



Inspiring Excellence

CSE471: System Analysis and Design
Project Report
Project Title: Online Job Portal

| Group No: 03, CSE471 Lab Section: 06, Fall 2023 | |
|---|----------------------|
| ID | Name |
| 21301524 | A N M Jubair Tanvir |
| 21301665 | Ashfaq Ahmad Saad |
| 21301510 | Asif Ahnaf Chowdhury |

Table of Contents

| Section No | Content | Page No |
|------------|-----------------------------------|---------|
| 1 | Introduction | 2 |
| 2 | Functional Requirements | 3 |
| 3 | User Manual | 4-12 |
| 4 | Frontend Development | 13-30 |
| 5 | Backend Development | 31-45 |
| 6 | Technology (Framework, Languages) | 45 |
| 7 | Github Repo Link | 45 |
| 8 | Individual Contribution | 46 |

Introduction

We worked on an exciting online job portal project that aims to change how people find jobs in Bangladesh. In our rapidly growing digital world, our platform acts like a helpful bridge connecting people looking for jobs with companies that are hiring. Our main goal is to make hiring and getting hired easier by creating a website that anyone can use easily, whether they are looking for a job or looking to hire someone. We focused on making the experience good for users and efficient. Our project includes cool features like showing real-time job numbers, creating personal profiles, and an easy way to apply for jobs. We used the latest technologies to make sure our website can run smoothly. This way, it can keep up with the needs of our fast-paced job world. The project is all about making the job market in Bangladesh more connected and efficient, providing a space where job seekers and employers can easily find each other.

Project Summary

In this project, we developed a frontend and backend web application through which, a job seeker can seek for a job and an employer can hire a skilful job seeker. Job seekers can see the different circulars of the different companies. He will have a profile where he can add all his professional information, even he can add his CV, and can apply through the system. As an employer, one can have a profile, he can see job seekers' profiles and see the list of applicants. A job seeker or employer before registration to the system can see the live statistics where he/she can see the total number of job seekers, the total number of employers and the total number of circulars published to date.

Functional Requirements

Module-1: User registration and authentication

1. Seeker's Registration
2. Employer's Registration
3. User login and logout
4. Forgot password
5. Verify mail

Module-2: Seeker's Profile

6. CV Add/Update
7. CV Download
8. Add Name, address, Bio, etc
9. Add and update Profile picture
10. Profile view count

Module-3: Employer's Profile

11. Post a Circular
12. Set Time limits for circulars
13. Show all the Job Circulars, Edit the Circular, Delete
14. Add Name, Profile picture, About etc.
15. View Applicants and their Profile.

Module-4: User feed

16. Filtering
17. Searching
18. Easy Apply
19. Live statistics
20. Contact us

User manual:

A. New user guide:

1. **Register** as a **job seeker** or an **employer**.

If you are a job seeker then register as a job seeker.

Online Job Portal

Home Job Seeker Employer Login

Looking for a job?
Please create an account

Register

Full name

Email

Password

Register

Otherwise, register as an employer.

Online Job Portal

Home Job Seeker Employer Login

Looking for an employee?
Please create an account

Employer Registration

Company name

Email

Password

Register

2. **Verify** your **email** before you log in for the first time.

You will see a text that reminds you to verify first.

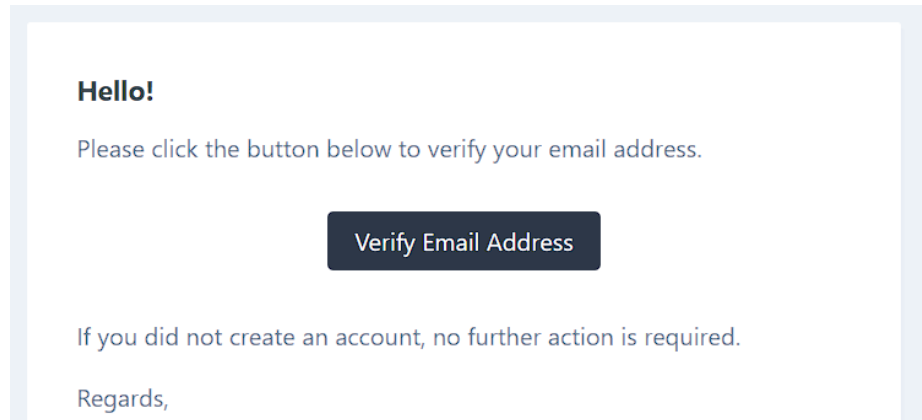
verify before first login

If you log in without verification, you will see this message.

Verify Account

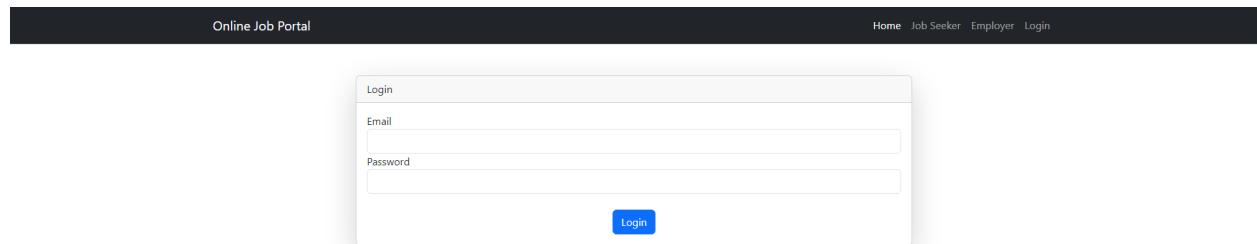
Your account is not verified. Please check your email and click on the link to verify your account.

Go check your email inbox, and verify your email.



3. Now you can **log in** and have full access.

Login as a job seeker or employer.

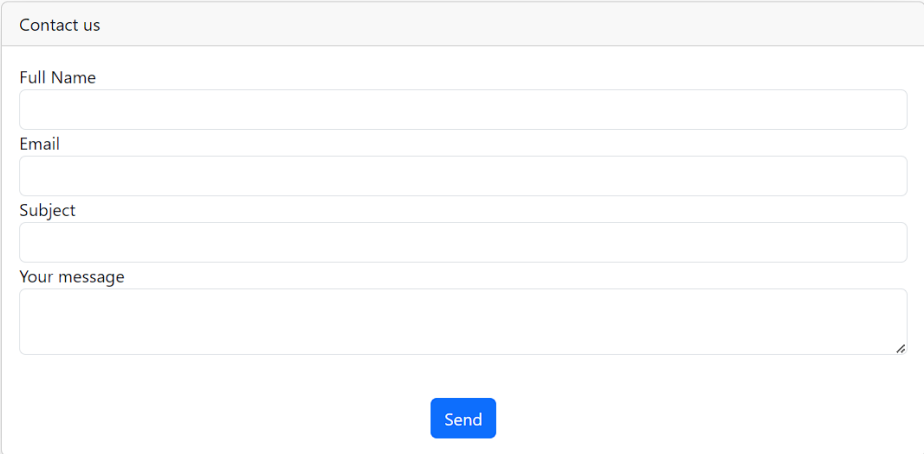


4. From the navigation bar, you can go see your profile, log out, contact us, and redirect to the home page.



a. **Contacting us**

You can share your concerns with us through this form. Make sure you add your account's email address if you need help regarding your account.

A contact form titled "Contact us" with a light gray header. It contains four input fields: "Full Name", "Email", "Subject", and "Your message". The "Your message" field is a larger text area. A blue "Send" button is located at the bottom right of the form.

Contact us

Full Name

Email

Subject

Your message

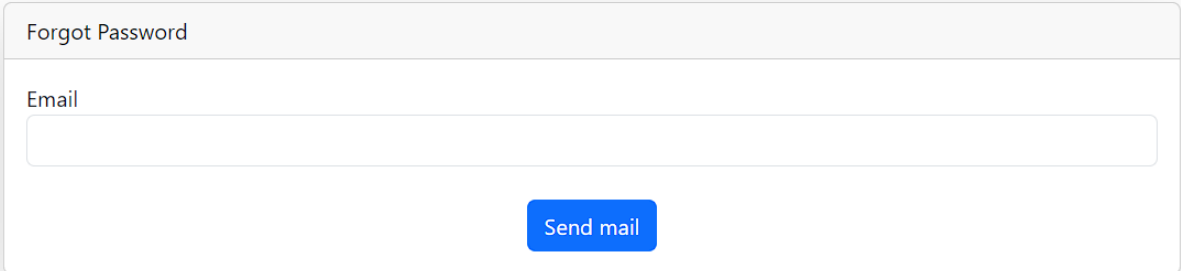
Send

b. **Logging out**

You can logout after you are done with your account. The logout button will redirect you to the home page

5. Set a new **password** if you have **forgotten** it. It happens to the best of us. Do not worry!

Just put your account's email address in the form shown below;

A form titled "Forgot Password" with a light gray header. It contains a single input field for "Email". A blue "Send mail" button is located at the bottom right of the form.

Forgot Password

Email

Send mail

Check your email inbox to set up a new password by clicking Reset Password.

Hello Asif Ahnaf Chowdhury

Reset Password

Here enter your new password, and confirm.

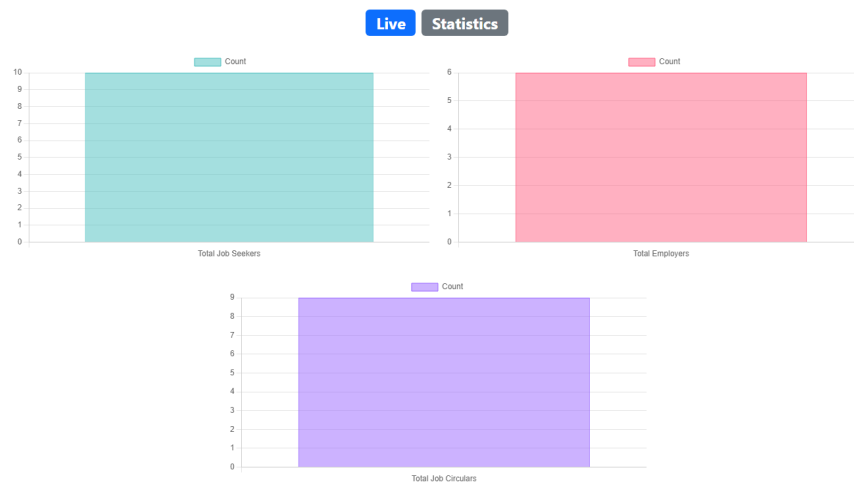
Reset Password

New passwords

Confirm new password

You should log in with your new password afterwards.

6. Also on the home page, you will see **live statistics**.



B. Seeker's guide:

1. As a seeker, you can view the **list of every circular** on our website.

You can **search** and **filter** for your desired jobs

Search Users by Skill

Filter by Skill:

Select Skill

| Name | Email | Skills |
|-------------------|------------------------|-------------------------|
| Ashfaq ahmad saad | ashfaqsaad10@gmail.com | Machine Learning,Gaming |
| Jubair Tanvir | jubair@gmail.com | Gaming |
| Ahnaf | ahnaf@gmail.com | Blockchain |

Search Results for "Gaming"

| Name | Email |
|-------------------|------------------------|
| Ashfaq ahmad saad | ashfaqsaad10@gmail.com |
| Jubair Tanvir | jubair@gmail.com |

C. Easily apply to the circulars.

From the circular list, you can directly apply for the jobs you like by pressing the apply button.

Software Eng

Meta

View Details
Apply

Web dev

Meta

View Details
Apply

SQA Job

Meta

View Details
Apply

Or, you can see the details by pressing ‘view details’ and then pressing the apply button.

Details for Web dev

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec laoreet sollicitudin iaculis. Aenean malesuada venenatis ex, quis tempus tortor imperdiet ut. Etiam in rhoncus libero, et tincidunt ante. Suspendisse non sapien mauris. Maecenas eget rhoncus metus. Fusce vel ipsum suscipit diam rutrum pharetra finibus at mi. Duis quis nisi a erat molestie pretium sed et justo. Sed eleifend ipsum vitae sem laoreet dictum. Pellentesque dignissim est vitae porta sollicitudin. Vivamus scelerisque lorem in fermentum eleifend. Phasellus nibh nisi, viverra vitae lectus vitae, interdum aliquam quam. Praesent et neque in ipsum sodales sodales a a dolor. Fusce sagittis ante ut mi auctor, non fringilla metus tempus. Sed sodales nunc lacus, nec placerat est mattis ac. Pellentesque ac magna urna. In et justo id odio molestie eleifend non et nisi. Aliquam erat volutpat. Praesent auctor consectetur sem at malesuada. Praesent suscipit elementum ultrices. Pellentesque ac diam vel est aliquet pellentesque. Sed fermentum felis enim, at elementum neque congue bibendum. Nunc rutrum volutpat dolor non efficitur. Suspendisse id gravida odio. Nullam venenatis, lacus eleifend lacinia dapibus, massa nunc pulvinar orci, at blandit turpis felis in leo. Sed lobortis justo sit amet ex aliquet, faucibus mattis nisi efficitur. Fusce condimentum sodales elit, ac feugiat turpis accumsan non. Quisque convallis varius posuere. Proin quis consequat magna. Maecenas sodales quam non metus gravida, a scelerisque lorem porttum nunc non, vestibulum nulla. Vivamus pharetra elit ac ullamcorper convallis. Integer eu mollis ex. In in velit efficitur, ullamcorper elit tincidunt, aliquet purus. Etiam sit amet pretium augue. Maecenas urna erat, tristique vitae lacinia sed, imperdiet in lacus. Nunc ante mi, interdum non mauris quis, iaculis facilisis magna. Etiam a consequat augue, at convallis lacus. Fusce rutrum id neque ut ultrices. Duis vitae tincidunt lectus. Maecenas ultrices nisl quis dui iaculis, nec viverra velit vulputate. Nunc sagittis sit amet magna tincidunt semper. Fusce vel iaculis nulla. Nulla aliquet lobortis diam, ac lobortis sapien blandit a. Pellentesque diam dolor, semper quis vulputate ut, tincidunt sit amet lacus. Duis eget rutrum nibh. Nulla in congue ex, quis sollicitudin diam. Integer aliquam egestas aliquam. Praesent eget porttitor mauris.

Apply
Close

View Details
Apply

SQA Job

Meta

View Details
Apply

2. Edit your profile to grab the attention of our employers.

a. How to add, download, and edit your CV

Online Job Portal

Home Logout

Hi, A N M JUBAIR TANVIR
[Download CV](#)

Update CV

Upload CV (PDF, DOC, DOCX)

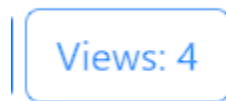
Choose File
No file chosen

Update CV

b. How to **edit information** in your profile

The screenshot shows a user profile for 'Ashfaq ahmad saad', a Full Stack Developer in the Bay Area, San Francisco, CA. The profile includes a 'Follow' button and a 'Message' button. Below the profile information are input fields for 'Website', 'Github', and 'Facebook'. To the right, there is an 'Edit' form with fields for 'Full Name', 'Email', 'Phone', 'Mobile', and 'Address'. A 'Save Changes' button is located below these fields. At the bottom of the edit form, there is a 'Skills' section with a text input containing 'Machine Learning,Gaming' and an 'Update Skills' button.

3. Look at the **view count** on your profile to know how many employers took an interest in you.



C. Employer's guide:

1. **Edit your profile** to uphold your company.

This is a duplicate of the screenshot above, showing the user profile for 'Ashfaq ahmad saad' and the 'Edit' form with fields for name, email, phone, mobile, address, and skills.

2. Announce circulars to our webpage.

a. How to **post circulars**.

Click the button to post a job circular.

Post a Job Circular

b. Set a **time limit** on the circulars.

Online Job Portal Home Logout

Create Job Circular

Title

Description

Deadline

dd-----yyyy --:-- --

Create Job Circular

c. You can also **edit circulars** afterwards.

By clicking on the edit button you can edit the current job posting.

| Title | Description | Deadline | Actions |
|-------------------|-------------|---------------------|--|
| Software engineer | Hiring | 2023-12-09 04:00:00 | Edit Download Delete |

d. View the **list of your circulars**

Online Job Portal Home Logout


Post a Job Circular

Your Job Circulars

| Title | Description | Deadline | Actions |
|--------------------------|--|---------------------|--|
| Software engineer | Hiring | 2023-12-09 04:00:00 | Edit Download Delete |
| Check 2 | Check 2 | 2023-12-09 14:52:00 | Edit Download Delete |
| Project Manager | We are hiring a project manager to lead a team of ML engineers. | 2023-12-09 00:00:00 | Edit Download Delete |
| Senior Software Engineer | We need a highly skilled ML engineer who will build some ML models | 2023-12-02 04:30:00 | Edit Download Delete |

3. View the **list of applicants** and their **profiles**.

Click the Applicants button



Meta

Full Stack Developer
Bay Area, San Francisco, CA

[Post Circular](#) [Applicants](#) [Message](#)

From this list, you can click on any Gmail and it will take you to the applicant's profile

Applicants

- asif.ahnaf.chowdhury@g.bracu.ac.bd
- asif.ahnaf.chowdhury@g.bracu.ac.bd
- asif.ahnaf.chowdhury@g.bracu.ac.bd
- asif.ahnaf.chowdhury@g.bracu.ac.bd

Frontend Development

1. A N M Jubair Tanvir

Home page: In this section there is a hero image with a title. This page represents what the system is about.

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Online Job Portal</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-T3c6CoIi6uLR9A9TneNEoa7Rxnz"
</head>
<body>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-C6RzsynM9kwdRMneT87bh95OGNyZPhcTNXj1"

  <nav class="navbar navbar-expand-lg bg-dark border-bottom border-body" data-bs-theme="dark">
    <div class="container">
      <a class="navbar-brand" href="#">Online Job Portal</a>
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded='
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav ms-auto">
          <li class="nav-item">
            <a class="nav-link active" aria-current="page" href="{{ route('live-statistics.index') }}">Home</a>
          </li>

          @if(!Auth::check())

          <li class="nav-item">
            <a class="nav-link" href="{{route('create.seeker')}}">Job Seeker</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{{route('create.employer')}}">Employer</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{{route('login')}}">Login</a>
          </li>

          @endif

          @if(Auth::check())
```

Employer's Registration: From the page an employer can register to the system. After completing the registration, he will be able to login and update his profile information. Without registration, he can't login to the system.

```
<div class="container mt-5">
  <div class="row">

    <div class="col-md-6">
      <h1>Looking for an employee?</h1>
      <h3>Please create an account</h3>
    </div>

    <div class="col-md-6">
      <div class="card">
        <div class="card-header">Employer Registration</div>
        <form action="{{route('store.employer')}}" method="POST">@csrf
        <div class="card-body">
          <div class="form-group">
            <label for="">Company name</label>
            <input type="text" name="name" class="form-control">
            @if ($errors->has('name'))
              <span class="text-danger">{{ $errors->first('name') }}</span>
            @endif
          </div>
          <div class="form-group">
            <label for="">Email</label>
            <input type="text" name="email" class="form-control">
            @if ($errors->has('email'))
              <span class="text-danger">{{ $errors->first('email') }}</span>
            @endif
          </div>
          <div class="form-group">
            <label for="">Password</label>
            <input type="password" name="password" class="form-control">
            @if ($errors->has('password'))
              <span class="text-danger">{{ $errors->first('password') }}</span>
            @endif
          </div>
        </div>
        <div class="form-group">
          <button class="btn btn-primary" type="submit">Register</button>
        </div>
      </div>
    </div>
  </div>
</div>
```

User login and logout: A user who is a job seeker or employer can login to the system. From this page, a user can login as a job seeker or an employer. But before that, a user must have to register as a job seeker or an employer. After he/she is done, he/she can logout. After the logout, the system will redirect to the login page again.

```
section('content')


<div class="row justify-content-center">
    <div class="col-md-8">
      <div class="card shadow-lg">
        <div class="card-header">Login</div>
        <form action="{{route('login.post')}}" method="post">
          @csrf
          <div class="card-body">
            <div class="form-group">
              <label for="">Email</label>
              <input type="text" name="email" class="form-control">
              @if($errors->has('email'))
              <span class="text-danger">{{ $errors->first('email')}}</span>
              @endif
            </div>
            <div class="form-group">
              <label for="">Password</label>
              <input type="password" name="password" class="form-control">
              @if($errors->has('password'))
              <span class="text-danger">{{ $errors->first('password')}}</span>
              @endif
            </div>
            <br>
            <div class="form-group text-center">
              <button class="btn btn-primary" type="submit">Login</button>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
</div>
</div>


```


CV add, update and download: A Job seeker can update his profile. As a part of that, he can add his CV, update his CV and download it.

```

<!-- resources/views/job_seeker/show.blade.php -->

@extends('layouts.app')

@section('content')
    <div class="container">
        <h2>Hi, {{Auth::user()->name}}</h2>

        <!-- Display CV -->
        @if (Auth::user()->cv_path)
            <p><a href="{{ asset('storage/' . Auth::user()->cv_path) }}" target="_blank">Download CV</a></p>
        @else
            <p>No CV uploaded</p>
        @endif

        <!-- Update CV Form -->
        <h2>Update CV</h2>

        <form action="{{ route('job_seeker.updateCV') }}" method="post" enctype="multipart/form-data">
            @csrf
            <div class="form-group">
                <label for="cv">Upload CV (PDF, DOC, DOCX)</label>
                <input type="file" class="form-control" id="cv" name="cv" required>
            </div>
            <button type="submit" class="btn btn-primary">Update CV</button>
        </form>
    </div>
@endsection

```

Post a Circular: An employer can post a job circular. He can post a job circular and by viewing it an applicant can apply for the particular job. Applicants can also see the deadline of the circular and see the job requirements. Employers can set the deadline.

```

@extends('layouts.app')

@section('content')
    <div class="container">
        <h2>Create Job Circular</h2>

        <form action="{{ route('employer.job_circular.store') }}" method="post">
            @csrf
            <div class="form-group">
                <label for="title">Title</label>
                <input type="text" class="form-control" id="title" name="title" required>
            </div>
            <div class="form-group">
                <label for="description">Description</label>
                <textarea class="form-control" id="description" name="description" required></textarea>
            </div>
            <div class="form-group">
                <label for="deadline">Deadline</label>
                <input type="datetime-local" class="form-control" id="deadline" name="deadline" required>
            </div>
            <button type="submit" class="btn btn-primary">Create Job Circular</button>
        </form>
    </div>
@endsection

```

Set Time limits for circulars, Show all the Job Circulars, Edit the Circular, Delete: On this page all the job circulars that are posted by the employer are shown. An employer can delete the existing job circulars. He/she can also edit the existing job circulars. The system will automatically delete the circulars after the deadline.

```

ion( content )


```

Click here to ask Blackbox to help you code faster
@extends('layouts.app')

@section('content')
 <div class="container">
 <h2>Edit Job Circular</h2>

 <form action="{{ route('employer.job_circular.update', $jobCircular) }}" method="post">
 @csrf
 @method('put')
 <div class="form-group">
 <label for="title">Title</label>
 <input type="text" class="form-control" id="title" name="title" value="{{ $jobCircular->title }}" required>
 </div>
 <div class="form-group">
 <label for="description">Description</label>
 <textarea class="form-control" id="description" name="description" required>{{ $jobCircular->description }}</textarea>
 </div>
 <div class="form-group">
 <label for="deadline">Deadline</label>
 <input type="datetime-local" class="form-control" id="deadline" name="deadline" value="{{ $jobCircular->deadline }}" required>
 </div>
 <button type="submit" class="btn btn-primary">Update Job Circular</button>
 </form>
 </div>
@endsection

```



17


```

Live Statistics: In this section, anyone can see the total number of job seekers, the total number of employers and the total number of circulars published to date. He/she will see this in graphical format.

```
@extends('layouts.app')

@section('content')

    <!-- Hero Section -->
    <div class="hero-section">
        
        <h2 class="text-white hero-title">Welcome to The Largest Online Job Portal of Bangladesh</h2>
    </div>

    <!-- Main Content Section -->
    <div class="container mt-5">
        <h2 class="text-center mb-4">
            <span class="badge bg-primary">Live</span>
            <span class="badge bg-secondary">Statistics</span>
        </h2>

        <div class="row justify-content-center">
            <div class="col-md-6 mb-4">
                <canvas id="jobSeekersChart"></canvas>
            </div>

            <div class="col-md-6 mb-4">
                <canvas id="employersChart"></canvas>
            </div>

            <div class="col-md-6 mb-4">
                <canvas id="jobCircularsChart"></canvas>
            </div>
        </div>
    </div>
</div>
```

```
<!-- Chart.js Script -->
<script>
    var jobSeekersData = {
        labels: ['Total Job Seekers'],
        datasets: [{
            label: 'Count',
            data: [({{ $totalJobSeekers }})],
            backgroundColor: 'rgba(75, 192, 192, 0.5)',
            borderColor: 'rgba(75, 192, 192, 1)',
            borderWidth: 1
        }]
    };

    var employersData = {
        labels: ['Total Employers'],
        datasets: [{
            label: 'Count',
            data: [({{ $totalEmployers }})],
            backgroundColor: 'rgba(255, 99, 132, 0.5)',
            borderColor: 'rgba(255, 99, 132, 1)',
            borderWidth: 1
        }]
    };

    var jobCircularsData = {
        labels: ['Total Job Circulars'],
        datasets: [{
            label: 'Count',
            data: [({{ $totalCirculars }})],
            backgroundColor: 'rgba(153, 102, 255, 0.5)',
            borderColor: 'rgba(153, 102, 255, 1)',
            borderWidth: 1
        }]
    };

    var jobSeekersCtx = document.getElementById('jobSeekersChart').getContext('2d');
    var employersCtx = document.getElementById('employersChart').getContext('2d');
    var jobCircularsCtx = document.getElementById('jobCircularsChart').getContext('2d');

    var jobSeekersChart = new Chart(jobSeekersCtx, {
```

```

resources > views > statistics > index.blade.php > script
84
85
86     var employersChart = new Chart(employersCtx, {
87         type: 'bar',
88         data: employersData,
89         options: {
90             scales: {
91                 y: {
92                     beginAtZero: true
93                 }
94             }
95         }
96     });
97
98     var jobCircularsChart = new Chart(jobCircularsCtx, {
99         type: 'bar',
100        data: jobCircularsData,
101        options: {
102            scales: {
103                y: {
104                    beginAtZero: true
105                }
106            }
107        }
108    });
109 </script>
110 @endsection
111
112 <style>
113     /* Navbar Styles (Assuming your navbar has the class 'navbar') */
114     .navbar {
115         position: relative;
116         z-index: 1000; /* Ensure a higher z-index than the hero section */
117     }
118
119     /* Hero Section Styles */
120     .hero-section {
121         position: relative;
122         overflow: hidden;
123         height: 100vh;
124         margin-top: -50px; /* Adjusted to ensure proper positioning */

```

2. Ashfaq Ahmad Saad

2.1 Seeker's Registration: This registration blade file helps the user complete the registration process. This takes inputs such as email and password and stores them in the profile table.

```
<div class="col-md-6">
  <h1>Looking for an employee?</h1>
  <h3>Please create a account:</h3>
</div>

<div class="col-md-6">
  <div class="card">
    <div class="card-header">Employer Registration</div>
    <form action="{{route('store.employer')}}" method="POST">@csrf
    <div class="card-body">
      <div class="form-group">
        <label for="">Company name</label>
        <input type="text" name="name" class="form-control">
        @if ($errors->has('name'))
        <span class="text-danger">{{ $errors->first('name') }}</span>
        @endif
      </div>
      <div class="form-group">
        <label for="">Email</label>
        <input type="text" name="email" class="form-control">
        @if ($errors->has('email'))
        <span class="text-danger">{{ $errors->first('email') }}</span>
        @endif
      </div>
      <div class="form-group">
        <label for="">Password</label>
        <input type="password" name="password" class="form-control">
        @if ($errors->has('password'))
        <span class="text-danger">{{ $errors->first('password') }}</span>
        @endif
      </div>
    </div>
    <div class="form-group">
      <button class="btn btn-primary" type="submit">Register</button>
    </div>
  </div>
</div>
```

```
@extends('layouts.app')

@section('content')

<div class="container mt-5">
  <div class="row">
    <div class="col-md-6">
      <h1>Looking for a job?</h1>
      <h3>Please create a account:</h3>
    </div>
    <div class="col-md-6">
      <div class="card">
        <div class="card-header">Register</div>
        <form action="{{route('store.seeker')}}" method="POST">@csrf
        <div class="card-body">
          <div class="form-group">
            <label for="">Full name</label>
            <input type="text" name="name" class="form-control">
            @if ($errors->has('name'))
            <span class="text-danger">{{ $errors->first('name') }}</span>
            @endif
          </div>
          <div class="form-group">
            <label for="">Email</label>
            <input type="text" name="email" class="form-control">
            @if ($errors->has('email'))
            <span class="text-danger">{{ $errors->first('email') }}</span>
            @endif
          </div>
          <div class="form-group">
            <label for="">Password</label>
            <input type="password" name="password" class="form-control">
            @if ($errors->has('password'))
            <span class="text-danger">{{ $errors->first('password') }}</span>
            @endif
          </div>
          <div class="form-group">
            <button class="btn btn-primary" type="submit">Register</button>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>

@endsection
```

2.2 Seeker's Profile:

This Blade template code creates a user profile page using Bootstrap styles. It displays the user's personal information, including a profile picture, name, role, and contact details. The template includes links to the user's website, GitHub, and Facebook accounts. An "Edit" button redirects to a route for profile editing. The page also shows the user's skills if available, presenting them in a visually organized manner. The overall design is clean and user-friendly, following Bootstrap conventions for a responsive and visually appealing layout.

```

sources > views / products / home.blade.php / div.container / div.main-body / div.row.gutters-sm / div.col-md-8 / div.card.mb-3
1
2
3
4 <link rel='stylesheet' href='https://cdn.jsdelivr.net/npm/bootstrap@4.4.1/dist/css/bootstrap.min.css'>
5
6 <script src='https://cdn.jsdelivr.net/npm/bootstrap@4.4.1/dist/js/bootstrap.bundle.min.js'></script>
7
8 <div class="container">
9   <div class="main-body">
10
11     <!-- Breadcrumb -->
12     <nav aria-label="breadcrumb" class="main-breadcrumb">
13       <ol class="breadcrumb">
14         <li class="breadcrumb-item"><a href="index.html">Home</a></li>
15         <li class="breadcrumb-item"><a href="javascript:void(0)">User</a></li>
16         <li class="breadcrumb-item active" aria-current="page">Employer profile</li>
17       </ol>
18     </nav>
19     <!-- /Breadcrumb -->
20
21     <div class="row gutters-sm">
22       <div class="col-md-4 mb-3">
23         <div class="card">
24           <div class="card-body">
25             <div class="d-flex flex-column align-items-center text-center">
26
27               <!-- Assuming $profile->dp contains the path to the profile picture -->
28               <!-- resources/views/upload.blade.php -->
29               <!-- resources/views/profiles/show.blade.php -->
30               @if($profile->dp)
31                 
32               @else
33                 
34               @endif
35
36

```

2.3. Edit page: This Blade template code is designed for a user profile editing page. It uses Bootstrap for styling and includes a form to upload a profile picture. The page is divided into two main sections: the left section displays the user's basic information, including an editable form for updating links to the website, GitHub, and Facebook. The right section contains a form for updating personal details like name, email, phone, mobile, and address. There is also an option to update the user's skills, displayed as a comma-separated list.

In summary, this template provides a comprehensive interface for users to edit and update their profile information, including a profile picture, contact links, and personal details. The use of Bootstrap ensures a clean and responsive design.

```
web.php change.blade.php X home.blade.php
resources > views > products > change.blade.php > div.container > div.main-body > div.row > div.col-lg-4 > div.card > div.card-body > div.d-flex.flex-column.align-items-center.text-center
1
2 <link rel='stylesheet' href='https://cdn.jsdelivr.net/npm/bootstrap@4.4.1/dist/css/bootstrap.min.css'>
3
4 <script src='https://cdn.jsdelivr.net/npm/bootstrap@4.4.1/dist/js/bootstrap.bundle.min.js'></script>
5
6
7 <div class="container">
8   <div class="main-body">
9     <div class="row">
10       <div class="col-lg-4">
11         <div class="card">
12           <div class="card-body">
13             <div class="d-flex flex-column align-items-center text-center">
14               {{--  --}}
15               <!-- resources/views/upload.blade.php -->
16               <form method="POST" action="{{ route('upload.image', ['email' => $profile->email]) }}" enctype="multipart/form-data">
17                 @csrf
18                 <input type="file" name="image" accept="image/*">
19                 <button type="submit">Upload Image</button>
20               </form>
21             </div>
22
23
24
25
26
27
28           <div class="mt-3">
29             <h4>{{ $profile->name }}</h4>
30             <p class="text-secondary mb-1">Full Stack Developer</p>
31             <p class="text-muted font-size-sm">Bay Area, San Francisco, CA</p>
32             <button class="btn btn-primary">Follow</button>
33             <button class="btn btn-outline-primary">Message</button>
34           </div>
35         </div>
36       </div>

```

2.4.Filter and search page:

This Blade template creates a user interface for searching and filtering users based on their skills. It includes a search bar and a skill filter dropdown, and displays search results in a table with columns for name, email, and skills. The design is enhanced with Bootstrap styling and JavaScript libraries such as jQuery and Popper.js for improved interactivity. Overall, it provides a user-friendly way to explore and filter user profiles by skill set.

```
resources > views > search > index.blade.php > html > body > div.container.mt-4 > form.mb-4 > div.form-group > select.form-control
1 <!-- resources/views/search/index.blade.php -->
2
3
4 <!DOCTYPE html>
5 <html lang="en">
6 <head>
7   <meta charset="UTF-8">
8   <meta name="viewport" content="width=device-width, initial-scale=1.0">
9   <title>Search Users by Skill</title>
10  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
11 </head>
12 <body>
13
14 <div class="container mt-4">
15   <h1 class="mb-4">Search Users by Skill</h1>
16
17   <form action="{{ route('search') }}" method="GET" class="mb-4">
18     <div class="input-group">
19       <input type="text" name="query" class="form-control" placeholder="Enter skill...">
20       <div class="input-group-append">
21         <button type="submit" class="btn btn-primary">Search</button>
22       </div>
23     </div>
24   </form>
25
26   <!-- Filter Section -->
27   <form action="{{ route('search.index') }}" method="GET" class="mb-4">
28     <div class="form-group">
29       <label for="filter">Filter by Skill:</label>
30       <select name="filter" class="form-control">
31         <option value="">Select Skill</option>
32         <option value="Machine Learning">Machine Learning</option>
33         <option value="Gaming">Gaming</option>
34         <option value="Blockchain">Blockchain</option>
35       </select>
36     </div>
37   </form>
38 </div>
```

2.5 Search Result page: This Blade template generates a search results page for users based on a specified skill query. It displays the results in a table format, including columns for name and email. The design is styled with Bootstrap, and the template provides a message if no results are found. Overall, it offers a clean and organized presentation of user search outcomes.


```
resources > views > search > results.blade.php > html > body > div.container.mt-4 > table.table.table-bordered.table-striped > tbody > tr
14
15 @if($results->isEmpty())
16 <p class="alert alert-info">No results found.</p>
17 @else
18 <table class="table table-bordered table-striped">
19 <thead class="thead-dark">
20 <tr>
21 <th>Name</th>
22 <th>Email</th>
23 <!-- Add more columns based on your needs -->
24 </tr>
25 </thead>
26 <tbody>
27 @foreach($results as $result)
28 <tr>
29 <td>{{ $result->name }}</td>
30 <td>{{ $result->email }}</td>
31 <!-- Add more columns here if needed -->
32 </tr>
33 @endforeach
34 </tbody>
35 </table>
36 @endif
37 </div>
38
39
40
41 <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
42 <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"></script>
43 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
44 </body>
45 </html>
```

2.6 Employer's Profile: This Blade template code creates a user profile page using Bootstrap styles. It displays the user's personal information, including a profile picture, name, role, and contact details. The template includes links to the user's website, GitHub, and Facebook accounts. An "Edit" button redirects to a route for profile editing. The page also shows the user's skills if available, presenting them in a visually organized manner. The overall design is clean and user-friendly, following Bootstrap conventions for a responsive and visually appealing layout.

```

sources > views > products > home.blade.php > <div class="container"> <div class="main-body"> <div class="row gutters-sm"> <div class="col-md-8"> <div class="card mb-3">
1
2
3
4 <link rel='stylesheet' href='https://cdn.jsdelivr.net/npm/bootstrap@4.4.1/dist/css/bootstrap.min.css'>
5
6 <script src='https://cdn.jsdelivr.net/npm/bootstrap@4.4.1/dist/js/bootstrap.bundle.min.js'></script>
7
8 <div class="container">
9 <div class="main-body">
10
11 <!-- Breadcrumb -->
12 <nav aria-label="breadcrumb" class="main-breadcrumb">
13 <ol class="breadcrumb">
14 <li class="breadcrumb-item"><a href="index.html">Home</a></li>
15 <li class="breadcrumb-item"><a href="javascript:void(0)">User</a></li>
16 <li class="breadcrumb-item active" aria-current="page">Employer profile</li>
17 </ol>
18 </nav>
19 <!-- /Breadcrumb -->
20
21 <div class="row gutters-sm">
22 <div class="col-md-4 mb-3">
23 <div class="card">
24 <div class="card-body">
25 <div class="d-flex flex-column align-items-center text-center">
26
27 <!-- Assuming $profile->dp contains the path to the profile picture -->
28 <!-- resources/views/upload.blade.php -->
29 <!-- resources/views/profiles/show.blade.php -->
30 @if($profile->dp)
31 
32 @else
33 
34 @endif
35
36

```

3. Asif Ahnaf Chowdhury

Unverified users will trigger this frontend after logging in

```

@extends('layouts.app')
@section('content')

<div class="container">
<div class="row justify-content-center mt-5">
<div class="card">
<div class="card-header">
Verify Account
</div>
<div class="card-body">
Your account is not verified. Please check your email and click on the link to verify your account.
</div>
</div>
</div>
</div>

@endsection

```

Contact Us form:

```
div class="card shadow-lg">
  <div class="card-header">Contact us</div>
  <form method="post" action="{{route('contact.submit')}}">@csrf
    <div class="card-body">
      <div class="form-group">
        <label for="">Full Name</label>
        <input type="text" name="name" class="form-control">
      </div>
      <div class="form-group">
        <label for="">Email</label>
        <input type="text" name="email" class="form-control">
      </div>
      <div class="form-group">
        <label for="">Subject</label>
        <input type="text" name="uSubject" class="form-control">
      </div>
      <div class="form-group">
        <div class="mb-3">
          <label for="validationTextarea">Your message</label>
          <textarea class="form-control" id="validationTextarea" name="content" maxLength=3000
            required></textarea>
        </div>
      </div>
      <br>
      <div class="form-group text-center">
        <button class="btn btn-primary" type="submit">Send</button>
      </div>
    </div>
  </form>
</div>
```

After contacting our mail will receive the mail in the following structure:

```
class ContactMail extends Mailable
{
    use Queueable, SerializesModels;

    /**
     * Create a new message instance.
     */
    1 reference | 0 overrides
    public function __construct(public string $name, public string $email, public string $uSubject, public string $content)
    {
        //
    }

    /**
     * Get the message envelope.
     */
    0 references | 0 overrides
    public function envelope(): Envelope
    {
        return new Envelope(
            subject: 'Contact Mail from Job portal user',
        );
    }

    0 references | 0 overrides
    public function build()
    {
        return $this->subject('Contact Mail from Job portal user')->replyTo($this->email)->markdown('emails.contact');
    }
}
```

```

1  @component('mail::message')
2  <b>Contact from {{ $name }} <{{ $email }}> </b><br>
3
4  <b>Subject: </b>{{ $uSubject }}<br>
5
6  <p>{{ $content }}</p> <br>
7
8  Thanks,<br>
9  {{ config('app.name') }}
10 @endcomponent

```

Forgot password:

```


### User will get a mail in the following structure:



```

@component('mail::message')
<p>Hello {{ $user->name }}</p>

@component('mail::button', ['url' => url('reset/'.$user->remember_token)])
Reset Password
@endcomponent

@endcomponent

```



27


```

Password reset:

```
<div class="card shadow-lg">
  <div class="card-header">Reset Password</div>
  <form action="" method="post">@csrf
    <div class="card-body">
      <div class="form-group">
        <label for="">New passwords</label>
        <input type="password" name="password" class="form-control" id='Password' required>
      </div>
      <br>
      <div class="form-group text-center">
        <button class="btn btn-primary" type="submit">Confirm new password</button>
      </div>
    </div>
  </form>
</div>
```

Easy apply:

```
@foreach ($circulars as $circular)
<div class="card mt-3">
  <div class="card-header">
    <div class="circular-title">
      <h2>{{ $circular->title }}</h2>
    </div>
  </div>
  <div class="card-body">
    <h3>{{ $circular->company->name }}</h3>
  </div>
  <div class="card-footer">
    <a class="btn btn-primary" onclick="openModal('{{ $circular->id }}')">View Details</a>
    <form action="{{route('apply')}}" method="POST" style="display: inline;">
      @csrf
      <input type="hidden" name="circularid" value="{{ $circular->id }}">
      <input type="hidden" name="companyid" value="{{ $circular->companyid }}">
      @if (auth()->user()->user_type == 'seeker')
      <button type="submit" class="btn btn-success">Apply</button>
      @endif
    </form>
  </div>
</div>
```

View details then apply:

```
<div id="detailsModal{{ $circular->id }}" class="modal">
  <div class="modal-content">
    <div class="modal-header">
      <h2>Details for {{ $circular->title }}</h2>
    </div>
    <div class="modal-body">
      <!-- Details content goes here -->
      <p>{{ $circular->detail }}</p>
    </div>

    <div class="modal-footer justify-content-center">

      <form action="{{route('apply')}}" method="POST" style="display: inline;">
        @csrf
        <input type="hidden" name="circularid" value="{{ $circular->id }}">
        <input type="hidden" name="companyid" value="{{ $circular->companyid }}">

        @if (auth()->user()->user_type == 'seeker')
        <button type="submit" class="btn btn-success">Apply</button>
        @endif

      </form>

      <button class="close-button btn btn-success"
        onclick="closeModal('{{ $circular->id }}')">Close</button>

    </div>
  </div>
```

Seeker's view count:

```
<button class="btn btn-primary">View</button>
<button class="btn btn-outline-primary disabled">Views: {{ auth()->user()->views
  }}</button>
</div>
v>
```

Applicant list:

```
<div class="card shadow-lg">
  <div class="card-header">Applicants</div>
  <div>
    {{-- // --}}
    <ul>
      @foreach($applicants as $applicant)
        <li>
          <a href="{{ route('view-profile', $applicant->seekerID) }}">{{ $applicant->seekerID
          }}</a>
        </li>
        <!-- Display other applicant information -->
      @endforeach
    </ul>
  </div>
</div>
```

Show applicant's profile:

```
<div class="d-flex flex-column align-items-center text-center">
  
  <div class="mt-3">
    <h4>{{ $seeker->name }}</h4>
  </div>
</div>
```

Backend Development

4. A N M Jubair Tanvir

Employer's Registration: Given the required information, the user can register as an employer. For this, a createEmployer() function is created. Another function is storeEmployer(). This function stores all the information of the employer in the central database. The information is stored in the user table of the database based on user type. If the user type is employer then he/she will be stored as an employer.

```
1 reference | 0 overrides
public function createEmployer()
{
    return view('user.employer-register');
}
```

```
1 reference | 0 overrides
public function storeEmployer()
{
    request()->validate([
        'name' => ['required', 'string', 'max:250'],
        'email' => ['required', 'string', 'email', 'max:250', 'unique:users'],
        'password' => ['required']
    ]);
    $user = User::create([
        'name' => request('name'),
        'email' => request('email'),
        'password' => bcrypt(request('password')),
        'user_type' => self::JOB_POSTER
    ]);

    $user->sendEmailVerificationNotification();
    return redirect()->route('login');
}
```


User login and logout: A user can login based on his user type. By checking the user type from the stored information in the database, one can login. If the user is already registered, then the system will check this authentication. For this, the auth() function is used. If the user is seeker then the system will redirect to his profile and if the user is an employer, the system will redirect the user to his profile. If the given information is wrong or false by the user then system won't let him/her to login.

```
1 reference | 0 overrides
public function login()
{
    return view('user.login');
}
```

```
1 reference | 0 overrides
public function postLogin(Request $request)
{
    $request->validate([
        'email' => 'required',
        'password'=> 'required'
    ]);

    $credentials = $request->only('email','password');

    if (auth::attempt($credentials)) {
        if (auth::user()->user_type == 'seeker') {
            return redirect()->route('job_seeker');
        } elseif (auth::user()->user_type == 'employer') {
            return redirect()->route('employer.job_circular.index');
        }
    }

    return 'Wrong email or password';
}
```

```

1 reference | 0 overrides
public function logout()
{
    auth()->logout();
    return redirect()->route('login');
}

```

CV add, update and download: To add CV there is a function named updateCV(). This function stores the CV and can update.

```

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

3 references | 0 implementations
class JobSeekerController extends Controller
{
    1 reference | 0 overrides
    public function show()
    {
        // dd("here at job seeker controller");
        return view('job_seeker.show');
    }

    1 reference | 0 overrides
    public function updateCV(Request $request)
    {
        $request->validate([
            'cv' => 'required|mimes:pdf,doc,docx|max:2048',
        ]);

        $cvPath = $request->file('cv')->store('cv');

        Auth::user()->update(['cv_path' => $cvPath]);

        return redirect()->route('job_seeker')->with('success', 'CV updated successfully.');
```

Set Time limits for circulars, Show all the Job Circulars, Edit the Circular, Delete: To implement these features there are several functions. There are store, update and download functions to do so.

```
2 references | 0 overrides
public function index()
{
    $jobCirculars = JobCircular::where('employer_id', Auth::id())->orderBy('created_at', 'desc')->get();
    return view('employer.job_circular.index', compact('jobCirculars'));
}

2 references | 0 overrides
public function create()
{
    return view('employer.job_circular.create');
}

2 references | 0 overrides
public function store(Request $request)
{
    // Validation and saving logic

    $request->validate([
        'title' => 'required|string',
        'description' => 'required|string',
        'deadline' => 'required|date',
    ]);

    $jobCircular = new JobCircular([
        'employer_id' => Auth::id(),
        'title' => $request->input('title'),
        'description' => $request->input('description'),
        'deadline' => $request->input('deadline'),
    ]);

    $jobCircular->save();

    // Schedule task to delete expired job circulars
    Artisan::call('schedule:run');

    return redirect()->route('employer.job_circular.index')->with('success', 'Job circular created successfully.');
```

```
2 references | 0 overrides
public function edit(JobCircular $jobCircular)
{
    return view('employer.job_circular.edit', compact('jobCircular'));
}

2 references | 0 overrides
public function update(Request $request, JobCircular $jobCircular)
{
    $request->validate([
        'title' => 'required|string',
        'description' => 'required|string',
        'deadline' => 'required|date',
    ]);

    $jobCircular->update([
        'title' => $request->input('title'),
        'description' => $request->input('description'),
        'deadline' => $request->input('deadline'),
    ]);

    return redirect()->route('employer.job_circular.index')->with('success', 'Job circular updated successfully.');
```

```

2 references | 0 overrides
public function download(JobCircular $jobCircular)
{
    $file = storage_path('app\\circulars\\' . $jobCircular->id . '.pdf');

    return response()->download($file, 'JobCircular.pdf');
}

```

Live Statistics: There is a function named index. This function counts the number of job seekers, employers and circulars. To do that, it uses the count() function.

```

2 references | 0 implementations
class StatisticsController extends Controller
{
    1 reference | 0 overrides
    public function index()
    {
        $totalJobSeekers = User::where('user_type', 'seeker')->count();
        $totalEmployers = User::where('user_type', 'employer')->count();
        $totalCirculars = JobCircular::count();

        return view('statistics.index', compact('totalJobSeekers', 'totalEmployers', 'totalCirculars'));
    }
}

```

5. Ashfaq Ahmad Saad

5.1 Seeker's Registration :These PHP functions, storeSeeker and storeEmployer, are part of a Laravel application for user registration. They validate input data for name, email, and password, create new users with hashed passwords, set user types for job seekers and job posters respectively, send email verification notifications, and redirect users to the login route after successful registration.

```
public function storeSeeker()
{
    request()->validate([
        'name' => ['required','string','max:250'],
        'email' => ['required','string','email', 'max:250', 'unique:users'],
        'password' => ['required']
    ]);
    $user = User::create([
        'name' => request('name'),
        'email' => request('email'),
        'password' => bcrypt(request('password')),
        'user_type' => self::JOB_SEEKER
    ]);

    $user->sendEmailVerificationNotification();
    return redirect()->route('login');
}

1 reference | 0 overrides
public function storeEmployer()
{
    request()->validate([
        'name' => ['required','string','max:250'],
        'email' => ['required','string','email', 'max:250', 'unique:users'],
        'password' => ['required']
    ]);
    $user = User::create([
        'name' => request('name'),
        'email' => request('email'),
        'password' => bcrypt(request('password')),
        'user_type' => self::JOB_POSTER
    ]);

    $user->sendEmailVerificationNotification();
    return redirect()->route('login');
}

1 reference | 0 overrides
public function storeSeeker()
{
    request()->validate([
        'name' => ['required','string','max:250'],
        'email' => ['required','string','email', 'max:250', 'unique:users'],
        'password' => ['required']
    ]);
    $user = User::create([
        'name' => request('name'),
        'email' => request('email'),
        'password' => bcrypt(request('password')),
        'user_type' => self::JOB_SEEKER
    ]);

    $user->sendEmailVerificationNotification();
    return redirect()->route('login');
}
```

5.2 Seeker's Profile:

This PHP function, show, is part of a Laravel application. It retrieves a user profile from the database based on the provided email address. If the profile is not found, it returns a 404 error. Otherwise, it renders a view ('products.home') with the retrieved profile data.

```
class ProfileController extends Controller
{
    public function show($email){
        $profile = Profile::where('email', $email)->first();

        if (!$profile) {
            abort(404);
        }

        return view('products.home', ['profile' => $profile]);
    }
}
```

5.3. Edit page: This set of functions is part of a Laravel application's ProfileController. Here's a summary of each function:

`change(Profile $profile):`

1. Renders a view ('products.change') to update user profile information.

`update(Profile $profile, Request $request):`

2. Validates and updates profile information based on the provided request data.

Redirects to the updated profile view with a success message.

`dpshow($email):`

3. Retrieves a user profile based on the provided email address.

Renders the 'profiles.show' view with the profile data.

`uploadImage(Request $request, $email):`

4. Validates and uploads a user's profile image.

Updates the profile with the image path and redirects to the profile view with a success message.

`updateSkills(Profile $profile, Request $request):`

5. Validates and updates user skills based on the provided request data.

Redirects to the profile view with a success message after the update.

```
p > Http > Controllers > ProfileController.php
21 public function change(Profile $profile){
22
23     return view('products.change',['profile' => $profile]);
24 }
25
26 public function update(Profile $profile, Request $request)
27 {
28     $data = $request->validate([
29         'email' => 'required|email|unique:profiles,email,' . $profile->id,
30         'name' => 'nullable|string',
31         'phone' => 'nullable|integer',
32         'mobile' => 'nullable|integer',
33         'address' => 'nullable|string',
34         'web' => 'nullable|string',
35         'git' => 'nullable|string',
36         'fab' => 'nullable|string',
37     ]);
38
39     $profile->update($data);
40
41     return redirect(route('profiles.show', ['email' => $data['email']]))->with('success', 'Profile Updated Successfully');
42 }
43
44 public function dpshow($email)
45 {
46     $profile = Profile::where('email', $email)->first();
47
48     if (!$profile) {
49         abort(404);
50     }
51
52     return view('profiles.show', ['profile' => $profile]);
53 }
54
55 public function uploadImage(Request $request, $email)
56 {
57
```

5.4.Filter and search page:search(Request \$request):

A.This function handles a search based on a user-provided query for skills.

It retrieves the query from the request, searches the 'skills' attribute in the 'Profile' model using the 'LIKE' operator, and retrieves the results.

The results, along with the original query, are then passed to the 'search.results' view.

index(Request \$request):

B.This function is associated with an index page that allows filtering profiles based on skills.

It retrieves a filter value from the request and applies it to the 'skills' attribute in the 'Profile' model using the 'LIKE' operator.

The filtered records are then passed to the 'search.index' view along with a list of all distinct skills available in the profiles.

```
public function search(Request $request)
{
    $query = $request->input('query');
    $results = Profile::where('skills', 'LIKE', '%' . $query . '%')->get();

    return view('search.results', compact('results', 'query'));
}

public function index(Request $request)
{
    $filter = $request->input('filter');
    $records = Profile::query(); // Use query() instead of all()

    if ($filter) {
        $records->where('skills', 'LIKE', '%' . $filter . '%');
    }

    $records = $records->get();

    $allSkills = Profile::distinct()->pluck('skills')->filter()->flatten();

    return view('search.index', compact('records', 'allSkills')); // Pass 'allSkills' to the view
}
```

5.5 Search Result page:

The search function is part of a Laravel application and performs a search based on a user-provided query. It uses the Profile model to search for profiles where the 'skills' attribute contains the specified query. The search results, along with the original query, are then passed to the 'search.results' view for display.

```
public function search(Request $request)
{
    $query = $request->input('query');
    $results = Profile::where('skills', 'LIKE', '%' . $query . '%')->get();

    return view('search.results', compact('results', 'query'));
}
```


2.6 Employer's Profile: This PHP function, show, is part of a Laravel application. It retrieves a user profile from the database based on the provided email address. If the profile is not found, it returns a 404 error. Otherwise, it renders a view ('products.home') with the retrieved profile data.

```
class ProfileController extends Controller
{
    public function show($email){
        $profile = Profile::where('email', $email)->first();

        if (!$profile) {
            abort(404);
        }

        return view('products.home', ['profile' => $profile]);
    }
}
```

6. Asif Ahnaf Chowdhury

6.1. Verify email address

After a new user is updated to our database we send a default verification mail by Laravel to verify our user and give them other accesses.

```
$user->sendEmailVerificationNotification();
return redirect()->route('login');
```

Verifying: The email generates a hash function and sends it along with user's id. Laravel's middleware matches these data with the DB and verifies the user.

```
Route::get('/email/verify/{id}/{hash}', function (EmailVerificationRequest $request) {
    $request->fulfill();

    return redirect('/dashboard');
})->middleware(['auth', 'signed'])->name('verification.verify');
```

We give access to the dashboard after checking if the user is verified:

```
Route::get('/dashboard', [DashboardController::class, 'index'])->middleware('verified')->name('dashboard');
Route::post('/dashboard', [DashboardController::class, 'apply'])->name('apply');
Route::get('/verify', [DashboardController::class, 'verify'])->name('verification.notice');
```

6.2. Contact us

Through a given form, users can contact the website's official email. We have used Laravel's Mail support whenever mail needs to be delivered.

```

3 references | 0 implementations
class ContactController extends Controller
{
    1 reference | 0 overrides
    public function show()
    {
        return view('contacts.show');
    }
    1 reference | 0 overrides
    public function submit(ContactRequest $request)
    {
        Mail::to('jobportal@gmail.com')->send(new ContactMail($request->name, $request->email, $request->uSubject, $request->content));
        return redirect()->route('contact.show');
    }
}

```

Setting up constraints in Contact Us form request as most email services provide limited storage:

```

0 references | 0 overrides
public function rules(): array
{
    return [
        'name' => 'required|string|max:255',
        'email' => 'required|email',
        'uSubject' => 'required|string|max:255',
        'content' => 'required|string',
    ];
}

```

6.3. Forgot Password

Submitting a wrong password and redirecting to 'Forgot password' by checking through auth support:

```

$credentials = $request->only('email','password');

if (auth::attempt($credentials)) {
    return redirect()->intended(route('dashboard'));
}
else
{
    return redirect()->intended(route('password.request'));
}
// return 'Wrong email or password';
}

```

Sending an email to reset password with a token of verification:

```

1 reference | 0 overrides
public function postForgot(Request $request)
{
    $user = User::where('email','=', $request->email)->first();
    if(!empty($user)){
        $user->remember_token = Str::random(60);
        $user->save();
        Mail::to($user->email)->send(new ForgotPasswordMail($user));
        echo "Email sent successfully";
        return redirect()->back()->with('success', 'Email sent successfully')
    }
    echo "Email does not exist";
    return redirect()->back()->with('error', 'Email does not exist');
}

```

```

2 references | 0 implementations
class ForgotPasswordMail extends Mailable
{
    use Queueable, SerializesModels;
    2 references
    public $user;

    /**
     * Create a new message instance.
     */
    1 reference | 0 overrides
    public function __construct($user)
    {
        $this->user = $user;
    }

    /**
     * Get the message envelope.
     */
    0 references | 0 overrides
    public function envelope(): Envelope
    {
        return new Envelope(
            subject: 'Forgot Password Mail',
        );
    }
}

```

Giving access to update password after matching token:

```

1 reference | 0 overrides
public function reset($token)
{
    $user = User::where('remember_token','=', $token)->first();
    if(!empty($user)){
        $data['user'] = $user;
        return view('user.reset', $data);
    }
    else
    {
        abort(404);
    }
}

```

```

Route::get('reset/{token}', [UserController::class, 'reset']->name('reset'));
Route::post('reset/{token}', [UserController::class, 'postReset']);

```

Updating new password to DB and double checking token:

```

1 reference | 0 overrides
public function postReset($token, Request $request)
{
    $user = User::where('remember_token','=', $token)->first();
    if(!empty($user)){
        $user->password = Hash::make($request->password);
        if(empty($user->email_verified_at)){
            $user->email_verified_at = date('Y-m-d H:i:s');
        }
        $user->remember_token = Str::random(60);
        $user->save();
        return redirect()->route('login');
    }
    else
    {
        abort(404);
    }
}

```

6.4. Easy apply

Seeker applying and automatically sending their information to the employer.

```
1 reference | 0 overrides
public function apply()
{
    $cid = request('circularid');
    $comid = request('companyid');
    // $compEmail = DB::select('select companyid
    Applicant::create([
        'circularID' => $cid,
        'seekerID' => auth()->user()->email,
        'employerEmail' => $comid,
    ]);
    return redirect()->route('dashboard');
}
```

6.5. View count

Updating seeker's views every time an employee visits their profile:

```
1 reference | 0 overrides
public function show($email)
{
    $seeker = User::where('email',$email)->first();
    if (!$seeker) {
        return abort(404);
    }
    $affectedRows = User::where('email', $email)->update([
        'views' => ($seeker->views + 1),
    ]);

    return view('applicant.show', compact('seeker'));
}
```

6.6. View Applicants and their Profile

Getting the list of applicants who applied to the current employer's circular:

```
public function list()
{
    $emp = auth()->user();
    $applicants = Applicant::where('employerEmail', $emp->id)
->get();
    return view('applicant.list', compact('applicants'));
}
```

Getting the viewer side's access to the profile through seeker's email:

```
Route::get('/view-profile/{email}', [ProfileController::class, 'show']->name('view-profile'));
Route::get('/applicants', [ProfileController::class, 'list']->name('applicants'));
```

Technology (Framework, Languages)

Framework: Laravel

Languages: HTML, CSS, JS, PHP, and MySQL.

GitHub Repository

Link: <https://github.com/AhnafCh/Job-Portal>

Individual Contribution

| ID | Name | Contributions (Features) |
|----------|----------------------|---|
| 21301524 | A N M Jubair Tanvir | <ul style="list-style-type: none"> ● Module-1:Employer's Registration, User login and logout ● Module-2: CV Add/Update, CV Download ● Module-3: Post a Circular, Set Time limits for circulars, Show all the Job Circulars, Edit the Circular, Delete ● Module-4: Live statistics |
| 21301665 | Ashfaq Ahmad Saad | <ul style="list-style-type: none"> ● Module-1:Seeker's Registration ● Module-2: Add Name, address, Bio, etc, Add and update Profile picture ● Module-3: Add Name, Profile picture, About etc. ● Module-4: Filtering,Searching |
| 21301510 | Asif Ahnaf Chowdhury | <ul style="list-style-type: none"> ● Module-1: Forgot password, Verify mail. ● Module-2: Profile view count. ● Module-3: View Applicants' Profile and CV. ● Module-4: Easy Apply, and Contact Us. |