

BiggerProject4

Develop the backend for a job portal using Node JS, Express JS, MongoDB. Perform important functionalities such as add new job, update job, apply for job, approve job application, etc.,

Procedure:

- Create a folder and open that folder in terminal
- Enter the following commands

```
PS E:\AWT LAB> cd .\myfolder\  
PS E:\AWT LAB\myfolder> npm init -y  
Wrote to E:\AWT LAB\myfolder\package.json:  
  
{  
  "name": "myfolder",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC"  
}  
  
PS E:\AWT LAB\myfolder> npm i express mongoose cors
```

- The following files and folder will be created in our folder

```
myfolder  
├── node_modules  
├── package-lock.json  
└── package.json
```

- Now create folder with name 'server' and create server.js inside it

server.js

```
const express = require('express');  
const mongoose = require('mongoose');  
const cors = require('cors');  
  
const app = express();  
const port = 5000;
```

```

// Use the cors middleware
app.use(cors())

// Connect to MongoDB
mongoose.connect('mongodb://127.0.0.1/job_portal', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});

// Define MongoDB schemas and models
const jobSchema = new mongoose.Schema({
  title: String,
  description: String,
  company: String,
  postedBy: String,
  applications: [{ applicant: String, status: String }],
});

const Job = mongoose.model('Job', jobSchema);

// Middleware to parse JSON requests
app.use(express.json());

// Routes

//Get all jobs
app.get('/jobs', async(req, res) => {
  try{
    const jobs = await Job.find();
    res.status(200).json({message: 'Jobs fetched successfully', jobs: jobs});
  }catch(error){
    res.status(500).json({ error: 'Internal Server Error: '+error });
  }
})

// Add a new job
app.post('/jobs/add', async (req, res) => {
  try {
    const { title, description, company, postedBy } = req.body;
    const job = new Job({ title, description, company, postedBy });
    await job.save();
    res.status(201).json({ message: 'Job added successfully', job });
  } catch (error) {
    res.status(500).json({ error: 'Internal Server Error' });
  }
});

// Update a job
app.put('/jobs/:jobId', async (req, res) => {
  try {
    const { jobId } = req.params;
    const { title, description, company } = req.body;
    const job = await Job.findByIdAndUpdate(

```

```

        jobId,
        { title, description, company },
        { new: true }
    );
    res.status(200).json({ message: 'Job updated successfully', job });
} catch (error) {
    res.status(500).json({ error: 'Internal Server Error' });
}
});

//Delete a job
app.delete('/jobs/delete/:jobId', async (req, res) => {
    try {
        const { jobId } = req.params;
        await Job.findByIdAndDelete(jobId);
        res.status(204).end(); // No content after successful deletion
    } catch (error) {
        console.error('Error deleting job:', error.message);
        res.status(500).json({ error: 'Internal Server Error' });
    }
});

// Apply for a job
app.post('/jobs/:jobId/apply', async (req, res) => {
    try {
        const { jobId } = req.params;
        const { applicant } = req.body;
        const job = await Job.findById(jobId);
        if (!job) {
            return res.status(404).json({ error: 'Job not found' });
        }
        job.applications.push({ applicant, status: 'Pending' });
        await job.save();
        res.status(200).json({ message: 'Application submitted successfully' });
    } catch (error) {
        res.status(500).json({ error: 'Internal Server Error' });
    }
});

// Approve a job application
app.put('/jobs/:jobId/approve/:applicationId', async (req, res) => {
    try {
        const { jobId, applicationId } = req.params;
        const job = await Job.findById(jobId);
        if (!job) {
            return res.status(404).json({ error: 'Job not found' });
        }
        const application = job.applications.id(applicationId);
        if (!application) {
            return res.status(404).json({ error: 'Application not found' });
        }
        application.status = 'Approved';
        await job.save();
        res.status(200).json({ message: 'Application approved successfully' });
    }

```

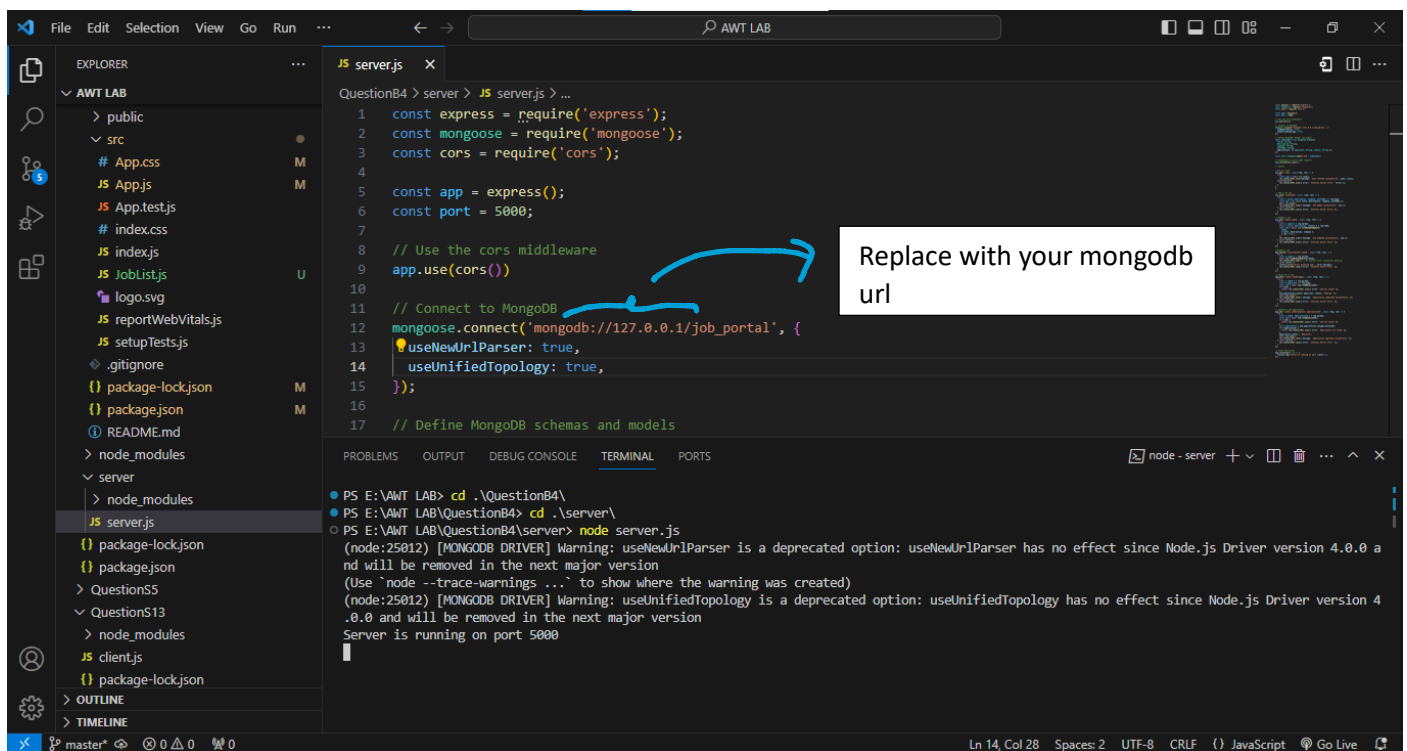
```

} catch (error) {
  res.status(500).json({ error: 'Internal Server Error' });
}
});

// Start the server
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});

```

- Run server.js



- Now create client folder using '***npx create-react-app client***' command
- Replace App.js code with below code

App.js

```
import React from 'react';
import JobList from './JobList';
import './App.css'

const App = () => {

  return (
    <div className='container'>
      <JobList />
    </div>
  );
};

export default App;
```

- Create JobList.js file in src folder

JobList.js

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

const JobList = () => {
  const [jobs, setJobs] = useState([]);
  const [selectedJob, setSelectedJob] = useState(null);
  const [updateFormData, setUpdateFormData] = useState({
    title: '',
    description: '',
    company: '',
    postedBy: '',
  });
};

useEffect(() => {
  // Fetch jobs from the backend when the component mounts
  const fetchJobs = async () => {
    try {
      const response = await axios.get('http://localhost:5000/jobs');
      setJobs(response.data.jobs);
    } catch (error) {
      console.error('Error fetching jobs:', error.message);
    }
  };

  fetchJobs();
}, []);
```

```

const applyForJob = async (jobId, applicant) => {
  try {
    await axios.post(`http://localhost:5000/jobs/${jobId}/apply`, { applicant });
    // Refresh the job list after applying
    const response = await axios.get('http://localhost:5000/jobs');
    setJobs(response.data.jobs);
  } catch (error) {
    console.error('Error applying for the job:', error.message);
  }
};

const approveApplication = async (jobId, applicationId) => {
  try {
    await axios.put(`http://localhost:5000/jobs/${jobId}/approve/${applicationId}`);
    // Refresh the job list after approving
    const response = await axios.get('http://localhost:5000/jobs');
    setJobs(response.data.jobs);
  } catch (error) {
    console.error('Error approving the application:', error.message);
  }
};

const createJob = async (e) => {
  try {
    e.preventDefault()
    await axios.post('http://localhost:5000/jobs/add', updateFormData);
    // Refresh the job list after creating
    clearFormDataAndSelection()
    const response = await axios.get('http://localhost:5000/jobs');
    setJobs(response.data.jobs);
  } catch (error) {
    console.error('Error creating job:', error.message);
  }
};

const updateJob = async (e) => {
  try {
    e.preventDefault()
    await axios.put(`http://localhost:5000/jobs/${selectedJob._id}`, updateFormData);
    // Refresh the job list after updating
    clearFormDataAndSelection()
    const response = await axios.get('http://localhost:5000/jobs');
    setJobs(response.data.jobs);
    setSelectedJob(null);
  } catch (error) {
    console.error('Error updating job:', error.message);
  }
};

const deleteJob = async (jobId) => {
  try {
    await axios.delete(`http://localhost:5000/jobs/delete/${jobId}`);
    // Refresh the job list after deleting
    const response = await axios.get('http://localhost:5000/jobs');
  }
};

```

```

    setJobs(response.data.jobs);
  } catch (error) {
    console.error('Error deleting job:', error.message);
  }
};

const handleFormDataChange = (e) => {
  setUpdateFormData({ ...updateFormData, [e.target.name]: e.target.value });
};

const openUpdateForm = (job) => {
  setSelectedJob(job);
  setUpdateFormData({
    title: job.title,
    description: job.description,
    company: job.company,
    postedBy: job.postedBy,
  });
};

const clearFormDataAndSelection = () => {
  setSelectedJob(null);
  setUpdateFormData({
    title: '',
    description: '',
    company: '',
    postedBy: '',
  });
};

return (
  <div>
    <h1>Job List</h1>
    {jobs.map((job) => (
      <div key={job._id} className='job'>
        <h2>{job.title}</h2>
        <p>{job.description}</p>
        <p>Company: {job.company}</p>
        <p>Posted By: {job.postedBy}</p>
        <button onClick={() => applyForJob(job._id, 'John Doe')}>Apply</button>
        <button onClick={() => openUpdateForm(job)}>Update</button>
        <button onClick={() => deleteJob(job._id)}>Delete</button>
        {job.applications.map((application) => (
          <div key={application._id}>
            <hr></hr>
            <p>Applicant: {application.applicant}</p>
            <p>Status: {application.status}</p>
            {application.status === 'Pending' && (
              <button onClick={() => approveApplication(job._id, application._id)}>
                Approve Application
              </button>
            )}
          </div>
        ))}
      </div>
    ))}
  </div>
)

```

```

    </div>
  )}

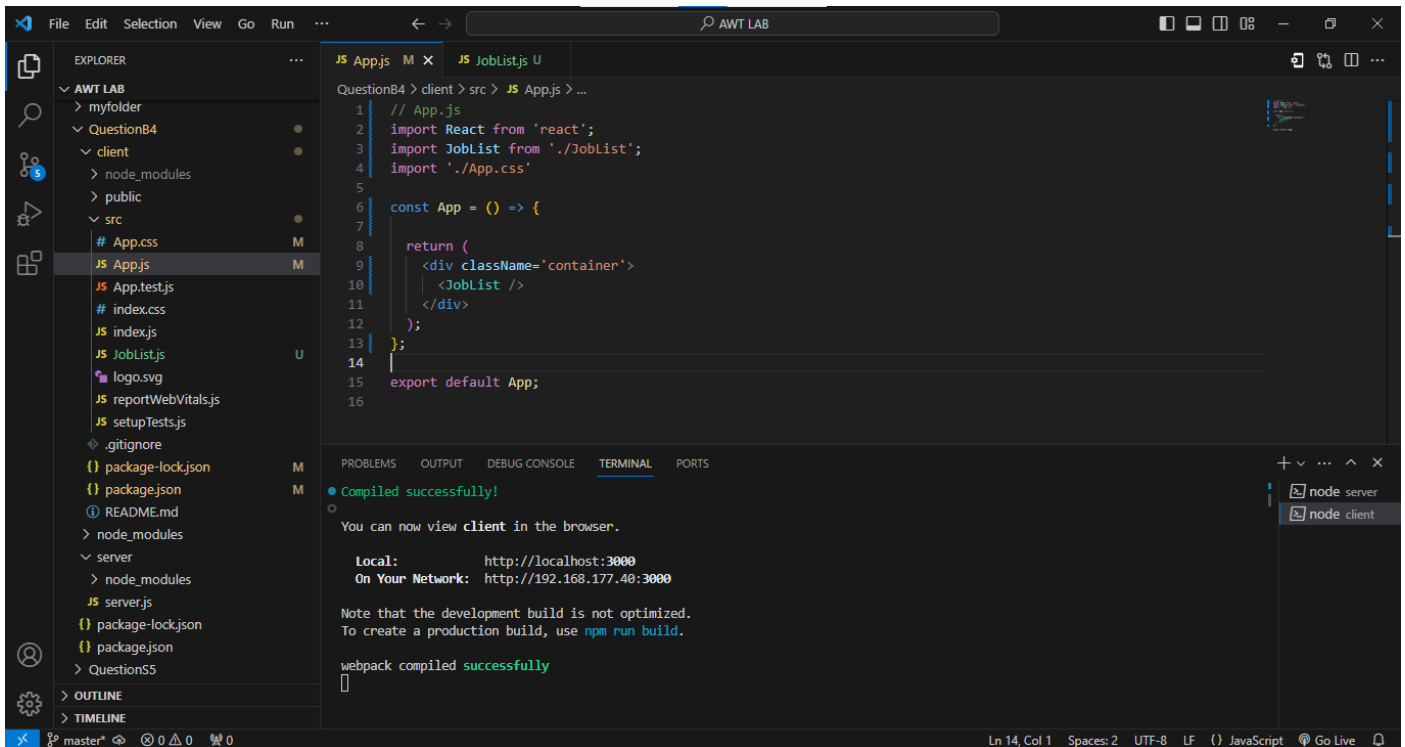
  <div className="form">
    <h2>{selectedJob ? 'Update Job' : 'Create Job'}</h2>
    <form onSubmit={selectedJob ? updateJob : createJob}>
      <label>Title:</label>
      <input
        type="text"
        name="title"
        value={updateFormData.title}
        onChange={handleFormDataChange}
        required
      />
      <label>Description:</label>
      <textarea
        name="description"
        value={updateFormData.description}
        onChange={handleFormDataChange}
        required
      />
      <label>Company:</label>
      <input
        type="text"
        name="company"
        value={updateFormData.company}
        onChange={handleFormDataChange}
        required
      />
      <label>Posted By:</label>
      <input
        type="text"
        name="postedBy"
        value={updateFormData.postedBy}
        onChange={handleFormDataChange}
        required
      />
      <button type="submit">{selectedJob ? 'Update' : 'Create'}</button>
      <button type="button" onClick={clearFormDataAndSelection}>
        Clear
      </button>
    </form>
  </div>

</div>
);
};

export default JobList;

```


- Run client using ***'npm start'*** command



```
QuestionB4 > client > src > JS Appjs > ...
1 // App.js
2 import React from 'react';
3 import JobList from './JobList';
4 import './App.css'
5
6 const App = () => {
7
8   return (
9     <div className='container'>
10       <JobList />
11     </div>
12   );
13 };
14
15 export default App;
16
```

Compiled successfully!

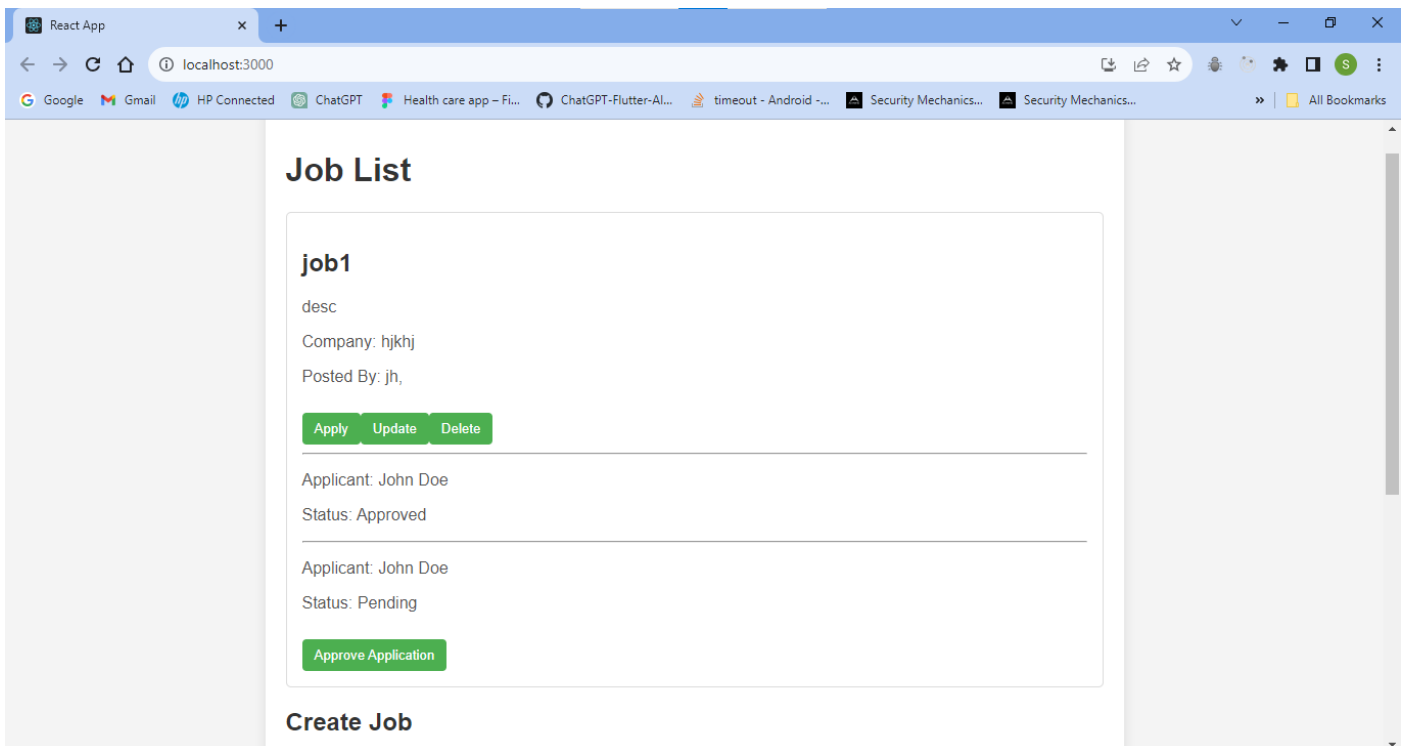
You can now view client in the browser.

Local: http://localhost:3000
On Your Network: http://192.168.177.40:3000

Note that the development build is not optimized.
To create a production build, use `npm run build`.

webpack compiled successfully

Output:



Node.js Static File Server

React App

+

localhost:3000

GoogleGmailHP ConnectedChatGPTHealth care app – Fi...ChatGPT-Flutter-AL...timeout - Android ~...Security Mechanics...Security Mechanics...All Bookmarks

Applicant: John Doe

Status: Approved

Create Job

Title:

Description:

Company:

Posted By:

Create

Clear