrect Password");

error.statusCode = 401;

throw error;

}

const token = jwt.sign(

{

userId: loadedUser.\_id.toString(),

userName: loadedUser.name,

role: loadedUser.role,

},

"thisistooconfidential",

{ expiresIn: "1h" }

);

res.status(200).json({

message: "Login Successful",

token: token,

});

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

Provider.js

const { validationResult } = require("express-validator");

const fs = require("fs");

const path = require("path");

const Job = require("../models/job");

const Applicant = require("../models/applicant");

const User = require("../models/user");

const { clearResume } = require("../util/helper");

exports.getStats = (req, res, next) => {

let jobsCount = 0;

let applicantsCount = 0;

Job.find({ providerId: req.userId })

.countDocuments()

.then((jobs) => {

jobsCount = jobs;

return Applicant.find({ providerId: req.userId }).countDocuments();

})

.then((applicants) => {

applicantsCount = applicants;

return res.status(200).json({

message: "Successfully fetched the stats",

stats: { jobsCount, applicantsCount },

});

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.getRecents = (req, res, next) => {

Job.find({ providerId: req.userId })

.sort({ createdAt: -1 })

.limit(3)

.lean()

.then((jobs) => {

return res

.status(200)

.json({

message: "Successfully fetched the recent jobs",

recentJobs: jobs,

});

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.getJobs = (req, res, next) => {

Job.find({ providerId: req.userId })

.lean()

.then((jobs) => {

res.status(200).json({

message: "Fetched the list of jobs",

jobs: jobs,

});

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.addJob = (req, res, next) => {

const errors = validationResult(req);

if (!errors.isEmpty()) {

const error = new Error("Validation failed");

error.statusCode = 422;

error.data = errors.array();

throw error;

}

const newJob = new Job({

...req.body,

providerId: req.userId,

});

let jobId;

newJob

.save()

.then((job) => {

jobId = job.\_id;

return User.findById(req.userId);

})

.then((user) => {

user.jobsPosted.push(jobId);

return user.save();

})

.then((result) => {

res.status(201).json({ message: "Job Added Successfully" });

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.getJob = (req, res, next) => {

const jobId = req.params.jobId;

Job.findOne({ \_id: jobId, providerId: req.userId })

.lean()

.then((job) => {

if (!job) {

const error = new Error("Job not found");

error.statusCode = 404;

throw error;

}

res

.status(200)

.json({ message: "Fetched the job Successfully", job: job });

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.editJob = (req, res, next) => {

const jobId = req.params.jobId;

const errors = validationResult(req);

if (!errors.isEmpty()) {

const error = new Error("Validation failed");

error.statusCode = 422;

error.data = errors.array();

throw error;

}

Job.findOneAndUpdate({ \_id: jobId, providerId: req.userId }, req.body, {

useFindAndModify: false,

})

.then((data) => {

if (!data) {

res.status(404).json({

message: `Cannot update job with id=${id}. Maybe job was not found!`,

});

} else res.status(200).json({ message: "Job was updated successfully." });

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.deleteJob = (req, res, next) => {

const jobId = req.params.jobId;

let resumes = [];

let applicants = [];

Job.findOneAndDelete({ \_id: jobId, providerId: req.userId })

.then((data) => {

if (!data) {

const error = new Error("Cannot delete job. Job not found!");

error.statusCode = 404;

throw error;

}

return User.findOneAndUpdate(

{ \_id: req.userId },

{ $pull: { jobsPosted: jobId } }

);

})

.then((result) => {

return Applicant.find({ jobId: jobId, providerId: req.userId }).then(

(docs) => {

docs.forEach((doc) => resumes.push(doc.resume));

docs.forEach((doc) => applicants.push(doc.\_id));

}

);

})

.then((result) => {

return Applicant.deleteMany({ \_id: { $in: applicants } });

})

.then((result) => {

resumes.forEach((resume) => clearResume(resume));

res.json({

message: "Job record was deleted successfully!",

});

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.getApplicantsForJob = (req, res, next) => {

const jobId = req.params.jobId;

const providerId = req.userId;

Applicant.find({

providerId: providerId,

jobId: jobId,

status: { $regex: "Applied", $options: "i" },

})

.populate("userId", "name")

.lean()

.then((applicants) => {

if (!applicants) {

return res

.status(200)

.json({ message: "Looks like no one has applied yet!" });

}

return res.status(200).json({

message: "Successfully fetched the applicants",

applicants: applicants,

});

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.getShortlistsForJob = (req, res, next) => {

const jobId = req.params.jobId;

const providerId = req.userId;

Applicant.find({

providerId: providerId,

jobId: jobId,

status: { $regex: "Shortlisted", $options: "i" },

})

.populate("userId", "name email")

.lean()

.then((applicants) => {

if (!applicants) {

return res

.status(200)

.json({ message: "Looks like no one has been shortlisted yet!" });

}

return res.status(200).json({

message: "Successfully fetched the applicants",

shortlists: applicants,

});

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.getApplicantResume = (req, res, next) => {

const applicantId = req.params.applicantItemId;

Applicant.findOne({ \_id: applicantId, providerId: req.userId })

.lean()

.then((applicant) => {

if (!applicant) {

return res.status(404).json({ message: "Applicant not found" });

}

const resumeFile = applicant.resume;

const resumePath = path.join(resumeFile);

fs.readFile(resumePath, (err, data) => {

if (err) {

return next(err);

}

res.setHeader("Content-type", "application/pdf");

res.send(data);

});

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.shortlistApplicant = (req, res, next) => {

const applicantItemId = req.params.applicantItemId;

Applicant.findById({ \_id: applicantItemId })

.then((applicant) => {

if (!applicant) {

return res.status(401).json({ message: "Applicant not found" });

}

if (applicant.providerId.toString() !== req.userId.toString()) {

const error = new Error("You are unauthorized to do the action!");

error.statusCode = 401;

throw error;

}

if (applicant.status === "Shortlisted") {

return res.status(409).json({ message: "Already shortlisted!" });

}

applicant.status = "Shortlisted";

return applicant.save();

})

.then((result) => {

res.status(200).json({ message: "Shortlisted the candidate!" });

})

.catch((err) => {

// console.log(err);

next(err);

});

};

exports.rejectApplicant = (req, res, next) => {

const applicantItemId = req.params.applicantItemId;

Applicant.findById(applicantItemId)

.then((applicant) => {

if (!applicant) {

return res.status(404).json({ message: "Applicant not found!" });

}

if (req.userId.toString() !== applicant.providerId.toString()) {

const error = new Error("You are unauthorized to do the action!");

error.statusCode = 401;

throw error;

}

clearResume(applicant.resume);

return Applicant.findByIdAndDelete(applicantItemId);

})

.then((result) => {

return res

.status(200)

.json({ message: "Applicant rejected successfully!" });

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

User.js

// const { validationResult } = require("express-validator");

const Job = require("../models/job");

const Applicant = require("../models/applicant");

const { clearResume } = require("../util/helper");

const { dateFormatter } = require("../util/helper");

exports.getAvailableJobs = (req, res, next) => {

let appliedJobs = [];

Applicant.find({ userId: req.userId })

.lean()

.then((applicants) => {

applicants.forEach((applicant) => appliedJobs.push(applicant.jobId));

return Job.find({ \_id: { $not: { $in: appliedJobs } } }).lean();

})

.then((jobs) => {

res.status(200).json({

message: "Fetched the list of jobs",

jobs: jobs,

});

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.getAppliedJobs = (req, res, next) => {

let appliedJobs = [];

const statusMap = new Map();

Applicant.find({ userId: req.userId })

.lean()

.then((applicants) => {

// console.log(applicants);

applicants.forEach((applicant) => {

appliedJobs.push(applicant.jobId);

statusMap.set(applicant.jobId.toString(), applicant.status);

});

return Job.find({ \_id: { $in: appliedJobs } }).lean();

})

.then((jobsApplied) => {

jobsApplied.forEach((applied) => {

applied.status = statusMap.get(applied.\_id.toString());

});

res.status(200).json({

message: "Fetched the list of jobs",

jobsApplied: jobsApplied,

});

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};

exports.applyJob = (req, res, next) => {

if (!req.file) {

const err = new Error("Resume not Found");

err.statusCode = 422;

throw err;

}

const jobId = req.params.jobId;

const userId = req.userId;

const providerId = req.body.providerId;

const resume = req.file.path.replace("\\", "/");

let status;

Applicant.findOne({ jobId: jobId, userId: userId })

.then((applicant) => {

if (applicant) {

clearResume(resume);

return res

.status(409)

.json({ message: "You have already applied for the job!" });

}

status = "Applied on " + dateFormatter();

const newApplicant = new Applicant({

jobId: jobId,

userId: userId,

resume: resume,

status: status,

providerId: providerId,

});

return newApplicant.save();

})

.then((result) => {

res.status(201).json({ message: "Successfully applied for the job!" });

})

.catch((err) => {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

});

};