`SRI KRISHNA COLLEGE OF THCHNOLOGY
An Autonomous Institution | Accredited by NAAC with 'A' Grade
Affiliated to Anna University | Approved by AICTE
KOVAIPUDUR, COIMBATORE 641042

# ONLINE JOB APPLICATION PORTAL
# FULL STACK APPLICATION

## A PROJECT REPORT

**Submitted by**

| | |
|---|---|
| **ABI C** | **727821TUIT006** |
| **KABARTHINI K** | **727821TUIT044** |
| **DHARSINE S** | **727821TUIT018** |

in partial fulfillment for the award of the degree

Of

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECHNOLOGY**

**NOVEMBER 2023**

# CERTIFICATE

# BONAFIDE CERTIFICATE

Certified that this project report **"ONLINE JOB APPLICATION PORATL-FULL STACK APPLICATION"** is the Bonafide work of **ABI C, KABARTHINI K, DHARSHINE S** who carried out the project work under my supervision.

**SIGNATURE**

**Mrs.P.Dhivya**
**SUPERVISOR**
Assistant professor,
Department of
Information Technology,
Sri Krishna College of Technology,
Coimbatore-641042 .

**SIGNATURE**

**Dr.S .SIAMALA DEVI**
**HEAD OF THE DEPARTMENT**
Assistant Professor,
Department of
Information technology,
Sri Krishna College of Technology,
Coimbatore-641042 .

Certified that the candidates were examined by us in the Project Work viva- voce examination held on _____ at Sri Krishna College of Technology, Coimbatore -641 042.

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

# ACKNOWLEGDEMENT

# ACKNOWLEDGEMENT

Dedicating this project to the **ALMIGHTY GOD** whose abundant grace and mercies enabled its successful completion.

We extend our deep gratitude to our beloved Principal, **Dr. M. G. Sumithra,** for her kindness and unwavering support throughout the project work.

We are grateful to our beloved Dean Academics affairs and assessment

**Dr. R. Rameshkumar,** for his tireless and relentless support.

We extend our heartfelt thanks to our beloved Dean Accreditation and Ranking **Dr. P. Manju,** for her advice and ethics inculcated during the entire period of our study.

We would like to express our deep gratitude to **Dr. J. Shanthini,** Head of Computing Science and **Dr. S. Siamala Devi,** Head of Information Technology , for her exceptional dedication and care towards success of this project.

We would like to extend our heartfelt gratitude to our Project guide **Mrs.P.Dhivya,** Assistant Professor, Department of Information Technology for her valuable guidance and suggestion in all aspects that aided us to ameliorate our skills.

We are thankful to all those who have directly and indirectly extended their help us in completing this project work successfully.

# ABSTRACT

# ABSTRACT

A job application portal is a vital digital platform that revolutionizes the recruitment process for job seekers and employers alike. It serves as a centralized hub where individuals seeking employment can effortlessly explore job listings, submit applications, and manage their professional profiles. For employers, the portal offers a convenient avenue to post job openings, review candidate applications, and communicate effectively with potential hires.

Key features include user authentication for secure access, comprehensive job listings categorized by industry and location, and robust application management tools allowing candidates to submit resumes and cover letters seamlessly.

Additionally, the portal facilitates communication between job seekers and employers, streamlining the hiring process and enhancing transparency. With automated notifications and an applicant tracking system, employers can efficiently manage applications, screen candidates, and schedule interviews. Overall, the job application portal represents a powerful solution that optimizes recruitment efforts, improves accessibility to job opportunities, and fosters a more transparent and efficient hiring process for all stakeholders involved.

# TABLE OF CONTENT

# TABLE OF CONTENT

# LIST OF FIGURES

# LIST OF FIGURES

# LIST OF ABBREVATIONS

# LIST OF ABBREVATIONS

| ABBREVATIONS | ACRONYMS |
|---|---|
| HTML | **HYPERTEXT MARKUP LANGUAGE** |
| CSS | **CASCADING STYLE SHEET** |
| JS | **JAVASCRIPT** |
| DOM | **DOCUMENT OBJECT MODEL** |

# INTRODUCTION

# CHAPTER 1
# INTRODUCTION

This chapter introduces the concept of job application portals and explores their transformative impact on the recruitment process. By providing an overview of their key features, benefits, and implications for both job seekers and employers, this chapter sets the stage for a comprehensive examination of how these portals have reshaped the recruitment landscape

## 1.1 PROBLEM STATEMENT

Traditional recruitment methods lack efficiency, accessibility, and transparency, hindering both job seekers and employers. Fragmented processes and biases in hiring often result in suboptimal outcomes. Addressing these challenges requires leveraging technology to create a streamlined and inclusive recruitment process.

## 1.2 OVERVIEW

This report provides a comprehensive analysis of job application portals and their transformative impact on modern recruitment practices. It explores the key features, benefits, and challenges associated with these platforms. Through case studies and industry insights, the report highlights the potential of job application portals to revolutionize talent acquisition strategies.

## 1.3 OBJECTIVE

The objectives of online job application portal are to evaluate the efficiency, accessibility, and transparency facilitated by job application portals in the recruitment process. Additionally, it aims to examine the role of these portals in promoting data-driven decision-making and to provide practical recommendations for their effective implementation. Through these objectives, the report seeks to enhance understanding and utilization of job application portals in modern talent acquisition efforts.

The report aims to assess the impact of job application portals on reducing recruitment costs and improving time-to-fill metrics. It also seeks to explore how these platforms contribute to enhancing the candidate experience and employer branding efforts. By addressing these objectives, the report aims to provide actionable insights for organizations looking to leverage job application portals for optimizing their recruitment strategies and achieving competitive advantage. By addressing these objectives, the report seeks to provide actionable insights and recommendations for organizations looking to leverage job application portals as strategic tools.

# METHODOLOGIES

# METHODOLOGIES

## 2.1 HISTORY

Throughout history, the evolution of recruitment methods has been closely intertwined with technological advancements and societal changes. Traditional recruitment practices, dating back centuries, relied on word-of-mouth referrals, newspaper advertisements, and in-person networking events to connect job seekers with potential employers. However, the emergence of the internet in the late 20th century revolutionized the recruitment landscape, paving the way for online job boards and early iterations of job application portals. Websites like Monster.com and CareerBuilder.com emerged as pioneers in the field, offering job seekers a digital platform to search for opportunities and submit applications online.

## 2.2 PURPOSE

The purpose of an online job application portal is to revolutionize and streamline the recruitment process for both job seekers and employers. By providing a centralized platform where job seekers can browse, apply, and manage their applications, and employers can post, review, and communicate with candidates, these portals aim to enhance efficiency, accessibility, and transparency in the hiring process.

Additionally, online job application portals serve to reduce recruitment costs, improve time-to-fill metrics, and enhance the candidate experience, ultimately contributing tostronger employer branding effort. Overall, the purpose of an online job application portal is to transform the way talent is sourced and acquired in the digital age, driving success.

# ENVIRONMENTAL SETUP

## 3.1 INSTALLATION OF VISUAL STUDIO CODE

A software package known as an integrated development environment, or IDE, integrates the essential tools required to design and test software. Developers use a range of tools for writing, creating, and testing software code. Examples of development tools include text editors, code libraries, compilers, and test environments. Developers can give commands immediately on VSC without needing to open a terminal process thanks to a built-in terminal system on VSC, which makes it easier to check for errors. The fact that VSC comes with a built-in GIT command that enables diff checking, staging files, and committing directly from the editor is another reason to choose it.

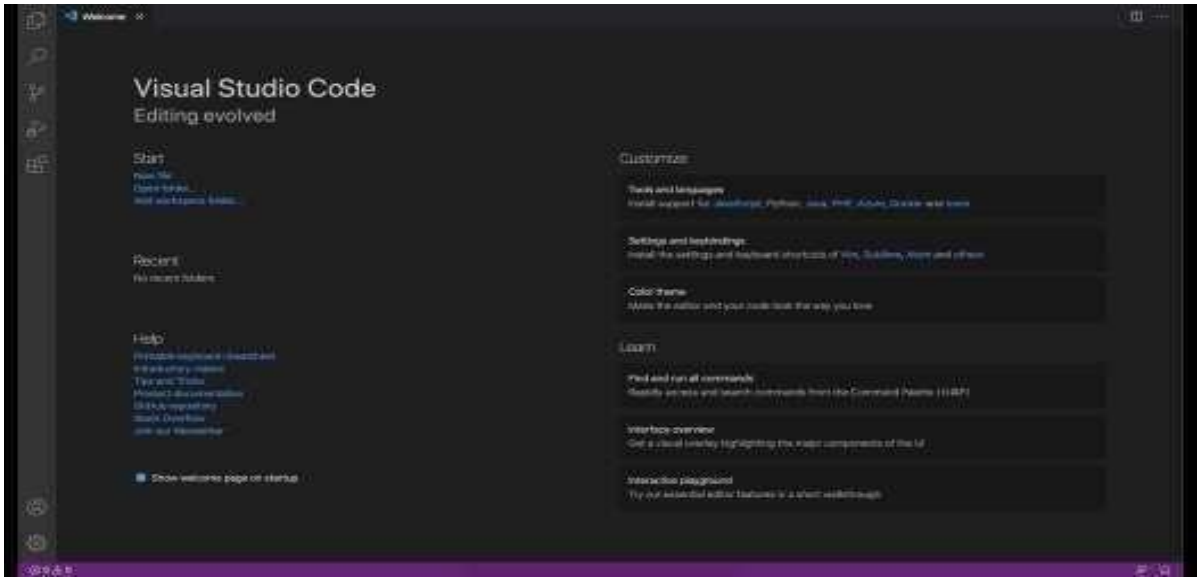


**Fig 3.1 Visual Studio**

# TOOLS AND TECHNOLOGIES USED

# CHAPTER 3

# ENVIRONMENT SETUP

Multiple technologies and applications were employed in the "buy casa" project.

4.1. HTML

4.2. CSS

4.3. JavaScript

## 4.1 HTML

Hypertext Markup Language is referred to as HTML. To create web pages, this language is utilized. The creation of interactive and responsive pages on the pages is also supported by this language, along with other languages like CSS, PHP, JAVASCRIPT, etc. Simply put, HTML5 is HTML in a more modern form. It enables brand-new functions, characteristics, HTML components, video and audio, as well as 2D and 3D images, all of which are helpful to users and web designers in their ability to easily add brand-new functions to websites. Figure illustrates the structure of HTML5.

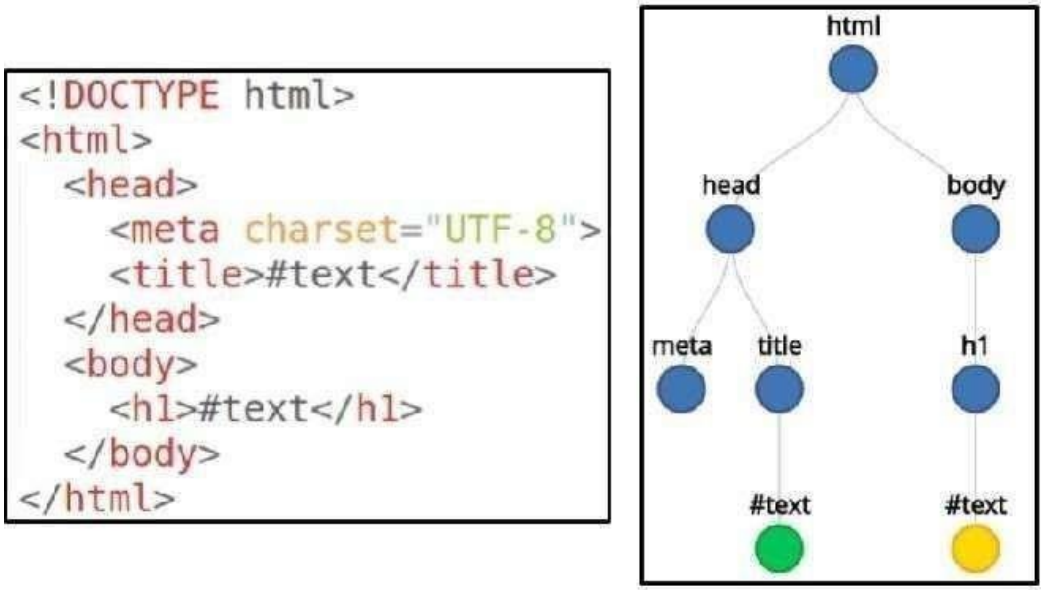


**Fig 4.1 HTML**

## 4.2 CSS – CASCADING STYLE SHEET

Cascading Style Sheets is the short name for CSS. CSS is used to specify the design, layout, and display variants for web pages on various devices and screen sizes. The basic organization of CSS is as

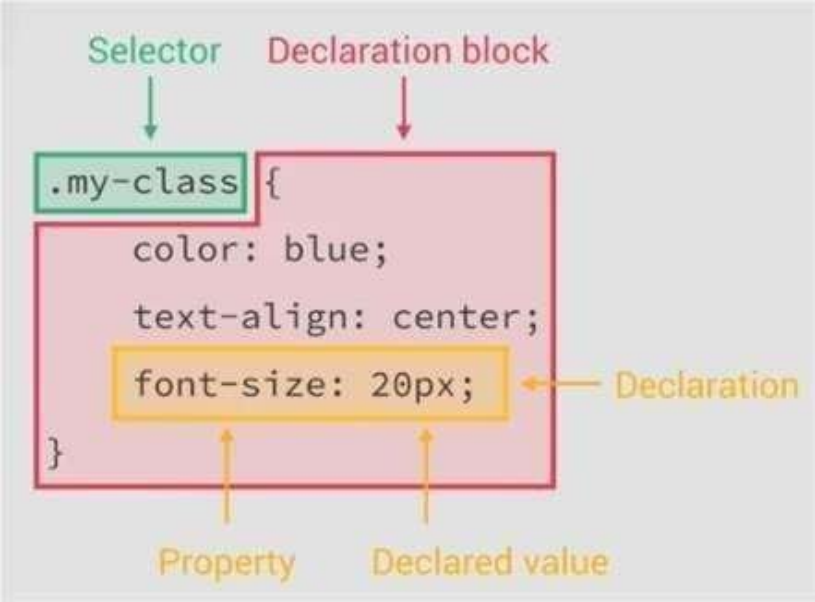


**Fig 4.2 CSS**

## 4.3 JAVASCRIPT

A JavaScript-based UI development library is called React. It is managed by Facebook and an open source development community. The library made its debut in May 2013 and is currently one of the frontend libraries for web development that is most frequently used. Our React project folders are as follows:


**Fig 4.3 Javascript**

The upcoming chapter will explain the UML diagrams involved such as usecase activity, sequence, ER, ERR diagrams.

# IMPLEMENTATION

## 5.1 FRONT-END

A front-end developer is a type of software developer who specializes in creating and designing the user interface (UI) and user experience (UX) of websites and web applications. The primary responsibility of a front-end developer is to ensure that the visual and interactive aspects of a website or application are user-friendly.


**Fig 5.7 Front-End**

## 5.1.1 HOME PAGE

A Home page is a page where the projects first and foremost page begins where it contains the project description and it contains top bar, side bar which displays activities, contact us, logout.

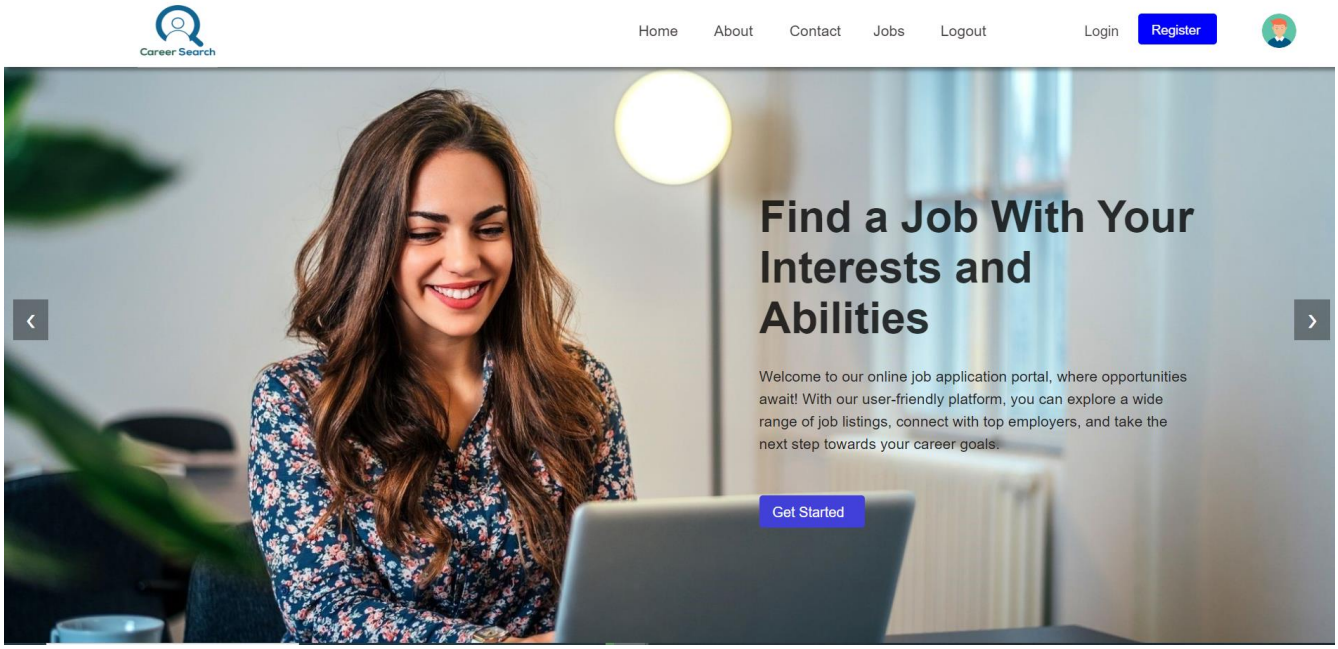


**Fig 5.8 Home page**

Js code :

```
import React, { useState, useEffect } from 'react';
import '../assets/css/Home.css'; // Import your CSS file for styling
function Home() {

  return (
    <div>

      <UserHeader />
      <div className="slider-container">
      {slides.map((slide, index) => (
<div
  key={index}
```

```
className={`slide ${index === currentSlide ? 'active' : ''}`}

        style={{ backgroundImage: `url(${slide.image})` }}

       <div className="content">
          <h5>{slide.title}</h5>
          <p>{slide.content}</p>
          <button>Get Started</button>
       </div>
    </div>
))}

            <button className="prev" onClick={prevSlide}>&#10094;</button>
            <button className="next" onClick={nextSlide}>&#10095;</button>
          </div>
          <div className='jobDes'>
            <div className='tittle'>
              <h5>2000+ Dream Job Openings</h5>
              <p>Explore thousands of job opportunies and find your dream job with our
comprehensive job search platform</p>
            </div>
          <div className='jobdetails'>
            <div className='jobcon1'>
              <div className='conbox'>
                <div className='image_con'>
                  <img src={web_developer} height="100px" width="100px"/>
                  <h4>Website Developer</h4>
                  <p>Remote work $4,000-$12,000</p>
                  <br/>
                  <p>we are seeking a talented web developer to join our team to create an engaging
and effective company profile landing page.</p>
                  <div className='but_con'>
                    <button>ReactJS</button>
                    <button>Javascript</button>
                    <button>Web Developer</button>
                  </div>
                </div>
              </div>
              <div className='conbox'>
                <div className='image_con'>
                  <img src={UI} height="100px" width="100px"/>
                  <h4>Senior UI Designer</h4>
                  <p>Remote work $4,000-$12,000</p>
                  <br/>
                  <p>we are looking for a talented and experienced UI-User Interface Designer to
develop a new social crawdfunging .</p>
                  <div className='but_con'>
                    <button>UI Designer</button>
                    <button>Adabe XD</button>
                    <button>UI UX Designer</button>
```

```
                </div>


            <div className='conbox'>
              <div className='image_con'>
                <img src={UX} height="100px" width="160px" style={{ marginLeft: '-35px' }}/>
                <h4>UX Copywriter</h4>
                <p>Remote work $4,000-$12,000</p>
                <br/>
                <p>we are seeking a talented UX Copywriter to join our team to create an engaging
and effective food delivery product  .</p>

                <div className='but_con'>
                  <button>Copywriter</button>
                  <button>Content Writer</button>
                  <button>UX Copy</button>
                </div>
              </div>
            </div>

          </div>
          <div className='jobcon2'>
          <div className='conbox'>
              <div className='image_con'>
                <img src={PM} height="100px" width="100px"/>
                <h4>Senior Project Manager</h4>
                <p>Remote work $4,000-$12,000</p>
                <br/>
                <p>we are looking for a experienced Project Manger to join our team to take care all
of incoming inquires from many clients.</p>
                <div className='but_con'>
                  <button>Project Manger</button>
                  <button>Scrum Master</button>
                  <button>PM</button>
                </div>
              </div>
            </div>
            <div className='conbox'>
              <div className='image_con'>
                <img src={app} height="100px" width="100px"/>
                <h4>App Developer</h4>
                <p>Remote work $4,000-$12,000</p>
                </div>
              </div>
            </div>
```

## 5.1.2 LOGIN PAGE

A login is a set of credentials used to authenticate a user. Most often, these consist of a username and password. However, a login may include other information, such as a PIN number, passcode, or passphrase. Some logins require a biometric identifier, such as a fingerprint or retina scan.

Logins are used by websites, computer applications, and mobile apps. They are a security measure designed to prevent unauthorized access to confidential data. When a login fails (i.e, the username and password combination does not match a user account), the user is disallowed access. Many systems block users from even trying to log in after multiple failed login attempts.

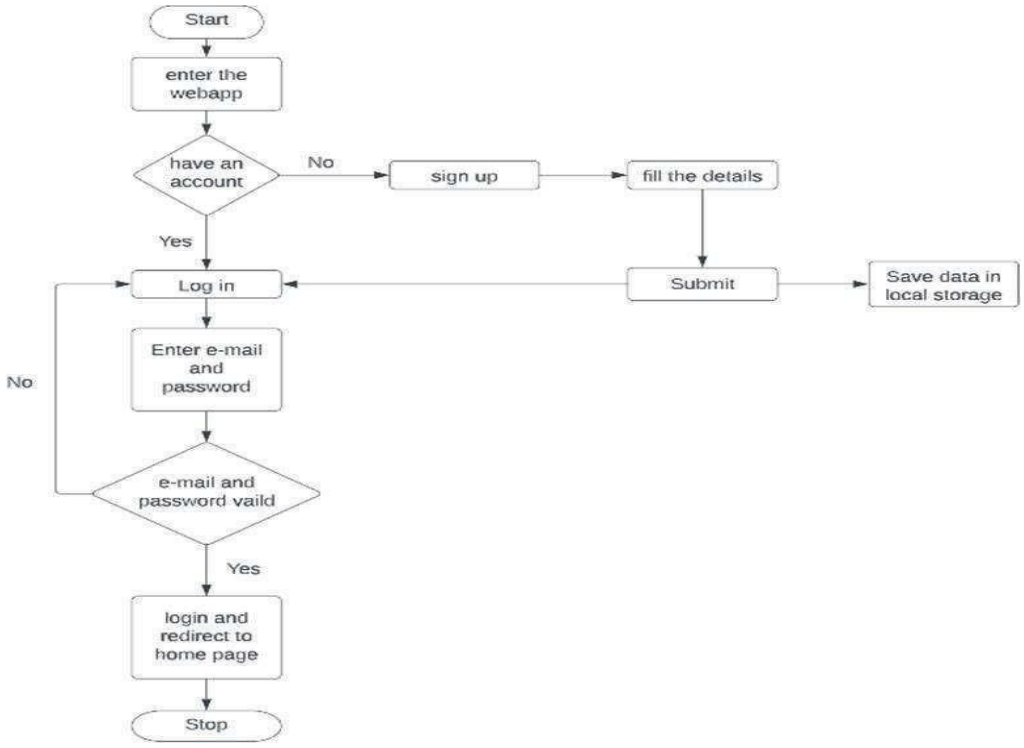


**Fig 5.9 Login Page Flowchart**

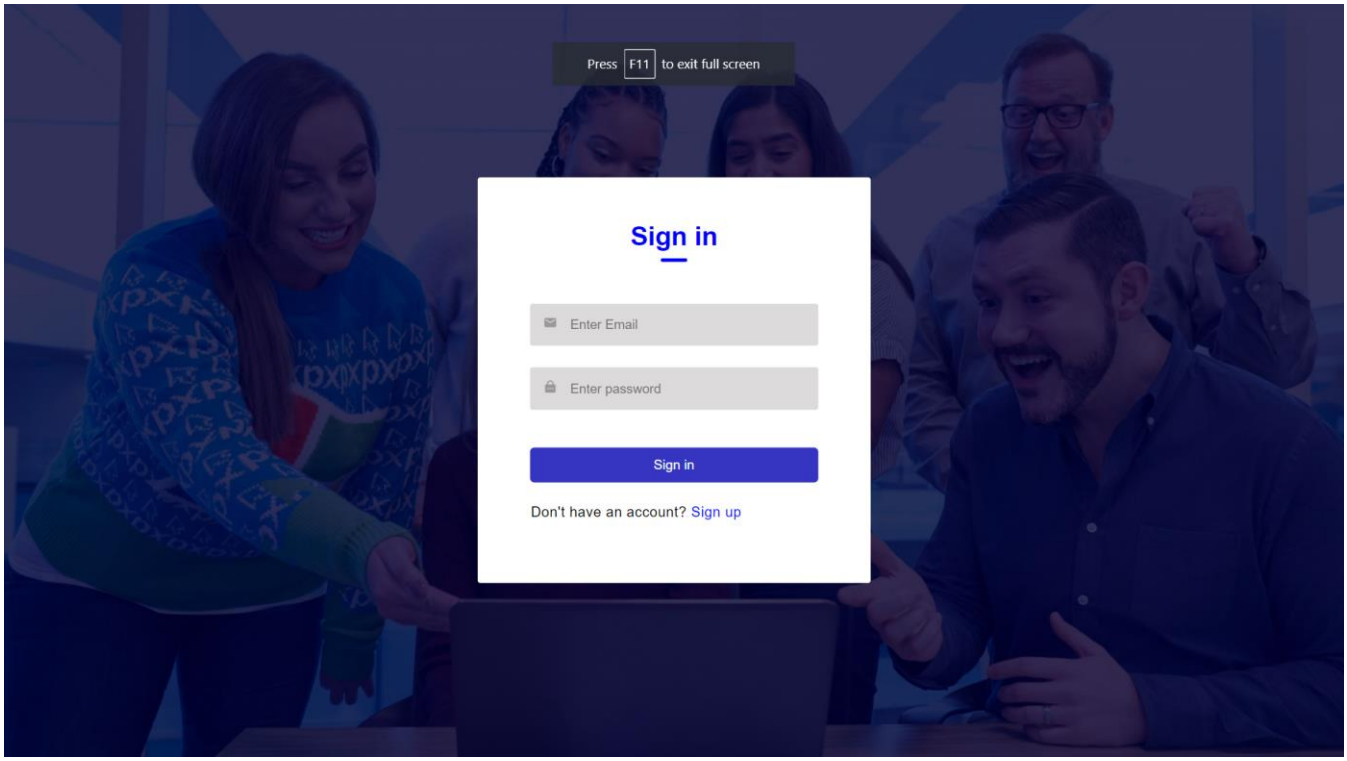


**Fig 5.10 Login Page**

## Js Code :

```
import React,{useState} from 'react'
import './Newregister.css';
import email_icon from '../Fpage/email.png'
import person_icon from '../Fpage/person.png'
import password_icon from '../Fpage/password.png'
import { useDispatch } from 'react-redux';
import { Link, useNavigate } from 'react-router-dom';
import { login } from './userSlice';
export default function Newregister() {

    const nav=useNavigate("");
    const[username,setUsername]=useState("");
    const[password,setPassword]=useState("");
    const dispatch=useDispatch();
```

```
    {
      e.preventDefault();
      if(username.length==0||password.length==0){
      alert("Enter all fields")
      }
      else if (username && password) {
        dispatch(login(username)); nav("/");
      }

  }
   return (
 <div className='bgrect'>
   <div className='bgrect1'>
    <div className='pic'><img src="https://drivefordti.com/wp-/>
    </div>
    <div className='boxy33'></div>
    <div className='bgrect2'>
    <div className='underline2'></div>

    <div className='header'>
        <div className='text'>Login</div>
        <div className='underline'></div>
      </div>
      <div className='fpassword'>Don't you have an account?</div>

    <div className='form1'>
    <form onSubmit={handleSubmit}>
<div className='image1'>
    <img src={person_icon} height={20}></img>
    </div>
    <div className='image2'>
    <img src={password_icon} height={25}></img>
    </div>
```

```
    <input type="text"  placeholder= 'username' value={username}
onChange={ (e)=>setUsername(e.target.value)} name='username'
required/><br></br>


    <input type="Password" placeholder='password' value={password}
onChange={ (e)=>setPassword(e.target.value)}
name='password'/><br></br>
   <Link to="/"> <button type="submit" className='button1'
onClick={handleSubmit}>Login</button></Link>
    </form></div>
    <Link to="/log"><button type="submit"
className='button2'>Register</button></Link>
    <div className='line3'></div>


   <div className='line4'></div>
    <div className='line5'></div>
 </div>
    </div>
    </div>
  )
}
```

### 5.1.3 REGISTER PAGE

A register page enables users and organizations to independently register and gain access to your system. It is common to have multiple signup pages depending on the types of people and organizations you want to register. In this article you will learn about the different types of signup pages, how to configure them and related functionality. Global Administrator access is required to create and modify signup pages.

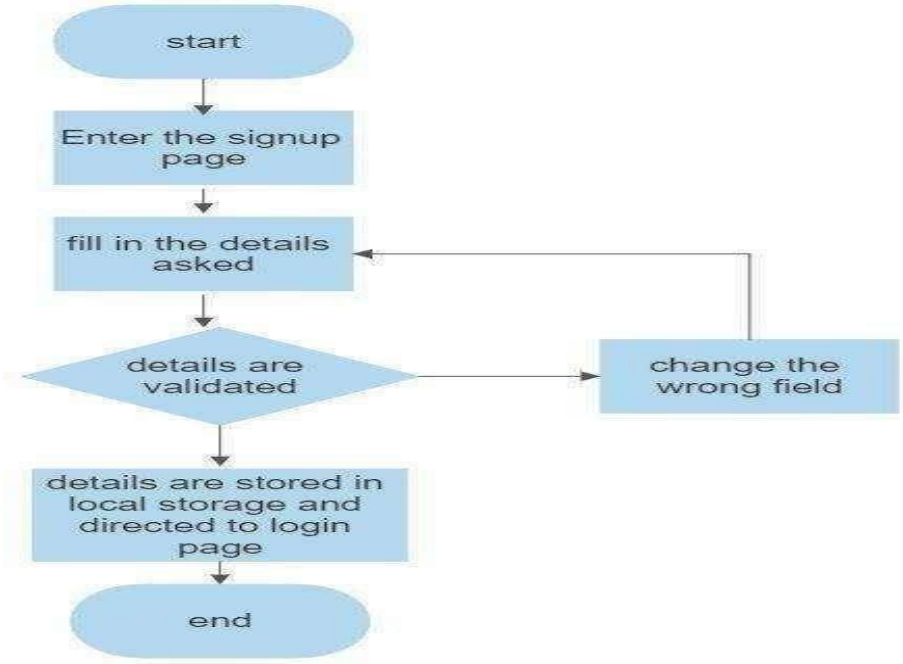


**Fig 5.11 Register Page Flowchart**

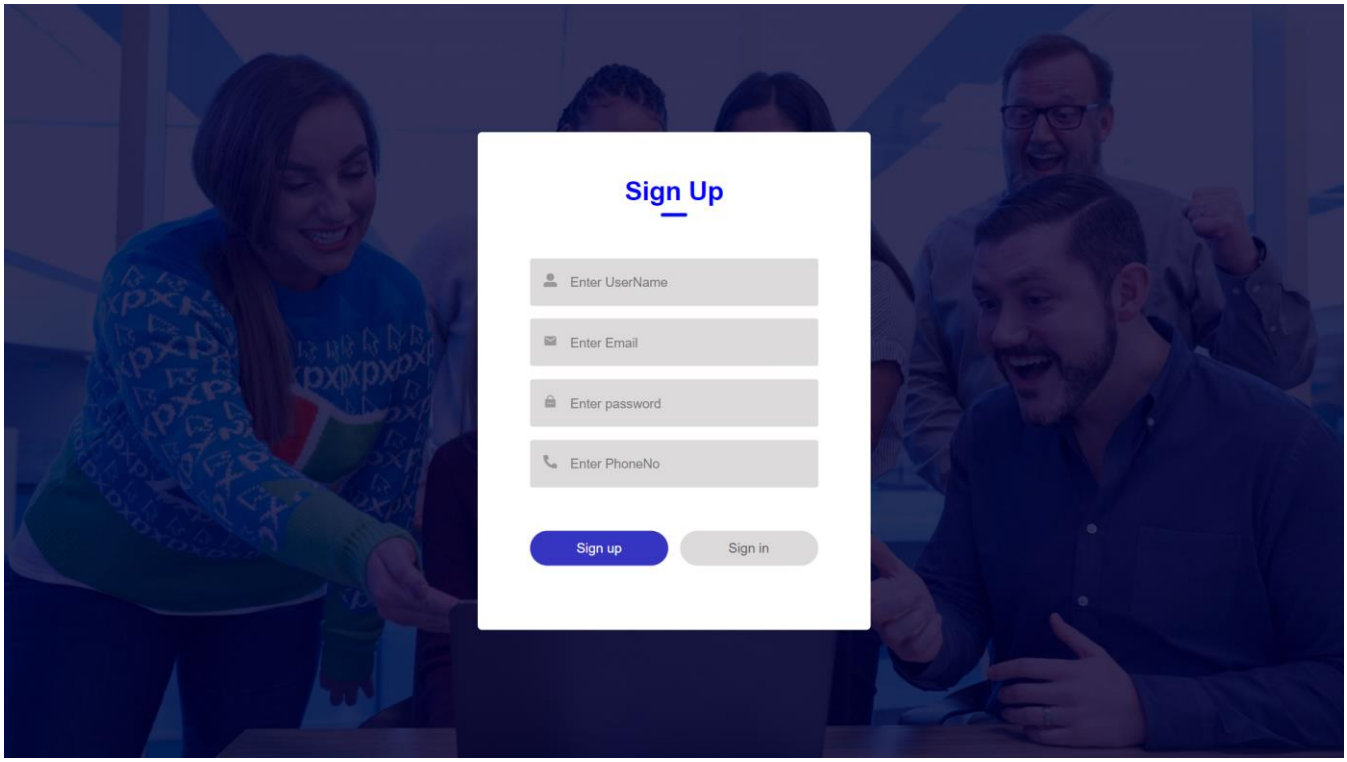


**Fig 5.12 Register Page**

## Js code:

```
import React from 'react'
import { Link } from 'react-router-dom';
import { useState } from 'react';
import { useNavigate } from 'react-router-dom';

import '../Fpage/person.png';
import '../Fpage/password.png';
import '../Fpage/email.png';
import '../Fpage/person.png';

export default function Register() {
   const nav=useNavigate("");
  const[username,setUsername]=useState("");
const[password,setPassword]=useState("");
const[gender,setGender]=useState("");
const[weight,setWeight]=useState("");
const[selects,setSelects]=useState();
const[age,setAge]=useState("");
  const authenticate=(e)=>{
    e.preventDefault();
    if(username.length===0||password.length==0){
      alert("Enter all fields")
    }

    else {
      nav("/log")
     }
  }
  return (
    <div className='full'>

       <div className='photo'>

            <div className='glass'>

            <div className='form2'>
            <form >
            <input type="text" placeholder= 'Username' value={username} onChange={
(e)=>setUsername(e.target.value)} name='username' required/><br></br>
```

```
            <input type="text" placeholder='Gender' value={gender} onChange={
(e)=>setGender(e.target.value)} name='gender'/><br></br>
            <input type="number" placeholder='Weight' value={weight} onChange={
(e)=>setWeight(e.target.value)} name='weight'/><br></br>
            <input type="number" placeholder='Age' value={age} onChange={
(e)=>setAge(e.target.value)} name='age'/><br></br>

            <Link to ="/login"><button type="submit" className='button'
onClick={authenticate}>Register</button></Link></form>
            </div>
            </div>
      </div>
    </div>
  )
}
```

## 5.1.4 JOB PAGE

This page display the list of jobs available for application.

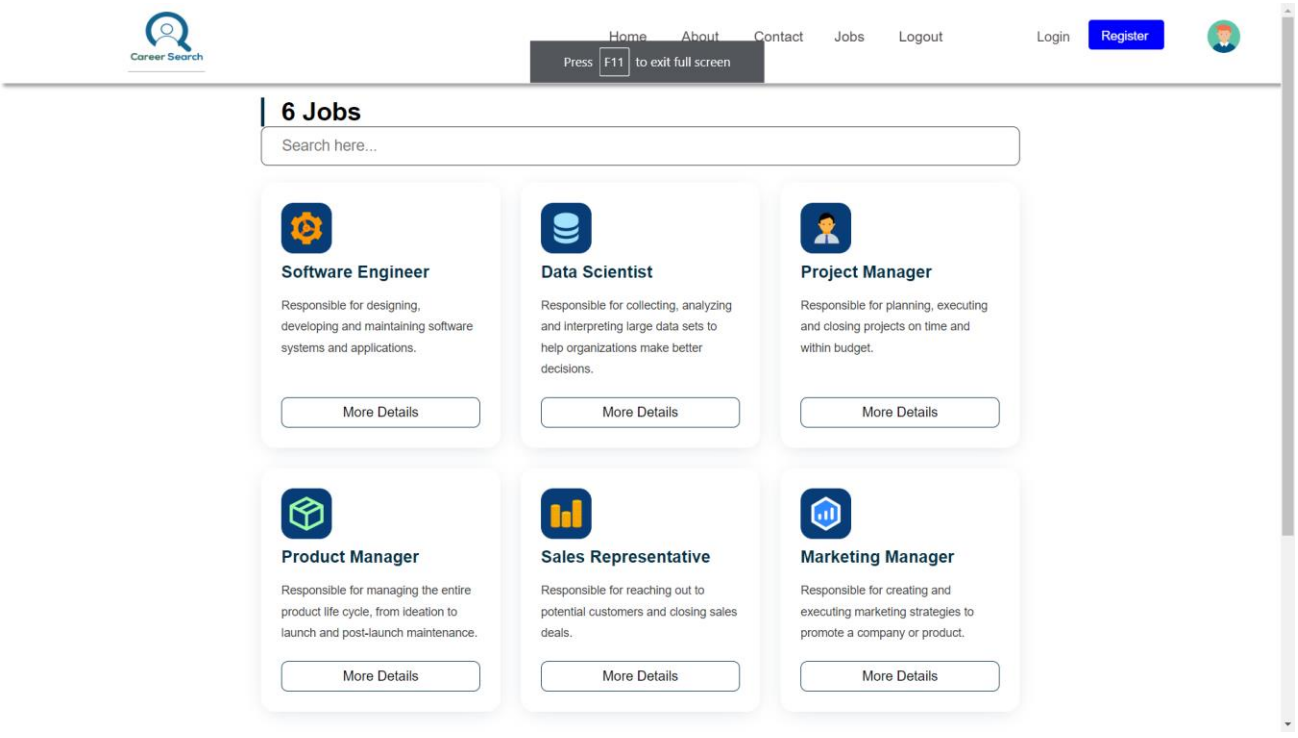


**Fig 5.13 Job Page**

**Js code :**

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import Nave from './Nave';
function TodayPage() {
  const [container, setContainer] = useState('Cup');
  const token = localStorage.getItem('token');
  const username = localStorage.getItem('username');
  const [lastDailyTotal, setLastDailyTotal] = useState(null);

 useEffect(() => {
    axios
      .get(http://localhost:8080/auth/history/${username})
      .then((response) => {
        if (response.data.length > 0) {
          setLastDailyTotal(response.data[response.data.length - 1].dailyTotal);
        }
      })
      .catch((error) => {
        console.error('Error fetching last dailyTotal:', error);
      });
  }, [username])
  const handleLogIntake = (selectedContainer) => {
    let iconAmount = 0;
    if (selectedContainer === 'Cup') {
      iconAmount = 250;
    } else if (selectedContainer === 'Glass') {
      iconAmount = 500;
    } else if (selectedContainer === 'Bottle') {
      iconAmount = 750;
    }
    const newTotal = lastDailyTotal !== null ? lastDailyTotal + iconAmount : iconAmount;
    setLastDailyTotal(newTotal);
    axios
      .post(http://localhost:8080/auth/log-container/${username}, {
        container: selectedContainer,
      })
      .then((response) => {

      })
```

```
.catch((error) => {
        console.error('Error logging intake:', error);
      });
  };
  return (
    <div>
      <div className="goalpage">
        <h2>Log Water Intake</h2>
      <div>
</select>
        </div>
        <button onClick={() => handleLogIntake(container)}>Log Intake</button>
          <label>Container:</label>
          <select value={container} onChange={(e) => setContainer(e.target.value)}>
            <option value="Cup">Cup</option>
            <option value="Glass">Glass</option>
            <option value="Bottle">Bottle</option>

          <label>Container:</label>
          <select value={container} onChange={(e) => setContainer(e.target.value)}>
            <option value="Cup">Cup</option>
            <option value="Glass">Glass</option>
            <option value="Bottle">Bottle</option>




        <h3>Last Daily Total</h3>
        <p>{lastDailyTotal !== null ? lastDailyTotal : 'Loading...'}</p>
        <Nave />
      </div>
    </div>
  );
}

export default TodayPage;
```

### 5.1.5 DETAILS PAGE

This page displays the details of the job.

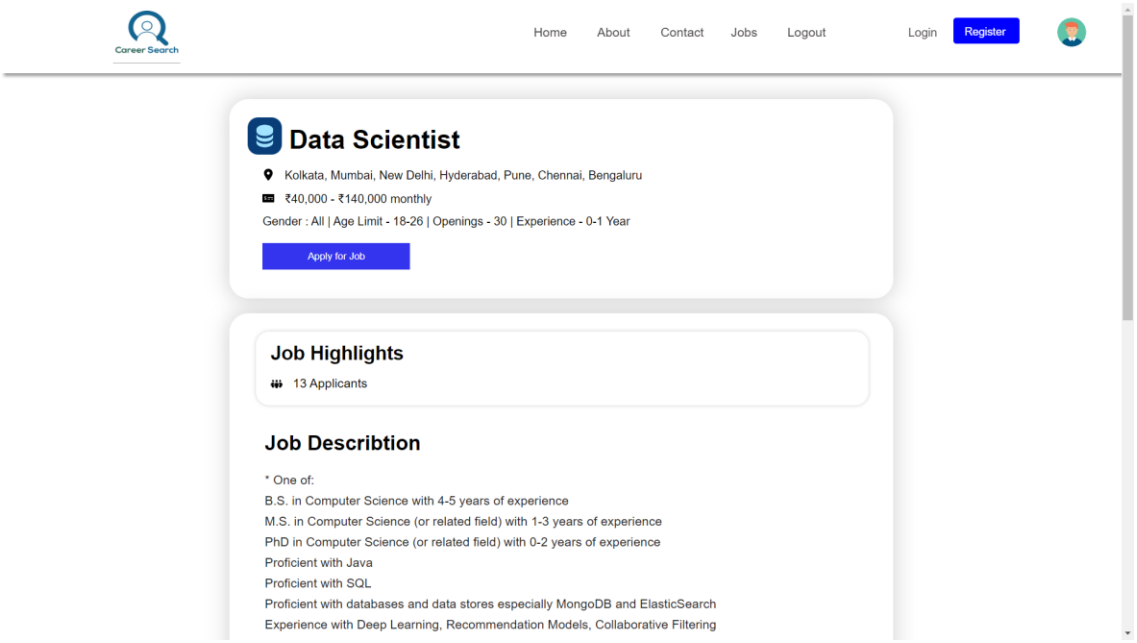


**Fig 5.14 Job details page**

## Js Code :

```
import React,{useState} from 'react'
import Chart from "react-apexcharts";
import './History.css'

export default function History() {
  const   [state,setState]=useState({
  options: {
     chart: {
       id:"basic-bar"
     },

 xaxis: {
       categories: ["Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"],
     },
     colors: ["black","black","black","black","black","black"],
   },
   series: [
```

```
      {
        name: "series-1",
        data: [2.5,1.7,2.1,3.4,4.0,3.0]
      }
    ]
  }
)
  return (
    <div className="hisfull">
    <div className='chart'>
        <div className='col-4'>
        <Chart
          options={state.options}
          series={state.series}
          type="area"
          width="900"
          height={400}
        />
        </div>
</div>
</div>
  )
}
```

### 5.1.6 JOB APPLICATION PAGE

Application process is designed to gather all the necessary information to help us understand your qualifications, experiences, and aspirations. This form is your opportunity to showcase your skills and tell us why you'd be a great fit for the position you're applying for. Please take your time to fill out each section accurately and thoroughly. Your responses will assist us in evaluating your candidacy and determining if there's a match between your profile and our company's needs.

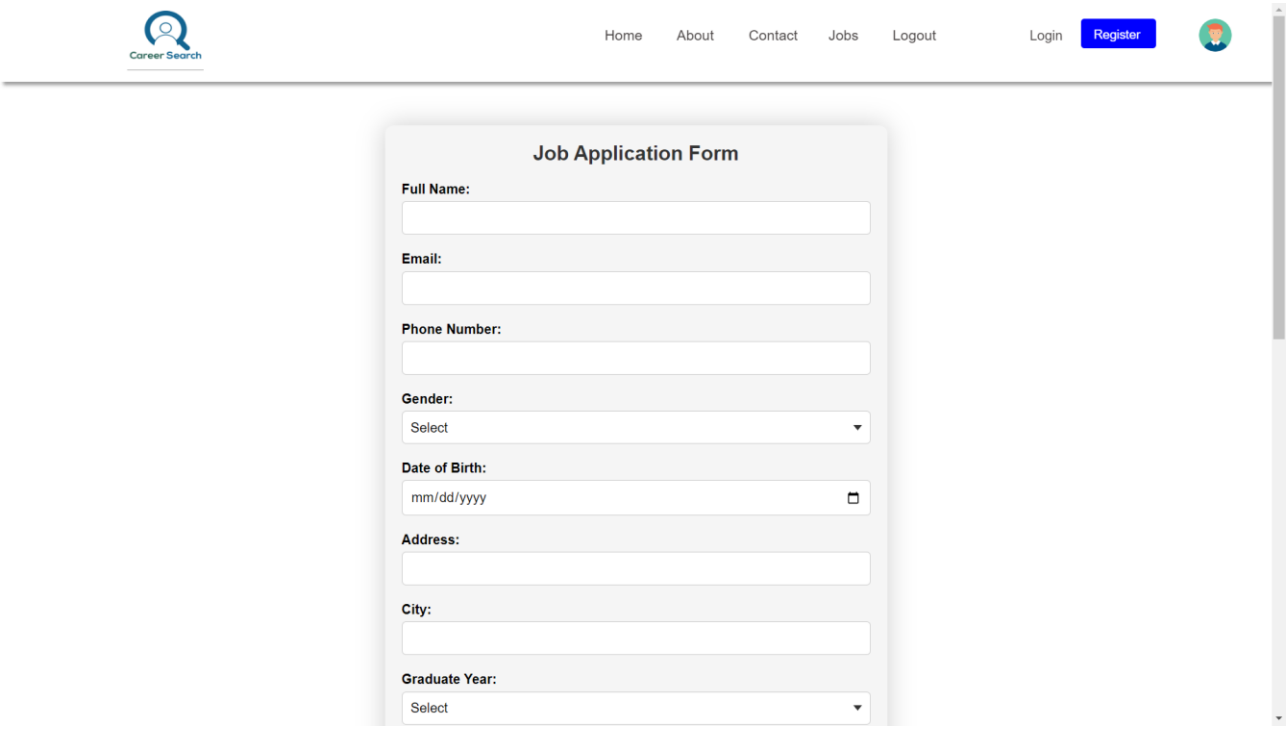


**Fig 5.15 Job Application page**

## Js Code :

```
import React ,{useState} from 'react'
import gify from '../Fpage/healthy-water-drinking-water.gif'
import './Suggestor.css'
import  suggcir from  '../Fpage/water_loader.gif'

export default function Suggestor() {
    const [age, setAge] = useState('');
    const [weight, setWeight] = useState('');
    const [activityLevel, setActivityLevel] = useState('');

    const [climate, setClimate] = useState('');
   const [waterIntake, setWaterIntake] = useState(null);
   const calculateWaterIntake = (e) => {
      e.preventDefault();
      const baseIntake = 0.035;
      const adjustedIntake = baseIntake * weight;
```

```
      let activityMultiplier = 1.0;
      if (activityLevel === 'low') {
      activityMultiplier = 0.5;
      } else if (activityLevel === 'moderate') {
        activityMultiplier = 0.75;
      } else if (activityLevel === 'high') {
        activityMultiplier = 1.0;
      }

      let climateMultiplier = 1.0;
      if (climate === 'hot') {
      climateMultiplier = 1.2;
      } else if (climate === 'cold') {
        climateMultiplier = 0.8;
      }

      const totalIntake = adjustedIntake * activityMultiplier * climateMultiplier;
      setWaterIntake(totalIntake);
    };
  return (
    <div className='suggfull'>
    <div className='giffy'><img src={gify} height={500} width={900}></img>
    <div className='sugg'>
    <h2>Water Intake Calculator</h2>

      <form onSubmit={calculateWaterIntake}>
      <button type="submit" >Calculate</button> <br />

<br/>
          <input type="number" placeholder='Enter your age' value={age}
 onChange={(e) => setAge(e.target.value)} />
     <br />
          <input type="number" placeholder='Enter your weight-kg)' value={weight}
 onChange={(e) => setWeight(e.target.value)} />
      <br />
          <select value={activityLevel} onChange={(e) =>
 setActivityLevel(e.target.value)}>
          <option value="">Select an Activity Level</option>
            <option value="low">Low</option>
            <option value="moderate">Moderate</option>
```

```
          <option value="high">High</option>
        </select>
       <br />

        <select value={climate} onChange={(e) => setClimate(e.target.value)}>
          <option value="">Select a Climate</option>
          <option value="average">Average</option>
          <option value="hot">Hot</option>
          <option value="cold">Cold</option>
        </select>

     </form></div>
     {waterIntake !== null && (
       <div className='recom'>
         <h3>daily intake<br></br> level</h3>
         <div className='result'>
         <p>{waterIntake.toFixed(2)}  l</p>
         </div>
       </div>
     )}
    </div>
   </div>
 )
```

### 5.1.7 PROFILE PAGE

The profile  Page is designed to collect valuable input from users, allowing them to share their thoughts, suggestions, and experiences to help us improve our services. Your feedback is essential in shaping our platform to better meet your needs and expectations.

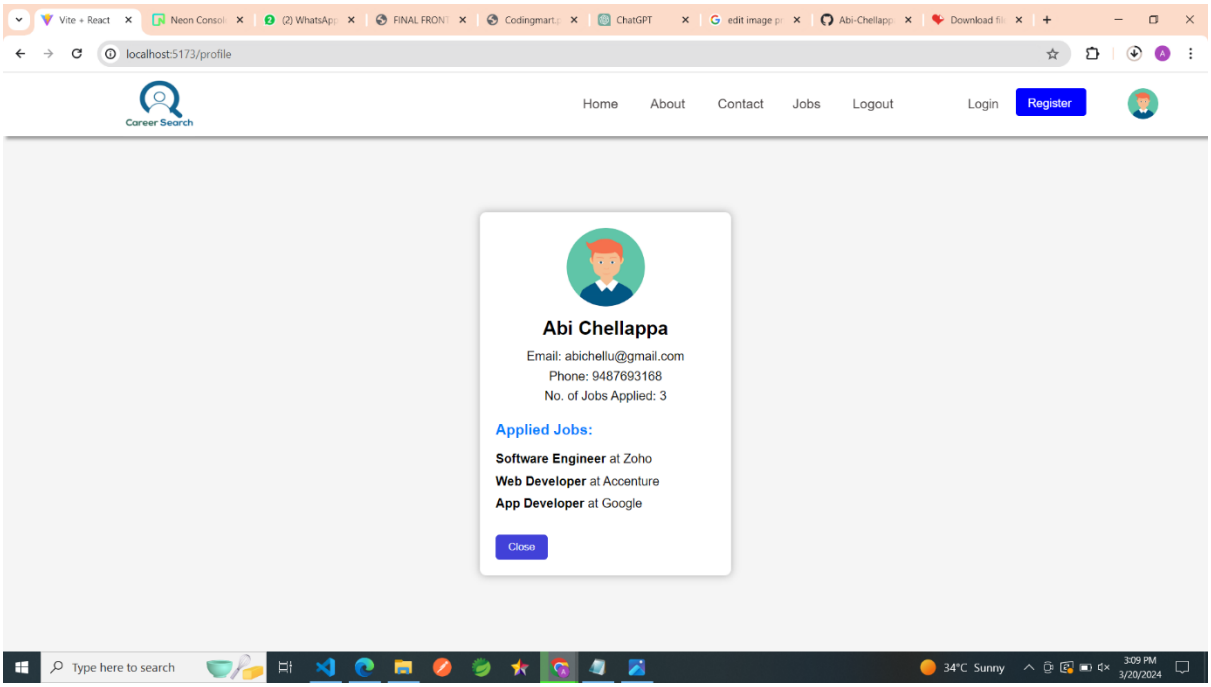


**Fig 5.16 Profile Page**

## Js Code :

```
import React, { useState } from 'react';
import './FeedbackForm.css'
function FeedbackForm() {
 const [feedback, setFeedback] = useState('');
 const [submitted, setSubmitted] = useState(false);
 const handleFeedbackChange = (e) => {
 setFeedback(e.target.value);
 };

 const handleSubmit = (e) => {
  e.preventDefault();
  setSubmitted(true);
 };
```

```
    return (
   <div className=”feedback-form”>
     <h2>Feedback Form</h2>
     {submitted ? (
       <p>Thank you for your feedback!</p>
     ) : (
       <form onSubmit={handleSubmit}>
         <label htmlFor=”feedback”>Please share your feedback:</label>
         <textarea
           id=”feedback”
           name=”feedback”
           rows="4"
           value={feedback}
           onChange={handleFeedbackChange}
           required
         ></textarea>
         <button type="submit">Submit Feedback</button>
       </form>
     )}
   </div>
  );
}

export default FeedbackForm;
```

## 5.1.8 ADMIN DASHBOARD PAGE

ThIS page serves as a comprehensive resource, addressing common questions and providing clear, concise answers to assist users in navigating our platform effortlessly. It's a valuable reference to quickly find solutions and gain insights into various aspects of our services.

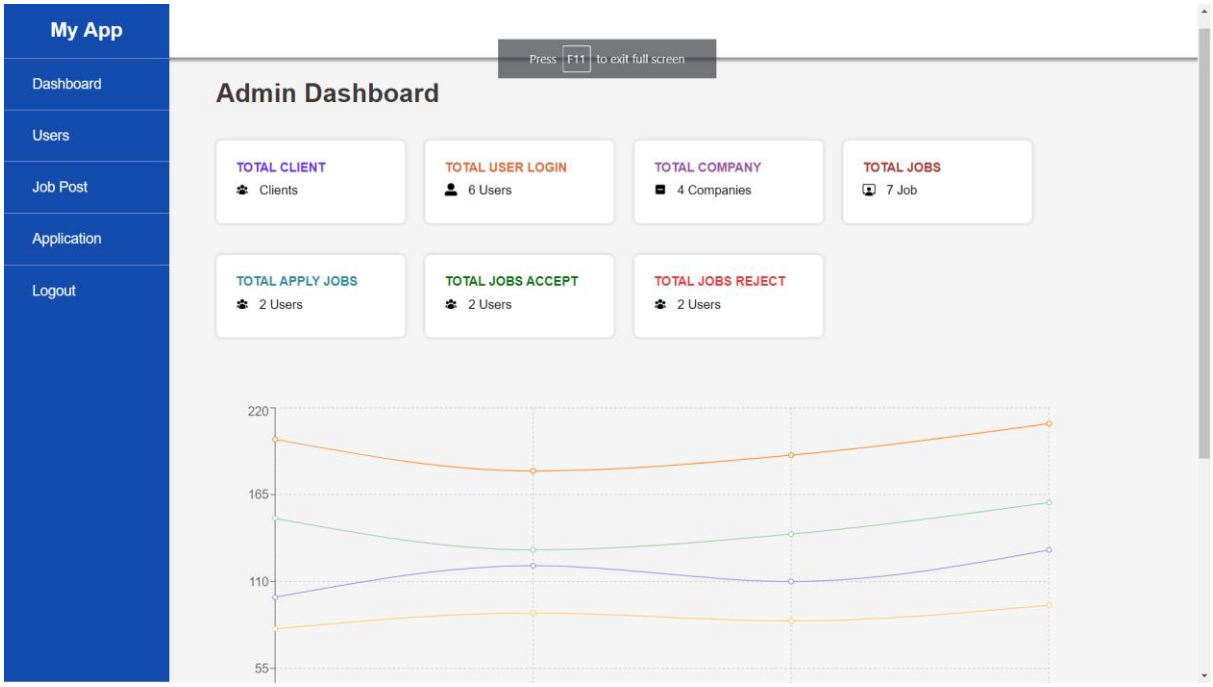


**Fig 5.17 Admin Dashboard Page**

## Js Code :

```
import React from 'react';
import './FAQPage.css'
function FAQPage()
{
  const faqItems = [
    {
      question: '1. How much water should I drink each day?',

   answer:
       'The recommended daily water intake varies based on factors like age, weight,
 activity level, and climate. ',
    },
    {
      question: '2. Why should I track my water consumption?',
      answer:
       'Tracking your water intake helps you stay hydrated, which is essential for overall
 health.',
    },
```

```
    {
      question: '3. Can I adjust my daily water intake goal?',

 answer:
       'Yes, you can customize your daily goal in the app settings. However, it is essential
to set a goal that aligns with your individual needs and lifestyle.',
    },

    {
       question: '3. Is it possible to drink too much water?',

  answer:
         'Yes, excessive water consumption can lead to a condition called hyponatremia,
where the body electrolyte balance is disrupted. It is essential to maintain a balance and
not overhydrate.',
      },
  ];

  return (
    <div className="faq-page">
      <h2>Frequently Asked Questions</h2>
      <ul className="faq-list">
        {faqItems.map((item, index) => (
          <li key={index} className="faq-item">
            <div className="faq-question">{item.question}</div>
            <div className="faq-answer">{item.answer}</div>
          </li>
        ))}


    </ul>
    </div>
  );
}

export default FAQPage;
```

## 5.1.1 ABOUT PAGE

Our About Us page offers insight into our organization's mission, values, and the team behind our platform, helping you get to know us better and understand our commitment to excellence. Discover our story, goals, and the passion driving our efforts to provide you with the best experience possible.

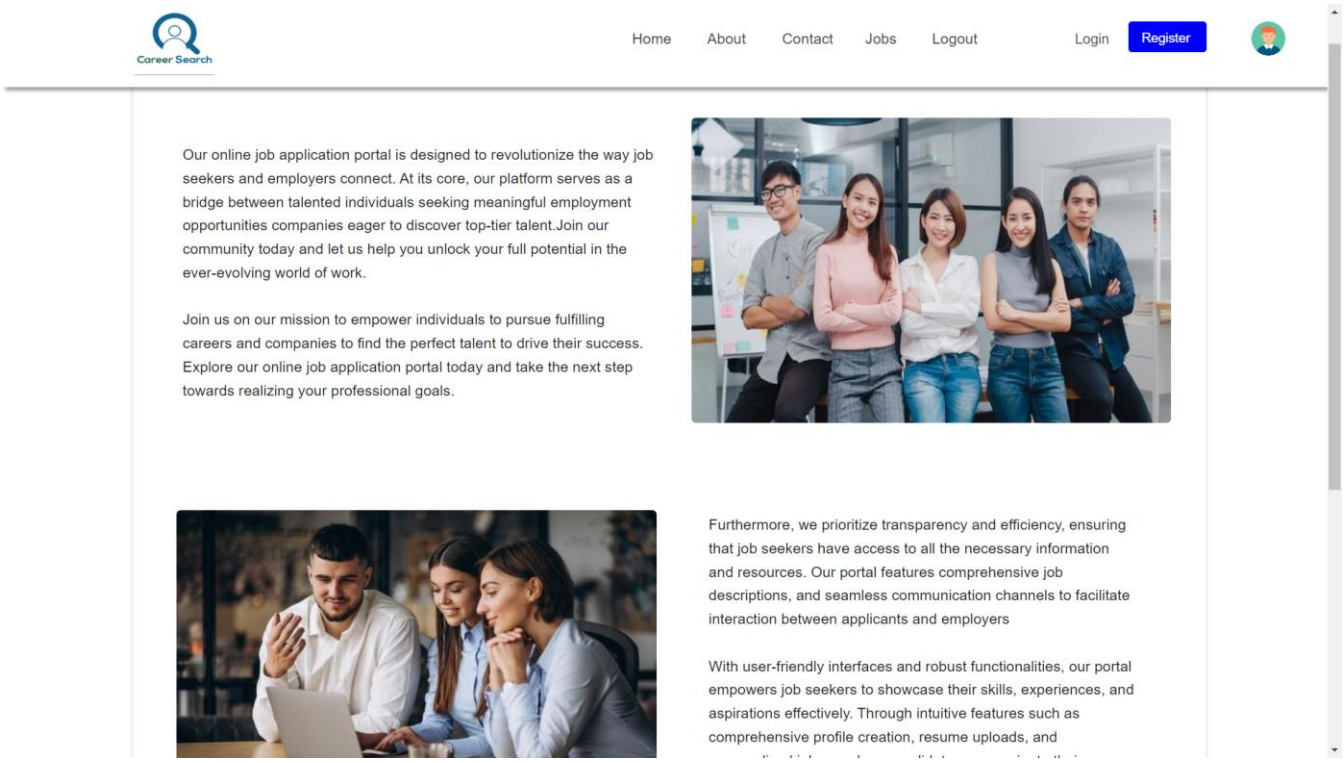


**Fig 5.18 About page**

## Js Code :

```
import pic1 from '../Fpage/pic 1.png';
import pic2 from '../Fpage/pic 2.png';
import pic3 from '../Fpage/pic3.png'
import './About.css'
export default function About() {
return (
```

```
<div>
<div className='Apic1'><img src={pic1} height={400}
        width={600}></img></div>
<div className='Apic2'><img src={pic2} height={400}
        width={600}></img></div>
<div className='Apic3'><img src={pic3} height={400}
        width={600}></img></div>

<div className='Aboxy1'><p>Users can review their historical water
        consumption data to track their progress over time. This can help them see
        whether they are meeting their hydration goals and making positive
        changes to their habits. By reviewing their past water intake, users can set
        realistic and personalized hydration goals. For instance, if they notice that
        they consistently fall short of their recommended daily water intake, they
        can set a goal to increase their consumption gradually.</p></div>

<div className='Aboxy2'><p>A water intake calculator takes into account
        various factors such as age, weight, activity level, and climate to provide
        personalized hydration goals. This helps individuals determine how much
        water they should drink daily based on their specific needs.The calculator
        helps users ensure they consume an adequate amount of water to meet
        their daily hydration requirements. </p></div>
<div className='Aboxy3'><p>Achieving daily water intake goals can be a source
        of personal accomplishment and motivation. Reminders keep individuals
        on track to reach their hydration objectives.</p></div>

</div>
    )
  }
```

### 5.1.2 USER DETAILS PAGE

This page outlines how we handle and protect your personal information, ensuring transparency in how your data is collected, used, and safeguarded. We are committed to maintaining your privacy and providing you with a secure and trustworthy experience on our platform.

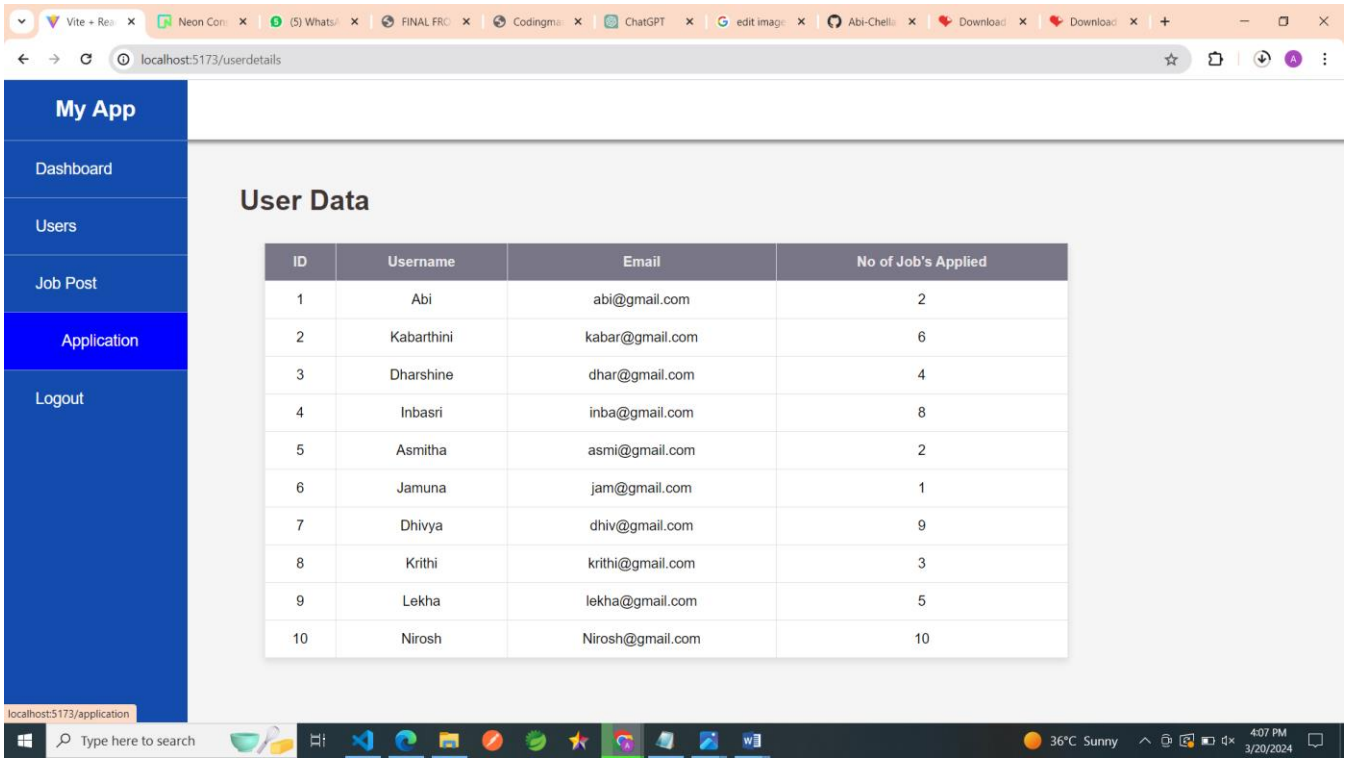


**Fig 5.19 user details page**

## Js Code:

```
import React from 'react'
import Sidebar from '../Bar/Sidebar';
import './Privacypolicy.css';
export default function Privacypolicy()
{
 return (
   <div>
     <Sidebar/>
     <div className='ppfull'>
          <div className='ppara'>
           <p>This Privacy Policy describes how our website collects, uses, and protects
personal information
           when you visit our website or use our services related to online wildlife
watching tour.</p>
          <div className='para'><br/>
           <p><span className='ppspan'>Personal Information: </span> We may collect
personal information, such as your name, email address, and location,
```

```
            </div>
           <div className='pphead1'>
             <h4>How We Use Your Information</h4>
             <div className='para'><br/>
           <p>We use the collected information for the following purposes:</p>
          </div>
           <div className='para'><br/>
            <p>1. To provide and improve our online wildlife watching tour services.</p>
            <p>2. To personalize your experience and offer relevant content.</P>
          </div>
          <div className='pphead1'>
          <p><b>Sharing Your Information</b></p>
          <p>We do not sell, trade, or rent your personal information to third parties. We may
  share your information with:</p>
          </div>
            <div className='para'><br/>
            <p>1.Partners and affiliates for marketing and promotional purposes.</p>
            <p>2.Third-party service providers that assist us in operating our website and
  delivering our services.</p>
            </div>
<div className='para'>
      <p>We take reasonable steps to protect your personal information from unauthorized
  access, disclosure, or alteration. However, no data transmission</p>
  </div>
  <div className='pphead1'>
          <p><b>Contact Us</b></p>
   </div>
  }
```

### 5.1.3 APPLICATION DETAILS PAGE

This page outlines the rules and guidelines governing the use of our platform, ensuring clarity and fairness in the interactions between users and our services. By accessing and using our platform, you agree to abide by these terms, promoting a safe and respectful online environment.

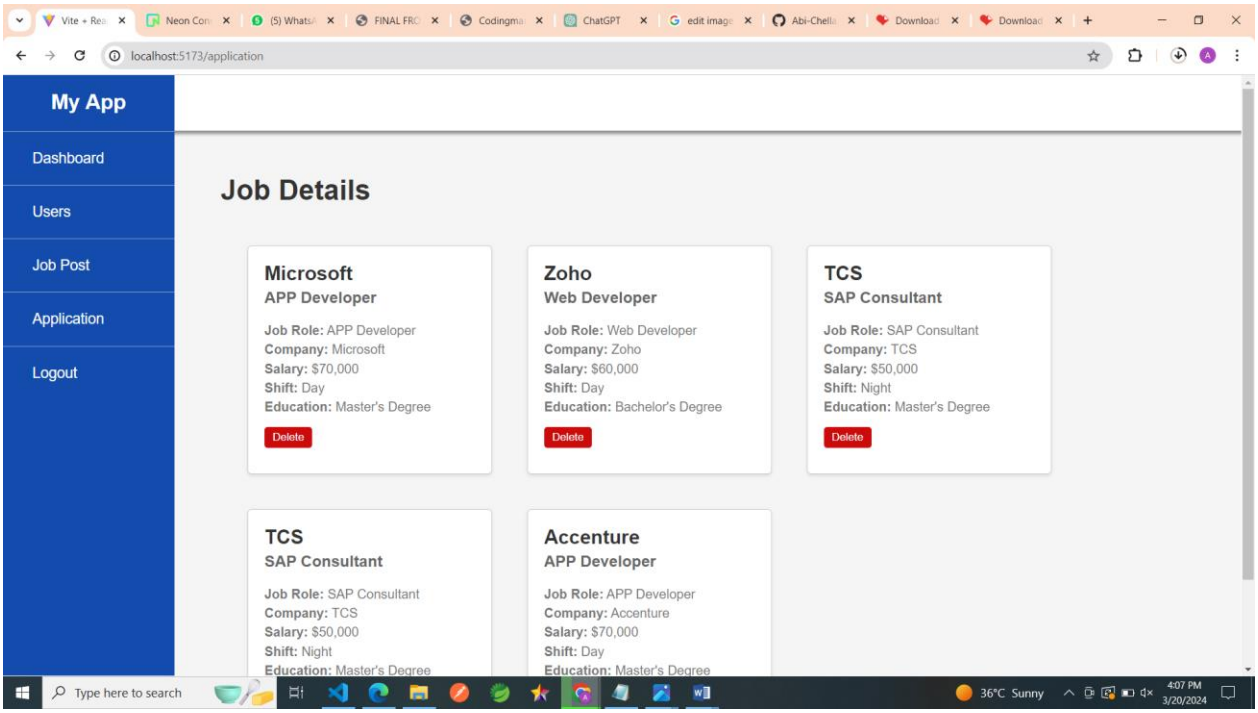


**Fig 5.20 Application Details page**

# Js Code:

```
import React from 'react'
import Sidebar from '../Bar/Sidebar';
export default function Termsandcondition()
{
 return (
   <div>
   <Sidebar/>
    <div className='tc-full'>
    <div className='wrapper flex_align_justify'>
    <div className='terms_condition'>
       <div className='tc_item tc_head flex_align_justify'>
          <div className='tc_icon'>
          <span className='tc_icon1'><BsIcons.BsTerminal/></span>
          </div>
    <div className='tc_text'>
```

```
          </div>
        </div>
        <div className='tc_item tc_body'>
          <ol>
            <li>
              <h3>1.Terms of use</h3>
              <p>By accessing and using this website, you agree to comply with and be
bound by these terms and conditions.</p>
            </li>
            <li>
              <h3>2.Services Offered</h3>
              <p>We provide online wildlife watching tour booking services. By booking a
tour through this website, you acknowledge that you have read and understood the
specific tour details and requirements.</p>
            </li><li>
              <h3>5.Changes to Terms and Conditions</h3>
              <p>We reserve the right to modify these terms and conditions at any time
without prior notice. Updated terms will be effective immediately upon posting on our
website.</p>
            </li> <li>
              <h3>6.Governing Laws</h3>
              <p>These terms and conditions are governed by and construed in accordance
exclusive jurisdiction of the courts </p></li>
        <div className='tc_accept'>
         <input type='radio' id='tc_accept'/>
         <label>I accept the <a href="#">terms and conditions</a></label>
          <div className='tc_item tc_foot flex_align'>
            <button className='tc_decline_btn'><Link to='/'>Decline</Link></button>
           <button className='tc_accept_btn'><Link to='/home'>Accept</Link></button>
        </div>
    </div>
    </div>
  )
```

### 5.1.4 FOOTER

The Footer page serves as a convenient navigational element at the bottom of our website, providing quick access to essential links, contact information, and important pages to enhance user accessibility and usability. It is designed to streamline your browsing experience and facilitate easy access to key resources.

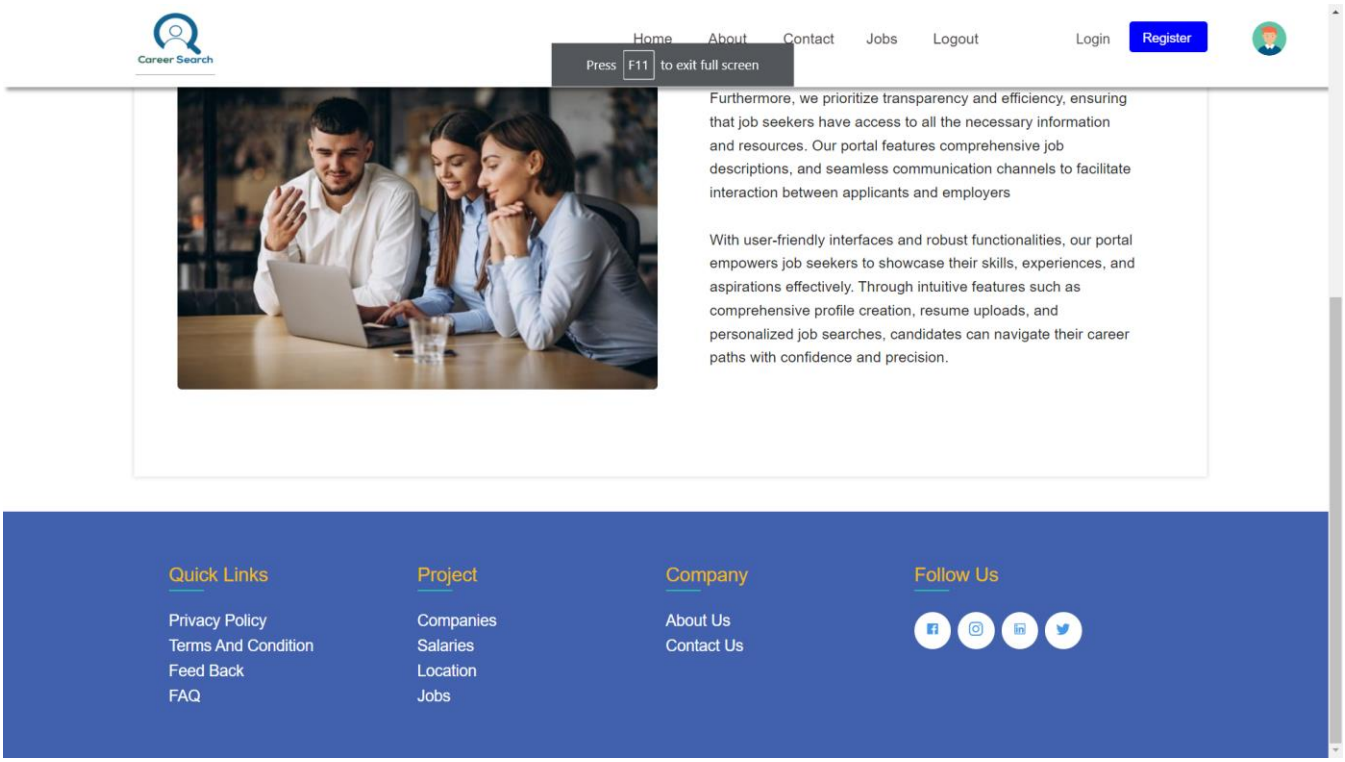


**Fig 5.21 Footer page**

## Js Code:

```
import React from 'react'
import { Link } from "react-router-dom";
export default function Footer() {
  return (
    <div className='footfull'>
    <footer>
```

```
      <div className='footer-col'>
        <h4>Quick Links</h4>
        <ul>
          <li><Link to='/pp'><span className='fp'>Privacy policy</span></Link></li>
          <li><Link to='/tc'><span className='fp'>Terms and
Condition</span></Link></li>
          <li><Link to='/feedback'><span className='fp'>Feed
Back</span></Link></li>
          <li><Link to='/faq'><span className='fp'>FAQ</span></Link></li></ul>
      </div>
      <div className='footer-col'>
        <h4>Project</h4>
        <ul>
          <li><Link to='/place'>place</Link></li>
          <li><Link to='/animel'>Animels</Link></li>
          <li><Link to='/video'>Videos</Link></li>
          <li><Link to='/image'>Images</Link> </ul></div>
      <div className='footer-col'>
        <h4>Company</h4><ul>
          <li><Link to='/about'>About Us</Link></li>
          <li><Link to='/contactus'>Contact Us</Link></li></ul></div>
      <div className='footer-col'>
        <h4>Follow Us</h4>
        <div className='sociellink'>
          <Link to='https://www.facebook.com/profile.php?id=100093687843994'><span
className='sicon'><AiIcons.AiFillFacebook/></span></Link>
          <Link to='https://www.instagram.com/'><span
className='sicon'><BsIcons.BsInstagram/></span></Link>
          <Link to='https://www.linkedin.com/in/abi-chellappa-193837256/'><span
className='sicon'><AiIcons.AiOutlineLinkedin/></span></Link>
          <Link to='https://twitter.com/'><span
className='sicon'><BsIcons.BsTwitter/></span></Link>
        </div></footer>
    </div>
  )}
```

# CHAPTER 6

# SYSTEM SPECIFICATION



In this chapter, we see the software that we have used to build the website. This chapter gives you a small description about the software used in the project.

## 6.1 INSTALLATION OF SPRING WITH VISUAL STUDIO

To setup a Spring backend in Visual Studio Code, follow these steps:

1. Install Visual Studio Code and Java support with the Coding Pack for Java or Java Extension Pack.

2. Install the Spring Boot Extension Pack for VS Code and Spring Boot Dashboard for VS Code extensions.

3. Clone or copy a Spring Getting Started guide, such as REST Service guide, from the GitHub repository.

4. Open the cloned or copied project in Visual Studio Code.

5. Use the Spring Boot Dashboard to manage project, navigate between files and start or stop the application.

6. Implement Spring Boot components such as controllers, services and repositories.

7. Create unit or Integration tests to validate your code and implement test-driven development.

8. Configure your application by setting up a data source, enabling security features or deploying to a server.


**FIGURE 6.1 SPRING WITH VISUAL STUDIO**

## 6.2 JAVA

Java is a class based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general -purpose programming language intended to let programmers write once, run anywhere meaning that compiled. Java code can run on all platforms that support Java without the need to recompile.

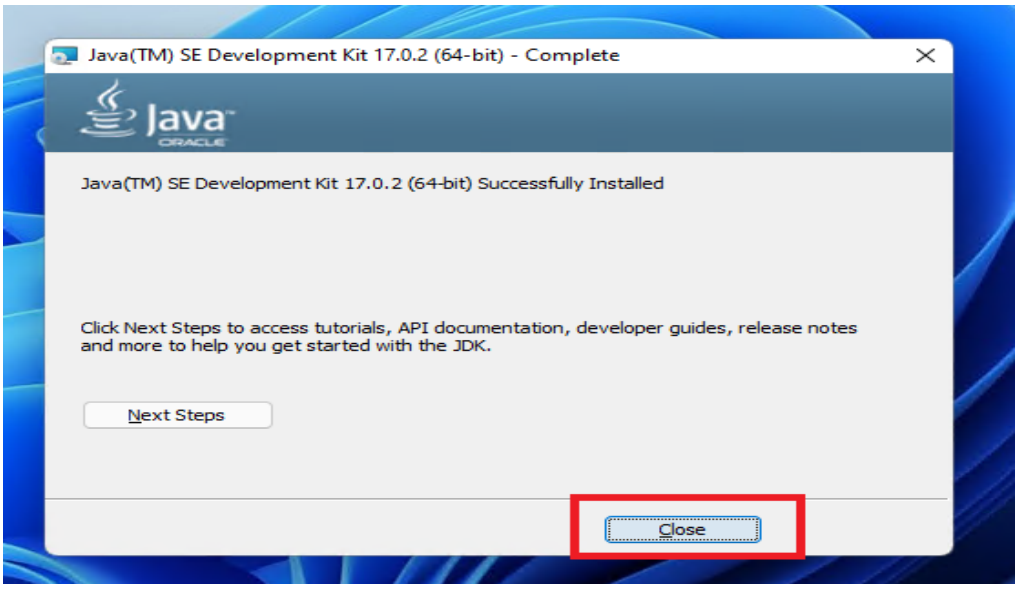


**FIGURE 2.2 JAVA JDK**

## 6.3 POSTGRESQL



PostgreSQL is a powerful, open-source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads.

1.Visit the official PostgreSQL website: PostgreSQL Downloads.

2.Choose the PostgreSQL Installer appropriate for your operating system (Windows, macOS, Linux).

3.Select the version that suits your needs (typically, choose the latest stable version).

4.Configure the PostgreSQL Server by setting the superuser password and selecting other server settings.

5.During installation, choose additional PostgreSQL components like pgAdmin, documentation, drivers etc. Allow the installer to finish the installation process, which might take a few minutes.

6.Once the installation is complete, you'll typically receive a confirmation message.

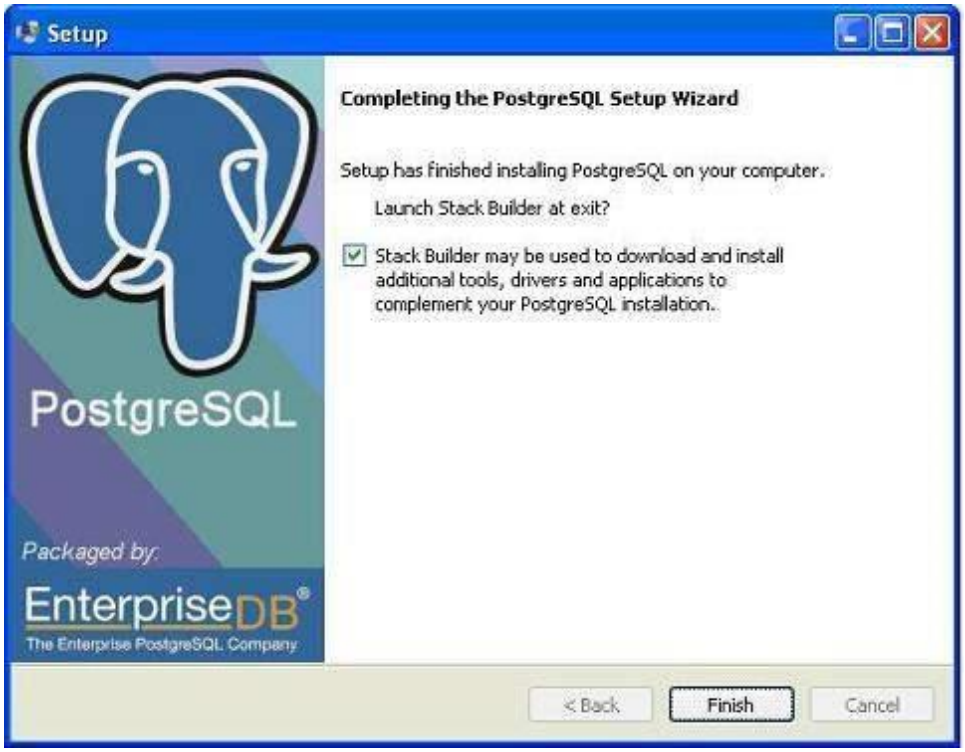


**FIGURE 6.3 POSTGRESQL INSTALLER**

# CHAPTER 7

# METHODOLOGIES

Methodologies and implementation play a crucial role in the project's success. The chosen methodologies provide a systematic framework for project management and development, ensuring adherence to best practices. Implementation involves executing the development tasks, integrating components, and deploying the application. This phase includes backend development, database integration, API creation, testing, deployment, and maintenance. By following the established methodologies and effectively implementing the project plan, the development team can ensure a structured and organized approach, leading to efficient collaboration, timely delivery, and the achievement of project objectives.

## 7.1 PROJECT APPROACH :

The Agile methodology offers an ideal approach for your project, combining flexibility and efficient progress. With a focus on iterative development and regular feedback, Agile enables you to tailor the methodology to your specific needs. By identifying user stories and creating a backlog, you can prioritize features and functionalities that align with your project's objectives. Breaking down the backlog into manageable tasks within sprints allows you to maintain a clear roadmap and work in a structured manner. As you proceed with development, you have the freedom to manage your time and resources effectively. Regular testing during and after each sprint ensures the quality and functionality of your application. Seeking feedback from users or stakeholders at the end of each sprint provides valuable insights for adaptation and refinement. This iterative process allows you to continually improve and adapt your project based on real-world input. Embracing the practice of continuous integration and automating deployment streamlines the delivery of new features and bug fixes. This ensures a seamless and efficient development workflow, enabling you to focus on enhancing your application.

By applying Agile methodologies tailored to your individual development process, you can maintain a structured and adaptable approach. This empowers you to incorporate user feedback, stay focused on your project's objectives, and deliver a high-quality application. The flexibility of Agile allows you to strike a balance between efficient progress and accommodating changes, ultimately leading to the successful completion of your project as a single developer.



## 7.2 TOOLS TO USE:

**1) Backend Tools:**

a) Code Editor:

i) VS Code with Spring Extension

b) Version Control System:

i) Git

ii) GitHub

c) Package Manager:

i) Maven

d) Backend Framework:

i) Java Spring Boot

ii) Spring Data JPA

**2) Database Management System:**

a) PostgreSQL

**3) API Development and Testing Tools:**

a) Swagger-UI

# CHAPTER 8



# IMPLEMETATION AND FUNCTIONALITY

The Job Searching portal backend is the linchpin of effective job management. It provides administrators with a comprehensive set of tools to seamlessly add, update, and remove jobs. The user-friendly interface simplifies their tasks, reducing the learning curve. Robust data validation and error-handling mechanisms are in place to safeguard the accuracy and integrity of stored information. Security is a top priority, with stringent access controls and user authentication measures protecting sensitive data. The architecture is designed with scalability in mind, capable of efficiently handling increased data loads and user traffic as the application expands.

## 8.1 API Request

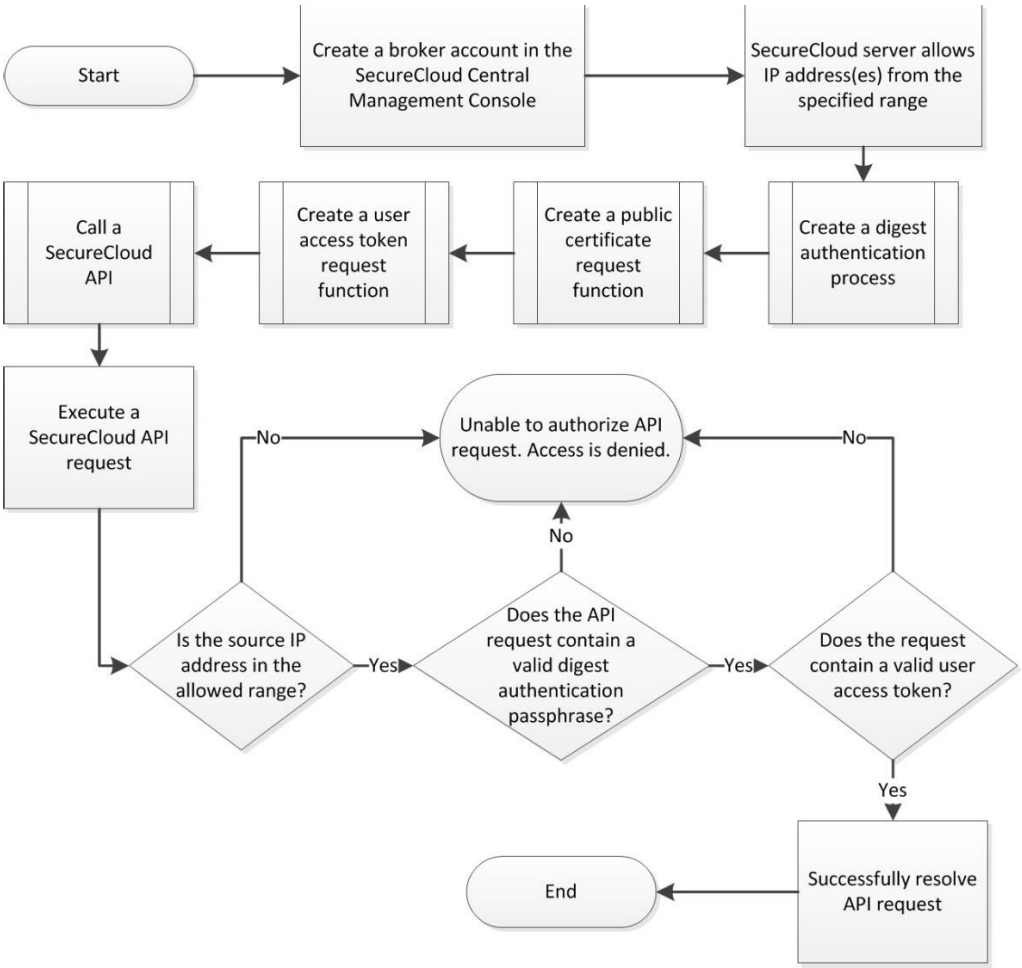


**Fig 8.1. REST API flowchart**

A Representational State Transfer (REST) API plays a pivotal role in the architecture of the Job Searching Portal, providing a structured and efficient means of communication between the frontend and backend components. In this context, the REST API serves as the intermediary that enables the exchange of data and requests, making it a cornerstone of the application's functionality.

The REST API of the Job Searching Portal adheres to RESTful principles, which are centered around a set of stateless operations for creating, retrieving, updating, and deleting data. It defines a clear structure for the endpoints, with each endpoint corresponding to a specific resource or action. For instance, endpoints might include "GET" to retrieve a list of available jobs, "POST /job " to add jobs to a user's job list, and "DELETE /job /{jobID}" to remove jobs from the list.

By adopting RESTful design, the API simplifies interactions with the application, ensuring that users can easily access job information, manage their shopping carts, and complete transactions. It provides data in a format that is widely understood, typically in JSON, allowing for seamless integration with various client applications, including web and mobile interfaces.

In addition to its role in enabling user interactions, the REST API is a fundamental component for potential future developments. It opens the door to third-party integrations, such as payment gateways, external inventory systems, or analytics services, that can enhance the application's functionality and expand its capabilities. Furthermore, the API empowers the application to be scalable, ensuring that it can accommodate growing user bases and evolving feature sets.

The REST API in the Job Searching Portal is not merely a technical component; it's the conduit through which the application's core functionalities are exposed and extended, ultimately contributing to a seamless and versatile user experience.

## 8.1 CRUD OPERATION



In the Job Searching portal, the implementation of CRUD (Create, Read, Update, Delete) operations is fundamental to the efficient management of the job inventory. The "Create" operation allows administrators to add new jobs by entering comprehensive details, which are then validated for accuracy and completeness before being securely stored in the database. "

Read" operations enable job seekers to seamlessly browse and explore the joblist, with search and filtering options enhancing their ability to find specific jobs and view detailed information.

The "Update" operation empowers administrators to modify job information, and a user-friendly interface ensures that this process is intuitive. Stringent data validation criteria are maintained to guarantee the accuracy and reliability of the edited information, which is subsequently updated in the database.

The "Delete" operation provides administrators with the means to remove jobs from the inventory, incorporating a confirmation prompt to prevent accidental deletion. Once confirmed, the job’s information is securely deleted from the database.

The successful implementation of these CRUD operations is essential for ensuring that job management within Job Searching portal is both efficient and user-friendly. These operations, when complemented by robust data validation, security measures, and intuitive interfaces, collectively contribute to a seamless user experience, making it easier for administrators to maintain the jobs and for recuiters to explore and interact with the job seekers.

**Coding:**

**Job Controller Class:**

```
package com.example.demo.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.example.demo.entity.Job;
import com.example.demo.service.JobService;

@RestController
@CrossOrigin
@RequestMapping("/auth")
public class JobController {
	@Autowired
	private JobService jjs;
	 @GetMapping("/getjob")
	 public List<Job> getJobDetails()
	 {
		return jjs.getJobDetails();
	 }
	 @PostMapping("/postjob")
	 public void saveData(@RequestBody Job je)
```

```
        {
                jjs.saveJob(je);
        }
        @PutMapping("/putjob{id}")
        public void updateData(@RequestBody Job je,@PathVariable Long id)
        {
                je.setId(id);
                jjs.updateJob(je);
        }
        @DeleteMapping("/deletejob/{id}")
        public void deleteData(@PathVariable Long id)
        {
                jjs.deleteJob(id);
        }
}
```

**Job Entity Class:**

```
package com.example.demo.entity;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Entity
@Getter
@Setter
@NoArgsConstructor
```

```
@AllArgsConstructor
public class Job {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   int jobId;
   String title;
   String company;
    String dept;
     String location;
     String responsiblity;
     String qualification;
}
```

**Job Service Class:**

```
package com.example.demo.service;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class JobService {
        @Autowired
   private JobRepository jr;
   public List<Job> getJobDetails(){
                return jr.findAll();
        }
        public void saveJob(Job je)
        {
                jr.save(je);
        }
```

```
	public void updateJob(Job je)
	{
	jr.save(je);
	}

	public void deleteJob(Long id)
	{
		jr.deleteById(id);
	}
}
```

**Job Repository Interface:**

```
package com.example.demo.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.example.demo.entity.Job;
@Repository
public interface JobRepository extends JpaRepository<Job, Long>
{
}
```

## 8.2 DATA RETRIEVEL PROCESS



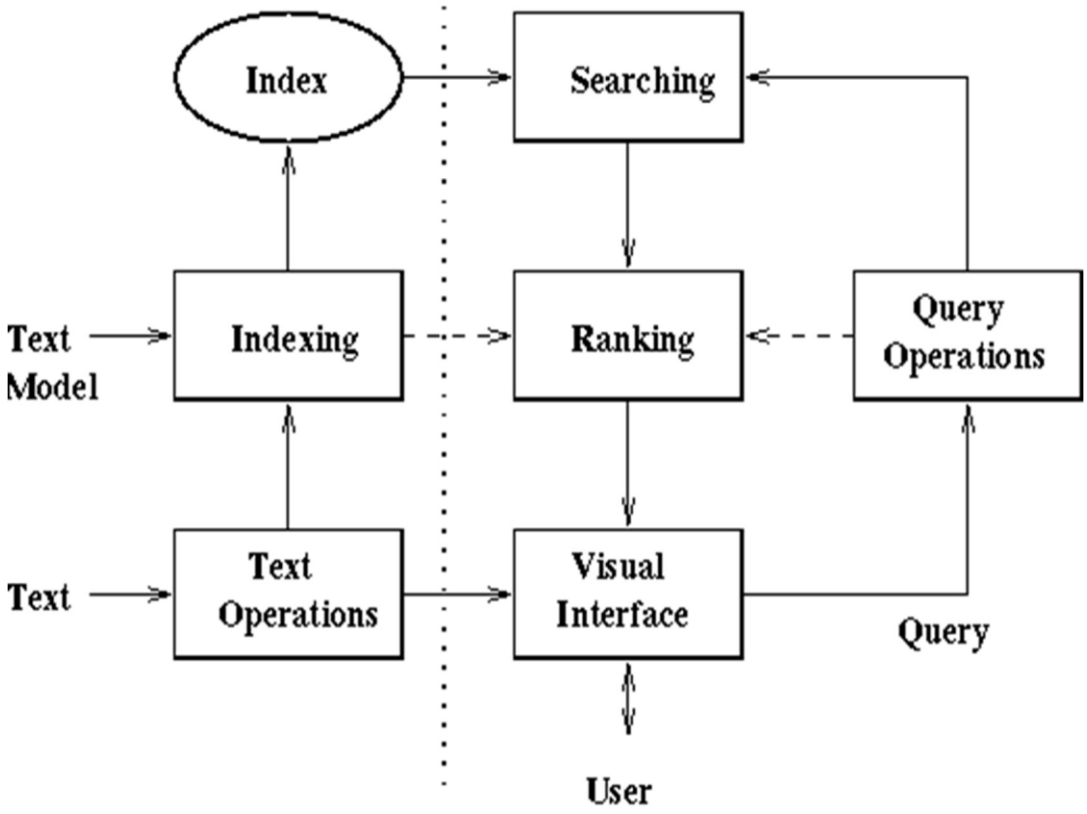


**Fig 8.2 Data Retrieval Process**

The data retrieval process in the backend of the Job Searching Portal is a pivotal component that facilitates the efficient and secure retrieval of data from the database and its transmission to the frontend or client applications. This process begins with a client request made to a specific API endpoint on the backend. Upon receiving the request, the backend's routing system directs it to the appropriate endpoint handler basedon the URL and HTTP method.

Before proceeding, the backend verifies user authentication and confirms if the necessary permissions exist to access the requested data. Following authorization, a database query is generated based on the request, specifying the criteria for data retrieval.

## 8.3 DATA UPDATE PROCESS



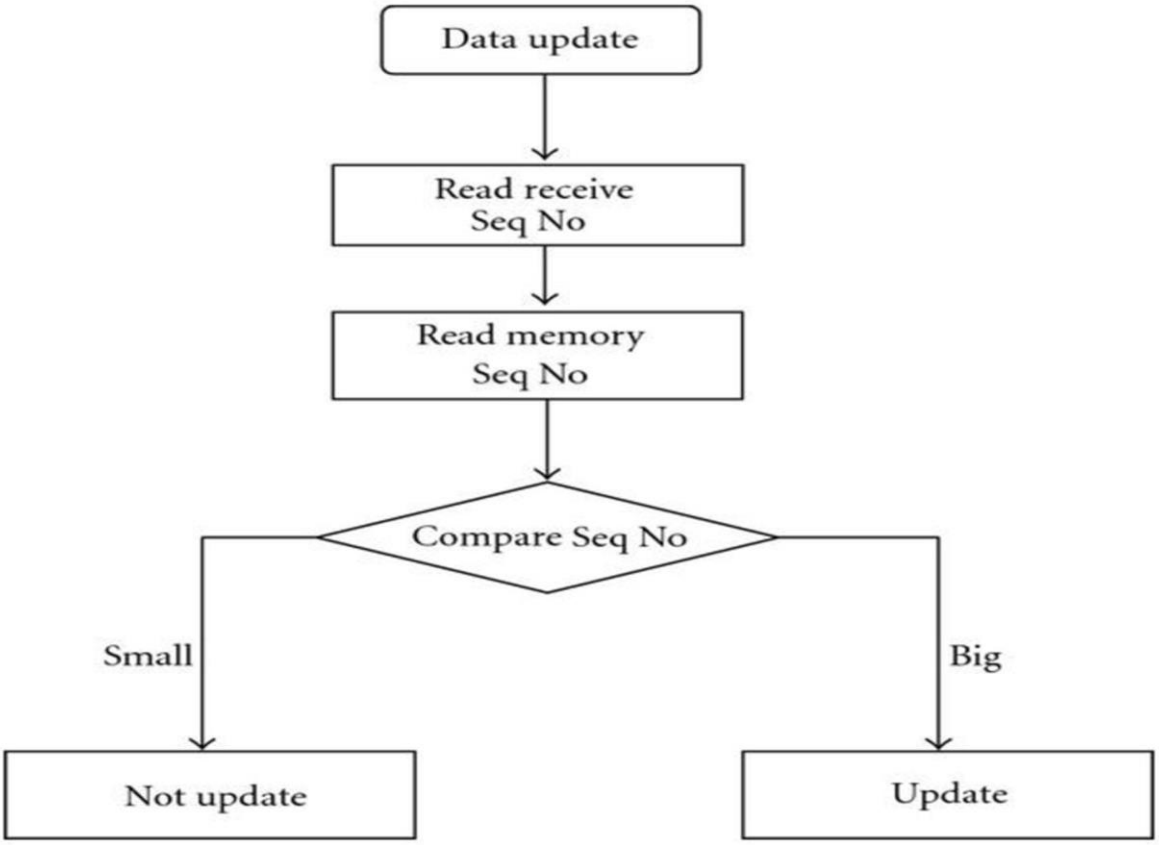


**Fig 8.3 Data Update Flowchart**

The data update process within the backend of the Job Searching Portal is a crucial mechanism that allows users and administrators to enact changes in the application's database.

Authentication and authorization are fundamental checkpoints in this process. The backend confirms the user's identity and verifies if the user possesses the necessary permissions to execute the data update operation. Unauthorized requests are diligently restricted.

Data validation is a subsequent step, whereby the backend scrutinizes the provided data to ensure that it adheres to the correct format and complies with established business rules and constraints. This phase plays a pivotal role in maintaining data integrity.

## 8.4 SECURITY AND AUTHENTICATION



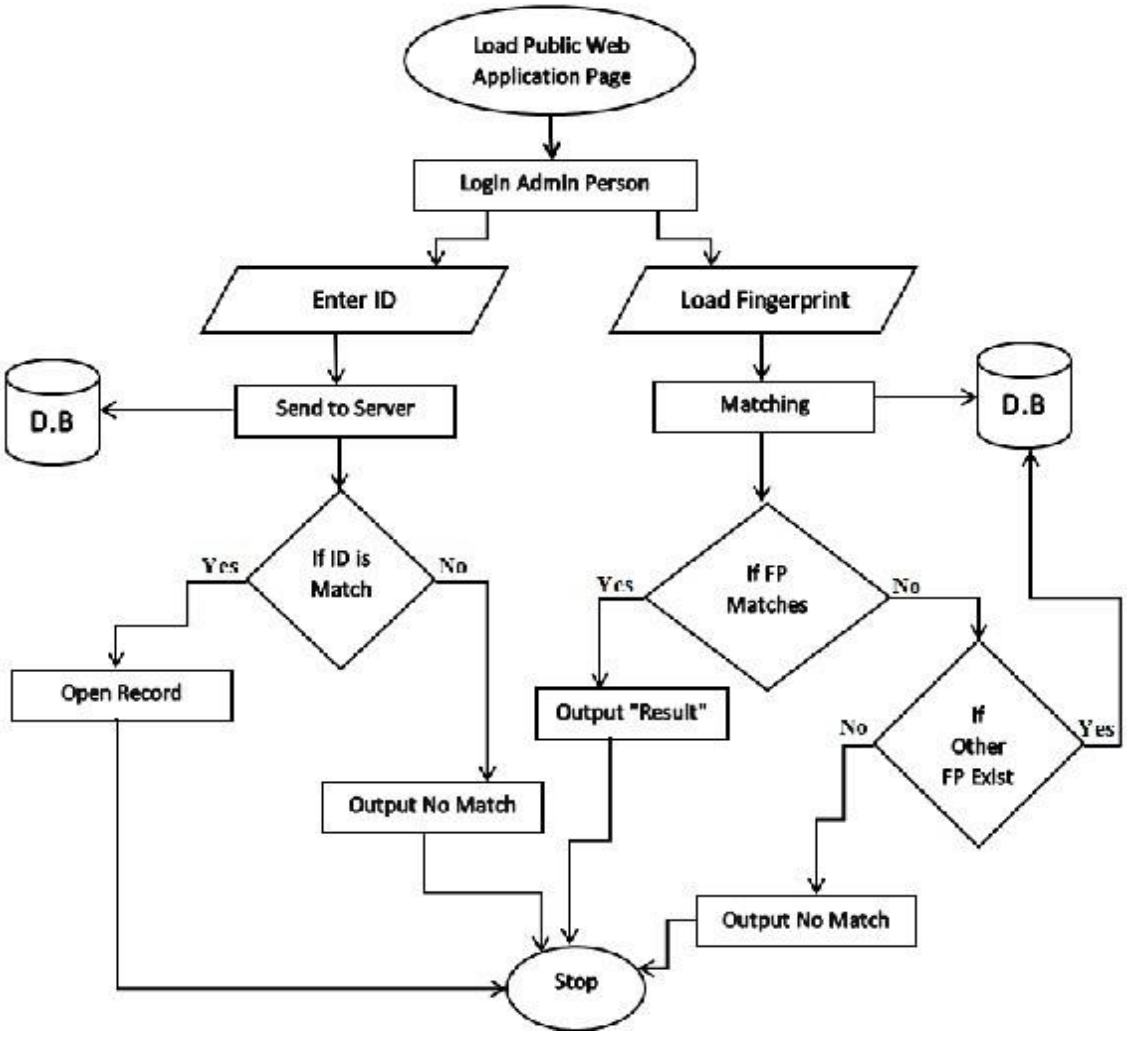


**Fig 8.4 Security And Authentication Flowchart**

Security and authentication lie at the heart of the Job Searching portal's robust infrastructure. To safeguard sensitive data and uphold the integrity of the system, a comprehensive set of security measures and authentication protocols have been implemented.

User authentication is a fundamental pillar, allowing users, including administrators and users , to register securely, leveraging email and strong, hashed passwords. A robust login system verifies user credentials and controls access to the application. Role-based access ensures that each user is assigned specific permissions based on their role, providing administrators with greater control over job management functions while limiting users primarily to read-only access.

Data encryption, both in transit and at rest, is a core component of the security framework. Secure communication channels protect data during interactions, while encryption of sensitive data in the database safeguards information in the event of a breach. Session management maintains secure user sessions, preventing unauthorized access or data exposure.

Utilizing security libraries and frameworks, like Spring Security, enhances the efficiency of authentication and access control. Additionally, protection against common security threats, such as cross-site scripting and SQL injection, is in place.

User awareness and education contribute to the overall security posture, ensuring that users are informed and capable of recognizing potential threats. Regular security audits and vulnerability assessments are conducted to proactively identify and address potential weaknesses, keeping the application resilient against emerging security risks. In sum, these measures collectively create a secure and trustworthy environment within the Job Searching Portal , protecting user data and maintaining the application's integrity.

**Coding:**

**Jwt Authentication Filter Class:**

```
package com.example.demo.config;
import java.io.IOException;
import org.springframework.lang.NonNull;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.stereotype.Component;
```

```
import org.springframework.web.filter.OncePerRequestFilter;
import com.example.demo.service.JwtService;
import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import lombok.RequiredArgsConstructor;

@Component
@RequiredArgsConstructor
public class JwtAuthenticationFilter extends OncePerRequestFilter{
        private final JwtService jwtservice ;
        private final UserDetailsService userDetailsService;
        @Override
        protected void doFilterInternal(
                        @NonNull HttpServletRequest request,
                        @NonNull HttpServletResponse response,
                        @NonNull FilterChain filterChain)
                       throws ServletException, IOException {
                if (request.getServletPath().contains("/api/v1/auth")) {
                    filterChain.doFilter(request, response);
                    return;}
                final String authHeader = request.getHeader("Authorization");
            final String token;
            final String username;
            if (authHeader == null || !authHeader.startsWith("Bearer ")) {
                filterChain.doFilter(request, response);
                return;
            }
              token = authHeader.substring(7);
              username = jwtservice.extractUsername(token);
```

```
                if (username != null &&
SecurityContextHolder.getContext().getAuthentication() == null) {
                UserDetails userDetails =
this.userDetailsService.loadUserByUsername(username);
                if (jwtservice.isTokenValid(token, userDetails)) {
                   UsernamePasswordAuthenticationToken
usernamePasswordAuthenticationToken = new
UsernamePasswordAuthenticationToken(
                         userDetails, null, userDetails.getAuthorities());
                   usernamePasswordAuthenticationToken
                         .setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticati
onToken);
                }
                filterChain.doFilter(request, response);
                }
        }
}
```

**Jwt Service Class:**

```
package com.example.demo.service;

import java.security.Key;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.function.Function;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Service;
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
```

```
import io.jsonwebtoken.io.Decoders;
import io.jsonwebtoken.security.Keys;

@Service
public class JwtService {

          @Value("${application.jwt.secret-key}")
           private String secretKey;

        public String extractUsername(String token) {
                // TODO Auto-generated method stub
                 return extractClaim(token, Claims::getSubject);
        }

          public <T> T extractClaim(String token, Function<Claims, T>
claimsResolver) {
              final Claims claims = extractAllClaims(token);
              return claimsResolver.apply(claims);
           }

        private Claims extractAllClaims(String token) {
                return Jwts
                                .parserBuilder()
                                .setSigningKey(getSigningKey())
                                .build()
                                .parseClaimsJws(token)
                                .getBody();
        }

        private Key getSigningKey() {
      byte[] keyBytes = Decoders.BASE64.decode(secretKey);
      return Keys.hmacShaKeyFor(keyBytes);
   }
          public String generateToken(UserDetails userDetails) {
              return createToken(new HashMap<>(), userDetails);
           }

           public String createToken(Map<String, Object> extraClaims, UserDetails
userDetails) {
              return Jwts
                    .builder()
                    .setClaims(extraClaims)
                    .setSubject(userDetails.getUsername())
                    .setIssuedAt(new Date(System.currentTimeMillis()))
                    .setExpiration(new Date(System.currentTimeMillis() + 24 * 60 * 60 *
1000))
                    .signWith(getSigningKey(), SignatureAlgorithm.HS256)
                    .compact();
```

```
    }
    public boolean isTokenValid(String token, UserDetails userDetails) {
        final String email = extractUsername(token);
        return (email.equals(userDetails.getUsername())) &&
!isTokenExpired(token);
    }

    private boolean isTokenExpired(String token) {
        return extractExpiration(token).before(new Date());
    }

    private Date extractExpiration(String token) {
        return extractClaim(token, Claims::getExpiration);
    }
}
```

# CONCLUSION

# CHAPTER 9

# CONCLUSION

In conclusion, the online job application portal represents a pivotal advancement in modern recruitment practices, offering a multifaceted solution to the challenges faced by both job seekers and employers. By providing a centralized platform for job listings, application management, and communication, these portals streamline the recruitment process, enhance efficiency, and foster transparency. Additionally, they contribute to reducing recruitment costs, improving time-to-fill metrics, and enhancing the candidate experience, ultimately strengthening employer branding efforts. With their customizable features, data-driven insights, and seamless user experience, online job application portals empower organizations to attract top talent, optimize recruitment strategies, and gain a competitive edge in the talent market. As technology continues to evolve, these portals will play an increasingly integral role in shaping the future of recruitment, driving success for individuals and organizations alike.

## 9.1 FUTURE SCOPE

The future scope for online job application portals is characterized by continuous innovation, driven by technological advancements and evolving trends in the global workforce. As these portals evolve to meet the changing needs of job seekers and employers, they will continue to serve as indispensable tools for talent acquisition, driving efficiency, accessibility, and fairness in the recruitment process..

# REFERENCES

# CHAPTER 10

# REFERENCES

[1] All Basics www.javatpoint.com

[2] The basic diagram of the HTML/HTML5 5/24/201844 http://www.modelrumahminimalis.co/html-menutemplatesfreedownload/69033.html/top-result-html-menu-templates-freedownloadinspirationalgenerous-menu-template-html-images-entry-levelresumetemplates-picture-2017-hdj5

[3] www.w3schools.com

[4] Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures.

[5] React Documentation: https://reactjs.org/docs/getting-started.html

[6] React Tutorial (MDN Web Docs): https://developer.mozilla.org/enUS/Learn/Tools_and_testing/client