import React from "react";

import { CompanyData } from "../../Data/Company"; // Ensure correct import path

// Component to display company details

function AboutComp() {

  return (

    <div className="bg-gray-900 text-white p-8 rounded-2xl shadow-lg max-w-4xl mx-auto my-12">

      <h2 className="text-4xl font-bold text-center text-blue-400 mb-6">About Our Company</h2>

      {/\* Overview Section \*/}

      <div className="bg-gray-800 p-6 rounded-lg shadow-md mb-6">

        <h3 className="text-2xl font-semibold text-blue-300 mb-3">Company Overview</h3>

        <p className="text-gray-300">{CompanyData.OverView}</p>

      </div>

      {/\* Details Section \*/}

      <div className="space-y-4">

        {Object.keys(CompanyData).map((key, index) => {

          if (key !== "OverView") {

            return (

              <div key={index} className="p-4 bg-gray-800 rounded-lg shadow-md">

                <strong className="text-blue-300">{key}:</strong>{" "}

                <span className="text-gray-300">

                  {Array.isArray(CompanyData[key as keyof typeof CompanyData])

                    ? (CompanyData[key as keyof typeof CompanyData] as string[]).join(", ")

                    : CompanyData[key as keyof typeof CompanyData]}

                </span>

              </div>

            );

          }

          return null;

        })}

      </div>

      {/\* Specialties Section \*/}

      {CompanyData.Specialties && (

        <div className="bg-gray-800 p-6 rounded-lg shadow-md mt-6">

          <h3 className="text-2xl font-semibold text-blue-300 mb-3">Specialties</h3>

          <ul className="list-disc pl-6 text-gray-300">

            {CompanyData.Specialties.map((specialty, index) => (

              <li key={index}>{specialty}</li>

            ))}

          </ul>

        </div>

      )}

      {/\* Link to Company Website \*/}

      <div className="mt-6 text-center">

        <a

          href={CompanyData.Website}

          target="\_blank"

          rel="noopener noreferrer"

          className="text-blue-400 underline text-lg"

        >

          Visit our website

        </a>

      </div>

    </div>

  );

}

export default AboutComp;

import React, { useState, useEffect } from "react";

import {

  Divider,

  FileInput,

  NumberInput,

  Textarea,

  TextInput,

  Button,

  Notification,

  Loader,

} from "@mantine/core";

import { IconPaperclip, IconCheck } from "@tabler/icons-react";

import { useNavigate } from "react-router-dom";

function ApplyJobComp() {

  const [showPreviewOptions, setShowPreviewOptions] = useState(false);

  const [showNotification, setShowNotification] = useState(false);

  const [countdown, setCountdown] = useState(5); // Timer starts from 5 seconds

  const [isLoading, setIsLoading] = useState(false); // Loading state

  const navigate = useNavigate();

  const [formData, setFormData] = useState({

    fullName: "",

    email: "",

    phoneNumber: "",

    website: "",

    resume: null as File | null,

    coverLetter: "",

  });

  useEffect(() => {

    window.scrollTo(0, 0);

  }, []);

  const handleEditClick = () => {

    setShowPreviewOptions(false);

    window.scrollTo({ top: 0, behavior: "smooth" });

  };

  const handleSubmit = () => {

    setShowNotification(true);

    setIsLoading(true); // Start loading

    let timeLeft = 5;

    const countdownInterval = setInterval(() => {

      timeLeft -= 1;

      setCountdown(timeLeft);

      if (timeLeft === 0) {

        clearInterval(countdownInterval);

        setIsLoading(false); // Stop loading

        navigate("/find-job"); // Redirects to find-job route after 5 seconds

      }

    }, 1000);

  };

  const handleChange = (field: string, value: string | number | File | null) => {

    setFormData((prev) => ({ ...prev, [field]: value }));

  };

  return (

    <>

      {showNotification && (

        <div className="fixed top-4 left-1/2 transform -translate-x-1/2 z-50 w-full max-w-md">

          <Notification

            icon={<IconCheck stroke={2} />}

            color="teal"

            title="Application Submitted"

            withCloseButton={false}

            withBorder

          >

            Redirecting in {countdown} seconds...

          </Notification>

        </div>

      )}

      {/\* Loading Overlay \*/}

      {isLoading && (

        <div className="fixed inset-0 bg-gray-900 bg-opacity-60 flex justify-center items-center z-50">

          <div className="flex flex-col items-center space-y-4">

            <Loader color="teal" size="xl" />

            <p className="text-white text-lg font-semibold">Submitting your application...</p>

          </div>

        </div>

      )}

      <div className="p-6 max-w-2xl mx-auto bg-white shadow-md rounded-lg text-gray-800 mt-16">

        <header className="text-center space-y-4">

          <div className="flex justify-center">

            <img

              src="https://upload.wikimedia.org/wikipedia/commons/2/2f/Google\_2015\_logo.svg"

              alt="Company Logo"

              className="w-20 h-20 object-contain"

            />

          </div>

          <h1 className="text-4xl font-extrabold text-blue-600">Software Engineer III</h1>

          <div className="flex justify-center space-x-6 text-sm text-gray-500">

            <span className="text-yellow-500 font-semibold">Google</span>

            <span className="text-green-500 font-semibold">3 days ago</span>

            <span className="text-pink-500 font-semibold">48 Applicants</span>

          </div>

        </header>

        <Divider my="xl" />

        <div className="text-xl font-semibold mb-5 text-gray-700">Submit Your Application</div>

        <form className="space-y-4 text-gray-700">

          <TextInput

            label="Full Name"

            withAsterisk

            placeholder="Enter name"

            value={formData.fullName}

            onChange={(e) => handleChange("fullName", e.target.value)}

          />

          <TextInput

            label="Email"

            withAsterisk

            placeholder="Enter email"

            type="email"

            value={formData.email}

            onChange={(e) => handleChange("email", e.target.value)}

          />

          <NumberInput

            label="Phone Number"

            placeholder="Enter phone number"

            hideControls

            min={0}

            max={9999999999}

            withAsterisk

            value={formData.phoneNumber}

            onChange={(value) => handleChange("phoneNumber", value ?? 0)}

          />

          <TextInput

            label="Personal Website"

            placeholder="Enter URL"

            value={formData.website}

            onChange={(e) => handleChange("website", e.target.value)}

          />

          <FileInput

            leftSection={<IconPaperclip stroke={1.5} />}

            label="Attach Your CV"

            placeholder="Upload resume"

            withAsterisk

            onChange={(file) => handleChange("resume", file)}

          />

          <Textarea

            placeholder="Type about yourself"

            label="Cover Letter"

            autosize

            minRows={4}

            withAsterisk

            value={formData.coverLetter}

            onChange={(e) => handleChange("coverLetter", e.target.value)}

          />

        </form>

        <div className="mt-6">

          {!showPreviewOptions ? (

            <Button

              fullWidth

              variant="gradient"

              gradient={{ from: "indigo", to: "cyan" }}

              size="md"

              onClick={() => setShowPreviewOptions(true)}

            >

              Preview

            </Button>

          ) : (

            <div className="flex space-x-4">

              <Button

                fullWidth

                variant="outline"

                color="blue"

                size="md"

                onClick={handleEditClick}

              >

                Edit

              </Button>

              <Button

                fullWidth

                variant="gradient"

                gradient={{ from: "green", to: "teal" }}

                size="md"

                onClick={handleSubmit}

                disabled={isLoading} // Prevent multiple submissions

              >

                Submit

              </Button>

            </div>

          )}

        </div>

      </div>

    </>

  );

}

export default ApplyJobComp;

import React from "react";

import { Divider, Button, Avatar, Tabs } from "@mantine/core";

import { IconBriefcase, IconMapPin } from "@tabler/icons-react";

import AboutComp from "../ApplyJob/AboutComp";

import ApplyJobComp from "./ApplyJobComp";

import CompanyJob from "./CompanyJob";

import CompanyEmployee from "./CompanyEmployee";

function Company() {

  return (

    <div className="bg-gray-900 text-white shadow-xl rounded-2xl overflow-hidden max-w-3xl mx-auto p-4 transition-all duration-300 hover:shadow-2xl">

      {/\* Banner and Avatar \*/}

      <div className="relative">

        <img

          className="rounded-t-2xl w-full h-56 object-cover"

          src="/color7.png"

          alt="Banner"

        />

        <div className="absolute bottom-0 left-1/2 transform -translate-x-1/2 translate-y-1/2">

          <img

            className="rounded-full h-36 w-36 border-4 border-gray-900 shadow-lg hover:scale-110 transition-all duration-300"

            src="/Company/google.png"

            alt="Avatar"

          />

        </div>

      </div>

      {/\* Profile Details \*/}

      <div className="text-center mt-20">

        <h2 className="text-2xl font-bold text-white flex items-center justify-center">

          Google

        </h2>

        {/\* Team Avatars \*/}

        <div className="flex justify-center mt-4">

          <Avatar.Group spacing="sm">

            <Avatar src="image1.png" radius="xl" />

            <Avatar src="image2.png" radius="xl" />

            <Avatar src="image3.png" radius="xl" />

            <Avatar radius="xl">+5</Avatar>

          </Avatar.Group>

        </div>

        {/\* Location \*/}

        <p className="flex justify-center items-center text-gray-400 mt-4 text-md">

          <IconMapPin className="h-5 w-5 mr-2 text-red-400" stroke={1.5} />

          <span className="text-gray-300 font-medium">Noida, India</span>

        </p>

        <Divider className="my-6 border-gray-700" />

      </div>

      {/\* Tabs Section \*/}

      <Tabs variant="outline" defaultValue="about" className="mt-4">

        <Tabs.List className="flex justify-center space-x-4">

          <Tabs.Tab value="about" className="text-gray-300 font-medium">

            About

          </Tabs.Tab>

          <Tabs.Tab value="jobs" className="text-gray-300 font-medium">

            Jobs

          </Tabs.Tab>

          <Tabs.Tab value="employee" className="text-gray-300 font-medium">

            Employees

          </Tabs.Tab>

        </Tabs.List>

        <Tabs.Panel value="about" className="p-4 text-gray-300 text-md leading-relaxed mb-5">

          <AboutComp/>

        </Tabs.Panel>

        <Tabs.Panel value="jobs" className="p-4 text-gray-300 text-md">

          <CompanyJob/>

        </Tabs.Panel>

        <Tabs.Panel value="employee" className="p-4 text-gray-300 text-md">

          <CompanyEmployee/>

        </Tabs.Panel>

      </Tabs>

      {/\* Action Buttons \*/}

      <div className="flex justify-center space-x-4 mt-6">

        <Button variant="filled" color="blue" className="px-6 py-2 font-medium text-white">

          Follow

        </Button>

        <Button variant="outline" color="gray" className="px-6 py-2 font-medium text-gray-300 border-gray-600">

          Visit Website

        </Button>

      </div>

    </div>

  );

}

export default Company;

import { IconBookmark, IconExternalLink } from "@tabler/icons-react";

import React, { useState } from "react";

// Helper function to generate a logo dynamically based on the company name

const generateLogo = (companyName: string) => {

  // You can customize the logo generation logic here.

  // A simple example where the logo is just the initials of the company name in a stylized background.

  const initials = companyName

    .split(" ")

    .map((word) => word[0].toUpperCase())

    .join("");

  return `https://via.placeholder.com/150/0000FF/808080?text=${initials}`; // Example dynamic logo URL

};

interface CompanyCardProps {

  companyName: string;

  jobTitle: string;

  applicants: number;

  logoUrl?: string; // Optional prop to accept a custom logo URL (PNG/JPG)

}

const CompanyCard: React.FC<CompanyCardProps> = ({ companyName, jobTitle, applicants, logoUrl }) => {

  const [isSaved, setIsSaved] = useState(false);

  const toggleSave = () => setIsSaved(!isSaved);

  // Use custom logo or generate logo dynamically based on the company name

  const logoToDisplay = logoUrl || generateLogo(companyName);

  return (

    <div className="p-4 bg-gray-800 rounded-xl shadow-md">

      <div className="flex justify-between items-center">

        {/\* Left Section: Logo & Job Info \*/}

        <div className="flex gap-4 items-center">

          <div className="p-3 bg-gray-700 rounded-xl">

            <img

              className="h-16 w-16 object-contain rounded-lg"

              src={logoToDisplay} // Use custom logo or generated one

              alt={`${companyName} logo`}

            />

          </div>

          <div>

            <h3 className="text-xl font-semibold text-blue-400">{jobTitle}</h3>

            <p className="text-lg font-bold text-yellow-400">{companyName}</p>

            <p className="text-gray-400 text-sm">{applicants} Applicants</p>

          </div>

        </div>

        {/\* Save Button \*/}

        <button

          className="p-2 rounded-full hover:bg-gray-700 transition"

          onClick={toggleSave}

          aria-label={isSaved ? "Unsave job" : "Save job"}

        >

          <IconExternalLink

            className={`transition-colors ${isSaved ? "text-yellow-400" : "text-gray-400"}`}

            stroke={2}

            fill={isSaved ? "#FACC15" : "none"}

          />

        </button>

      </div>

    </div>

  );

};

export default CompanyCard;

import React from 'react'

import talentsData from '../FindTalent/TalentData'

import TalentCard from '../FindTalent/TalentCard'

function CompanyEmployee() {

  return (

    <div>

      <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-2 lg:grid-cols-2 gap-3 mt-7">

        {talentsData.map((talent) => (

          <TalentCard key={talent.id} {...talent} />

        ))}

      </div>

    </div>

  )

}

export default CompanyEmployee

import React from 'react';

import JobCard from '../FindJob/JobCard';

import { motion } from 'framer-motion';

import { jobList } from '../../Data/JobData';

function CompanyJob() {

  return (

    <div className="p-10 min-h-screen flex flex-col items-center bg-gradient-to-br from-purple-500 via-pink-500 to-orange-500">

      <h2 className="text-4xl font-extrabold text-white mb-10 text-center relative">

        <span className="relative z-10">Explore Job Opportunities</span>

      </h2>

      {jobList.length > 0 ? (

        <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-2 gap-6 w-full max-w-7xl">

          {jobList.map((job) => (

            <motion.div

              key={job.id}

              className="bg-white rounded-2xl shadow-lg hover:shadow-2xl transition-transform duration-300 border border-gray-300 p-4 flex flex-col justify-between"

              whileHover={{ scale: 1.05, rotateY: 10 }}

              whileTap={{ scale: 0.95 }}

              style={{ perspective: 1000 }}

            >

              <JobCard {...job} />

            </motion.div>

          ))}

        </div>

      ) : (

        <div className="flex flex-col items-center justify-center h-60 text-white text-center">

          <p className="text-2xl font-semibold">No jobs available at the moment.</p>

          <p className="text-lg mt-4">Check back later for new opportunities.</p>

        </div>

      )}

    </div>

  );

}

export default CompanyJob;

import { SimilarCompanies } from "../../Data/Company";

import CompanyCard from "../CompanyProfile/CompanyCard";

const SimilarCompany = () => {

  return (

    <div className="grid grid-cols-1 sm:grid-cols-1 md:grid-cols-2 lg:grid-cols-1 gap-3 p-2">

      {SimilarCompanies.map(({ name, employees }) => (

        <CompanyCard

          key={name}

          companyName={name}

          jobTitle={`Employees: ${employees}`}

          applicants={employees}

        />

      ))}

    </div>

  );

};

export default SimilarCompany;

import React, { memo, useState } from "react";

import { IconBookmark, IconClock } from "@tabler/icons-react";

import { Link } from "react-router-dom";

export interface Job {

  id: string;

  jobTitle: string;

  companyName: string;

  applicants: number;

  experience: string;

  jobType: string;

  location: string;

  salary: string;

  postDayAgo: number;

  description: string;

}

const generateCompanyLogoSVG = (companyName: string) => {

  const initials = companyName

    .split(" ")

    .map((word) => word[0]?.toUpperCase())

    .join("");

  const bgColor = "#F59E0B";

  const textColor = "#FFFFFF";

  const svg = `

    <svg xmlns="http://www.w3.org/2000/svg" width="100" height="100">

      <rect width="100%" height="100%" fill="${bgColor}" rx="15"/>

      <text x="50%" y="55%" font-size="32" font-family="Arial" font-weight="bold" fill="${textColor}" text-anchor="middle">

        ${initials}

      </text>

    </svg>

  `;

  return `data:image/svg+xml;base64,${btoa(unescape(encodeURIComponent(svg)))}`;

};

const JobCard: React.FC<Job> = memo(

  ({ id, jobTitle, companyName, applicants, experience, jobType, location, salary, postDayAgo, description }) => {

    const jobDetails = [experience, jobType, location];

    const [isSaved, setIsSaved] = useState(false);

    const toggleSave = () => {

      setIsSaved((prev) => !prev);

    };

    return (

      <div className="p-6 bg-gray-900 text-gray-100 shadow-xl rounded-2xl border border-gray-700 flex flex-col gap-5 transition-all hover:shadow-2xl hover:scale-105">

        <div className="flex justify-between items-center">

          <div className="flex gap-4 items-center">

            <div className="p-3 bg-gray-700 rounded-xl">

              <img

                className="h-16 w-16 object-contain rounded-lg"

                src={generateCompanyLogoSVG(companyName)}

                alt={`${companyName} logo`}

              />

            </div>

            <div>

              <h3 className="text-xl font-semibold text-blue-400">{jobTitle}</h3>

              <p className="text-lg font-bold text-gradient bg-gradient-to-r from-yellow-400 to-pink-400 bg-clip-text text-transparent">

                {companyName}

              </p>

              <p className="text-gray-400 text-sm">{applicants} Applicants</p>

            </div>

          </div>

          <button

            className="p-2 rounded-full hover:bg-gray-700 transition"

            onClick={toggleSave}

          >

            <IconBookmark

              className={`transition-colors ${isSaved ? "text-yellow-400" : "text-gray-400"}`}

              stroke={2}

              fill={isSaved ? "#FACC15" : "none"}

            />

          </button>

        </div>

        <div className="flex flex-wrap gap-2">

          {jobDetails.map((item, index) => (

            <span key={index} className="px-3 py-1 bg-indigo-500 text-white font-medium rounded-full text-xs">

              {item}

            </span>

          ))}

        </div>

        <p className="text-gray-300 text-sm line-clamp-2">{description}</p>

        <div className="flex justify-between items-center mt-3">

          <span className="text-lg font-bold text-green-400">

            {salary.includes("LPA") ? `₹ ${salary}` : salary}

          </span>

          <div className="flex items-center gap-1 text-gray-400 text-sm">

            <IconClock className="h-5 w-5" stroke={1.5} />

            {postDayAgo} days ago

          </div>

        </div>

        <Link to="/job">

          <button className="mt-3 py-2 px-5 bg-gradient-to-r from-blue-500 to-purple-600 text-white rounded-lg font-medium text-sm hover:from-blue-600 hover:to-purple-700 transition-all">

            Apply Now

          </button>

        </Link>

      </div>

    );

  }

);

export default JobCard;

import React, { memo, useState } from "react";

import Sort from "./Sort";

import JobCard from "../FindJob/JobCard";

export interface Job {

  id: string;

  jobTitle: string;

  companyName: string;

  applicants: number;

  experience: string;

  jobType: string;

  location: string;

  salary: string;

  postDayAgo: number;

  description: string;

}

const isValidJob = (job: any): job is Job => {

  return (

    typeof job === "object" &&

    job !== null &&

    typeof job.id === "string" &&

    typeof job.jobTitle === "string" &&

    typeof job.companyName === "string" &&

    typeof job.applicants === "number" &&

    typeof job.experience === "string" &&

    typeof job.jobType === "string" &&

    typeof job.location === "string" &&

    typeof job.salary === "string" &&

    typeof job.postDayAgo === "number" &&

    typeof job.description === "string"

  );

};

const getValidJobs = (data: any): Job[] => {

  if (!Array.isArray(data)) {

    console.error("Invalid job data format - expected an array", data);

    return [];

  }

  return data.filter(isValidJob);

};

const Jobs: React.FC = () => {

  const [jobList] = useState<Job[]>(getValidJobs([])); // Assuming an empty array as a placeholder for importedJobList

  return (

    <div className="max-w-7xl mx-auto p-6 bg-white shadow-lg rounded-2xl border border-gray-200">

      {/\* Header Section \*/}

      <div className="flex justify-between items-center mb-6 border-b pb-4">

        <h2 className="text-3xl font-bold text-gray-800">Recommended Jobs</h2>

        <Sort />

      </div>

      {/\* Job List \*/}

      {jobList.length > 0 ? (

        <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4 gap-6">

          {jobList.map((job) => (

            <JobCard key={job.id} {...job} />

          ))}

        </div>

      ) : (

        <div className="flex flex-col items-center text-gray-500 text-center py-10">

          <p className="mb-3 text-lg">No jobs available at the moment.</p>

        </div>

      )}

    </div>

  );

};

export default memo(Jobs);

import { useEffect, useState, useMemo, useCallback } from "react";

import { Combobox, Pill, PillsInput, useCombobox } from "@mantine/core";

import { IconChevronDown, IconX } from "@tabler/icons-react";

interface MultiInputProps {

  options: string[];

}

const MultiInput: React.FC<MultiInputProps> = ({ options = [] }) => {

  const [search, setSearch] = useState<string>("");

  const [selectedItems, setSelectedItems] = useState<string[]>([]);

  const combobox = useCombobox({

    onDropdownClose: () => combobox.resetSelectedOption(),

    onDropdownOpen: () => combobox.updateSelectedOptionIndex("active"),

  });

  useEffect(() => {

    setSearch("");

  }, [options]);

  const handleSelect = useCallback((item: string) => {

    setSearch("");

    setSelectedItems((prev) =>

      prev.includes(item) ? prev.filter((s) => s !== item) : [...prev, item]

    );

  }, []);

  const handleRemove = useCallback((item: string) => {

    setSelectedItems((prev) => prev.filter((s) => s !== item));

  }, []);

  const filteredOptions = useMemo(

    () => options.filter((opt) => opt.toLowerCase().includes(search.toLowerCase())),

    [options, search]

  );

  return (

    <div className="w-full max-w-md p-4 bg-gray-800 text-white rounded-lg shadow-md">

      <Combobox store={combobox} onOptionSubmit={(item) => handleSelect(item as string)} withinPortal={false}>

        <Combobox.DropdownTarget>

          <PillsInput

            variant="unstyled"

            rightSection={<IconChevronDown size={18} className="text-gray-300" />}

            onClick={() => search === "" && combobox.openDropdown()}

            className="w-full px-4 py-2 border border-gray-600 rounded-md bg-gray-800 text-white"

          >

            <Pill.Group className="flex flex-wrap gap-2">

              {selectedItems.slice(0, 2).map((item) => (

                <Pill

                  key={item}

                  withRemoveButton

                  onRemove={() => handleRemove(item)}

                  className="text-sm bg-gray-700 text-white px-3 py-1 rounded-lg"

                >

                  {item}

                </Pill>

              ))}

              {selectedItems.length > 2 && (

                <Pill className="bg-gray-600 text-sm px-3 py-1 rounded-lg">

                  +{selectedItems.length - 2}

                </Pill>

              )}

            </Pill.Group>

            <input

              value={search}

              placeholder="Search..."

              onChange={(e) => setSearch(e.currentTarget.value)}

              className="text-sm w-full bg-gray-800 text-white border-none px-2 py-1 focus:outline-none"

            />

          </PillsInput>

        </Combobox.DropdownTarget>

        <Combobox.Dropdown className="w-full max-w-md bg-gray-800 border border-gray-700 shadow-lg rounded-lg">

          <Combobox.Options>

            {filteredOptions.map((item) => (

              <Combobox.Option value={item} key={item}>

                <label className="flex items-center gap-2 cursor-pointer p-3 hover:bg-gray-700 rounded">

                  <input

                    type="checkbox"

                    checked={selectedItems.includes(item)}

                    onChange={() => handleSelect(item)}

                    className="w-5 h-5 border-2 rounded-md bg-gray-700 border-gray-500"

                  />

                  {item}

                </label>

              </Combobox.Option>

            ))}

            {search.trim() && !options.includes(search) && (

              <Combobox.Option value={search}>

                <span className="text-gray-400 block p-3">+ Add "{search}"</span>

              </Combobox.Option>

            )}

            {search.trim() && filteredOptions.length === 0 && (

              <Combobox.Empty className="text-gray-400 p-3">No results found</Combobox.Empty>

            )}

          </Combobox.Options>

        </Combobox.Dropdown>

      </Combobox>

      {selectedItems.length > 0 && (

        <button

          onClick={() => setSelectedItems([])}

          className="mt-3 px-4 py-2 bg-red-600 text-white rounded-lg flex items-center gap-1 hover:bg-red-700"

        >

          <IconX size={16} /> Clear Selection

        </button>

      )}

    </div>

  );

};

export default MultiInput;

import React, { useState } from "react";

import { IconBriefcase, IconMapPin, IconRecharging, IconSearch } from "@tabler/icons-react";

// Define the type for dropdown data

interface DropdownItem {

  title: string;

  icon: React.ElementType;

  options: string[];

}

// Dropdown data

const dropdownData: DropdownItem[] = [

  {

    title: "Job Title",

    icon: IconSearch,

    options: [

      "Designer", "Developer", "Manager", "Tester", "Engineer",

      "Marketing Specialist", "Data Analyst", "Business Analyst",

      "Product Manager", "HR Manager", "Finance Manager",

      "Accountant", "Sales Manager", "Marketing Manager",

      "Customer Support", "Customer Success", "Customer Service",

      "Customer Care"

    ],

  },

  {

    title: "Location",

    icon: IconMapPin,

    options: [

      "Delhi", "Bangalore", "Mumbai", "Noida", "Hyderabad",

      "Lucknow", "Pune", "Chennai", "Kolkata", "Ahmedabad",

      "Jaipur", "Chandigarh", "Indore", "Surat", "Nagpur"

    ],

  },

  {

    title: "Experience",

    icon: IconBriefcase,

    options: [

      "Fresher", "1-2 Years", "2-3 Years", "3-4 Years", "4-5 Years",

      "5-6 Years", "6-7 Years", "7-8 Years", "8-9 Years",

      "9-10 Years", "10+ Years"

    ],

  },

  {

    title: "Job Type",

    icon: IconRecharging,

    options: ["Full Time", "Part Time", "Internship", "Contract", "Remote", "Temporary", "Freelance"],

  }

];

// Dropdown Component

const Dropdown: React.FC<{ item: DropdownItem }> = ({ item }) => {

  return (

    <div className="flex flex-col w-1/5 min-w-[150px]">

      <label className="text-sm font-semibold mb-1 flex items-center gap-2">

        {React.createElement(item.icon, { size: 18 })} {item.title}

      </label>

      <select className="border border-gray-600 p-2 rounded bg-gray-800 text-white">

        {item.options.map((option, index) => (

          <option key={index} value={option}>{option}</option>

        ))}

      </select>

    </div>

  );

};

// Search Bar Component

const SearchBar: React.FC = () => {

  const [salaryRange, setSalaryRange] = useState<[number, number]>([20, 80]);

  const handleSliderChange = (e: React.ChangeEvent<HTMLInputElement>, index: number) => {

    setSalaryRange((prev) => {

      const newValue: [number, number] = [...prev];

      newValue[index] = Number(e.target.value);

      return newValue;

    });

  };

  return (

    <div className="flex flex-col gap-4 p-6 bg-gray-900 shadow-xl rounded-lg border border-gray-700 text-white">

      {/\* Dropdowns in Single Line \*/}

      <div className="flex flex-wrap gap-4 justify-between w-full">

        {dropdownData.map((item, index) => (

          <Dropdown key={index} item={item} />

        ))}

      </div>

      {/\* Salary Range Slider \*/}

      <div className="flex flex-col w-full mt-4">

        <label className="text-sm font-semibold mb-2">Salary Range</label>

        <div className="flex items-center gap-2">

          <input

            type="range"

            min="0"

            max="100"

            value={salaryRange[0]}

            onChange={(e) => handleSliderChange(e, 0)}

            className="w-24 accent-blue-500"

          />

          <span>{salaryRange[0]}k</span>

          <input

            type="range"

            min="0"

            max="100"

            value={salaryRange[1]}

            onChange={(e) => handleSliderChange(e, 1)}

            className="w-24 accent-blue-500"

          />

          <span>{salaryRange[1]}k</span>

        </div>

      </div>

      {/\* Search Button \*/}

      <div className="w-full flex justify-center mt-4">

        <button className="bg-blue-600 hover:bg-blue-700 text-white font-semibold py-2 px-6 rounded-lg transition">

          Search Jobs

        </button>

      </div>

    </div>

  );

};

export default SearchBar;

import { useState } from "react";

import { Button, Combobox, useCombobox } from "@mantine/core";

import { IconAdjustmentsAlt } from "@tabler/icons-react";

const optionsList = [

  "Relevance",

  "Most Recent",

  "Salary (Low to High)",

  "Salary (High to Low)",

];

const Sort = () => {

  const [selectedItem, setSelectedItem] = useState<string>(optionsList[0]);

  const combobox = useCombobox({

    onDropdownClose: () => combobox.resetSelectedOption(),

  });

  return (

    <Combobox

      store={combobox}

      position="bottom-start"

      withArrow

      onOptionSubmit={(val) => {

        setSelectedItem(val);

        combobox.closeDropdown();

      }}

    >

      <Combobox.Target>

        <Button

          onClick={() => combobox.toggleDropdown()}

          variant="outline"

          className="flex items-center gap-2 border border-gray-300 p-2 rounded-lg w-full"

          aria-label="Sort Options"

        >

          {selectedItem} <IconAdjustmentsAlt className="text-yellow-500" stroke={1.75} />

        </Button>

      </Combobox.Target>

      <Combobox.Dropdown>

        <Combobox.Options>

          {optionsList.map((item) => (

            <Combobox.Option value={item} key={item}>

              {item}

            </Combobox.Option>

          ))}

        </Combobox.Options>

      </Combobox.Dropdown>

    </Combobox>

  );

};

export default Sort;

import React, { useState } from "react";

import {

  IconBriefcase,

  IconMapPin,

  IconRecharging,

  IconSearch,

  IconUserCircle,

} from "@tabler/icons-react";

// Define the type for dropdown data

export interface SearchItem {

  title: string;

  icon: React.ElementType;

  options: string[];

}

// Dropdown data

const SearchField: SearchItem[] = [

  {

    title: "Job Title",

    icon: IconSearch,

    options: [

      "Designer", "Developer", "Manager", "Tester", "Engineer",

      "Marketing Specialist", "Data Analyst", "Business Analyst",

      "Product Manager", "HR Manager", "Finance Manager",

      "Accountant", "Sales Manager", "Marketing Manager",

      "Customer Support", "Customer Success", "Customer Service",

      "Customer Care"

    ],

  },

  {

    title: "Location",

    icon: IconMapPin,

    options: [

      "Delhi", "Bangalore", "Mumbai", "Noida", "Hyderabad",

      "Lucknow", "Pune", "Chennai", "Kolkata", "Ahmedabad",

      "Jaipur", "Chandigarh", "Indore", "Surat", "Nagpur"

    ],

  },

  {

    title: "Skill",

    icon: IconRecharging,

    options: [

      "HTML", "CSS", "JavaScript", "TypeScript", "ReactJS",

      "Angular", "VueJS", "Svelte", "NodeJS", "ExpressJS",

      "Python", "Django", "Flask", "Ruby", "Ruby on Rails",

      "PHP", "Laravel", "Java", "Spring Boot", "Kotlin",

      "Swift", "Objective-C", "C", "C++", "C#",

      ".NET", "Go", "Rust", "SQL", "PostgreSQL",

      "MySQL", "MongoDB", "GraphQL", "Firebase", "AWS",

      "Azure", "Google Cloud", "Docker", "Kubernetes", "Terraform",

      "Jenkins", "Git", "GitHub", "GitLab", "Bash",

      "PowerShell", "MATLAB", "R", "Scala", "Perl"

    ],

  },

];

const MultiInput: React.FC<{ options: string[] }> = ({ options }) => {

  return (

    <select className="border border-gray-300 p-2 rounded-lg w-full bg-gray-900 text-white shadow-md focus:ring-2 focus:ring-yellow-400">

      {options.map((option, index) => (

        <option key={index} value={option} className="text-white">

          {option}

        </option>

      ))}

    </select>

  );

};

const SearchBar: React.FC = () => {

  const [salaryRange, setSalaryRange] = useState<[number, number]>([20, 80]);

  const handleSliderChange = (index: number, value: number) => {

    setSalaryRange((prev) => {

      const newRange = [...prev] as [number, number];

      newRange[index] = value;

      return newRange;

    });

  };

  return (

    <div className="p-6 bg-gradient-to-r from-gray-800 to-gray-900 text-white shadow-lg rounded-xl border border-gray-600 flex flex-wrap gap-4 backdrop-blur-lg justify-center items-center">

      {/\* Talent Name Field \*/}

      <div className="flex items-center gap-3 w-full sm:w-auto border px-3 py-2 rounded-lg bg-gray-700 shadow-md">

        <IconUserCircle size={24} className="text-yellow-400" />

        <input

          type="text"

          placeholder="Talent Name"

          className="bg-transparent outline-none w-full text-white placeholder-gray-400"

        />

      </div>

      {/\* Dropdown Fields \*/}

      {SearchField.map((item, index) => (

        <div key={index} className="flex flex-col w-full sm:w-auto">

          <label className="text-sm font-semibold mb-1 flex items-center gap-2">

            {React.createElement(item.icon, {

              size: 18,

              className: "text-yellow-400",

            })}

            {item.title}

          </label>

          <MultiInput options={item.options} />

        </div>

      ))}

      {/\* Salary Range \*/}

      <div className="flex flex-col w-full sm:w-auto">

        <label className="text-sm font-semibold mb-1">Salary Range</label>

        <div className="flex items-center gap-3">

          <input

            type="range"

            min="0"

            max="100"

            value={salaryRange[0]}

            onChange={(e) => handleSliderChange(0, Number(e.target.value))}

            className="w-24 accent-yellow-400"

          />

          <span className="text-yellow-400 font-medium">{salaryRange[0]}k</span>

          <input

            type="range"

            min="0"

            max="100"

            value={salaryRange[1]}

            onChange={(e) => handleSliderChange(1, Number(e.target.value))}

            className="w-24 accent-yellow-400"

          />

          <span className="text-yellow-400 font-medium">{salaryRange[1]}k</span>

        </div>

      </div>

    </div>

  );

};

export default SearchBar;

import React from "react";

import Sort from "../FindJob/Sort";

import TalentCard from "./TalentCard";

import talentsData from "./TalentData";

const Talent: React.FC = () => {

  return (

    <section className="px-5 py-5">

      <header className="flex justify-between items-center mt-5">

        <h2 className="text-2xl font-semibold">Talents</h2>

        <Sort />

      </header>

      <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6 mt-10">

        {talentsData.map((talent) => (

          <TalentCard key={talent.id} {...talent} />

        ))}

      </div>

    </section>

  );

};

export default Talent;

import React, { memo, useRef, useState } from "react";

import { useNavigate } from "react-router-dom";

import { DateInput, TimeInput } from "@mantine/dates";

import {

  IconHeart,

  IconMapPin,

  IconUser,

  IconMessage,

  IconCalendarMonth,

} from "@tabler/icons-react";

import { Divider, Modal } from "@mantine/core";

import { useDisclosure } from "@mantine/hooks";

export interface Talents {

  id: number;

  name: string;

  role: string;

  company: string;

  topSkill: string[];

  about: string;

  location: string;

  salary: string;

  posted?: boolean;

  invited?: boolean;

}

const TalentCard: React.FC<Talents> = memo(

  ({ id, name, role, company, topSkill, about, location, salary, posted, invited }) => {

    const navigate = useNavigate();

    const [showMessage, setShowMessage] = useState(false);

    const [message, setMessage] = useState("");

    const [isLiked, setIsLiked] = useState(false);

    const [opened, { open, close }] = useDisclosure(false);

    const [date, setDate] = useState<Date | null>(null);

    const timeRef = useRef<HTMLInputElement>(null);

    return (

      <div className="p-6 bg-white shadow-lg rounded-2xl border border-gray-200 flex flex-col gap-4 transition-all hover:shadow-2xl hover:scale-[1.02]">

        <div className="flex justify-between items-center">

          <div className="flex gap-4 items-center">

            <div className="p-3 bg-gray-100 rounded-full w-16 h-16 flex items-center justify-center text-xl font-semibold text-blue-600">

              {name.charAt(0)}

            </div>

            <div>

              <h3 className="text-xl font-semibold text-gray-900">{name}</h3>

              <p className="text-gray-600 text-sm">{role} at {company}</p>

            </div>

          </div>

          <button

            className="p-2 rounded-full hover:bg-gray-100 transition"

            onClick={() => setIsLiked(!isLiked)}

          >

            <IconHeart className={`transition-colors ${isLiked ? "text-red-500 fill-red-500" : "text-gray-500"}`} stroke={2} />

          </button>

        </div>

        <div className="flex flex-wrap gap-2">

          {topSkill.map((skill, index) => (

            <span key={index} className="px-3 py-1 bg-green-100 text-green-700 font-medium rounded-full text-xs">

              {skill}

            </span>

          ))}

        </div>

        <p className="text-gray-700 text-sm line-clamp-2">{about}</p>

        <Divider color="gray-500" size="xs"/>

        {invited ? (

          <div className="flex gap-1 items-center text-gray-700">

            <IconCalendarMonth stroke={1.5} /> Interview | April 5, 2025, 04:30 PM

          </div>

        ) : (

          <p className="text-gray-900 font-semibold">Salary: {salary}</p>

        )}

        <div className="flex items-center gap-1 text-gray-500 text-sm">

          <IconMapPin className="h-5 w-5" stroke={1.5} />

          {location}

        </div>

        <Divider color="gray" size="xs"/>

        <div className="flex gap-4 mt-3">

          {!invited ? (

            <>

              <button

                className="py-2 px-4 bg-gray-700 text-white rounded-lg font-medium text-sm hover:bg-gray-800 transition-all flex items-center"

                onClick={() => navigate("/talent-profile")}

              >

                <IconUser className="mr-1" /> Profile

              </button>

              {posted ? (

                <button onClick={open} className="py-2 px-4 bg-blue-600 text-white rounded-lg font-medium text-sm hover:bg-blue-700 transition-all flex items-center">

                  <IconCalendarMonth className="mr-1" stroke={2} /> Schedule

                </button>

              ) : (

                <button

                  className="py-2 px-4 bg-green-600 text-white rounded-lg font-medium text-sm hover:bg-green-700 transition-all flex items-center"

                  onClick={() => setShowMessage(!showMessage)}

                >

                  <IconMessage className="mr-1" /> Message

                </button>

              )}

            </>

          ) : (

            <div className="flex justify-center gap-4 w-full">

              <button className="py-2 px-6 bg-green-600 text-white rounded-lg font-medium text-sm hover:bg-green-700 transition-all">Accept</button>

              <button className="py-2 px-6 bg-red-600 text-white rounded-lg font-medium text-sm hover:bg-red-700 transition-all">Reject</button>

            </div>

          )}

        </div>

        {showMessage && (

          <div className="mt-4 p-4 bg-gray-100 rounded-lg">

            <h4 className="text-lg font-semibold">Send a Message</h4>

            <textarea

              className="w-full p-2 border border-gray-300 rounded-lg"

              placeholder="Type your message..."

              value={message}

              onChange={(e) => setMessage(e.target.value)}

            ></textarea>

            <div className="mt-2 flex gap-2">

              <button className="py-2 px-4 bg-blue-600 text-white rounded-lg font-medium text-sm hover:bg-blue-700 transition-all">Send</button>

              <button className="py-2 px-4 bg-red-600 text-white rounded-lg font-medium text-sm hover:bg-red-700 transition-all" onClick={() => setShowMessage(false)}>Cancel</button>

            </div>

          </div>

        )}

        <Modal opened={opened} onClose={close} title="Schedule Interview" centered>

          <div className="flex flex-col gap-3">

            <DateInput

              value={date}

              minDate={new Date()}

              onChange={setDate}

              label="Date"

              placeholder="Enter Date"

            />

            <TimeInput label="Time" ref={timeRef} onClick={() => timeRef.current?.showPicker()} />

            <button onClick={close} className="py-2 px-4 bg-blue-600 text-white rounded-lg font-medium text-sm hover:bg-blue-700 transition-all flex items-center">

              <IconCalendarMonth className="mr-1" stroke={2} /> Schedule

            </button>

          </div>

        </Modal>

      </div>

    );

  }

);

export default TalentCard;

import { ReactNode } from "react";

export interface Talent {

  [x: string]: ReactNode | number | string | string[] | Experience[] | Certification[] | undefined;

  id: number;

  name: string;

  role: string;

  company: string;

  topSkill: string[];

  about: string;

  location: string;

  salary: string;

  experience: Experience[];

  certifications?: Certification[];

}

interface Experience {

  currentlyWorking: any;

  title: string;

  company: string;

  location: string;

  startDate: string;

  endDate: string;

  description: string;

}

interface Certification {

  title: string;

  company: string;

  location: string;

  issueDate: string;

  certificateID: string;

}

const talentsData: Talent[] = [

  {

    id: 1,

    name: "Arjun Mehta",

    role: "Frontend Developer",

    company: "Tech Solutions",

    topSkill: ["React", "JavaScript", "CSS"],

    about: "Arjun is a skilled frontend developer with a passion for creating intuitive and responsive user interfaces.",

    location: "Bangalore, India",

    salary: "10 - 12 LPA",

    experience: [

      {

        title: "Software Engineer",

        company: "Google",

        location: "Hyderabad, India",

        startDate: "Jan 2023",

        endDate: "Present",

        description: "Developing scalable frontend solutions using React and TypeScript.",

        currentlyWorking: undefined

      },

    ],

    certifications: [

      {

        title: "React Developer",

        company: "Coursera",

        location: "India",

        issueDate: "Mar 2023",

        certificateID: "React123",

      },

    ],

  },

  {

    id: 2,

    name: "Priya Sharma",

    role: "Backend Developer",

    company: "Innovatech",

    topSkill: ["Node.js", "Express", "MongoDB"],

    about: "Priya specializes in building robust backend systems and APIs for scalable applications.",

    location: "Mumbai, India",

    salary: "12 - 14 LPA",

    experience: [

      {

        title: "Backend Developer",

        company: "Amazon",

        location: "Pune, India",

        startDate: "Jun 2019",

        endDate: "Feb 2021",

        description: "Built and optimized backend services for Amazon's e-commerce platform, focusing on scalability and reliability.",

        currentlyWorking: undefined

      },

    ],

    certifications: [

      {

        title: "Node.js Specialist",

        company: "Udemy",

        location: "India",

        issueDate: "Jan 2022",

        certificateID: "Node456",

      },

    ],

  },

  {

    id: 3,

    name: "Ravi Patel",

    role: "UI/UX Designer",

    company: "Design Hub",

    topSkill: ["Figma", "Adobe XD", "Prototyping"],

    about: "Ravi is a creative designer focused on delivering user-friendly and visually appealing designs.",

    location: "Delhi, India",

    salary: "8 - 10 LPA",

    experience: [

      {

        title: "Frontend Developer",

        company: "Facebook",

        location: "Bangalore, India",

        startDate: "Mar 2021",

        endDate: "Dec 2022",

        description: "Developed and maintained user-facing features for Facebook's web applications, ensuring seamless user experiences and high performance.",

        currentlyWorking: undefined

      },

    ],

    certifications: [

      {

        title: "UI/UX Design Expert",

        company: "LinkedIn Learning",

        location: "India",

        issueDate: "Feb 2023",

        certificateID: "UIUX789",

      },

    ],

  },

  {

    id: 4,

    name: "Ananya Iyer",

    role: "DevOps Engineer",

    company: "CloudWorks",

    topSkill: ["AWS", "Docker", "Kubernetes"],

    about: "Ananya ensures smooth deployment pipelines and manages cloud infrastructure efficiently.",

    location: "Hyderabad, India",

    salary: "15 - 18 LPA",

    experience: [

      {

        title: "Backend Developer",

        company: "Amazon",

        location: "Chennai, India",

        startDate: "Jun 2019",

        endDate: "Feb 2021",

        description: "Built and optimized backend services for Amazon's e-commerce platform, focusing on scalability and reliability.",

        currentlyWorking: undefined

      },

    ],

    certifications: [

      {

        title: "AWS Certified Solutions Architect",

        company: "AWS",

        location: "India",

        issueDate: "May 2023",

        certificateID: "AWS123",

      },

    ],

  },

  {

    id: 5,

    name: "Karan Malhotra",

    role: "Data Scientist",

    company: "Data Insights",

    topSkill: ["Python", "R", "Machine Learning"],

    about: "Karan is an expert in analyzing data and building predictive models to drive business decisions.",

    location: "Pune, India",

    salary: "18 - 20 LPA",

    experience: [

      {

        title: "Data Scientist",

        company: "Microsoft",

        location: "Mumbai, India",

        startDate: "Jan 2018",

        endDate: "May 2019",

        description: "Analyzed large datasets to derive actionable insights and developed predictive models to enhance business decision-making.",

        currentlyWorking: undefined

      },

    ],

    certifications: [

      {

        title: "Machine Learning Specialist",

        company: "edX",

        location: "India",

        issueDate: "Jun 2022",

        certificateID: "ML456",

      },

    ],

  },

  {

    id: 6,

    name: "Neha Kapoor",

    role: "Full Stack Developer",

    company: "CodeCraft",

    topSkill: ["React", "Node.js", "PostgreSQL"],

    about: "Neha is a versatile developer proficient in both frontend and backend technologies.",

    location: "Chennai, India",

    salary: "12 - 14 LPA",

    experience: [

      {

        title: "UI/UX Designer",

        company: "Adobe",

        location: "Delhi, India",

        startDate: "Aug 2016",

        endDate: "Dec 2017",

        description: "Designed intuitive user interfaces and improved user experiences for Adobe's suite of creative tools.",

        currentlyWorking: undefined

      },

    ],

    certifications: [

      {

        title: "Full Stack Developer",

        company: "Pluralsight",

        location: "India",

        issueDate: "Apr 2023",

        certificateID: "FullStack789",

      },

    ],

  },

  {

    id: 7,

    name: "Vikram Singh",

    role: "Quality Assurance Engineer",

    company: "Testify",

    topSkill: ["Selenium", "Cypress", "Manual Testing"],

    about: "Vikram ensures the delivery of high-quality software through rigorous testing processes.",

    location: "Kolkata, India",

    salary: "8 - 10 LPA",

    experience: [

      {

        title: "DevOps Engineer",

        company: "Netflix",

        location: "Bangalore, India",

        startDate: "Feb 2015",

        endDate: "Jul 2016",

        description: "Implemented CI/CD pipelines and managed cloud infrastructure to ensure smooth deployment of Netflix's streaming services.",

        currentlyWorking: undefined

      },

    ],

    certifications: [

      {

        title: "Certified QA Engineer",

        company: "ISTQB",

        location: "India",

        issueDate: "Dec 2022",

        certificateID: "QA123",

      },

    ],

  },

  {

    id: 8,

    name: "Aditi Nair",

    role: "Cybersecurity Analyst",

    company: "SecureNet",

    topSkill: ["Ethical Hacking", "Network Security", "Firewalls"],

    about: "Aditi is dedicated to protecting systems and networks from cyber threats.",

    location: "Ahmedabad, India",

    salary: "14 - 16 LPA",

    experience: [

      {

        title: "Mobile App Developer",

        company: "Apple",

        location: "Pune, India",

        startDate: "May 2013",

        endDate: "Jan 2015",

        description: "Developed high-performance mobile applications for iOS, focusing on usability and seamless integration with Apple ecosystems.",

        currentlyWorking: undefined

      },

    ],

    certifications: [

      {

        title: "Certified Ethical Hacker",

        company: "EC-Council",

        location: "India",

        issueDate: "Jul 2023",

        certificateID: "CEH123",

      },

    ],

  },

  {

    id: 9,

    name: "Rohan Desai",

    role: "Mobile App Developer",

    company: "Appify",

    topSkill: ["Flutter", "React Native", "Swift"],

    about: "Rohan develops cross-platform mobile applications with a focus on performance and usability.",

    location: "Jaipur, India",

    salary: "10 - 12 LPA",

    experience: [

      {

        title: "Cybersecurity Analyst",

        company: "IBM",

        location: "Ahmedabad, India",

        startDate: "Nov 2011",

        endDate: "Apr 2013",

        description: "Monitored and secured IBM's network infrastructure, identifying and mitigating potential cyber threats.",

        currentlyWorking: undefined

      },

    ],

    certifications: [

      {

        title: "Mobile App Developer",

        company: "Google",

        location: "India",

        issueDate: "Aug 2023",

        certificateID: "MobileDev123",

      },

    ],

  },

  {

    id: 10,

    name: "Meera Joshi",

    role: "AI Engineer",

    company: "NeuralTech",

    topSkill: ["TensorFlow", "PyTorch", "NLP"],

    about: "Meera works on cutting-edge AI solutions, including natural language processing and computer vision.",

    location: "Lucknow, India",

    salary: "20 - 25 LPA",

    experience: [

      {

        title: "Quality Assurance Engineer",

        company: "Intel",

        location: "Chennai, India",

        startDate: "Jul 2009",

        endDate: "Oct 2011",

        description: "Conducted rigorous testing of Intel's software and hardware products to ensure quality and reliability.",

        currentlyWorking: undefined

      },

    ],

    certifications: [

      {

        title: "AI Specialist",

        company: "DeepLearning.AI",

        location: "India",

        issueDate: "Sep 2023",

        certificateID: "AI123",

      },

    ],

  },

  {

    id: 11,

    name: "Sanjay Verma",

    role: "Cloud Architect",

    company: "CloudSphere",

    topSkill: ["Azure", "GCP", "Infrastructure Design"],

    about: "Sanjay designs and implements scalable cloud architectures for enterprise applications.",

    location: "Surat, India",

    salary: "22 - 25 LPA",

    experience: [

      {

        title: "Cloud Architect",

        company: "Oracle",

        location: "Mumbai, India",

        startDate: "Jan 2007",

        endDate: "Jun 2009",

        description: "Designed and implemented scalable cloud solutions for enterprise clients, leveraging Oracle's cloud technologies.",

        currentlyWorking: undefined

      },

    ],

    certifications: [

      {

        title: "Cloud Architect Professional",

        company: "Google Cloud",

        location: "India",

        issueDate: "Oct 2023",

        certificateID: "Cloud123",

      },

    ],

  },

];

export default talentsData;

import React, { useState } from "react";

import { motion } from "framer-motion";

import { useLocation } from "react-router-dom";

import {

  IconBrandFacebook,

  IconBrandInstagram,

  IconBrandLinkedin,

  IconBrandX,

  IconBrandYoutube,

  IconMenu2,

} from "@tabler/icons-react";

const footerLinks = [

  {

    title: "Company",

    links: ["About Us", "Careers", "Contact", "Blog"],

  },

  {

    title: "Resources",

    links: ["Help Center", "Privacy Policy", "Terms of Service"],

  },

  {

    title: "Support",

    links: ["FAQs", "Customer Support", "Feedback"],

  },

];

const SocialIcons = () => {

  const location = useLocation();

  if (location.pathname === "/signup") return null;

  const icons = [

    { icon: IconBrandFacebook, link: "#", color: "bg-blue-600" },

    { icon: IconBrandInstagram, link: "#", color: "bg-pink-500" },

    { icon: IconBrandX, link: "#", color: "bg-gray-900" },

    { icon: IconBrandLinkedin, link: "#", color: "bg-blue-700" },

    { icon: IconBrandYoutube, link: "#", color: "bg-red-600" },

  ];

  return (

    <div className="flex gap-4 mt-4">

      {icons.map((item, index) => (

        <motion.a

          key={index}

          href={item.link}

          whileHover={{ scale: 1.2 }}

          whileTap={{ scale: 0.9 }}

          className={`${item.color} p-3 rounded-full text-white transition shadow-md hover:shadow-lg`}

        >

          <item.icon size={22} stroke={2} />

        </motion.a>

      ))}

    </div>

  );

};

const FooterLinks = () => (

  <motion.div

    className="flex flex-wrap justify-center md:justify-end gap-10 mt-6 md:mt-0"

    initial={{ opacity: 0, y: 20 }}

    animate={{ opacity: 1, y: 0 }}

    transition={{ duration: 0.8 }}

  >

    {footerLinks.map((section, index) => (

      <div key={index} className="flex flex-col gap-2 text-center md:text-left">

        <h2 className="text-lg font-semibold text-gray-200">{section.title}</h2>

        {section.links.map((link, linkIndex) => (

          <a

            key={linkIndex}

            href="#"

            className="text-gray-400 hover:text-gray-200 transition-transform hover:translate-x-1"

          >

            {link}

          </a>

        ))}

      </div>

    ))}

  </motion.div>

);

const Footer = () => {

  const location = useLocation();

  const [isMobileMenuOpen, setIsMobileMenuOpen] = useState(false);

  if (location.pathname === "/signup" || location.pathname === "/login") return null;

  return (

    <motion.footer

      className="bg-gray-900 text-white py-10 px-6 md:px-16 flex flex-col md:flex-row items-center md:items-start justify-between shadow-lg"

      initial={{ opacity: 0 }}

      animate={{ opacity: 1 }}

      transition={{ duration: 0.8 }}

    >

      <div className="w-full md:w-1/3 flex flex-col items-center md:items-start gap-6">

        <div className="flex items-center gap-4">

          <button

            className="md:hidden"

            onClick={() => setIsMobileMenuOpen(!isMobileMenuOpen)}

          >

            <IconMenu2 stroke={2} className="h-8 w-8 text-white hover:text-blue-400 transition-transform hover:rotate-90" />

          </button>

          <div className="bg-white/10 p-3 rounded-full flex justify-center items-center shadow-md border border-white/20">

            <img

              src="/career.jpeg"

              alt="CareerSync Logo"

              className="h-12 w-12 object-cover rounded-full"

            />

          </div>

          <h1 className="text-2xl font-bold text-transparent bg-clip-text bg-gradient-to-r from-blue-400 to-green-300 drop-shadow-lg">

            CAREERSYNC

          </h1>

        </div>

        <p className="text-gray-300 text-sm text-center md:text-left leading-relaxed">

          Job portal with user profiles, skill updates, certifications, work experience, and admin job postings.

        </p>

        <SocialIcons />

      </div>

      <FooterLinks />

    </motion.footer>

  );

};

export default Footer;

// React core & hooks

import { useState, useEffect, useRef } from "react";

// For navigation

import { Link, useLocation } from "react-router-dom";

// Icons

import { IconBell, IconSettings, IconMenu2 } from "@tabler/icons-react";

// Animation library

import { motion } from "framer-motion";

// Import Profile Menu component

import ProfileMenu from "./ProfileMenu";

// Redux for accessing global state

import { useSelector } from "react-redux";

// Button component (missing in original)

import { Button } from "@mantine/core";

// Navigation menu items (SignUp removed)

const navItems = [

  { name: "Find Job", path: "/find-job" },

  { name: "Find Talent", path: "/find-talent" },

  { name: "Post Job", path: "/post-job" },

  { name: "Posted Job", path: "/posted-job" },

  { name: "Job History", path: "/job-history" },

];

function Header() {

  // State variables for toggles

  const [showSettings, setShowSettings] = useState(false);

  const [showNotifications, setShowNotifications] = useState(false);

  const [isMobileMenuOpen, setIsMobileMenuOpen] = useState(false);

  // Refs to detect outside clicks

  const settingsRef = useRef<HTMLDivElement | null>(null);

  const notificationsRef = useRef<HTMLDivElement | null>(null);

  const mobileMenuRef = useRef<HTMLDivElement | null>(null);

  const location = useLocation();

  const user = useSelector((state: any) => state.user); // Get logged-in user

  // Close dropdowns when clicked outside

  useEffect(() => {

    const handleClickOutside = (event: MouseEvent) => {

      if (settingsRef.current && !settingsRef.current.contains(event.target as Node))

        setShowSettings(false);

      if (notificationsRef.current && !notificationsRef.current.contains(event.target as Node))

        setShowNotifications(false);

      if (mobileMenuRef.current && !mobileMenuRef.current.contains(event.target as Node))

        setIsMobileMenuOpen(false);

    };

    document.addEventListener("mousedown", handleClickOutside);

    return () => document.removeEventListener("mousedown", handleClickOutside);

  }, []);

  // Hide header on signup/signin pages

  if (location.pathname === "/signup" || location.pathname === "/signin") return null;

  return (

    <header className="w-full h-20 bg-gray-900 text-white flex justify-between items-center px-6 md:px-12 shadow-lg border-b border-gray-800 relative">

      {/\* Logo & Menu Button \*/}

      <div className="flex items-center gap-4">

        <button className="md:hidden" onClick={() => setIsMobileMenuOpen(!isMobileMenuOpen)}>

          <IconMenu2 className="h-8 w-8 text-white hover:text-blue-400" />

        </button>

        <div className="bg-white/10 p-2 rounded-full shadow-md border border-white/20">

          <img src="/career.jpeg" alt="Career Logo" className="h-8 w-8 object-cover rounded-full" />

        </div>

        <h1 className="text-2xl font-extrabold text-transparent bg-clip-text bg-gradient-to-r from-blue-400 to-green-300 drop-shadow-lg">

          CAREERSYNC

        </h1>

      </div>

      {/\* Desktop Navigation Menu \*/}

      <nav className="hidden md:flex gap-4">

        {navItems.map(({ name, path }) => (

          <motion.div key={name} whileHover={{ scale: 1.1 }} whileTap={{ scale: 0.95 }}>

            <Link

              to={path}

              className="relative px-6 py-3 font-semibold rounded-lg transition-all duration-300 shadow-md border text-gray-300 bg-gray-800 hover:bg-gray-700 hover:scale-105 border-transparent"

            >

              {name}

            </Link>

          </motion.div>

        ))}

      </nav>

      {/\* Right Section: User Menu, Settings, Notifications \*/}

      <div className="flex gap-6 items-center relative">

        {/\* Conditional render based on login \*/}

        {user ? (

          <ProfileMenu />

        ) : (

          <Link to="/login" className="text-gray-300 hover:text-blue-400">

            <Button variant="subtle" color="yellow">Login</Button>

          </Link>

        )}

        {/\* Settings Dropdown \*/}

        <div className="relative" ref={settingsRef}>

          <button onClick={() => setShowSettings(!showSettings)} className="relative hover:rotate-12 transition-transform duration-300">

            <IconSettings className="h-8 w-8 text-white hover:text-blue-400" />

          </button>

          {showSettings && (

            <div className="absolute right-0 mt-2 w-48 bg-gray-800 backdrop-blur-md text-white text-sm rounded-lg p-4 shadow-lg border border-gray-700">

              <p className="hover:text-blue-400 cursor-pointer py-2">Profile Settings</p>

              <p className="hover:text-blue-400 cursor-pointer py-2">Privacy & Security</p>

              <p className="hover:text-blue-400 cursor-pointer py-2">Logout</p>

            </div>

          )}

        </div>

        {/\* Notifications Dropdown \*/}

        <div className="relative" ref={notificationsRef}>

          <button onClick={() => setShowNotifications(!showNotifications)} className="relative hover:scale-110 transition-all duration-300">

            <IconBell className="h-8 w-8 text-white hover:text-blue-400" />

          </button>

          {showNotifications && (

            <div className="absolute right-0 mt-2 w-52 bg-gray-800 backdrop-blur-md text-white text-sm rounded-lg p-4 shadow-lg border border-gray-700">

              <p className="hover:text-blue-400 cursor-pointer py-2">New Job Alerts</p>

              <p className="hover:text-blue-400 cursor-pointer py-2">Messages</p>

              <p className="hover:text-blue-400 cursor-pointer py-2">Application Updates</p>

            </div>

          )}

        </div>

      </div>

      {/\* Mobile Menu \*/}

      {isMobileMenuOpen && (

        <div ref={mobileMenuRef} className="absolute top-20 left-0 w-full bg-gray-900 text-white p-6 shadow-xl md:hidden">

          {navItems.map(({ name, path }) => (

            <Link key={name} to={path} className="block py-2 text-gray-300 hover:text-blue-400">{name}</Link>

          ))}

        </div>

      )}

    </header>

  );

}

export default Header;

import React from "react";

import { Link, useLocation } from "react-router-dom";

import { motion } from "framer-motion";

import clsx from "clsx";

interface NavItemProps {

  name: string;

  path: string;

  isActive: boolean;

}

const navItems = [

  { name: "Find Job", path: "/find-job" },

  { name: "Find Talent", path: "/find-talent" },

  { name: "Post Job", path: "/post-job" },

  { name: "Posted Job", path: "/posted-job" },

  { name: "Job History", path: "/job-history" },

];

const NavItem: React.FC<NavItemProps> = ({ name, path, isActive }) => (

  <motion.div whileHover={{ scale: 1.1 }} whileTap={{ scale: 0.95 }} className="relative">

    <Link

      to={path}

      aria-current={isActive ? "page" : undefined}

      className={clsx(

        "relative px-6 py-3 font-medium rounded-xl transition-all duration-300 border text-sm md:text-base flex items-center justify-center",

        isActive

          ? "text-white bg-gradient-to-r from-blue-500 to-purple-600 border-blue-500 shadow-lg"

          : "text-gray-300 bg-gray-800 hover:bg-gray-700 hover:border-gray-500 border-gray-700"

      )}

    >

      {name}

    </Link>

    {isActive && (

      <motion.span

        layoutId="underline"

        className="absolute -bottom-1 left-1/2 transform -translate-x-1/2 w-3/4 h-[3px] bg-blue-500 rounded-full"

      />

    )}

  </motion.div>

);

const NavLinks: React.FC = () => {

  const location = useLocation();

  return (

    <nav className="w-full max-w-5xl mx-auto flex flex-wrap justify-center items-center gap-6 p-4 bg-gray-900 bg-opacity-95 backdrop-blur-md shadow-xl rounded-3xl border border-gray-700">

      {navItems.map(({ name, path }) => (

        <NavItem key={name} name={name} path={path} isActive={location.pathname === path} />

      ))}

    </nav>

  );

};

export default NavLinks;

// React imports

import { useState, useEffect } from 'react';

// Mantine UI components

import { Menu, Button, Switch } from '@mantine/core';

// Tabler icons

import {

  IconSettings, IconMessageCircle, IconFileCv, IconMoon, IconSun,

  IconMoonStars, IconLogout2, IconArrowsLeftRight, IconUserCircle

} from '@tabler/icons-react';

// Routing

import { Link } from 'react-router-dom';

// Redux

import { useDispatch, useSelector } from 'react-redux';

import { removeUser } from '../../Slices/UserSlice';

function ProfileMenu() {

  // Get dark mode value from localStorage

  const [darkMode, setDarkMode] = useState(() => localStorage.getItem('darkMode') === 'true');

  const [opened, setOpened] = useState(false);

  const user = useSelector((state: any) => state.user);

  const dispatch = useDispatch();

  // Logout handler

  const handleLogout = () => {

    dispatch(removeUser());

  };

  // Set dark mode class to html root

  useEffect(() => {

    document.documentElement.classList.toggle('dark', darkMode);

    localStorage.setItem('darkMode', JSON.stringify(darkMode));

  }, [darkMode]);

  return (

    <Menu shadow="md" width={200} opened={opened} onOpen={() => setOpened(true)} onClose={() => setOpened(false)}>

      {/\* Profile trigger section \*/}

      <Menu.Target>

        <div className="flex items-center gap-3 cursor-pointer transition-transform duration-300 hover:scale-105">

          <span className="text-lg font-semibold text-gray-300">{user.name}</span>

          <img src="https://randomuser.me/api/portraits/men/75.jpg" alt="User Avatar" className="h-12 w-12 rounded-full border-2 border-blue-400 shadow-md" />

        </div>

      </Menu.Target>

      {/\* Dropdown content \*/}

      <Menu.Dropdown>

        <Link to="/Profile">

          <Menu.Item leftSection={<IconUserCircle size={14} />}>Profile</Menu.Item>

        </Link>

        <Menu.Item leftSection={<IconMessageCircle size={14} />}>Messages</Menu.Item>

        <Menu.Item leftSection={<IconFileCv size={14} />}>Resume</Menu.Item>

        {/\* Dark mode toggle \*/}

        <Menu.Item

          leftSection={<IconMoon stroke={2} size={14} />}

          rightSection={

            <Switch

              checked={darkMode}

              onChange={(e) => setDarkMode(e.currentTarget.checked)}

              size="md"

              color={darkMode ? 'yellow' : 'blue'}

              onLabel={<IconSun size={16} stroke={2.5} color="var(--mantine-color-yellow-4)" />}

              offLabel={<IconMoonStars size={16} stroke={2.5} color="var(--mantine-color-blue-6)" />}

            />

          }

        >

          Dark Mode

        </Menu.Item>

        <Menu.Divider />

        <Menu.Label>Danger Zone</Menu.Label>

        <Menu.Item leftSection={<IconArrowsLeftRight size={14} />}>Transfer my data</Menu.Item>

        {/\* Logout item \*/}

        <Menu.Item onClick={handleLogout} color="red" leftSection={<IconLogout2 stroke={2} size={14} />}>

          Logout

        </Menu.Item>

      </Menu.Dropdown>

    </Menu>

  );

}

export default ProfileMenu;

import React from "react";

import { Divider } from "@mantine/core";

import { IconBookmark } from "@tabler/icons-react";

import { Link, useNavigate } from "react-router-dom";

import { card, jobDescription, qualification, responsibilities, skills } from "../../Data/CardData";

interface JobDeskProps {

    edit?: boolean;

}

const JobDesk: React.FC<JobDeskProps> = ({ edit = false }) => {

    const navigate = useNavigate();

    const handleApplyClick = () => {

        navigate("/apply-job");

    };

    return (

        <div className="p-8 bg-gray-900 shadow-2xl rounded-3xl text-white space-y-10">

            {/\* Job Header \*/}

            <header className="text-center space-y-4">

                <div className="flex justify-center">

                    <img

                        src="https://upload.wikimedia.org/wikipedia/commons/2/2f/Google\_2015\_logo.svg"

                        alt="Company Logo"

                        className="w-20 h-20 object-contain"

                    />

                </div>

                <h1 className="text-4xl font-extrabold text-blue-500">Software Engineer III</h1>

                <div className="flex justify-center space-x-6 text-sm text-gray-400">

                    <span className="text-yellow-400 font-semibold">Google</span>

                    <span className="text-green-400 font-semibold">3 days ago</span>

                    <span className="text-pink-400 font-semibold">48 Applicants</span>

                </div>

            </header>

            {/\* Apply Section \*/}

            <div className="flex justify-center gap-6">

                <button

                    className="px-6 py-3 bg-blue-600 text-white font-bold rounded-xl hover:bg-blue-700 transition-all shadow-lg"

                    onClick={handleApplyClick}

                >

                   {edit ? "Edit" : "Apply Now"}

                </button>

                {edit && (

                    <button

                        className="px-6 py-3 bg-red-600 text-white font-bold rounded-xl hover:bg-red-700 transition-all shadow-lg"

                    >

                        Delete

                    </button>

                )}

                <IconBookmark

                    className="cursor-pointer text-gray-400 hover:text-blue-400 transition-all"

                    stroke={2}

                    size={30}

                />

            </div>

            <Divider color="gray" />

            {/\* Job Details \*/}

            <section>

                <h2 className="text-2xl font-semibold text-purple-400 mb-4">Job Details</h2>

                <div className="grid grid-cols-2 md:grid-cols-4 gap-6 bg-gray-800 p-6 rounded-xl text-white text-sm shadow-md">

                    {card.map((item, index) => (

                        <div key={index} className="flex items-center gap-3">

                            <item.icon className="text-yellow-400" stroke={1.5} size={24} />

                            <div>

                                <div className="text-yellow-400 text-sm font-semibold">{item.name}</div>

                                <div className="text-green-400 font-bold text-base">{item.value}</div>

                            </div>

                        </div>

                    ))}

                </div>

            </section>

            <Divider color="gray" />

            {/\* Required Skills \*/}

            <section>

                <h2 className="text-2xl font-semibold text-purple-400 mb-4">Required Skills</h2>

                <div className="flex flex-wrap gap-3">

                    {skills.map((skill, index) => (

                        <span key={index} className="px-5 py-2 bg-gray-800 text-yellow-400 font-semibold rounded-full text-sm shadow-md">

                            {skill}

                        </span>

                    ))}

                </div>

            </section>

            <Divider color="gray" />

            {/\* Additional Sections \*/}

            {[

                { title: "About the Job", content: jobDescription },

                { title: "Responsibilities", content: responsibilities },

                { title: "Qualifications", content: qualification }

            ].map((section, index) => (

                <section key={index}>

                    <h2 className="text-2xl font-semibold text-purple-400 mb-4">{section.title}</h2>

                    <div className="text-gray-300 space-y-3 leading-relaxed" dangerouslySetInnerHTML={{ \_\_html: section.content }} />

                </section>

            ))}

            <Divider color="gray" />

            {/\* Company Section \*/}

            <footer className="bg-gray-800 p-6 rounded-xl shadow-lg flex flex-col md:flex-row items-center md:justify-between space-y-6 md:space-y-0">

                <div className="flex items-center gap-5">

                    <img src="Company/google.png" alt="Company Logo" className="w-20 h-20 rounded-full shadow-lg border-4 border-gray-700" />

                    <div>

                        <h1 className="text-2xl font-bold text-blue-500">Google</h1>

                        <p className="text-gray-400 text-sm">Innovating the Future</p>

                    </div>

                </div>

                <div className="text-center">

                    <h3 className="text-lg font-semibold text-yellow-400">10k+ Employees</h3>

                    <p className="text-gray-400 text-sm">Global Presence</p>

                </div>

                <Link to="/company">

                    <button className="px-6 py-3 bg-blue-600 text-white font-bold rounded-xl hover:bg-blue-700 transition-all shadow-lg">

                        Visit Company Page

                    </button>

                </Link>

            </footer>

        </div>

    );

};

export default JobDesk;

import React from "react";

import JobCard from "../FindJob/JobCard";import { JSX } from "react/jsx-runtime";

import { jobList } from "../../Data/JobData";

interface Job {

  id: string;

  jobTitle: string;

  companyName: string;

  applicants: number;

  experience: string;

  jobType: string;

  location: string;

  salary: string;

  postDayAgo: number;

  description: string;

}

interface RecommendedJobProps {

  currentIndex: number;

  jobsPerPage: number;

  jobs?: Job[];

}

const RecommendedJob: React.FC<RecommendedJobProps> = ({ currentIndex, jobsPerPage, jobs = jobList }) => {

  const visibleJobs = jobs.slice(currentIndex, currentIndex + jobsPerPage);

  return (

    <div className="w-full flex flex-col gap-4">

      {visibleJobs.length > 0 ? (

        visibleJobs.map((job: JSX.IntrinsicAttributes & Job) => <JobCard key={job.id} {...job} />)

      ) : (

        <p className="text-gray-400">No more recommended jobs available.</p>

      )}

    </div>

  );

};

export default RecommendedJob;

import React, { memo, useState } from "react";

import { IconBookmark, IconCalendarMonth, IconClock } from "@tabler/icons-react";

import { Button, Divider } from "@mantine/core";

export interface Job {

  id: string;

  jobTitle: string;

  companyName: string;

  applicants: number;

  experience: string;

  jobType: string;

  location: string;

  salary: string;

  postDayAgo: number;

  description: string;

  applied?: boolean;

  offered?: boolean;

  interviewing?: boolean;

  saved?: boolean;

}

const generateCompanyLogoSVG = (companyName: string): string => {

  const initials = companyName

    .split(" ")

    .map((word) => word[0]?.toUpperCase())

    .join("");

  const svg = `

    <svg xmlns="http://www.w3.org/2000/svg" width="100" height="100">

      <rect width="100%" height="100%" fill="#F59E0B" rx="15"/>

      <text x="50%" y="55%" font-size="32" font-family="Arial" font-weight="bold" fill="#FFFFFF" text-anchor="middle">

        ${initials}

      </text>

    </svg>

  `;

  return `data:image/svg+xml;base64,${btoa(svg)}`;

};

const Card: React.FC<Job> = memo(({

  jobTitle, companyName, applicants, experience, jobType, location, salary, postDayAgo, description, applied, offered, interviewing, saved

}) => {

  const [isSaved, setIsSaved] = useState(saved || false);

  return (

    <div className="p-6 bg-gray-900 text-gray-100 shadow-xl rounded-2xl border border-gray-700 flex flex-col gap-5 transition-all hover:shadow-2xl hover:scale-105">

      <div className="flex justify-between items-center">

        <div className="flex gap-4 items-center">

          <div className="p-3 bg-gray-700 rounded-xl">

            <img

              className="h-16 w-16 object-contain rounded-lg"

              src={generateCompanyLogoSVG(companyName)}

              alt={`${companyName} logo`}

            />

          </div>

          <div>

            <h3 className="text-xl font-semibold text-blue-400">{jobTitle}</h3>

            <p className="text-lg font-bold bg-gradient-to-r from-yellow-400 to-pink-400 bg-clip-text text-transparent">

              {companyName}

            </p>

            <p className="text-gray-400 text-sm">{applicants} Applicants</p>

          </div>

        </div>

        <button

          className="p-2 rounded-full hover:bg-gray-700 transition"

          onClick={() => setIsSaved((prev) => !prev)}

        >

          <IconBookmark

            className={`transition-colors ${isSaved ? "text-yellow-400" : "text-gray-400"}`}

            stroke={2}

            fill={isSaved ? "#FACC15" : "none"}

          />

        </button>

      </div>

      <div className="flex flex-wrap gap-2">

        {[experience, jobType, location].map((item, index) => (

          <span

            key={index}

            className="px-3 py-1 bg-indigo-500 text-white font-medium rounded-full text-xs"

          >

            {item}

          </span>

        ))}

      </div>

      <p className="text-gray-300 text-sm line-clamp-2">{description}</p>

      <div className="flex justify-between items-center mt-3">

        <span className="text-lg font-bold text-green-400">

          {salary.includes("LPA") ? `₹ ${salary}` : salary}

        </span>

        <div className="flex items-center gap-1 text-gray-400 text-sm">

          <IconClock className="h-5 w-5" stroke={1.5} />

          {applied || interviewing? "Applied" : offered ? "Offered" : "Posted"} {postDayAgo} days ago

        </div>

      </div>

      {offered && (

        <div className="flex gap-3 mt-3">

          <Divider size="xs" />

          <Button variant="filled" color="green">Accept</Button>

          <Button variant="outline" color="red">Reject</Button>

        </div>

      )}

      {interviewing && (

        <div className="flex gap-1 items-center text-gray-400 mt-2">

          <IconCalendarMonth className="text-yellow-400 w-5 h-5" stroke={1.5} />

          <span>Sat April 5, 2025 • <span className="text-lime-500">04:30 PM</span></span>

        </div>

      )}

    </div>

  );

});

export default Card;

import React from "react";

import { Tabs } from "@mantine/core";

import Card from "./Card";

import { jobList } from "../../Data/JobData";

function JobHistory() {

  return (

    <div>

      <div className="text-2xl font-semibold mb-5">Job History</div>

      <div>

        <Tabs variant="outline" defaultValue="about" className="mt-4">

          <Tabs.List className="flex justify-center space-x-4">

            <Tabs.Tab value="applied" className="text-gray-300 font-medium">

              Applied

            </Tabs.Tab>

            <Tabs.Tab value="saved" className="text-gray-300 font-medium">

              Saved

            </Tabs.Tab>

            <Tabs.Tab value="offered" className="text-gray-300 font-medium">

              Offered

            </Tabs.Tab>

            <Tabs.Tab value="interviewing" className="text-gray-300 font-medium">

              Interviewing

            </Tabs.Tab>

          </Tabs.List>

          <Tabs.Panel value="applied" className="p-4 text-gray-300 text-md leading-relaxed mb-5">

            <div className="grid grid-cols-2 gap-4">

              {jobList.map((job, index) => (

                <Card key={index} {...job} applied={true} />

              ))}

            </div>

          </Tabs.Panel>

          <Tabs.Panel value="saved" className="p-4 text-gray-300 text-md">

            <div className="grid grid-cols-2 gap-4">

              {jobList.map((job, index) => (

                <Card key={index} {...job} saved={true} />

              ))}

            </div>

          </Tabs.Panel>

          <Tabs.Panel value="offered" className="p-4 text-gray-300 text-md">

            <div className="grid grid-cols-2 gap-4">

              {jobList.map((job, index) => (

                <Card key={index} {...job} offered={true} />

              ))}

            </div>

          </Tabs.Panel>

          <Tabs.Panel value="interviewing" className="p-4 text-gray-300 text-md">

            <div className="grid grid-cols-2 gap-4">

              {jobList.map((job, index) => (

                <Card key={index} {...job} interviewing={true} />

              ))}

            </div>

          </Tabs.Panel>

        </Tabs>

      </div>

    </div>

  );

}

export default JobHistory;

import React, { useState } from "react";

import Marquee from "react-fast-marquee";

const companies = [

  { name: "Google", logo: "google.png" },

  { name: "Microsoft", logo: "microsoft.jpg" },

  { name: "Infosys", logo: "infosys.jpg" },

  { name: "Intel", logo: "intel.jpg" },

  { name: "Sony", logo: "sony.jpg" },

  { name: "Tata", logo: "tata.jpg" },

  { name: "Wipro", logo: "wipro.png" },

  { name: "HCL", logo: "hcl.png" },

  { name: "TCS", logo: "tcs.jpg" },

  { name: "Accenture", logo: "accenture.png" },

  { name: "Honeywell", logo: "honeywell.png" },

  { name: "Cognizant", logo: "cognizant.png" },

];

function CompaniesComponent() {

  const [isPaused, setIsPaused] = useState(false);

  return (

    <div className="bg-gray-900 py-8">

      {/\* Title Section \*/}

      <h2 className="text-center text-white font-extrabold text-2xl md:text-4xl mb-6">

        Trusted By <span className="text-yellow-400 text-3xl md:text-5xl">1K+</span> Companies

      </h2>

      {/\* Marquee Section \*/}

      <Marquee

        speed={50}

        gradient={true}

        gradientWidth={100}

        className="py-2"

        pauseOnHover={true}

        play={!isPaused}

      >

        {companies.map((company, index) => (

          <div

            key={index}

            className="mx-6 flex items-center"

            onMouseEnter={() => setIsPaused(true)}

            onMouseLeave={() => setIsPaused(false)}

          >

            <div

              className={`p-2 border-4 border-gray-700 shadow-lg transition-transform transform hover:scale-110 ${

                index === 0 ? "w-16 h-16 md:w-20 md:h-20" : "w-24 h-24 md:w-28 md:h-28"

              } rounded-full bg-white`}

            >

              <img

                src={`/Company/${company.logo}`}

                alt={company.name}

                className="w-full h-full rounded-full object-cover"

              />

            </div>

          </div>

        ))}

      </Marquee>

    </div>

  );

}

export default CompaniesComponent;

import { TextInput, Avatar, Button } from '@mantine/core';

import React, { useEffect, useState } from 'react';

import './DreamJob.css';

import { IconClock, IconSearch, IconUsers } from '@tabler/icons-react';

function DreamJob() {

  const [bgPosition, setBgPosition] = useState(0);

  const [tilt, setTilt] = useState({ x: 0, y: 0 });

  useEffect(() => {

    const interval = setInterval(() => {

      setBgPosition((prev) => (prev + 1) % 100);

    }, 50);

    return () => clearInterval(interval);

  }, []);

  const handleMouseMove = (event: React.MouseEvent<HTMLDivElement>) => {

    const { clientX, clientY } = event;

    const x = (clientX / window.innerWidth - 0.5) \* 15;

    const y = (clientY / window.innerHeight - 0.5) \* 15;

    setTilt({ x, y });

  };

  return (

    <div

      className="relative min-h-screen flex flex-col md:flex-row items-center px-8 md:px-16 py-16 gap-12 overflow-hidden parallax-container"

      onMouseMove={handleMouseMove}

    >

      <div

        className="absolute inset-0 bg-animated backdrop-blur-xl"

        style={{ backgroundPosition: `${bgPosition}% ${bgPosition}%` }}

      ></div>

      {/\* Left Section \*/}

      <div className="relative z-10 flex flex-col w-full md:w-1/2 space-y-6 text-center md:text-left">

        <h1 className="text-5xl md:text-7xl font-extrabold leading-tight text-gray-900 drop-shadow-xl">

          Find Your <br className="hidden md:block" />

          <span className="bg-gradient-to-r from-blue-600 via-purple-600 to-pink-500 text-transparent bg-clip-text">

            Dream Job

          </span>

        </h1>

        <p className="text-gray-300 text-lg max-w-lg leading-relaxed">

          A good life begins with a great company. Start exploring thousands of jobs in one place.

        </p>

        {/\* Search Bar \*/}

        <div className="flex flex-col md:flex-row gap-4 mt-6 p-6 bg-white/20 backdrop-blur-md shadow-2xl rounded-xl border border-white/30">

          <TextInput size="md" radius="md" variant="filled" placeholder="🔍 Job Title" className="shadow-lg rounded-lg" />

          <TextInput size="md" radius="md" variant="filled" placeholder="📍 Location" className="shadow-lg rounded-lg" />

          <Button size="md" radius="md" className="bg-blue-600 hover:bg-blue-700 text-white px-6 flex items-center shadow-xl transform transition duration-300 hover:scale-105">

            <IconSearch className="h-5 w-5 mr-2" stroke={2} /> Search

          </Button>

        </div>

      </div>

      {/\* Right Section \*/}

      <div

        className="relative z-10 w-full md:w-1/2 flex flex-col items-center space-y-6 parallax-item"

        style={{ transform: `rotateY(${tilt.x}deg) rotateX(${tilt.y}deg)` }}

      >

        <div className="text-center p-6 bg-white/20 backdrop-blur-md shadow-xl rounded-xl border border-white/30 relative">

          <p className="text-3xl font-bold bg-gradient-to-r from-blue-600 via-purple-600 to-pink-500 text-transparent bg-clip-text drop-shadow-md">

            10K+ Got Jobs

          </p>

          <div className="relative flex items-center justify-center mt-4">

            <div className="floating-avatars flex space-x-2">

              <img src="/avatar-9.png" alt="User" className="h-12 w-12 object-cover rounded-full border border-white shadow-lg" />

              <img src="/avatar-8.png" alt="User" className="h-12 w-12 object-cover rounded-full border border-white shadow-lg" />

              <Avatar radius="xl" size={50} className="bg-gradient-to-r from-blue-600 to-purple-600 text-white flex items-center justify-center border border-white shadow-lg text-sm font-semibold">

                +9K

              </Avatar>

            </div>

          </div>

        </div>

        {/\* Job Card \*/}

        <div className="relative group">

          <img src="/coding.png" alt="Coding" className="w-80 h-auto rounded-xl shadow-2xl transform transition duration-500 group-hover:scale-110 group-hover:rotate-3" />

          <div className="absolute inset-0 bg-gradient-to-r from-blue-500 to-purple-500 opacity-30 rounded-xl"></div>

        </div>

        {/\* Google Job \*/}

        <div className="text-white text-sm font-semibold mt-2 flex items-center space-x-4 p-4 bg-white/10 rounded-lg shadow-lg">

          <img src="/Google.png" alt="Google" className="h-10 w-10 rounded-full border border-white shadow-lg" />

          <div>

            <div className="text-lg font-bold text-gray-200">Software Engineer</div>

            <div className="text-gray-400">Delhi, India</div>

          </div>

        </div>

        {/\* Job Meta Info \*/}

        <div className="flex gap-4 mt-4 text-white text-sm font-medium items-center justify-center bg-white/10 py-2 px-4 rounded-lg shadow-md">

          <span className="flex items-center gap-1">

            <IconClock size={16} className="text-blue-400" />

            1 Day ago

          </span>

          <span className="flex items-center gap-1">

            <IconUsers size={16} className="text-green-400" />

            1230 Applicants

          </span>

        </div>

      </div>

    </div>

  );

}

export default DreamJob;

import React, { useRef, useState } from 'react';

import { Carousel } from '@mantine/carousel';

import { useInterval } from '@mantine/hooks';

import { rem } from '@mantine/core';

const jobCategories = [

    { title: 'Driver Jobs', description: 'Find driving opportunities that match your expertise and schedule.' },

    { title: 'IT Jobs', description: 'Explore tech roles in software development, data analysis, and more.' },

    { title: 'Healthcare Jobs', description: 'Discover opportunities in nursing, caregiving, and medical support.' },

    { title: 'Finance Jobs', description: 'Grow your career in banking, accounting, and financial analysis.' },

    { title: 'Education Jobs', description: 'Find teaching, tutoring, and academic research opportunities.' },

    { title: 'Engineering Jobs', description: 'Explore mechanical, civil, and software engineering roles.' }

];

const carouselItems = [

    { title: 'Digital Marketing', imgSrc: '/digital.png', description: 'Discover the latest trends in digital marketing and grow your career.', jobsPosted: '500+ Job Posted' },

    { title: 'Game Development', imgSrc: '/game.png', description: 'Join the gaming industry and create immersive experiences.', jobsPosted: '300+ Job Posted' },

    { title: 'Freelancing', imgSrc: '/freelancing.png', description: 'Work on your own terms with exciting freelancing opportunities.', jobsPosted: '700+ Job Posted' },

    { title: 'Coding', imgSrc: '/coding.png', description: 'Build innovative solutions and advance your programming skills.', jobsPosted: '1,000+ Job Posted' },

    { title: 'AI Development', imgSrc: '/ai.png', description: 'Shape the future by working on cutting-edge AI technologies.', jobsPosted: '400+ Job Posted' },

    { title: 'Cybersecurity', imgSrc: '/cyber.png', description: 'Protect digital assets with cybersecurity expertise.', jobsPosted: '600+ Job Posted' },

    { title: 'Blockchain', imgSrc: '/blockchain.png', description: 'Enter the world of decentralized technology and innovation.', jobsPosted: '350+ Job Posted' }

];

function JobCategory() {

    const [embla, setEmbla] = useState<any>(null);

    useInterval(() => {

        if (embla) {

            embla.scrollNext();

        }

    }, 3000);

    return (

        <div className="bg-gray-900 py-12 px-6">

            {/\* Job Categories Section \*/}

            <div className="max-w-7xl mx-auto text-center">

                <h2 className="text-white font-extrabold text-3xl md:text-5xl mb-4">

                    Browse <span className="text-yellow-400">Job Categories</span>

                </h2>

                <p className="text-gray-300 text-lg md:text-xl mb-8">

                    Explore job opportunities tailored to your skills. Start your career journey today.

                </p>

                <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-3 gap-6">

                    {jobCategories.map((category, index) => (

                        <div

                            key={index}

                            className="bg-gray-800  p-6 rounded-lg shadow-lg hover:shadow-xl transition-shadow transform hover:scale-105"

                        >

                            <h3 className="text-yellow-400 font-bold text-xl mb-2">{category.title}</h3>

                            <p className="text-gray-300">{category.description}</p>

                        </div>

                    ))}

                </div>

            </div>

            {/\* Trending Job Sectors Section \*/}

            <div className="mt-12">

                <h2 className="text-white font-extrabold text-3xl md:text-4xl text-center mb-6">

                    Trending <span className="text-yellow-400">Job Sectors</span>

                </h2>

                <Carousel

                    getEmblaApi={setEmbla}

                    slideSize="20%"

                    slideGap="md"

                    loop

                    align="start"

                    className="max-w-6xl mx-auto"

                    withIndicators

                    height={rem(250)}

                >

                    {carouselItems.map((item, index) => (

                        <Carousel.Slide key={index}>

                            <div className="flex flex-col justify-center p-6 rounded-lg bg-gray-800 text-white text-center hover:shadow-xl transform hover:scale-105 transition-all duration-300">

                                <div className="flex justify-center p-2 rounded-full bg-gray-700">

                                    <img

                                        src={item.imgSrc}

                                        alt={item.title}

                                        className="h-16 w-16 object-cover rounded-full"

                                    />

                                </div>

                                <h4 className="text-yellow-400 font-bold text-lg mt-4">{item.title}</h4>

                                <p className="text-gray-300 text-sm mt-2">{item.description}</p>

                                <p className="text-gray-400 text-xs mt-3">{item.jobsPosted}</p>

                            </div>

                        </Carousel.Slide>

                    ))}

                </Carousel>

            </div>

        </div>

    );

}

export default JobCategory;

import { Button, TextInput } from '@mantine/core';

import React, { useState } from 'react';

const Subscribe: React.FC = () => {

  const [email, setEmail] = useState('');

  const [error, setError] = useState('');

  const isValidEmail = (email: string) => /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email);

  const handleSubscribe = () => {

    if (!email.trim()) {

      setError('Email is required.');

      return;

    }

    if (!isValidEmail(email)) {

      setError('Please enter a valid email address.');

      return;

    }

    setError('');

    alert('Subscribed successfully!');

  };

  return (

    <div className="mt-16 mx-auto max-w-3xl px-8 py-10 bg-gradient-to-r from-indigo-600 to-purple-700 rounded-3xl shadow-2xl text-center md:flex md:items-center md:justify-between md:gap-6">

      <h2 className="text-white font-bold text-3xl md:text-4xl lg:text-5xl leading-snug md:w-2/3">

        Never Miss <span className="text-yellow-300">Job Offers</span> Again!

      </h2>

      <div className="flex flex-col md:flex-row items-center gap-4 mt-6 md:mt-0 w-full md:w-auto">

        <div className="w-full md:w-72">

          <TextInput

            className="[&\_input]:text-gray-900 w-full text-lg px-4 py-3 rounded-lg"

            placeholder="Enter your email"

            size="md"

            radius="md"

            value={email}

            onChange={(e) => setEmail(e.target.value)}

            error={error.length > 0}

          />

          {error && <p className="text-red-500 text-sm mt-2">{error}</p>}

        </div>

        <Button

          size="md"

          color="yellow"

          variant="filled"

          radius="xl"

          className="font-semibold shadow-lg px-6 py-3 text-lg transition-transform transform hover:scale-105 hover:shadow-2xl"

          onClick={handleSubscribe}

        >

          Subscribe

        </Button>

      </div>

    </div>

  );

};

export default Subscribe;

import React, { useRef } from "react";

import { Avatar, Rating } from "@mantine/core";

import { testimonials } from "../../Data/Data";

const Testimonial: React.FC = () => {

  const scrollRef = useRef<HTMLDivElement>(null);

  const handleScroll = (direction: "left" | "right") => {

    if (scrollRef.current) {

      const scrollAmount = 300;

      scrollRef.current.scrollBy({

        left: direction === "left" ? -scrollAmount : scrollAmount,

        behavior: "smooth",

      });

    }

  };

  return (

    <section className="max-w-7xl mx-auto text-center px-6 lg:px-8 py-12 relative">

      <h2 className="text-white font-extrabold text-4xl md:text-5xl mb-8">

        What <span className="text-yellow-400">Users</span> Say About Us

      </h2>

      <div className="relative flex items-center">

        {/\* Left Scroll Button \*/}

        <button

          aria-label="Scroll Left"

          className="absolute left-0 z-10 bg-yellow-400 text-black p-2 rounded-full shadow-md hover:bg-yellow-500 transition"

          onClick={() => handleScroll("left")}

        >

          <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" strokeWidth="2" strokeLinecap="round" strokeLinejoin="round">

            <polyline points="15 18 9 12 15 6"></polyline>

          </svg>

        </button>

        {/\* Testimonials Container \*/}

        <div

          ref={scrollRef}

          className="flex space-x-6 overflow-x-auto overflow-hidden snap-x snap-mandatory hide-scrollbar px-4"

        >

          {testimonials.map(({ id, avatar, name, rating, testimonial }) => (

            <div key={id} className="flex-shrink-0 w-80 snap-center">

              <div className="border-2 border-yellow-400 p-6 rounded-lg bg-gray-800 shadow-lg flex flex-col items-center">

                <Avatar src={avatar} alt={name} className="h-16 w-16 rounded-full" />

                <div className="mt-4 text-center">

                  <p className="text-lg font-semibold text-yellow-400">{name}</p>

                  <Rating value={rating} fractions={2} readOnly />

                </div>

                <p className="text-sm text-green-400 mt-4 text-center">{testimonial}</p>

              </div>

            </div>

          ))}

        </div>

        {/\* Right Scroll Button \*/}

        <button

          aria-label="Scroll Right"

          className="absolute right-0 z-10 bg-yellow-400 text-black p-2 rounded-full shadow-md hover:bg-yellow-500 transition"

          onClick={() => handleScroll("right")}

        >

          <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" strokeWidth="2" strokeLinecap="round" strokeLinejoin="round">

            <polyline points="9 18 15 12 9 6"></polyline>

          </svg>

        </button>

      </div>

    </section>

  );

};

export default Testimonial;

import React from "react";

import { Avatar } from "@mantine/core";

import { workerData } from "../../Data/Data";

const Working: React.FC = () => {

  return (

    <section className="bg-gray-900 py-16">

      {/\* Heading Section \*/}

      <div className="max-w-7xl mx-auto text-center px-6 lg:px-8">

        <h2 className="text-white font-extrabold text-4xl md:text-5xl mb-6">

          How <span className="text-yellow-400">It</span> Works

        </h2>

        <p className="text-gray-300 text-lg md:text-xl mb-10 max-w-2xl mx-auto">

          Explore exciting job opportunities that match your skills and career goals.

          Take the next step towards your dream job today!

        </p>

      </div>

      {/\* Content Section \*/}

      <div className="max-w-7xl mx-auto px-6 lg:px-8 grid grid-cols-1 md:grid-cols-2 items-center gap-10">

        {/\* Left Image Section \*/}

        <div className="relative w-full">

          <img

            src="Girl.png"

            alt="Career Journey"

            className="w-full h-auto rounded-lg shadow-lg"

          />

          {/\* Profile Completion Box \*/}

          <div className="absolute top-[15%] right-0 flex flex-col items-center gap-4 border border-white rounded-xl py-4 px-2 backdrop-blur-md bg-gray-800/50 w-36">

            <Avatar src="/images/avatar-8.png" alt="User Avatar" className="h-16 w-16" />

            <div className="text-sm font-semibold text-center text-white">

              Complete your profile

            </div>

            <div className="text-sm font-semibold text-yellow-400">70% Completed</div>

          </div>

        </div>

        {/\* Right Features List \*/}

        <div className="w-full flex flex-col gap-6">

          {workerData.map(({ id, image, title, description }) => (

            <div

              key={id}

              className="flex items-center gap-4 p-6 bg-gray-800 rounded-xl shadow-lg transition-transform duration-300 hover:scale-105"

            >

              <img

                src={image}

                alt={title}

                className="w-14 h-14 bg-yellow-400 p-1 rounded-full object-contain"

              />

              <div>

                <h3 className="text-white font-semibold text-xl">{title}</h3>

                <p className="text-gray-400 text-sm md:text-base">{description}</p>

              </div>

            </div>

          ))}

        </div>

      </div>

    </section>

  );

};

export default Working;

import { Tabs } from '@mantine/core'

import React from 'react'

import PostedJobCard from './PostedJobCard'

import { activeJob } from '../../Data/PostedJobData'

function PostedJob() {

  return (

    <div className='w-1/6 mt-5'>

      <div className="text-2xl font-semibold mb-5">Job</div>

      <div>

      <Tabs autoContrast variant="pills" defaultValue="gallery">

      <Tabs.List className='[&\_button[aria-selected="false"]]'>

        <Tabs.Tab value="active">Active [4]</Tabs.Tab>

        <Tabs.Tab value="draft">Draft [1]</Tabs.Tab>

      </Tabs.List>

      <Tabs.Panel value="active">

        <div className='flex flex-col gap-3  mt-5'>

          {

            activeJob.map((item,index)=><PostedJobCard key={index} {...item}/>)

          }

        </div>

        </Tabs.Panel>

      <Tabs.Panel value="draft">s</Tabs.Panel>

    </Tabs>

      </div>

    </div>

  )

}

export default PostedJob

import React from 'react'

function PostedJobCard(props:any) {

  return (

    <div className='rounded-xl p-2 border-l-2 border-l-yellow-400'>

      <div className='text-sm font-medium'>{props.JobTitle}</div>

      <div className='text-xs'>{props.Location}</div>

      <div className='text-xs'>{props.Posted}</div>

    </div>

  )

}

export default PostedJobCard

import React from 'react';

import { Badge, Tabs } from '@mantine/core';

import JobDesk from '../JobDescription/JobDesk';

import talentsData from '../FindTalent/TalentData';

import TalentCard from '../FindTalent/TalentCard';

function PostedJobDesc() {

  return (

    <div className="w-3/4 mt-5 px-5">

      {/\* Job Title with Badge \*/}

      <div className="text-2xl font-semibold flex items-center">

        Job <Badge variant="light" color="yellow">Badge</Badge>

      </div>

      {/\* Job Location \*/}

      <div className="font-medium mb-5">New York, United States</div>

      {/\* Tabs Section \*/}

      <Tabs variant="outline" defaultValue="overView" className="mt-4">

        <Tabs.List className="flex justify-center space-x-4">

          <Tabs.Tab value="overView" className="text-gray-600 font-medium">

            Overview

          </Tabs.Tab>

          <Tabs.Tab value="applicant" className="text-gray-600 font-medium">

            Applicants

          </Tabs.Tab>

          <Tabs.Tab value="invited" className="text-gray-600 font-medium">

            Invited

          </Tabs.Tab>

        </Tabs.List>

        {/\* Overview Panel \*/}

        <Tabs.Panel value="overView" className="p-4 text-gray-700 text-md leading-relaxed mb-5 w-full">

          <JobDesk edit />

        </Tabs.Panel>

        {/\* Applicants Panel \*/}

        <Tabs.Panel value="applicant" className="grid grid-cols-2 gap-4 p-4 text-gray-700 text-md">

          {talentsData.map((talent) => (

            <TalentCard key={talent.id} {...talent} posted />

          ))}

        </Tabs.Panel>

        {/\* Invited Panel \*/}

        <Tabs.Panel value="invited" className="grid grid-cols-2 gap-4 p-4 text-gray-700 text-md">

          {talentsData.map((talent) => (

            <TalentCard key={talent.id} {...talent} invited />

          ))}

        </Tabs.Panel>

      </Tabs>

    </div>

  );

}

export default PostedJobDesc;

import React, { useState } from "react";

import { SelectInput } from "../PostJob/SelectInput";

import TextEditor from "./TextEditor";

import { fields, content } from "../../Data/PostJobData";

import { useForm } from "@mantine/form";

import { notifications } from "@mantine/notifications";

import { LoadingOverlay } from "@mantine/core";

import { postJob, resetJobState } from "../../Slices/PostJobSlice";

import { useDispatch, useSelector } from "react-redux";

import { AppDispatch, RootState } from "../../Store";

interface FormValues {

  jobTitle: string;

  company: string;

  about: string;

  experience: string;

  jobType: string;

  location: string;

  packageOffer: string;

  description: string;

  skillsRequired: string[];

  jobStatus: "OPEN" | "CLOSE" | "DRAFT";

}

const PostJob = () => {

  const dispatch = useDispatch<AppDispatch>();

  const { loading } = useSelector((state: RootState) => state.postJob as { loading: boolean });

  const [editorContent, setEditorContent] = useState(content);

  const form = useForm<FormValues>({

    initialValues: {

      jobTitle: "",

      company: "",

      about: "",

      experience: "",

      jobType: "",

      location: "",

      packageOffer: "",

      description: content,

      skillsRequired: [],

      jobStatus: "OPEN",

    },

    validate: {

      jobTitle: (value) => (value ? null : "Job title is required"),

      company: (value) => (value ? null : "Company is required"),

      jobType: (value) => (value ? null : "Job type is required"),

      location: (value) => (value ? null : "Location is required"),

      packageOffer: (value) => (value ? null : "Salary is required"),

      skillsRequired: (value) => (value.length > 0 ? null : "At least one skill is required"),

    },

  });

  const handleEditorChange = (content: string) => {

    setEditorContent(content);

    form.setFieldValue("description", content);

  };

  const handleSubmit = async (status: "OPEN" | "DRAFT") => {

    form.setFieldValue("jobStatus", status);

    const values = form.values;

    const validationResult = form.validate();

    if (validationResult.hasErrors) {

      notifications.show({

        title: "Validation Error",

        message: "Please fill all required fields",

        color: "red",

      });

      return;

    }

    try {

      const jobData = {

        ...values,

        description: editorContent,

      };

      await dispatch(postJob(jobData)).unwrap();

      notifications.show({

        title: "Success",

        message: `Job ${status === "OPEN" ? "published" : "saved as draft"} successfully!`,

        color: "green",

      });

      form.reset();

      setEditorContent(content);

      dispatch(resetJobState());

    } catch (error: any) {

      notifications.show({

        title: "Error",

        message: error?.message || "Failed to post job. Please try again.",

        color: "red",

      });

    }

  };

  return (

    <div className="w-full max-w-4xl mx-auto p-10 bg-gray-800 shadow-xl rounded-lg text-gray-200 relative">

      <LoadingOverlay visible={loading} loaderProps={{ size: "lg" }} />

      <h2 className="text-4xl font-extrabold mb-8 text-center text-gray-100">

        Post a Job

      </h2>

      <div className="grid grid-cols-1 md:grid-cols-2 gap-8 mb-8">

        {fields.map((field, index) => {

          const fieldName = field.label.toLowerCase().replace(/\s+/g, '');

          // Only use SelectInput for fields that need multi-select (skillsRequired)

          if (fieldName === 'skillsrequired') {

            return (

              <SelectInput

                key={index}

                label={field.label}

                placeholder={field.placeholder}

                options={field.options}

                name={fieldName}

                value={form.values[fieldName as keyof FormValues] || []}

                onChange={(value) => form.setFieldValue(fieldName, value)}

                error={form.errors[fieldName as keyof FormValues] as string | undefined}

              />

            );

          }

          // For other fields, use regular input

          return (

            <div key={index} className="flex flex-col space-y-2">

              <label className="text-lg font-semibold text-gray-300">

                {field.label} <span className="text-red-600">\*</span>

              </label>

              <input

                type="text"

                value={form.values[fieldName as keyof FormValues] || ''}

                onChange={(e) => form.setFieldValue(fieldName, e.target.value)}

                placeholder={field.placeholder}

                className="p-3 text-base border border-gray-400 rounded-lg shadow-md focus:ring-2 focus:ring-indigo-500 transition-all duration-200 bg-white text-gray-800 placeholder-gray-500 w-full"

              />

              {form.errors[fieldName as keyof FormValues] && (

                <p className="text-red-500 text-sm">

                  {form.errors[fieldName as keyof FormValues] as string}

                </p>

              )}

            </div>

          );

        })}

      </div>

      <div className="mb-8">

        <label className="block text-lg font-medium text-gray-300 mb-2">

          Job Description

        </label>

        <div className="border border-gray-500 rounded-lg p-4 bg-gray-700">

          <TextEditor

            content={editorContent}

            onChange={handleEditorChange}

          />

        </div>

      </div>

      <div className="flex justify-end gap-4">

        <button

          className="bg-green-600 text-gray-100 px-4 py-2 rounded-lg shadow hover:bg-green-700"

          onClick={() => handleSubmit("OPEN")}

          disabled={loading}

          aria-label="Publish Job"

        >

          Publish Job

        </button>

        <button

          className="bg-yellow-600 text-gray-100 px-4 py-2 rounded-lg shadow hover:bg-yellow-700"

          onClick={() => handleSubmit("DRAFT")}

          disabled={loading}

          aria-label="Save as Draft"

        >

          Save as Draft

        </button>

      </div>

    </div>

  );

};

export default PostJob;

// SelectInput.tsx

import React, { useState } from "react";

import { InputBase, Combobox, useCombobox, CloseButton, Button } from "@mantine/core";

import { motion } from "framer-motion";

interface SelectInputProps {

    label: string;

    placeholder: string;

    options: string[];

    name: string;

    value: any;

    onChange: (value: any) => void;

    error?: string;

}

export const SelectInput: React.FC<SelectInputProps> = ({

    label,

    placeholder,

    options,

    name,

    value,

    onChange,

    error,

}) => {

    const combobox = useCombobox({

        onDropdownClose: () => combobox.resetSelectedOption(),

    });

    const [search, setSearch] = useState("");

    // Ensure value is always an array

    const safeValue = Array.isArray(value) ? value : [];

    const filteredOptions = options.filter(

        (item) =>

            item.toLowerCase().includes(search.toLowerCase().trim()) &&

            !safeValue.includes(item)

    );

    const handleSelect = (val: string) => {

        if (!safeValue.includes(val)) {

            onChange([...safeValue, val]);

        }

        setSearch("");

        combobox.closeDropdown();

    };

    const handleRemove = (val: string) => {

        onChange(safeValue.filter((item: string) => item !== val));

    };

    const handleClearAll = () => {

        onChange([]);

    };

    return (

        <div className="flex flex-col w-full space-y-4">

            <label className="text-lg font-semibold text-gray-300">

                {label} <span className="text-red-600">\*</span>

            </label>

            <div className="flex flex-wrap gap-2 items-center w-full">

                {safeValue.length <= 2 ? (

                    safeValue.map((skill: string) => (

                        <div

                            key={skill}

                            className="flex items-center bg-indigo-200 text-indigo-900 px-3 py-1 rounded-lg text-sm w-auto"

                        >

                            {skill}

                            <CloseButton

                                onClick={() => handleRemove(skill)}

                                className="ml-2 text-indigo-600 hover:text-indigo-800"

                                size="xs"

                            />

                        </div>

                    ))

                ) : (

                    <div className="flex items-center bg-indigo-200 text-indigo-900 px-3 py-1 rounded-lg text-sm w-auto">

                        {safeValue.slice(0, 2).join(", ")} +{safeValue.length - 2}

                    </div>

                )}

                {safeValue.length > 0 && (

                    <Button

                        variant="outline"

                        size="xs"

                        color="red"

                        onClick={handleClearAll}

                        className="ml-auto text-red-600 border-red-600 hover:bg-red-100"

                    >

                        Clear All

                    </Button>

                )}

            </div>

            <Combobox store={combobox} onOptionSubmit={handleSelect}>

                <Combobox.Target>

                    <InputBase

                        required

                        rightSection={<span className="text-indigo-600">▼</span>}

                        rightSectionPointerEvents="none"

                        onClick={() => combobox.openDropdown()}

                        onFocus={() => combobox.openDropdown()}

                        onBlur={() => combobox.closeDropdown()}

                        placeholder={placeholder}

                        value={search}

                        onChange={(event) => setSearch(event.currentTarget.value)}

                        className="p-3 text-base border border-gray-400 rounded-lg shadow-md focus:ring-2 focus:ring-indigo-500 transition-all duration-200 bg-white text-gray-800 placeholder-gray-500 w-full"

                        error={error}

                    />

                </Combobox.Target>

                {combobox.dropdownOpened && (

                    <Combobox.Dropdown>

                        <motion.div

                            initial={{ opacity: 0, y: -10 }}

                            animate={{ opacity: 1, y: 0 }}

                            exit={{ opacity: 0, y: -10 }}

                            className="mt-1 bg-white border border-gray-300 rounded-lg shadow-lg overflow-hidden"

                        >

                            <div className="max-h-40 overflow-y-auto scrollbar-thin scrollbar-thumb-indigo-400 scrollbar-track-gray-200">

                                <Combobox.Options>

                                    {filteredOptions.length > 0 ? (

                                        filteredOptions.map((item) => (

                                            <Combobox.Option

                                                key={item}

                                                value={item}

                                                className="px-4 py-2 text-gray-800 hover:bg-indigo-100 hover:text-indigo-900 cursor-pointer transition-all"

                                            >

                                                {item}

                                            </Combobox.Option>

                                        ))

                                    ) : (

                                        <Combobox.Empty className="px-4 py-2 text-gray-500">

                                            No results found

                                        </Combobox.Empty>

                                    )}

                                </Combobox.Options>

                            </div>

                        </motion.div>

                    </Combobox.Dropdown>

                )}

            </Combobox>

            {error && <p className="text-red-500 text-sm">{error}</p>}

        </div>

    );

};

export default SelectInput;

// TextEditor.tsx

import { RichTextEditor, Link } from '@mantine/tiptap';

import { useEditor } from '@tiptap/react';

import Highlight from '@tiptap/extension-highlight';

import StarterKit from '@tiptap/starter-kit';

import Underline from '@tiptap/extension-underline';

import TextAlign from '@tiptap/extension-text-align';

import Superscript from '@tiptap/extension-superscript';

import SubScript from '@tiptap/extension-subscript';

interface TextEditorProps {

  content: string;

  onChange: (content: string) => void;

}

const TextEditor: React.FC<TextEditorProps> = ({ content, onChange }) => {

    const editor = useEditor({

        extensions: [

            StarterKit,

            Underline,

            Link,

            Superscript,

            SubScript,

            Highlight,

            TextAlign.configure({ types: ['heading', 'paragraph'] }),

        ],

        content,

        onUpdate: ({ editor }) => {

            onChange(editor.getHTML());

        },

    });

    return (

        <RichTextEditor editor={editor}>

            <RichTextEditor.Toolbar sticky stickyOffset={60}>

                <RichTextEditor.ControlsGroup>

                    <RichTextEditor.Bold />

                    <RichTextEditor.Italic />

                    <RichTextEditor.Underline />

                    <RichTextEditor.Strikethrough />

                    <RichTextEditor.ClearFormatting />

                    <RichTextEditor.Highlight />

                    <RichTextEditor.Code />

                </RichTextEditor.ControlsGroup>

                <RichTextEditor.ControlsGroup>

                    <RichTextEditor.H4 />

                </RichTextEditor.ControlsGroup>

                <RichTextEditor.ControlsGroup>

                    <RichTextEditor.Blockquote />

                    <RichTextEditor.Hr />

                    <RichTextEditor.BulletList />

                    <RichTextEditor.OrderedList />

                    <RichTextEditor.Subscript />

                    <RichTextEditor.Superscript />

                </RichTextEditor.ControlsGroup>

                <RichTextEditor.ControlsGroup>

                    <RichTextEditor.Link />

                    <RichTextEditor.Unlink />

                </RichTextEditor.ControlsGroup>

                <RichTextEditor.ControlsGroup>

                    <RichTextEditor.AlignLeft />

                    <RichTextEditor.AlignCenter />

                    <RichTextEditor.AlignJustify />

                    <RichTextEditor.AlignRight />

                </RichTextEditor.ControlsGroup>

                <RichTextEditor.ControlsGroup>

                    <RichTextEditor.Undo />

                    <RichTextEditor.Redo />

                </RichTextEditor.ControlsGroup>

            </RichTextEditor.Toolbar>

            <RichTextEditor.Content />

        </RichTextEditor>

    );

}

export default TextEditor;

import React, { useState } from "react";

import { IconEdit, IconTrash, IconCheck, IconX } from "@tabler/icons-react";

interface Certification {

  title: string;

  company: string;

  location?: string;

  issueDate: string;

  certificateID: string;

}

interface CertiCardProps extends Certification {

  onEdit?: (updatedCert: Certification) => void;

  onDelete?: () => void;

  editable?: boolean;

}

/\*\*

 \* Certification card component that displays and allows editing of certification details

 \*/

const CertiCard: React.FC<CertiCardProps> = ({

  title,

  company,

  location = "Global",

  issueDate,

  certificateID,

  onEdit,

  onDelete,

  editable = false,

}) => {

  const [isEditing, setIsEditing] = useState(false);

  const [editedCert, setEditedCert] = useState<Certification>({

    title,

    company,

    location,

    issueDate,

    certificateID,

  });

  /\*\*

   \* Formats company name for logo path

   \* @param companyName - Name of the company

   \* @returns Formatted string for logo path

   \*/

  const formatCompanyLogoPath = (companyName: string) => {

    return companyName.toLowerCase().replace(/\s+/g, "-");

  };

  /\*\*

   \* Handles image loading errors by setting a default image

   \*/

  const handleImageError = (e: React.SyntheticEvent<HTMLImageElement>) => {

    const target = e.target as HTMLImageElement;

    target.src = "/logos/default.jpeg";

    target.onerror = null;

  };

  /\*\*

   \* Handles input changes during editing

   \* @param field - Field to update

   \* @param value - New value

   \*/

  const handleInputChange = (field: keyof Certification, value: string) => {

    setEditedCert(prev => ({ ...prev, [field]: value }));

  };

  /\*\*

   \* Saves the edited certification

   \*/

  const handleSave = () => {

    if (!editedCert.title || !editedCert.company || !editedCert.issueDate || !editedCert.certificateID) {

      return;

    }

    onEdit?.(editedCert);

    setIsEditing(false);

  };

  /\*\*

   \* Cancels editing and resets to original values

   \*/

  const handleCancel = () => {

    setEditedCert({ title, company, location, issueDate, certificateID });

    setIsEditing(false);

  };

  return (

    <div className="p-6 bg-white dark:bg-gray-800 text-gray-900 dark:text-white shadow-md rounded-xl border border-gray-200 dark:border-gray-700 flex flex-col gap-4 transition-all hover:shadow-lg hover:-translate-y-1 relative">

      {/\* Editable controls \*/}

      {editable && (

        <div className="absolute top-2 right-2 flex gap-2">

          {!isEditing ? (

            <>

              <button

                onClick={() => setIsEditing(true)}

                className="p-1 text-blue-600 hover:bg-blue-100 dark:hover:bg-blue-900 rounded-full"

                aria-label="Edit certification"

              >

                <IconEdit size={18} />

              </button>

              <button

                onClick={onDelete}

                className="p-1 text-red-600 hover:bg-red-100 dark:hover:bg-red-900 rounded-full"

                aria-label="Delete certification"

              >

                <IconTrash size={18} />

              </button>

            </>

          ) : (

            <>

              <button

                onClick={handleSave}

                className="p-1 text-green-600 hover:bg-green-100 dark:hover:bg-green-900 rounded-full"

                aria-label="Save changes"

              >

                <IconCheck size={18} />

              </button>

              <button

                onClick={handleCancel}

                className="p-1 text-red-600 hover:bg-red-100 dark:hover:bg-red-900 rounded-full"

                aria-label="Cancel editing"

              >

                <IconX size={18} />

              </button>

            </>

          )}

        </div>

      )}

      {/\* Certification content \*/}

      <div className="flex gap-4 items-start">

        <div className="p-2 bg-gray-100 dark:bg-gray-700 rounded-lg flex-shrink-0">

          <img

            className="h-14 w-14 object-contain rounded-md"

            src={`/logos/${formatCompanyLogoPath(company)}.jpeg`}

            alt={`${company} logo`}

            onError={handleImageError}

            loading="lazy"

          />

        </div>

        <div className="flex-1">

          {isEditing ? (

            <div className="space-y-3">

              <input

                className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-700"

                value={editedCert.title}

                onChange={(e) => handleInputChange("title", e.target.value)}

                placeholder="Certification Title\*"

                required

                aria-label="Certification title"

              />

              <input

                className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-700"

                value={editedCert.company}

                onChange={(e) => handleInputChange("company", e.target.value)}

                placeholder="Issuing Company\*"

                required

                aria-label="Issuing company"

              />

              <input

                className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-700"

                value={editedCert.location || ""}

                onChange={(e) => handleInputChange("location", e.target.value)}

                placeholder="Location (optional)"

                aria-label="Location"

              />

              <div className="flex gap-2">

                <input

                  type="date"

                  className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-700"

                  value={editedCert.issueDate}

                  onChange={(e) => handleInputChange("issueDate", e.target.value)}

                  required

                  aria-label="Issue date"

                />

                <input

                  className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-700"

                  value={editedCert.certificateID}

                  onChange={(e) => handleInputChange("certificateID", e.target.value)}

                  placeholder="Certificate ID\*"

                  required

                  aria-label="Certificate ID"

                />

              </div>

            </div>

          ) : (

            <>

              <h3 className="text-lg font-semibold text-gray-900 dark:text-white">{title}</h3>

              <div className="flex flex-wrap items-center gap-1 text-sm text-gray-600 dark:text-gray-300 mt-1">

                <span className="font-medium">{company}</span>

                {location && <span className="text-gray-400">•</span>}

                <span>{location}</span>

              </div>

              <div className="mt-2 flex flex-wrap items-center gap-3 text-xs text-gray-500 dark:text-gray-400">

                <span>

                  Issued: {new Date(issueDate).toLocaleDateString("en-US", {

                    year: "numeric",

                    month: "long",

                  })}

                </span>

                <span className="bg-gray-100 dark:bg-gray-700 px-2 py-1 rounded">

                  ID: {certificateID}

                </span>

              </div>

            </>

          )}

        </div>

      </div>

    </div>

  );

};

export default CertiCard;

import React, { useState } from "react";

import { IconEdit, IconTrash, IconCheck, IconX } from "@tabler/icons-react";

interface Experience {

  title: string;

  company: string;

  location: string;

  startDate: string;

  endDate: string;

  description: string;

  currentlyWorking: boolean;

}

interface ExpCardProps extends Experience {

  onEdit?: (updatedExp: Experience) => void;

  onDelete?: () => void;

  editable?: boolean;

}

const ExpCard: React.FC<ExpCardProps> = ({

  title,

  company,

  location,

  startDate,

  endDate,

  description,

  currentlyWorking = false,

  onEdit,

  onDelete,

  editable = false,

}) => {

  const [isEditing, setIsEditing] = useState(false);

  const [editedExp, setEditedExp] = useState<Experience>({

    title,

    company,

    location,

    startDate,

    endDate,

    description,

    currentlyWorking,

  });

  const formatDate = (dateString: string) => {

    if (dateString === "Present") return "Present";

    return new Date(dateString).toLocaleDateString("en-US", {

      year: "numeric",

      month: "short",

    });

  };

  const handleInputChange = (field: keyof Experience, value: string) => {

    setEditedExp(prev => ({ ...prev, [field]: value }));

  };

  const handleCurrentlyWorkingChange = (e: React.ChangeEvent<HTMLInputElement>) => {

    setEditedExp(prev => ({

      ...prev,

      currentlyWorking: e.target.checked,

      endDate: e.target.checked ? "Present" : prev.endDate,

    }));

  };

  const handleSave = () => {

    if (!editedExp.title || !editedExp.company || !editedExp.location || !editedExp.startDate ||

        (!editedExp.currentlyWorking && !editedExp.endDate)) {

      return;

    }

    onEdit?.(editedExp);

    setIsEditing(false);

  };

  const handleCancel = () => {

    setEditedExp({

      title,

      company,

      location,

      startDate,

      endDate,

      description,

      currentlyWorking,

    });

    setIsEditing(false);

  };

  return (

    <div className="p-6 bg-white dark:bg-gray-800 text-gray-900 dark:text-white shadow-md rounded-xl border border-gray-200 dark:border-gray-700 flex flex-col gap-3 transition-all hover:shadow-lg hover:-translate-y-1 relative">

      {editable && (

        <div className="absolute top-2 right-2 flex gap-2">

          {!isEditing ? (

            <>

              <button

                onClick={() => setIsEditing(true)}

                className="p-1 text-blue-600 hover:bg-blue-100 dark:hover:bg-blue-900 rounded-full"

                aria-label="Edit experience"

              >

                <IconEdit size={18} />

              </button>

              <button

                onClick={onDelete}

                className="p-1 text-red-600 hover:bg-red-100 dark:hover:bg-red-900 rounded-full"

                aria-label="Delete experience"

              >

                <IconTrash size={18} />

              </button>

            </>

          ) : (

            <>

              <button

                onClick={handleSave}

                className="p-1 text-green-600 hover:bg-green-100 dark:hover:bg-green-900 rounded-full"

                aria-label="Save changes"

              >

                <IconCheck size={18} />

              </button>

              <button

                onClick={handleCancel}

                className="p-1 text-red-600 hover:bg-red-100 dark:hover:bg-red-900 rounded-full"

                aria-label="Cancel editing"

              >

                <IconX size={18} />

              </button>

            </>

          )}

        </div>

      )}

      {isEditing ? (

        <div className="space-y-3">

          <input

            className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-700"

            value={editedExp.title}

            onChange={(e) => handleInputChange("title", e.target.value)}

            placeholder="Job Title\*"

            required

            aria-label="Job title"

          />

          <input

            className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-700"

            value={editedExp.company}

            onChange={(e) => handleInputChange("company", e.target.value)}

            placeholder="Company\*"

            required

            aria-label="Company"

          />

          <input

            className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-700"

            value={editedExp.location}

            onChange={(e) => handleInputChange("location", e.target.value)}

            placeholder="Location\*"

            required

            aria-label="Location"

          />

          <div className="flex gap-2 items-center">

            <input

              type="date"

              className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-700"

              value={editedExp.startDate}

              onChange={(e) => handleInputChange("startDate", e.target.value)}

              required

              aria-label="Start date"

            />

            {!editedExp.currentlyWorking && (

              <input

                type="date"

                className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-700"

                value={editedExp.endDate === "Present" ? "" : editedExp.endDate}

                onChange={(e) => handleInputChange("endDate", e.target.value)}

                required={!editedExp.currentlyWorking}

                aria-label="End date"

              />

            )}

            <label className="flex items-center gap-2 whitespace-nowrap">

              <input

                type="checkbox"

                checked={editedExp.currentlyWorking}

                onChange={handleCurrentlyWorkingChange}

                aria-label="Currently working here"

              />

              Currently Working

            </label>

          </div>

          <textarea

            className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-700"

            value={editedExp.description}

            onChange={(e) => handleInputChange("description", e.target.value)}

            placeholder="Job description"

            rows={4}

            aria-label="Job description"

          />

        </div>

      ) : (

        <>

          <div className="flex justify-between items-start">

            <div>

              <h3 className="text-lg font-semibold">{title}</h3>

              <p className="text-gray-600 dark:text-gray-300">

                {company} • {location}

              </p>

            </div>

            <span className="text-sm bg-blue-100 dark:bg-blue-900 text-blue-800 dark:text-blue-200 px-3 py-1 rounded-full">

              {formatDate(startDate)} - {formatDate(endDate)}

              {currentlyWorking && " (Current)"}

            </span>

          </div>

          <p className="text-gray-700 dark:text-gray-300 text-sm mt-2 leading-relaxed">

            {description}

          </p>

        </>

      )}

    </div>

  );

};

export default ExpCard;

import {

  IconMapPin,

  IconBriefcase,

  IconCurrencyRupee,

  IconUser,

  IconCertificate,

  IconTools,

  IconBuildingSkyscraper,

  IconEdit,

  IconDeviceFloppy,

  IconTrash,

  IconPlus,

  IconX,

  IconUpload,

} from "@tabler/icons-react";

import React, { useState, useRef, useEffect } from "react";

import { Talent } from "../FindTalent/TalentData";

import { Divider } from "@mantine/core";

import ExpCard from "./ExpCard";

import { getProfile, updateProfile as updateProfileApi } from "../../Services/ProfileService";

import { setProfile, updateProfile } from "../../Slices/ProfileSlice";

import CertiCard from "./CertiCard";

import { useSelector, useDispatch } from "react-redux";

import { toast } from "react-toastify";

// In Profile.tsx

const mapProfileToTalent = (profileData: any): Talent => {

  return {

    id: profileData.id,

    name: profileData.name || "",

    email: profileData.email,

    role: profileData.jobTitle || "",

    company: profileData.company || "",

    location: profileData.location || "",

    about: profileData.about || "",

    topSkill: profileData.skills || [],

    salary: profileData.salary || "",

    experience: profileData.experiences?.map((exp: any) => ({

      title: exp.title,

      company: exp.company,

      location: exp.location,

      startDate: exp.startDate,

      endDate: exp.currentlyWorking ? "Present" : exp.endDate,

      description: exp.description,

      currentlyWorking: exp.currentlyWorking

    })) || [],

    certifications: profileData.certifications?.map((cert: any) => ({

      title: cert.name,

      company: cert.issuer,

      location: cert.location,

      issueDate: cert.issueDate,

      certificateID: cert.certificateId

    })) || [],

    avatar: profileData.avatar || "/avatar-7.png",

    banner: profileData.banner || "/color7.png"

  };

};

const mapTalentToProfile = (talentData: Talent): any => {

  return {

    id: talentData.id,

    email: talentData.email,

    jobTitle: talentData.role,

    company: talentData.company,

    location: talentData.location,

    about: talentData.about,

    skills: talentData.topSkill,

    salary: talentData.salary,

    experiences: talentData.experience?.map(exp => ({

      title: exp.title,

      company: exp.company,

      location: exp.location,

      startDate: exp.startDate,

      endDate: exp.currentlyWorking ? null : exp.endDate,

      currentlyWorking: exp.currentlyWorking,

      description: exp.description

    })) || [],

    certifications: talentData.certifications?.map(cert => ({

      name: cert.title,

      issuer: cert.company,

      location: cert.location,

      issueDate: cert.issueDate,

      certificateId: cert.certificateID

    })) || [],

    avatar: talentData.avatar,

    banner: talentData.banner

  };

};

// Enhanced Type definitions

interface Certification {

  name: string;

  issuer: string;

  issueDate: string;

  certificateId: string;

  location?: string;

}

interface Experience {

  title: string;

  company: string;

  location: string;

  startDate: string;

  endDate: string;

  description: string;

  currentlyWorking: boolean;

}

interface EditModes {

  about: boolean;

  skills: boolean;

  salary: boolean;

  personal: boolean;

}

interface ProfileProps {

  talent: Talent | null;

  onBack: () => void;

  onSave?: (updatedTalent: Talent) => void;

}

const Profile: React.FC<ProfileProps> = ({ talent, onBack, onSave }) => {

  const dispatch = useDispatch();

  const user = useSelector((state: any) => state.user);

  const profile = useSelector((state: any) => state.profile);

  // State management

  const [editMode, setEditMode] = useState<EditModes>({

    about: false,

    skills: false,

    salary: false,

    personal: false,

  });

  const [editableTalent, setEditableTalent] = useState<Talent | null>(talent);

  const [isAddingCert, setIsAddingCert] = useState(false);

  const [isAddingExp, setIsAddingExp] = useState(false);

  const [isLoading, setIsLoading] = useState(false);

  const [newCert, setNewCert] = useState<Certification>({

    name: "",

    issuer: "",

    location: "",

    issueDate: new Date().toISOString().split("T")[0],

    certificateId: "",

  });

  const [newExp, setNewExp] = useState<Experience>({

    title: "",

    company: "",

    location: "",

    startDate: new Date().toISOString().split("T")[0],

    endDate: "",

    description: "",

    currentlyWorking: false,

  });

  // Refs

  const bannerInputRef = useRef<HTMLInputElement>(null);

  const avatarInputRef = useRef<HTMLInputElement>(null);

  useEffect(() => {

    const fetchProfileData = async () => {

      try {

        setIsLoading(true);

        const data = await getProfile(user.id);

        dispatch(setProfile(data));

        setEditableTalent(mapProfileToTalent(data));

      } catch (error) {

        console.error("Error fetching profile:", error);

        toast.error("Failed to load profile data");

      } finally {

        setIsLoading(false);

      }

    };

    if (user?.id) {

      fetchProfileData();

    }

  }, [user?.id, dispatch]);

  useEffect(() => {

    if (talent) {

      setEditableTalent(talent);

    }

  }, [talent]);

  // Map backend profile data to frontend talent format

  const mapProfileToTalent = (profileData: any): Talent => {

    return {

      id: profileData.id,

      name: profileData.name || "",

      email: profileData.email,

      role: profileData.jobTitle || "",

      company: profileData.company || "",

      location: profileData.location || "",

      about: profileData.about || "",

      topSkill: profileData.skills || [],

      salary: profileData.salary || "",

      experience: profileData.experiences?.map((exp: any) => ({

        title: exp.title,

        company: exp.company,

        location: exp.location,

        startDate: exp.startDate,

        endDate: exp.currentlyWorking ? "Present" : exp.endDate,

        description: exp.description,

        currentlyWorking: exp.currentlyWorking

      })) || [],

      certifications: profileData.certifications?.map((cert: any) => ({

        title: cert.name,

        company: cert.issuer,

        location: cert.location,

        issueDate: cert.issueDate,

        certificateID: cert.certificateId

      })) || [],

      avatar: profileData.avatar || "/avatar-7.png",

      banner: profileData.banner || "/color7.png"

    };

  };

  // Map frontend talent data to backend profile format

  const mapTalentToProfile = (talentData: Talent): any => {

    return {

      id: talentData.id,

      email: talentData.email,

      jobTitle: talentData.role,

      company: talentData.company,

      location: talentData.location,

      about: talentData.about,

      skills: talentData.topSkill,

      salary: talentData.salary,

      experiences: talentData.experience?.map(exp => ({

        title: exp.title,

        company: exp.company,

        location: exp.location,

        startDate: exp.startDate,

        endDate: exp.currentlyWorking ? null : exp.endDate,

        currentlyWorking: exp.currentlyWorking,

        description: exp.description

      })) || [],

      certifications: talentData.certifications?.map(cert => ({

        name: cert.title,

        issuer: cert.company,

        location: cert.location,

        issueDate: cert.issueDate,

        certificateId: cert.certificateID

      })) || [],

      avatar: talentData.avatar,

      banner: talentData.banner

    };

  };

  const toggleEdit = (field: keyof EditModes) => {

    setEditMode(prev => ({ ...prev, [field]: !prev[field] }));

  };

  const handleInputChange = (field: string, value: any) => {

    setEditableTalent(prev => prev ? { ...prev, [field]: value } : null);

  };

  // Certification handlers

  const handleCertEdit = (index: number, updatedCert: Certification & { location: string }) => {

    if (!editableTalent?.certifications) return;

    const updatedCerts = [...editableTalent.certifications];

    updatedCerts[index] = {

      title: updatedCert.name,

      company: updatedCert.issuer,

      location: updatedCert.location,

      issueDate: updatedCert.issueDate,

      certificateID: updatedCert.certificateId

    };

    const updatedTalent = { ...editableTalent, certifications: updatedCerts };

    setEditableTalent(updatedTalent);

    handleSave(updatedTalent);

  };

  const handleCertDelete = (index: number) => {

    if (!editableTalent?.certifications) return;

    const updatedCerts = editableTalent.certifications.filter((\_, i) => i !== index);

    const updatedTalent = { ...editableTalent, certifications: updatedCerts };

    setEditableTalent(updatedTalent);

    handleSave(updatedTalent);

  };

  const handleAddCert = () => {

    if (!newCert.name || !newCert.issuer || !editableTalent) {

      toast.error("Please fill all required fields");

      return;

    }

    const certToAdd = {

      title: newCert.name,

      company: newCert.issuer,

      location: newCert.location || "",

      issueDate: newCert.issueDate,

      certificateID: newCert.certificateId

    };

    const updatedCerts = [...(editableTalent.certifications || []), certToAdd];

    const updatedTalent = { ...editableTalent, certifications: updatedCerts };

    setEditableTalent(updatedTalent);

    handleSave(updatedTalent);

    setNewCert({

      name: "",

      issuer: "",

      location: "",

      issueDate: new Date().toISOString().split("T")[0],

      certificateId: "",

    });

    setIsAddingCert(false);

  };

  // Main save function that dispatches to backend

  const handleSave = async (updatedTalent: Talent) => {

    try {

      setIsLoading(true);

      const profileData = mapTalentToProfile(updatedTalent);

      const response = await updateProfileApi(user.id, profileData);

      if (response) {

        dispatch(updateProfile(response));

        toast.success("Profile updated successfully");

      }

    } catch (error) {

      console.error("Error updating profile:", error);

      toast.error("Failed to update profile");

    } finally {

      setIsLoading(false);

    }

  };

  // Experience handlers

  const handleExpEdit = (index: number, updatedExp: Experience) => {

    if (!editableTalent?.experience) return;

    const updatedExps = [...editableTalent.experience];

    updatedExps[index] = {

      ...updatedExp,

      endDate: updatedExp.currentlyWorking ? "Present" : updatedExp.endDate

    };

    const updatedTalent = { ...editableTalent, experience: updatedExps };

    setEditableTalent(updatedTalent);

    handleSave(updatedTalent);

  };

  const handleExpDelete = (index: number) => {

    if (!editableTalent?.experience) return;

    const updatedExps = editableTalent.experience.filter((\_, i) => i !== index);

    const updatedTalent = { ...editableTalent, experience: updatedExps };

    setEditableTalent(updatedTalent);

    handleSave(updatedTalent);

  };

  const handleAddExp = () => {

    if (!newExp.title || !newExp.company || !editableTalent) {

      toast.error("Please fill all required fields");

      return;

    }

    const expToAdd = {

      ...newExp,

      endDate: newExp.currentlyWorking ? "Present" : newExp.endDate

    };

    const updatedExps = [...(editableTalent.experience || []), expToAdd];

    const updatedTalent = { ...editableTalent, experience: updatedExps };

    setEditableTalent(updatedTalent);

    handleSave(updatedTalent);

    setNewExp({

      title: "",

      company: "",

      location: "",

      startDate: new Date().toISOString().split("T")[0],

      endDate: "",

      description: "",

      currentlyWorking: false,

    });

    setIsAddingExp(false);

  };

  // Image upload handlers

  const handleImageUpload = (e: React.ChangeEvent<HTMLInputElement>, type: 'banner' | 'avatar') => {

    if (!e.target.files?.[0] || !editableTalent) return;

    const file = e.target.files[0];

    if (file.size > 2 \* 1024 \* 1024) { // 2MB limit

      toast.error("Image size should be less than 2MB");

      return;

    }

    const reader = new FileReader();

    reader.onload = (event) => {

      if (event.target?.result) {

        const updatedTalent = {

          ...editableTalent,

          [type]: event.target.result as string

        };

        setEditableTalent(updatedTalent);

        handleSave(updatedTalent);

      }

    };

    reader.readAsDataURL(file);

  };

  if (isLoading) {

    return <div className="flex justify-center items-center h-64">

      <div className="animate-spin rounded-full h-12 w-12 border-t-2 border-b-2 border-blue-500"></div>

    </div>;

  }

  if (!editableTalent) {

    return <p className="text-center text-gray-500 text-lg">No profile selected</p>;

  }

  return (

    <div className="w-full max-w-2xl mx-auto bg-white dark:bg-gray-800 shadow-2xl rounded-2xl overflow-hidden p-8 transition-all duration-300 transform hover:scale-[1.02]">

      {/\* Banner and Avatar Section \*/}

      <div className="relative">

        <div className="relative">

          <img

            className="rounded-t-lg w-full h-52 object-cover"

            src={editableTalent.banner as string || "/color7.png"}

            alt="Profile banner"

          />

          {editMode.personal && (

            <button

              onClick={() => bannerInputRef.current?.click()}

              className="absolute top-2 right-2 bg-white/90 hover:bg-white text-gray-800 p-2 rounded-full shadow-md"

              aria-label="Upload banner image"

            >

              <IconUpload size={18} />

              <input

                type="file"

                ref={bannerInputRef}

                onChange={(e) => handleImageUpload(e, 'banner')}

                accept="image/\*"

                className="hidden"

              />

            </button>

          )}

        </div>

        <div className="relative">

          <img

            className="rounded-full h-32 w-32 -bottom-16 absolute left-1/2 transform -translate-x-1/2 border-4 border-white dark:border-gray-800 shadow-lg hover:scale-110 transition-all duration-300"

            src={editableTalent.avatar as string || "/avatar-7.png"}

            alt="Profile avatar"

          />

          {editMode.personal && (

            <button

              onClick={() => avatarInputRef.current?.click()}

              className="absolute -bottom-10 left-1/2 transform translate-x-8 bg-white/90 hover:bg-white text-gray-800 p-2 rounded-full shadow-md"

              aria-label="Upload avatar image"

            >

              <IconUpload size={18} />

              <input

                type="file"

                ref={avatarInputRef}

                onChange={(e) => handleImageUpload(e, 'avatar')}

                accept="image/\*"

                className="hidden"

              />

            </button>

          )}

        </div>

      </div>

      {/\* Profile Details \*/}

      <div className="text-center mt-16">

        {/\* Name \*/}

        <div className="flex items-center justify-center gap-2">

          <h2 className="text-3xl font-bold flex items-center justify-center text-gray-900 dark:text-white">

            <IconUser className="h-7 w-7 mr-2 text-blue-600" stroke={1.5} />

            {editMode.personal ? (

              <input

                type="text"

                className="border rounded-md p-1 text-center"

                value={editableTalent.name}

                onChange={(e) => handleInputChange("name", e.target.value)}

                aria-label="Name"

              />

            ) : (

              editableTalent.name

            )}

          </h2>

          <button

            onClick={() => toggleEdit("personal")}

            className="flex items-center gap-1 text-sm text-blue-600 dark:text-blue-400 hover:underline"

            aria-label={editMode.personal ? "Save personal info" : "Edit personal info"}

          >

            {editMode.personal ? (

              <>

                <IconDeviceFloppy size={18} /> Save

              </>

            ) : (

              <>

                <IconEdit size={18} /> Edit

              </>

            )}

          </button>

        </div>

        {/\* Role and Company \*/}

        <p className="text-gray-600 dark:text-gray-300 flex items-center justify-center mt-2 text-lg">

          <IconBriefcase className="h-5 w-5 mr-1 text-gray-500" stroke={1.5} />

          {editMode.personal ? (

            <div className="flex gap-2">

              <input

                type="text"

                className="border rounded-md p-1 text-center w-32"

                value={editableTalent.role}

                onChange={(e) => handleInputChange("role", e.target.value)}

                aria-label="Role"

              />

              <span>•</span>

              <input

                type="text"

                className="border rounded-md p-1 text-center w-32"

                value={editableTalent.company}

                onChange={(e) => handleInputChange("company", e.target.value)}

                aria-label="Company"

              />

            </div>

          ) : (

            `${editableTalent.role} • ${editableTalent.company}`

          )}

        </p>

        {/\* Location \*/}

        <p className="flex justify-center items-center text-gray-500 dark:text-gray-400 mt-2 text-md">

          <IconMapPin className="h-5 w-5 mr-1 text-red-500" stroke={1.5} />

          {editMode.personal ? (

            <input

              type="text"

              className="border rounded-md p-1 text-center"

              value={editableTalent.location}

              onChange={(e) => handleInputChange("location", e.target.value)}

              aria-label="Location"

            />

          ) : (

            editableTalent.location

          )}

        </p>

        <Divider className="my-6" />

        {/\* About Section \*/}

        <div className="mt-6 px-4">

          <div className="flex items-center justify-between">

            <h3 className="text-lg font-semibold text-gray-800 dark:text-white flex items-center">

              <IconBuildingSkyscraper className="h-5 w-5 mr-2 text-blue-500" stroke={1.5} />

              About

            </h3>

            <button

              onClick={() => toggleEdit("about")}

              className="flex items-center gap-1 text-sm text-blue-600 dark:text-blue-400 hover:underline"

              aria-label={editMode.about ? "Save about section" : "Edit about section"}

            >

              {editMode.about ? (

                <>

                  <IconDeviceFloppy size={18} /> Save

                </>

              ) : (

                <>

                  <IconEdit size={18} /> Edit

                </>

              )}

            </button>

          </div>

          {editMode.about ? (

            <textarea

              className="mt-2 w-full border rounded-md p-2 text-gray-800 dark:text-white dark:bg-gray-700"

              value={editableTalent.about}

              onChange={(e) => handleInputChange("about", e.target.value)}

              rows={4}

              aria-label="About section"

            />

          ) : (

            <p className="mt-2 text-gray-700 dark:text-gray-300 text-md leading-relaxed border-l-4 border-blue-500 pl-4 italic">

              "{editableTalent.about}"

            </p>

          )}

        </div>

        <Divider className="my-6" />

        {/\* Skills Section \*/}

        <div className="mt-6">

          <div className="flex items-center justify-between">

            <h3 className="text-lg font-semibold text-gray-800 dark:text-white flex items-center">

              <IconTools className="h-5 w-5 mr-2 text-blue-500" stroke={1.5} />

              Top Skills

            </h3>

            <button

              onClick={() => toggleEdit("skills")}

              className="flex items-center gap-1 text-sm text-blue-600 dark:text-blue-400 hover:underline"

              aria-label={editMode.skills ? "Save skills" : "Edit skills"}

            >

              {editMode.skills ? (

                <>

                  <IconDeviceFloppy size={18} /> Save

                </>

              ) : (

                <>

                  <IconEdit size={18} /> Edit

                </>

              )}

            </button>

          </div>

          {editMode.skills ? (

            <div className="mt-2">

              <textarea

                className="w-full border rounded-md p-2 text-gray-800 dark:text-white dark:bg-gray-700"

                value={editableTalent.topSkill?.join(", ") || ""}

                onChange={(e) =>

                  handleInputChange("topSkill", e.target.value.split(",").map((s) => s.trim()))

                }

                placeholder="Enter skills separated by commas"

                rows={3}

                aria-label="Skills input"

              />

              <p className="text-xs text-gray-500 dark:text-gray-400 mt-1">

                Separate skills with commas

              </p>

            </div>

          ) : (

            <div className="flex flex-wrap justify-center mt-3 gap-2">

              {editableTalent.topSkill?.map((skill, index) => (

                <span

                  key={index}

                  className="bg-gradient-to-r from-blue-400 to-blue-600 text-white px-4 py-1 rounded-full text-sm font-medium shadow-md hover:shadow-lg transform hover:scale-105 transition-all"

                >

                  {skill}

                </span>

              ))}

            </div>

          )}

        </div>

        {/\* Salary Section \*/}

        <div className="mt-6 text-center">

          <div className="flex justify-center items-center gap-2">

            <h3 className="text-xl font-semibold text-green-700 dark:text-green-400 flex items-center">

              <IconCurrencyRupee className="h-6 w-6 mr-1 text-green-600" stroke={1.5} />

              Salary:

            </h3>

            <button

              onClick={() => toggleEdit("salary")}

              className="flex items-center gap-1 text-sm text-blue-600 dark:text-blue-400 hover:underline"

              aria-label={editMode.salary ? "Save salary" : "Edit salary"}

            >

              {editMode.salary ? (

                <>

                  <IconDeviceFloppy size={18} /> Save

                </>

              ) : (

                <>

                  <IconEdit size={18} /> Edit

                </>

              )}

            </button>

          </div>

          {editMode.salary ? (

            <div className="mt-2">

              <input

                type="text"

                className="px-4 py-2 border rounded-md text-center text-lg w-48"

                value={editableTalent.salary}

                onChange={(e) => handleInputChange("salary", e.target.value)}

                aria-label="Salary input"

              />

            </div>

          ) : (

            <p className="text-xl text-green-700 dark:text-green-400 mt-1">

              {editableTalent.salary}

            </p>

          )}

        </div>

        {/\* Experience Section \*/}

        <div className="mt-6 px-3">

          <div className="flex justify-between items-center">

            <h3 className="text-2xl font-semibold text-gray-800 dark:text-white flex items-center">

              <IconBriefcase className="h-6 w-6 mr-2 text-blue-500" stroke={1.5} />

              Experience

            </h3>

            <button

              onClick={() => setIsAddingExp(true)}

              className="flex items-center gap-1 text-sm bg-blue-600 hover:bg-blue-700 text-white px-3 py-1 rounded-full"

              aria-label="Add experience"

            >

              <IconPlus size={16} /> Add Experience

            </button>

          </div>

          {isAddingExp && (

            <div className="mt-4 p-4 bg-gray-100 dark:bg-gray-700 rounded-lg">

              <div className="space-y-3">

                <div className="flex gap-2">

                  <input

                    className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-600"

                    value={newExp.title}

                    onChange={(e) => setNewExp({ ...newExp, title: e.target.value })}

                    placeholder="Job Title"

                    aria-label="Job title"

                  />

                  <input

                    className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-600"

                    value={newExp.company}

                    onChange={(e) => setNewExp({ ...newExp, company: e.target.value })}

                    placeholder="Company"

                    aria-label="Company"

                  />

                </div>

                <input

                  className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-600"

                  value={newExp.location}

                  onChange={(e) => setNewExp({ ...newExp, location: e.target.value })}

                  placeholder="Location"

                  aria-label="Location"

                />

                <div className="flex gap-2 items-center">

                  <input

                    type="date"

                    className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-600"

                    value={newExp.startDate}

                    onChange={(e) => setNewExp({ ...newExp, startDate: e.target.value })}

                    aria-label="Start date"

                  />

                  {!newExp.currentlyWorking && (

                    <input

                      type="date"

                      className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-600"

                      value={newExp.endDate === "Present" ? "" : newExp.endDate}

                      onChange={(e) => setNewExp({ ...newExp, endDate: e.target.value })}

                      aria-label="End date"

                    />

                  )}

                  <label className="flex items-center gap-2 whitespace-nowrap">

                    <input

                      type="checkbox"

                      checked={newExp.currentlyWorking}

                      onChange={(e) => {

                        setNewExp({

                          ...newExp,

                          currentlyWorking: e.target.checked,

                          endDate: e.target.checked ? "Present" : newExp.endDate

                        });

                      }}

                      aria-label="Currently working here"

                    />

                    Currently Working

                  </label>

                </div>

                <textarea

                  className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-600"

                  value={newExp.description}

                  onChange={(e) => setNewExp({ ...newExp, description: e.target.value })}

                  placeholder="Job description"

                  rows={4}

                  aria-label="Job description"

                />

                <div className="flex justify-end gap-2">

                  <button

                    onClick={() => setIsAddingExp(false)}

                    className="px-3 py-1 bg-gray-300 dark:bg-gray-600 rounded-md"

                    aria-label="Cancel adding experience"

                  >

                    Cancel

                  </button>

                  <button

                    onClick={handleAddExp}

                    className="px-3 py-1 bg-blue-600 text-white rounded-md"

                    aria-label="Save experience"

                  >

                    Save

                  </button>

                </div>

              </div>

            </div>

          )}

          <div className="flex flex-col gap-4 mt-4">

            {editableTalent.experience && editableTalent.experience.length > 0 ? (

              editableTalent.experience.map((exp, index) => (

                <ExpCard

                  key={index}

                  {...exp}

                  onEdit={(updatedExp) => handleExpEdit(index, updatedExp)}

                  onDelete={() => handleExpDelete(index)}

                  editable                />

              ))

            ) : (

              <p className="text-gray-500 dark:text-gray-400 text-center py-4">

                No experience details available.

              </p>

            )}

          </div>

        </div>

        {/\* Certification Section \*/}

        <div className="mt-6 px-3">

        <div className="flex justify-between items-center">

          <h3 className="text-2xl font-semibold text-gray-800 dark:text-white flex items-center">

            <IconCertificate className="h-6 w-6 mr-2 text-blue-500" stroke={1.5} />

            Certifications

          </h3>

          <button

            onClick={() => setIsAddingCert(true)}

            className="flex items-center gap-1 text-sm bg-blue-600 hover:bg-blue-700 text-white px-3 py-1 rounded-full"

            aria-label="Add certification"

          >

            <IconPlus size={16} /> Add Certification

          </button>

        </div>

        {isAddingCert && (

          <div className="mt-4 p-4 bg-gray-100 dark:bg-gray-700 rounded-lg">

            <div className="space-y-3">

              <input

                className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-600"

                value={newCert.name}

                onChange={(e) => setNewCert({ ...newCert, name: e.target.value })}

                placeholder="Certification Name\*"

                required

                aria-label="Certification name"

              />

              <input

                className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-600"

                value={newCert.issuer}

                onChange={(e) => setNewCert({ ...newCert, issuer: e.target.value })}

                placeholder="Issuing Organization\*"

                required

                aria-label="Issuing organization"

              />

              <input

                className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-600"

                value={newCert.location || ""}

                onChange={(e) => setNewCert({ ...newCert, location: e.target.value })}

                placeholder="Location (optional)"

                aria-label="Location"

              />

              <div className="flex gap-2">

                <input

                  type="date"

                  className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-600"

                  value={newCert.issueDate}

                  onChange={(e) => setNewCert({ ...newCert, issueDate: e.target.value })}

                  required

                  aria-label="Issue date"

                />

                <input

                  className="w-full p-2 border rounded-md text-gray-900 dark:text-white dark:bg-gray-600"

                  value={newCert.certificateId}

                  onChange={(e) => setNewCert({ ...newCert, certificateId: e.target.value })}

                  placeholder="Certificate ID\*"

                  required

                  aria-label="Certificate ID"

                />

              </div>

              <div className="flex justify-end gap-2">

                <button

                  onClick={() => setIsAddingCert(false)}

                  className="px-3 py-1 bg-gray-300 dark:bg-gray-600 rounded-md"

                  aria-label="Cancel adding certification"

                >

                  Cancel

                </button>

                <button

                  onClick={handleAddCert}

                  className="px-3 py-1 bg-blue-600 text-white rounded-md"

                  aria-label="Save certification"

                >

                  Save

                </button>

              </div>

            </div>

          </div>

        )}

        <div className="flex flex-col gap-4 mt-4">

          {editableTalent.certifications && editableTalent.certifications.length > 0 ? (

            editableTalent.certifications.map((cert, index) => (

              <CertiCard

                key={index}

                title={cert.title}

                company={cert.company}

                location={cert.location}

                issueDate={cert.issueDate}

                certificateID={cert.certificateID}

                onEdit={(updatedCert) =>

                  handleCertEdit(index, {

                    name: updatedCert.title,

                    issuer: updatedCert.company,

                    location: updatedCert.location || "",

                    issueDate: updatedCert.issueDate,

                    certificateId: updatedCert.certificateID

                  })

                }

                onDelete={() => handleCertDelete(index)}

                editable

              />

            ))

          ) : (

            <p className="text-gray-500 dark:text-gray-400 text-center py-4">

              No certifications available.

            </p>

          )}

        </div>

      </div>

        {/\* Back Button \*/}

        <div className="mt-8 flex justify-center">

          <button

            className="py-2 px-6 bg-gray-600 hover:bg-gray-700 text-white text-lg rounded-lg shadow-lg transform hover:scale-110 transition-all"

            onClick={onBack}

            aria-label="Go back"

          >

            Back

          </button>

        </div>

      </div>

    </div>

  );

};

export default Profile;

import React, { useState } from "react";

import { InputBase, Combobox, useCombobox, CloseButton, Button } from "@mantine/core";

import { motion } from "framer-motion";

interface SelectInputProps {

    label: string;

    placeholder: string;

    options: string[];

}

export const SelectInput: React.FC<SelectInputProps> = ({

    label,

    placeholder,

    options,

}) => {

    const combobox = useCombobox({

        onDropdownClose: () => combobox.resetSelectedOption(),

    });

    const [selectedValues, setSelectedValues] = useState<string[]>([]);

    const [search, setSearch] = useState("");

    const filteredOptions = options.filter(

        (item) =>

            item.toLowerCase().includes(search.toLowerCase().trim()) &&

            !selectedValues.includes(item)

    );

    const handleSelect = (val: string) => {

        if (!selectedValues.includes(val)) {

            setSelectedValues([...selectedValues, val]);

        }

        setSearch("");

        combobox.closeDropdown();

    };

    const handleRemove = (val: string) => {

        setSelectedValues(selectedValues.filter((item) => item !== val));

    };

    const handleClearAll = () => {

        setSelectedValues([]);

    };

    return (

        <div className="flex flex-col w-full space-y-4">

            <label className="text-lg font-semibold text-indigo-800">

                {label} <span className="text-red-600">\*</span>

            </label>

            <div className="flex flex-wrap gap-2 items-center w-full">

                {selectedValues.length <= 2 ? (

                    selectedValues.map((skill) => (

                        <div

                            key={skill}

                            className="flex items-center bg-indigo-200 text-indigo-900 px-3 py-1 rounded-lg text-sm w-auto"

                        >

                            {skill}

                            <CloseButton

                                onClick={() => handleRemove(skill)}

                                className="ml-2 text-indigo-600 hover:text-indigo-800"

                                size="xs"

                            />

                        </div>

                    ))

                ) : (

                    <div className="flex items-center bg-indigo-200 text-indigo-900 px-3 py-1 rounded-lg text-sm w-auto">

                        {selectedValues.slice(0, 2).join(", ")} +{selectedValues.length - 2}

                    </div>

                )}

                {selectedValues.length > 0 && (

                    <Button

                        variant="outline"

                        size="xs"

                        color="red"

                        onClick={handleClearAll}

                        className="ml-auto text-red-600 border-red-600 hover:bg-red-100"

                    >

                        Clear All

                    </Button>

                )}

            </div>

            <Combobox store={combobox} onOptionSubmit={handleSelect}>

                <Combobox.Target>

                    <InputBase

                        required

                        rightSection={<span className="text-indigo-600">▼</span>}

                        rightSectionPointerEvents="none"

                        onClick={() => combobox.openDropdown()}

                        onFocus={() => combobox.openDropdown()}

                        onBlur={() => combobox.closeDropdown()}

                        placeholder={placeholder}

                        value={search}

                        onChange={(event) => setSearch(event.currentTarget.value)}

                        className="p-3 text-base border border-gray-400 rounded-lg shadow-md focus:ring-2 focus:ring-indigo-500 transition-all duration-200 bg-white text-gray-800 placeholder-gray-500 w-full"

                    />

                </Combobox.Target>

                {combobox.dropdownOpened && (

                    <Combobox.Dropdown>

                        <motion.div

                            initial={{ opacity: 0, y: -10 }}

                            animate={{ opacity: 1, y: 0 }}

                            exit={{ opacity: 0, y: -10 }}

                            className="mt-1 bg-white border border-gray-300 rounded-lg shadow-lg overflow-hidden"

                        >

                            <div className="max-h-40 overflow-y-auto scrollbar-thin scrollbar-thumb-indigo-400 scrollbar-track-gray-200">

                                <Combobox.Options>

                                    {filteredOptions.length > 0 ? (

                                        filteredOptions.map((item) => (

                                            <Combobox.Option

                                                key={item}

                                                value={item}

                                                className="px-4 py-2 text-gray-800 hover:bg-indigo-100 hover:text-indigo-900 cursor-pointer transition-all"

                                            >

                                                {item}

                                            </Combobox.Option>

                                        ))

                                    ) : (

                                        <Combobox.Empty className="px-4 py-2 text-gray-500">

                                            No results found

                                        </Combobox.Empty>

                                    )}

                                </Combobox.Options>

                            </div>

                        </motion.div>

                    </Combobox.Dropdown>

                )}

            </Combobox>

        </div>

    );

};

export default SelectInput;

import React, { useState } from "react";

import {

  TextInput,

  PasswordInput,

  Button,

  Checkbox,

  Anchor,

  Modal,

  Text,

  LoadingOverlay,

} from "@mantine/core";

import { IconAt, IconLock } from "@tabler/icons-react";

import { useNavigate } from "react-router-dom";

import { useDispatch } from "react-redux";

import UserService from "../../Services/UserService";

import ResetPassword from "./ResetPassword";

import { setUser } from "../../Slices/UserSlice";

function Login() {

  const dispatch = useDispatch();

  const navigate = useNavigate();

  // 🔐 Login States

  const [email, setEmail] = useState("");

  const [password, setPassword] = useState("");

  const [acceptedTerms, setAcceptedTerms] = useState(false);

  const [loading, setLoading] = useState(false);

  // 🔁 Forgot Password States

  const [forgotPasswordModalOpen, setForgotPasswordModalOpen] = useState(false);

  const [resetEmail, setResetEmail] = useState("");

  const [resetLoading, setResetLoading] = useState(false);

  const [resetMessage, setResetMessage] = useState("");

  const [showResetFlow, setShowResetFlow] = useState(false);

  // 🧠 Handle Login

  const handleSubmit = async (e: React.FormEvent) => {

    e.preventDefault();

    setLoading(true);

    if (!email || !password) {

      alert("Please fill in all fields.");

      setLoading(false);

      return;

    }

    if (!acceptedTerms) {

      alert("Please accept the Terms & Conditions.");

      setLoading(false);

      return;

    }

    try {

      const res = await UserService.loginUser({ email, password });

      if (res && typeof res === "object" && res.user) {

        dispatch(setUser(res.user));

        localStorage.setItem("currentUser", JSON.stringify(res.user));

        alert("Login Successful!");

        navigate("/Homepage");

      } else {

        console.error("Unexpected Login Response:", res);

        alert(res?.message || "Unexpected response from server. Please contact support.");

      }

    } catch (error: any) {

      console.error("Login Error:", error);

      const message = error?.response?.data?.message || error?.message || "Login failed. Please try again.";

      alert(message);

    } finally {

      setLoading(false);

    }

  };

  // 🔁 Handle Forgot Password

  const handleForgotPassword = async () => {

    if (!resetEmail) {

      setResetMessage("Please enter your email address.");

      return;

    }

    try {

      setResetLoading(true);

      const response = await UserService.sendOtp(resetEmail);

      setResetMessage(response?.message || "OTP sent successfully.");

      setForgotPasswordModalOpen(false); // Close email modal

      setShowResetFlow(true); // Open full reset modal

    } catch (error: any) {

      console.error("Forgot Password OTP Error:", error);

      const message = error?.response?.data?.message || error?.message || "Error while sending OTP.";

      setResetMessage(message);

    } finally {

      setResetLoading(false);

    }

  };

  return (

    <div style={{ position: "relative" }}>

      <LoadingOverlay

        visible={loading}

        zIndex={1000}

        overlayProps={{ radius: "sm", blur: 2 }}

        loaderProps={{ color: "pink", type: "bars" }}

      />

      {/\* 🔐 Login Form \*/}

      <form

        onSubmit={handleSubmit}

        className="w-full max-w-md bg-[#1f1f2e] p-8 rounded-xl shadow-lg border border-yellow-500 mx-auto"

      >

        <h2 className="text-3xl text-yellow-400 font-bold text-center mb-6">Sign In</h2>

        <TextInput

          label="Email"

          placeholder="Enter your email"

          value={email}

          onChange={(e) => setEmail(e.currentTarget.value)}

          leftSection={<IconAt size={18} className="text-yellow-300" />}

          styles={{

            input: { backgroundColor: "#2a2a3a", color: "#fefce8" },

            label: { color: "#ffeb8a" },

          }}

          className="mb-4"

          required

        />

        <PasswordInput

          label="Password"

          placeholder="Enter your password"

          value={password}

          onChange={(e) => setPassword(e.currentTarget.value)}

          leftSection={<IconLock size={18} className="text-yellow-300" />}

          styles={{

            input: { backgroundColor: "#2a2a3a", color: "#fefce8" },

            label: { color: "#ffeb8a" },

          }}

          className="mb-6"

          required

        />

        <Checkbox

          checked={acceptedTerms}

          onChange={(e) => setAcceptedTerms(e.currentTarget.checked)}

          label={

            <span className="text-yellow-300">

              I accept{" "}

              <Anchor href="#" className="text-yellow-400 hover:underline">

                Terms & Conditions

              </Anchor>

            </span>

          }

          className="mb-6"

        />

        <Button

          type="submit"

          fullWidth

          loading={loading}

          className="bg-yellow-500 hover:bg-yellow-600 text-gray-900 font-bold"

        >

          Login

        </Button>

        <Anchor

          onClick={() => {

            setResetMessage("");

            setForgotPasswordModalOpen(true);

          }}

          className="text-yellow-300 hover:underline mt-4 block text-center cursor-pointer"

        >

          Forgot Password?

        </Anchor>

      </form>

      {/\* 🔁 Forgot Password Email Modal \*/}

      <Modal

        opened={forgotPasswordModalOpen}

        onClose={() => {

          setForgotPasswordModalOpen(false);

          setResetEmail("");

          setResetMessage("");

        }}

        title="Reset Password"

        centered

      >

        <TextInput

          label="Enter your email"

          placeholder="Your email address"

          value={resetEmail}

          onChange={(e) => setResetEmail(e.currentTarget.value)}

          leftSection={<IconAt size={18} className="text-yellow-300" />}

          styles={{

            input: { backgroundColor: "#2a2a3a", color: "#fefce8" },

            label: { color: "#ffeb8a" },

          }}

          required

        />

        {resetMessage && (

          <Text className="mt-2 text-yellow-400 font-medium">{resetMessage}</Text>

        )}

        <Button

          fullWidth

          loading={resetLoading}

          onClick={handleForgotPassword}

          className="bg-yellow-500 hover:bg-yellow-600 text-gray-900 font-bold mt-4"

        >

          Send OTP

        </Button>

      </Modal>

      {/\* 🔁 Full Reset Flow Modal with OTP & New Password \*/}

      {showResetFlow && resetEmail && (

        <ResetPassword

          opened={showResetFlow}

          close={() => {

            setShowResetFlow(false);

            setResetEmail("");

          }}

          redirectToLogin={true}

          initialEmail={resetEmail}

        />

      )}

    </div>

  );

}

export default Login;

// ResetPassword.tsx

// 🔐 Improved Modal Component to reset password via Email → OTP → New Password

import React, { useState, useEffect } from "react";

import {

  Modal,

  TextInput,

  Button,

  PinInput,

  PasswordInput,

  Text,

} from "@mantine/core";

import { IconAt, IconLock } from "@tabler/icons-react";

import UserService from "../../Services/UserService";

import { successNotification, errorNotification } from "../../Services/NotificationService";

import { useNavigate } from "react-router-dom";

interface ResetPasswordProps {

  opened: boolean;

  close: () => void;

  redirectToLogin?: boolean;

  initialEmail?: string; // 🌟 Add to retain email from Login

}

const ResetPassword: React.FC<ResetPasswordProps> = ({

  opened,

  close,

  redirectToLogin = true,

  initialEmail = "",

}) => {

  const [email, setEmail] = useState(initialEmail);

  const [otp, setOtp] = useState("");

  const [password, setPassword] = useState("");

  const [otpSent, setOtpSent] = useState(false);

  const [verified, setVerified] = useState(false);

  const [feedback, setFeedback] = useState("");

  const [loading, setLoading] = useState(false);

  const [resendTimer, setResendTimer] = useState(0);

  const navigate = useNavigate();

  useEffect(() => {

    let timer: NodeJS.Timeout;

    if (resendTimer > 0) {

      timer = setTimeout(() => setResendTimer(resendTimer - 1), 1000);

    }

    return () => clearTimeout(timer);

  }, [resendTimer]);

  useEffect(() => {

    setEmail(initialEmail); // update if new email passed

  }, [initialEmail]);

  useEffect(() => {

    if (!opened) {

      // 💥 Reset only when modal closes to preserve flow

      setOtp("");

      setOtpSent(false);

      setVerified(false);

      setPassword("");

      setFeedback("");

      setResendTimer(0);

    }

  }, [opened]);

  const handleSendOtp = async () => {

    if (!email) {

      setFeedback("Please enter your email address.");

      return;

    }

    try {

      setLoading(true);

      const res = await UserService.sendOtp(email);

      setFeedback(res?.message || "OTP sent to your email.");

      setOtpSent(true);

      setResendTimer(60);

      successNotification("OTP Sent", res?.message || "Check your inbox!");

    } catch (error: any) {

      console.error("Send OTP Error:", error);

      const message = error?.response?.data?.message || error?.message || "Error sending OTP. Please try again.";

      if (message.includes("non unique result")) {

        setFeedback("Multiple accounts found with this email. Contact support.");

      } else {

        setFeedback(message);

      }

      errorNotification("Send OTP Failed", message);

    } finally {

      setLoading(false);

    }

  };

  const handleVerifyOtp = async () => {

    if (!otp) {

      setFeedback("Please enter the OTP you received.");

      return;

    }

    try {

      setLoading(true);

      const res = await UserService.verifyOtp(email, otp);

      setFeedback(res?.message || "OTP verified. You can now set a new password.");

      setVerified(true);

      successNotification("OTP Verified", res?.message || "Set your new password.");

    } catch (error: any) {

      console.error("Verify OTP Error:", error);

      const message = error?.response?.data?.message || error?.message || "OTP verification failed. Please try again.";

      setFeedback(message);

      errorNotification("OTP Verification Failed", message);

    } finally {

      setLoading(false);

    }

  };

  const handleResetPassword = async () => {

    if (!verified) {

      setFeedback("Please verify your OTP before resetting the password.");

      return;

    }

    if (!email) {

      setFeedback("Email is missing. Please start the reset process again.");

      return;

    }

    if (!password || password.length < 8) {

      setFeedback("Password must be at least 8 characters long.");

      return;

    }

    try {

      setLoading(true);

      const res = await UserService.changePass(email, password);

      successNotification("Password Reset", res?.message || "Your password has been updated.");

      setFeedback(res?.message || "Your password has been updated.");

      setTimeout(() => {

        close();

        if (redirectToLogin) {

          navigate("/login");

        }

      }, 1500);

    } catch (error: any) {

      console.error("Password Reset Error:", error);

      const message = error?.response?.data?.message || error?.message || "Error resetting password. Please try again.";

      if (message.includes("non unique result")) {

        setFeedback("Multiple accounts found with this email. Contact support.");

      } else {

        setFeedback(message);

      }

      errorNotification("Password Reset Failed", message);

    } finally {

      setLoading(false);

    }

  };

  return (

    <Modal opened={opened} onClose={close} title="Reset Password" centered>

      <TextInput

        label="Email"

        placeholder="Enter your email"

        value={email}

        onChange={(e) => setEmail(e.currentTarget.value)}

        leftSection={<IconAt size={18} className="text-yellow-300" />}

        styles={{

          input: { backgroundColor: "#2a2a3a", color: "#fefce8" },

          label: { color: "#ffeb8a" },

        }}

        className="mb-4"

        required

        disabled={otpSent}

      />

      {feedback && <Text className="text-yellow-300 mb-3">{feedback}</Text>}

      {!otpSent && (

        <Button

          fullWidth

          loading={loading}

          onClick={handleSendOtp}

          className="bg-yellow-500 hover:bg-yellow-600 text-gray-900 font-bold mb-4"

        >

          Send OTP

        </Button>

      )}

      {otpSent && !verified && (

        <>

          <PinInput

            length={6}

            value={otp}

            onChange={setOtp}

            type="number"

            size="md"

            gap="lg"

            className="mx-auto mb-3"

          />

          <Button

            fullWidth

            loading={loading}

            onClick={handleVerifyOtp}

            className="bg-yellow-500 hover:bg-yellow-600 text-gray-900 font-bold mb-2"

          >

            Verify OTP

          </Button>

          <Button

            fullWidth

            variant="outline"

            onClick={handleSendOtp}

            disabled={loading || resendTimer > 0}

            className="border-yellow-500 text-yellow-400 hover:bg-yellow-900 mb-4"

          >

            {resendTimer > 0 ? `Resend OTP (${resendTimer}s)` : "Resend OTP"}

          </Button>

        </>

      )}

      {verified && (

        <>

          <PasswordInput

            label="New Password"

            placeholder="Enter your new password"

            value={password}

            onChange={(e) => setPassword(e.currentTarget.value)}

            leftSection={<IconLock size={18} className="text-yellow-300" />}

            styles={{

              input: { backgroundColor: "#2a2a3a", color: "#fefce8" },

              label: { color: "#ffeb8a" },

            }}

            className="mb-4"

            required

          />

          <Button

            fullWidth

            loading={loading}

            onClick={handleResetPassword}

            className="bg-yellow-500 hover:bg-yellow-600 text-gray-900 font-bold mb-4"

          >

            Reset Password

          </Button>

        </>

      )}

    </Modal>

  );

};

export default ResetPassword;

// SignUp.tsx

// 🎯 User Registration Component with validation, error handling, and clean design.

import React, { useState } from "react";

import {

  PasswordInput,

  TextInput,

  Button,

  Checkbox,

  Anchor,

  Radio,

  Group,

  LoadingOverlay,  // ✅ Add this import

} from "@mantine/core";

import { IconAt, IconLock, IconUser } from "@tabler/icons-react";

import UserService from "../../Services/UserService";

import { successNotification, errorNotification } from "../../Services/NotificationService";

interface Props {

  onSignupSuccess: () => void;

}

function SignUp({ onSignupSuccess }: Props) {

  const [name, setName] = useState("");

  const [email, setEmail] = useState("");

  const [password, setPassword] = useState("");

  const [confirmPassword, setConfirmPassword] = useState("");

  const [accountType, setAccountType] = useState<"APPLICANT" | "EMPLOYER">("APPLICANT");

  const [acceptedTerms, setAcceptedTerms] = useState(false);

  const [loading, setLoading] = useState(false); // ✅ Keep only this

  const [submissionError, setSubmissionError] = useState("");

  const [errors, setErrors] = useState({

    name: "",

    email: "",

    password: "",

    confirmPassword: "",

  });

  const validateForm = () => {

    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

    const isPasswordValid = password.length >= 8;

    const passwordsMatch = password === confirmPassword;

    const newErrors = {

      name: name.trim() ? "" : "Full name is required.",

      email: emailRegex.test(email) ? "" : "Invalid email format.",

      password: isPasswordValid ? "" : "Password must be at least 8 characters.",

      confirmPassword: passwordsMatch ? "" : "Passwords do not match.",

    };

    setErrors(newErrors);

    return Object.values(newErrors).every((err) => err === "");

  };

  const handleSubmit = async (e: React.FormEvent) => {

    e.preventDefault();

    setSubmissionError("");

    if (!acceptedTerms) {

      errorNotification("Terms & Conditions", "You must accept the terms to proceed.");

      return;

    }

    if (!validateForm()) return;

    const formData = { name, email, password, accountType };

    setLoading(true);

    try {

      const res = await UserService.registerUser(formData);

      successNotification("Account Created", res?.message || "Your account was created successfully!");

      onSignupSuccess();

    } catch (error: any) {

      console.error("Signup Error:", error);

      const message = error?.response?.data?.message || error?.message || "An error occurred. Please check your input or try again later.";

      if (error.response?.status === 409 || message.toLowerCase().includes("already exists")) {

        setSubmissionError("An account with this email already exists.");

      } else {

        setSubmissionError(message);

      }

      errorNotification("Signup Failed", message);

    } finally {

      setLoading(false);

    }

  };

  return (

    <>

      <LoadingOverlay

        visible={loading}

        zIndex={1000}

        overlayProps={{ radius: 'sm', blur: 2 }}

        loaderProps={{ color: 'pink', type: 'bars' }}

      />

      <form

        onSubmit={handleSubmit}

        className="w-full max-w-lg mx-auto bg-[#1f1f2e] p-10 rounded-2xl shadow-2xl border border-yellow-500 space-y-6"

      >

        <h2 className="text-3xl text-yellow-400 font-extrabold text-center tracking-wide">

          Create Your Account

        </h2>

        {submissionError && (

          <div className="text-red-400 text-center font-semibold">

            {submissionError}

          </div>

        )}

        <TextInput

          label="Full Name"

          placeholder="Enter your full name"

          value={name}

          onChange={(e) => setName(e.currentTarget.value)}

          error={errors.name}

          leftSection={<IconUser size={18} className="text-yellow-300" />}

          styles={{

            input: { backgroundColor: "#2a2a3a", color: "#fefce8", borderColor: "#444" },

            label: { color: "#ffeb8a", fontWeight: 600 },

          }}

          required

        />

        <TextInput

          label="Email"

          placeholder="Enter your email"

          value={email}

          onChange={(e) => setEmail(e.currentTarget.value)}

          error={errors.email}

          leftSection={<IconAt size={18} className="text-yellow-300" />}

          styles={{

            input: { backgroundColor: "#2a2a3a", color: "#fefce8", borderColor: "#444" },

            label: { color: "#ffeb8a", fontWeight: 600 },

          }}

          required

        />

        <PasswordInput

          label="Password"

          placeholder="Create a password"

          value={password}

          onChange={(e) => setPassword(e.currentTarget.value)}

          error={errors.password}

          leftSection={<IconLock size={18} className="text-yellow-300" />}

          styles={{

            input: { backgroundColor: "#2a2a3a", color: "#fefce8", borderColor: "#444" },

            label: { color: "#ffeb8a", fontWeight: 600 },

          }}

          required

        />

        <PasswordInput

          label="Confirm Password"

          placeholder="Confirm your password"

          value={confirmPassword}

          onChange={(e) => setConfirmPassword(e.currentTarget.value)}

          error={errors.confirmPassword}

          leftSection={<IconLock size={18} className="text-yellow-300" />}

          styles={{

            input: { backgroundColor: "#2a2a3a", color: "#fefce8", borderColor: "#444" },

            label: { color: "#ffeb8a", fontWeight: 600 },

          }}

          required

        />

        <Radio.Group

          label="Select Account Type"

          value={accountType}

          onChange={(value) => setAccountType(value as "APPLICANT" | "EMPLOYER")}

          styles={{ label: { color: "#ffeb8a", fontWeight: 700, fontSize: "1.1rem" } }}

        >

          <Group mt="xs" className="gap-4">

            <Radio

              value="APPLICANT"

              label="Applicant"

              className={`px-6 py-3 rounded-lg ${

                accountType === "APPLICANT"

                  ? "border-4 border-yellow-400 bg-yellow-200 text-black"

                  : "border border-slate-700 bg-slate-800 hover:bg-slate-700 text-white"

              }`}

            />

            <Radio

              value="EMPLOYER"

              label="Employer"

              className={`px-6 py-3 rounded-lg ${

                accountType === "EMPLOYER"

                  ? "border-4 border-yellow-400 bg-yellow-200 text-black"

                  : "border border-slate-700 bg-slate-800 hover:bg-slate-700 text-white"

              }`}

            />

          </Group>

        </Radio.Group>

        <Checkbox

          checked={acceptedTerms}

          onChange={(e) => setAcceptedTerms(e.currentTarget.checked)}

          label={

            <span className="text-yellow-300 text-sm">

              I accept{" "}

              <Anchor href="#" className="text-yellow-400 underline">

                Terms & Conditions

              </Anchor>

            </span>

          }

        />

        <Button

          type="submit"

          fullWidth

          loading={loading}

          size="md"

          radius="md"

          className="bg-yellow-500 hover:bg-yellow-600 text-gray-900 font-bold transition duration-200"

        >

          Sign Up

        </Button>

      </form>

    </>

  );

}

export default SignUp;

import React from "react";

interface CertiCardProps {

  title: string;

  company: string;

  location?: string;

  issueDate: string;

  certificateID: string;

}

export function CertiCard({

  title,

  company,

  location = "Global",

  issueDate,

  certificateID,

}: CertiCardProps) {

  const formattedCompany = company.toLowerCase().replace(/\s+/g, "-");

  return (

    <div className="p-6 bg-gray-900 text-white shadow-lg rounded-2xl border border-gray-700 flex flex-col gap-4 transition-all hover:shadow-2xl hover:scale-105">

      {/\* Header Section \*/}

      <div className="flex gap-4 items-center">

        {/\* Company Logo \*/}

        <div className="p-3 bg-gray-800 rounded-xl flex items-center justify-center">

          <img

            className="h-16 w-16 object-contain rounded-lg aspect-square"

            src={`/logos/${formattedCompany}.jpeg`}

            alt={`${company} logo`}

            onError={(e) => {

              const target = e.target as HTMLImageElement;

              target.src = "/logos/default.jpeg"; // Fallback image

            }}

          />

        </div>

        {/\* Certification Info \*/}

        <div>

          <h3 className="text-xl font-semibold text-white">{title}</h3>

          <p className="text-gray-400 text-sm">{company} • {location}</p>

          <p className="text-gray-400 text-sm">Issued: {issueDate}</p>

          <p className="text-gray-400 text-sm">ID: {certificateID}</p>

        </div>

      </div>

    </div>

  );

}

import React from 'react';

interface ExpCardProps {

  title: string;

  company: string;

  location: string;

  startDate: string;

  endDate: string;

  description: string;

}

const ExpCard: React.FC<ExpCardProps> = ({ title, company, location, startDate, endDate, description }) => {

  return (

    <div className="p-6 bg-gray-900 text-white shadow-lg rounded-xl border border-gray-700 flex flex-col gap-4 transition-all hover:shadow-2xl hover:scale-105">

      <h3 className="text-xl font-semibold">{title}</h3>

      <p className="text-gray-400">{company} - {location}</p>

      <p className="text-sm text-gray-300">{startDate} - {endDate}</p>

      <p className="text-gray-300 text-sm mt-2">{description}</p>

    </div>

  );

};

export default ExpCard;

import { IconMapPin, IconMessage, IconBriefcase, IconCurrencyRupee, IconUser, IconCertificate, IconTools, IconBuildingSkyscraper } from '@tabler/icons-react';

import React from 'react';

import { Talent } from '../FindTalent/TalentData';

import { Divider } from '@mantine/core';

import ExpCard from './ExpCard';

import { CertiCard } from './CertiCard';

interface ProfileProps {

  talent: Talent | null;

  onBack: () => void;

}

const Profile: React.FC<ProfileProps> = ({ talent, onBack }) => {

  if (!talent) {

    return <p className="text-center text-gray-500 text-lg">No profile selected</p>;

  }

  return (

    <div className="w-full max-w-2xl mx-auto bg-white shadow-2xl rounded-2xl overflow-hidden p-8 transition-all duration-300 transform hover:scale-[1.02]">

      {/\* Banner and Avatar Section \*/}

      <div className="relative">

        <img className="rounded-t-lg w-full h-52 object-cover" src="/color7.png" alt="Banner" />

        <img

          className="rounded-full h-32 w-32 -bottom-16 absolute left-1/2 transform -translate-x-1/2 border-4 border-white shadow-lg hover:scale-110 transition-all duration-300"

          src="/avatar-7.png"

          alt="Avatar"

        />

      </div>

      {/\* Profile Details \*/}

      <div className="text-center mt-16">

        <h2 className="text-3xl font-bold flex items-center justify-center text-gray-900">

          <IconBriefcase className="h-7 w-7 mr-2 text-blue-600" stroke={1.5} />

          {talent.name}

        </h2>

        <p className="text-gray-600 flex items-center justify-center mt-2 text-lg">

          <IconUser className="h-5 w-5 mr-1 text-gray-500" stroke={1.5} /> {talent.role} • {talent.company}

        </p>

        <p className="flex justify-center items-center text-gray-500 mt-2 text-md">

          <IconMapPin className="h-5 w-5 mr-1 text-red-500" stroke={1.5} /> {talent.location}

        </p>

        <Divider className="my-6" />

        {/\* About Section \*/}

        <div className="mt-6 px-4">

          <h3 className="text-lg font-semibold text-gray-800 flex items-center">

            <IconBuildingSkyscraper className="h-5 w-5 mr-2 text-blue-500" stroke={1.5} />

            About:

          </h3>

          <p className="mt-2 text-gray-700 text-md leading-relaxed border-l-4 border-blue-500 pl-4 italic">

            "{talent.about}"

          </p>

        </div>

        <Divider className="my-6" />

        {/\* Skills Section \*/}

        <div className="mt-6">

          <h3 className="text-lg font-semibold text-gray-800 flex items-center">

            <IconTools className="h-5 w-5 mr-2 text-blue-500" stroke={1.5} />

            Top Skills:

          </h3>

          <div className="flex flex-wrap justify-center mt-3 gap-2">

            {talent.topSkill.map((skill, index) => (

              <span

                key={index}

                className="bg-gradient-to-r from-blue-400 to-blue-600 text-white px-4 py-1 rounded-full text-sm font-medium shadow-md hover:shadow-lg transform hover:scale-105 transition-all"

              >

                {skill}

              </span>

            ))}

          </div>

        </div>

        {/\* Salary Section \*/}

        <p className="mt-6 font-semibold text-xl text-green-700 flex justify-center items-center">

          <IconCurrencyRupee className="h-6 w-6 mr-1 text-green-600" stroke={1.5} /> Salary: {talent.salary}

        </p>

        {/\* Experience Section \*/}

        <div className='mt-6 px-3'>

          <h3 className="text-2xl font-semibold text-gray-800 flex items-center">

            <IconBriefcase className="h-6 w-6 mr-2 text-blue-500" stroke={1.5} />

            Experience

          </h3>

          <div className="flex flex-col gap-4 mt-4">

            {talent.experience && talent.experience.length > 0 ? (

              talent.experience.map((exp, index) => (

                <ExpCard

                  key={index}

                  title={exp.title}

                  company={exp.company}

                  location={exp.location}

                  startDate={exp.startDate}

                  endDate={exp.endDate}

                  description={exp.description}

                />

              ))

            ) : (

              <p className="text-gray-500 text-center">No experience details available.</p>

            )}

          </div>

        </div>

        {/\* Certification Section \*/}

        <div className="mt-6 px-3">

          <h3 className="text-2xl font-semibold text-gray-800 flex items-center">

            <IconCertificate className="h-6 w-6 mr-2 text-blue-500" stroke={1.5} />

            Certifications

          </h3>

          <div className="flex flex-col gap-4 mt-4">

            {talent.certifications && talent.certifications.length > 0 ? (

              talent.certifications.map((cert, index) => (

                <CertiCard

                  key={index}

                  title={cert.title}

                  company={cert.company}

                  issueDate={cert.issueDate}

                  certificateID={cert.certificateID}

                />

              ))

            ) : (

              <p className="text-gray-500 text-center">No certifications available.</p>

            )}

          </div>

        </div>

        {/\* Action Buttons \*/}

        <div className="mt-8 flex justify-center gap-6">

          <button className="py-2 px-6 bg-gradient-to-r from-blue-500 to-blue-700 text-white text-lg rounded-lg shadow-lg transform hover:scale-110 transition-all flex items-center hover:shadow-xl">

            <IconMessage className="mr-2" /> Message

          </button>

          <button

            className="py-2 px-6 bg-gray-600 text-white text-lg rounded-lg shadow-lg transform hover:scale-110 transition-all hover:bg-gray-700"

            onClick={onBack}

          >

            Back

          </button>

        </div>

      </div>

    </div>

  );

};

export default Profile;

import TalentCard from "../FindTalent/TalentCard";

import talentsData from "../FindTalent/TalentData";

function RecommendedTalent() {

  return (

    <div>

      <h1 className='text-xl font-semibold mb-5'>Recommended Talent</h1>

      <div className="flex flex-wrap gap-5">

        {talentsData.map((talent: any, index: any) => (

          <TalentCard key={index} {...talent} />

        ))}

      </div>

    </div>

  );

}

export default RecommendedTalent;

// CardData.ts

import { IconMapPin, IconPremiumRights, IconRecharging, IconBriefcase } from "@tabler/icons-react";

export const card = [

    {

        name: "Location",

        icon: IconMapPin,

        value: "Delhi",

    },

    {

        name: "Experience",

        icon: IconBriefcase,

        value: "Expert",

    },

    {

        name: "Salary",

        icon: IconPremiumRights,

        value: "32 LPA",

    },

    {

        name: "Job Type",

        icon: IconRecharging,

        value: "Full Time",

    }

];

export const skills = [

    "JavaScript", "TypeScript", "Python", "Java", "C++", "C#", "Ruby", "Go", "Swift", "Kotlin", "PHP", "SQL",

    "HTML", "CSS", "Rust", "Scala", "Perl", "Dart", "R", "MATLAB"

];

export const jobDescription = `

    At "CAREERSYNC", we are a passionate, fun-loving, and growing team.

    We are looking for enthusiastic programmers who love solving technical challenges

    and learning new technologies. As a part of our team, you will have the opportunity

    to work on cutting-edge projects, collaborate with talented professionals, and

    contribute to innovative solutions that make a real impact. We value creativity,

    dedication, and a growth mindset. If you are someone who thrives in a dynamic

    environment and is eager to expand your skill set, this is the perfect place for you.

    Join us to be a part of a culture that fosters learning, innovation, and excellence.</p>

`;

export const responsibilities = `

    <ul>

        <li>Develop and maintain high-quality software solutions using modern programming languages and frameworks.</li>

        <li>Collaborate with cross-functional teams to design, develop, and implement innovative features.</li>

        <li>Participate in code reviews to ensure code quality and adherence to best practices.</li>

        <li>Identify and resolve technical challenges and bugs in a timely manner.</li>

        <li>Stay updated with the latest industry trends and technologies to continuously improve skills.</li>

        <li>Contribute to the overall success of projects by meeting deadlines and delivering exceptional results.</li>

    </ul>

`;

export const qualification = `

    <ul>

        <li>Master's degree in Computer Science, Information Technology, or a related field.</li>

        <li>3+ years of professional experience in software development or a related role.</li>

        <li>Proficiency in programming languages such as JavaScript, TypeScript, Python, or Java.</li>

        <li>Strong understanding of web development technologies including HTML, CSS, and modern JavaScript frameworks.</li>

        <li>Experience with version control systems like Git and collaborative development workflows.</li>

        <li>Excellent problem-solving skills and the ability to work independently or as part of a team.</li>

        <li>Strong communication skills and the ability to effectively convey technical concepts to non-technical stakeholders.</li>

    </ul>

`;

export const CompanyData = {

    Name: "Google",

    OverView:

      "Google, a multinational technology company, is renowned for its search engine, online advertising technologies, cloud computing, software, and hardware. Its mission is to organize the world's information and make it universally accessible and useful. Google's diverse portfolio includes products like Android, Chrome, YouTube, and Google Cloud. The company's influence spans across various sectors, impacting communication, entertainment, and productivity.",

    Industry: "Internet and Search, Technology",

    Website: "https://www.google.com",

    Size: "183,323",

    Headquarter: "Mountain View, California, United States",

    Specialties: [

      "Search Engine Technology",

      "Artificial Intelligence and Machine Learning",

      "Cloud Computing Infrastructure (Google Cloud)",

      "Online Advertising and Marketing",

      "Mobile Operating Systems (Android)",

    ],

  }

  export const SimilarCompanies = [

    { name: "Meta", employees: 21456 },

    { name: "Google", employees: 13579 },

    { name: "Amazon", employees: 98765 },

    { name: "Microsoft", employees: 54321 },

    { name: "Apple", employees: 24680 },

    { name: "Tesla", employees: 11223 },

    { name: "Netflix", employees: 55667 },

    { name: "Twitter", employees: 77889 },

    { name: "LinkedIn", employees: 99001 },

    { name: "Spotify", employees: 33445 },

  ];

import React from "react";

import { IconRecharging, IconSearch, IconMapPin, IconBriefcase } from "@tabler/icons-react";

export interface WorkerData {

  id: number;

  image: string;

  title: string;

  description: string;

}

export const workerData: WorkerData[] = [

  {

    id: 1,

    image: "cv.png",

    title: "Build your resume",

    description: "Create a professional resume with ease and stand out in the job market.",

  },

  {

    id: 2,

    image: "job.png",

    title: "Find the right job",

    description: "Search and apply for jobs that match your skills and aspirations.",

  },

  {

    id: 3,

    image: "get-hire.png",

    title: "Apply and get hired",

    description: "Track your applications and get hired by top companies.",

  },

];

export interface TestimonialItem {

  id: number;

  name: string;

  rating: number;

  testimonial: string;

  avatar: string;

}

export const testimonials: TestimonialItem[] = [

  {

    id: 1,

    name: "Ram Raj",

    rating: 3.5,

    testimonial: "I had a great experience using this service. Highly recommended!",

    avatar: "https://source.unsplash.com/80x80/?face&sig=1",

  },

  {

    id: 2,

    name: "Ramesh",

    rating: 4.5,

    testimonial: "Excellent service and very professional team. Highly recommended!",

    avatar: "https://source.unsplash.com/80x80/?face&sig=2",

  },

  {

    id: 3,

    name: "Mohan",

    rating: 3,

    testimonial: "Had a great experience, everything was smooth and efficient!",

    avatar: "https://source.unsplash.com/80x80/?face&sig=3",

  },

  {

    id: 4,

    name: "Raju",

    rating: 4,

    testimonial: "Very satisfied with the service. Quick response and friendly staff.",

    avatar: "https://source.unsplash.com/80x80/?face&sig=4",

  },

  {

    id: 5,

    name: "Sita Verma",

    rating: 5,

    testimonial: "Absolutely fantastic! The best service I have ever used.",

    avatar: "https://source.unsplash.com/80x80/?face&sig=5",

  },

];

export const footerLinks = [

  { title: "Product", links: ["Find Jobs", "Find Company", "Find Employee"] },

  { title: "Company", links: ["About Us", "Contact Us", "Terms & Conditions"] },

  { title: "Support", links: ["Help & Support", "Feedback", "FAQ"] },

];

// src/data/JobData.ts

import { IconBriefcase, IconMapPin, IconRecharging, IconSearch } from "@tabler/icons-react";

export interface DropdownItem {

  title: string;

  icon: React.ElementType;

  options: string[];

}

export const dropdownData: DropdownItem[] = [

  {

    title: "Job Title",

    icon: IconSearch,

    options: [

      "Designer", "Developer", "Manager", "Tester", "Engineer",

      "Marketing Specialist", "Data Analyst", "Business Analyst",

      "Product Manager", "HR Manager", "Finance Manager",

      "Accountant", "Sales Manager", "Marketing Manager",

      "Customer Support", "Customer Success", "Customer Service",

      "Customer Care"

    ],

  },

  {

    title: "Location",

    icon: IconMapPin,

    options: [

      "Delhi", "Bangalore", "Mumbai", "Noida", "Hyderabad",

      "Lucknow", "Pune", "Chennai", "Kolkata", "Ahmedabad",

      "Jaipur", "Chandigarh", "Indore", "Surat", "Nagpur"

    ],

  },

  {

    title: "Experience",

    icon: IconBriefcase,

    options: [

      "Fresher", "1-2 Years", "2-3 Years", "3-4 Years", "4-5 Years",

      "5-6 Years", "6-7 Years", "7-8 Years", "8-9 Years",

      "9-10 Years", "10+ Years"

    ],

  },

  {

    title: "Job Type",

    icon: IconRecharging,

    options: ["Full Time", "Part Time", "Internship", "Contract", "Remote", "Temporary", "Freelance"],

  }

];

export interface Job {

  id: string;

  jobTitle: string;

  companyName: string;

  applicants: number;

  experience: string;

  jobType: string;

  location: string;

  salary: string;

  postDayAgo: number;

  description: string;

}

export const jobList: Job[] = [

  {

    id: "1",

    jobTitle: "Product Designer",

    companyName: "Meta",

    applicants: 25,

    experience: "Entry Level",

    jobType: "Full Time",

    location: "Remote",

    salary: "20 LPA",

    postDayAgo: 12,

    description: "Responsible for designing and prototyping user experiences for Meta's core products. Collaborate with engineers and researchers to build intuitive designs."

  },

  {

    id: "2",

    jobTitle: "Frontend Developer",

    companyName: "Google",

    applicants: 30,

    experience: "Mid Level",

    jobType: "Full Time",

    location: "Bangalore",

    salary: "25 LPA",

    postDayAgo: 15,

    description: "Develop high-performance web applications using React and Angular. Work closely with designers and backend engineers to build seamless user experiences."

  },

  {

    id: "3",

    jobTitle: "Data Scientist",

    companyName: "Microsoft",

    applicants: 40,

    experience: "Senior Level",

    jobType: "Full Time",

    location: "Hyderabad",

    salary: "35 LPA",

    postDayAgo: 10,

    description: "Analyze large datasets to derive insights and drive business decisions. Expertise in Python, machine learning, and deep learning frameworks required."

  },

  {

    id: "4",

    jobTitle: "Software Engineer",

    companyName: "Amazon",

    applicants: 50,

    experience: "Entry Level",

    jobType: "Full Time",

    location: "Chennai",

    salary: "18 LPA",

    postDayAgo: 8,

    description: "Work on backend services using Java and AWS. Design scalable and reliable systems to support millions of users."

  },

  {

    id: "5",

    jobTitle: "DevOps Engineer",

    companyName: "Netflix",

    applicants: 20,

    experience: "Mid Level",

    jobType: "Full Time",

    location: "Remote",

    salary: "30 LPA",

    postDayAgo: 14,

    description: "Automate cloud infrastructure and CI/CD pipelines for Netflix's streaming platform. Experience with Docker, Kubernetes, and AWS required."

  },

  {

    id: "6",

    jobTitle: "Cyber Security Analyst",

    companyName: "IBM",

    applicants: 22,

    experience: "Senior Level",

    jobType: "Full Time",

    location: "Pune",

    salary: "28 LPA",

    postDayAgo: 7,

    description: "Monitor and secure enterprise systems against cyber threats. Experience in ethical hacking and penetration testing is a plus."

  },

  {

    id: "15",

    jobTitle: "Backend Developer",

    companyName: "Swiggy",

    applicants: 35,

    experience: "Mid Level",

    jobType: "Full Time",

    location: "Bangalore",

    salary: "22 LPA",

    postDayAgo: 5,

    description: "Build and optimize RESTful APIs using Node.js and Express. Design scalable database architectures using MongoDB and PostgreSQL."

  },

  {

    id: "7",

    jobTitle: "AI/ML Engineer",

    companyName: "Tesla",

    applicants: 18,

    experience: "Senior Level",

    jobType: "Full Time",

    location: "California, USA",

    salary: "50 LPA",

    postDayAgo: 20,

    description: "Develop deep learning models for Tesla's autonomous vehicles. Expertise in computer vision and reinforcement learning required."

  },

  {

    id: "8",

    jobTitle: "Full Stack Developer",

    companyName: "Paytm",

    applicants: 27,

    experience: "Mid Level",

    jobType: "Full Time",

    location: "Noida",

    salary: "20 LPA",

    postDayAgo: 6,

    description: "Develop and maintain full-stack applications using React.js and Spring Boot. Hands-on experience with microservices architecture is required."

  },

  {

    id: "9",

    jobTitle: "Game Developer",

    companyName: "Ubisoft",

    applicants: 15,

    experience: "Entry Level",

    jobType: "Full Time",

    location: "Pune",

    salary: "16 LPA",

    postDayAgo: 9,

    description: "Work on 3D game development using Unity and Unreal Engine. Strong skills in C++ and 3D rendering techniques are essential."

  },

  {

    id: "10",

    jobTitle: "Blockchain Developer",

    companyName: "Coinbase",

    applicants: 19,

    experience: "Mid Level",

    jobType: "Full Time",

    location: "Remote",

    salary: "35 LPA",

    postDayAgo: 11,

    description: "Develop and optimize smart contracts using Solidity for Ethereum-based applications. Experience with Web3.js and blockchain security is required."

  },

  {

    id: "11",

    jobTitle: "Cloud Engineer",

    companyName: "Google Cloud",

    applicants: 32,

    experience: "Senior Level",

    jobType: "Full Time",

    location: "Bangalore",

    salary: "40 LPA",

    postDayAgo: 4,

    description: "Design and implement cloud solutions on Google Cloud Platform. Expertise in Terraform, Kubernetes, and networking is a must."

  },

  {

    id: "12",

    jobTitle: "Technical Writer",

    companyName: "Atlassian",

    applicants: 12,

    experience: "Entry Level",

    jobType: "Full Time",

    location: "Remote",

    salary: "12 LPA",

    postDayAgo: 13,

    description: "Write and maintain documentation for Atlassian's suite of developer tools. Strong understanding of APIs and DevOps is a plus."

  },

  {

    id: "13",

    jobTitle: "UI/UX Designer",

    companyName: "Adobe",

    applicants: 23,

    experience: "Mid Level",

    jobType: "Full Time",

    location: "Delhi",

    salary: "18 LPA",

    postDayAgo: 16,

    description: "Create intuitive and aesthetically pleasing interfaces for Adobe's creative suite. Proficiency in Figma and Adobe XD is required."

  },

  {

    id: "14",

    jobTitle: "Data Analyst",

    companyName: "Flipkart",

    applicants: 40,

    experience: "Entry Level",

    jobType: "Full Time",

    location: "Bangalore",

    salary: "14 LPA",

    postDayAgo: 3,

    description: "Analyze customer purchase trends using SQL and Python. Generate insights to improve sales and customer engagement."

  }

];

export default ["dropdownData","jobList"];

export const activeJob = [

    { "JobTitle": "Cloud Engineer", "Location": "Bangalore", "Posted": "1 hour ago" },

    { "JobTitle": "Finance Manager", "Location": "Pune", "Posted": "4 hours ago" },

    { "JobTitle": "AI Engineer", "Location": "Hyderabad", "Posted": "2 days ago" },

    { "JobTitle": "Digital Marketing Manager", "Location": "Noida", "Posted": "2 days ago" },

    { "JobTitle": "Content Writer", "Location": "Pune", "Posted": "1 day ago" },

    { "JobTitle": "SEO Specialist", "Location": "Chennai", "Posted": "3 days ago" },

    { "JobTitle": "Data Analyst", "Location": "Hyderabad", "Posted": "3 days ago" },

    { "JobTitle": "Project Manager", "Location": "Gurgaon", "Posted": "4 days ago" },

    { "JobTitle": "HR Executive", "Location": "Delhi", "Posted": "5 days ago" },

    { "JobTitle": "Full Stack Developer", "Location": "Mumbai", "Posted": "5 days ago" },

    { "JobTitle": "Mobile App Developer", "Location": "Kolkata", "Posted": "6 days ago" },

    { "JobTitle": "Software Engineer", "Location": "Bangalore", "Posted": "1 week ago" },

    { "JobTitle": "Cybersecurity Analyst", "Location": "Noida", "Posted": "1 week ago" },

    { "JobTitle": "Graphic Designer", "Location": "Mumbai", "Posted": "2 weeks ago" },

    { "JobTitle": "Machine Learning Engineer", "Location": "Delhi", "Posted": "3 weeks ago" }

];

export const drafted = [

    { "JobTitle": "Cloud Engineer", "Location": "Bangalore", "Posted": "1 hour ago" },

    { "JobTitle": "Blockchain Developer", "Location": "Hyderabad", "Posted": "3 hours ago" },

    { "JobTitle": "Finance Manager", "Location": "Pune", "Posted": "4 hours ago" },

    { "JobTitle": "DevOps Engineer", "Location": "Gurgaon", "Posted": "5 hours ago" },

    { "JobTitle": "AI Engineer", "Location": "Hyderabad", "Posted": "2 days ago" },

    { "JobTitle": "Digital Marketing Manager", "Location": "Noida", "Posted": "2 days ago" },

    { "JobTitle": "Content Writer", "Location": "Pune", "Posted": "1 day ago" },

    { "JobTitle": "SEO Specialist", "Location": "Chennai", "Posted": "3 days ago" },

    { "JobTitle": "Data Analyst", "Location": "Hyderabad", "Posted": "3 days ago" },

    { "JobTitle": "Project Manager", "Location": "Gurgaon", "Posted": "4 days ago" },

    { "JobTitle": "HR Executive", "Location": "Delhi", "Posted": "5 days ago" },

    { "JobTitle": "Full Stack Developer", "Location": "Mumbai", "Posted": "5 days ago" },

    { "JobTitle": "Mobile App Developer", "Location": "Kolkata", "Posted": "6 days ago" },

    { "JobTitle": "Business Analyst", "Location": "Pune", "Posted": "6 days ago" },

    { "JobTitle": "Software Engineer", "Location": "Bangalore", "Posted": "1 week ago" },

    { "JobTitle": "Cybersecurity Analyst", "Location": "Noida", "Posted": "1 week ago" },

    { "JobTitle": "Network Administrator", "Location": "Chennai", "Posted": "1 week ago" },

    { "JobTitle": "Product Manager", "Location": "Bangalore", "Posted": "2 weeks ago" },

    { "JobTitle": "Graphic Designer", "Location": "Mumbai", "Posted": "2 weeks ago" },

    { "JobTitle": "Machine Learning Engineer", "Location": "Delhi", "Posted": "3 weeks ago" }

];

// Ensure it's recognized as a module

export {};

const fields = [

    {

        label: "Job Title",

        placeholder: "Enter Job Title",

        options: ["Designer", "Developer", "Product Manager", "Market Specialist", "Data Analyst", "Customer Service", "Content Writer", "Sales Executive"]

    },

    {

        label: "Company",

        placeholder: "Enter Company Name",

        options: [

            "Google", "Microsoft", "Apple", "Amazon", "Facebook",

            "IBM", "Intel", "Cisco", "Oracle", "Samsung",

            "Sony", "Dell", "HP", "Accenture", "Capgemini",

            "Deloitte", "TCS", "Infosys", "Wipro", "Cognizant"

        ]

    },

    {

        label:"Experience",

        placeholder:"Enter Experience",

        options:["Entry Level","Intermediate Level","Expert"]

    },

    {

        label:"Job Type",

        placeholder:"Enter Job Type",

        options:["Full Time","Part Time","Freelance","Internship","Contract"]

    },

    {

        label:"Location",

        placeholder:"Enter Location",

        options:[

            "Bangalore", "Hyderabad", "Pune", "Chennai", "Mumbai",

            "Delhi", "Noida", "Gurgaon", "Kolkata", "Ahmedabad",

            "Thiruvananthapuram", "Kochi", "Jaipur", "Indore", "Nagpur",

            "Lucknow", "Chandigarh", "Bhubaneswar", "Coimbatore", "Mysore"

        ]

    },

    {

        label:"Salary",

        placeholder:"Enter Salary",

        options:[

            "3 LPA", "4 LPA", "5 LPA", "6 LPA", "7 LPA",

            "8 LPA", "9 LPA", "10 LPA", "12 LPA", "15 LPA",

            "18 LPA", "20 LPA", "25 LPA", "30 LPA", "35 LPA",

            "40 LPA", "50 LPA", "60 LPA", "75 LPA", "95 LPA"

        ]

    },

    {

        label:"Skills",

        placeholder:"Enter Skills",

        options:[

            "JavaScript", "TypeScript", "React", "Angular", "Vue.js",

            "Node.js", "Express.js", "Python", "Django", "Flask",

            "Java", "Spring Boot", "Kotlin", "Swift", "C#",

            ".NET", "Ruby on Rails", "PHP", "Laravel", "Go",

            "Rust", "C++", "C", "SQL", "NoSQL",

            "GraphQL", "Docker", "Kubernetes", "AWS", "Azure"

        ]

    }

];

const content = `<h1>About The Job</h1> <p>Write Description Here...</p> <h4>Responsibilities</h4> <ul> <li>Write Responsibilities Here...</li></ul> <h4>Qualifications & Skill Set</h4> <ul> <li> Add Qualification & Skill Set Here...</li></ul>`;

export { fields, content };

import { Divider } from '@mantine/core';

import React from 'react';

import { Link } from 'react-router-dom';

import { IconArrowLeft } from '@tabler/icons-react';

import ApplyJobComp from '../Components/CompanyProfile/ApplyJobComp';

function ApplyJobPage() {

  return (

    <div className="min-h-screen bg-gray-900 text-white font-[poppins] p-6 flex flex-col items-center relative">

      <Divider size="xs" className="my-4 w-full" />

      <Link to="/jobs">

        <button

          className="absolute top-4 left-4 flex items-center gap-2 py-2 px-4 bg-gradient-to-r from-purple-500 to-pink-500 text-white rounded-lg shadow-lg hover:from-purple-600 hover:to-pink-600 transition-all">

          <IconArrowLeft stroke={2} /> Back

        </button>

      </Link>

      <ApplyJobComp/>

    </div>

  );

}

export default ApplyJobPage;

import { BrowserRouter, Routes, Route, Navigate } from "react-router-dom"

import Footer from "../Components/Footer/Footer"

import Header from "../Components/Header/Header"

import ApplyJobPage from "./ApplyJobPage"

import CompanyPage from "./CompanyPage"

import FindJob from "./FindJob"

import FindTalentPage from "./FindTalentPage"

import Homepage from "./Homepage"

import JobDesPage from "./JobDesPage"

import JobHistoryPage from "./JobHistoryPage"

import PostedJobPage from "./PostedJobPage"

import PostJobPage from "./PostJobPage"

import ProfilePage from "./ProfilePage"

import SignupPage from "./SignupPage"

import TalentProfile from "./TalentProfile"

import { useSelector } from "react-redux"

const AppRoutes = () => {

    const user=useSelector((state:any)=>state.user);

    return(

    <BrowserRouter>

    <Header />

    <Routes>

      <Route path="/find-job" element={<FindJob />} />

      <Route path="/find-talent" element={<FindTalentPage />}/>

      <Route path="job" element={<JobDesPage/>}/>

      <Route path="apply-job" element={<ApplyJobPage/>}/>

      <Route path="/talent-profile" element={<TalentProfile />}/>

      <Route path="/company" element={<CompanyPage/>} />

      <Route path="/posted-job" element={<PostedJobPage/>}/>

      <Route path="/job-history" element={<JobHistoryPage/>}/>

      <Route path="/post-job" element={<PostJobPage/>}/>

      <Route path="/signup" element={user?<Navigate to="/"/>:<SignupPage/>}/>

      <Route path="/login" element={user?<Navigate to="/"/>:<SignupPage/>}/>

      <Route path="/profile" element={<ProfilePage/>}/>

      <Route path="\*" element={<Homepage />} />

    </Routes>

    <Footer/>

    </BrowserRouter>

)}

export default AppRoutes;

import React from "react";

import { useNavigate } from "react-router-dom";

import { Divider } from "@mantine/core";

import { IconArrowLeft } from "@tabler/icons-react";

import SimilarCompany from "../Components/CompanyProfile/SimilarCompany";

import Company from "../Components/CompanyProfile/Company";

function CompanyPage() {

  const navigate = useNavigate();

  return (

    <div className="min-h-screen bg-gray-900 text-white font-[Poppins] p-6 flex flex-col items-center relative">

      {/\* Back Button \*/}

      <button

        className="absolute top-4 left-4 py-2 px-4 bg-gradient-to-r from-purple-500 to-pink-500 text-white rounded-lg flex items-center gap-2 hover:from-purple-600 hover:to-pink-600 transition-all"

        onClick={() => navigate(-1)}

        aria-label="Go Back"

      >

        <IconArrowLeft stroke={2} /> Back

      </button>

      {/\* Divider \*/}

      <Divider size="xs" className="my-6 w-full" />

      {/\* Company and Similar Companies Section \*/}

      <main className="flex flex-col md:flex-row gap-6 w-full max-w-6xl">

        <Company />

        <SimilarCompany />

      </main>

    </div>

  );

}

export default CompanyPage;

import React from "react";

import { Link } from "react-router-dom";

import SearchBar from "../Components/FindJob/SearchBar"; // Ensure correct import path

import Jobs from "../Components/FindJob/Jobs";

import { Divider } from "@mantine/core";

const FindJob: React.FC = () => {

  return (

    <div className="min-h-[90vh] bg-gray-700 font-[poppins] p-6">

      <SearchBar />

      <Divider className="my-4" />

      <Jobs />

    </div>

  );

};

export default FindJob;

import { Divider } from "@mantine/core"

import SearchBar from "../Components/FindTalent/SearchBar"

import Talent from "../Components/FindTalent/Talent"

function FindTalentPage() {

  return (

    <div className="min-h-[90vh] bg-gray-700 font-[poppins]">

        <Divider size="xs" mx="md"/>

        <SearchBar/>

        <Divider size="xs" mx="md"/>

        <Talent/>

    </div>

  )

}

export default FindTalentPage;

import Companies from "../Components/LandingPage/Companies";

import JobCategory from "../Components/LandingPage/JobCategory";

import Working from "../Components/LandingPage/Working";

import Testimonial from "../Components/LandingPage/Testimonial";

import Subscribe from "../Components/LandingPage/Subscribe";

import DreamJob from "../Components/LandingPage/DreamJob";

const Homepage = () => {

  return (

    <div className="relative min-h-screen font-poppins bg-gray-900 overflow-hidden flex flex-col text-gray-100">

      <DreamJob/>

      <Companies/>

      <JobCategory/>

      <Working/>

      <Testimonial/>

      <Subscribe/>

    </div>

  );

};

export default Homepage;

import { Divider } from "@mantine/core";

import { IconArrowLeft } from "@tabler/icons-react";

import { useNavigate } from "react-router-dom";

import JobDesk from "../Components/JobDescription/JobDesk";

import RecommendedJob from "../Components/JobDescription/RecommendedJob";

import { useState } from "react";

const JobDesPage: React.FC = () => {

  const navigate = useNavigate();

  const [currentIndex, setCurrentIndex] = useState(0);

  const totalJobs = 12;

  const jobsPerPage = 6;

  const totalPages = Math.ceil(totalJobs / jobsPerPage);

  const handleNext = () => {

    setCurrentIndex((prevIndex) => (prevIndex + 1) % totalPages);

  };

  const handlePrev = () => {

    setCurrentIndex((prevIndex) => (prevIndex - 1 + totalPages) % totalPages);

  };

  return (

    <div className="min-h-screen bg-gray-900 text-white font-[poppins] p-6 flex flex-col items-center relative">

      {/\* Back Button \*/}

      <button

        className="absolute top-4 left-4 flex items-center gap-2 py-2 px-4 bg-gradient-to-r from-purple-500 to-pink-500 text-white rounded-lg shadow-lg hover:from-purple-600 hover:to-pink-600 transition-all"

        onClick={() => navigate("/find-job")}

      >

        <IconArrowLeft stroke={2} /> Back

      </button>

      {/\* Header Section \*/}

      <div className="w-full max-w-6xl text-center mt-12">

        <h1 className="text-3xl font-bold text-purple-400">Job Details</h1>

        <Divider size="xs" className="my-4 w-full" />

      </div>

      {/\* Main Content Section \*/}

      <div className="flex flex-col md:flex-row gap-6 w-full max-w-6xl mt-6">

        {/\* Job Description Section \*/}

        <div className="flex-1 bg-gray-800 p-6 rounded-2xl shadow-lg border border-gray-700">

          <h2 className="text-2xl font-semibold text-purple-400 mb-4">Job Description</h2>

          <JobDesk />

        </div>

        {/\* Recommended Jobs Section \*/}

        <div className="md:w-1/3 w-full bg-gray-800 p-6 rounded-2xl shadow-lg border border-gray-700 flex flex-col items-center">

          <h2 className="text-2xl font-semibold text-pink-400 mb-4">Recommended Jobs</h2>

          <RecommendedJob currentIndex={currentIndex} jobsPerPage={jobsPerPage} />

          <div className="flex gap-4 mt-4">

            <button

              className="py-2 px-4 bg-gradient-to-r from-blue-500 to-green-500 text-white rounded-lg shadow-lg hover:from-blue-600 hover:to-green-600 transition-all"

              onClick={handlePrev}

            >

              Previous

            </button>

            <button

              className="py-2 px-4 bg-gradient-to-r from-green-500 to-blue-500 text-white rounded-lg shadow-lg hover:from-green-600 hover:to-blue-600 transition-all"

              onClick={handleNext}

            >

              Next

            </button>

          </div>

        </div>

      </div>

    </div>

  );

};

export default JobDesPage;

import { Divider } from '@mantine/core'

import React from 'react'

import JobHistory from '../Components/JobHistory/JobHistory'

function JobHistoryPage() {

  return (

    <div className="min-h-[900vh] gap-6 w-full max-w-6xl my-5">

        <Divider size="xs"/>

        <div className="my-5"></div>

        <JobHistory/>

    </div>

  )

}

export default JobHistoryPage

import React from 'react'

import PostedJob from '../Components/PostedJob/PostedJob';

import PostedJobDesc from '../Components/PostedJob/PostedJobDesc';

function PostedJobPage() {

  return (

    <div>

      <main className="flex flex-col md:flex-row gap-6 w-full max-w-6xl">

        <PostedJob/>

        <PostedJobDesc/>

      </main>

    </div>

  )

}

export default PostedJobPage;

import { Divider } from '@mantine/core'

import React from 'react'

import PostJob from '../Components/PostJob/PostJob'

function PostJobPage() {

  return (

    <div className="min-h-screen bg-gray-900 text-white font-[poppins] p-6 flex flex-col items-center relative">

      <Divider size="xs" className="my-4 w-full" />

      <PostJob/>

    </div>

  )

}

export default PostJobPage

import { Divider } from "@mantine/core";

import { IconArrowLeft } from "@tabler/icons-react";

import { useNavigate } from "react-router-dom";

import talentsData from "../Components/FindTalent/TalentData";

import Profile from "../Components/Profile/Profile";

const ProfilePage: React.FC = () => {

  const navigate = useNavigate();

  const selectedTalent = talentsData[1]; // Example talent, can be dynamic in real use

  return (

    <div className="min-h-screen bg-gray-900 text-white font-[Poppins] p-6 flex flex-col items-center relative">

      {/\* Back Button \*/}

      <button

        onClick={() => navigate("/find-talent")}

        className="absolute top-4 left-4 py-2 px-4 bg-gradient-to-r from-purple-700 to-pink-600 text-white rounded-lg flex items-center gap-2 hover:from-purple-800 hover:to-pink-700 transition-all shadow-xl"

      >

        <IconArrowLeft stroke={2} /> Back

      </button>

        {/\* Divider \*/}

        <Divider size="xs" className="mb-6 border-t border-gray-600" />

        {/\* Profile Component \*/}

        <div className="grid grid-cols-1 lg:grid-cols-1 gap-6">

          <Profile talent={selectedTalent} onBack={() => navigate("/talent-profile")} />

        </div>

      </div>

  );

};

export default ProfilePage;

// SignupPage.tsx

import React, { useState } from "react";

import { motion, AnimatePresence } from "framer-motion";

import Login from "../Components/SignUp/Login";

import SignUp from "../Components/SignUp/SignUp";

function SignupPage() {

  const [isLogin, setIsLogin] = useState(true);

  const toggleForm = () => setIsLogin(!isLogin);

  const handleSignupSuccess = () => {

    alert("Account created successfully! You can now login.");

    setIsLogin(true);

  };

  const formVariants = {

    hidden: { opacity: 0, x: isLogin ? -100 : 100 },

    visible: {

      opacity: 1,

      x: 0,

      transition: { duration: 0.5, ease: "easeInOut" }

    },

    exit: {

      opacity: 0,

      x: isLogin ? -100 : 100,

      transition: { duration: 0.4, ease: "easeInOut" }

    }

  };

  return (

    <div className="min-h-screen flex justify-center items-center bg-gradient-to-r from-gray-900 to-black px-4">

      <div className="w-full max-w-5xl flex flex-col md:flex-row rounded-3xl overflow-hidden shadow-2xl border border-yellow-600 bg-[#1f1f2e]">

        <div className="w-full md:w-1/2 flex flex-col justify-center items-center p-8 bg-gradient-to-br from-yellow-400 to-yellow-600 text-white">

          <img

            src="/career.jpeg"

            alt="Career Logo"

            className="w-16 h-16 rounded-full shadow-lg border-2 border-white mb-4"

          />

          <h1 className="text-3xl font-bold mb-2 tracking-wide">CAREERSYNC</h1>

          <p className="text-lg font-semibold text-center">Find the job made for you</p>

          <button

            onClick={toggleForm}

            className="mt-6 px-6 py-2 bg-white text-yellow-700 font-semibold rounded-full shadow-md hover:bg-yellow-200 transition"

          >

            {isLogin ? "Create an Account" : "Already have an Account?"}

          </button>

        </div>

        <div className="w-full md:w-1/2 flex justify-center items-center p-8 bg-[#2a2a3a] text-white">

          <AnimatePresence mode="wait">

            {isLogin ? (

              <motion.div

                key="login"

                variants={formVariants}

                initial="hidden"

                animate="visible"

                exit="exit"

                className="w-full"

              >

                <Login />

              </motion.div>

            ) : (

              <motion.div

                key="signup"

                variants={formVariants}

                initial="hidden"

                animate="visible"

                exit="exit"

                className="w-full"

              >

                <SignUp onSignupSuccess={handleSignupSuccess} />

              </motion.div>

            )}

          </AnimatePresence>

        </div>

      </div>

    </div>

  );

}

export default SignupPage;

import { Divider } from "@mantine/core";

import { IconArrowLeft } from "@tabler/icons-react";

import { useNavigate } from "react-router-dom";

import Profile from "../Components/TalentProfile/Profile";

import talentsData from "../Components/FindTalent/TalentData";

import RecommendedTalent from "../Components/TalentProfile/RecommendedTalent";

const TalentProfile: React.FC = () => {

  const navigate = useNavigate();

  const selectedTalent = talentsData[1]; // Select the first talent as default for demo

  return (

    <div className="min-h-screen bg-gray-900 text-white font-[poppins] p-6 flex flex-col items-center relative">

      {/\* Back Button \*/}

      <button

        className="absolute top-4 left-4 py-2 px-4 bg-gradient-to-r from-purple-500 to-pink-500 text-white rounded-lg flex items-center gap-2 hover:from-purple-600 hover:to-pink-600 transition-all"

        onClick={() => navigate("/find-talent")}

      >

        <IconArrowLeft stroke={2} /> Back

      </button>

      <Divider size="xs" className="my-4 w-full" />

      {/\* Main Content Section \*/}

      <div className="w-full flex flex-col lg:flex-row gap-6 items-start justify-center">

        {/\* Profile Section \*/}

        <div className="lg:w-2/3 w-full">

          <Profile talent={selectedTalent} onBack={() => navigate("/find-talent")} />

        </div>

        {/\* Recommended Talent Section \*/}

        <div className="lg:w-1/3 w-full">

          <RecommendedTalent />

        </div>

      </div>

    </div>

  );

};

export default TalentProfile;

const signupValidation = (

    fieldName: string,

    value: string,

    passwordValue?: string

  ): string | undefined => {

    switch (fieldName) {

      case "fullName":

        if (value.trim().length < 3) {

          return "Full Name must be at least 3 characters long";

        }

        break;

      case "email":

        const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

        if (!emailRegex.test(value)) {

          return "Invalid email address";

        }

        break;

      case "password":

        const passwordRegex =

          /^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{8,20}$/;

        if (!passwordRegex.test(value)) {

          return "Password must be 8-20 characters long and include uppercase, lowercase, number, and special character";

        }

        break;

      case "confirmPassword":

        if (value !== passwordValue) {

          return "Passwords do not match";

        }

        break;

      default:

        return "Invalid input field";

    }

    return undefined; // No error

  };

  export default signupValidation;

const setItem=(key:string,value:any)=>{

    localStorage.setItem(key, JSON.stringify(value));

}

const getItem=(key:string)=>{

    return JSON.parse(localStorage.getItem(key) as string);

}

const removeItem=(key:string)=>{

    localStorage.removeItem(key);

}

export {setItem,getItem,removeItem};

import { notifications } from "@mantine/notifications";

import { IconX, IconCheck } from "@tabler/icons-react";

// Show a success notification

export const successNotification = (title: string, message: string) => {

  notifications.show({

    title,

    message,

    color: "green",

    icon: <IconCheck size={20} />,

    withBorder: true,

    withCloseButton: true,

    className: "bg-[#1f1f2e] text-yellow-400",

  });

};

// Show an error notification

export const errorNotification = (title: string, message: string) => {

  notifications.show({

    title,

    message,

    color: "red",

    icon: <IconX size={20} />,

    withBorder: true,

    withCloseButton: true,

    className: "bg-[#1f1f2e] text-yellow-400",

  });

};

// Simple default error

export const showGenericError = () => {

  errorNotification("Error", "An error occurred. Please try again.");

};

// services/postJobService.tsx

import axios from 'axios';

interface JobPostData {

    jobTitle: string;

    company: string;

    about: string;

    experience: string;

    jobType: string;

    location: string;

    packageOffer: string;

    description: string;

    skillsRequired: string[];

    jobStatus: "OPEN" | "CLOSE" | "DRAFT";

}

export const postJob = async (jobData: JobPostData) => {

    try {

        const response = await axios.post('http://localhost:8080/jobs/post', {

            ...jobData,

            packageOffer: Number(jobData.packageOffer.replace(' LPA', '')),

        });

        return response.data;

    } catch (error) {

        throw error;

    }

};

import axios from "axios";

const API\_BASE\_URL = "http://localhost:8080/profiles/";

/\*\*

 \* Fetches profile data for a given user ID

 \* @param id - User ID

 \* @returns Promise with profile data

 \*/

export const getProfile = async (id: number): Promise<any> => {

  try {

    const response = await axios.get<any>(`${API\_BASE\_URL}${id}`);

    return response.data;

  } catch (error) {

    if (axios.isAxiosError(error)) {

      throw new Error(

        error.response?.data?.message || "Failed to fetch profile"

      );

    }

    throw new Error("Unknown error occurred while fetching profile");

  }

};

/\*\*

 \* Updates profile data for a given user ID

 \* @param id - User ID

 \* @param profileData - Partial profile data to update

 \* @returns Promise with updated profile data

 \*/

export const updateProfile = async (

  id: number,

  profileData: Partial<any> // Changed 'Profile' to 'any' to fix the error

): Promise<any> => { // Changed 'Profile' to 'any' to fix the error

  try {

    const response = await axios.put<any>(

      `${API\_BASE\_URL}${id}`,

      profileData

    );

    return response.data;

  } catch (error) {

    if (axios.isAxiosError(error)) {

      throw new Error(

        error.response?.data?.message || "Failed to update profile"

      );

    }

    throw new Error("Unknown error occurred while updating profile");

  }

};

// Additional profile-related API calls can be added here

// UserService.tsx

// ✅ Service: Handles API Calls for Registration, Login, OTP, and Password Change.

import axios from "axios";

// 📌 Base URL for User-related API endpoints

const BASE\_URL = "http://localhost:8080/users/";

// 🔥 Register a new user with name, email, password, and account type.

const registerUser = async (user: { name: string; email: string; password: string; accountType: string }) => {

  try {

    const { data } = await axios.post(`${BASE\_URL}register`, user, {

      headers: { "Content-Type": "application/json" },

    });

    return data;

  } catch (error: any) {

    console.error("Register User Error:", error.response?.data || error.message);

    throw new Error(error?.response?.data?.message || "Registration failed. Please try again.");

  }

};

// 🔥 Login user with email and password.

const loginUser = async (credentials: { email: string; password: string }) => {

  try {

    const { data } = await axios.post(`${BASE\_URL}login`, credentials, {

      headers: { "Content-Type": "application/json" },

    });

    return data;

  } catch (error: any) {

    console.error("Login Error:", error.response?.data || error.message);

    throw new Error(error?.response?.data?.message || "Login failed. Please check your credentials.");

  }

};

// 🔥 Send OTP to user's email for password reset or verification.

const sendOtp = async (email: string) => {

  try {

    const { data } = await axios.post(`${BASE\_URL}sendOtp/${encodeURIComponent(email)}`, null, {

      headers: { "Content-Type": "application/json" },

    });

    return data;

  } catch (error: any) {

    console.error("Send OTP Error:", error.response?.data || error.message);

    throw new Error(error?.response?.data?.message || "Failed to send OTP. Try again later.");

  }

};

// 🔥 Verify the OTP received on email.

const verifyOtp = async (email: string, otp: string) => {

  try {

    const { data } = await axios.get(`${BASE\_URL}verifyOtp/${encodeURIComponent(email)}/${otp}`);

    return data;

  } catch (error: any) {

    console.error("Verify OTP Error:", error.response?.data || error.message);

    throw new Error(error?.response?.data?.message || "OTP verification failed.");

  }

};

// 🔥 Change password for a user using email and new password.

const changePass = async (email: string, newPassword: string) => {

  try {

    const { data } = await axios.post(`${BASE\_URL}changePass`, { email, newPassword }, {

      headers: { "Content-Type": "application/json" },

    });

    return data;

  } catch (error: any) {

    console.error("Change Password Error:", error.response?.data || error.message);

    throw new Error(error?.response?.data?.message || "Password change failed. Make sure your email and password are correct.");

  }

};

// 💡 Export all service functions as a single object.

export default { registerUser, loginUser, sendOtp, verifyOtp, changePass };

const formatDate = (dateString: string) => {

    const date = new Date(dateString);

    const options = { year: 'numeric' as const, month: 'long' as const, day: 'numeric' as const };

    return date.toLocaleDateString('en-US', options);

}

export default formatDate;

// PostJobSlice.tsx

import { createSlice, createAsyncThunk, PayloadAction } from '@reduxjs/toolkit';

import { postJob as postJobService } from '../Services/PostJobService';

interface JobState {

  loading: boolean;

  error: string | null;

  success: boolean;

}

const initialState: JobState = {

  loading: false,

  error: null,

  success: false,

};

// ✅ Create AsyncThunk properly

export const postJob = createAsyncThunk(

  'postJob/post',

  async (jobData: any, thunkAPI) => {

    try {

      const response = await postJobService(jobData);

      return response;

    } catch (error: any) {

      return thunkAPI.rejectWithValue(error.message || 'Failed to post job');

    }

  }

);

const postJobSlice = createSlice({

  name: 'postJob',

  initialState,

  reducers: {

    resetJobState: (state) => {

      state.loading = false;

      state.error = null;

      state.success = false;

    },

  },

  extraReducers: (builder) => {

    builder

      .addCase(postJob.pending, (state) => {

        state.loading = true;

        state.error = null;

        state.success = false;

      })

      .addCase(postJob.fulfilled, (state) => {

        state.loading = false;

        state.success = true;

      })

      .addCase(postJob.rejected, (state, action) => {

        state.loading = false;

        state.error = action.payload as string || 'Failed to post job';

      });

  },

});

export const { resetJobState } = postJobSlice.actions;

export default postJobSlice.reducer;

import { createSlice, PayloadAction } from "@reduxjs/toolkit";

interface Skill {

  id: number;

  name: string;

}

interface Certification {

  title: string;

  company: string;

  location?: string;

  issueDate: string;

  certificateID: string;

}

interface Experience {

  title: string;

  company: string;

  location: string;

  startDate: string;

  endDate: string;

  description: string;

  currentlyWorking: boolean;

}

interface ProfileData {

  id: number;

  name: string;

  role: string;

  company: string;

  location: string;

  about: string;

  skills: Skill[];

  salary: string;

  experience: Experience[];

  certifications: Certification[];

  avatar?: string;

  banner?: string;

}

interface ProfileState {

  data: ProfileData | null;

  loading: boolean;

  error: string | null;

}

const initialState: ProfileState = {

  data: null,

  loading: false,

  error: null,

};

const profileSlice = createSlice({

  name: "profile",

  initialState,

  reducers: {

    setProfile: (state, action: PayloadAction<ProfileData>) => {

      state.data = action.payload;

      state.loading = false;

      state.error = null;

    },

    setProfileLoading: (state) => {

      state.loading = true;

      state.error = null;

    },

    setProfileError: (state, action: PayloadAction<string>) => {

      state.loading = false;

      state.error = action.payload;

    },

    updateProfile: (state, action: PayloadAction<Partial<ProfileData>>) => {

      if (state.data) {

        state.data = { ...state.data, ...action.payload };

      }

    },

    clearProfile: (state) => {

      state.data = null;

      state.loading = false;

      state.error = null;

    },

  },

});

export const {

  setProfile,

  setProfileLoading,

  setProfileError,

  updateProfile,

  clearProfile,

} = profileSlice.actions;

export default profileSlice.reducer;

import { createSlice } from "@reduxjs/toolkit";

import { getItem, setItem, removeItem } from "../Services/LocalStorageService";

const UserSlice=createSlice({

    name:"user",

    initialState:getItem("user"),

    reducers:{

        setUser:(state,action)=>{

            setItem("user",action.payload);

            state:getItem("user");

            return state;

        },

        removeUser:(state)=>{

            state=null;

            removeItem("user");

            return state;

        }

    }

    });

export const {setUser,removeUser}=UserSlice.actions;

export default UserSlice.reducer;

import React from 'react';

import { MantineProvider } from '@mantine/core';

import Homepage from './Pages/Homepage';

import { BrowserRouter, Routes, Route } from 'react-router-dom';

import '@mantine/carousel/styles.css'; // Ensure styles are imported

import "@mantine/core/styles.css"; // ✅ Correct way to import styles

import '@mantine/tiptap/styles.css';

import '@mantine/dates/styles.css';

import { Provider } from 'react-redux';

import AppRoutes from './Pages/AppRoutes';

import Store from './Store';

const App = () => {

  return (

    <Provider store={Store}>

    <MantineProvider defaultColorScheme='dark'>

     <AppRoutes/>

    </MantineProvider>

    </Provider>

  );

}

export default App;

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(

  document.getElementById('root') as HTMLElement

);

root.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

// LocalStorageService.ts

export const getItem = (key: string): any => {

    const item = localStorage.getItem(key);

    return item ? JSON.parse(item) : null;

  };

  export const setItem = (key: string, value: any): void => {

    localStorage.setItem(key, JSON.stringify(value));

  };

  export const removeItem = (key: string): void => {

    localStorage.removeItem(key);

  };

// Store.tsx

import { configureStore } from "@reduxjs/toolkit"; // ✅ Correct import at top

import userReducer from "./Slices/UserSlice";

import profileReducer from "./Slices/ProfileSlice";

import postJobReducer from "./Slices/PostJobSlice"; // ✅ Correct import

// ✅ Only one configureStore

 const store = configureStore({

  reducer: {

    user: userReducer,

    profile: profileReducer,

    postJob: postJobReducer,

  },

});

// ✅ Types for Redux Toolkit

export type RootState = ReturnType<typeof store.getState>;

export type AppDispatch = typeof store.dispatch;

export default store; // ✅ Default export

package com.jobportal;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.scheduling.annotation.EnableScheduling;

@SpringBootApplication

@EnableScheduling

public class JobPortalApplication {

public static void main(String[] args) {

SpringApplication.run(JobPortalApplication.class, args);

}

}

package com.jobportal;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.data.mongodb.core.MongoTemplate;

import com.mongodb.client.MongoClient;

import com.mongodb.client.MongoClients;

@Configuration

public class MongoConfig {

@Value("${spring.data.mongodb.uri}")

private String mongoUri;

@Value("${spring.data.mongodb.database}")

private String database;

/\*\*

\* ✅ Creates a MongoClient bean using the URI from application.properties.

\* @return MongoClient

\*/

@Bean

public MongoClient mongoClient() {

return MongoClients.create(mongoUri);

}

/\*\*

\* ✅ Provides a configured MongoTemplate bean.

\* Used for MongoDB database operations.

\* @return MongoTemplate

\*/

@Bean

public MongoTemplate mongoTemplate() {

return new MongoTemplate(mongoClient(), database);

}

}

package com.jobportal;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration

public class SecurityConfig {

// Password encoder bean using BCrypt

@Bean

PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

// Security filter chain for defining security rules

@Bean

SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.csrf(csrf -> csrf.disable()) // Disable CSRF for development (enable in production)

.authorizeHttpRequests(auth -> auth

.anyRequest().permitAll() // Allow all requests (customize as needed)

);

return http.build();

}

}

package com.jobportal.api;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.validation.annotation.Validated;

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.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.jobportal.dto.JobDTO;

import com.jobportal.exception.JobPortalException;

import com.jobportal.service.JobService;

@RestController

@RequestMapping("/jobs")

@Validated

public class JobAPI {

@Autowired

private JobService jobService;

@PostMapping("/post")

public ResponseEntity<JobDTO>postJob(@RequestBody JobDTO jobDTO) throws JobPortalException

{

return new ResponseEntity<>(jobService.postJob(jobDTO),HttpStatus.CREATED);

}

@GetMapping("/getAll")

public ResponseEntity<List<JobDTO>> getAllJobs() throws JobPortalException {

return new ResponseEntity<>(jobService.getAllJobs(), HttpStatus.OK);

}

@GetMapping("/get/{id}")

public ResponseEntity<JobDTO> getJob(@PathVariable Long id) throws JobPortalException {

return new ResponseEntity<>(jobService.getJob(id), HttpStatus.OK);

}

}

// ProfileAPI.java

package com.jobportal.api;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.validation.annotation.Validated;

import org.springframework.web.bind.annotation.\*;

import com.jobportal.dto.ProfileDTO;

import com.jobportal.exception.JobPortalException;

import com.jobportal.service.ProfileService;

import jakarta.validation.Valid;

import jakarta.validation.constraints.Email;

@RestController

@RequestMapping("/api/profiles")

@Validated

public class ProfileAPI {

@Autowired

private ProfileService profileService;

@GetMapping("/{id}")

public ResponseEntity<ProfileDTO> getProfile(@PathVariable Long id) throws JobPortalException {

ProfileDTO profile = profileService.getProfile(id);

return new ResponseEntity<>(profile, HttpStatus.OK);

}

@PutMapping("/{id}")

public ResponseEntity<ProfileDTO> updateProfile(

@PathVariable Long id,

@Valid @RequestBody ProfileDTO profileDTO

) throws JobPortalException {

if (!id.equals(profileDTO.getId())) {

throw new JobPortalException("ID in path and body must match");

}

ProfileDTO updatedProfile = profileService.updateProfile(profileDTO);

return new ResponseEntity<>(updatedProfile, HttpStatus.OK);

}

@PostMapping

public ResponseEntity<Long> createProfile(

@RequestParam @Email String email

) throws JobPortalException {

Long profileId = profileService.createProfile(email);

return new ResponseEntity<>(profileId, HttpStatus.CREATED);

}

}

// ✅ UserAPI.java — Fully Enhanced & Cleaned

package com.jobportal.api;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.validation.annotation.Validated;

import org.springframework.web.bind.annotation.\*;

import com.jobportal.dto.LoginDTO;

import com.jobportal.dto.ResponseDTO;

import com.jobportal.dto.UserDTO;

import com.jobportal.exception.JobPortalException;

import com.jobportal.service.UserService;

import jakarta.validation.Valid;

import jakarta.validation.constraints.Email;

import jakarta.validation.constraints.Pattern;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.util.HashMap;

import java.util.Map;

@Validated

@CrossOrigin

@RestController

@RequestMapping("/users")

public class UserAPI {

private static final Logger logger = LoggerFactory.getLogger(UserAPI.class);

@Autowired

private UserService userService;

/\*\*

\* ✅ Register New User

\*/

@PostMapping("/register")

public ResponseEntity<Map<String, Object>> registerUser(@RequestBody @Valid UserDTO userDTO) throws JobPortalException {

UserDTO savedUser = userService.registerUser(userDTO);

Map<String, Object> response = new HashMap<>();

response.put("user", savedUser);

response.put("message", "Registration successful.");

return new ResponseEntity<>(response, HttpStatus.CREATED);

}

/\*\*

\* ✅ User Login

\*/

@PostMapping("/login")

public ResponseEntity<?> loginUser(@RequestBody @Valid LoginDTO loginDTO) {

try {

UserDTO loggedInUser = userService.loginUser(loginDTO);

Map<String, Object> response = new HashMap<>();

response.put("user", loggedInUser);

response.put("message", "Login successful.");

return new ResponseEntity<>(response, HttpStatus.OK);

} catch (JobPortalException e) {

logger.error("Login failed: {}", e.getMessage());

return new ResponseEntity<>(new ResponseDTO("Login failed: " + e.getMessage()), HttpStatus.UNAUTHORIZED);

}

}

/\*\*

\* ✅ Send OTP to User Email

\*/

@PostMapping("/sendOtp/{email}")

public ResponseEntity<ResponseDTO> sendOtp(@PathVariable @Email(message = "{user.email.invalid}") String email) {

try {

boolean status = userService.sendOtp(email);

return new ResponseEntity<>(

new ResponseDTO(status ? "OTP sent successfully." : "Failed to send OTP."),

status ? HttpStatus.OK : HttpStatus.INTERNAL\_SERVER\_ERROR

);

} catch (JobPortalException e) {

logger.error("Error sending OTP to {}: {}", email, e.getMessage());

return new ResponseEntity<>(new ResponseDTO("Error: " + e.getMessage()), HttpStatus.BAD\_REQUEST);

} catch (Exception e) {

logger.error("Unexpected error sending OTP to {}: {}", email, e.getMessage());

return new ResponseEntity<>(new ResponseDTO("Unexpected error: " + e.getMessage()), HttpStatus.INTERNAL\_SERVER\_ERROR);

}

}

/\*\*

\* ✅ Verify OTP

\*/

@GetMapping("/verifyOtp/{email}/{otp}")

public ResponseEntity<ResponseDTO> verifyOtp(

@PathVariable @Email(message = "{user.email.invalid}") String email,

@PathVariable @Pattern(regexp = "^[0-9]{6}$", message = "otp.invalid") String otp) {

try {

boolean status = userService.verifyOtp(email, otp);

return new ResponseEntity<>(

new ResponseDTO(status ? "OTP has been verified." : "OTP verification failed."),

status ? HttpStatus.OK : HttpStatus.BAD\_REQUEST

);

} catch (JobPortalException e) {

logger.error("OTP verification error for {}: {}", email, e.getMessage());

return new ResponseEntity<>(new ResponseDTO("Verification failed: " + e.getMessage()), HttpStatus.BAD\_REQUEST);

} catch (Exception e) {

logger.error("Unexpected OTP error for {}: {}", email, e.getMessage());

return new ResponseEntity<>(new ResponseDTO("Unexpected error: " + e.getMessage()), HttpStatus.INTERNAL\_SERVER\_ERROR);

}

}

/\*\*

\* ✅ Change Password

\*/

@PostMapping("/changePass")

public ResponseEntity<ResponseDTO> changePassword(@RequestBody @Valid LoginDTO loginDTO) {

try {

return new ResponseEntity<>(userService.changePassword(loginDTO), HttpStatus.OK);

} catch (JobPortalException e) {

logger.error("Password change error: {}", e.getMessage());

return new ResponseEntity<>(new ResponseDTO("Password reset failed: " + e.getMessage()), HttpStatus.BAD\_REQUEST);

} catch (Exception e) {

logger.error("Unexpected error changing password: {}", e.getMessage());

return new ResponseEntity<>(new ResponseDTO("Unexpected error: " + e.getMessage()), HttpStatus.INTERNAL\_SERVER\_ERROR);

}

}

}

**package** com.jobportal.dto;

**public** **enum** AccountType {

***APPLICANT*** , ***EMPLOYER***

}

**package** com.jobportal.dto;

**import** java.time.LocalDateTime;

**public** **class** Applicant {

**private** Long applicantId;

**private** LocalDateTime timestamp;

**private** ApplicationStatus applicationStatus;

}

**package** com.jobportal.dto;

**public** **enum** ApplicationStatus {

***APPLIED***, ***INTERVIWING*** , ***OFFERED***

}

package com.jobportal.dto;

import lombok.AllArgsConstructor;

import lombok.Data;

@Data

@AllArgsConstructor

public class AuthResponse {

private String message;

private String token;

}

// Certification.java

package com.jobportal.dto;

import java.time.LocalDate;

import jakarta.validation.constraints.NotBlank;

import jakarta.validation.constraints.NotNull;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Certification {

@NotBlank(message = "Certification name is required")

private String name;

@NotBlank(message = "Issuer is required")

private String issuer;

private String location;

@NotNull(message = "Issue date is required")

private LocalDate issueDate;

@NotBlank(message = "Certificate ID is required")

private String certificateId;

}

// Experience.java

package com.jobportal.dto;

import java.time.LocalDate;

import jakarta.validation.constraints.NotBlank;

import jakarta.validation.constraints.NotNull;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Experience {

@NotBlank(message = "Job title is required")

private String title;

@NotBlank(message = "Company is required")

private String company;

@NotBlank(message = "Location is required")

private String location;

@NotNull(message = "Start date is required")

private LocalDate startDate;

private LocalDate endDate;

@NotNull(message = "Working status is required")

private Boolean currentlyWorking;

@NotBlank(message = "Description is required")

private String description;

}

package com.jobportal.dto;

import java.time.LocalDateTime;

import java.util.List;

import com.jobportal.entity.Job;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

public class JobDTO {

private Long id;

private String jobTitle;

private String company;

private List<Applicant> applicants;

private String about;

private String experience;

private String jobType;

private String location;

private Long packageOffer;

private LocalDateTime postTime;

private String description;

private List<String> skillsRequired;

private JobStatus jobStatus;

public Job toEntity() {

return new Job(

this.id, this.jobTitle, this.company, null, // applicants not included to avoid circular reference

this.about, this.experience, this.jobType, this.location,

this.packageOffer, this.postTime, this.description,

this.skillsRequired, this.jobStatus

);

}

}

**package** com.jobportal.dto;

**public** **enum** JobStatus {

***OPEN***,***CLOSE***,***DRAFT***

}

package com.jobportal.dto;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

public class LoginDTO {

private String email;

private String password;

}

// ProfileDTO.java

package com.jobportal.dto;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

import com.jobportal.entity.Profile;

import jakarta.validation.constraints.Email;

import jakarta.validation.constraints.NotBlank;

import jakarta.validation.constraints.NotNull;

import java.util.List;

@Data

@NoArgsConstructor

@AllArgsConstructor

public class ProfileDTO {

@NotNull(message = "ID is required")

private Long id;

@NotBlank(message = "Email is required")

@Email(message = "Email should be valid")

private String email;

@NotBlank(message = "Job title is required")

private String jobTitle;

@NotBlank(message = "Company is required")

private String company;

@NotBlank(message = "Location is required")

private String location;

@NotBlank(message = "About section is required")

private String about;

@NotNull(message = "Skills cannot be null")

private List<@NotBlank String> skills;

@NotNull(message = "Experiences cannot be null")

private List<@NotNull Experience> experiences;

@NotNull(message = "Certifications cannot be null")

private List<@NotNull Certification> certifications;

private String avatar;

private String banner;

private String salary;

public Profile toEntity() {

return new Profile(

this.id, this.email, this.jobTitle, this.company, this.location,

this.about, this.skills, this.experiences, this.certifications,

this.avatar, this.banner, this.salary

);

}

}

package com.jobportal.dto;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

@NoArgsConstructor

public class ResponseDTO {

String message;

}

package com.jobportal.dto;

import com.jobportal.entity.User;

import jakarta.validation.constraints.Email;

import jakarta.validation.constraints.NotBlank;

import jakarta.validation.constraints.Pattern;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

/\*\*

\* Data Transfer Object for User entity.

\*/

@Data

@NoArgsConstructor

@AllArgsConstructor

public class UserDTO {

private Long id;

@NotBlank(message = "{user.name.absent}")

private String name;

@NotBlank(message = "{user.email.absent}")

@Email(message = "{user.email.invalid}")

private String email;

@NotBlank(message = "{user.password.absent}")

@Pattern(

regexp = "^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\\d)(?=.\*[@#$%^&+=!]).{8,20}$",

message = "{user.password.invalid}"

)

private String password;

private AccountType accountType;

private Long profileId;

/\*\*

\* Convert DTO to User entity.

\*/

public User toEntity() {

return new User(

this.id, this.name, this.email, this.password,

this.accountType, this.profileId);

}

}

**package** com.jobportal.entity;

**import** com.jobportal.dto.ApplicationStatus;

**import** lombok.Data;

**import** com.jobportal.dto.Applicant;

**import** java.time.LocalDateTime;

**import** org.springframework.data.annotation.Id;

@Data

**public** **class** Application {

@Id

**private** Long id;

**private** Applicant applicant;

**private** ApplicationStatus status;

**private** LocalDateTime applicationDate;

}

package com.jobportal.entity;

import java.time.LocalDateTime;

import java.util.List;

import org.springframework.data.annotation.Id;

import org.springframework.data.mongodb.core.mapping.Document;

import com.jobportal.dto.JobDTO;

import com.jobportal.dto.JobStatus;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Document(collection="jobs")

public class Job {

@Id

private Long id;

private String jobTitle;

private String company;

private List<Application> applications;

private String about;

private String experience;

private String jobType;

private String location;

private Long packageOffer;

private LocalDateTime postTime;

private String description;

private List<String> skillsRequired;

private JobStatus jobStatus;

public JobDTO toDTO() {

return new JobDTO(

this.id, this.jobTitle, this.company, null, // applicants not included to avoid circular reference

this.about, this.experience, this.jobType, this.location,

this.packageOffer, this.postTime, this.description,

this.skillsRequired, this.jobStatus

);

}

}

package com.jobportal.entity;

import java.time.LocalDateTime;

import org.springframework.data.annotation.Id;

import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "otps")

public class OTP {

@Id

private String id;

private String email;

private String code;

private LocalDateTime creationTime;

public OTP() {}

public OTP(String email, String code, LocalDateTime creationTime) {

this.email = email;

this.code = code;

this.creationTime = creationTime;

}

public String getId() { return id; }

public void setId(String id) { this.id = id; }

public String getEmail() { return email; }

public void setEmail(String email) { this.email = email; }

public String getCode() { return code; }

public void setCode(String code) { this.code = code; }

public LocalDateTime getCreationTime() { return creationTime; }

public void setCreationTime(LocalDateTime creationTime) { this.creationTime = creationTime; }

public boolean isExpired() {

return this.creationTime.plusMinutes(10).isBefore(LocalDateTime.now());

}

}

package com.jobportal.entity;

import org.springframework.data.annotation.Id;

import org.springframework.data.mongodb.core.mapping.Document;

import com.jobportal.dto.Certification;

import com.jobportal.dto.Experience;

import com.jobportal.dto.ProfileDTO;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

import java.util.List;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Document(collection = "profiles")

public class Profile {

@Id

private Long id;

private String email;

private String jobTitle;

private String company;

private String location;

private String about;

private List<String> skills;

private List<Experience> experiences;

private List<Certification> certifications;

private String avatar;

private String banner;

private String salary;

public ProfileDTO toDTO() {

return new ProfileDTO(

this.id, this.email, this.jobTitle, this.company, this.location,

this.about, this.skills, this.experiences, this.certifications,

this.avatar, this.banner, this.salary

);

}

}

package com.jobportal.entity;

import org.springframework.data.annotation.Id;

import org.springframework.data.mongodb.core.mapping.Document;

import lombok.Data;

@Document(collection="sequence")

@Data

public class Sequence {

@Id

private String id;

private Long seq;

}

package com.jobportal.entity;

import org.springframework.data.annotation.Id;

import org.springframework.data.mongodb.core.index.Indexed;

import org.springframework.data.mongodb.core.mapping.Document;

import com.jobportal.dto.AccountType;

import com.jobportal.dto.UserDTO;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

/\*\*

\* MongoDB document for User details.

\*/

@Data

@NoArgsConstructor

@AllArgsConstructor

@Document(collection = "users")

public class User {

@Id

private Long id;

private String name;

// Ensures email is unique

@Indexed(unique = true)

private String email;

private String password;

private AccountType accountType;

private Long profileId;

/\*\*

\* Convert User entity to UserDTO.

\*/

public UserDTO toDTO() {

return new UserDTO(

this.id, this.name, this.email, this.password,

this.accountType, this.profileId);

}

}

**package** com.jobportal.exception;

**public** **class** JobPortalException **extends** Exception {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** JobPortalException(String message) {

**super**(message);

}

}

package com.jobportal.repository;

import org.springframework.data.mongodb.repository.MongoRepository;

import com.jobportal.entity.Job;

public interface JobRepository extends MongoRepository<Job,Long> {

}

package com.jobportal.repository;

import java.time.LocalDateTime;

import java.util.List;

import java.util.Optional;

import org.springframework.data.mongodb.repository.MongoRepository;

import com.jobportal.entity.OTP;

public interface OTPRepository extends MongoRepository<OTP, String> {

Optional<OTP> findByEmail(String email);

List<OTP> findByCreationTimeBefore(LocalDateTime expiry);

}

package com.jobportal.repository;

import org.springframework.data.mongodb.repository.MongoRepository;

import com.jobportal.entity.Profile;

import java.util.Optional;

public interface ProfileRepository extends MongoRepository<Profile, Long> {

Optional<Profile> findByEmail(String email);

}

// ✅ UserRepository.java

package com.jobportal.repository;

import java.util.List;

import java.util.Optional;

import org.springframework.data.mongodb.repository.MongoRepository;

import com.jobportal.entity.User;

public interface UserRepository extends MongoRepository<User, Long> {

List<User> findByEmail(String email); // For admin use / check duplicates

Optional<User> findFirstByEmail(String email); // 💡 Enforce uniqueness for login/OTP/reset

}

package com.jobportal.service;

import java.util.List;

import com.jobportal.dto.JobDTO;

import com.jobportal.exception.JobPortalException;

public interface JobService {

public JobDTO postJob(JobDTO jobDTO) throws JobPortalException;

public List<JobDTO> getAllJobs();

public JobDTO getJob(Long id) throws JobPortalException;

}

package com.jobportal.service;

import java.time.LocalDateTime;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.jobportal.dto.JobDTO;

import com.jobportal.exception.JobPortalException;

import com.jobportal.repository.JobRepository;

import com.jobportal.utility.Utilities;

@Service("jobService")

public class JobServiceImpl implements JobService {

@Autowired

private JobRepository jobRepository;

@Override

public JobDTO postJob(JobDTO jobDTO) throws JobPortalException {

jobDTO.setId(Utilities.getNextSequence("jobs"));

jobDTO.setPostTime(LocalDateTime.now());

return jobRepository.save(jobDTO.toEntity()).toDTO();

}

@Override

public List<JobDTO> getAllJobs() {

// TODO Auto-generated method stub

return jobRepository.findAll()

.stream().map((x)->x.toDTO()).toList(); }

@Override

public JobDTO getJob(Long id) throws JobPortalException {

// TODO Auto-generated method stub

return jobRepository.findById(id).orElseThrow(()->new JobPortalException("JOB\_NOT\_FOUND")).toDTO();

}

}

package com.jobportal.service;

import com.jobportal.dto.ProfileDTO;

import com.jobportal.exception.JobPortalException;

public interface ProfileService {

Long createProfile(String email) throws JobPortalException;

ProfileDTO getProfile(Long id) throws JobPortalException;

ProfileDTO updateProfile(ProfileDTO profileDTO) throws JobPortalException;

}

// ProfileServiceImpl.java

package com.jobportal.service;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.jobportal.dto.ProfileDTO;

import com.jobportal.entity.Profile;

import com.jobportal.repository.ProfileRepository;

import com.jobportal.utility.Utilities;

import com.jobportal.exception.JobPortalException;

import java.util.ArrayList;

import java.util.Optional;

@Service("profileService")

public class ProfileServiceImpl implements ProfileService {

@Autowired

private ProfileRepository profileRepository;

@Override

public Long createProfile(String email) throws JobPortalException {

if (email == null || email.isEmpty()) {

throw new JobPortalException("Email is required");

}

Optional<Profile> existingProfile = profileRepository.findByEmail(email);

if (existingProfile.isPresent()) {

throw new JobPortalException("Profile already exists for this email");

}

Profile profile = new Profile();

profile.setId(Utilities.getNextSequence("profiles"));

profile.setEmail(email);

profile.setJobTitle("");

profile.setCompany("");

profile.setLocation("");

profile.setAbout("");

profile.setSkills(new ArrayList<>());

profile.setExperiences(new ArrayList<>());

profile.setCertifications(new ArrayList<>());

return profileRepository.save(profile).getId();

}

@Override

public ProfileDTO getProfile(Long id) throws JobPortalException {

if (id == null) {

throw new JobPortalException("Profile ID is required");

}

Profile profile = profileRepository.findById(id)

.orElseThrow(() -> new JobPortalException("Profile not found with ID: " + id));

ProfileDTO dto = new ProfileDTO();

dto.setId(profile.getId());

dto.setEmail(profile.getEmail());

dto.setJobTitle(profile.getJobTitle());

dto.setCompany(profile.getCompany());

dto.setLocation(profile.getLocation());

dto.setAbout(profile.getAbout());

dto.setSkills(profile.getSkills());

dto.setExperiences(profile.getExperiences());

dto.setCertifications(profile.getCertifications());

return dto;

}

@Override

public ProfileDTO updateProfile(ProfileDTO profileDTO) throws JobPortalException {

if (profileDTO.getId() == null) {

throw new JobPortalException("Profile ID is required");

}

Profile existingProfile = profileRepository.findById(profileDTO.getId())

.orElseThrow(() -> new JobPortalException("Profile not found with ID: " + profileDTO.getId()));

existingProfile.setJobTitle(profileDTO.getJobTitle());

existingProfile.setCompany(profileDTO.getCompany());

existingProfile.setLocation(profileDTO.getLocation());

existingProfile.setAbout(profileDTO.getAbout());

existingProfile.setSkills(profileDTO.getSkills());

existingProfile.setExperiences(profileDTO.getExperiences());

existingProfile.setCertifications(profileDTO.getCertifications());

Profile updatedProfile = profileRepository.save(existingProfile);

return getProfile(updatedProfile.getId());

}

}

package com.jobportal.service;

import com.jobportal.dto.LoginDTO;

import com.jobportal.dto.ResponseDTO;

import com.jobportal.dto.UserDTO;

import com.jobportal.exception.JobPortalException;

public interface UserService {

/\*\*

\* Registers a new user after validating uniqueness.

\*/

UserDTO registerUser(UserDTO userDTO) throws JobPortalException;

/\*\*

\* Authenticates a user by email and password.

\*/

UserDTO loginUser(LoginDTO loginDTO) throws JobPortalException;

/\*\*

\* Sends an OTP to the user's email.

\*/

boolean sendOtp(String email) throws Exception;

/\*\*

\* Verifies OTP for a user's email.

\*/

boolean verifyOtp(String email, String otp) throws JobPortalException;

/\*\*

\* Changes user password after OTP validation.

\*/

ResponseDTO changePassword(LoginDTO loginDTO) throws JobPortalException;

}

package com.jobportal.service;

import java.time.LocalDateTime;

import java.util.List;

import jakarta.mail.internet.MimeMessage;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.mail.javamail.JavaMailSender;

import org.springframework.mail.javamail.MimeMessageHelper;

import org.springframework.scheduling.annotation.Scheduled;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.stereotype.Service;

import com.jobportal.dto.LoginDTO;

import com.jobportal.dto.ResponseDTO;

import com.jobportal.dto.UserDTO;

import com.jobportal.entity.User;

import com.jobportal.entity.OTP;

import com.jobportal.exception.JobPortalException;

import com.jobportal.repository.OTPRepository;

import com.jobportal.repository.UserRepository;

import com.jobportal.utility.Data;

import com.jobportal.utility.Utilities;

@Service(value = "userService")

public class UserServiceImpl implements UserService {

@Autowired

private UserRepository userRepository;

@Autowired

private PasswordEncoder passwordEncoder;

@Autowired

private ProfileService profileService;

@Autowired

private OTPRepository otpRepository;

@Autowired

private JavaMailSender mailSender;

@Override

public UserDTO registerUser(UserDTO userDTO) throws JobPortalException {

List<User> users = userRepository.findByEmail(userDTO.getEmail());

if (!users.isEmpty()) {

throw new JobPortalException("A user with this email already exists.");

}

userDTO.setProfileId(profileService.createProfile(userDTO.getEmail()));

userDTO.setId(Utilities.getNextSequence("users"));

userDTO.setPassword(passwordEncoder.encode(userDTO.getPassword()));

User user = userDTO.toEntity();

User savedUser = userRepository.save(user);

return savedUser.toDTO();

}

@Override

public UserDTO loginUser(LoginDTO loginDTO) throws JobPortalException {

List<User> users = userRepository.findByEmail(loginDTO.getEmail());

if (users.isEmpty()) {

throw new JobPortalException("User not found with this email.");

} else if (users.size() > 1) {

throw new JobPortalException("Multiple accounts found with this email. Contact support.");

}

User user = users.get(0);

if (!passwordEncoder.matches(loginDTO.getPassword(), user.getPassword())) {

throw new JobPortalException("Invalid email or password.");

}

return user.toDTO();

}

@Override

public boolean sendOtp(String email) throws JobPortalException {

List<User> users = userRepository.findByEmail(email);

if (users.isEmpty()) {

throw new JobPortalException("No user found with this email.");

} else if (users.size() > 1) {

throw new JobPortalException("Multiple accounts found with this email. Contact support.");

}

User user = users.get(0);

String generatedOtp = Utilities.generateOTP();

OTP otp = new OTP(email, generatedOtp, LocalDateTime.now());

otpRepository.save(otp);

try {

MimeMessage message = mailSender.createMimeMessage();

MimeMessageHelper helper = new MimeMessageHelper(message, true);

helper.setTo(email);

helper.setSubject("CareerSync Verification: Your OTP Code");

helper.setText(Data.getMessageBody(user.getName(), generatedOtp), true);

mailSender.send(message);

return true;

} catch (Exception e) {

e.printStackTrace();

throw new JobPortalException("Failed to send OTP email. Please try again later.");

}

}

@Override

public boolean verifyOtp(String email, String otpCode) throws JobPortalException {

OTP otpData = otpRepository.findByEmail(email)

.orElseThrow(() -> new JobPortalException("OTP not found for this email."));

if (!otpData.getCode().equals(otpCode)) {

throw new JobPortalException("The OTP you entered is incorrect.");

}

if (otpData.isExpired()) {

throw new JobPortalException("The OTP has expired. Please request a new one.");

}

return true;

}

@Override

public ResponseDTO changePassword(LoginDTO loginDTO) throws JobPortalException {

List<User> users = userRepository.findByEmail(loginDTO.getEmail());

if (users.isEmpty()) {

throw new JobPortalException("No user found with this email.");

} else if (users.size() > 1) {

throw new JobPortalException("Multiple accounts found with this email. Contact support.");

}

User user = users.get(0);

user.setPassword(passwordEncoder.encode(loginDTO.getPassword()));

userRepository.save(user);

return new ResponseDTO("Password changed successfully.");

}

@Scheduled(fixedRate = 60000)

public void removeExpiredOTPs() {

LocalDateTime expiryTime = LocalDateTime.now().minusMinutes(10);

List<OTP> expiredOTPs = otpRepository.findByCreationTimeBefore(expiryTime);

if (!expiredOTPs.isEmpty()) {

otpRepository.deleteAll(expiredOTPs);

System.out.println("Cleaned up " + expiredOTPs.size() + " expired OTP(s).");

}

}

}

**package** com.jobportal.utility;

**public** **class** Data {

**public** **static** String getMessageBody(String name, String otp) {

**return** "<!DOCTYPE html>" +

"<html><head><meta charset='UTF-8'><title>CareerSync OTP Verification</title></head>" +

"<body style='font-family: Arial, sans-serif; background-color: #f4f8fb; padding: 20px;'>" +

"<div style='max-width: 600px; margin: auto; background: white; padding: 30px; border-radius: 12px; box-shadow: 0 4px 8px rgba(0,0,0,0.05);'>" +

"<h2 style='color: #2c3e50;'>Hello, " + name + " 👋</h2>" +

"<p style='font-size: 16px; color: #333;'>Welcome to <strong style='color:#1a73e8;'>CareerSync</strong>! Your One-Time Password (OTP) is:</p>" +

"<div style='font-size: 30px; font-weight: bold; color: #1a73e8; margin: 20px 0; text-align:center;'>" + otp + "</div>" +

"<p style='font-size: 14px; color: #555;'>This OTP will expire in 10 minutes. Please do not share it with anyone.</p>" +

"<hr style='margin: 30px 0;'>" +

"<p style='font-size: 12px; color: #999;'>If you did not request this, simply ignore this message. Thank you for choosing <strong>CareerSync</strong>.</p>" +

"</div></body></html>";

}

}

package com.jobportal.utility;

import java.time.LocalDateTime;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

public class ErrorInfo {

private String errorMessage;

private Integer errorCode;

private LocalDateTime timeStamp;

}

package com.jobportal.utility;

import java.time.LocalDateTime;

import java.util.stream.Collectors;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.core.env.Environment;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.validation.ObjectError;

import org.springframework.web.bind.MethodArgumentNotValidException;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.bind.annotation.RestControllerAdvice;

import com.jobportal.exception.JobPortalException;

import jakarta.validation.ConstraintViolation;

import jakarta.validation.ConstraintViolationException;

@RestControllerAdvice

public class ExceptionControllerAdvice {

@Autowired

private Environment environment;

// Handle generic exceptions

@ExceptionHandler(Exception.class)

public ResponseEntity<ErrorInfo> generalException(Exception exception) {

ErrorInfo error = new ErrorInfo(

exception.getMessage(),

HttpStatus.INTERNAL\_SERVER\_ERROR.value(),

LocalDateTime.now()

);

return new ResponseEntity<>(error, HttpStatus.INTERNAL\_SERVER\_ERROR);

}

@ExceptionHandler(JobPortalException.class)

public ResponseEntity<ErrorInfo> generalException(JobPortalException exception) {

String msg=environment.getProperty(exception.getMessage());

ErrorInfo error = new ErrorInfo(

msg,HttpStatus.INTERNAL\_SERVER\_ERROR.value(),

LocalDateTime.now()

);

return new ResponseEntity<>(error, HttpStatus.INTERNAL\_SERVER\_ERROR);

}

// Handle validation exceptions (both types)

@ExceptionHandler({MethodArgumentNotValidException.class, ConstraintViolationException.class})

public ResponseEntity<ErrorInfo> validatorExceptionHandler(Exception exception) {

String msg = "";

if (exception instanceof MethodArgumentNotValidException manvException) {

msg = manvException.getAllErrors()

.stream()

.map(ObjectError::getDefaultMessage)

.collect(Collectors.joining(", "));

} else if (exception instanceof ConstraintViolationException cvException) {

msg = cvException.getConstraintViolations()

.stream()

.map(ConstraintViolation::getMessage)

.collect(Collectors.joining(", "));

}

ErrorInfo error = new ErrorInfo(

msg,

HttpStatus.BAD\_REQUEST.value(),

LocalDateTime.now()

);

return new ResponseEntity<>(error, HttpStatus.BAD\_REQUEST);

}

}

package com.jobportal.utility;

import jakarta.annotation.PostConstruct;

import java.security.SecureRandom;

import org.springframework.data.mongodb.core.FindAndModifyOptions;

import org.springframework.data.mongodb.core.MongoOperations;

import org.springframework.data.mongodb.core.query.Criteria;

import org.springframework.data.mongodb.core.query.Query;

import org.springframework.data.mongodb.core.query.Update;

import org.springframework.stereotype.Component;

import com.jobportal.entity.Sequence;

import com.jobportal.exception.JobPortalException;

/\*\*

\* Utility class for generating sequential IDs using MongoDB.

\* Provides static methods for both sequence generation and OTP code creation.

\*/

@Component

public class Utilities {

// Static reference to MongoOperations for static method access

private static MongoOperations staticMongoOperations;

// Instance MongoOperations, injected by Spring

private final MongoOperations mongoOperations;

/\*\*

\* Constructor-based injection of MongoOperations.

\* This enables MongoDB operations through Spring's dependency injection.

\*

\* @param mongoOperations the MongoOperations bean injected by Spring

\*/

public Utilities(MongoOperations mongoOperations) {

this.mongoOperations = mongoOperations;

}

/\*\*

\* Initializes the static MongoOperations reference after the bean is constructed.

\* Allows static methods to use MongoOperations.

\*/

@PostConstruct

public void init() {

staticMongoOperations = this.mongoOperations;

}

/\*\*

\* Static method to get the next sequence number for a given key.

\* Uses MongoDB's atomic findAndModify to safely increment sequences.

\*

\* @param key the name of the sequence (e.g., "jobId", "userId")

\* @return next sequence number

\* @throws JobPortalException if the sequence cannot be generated

\*/

public static Long getNextSequence(String key) throws JobPortalException {

Query query = new Query(Criteria.where("\_id").is(key)); // Match document by key

Update update = new Update().inc("seq", 1); // Increment sequence field

FindAndModifyOptions options = FindAndModifyOptions.options()

.returnNew(true) // Return updated document

.upsert(true); // Insert if absent

Sequence sequence = staticMongoOperations.findAndModify(

query, update, options, Sequence.class

);

// Throw exception if sequence could not be generated

if (sequence == null) {

throw new JobPortalException("Unable to get sequence ID for key: " + key);

}

return sequence.getSeq(); // Return the updated sequence number

}

/\*\*

\* Generates a secure random 6-digit OTP code.

\* Uses SecureRandom for cryptographic strength randomness.

\*

\* @return A 6-digit OTP code as String.

\*/

public static String generateOTP() {

StringBuilder otp = new StringBuilder();

SecureRandom random = new SecureRandom();

for (int i = 0; i < 6; i++) {

otp.append(random.nextInt(10)); // Append single digit [0-9]

}

return otp.toString(); // Return the 6-digit OTP code

}

}

# 🌐 Spring Boot Application Name

spring.application.name=Job-Portal

# 🗄 MongoDB Configuration

spring.data.mongodb.auto-index-creation=true

spring.data.mongodb.uri=mongodb://localhost:27017/jobportal

spring.data.mongodb.database=jobportal

# 📧 Email Server Settings (Gmail SMTP)

spring.mail.host=smtp.gmail.com

spring.mail.port=587

spring.mail.username=Vickyprem8340@gmail.com

spring.mail.password=axbonhbqorvl wphj

spring.mail.properties.mail.smtp.auth=true

spring.mail.properties.mail.smtp.starttls.enable=true

# 💡 Custom Application Messages

USER\_FOUND=User has registered already.

USER\_NOT\_FOUND=USER is not registered.

INVALID\_CREDENTIALS=Invalid Credentials.

OTP\_NOT\_FOUND=OTP is expired

OTP\_INCORRECT=OTP is incorrect

JOB\_NOT\_FOUND=Job not found

user.name.absent=Name is null or blank.

user.email.absent=Email is null or blank.

user.email.invalid=Email is invalid.

user.password.absent=Password is null or blank.

user.password.invalid=Password is invalid.

otp.invalid=OTP is invalid.