

Right Resource Fit

Description

Right Resource Fit is an intuitive web-based platform designed to bridge the gap between businesses and skilled professionals. It empowers companies to efficiently find, evaluate, and onboard the most suitable candidates for their project-specific needs, while also providing individuals with an avenue to showcase their talents and find opportunities tailored to their skill sets. Using advanced filtering, skill-based matching, and availability tracking, the platform aims to reduce the time and effort required for resource allocation.

Key features include a dynamic matching algorithm, user-friendly profiles with integrated portfolios, and a seamless dashboard for managing applications, projects, and notifications.

Purpose

- The purpose of Right Resource Fit is to simplify and enhance the resource-matching process for both businesses and freelancers by:
- **Optimizing Talent Discovery:** Enable businesses to identify the right candidates quickly based on skills, experience, and real-time availability.
- **Empowering Professionals:** Provide individuals with tools to present their expertise and gain access to projects aligned with their capabilities.
- **Streamlining Processes:** Replace traditional, time-intensive hiring workflows with a smart, technology-driven approach.
- **Encouraging Collaboration:** Foster connections between businesses and professionals, facilitating a mutually beneficial ecosystem.

By addressing inefficiencies in traditional recruitment and resource allocation processes, Right Resource Fit aims to become a go-to platform for short-term hiring and project-based collaboration.

Getting Started

Prerequisites

Before setting up and running the application, ensure the following software and tools are installed on your system:

- Node.js and npm
Download and install from Node.js official website.
npm comes bundled with Node.js.
- MongoDB
Install and set up MongoDB locally or use a cloud-hosted solution like MongoDB Atlas.
- SMTP Service
Configure an SMTP service (e.g., Gmail SMTP) for sending OTPs and notifications.
- IDE

Use a development environment like Visual Studio Code for code editing and debugging.

Installation

Follow these steps to set up the project:

- **Clone the Repository**
git clone https://github.com/springboardmentor106/RightResourceFit_Infosys_Internship_Oct2024_Team_02.git
cd right-resource-fit
- **Install Dependencies**
npm install

Running the Application

- Start the Backend
Navigate to the backend directory:
cd backend
Run the backend server (ensure the backend runs on a specific port, e.g., 1000):
nodemon server.js
- Start the Frontend
Navigate to the frontend directory:
cd frontend
Run the development server:
npm start

Accessing the Application

- Frontend: Visit <http://localhost:3000> (or the port specified during the setup).
- Backend API: Accessible at <http://localhost:1000> for testing endpoints.

Environment Setup for Right Resource Fit

To run the project seamlessly, you need to set up environment variables for both the backend and frontend. Create a .env file in the root directory of the backend project and add the required configurations as shown below.

Required Environment Variables

Variable Name	Description	Example Value
USER_DB_URL	Connection string for the user database.	mongodb://localhost:27017/users
JOB_DB_URL	Connection string for the job postings database.	mongodb://localhost:27017/jobs
NOTIFICATION_DB_URL	Connection string for the notifications database.	mongodb://localhost:27017/notifications
APPLICATION_DB_URL	Connection string for the applications database.	mongodb://localhost:27017/applications
JWT_SECRET	Secret key for signing JWT tokens.	supersecretkey123
SMTP_HOST	SMTP host for sending emails.	smtp.gmail.com
SMTP_PORT	SMTP port for the email server.	587
SMTP_USER	Email address used for sending emails.	your-email@example.com
SMTP_PASS	Password or app-specific password for the email.	your-email-password

Example .env Configuration

Here's an example .env file for your project:

MongoDB URLs

USER_DB_URL=mongodb://localhost:27017/users
JOB_DB_URL=mongodb://localhost:27017/jobs
NOTIFICATION_DB_URL=mongodb://localhost:27017/notifications
APPLICATION_DB_URL=mongodb://localhost:27017/applications

JWT Secret

JWT_SECRET=supersecretkey123

SMTP Configuration

SMTP_HOST=smtp.gmail.com

SMTP_PORT=587

SMTP_USER=your-email@example.com

SMTP_PASS=your-email-password

Steps to Set Up the Environment

- Create a .env file in the root of your backend directory.
- Copy the example configuration above and paste it into the .env file.
- Replace placeholder values (like your-email@example.com and your-email-password) with your actual configuration.

Additional Notes

- For cloud-hosted MongoDB (e.g., MongoDB Atlas), replace mongodb://localhost:27017/<db> with your cluster's connection string.
- If you're using a third-party email service, adjust the SMTP_HOST, SMTP_PORT, SMTP_USER, and SMTP_PASS accordingly.
- Keep the .env file secure and do not commit it to version control. Add .env to your .gitignore file.

Folder Structure

A well-organized folder structure helps maintain clarity and efficiency in development. Below is a suggested folder structure for your Right Resource Fit project, leveraging the MERN stack.

```
right-resource-fit/
├── backend/
│   ├── src/
│   │   ├── config/      # Config files (db, mailer)
│   │   ├── controllers/ # Business logic
│   │   ├── models/      # MongoDB schemas
│   │   ├── routes/      # API endpoints
│   │   └── server.js     # Backend entry point
│   ├── package.json
│   └── .env
├── frontend/
│   ├── public/          # Static files (HTML, assets)
│   ├── src/
│   │   ├── components/  # Reusable React components
│   │   ├── pages/       # Page-level components
│   │   ├── App.js       # Main React component
│   │   └── index.js      # Frontend entry point
│   ├── package.json
│   └── .env
├── .gitignore
├── README.md
└── LICENSE
```

Key Features

- Dynamic Matching Algorithm

Matches candidates to jobs based on skills, experience, and availability.

- User Profiles

Comprehensive profiles for users, showcasing their skills, portfolios, and work history.

- Job Posting and Management

Businesses can post jobs with specific requirements, manage active listings, and track applications.

- Application Tracking

Real-time application tracking for both candidates and businesses.

- Notifications System

Email and in-app notifications for job updates, application status, and reminders.

- Secure Authentication

User authentication using JWT for secure login and session management.

- Responsive Design

Frontend designed to be mobile and desktop friendly for a seamless user experience.

- Admin Dashboard

Admin tools to monitor platform activity, manage users, and generate reports.

Project Components

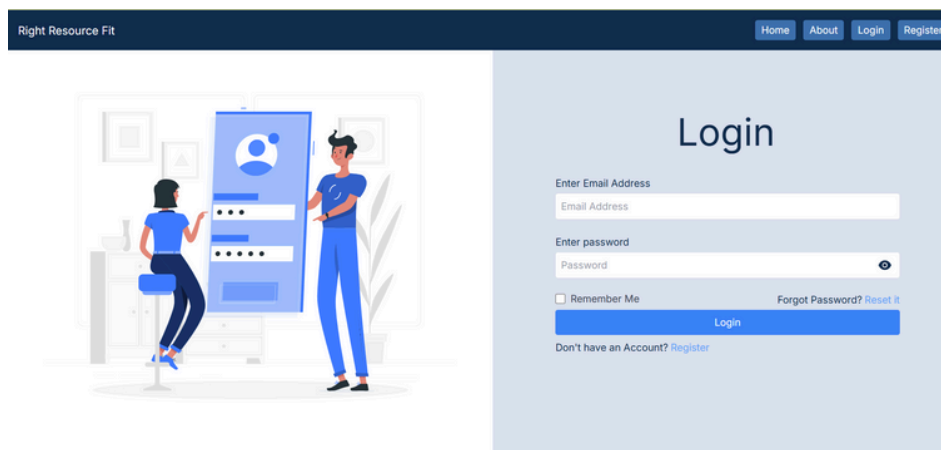
Here's a list of key components in the Right Resource Fit project, along with brief descriptions of their functionality:

Login Component

Description: Manages the user login process, handling inputs for email and password.

Key Features:

- Password Visibility Toggle: Allows users to show/hide their password for easy input.
- 'Remember Me' Functionality: Saves user login credentials (using cookies or localStorage) for subsequent visits without needing to re-enter credentials.
- Form Validation: Ensures that the user provides valid credentials before submission.
- Error Handling: Displays error messages in case of incorrect credentials.

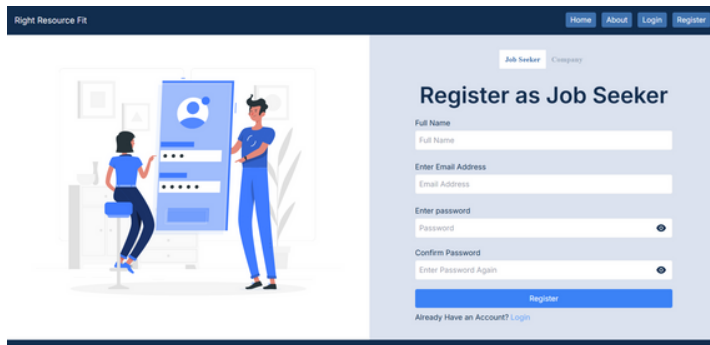


Register Component

Provides signup functionality for both job seekers (freelancers) and businesses (companies).

Key Features:

- User Role Selection: Allows users to select whether they are a job seeker or a company.
- Form Fields: Includes fields for email, password, name, and additional details (e.g., portfolio for job seekers, company name for businesses).
- Validation: Ensures that all necessary fields are filled out correctly and securely.
- Submit & Success Handling: On successful registration, redirects users to their appropriate dashboard (job seeker or business).



Right Resource Fit

Home About Login Register

Job Seeker Company

Register as Job Seeker

Full Name
Full Name

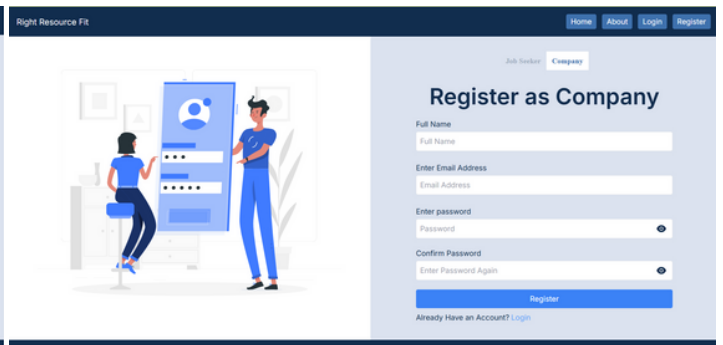
Enter Email Address
Email Address

Enter password
Password

Confirm Password
Enter Password Again

Register

Already Have an Account? [Login](#)



Right Resource Fit

Home About Login Register

Job Seeker Company

Register as Company

Full Name
Full Name

Enter Email Address
Email Address

Enter password
Password

Confirm Password
Enter Password Again

Register

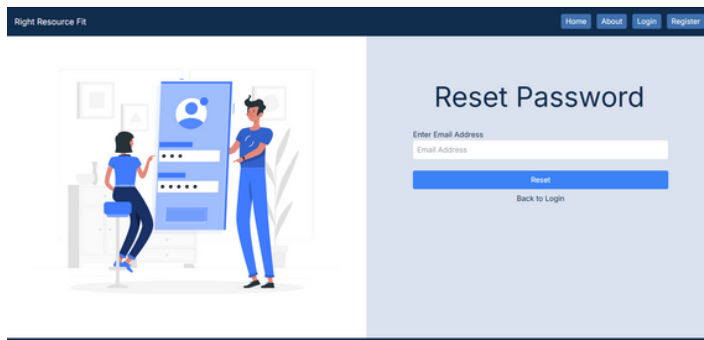
Already Have an Account? [Login](#)

Reset Password & OTP Component

Description: Implements the functionality for password recovery using an OTP sent via email.

Key Features:

- OTP Generation: Generates a one-time password (OTP) and sends it to the user's email via Gmail SMTP.
- OTP Validation: Users input the OTP received in their email to verify their identity.
- Password Reset: Allows users to set a new password after OTP verification.
- Security: Ensures the OTP is valid for a limited time and can only be used once for safety.



Right Resource Fit

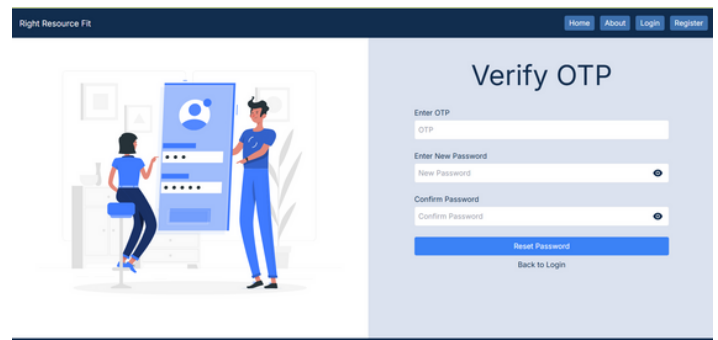
Home About Login Register

Reset Password

Enter Email Address
Email Address

Reset

[Back to Login](#)



Right Resource Fit

Home About Login Register

Verify OTP

Enter OTP
OTP

Enter New Password
New Password

Confirm Password
Confirm Password

Reset Password

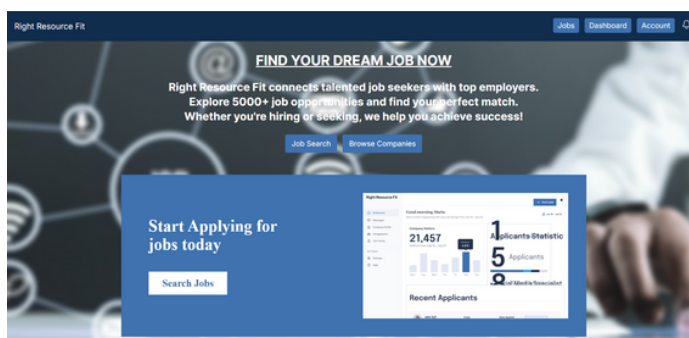
[Back to Login](#)

Homepage Component

Serves as the landing page, showcasing platform features and providing navigation options.

Key Features:

- Call-to-Action Buttons: Encourages users to sign up or explore further.
- Feature Highlights: Briefly outlines the platform's benefits.
- Footer: Contains quick links and contact details.
- Navbar: Contains the links of Dashboard, Account and Logout



Right Resource Fit

Jobs Dashboard Account

FIND YOUR DREAM JOB NOW

Right Resource Fit connects talented job seekers with top employers.
Explore 5000+ job opportunities and find your perfect match.
Whether you're hiring or seeking, we help you achieve success!

[Job Search](#) [Browse Companies](#)

Start Applying for jobs today

[Search Jobs](#)

Applicant Statistics

21,457 Applicants

1 Applicants

5 Applicants

Recent Applicants

Jobseeker Homepage



Right Resource Fit

Post Jobs Dashboard Account

START POSTING JOBS NOW

Right Resource Fit connects talented job seekers with top employers.
Explore 5000+ job opportunities and find your perfect match.
You're Hiring, we help you achieve success!

[Post Job](#)

Start Hiring today

[Post Jobs](#)

Applicant Statistics

21,457 Applicants

1 Applicants

5 Applicants

Recent Applicants

Company Homepage

Jobseeker Profile Component

Description: Allows job seekers to create and update their profiles.

Key Features:

- Profile Image Upload: Users can upload and preview a profile image.
- Editable Fields: Supports editing personal details like name, email, phone, and address.
- Validation: Ensures all fields are correctly filled.
- Success Notification: Displays confirmation upon successful updates.

Right Resource Fit

[Post Jobs](#)
[Dashboard](#)
[Account](#)

Dashboard

Messages

Profile

All Applications

My Schedule

Settings

Help Center

Basic Info

This is the user info you can update anytime

User Profile

This image will be shown publicly as the user profile

Click to replace or drag and drop

SVG, PNG, JPG or GIF (max. 400 x 400px)

Preview

User Name

Manas Raj

Enter Email Address

manasraj1357@gmail.com

Phone Number

9155557919

Date of Birth

Dashboard Component

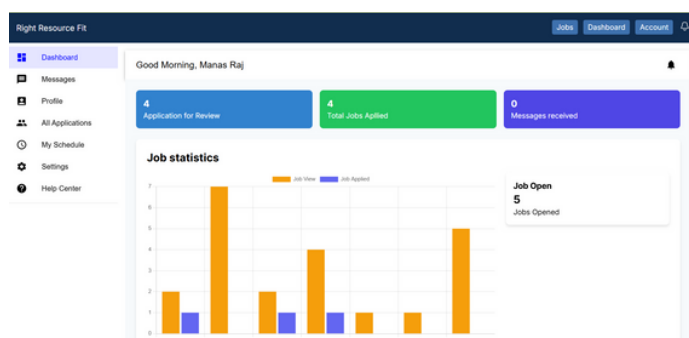
Provides a personalized dashboard for users based on their role.

Key Features:

- Role-Specific Content: Displays job opportunities for seekers and application summaries for businesses.
- Notifications: Alerts users to new messages, updates, or application statuses.
- Quick Actions: Includes shortcuts to frequently used features like profile updates or job posting.



Company Dashboard



Jobseeker Dashboard

Job List Component

Displays a list of jobs for seekers to browse and apply to.

Key Features:

- Filters: Allows filtering jobs by category, location, or type (remote/on-site).
- Pagination: Ensures seamless navigation through large job lists.

Right Resource Fit

Enter job title | Enter location | Enter skills (comma-separated) | Search | Clear All

Salary Range

Min | Max

Job type

☐ All (2567)
☐ Full-Time (450)
☐ Part-Time (145)
☐ Internship (65)
☐ Contract (12)

Work mode

☐ On-site
☐ Remote (180)
☐ Hybrid (200)

AutoCAD Designer

Full-Time | Salary: 30000 - 45000

Company | Remote...

View Details | Apply Now

Circuit designer

Full-Time | Salary: 76000 - 100000

Company | Mumbai...

View Details | Apply Now

Software Developer

Full-Time | Salary: 80000 - 100000

Company | Delhi...

View Details | Apply Now

Adobe Designer

Full-Time | Salary: 39000 - 56000

Company | Mumbai...

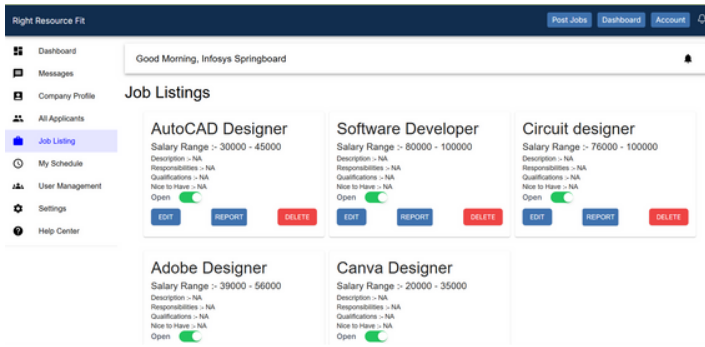
View Details | Apply Now

Job Listing Component

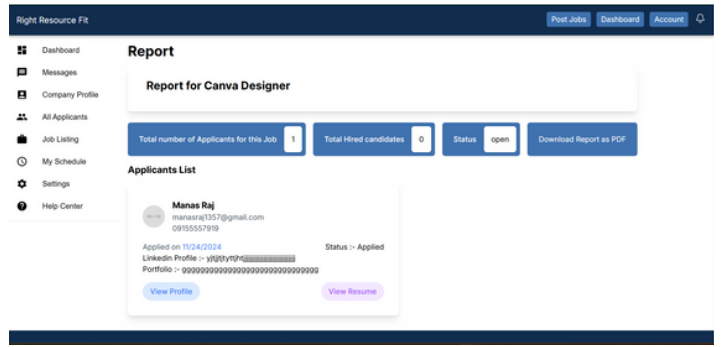
Displays all of the jobs for Company to Edit, Delete and View Report

Key Features:

- Pagination: Ensures seamless navigation through large job lists.
- Delete: Company able to delete the Job.
- Edit: Company can edit their Job posting.
- View Report: Company can view reports of applications for a particular Job post.
- Job Status Change: Company can change the Job Status to Open or Close



Company Joblisting



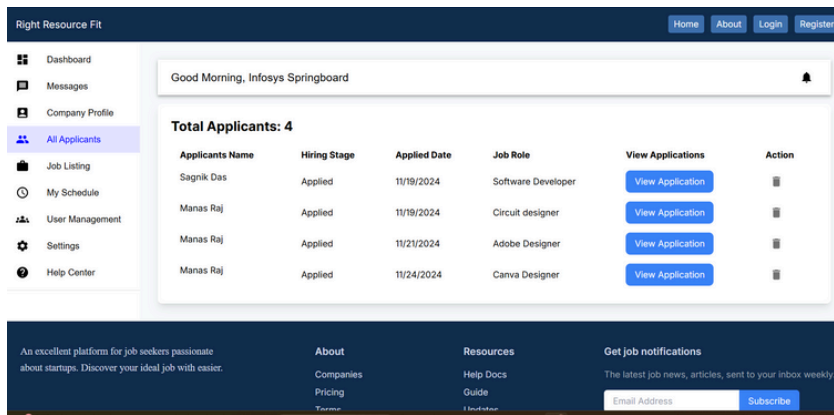
Company Joblisting Report

Application Management Component

Enables users to view and manage job applications.

Key Features:

- Status Tracking: Displays the status of applications (e.g., pending, shortlisted).
- Delete Application: Allows Company to delete applications if needed.
- Details View: Provides detailed job descriptions and application history.

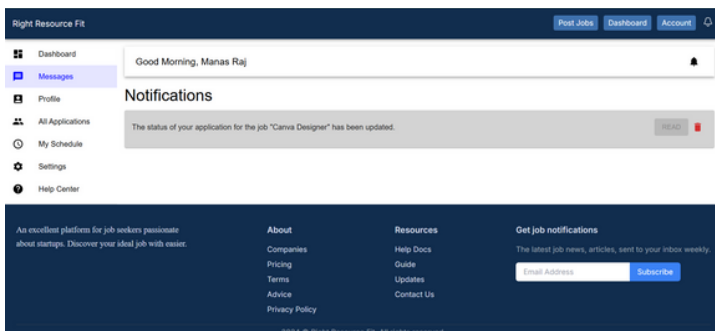


Messages Component

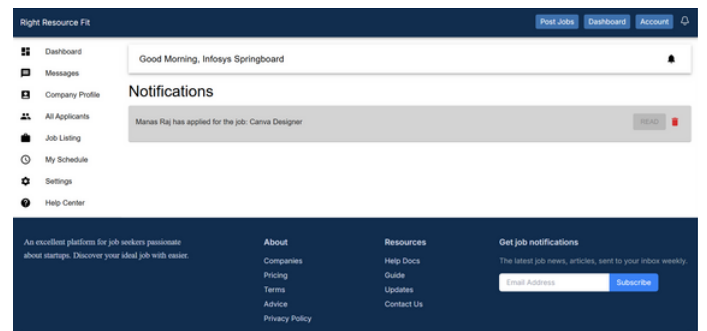
Manages communication between job seekers and companies, enabling seamless messaging.

Key Features:

- Inbox View: Displays a list of conversations with the latest messages previewed..
- Unread Indicators: Highlights conversations with unread messages.
- Responsive Design: Optimized for both desktop and mobile views for user convenience.



Jobseeker Messages



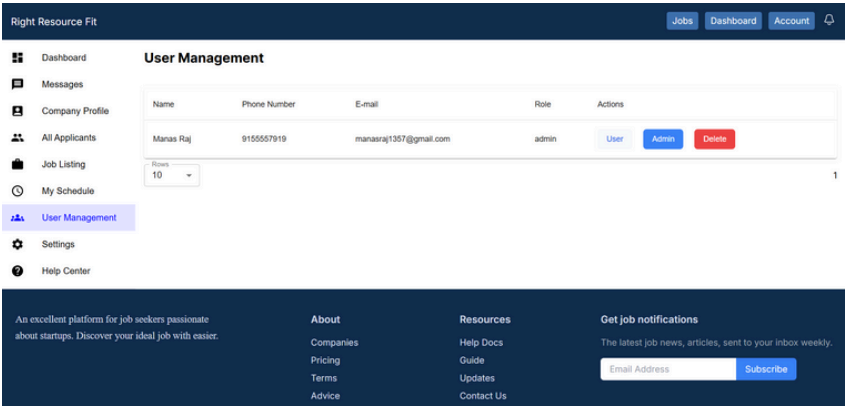
Company Messages

User Management Component

Handles the management of users, allowing administrators to view, edit, and manage user details.

Key Features:

- User List:** Displays all registered users (job seekers and companies) in a tabular format with filters for easy navigation.
- Role Management:** Supports assigning or modifying user roles (e.g., job seeker, company, admin).
- Deactivate/Activate Accounts:** Enables suspension or reactivation of user accounts for security or compliance purposes.

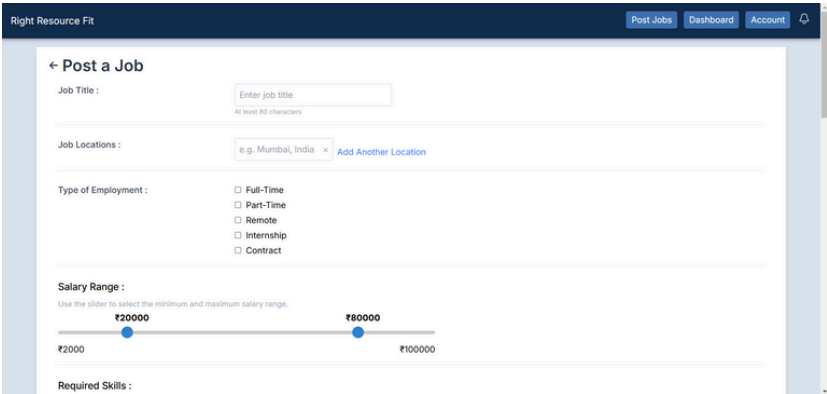


Job Posting Component

Manages job listings, enabling employers to create, edit, and manage job opportunities.

Key Features:

- Job Listing Creation:** Employers can create job posts by entering details like title, description, and requirements.
- Job Categories:** Employers can categorize job posts (e.g., Full-time, Part-time, Remote) for easier filtering.

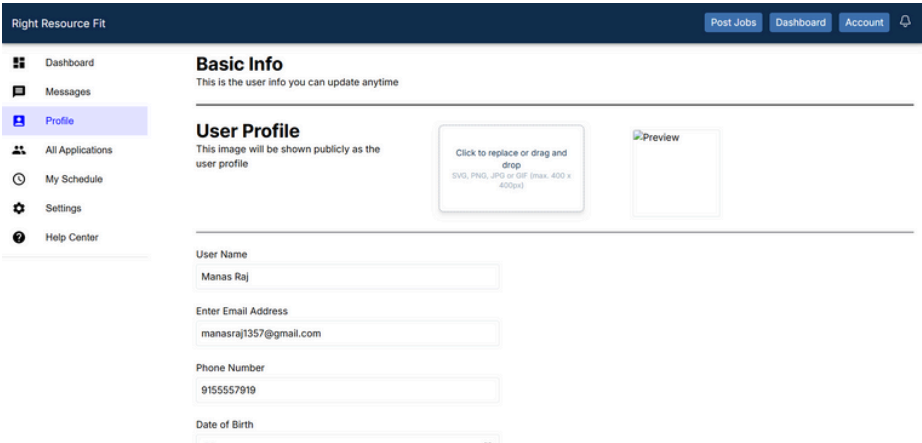


Jobseeker Profile Component

Allows job seekers to create, manage, and update their personal and professional details.

Key Features:

Profile Creation: Job seekers can create profiles by adding personal details, education, and work experience.



Job Description and Apply Job Component

Displays job details and allows job seekers to apply for positions.

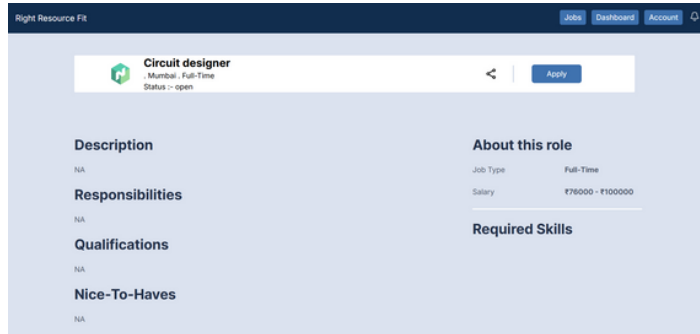
Key Features:

Job Description: Provides detailed information about the job, including title, responsibilities, qualifications, and company details.

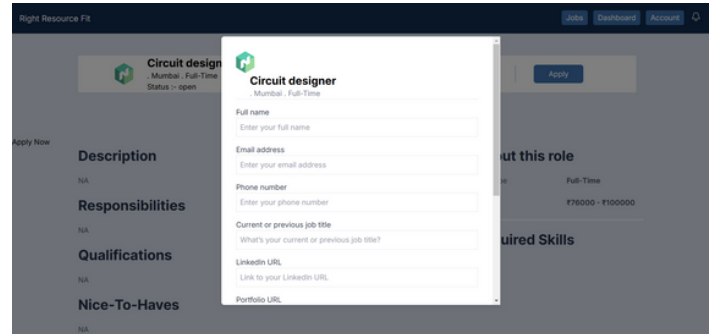
Apply for Job: Job seekers can apply directly by submitting their resume or profile with a single click.

Application Confirmation: After applying, job seekers receive confirmation of their application submission.

Application Tracking: Job seekers can track the status of their applications (e.g., Pending, Reviewed, Rejected).



Job Description



Application form

Jobseeker Application Component

Allows job seekers to apply for jobs and manage their applications.

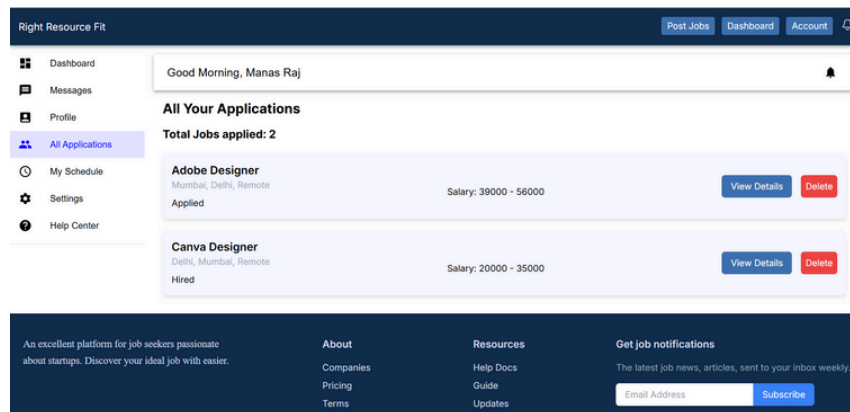
Key Features:

View Job Descriptions: Job seekers can view detailed job descriptions, including title, responsibilities, requirements, and company information.

Application Confirmation: Upon applying, job seekers receive immediate confirmation of their application submission.

Track Application Status: Job seekers can monitor the progress of their applications (e.g., Pending, Under Review, Accepted, Rejected).

Saved Applications: Job seekers can save jobs they are interested in to apply later or for future reference.



API Components

User Authentication & Management

User Registration

- **Route:** POST <http://localhost:1000/signup>
- **Description:** Register a new user account.

Input:

```
{
  "username": "Manas Raj",
  "email": "raj@gmail.com",
  "password": "Raj@123"
}
```

Output:

- 200 OK: { "message": "Signup successful" }
- 400 Bad Request: { "error": "Invalid input" }

User Login

- **Route:** POST <http://localhost:1000/login>
- **Description:** Authenticate a user.

Input:

```
{  
  "email": "raj@gmail.com",  
  "password": "Raj@123"  
}
```

Output:

- 200 OK: { "message": "Login successful" }
- 401 Unauthorized: { "error": "Invalid credentials" }

Password Reset Request

- **Route:** POST <http://localhost:1000/send-otp>
- **Description:** Send OTP for password reset.

Input:

```
{  
  "email": "raj@gmail.com"  
}
```

Output:

- 200 OK: { "message": "OTP sent successfully" }
- 400 Bad Request: { "error": "Failed to send OTP" }

Verify OTP for Password Reset

- **Route:** POST <http://localhost:1000/verify-otp>
- **Description:** Verify OTP sent to email for password reset.

Input:

```
{  
  "otp": "698214"  
}
```

Output:

- 200 OK: { "message": "OTP verified successfully" }
- 400 Bad Request: { "error": "Invalid OTP" }

User Logout

- **Route:** POST <http://localhost:1000/logout>
- **Description:** Logout a user.

Input: None**Output:**

200 OK: { "message": "Logged out successfully" }

Job Applications**Get All Applications for a Job**

- **Route:** GET <http://localhost:1000/applications>
- **Description:** Retrieve all applications for a specific job.

Input: None**Output:**

200 OK: [{ "applicationId": "1", "userId": "2", "status": "Pending" }, ...]

Apply for a Job

Route: POST <http://localhost:1000/apply/:jobId/:userId>

Description: This route allows a user to submit a job application. The user provides details like their resume and other application information (e.g., full name, email, phone, LinkedIn, portfolio, additional information), and it gets stored in the JobApplication model.

Input:

```
{
  "userId": "671f1b74f350e165d3958f65",
  "jobId": "673ff1845ca9da30126554c8",
  "fullName": "Manas Raj",
  "email": "raj@gmail.com",
  "phone": "915555XXXX",
  "jobTitle": "Software Developer",
  "linkedin": "https://www.linkedin.com/in/raj",
  "portfolio": "https://raj.dev",
  "additionalInfo": "I am passionate about building scalable applications and am eager to contribute to your company's success.",
  "resume": "<Binary data of resume file>"
}
```

Output:

- 200 OK: Response: { "message": "Application submitted successfully" }
- 400 Bad Request: Response: { "error": "Invalid application data" }

Delete Application

- **Route:** DELETE <http://localhost:1000/applications/:id>
- **Description:** Delete a specific application.

Input:

id: 67436b52d9cbbf62f5a7b9ec

Output:

- 200 OK: { "message": "Application deleted successfully" }
- 404 Not Found: { "error": "Application not found" }

Get Application by ID

- **Route:** GET <http://localhost:1000/applications/:id>
- **Description:** Retrieve an application by its ID.

Input:

id: 67436b52d9cbbf62f5a7b9ec

Output:

200 OK: { "applicationId": "1", "userId": "2", "status": "Pending" }

Update Application Status

- **Route:** PUT <http://localhost:1000/applications/:id>
- **Description:** Update the status of an application.

Input:

id: ID of the application

Body: { "status": "Hired" }

Output:

- 200 OK: { "message": "Application status updated" }
- 400 Bad Request: { "error": "Invalid status" }

Job Postings

Get Specific Job by ID

- **Route:** GET <http://localhost:1000/jobs/:id>
- **Description:** Retrieve a specific job posting by its ID.

Input:

id: 673ff1845ca9da30126554c8

Output:

200 OK: { "jobId": "673ff1845ca9da30126554c8", "title": "Software Engineer", "status": "Active" }

Create Job Posting

- **Route:** POST <http://localhost:1000/jobs>
- **Description:** Create a new job posting.

Input:

Body: { "title": "Software Engineer", "description": "Job description", "location": "Remote"..... }

Output:

200 OK: { "message": "Job posted successfully" }

Get All Job Postings

- **Route:** GET <http://localhost:1000/jobs>
- **Description:** Retrieve all job postings.

Input: None

Output:

200 OK: [{ "jobId": "1", "title": "Software Engineer", "status": "Active" }, ...]

Update Job Posting

- **Route:** PUT <http://localhost:1000/jobs/:id>
- **Description:** Update a job posting.

Input:

id: 673ff1845ca9da30126554c8

Body: { "title": "Senior Software Engineer", "description": "Updated description" }

Output:

200 OK: { "message": "Job updated successfully" }

Delete Job Posting

- **Route:** DELETE <http://localhost:1000/jobs/:id>
- **Description:** Delete a specific job posting.

Input:

id: 673ff1845ca9da30126554c8

Output:

200 OK: { "message": "Job deleted successfully" }

Jobseeker Profile

Save Jobseeker Profile

- **Route:** POST <http://localhost:1000/jobseeker-profile/save>
- **Description:** Save a new jobseeker profile.

Input:

Body: { "name": "John Doe", "location": "New York"..... }

Output:

200 OK: { "message": "Profile saved successfully" }

Get Jobseeker Profiles

- **Route:** GET <http://localhost:1000/jobseeker-profiles>
- **Description:** Retrieve a list of all jobseeker profiles.

Input: None

Output:

200 OK: [{ "userId": "1", "name": "John Doe",... }]

Get Specific Jobseeker Profile by ID

- **Route:** GET <http://localhost:1000/jobseeker-profile/userId>
- **Description:** Retrieve a specific jobseeker profile by user ID.

Input:

userId: ID of the jobseeker

Output:

200 OK: { "userId": "1", "name": "John Doe",..... }

Notifications

Get Notifications for a Company

- **Route:** GET <http://localhost:1000/notifications/:companyId>
- **Description:** Get notifications for a specific company.

Input:

companyId: 674552fc0605c2a53f2168b7

Output:

200 OK: [{ "notificationId": "674367c0d9cbbf62f5a7b84f", "message": "New job posting!" }, ...]

Mark Notification as Read

- **Route:** PUT <http://localhost:1000/notifications/:notificationId/read>
- **Description:** Mark a notification as read.

Input:

notificationId: 674367c0d9cbbf62f5a7b84f

Output:

200 OK: { "message": "Notification marked as read" }

Delete Notification

- **Route:** DELETE <http://localhost:1000/notifications/:notificationId>
- **Description:** Delete a notification.

Input:

notificationId: 674367c0d9cbbf62f5a7b84f

Output:

200 OK: { "message": "Notification deleted successfully" }

Get Message Count for a User

- **Route:** GET <http://localhost:1000/messages/count/:userId>
- **Description:** Get the count of unread messages for a specific user.

Input:

userId: 671f1b74f350e165d3958f65

Output:

200 OK: { "messageCount": 5 }

Error Handling and Troubleshooting

Effective error handling and troubleshooting ensure the smooth operation of the Right Resource Fit platform. Below are some common errors that users and developers may encounter, along with steps for troubleshooting and resolving them.

Gmail SMTP Not Working

Error Description: The platform uses SMTP to send email notifications (e.g., OTPs, confirmation emails). If SMTP fails, emails won't be sent.

Troubleshooting Steps:

Check Gmail Settings: Ensure that less secure app access is enabled in your Gmail account. Gmail may block connections from apps it considers less secure.

- Navigate to your Google Account settings.
- Under "Less secure app access," turn on access.

Check Credentials: Double-check the email and password provided in the configuration to ensure they are correct.

- Check SMTP Settings:
- SMTP Server: smtp.gmail.com
- Port: 587 (for TLS)
- Ensure the secure: true flag is set in your SMTP configuration.
- Check 2-Step Verification: If you have 2-step verification enabled, you need to generate and use an App Password instead of your regular Gmail password.

Port Conflicts

Error Description: The application may fail to start due to port conflicts (e.g., the server tries to bind to port 3000, but it's already being used by another application).

Troubleshooting Steps:

Check Which Service is Using the Port:

On Linux/macOS, you can use the following command to check:

```
lsof -i :3000
```

On Windows, use:

```
netstat -ano | findstr :3000
```

This will show you which process is using the port.

Stop the Conflicting Process:

If you find a process using the port, you can stop it by identifying its PID and killing it.

On Linux/macOS, run:

```
kill -9 <PID>
```

On Windows, use:

```
taskkill /PID <PID> /F
```

Change the Port: If you can't stop the conflicting process or prefer to use a different port, update your application's configuration file to use a new port number (e.g., 3001 or 8080). Example (Node.js Express App):

Database Connection Failures

Error Description: The application may fail to connect to the database due to incorrect configuration or service unavailability.

Troubleshooting Steps:

Check Database Credentials: Ensure that the database username, password, and connection string in the configuration are correct.

- For MongoDB, check the connection string format and the database's host URL.
- Ensure Database is Running: Make sure that the MongoDB or other database service is running on your local machine or remote server.
- Check Database Access Permissions: Ensure that the database user has sufficient permissions to perform operations (e.g., read, write).

API Request Failures

Error Description: API requests may fail due to misconfiguration or incorrect endpoint usage.

Troubleshooting Steps:

- **Check Endpoint URL:** Ensure that the correct API endpoint is being used. Check the route and URL in the backend and the client-side code.
- Check API Authentication: If your API requires authentication (e.g., API keys or tokens), ensure that these credentials are correctly included in the request headers.
- Examine Response Codes: Check the HTTP status codes returned by the API. Common errors include:
 - 404 (Not Found): The requested endpoint is incorrect or missing.
 - 500 (Internal Server Error): There's an issue with the server; check logs for more details.
 - 401 (Unauthorized): Authentication issues, such as missing or invalid API keys.

File Upload Failures

Error Description: Users may encounter issues when uploading files (e.g., profile images, portfolio items).

Troubleshooting Steps:

- **Check File Size Limits:** Ensure that the file size does not exceed the server's upload limits. In Node.js, for example, you can increase the upload size in the body parser middleware
- Verify File Type: Ensure that only allowed file types (e.g., .jpg, .png, .pdf) are uploaded. Validate file types both on the client-side and server-side. Example (Node.js File Type Validation)
- Check Disk Space: Ensure the server has enough disk space to store the uploaded files.

Authentication Issues (JWT or Session Timeout)

Error Description: Users may experience issues logging in or maintaining sessions due to token expiration or invalid credentials.

Troubleshooting Steps:

- **Check Token Expiry:** If you are using JWT tokens, ensure that the token hasn't expired. JWT tokens typically have an expiration time encoded in them.
- **Ensure Correct Login Credentials:** Check that the user is providing the correct username and password and that your backend is validating these correctly.
- **Check Session Storage:** If using sessions, ensure the session cookie is correctly stored and sent with each subsequent request.

Frontend Rendering Issues

Error Description: The frontend UI may fail to load or display incorrectly due to errors in rendering or incorrect data passed to components.

Troubleshooting Steps:

- **Check JavaScript Errors:** Use the browser's developer tools (F12) to check for JavaScript errors in the console.
- **Look for error messages related to missing variables, incorrect function calls, or syntax issues.**
- **Inspect DOM Elements:** Ensure the correct data is being passed to your UI components. Use `console.log()` to log the data being passed into templates or components.
- **Ensure Dependencies Are Installed:** Check if all the required frontend dependencies are installed (e.g., React.js). Run `npm install` or `yarn install` to install any missing packages.

General Troubleshooting Tips

- **Clear Cache:** Sometimes, browser or application caches can cause issues. Clear the cache or try running the application in incognito mode.
- **Check Server Logs:** Always check the server logs for detailed error messages. They can provide more context on what's going wrong.
- **Use Debugging Tools:** Utilize debugging tools like Chrome DevTools, Visual Studio Code debugger, or backend-specific debuggers to step through the code and identify issues.

Maintenance and Support

Procedures for Regular Maintenance

Backing Up the Database

- Use MongoDB's built-in backup tools or cloud-based solutions like MongoDB Atlas to schedule regular backups.
- Store backups securely in a cloud storage service (e.g., AWS S3, Google Cloud Storage) to prevent data loss.
- Rotating API Keys or Credentials

Regularly update sensitive credentials stored in environment variables (e.g., .env files).

- Use tools like AWS Secrets Manager or Azure Key Vault for secure storage.
- Monitoring Logs and Server Health
- Implement logging with tools like Winston or ELK Stack for application logs.
- Use monitoring tools like New Relic, Prometheus, or Datadog to track server performance and uptime.

How Users Can Report Issues or Request Support

- Provide a dedicated Support Email Address (e.g., support@rightresourcefit.com).
- Include a link to a Support Pagewhere users can:
- Submit issue reports.
- Access self-help guides or FAQs.
- Set up a ticketing system using tools like Zendesk or Jira for efficient issue tracking.

Version Control and Updates

Using Git for Version Control

Branching Strategies:

- Use the main branch for production-ready code.
- Create feature branches for new features or bug fixes.
- Use release branches for testing updates before merging into the main branch.

Commit Messages:

- Follow a standard format such as [feature]: Add user authentication or [fix]: Resolve login bug.
- Applying Updates and Migrations Safely
- Test updates in a staging environment before deploying to production.
- Perform database migrations using tools like Mongoose migration scripts.

Glossary or FAQ

Glossary

- Matching Algorithm: A system that matches candidates to jobs based on their skills, experience, and availability.
- Skills Portfolio: A section in a user profile where professionals showcase their expertise, certifications, and completed projects.
- Availability Tracking: A feature that allows businesses to view real-time availability of candidates.

FAQs

- Why is the Gmail SMTP not working?

Ensure that "Allow less secure apps" is enabled in Gmail settings or use OAuth 2.0 for secure access.

- What to do if I encounter a port conflict?

Use the command `lsof -i :3000` to identify the service occupying the port and stop it.

- How can I reset my password?

Go to the login page and click on "Forgot Password." Follow the instructions to reset it via email.

Account and Profiles

- How can I update my profile details?

Log in to your account, navigate to the Profile section in the dashboard, and click Edit Profile to make updates.

- Why was my profile rejected?

Profiles are reviewed for accuracy and completeness. Ensure that all required fields are filled out and that your portfolio includes valid work examples.

Jobs and Applications

- How do I post a Job as a business?

Navigate to the Post a Project section, fill in the Job details (e.g., skills required, budget, and deadline), and submit it for review.

- Can I apply for multiple Jobs at the same time?

Yes, professionals can apply for multiple Jobs, but it's advised to prioritize projects that align closely with their skill set and availability.

- What happens if a project is canceled after it starts?

If a project is canceled, payments already made will follow the platform's refund or dispute resolution policy.

Technical Issues

- Why am I unable to upload my portfolio?

Ensure that your files meet the size and format requirements (e.g., PDF, PNG, JPG, under 10MB). If the issue persists, clear your browser cache and try again.

- I'm not receiving email notifications. What should I do?

Check your spam or junk folder. Add the platform's email address to your safe sender list. If the problem continues, verify that your email address is correctly configured in your profile.

Future Enhancements

Integration with Third-Party Tools

- Add integration with tools like Slack for notifications and Zapier for workflow automation.

AI-Driven Recommendations

- Implement AI algorithms to recommend the best candidates or projects based on past trends.

Enhanced Reporting and Analytics

- Provide detailed insights into hiring trends, project performance, and user engagement through an analytics dashboard.

