OPPORTUNITY HUB – JOB SEEK PORTAL

TEAM MEMBERS:

ROLL NUMBER	NAME
RASHMIKA K R	21ALR076
RITHIK CHANDRASEKAR	21ALR081
PADMAPRIYA R	21ALR063

INTRODUCTION:

Opportunity Hub is a premier web development project designed to revolutionize the job market by serving as a dynamic and interactive platform for job seekers and employers. This innovative portal aims to bridge the gap between talent and opportunity, creating a seamless experience for both parties.

For Job Seekers:

Opportunity Hub offers a comprehensive and user-friendly interface where job seekers can showcase their skills, qualifications, and field specialities. By creating detailed profiles, job seekers can highlight their expertise, making it easier for potential employers to find and evaluate their capabilities. The platform provides robust search functionalities, allowing job seekers to explore various job postings that match their skill sets and career aspirations.

For Employers:

Employers can leverage Opportunity Hub to post job vacancies and search for potential candidates that fit their organizational needs. The portal's sophisticated filtering and matching algorithms ensure that employers can efficiently identify and connect with the most suitable candidates. By accessing a diverse pool of talent, employers can streamline their recruitment process and enhance their workforce with highly qualified professionals.

Key Features:

- **Profile Management:** Detailed profiles for both job seekers and employers, ensuring clarity and visibility.
- **Advanced Search:** Powerful search and filtering options to match job seekers with relevant job postings and vice versa.
- **Direct Communication:** Integrated messaging system for seamless interaction between job seekers and employers.
- **Job Alerts:** Personalized notifications for job seekers about new opportunities that match their profiles.
- **Analytics:** Insightful data and analytics to help employers make informed hiring decisions.

Opportunity Hub is more than just a job portal; it is a comprehensive ecosystem designed to foster meaningful connections between job seekers and employers. By facilitating efficient and effective recruitment processes, Opportunity Hub stands as a testament to modern web development's potential to transform the employment landscape.

Join Opportunity Hub today and be part of a platform where opportunities meet talent.

REQUIREMENTS:

Front-End Programming

1. **React:** React is the cornerstone of our front-end development for Opportunity Hub. This popular JavaScript library allows us to build a dynamic and responsive user interface with a component-based architecture. React enables the creation of reusable UI components that efficiently update in response to data changes. By leveraging the virtual DOM, React ensures optimal performance, even in complex web applications, enhancing the user experience for both job seekers and employers.

2. **CSS (Cascading Style Sheets):** CSS is integral to defining the visual appearance and layout of Opportunity Hub. It allows our developers to apply consistent styles, such as colours, fonts, spacing, and positioning, across all HTML elements. This results in a visually appealing and uniform interface, ensuring a pleasant and engaging experience for users as they navigate the portal.

3. **JavaScript:** JavaScript brings interactivity and dynamic behaviour to the front end of Opportunity Hub. It enables developers to manipulate HTML and CSS, handle user interactions, validate forms, and perform calculations. JavaScript is also crucial for making asynchronous requests to the server, facilitating smooth and responsive user experiences.

Back-End Programming

1. **Node.js:** Node.js forms the backbone of our back-end development. This JavaScript runtime environment allows us to execute server-side code and build scalable network applications. Node.js is known for its event-driven architecture and non-blocking I/O operations, making it ideal for handling the real-time interactions and data processing required by Opportunity Hub.

2. **Express:** Express is a lightweight and flexible Node.js web application framework that we use to build our server-side logic. It simplifies the creation of robust APIs and web servers, providing a streamlined approach to managing routes, middleware, and request handling. Express plays a crucial role in facilitating communication between the front-end and back-end components of our application.

3. **MongoDB:** MongoDB is our choice for database management, fitting seamlessly into the MERN stack. As a NoSQL database, MongoDB offers flexibility in storing and querying data in a JSON-like format. This allows for efficient handling of large volumes of unstructured data, ensuring quick retrieval and storage of job postings, user profiles, and other critical information. MongoDB's scalability and performance make it an excellent choice for supporting the dynamic data needs of Opportunity Hub.

Connecting Front-End and Back-End

APIs (Application Programming Interfaces): Our backend developers build robust APIs using Express to facilitate communication between the front-end React components and the back-end Node.js server. These APIs define the protocols and rules for data exchange, ensuring secure and efficient interactions. We utilize RESTful API standards to structure our endpoints, enabling seamless data retrieval, submission, and updates. This integration ensures that job seekers and employers experience a smooth and responsive interface, with real-time updates and interactions.

Additional Features

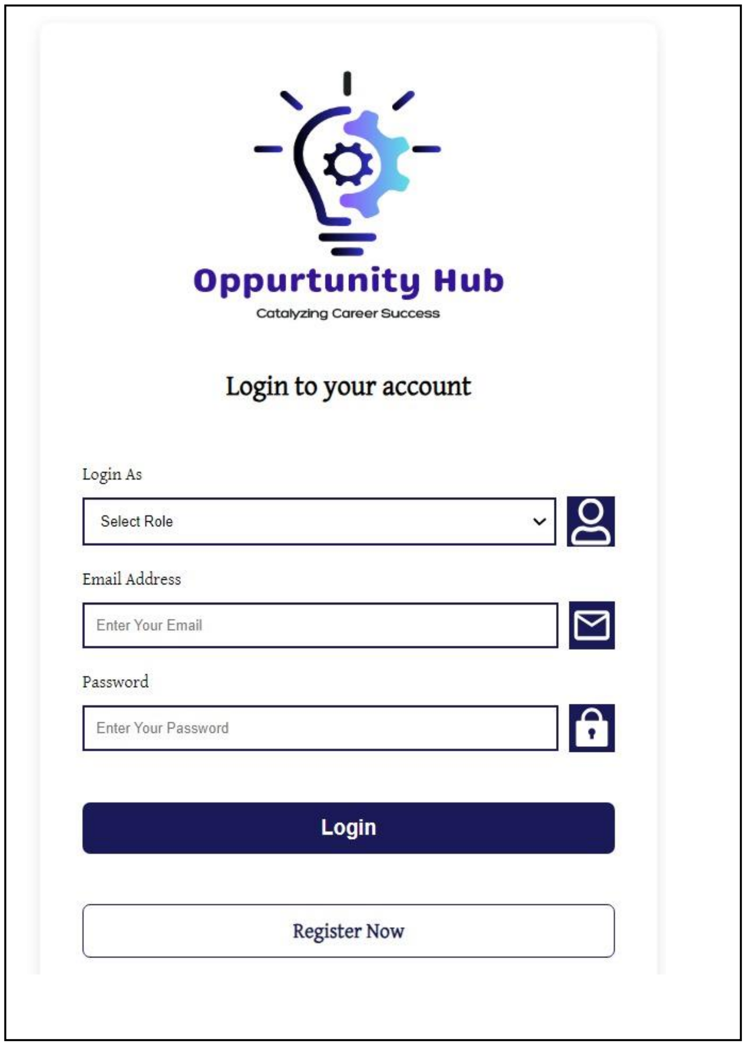
1. **Authentication and Security:** Opportunity Hub implements robust authentication mechanisms to ensure the security and privacy of user data. We utilize JWT (JSON Web Tokens) for secure user authentication and authorization, ensuring that only authorized users can access specific resources. Implementing HTTPS and other security best practices, such as input validation and sanitation, helps protect against common web vulnerabilities like SQL injection and cross-site scripting (XSS).

2. **Search and Filtering:** Advanced search and filtering capabilities are integrated into Opportunity Hub, allowing users to quickly find relevant job postings or candidates. By utilizing MongoDB's powerful querying and indexing features, we ensure fast and accurate search results, enhancing the overall user experience.

By leveraging the MERN stack – MongoDB, Express, React, and Node.js – Opportunity Hub delivers a powerful and cohesive web application. This technology stack enables us to create a scalable, high-performance platform that effectively connects job seekers with employers, fostering opportunities and facilitating meaningful professional connections.

UI DESIGN:

Register/Login page



Login to your account

Login As

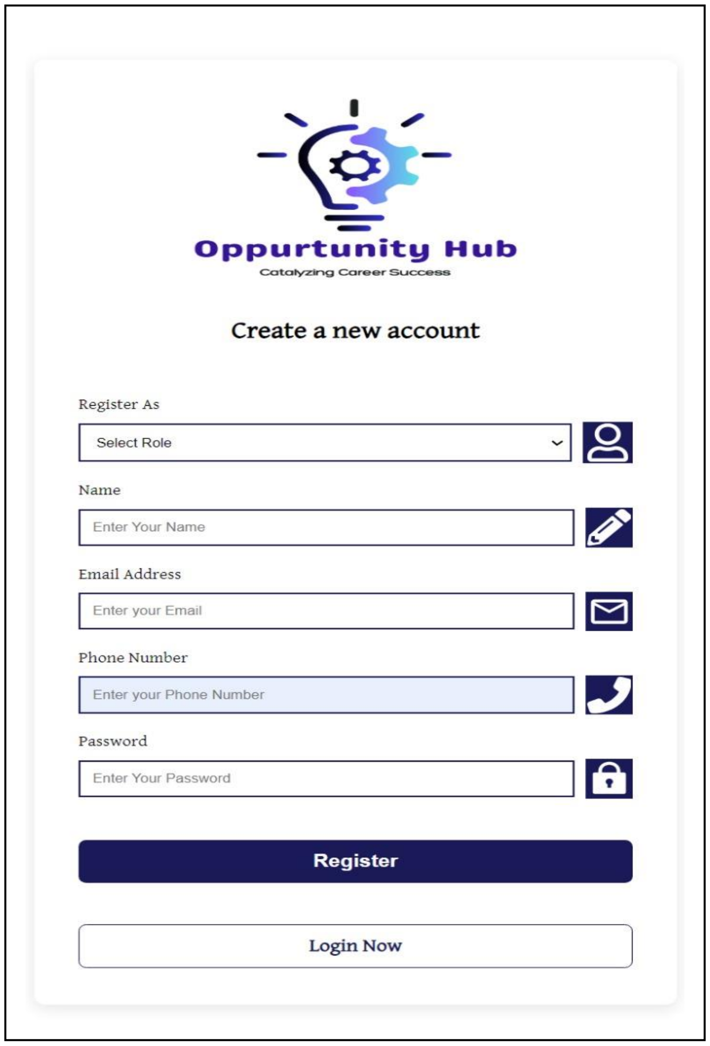
Select Role

Email Address

Enter Your Email

Password

Enter Your Password



Create a new account

Register As

Select Role

Name

Enter Your Name

Email Address

Enter your Email

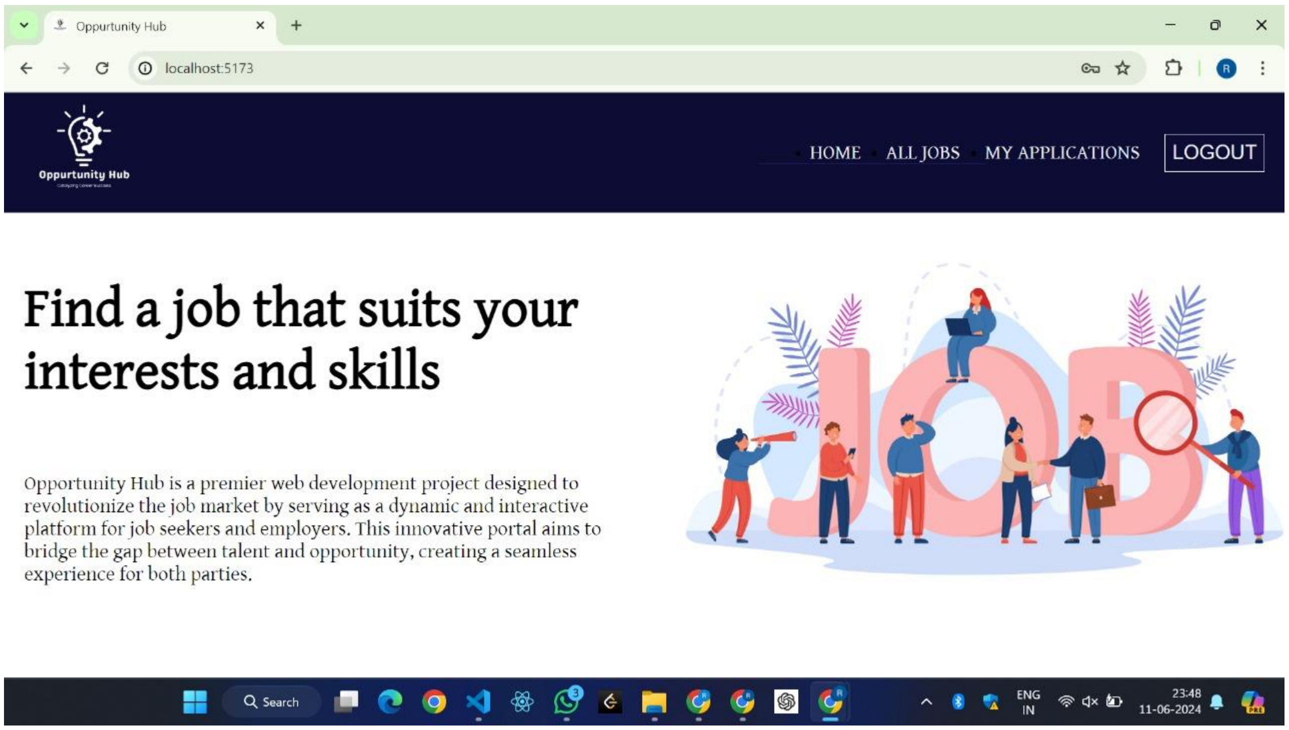
Phone Number

Enter your Phone Number

Password

Enter Your Password

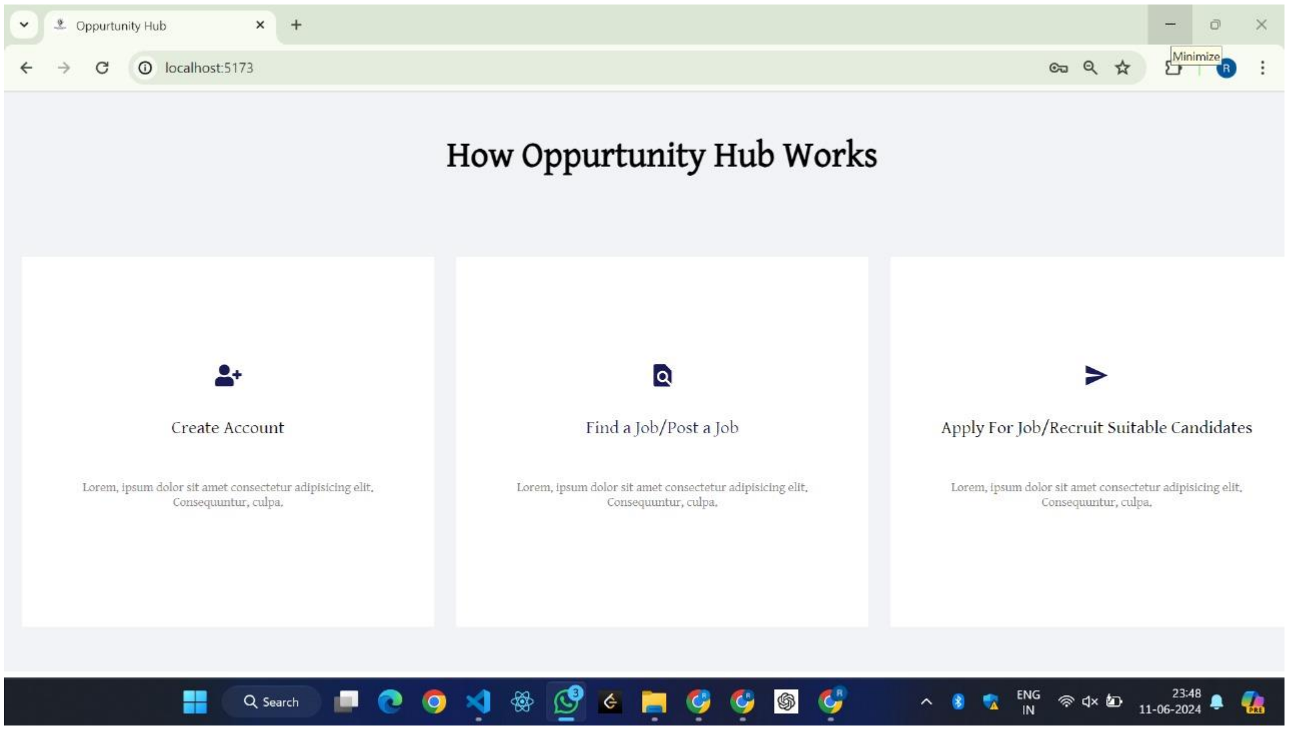
Home page



Find a job that suits your interests and skills

Oppurtunity Hub is a premier web development project designed to revolutionize the job market by serving as a dynamic and interactive platform for job seekers and employers. This innovative portal aims to bridge the gap between talent and opportunity, creating a seamless experience for both parties.

About Section



How Oppurtunity Hub Works

Create Account

Lorem, ipsum dolor sit amet consectetur adipisicing elit, Consequuntur, culpa.

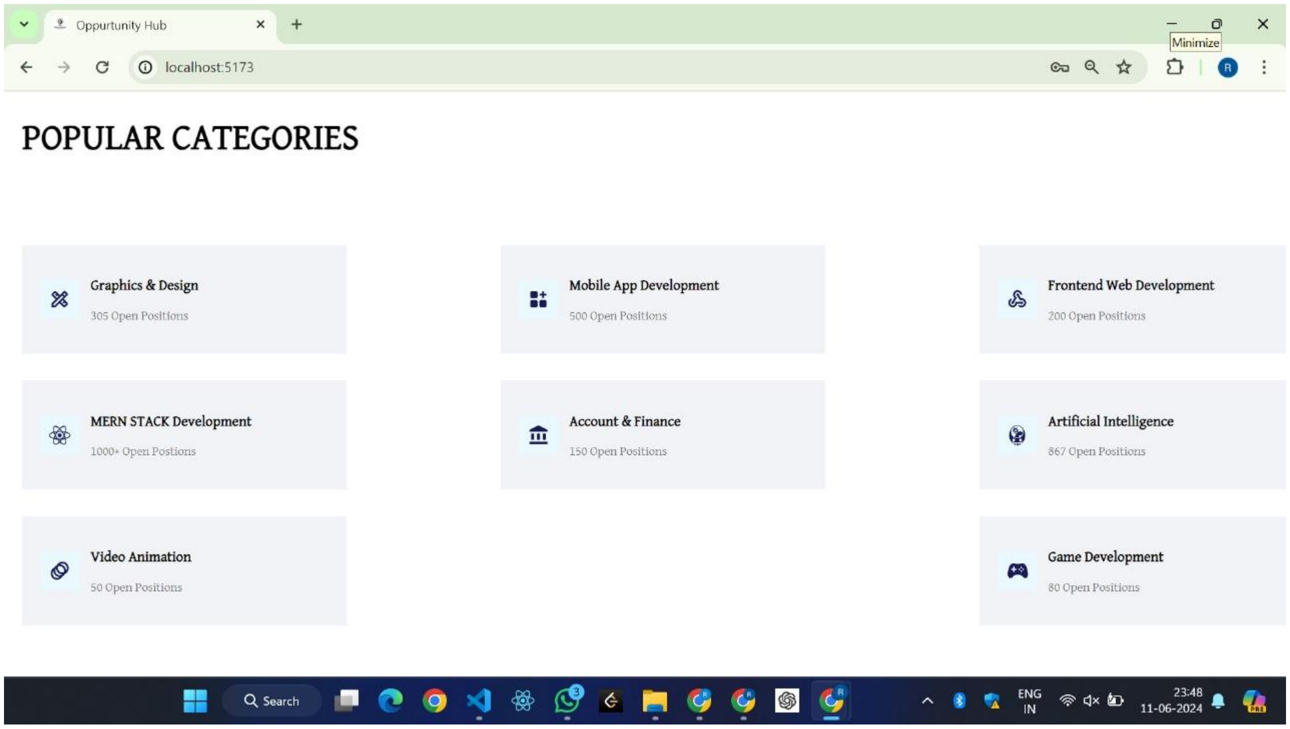
Find a Job/Post a Job

Lorem, ipsum dolor sit amet consectetur adipisicing elit, Consequuntur, culpa.

Apply For Job/Recruit Suitable Candidates

Lorem, ipsum dolor sit amet consectetur adipisicing elit, Consequuntur, culpa.

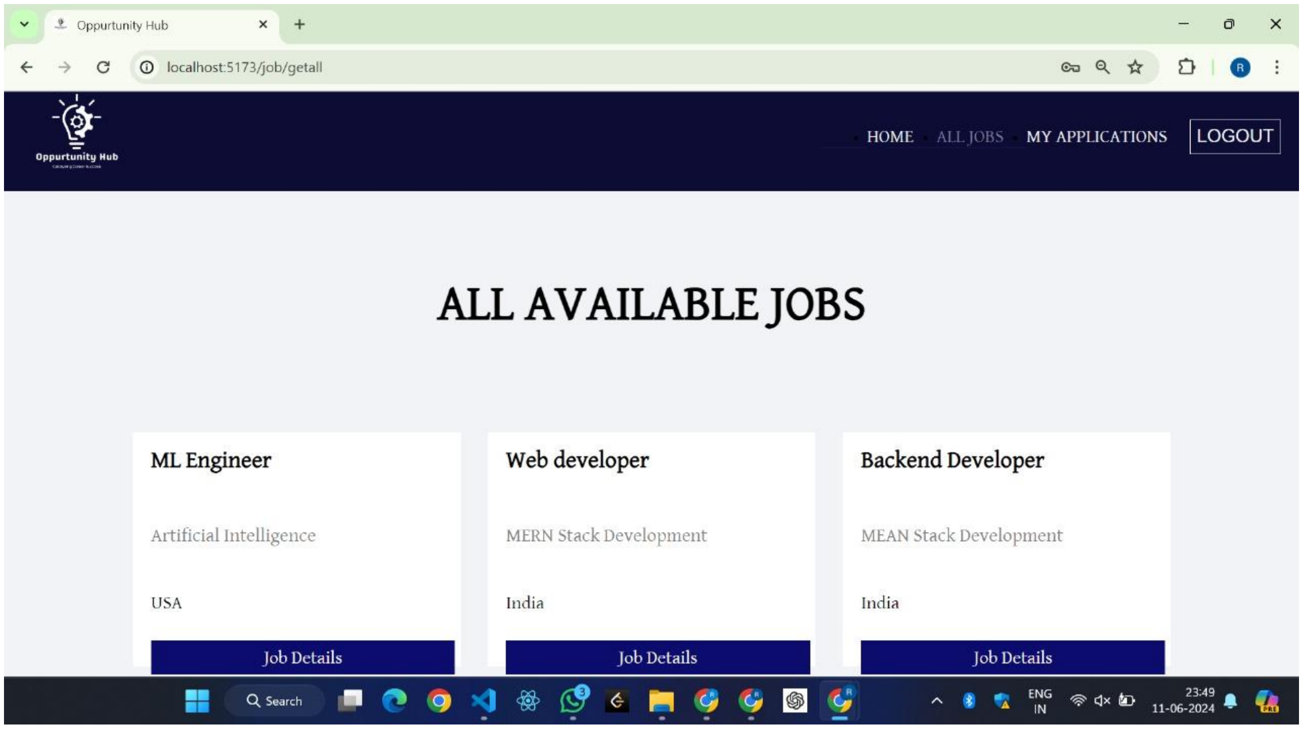
About Section



POPULAR CATEGORIES

Graphics & Design 305 Open Positions	Mobile App Development 500 Open Positions	Frontend Web Development 200 Open Positions
MERN STACK Development 1000+ Open Postions	Account & Finance 150 Open Positions	Artificial Intelligence 867 Open Positions
Video Animation 50 Open Positions		Game Development 80 Open Positions

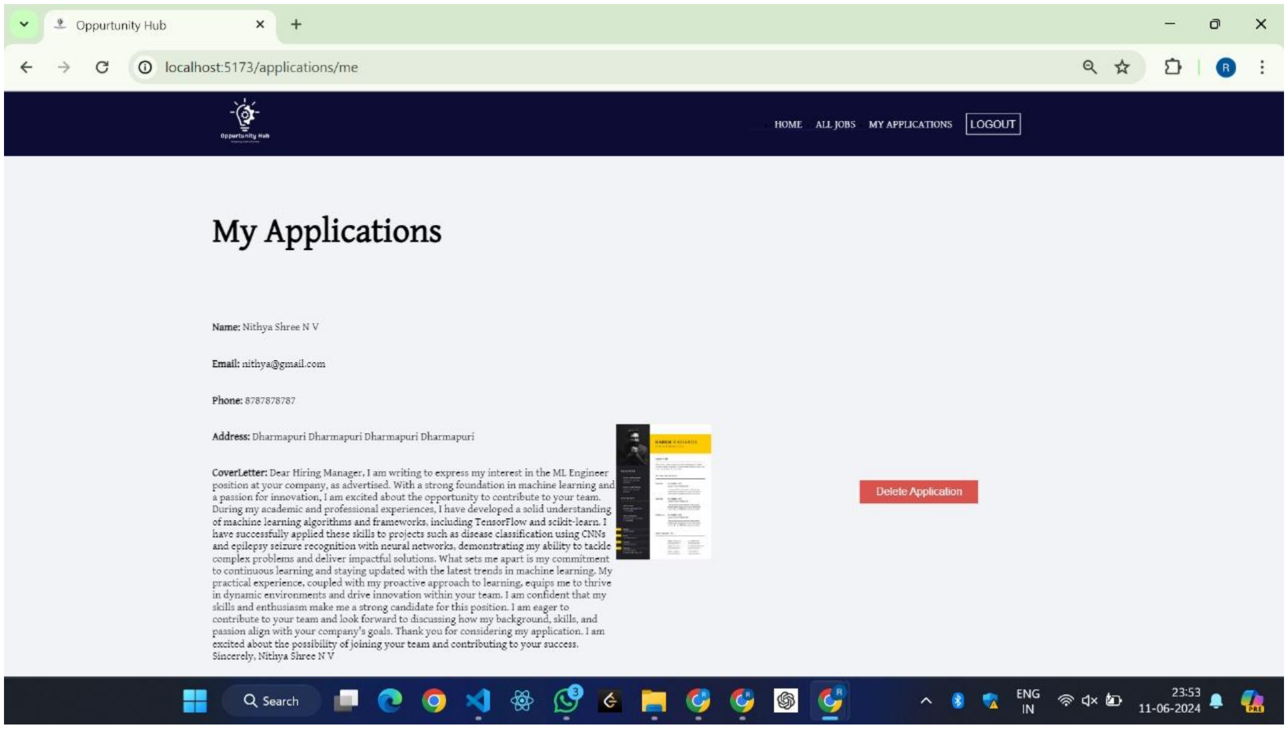
JOB Seeker's All Jobs Page



ALL AVAILABLE JOBS

ML Engineer	Web developer	Backend Developer
Artificial Intelligence	MERN Stack Development	MEAN Stack Development
USA	India	India
Job Details	Job Details	Job Details

JOB Seeker's My Application Page



My Applications

Name: Nithya Shree N V

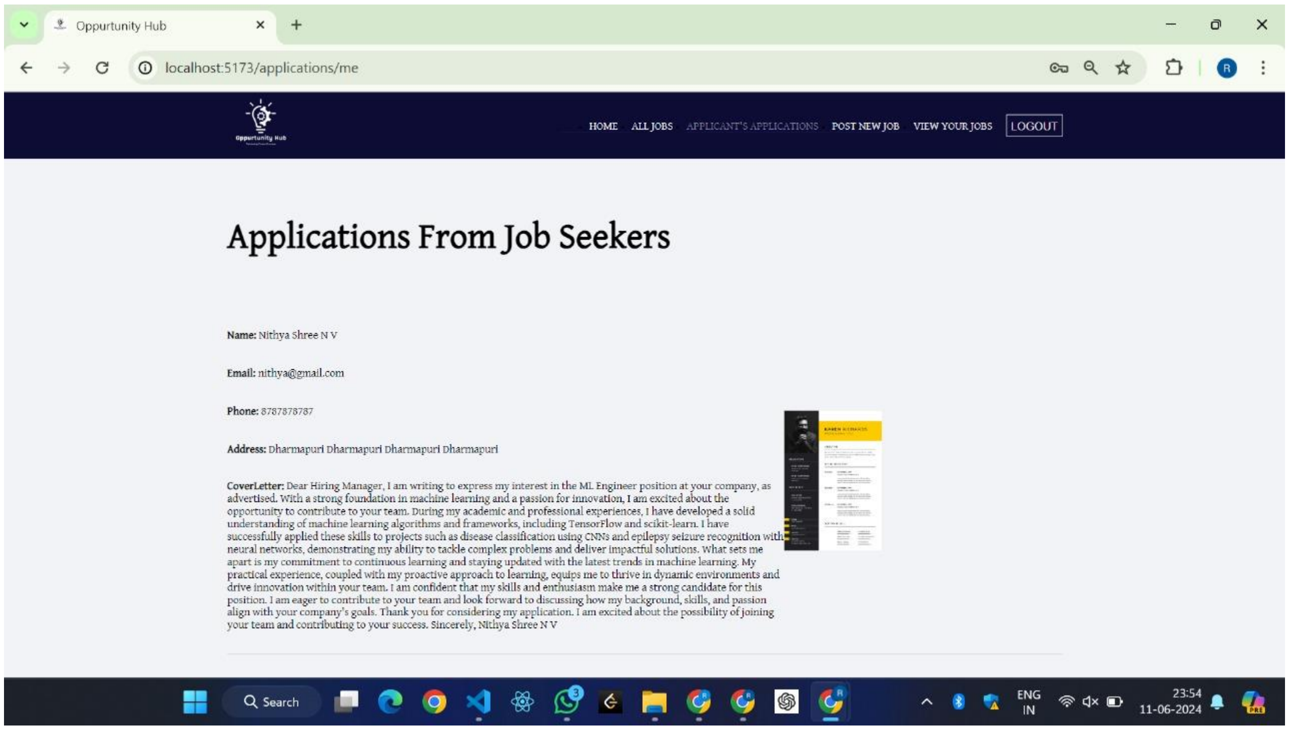
Email: nithya@gmail.com

Phone: 8787878787

Address: Dharmapuri Dharmapuri Dharmapuri Dharmapuri

CoverLetter: Dear Hiring Manager, I am writing to express my interest in the ML Engineer position at your company, as advertised. With a strong foundation in machine learning and a passion for innovation, I am excited about the opportunity to contribute to your team. During my academic and professional experiences, I have developed a solid understanding of machine learning algorithms and frameworks, including TensorFlow and scikit-learn. I have successfully applied these skills to projects such as disease classification using CNNs and epilepsy seizure recognition with neural networks, demonstrating my ability to tackle complex problems and deliver impactful solutions. What sets me apart is my commitment to continuous learning and staying updated with the latest trends in machine learning. My practical experience, coupled with my proactive approach to learning, equips me to thrive in dynamic environments and drive innovation within your team. I am confident that my skills and enthusiasm make me a strong candidate for this position. I am eager to contribute to your team and look forward to discussing how my background, skills, and passion align with your company's goals. Thank you for considering my application. I am excited about the possibility of joining your team and contributing to your success. Sincerely, Nithya Shree N V

Employer's Applicant Applications page



Applications From Job Seekers

Name: Nithya Shree N V

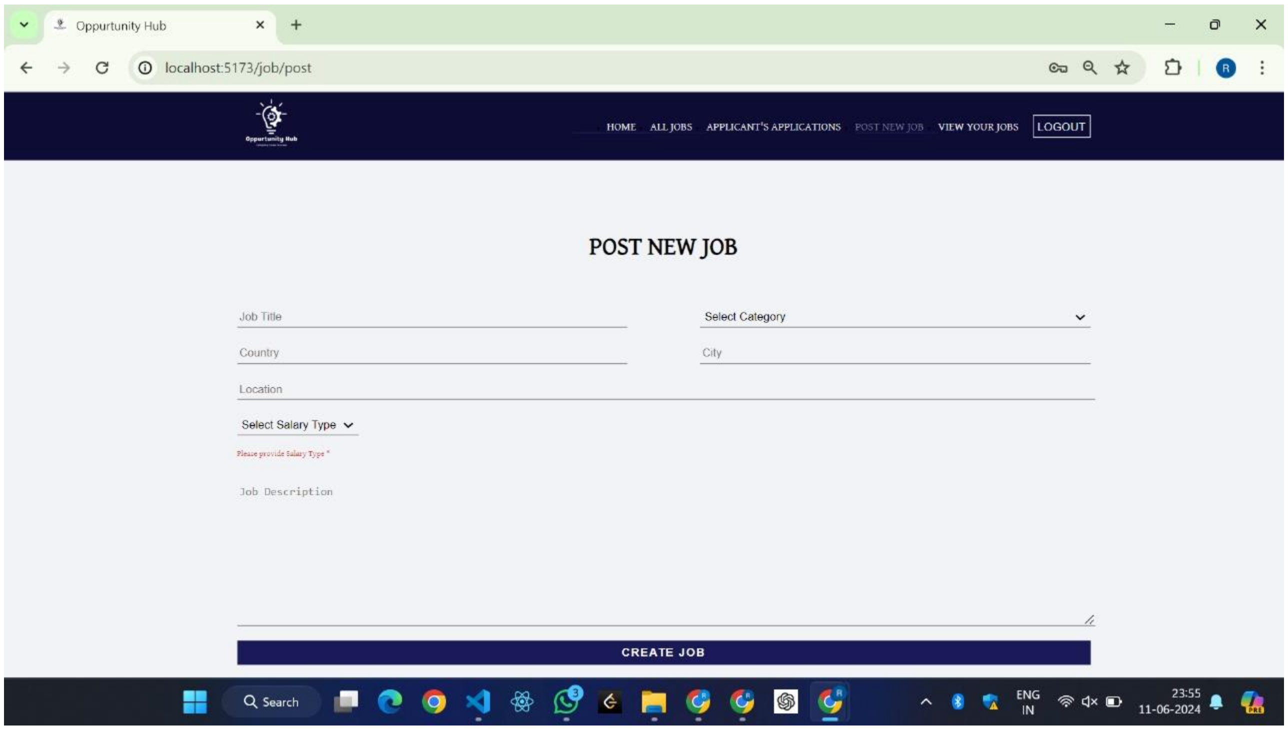
Email: nithya@gmail.com

Phone: 8787878787

Address: Dharmapuri Dharmapuri Dharmapuri Dharmapuri

CoverLetter: Dear Hiring Manager, I am writing to express my interest in the ML Engineer position at your company, as advertised. With a strong foundation in machine learning and a passion for innovation, I am excited about the opportunity to contribute to your team. During my academic and professional experiences, I have developed a solid understanding of machine learning algorithms and frameworks, including TensorFlow and scikit-learn. I have successfully applied these skills to projects such as disease classification using CNNs and epilepsy seizure recognition with neural networks, demonstrating my ability to tackle complex problems and deliver impactful solutions. What sets me apart is my commitment to continuous learning and staying updated with the latest trends in machine learning. My practical experience, coupled with my proactive approach to learning, equips me to thrive in dynamic environments and drive innovation within your team. I am confident that my skills and enthusiasm make me a strong candidate for this position. I am eager to contribute to your team and look forward to discussing how my background, skills, and passion align with your company's goals. Thank you for considering my application. I am excited about the possibility of joining your team and contributing to your success. Sincerely, Nithya Shree N V

Employer’s Post New JOB page



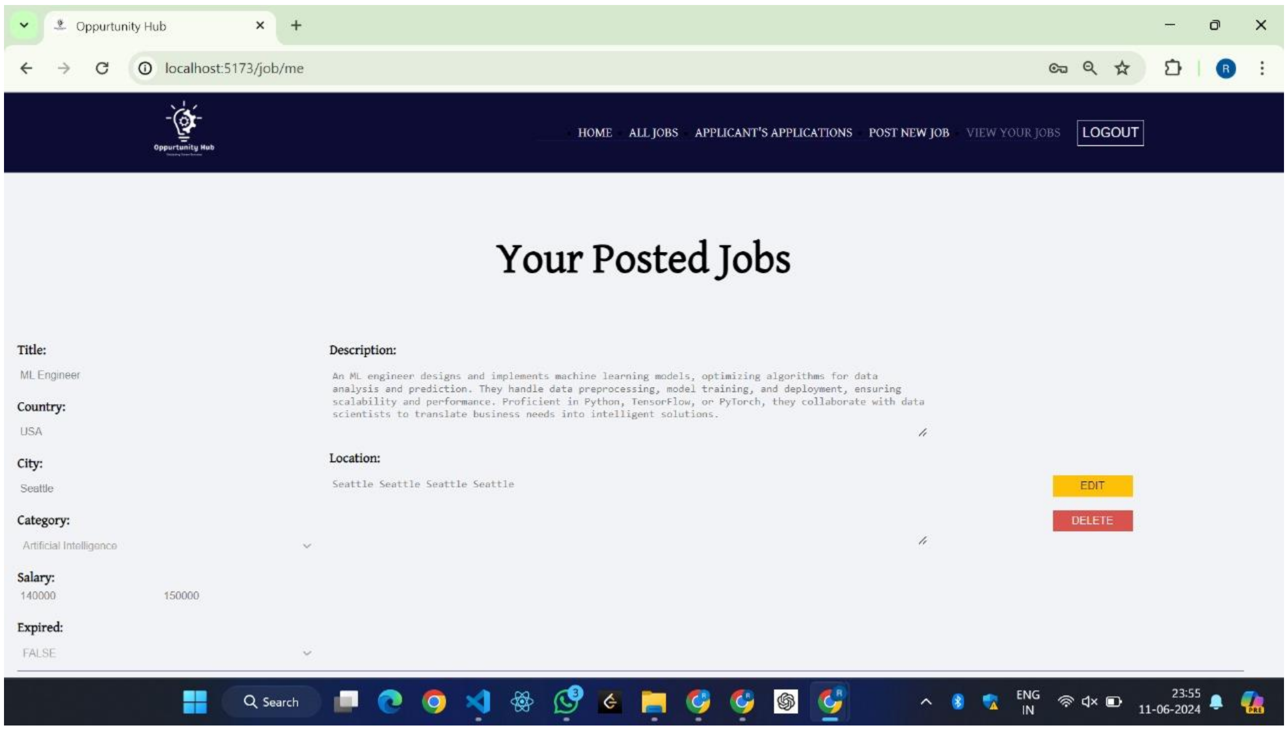
POST NEW JOB

Job Title	Select Category
Country	City
Location	
Select Salary Type	

Please provide Salary Type *

Job Description

Employer’s Your Posted JOB page



Your Posted Jobs

Title: ML Engineer

Country: USA

City: Seattle

Category: Artificial Intelligence

Salary: 140000 150000

Expired: FALSE

Description:

An ML engineer designs and implements machine learning models, optimizing algorithms for data analysis and prediction. They handle data preprocessing, model training, and deployment, ensuring scalability and performance. Proficient in Python, TensorFlow, or PyTorch, they collaborate with data scientists to translate business needs into intelligent solutions.

Location:

Seattle Seattle Seattle Seattle

SAMPLE CODING:

frontend/index.html

```
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<link rel="icon" type="image/png" href="icon.png" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>Oppurtunity Hub</title>
</head>
<body>
<div id="root"></div>
<script type="module" src="/src/main.jsx"></script>
</body>
</html>
```

frontend/src/main.jsx

```
import React, { createContext, useState } from "react";
import ReactDOM from "react-dom/client";
import App from "./App.jsx";

export const Context = createContext({
  isAuthorized: false,
});

const AppWrapper = () => {
  const [isAuthorized, setIsAuthorized] = useState(false);
  const [user, setUser] = useState({});

  return (
    <Context.Provider
      value={{
        isAuthorized,
        setIsAuthorized,
        user,
        setUser,
      }}
    >
      <App />
    </Context.Provider>
  );
};

ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <AppWrapper />
  </React.StrictMode>
);
```

frontend/app.jsx

```
import React, { useContext, useEffect } from "react";
import "./App.css";
import { Context } from "./main";
import { BrowserRouter, Route, Routes } from "react-router-dom";
import Login from "./components/Auth/Login";
import Register from "./components/Auth/Register";
import { Toaster } from "react-hot-toast";
import axios from "axios";
import Navbar from "./components/Layout/Navbar";
import Footer from "./components/Layout/Footer";
import Home from "./components/Home/Home";
import Jobs from "./components/Job/Jobs";
import JobDetails from "./components/Job/JobDetails";
import Application from "./components/Application/Application";
import MyApplications from "./components/Application/MyApplications";
import PostJob from "./components/Job/PostJob";
import NotFound from "./components/NotFound/NotFound";
import MyJobs from "./components/Job/MyJobs";

const App = () => {
  const { isAuthorized, setIsAuthorized, setUser } = useContext(Context);
  useEffect(() => {
    const fetchUser = async () => {
      try {
        const response = await axios.get(
          "https://localhost:5173/api/v1/user/getuser",
          {
            withCredentials: true,
          }
        );
        setUser(response.data.user);
        setIsAuthorized(true);
      } catch (error) {
        setIsAuthorized(false);
      }
    };
    fetchUser();
  }, [isAuthorized]);

  return (
    <>
      <BrowserRouter>
        <Navbar />
        <Routes>
          <Route path="/login" element={<Login />} />
          <Route path="/register" element={<Register />} />
          <Route path="/" element={<Home />} />
          <Route path="/job/getall" element={<Jobs />} />
          <Route path="/job/:id" element={<JobDetails />} />
          <Route path="/application/:id" element={<Application />} />
          <Route path="/applications/me" element={<MyApplications />} />
          <Route path="/job/post" element={<PostJob />} />
          <Route path="/job/me" element={<MyJobs />} />
          <Route path="*" element={<NotFound />} />
        </Routes>
        <Footer />
        <Toaster />
      </BrowserRouter>
    </>
  );
};

export default App;
```

Backend/server.js

```
import app from "./app.js";
import cloudinary from "cloudinary";

cloudinary.v2.config({
  cloud_name: process.env.CLOUDINARY_CLIENT_NAME,
  api_key: process.env.CLOUDINARY_CLIENT_API,
  api_secret: process.env.CLOUDINARY_CLIENT_SECRET,
});
```

```
app.listen(process.env.PORT, () => {
  console.log(`Server running at port ${process.env.PORT}`);
});
```

Backend/database/dbConnection.js

```
import mongoose from "mongoose";

export const dbConnection = () => {
  mongoose
    .connect(process.env.MONGO_URI, {
      dbName: "jobSeek",
    })
    .then(() => {
      console.log("Connected to database.");
    })
    .catch((err) => {
      console.log(`Some Error occured. ${err}`);
    });
}
```

Backend/app.js

```
import express from "express";
import dotenv from "dotenv";
import cors from "cors";
import cookieParser from "cookie-parser";
import fileUpload from "express-fileupload";
import jobRouter from "./routes/jobRouter.js";
import userRouter from "./routes/userRouter.js";
import applicationRouter from "./routes/applicationRouter.js";
import { dbConnection } from "./database/dbConnection.js";
import { errorMiddleware } from "./middlewares/error.js";

const app = express();
dotenv.config({path:"./config/config.env" });
app.use(
    cors({
      origin: [process.env.FRONTEND_URL],
      method: ["GET", "POST", "DELETE", "PUT"],
      credentials: true,
    })
  );

  app.use(cookieParser());
  app.use(express.json());
  app.use(express.urlencoded({ extended: true }));
  dbConnection();

app.use(
    fileUpload({
      useTempFiles: true,
      tempFileDir: "/tmp/",
    })
  );
  app.use("/api/v1/user", userRouter);
  app.use("/api/v1/job", jobRouter);
  app.use("/api/v1/application", applicationRouter);
  app.use(errorMiddleware);

export default
```

Backend/models/userSchema.js

```
import mongoose from "mongoose";
import validator from "validator";
import bcrypt from "bcrypt";
import jwt from "jsonwebtoken";

const userSchema = new mongoose.Schema({
  name: {
    type: String,
    required: [true, "Please enter your Name!"],
    minLength: [3, "Name must contain at least 3 Characters!"],
    maxLength: [30, "Name cannot exceed 30 Characters!"],
  },
  email: {
    type: String,
    required: [true, "Please enter your Email!"],
    validate: [validator.isEmail, "Please provide a valid Email!"],
  },
  phone: {
    type: String,
    required: [true, "Please enter your Phone Number!"],
    validate: {
      validator: function(value) {
        // Check if the phone number is exactly 10 digits and contains only digits
        return /^[6-9]\d{9}$/.test(value);
      },
      message: "Phone number must be exactly 10 digits long and contain only digits!",
    },
  },
  password: {
    type: String,
    required: [true, "Please provide a Password!"],
    minLength: [8, "Password must contain at least 8 characters!"],
    maxLength: [32, "Password cannot exceed 32 characters!"],
    select: false,
  },
  role: {
    type: String,
    required: [true, "Please select a role"],
    enum: ["Job Seeker", "Employer", "Data Analyst", "ML Engineer"],
  },
  createdAt: {
    type: Date,
    default: Date.now,
  },
});

// Pre-save hook to hash the password before saving
userSchema.pre("save", async function(next) {
  if (!this.isModified("password")) {
    return next();
  }
  try {
    const salt = await bcrypt.genSalt(10);
    this.password = await bcrypt.hash(this.password, salt);
    next();
  } catch (error) {
    next(error);
  }
});

// Method to compare passwords
userSchema.methods.comparePassword = async function(enteredPassword) {
  return await bcrypt.compare(enteredPassword, this.password);
};

// Method to generate JWT token
userSchema.methods.getJWTToken = function() {
  return jwt.sign({ id: this._id }, process.env.JWT_SECRET_KEY, {
    expiresIn: process.env.JWT_EXPIRE,
  });
};

export const User = mongoose.model("User", userSchema);
```

Backend/models/applicationSchema.js

```
import mongoose from "mongoose";
import validator from "validator";

const applicationSchema = new mongoose.Schema({
  name: {
    type: String,
    required: [true, "Please enter your Name!"],
    minLength: [3, "Name must contain at least 3 Characters!"],
    maxLength: [30, "Name cannot exceed 30 Characters!"],
  },
  email: {
    type: String,
    required: [true, "Please enter your Email!"],
    validate: [validator.isEmail, "Please provide a valid Email!"],
  },
  coverLetter: {
    type: String,
    required: [true, "Please provide a cover letter!"],
  },
  phone: {
    type: Number,
    required: [true, "Please enter your Phone Number!"],
  },
  address: {
    type: String,
    required: [true, "Please enter your Address!"],
  },
  resume: {
    public_id: {
      type: String,
      required: true,
    },
    url: {
      type: String,
      required: true,
    },
  },
  applicantID: {
    user: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "User",
      required: true,
    },
    role: {
      type: String,
      enum: ["Job Seeker"],
      required: true,
    },
  },

  employerID: {
    user: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "User",
      required: true,
    },
    role: {
      type: String,
      enum: ["Employer"],
      required: true,
    },
  },
});

export const Application = mongoose.model("Application", applicationSchema);
```

Backend/models/jobSchema.js

```
import mongoose from "mongoose";

const jobSchema = new mongoose.Schema({
  title: {
    type: String,
    required: [true, "Please provide a title."],
    minLength: [3, "Title must contain at least 3 Characters!"],
    maxLength: [30, "Title cannot exceed 30 Characters!"],
  },
  description: {
    type: String,
    required: [true, "Please provide decription."],
    minLength: [30, "Description must contain at least 30 Characters!"],
    maxLength: [500, "Description cannot exceed 500 Characters!"],
  },
  category: {
    type: String,
    required: [true, "Please provide a category."],
  },
  country: {
    type: String,
    required: [true, "Please provide a country name."],
  },
  city: {
    type: String,
    required: [true, "Please provide a city name."],
  },
  location: {
    type: String,
    required: [true, "Please provide location."],
    minLength: [20, "Location must contian at least 20 characters!"],
  },
  fixedSalary: {
    type: Number,
    minLength: [4, "Salary must contain at least 4 digits"],
    maxLength: [9, "Salary cannot exceed 9 digits"],
  },
  salaryFrom: {
    type: Number,
    minLength: [4, "Salary must contain at least 4 digits"],
    maxLength: [9, "Salary cannot exceed 9 digits"],
  },
  salaryTo: {
    type: Number,
    minLength: [4, "Salary must contain at least 4 digits"],
    maxLength: [9, "Salary cannot exceed 9 digits"],
  },
  expired: {
    type: Boolean,
    default: false,
  },
  jobPostedOn: {
    type: Date,
    default: Date.now,
  },
  postedBy: {
    type: mongoose.Schema.ObjectId,
    ref: "User",
    required: true,
  },
});

export const Job = mongoose.model("Job", jobSchema);
```

CONCLUSION:

Opportunity Hub stands as a testament to the innovative potential of web development, utilizing the robust MERN stack—MongoDB, Express, React, and Node.js—to create a dynamic and efficient job seek portal. By seamlessly integrating front-end and back-end technologies, Opportunity Hub offers a powerful platform that effectively bridges the gap between job seekers and employers.

For job seekers, the platform provides a user-friendly interface to showcase their skills and find relevant job opportunities, while employers benefit from advanced search and filtering capabilities to discover and connect with qualified candidates. Enhanced by real-time notifications, robust authentication, and comprehensive data analytics, Opportunity Hub ensures a secure, responsive, and engaging user experience.

In essence, Opportunity Hub is more than just a job portal; it is a comprehensive ecosystem designed to foster meaningful professional connections and streamline the recruitment process. By leveraging modern web technologies, Opportunity Hub is poised to transform the employment landscape, making it easier for talent and opportunities to find each other.