JobConnect - MERN Stack Job Application Portal

1. Project Overview

Project Title: JobConnect
Description: JobConnect is a MERN stack job portal designed to connect job seekers with employers. It features role-based access for Admins, Job Seekers, and Companies. Employers can post job listings, which are published upon admin approval. Users upload their resumes and profile pictures via Cloudinary, and all core interactions (posting, applying, approving) are managed via secure APIs. Additionally, JobConnect includes automated email notifications using Nodemailer for registrations, application events, and status updates. Job seekers can only apply for jobs that match their qualifications or job requirements as specified by the employer. A qualification flag (Qualified / Not Qualified) is shown to the job seeker before application and is stored in the application record upon submission.

Duration: 5 days
Developer: Silas HAKUZWIMANA
Stack: MongoDB, Express.js, Vite + React.js, Node.js, Cloudinary, TailwindCSS, Nodemailer

2. Key Features by Role

2.1 Job Seeker

- Register/Login (2FA enabled)
- Upload profile picture and resume
- View and apply for approved jobs (only if requirements match)
- View application history/status
- Bookmark jobs
- See qualification flag for each job (Qualified / Not Qualified) before applying
- Qualification flag is submitted and saved along with the application
- Receive email notifications on registration, application submission, and status updates

2.2 Company (Employer)

- Register (default role is "jobseeker")
- Admin approval required to activate as a company
- Post, edit, and delete jobs
- Define job requirements
- View own job listings
- View applications on posted jobs
- Receive email notification upon approval and new applications

2.3 Admin

- Login
- Approve or reject new companies
- Approve or reject job listings
- View and manage all users, jobs, and applications
- View dashboard statistics
- Send custom email notifications to users (optional)

3. System Architecture

3.1 Frontend (React.js)

- SPA with protected routes using React Router
- Global state management using Context API
- TailwindCSS for styling
- Flag-based job display (Qualified / Not Qualified) before applying

3.2 Backend (Express.js)

- RESTful API architecture
- JWT-based authentication with 2FA during login
- Role-based access middleware
- Nodemailer integration for email notifications
- Cloudinary integration for file uploads
- Validation logic for job application based on job requirements
- Flag setting on job view and submission to indicate qualification

3.3 Database (MongoDB)

- Collections: Users, Jobs, Applications
- Mongoose for schema validation and relations

4. Folder Structure

Backend

```
backend/
├── config/
│   └── db.js
│   └── cloudinary.js
│   └── mailer.js
├── controllers/
├── middlewares/
├── models/
├── routes/
├── utils/
├── server.js
└── .env
```

Frontend

```
frontend/
├── public/
├── src/
│   ├── assets/
│   ├── components/
│   ├── pages/
│   ├── context/
│   ├── services/
│   ├── App.jsx
│   └── main.jsx
├── tailwind.config.js
└── package.json
```

5. Database Schemas

User Schema

```
{
  name: String,
  email: String,
  password: String,
  role: { type: String, enum: ['admin', 'jobseeker', 'company'], default:
'jobseeker' },
  isApproved: Boolean,
  profileImage: String,
  resume: String,
  otp: String, // for 2FA
  otpExpiresAt: Date,
  qualifications: [String] // For job match validation
}
```

Job Schema

```
{
  title: String,
  description: String,
  company: ObjectId (User),
  location: String,
  type: String,
  salaryRange: String,
  requirements: [String],
  isApproved: Boolean,
  createdAt: Date
}
```

Application Schema

```
{
  job: ObjectId (Job),
  applicant: ObjectId (User),
  resumeUrl: String,
  status: { type: String, enum: ['pending', 'reviewed', 'rejected'] },
  appliedAt: Date,
```

```
  isQualified: Boolean // Set before submission if applicant meets job
requirements
}
```

6. APIs Overview

Auth

- POST `/api/auth/register`
- POST `/api/auth/login` (with OTP verification)
- POST `/api/auth/verify-otp`

Job

- GET `/api/jobs` (public)
- POST `/api/company/jobs` (company)
- PUT `/api/admin/jobs/:id` (admin)

Application

- POST `/api/jobs/:id/apply` (jobseeker, only if requirements match)
- GET `/api/applications/mine` (jobseeker)
- PUT `/api/applications/:id/status` (admin or company)

User

- POST `/api/users/upload-profile`
- POST `/api/users/upload-resume`
- PUT `/api/users/update-qualifications`

Admin

- GET `/api/admin/users`
- PUT `/api/admin/approve-user/:id`
- POST `/api/admin/send-email`

7. Cloudinary Integration

- All files (profile pictures, resumes) uploaded via `multer-storage-cloudinary`
- URLs are stored in MongoDB

8. Security & Middleware

- Password hashing using `bcrypt`
- JWT authentication
- OTP-based 2FA login
- Middleware for verifying token and role-based authorization
- Middleware to validate job application eligibility based on requirements
- Flagging system for application qualification, set before and stored after submission

9. Deployment Plan

Frontend

- Vite or Create React App build
- Deployed on Vercel or Netlify

Backend

- Deployed on Render or Cyclic
- MongoDB Atlas for database
- Cloudinary for file storage
- Email service via Nodemailer + SMTP or Gmail

10. Testing & QA

- Manual testing of all CRUD operations
- Test job application flow end-to-end
- Check uploads for resume/profile image
- Verify email delivery and OTP flow
- Test job seeker eligibility enforcement
- Verify qualification flags for applied jobs (pre and post submission)

11. Future Improvements

- Add real-time notifications
- Messaging system between job seeker and employer
- Admin dashboard analytics with charts
- Resume builder tool
- In-app notification system for status updates
- AI-based job matching engine

12. Conclusion

JobConnect is a complete, real-world job portal designed using modern web technologies. It’s secure, scalable, and fulfills the typical features expected in a professional job application platform. The system enforces role-based access, media storage, 2FA login, and administrator moderation. With dynamic job requirement checks and qualification flags, it ensures that only suitable candidates apply, increasing success rates for both seekers and companies.

13. References

- https://reactjs.org
- https://expressjs.com
- https://mongoosejs.com
- https://www.mongodb.com
- https://cloudinary.com
- https://tailwindcss.com
- https://nodemailer.com