

```
// src/api/axios.js

import axios from "axios";

const RAW_BASE =
  (import.meta && import.meta.env && import.meta.env.VITE_API_BASE) ||
  "http://localhost:5000/api";

const BASE_URL = RAW_BASE.replace(/\//g, "");
export const SOCKET_ORIGIN = BASE_URL.replace(/api/, "");

const API = axios.create({
  baseURL: BASE_URL,
  withCredentials: false,
  timeout: 20000,
});

// ----- helpers -----
function absUrl(config) {
  const base = config.baseURL || "";
  const url = config.url || "";
  if (/^https?:\/\/i.test(url)/) return url;
  return `${base}${url.startsWith("/") ? "" : "/"}` + url;
}

function isAdminApi(url) {
  return /\api\/(auth\|admin)\b/i.test(url);
}

function isCompanyApi(url) {
  return /\api\company\b/i.test(url);
}

function isAdminPage() {
  try { return typeof window !== "undefined" && window.location.pathname.startsWith("/admin"); }
}
```

```
        catch { return false; }

    }

// ----- REQUEST -----
API.interceptors.request.use((config) => {

    const url = absUrl(config);

    const adminToken = localStorage.getItem("adminToken");

    const userToken = localStorage.getItem("userToken");

    let token;

    if (isAdminApi(url)) {

        token = adminToken; // admin APIs -> admin token

    } else if (isCompanyApi(url)) {

        token = (isAdminPage() ? adminToken : null) || userToken; // company APIs on admin pages use admin token

    } else {

        token = userToken;

    }

    if (token) {

        config.headers = config.headers || {};

        config.headers.Authorization = `Bearer ${token}`;

    }

    return config;

});

// ----- RESPONSE -----
let redirecting = false;
API.interceptors.response.use(
    (res) => res,
    (err) => {
```

```

const status = err?.response?.status;
const url = absUrl(err?.config || {});
const adminApi = isAdminApi(url);
const onAdmin = isAdminPage();

if ((status === 401 || status === 403) && !redirecting) {
  if (adminApi) {
    redirecting = true;
    try {
      localStorage.removeItem("adminToken");
      localStorage.removeItem("adminUser");
      localStorage.removeItem("adminEmail");
      window.location.replace("/admin/login");
    } finally {}
  } else {
    if (onAdmin) return Promise.reject(err); // stay on admin UI, show error
    redirecting = true;
    try {
      localStorage.removeItem("userToken");
      localStorage.removeItem("userRole");
      window.location.replace("/login");
    } finally {}
  }
  return Promise.reject(err);
};

export default API;

```

---

```
import { createContext, useContext, useState } from "react";
```

```
// Create Context

const TabsContext = createContext();

export function Tabs({ children, defaultValue }) {
  const [selectedTab, setSelectedTab] = useState(defaultValue || "");

  return (
    <TabsContext.Provider value={{ selectedTab, setSelectedTab }}>
      <div className="w-full">{children}</div>
    </TabsContext.Provider>
  );
}

export function TabsList({ children }) {
  return <div className="flex border-b mb-4">{children}</div>;
}

export function TabsTrigger({ children, value }) {
  const { selectedTab, setSelectedTab } = useContext(TabsContext);
  const isActive = selectedTab === value;

  return (
    <button
      onClick={() => setSelectedTab(value)}
      className={`px-4 py-2 text-sm font-medium transition-colors ${{
        isActive
          ? "border-b-2 border-blue-600 text-blue-600"
          : "text-gray-500 hover:text-blue-500"
      }}`}
    >

```

```
{children}  
</button>  
);  
}  
  
export function TabsContent({ value, children }) {  
  const { selectedTab } = useContext(TabsContext);  
  return selectedTab === value ? <div className="mt-4">{children}</div> : null;  
}
```

---

```
// File: components/university/StudentAnalyticsCard.jsx  
import { GraduationCap, BarChart2 } from "lucide-react";  
  
export default function StudentAnalyticsCard() {  
  return (  
    <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-6 mt-6">  
      <div className="bg-white p-5 rounded-xl shadow hover:shadow-md transition">  
        <div className="flex items-center gap-3">  
          <GraduationCap className="text-blue-600" />  
          <h4 className="text-gray-700 font-semibold">Total Students</h4>  
        </div>  
        <p className="text-2xl font-bold mt-2 text-blue-800">3,200+</p>  
      </div>  
  
      <div className="bg-white p-5 rounded-xl shadow hover:shadow-md transition">  
        <div className="flex items-center gap-3">  
          <BarChart2 className="text-green-600" />  
          <h4 className="text-gray-700 font-semibold">Placement Rate</h4>  
        </div>  
        <p className="text-2xl font-bold mt-2 text-green-700">88%</p>  
      </div>  
    </div>  
  );  
}
```

```
<div className="bg-white p-5 rounded-xl shadow hover:shadow-md transition">
  <div className="flex items-center gap-3">
    <GraduationCap className="text-purple-600" />
    <h4 className="text-gray-700 font-semibold">Active Internships</h4>
  </div>
  <p className="text-2xl font-bold mt-2 text-purple-700">120+</p>
</div>

<div className="bg-white p-5 rounded-xl shadow hover:shadow-md transition">
  <div className="flex items-center gap-3">
    <BarChart2 className="text-yellow-600" />
    <h4 className="text-gray-700 font-semibold">Industry Partners</h4>
  </div>
  <p className="text-2xl font-bold mt-2 text-yellow-700">45</p>
</div>
</div>
);

}
```

---

```
// src/components/AdminRoute.jsx

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

export default function AdminRoute({ children }) {
  const token = localStorage.getItem("adminToken");
  if (!token) return <Navigate to="/admin/login" replace />;
  return children;
}
```

---

```
import { useEffect, useState, useMemo } from "react";
import API from "../api/axios";
```

```
import { BrainCircuit, Sparkles, Target, Briefcase, CalendarDays, MessageSquare, ArrowUpRight, Loader2 } from "lucide-react";
import { Link } from "react-router-dom";

const Card = ({ children, className="" }) => (
  <div className={'rounded-2xl border bg-white/80 backdrop-blur p-5 ${className}' } />
);

const Chip = ({ children, tone="sky" }) => (
  <span className={'inline-flex items-center px-2.5 py-0.5 rounded-full text-xs font-medium bg-${tone}-50 text-${tone}-700 border border-${tone}-200`}>
    {children}
  </span>
);

export default function AICareerGuidance() {
  const [data, setData] = useState(null);
  const [ask, setAsk] = useState("");
  const [answer, setAnswer] = useState("");
  const [loading, setLoading] = useState(true);
  const [asking, setAsking] = useState(false);

  const load = async () => {
    setLoading(true);
    try {
      const res = await API.get("/student/ai/guidance");
      setData(res.data || {});
    } catch {
      setData(null);
    } finally {
      setLoading(false);
    }
  }
}
```

```
};

useEffect(()=>{ load(); }, []);

const readinessPct = useMemo(() => {
  const r = data?.metrics?.readiness ?? 0;
  return Math.round((r/10)*100);
}, [data]);

const askAI = async (e) => {
  e.preventDefault();
  if (!ask.trim()) return;
  setAsking(true);
  setAnswer("");
  try {
    const res = await API.post("/student/ai/ask", { q: ask.trim() });
    setAnswer(res.data?.answer || "");
  } finally {
    setAsking(false);
  }
};

if (loading) {
  return (
    <div className="max-w-6xl mx-auto p-6">
      <Card className="grid place-items-center h-48 text-slate-500">
        <Loader2 className="animate-spin" size={18} /> <span className="ml-2">Preparing your
        guidance...</span>
      </Card>
    </div>
  );
}
```

```
}

if (!data) {
  return (
    <div className="max-w-6xl mx-auto p-6">
      <Card className="text-center">
        <h3 className="font-semibold">No guidance available</h3>
        <p className="text-sm text-slate-600 mt-1">Try taking a few skill tests or apply to roles to unlock personalized insights.</p>
      </Card>
    </div>
  );
}

const strengths = data.metrics?.strengths || [];
const gaps = data.metrics?.gaps || [];
const matches = data.jobMatches || [];
const plan = data.weeklyPlan || [];
const gapRes = data.gapResources || [];
const hacks = data.upcomingHackathons || [];
const coachNote = data.coachNote;

return (
  <div className="max-w-7xl mx-auto p-6 space-y-6">
    {/* header */}
    <div className="flex items-center justify-between">
      <div className="flex items-center gap-3">
        <div className="h-10 w-10 rounded-xl bg-sky-50 text-sky-700 border border-sky-200 grid place-items-center">
          <BrainCircuit size={18}/>
        </div>
        <div>
```

```
<h1 className="text-xl font-semibold tracking-tight">AI Career Guidance</h1>
<p className="text-xs text-slate-600">Tailored from your skills, activity & goals</p>
</div>
</div>

<div className="hidden sm:flex items-center gap-2">
  <div className="text-right">
    <div className="text-xs text-slate-500">Readiness</div>
    <div className="text-lg font-semibold">{readinessPct}%</div>
  </div>
  <div className="h-8 w-28 bg-slate-100 rounded-full overflow-hidden">
    <div className="h-full bg-gradient-to-r from-sky-300 to-emerald-300"
      style={{width: `${readinessPct}%`}} />
  </div>
  </div>
</div>

/* coach note */
{coachNote && (
  <Card className="border-sky-200">
    <div className="flex items-start gap-3">
      <Sparkles className="text-sky-600 mt-0.5" size={18}/>
      <div>
        <div className="font-medium">Coach Note</div>
        <p className="text-sm text-slate-700 mt-1 whitespace-pre-wrap">{coachNote}</p>
      </div>
    </div>
  </Card>
)};

/* grid */
<div className="grid lg:grid-cols-3 gap-4">
```

```

/* strengths/gaps */

<Card className="lg:col-span-1">
  <div className="flex items-center gap-2 mb-3">
    <Target className="text-emerald-600" size={18}/>
    <h3 className="font-semibold">Strengths & Gaps</h3>
  </div>
  <div className="space-y-3">
    <div>
      <div className="text-xs text-slate-500 mb-1">Strengths</div>
      {strengths.length ? (
        <div className="flex flex-wrap gap-2">
          {strengths.map(s => <Chip key={s} tone="emerald">{s}</Chip>)}
        </div>
      ) : <div className="text-sm text-slate-500">No strengths detected yet.</div>}
    </div>
    <div>
      <div className="text-xs text-slate-500 mb-1">Gaps</div>
      {gaps.length ? (
        <div className="flex flex-wrap gap-2">
          {gaps.map(s => <Chip key={s} tone="rose">{s}</Chip>)}
        </div>
      ) : <div className="text-sm text-slate-500">No gaps detected. Keep going!</div>}
    </div>
  </div>

  {gapRes.length>0 && (
    <div className="mt-4">
      <div className="text-xs text-slate-500 mb-1">Suggested Resources</div>
      <ul className="space-y-2 text-sm">
        {gapRes.map(gr => (
          <li key={gr.skill} className="rounded-lg p-2 bg-slate-50/60">

```

```

<div className="font-medium">{gr.skill}</div>

{gr.suggestions?.length ? (
  <ul className="list-disc ml-5 mt-1 space-y-1">
    {gr.suggestions.map((r,i)=>(
      <li key={i}>
        <a className="text-sky-700 underline" href={r.link} target="_blank"
        rel="noreferrer">{r.title}</a>
      </li>
    )))
  </ul>
) : <div className="text-slate-500">Add resources in roadmap to surface here.</div>}
</li>
))}

</ul>
</div>
)}

</Card>

/* job matches */

<Card className="lg:col-span-2">
  <div className="flex items-center justify-between mb-3">
    <div className="flex items-center gap-2">
      <Briefcase className="text-indigo-600" size={18}/>
      <h3 className="font-semibold">Best Job Matches</h3>
    </div>
    <Link to="/student/jobs" className="text-sm text-indigo-700 underline">Browse all</Link>
  </div>
  {matches.length==0 ? (
    <div className="text-sm text-slate-500">No matches yet. Try adding skills or taking
    tests.</div>
  ) : (
    <div className="grid md:grid-cols-2 gap-3">

```

```

{matches.map(m=>
  <div key={m._id} className="rounded-xl border p-4 bg-white hover:shadow-sm transition">
    <div className="flex items-start justify-between gap-3">
      <div>
        <div className="font-medium">{m.title}</div>
        <div className="text-xs text-slate-500">{m.company || "—" • {m.location}}</div>
      </div>
      <Chip tone="violet">{m.type}</Chip>
    </div>
    <div className="mt-2 text-xs text-slate-600">
      Skills: {(m.skills || []).join(", ") || "—"}
    </div>
    <div className="mt-3 flex items-center justify-between">
      <div className="text-xs text-slate-500">Fit score</div>
      <div className="font-semibold">{m.fit}</div>
    </div>
    <div className="mt-3 flex justify-end">
      <Link to="/student/jobs" className="inline-flex items-center gap-1 text-sm text-indigo-700 underline">
        Apply <ArrowUpRight size={14}/>
      </Link>
    </div>
  </div>
))}

</div>
)
</Card>
</div>

/* weekly plan + hackathons + ask ai */

<div className="grid lg:grid-cols-3 gap-4">
```

```

<Card className="lg:col-span-1">
  <div className="flex items-center gap-2 mb-3">
    <CalendarDays className="text-fuchsia-600" size={18}/>
    <h3 className="font-semibold">Weekly Plan</h3>
  </div>
  <ul className="text-sm space-y-2">
    {plan.map((p,i)=>(
      <li key={i} className="flex items-center justify-between rounded-lg bg-slate-50/70 p-2">
        <span className="font-medium">{p.day}</span>
        <span className="text-slate-600">{p.task}</span>
        <span className="text-slate-500">{p.minutes}m</span>
      </li>
    )))
  </ul>
</Card>

```

```

<Card className="lg:col-span-1">
  <div className="flex items-center gap-2 mb-3">
    <Sparkles className="text-amber-600" size={18}/>
    <h3 className="font-semibold">Upcoming Hackathons</h3>
  </div>
  {hacks.length==0 ? (
    <div className="text-sm text-slate-500">No upcoming events.</div>
  ) : (
    <ul className="text-sm space-y-2">
      {hacks.map(h=>(
        <li key={h._id} className="rounded-lg p-2 bg-slate-50/60">
          <div className="font-medium">{h.title}</div>
          <div className="text-xs text-slate-600">{new Date(h.startAt).toLocaleString()}</div>
          <div className="mt-1">

```

```

        <Link to={`/student/hackathons/${h._id}`} className="text-amber-700 underline text-xs">View</Link>
      </div>
    </li>
  ))}
</ul>
)
</Card>

<Card className="lg:col-span-1">
  <div className="flex items-center gap-2 mb-3">
    <MessageSquare className="text-emerald-600" size={18}/>
    <h3 className="font-semibold">Ask AI</h3>
  </div>
  <form onSubmit={askAI} className="space-y-2">
    <textarea
      value={ask}
      onChange={e=>setAsk(e.target.value)}
      rows={4}
      className="w-full border rounded-xl px-3 py-2 bg-white/70 focus:outline-none focus:ring-2 focus:ring-emerald-200"
      placeholder="Ask about interview prep, skill gaps, or which roles to target...">
    />
    <button disabled={asking} className="w-full rounded-xl bg-emerald-600 text-white py-2 hover:bg-emerald-700">
      {asking ? "Thinking..." : "Ask"}
    </button>
  </form>
  {answer && (
    <div className="mt-3 text-sm text-slate-700 whitespace-pre-wrap bg-emerald-50/60 border border-emerald-200 rounded-xl p-3">
      {answer}
    </div>
  )
)
</Card>

```

```
        </div>
    )}
</Card>
</div>
</div>
);
}
```

---

```
import { useEffect, useMemo, useState } from "react";
import API from "../api/axios";
import {
    BrainCircuit, ChevronDown, CheckCircle2, Circle, Link as LinkIcon, BookOpen
} from "lucide-react";

/* ----- tiny UI helpers ----- */
const Card = ({ children, className = "" }) => (
    <div className={`bg-white/80 backdrop-blur rounded-2xl border p-6 ${className}`}>{children}</div>
);

const Chip = ({ children, tone = "blue", className = "" }) => (
    <span className={`inline-flex items-center gap-1 px-2.5 py-1 rounded-full text-xs ring-1 bg-${tone}-50 text-${tone}-700 ring-${tone}-200 ${className}`}>{children}</span>
);

export default function CareerRoadmap() {
    const [roadmap, setRoadmap] = useState({});
    const [loading, setLoading] = useState(true);
    const [open, setOpen] = useState({}); // which skill sections are expanded
    const [completed, setCompleted] = useState(() => {
        try {
            return JSON.parse(localStorage.getItem("roadmap.completed")) || "{}";
        }
    });
}
```

```
    } catch { return {}; }

});

useEffect(() => {
  const fetchRoadmap = async () => {
    try {
      const res = await API.get("/student/career-roadmap");
      const data = res.data?.roadmap || {};
      setRoadmap(data);

      // open the first skill by default
      const firstKey = Object.keys(data)[0];
      setOpen((o) => (firstKey ? { ...o, [firstKey]: true } : o));
    } catch (err) {
      console.error("Failed to fetch roadmap", err);
    } finally {
      setLoading(false);
    }
  };
  fetchRoadmap();
}, []);

// persist completion
useEffect(() => {
  localStorage.setItem("roadmap.completed", JSON.stringify(completed));
}, [completed]);

const skillNames = Object.keys(roadmap);

const toggleOpen = (k) => setOpen((o) => ({ ...o, [k]: !o[k] }));
```

```

const toggleStep = (skill, stepIdx) => {
  setCompleted((c) => {
    const prev = new Set(c[skill] || []);
    if (prev.has(stepIdx)) prev.delete(stepIdx);
    else prev.add(stepIdx);
    return { ...c, [skill]: Array.from(prev) };
  });
};

const resetSkill = (skill) =>
  setCompleted((c) => ({ ...c, [skill]: [] }));

const markAllSkill = (skill) =>
  setCompleted((c) => ({
    ...c,
    [skill]: roadmap[skill]?.map((_, idx) => idx) || [],
  }));

if (loading) {
  return (
    <div className="max-w-6xl mx-auto py-10 px-4 space-y-6">
      <Card className="animate-pulse">
        <div className="h-6 w-56 rounded bg-slate-200" />
        <div className="mt-4 h-24 w-full rounded bg-slate-200" />
      </Card>
    </div>
  );
}

if (!skillNames.length) {
  return (

```

```

<div className="max-w-5xl mx-auto p-10 text-center">
  <div className="mx-auto h-12 w-12 rounded-xl bg-blue-50 text-blue-700 grid place-items-center">
    <BrainCircuit />
  </div>
  <h2 className="mt-3 text-lg font-semibold">No personalized roadmap yet</h2>
  <p className="text-sm text-gray-600">Start taking skill tests to unlock your tailored plan.</p>
</div>
);

}

return (
<div className="max-w-6xl mx-auto py-8 px-4 space-y-6">
  <div className="flex items-center justify-between">
    <h2 className="text-2xl font-semibold tracking-tight flex items-center gap-2">
      <BrainCircuit className="text-blue-600" /> Career Roadmap
    </h2>
    <Chip tone="blue">Guided • Actionable • Minimal</Chip>
  </div>
</div>

{skillNames.map((skill) => {
  const steps = Array.isArray(roadmap[skill]) ? roadmap[skill] : [];
  const doneList = new Set(completed[skill] || []);
  const done = doneList.size;
  const total = steps.length;
  const pct = total ? Math.round((done / total) * 100) : 0;

  return (
    <Card key={skill} className="overflow-hidden">
      {/* header */}
      <button

```

```
onClick={() => toggleOpen(skill)}
```

```
className="w-full flex items-center justify-between text-left"
```

```
>
```

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

```
<div className="h-9 w-9 rounded-xl bg-blue-50 text-blue-700 grid place-items-center ring-1 ring-blue-200">
```

```
<BookOpen size={16} />
```

```
</div>
```

```
<div>
```

```
<div className="font-semibold">{skill} Roadmap</div>
```

```
<div className="text-xs text-gray-600">{done}/{total} steps completed</div>
```

```
</div>
```

```
</div>
```

```
</div>
```

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

```
<span className="text-xs text-gray-600">{pct}%</span>
```

```
<div className="w-28 h-2 bg-gray-100 rounded-full overflow-hidden">
```

```
<div className="h-full bg-blue-500" style={{ width: `${pct}` }} />
```

```
</div>
```

```
<ChevronDown
```

```
size={18}
```

```
className={`text-gray-500 transition-transform ${open[skill] ? "rotate-180" : ""}`}
```

```
/>
```

```
</div>
```

```
</button>
```

```
/* body */
```

```
{open[skill] && (
```

```
<div className="mt-5">
```

```
<div className="flex items-center gap-2 mb-4">
```

```
<button
```

```
    onClick={() => markAllSkill(skill)}
```

```
    className="px-3 py-1.5 rounded-lg border text-sm hover:bg-gray-50"
```

```
>
```

```
    Mark all done
```

```
</button>
```

```
<button
```

```
    onClick={() => resetSkill(skill)}
```

```
    className="px-3 py-1.5 rounded-lg border text-sm hover:bg-gray-50"
```

```
>
```

```
    Reset
```

```
</button>
```

```
</div>
```

  

```
<ul className="space-y-4">
```

```
{steps.map((step, idx) => {
```

```
    const isDone = doneList.has(idx);
```

```
    return (
```

```
        <li
```

```
            key={`${skill}-${idx}`}
```

```
            className={`rounded-xl border p-4 ${
```

```
                isDone
```

```
                    ? "bg-emerald-50/50 border-emerald-200"
```

```
                    : "bg-blue-50/30 border-blue-100"
```

```
                }`}
```

```
>
```

```
        <div className="flex items-start justify-between gap-3">
```

```
            <div className="flex items-start gap-3">
```

```
                <button
```

```
                    onClick={() => toggleStep(skill, idx)}
```

```
                    className={`mt-0.5 h-6 w-6 rounded-full grid place-items-center`}
```

```
                    ${isDone ? "text-emerald-600" : "text-gray-400"}`}
```

```
        title={isDone ? "Completed" : "Mark completed"}
```

```
>
```

```
    {isDone ? <CheckCircle2 /> : <Circle />}
```

```
</button>
```

```
<div>
```

```
    <div className="text-xs uppercase tracking-wide text-blue-700 font-bold">
```

```
        {step.level}
```

```
</div>
```

```
    <h4 className="font-semibold leading-tight mt-0.5">{step.title}</h4>
```

```
    {step.description && (
```

```
        <p className="text-sm text-gray-600 mt-1">{step.description}</p>
```

```
    )}
```

```
</div>
```

```
</div>
```

```
<Chip tone={isDone ? "emerald" : "slate"}>
```

```
    {isDone ? "Done" : "Planned"}
```

```
</Chip>
```

```
</div>
```

```
{/* topics */}
```

```
{Array.isArray(step.topics) && step.topics.length > 0 && (
```

```
    <div className="mt-3">
```

```
        <p className="text-xs font-semibold text-gray-700 mb-1">Topics</p>
```

```
        <ul className="text-sm text-gray-700 list-disc pl-5 space-y-1">
```

```
            {step.topics.map((t, i) => (
```

```
                <li key={i}>{t}</li>
```

```
            ))}
```

```
        </ul>
```

```
</div>
```

```
)}
```

```
/* resources */

{Array.isArray(step.resources) && step.resources.length > 0 && (
  <div className="mt-3">
    <p className="text-xs font-semibold text-gray-700 mb-1">Resources</p>
    <ul className="text-sm text-blue-700 space-y-1">
      {step.resources.map((r, i) => (
        <li key={i} className="flex items-center gap-2">
          <LinkIcon size={14} className="text-blue-500" />
          <a
            href={r.link}
            target="_blank"
            rel="noreferrer"
            className="underline hover:text-blue-900"
          >
            {r.title}
          </a>
        </li>
      )));
    </ul>
  </div>
);
})}

</li>
</ul>
</div>
)
);
});
```

```
</div>  
);  
}  
  
-----
```

```
import { FaFacebookF, FaLinkedinIn, FaTwitter } from "react-icons/fa";  
  
function Footer() {  
  return (  
    <footer className="bg-gradient-to-tr from-slate-900 to-gray-800 text-white pt-12 pb-6">  
      <div className="max-w-7xl mx-auto px-6 grid grid-cols-1 sm:grid-cols-2 md:grid-cols-4 gap-10 text-sm">  
  
        {/* Brand */}  
        <div>  
          <h3 className="text-xl font-bold text-blue-400 mb-3">StudentConnect</h3>  
          <p className="text-gray-300">  
            Bridging students, universities & companies through AI-powered career tools.  
          </p>  
        </div>  
  
        {/* Navigation */}  
        <div>  
          <h4 className="text-lg font-semibold mb-3">Quick Links</h4>  
          <ul className="space-y-2 text-gray-300">  
            <li><a href="#" className="hover:text-white transition">About</a></li>  
            <li><a href="#" className="hover:text-white transition">Blog</a></li>  
            <li><a href="#" className="hover:text-white transition">Help Center</a></li>  
            <li><a href="#" className="hover:text-white transition">Privacy Policy</a></li>  
          </ul>  
        </div>  
    </div>  
  </footer>  
}
```

```
/* Resources */

<div>

  <h4 className="text-lg font-semibold mb-3">Resources</h4>
  <ul className="space-y-2 text-gray-300">
    <li><a href="#" className="hover:text-white transition">Universities</a></li>
    <li><a href="#" className="hover:text-white transition">Companies</a></li>
    <li><a href="#" className="hover:text-white transition">Student Toolkit</a></li>
    <li><a href="#" className="hover:text-white transition">FAQs</a></li>
  </ul>
</div>

/* Social */

<div>

  <h4 className="text-lg font-semibold mb-3">Connect With Us</h4>
  <div className="flex space-x-4 text-lg">
    <a href="#" className="hover:text-blue-400 transition"><FaFacebookF /></a>
    <a href="#" className="hover:text-blue-400 transition"><FaTwitter /></a>
    <a href="#" className="hover:text-blue-400 transition"><FaLinkedinIn /></a>
  </div>
</div>
</div>

<div className="border-t border-slate-700 mt-10 pt-4 text-center text-gray-400 text-xs px-6">
  © {new Date().getFullYear()} StudentConnect. All rights reserved.
</div>
</footer>
);

}

export default Footer;
```

---

```
import { useState, useEffect } from "react";
import { Link, useLocation } from "react-router-dom";
import { HiMenuAlt3, HiX } from "react-icons/hi";
import { motion, AnimatePresence } from "framer-motion";

function Header() {
  const [menuOpen, setMenuOpen] = useState(false);
  const [scrolled, setScrolled] = useState(false);
  const location = useLocation();

  const toggleMenu = () => setMenuOpen(!menuOpen);

  useEffect(() => {
    const handleScroll = () => setScrolled(window.scrollY > 50);
    window.addEventListener("scroll", handleScroll);
    return () => window.removeEventListener("scroll", handleScroll);
  }, []);

  // Close mobile menu on route change
  useEffect(() => {
    setMenuOpen(false);
  }, [location.pathname]);

  return (
    <header
      className={`fixed top-0 left-0 w-full z-50 transition-colors duration-300 ${scrolled ? "bg-white text-gray-900 shadow-md" : "bg-transparent text-white"}`}
    >
  
```

```
<div className="max-w-7xl mx-auto px-6 py-4 flex justify-between items-center">

 {/* Logo */}

<Link
  to="/"
  className={`text-xl font-extrabold tracking-tight ${

    scrolled ? "text-gray-900" : "text-white"

  }`}

>
  Student<span className="text-blue-600">Connect</span>
</Link>

 {/* Desktop Navigation */}

<nav className="hidden md:flex items-center gap-6 text-sm font-medium">

[

  { name: "Home", to: "/" },

  { name: "Login", to: "/login" },

  { name: "Register", to: "/register" },

  { name: "Pricing", to: "/pricing" },

].map((link) => (

  <Link
    key={link.name}
    to={link.to}
    className={`relative group transition-colors ${

      scrolled

      ? "text-gray-700 hover:text-blue-600"

      : "text-white hover:text-blue-300"

    }`}

>
  {link.name}

```

```

    <span className="absolute left-0 -bottom-1 w-0 h-[2px] bg-blue-600 transition-all group-hover:w-full"></span>
    </Link>
  )})
  <Link to="/register">
    <button className="ml-4 bg-blue-600 text-white px-5 py-2 rounded-md font-semibold hover:bg-blue-700 transition">
      Get Started
    </button>
  </Link>
</nav>

/* Mobile Menu Toggle */

<div className="md:hidden">
  <button
    onClick={toggleMenu}
    className={`${text-2xl focus:outline-none ${
      scrolled ? "text-gray-900" : "text-white"
    }}`}>
    {menuOpen ? <HiX /> : <HiMenuAlt3 />}
  </button>
</div>
</div>

/* Mobile Menu */

<AnimatePresence>
  {menuOpen && (
    <motion.div
      initial={{ height: 0, opacity: 0 }}
      animate={{ height: "auto", opacity: 1 }}
      exit={{ height: 0, opacity: 0 }}>

```

```
transition={{ duration: 0.3 }}

className={`md:hidden border-t ${

scrolled ? "bg-white border-gray-200" : "bg-gray-900 border-gray-700"

}`}

>

<div className="px-6 py-4 flex flex-col space-y-4 text-sm font-medium">

{[

{ name: "Home", to: "/" },

{ name: "Login", to: "/login" },

{ name: "Register", to: "/register" },

].map((link) => (

<Link

key={link.name}

to={link.to}

onClick={toggleMenu}

className={`${

scrolled

? "text-gray-800 hover:text-blue-600"

: "text-white hover:text-blue-300"

}`}

>

{link.name}

</Link>

))}

<Link to="/register" onClick={toggleMenu}>

<button

className={`w-full px-5 py-2 rounded-md font-semibold transition ${

scrolled

? "bg-blue-600 text-white hover:bg-blue-700"

: "bg-blue-500 text-white hover:bg-blue-600"

}`}


```

```
>
    Get Started
  </button>
</Link>
</div>
</motion.div>
)}
</AnimatePresence>
</header>
);
}

export default Header;
```

---

```
// src/components/InternshipRecommendations.jsx
import { Briefcase } from "lucide-react";

const internships = [
  {
    company: "TechNova",
    role: "Frontend Intern",
    type: "Remote",
    skills: ["React", "Tailwind", "Figma"],
  },
  {
    company: "AIWorks",
    role: "ML Intern",
    type: "On-site",
    skills: ["Python", "TensorFlow", "Data Analysis"],
  },
];
```

```
{  
  company: "CodeCraft",  
  role: "Backend Intern",  
  type: "Remote",  
  skills: ["Node.js", "MongoDB", "REST API"],  
},  
];  
  
function InternshipRecommendations() {  
  return (  
    <div className="bg-white shadow-md rounded-xl p-6">  
      <div className="flex items-center gap-2 mb-4">  
        <Briefcase className="text-blue-600" size={20} />  
        <h2 className="text-lg font-semibold text-gray-800">Recommended Internships</h2>  
      </div>  
  
      <div className="space-y-4">  
        {internships.map((job, idx) => (  
          <div  
            key={idx}  
            className="border border-gray-200 rounded-md p-4 hover:shadow-md transition-all"  
          >  
            <div className="flex items-center justify-between">  
              <div>  
                <h3 className="font-semibold text-blue-700 text-sm">{job.role}</h3>  
                <p className="text-sm text-gray-600">{job.company} • {job.type}</p>  
              </div>  
              <button className="text-sm bg-blue-600 text-white px-3 py-1 rounded hover:bg-blue-700">  
                Apply  
              </button>  
            </div>  
          </div>  
        ))}  
      </div>  
    </div>  
  );  
}
```

```

<div className="mt-2 flex flex-wrap gap-2">
  {job.skills.map((skill, i) => (
    <span
      key={i}
      className="text-xs bg-gray-100 text-gray-700 px-2 py-1 rounded-full"
    >
      {skill}
    </span>
  )));
  </div>
</div>
))}

</div>
</div>
);

}

export default InternshipRecommendations;

```

```

// src/components/NotificationPanel.jsx

import { BellRing } from "lucide-react";

const notifications = [
  {
    type: "badge",
    message: "You earned the 'AI Learner' badge!",
    time: "2 mins ago",
    read: false,
  },
  {
    type: "job",
  }
];

```

```
    message: "New internship match: Frontend at CodeCraft.",  
    time: "1 hour ago",  
    read: true,  
,  
{  
  type: "event",  
  message: "Reminder: Webinar on UI/UX tomorrow at 3PM.",  
  time: "5 hours ago",  
  read: false,  
,  
{  
  type: "university",  
  message: "Your mentor endorsed your React skill.",  
  time: "Yesterday",  
  read: true,  
,  
];
```

```
function NotificationPanel() {  
  return (  
    <div className="bg-white shadow-md rounded-xl p-6">  
      <div className="flex items-center gap-2 mb-4">  
        <BellRing className="text-blue-600" size={20} />  
        <h2 className="text-lg font-semibold text-gray-800">Notifications</h2>  
      </div>  
  
      <ul className="space-y-3 max-h-60 overflow-y-auto pr-2">  
        {notifications.map((note, idx) => (  
          <li  
            key={idx}
```

```
    className={`border rounded-md px-4 py-2 text-sm transition shadow-sm hover:shadow-md cursor-pointer ${{
      note.read ? "bg-gray-50 text-gray-600" : "bg-blue-50 text-blue-700 border-blue-200"
    }}}
  >
  <p className="font-medium">{note.message}</p>
  <span className="text-xs text-gray-400">{note.time}</span>
</li>
))}

</ul>
</div>
);

}

export default NotificationPanel;
```

---

```
// components/ProtectedRoute.jsx

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

export default function ProtectedRoute({ children }) {
  const token = localStorage.getItem("userToken");
  return token ? children : <Navigate to="/login" replace />;
}
```

---

```
// src/components/RequireRole.jsx

import { useEffect, useState } from "react";
import { Navigate, useLocation } from "react-router-dom";

export default function RequireRole({ role, children }) {
  const [ready, setReady] = useState(false);
  const [ok, setOk] = useState(false);
```

```

const location = useLocation();

useEffect(() => {
  const token = localStorage.getItem("userToken");
  const userRole = localStorage.getItem("userRole");
  setOk(Boolean(token) && userRole === role);
  setReady(true);
}, [role]);

if (!ready) {
  return (
    <div className="min-h-[30vh] grid place-items-center text-gray-500">
      Checking access...
    </div>
  );
}

if (!ok) {
  return <Navigate to="/login" replace state={{ from: location }} />;
}

return children;
}

```

```

import { Link, useLocation, useNavigate } from "react-router-dom";
import {
  Briefcase,
  BookOpen,
  BadgeCheck,
  LayoutDashboard,
  LogOut,
}

```

```
Bell,
UserCircle,
SquarePen,
ChartColumn,
Video,
Sparkles,
PanelLeftClose,
PanelLeftOpen,
} from "lucide-react";
import { useEffect, useMemo, useState } from "react";

const navItems = [
  { label: "Dashboard", icon: LayoutDashboard, path: "/student/dashboard" },
  { label: "Internships", icon: Briefcase, path: "/student/internships" },
  { label: "Career Roadmap", icon: BookOpen, path: "/student/roadmap" },
  { label: "Badges", icon: BadgeCheck, path: "/student/badges" },
  { label: "TestSkill", icon: SquarePen, path: "/student/testskill" },
  { label: "Progress", icon: ChartColumn, path: "/student/progress" },
  { label: "Job Board", icon: ChartColumn, path: "/student/jobs" },
  { label: "Webinars", icon: Video, path: "/student/webinars" },
  { label: "Application Status", icon: Video, path: "/student/status" },
  { label: "Hackathon", icon: Sparkles, path: "/student/hackathons" },
];
}

function Sidebar({ isOpen, setIsOpen, isMobile }) {
  const location = useLocation();
  const navigate = useNavigate();
  const [studentName, setStudentName] = useState("Student");

  // NEW: collapsed state persisted
  const [collapsed, setCollapsed] = useState(() => {
```

```
try {
    return localStorage.getItem("sidebar.collapsed") === "1";
} catch {
    return false;
}
});

// Apply CSS var so main can offset with ml-[var(--sidebar-width)]
useEffect(() => {
    const width = collapsed ? "80px" : "270px";
    document.documentElement.style.setProperty("--sidebar-width", width);
    try {
        localStorage.setItem("sidebar.collapsed", collapsed ? "1" : "0");
    } catch {}
}, [collapsed]);

useEffect(() => {
    const stored = localStorage.getItem("studentName");
    if (stored) setStudentName(stored);
}, []);

const handleLogout = () => {
    localStorage.removeItem("userToken");
    localStorage.removeItem("userRole");
    navigate("/login");
};

const isActive = (p) => location.pathname === p;

const Brand = useMemo(
() => (
```

```
<div className="flex items-center gap-3">
  <div className="relative">
    <div className="h-9 w-9 rounded-xl bg-gradient-to-br from-blue-500 to-indigo-600" />
    <div className="absolute inset-0 rounded-xl ring-2 ring-white/50" />
  </div>
  {!collapsed && (
    <div className="leading-tight">
      <div className="font-semibold text-gray-900">StudentConnect</div>
      <div className="text-xs text-gray-500">Student portal</div>
    </div>
  )}
</div>
),
[collapsed]
);

return (
<aside
  className={[
    "fixed md:sticky top-0 left-0 h-full md:h-screen bg-white/80 backdrop-blur border-r z-40",
    "transform transition-transform duration-300 flex flex-col justify-between",
    isOpen || !isMobile ? "translate-x-0" : "-translate-x-full",
    ].join(" ")}

  style={{
    maxHeight: "100vh",
    width: collapsed && !isMobile ? "80px" : "270px",
  }}
>
<div className="flex flex-col h-full">
  {/* Top: Brand + Collapse/Close */}
  <div className="px-4 py-3 border-b bg-white/60 backdrop-blur-md flex items-center justify-between">
```

```
{Brand}
```

```
<div className="flex items-center gap-2">  
  {!isMobile && (  
    <button  
      onClick={() => setCollapsed((c) => !c)}  
      className="h-8 w-8 grid place-items-center rounded-lg hover:bg-gray-100 text-gray-700"  
      title={collapsed ? "Expand sidebar" : "Collapse sidebar"}  
    >  
    {collapsed ? <PanelLeftOpen size={18} /> : <PanelLeftClose size={18} />}  
  </button>  
  )}  
  {isMobile && (  
    <button  
      onClick={() => setIsOpen(false)}  
      className="h-8 w-8 grid place-items-center rounded-lg hover:bg-gray-100 text-gray-600"  
      aria-label="Close sidebar"  
    >  
    X  
  </button>  
  )}  
</div>  
</div>  
  
/* Profile snippet (hidden when collapsed) */  
;!collapsed && (  
  <div className="px-5 py-3 border-b">  
    <div className="flex items-center gap-3">  
      <div className="h-9 w-9 rounded-full bg-blue-100 text-blue-700 grid place-items-center">  
        <UserCircle size={18} />  
      </div>
```

```

<div className="min-w-0">
  <div className="text-sm font-medium text-gray-900 truncate">{studentName}</div>
  <div className="text-xs text-gray-500 truncate">Student</div>
</div>

  <button className="ml-auto relative text-gray-600 hover:text-blue-600"
    title="Notifications">
    <Bell size={18} />
    <span className="absolute -top-1 -right-1 h-2 w-2 rounded-full bg-red-500" />
  </button>
</div>
</div>
)}

/* Nav Links */
<div className="flex-1 overflow-y-auto">
  <nav className="p-3 space-y-1">
    {navItems.map((item) => {
      const Icon = item.icon;
      const active = isActive(item.path);
      return (
        <Link
          to={item.path}
          key={item.path}
          onClick={() => isMobile && setIsOpen(false)}
          className={[
            "group flex items-center gap-3 px-3 py-2 rounded-xl text-sm transition",
            active ? "bg-blue-600 text-white shadow-sm" : "text-gray-700 hover:bg-gray-100",
            collapsed ? "justify-center" : "",
          ].join(" ")}

          title={collapsed ? item.label : undefined}
        >

```

```
<span
  className={[
    "h-8 w-8 grid place-items-center rounded-lg border flex-shrink-0",
    active
      ? "border-white/20 bg-white/10 text-white"
      : "border-gray-200 bg-white text-gray-700 group-hover:border-gray-300",
  ].join(" ")}

>

<Icon size={16} />
</span>

{!collapsed && <span className="truncate">{item.label}</span>}

{active && !collapsed && <span className="ml-auto h-5 w-1 rounded-full bg-white/70"
/>}

</Link>

);

}}}

</nav>

</div>

/* Footer actions */

<div className="px-4 py-3 border-t bg-white/60 backdrop-blur-md">

<button
  onClick={handleLogout}
  className={[
    "w-full inline-flex items-center justify-center gap-2 text-sm font-medium",
    "text-red-600 hover:text-red-700 border border-red-200 hover:border-red-300 rounded-xl
py-2",
    collapsed ? "px-0" : "",
  ].join(" ")}

  title={collapsed ? "Logout" : undefined}

>

<LogOut size={16} />
```

```
        {!collapsed && "Logout"}  
      </button>  
    </div>  
  </div>  
  </aside>  
);  
}  
  
export default Sidebar;
```

---

```
// src/components/SkillsProgressChart.jsx  
import { useEffect, useState } from "react";  
import API from "../api/axios";  
import { Bar } from "react-chartjs-2";  
import {  
  Chart as ChartJS,  
  BarElement,  
  CategoryScale,  
  LinearScale,  
  Tooltip,  
  Legend,  
} from "chart.js";  
import { Gauge } from "lucide-react";  
  
ChartJS.register(BarElement, CategoryScale, LinearScale, Tooltip, Legend);  
  
const skillOptions = {  
  responsive: true,  
  plugins: {  
    legend: { display: false },  
    tooltip: {
```

```
callbacks: {
  label: (context) => `${context.parsed.y}%`,
},
},
},
scales: {
y: {
beginAtZero: true,
max: 100,
ticks: {
stepSize: 20,
},
},
},
},
};

function SkillsProgressChart() {
const [chartData, setChartData] = useState(null);

useEffect(() => {
  const fetchSkillData = async () => {
    try {
      const res = await API.get("/student/skill-progress");
      const data = res.data;

      const labels = data.map((d) => d.skill);
      const averages = data.map((d) =>
        Math.round(
          (d.history.reduce((acc, h) => acc + h.score, 0) / d.history.length / d.history[0].total) * 100
        )
      );
    }
  };
});
```

```
        setChartData({
          labels,
          datasets: [
            {
              label: "Proficiency (%)",
              data: averages,
              backgroundColor: "#3b82f6",
              borderRadius: 6,
            },
          ],
        });
      } catch (err) {
        console.error("Failed to load skill chart data", err);
      }
    });

    fetchSkillData();
  }, []);
}

if (!chartData) return null;

return (
  <div className="bg-white shadow-md rounded-md p-6">
    <div className="flex items-center gap-2 mb-4">
      <Gauge className="text-blue-600" size={20} />
      <h2 className="text-lg font-semibold text-gray-800">My Skills Progress</h2>
    </div>
    <Bar data={chartData} options={skillOptions} height={200} />
  </div>
);
```

```
}
```

```
export default SkillsProgressChart;
```

---

```
// src/components/Topbar.jsx

import { Bell, UserCircle, Menu } from "lucide-react";
import { Link } from "react-router-dom";
import { useEffect, useState } from "react";

function Topbar({ toggleSidebar, isMobile }) {
  const [studentName, setStudentName] = useState("Student");

  useEffect(() => {
    const stored = localStorage.getItem("studentName");
    if (stored) setStudentName(stored);
  }, []);

  return (
    <header className="bg-white shadow px-4 sm:px-6 py-4 flex items-center justify-between border-b relative z-30">
      {/* Hamburger for mobile */}
      {isMobile && (
        <button onClick={toggleSidebar} className="text-gray-700 p-2 rounded-md bg-white shadow-md">
          <Menu size={22} />
        </button>
      )}
    
```

{/\* Centered title on mobile \*/}

```
<h1
  className={`text-lg font-semibold text-gray-800 ${

  isMobile ? "absolute left-1/2 -translate-x-1/2" : ""`}}
```

```

    `}`}
  >
  Student Dashboard
</h1>

/* Notification and Profile (Desktop Only) */

!isMobile && (
  <div className="flex items-center gap-4">
    <button className="relative text-gray-600 hover:text-blue-600">
      <Bell size={20} />
      <span className="absolute top-0 right-0 w-2 h-2 bg-red-500 rounded-full animate-ping" />
      <span className="absolute top-0 right-0 w-2 h-2 bg-red-500 rounded-full" />
    </button>

    <Link to="/student/profile" className="flex items-center gap-2 hover:underline">
      <UserCircle size={28} className="text-gray-500" />
      <span className="text-sm font-medium text-gray-800 hidden sm:block">{studentName}</span>
    </Link>
  </div>
)});

</header>
);

}

export default Topbar;

```

---

```

// src/components/UniversityBadges.jsx

import { BadgeCheck } from "lucide-react";

const badges = [

```

```
{ title: "React Pro", status: "verified", color: "green" },
{ title: "AI Learner", status: "verified", color: "yellow" },
{ title: "Data Wrangler", status: "pending", color: "gray" },
{ title: "UI Designer", status: "verified", color: "purple" },
{ title: "Backend Champ", status: "pending", color: "gray" },
];

function UniversityBadges() {
  return (
    <div className="bg-white shadow-md rounded-xl p-6">
      <div className="flex items-center gap-2 mb-4">
        <BadgeCheck className="text-blue-600" size={20} />
        <h2 className="text-lg font-semibold text-gray-800">University-Verified Badges</h2>
      </div>

      <div className="flex gap-3 overflow-x-auto pb-2">
        {badges.map((badge, idx) => (
          <div
            key={idx}
            className={`${`min-w-max px-4 py-2 rounded-full text-xs font-semibold text-${badge.color}-700`}
            bg-${badge.color}-100 capitalize shadow-sm`}
          >
            {badge.title} {badge.status === "verified" ? "✓" : "⌚ "}
          </div>
        )));
      </div>
    </div>
  );
}

export default UniversityBadges;
```

```
// src/components/WebinarsSchedule.jsx

import { CalendarDays } from "lucide-react";

const events = [
  {
    date: "Apr 15",
    title: "UI/UX Design Webinar",
    time: "3:00 PM - 4:00 PM",
    speaker: "Jane Doe (UX Lead, CreativX)"
  },
  {
    date: "Apr 22",
    title: "Full Stack Career Talk",
    time: "11:00 AM - 12:30 PM",
    speaker: "Rahul Mehta (CTO, Stackify)"
  },
  {
    date: "May 02",
    title: "AI in Real World",
    time: "5:00 PM - 6:00 PM",
    speaker: "Dr. Kavita Nair (AI Researcher)"
  }
];

function WebinarsSchedule() {
  return (
    <div className="bg-white shadow-md rounded-xl p-6">
      <div className="flex items-center gap-2 mb-4">
        <CalendarDays className="text-blue-600" size={20} />
        <h2 className="text-lg font-semibold text-gray-800">Upcoming Webinars & Events</h2>
      </div>
    </div>
  );
}
```

```

<ul className="space-y-4">
  {events.map((event, idx) => (
    <li
      key={idx}
      className="border-l-4 border-blue-500 pl-4 py-2 bg-blue-50/30 rounded-md hover:bg-blue-100/50"
    >
      <p className="text-sm text-gray-700 font-medium">{event.date}</p>
      <h3 className="text-md text-blue-700 font-semibold">{event.title}</h3>
      <p className="text-sm text-gray-600">{event.time} • {event.speaker}</p>
    </li>
  )));
</ul>
</div>
);

}

export default WebinarsSchedule;

```

```

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

/**
 * Manages a single local MediaStream (mic/cam) and optional screen share.
 * No signaling here — just local preview and control wiring.
 */
export default function useLocalMedia() {
  const videoRef = useRef(null);
  const [stream, setStream] = useState(null);
  const [muted, setMuted] = useState(true);
  const [cameraOff, setCameraOff] = useState(true);

```

```
const [presenting, setPresenting] = useState(false);
const screenTrackRef = useRef(null);

// init audio+video on first use (lazy)
const ensureStream = useCallback(async () => {
  if (stream) return stream;

  const s = await navigator.mediaDevices.getUserMedia({ audio: true, video: true });

  // start muted/camera off by default (like Meet waiting state)
  s.getAudioTracks().forEach(t => (t.enabled = false));
  s.getVideoTracks().forEach(t => (t.enabled = false));

  setStream(s);

  return s;
}, [stream]);

// attach to <video>
useEffect(() => {
  if (videoRef.current) videoRef.current.srcObject = stream || null;
}, [stream]);

const toggleMute = useCallback(async () => {
  const s = await ensureStream();

  const enabled = s.getAudioTracks().some(t => t.enabled);
  s.getAudioTracks().forEach(t => (t.enabled = !enabled));
  setMuted(enabled);
}, [ensureStream]);

const toggleCamera = useCallback(async () => {
  const s = await ensureStream();

  const enabled = s.getVideoTracks().some(t => t.enabled);
  s.getVideoTracks().forEach(t => (t.enabled = !enabled));
  setCameraOff(enabled);
}
```

```
    }, [ensureStream]);
```

  

```
const togglePresent = useCallback(async () => {
  if (!presenting) {
    try {
      const ds = await navigator.mediaDevices.getDisplayMedia({ video: true, audio: false });
      const track = ds.getVideoTracks()[0];
      screenTrackRef.current = track;
      setPresenting(true);
      track.onended = () => {
        setPresenting(false);
        screenTrackRef.current = null;
      };
      // For local preview: swap into the <video> temporarily
      const s = new MediaStream([track]);
      setStream(s);
    } catch {
      /* user cancelled */
    }
  } else if (screenTrackRef.current) {
    screenTrackRef.current.stop();
    screenTrackRef.current = null;
    setPresenting(false);
    // restore original cam stream but keep previous on/off state
    const s = await ensureStream();
    setStream(s);
  }
}, [ensureStream, presenting]);
```

  

```
const stopAll = useCallback(() => {
  if (screenTrackRef.current) {
```

```

try { screenTrackRef.current.stop(); } catch {}

screenTrackRef.current = null;

}

setPresenting(false);

setMuted(true);

setCameraOff(true);

if (stream) {

  stream.getTracks().forEach(t => t.stop());

  setStream(null);

}

}, [stream]);



return {

videoRef,

muted, cameraOff, presenting,

ensureStream,

toggleMute, toggleCamera, togglePresent, stopAll,

};

}

```

```

// src/hooks/useSmartState.js

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


const read = (key, fallback) => {

try {

  const raw = localStorage.getItem(key);

  if (!raw) return fallback;

  return JSON.parse(raw);

} catch {

  return fallback;

}

```

```
};

export default function useSmartState(key, initial) {
  const init = useRef(true);
  const [val, setVal] = useState(() => read(key, initial));

  useEffect(() => {
    if (init.current) { init.current = false; return; }
    try { localStorage.setItem(key, JSON.stringify(val)); } catch {}
  }, [key, val]);

  return [val, setVal];
}
```

---

```
// src/layouts/StudentLayout.jsx

import Sidebar from "../components/Sidebar";
import Topbar from "../components/Topbar";
import { Outlet } from "react-router-dom";
import { useState, useEffect } from "react";

function StudentLayout() {
  const [sidebarOpen, setSidebarOpen] = useState(false);
  const [isMobile, setIsMobile] = useState(window.innerWidth < 768);

  useEffect(() => {
    const handleResize = () => setIsMobile(window.innerWidth < 768);
    window.addEventListener("resize", handleResize);
    return () => window.removeEventListener("resize", handleResize);
  }, []);

  return (

```

```
<div className="flex min-h-screen bg-gray-50">
  <Sidebar isOpen={sidebarOpen} setIsOpen={setSidebarOpen} isMobile={isMobile} />
  <div className="flex-1 flex flex-col">
    <Topbar toggleSidebar={() => setSidebarOpen(true)} isMobile={isMobile} />
    <main className="p-4 sm:p-6 md:p-10">
      <Outlet />
    </main>
  </div>
</div>
);

}

export default StudentLayout;
```

---

```
// src/pages/Admin.jsx

import { useEffect, useMemo, useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import API from "../../api/axios";

import SkillManager from "../admin/SkillManager";
import SetManager from "../admin/SetManager";
import QuestionManager from "../admin/QuestionManager";
import SkillTestResults from "../admin/SkillTestResults";

import {
  LayoutDashboard, Briefcase, ClipboardList, Video, Trophy, Handshake,
  Loader2, Pencil, Trash2, RefreshCcw, Sparkles, BarChart3, Upload
} from "lucide-react";

const apiOrigin =
  (API?.defaults?.baseUrl && new URL(API.defaults.baseUrl).origin) ||
```

```
window.location.origin;

const toAbsolute = (p) =>
!p ? null : p.startsWith("http") ? p : `${apiOrigin}${p.startsWith("/") ? p : `/ ${p}`}`;

/* ----- schedule mini control (copied + simplified from company) ----- */
function ScheduleButton({ application, onUpdated }) {
  const [open, setOpen] = useState(false);
  const [busy, setBusy] = useState(false);
  const iv = application?.interview || {};
  const [form, setForm] = useState({
    startsAt: iv?.startsAt ? new Date(iv.startsAt).toISOString().slice(0, 16) : "",
    durationMins: iv?.durationMins || 45,
    stage: iv?.stage || "screening",
  });
}

const schedule = async () => {
  try {
    setBusy(true);
    if (!form.startsAt) return alert("Pick date & time");
    const body = {
      startsAt: form.startsAt,
      durationMins: Number(form.durationMins || 45),
      stage: form.stage || "screening",
    };
    if (iv?.roomId) {
      await API.put(`/company/applications/${application._id}/reschedule-interview`, body);
    } else {
      await API.post(`/company/applications/${application._id}/schedule-interview`, body);
    }
    setOpen(false);
  } catch (err) {
    console.error(err);
  }
}
```

```
        onUpdated?.();

    } catch (e) {
        alert(e?.response?.data?.message || "Scheduling failed");
    } finally {
        setBusy(false);
    }
};

const cancel = async () => {
    if (!iv?.roomId) return;
    if (!window.confirm("Cancel this interview?")) return;
    try {
        setBusy(true);
        await API.delete(`/company/applications/${application._id}/cancel-interview`);
        setOpen(false);
        onUpdated?.();
    } catch (e) {
        alert(e?.response?.data?.message || "Cancel failed");
    } finally {
        setBusy(false);
    }
};

return (
    <>
    <button onClick={() => setOpen(true)} className="text-indigo-600 underline">
        {iv?.roomId ? "Reschedule" : "Schedule"}
    </button>

    {open && (
        <div className="fixed inset-0 z-50 bg-black/40 grid place-items-center p-4">

```

```
<div className="bg-white w-full max-w-md rounded-2xl p-5">

  <div className="flex items-center justify-between mb-3">
    <div className="font-semibold">
      {iv?.roomId ? "Reschedule Interview" : "Schedule Interview"}
    </div>

    <button onClick={() => setOpen(false)} className="text-gray-500">
      ×
    </button>
  </div>

  <div className="space-y-3">
    <div>

      <label className="text-xs text-gray-500">Stage</label>
      <select
        value={form.stage}
        onChange={(e) => setForm((f) => ({ ...f, stage: e.target.value }))}
        className="w-full border rounded-lg px-3 py-2"
      >
        <option value="screening">Screening</option>
        <option value="technical">Technical</option>
        <option value="behavioral">Behavioral</option>
        <option value="panel">Panel</option>
        <option value="final">Final</option>
      </select>
    </div>

    <div>
      <label className="text-xs text-gray-500">Start time</label>
      <input
        type="datetime-local"
        value={form.startsAt}
      </input>
    </div>
  </div>
</div>
```

```
onChange={(e) => setForm((f) => ({ ...f, startsAt: e.target.value }))}

className="w-full border rounded-lg px-3 py-2"

/>>

<div className="text-[11px] text-gray-500 mt-1">

  Uses your local timezone: {Intl.DateTimeFormat().resolvedOptions().timeZone}

</div>

</div>

<div>

  <label className="text-xs text-gray-500">Duration (mins)</label>

  <input

    type="number"

    min={15}

    step={5}

    value={form.durationMins}

    onChange={(e) =>

      setForm((f) => ({ ...f, durationMins: e.target.value }))

    }

    className="w-full border rounded-lg px-3 py-2"

  />

</div>

<div className="flex items-center justify-between pt-2">

  {iv?.roomId ? (

    <button

      onClick={cancel}

      disabled={busy}

      className="px-3 py-2 rounded-lg border text-red-600"

    >

      Cancel Interview

    </button>

  ) : null}

</div>
```

```
) : (  
    <span />  
)}  
  
<div className="flex gap-2">  
    <button onClick={() => setOpen(false)} className="px-3 py-2 rounded-lg border">  
        Close  
    </button>  
    <button  
        onClick={schedule}  
        disabled={busy}  
        className="px-3 py-2 rounded-lg bg-blue-600 text-white"  
    >  
        {busy ? "Saving..." : iv?.roomId ? "Save" : "Schedule"}  
    </button>  
    </div>  
</div>  
  
{!!iv?.roomId && (  
    <div className="border rounded-lg p-3 bg-gray-50 text-xs">  
        <div className="font-medium mb-1">Join Links</div>  
        <div className="flex flex-col gap-1">  
            <Link  
                to={`/company/webinar/${iv.roomId}`}  
                className="text-indigo-700 underline"  
            >  
                Company (Host)  
            </Link>  
            <Link  
                to={`/student/webinar/${iv.roomId}`}  
                className="text-blue-700 underline"  
            >
```

```

        >
        Student (Viewer)
      </Link>
    </div>
  </div>
}

</div>
</div>
</div>
</div>
);

}

/* ----- Admin Section Components ----- */
/** A. Analytics (cards from existing data) */
function AdminAnalytics() {
  const [loading, setLoading] = useState(true);
  const [jobs, setJobs] = useState([]);
  const [apps, setApps] = useState([]);
  const [webinars, setWebinars] = useState([]);
  const [hackathons, setHackathons] = useState([]);
  const [partnerships, setPartnerships] = useState([]);
  const [results, setResults] = useState([]);

  const [errMsg, setErrMsg] = useState("");

  const load = async () => {
    setLoading(true);
    setErrMsg("");
    try {

```

```

const [j, a, w, h, p, r] = await Promise.all([
  API.get("/company/jobs"),
  API.get("/company/applications"),
  API.get("/company/webinars"),
  API.get("/company/hackathons"),
  API.get("/company/partnerships"),
  API.get("/admin/skill-test-results"),
]);

setJobs(j.data?.items || j.data || []);
setApps(a.data?.items || a.data || []);
setWebinars(w.data?.items || w.data || []);
setHackathons(h.data?.items || h.data || []);
setPartnerships(p.data?.items || p.data || []);
setResults(r.data || []);

} catch (e) {
  setErrMsg(e?.response?.data?.message || e.message || "Failed to load analytics");
} finally {
  setLoading(false);
}
};

useEffect(() => {
  load();
}, []);
}

const cards = useMemo(
() => [
  { label: "Open Jobs", value: jobs.filter((j) => j.status === "open").length, icon: <Briefcase size={18}>/> },
  { label: "Applications", value: apps.length, icon: <ClipboardList size={18}>/> },
]

```

```

        label: "Upcoming Webinars",
        value: webinars.filter((w) => new Date(w.startsAt) > new Date()).length,
        icon: <Video size={18} />,
    },
    { label: "Live Hackathons", value: hackathons.filter((h) => h.status === "live").length, icon: <Trophy size={18} /> },
    { label: "Partnerships", value: partnerships.length, icon: <Handshake size={18} /> },
    { label: "Skill Tests Recorded", value: results.length, icon: <BarChart3 size={18} /> },
],
[jobs, apps, webinars, hackathons, partnerships, results]
);

return (
<div className="space-y-6">
<div className="flex items-center justify-between">
<h2 className="text-2xl font-bold text-slate-800">Admin Analytics</h2>
<button
    onClick={load}
    className="px-3 py-2 rounded-lg border text-sm flex items-center gap-2"
>
    <RefreshCcw size={16} /> Refresh
</button>
{errMsg && <div className="text-sm text-red-600">{errMsg}</div>}
</div>

{!loading ? (
<div className="grid sm:grid-cols-2 lg:grid-cols-3 gap-4">
{[...Array(6)].map((_, i) => (
<div key={i} className="p-5 border rounded-2xl bg-white/70">
<div className="h-4 w-24 bg-slate-200 rounded mb-3" />
<div className="h-8 w-16 bg-slate-200 rounded" />

```

```

        </div>
    )})
</div>
) : (
<div className="grid sm:grid-cols-2 lg:grid-cols-3 gap-4">
{cards.map((c) => (
<div key={c.label} className="p-5 border rounded-2xl bg-white/80 shadow-sm">
<div className="flex items-center justify-between">
<div>
<p className="text-xs text-slate-500">{c.label}</p>
<p className="text-2xl font-semibold mt-1">{c.value}</p>
</div>
<div className="h-10 w-10 rounded-xl grid place-items-center bg-indigo-50 text-indigo-700 border border-indigo-200">
{c.icon}
</div>
</div>
</div>
))})
</div>
)}
</div>
);
}

/** B. Jobs Manager (uses existing /company/jobs endpoints) */
function JobsManager() {
const [skills, setSkills] = useState([]);
const [jobs, setJobs] = useState([]);
const [loading, setLoading] = useState(true);
const [editing, setEditing] = useState(null);

```

```

const load = async () => {
    setLoading(true);
    try {
        const [s, j] = await Promise.all([API.get("/admin/skills"), API.get("/company/jobs")]);
        setSkills(s.data || []);
        setJobs(j.data?.items || j.data || []);
    } catch (e) {
        console.error("JobsManager load failed:", e?.response?.data || e.message);
    } finally {
        setLoading(false);
    }
};

useEffect(() => { load(); }, []);

const toggle = async (id) => {
    try {
        await API.patch(`/company/jobs/${id}/toggle`);
        await load();
    } catch (e) {
        alert(e?.response?.data?.message || "Toggle failed");
    }
};

const del = async (id) => {
    if (!window.confirm("Delete this job posting?")) return;
    try { await API.delete(`/company/jobs/${id}`); load(); } catch (e) { alert(e?.response?.data?.message || "Delete failed"); }
};

const saveEdit = async (e) => {

```

```
e.preventDefault();

if (!editing?._id) return;

try {
  const p = {
    title: editing.title,
    type: editing.type,
    location: editing.location,
    description: editing.description,
    minScore: Number(editing.minScore || 0),
    isFeatured: !!editing.isFeatured,
    status: editing.status,
    skills: editing.skills,
  };
  await API.put(`/company/jobs/${editing._id}`, p);
  setEditing(null);
  await load();
} catch (e2) {
  alert(e2?.response?.data?.message || "Update job failed");
}
};

return (
  <div className="space-y-4">
    <div className="flex items-center justify-between">
      <h3 className="text-xl font-semibold">All Jobs</h3>
      <button
        onClick={load}
        className="px-3 py-2 rounded-lg border text-sm flex items-center gap-2"
      >
        <RefreshCcw size={16} /> Refresh
      </button>
    </div>
  </div>
);
```

```

</div>

{loading ? (
  <div className="text-gray-500 flex items-center gap-2">
    <Loader2 className="animate-spin" />
    <span>Loading jobs...</span>
  </div>
) : jobs.length === 0 ? (
  <p className="text-sm text-gray-500">No jobs found.</p>
) : (
  <div className="overflow-x-auto">
    <table className="w-full text-sm overflow-hidden rounded-2xl border border-slate-200">
      <thead className="bg-slate-50/80">
        <tr>
          <th className="p-3 text-left">Title</th>
          <th className="p-3">Type</th>
          <th className="p-3">Status</th>
          <th className="p-3">Skills</th>
          <th className="p-3">MinScore</th>
          <th className="p-3">Actions</th>
        </tr>
      </thead>
      <tbody>
        {jobs.map((j) => (
          <tr key={j._id} className="border-b">
            <td className="p-3">{j.title}</td>
            <td className="p-3 text-center">{j.type}</td>
            <td className="p-3 text-center">{j.status}</td>
            <td className="p-3">
              {((j.skills || []))
                .map((s) => (typeof s === "string" ? s : s.name))}
            </td>
          </tr>
        ))
      </tbody>
    </table>
  </div>
)

```

```
.join(", "))

</td>

<td className="p-3 text-center">j.minScore</td>

<td className="p-3">
  <div className="flex items-center gap-3 justify-center">
    <button
      onClick={() =>
        setEditing({
          _id: j._id,
          title: j.title || "",
          type: j.type || "job",
          location: j.location || "",
          description: j.description || "",
          minScore: j.minScore ?? 0,
          isFeatured: !!j.isFeatured,
          status: j.status || "open",
          skills: (j.skills || []).map((s) =>
            typeof s === "string" ? s : s._id
          ),
        })
      }
    >
      <Pencil size={16} />
      Edit
    </button>
    <button
      onClick={() => toggle(j._id)}
      className="text-indigo-700 hover:underline"
    >
      {j.status === "open" ? "Close" : "Open"}
    </button>
  </div>
</td>
```

```

        </button>

        <button
            onClick={() => del(j._id)}
            className="text-rose-700 hover:underline flex items-center gap-1"
        >
            <Trash2 size={16} />
            Delete
        </button>
    </div>
</td>
</tr>
))}

</tbody>
</table>
</div>
)
}

```

```

/* Edit modal */

{editing && (
    <div className="fixed inset-0 bg-black/40 z-50 flex items-center justify-center p-4">
        <div className="bg-white/90 backdrop-blur w-full max-w-2xl rounded-2xl shadow-xl p-6 border border-slate-200">
            <div className="flex items-center justify-between mb-4">
                <h3 className="text-lg font-semibold">Edit Job</h3>
                <button
                    onClick={() => setEditing(null)}
                    className="text-gray-500 hover:text-gray-700"
                >
                    ×
                </button>
            </div>

```

```
<form onSubmit={saveEdit} className="grid md:grid-cols-2 gap-4">

  <input
    value={editing.title}
    onChange={(e) => setEditing((j) => ({ ...j, title: e.target.value }))}
    required
    placeholder="Title"
    className="border rounded-xl px-3 py-2 bg-white/70"
  />

  <select
    value={editing.type}
    onChange={(e) => setEditing((j) => ({ ...j, type: e.target.value }))}
    className="border rounded-xl px-3 py-2 bg-white/70"
  >
    <option value="job">Job</option>
    <option value="internship">Internship</option>
  </select>

  <input
    value={editing.location}
    onChange={(e) => setEditing((j) => ({ ...j, location: e.target.value }))}
    placeholder="Location (Remote / City)"
    className="border rounded-xl px-3 py-2 bg-white/70"
  />

  <input
    type="number"
    min="0"
    max="10"
    step="0.1"
    value={editing.minScore}
    onChange={(e) => setEditing((j) => ({ ...j, minScore: e.target.value }))}
    placeholder="Min test score (0-10)"
```

```
    className="border rounded-xl px-3 py-2 bg-white/70"
  />
<div className="md:col-span-2">
  <textarea
    value={editing.description}
    onChange={(e) =>
      setEditing((j) => ({ ...j, description: e.target.value }))
    }
    placeholder="Role description"
    className="w-full border rounded-xl px-3 py-2 bg-white/70"
  />
</div>
<div className="md:col-span-2">
  <p className="text-xs text-gray-500 mb-1">Required skills</p>
  <div className="flex flex-wrap gap-3">
    {skills.map((sk) => {
      const checked = editing.skills.includes(sk._id);
      return (
        <label key={sk._id} className="text-sm flex items-center gap-2">
          <input
            type="checkbox"
            checked={checked}
            onChange={() =>
              setEditing((prev) => {
                const exists = prev.skills.includes(sk._id);
                return {
                  ...prev,
                  skills: exists
                    ? prev.skills.filter((x) => x !== sk._id)
                    : [...prev.skills, sk._id],
                };
              });
            }
          />
        </label>
      );
    });
  </div>
</div>
```

```
        })
    }
  />
  {sk.name}
</label>
);
}}}
</div>
</div>
<label className="flex items-center gap-2 text-sm">
<input
  type="checkbox"
  checked={editing.isFeatured}
  onChange={(e) =>
    setEditing((j) => ({ ...j, isFeatured: e.target.checked }))
  }
/>
  Featured listing
</label>
<select
  value={editing.status}
  onChange={(e) => setEditing((j) => ({ ...j, status: e.target.value }))}
  className="border rounded-xl px-3 py-2 bg-white/70"
>
  <option value="open">Open</option>
  <option value="closed">Closed</option>
  <option value="paused">Paused</option>
</select>
<div className="md:col-span-2 flex items-center gap-3 pt-2">
  <button
    type="button"
```

```

        onClick={() => setEditing(null)}
        className="px-4 py-2 rounded-lg border"
      >
  Cancel
</button>
<button type="submit" className="px-4 py-2 rounded-lg bg-blue-600 text-white">
  Save Changes
</button>
</div>
</form>
</div>
</div>
)
};

</div>
}

/** C. Applications / Screening (company table adapted) */
function ApplicationsManager() {
  const [applications, setApplications] = useState([]);
  const [loading, setLoading] = useState(true);

  const load = async () => {
    setLoading(true);
    try {
      const res = await API.get("/company/applications");
      setApplications(res.data?.applications || res.data || []);
    } catch (e) {
      console.error("Applications load failed:", e?.response?.data || e.message);
      setApplications([]);
    } finally {

```

```
    setLoading(false);

}

};

useEffect(() => {
    load();
}, []);

const update = async (id, status) => {
    try {
        await API.patch(`/company/applications/${id}`, { status });
        await load();
    } catch (e) {
        alert(e?.response?.data?.message || "Update failed");
    }
};

return (
    <div className="space-y-4">
        <div className="flex items-center justify-between">
            <h3 className="text-xl font-semibold">Candidate Applications</h3>
            <button
                onClick={load}
                className="px-3 py-2 rounded-lg border text-sm flex items-center gap-2"
            >
                <RefreshCcw size={16} /> Refresh
            </button>
        </div>
    </div>

    {loading ? (
        <div className="text-gray-500 flex items-center gap-2">
            <Loader2 className="animate-spin" />
    
```

```

<span>Loading applications...</span>
</div>

) : applications.length === 0 ? (
  <p className="text-sm text-gray-500">No applications yet.</p>
) : (
  <div className="overflow-x-auto">
    <table className="w-full text-sm overflow-hidden rounded-2xl border border-slate-200">
      <thead className="bg-slate-50/80">
        <tr>
          <th className="p-3 text-left">Candidate</th>
          <th className="p-3">Job</th>
          <th className="p-3">Resume</th>
          <th className="p-3">ResumeScore</th>
          <th className="p-3">TestScore</th>
          <th className="p-3">Fit</th>
          <th className="p-3">Status</th>
          <th className="p-3">Interview</th>
          <th className="p-3">Action</th>
        </tr>
      </thead>
      <tbody>
        {applications.map((a) => {
          const iv = a.interview;
          const windowText = iv?.startsAt
            ? `${new Date(iv.startsAt).toLocaleString()} • ${
                iv.durationMins || 45
              }m ${iv.stage || "screening"}`
            : "—";
          return (
            <tr key={a._id} className="border-b">
              <td className="p-3">

```

```
{a.student?.name}{" "}  
<span className="text-gray-400">{a.student?.email}</span>  
</td>  
<td className="p-3 text-center">{a.job?.title}</td>  
<td className="p-3 text-center">  
{a.cvUrl ? (  
    <a  
        className="text-blue-700 underline"  
        href={toAbsolute(a.cvUrl)}  
        target="_blank"  
        rel="noreferrer"  
        download  
    >  
        Resume  
    </a>  
) : (  
    "_"  
)  
</td>  
<td className="p-3 text-center">  
{a.screening?.resumeScore ??"—"}  
</td>  
<td className="p-3 text-center">  
{a.screening?.testScore != null  
? Math.round(a.screening.testScore * 10) / 10  
:"—"}  
</td>  
<td className="p-3 text-center font-semibold">  
{a.screening?.fitScore ??"—"}  
</td>  
<td className="p-3 text-center">{a.status}</td>
```

```
<td className="p-3 text-center">  
  <div className="text-xs">{windowText}</div>  
  {iv?.roomId && (  
    <div className="flex items-center justify-center gap-3 mt-1">  
      <Link  
        to={`/company/webinar/${iv.roomId}`}  
        className="text-indigo-700 underline text-xs"  
      >  
        Join (Host)  
      </Link>  
      <Link  
        to={`/student/webinar/${iv.roomId}`}  
        className="text-blue-700 underline text-xs"  
      >  
        Join (Viewer)  
      </Link>  
    </div>  
  )}  
</td>  
<td className="p-3">  
  <div className="flex gap-3 justify-center">  
    <button  
      onClick={() => update(a._id, "shortlisted")}  
      className="text-blue-700 hover:underline"  
    >  
      Shortlist  
    </button>  
    <ScheduleButton application={a} onUpdated={load} />  
    <button  
      onClick={() => update(a._id, "offer")}  
      className="text-emerald-700 hover:underline"  
    >
```

```

        >
        Offer
      </button>
      <button
        onClick={() => update(a._id, "rejected")}
        className="text-rose-700 hover:underline"
      >
        Reject
      </button>
    </div>
  </td>
</tr>
);
}

</tbody>
</table>
</div>
)
;
</div>
);

}

/* D. Webinars / Hackathons / Partnerships (reader + manage) */
function SimpleListManager({ title, endpoint, columns, rowRender }) {
  const [rows, setRows] = useState([]);
  const [loading, setLoading] = useState(true);
  const load = async () => {
    setLoading(true);
    try {
      const res = await API.get(endpoint);
      setRows(res.data?.items || res.data || []);
    }
  }
}

```

```
        } catch (e) {
            console.error(`#${title} load failed:`, e?.response?.data || e.message);
            setRows([]);
        } finally {
            setLoading(false);
        }
    };
    useEffect(() => {
        load();
    }, [endpoint]);
}

return (
    <div className="space-y-3">
        <div className="flex items-center justify-between">
            <h3 className="text-xl font-semibold">{title}</h3>
            <button
                onClick={load}
                className="px-3 py-2 rounded-lg border text-sm flex items-center gap-2"
            >
                <RefreshCcw size={16} /> Refresh
            </button>
        </div>
        {loading ? (
            <div className="text-gray-500 flex items-center gap-2">
                <Loader2 className="animate-spin" />
                <span>Loading...</span>
            </div>
        ) : rows.length === 0 ? (
            <p className="text-sm text-gray-500">No data.</p>
        ) : (
            <div className="overflow-x-auto">
```

```

        <table className="w-full text-sm overflow-hidden rounded-2xl border border-slate-200">
          <thead className="bg-slate-50/80">
            <tr>{columns.map((c) => <th key={c} className="p-3 text-left">{c}</th>)}</tr>
          </thead>
          <tbody>{rows.map((r) => rowRender(r, load))}</tbody>
        </table>
      </div>
    )}

</div>

);

}

```

```

function WebinarsManager() {
  return (
    <SimpleListManager
      title="Webinars"
      endpoint="/company/webinars"
      columns={["Title", "When", "Room", "Actions"]}
      rowRender={(w, reload) => (
        <tr key={w._id} className="border-b">
          <td className="p-3">{w.title}</td>
          <td className="p-3">
            {new Date(w.startsAt).toLocaleString()} • {w.durationMins}m
          </td>
          <td className="p-3">{w.roomId || "-"}</td>
          <td className="p-3">
            <div className="flex gap-3">
              <Link
                to={`/company/webinar/${w.roomId}`}
                className="text-indigo-700 underline text-sm"
              >

```

```

    Open Studio

</Link>

<button
  onClick={async () => {
    await API.delete(`/company/webinars/${w._id}`);
    reload();
  }}
  className="text-rose-600 text-sm underline"
>
  Delete
</button>
</div>
</td>
</tr>
)};

/>
);

}

function HackathonsManager() {
  return (
    <SimpleListManager
      title="Hackathons"
      endpoint="/company/hackathons"
      columns={['Title', 'Window', 'Status', 'Actions']}
      rowRender={(h, reload) => (
        <tr key={h._id} className="border-b">
          <td className="p-3">{h.title}</td>
          <td className="p-3">
            {new Date(h.startAt).toLocaleString()} →{" "}
            {new Date(h.endAt).toLocaleString()}
          </td>
        </tr>
      )}
    )
  );
}

```

```
</td>

<td className="p-3">{h.status}</td>

<td className="p-3">
  <div className="flex gap-3">
    <select
      value={h.status || "upcoming"}
      onChange={async (e) => {
        await API.patch('/company/hackathons/${h._id}/status', {
          status: e.target.value,
        });
        reload();
      }}
      className="border rounded px-2 py-1 text-sm"
    >
      <option value="upcoming">Upcoming</option>
      <option value="live">Live</option>
      <option value="ended">Ended</option>
    </select>
    <button
      onClick={async () => {
        await API.delete('/company/hackathons/${h._id}');
        reload();
      }}
      className="text-rose-600 text-sm underline"
    >
      Delete
    </button>
  </div>
</td>
</tr>
)}
```

```

        />
    );
}

function PartnershipsManager() {
    return (
        <SimpleListManager
            title="University Partnerships"
            endpoint="/company/partnerships"
            columns={[ "Title", "University", "Status", "Actions" ] }
            rowRender={({p, reload}) => (
                <tr key={p._id} className="border-b">
                    <td className="p-3">{p.title}</td>
                    <td className="p-3">{p.university}</td>
                    <td className="p-3">{p.status}</td>
                    <td className="p-3">
                        <div className="flex gap-3">
                            <select
                                value={p.status || "proposal"}
                                onChange={(e) => {
                                    await API.put(`/company/partnerships/${p._id}`, {
                                        ...p,
                                        status: e.target.value,
                                    });
                                    reload();
                                }}
                            >
                                <option value="proposal">Proposal</option>
                                <option value="active">Active</option>
                                <option value="completed">Completed</option>
                            </select>
                        </div>
                    </td>
                </tr>
            )}
            className="border rounded px-2 py-1 text-sm"
        >
        <option value="proposal">Proposal</option>
        <option value="active">Active</option>
        <option value="completed">Completed</option>
    </SimpleListManager>
)
}

```

```

        </select>

        <button
            onClick={async () => {
                await API.delete(`/company/partnerships/${p._id}`);
                reload();
            }}
            className="text-rose-600 text-sm underline"
        >
            Delete
        </button>
    </div>
</td>
</tr>
)
);
}
/>
);
}

}

```

```

/** E. Bulk CSV Import/Export (skills, sets, questions, test-results) */

function BulkImportExport() {

    const [type, setType] = useState("skills");
    const [file, setFile] = useState(null);
    const [busy, setBusy] = useState(false);
    const [log, setLog] = useState([]);

    const append = (line) => setLog((l) => [...l, line]);

    const parseCSV = async (f) => {
        const text = await f.text();
        const rows = text
            .split(/\r?\n/)

```

```
.map((r) => r.trim())
.filter(Boolean)
.map((r) => r.split(",").map((c) => c.trim())));
if (rows.length && rows[0].some((c) => /name|title|question|skill/i.test(c))) rows.shift();
return rows;
};

const importNow = async () => {
  if (!file) return alert("Choose a CSV file");
  setBusy(true);
  setLog([]);
  try {
    const rows = await parseCSV(file);
    append(`Parsed ${rows.length} rows`);

    if (type === "skills") {
      for (const [name] of rows) {
        if (!name) continue;
        try {
          await API.post("/admin/skills", { name });
          append(`✓ Skill: ${name}`);
        } catch (e) {
          append(`✗ Skill: ${name} — ${e?.response?.data?.message || e.message}`);
        }
      }
    } else if (type === "sets") {
      for (const [title, skill] of rows) {
        if (!title || !skill) {
          append("✗ Set: missing title or skillId");
          continue;
        }
      }
    }
  }
};
```

```

try {

    await API.post("/admin/question-sets", { title, skill, description: "" });

    append(`✓ Set: ${title}`);

} catch (e) {

    append(`✗ Set: ${title} — ${e?.response?.data?.message || e.message}`);

}

}

} else if (type === "questions") {

for (const [set, question, o1, o2, o3, o4, answer] of rows) {

if (!set || !question || !o1 || !o2 || !o3 || !o4 || !answer) {

    append("✗ Question: missing fields");

    continue;

}

try {

    await API.post("/admin/questions", {

        set,

        question,

        options: [o1, o2, o3, o4],

        answer,

    });

    append(`✓ Q: ${question.slice(0, 50)})`);

} catch (e) {

    append(`✗ Q: ${question.slice(0, 50)} — ${e?.response?.data?.message || e.message}`);

}

}

} else if (type === "results") {

append("✗ Import for results not supported.");

}

append("Done.");

} catch (e) {

```

```
append(`× Import error: ${e.message}`);
} finally {
  setBusy(false);
}

};

const exportCSV = async () => {
  try {
    let rows = [];
    if (type === "skills") {
      const res = await API.get("/admin/skills");
      rows = (res.data || []).map((s) => [s._id, s.name]);
    } else if (type === "sets") {
      const res = await API.get("/admin/question-sets");
      rows = (res.data || []).map((s) => [s._id, s.title, s.skill?._id || s.skill || ""]);
    } else if (type === "questions") {
      alert(
        "Exporting all questions typically needs a dedicated endpoint. Export skills/sets for now."
      );
      return;
    } else if (type === "results") {
      const res = await API.get("/admin/skill-test-results");
      rows = (res.data || []).map((r) => [
        r.user?.name || "",
        r.user?.email || "",
        r.skill?.name || "",
        r.score,
        r.total,
        new Date(r.createdAt).toLocaleString(),
      ]);
    }
  }
}
```

```

const csv = rows.map((r) => r.map((c) => String(c).replace(/,/g, " ")).join(",")).join("\n");

const blob = new Blob([csv], { type: "text/csv;charset=utf-8;" });

const url = URL.createObjectURL(blob);

const a = document.createElement("a");

a.href = url;

a.download = `${type}.csv`;

a.click();

URL.revokeObjectURL(url);

} catch (e) {

alert(e?.response?.data?.message || e.message || "Export failed");

}

};

return (

<div className="space-y-4">

<h3 className="text-xl font-semibold">Bulk Import / Export (CSV)</h3>

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

<select

value={type}

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

className="border rounded-lg px-3 py-2"

>

<option value="skills">Skills</option>

<option value="sets">Question Sets</option>

<option value="questions">Questions</option>

<option value="results">Skill Test Results</option>

</select>

<label className="flex items-center gap-2 text-sm">

<input type="file" accept=".csv" onChange={(e) => setFile(e.target.files?[0] || null)} />

{file?.name && <span className="text-gray-600">{file.name}</span>}

</label>

```

```

<button
    disabled={busy}
    onClick={importNow}
    className="px-3 py-2 rounded-lg bg-blue-600 text-white flex items-center gap-2"
>
    {busy ? <Loader2 size={16} className="animate-spin" /> : <Upload size={16} />}
    Import
</button>

<button onClick={exportCSV} className="px-3 py-2 rounded-lg border flex items-center gap-2">
    <Upload size={16} />
    Export
</button>
</div>

<div className="border rounded-lg p-3 bg-gray-50 text-xs max-h-48 overflow-auto">
    {log.length === 0 ? (
        <div className="text-gray-500">No logs yet.</div>
    ) : (
        log.map((l, i) => <div key={i}>{l}</div>)
    )}
</div>

<div className="text-xs text-gray-500">
    Hints:
    <ul className="list-disc ml-5 mt-1 space-y-1">
        <li>
            <b>Skills:</b> <code>name</code>
        </li>
        <li>
            <b>Sets:</b> <code>title, skillId</code> (get skillId from “Export Skills”)
        </li>
    <ul>

```

```

<li>
  <b>Questions:</b>{" "}
  <code>setId, question, opt1, opt2, opt3, opt4, answer</code> (answer must match one
  option exactly)
</li>
</ul>
</div>
</div>
);

}

/* ----- Main Admin Page ----- */
const menu = [
  { id: "analytics", label: "Analytics", icon: <LayoutDashboard size={16} /> },
  { id: "jobs", label: "Jobs", icon: <Briefcase size={16} /> },
  { id: "applications", label: "Applications", icon: <ClipboardList size={16} /> },
  { id: "webinars", label: "Webinars", icon: <Video size={16} /> },
  { id: "hackathons", label: "Hackathons", icon: <Trophy size={16} /> },
  { id: "partnerships", label: "Partnerships", icon: <Handshake size={16} /> },
  { id: "skills", label: "Skills", icon: <Sparkles size={16} /> },
  { id: "sets", label: "Question Sets", icon: <Sparkles size={16} /> },
  { id: "questions", label: "Questions", icon: <Sparkles size={16} /> },
  { id: "results", label: "Results", icon: <BarChart3 size={16} /> },
  { id: "bulk", label: "Bulk Import/Export", icon: <Upload size={16} /> },
];

export default function Admin() {
  const navigate = useNavigate();
  const [activeTab, setActiveTab] = useState("analytics");

  // soft guard: if no admin token, go to admin login (AdminRoute also guards)
}

```

```
useEffect(() => {
  if (!localStorage.getItem("adminToken")) navigate("/admin/login", { replace: true });
}, [navigate]);

const adminEmail =
  localStorage.getItem("adminEmail") ||
  () => {
  try {
    const u = JSON.parse(localStorage.getItem("adminUser") || "{}");
    return u.email || "";
  } catch { return ""; }
})();

const logout = () => {
  localStorage.removeItem("adminToken");
  localStorage.removeItem("adminUser");
  localStorage.removeItem("adminEmail");
  navigate("/admin/login", { replace: true });
};

return (
  <div className="flex min-h-screen bg-gray-50">
    {/* Sidebar */}
    <aside className="w-64 bg-gray-900 text-white p-6 space-y-6">
      <div className="flex items-center justify-between mb-2">
        <h2 className="text-2xl font-bold">Admin Panel</h2>
        <button onClick={logout} className="text-xs underline opacity-80 hover:opacity-100" title="Logout">
          Logout
        </button>
      </div>
    </aside>
    <div className="flex-grow flex flex-col py-6 px-6 sm:px-8 sm:py-8 sm:space-y-6">
      <div>
        <h3>Dashboard</h3>
        <p>Welcome to the Admin Dashboard!</p>
        <ul>
          <li>Item 1</li>
          <li>Item 2</li>
          <li>Item 3</li>
        </ul>
      </div>
    </div>
  </div>
)
```

```

<div className="text-xs text-slate-300 mb-4 break-all">{adminEmail}</div>
<ul className="space-y-2">
  {menu.map((item) => (
    <li
      key={item.id}
      className={`${`px-4 py-2 rounded cursor-pointer flex items-center gap-2 ${{
        activeTab === item.id ? "bg-blue-600" : "hover:bg-gray-800"
      }}`}
      onClick={() => setActiveTab(item.id)}
    >
      {item.icon} {item.label}
    </li>
  )))
</ul>
</aside>

/* Canvas */

<main className="flex-1 p-6 overflow-y-auto">
  {activeTab === "analytics" && <AdminAnalytics />}
  {activeTab === "jobs" && <JobsManager />}
  {activeTab === "applications" && <ApplicationsManager />}
  {activeTab === "webinars" && <WebinarsManager />}
  {activeTab === "hackathons" && <HackathonsManager />}
  {activeTab === "partnerships" && <PartnershipsManager />}

  {activeTab === "skills" && <SkillManager />}
  {activeTab === "sets" && <SetManager />}
  {activeTab === "questions" && <QuestionManager />}
  {activeTab === "results" && <SkillTestResults />}

  {activeTab === "bulk" && <BulkImportExport />}

```

```
</main>
</div>
);
}



---



```
// src/pages/admin/AdminLogin.jsx
import { useEffect, useState } from "react";
import { useNavigate } from "react-router-dom";
import API from "../../api/axios";

export default function AdminLogin() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [busy, setBusy] = useState(false);
  const [err, setErr] = useState("");
  const [info, setInfo] = useState("");
  const navigate = useNavigate();

  // Always reset admin session on login page to avoid stuck states
  useEffect(() => {
    localStorage.removeItem("adminToken");
    localStorage.removeItem("adminUser");
    localStorage.removeItem("adminEmail");
  }, []);

  const submit = async (e) => {
    e.preventDefault();
    setBusy(true); setErr(""); setInfo("");
    try {
      const res = await API.post("/auth/admin/login", { email, password });
      const token = res?.data?.token;
```


```

```
const user = res?.data?.user || {};
if (!token) throw new Error("Login response did not include a token");

localStorage.setItem("adminToken", token);
localStorage.setItem("adminUser", JSON.stringify(user));
if (user.email) localStorage.setItem("adminEmail", user.email);

// verify before navigation (prevents bounce-back)
await API.get("/auth/admin/me");

setInfo("Login successful. Redirecting...");
navigate("/admin", { replace: true });

} catch (e2) {
  setErr(e2?.response?.data?.message || e2.message || "Login failed");
} finally {
  setBusy(false);
}
};

return (
  <div className="min-h-screen flex items-center justify-center bg-gray-50 px-4">
    <form onSubmit={submit} className="w-full max-w-sm bg-white border rounded-xl p-6 shadow-sm">
      <h1 className="text-2xl font-semibold mb-1">Admin Sign in</h1>
      <p className="text-sm text-gray-500 mb-4">Access the control panel</p>
      {err && <div className="mb-3 text-sm text-red-600">{err}</div>}
      {info && <div className="mb-3 text-sm text-emerald-600">{info}</div>}

      <label className="block text-sm mb-1">Email</label>
      <input
        type="email"

```

```
    className="w-full border rounded-lg px-3 py-2 mb-3"
    placeholder="admin@yourdomain.com"
    value={email}
    onChange={(e)=>setEmail(e.target.value)}
    autoComplete="username"
    required
/>

```

```
<label className="block text-sm mb-1">Password</label>
<input
  type="password"
  className="w-full border rounded-lg px-3 py-2 mb-4"
  placeholder="*****"
  value={password}
  onChange={(e)=>setPassword(e.target.value)}
  autoComplete="current-password"
  required
/>

```

```
<button disabled={busy} className="w-full bg-blue-600 text-white rounded-lg py-2 font-medium">
  {busy ? "Signing in..." : "Sign in"}
</button>
</form>
</div>
);
}
```

---

```
import { useEffect, useState } from "react";
import API from "../../api/axios";
```

```
function AdminResults() {
  const [results, setResults] = useState([]);

  useEffect(() => {
    fetchResults();
  }, []);

  const fetchResults = async () => {
    try {
      const res = await API.get("/admin/results");
      setResults(res.data);
    } catch (err) {
      console.error("Failed to fetch results", err);
    }
  };
}

return (
  <div className="p-6">
    <h2 className="text-2xl font-bold mb-4">📊 Skill Test Results</h2>
    <div className="overflow-x-auto">
      <table className="min-w-full border border-gray-300">
        <thead className="bg-gray-100">
          <tr>
            <th className="px-4 py-2">Student</th>
            <th className="px-4 py-2">Skill</th>
            <th className="px-4 py-2">Score</th>
            <th className="px-4 py-2">Total</th>
            <th className="px-4 py-2">Percentage</th>
            <th className="px-4 py-2">Date</th>
          </tr>
        </thead>
```

```

<tbody>
  {results.map((r) => (
    <tr key={r._id} className="border-t">
      <td className="px-4 py-2">{r.studentId?.name || "N/A"}</td>
      <td className="px-4 py-2">{r.skill?.name}</td>
      <td className="px-4 py-2">{r.score}</td>
      <td className="px-4 py-2">{r.total}</td>
      <td className="px-4 py-2">{r.percentage.toFixed(2)}%</td>
      <td className="px-4 py-2">{new Date(r.createdAt).toLocaleString()}</td>
    </tr>
  )));
</tbody>
</table>
</div>
</div>
);

}

export default AdminResults;

```

---

```

// src/admin/HomePageManager.jsx
import { useState, useEffect } from "react";
import axios from "axios";

export default function HomePageManager() {
  const [hero, setHero] = useState({ heading: "", subheading: "" });
  const [highlights, setHighlights] = useState([]);
  const [loading, setLoading] = useState(true);
  const [message, setMessage] = useState("");

  useEffect(() => {

```

```
axios.get("/api/homepage").then((res) => {
  setHero(res.data.hero);
  setHighlights(res.data.highlights);
  setLoading(false);
});

}, []);

const updateHomepage = async () => {
  try {
    await axios.post("/api/homepage/update", { hero, highlights });
    setMessage("Homepage updated successfully");
  } catch (err) {
    console.error(err);
    setMessage("Update failed");
  }
};

const updateHighlight = (index, field, value) => {
  const updated = [...highlights];
  updated[index][field] = value;
  setHighlights(updated);
};

if (loading) return <div className="p-6">Loading...</div>

return (
  <div className="p-6 space-y-6">
    <h2 className="text-2xl font-bold">Manage Home Page</h2>

    {/* Hero Section */}
    <div className="bg-white p-6 rounded shadow">
```

```
<h3 className="text-lg font-semibold mb-4">Hero Section</h3>

<input
  type="text"
  value={hero.heading}
  onChange={(e) => setHero({ ...hero, heading: e.target.value })}
  placeholder="Main Heading"
  className="border p-2 w-full mb-2"
/>

<textarea
  value={hero.subheading}
  onChange={(e) => setHero({ ...hero, subheading: e.target.value })}
  placeholder="Subheading"
  className="border p-2 w-full"
></textarea>

</div>
```

```
{/* Highlights Section */}

<div className="bg-white p-6 rounded shadow">
  <h3 className="text-lg font-semibold mb-4">Highlights</h3>
  {highlights.map((item, index) => (
    <div key={index} className="mb-4">
      <input
        type="text"
        value={item.label}
        onChange={(e) => updateHighlight(index, "label", e.target.value)}
        placeholder="Label"
        className="border p-2 w-full mb-2"
      />
      <input
        type="text"
        value={item.desc}
      />
    
```

```

        onChange={(e) => updateHighlight(index, "desc", e.target.value)}
        placeholder="Description"
        className="border p-2 w-full"

      />
    </div>
  )})
</div>

/* Submit Button */
<button
  onClick={updateHomepage}
  className="bg-blue-600 text-white px-6 py-2 rounded hover:bg-blue-700"
>
  Save Changes
</button>

{message && <p className="text-sm text-green-600 mt-4">{message}</p>}
</div>
);
}

```

---

```

import { useEffect, useState } from "react";
import API from "../../api/axios";
import { Trash2 } from "lucide-react";

export default function QuestionManager() {
  const [skills, setSkills] = useState([]);
  const [sets, setSets] = useState([]);
  const [questions, setQuestions] = useState([]);
  const [selectedSkill, setSelectedSkill] = useState("");
  const [selectedSet, setSelectedSet] = useState("");

```

```
const [form, setForm] = useState({  
    question: "",  
    options: ["", "", "", ""],  
    answer: "",  
});  
  
useEffect(() => {  
    fetchSkills();  
}, []);  
  
useEffect(() => {  
    setSelectedSet("");  
    setSets([]);  
    setQuestions([]);  
    fetchSets();  
}, [selectedSkill]);  
  
useEffect(() => {  
    setQuestions([]);  
    fetchQuestions();  
}, [selectedSet]);  
  
const fetchSkills = async () => {  
    try {  
        const res = await API.get("/admin/skills");  
        setSkills(res.data);  
    } catch (err) {  
        console.error("Failed to fetch skills");  
    }  
};
```

```
const fetchSets = async () => {
  if (!selectedSkill) return;
  try {
    const res = await API.get("/admin/question-sets");
    const filtered = res.data.filter(
      (set) => set.skill && set.skill._id === selectedSkill
    );
    setSets(filtered);
  } catch (err) {
    console.error("Failed to fetch sets", err.message);
  }
};

const fetchQuestions = async () => {
  if (!selectedSet) return;
  try {
    const res = await API.get(` /admin/questions/set/${selectedSet}`);
    setQuestions(res.data);
  } catch (err) {
    console.error("Failed to fetch questions");
  }
};

const handleInputChange = (index, value) => {
  const newOptions = [...form.options];
  newOptions[index] = value;
  setForm({ ...form, options: newOptions });
};

const handleAddQuestion = async (e) => {
```

```
e.preventDefault();

if (!form.question || form.options.some((o) => !o) || !form.answer) {
    return alert("Please fill in all fields.");
}

try {
    await API.post("/admin/questions", {
        set: selectedSet,
        question: form.question,
        options: form.options,
        answer: form.answer,
    });
    setForm({ question: "", options: ["", "", "", ""], answer: "" });
    fetchQuestions();
} catch (err) {
    alert("Failed to add question");
}
};

const handleDelete = async (id) => {
    if (!window.confirm("Delete this question?")) return;
    try {
        await API.delete(`admin/questions/${id}`);
        fetchQuestions();
    } catch (err) {
        alert("Failed to delete question");
    }
};

return (
<div className="bg-white p-6 rounded-xl shadow">
```

```
<h2 className="text-2xl font-bold mb-6 text-purple-700">🧠 Manage Questions</h2>
```

```
<div className="flex gap-6 mb-6">  
  <div>  
    <label className="block mb-1 font-medium">Select Skill:</label>  
    <select  
      className="border rounded px-3 py-2 w-52"  
      value={selectedSkill}  
      onChange={(e) => setSelectedSkill(e.target.value)}  
    >  
      <option value="">-- Skill --</option>  
      {skills.map((s) => (  
        <option key={s._id} value={s._id}>  
          {s.name}  
        </option>  
      ))}  
    </select>  
  </div>
```

```
  {sets.length > 0 && (  
    <div>  
      <label className="block mb-1 font-medium">Select Set:</label>  
      <select  
        className="border rounded px-3 py-2 w-60"  
        value={selectedSet}  
        onChange={(e) => setSelectedSet(e.target.value)}  
      >  
        <option value="">-- Set --</option>  
        {sets.map((s) => (  
          <option key={s._id} value={s._id}>  
            {s.title}
```

```
        </option>
    )})
</select>
</div>
)}

</div>

/* Add Question Form */

{selectedSet && (
<form onSubmit={handleAddQuestion} className="mb-8 grid grid-cols-2 gap-4">
<div className="col-span-2">
<label className="block mb-1 font-medium">Question</label>
<input
  type="text"
  value={form.question}
  onChange={(e) => setForm({ ...form, question: e.target.value })}
  className="border rounded px-3 py-2 w-full"
/>
</div>

{[0, 1, 2, 3].map((i) => (
<div key={i}>
<label className="block mb-1 font-medium">Option {i + 1}</label>
<input
  type="text"
  value={form.options[i]}
  onChange={(e) => handleInputChange(i, e.target.value)}
  className="border rounded px-3 py-2 w-full"
/>
</div>
))}
```

```

<div className="col-span-2">
  <label className="block mb-1 font-medium">Correct Answer</label>
  <select
    className="border rounded px-3 py-2 w-full"
    value={form.answer}
    onChange={(e) => setForm({ ...form, answer: e.target.value })}>
    <option value="">-- Choose Answer --</option>
    {form.options.map((opt, idx) => (
      <option key={idx} value={opt}>
        {opt || `Option ${idx + 1}`}
      </option>
    ))}
  </select>
</div>

<div className="col-span-2">
  <button
    type="submit"
    className="bg-purple-600 text-white px-6 py-2 rounded hover:bg-purple-700">
    >
    Add Question
  </button>
</div>
</form>
}

/* Question List */
<h3 className="text-lg font-semibold mb-3">All Questions</h3>
<table className="w-full text-sm bg-white rounded shadow">

```

```

<thead>
  <tr className="bg-gray-100">
    <th className="py-2 px-4 text-left">#</th>
    <th className="py-2 px-4 text-left">Question</th>
    <th className="py-2 px-4 text-left">Answer</th>
    <th className="py-2 px-4 text-left">Action</th>
  </tr>
</thead>
<tbody>
  {questions.length === 0 ? (
    <tr>
      <td colSpan="4" className="text-center py-6 text-gray-500">
        No questions added yet.
      </td>
    </tr>
  ) : (
    questions.map((q, idx) => (
      <tr key={q._id} className="border-t">
        <td className="py-2 px-4">{idx + 1}</td>
        <td className="py-2 px-4">{q.question}</td>
        <td className="py-2 px-4">{q.answer}</td>
        <td className="py-2 px-4">
          <button
            onClick={() => handleDelete(q._id)}
            className="text-red-600 hover:underline"
          >
            <Trash2 size={16} />
          </button>
        </td>
      </tr>
    )))

```

```

        )}
      </tbody>
    </table>
  </div>
);
}

```

---

```

import { useEffect, useState } from "react";
import API from "../../api/axios";
import { Pencil, Trash2, Plus, Check, X } from "lucide-react";

export default function SetManager() {
  const [skills, setSkills] = useState([]);
  const [selectedSkill, setSelectedSkill] = useState("");
  const [sets, setSets] = useState([]);
  const [newSetTitle, setNewSetTitle] = useState("");
  const [EditMode, setEditMode] = useState(null);
  const [editValue, setEditValue] = useState("");

  useEffect(() => {
    fetchSkills();
  }, []);

  useEffect(() => {
    if (selectedSkill) fetchSets();
    else setSets([]);
  }, [selectedSkill]);

  const fetchSkills = async () => {
    try {
      const res = await API.get("/admin/skills");

```

```
    setSkills(res.data);

} catch (err) {
    console.error("Failed to fetch skills", err.message);
}

};

const fetchSets = async () => {
    try {
        const res = await API.get("/admin/question-sets");
        const filtered = res.data.filter(
            (set) =>
            set.skill && (set.skill._id === selectedSkill || set.skill === selectedSkill)
        );
        setSets(filtered);
    } catch (err) {
        console.error("Failed to fetch sets", err.message);
    }
};

const handleAddSet = async (e) => {
    e.preventDefault();
    if (!newSetTitle.trim() || !selectedSkill) return;

    try {
        await API.post("/admin/question-sets", {
            title: newSetTitle.trim(),
            skill: selectedSkill,
            description: "",
        });
        setNewSetTitle("");
        fetchSets();
    }
};
```

```
    } catch (err) {
      console.error("Error adding set", err.message);
      alert("Error adding set: " + (err.response?.data?.message || err.message));
    }
  };

const handleDelete = async (id) => {
  if (!window.confirm("Delete this question set?")) return;
  try {
    await API.delete(`/admin/question-sets/${id}`);
    fetchSets();
  } catch (err) {
    alert("Error deleting set");
  }
};

const startEdit = (id, currentTitle) => {
  setEditMode(id);
  setEditValue(currentTitle);
};

const cancelEdit = () => {
  setEditMode(null);
  setEditValue("");
};

const saveEdit = async (id) => {
  if (!editValue.trim()) return;
  try {
    await API.put(`/admin/question-sets/${id}`, {
      title: editValue.trim(),
    });
  } catch (err) {
    console.error("Error saving edit", err.message);
    alert("Error saving edit: " + (err.response?.data?.message || err.message));
  }
};
```

```

    skill: selectedSkill,
    description: "",
  });
  setEditMode(null);
  setEditValue("");
  fetchSets();
} catch (err) {
  alert("Failed to update set");
}
};

return (
  <div className="bg-white p-6 rounded-xl shadow">
    <h2 className="text-2xl font-bold mb-6 text-indigo-700"> Manage Question Sets</h2>

    <div className="mb-6 flex items-center gap-4">
      <label className="font-medium">Select Skill:</label>
      <select
        className="border rounded px-3 py-2 w-64"
        value={selectedSkill}
        onChange={(e) => setSelectedSkill(e.target.value)}
      >
        <option value="">-- Choose Skill --</option>
        {skills.map((s) => (
          <option key={s._id} value={s._id}>
            {s.name}
          </option>
        )));
      </select>
    </div>
  </div>
)

```

```

{selectedSkill && (
  <form onSubmit={handleAddSet} className="mb-6 flex gap-4">
    <input
      type="text"
      placeholder="Enter Set Title"
      value={newSetTitle}
      onChange={(e) => setNewSetTitle(e.target.value)}
      className="border px-3 py-2 rounded w-1/2"
    />
    <button
      type="submit"
      className="bg-indigo-600 text-white px-4 py-2 rounded flex items-center gap-2"
    >
      <Plus size={18} /> Add Set
    </button>
  </form>
)}
```

```

<table className="w-full bg-white rounded shadow text-sm">
  <thead>
    <tr className="bg-gray-100 text-left">
      <th className="py-2 px-4">#</th>
      <th className="py-2 px-4">Set Title</th>
      <th className="py-2 px-4">Actions</th>
    </tr>
  </thead>
  <tbody>
    {sets.length === 0 ? (
      <tr>
        <td colSpan="3" className="text-center py-6 text-gray-500">
          No sets found for selected skill.
        </td>
      </tr>
    ) : sets.map((set, index) => (
      <tr key={index}>
        <td>{index + 1}</td>
        <td>{set.title}</td>
        <td>
          <button
            type="button"
            onClick={() => handleEditSet(set.id)}
            className="text-indigo-600 hover:underline"
          >Edit</button>
          <button
            type="button"
            onClick={() => handleDeleteSet(set.id)}
            className="text-red-600 hover:underline"
          >Delete</button>
        </td>
      </tr>
    ))}
  </tbody>
</table>
```

```
</td>
</tr>
) : (
sets.map((set, idx) => (
<tr key={set._id} className="border-t">
<td className="py-2 px-4">{idx + 1}</td>
<td className="py-2 px-4">
{editMode === set._id ? (
<input
type="text"
value={editValue}
onChange={(e) => setEditValue(e.target.value)}
className="border px-2 py-1 rounded w-full"
/>
) : (
set.title
)}
</td>
<td className="py-2 px-4 flex gap-2">
{editMode === set._id ? (
<>
<button
onClick={() => saveEdit(set._id)}
className="text-green-600 hover:underline"
>
<Check size={18} />
</button>
<button
onClick={cancelEdit}
className="text-gray-500 hover:underline"
>

```

```

        <X size={18} />
      </button>
    </>
  ) : (
  <>
    <button
      onClick={() => startEdit(set._id, set.title)}
      className="text-blue-600 hover:underline"
    >
      <Pencil size={18} />
    </button>
    <button
      onClick={() => handleDelete(set._id)}
      className="text-red-600 hover:underline"
    >
      <Trash2 size={18} />
    </button>
  </>
)
  >
</td>
</tr>
))
)
</tbody>
</table>
</div>
);
}

```

---

```

// src/admin/SkillManager.jsx
import { useEffect, useState } from "react";

```

```
import API from "../../api/axios";
import { Pencil, Trash2, Plus, Check, X, RefreshCcw, Loader2 } from "lucide-react";

export default function SkillManager() {
  const [skills, setSkills] = useState([]);
  const [newSkill, setNewSkill] = useState("");
  const [editId, setEditId] = useState(null);
  const [editValue, setEditValue] = useState("");
  const [loading, setLoading] = useState(true);

  const load = async () => {
    setLoading(true);
    try {
      const res = await API.get("/admin/skills");
      setSkills(res.data || []);
    } catch (e) {
      console.error("Skills load failed:", e?.response?.data || e.message);
    } finally {
      setLoading(false);
    }
  };
  useEffect(()=>{ load(); }, []);

  const add = async (e) => {
    e.preventDefault();
    if (!newSkill.trim()) return;
    try {
      await API.post("/admin/skills", { name: newSkill.trim() });
      setNewSkill("");
      await load();
    } catch (e) {
```

```

        alert(e?.response?.data?.message || "Add failed");
    }
};

const save = async (id) => {
    if (!editValue.trim()) return;
    try {
        await API.put(`admin/skills/${id}`, { name: editValue.trim() });
        setEditId(null); setEditValue("");
        await load();
    } catch (e) {
        alert(e?.response?.data?.message || "Update failed");
    }
};

const del = async (id) => {
    if (!window.confirm("Delete this skill?")) return;
    try {
        await API.delete(`admin/skills/${id}`);
        await load();
    } catch (e) {
        alert(e?.response?.data?.message || "Delete failed");
    }
};

return (
    <div className="bg-white p-6 rounded-xl shadow">
        <div className="flex items-center justify-between mb-4">
            <h2 className="text-2xl font-bold text-indigo-700">✿ Manage Skills</h2>
            <button onClick={load} className="px-3 py-2 rounded-lg border text-sm flex items-center gap-2">

```

```

<RefreshCcw size={16}/> Refresh
</button>
</div>

<form onSubmit={add} className="mb-6 flex gap-3">
  <input
    type="text"
    placeholder="New skill name"
    value={newSkill}
    onChange={(e)=>setNewSkill(e.target.value)}
    className="border px-3 py-2 rounded w-64"
  />
  <button className="bg-indigo-600 text-white px-4 py-2 rounded flex items-center gap-2">
    <Plus size={18}/> Add Skill
  </button>
</form>

{loading ? (
  <div className="text-gray-500 flex items-center gap-2"><Loader2 className="animate-spin"/>
  Loading...</div>
) : (
  <table className="w-full bg-white rounded shadow text-sm">
    <thead>
      <tr className="bg-gray-100 text-left">
        <th className="py-2 px-4">#</th>
        <th className="py-2 px-4">Skill</th>
        <th className="py-2 px-4">Actions</th>
      </tr>
    </thead>
    <tbody>
      {skills.length === 0 ? (

```

```

<tr><td colSpan="3" className="text-center py-6 text-gray-500">No skills yet.</td></tr>

) : skills.map((s, idx)=>(
  <tr key={s._id} className="border-t">
    <td className="py-2 px-4">{idx+1}</td>
    <td className="py-2 px-4">
      {editId === s._id ? (
        <input value={editValue} onChange={(e)=>setEditValue(e.target.value)}
        className="border px-2 py-1 rounded w-full" />
      ) : s.name}
    </td>
    <td className="py-2 px-4 flex gap-2">
      {editId === s._id ? (
        <>
          <button onClick={()=>save(s._id)} className="text-green-600"><Check
size={18}/></button>
          <button onClick={()=>{ setEditId(null); setEditValue(""); }} className="text-gray-500"><X
size={18}/></button>
        </>
      ) : (
        <>
          <button onClick={()=>{ setEditId(s._id); setEditValue(s.name); }} className="text-blue-
600"><Pencil size={18}/></button>
          <button onClick={()=>del(s._id)} className="text-red-600"><Trash2
size={18}/></button>
        </>
      )}
    </td>
  </tr>
))}

</tbody>
</table>
)}


</div>

```

```
);

}



---



```
import { useEffect, useState } from "react";
import API from "../../api/axios";
import { Search, Download, Loader2 } from "lucide-react";
import { CSVLink } from "react-csv";

function SkillTestResults() {
  const [results, setResults] = useState([]);
  const [filtered, setFiltered] = useState([]);
  const [loading, setLoading] = useState(true);
  const [query, setQuery] = useState("");
  const [sortKey, setSortKey] = useState("date");
  const [sortOrder, setSortOrder] = useState("desc");

  const [currentPage, setCurrentPage] = useState(1);
  const resultsPerPage = 10;

  useEffect(() => {
    fetchResults();
  }, []);

  const csvHeaders = [
    { label: "Name", key: "user.name" },
    { label: "Email", key: "user.email" },
    { label: "Skill", key: "skill.name" },
    { label: "Score", key: "score" },
    { label: "Total", key: "total" },
    { label: "Date", key: "createdAt" }
  ];
}
```


```

```
const csvData = results.map((r) => ({
  "user.name": r.user?.name || "N/A",
  "user.email": r.user?.email || "N/A",
  "skill.name": r.skill?.name || "N/A",
  score: r.score,
  total: r.total,
  createdAt: new Date(r.createdAt).toLocaleString()
}));
```

  

```
const fetchResults = async () => {
  try {
    const res = await API.get("/admin/skill-test-results");
    setResults(res.data);
    setFiltered(res.data);
  } catch (err) {
    console.error("Failed to load results:", err.response?.data || err.message);
  } finally {
    setLoading(false);
  }
};
```

  

```
useEffect(() => {
  const q = query.toLowerCase();
  const filteredData = results.filter(
    (r) =>
      r.user?.name?.toLowerCase().includes(q) ||
      r.user?.email?.toLowerCase().includes(q) ||
      r.skill?.name?.toLowerCase().includes(q)
  );
  setFiltered(filteredData);
});
```

```

setcurrentPage(1);

}, [query, results]);

const sorted = [...filtered].sort((a, b) => {
  if (sortKey === "date") {
    return sortOrder === "asc"
      ? new Date(a.createdAt) - new Date(b.createdAt)
      : new Date(b.createdAt) - new Date(a.createdAt);
  } else if (sortKey === "score") {
    return sortOrder === "asc" ? a.score - b.score : b.score - a.score;
  } else if (sortKey === "name") {
    return sortOrder === "asc"
      ? (a.user?.name || "").localeCompare(b.user?.name || "")
      : (b.user?.name || "").localeCompare(a.user?.name || "");
  }
  return 0;
});

const indexOfLast = currentPage * resultsPerPage;
const indexOfFirst = indexOfLast - resultsPerPage;
const currentResults = sorted.slice(indexOfFirst, indexOfLast);
const totalPages = Math.ceil(filtered.length / resultsPerPage);

return (
  <div className="max-w-7xl mx-auto py-10 px-4">
    <div className="flex justify-between items-center mb-6 flex-wrap gap-3">
      <h2 className="text-3xl font-bold text-gray-800">  Skill Test Results </h2>

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

```

```
        type="text"
        placeholder="Search by name, email, skill..."
        className="border rounded-lg px-4 py-2 w-64"
        value={query}
        onChange={(e) => setQuery(e.target.value)}
      />
<Search className="absolute right-2 top-2.5 text-gray-500" size={18} />
</div>

<select
  className="border px-3 py-2 rounded-lg"
  value={sortKey}
  onChange={(e) => setSortKey(e.target.value)}
>
  <option value="date">Sort by Date</option>
  <option value="score">Sort by Score</option>
  <option value="name">Sort by Name</option>
</select>

<button
  className="bg-blue-600 text-white px-4 py-2 rounded hover:bg-blue-700"
  onClick={() => setSortOrder((prev) => (prev === "asc" ? "desc" : "asc"))}
>
  {sortOrder === "asc" ? "↑ Asc" : "↓ Desc"}
</button>

<CSVLink
  headers={csvHeaders}
  data={csvData}
  filename="skill-test-results.csv"
  className="flex items-center gap-1 border px-4 py-2 rounded hover:bg-gray-100"
```

```

>
<Download size={18} />
Export CSV
</CSVLink>
</div>
</div>

{loading ? (
  <div className="text-center py-20 text-gray-500">
    <Loader2 className="mx-auto animate-spin" size={32} />
    <p className="mt-2">Loading results...</p>
  </div>
) : currentResults.length === 0 ? (
  <p className="text-gray-500 text-center">No matching results found.</p>
) : (
  <>
    <div className="overflow-x-auto border rounded-lg">
      <table className="w-full text-sm text-left table-auto">
        <thead className="bg-gray-100 text-gray-700 uppercase text-xs">
          <tr>
            <th className="px-4 py-3">Name</th>
            <th className="px-4 py-3">Email</th>
            <th className="px-4 py-3">Skill</th>
            <th className="px-4 py-3">Score</th>
            <th className="px-4 py-3">Total</th>
            <th className="px-4 py-3">Date</th>
          </tr>
        </thead>
        <tbody>
          {currentResults.map((r) => (
            <tr key={r._id} className="border-b hover:bg-gray-50">

```

```

<td className="px-4 py-2">{r.user?.name || "N/A"}</td>
<td className="px-4 py-2">{r.user?.email || "N/A"}</td>
<td className="px-4 py-2">{r.skill?.name || "N/A"}</td>
<td className="px-4 py-2 font-semibold">
  {r.score}
  <span
    className={`text-xs font-medium ml-1 px-2 py-0.5 rounded-full
    ${r.score / r.total >= 0.8 ? "bg-green-100 text-green-700"
    : r.score / r.total >= 0.5 ? "bg-yellow-100 text-yellow-700"
    : "bg-red-100 text-red-700`}
  >
    {Math.round((r.score / r.total) * 100)}%
  </span>
</td>
<td className="px-4 py-2">{r.total}</td>
<td className="px-4 py-2">{new Date(r.createdAt).toLocaleString()}</td>
</tr>
))}>
</tbody>
</table>
</div>

<div className="mt-6 flex justify-center gap-2">
  {[...Array(totalPages)].map((_, i) => (
    <button
      key={i}
      onClick={() => setCurrentPage(i + 1)}
      className={`w-8 h-8 text-sm rounded-full border ${
        currentPage === i + 1
        ? "bg-blue-600 text-white"
        : "border-gray-300 text-gray-700"
      `}
    >{i + 1}</button>
  ))}
</div>

```

```
: "bg-white text-gray-700"
`}`}
>
{i + 1}
</button>
))}
</div>
</>
)}
</div>
);
}
```

```
export default SkillTestResults;
```

---

```
import { useEffect, useMemo, useState } from "react";
import API from "../../api/axios";
import useSmartState from "../../hooks/useSmartState";
import {
  Briefcase, Users2, Trophy, Video, Handshake, Building2, Pencil, Trash2, X,
  Bell, Search, ChevronDown, Settings, ClipboardList, Sparkles, Plus,
  ShieldCheck, BarChart3, Globe2, MapPin, Phone, Mail, Menu, LogOut, Calendar,
} from "lucide-react";
import { Link, useNavigate } from "react-router-dom";

/* ----- small helpers (PASTEL SAFE) ----- */
const apiOrigin =
  (API.defaults.baseURL && new URL(API.defaults.baseURL).origin) ||
  window.location.origin;
```

```

const toAbsolute = (p) => (!p ? null : p.startsWith("http") ? p : `${apiOrigin}${p.startsWith("/") ? p : `/${p}`}`);

```

/\* Static class maps to avoid purge issues \*/

```

const toneMap = {
  blue: { chip: "bg-blue-50 text-blue-800 border-blue-200", icon: "bg-blue-50 text-blue-700 border-blue-200" },
  indigo: { chip: "bg-indigo-50 text-indigo-800 border-indigo-200", icon: "bg-indigo-50 text-indigo-700 border-indigo-200" },
  violet: { chip: "bg-violet-50 text-violet-800 border-violet-200", icon: "bg-violet-50 text-violet-700 border-violet-200" },
  emerald: { chip: "bg-emerald-50 text-emerald-800 border-emerald-200", icon: "bg-emerald-50 text-emerald-700 border-emerald-200" },
  slate: { chip: "bg-slate-50 text-slate-800 border-slate-200", icon: "bg-slate-50 text-slate-700 border-slate-200" },
};

```

```

const Badge = ({ children, tone = "blue" }) => {
  const t = toneMap[tone] || toneMap.blue;
  return (
    <span className={`inline-flex items-center gap-1 px-3 py-1 rounded-full text-xs font-medium border ${t.chip}`}>
      {children}
    </span>
  );
};

const Card = ({ className = "", children }) => (
  <div className={`bg-white/80 backdrop-blur rounded-3xl border border-slate-200/60 shadow-[0_2px_0_rgba(0,0,0,0.02),0_14px_30px_-12px_rgba(2,6,23,0.18)] ${className}`}>
    {children}
  </div>
);

```

```

const SectionCard = ({ title, icon, children, actions }) => (
  <Card className="p-6 md:p-8">
    <div className="flex items-center justify-between mb-4">
      <div className="flex items-center gap-3">
        {icon}
        <h3 className="text-lg font-semibold text-gray-800">{title}</h3>
      </div>
      <div className="flex gap-2">{actions}</div>
    </div>
    {children}
  </Card>
);

/* Reusable pastel buttons */

const Btn = ({ className = "", ...props }) => (
  <button
    {...props}
    className={`inline-flex items-center justify-center rounded-xl px-3.5 py-2 text-sm font-medium shadow-sm transition ${className}`}
  />
);

const BtnPrimary = (p) => <Btn {...p} className={`bg-blue-600 text-white hover:bg-blue-700 ${p.className || ""}`}>;
const BtnGhost = (p) => <Btn {...p} className={`border border-slate-200 bg-white/70 hover:bg-white ${p.className || ""}`}>;
const BtnDanger = (p) => <Btn {...p} className={`bg-rose-600 text-white hover:bg-rose-700 ${p.className || ""}`}>;
const BtnLite = (p) => <Btn {...p} className={`bg-slate-100 text-slate-700 hover:bg-slate-200 ${p.className || ""}`}>

const takeArray = (v) => {
  if (Array.isArray(v)) return v;
  if (!v || typeof v !== "object") return [];
}

```

```

    return v.items || v.data || v.results || v.rows || v.applications || v.jobs || v.webinars || v.hackathons || v.partnerships || [];
};

const fmtDT = (iso) => (iso ? new Date(iso).toLocaleString() : "-");

/* ----- inline scheduling (unchanged behavior, softer UI) ----- */

function ScheduleButton({ application, onUpdated }) {
  const [open, setOpen] = useState(false);
  const [busy, setBusy] = useState(false);
  const iv = application.interview || {};
  const [form, setForm] = useState({
    startsAt: iv.startsAt ? new Date(iv.startsAt).toISOString().slice(0,16) : "",
    durationMins: iv.durationMins || 45,
    stage: iv.stage || "screening",
  });
}

const schedule = async () => {
  try {
    setBusy(true);
    if (!form.startsAt) return alert("Pick date & time");
    const body = {
      startsAt: form.startsAt,
      durationMins: Number(form.durationMins || 45),
      stage: form.stage || "screening",
    };
    if (iv?.roomId) {
      await API.put(`/company/applications/${application._id}/reschedule-interview`, body);
    } else {
      await API.post(`/company/applications/${application._id}/schedule-interview`, body);
    }
    setOpen(false);
  } catch (err) {
    console.error(err);
  }
}

```

```
        onUpdated?.();

    } catch (e) {
        alert(e?.response?.data?.message || "Scheduling failed");
    } finally {
        setBusy(false);
    }
};

const cancel = async () => {
    if (!iv?.roomId) return;
    if (!window.confirm("Cancel this interview?")) return;
    try {
        setBusy(true);
        await API.delete(`/company/applications/${application._id}/cancel-interview`);
        setOpen(false);
        onUpdated?.();
    } catch (e) {
        alert(e?.response?.data?.message || "Cancel failed");
    } finally {
        setBusy(false);
    }
};

return (
    <>
    <button onClick={() => setOpen(true)} className="text-indigo-700 hover:underline">
        {iv?.roomId ? "Reschedule" : "Schedule"}
    </button>

    {open && (
        <div className="fixed inset-0 z-50 bg-black/40 grid place-items-center p-4">

```

```
<div className="bg-white/90 backdrop-blur w-full max-w-md rounded-2xl border border-slate-200 p-5 shadow-xl">

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

    <div className="font-semibold">{iv?.roomId ? "Reschedule Interview" : "Schedule Interview"}</div>

    <button onClick={() => setOpen(false)} className="text-gray-500">x</button>

  </div>

  <div className="space-y-3">

    <div>

      <label className="text-xs text-gray-600">Stage</label>

      <select
        value={form.stage}
        onChange={(e)=>setForm(f=>({...f, stage: e.target.value}))}
        className="w-full border rounded-xl px-3 py-2 bg-white/70"
      >

        <option value="screening">Screening</option>
        <option value="technical">Technical</option>
        <option value="behavioral">Behavioral</option>
        <option value="panel">Panel</option>
        <option value="final">Final</option>
      </select>

    </div>

    <div>

      <label className="text-xs text-gray-600">Start time</label>

      <input
        type="datetime-local"
        value={form.startsAt}
        onChange={(e)=>setForm(f=>({...f, startsAt: e.target.value}))}
        className="w-full border rounded-xl px-3 py-2 bg-white/70"
      />

    </div>

  </div>

</div>
```

```
<div className="text-[11px] text-gray-500 mt-1">  
  Your timezone: {Intl.DateTimeFormat().resolvedOptions().timeZone}  
</div>  
</div>
```

```
<div>  
  <label className="text-xs text-gray-600">Duration (mins)</label>  
  <input  
    type="number" min={15} step={5}  
    value={form.durationMins}  
    onChange={(e)=>setForm(f=>({...f, durationMins: e.target.value}))}  
    className="w-full border rounded-xl px-3 py-2 bg-white/70"  
  />  
</div>
```

```
<div className="flex items-center justify-between pt-2">  
  {iv?.roomId ? (  
    <BtnGhost onClick={cancel} disabled={busy} className="text-rose-600">  
      Cancel Interview  
    </BtnGhost>  
  ) : <span />}
```

```
<div className="flex gap-2">  
  <BtnGhost onClick={()=>setOpen(false)}>Close</BtnGhost>  
  <BtnPrimary onClick={schedule} disabled={busy}>  
    {busy ? "Saving..." : (iv?.roomId ? "Save" : "Schedule")}  
  </BtnPrimary>  
</div>  
</div>
```

```
{!!iv?.roomId && (
```

```

<div className="border rounded-xl p-3 bg-slate-50/70 text-xs">
  <div className="font-medium mb-1">Join Links</div>
  <div className="flex flex-col gap-1">
    <Link to={`/company/webinar/${iv.roomId}`} className="text-indigo-700 underline">Company (Host)</Link>
    <Link to={`/student/webinar/${iv.roomId}`} className="text-blue-700 underline">Student (Viewer)</Link>
  </div>
</div>
</div>
</div>
</div>
</div>
</div>
);
</>
};

}

```

```

function HackStatusControl({ value, onChange }) {
  const [v, setV] = useState(value || "upcoming");
  useEffect(() => { setV(value || "upcoming"); }, [value]);
  return (
    <div className="flex items-center gap-2">
      <select
        value={v}
        onChange={({e}) => setV(e.target.value)}
        className="border rounded-xl px-3 py-1.5 text-sm bg-white/70"
      >
        <option value="upcoming">Upcoming</option>
        <option value="live">Live</option>
        <option value="ended">Ended</option>
      </select>
    </div>
  );
}


```

```

        <BtnPrimary onClick={() => onChange(v)} className="px-3 py-1.5">Save</BtnPrimary>
      </div>
    );
}

function EditHackathonButton({ hack, onSave }){
  const [open, setOpen] = useState(false);

  return (
    <>
      <BtnGhost onClick={() => setOpen(true)} className="px-3 py-1.5 text-sm">Edit</BtnGhost>
      {open && (
        <div className="fixed inset-0 z-50 bg-black/40 grid place-items-center p-4">
          <div className="bg-white/90 backdrop-blur rounded-2xl w-full max-w-2xl p-6 border border-slate-200">
            <div className="flex items-center justify-between mb-3">
              <h3 className="font-semibold">Edit Hackathon</h3>
              <button onClick={() => setOpen(false)} className="text-gray-500">Close</button>
            </div>
            <form
              onSubmit={async (e) => {
                e.preventDefault();

                const f = new FormData(e.currentTarget);

                await API.put(`company/hackathons/${hack._id}`, {
                  title: f.get("title"),
                  prize: f.get("prize"),
                  brief: f.get("brief"),
                  rules: f.get("rules"),
                  resources: f.get("resources"),
                  startAt: f.get("startAt"),
                  endAt: f.get("endAt"),
                  visibility: f.get("visibility"),
                });
              }}
            >
              <div>
                <div>
                  <div>
                    <div>
                      <div>
                        <div>
                          <div>
                            <div>
                              <div>
                                <div>
                                  <div>
                                    <div>
                                      <div>
                                        <div>
                                          <div>
                                            <div>
                                              <div>
                                                <div>
                                                  <div>
                                                    <div>
                                                      <div>
                                                        <div>
                                                          <div>
                                                            <div>
                                                              <div>
                                                                <div>
                                                                  <div>
                                                                    <div>
                                                                      <div>
                                                                        <div>
                                                                          <div>
                                                                            <div>
                                                                              <div>
                                                                                <div>
                                                                                  <div>
                                                                                    <div>
                                                                                      <div>
                                                                                        <div>
              </div>
            </form>
          </div>
        </div>
      )};
    )
}

```

```

        bannerUrl: f.get("bannerUrl"),
    });

    setOpen(false);

    onSaved?.();

}

className="grid md:grid-cols-2 gap-4"

>

<input name="title" defaultValue={hack.title} placeholder="Hackathon Title"
className="border rounded-xl px-3 py-2 bg-white/70" />

<input name="prize" defaultValue={hack.prize} placeholder="Prize / Awards"
className="border rounded-xl px-3 py-2 bg-white/70" />

<textarea name="brief" defaultValue={hack.brief} placeholder="Brief / Problem Statement"
className="md:col-span-2 border rounded-xl px-3 py-2 bg-white/70" />

<textarea name="rules" defaultValue={hack.rules} placeholder="Rules / Evaluation Criteria"
className="md:col-span-2 border rounded-xl px-3 py-2 bg-white/70" />

<textarea name="resources" defaultValue={hack.resources} placeholder="Resources /
Datasets / Links" className="md:col-span-2 border rounded-xl px-3 py-2 bg-white/70" />

<input type="datetime-local" name="startAt" defaultValue={hack.startAt?.slice(0,16)}
className="border rounded-xl px-3 py-2 bg-white/70" />

<input type="datetime-local" name="endAt" defaultValue={hack.endAt?.slice(0,16)}
className="border rounded-xl px-3 py-2 bg-white/70" />

<select name="visibility" defaultValue={hack.visibility || "public"} className="border
rounded-xl px-3 py-2 bg-white/70">

    <option value="public">Public</option>
    <option value="private">Private</option>
</select>

<input name="bannerUrl" defaultValue={hack.bannerUrl || ""} placeholder="Banner image
URL (optional)" className="border rounded-xl px-3 py-2 bg-white/70" />

<div className="md:col-span-2 flex gap-2 pt-2">

    <BtnGhost type="button" onClick={() => setOpen(false)}>Cancel</BtnGhost>
    <BtnPrimary type="submit">Save Changes</BtnPrimary>
</div>
</form>
</div>

```

```

        </div>
    )}
</>
);
}

function HackathonRegistrations({ hackathonId }) {
  const [rows, setRows] = useState(null);
  const load = async () => {
    try {
      const res = await API.get(`/company/hackathons/${hackathonId}/registrations`);
      setRows(res.data || []);
    } catch { setRows([]); }
  };
  useEffect(() => { load(); }, [hackathonId]);
  if (!rows) return <div className="text-sm text-gray-500">Loading...</div>;
  if (rows.length === 0) return <div className="text-sm text-gray-500">No registrations yet.</div>;
  return (
    <div className="max-h-64 overflow-auto text-sm">
      <table className="w-full text-sm overflow-hidden rounded-2xl border border-slate-200">
        <thead className="bg-slate-50/80">
          <tr><th className="p-3 text-left">Student</th><th className="p-3 text-left">Team</th><th className="p-3">When</th></tr>
        </thead>
        <tbody>
          {rows.map(r => (
            <tr key={r._id} className="border-b">
              <td className="p-3">{r.student?.name} <span className="text-gray-400">{r.student?.email}</span></td>
              <td className="p-3">{r.teamName || "-"}</td>
              <td className="p-3 text-center">{new Date(r.createdAt).toLocaleString()}</td>
            </tr>
          ))
        )
      </table>
    </div>
  );
}

```

```

        ))}
      </tbody>
    </table>
  </div>
);

}

function HackathonSubmissions({ hackathonId }) {
  const [rows, setRows] = useState(null);
  const [edit, setEdit] = useState({});
  const [busyId, setBusyId] = useState(null);

  const load = async () => {
    try {
      const res = await API.get(` /company/hackathons/${hackathonId}/submissions`);
      const list = res.data || [];
      setRows(list);
      const cache = {};
      list.forEach(s => { cache[s._id] = { score: s.score ?? "", feedback: s.feedback ?? "" }; });
      setEdit(cache);
    } catch {
      setRows([]);
      setEdit({});
    }
  };
  useEffect(() => { load(); }, [hackathonId]);

  const onChange = (id, field, value) => {
    setEdit(prev => { ...prev, [id]: { ...((prev[id] || {})), [field]: value } }));
  };
}

```

```

const save = async (id) => {
  try {
    setBusyId(id);
    const { score, feedback } = edit[id] || {};
    await API.post(`/company/hackathons/${hackathonId}/judge/${id}`, {
      score: Number(score || 0), feedback: feedback || ""
    });
    await load();
  } catch (e) {
    alert(e?.response?.data?.message || "Save failed");
  } finally {
    setBusyId(null);
  }
};

if (!rows) return <div className="text-sm text-gray-500">Loading...</div>;
if (rows.length === 0) return <div className="text-sm text-gray-500">No submissions yet.</div>

return (
  <div className="max-h-64 overflow-auto text-sm">
    <table className="w-full text-sm overflow-hidden rounded-2xl border border-slate-200">
      <thead className="bg-slate-50/80">
        <tr>
          <th className="p-3 text-left">Student</th>
          <th className="p-3">Repo</th>
          <th className="p-3">Demo</th>
          <th className="p-3">File</th>
          <th className="p-3">Score</th>
          <th className="p-3">Feedback</th>
          <th className="p-3">Rank</th>
          <th className="p-3">Action</th>
        </tr>
      </thead>
      <tbody>
        {rows.map((row) => {
          return <tr key={row.id}>
            <td>{row.student}</td>
            <td>{row.repo}</td>
            <td>{row.demo}</td>
            <td>{row.file}</td>
            <td>{row.score}</td>
            <td>{row.feedback}</td>
            <td>{row.rank}</td>
            <td>{row.action}</td>
          </tr>
        })}
      </tbody>
    </table>
  </div>
);

```

```

        </tr>
    </thead>
    <tbody>
        {rows.map(s => {
            const cur = edit[s._id] || { score: "", feedback: "" };

            return (
                <tr key={s._id} className="border-b">
                    <td className="p-3">{s.student?.name} <span className="text-gray-400">{s.student?.email}</span></td>
                    <td className="p-3 text-center">{s.repoUrl ? <a className="text-blue-700 underline" href={s.repoUrl} target="_blank" rel="noreferrer">Repo</a> : "—" }</td>
                    <td className="p-3 text-center">{s.demoUrl ? <a className="text-blue-700 underline" href={s.demoUrl} target="_blank" rel="noreferrer">Demo</a> : "—" }</td>
                    <td className="p-3 text-center">{s.fileUrl ? <a className="text-blue-700 underline" href={s.fileUrl} target="_blank" rel="noreferrer" download>File</a> : "—" }</td>
                <td className="p-3">
                    <input
                        type="number" min="0" max="100"
                        value={cur.score}
                        onChange={(e)=>onChange(s._id, "score", e.target.value)}
                        className="w-20 border rounded-xl px-2 py-1 bg-white/70"
                    />
                </td>
                <td className="p-3">
                    <input
                        value={cur.feedback}
                        onChange={(e)=>onChange(s._id, "feedback", e.target.value)}
                        className="w-48 border rounded-xl px-2 py-1 bg-white/70"
                    />
                </td>
                <td className="p-3 text-center">{s.rank ?? "—" }</td>
                <td className="p-3">

```

```

        <BtnPrimary
            onClick={()=>save(s._id)}
            disabled={busyId === s._id}
            className="px-2 py-1"
        >
            {busyId === s._id ? "Saving..." : "Save"}
        </BtnPrimary>
    </td>
</tr>
);
}
</tbody>
</table>
</div>
);

}

/* ----- schedule modal (kept; softened styles) ----- */
function ScheduleModal({ open, onClose, mode = "create", application, onSave } ) {
    const iv = application?.interview || null;
    const [startsAt, setStartsAt] = useState(() => {
        if (iv?.startsAt) return iv.startsAt.slice(0, 16);
        const d = new Date(Date.now() + 48 * 3600 * 1000);
        d.setMinutes(0, 0, 0);
        d.setHours(10);
        return d.toISOString().slice(0, 16);
    });
    const [durationMins, setDurationMins] = useState(iv?.durationMins || 45);
    const [stage, setStage] = useState(iv?.stage || "technical");
    const [notes, setNotes] = useState(iv?.notes || "");
    const [saving, setSaving] = useState(false);

```

```
useEffect(() => {
  if (!open) return;

  const cur = application?.interview;
  if (cur) {
    setStartsAt(cur.startsAt ? cur.startsAt.slice(0, 16) : startsAt);
    setDurationMins(cur.durationMins || 45);
    setStage(cur.stage || "technical");
    setNotes(cur.notes || "");
  }
  // eslint-disable-next-line react-hooks/exhaustive-deps
}, [open, application?._id]);

const submit = async (e) => {
  e.preventDefault();
  if (!application?._id) return;
  setSaving(true);
  try {
    if (mode === "create" || !iv) {
      await API.post(`/company/applications/${application._id}/schedule-interview`, {
        startsAt,
        durationMins,
        stage,
        notes,
      });
    } else {
      await API.patch(`/company/applications/${application._id}/reschedule-interview`, {
        startsAt,
        durationMins,
        stage,
        notes,
      });
    }
  } catch (err) {
    setSaving(false);
    console.error(err);
  }
};
```

```
    });
}

onSaved?.();
onClose?.();
} catch (err) {
  alert(err?.response?.data?.message || "Failed to save interview");
} finally {
  setSaving(false);
}

};

if (!open) return null;
return (
  <div className="fixed inset-0 z-50 bg-black/40 flex items-center justify-center p-4">
    <div className="bg-white/90 backdrop-blur w-full max-w-xl rounded-2xl shadow-xl overflow-hidden border border-slate-200">
      <div className="px-6 py-4 border-b flex items-center justify-between">
        <div className="flex items-center gap-2">
          <Calendar className="text-blue-600" size={18} />
          <h3 className="text-lg font-semibold">
            {mode === "create" ? "Schedule Interview" : "Reschedule Interview"}
          </h3>
        </div>
        <button onClick={onClose} className="p-2 rounded-lg hover:bg-slate-100">
          <X size={18} />
        </button>
      </div>

      <form onSubmit={submit} className="p-6 space-y-4">
        <div className="grid md:grid-cols-2 gap-4">
          <div>
```

```
<label className="text-sm text-gray-600">Date & time</label>
<input
  type="datetime-local"
  required
  value={startsAt}
  onChange={(e) => setStartsAt(e.target.value)}
  className="w-full border rounded-xl px-3 py-2 bg-white/70"
/>
</div>
<div>
  <label className="text-sm text-gray-600">Duration (mins)</label>
  <input
    type="number" min={15} max={180}
    value={durationMins}
    onChange={(e) => setDurationMins(e.target.value)}
    className="w-full border rounded-xl px-3 py-2 bg-white/70"
  />
</div>
<div>
  <label className="text-sm text-gray-600">Stage</label>
  <select
    value={stage}
    onChange={(e) => setStage(e.target.value)}
    className="w-full border rounded-xl px-3 py-2 bg-white/70"
  >
    <option value="screening">Screening</option>
    <option value="technical">Technical</option>
    <option value="hr">HR</option>
    <option value="final">Final</option>
  </select>
</div>
```

```

<div className="md:col-span-2">
  <label className="text-sm text-gray-600">Notes (visible to company only)</label>
  <textarea
    value={notes}
    onChange={(e) => setNotes(e.target.value)}
    className="w-full border rounded-xl px-3 py-2 bg-white/70"
    rows={3}
    placeholder="Panel members, agenda, topics...">
  />
</div>
</div>

<div className="pt-2 flex justify-end gap-2">
  <BtnGhost type="button" onClick={onClose}>Cancel</BtnGhost>
  <BtnPrimary disabled={saving}>{saving ? "Saving..." : (mode === "create" ? "Schedule" :
  "Update")}</BtnPrimary>
</div>
</form>
</div>
</div>
);

}

/* ----- dashboard ----- */
export default function CompanyDashboard() {
  const navigate = useNavigate();

  /* data */
  const [profile, setProfile] = useState(null);
  const [skills, setSkills] = useState([]);
  const [jobs, setJobs] = useState([]);

```

```
const [webinars, setWebinars] = useState([]);

const [hackathons, setHackathons] = useState([]);

const [partnerships, setPartnerships] = useState([]);

const [applications, setApplications] = useState([]);

const [talent, setTalent] = useState([]);

const [creatingWebinar, setCreatingWebinar] = useState(false);

/* ui */

const [activeTab, setActiveTab] = useSmartState("company.activeTab", "overview");

const [loading, setLoading] = useState(true);

const [savingProfile, setSavingProfile] = useState(false);

const [editingJob, setEditingJob] = useState(null);

const [mobileTabsOpen, setMobileTabsOpen] = useState(false);

/* search */

const [search, setSearch] = useSmartState("company.talent.search", { skillIds: [], minScore: 6 });

/* schedule modal state */

const [scheduleOpen, setScheduleOpen] = useState(false);

const [scheduleMode, setScheduleMode] = useState("create");

const [scheduleApp, setScheduleApp] = useState(null);

const tabs = [
  { id: "overview", label: "Overview" },
  { id: "profile", label: "Profile" },
  { id: "screening", label: "Screening" },
  { id: "talent", label: "Talent" },
  { id: "jobs", label: "Jobs" },
  { id: "hackathons", label: "Hackathons" },
  { id: "webinars", label: "Webinars" },
  { id: "partnerships", label: "Partnerships" },
]
```

```
];

const fetchAll = async () => {
  setLoading(true);
  try {
    const [p, s, j, w, h, pa, apps] = await Promise.all([
      API.get("/company/profile"),
      API.get("/admin/skills"),
      API.get("/company/jobs"),
      API.get("/company/webinars"),
      API.get("/company/hackathons"),
      API.get("/company/partnerships"),
      API.get("/company/applications"),
    ]);
    setProfile(p.data || null);
    setSkills(takeArray(s.data));
    setJobs(takeArray(j.data));
    setWebinars(takeArray(w.data));
    setHackathons(takeArray(h.data));
    setPartnerships(takeArray(pa.data));
    setApplications(takeArray(apps.data));
  } catch (e) {
    console.error("Load dashboard failed:", e?.response?.data || e.message);
  } finally {
    setLoading(false);
  }
};

useEffect(() => { fetchAll(); }, []);
```

```

const kpis = useMemo(() => {
  const openJobs = jobs.filter((j) => j.status === "open").length;
  const totalApps = applications.length;
  const upcomingWebinars = webinars.filter((w) => new Date(w.startsAt) > new Date()).length;
  const activePartnerships = partnerships.filter((p) => p.status === "active").length;
  return [
    { label: "Open Roles", value: openJobs, icon: <Briefcase size={18} />, tone: "blue" },
    { label: "Applications", value: totalApps, icon: <ClipboardList size={18} />, tone: "indigo" },
    { label: "Upcoming Webinars", value: upcomingWebinars, icon: <Video size={18} />, tone: "violet" },
    { label: "Active Partnerships", value: activePartnerships, icon: <Handshake size={18} />, tone: "emerald" },
  ];
}, [jobs, applications, webinars, partnerships]);

/* profile */
const saveProfile = async (e) => {
  e.preventDefault();
  setSavingProfile(true);
  try {
    const f = new FormData(e.currentTarget);
    const body = {
      name: f.get("name"),
      website: f.get("website"),
      logoUrl: f.get("logoUrl"),
      description: f.get("description"),
      size: f.get("size"),
      locations: (f.get("locations") || "").split(",").map(v => v.trim()).filter(Boolean),
      domains: (f.get("domains") || "").split(",").map(v => v.trim()).filter(Boolean),
      contactEmail: f.get("contactEmail"),
      contactPhone: f.get("contactPhone"),
    };
  }
}

```

```
if (profile?._id) {
    const res = await API.put("/company/profile", body);
    setProfile(res.data);
} else {
    try {
        const res = await API.post("/company/profile", body);
        setProfile(res.data);
    } catch (err) {
        if (err?.response?.status === 409) {
            const res2 = await API.put("/company/profile", body);
            setProfile(res2.data);
        } else {
            throw err;
        }
    }
}

} catch (err) {
    console.error("Save profile failed:", err?.response?.data || err.message);
    alert(err?.response?.data?.message || "Failed to save profile");
} finally {
    setSavingProfile(false);
}
};

/* applications */
const updateAppStatus = async (id, status) => {
try {
    await API.patch(`/company/applications/${id}`, { status });
    const refreshed = await API.get("/company/applications");
    setApplications(takeArray(refreshed.data));
}
```

```

} catch (e) {
  console.error("Update application failed:", e?.response?.data || e.message);
}

};

/* schedule actions */

const openScheduleCreate = (a) => { setScheduleMode("create"); setScheduleApp(a); setScheduleOpen(true); };

const openScheduleEdit = (a) => { setScheduleMode("edit"); setScheduleApp(a); setScheduleOpen(true); };

const cancellInterview = async (a) => {
  if (!window.confirm("Cancel this interview?")) return;
  try {
    await API.delete(`/company/applications/${a._id}/cancel-interview`);
    fetchAll();
  } catch (e) {
    alert(e?.response?.data?.message || "Cancel failed");
  }
};

/* talent search */

const runTalentSearch = async () => {
  try {
    const params = new URLSearchParams();
    search.skillIds.forEach((id) => params.append("skillIds", id));
    params.append("minScore", search.minScore);
    const res = await API.get(`/company/talent-search?${params.toString()}`);
    setTalent(res.data);
  } catch (e) {
    console.error("Talent search failed:", e?.response?.data || e.message);
  }
};

```

```

useEffect(() => { if (search.skillIds.length) runTalentSearch(); else setTalent([]); }, [search]);

const toggleSearchSkill = (id) =>
  setSearch((prev) => ({ ...prev, skillIds: prev.skillIds.includes(id) ? prev.skillIds.filter(x => x !== id) : [...prev.skillIds, id] }));

/* jobs */

const [editingJobState, setEditingJobState] = useState(null);
const openEditJob = (job) => {
  const skillIds = (job.skills || []).map((s) => (typeof s === "string" ? s : s._id));
  setEditingJobState({
    _id: job._id, title: job.title || "", type: job.type || "job",
    location: job.location || "", description: job.description || "",
    minScore: job.minScore ?? 0, isFeatured: !job.isFeatured, status: job.status || "open",
    skills: skillIds,
  });
};

const closeEditJob = () => setEditingJobState(null);
const toggleEditSkill = (id) =>
  setEditingJobState((prev) => (!prev ? prev : ({ ...prev, skills: prev.skills.includes(id) ? prev.skills.filter(x => x !== id) : [...prev.skills, id] })));

const createJob = async (e) => {
  e.preventDefault();
  const formEl = e.currentTarget;
  const f = new FormData(formEl);
  const payload = {
    title: (f.get("title") || "").trim(),
    type: f.get("type") || "job",
    location: (f.get("location") || "").trim(),
    skills: Array.from(f.getAll("skills") || []).map(String),
    description: (f.get("description") || "").trim(),
  }
}

```

```
minScore: Number(f.get("minScore") || 0),
isFeatured: f.get("isFeatured") === "on",
};

try {
  await API.post("/company/jobs", payload);
  formEl.reset();
  fetchAll();
} catch (err) {
  alert(err?.response?.data?.message || "Create job failed");
}

};

const saveJobUpdates = async (e) => {
  e.preventDefault();
  if (!editingJobState?._id) return;
  try {
    const p = {
      title: editingJobState.title,
      type: editingJobState.type,
      location: editingJobState.location,
      description: editingJobState.description,
      minScore: Number(editingJobState.minScore || 0),
      isFeatured: !!editingJobState.isFeatured,
      status: editingJobState.status,
      skills: editingJobState.skills,
    };
    await API.put(`/company/jobs/${editingJobState._id}`, p);
    closeEditJob();
    fetchAll();
  } catch (e2) {
    alert(e2?.response?.data?.message || "Update job failed");
  }
};
```

```
        }

    };

const deleteJob = async (id) => {
    if (!window.confirm("Delete this job posting?")) return;
    try { await API.delete(`/company/jobs/${id}`); fetchAll(); }
    catch (e) { alert(e?.response?.data?.message || "Delete failed"); }
};

/* webinars — internal studio */
async function createWebinar(e) {
    e.preventDefault();
    if (creatingWebinar) return;
    setCreatingWebinar(true);

    const formEl = e.currentTarget;
    const f = new FormData(formEl);

    const payload = {
        title: (f.get("title") || "").trim(),
        speaker: (f.get("speaker") || "").trim(),
        description: (f.get("description") || "").trim(),
        startsAt: f.get("startsAt") || null,
        durationMins: Number(f.get("durationMins") || 60),
        visibility: "public",
    };

    try {
        const res = await API.post("/company/webinars", payload);
        console.log("Webinar created", res.data);
        formEl.reset();
    }
}
```

```

        await fetchAll();

    } catch (err) {
        console.error("Create webinar failed:", err?.response?.data || err.message);
        alert(err?.response?.data?.message || "Create webinar failed");
    } finally {
        setCreatingWebinar(false);
    }
}

/* navbar logout */
const logout = () => {
    localStorage.removeItem("userToken");
    localStorage.removeItem("userRole");
    window.location.href = "/login";
};

return (
<div className="min-h-screen bg-gradient-to-b from-rose-50 via-sky-50 to-emerald-50">
    /* navbar */
    <nav className="sticky top-0 z-40 bg-white/80 backdrop-blur border-b">
        <div className="max-w-7xl mx-auto px-4">
            <div className="h-14 flex items-center justify-between">
                <div className="flex items-center gap-3">
                    <button className="md:hidden p-2 rounded-lg hover:bg-gray-100" onClick={() => setMobileTabsOpen(s => !s)}>
                        <Menu size={18} />
                    </button>
                    <div className="flex items-center gap-2">
                        {profile?.logoUrl ? (
                            <img src={profile.logoUrl} alt="logo" className="h-7 w-7 rounded-md object-cover border border-slate-200" />
                        ) : (

```

```
        <div className="h-7 w-7 rounded-md bg-blue-600 text-white grid place-items-center text-xs font-bold">C</div>
    )}
    <span className="font-semibold">Company Portal</span>
</div>
</div>

<div className="hidden md:flex items-center flex-1 max-w-lg mx-4">
    <div className="relative w-full">
        <Search className="absolute left-3 top-2.5 text-gray-400" size={16} />
        <input
            type="text"
            placeholder="Search jobs, candidates, webinars...">
        <span className="w-full border rounded-xl pl-9 pr-3 py-2 text-sm bg-white/70 focus:outline-none focus:ring-2 focus:ring-blue-200">
            />
    </div>
</div>

<div className="flex items-center gap-2">
    <button className="relative p-2 rounded-lg hover:bg-gray-100">
        <Bell size={18} />
        <span className="absolute -top-0.5 -right-0.5 h-2 w-2 bg-red-500 rounded-full" />
    </button>
    <div className="h-8 w-8 rounded-xl grid place-items-center bg-slate-100 text-slate-700 border border-slate-200">
        <ShieldCheck size={16} />
    </div>
    <button className="hidden md:flex items-center gap-2 px-3 py-1.5 rounded-xl border bg-white/70 hover:bg-white">
        <span className="text-sm">{profile?.name || "Profile"}</span>
        <ChevronDown size={16} />
    </button>
</div>
```

```
</button>

<Btn className="hidden md:inline-flex bg-gray-900 text-white hover:bg-gray-800"
onClick={logout}>

  <LogOut size={16} /> <span className="text-sm ml-1">Logout</span>

</Btn>
</div>
</div>

<div className={`border-t md:border-none ${mobileTabsOpen ? "block" : "hidden"
md:block"}`}>

  <div className="flex items-center gap-1 py-2 overflow-x-auto">

    {tabs.map((t) => {

      const active = activeTab === t.id;

      return (
        <button
          key={t.id}
          onClick={() => { setActiveTab(t.id); setMobileTabsOpen(false); }}
          className={`px-4 py-2 rounded-xl text-sm font-medium transition ${
            active ? "bg-blue-600 text-white shadow" : "text-gray-700 hover:bg-gray-100"
          }`}
        >
          {t.label}
        </button>
      );
    })}
  </div>
</div>
</div>
</nav>

/* body */

<div className="max-w-7xl mx-auto p-4 md:p-6 space-y-6">
```

```
{loading ? (
  <Card className="p-10 grid place-items-center text-gray-500">Loading dashboard...</Card>
) : (
  <>
    /* OVERVIEW */
  </>
  {activeTab === "overview" && (
    <>
      <div className="grid sm:grid-cols-2 lg:grid-cols-4 gap-4">
        {kpis.map((k) => {
          const t = toneMap[k.tone] || toneMap.blue;
          return (
            <Card key={k.label} className="p-5">
              <div className="flex items-center justify-between">
                <div>
                  <p className="text-xs text-gray-500">{k.label}</p>
                  <p className="text-2xl font-semibold mt-1">{k.value}</p>
                </div>
                <div className={`h-10 w-10 rounded-2xl grid place-items-center border ${t.icon}`}>
                  {k.icon}
                </div>
              </div>
            </Card>
          );
        });
      </>
    </div>
  )};
</div>

<Card className="p-5">
  <div className="flex items-center gap-2 mb-3">
    <Sparkles className="text-blue-600" size={18} />
    <h3 className="font-semibold">Quick Actions</h3>
  </div>
```

```

<div className="flex flex-wrap gap-2">
  <BtnGhost onClick={() => setActiveTab("jobs")}><Plus size={16} className="mr-1" /> Post a Job</BtnGhost>
  <BtnGhost onClick={() => setActiveTab("talent")}><Users2 size={16} className="mr-1" /> Find Talent</BtnGhost>
  <BtnGhost onClick={() => setActiveTab("webinars")}><Video size={16} className="mr-1" /> Host Webinar</BtnGhost>
  <BtnGhost onClick={() => setActiveTab("hackathons")}><Trophy size={16} className="mr-1" /> Launch Hackathon</BtnGhost>
  <BtnGhost onClick={() => setActiveTab("partnerships")}><Handshake size={16} className="mr-1" /> Partner with University</BtnGhost>
</div>
</Card>

<div className="grid lg:grid-cols-2 gap-4">
  <SectionCard title="Open Roles" icon=<Briefcase className="text-blue-600" />>
  {jobs.filter((j) => j.status === "open").length === 0 ? (
    <p className="text-sm text-gray-500">No open roles. Create one from Jobs tab.</p>
  ) : (
    <ul className="divide-y">
      {jobs.filter((j) => j.status === "open").slice(0, 5).map((j) => (
        <li key={j._id} className="py-3 flex justify-between items-center">
          <div>
            <p className="font-medium">{j.title}</p>
            <div className="flex items-center gap-2 text-xs text-gray-500">
              <Badge tone="blue">{j.type}</Badge>
              <span className="flex items-center gap-1"><MapPin size={12} /> {j.location || "--"}</span>
            </div>
          </div>
        </li>
      ))}
    </ul>
  )}
<button onClick={() => setActiveTab("jobs")}>Manage</button>
</div>

```

```

        ))}
      </ul>
    )}
</SectionCard>

<SectionCard title="Upcoming Webinars" icon={<Video className="text-blue-600" />}>
{webinars.filter((w) => new Date(w.startsAt) > new Date()).length === 0 ? (
  <p className="text-sm text-gray-500">No upcoming webinars scheduled.</p>
) : (
  <ul className="space-y-3">
    {webinars.filter((w) => new Date(w.startsAt) > new Date()).slice(0, 5).map((w) => (
      <li key={w._id} className="border rounded-xl p-3 bg-white/60">
        <div className="flex items-center justify-between">
          <div>
            <p className="font-medium">{w.title}</p>
            <p className="text-xs text-gray-500">
              {new Date(w.startsAt).toLocaleString()} • {w.durationMins} mins
            </p>
          </div>
        <div className="flex gap-3">
          <Link to={`/student/webinar/${w.roomId}`} className="text-blue-700 text-sm underline">
            Join
          </Link>
          <Link to={`/company/webinar/${w.roomId}`} className="text-indigo-700 text-sm underline">
            Open Studio
          </Link>
        </div>
      </li>
    )));
}
</SectionCard>

```

```

        ))}
      </ul>
    )}
  </SectionCard>
</div>
</>
)}

/* PROFILE */

{activeTab === "profile" && (
  <SectionCard title="Company Profile" icon={<Settings className="text-blue-600" />}>
    <form onSubmit={saveProfile} className="grid md:grid-cols-2 gap-4">
      <input name="name" required placeholder="Company Name"
        defaultValue={profile?.name || ""} className="border rounded-xl px-3 py-2 bg-white/70" />
      <input name="website" placeholder="Website" defaultValue={profile?.website || ""}
        className="border rounded-xl px-3 py-2 bg-white/70" />
      <input name="logoUrl" placeholder="Logo URL" defaultValue={profile?.logoUrl || ""}
        className="border rounded-xl px-3 py-2 bg-white/70" />
      <select name="size" defaultValue={profile?.size || "1-10"} className="border rounded-xl
        px-3 py-2 bg-white/70">
        {[["1-10","11-50","51-200","201-500","500+"]].map(s => <option key={s} value={s}>{s}
          employees</option>)}
      </select>
      <input name="contactEmail" placeholder="Contact Email"
        defaultValue={profile?.contactEmail || ""} className="border rounded-xl px-3 py-2 bg-white/70" />
      <input name="contactPhone" placeholder="Contact Phone"
        defaultValue={profile?.contactPhone || ""} className="border rounded-xl px-3 py-2 bg-white/70" />
      <input name="locations" placeholder="Locations (comma separated)"
        defaultValue={profile?.locations?.join(", ") || ""} className="border rounded-xl px-3 py-2 bg-
        white/70 md:col-span-2" />
      <input name="domains" placeholder="Domains/Industry (comma separated)"
        defaultValue={profile?.domains?.join(", ") || ""} className="border rounded-xl px-3 py-2 bg-
        white/70 md:col-span-2" />
      <textarea name="description" placeholder="About company"
        defaultValue={profile?.description || ""} className="border rounded-xl px-3 py-2 bg-white/70
        md:col-span-2" />
    </form>
  </SectionCard>
)
}

```

```

<BtnPrimary disabled={savingProfile} className="md:justify-self-start">
  {savingProfile ? "Saving..." : (profile?._id ? "Update Profile" : "Create Profile")}
</BtnPrimary>
</form>

<div className="mt-6 grid md:grid-cols-2 gap-4">
  <Card className="p-5">
    <div className="flex items-center gap-3">
      {profile?.logoUrl ? <img src={profile.logoUrl} alt="logo" className="h-10 w-10 rounded-xl object-cover border border-slate-200" /> : <div className="h-10 w-10 rounded-xl bg-gray-200" />}
      <div>
        <div className="font-semibold">{profile?.name || "-"}</div>
        <div className="text-xs text-gray-500">{profile?.website || "-"}</div>
      </div>
    </div>
    <p className="text-sm text-gray-600 mt-3">{profile?.description || "—"}</p>
    <div className="mt-3 flex flex-wrap gap-2 text-xs text-gray-600">
      {profile?.domains?.map((d,i) => <Badge key={i} tone="indigo">{d}</Badge>)}
    </div>
  </Card>
  <Card className="p-5 text-sm text-gray-600">
    <div className="flex items-center gap-2"><Globe2 size={14} /> {profile?.website || "—"}</div>
    <div className="flex items-center gap-2 mt-2"><Mail size={14} /> {profile?.contactEmail || "—"}</div>
    <div className="flex items-center gap-2 mt-2"><Phone size={14} /> {profile?.contactPhone || "—"}</div>
    <div className="flex items-center gap-2 mt-2"><MapPin size={14} /> {((profile?.locations || []).join(", ") || "—")}</div>
    <div className="flex items-center gap-2 mt-2"><BarChart3 size={14} /> Size: {profile?.size || "—"}</div>
  </Card>
</div>

```

```

        </SectionCard>
    )}

/* SCREENING */
{activeTab === "screening" && (
    <SectionCard
        title="Smart Candidate Screening"
        icon={<ClipboardList className="text-blue-600" />}
    >
        {applications.length === 0 ? (
            <p className="text-sm text-gray-500">No applications yet.</p>
        ) : (
            <>
                {/* Mobile cards */}
                <div className="md:hidden space-y-3">
                    {applications.map((a) => {
                        const iv = a.interview;
                        const windowText = iv?.startsAt
                            ? `${new Date(iv.startsAt).toLocaleString()} • ${iv.durationMins || 45}m ${iv.stage || "screening"}`
                            : "—";
                        const name = a.student?.name || "-";
                        const email = a.student?.email || "-";
                        const fit = a.screening?.fitScore ?? null;
                        const test = a.screening?.testScore != null ? Math.round(a.screening.testScore * 10) /
                            10 : null;
                        const resumeScore = a.screening?.resumeScore ?? null;

                        const fitTone =
                            fit == null ? "bg-slate-100 text-slate-600" :
                            fit >= 80 ? "bg-emerald-50 text-emerald-700 border-emerald-200" :
                            fit >= 60 ? "bg-amber-50 text-amber-700 border-amber-200" :
                    )}
                </div>
            </>
        )
    )
}

```

```
"bg-rose-50 text-rose-700 border-rose-200";  
  
return (  
  <div key={a._id} className="rounded-2xl border border-slate-200 bg-white/70 p-4">  
    <div className="flex items-start justify-between gap-3">  
      <div>  
        <div className="font-medium text-gray-900">{name}</div>  
        <div className="text-xs text-gray-500">{email}</div>  
        <div className="mt-1 text-xs text-gray-500">{a.job?.title || "—"}</div>  
      </div>  
      <span className={`text-xs px-2.5 py-1 rounded-full border ${fitTone}`}>  
        Fit {fit ?? "—"}  
      </span>  
    </div>  
  
    <div className="mt-4 grid grid-cols-2 gap-3 text-xs">  
      <div className="col-span-2">  
        <div className="flex items-center justify-between mb-1">  
          <span className="text-gray-600">Resume Score</span>  
          <b className="text-gray-800">{resumeScore ?? "—"}</b>  
        </div>  
        <div className="h-2 w-full rounded-full bg-slate-100 overflow-hidden">  
          <div  
            className="h-full rounded-full bg-indigo-400"  
            style={{ width: resumeScore != null ? `${Math.min(100, resumeScore)}%` : "0%"}}>  
        </div>  
      </div>  
    </div>  
  
  <div>
```

```
<div className="flex items-center justify-between mb-1">
  <span className="text-gray-600">Test Score</span>
  <b className="text-gray-800">{test ?? "—"}</b>
</div>

<div className="h-2 w-full rounded-full bg-slate-100 overflow-hidden">
  <div
    className="h-full rounded-full bg-blue-400"
    style={{ width: test != null ? `${Math.min(100, test * 10)}%` : "0%" }}
  />
</div>
</div>

<div>
  <div className="flex items-center justify-between mb-1">
    <span className="text-gray-600">Status</span>
    <b className="text-gray-800 capitalize">{a.status}</b>
  </div>
  <div className="h-2 w-full rounded-full bg-slate-100 overflow-hidden">
    <div
      className="h-full rounded-full bg-sky-400"
      style={{ width:
        a.status === "shortlisted" ? "60%":
        a.status === "offer" ? "100%":
        a.status === "rejected" ? "20%" : "40%"
      }}
    />
  </div>
</div>
</div>

<div className="mt-4 rounded-xl border border-slate-200 bg-slate-50/60 p-2">
```

```

<div className="text-[11px] text-gray-600">{windowText}</div>

{iv?.roomId && (
  <div className="flex items-center gap-3 mt-1">
    <Link to={`/company/webinar/${iv.roomId}`} className="text-indigo-700 underline text-xs">
      Join (Host)
    </Link>
    <Link to={`/student/webinar/${iv.roomId}`} className="text-blue-700 underline text-xs">
      Join (Viewer)
    </Link>
  </div>
)};

</div>

<div className="mt-4 flex flex-wrap gap-2">
  <button
    onClick={() => updateAppStatus(a._id, "shortlisted")}
    className="px-3 py-1.5 rounded-xl border bg-white hover:bg-slate-50 text-sm text-blue-700"
  >
    Shortlist
  </button>

  <ScheduleButton
    application={a}
    onUpdated={async () => {
      const refreshed = await API.get("/company/applications");
      setApplications(refreshed.data || []);
    }}
  />

```

```

        <button
            onClick={() => updateAppStatus(a._id, "offer")}
            className="px-3 py-1.5 rounded-xl bg-emerald-600 text-white text-sm hover:bg-emerald-700"
        >
            Offer
        </button>
        <button
            onClick={() => updateAppStatus(a._id, "rejected")}
            className="px-3 py-1.5 rounded-xl bg-rose-600 text-white text-sm hover:bg-rose-700"
        >
            Reject
        </button>
    </div>
</div>
);

}}}
</div>

/* Desktop table */

<div className="hidden md:block overflow-x-auto">
    <table className="w-full text-sm overflow-hidden rounded-2xl border border-slate-200">
        <thead className="bg-slate-50/80">
            <tr>
                <th className="p-3 text-left">Candidate</th>
                <th className="p-3">Job</th>
                <th className="p-3">Resume</th>
                <th className="p-3">ResumeScore</th>
                <th className="p-3">TestScore</th>
                <th className="p-3">Fit</th>
            </tr>
        </thead>
        <tbody>
            {applicants.map((a) => {
                return (
                    <tr key={a._id}>
                        <td>{a.name}</td>
                        <td>{a.job}</td>
                        <td>{a.resume}</td>
                        <td>{a.resumeScore}</td>
                        <td>{a.testScore}</td>
                        <td>{a.fit}</td>
                    </tr>
                );
            })}
        </tbody>
    </table>
</div>

```

```
 Status | Interview | Action |
{applications.map((a) => {
  const iv = a.interview;
  const windowText = iv?.startsAt
    ? `${new Date(iv.startsAt).toLocaleString()} • ${iv.durationMins}m ${iv.stage}
    || "screening")` ||
    : "--";
  const fit = a.screening?.fitScore ?? null;
  const test = a.screening?.testScore != null ? Math.round(a.screening.testScore * 10)
    / 10 : null;
  const resumeScore = a.screening?.resumeScore ?? null;

  // pastel chips for FIT
  const fitChip =
    fit == null ? "bg-slate-100 text-slate-700 border-slate-200" :
    fit >= 80 ? "bg-emerald-50 text-emerald-700 border-emerald-200" :
    fit >= 60 ? "bg-amber-50 text-amber-700 border-amber-200" :
    "bg-rose-50 text-rose-700 border-rose-200";

  return (
    <tr key={a._id} className="border-b">
      <td className="p-3">
        <div className="flex items-center gap-3">
          <div className="h-8 w-8 rounded-xl bg-slate-100 border border-slate-200 grid
place-items-center text-xs font-semibold text-slate-700">
            {(a.student?.name || "?").slice(0, 2).toUpperCase()}
          </div>
        </div>
      </td>
    </tr>
  );
});

```

```
<div>
  <div className="font-medium text-gray-900">{a.student?.name}</div>
  <div className="text-xs text-gray-500">{a.student?.email}</div>
</div>
</div>
</td>

<td className="p-3 text-center">{a.job?.title}</td>

<td className="p-3 text-center">
  {a.cvUrl ? (
    <a
      className="text-blue-700 underline"
      href={toAbsolute(a.cvUrl)}
      target="_blank"
      rel="noreferrer"
      download
    >
      Resume
    </a>
  ) : (
    "_"
  )}
</td>

/* Resume score with bar */
<td className="p-3">
  <div className="min-w-[120px] mx-auto">
    <div className="flex items-center justify-between text-xs mb-1">
      <span className="text-gray-600">Score</span>
      <b className="text-gray-800">{resumeScore ?? "—"}</b>
    </div>
  </div>
</td>
```

```

        </div>

        <div className="h-2 w-full rounded-full bg-slate-100 overflow-hidden">
            <div
                className="h-full rounded-full bg-indigo-400"
                style={{ width: resumeScore != null ? `${Math.min(100, resumeScore)}%` :
                    "0%" }}>
            />
        </div>
    </div>
</td>

/* Test score with bar */

<td className="p-3">
    <div className="min-w-[120px] mx-auto">
        <div className="flex items-center justify-between text-xs mb-1">
            <span className="text-gray-600">Score</span>
            <b className="text-gray-800">{test ?? "—"}</b>
        </div>
        <div className="h-2 w-full rounded-full bg-slate-100 overflow-hidden">
            <div
                className="h-full rounded-full bg-blue-400"
                style={{ width: test != null ? `${Math.min(100, test * 10)}%` : "0%" }}>
            />
        </div>
    </div>
</td>

/* Fit chip */

<td className="p-3 text-center">
    <span className={`inline-flex items-center px-2.5 py-1 rounded-full border text-
xs ${fitChip}`}>
        {fit ?? "—"}
```

```

        </span>
    </td>

    {/* Status pill */}
    <td className="p-3 text-center">
        <span
            className={`inline-flex items-center px-2.5 py-1 rounded-full border text-xs ${{
                a.status === "shortlisted"
                    ? "bg-blue-50 text-blue-700 border-blue-200"
                : a.status === "offer"
                    ? "bg-emerald-50 text-emerald-700 border-emerald-200"
                : a.status === "rejected"
                    ? "bg-rose-50 text-rose-700 border-rose-200"
                : "bg-slate-50 text-slate-700 border-slate-200"
            }}}
        >
            {a.status}
        </span>
    </td>

    {/* Interview window + links */}
    <td className="p-3 text-center">
        <div className="text-xs">{windowText}</div>
        {iv?.roomId && (
            <div className="flex items-center justify-center gap-3 mt-1">
                <Link
                    to={`/company/webinar/${iv.roomId}`}
                    className="text-indigo-700 underline text-xs"
                >
                    Join (Host)
                </Link>
            </div>
        )}
    </td>

```

```

        <Link
          to={`/student/webinar/${iv.roomId}`}
          className="text-blue-700 underline text-xs"
        >
          Join (Viewer)
        </Link>
      </div>
    )}

</td>

 {/* Actions */}

<td className="p-3">
  <div className="flex gap-2 justify-center">
    <button
      onClick={() => updateAppStatus(a._id, "shortlisted")}
      className="px-3 py-1.5 rounded-xl border bg-white hover:bg-slate-50 text-sm
text-blue-700"
    >
      Shortlist
    </button>

    <ScheduleButton
      application={a}
      onUpdated={async () => {
        const refreshed = await API.get("/company/applications");
        setApplications(refreshed.data || []);
      }}
    />

    <button
      onClick={() => updateAppStatus(a._id, "offer")}

```

```

        className="px-3 py-1.5 rounded-xl bg-emerald-600 text-white text-sm
        hover:bg-emerald-700"
      >
        Offer
      </button>
      <button
        onClick={() => updateAppStatus(a._id, "rejected")}
        className="px-3 py-1.5 rounded-xl bg-rose-600 text-white text-sm hover:bg-
        rose-700"
      >
        Reject
      </button>
    </div>
  </td>
</tr>
);
}
}
</tbody>
</table>
</div>
</>
)
}
</SectionCard>
}

```

```

/* TALENT */
{activeTab === "talent" && (
  <SectionCard
    title="AI-Powered Talent Search"
    icon={<Users2 className="text-blue-600" />}
    actions={

```

```

<div className="flex items-center gap-3">
  <input
    type="number" min={0} max={10} value={search.minScore}
    onChange={(e) => setSearch((s) => ({ ...s, minScore: e.target.value }))}
    className="w-24 border rounded-xl px-3 py-1 text-sm bg-white/70"
    placeholder="Min score"
    title="Min average score (0-10)"
  />
  <BtnPrimary onClick={runTalentSearch}>Search</BtnPrimary>
</div>
}

>

<div className="flex flex-wrap gap-2 mb-4">
  {skills.map((sk) => {
    const active = search.skillIds.includes(sk._id);
    return (
      <button
        key={sk._id}
        onClick={() => toggleSearchSkill(sk._id)}
        className={`px-3 py-1 rounded-full border text-sm ${active ? "bg-blue-600 text-white" : "bg-white/70 hover:bg-white"}`}
      >
        {sk.name}
      </button>
    );
  })}
</div>

{talent.length === 0 ? (
  <p className="text-sm text-gray-500">Pick skills and run search to see ranked candidates.</p>
) : (

```

```

<div className="overflow-x-auto">
  <table className="w-full text-sm overflow-hidden rounded-2xl border border-slate-200">
    <thead className="bg-slate-50/80">
      <tr>
        <th className="p-3 text-left">Student</th>
        <th className="p-3 text-left">University</th>
        <th className="p-3">Avg Req Skill Score</th>
        <th className="p-3">Fit Score</th>
      </tr>
    </thead>
    <tbody>
      {talent.map((t, i) => (
        <tr key={i} className="border-b">
          <td className="p-3">{t.student.name} <span className="text-gray-400">{t.student.email}</span></td>
          <td className="p-3">{t.student.university || "-"}</td>
          <td className="p-3 text-center">{t.avgRequiredSkillScore}</td>
          <td className="p-3 text-center font-semibold">{t.fitScore}</td>
        </tr>
      )))
    </tbody>
  </table>
</div>
)}

</SectionCard>
}

/* JOBS */

{activeTab === "jobs" && (
<SectionCard
  title="Job & Internship Postings"

```

```

icon=<Briefcase className="text-blue-600" />

>

/* Create form */

<form onSubmit={createJob} className="grid md:grid-cols-2 gap-4 mb-6">

  <input name="title" required placeholder="Title" className="border rounded-xl px-3 py-2 bg-white/70" />

  <select name="type" className="border rounded-xl px-3 py-2 bg-white/70">

    <option value="job">Job</option>

    <option value="internship">Internship</option>

  </select>

  <input name="location" placeholder="Location (Remote / City)" className="border rounded-xl px-3 py-2 bg-white/70" />

  <input name="minScore" type="number" min="0" max="10" step="0.1" placeholder="Min test score (0-10)" className="border rounded-xl px-3 py-2 bg-white/70" />

  <textarea name="description" placeholder="Role description" className="md:col-span-2 border rounded-xl px-3 py-2 bg-white/70" />

  <div className="md:col-span-2">

    <p className="text-xs text-gray-500 mb-1">Required skills</p>

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

      {skills.map((sk) => (
        <label key={sk._id} className="text-sm inline-flex items-center gap-2 px-3 py-1 rounded-full border bg-white hover:bg-slate-50">

          <input type="checkbox" name="skills" value={sk._id} />

          {sk.name}

        </label>
      )))
    </div>
  </div>

  <label className="flex items-center gap-2 text-sm">

    <input type="checkbox" name="isFeatured" /> Featured listing

  </label>

  /* Use your existing BtnPrimary; if you don't have it, replace with a normal button */

  <BtnPrimary className="md:justify-self-start">Post</BtnPrimary>

```

```

        </form>

/* Mobile cards */

<div className="md:hidden space-y-3">
{jobs.map((j) => {
  const typeTone =
    j.type === "internship"
      ? "bg-emerald-50 text-emerald-700 border-emerald-200"
      : "bg-blue-50 text-blue-700 border-blue-200";
  const statusTone =
    j.status === "open"
      ? "bg-sky-50 text-sky-700 border-sky-200"
      : j.status === "paused"
        ? "bg-amber-50 text-amber-700 border-amber-200"
        : "bg-rose-50 text-rose-700 border-rose-200";
  const minPct = Math.min(100, Math.max(0, (Number(j.minScore || 0) / 10) * 100));
  return (
    <div key={j._id} className="rounded-2xl border border-slate-200 bg-white/70 p-4">
      <div className="flex items-start justify-between gap-3">
        <div>
          <div className="flex items-center gap-2 flex-wrap">
            <h4 className="font-semibold text-gray-900">{j.title}</h4>
            {j.isFeatured ? (
              <span className="text-[11px] px-2 py-0.5 rounded-full border bg-fuchsia-50 text-fuchsia-700 border-fuchsia-200">
                Featured
              </span>
            ) : null}
        </div>
        <div className="mt-1 text-xs text-gray-500">{j.location || "—"}</div>
      </div>
    </div>
  )
})

```

```
</div>

<div className="flex flex-col items-end gap-1">
  <span className={`text-xs px-2.5 py-1 rounded-full border ${typeTone}`}>{j.type}</span>
  <span className={`text-xs px-2.5 py-1 rounded-full border ${statusTone}`}>{j.status}</span>
</div>
</div>

<p className="mt-3 text-sm text-gray-700 line-clamp-3">{j.description}</p>

<div className="mt-3">
  <div className="flex items-center justify-between text-xs mb-1">
    <span className="text-gray-600">Min Test Score</span>
    <b className="text-gray-800">{j.minScore ?? 0}/10</b>
  </div>
  <div className="h-2 w-full rounded-full bg-slate-100 overflow-hidden">
    <div className="h-full rounded-full bg-indigo-400" style={{ width: `${minPct}%` }} />
  </div>
</div>

<div className="mt-3 flex flex-wrap gap-2">
  {({skills || []}).map((s, i) => (
    <span
      key={`${j._id}-${i}`}
      className="text-xs px-2.5 py-1 rounded-full border bg-slate-50 text-slate-700 border-slate-200">
      >
        {typeof s === "string" ? s : s.name}
      </span>
    )))
</div>
```

```

<div className="mt-4 flex flex-wrap gap-2">

  <button onClick={() => openEditJob(j)} className="px-3 py-1.5 rounded-xl border bg-white hover:bg-slate-50 text-sm text-blue-700">

    <span className="inline-flex items-center gap-1"><Pencil size={14}/> Edit</span>

  </button>

  <button

    onClick={async () => { await API.patch(`/company/jobs/${j._id}/toggle`); fetchAll(); }}

    className="px-3 py-1.5 rounded-xl border bg-white hover:bg-slate-50 text-sm text-indigo-700"

  >

    {j.status === "open" ? "Close" : "Open"}

  </button>

  <button onClick={() => deleteJob(j._id)} className="px-3 py-1.5 rounded-xl bg-rose-600 text-white text-sm hover:bg-rose-700">

    <span className="inline-flex items-center gap-1"><Trash2 size={14}/> Delete</span>

  </button>

</div>

</div>

);

)}

</div>

/* Desktop table */

<div className="hidden md:block overflow-x-auto">

<table className="w-full text-sm overflow-hidden rounded-2xl border border-slate-200">

<thead className="bg-slate-50/80">

<tr>

  <th className="p-3 text-left">Title</th>

  <th className="p-3">Type</th>

  <th className="p-3">Status</th>

  <th className="p-3">Skills</th>

  <th className="p-3">MinScore</th>


```

```

<th className="p-3">Actions</th>
</tr>
</thead>
<tbody>
{jobs.map((j) => {
  const typeChip =
    j.type === "internship"
      ? "bg-emerald-50 text-emerald-700 border-emerald-200"
      : "bg-blue-50 text-blue-700 border-blue-200";
  const statusChip =
    j.status === "open"
      ? "bg-sky-50 text-sky-700 border-sky-200"
      : j.status === "paused"
        ? "bg-amber-50 text-amber-700 border-amber-200"
        : "bg-rose-50 text-rose-700 border-rose-200";
  const minPct = Math.min(100, Math.max(0, (Number(j.minScore || 0) / 10) * 100));
  return (
    <tr key={j._id} className="border-b">
      <td className="p-3">
        <div className="flex items-center gap-2 flex-wrap">
          <span className="font-medium text-gray-900">{j.title}</span>
          {j.isFeatured ? (
            <span className="text-[11px] px-2 py-0.5 rounded-full border bg-fuchsia-50 text-fuchsia-700 border-fuchsia-200">
              Featured
            </span>
          ) : null}
          <div className="text-xs text-gray-500">• {j.location || "--"}</div>
        </div>
      </td>
    </tr>
  );
});

```

```
<td className="p-3 text-center">
    <span className={`inline-flex px-2.5 py-1 rounded-full border text-xs
${typeChip}`}>{j.type}</span>
</td>

<td className="p-3 text-center">
    <span className={`inline-flex px-2.5 py-1 rounded-full border text-xs
${statusChip}`}>{j.status}</span>
</td>

<td className="p-3">
    <div className="flex flex-wrap gap-2">
        {((j.skills || []).map((s, i) => (
            <span
                key={`${j._id}-${i}`}
                className="text-xs px-2.5 py-1 rounded-full border bg-slate-50 text-slate-700 border-slate-200"
            >
                {typeof s === "string" ? s : s.name}
            </span>
        )))
    </div>
</td>

<td className="p-3">
    <div className="min-w-[140px] mx-auto">
        <div className="flex items-center justify-between text-xs mb-1">
            <span className="text-gray-600">Min</span>
            <b className="text-gray-800">{j.minScore ?? 0}/10</b>
        </div>
        <div className="h-2 w-full rounded-full bg-slate-100 overflow-hidden">
            <div className="h-full rounded-full bg-indigo-400" style={{ width: `${minPct}%` }} />
        </div>
    </div>
</td>
```

```

<td className="p-3">
  <div className="flex items-center gap-3 justify-center">
    <button
      onClick={() => openEditJob(j)}
      className="text-blue-700 hover:underline flex items-center gap-1"
      title="Edit job"
    >
      <Pencil size={16} /> Edit
    </button>
    <button
      onClick={async () => { await API.patch(`/company/jobs/${j._id}/toggle`); fetchAll(); }}
      className="text-indigo-700 hover:underline"
    >
      {j.status === "open" ? "Close" : "Open"}
    </button>
    <button
      onClick={() => deleteJob(j._id)}
      className="text-rose-700 hover:underline flex items-center gap-1"
      title="Delete job"
    >
      <Trash2 size={16} /> Delete
    </button>
  </div>
</td>
</tr>
);
}};

</tbody>
</table>
</div>

```

```
/* Edit Job Modal (unchanged API/logic, pastel styling) */

{editingJobState && (
  <div className="fixed inset-0 bg-black/40 z-50 flex items-center justify-center p-4">
    <div className="bg-white/90 backdrop-blur w-full max-w-2xl rounded-2xl shadow-xl p-6 border border-slate-200">
      <div className="flex items-center justify-between mb-4">
        <h3 className="text-lg font-semibold">Edit Job</h3>
        <button onClick={closeEditJob} className="text-gray-500 hover:text-gray-700">
          <X size={18} />
        </button>
      </div>

      <form onSubmit={saveJobUpdates} className="grid md:grid-cols-2 gap-4">
        <input
          value={editingJobState.title}
          onChange={(e) => setEditingJobState((j) => ({ ...j, title: e.target.value }))}
          required
          placeholder="Title"
          className="border rounded-md px-3 py-2 bg-white/70"
        />
        <select
          value={editingJobState.type}
          onChange={(e) => setEditingJobState((j) => ({ ...j, type: e.target.value }))}
          className="border rounded-md px-3 py-2 bg-white/70"
        >
          <option value="job">Job</option>
          <option value="internship">Internship</option>
        </select>
        <input
          value={editingJobState.location}
          onChange={(e) => setEditingJobState((j) => ({ ...j, location: e.target.value }))}
        </div>
    
```

```

placeholder="Location (Remote / City)"
className="border rounded-xl px-3 py-2 bg-white/70"
/>
<input
  type="number"
  min="0"
  max="10"
  step="0.1"
  value={editingJobState.minScore}
  onChange={(e) => setEditingJobState((j) => ({ ...j, minScore: e.target.value }))}
  placeholder="Min test score (0-10)"
  className="border rounded-xl px-3 py-2 bg-white/70"
/>
<div className="md:col-span-2">
  <textarea
    value={editingJobState.description}
    onChange={(e) => setEditingJobState((j) => ({ ...j, description: e.target.value }))}
    placeholder="Role description"
    className="w-full border rounded-xl px-3 py-2 bg-white/70"
  />
</div>
<div className="md:col-span-2">
  <p className="text-xs text-gray-500 mb-1">Required skills</p>
  <div className="flex flex-wrap gap-3">
    {skills.map((sk) => {
      const checked = editingJobState.skills.includes(sk._id);
      return (
        <label key={sk._id} className="text-sm inline-flex items-center gap-2 px-3 py-1 rounded-full border bg-white hover:bg-slate-50">
          <input type="checkbox" checked={checked} onChange={() => toggleEditSkill(sk._id)} />
          {sk.name}
        </label>
      )
    })
  </div>
</div>

```

```

        </label>
    );
  }})
</div>
</div>

<label className="flex items-center gap-2 text-sm">
  <input
    type="checkbox"
    checked={editingJobState.isFeatured}
    onChange={(e) => setEditingJobState((j) => ({ ...j, isFeatured: e.target.checked }))}
  />
  Featured listing
</label>

<select
  value={editingJobState.status}
  onChange={(e) => setEditingJobState((j) => ({ ...j, status: e.target.value }))}
  className="border rounded-xl px-3 py-2 bg-white/70"
>
  <option value="open">Open</option>
  <option value="closed">Closed</option>
  <option value="paused">Paused</option>
</select>

<div className="md:col-span-2 flex items-center gap-3 pt-2">
  /* Use your existing buttons; if not available, swap BtnGhost/BtnPrimary with standard
  buttons */
  <BtnGhost type="button" onClick={closeEditJob}>Cancel</BtnGhost>
  <BtnPrimary type="submit">Save Changes</BtnPrimary>
</div>
</form>
</div>
</div>

```

```

        )}
      </SectionCard>
    )}

/* HACKATHONS */
{activeTab === "hackathons" && (
  <SectionCard title="Live Hackathons & Talent Challenges" icon={<Trophy className="text-blue-600" />}>
    <form
      onSubmit={async (e) => {
        e.preventDefault();
        const f = new FormData(e.currentTarget);
        try {
          await API.post("/company/hackathons", {
            title: f.get("title"),
            prize: f.get("prize"),
            brief: f.get("brief"),
            rules: f.get("rules"),
            resources: f.get("resources"),
            skills: f.getAll("skills"),
            startAt: f.get("startAt"),
            endAt: f.get("endAt"),
            visibility: f.get("visibility") || "public",
            bannerUrl: f.get("bannerUrl"),
          });
          e.currentTarget.reset();
          fetchAll();
        } catch (err) {
          alert(err?.response?.data?.message || "Create failed");
        }
      }}
    </SectionCard>
  )
}


```

```
        )}

        className="grid md:grid-cols-2 gap-4 mb-6"

      >

        <input name="title" required placeholder="Hackathon Title" className="border rounded-xl px-3 py-2 bg-white/70" />

        <input name="prize" placeholder="Prize / Awards" className="border rounded-xl px-3 py-2 bg-white/70" />

        <textarea name="brief" placeholder="Brief / Problem Statement" className="md:col-span-2 border rounded-xl px-3 py-2 bg-white/70" />

        <textarea name="rules" placeholder="Rules / Evaluation Criteria" className="md:col-span-2 border rounded-xl px-3 py-2 bg-white/70" />

        <textarea name="resources" placeholder="Resources / Datasets / Links" className="md:col-span-2 border rounded-xl px-3 py-2 bg-white/70" />

      <div className="md:col-span-2">

        <p className="text-xs text-gray-500 mb-1">Target skills</p>

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

          {skills.map((sk) => (
            <label key={sk._id} className="text-sm flex items-center gap-2">
              <input type="checkbox" name="skills" value={sk._id} />
              {sk.name}
            </label>
          )))
        </div>
      </div>

      <input type="datetime-local" name="startAt" className="border rounded-xl px-3 py-2 bg-white/70" />

      <input type="datetime-local" name="endAt" className="border rounded-xl px-3 py-2 bg-white/70" />

      <select name="visibility" className="border rounded-xl px-3 py-2 bg-white/70">
        <option value="public">Public</option>
        <option value="private">Private</option>
      </select>

      <input name="bannerUrl" placeholder="Banner image URL (optional)" className="border rounded-xl px-3 py-2 bg-white/70" />
```

```

<BtnPrimary className="md:justify-self-start">Create</BtnPrimary>
</form>

{hackathons.length === 0 ? (
  <p className="text-sm text-gray-500">No hackathons yet.</p>
) : (
  <ul className="space-y-5">
    {hackathons.map((h) => (
      <li key={h._id} className="border rounded-2xl p-5 bg-white/70 hover:shadow-sm transition">
        <div className="flex flex-col gap-3 md:flex-row md:items-start md:justify-between">
          <div className="flex gap-3">
            {h.bannerUrl ? (
              <img src={h.bannerUrl} alt="" className="h-16 w-24 object-cover rounded-xl border border-slate-200" />
            ) : (
              <div className="h-16 w-24 rounded-xl bg-gray-100 border" />
            )}
          <div>
            <div className="flex items-center gap-2">
              <div className="font-semibold">{h.title}</div>
              <span className="text-xs px-2 py-0.5 rounded-full border bg-slate-50">{h.status}</span>
            </div>
            <div className="text-xs text-gray-500 mt-0.5">
              {new Date(h.startAt).toLocaleString()} → {new Date(h.endAt).toLocaleString()} •
            {h.visibility}
            </div>
            {h.prize && <div className="text-xs mt-1">Prize: <b>{h.prize}</b></div>}
            <div className="text-xs mt-1 text-gray-600">
              {(h.skills || []).map((s) => (typeof s === "string" ? s : s.name)).join(", ")}
            </div>
          </div>
        </div>
      </li>
    ))}
  </ul>
)

```

```
</div>

</div>

<div className="flex flex-wrap gap-2">
  <HackStatusControl
    value={h.status}
    onChange={async (next) => { await
      API.patch(`company/hackathons/${h._id}/status`, { status: next }); fetchAll(); }}
  />
  <BtnGhost
    onClick={async () => {
      const shareUrl = `${window.location.origin}/student/hackathons/${h._id}`;
      try { await navigator.clipboard.writeText(shareUrl); alert("Share link copied!"); }
      catch { window.prompt("Copy link", shareUrl); }
    }}
    title="Copy public link"
    className="px-3 py-1.5 text-sm"
  >
    Share
  </BtnGhost>
  <EditHackathonButton hack={h} onSaved={fetchAll} />
</div>
</div>

<div className="grid md:grid-cols-2 gap-4 mt-4">
  <Card className="p-4">
    <div className="font-medium mb-2">Registrations</div>
    <HackathonRegistrations hackathonId={h._id} />
  </Card>
<Card className="p-4">
```

```

        <div className="font-medium mb-2">Submissions & Judging</div>
        <HackathonSubmissions hackathonId={h._id} />
        <div className="mt-3">
          <BtnPrimary
            onClick={async () => {
              await API.post(`/company/hackathons/${h._id}/publish-leaderboard`);
              alert("Leaderboard published");
            }}
          >
            Publish Leaderboard
          </BtnPrimary>
        </div>
      </Card>
    </div>
  </li>
))}

</ul>
)
</SectionCard>
)

/* WEBINARS */
{activeTab === "webinars" && (
  <SectionCard title="Employer Branding & Webinars" icon={<Video className="text-blue-600" />}>
    <form onSubmit={createWebinar} className="grid md:grid-cols-2 gap-4 mb-6">
      <input name="title" required placeholder="Title" className="border rounded-xl px-3 py-2 bg-white/70" />
      <input name="speaker" placeholder="Speaker" className="border rounded-xl px-3 py-2 bg-white/70" />
      <input type="datetime-local" name="startsAt" className="border rounded-xl px-3 py-2 bg-white/70" />
    </form>
  </SectionCard>
)
}

```

```

        <input type="number" name="durationMins" placeholder="Duration (mins)" className="border rounded-xl px-3 py-2 bg-white/70" />

        <textarea name="description" placeholder="Description" className="md:col-span-2 border rounded-xl px-3 py-2 bg-white/70" />

        <BtnPrimary disabled={creatingWebinar} className="md:justify-self-start">
            {creatingWebinar ? "Creating..." : "Schedule"}
        </BtnPrimary>
    </form>

    {webinars.length === 0 ? (
        <p className="text-sm text-gray-500">No webinars yet.</p>
    ) : (
        <ul className="space-y-3">
            {webinars.map((w) => (
                <li key={w._id} className="border rounded-xl p-3 bg-white/70">
                    <div className="flex items-center justify-between">
                        <div>
                            <p className="font-medium">{w.title}</p>
                            <p className="text-xs text-gray-500">
                                {new Date(w.startsAt).toLocaleString()} • {w.durationMins} mins
                            </p>
                        </div>
                    <div className="flex gap-3">
                        <Link to={`/student/webinars/webinar/${w.roomId}`} className="text-blue-700 text-sm underline">
                            Join (Viewer)
                        </Link>
                        <Link to={`/company/webinar/${w.roomId}`} className="text-indigo-700 text-sm underline">
                            Open Studio
                        </Link>
                    </div>
                </li>
            ))
        </ul>
    )
}

```

```

        </div>
    </li>
  )})
</ul>
)}

</SectionCard>
)}

/* PARTNERSHIPS */

{activeTab === "partnerships" && (
  <SectionCard title="University Partnerships & Research" icon={<Handshake className="text-blue-600" />}>
    <form
      onSubmit={async (e) => {
        e.preventDefault();
        const f = new FormData(e.currentTarget);
        await API.post("/company/partnerships", {
          university: f.get("university"),
          title: f.get("title"),
          details: f.get("details"),
          status: f.get("status"),
        });
        e.currentTarget.reset();
        fetchAll();
      }}
      className="grid md:grid-cols-2 gap-4 mb-6"
    >
      <input name="university" required placeholder="University name" className="border rounded-xl px-3 py-2 bg-white/70" />
      <select name="status" className="border rounded-xl px-3 py-2 bg-white/70">
        <option value="proposal">Proposal</option>
        <option value="active">Active</option>

```

```

        <option value="completed">Completed</option>
    </select>
    <input name="title" required placeholder="Initiative title" className="border rounded-xl px-3 py-2 bg-white/70 md:col-span-2" />
    <textarea name="details" placeholder="Details" className="border rounded-xl px-3 py-2 bg-white/70 md:col-span-2" />
    <BtnPrimary className="md:justify-self-start">Save</BtnPrimary>
</form>

{partnerships.length === 0 ? (
    <p className="text-sm text-gray-500">No collaborations yet.</p>
) : (
    <ul className="space-y-3">
        {partnerships.map((p) => (
            <li key={p._id} className="border rounded-xl p-3 bg-white/70">
                <div className="flex items-center justify-between">
                    <div>
                        <p className="font-medium">{p.title}</p>
                        <p className="text-xs text-gray-500">{p.university} • {p.status}</p>
                        <p className="text-xs mt-1">{p.details}</p>
                    </div>
                    <Building2 className="text-gray-400" />
                </div>
            </li>
        )))
    </ul>
)
}
</SectionCard>
)}
</>
)}
</div>

```

```
<ScheduleModal
  open={scheduleOpen}
  onClose={() => setScheduleOpen(false)}
  mode={scheduleMode}
  application={scheduleApp}
  onSaved={fetchAll}
/>
</div>
);
}

import { useEffect, useRef, useState } from "react";
import { useParams, Link } from "react-router-dom";
import io from "socket.io-client";

const socket = io(import.meta.env.VITE_API_BASE?.replace("/api","") || "http://localhost:5000", {
  transports: ["websocket"]
});

export default function CompanyInterviewRoom() {
  const { roomId } = useParams();
  const [connected, setConnected] = useState(false);
  const videoMine = useRef(null);
  const videoPeer = useRef(null);
  const pcRef = useRef(null);
  const localStreamRef = useRef(null);

  useEffect(() => {
    socket.on("connect", () => setConnected(true));
    socket.on("disconnect", () => setConnected(false));
    return () => { socket.off("connect"); socket.off("disconnect"); };
  }, []);
}
```

```

useEffect(() => {
  if (!roomId) return;

  (async () => {
    localStreamRef.current = await navigator.mediaDevices.getUserMedia({ video: true, audio: true });
    if (videoMine.current) videoMine.current.srcObject = localStreamRef.current;

    pcRef.current = new RTCPeerConnection();
    localStreamRef.current.getTracks().forEach(t => pcRef.current.addTrack(t, localStreamRef.current));
    pcRef.current.ontrack = (e) => { if (videoPeer.current) videoPeer.current.srcObject = e.streams[0]; };
  });

  socket.emit("createWebinar", { roomId, webinarName: "Interview", hostName: "Interviewer" });

  socket.on("signal-offer", async ({ sdp }) => {
    await pcRef.current.setRemoteDescription(new RTCSessionDescription(sdp));
    const answer = await pcRef.current.createAnswer();
    await pcRef.current.setLocalDescription(answer);
    socket.emit("signal-answer", { target: socket.id /* server maps host->student */ , sdp: answer });
  });

  socket.on("signal-ice", async ({ candidate }) => {
    if (candidate) await pcRef.current.addIceCandidate(new RTCIceCandidate(candidate));
  });

  pcRef.current.onicecandidate = (e) => {
    if (e.candidate) socket.emit("signal-ice", { target: "student", candidate: e.candidate });
  };
})();

```

```

    return () => {
      socket.off("signal-offer");
      socket.off("signal-ice");
      pcRef.current?.close();
      localStreamRef.current?.getTracks().forEach(t => t.stop());
    };
  }, [roomId]);
}

return (
  <div className="min-h-screen bg-gray-50 p-4">
    <div className="max-w-5xl mx-auto">
      <div className="flex items-center justify-between mb-3">
        <Link to="/company/dashboard" className="text-blue-600 underline">Back</Link>
        <span className={`text-xs px-2 py-1 rounded ${connected ? "bg-green-100 text-green-700" : "bg-gray-200 text-gray-600"}`}>{connected?"Connected":"Disconnected"}</span>
      </div>
      <div className="grid md:grid-cols-2 gap-3">
        <video ref={videoPeer} autoPlay playsInline className="w-full bg-black rounded-xl aspect-video" />
        <video ref={videoMine} autoPlay playsInline muted className="w-full bg-black rounded-xl aspect-video" />
      </div>
    </div>
  </div>
);
}

```

```

import { useEffect, useState } from "react";
import { useParams, Link } from "react-router-dom";
import API from "../../api/axios";
import { ArrowLeft, Megaphone, Scale, Users2, Trophy, Save, Check } from "lucide-react";

```

```

const Field = (p) => <input {...p} className={"border rounded-lg px-3 py-2 " + (p.className || "")} />

export default function HackathonManager() {
  const { id } = useParams();
  const [h, setH] = useState(null);
  const [subs, setSubs] = useState([]);
  const [announceMsg, setAnnounceMsg] = useState("");
  const [saving, setSaving] = useState(false);

  const load = async () => {
    const [d, s] = await Promise.all([
      API.get(`/company/hackathons/${id}`) // we'll reuse /company/hackathons/:id via PUT route
      (GET fallback below)
      .catch(async () => ({ data: (await API.get("/company/hackathons")).data.find(x=>x._id==id) })),
      API.get(`/company/hackathons/${id}/submissions`)
    ]);
    setH(d.data);
    setSubs(s.data || []);
  };

  useEffect(()=>{ load(); }, [id]);
}

if (!h) return <div className="p-6">Loading...</div>

const update = async (patch) => {
  setSaving(true);
  try {
    const res = await API.put(`/company/hackathons/${id}`, patch);
    setH(res.data);
  } catch (e) {

```

```

        alert(e?.response?.data?.message || "Save failed");

    } finally { setSaving(false); }

};

const announce = async () => {
    if (!announceMsg.trim()) return;

    await API.post(`/company/hackathons/${id}/announce`, { message: announceMsg });
    setAnnounceMsg("");
    load();
};

const score = async (subId, rubricKey, val) => {
    const sub = subs.find(s => s._id === subId);
    const items = (h.rubric || []).map(r => {
        if (r.key === rubricKey) return { rubricKey: r.key, score: Number(val) };
        const me = (sub.judgeScores || []).find(js => js.judge === "me"); // UI only; backend derives judge from token
        const prev = me?.items?.find(i => i.rubricKey === r.key)?.score || 0;
        return { rubricKey: r.key, score: prev };
    });
    const res = await API.post(`/company/hackathons/${id}/submissions/${subId}/score`, { items });
    setSubs(s => s.map(x => x._id === subId ? res.data : x));
};

const finalize = async () => {
    if (!window.confirm("Finalize results? This will set ranks and mark finished.")) return;
    await API.post(`/company/hackathons/${id}/finalize`);
    load();
};

return (

```

```
<div className="max-w-6xl mx-auto p-4 space-y-6">
  <div className="flex items-center justify-between">
    <Link to="/company" className="inline-flex items-center gap-2 text-sm text-gray-600">
      <ArrowLeft size={16}/> Back
    </Link>
    <div className="text-xs px-2 py-1 rounded-full border">{h.phase}</div>
  </div>

  <div className="bg-white border rounded-2xl p-6">
    <div className="flex items-center justify-between">
      <div>
        <h1 className="text-xl font-semibold">{h.title}</h1>
        <p className="text-sm text-gray-500">{new Date(h.startAt).toLocaleString()} → {new Date(h.endAt).toLocaleString()}</p>
      </div>
      <button onClick={finalize} className="px-3 py-2 rounded-lg bg-emerald-600 text-white flex items-center gap-2">
        <Trophy size={16}/> Finalize
      </button>
    </div>
  </div>

  <div className="grid md:grid-cols-2 gap-6 mt-6">
    <div>
      <h3 className="font-medium mb-2">Settings</h3>
      <div className="space-y-2">
        <Field defaultValue={h.prize} placeholder="Prize"
          onBlur={(e)=>update({ prize: e.target.value })}/>
        <Field defaultValue={h.bannerUrl} placeholder="Banner URL"
          onBlur={(e)=>update({ bannerUrl: e.target.value })}/>
        <select defaultValue={h.phase} onChange={(e)=>update({ phase: e.target.value })}>
          {["draft","upcoming","open","judging","finished"].map(p => <option key={p}>{p}</option>)}
        </select>
      </div>
    </div>
  </div>
</div>
```

```

        </select>
    </div>
    {saving && <div className="text-xs text-gray-400 mt-2 flex items-center gap-1"><Save
size={14}/> Saving...</div>}
</div>

<div>
    <h3 className="font-medium mb-2 flex items-center gap-2"><Megaphone size={16}/>
Announcements</h3>
    <div className="flex gap-2">
        <Field value={announceMsg} onChange={(e)=>setAnnounceMsg(e.target.value)}
placeholder="Share schedule changes, links, etc." className="flex-1"/>
        <button onClick={announce} className="px-3 py-2 rounded-lg border">Post</button>
    </div>
    <ul className="mt-3 space-y-2 text-sm">
        {(h.announcements| []).slice().reverse().map((a,i)=>(
            <li key={i} className="border rounded-lg p-2">{a.message} <span className="text-gray-
400 text-xs">• {new Date(a.createdAt).toLocaleString()}</span></li>
        ))}
    </ul>
    </div>
</div>

<div className="mt-8">
    <h3 className="font-medium mb-2 flex items-center gap-2"><Scale size={16}/> Rubric</h3>
    <div className="grid md:grid-cols-2 gap-3">
        {(h.rubric| []).map((r,i)=>(
            <div key={i} className="border rounded-lg p-3 text-sm flex items-center justify-between">
                <div>{r.label}</div>
                <div className="text-gray-500">/{r.max}</div>
            </div>
        )))
    </div>

```

```
</div>
</div>

<div className="mt-8">
  <h3 className="font-medium mb-3 flex items-center gap-2"><Users2 size={16}/>
  Submissions</h3>
  {subs.length === 0 ? (
    <p className="text-sm text-gray-500">No submissions yet.</p>
  ) : (
    <div className="overflow-x-auto">
      <table className="w-full text-sm">
        <thead className="bg-gray-50">
          <tr>
            <th className="p-2 text-left">Team</th>
            <th className="p-2">Title</th>
            <th className="p-2">Repo</th>
            <th className="p-2">Demo</th>
            <th className="p-2">Final</th>
            {{(h.rubric || []).map(r => <th key={r.key} className="p-2">{r.label}</th>)}}
          </tr>
        </thead>
        <tbody>
          {subs.map(s => (
            <tr key={s._id} className="border-b">
              <td className="p-2">{s.team?.name}</td>
              <td className="p-2 text-center">{s.title}</td>
              <td className="p-2 text-center">{s.repoUrl ? <a className="text-blue-600 underline" href={s.repoUrl} target="_blank" rel="noreferrer">Repo</a> : "-"}</td>
              <td className="p-2 text-center">{s.demoUrl ? <a className="text-indigo-600 underline" href={s.demoUrl} target="_blank" rel="noreferrer">Demo</a> : "-"}</td>
              <td className="p-2 text-center font-semibold">{s.finalScore ?? "-"}</td>
            {{(h.rubric || []).map(r => {

```

```

        const max = r.max || 10;

        return (
            <td key={r.key} className="p-2">
                <input type="number" min="0" max={max}
                    onBlur={(e)=>score(s._id, r.key, e.target.value)}
                    className="w-20 border rounded px-2 py-1" placeholder={`0-${max}`}>
            </td>
        );
    )}

    </tr>
)})

</tbody>
</table>
</div>
)
</div>
</div>
</div>
);

}

```

---

```

import React, { useEffect, useRef, useState } from "react";
import io from "socket.io-client";
import Peer from "simple-peer";

const socket = io("http://localhost:5000");

function WebinarRoom({ roomId, username }) {
    const [peers, setPeers] = useState([]);
    const userVideo = useRef();
    const peersRef = useRef([]);

    useEffect(() => {
        const peer = new Peer({
            initiator: true,
            stream: userVideo.current
        });

        peer.on("open", () => {
            socket.emit("join", { roomId, username });
            socket.on("offer", (offer) => {
                peer.setOffer(offer);
                peer.on("answer", (answer) => {
                    socket.emit("answer", answer);
                });
            });
        });
    }, []);
}
```

```
useEffect(() => {

  navigator.mediaDevices.getUserMedia({ video: true, audio: true }).then(stream => {
    userVideo.current.srcObject = stream;
    socket.emit("joinWebinar", { roomId, username });

    socket.on("userJoined", ({ username }) => {
      console.log(` ${username} joined`);
    });

    socket.on("signal", ({ id, data }) => {
      const peer = peersRef.current.find(p => p.peerID === id);
      if (peer) {
        peer.peer.signal(data);
      }
    });
  });

  return (
    <div className="webinar-container">
      <video ref={userVideo} autoPlay playsInline muted className="video-player" />
      {peers.map((peer, index) => (
        <Video key={index} peer={peer.peer} />
      ))}
    </div>
  );
}

function Video({ peer }) {
  const ref = useRef();
}
```

```

useEffect(() => {
  peer.on("stream", stream => {
    ref.current.srcObject = stream;
  });
}, [peer]);

return <video ref={ref} autoPlay playsInline className="video-player" />;
}

export default WebinarRoom;

```

---

```

import { useEffect, useMemo, useRef, useState } from "react";
import { Link, useParams } from "react-router-dom";
import io from "socket.io-client";
import {
  ShieldCheck,
  ArrowLeft,
  Users,
  Info,
  MessageCircle,
  MicOff,
  VideoOff,
  ScreenShare,
  PhoneOff,
  Maximize2,
  Minimize2,
  Check,
  X as XIcon,
  UserPlus,
  UserX
} from "lucide-react";
import useLocalMedia from "../../hooks/useLocalMedia";

const socket = io(import.meta.env.VITE_API_BASE || "http://localhost:5000", { transports: ["websocket"] });
const PANEL_W = 380;

const Pill = ({ active, onClick, title, children }) => (
  <button type="button" title={title} onClick={onClick}>
    <span>{`px-3 h-9 rounded-full border text-sm flex items-center gap-2 transition`}</span>
    ${active ? "bg-blue-600 text-white border-blue-600" : "bg-white/90 hover:bg-white border-black/10 text-gray-800"}>
    {children}
  </button>
)

```

```

        </button>
    );

const ChatPanel = ({ messages, onSend }) => {
    const inputRef = useRef(null);
    return (
        <div className="h-full flex flex-col">
            <div className="px-4 h-12 border-b border-black/10 flex items-center font-semibold">In-call
            messages</div>
            <div className="flex-1 overflow-y-auto p-4 space-y-2 text-sm">
                {messages.length === 0 ? <p className="text-gray-500">No messages yet.</p> :
                    messages.map((m, i) => (<div key={i}>{m.system ? <span className="italic text-gray-
                    500">{m.message}</span> : (<><b className="text-purple-700">{m.username}:
                    </b><span>{m.message}</span></>)}</div>))}
            </div>
            <div className="p-3 border-t border-black/10 flex gap-2">
                <input ref={inputRef} className="flex-1 border rounded-md px-3 h-10" placeholder="Send a
                message to everyone"
                    onKeyDown={(e)=>{ if(e.key==="Enter"){ const v=inputRef.current?.value?.trim(); if(v){
                    onSend(v); inputRef.current.value=""; }}}}/>
                <button onClick={()=>{ const v=inputRef.current?.value?.trim(); if(v){ onSend(v);
                    inputRef.current.value=""; }}}> class="h-10 px-4 rounded-md bg-blue-600 text-
                    white">Send</button>
            </div>
        </div>
    );
};

const HostInfoPanel = () => (
    <div className="h-full flex flex-col">
        <div className="px-4 h-12 border-b border-black/10 flex items-center font-semibold">Info</div>
        <div className="p-4 text-sm space-y-3">
            <p>Admit or deny attendees from the People tab.</p>

```

```

<div>
  <p className="font-medium mb-1">Shortcuts</p>
  <ul className="list-disc ml-5 space-y-1">
    <li><kbd className="px-1 py-0.5 bg-gray-100 rounded">G</kbd> — Open People</li>
    <li><kbd className="px-1 py-0.5 bg-gray-100 rounded">C</kbd> — Chat</li>
    <li><kbd className="px-1 py-0.5 bg-gray-100 rounded">I</kbd> — Info</li>
    <li><kbd className="px-1 py-0.5 bg-gray-100 rounded">M</kbd> — Mute</li>
    <li><kbd className="px-1 py-0.5 bg-gray-100 rounded">V</kbd> — Camera</li>
    <li><kbd className="px-1 py-0.5 bg-gray-100 rounded">P</kbd> — Present</li>
    <li><kbd className="px-1 py-0.5 bg-gray-100 rounded">F</kbd> — Fullscreen</li>
  </ul>
</div>
</div>
</div>
);


```

```

const PeoplePanel = ({ waiting, onToggle, selected, onAdmit, onDeny, onAdmitSelected,
onDenySelected, participants }) => (
  <div className="h-full flex flex-col">
    <div className="px-4 h-12 border-b border-black/10 flex items-center justify-between">
      <span className="font-semibold">People</span>
      <div className="flex items-center gap-2">
        <button onClick={onAdmitSelected} className="px-2 h-8 rounded-md bg-emerald-600 text-white text-xs flex items-center gap-1">
          <Check size={14}/> Admit selected
        </button>
        <button onClick={onDenySelected} className="px-2 h-8 rounded-md bg-red-600 text-white text-xs flex items-center gap-1">
          <XIcon size={14}/> Deny selected
        </button>
      </div>
    </div>
  </div>
);


```

```

<div className="flex-1 overflow-y-auto text-sm">
  <div className="p-4 border-b border-black/10">
    <p className="font-medium mb-2">Waiting room</p>
    {waiting.length === 0 ? <p className="text-gray-500">No one waiting.</p> :
      waiting.map((u)=>
        <div key={u.id} className="flex items-center justify-between py-2">
          <label className="flex items-center gap-2">
            <input type="checkbox" checked={!selected[u.id]} onChange={()=>onToggle(u.id)}/>
            <span>{u.name}</span>
          </label>
          <div className="flex items-center gap-2">
            <button onClick={()=>onAdmit(u.id)} className="px-2 h-8 rounded-md bg-emerald-600 text-white text-xs flex items-center gap-1"><UserPlus size={14}/> Admit</button>
            <button onClick={()=>onDeny(u.id)} className="px-2 h-8 rounded-md bg-red-600 text-white text-xs flex items-center gap-1"><UserX size={14}/> Deny</button>
          </div>
        </div>
      )
    )
  </div>
  <div className="p-4">
    <p className="font-medium mb-2">In the meeting</p>
    {participants.length === 0 ? <p className="text-gray-500">Only you right now.</p> :
      participants.map((u)=>(<div key={u.id} className="py-1">{u.name}</div>))
    }
  </div>
</div>
);

export default function WebinarStudio() {
  const { roomId } = useParams();
  const stageRef = useRef(null);

```

```

const {
  videoRef, muted, cameraOff, presenting,
  toggleMute, toggleCamera, togglePresent, stopAll, ensureStream
} = useLocalMedia();

const [connected, setConnected] = useState(false);
const [panel, setPanel] = useState(null); // "people" | "chat" | "info" | null
const [messages, setMessages] = useState([]);
const [waiting, setWaiting] = useState([]);
const [selectedWait, setSelectedWait] = useState({});
const [participants, setParticipants] = useState([]);
const containerMinH = useMemo(() => "min-h-[600px]", []);

useEffect(() => {
  if (!roomId) return;
  socket.on("connect", () => setConnected(true));
  socket.on("disconnect", () => setConnected(false));
  socket.on("chatMessage", (payload) => setMessages((m) => [...m, payload]));
  socket.on("userJoined", ({ username }) => {
    setWaiting((w) => [...w, { id: crypto.randomUUID(), name: username }]);
  });
  return () => { socket.off("connect"); socket.off("disconnect"); socket.off("chatMessage");
  socket.off("userJoined"); };
}, [roomId]);

useEffect(() => { ensureStream().catch(()=>{}); }, [ensureStream]);

const sendChat = (txt) => socket.emit("chatMessage", { roomId, username: "Host", message: txt });
const requestFullscreen = () => {
  const el = stageRef.current;
  if (!el) return;

```

```

if (document.fullscreenElement) document.exitFullscreen?();
else el.requestFullscreen?();
};

const togglePanel = (next) => setPanel((p) => (p === next ? null : next));

// people controls

const toggleSelect = (id) => setSelectedWait((s) => ({ ...s, [id]: !s[id] }));
const admitOne = (id) => {
  setWaiting((w) => w.filter((x) => x.id !== id));
  const user = waiting.find((x) => x.id === id);
  if (user) setParticipants((p) => [...p, user]);
};
const denyOne = (id) => setWaiting((w) => w.filter((x) => x.id !== id));
const admitSelected = () => {
Object.keys(selectedWait).filter(k=>selectedWait[k]).forEach(admitOne); setSelectedWait({}); };
const denySelected = () => { Object.keys(selectedWait).filter(k=>selectedWait[k]).forEach(denyOne); setSelectedWait({}); };

// keyboard shortcuts

useEffect(() => {
  const h = (e) => {
    const k = e.key.toLowerCase();
    if (k === "g") setPanel(p => (p === "people" ? null : "people"));
    if (k === "c") setPanel(p => (p === "chat" ? null : "chat"));
    if (k === "i") setPanel(p => (p === "info" ? null : "info"));
    if (k === "m") toggleMute();
    if (k === "v") toggleCamera();
    if (k === "p") togglePresent();
    if (k === "f") requestFullscreen();
  };
  window.addEventListener("keydown", h);
  return () => window.removeEventListener("keydown", h);
});

```

```

    }, [toggleMute, toggleCamera, togglePresent]);

return (
  <div className="min-h-screen bg-gray-950 text-white">
    <div className="max-w-7xl mx-auto p-4 space-y-4">
      <div className="flex items-center justify-between">
        <Link to="/company/dashboard" className="inline-flex items-center gap-2 text-sm text-blue-400">
          <ArrowLeft size={16} /> Back to dashboard
        </Link>
        <div className="flex items-center gap-2">
          <span className="text-xs px-2 py-1 rounded bg-emerald-600/20 text-emerald-300">Live</span>
          <span className="text-xs px-2 py-1 rounded bg-slate-800">{roomId}</span>
        </div>
      </div>
    </div>

    <div className="rounded-3xl bg-gradient-to-b from-slate-800 to-slate-900 border border-white/10 overflow-hidden">
      <div ref={stageRef} className={`relative flex w-full ${containerMinH}`}>
        {/* STAGE */}
        <div className="relative flex-1 bg-slate-900">
          <video ref={videoRef} className="w-full h-full object-cover opacity-85" autoPlay playsInline muted />
        </div>

        <div className="absolute left-4 top-4 flex items-center gap-2">
          <ShieldCheck className="text-emerald-300" size={18} />
          <span className="text-xs text-white/80">Host Controls Enabled</span>
        </div>

        {/* controls */}
        <div className="absolute bottom-4 left-0 right-0 flex justify-center" style={{ paddingRight: panel ? PANEL_W : 0 }}>

```

```

<div className="backdrop-blur bg-black/40 border border-white/10 rounded-full px-3 py-2
flex items-center gap-2">

    <button onClick={toggleMute} className="w-9 h-9 grid place-items-center rounded-full
    hover:bg-white/10" title="Mute (M)">
        <MicOff size={18} className={muted ? "opacity-100" : "opacity-50"} />
    </button>

    <button onClick={toggleCamera} className="w-9 h-9 grid place-items-center rounded-full
    hover:bg-white/10" title="Camera (V)">
        <VideoOff size={18} className={cameraOff ? "opacity-100" : "opacity-50"} />
    </button>

    <button onClick={togglePresent} className="w-9 h-9 grid place-items-center rounded-full
    hover:bg-white/10" title="Present (P)">
        <ScreenShare size={18} className={presenting ? "opacity-100" : "opacity-50"} />
    </button>

    <button onClick={requestFullscreen} className="w-9 h-9 grid place-items-center rounded-full
    hover:bg-white/10" title="Fullscreen (F)">
        {document.fullscreenElement ? <Minimize2 size={18} /> : <Maximize2 size={18} />}
    </button>

    <button onClick={stopAll} className="w-9 h-9 grid place-items-center rounded-full bg-red-
    600 hover:bg-red-700 ml-1" title="End">
        <PhoneOff size={18} />
    </button>
</div>
</div>

/* dock */

<div className="absolute bottom-5 flex gap-2" style={{ right: (panel ? PANEL_W : 0) + 20 }}>

    <Pill active={panel === "info"} onClick={() => togglePanel("info")} title="Info (I)">
        <Info size={16} /> Info
    </Pill>

    <Pill active={panel === "people"} onClick={() => togglePanel("people")} title="People (G)">
        <Users size={16} /> People
    </Pill>
</div>

```

```

<Pill active={panel === "chat"} onClick={() => togglePanel("chat")} title="Chat (C)">
  <MessageCircle size={16} /> Chat
</Pill>
</div>
</div>

/* PANEL */

<div className="h-full bg-white text-zinc-900 border-l border-black/10 transition-[width] duration-200 ease-out overflow-hidden shadow-[inset_8px_0_16px_rgba(0,0,0,0.05)]"
  style={{ width: panel ? PANEL_W : 0 }}>
  {panel === "chat" && <ChatPanel messages={messages} onSend={sendChat} />}
  {panel === "people" && (
    <PeoplePanel
      waiting={waiting}
      selected={selectedWait}
      onToggle={toggleSelect}
      onAdmit={admitOne}
      onDeny={denyOne}
      onAdmitSelected={admitSelected}
      onDenySelected={denySelected}
      participants={participants}
    />
  )}
  {panel === "info" && <HostInfoPanel />}
</div>
</div>
</div>
</div>
);

{
}

```

---

```
Students :  
// src/pages/student/AIGuidance.jsx  
  
import { useEffect, useMemo, useState } from "react";  
  
import { Link } from "react-router-dom";  
  
import API from "../../api/axios";  
  
import {  
    Sparkles, Target, Lightbulb, Briefcase, CalendarDays, Star, StarOff,  
    Plus, Check, Download, ChevronDown, ChevronUp, Wand2  
} from "lucide-react";  
  
/* ----- helpers ----- */  
  
const Card = ({ children, className = "" }) => (  
    <div className={"rounded-2xl border border-slate-200/60 bg-white/80 shadow-sm " +  
        className}>  
        {children}  
    </div>  
);  
  
const toDateInput = (d) => new Date(d).toISOString().slice(0, 16); // yyyy-mm-ddThh:mm  
  
function icsForTask({ title, startISO, durationMins = 45, description = "Study Session" }) {  
    const uid = Math.random().toString(36).slice(2);  
    const dtStart = new Date(startISO);  
    const dtEnd = new Date(dtStart.getTime() + durationMins * 60 * 1000);  
    const fmt = (x) => x.toISOString().replace(/[-]/g, "").split(".").[0] + "Z"; // UTC basic  
    return new Blob([`BEGIN:VCALENDAR  
VERSION:2.0  
PRODID://StudentConnect//Guidance//EN  
BEGIN:VEVENT  
UID:${uid}  
DTSTAMP:${fmt(new Date())}`])  
}
```

```

DTSTART:${fmt(dtStart)}

DTEND:${fmt(dtEnd)}

SUMMARY:${title.replace(/\n/g, " ")}

DESCRIPTION:${description.replace(/\n/g, " ")}

END:VEVENT

END:VCALENDAR`], { type: "text/calendar;charset=utf-8" });

}

function downloadBlob(blob, filename) {

const url = URL.createObjectURL(blob);

const a = document.createElement("a");

a.href = url; a.download = filename; a.click();

URL.revokeObjectURL(url);

}

/* ----- page ----- */

export default function AIGuidance() {

const [skills, setSkills] = useState([]); // /student/skill-progress

const [jobs, setJobs] = useState([]); // /student/jobs

const [loading, setLoading] = useState(true);

// UI/State

const [targetSkill, setTargetSkill] = useState("");

const [boost, setBoost] = useState(0);

const [showPlan, setShowPlan] = useState(true);

// localStorage keys

const PLAN_KEY = "student.guidance.weeklyPlan";

const BOOK_KEY = "student.guidance.bookmarks";

const [plan, setPlan] = useState(() => {

try { return JSON.parse(localStorage.getItem(PLAN_KEY) || "[]"); }

```

```

    catch { return []; }

});

const [bookmarks, setBookmarks] = useState(() => {
  try { return new Set(JSON.parse(localStorage.getItem(BOOK_KEY) || "[]")); }
  catch { return new Set(); }
});

useEffect(() => { localStorage.setItem(PLAN_KEY, JSON.stringify(plan)); }, [plan]);
useEffect(() => { localStorage.setItem(BOOK_KEY, JSON.stringify(Array.from(bookmarks))); }, [bookmarks]);

useEffect(() => {
  document.title = "AI Career Guidance | StudentConnect";
  (async () => {
    setLoading(true);
    try {
      const [sp, jb] = await Promise.allSettled([
        API.get("/student/skill-progress"),
        API.get("/student/jobs"),
      ]);
      const s = Array.isArray(sp.value?.data) ? sp.value.data : [];
      setSkills(s);
      setJobs(Array.isArray(jb.value?.data) ? jb.value.data : []);
      const flat = s.map(sk => ({
        name: sk.skill?.name || sk.skill,
        avg: sk.history.length ? (sk.history.reduce((a, b) => a + b.score, 0) / sk.history.length) : 0,
      }));
      const lowest = flat.filter(x => x.name).sort((a, b) => a.avg - b.avg)[0]?.name || "";
      setTargetSkill(lowest);
    } finally {
      setLoading(false);
    }
  })();
});

```

```

        }
    })();
}, []);
}

const flatSkills = useMemo(() => (
    skills.map(s => ({
        name: s.skill?.name || s.skill,
        avg: s.history.length ? (s.history.reduce((a, b) => a + b.score, 0) / s.history.length) : 0,
        latest: s.history.slice().sort((a,b)=> new Date(b.createdAt)-new Date(a.createdAt))[0]?score ?? 0,
    }))
), [skills]);

const strengths = useMemo(
() => flatSkills.filter(x => x.avg >= 7).sort((a,b)=>b.avg-a.avg).slice(0, 3),
[flatSkills]
);

const gaps = useMemo(
() => flatSkills.filter(x => x.avg < 5).sort((a,b)=>a.avg-b.avg).slice(0, 3),
[flatSkills]
);

// Match scoring with boost
const matches = useMemo(() => {
    const boosted = new Map(flatSkills.map(s => [s.name?.toLowerCase(), s.avg]));
    if (targetSkill) {
        const k = targetSkill.toLowerCase();
        boosted.set(k, Math.min(10, (boosted.get(k) || 0) + Number(boost)));
    }
    const scoreJob = (j) => {
        const names = (j.skills || []).map(s => (typeof s === "string" ? s : s.name || "").toLowerCase());
        return Math.round(names.reduce((acc, n) => acc + (boosted.get(n) || 0), 0));
    }

```

```
};

return jobs.map(j => ({ job: j, match: scoreJob(j)})).sort((a,b)=>b.match-a.match).slice(0, 6);

}, [jobs, flatSkills, targetSkill, boost]);
```

```
const expectedGain = useMemo(() => {

if (!targetSkill || boost <= 0 || jobs.length === 0) return 0;

const base = jobs.slice(0, 20).map(j => {

const names = (j.skills || []).map(s => (typeof s === "string" ? s : s.name || "").toLowerCase());

const baseSum = flatSkills.reduce((acc, sk) => acc + (names.includes((sk.name || "")).toLowerCase() ? sk.avg : 0), 0);

const boostedSum = flatSkills.reduce((acc, sk) => {

let v = sk.avg;

if ((sk.name || "").toLowerCase() === targetSkill.toLowerCase()) v = Math.min(10, v + Number(boost));

return acc + (names.includes((sk.name || "").toLowerCase()) ? v : 0);

}, 0);

return boostedSum - baseSum;

});

return Math.round((base.reduce((a,b)=>a+b, 0) / base.length) || 0);

}, [jobs, flatSkills, targetSkill, boost]);
```

```
// ✅ MOVE THIS HOOK ABOVE ANY EARLY RETURN

const suggested = useMemo(() => {

const arr = [];

if (gaps[0]?.name) arr.push(`Practice ${gaps[0].name}: 3 problems + 30m notes`);

if (strengths[0]?.name) arr.push(`Mini-project using ${strengths[0].name} and publish on GitHub`);

arr.push("Watch 1 webinar & take 5 bullet notes");

return arr;

}, [gaps, strengths]);
```

```
/* ----- plan actions ----- */

const addPlanItem = (title) => {
```

```

const item = {
  id: Math.random().toString(36).slice(2),
  title: title.trim(),
  when: toDateInput(new Date(Date.now() + 24*3600*1000)),
  minutes: 45, done: false, skill: targetSkill || "",
};

setPlan(p => [item, ...p]);
};

const togglePlanDone = (id) => setPlan(p => p.map(x => x.id === id ? { ...x, done: !x.done } : x));
const updatePlanItem = (id, patch) => setPlan(p => p.map(x => x.id === id ? { ...x, ...patch } : x));
const removePlanItem = (id) => setPlan(p => p.filter(x => x.id !== id));

const exportPlanICS = () => {
  const fmt = (x) => x.toISOString().replace(/[:-]/g, "").split(".")[0] + "Z";
  const events = plan.map(t => {
    const start = new Date(t.when);
    const end = new Date(start.getTime() + (t.minutes || 45) * 60000);
    return `BEGIN:VEVENT
UID:${t.id}
DTSTAMP:${fmt(new Date())}
DTSTART:${fmt(start)}
DTEND:${fmt(end)}
SUMMARY:${t.title.replace(/\n/g, " ")}
DESCRIPTION:Skill: ${t.skill || "-"}
END:VEVENT`;
  }).join("\n");
  downloadBlob(new Blob([`BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//StudentConnect//Guidance Plan//EN
${events}
END:VCALENDAR`], { type: "text/calendar;charset=utf-8" }), "WeeklyPlan.ics");
};

```

```

/* ----- bookmarks ----- */

const toggleBookmark = (id) => {
  setBookmarks(prev => {
    const next = new Set(prev);
    next.has(id) ? next.delete(id) : next.add(id);
    return next;
  });
};

// Early return is now AFTER all hooks above
if (loading) return <div className="max-w-6xl mx-auto p-6">Loading guidance...</div>

return (
  <div className="max-w-6xl mx-auto p-6 space-y-6">
    <div className="flex items-center justify-between">
      <h1 className="text-2xl font-semibold tracking-tight">AI Career Guidance</h1>
      <Link to="/student/dashboard" className="text-sky-700 underline">Back to dashboard</Link>
    </div>

    <Card className="p-5 bg-gradient-to-br from-sky-50/70 to-white">
      <div className="flex items-center gap-2">
        <Sparkles className="text-sky-600" />
        <h2 className="font-semibold">Personalized Snapshot</h2>
      </div>
      <p className="text-sm text-slate-600 mt-1">
        Built from your test history and current openings. Simulate progress, shortlist roles, and export
        your weekly plan.
      </p>
    </Card>

```

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

  <Card className="p-5">

    <div className="flex items-center gap-2 mb-2">
      <Target className="text-emerald-600" />
      <h3 className="font-semibold">Strengths</h3>
    </div>

    {strengths.length === 0 ? (
      <p className="text-sm text-slate-600">Take a few skill tests to reveal your strengths.</p>
    ) : (
      <ul className="text-sm space-y-2">
        {strengths.map(s => (
          <li key={s.name} className="flex items-center justify-between border rounded-xl px-3 py-2">
            <span>{s.name}</span>
            <span className="text-emerald-700 text-xs px-2 py-1 rounded-full border border-emerald-200 bg-emerald-50">
              Avg {s.avg.toFixed(1)}
            </span>
          </li>
        )));
      </ul>
    )}
  </Card>

  <Card className="p-5">
    <div className="flex items-center gap-2 mb-2">
      <Lightbulb className="text-amber-600" />
      <h3 className="font-semibold">Gaps to Improve</h3>
    </div>

    {gaps.length === 0 ? (
      <p className="text-sm text-slate-600">Nice! No major gaps detected yet.</p>
    ) : (

```

```

<ul className="text-sm space-y-2">
  {gaps.map(s => (
    <li key={s.name} className="flex items-center justify-between border rounded-xl px-3 py-2">
      <span>{s.name}</span>
      <span className="text-amber-700 text-xs px-2 py-1 rounded-full border border-amber-200 bg-amber-50">
        Avg {s.avg.toFixed(1)}
      </span>
    </li>
  )))
</ul>
)}

</Card>
</div>

<Card className="p-5">
  <div className="flex flex-col md:flex-row md:items-end md:justify-between gap-3">
    <div>
      <div className="flex items-center gap-2">
        <Briefcase className="text-indigo-600" />
        <h3 className="font-semibold">Top Matches</h3>
      </div>
      <p className="text-sm text-slate-600">Simulate improving a skill to see how your matches move.</p>
    </div>
    <div className="flex items-center gap-2">
      <label className="text-sm text-slate-600">Target skill</label>
      <select value={targetSkill} onChange={(e) => setTargetSkill(e.target.value)} className="border rounded-lg px-3 py-1.5 bg-white">
        <option value="">Select...</option>
        {flatSkills.filter(s=>s.name).map(s => (<option key={s.name} value={s.name}>{s.name}</option>))}
      </select>
    </div>
  </div>
</Card>

```

```

        </select>

        <label className="text-sm text-slate-600">Boost</label>
        <input type="range" min="0" max="3" step="1" value={boost} onChange={(e)=>setBoost(e.target.value)} />
        <span className="text-sm font-medium w-8 text-right">{boost}</span>
    </div>
</div>

{expectedGain > 0 && (
    <div className="mt-2 text-xs text-emerald-700 px-2 py-1 rounded-md bg-emerald-50 inline-flex items-center gap-1">
        <Wand2 size={14}/> Expected average match gain: +{expectedGain}
    </div>
)
}

{matches.length === 0 ? (
    <p className="text-sm text-slate-600 mt-3">No jobs available yet. Check again soon.</p>
) : (
    <ul className="grid md:grid-cols-2 gap-3 mt-4">
        {matches.map(({ job, match }) => {
            const id = job._id;
            const saved = bookmarks.has(id);
            return (
                <li key={id} className="border rounded-xl p-4">
                    <div className="flex items-start justify-between">
                        <div>
                            <div className="font-medium">{job.title}</div>
                            <div className="text-xs text-slate-500">{job.company?.name || "—" • {job.location || "Remote"}</div>
                        </div>
                    <div className="flex items-center gap-2">

```

```

        <span className="text-xs px-2 py-1 rounded-full border bg-indigo-50 text-indigo-700">Match {match}</span>

        <button onClick={() => toggleBookmark(id)} className="p-1.5 rounded-lg border hover:bg-slate-50" title={saved ? "Remove from shortlist" : "Save to shortlist"}>
          {saved ? <Star className="text-amber-500" size={16}/> : <StarOff size={16}/>}
        </button>
      </div>
    </div>
    <p className="text-sm text-slate-600 mt-2 line-clamp-3">{job.description}</p>
    <div className="mt-3 flex flex-wrap gap-2">
      {((job.skills || []).slice(0,5).map((s, i) => (
        <span key={i} className="text-xs px-2 py-0.5 rounded-full border bg-slate-50">
          {typeof s === "string" ? s : (s.name || "")}
        </span>
      )))
    </div>
  </li>
);
}
</ul>
)
</Card>

<Card className="p-5">
  <div className="flex items-center justify-between">
    <div className="flex items-center gap-2">
      <CalendarDays className="text-sky-600" />
      <h3 className="font-semibold">Weekly Plan</h3>
    </div>
    <button onClick={()=>setShowPlan(s => ls)} className="text-sm text-slate-600 inline-flex items-center gap-1">
      {showPlan ? <>Hide <ChevronUp size={16}/></> : <>Show <ChevronDown size={16}/></>}
    </button>
  </div>

```

```

        </button>
    </div>

    {showPlan && (
        <>
        <div className="mt-3 flex flex-wrap gap-2">
            {suggested.map((s,i)=>(
                <button key={i} onClick={()=>addPlanItem(s)} className="text-xs px-2.5 py-1 rounded-full border bg-slate-50 hover:bg-slate-100" title="Add to plan">
                    <Plus size={12} className="inline mr-1"/> {s}
                </button>
            )))
        </div>

        <div className="mt-3 flex gap-2">
            <input
                placeholder="Add your own task (e.g., 'Finish portfolio project README')"
                className="flex-1 border rounded-lg px-3 py-2"
                onKeyDown={(e)=>{
                    if (e.key === "Enter") {
                        const v = e.currentTarget.value.trim();
                        if (v) { addPlanItem(v); e.currentTarget.value=""; }
                    }
                }}
            />
            <button className="px-3 py-2 rounded-lg border bg-white hover:bg-slate-50" disabled
                title="Type above and press Enter">
                Add
            </button>
        </div>
    )} {plan.length === 0 ? (

```

```

<p className="text-sm text-slate-600 mt-3">No tasks yet. Use the suggestions or add your own.</p>

):(
<ul className="mt-4 space-y-3">
{plan.map((t)=>
<li key={t.id} className="border rounded-xl p-3">
<div className="flex flex-col md:flex-row md:items-center md:justify-between gap-3">
<div className="flex items-start gap-2">
<button
    onClick={()=>togglePlanDone(t.id)}
    className={`${`mt-0.5 h-5 w-5 rounded border grid place-items-center ${t.done ? "bg-emerald-500 border-emerald-600 text-white" : "bg-white"}`}`}
    title="Mark done"
    >
    {t.done && <Check size={14}/>}
    </button>
<div>
    <div className={`${`font-medium ${t.done ? "line-through text-slate-400" : ""}`}}>{t.title}</div>
    <div className="text-xs text-slate-500">Skill: {t.skill || "-"}</div>
</div>
</div>

<div className="flex flex-wrap items-center gap-2">
<input type="datetime-local" value={t.when} onChange={(e)=>updatePlanItem(t.id, {when: e.target.value})} className="border rounded-lg px-2 py-1 text-sm" title="When" />
<input type="number" min={15} step={5} value={t.minutes} onChange={(e)=>updatePlanItem(t.id, {minutes: Number(e.target.value) || 45})} className="w-20 border rounded-lg px-2 py-1 text-sm" title="Duration (mins)" />
<button
    onClick={()=>{
        const blob = icsForTask({
            title: t.title,
    })
    const file = new File([blob], `${t.title}.ics`, {type: "text/calendar"})
    const url = URL.createObjectURL(file)
    window.open(url)
    updatePlanItem(t.id, {when: Date.now(), minutes: t.minutes})
    setPlan(plan.filter(item => item.id !== t.id))
    toast.success("Task added to calendar")
    }
    >
    Add to calendar
    </button>
</div>

```

```
        startISO: t.when,
        durationMins: t.minutes,
        description: `Skill: ${t.skill || "-"},
    });
    downloadBlob(blob, `${t.title.replace(/\W+/g, "_")}.ics`);
}
}

className="px-2.5 py-1.5 rounded-lg border text-sm hover:bg-slate-50 inline-flex items-center gap-1"
title="Add to calendar"
>
<button onClick={()=>removePlanItem(t.id)} className="px-2.5 py-1.5 rounded-lg border text-sm hover:bg-rose-50" title="Remove">
    Remove
</button>
</div>
</div>
</li>
))}

</ul>
}

{plan.length > 0 && (
<div className="mt-4 flex items-center gap-2">
    <button onClick={exportPlanICS} className="px-3 py-2 rounded-lg border hover:bg-slate-50 inline-flex items-center gap-2" title="Download a calendar file with all tasks">
        <Download size={16}/> Export weekly plan (ICS)
    </button>
    </div>
)
</>
```

```

        )}
      </Card>

      <Card className="p-5">
        <div className="flex items-center gap-2 mb-2">
          <Wand2 className="text-violet-600" />
          <h3 className="font-semibold">Micro-project Idea</h3>
        </div>
        <p className="text-sm text-slate-700">
          {strengths[0]?.name
            ? `Build a tiny app or script that showcases your ${strengths[0].name}. Keep it under 6 hours and publish on GitHub with a 200-word README.`
            : "Pick one strength and ship a tiny app in under 6 hours. Publish with a clear README."}
        </p>
      </Card>
    </div>
  );
}

}

```

```

import CareerRoadmap from "../../components/CareerRoadmap";

function CareerRoadmapPage() {
  return (
    <div className="min-h-screen bg-gray-50 p-6">
      <CareerRoadmap />
    </div>
  );
}

export default CareerRoadmapPage;

```

```
import { useEffect, useMemo, useState } from "react";
import API from "../../api/axios";
import { useParams } from "react-router-dom";
import { Calendar, Clock, Users, Link as LinkIcon, FileDown, Gift, ChevronRight } from "lucide-react";

const toAbs = (url) => {
  if (!url) return null;
  const origin = (API.defaults.baseURL && new URL(API.defaults.baseURL).origin) ||
    window.location.origin;
  return url.startsWith("http") ? url : `${origin}${url.startsWith("/") ? url : `/${url}`}`;
};

export default function HackathonDetail() {
  const { id } = useParams();
  const [hack, setHack] = useState(null);
  const [reg, setReg] = useState(null);
  const [mySub, setMySub] = useState(null);
  const [leader, setLeader] = useState([]);
  const [busy, setBusy] = useState(false);

  // modals
  const [showRegister, setShowRegister] = useState(false);
  const [teamName, setTeamName] = useState("");
  const [showUnreg, setShowUnreg] = useState(false);

  const loadAll = async () => {
    const [h, r, s, l] = await Promise.allSettled([
      API.get(`/student/hackathons/${id}`),
      API.get(`/student/hackathons/${id}/registration`),
      API.get(`/student/hackathons/${id}/my-submission`),
      API.get(`/student/hackathons/${id}/leaderboard`),
    ]);
  }
}
```

```

    ]);

    setHack(h.value?.data || null);
    setReg(r.value?.data || null);
    setMySub(s.value?.data || null);
    setLeader(l.value?.data || []);
}

useEffect(()=>{ loadAll(); }, [id]);

const now = new Date();
const started = hack ? now >= new Date(hack.startAt) : false;
const ended = hack ? now > new Date(hack.endAt) : false;

const countdown = useMemo(() => {
  if (!hack) return "";
  const target = started && !ended ? new Date(hack.endAt) : new Date(hack.startAt);
  let diff = +target - +now;
  if (diff <= 0) return "0d 0h 0m";
  const d = Math.floor(diff / (1000*60*60*24));
  const h = Math.floor((diff / (1000*60*60)) % 24);
  const m = Math.floor((diff / (1000*60)) % 60);
  return `${d}d ${h}h ${m}m`;
}, [hack]);

const doRegister = async () => {
  setBusy(true);
  try {
    await API.post('/student/hackathons/${id}/register', { teamName: teamName.trim() });
    setShowRegister(false);
    setTeamName("");
    await loadAll();
  }
}

```

```
        } finally {
            setBusy(false);
        }
    };

const doUnregister = async () => {
    setBusy(true);
    try {
        await API.delete(`student/hackathons/${id}/register`);
        setShowUnreg(false);
        await loadAll();
    } finally {
        setBusy(false);
    }
};

const submit = async (e) => {
    e.preventDefault();
    setBusy(true);
    const f = new FormData(e.currentTarget);
    try {
        await API.post(`student/hackathons/${id}/submit`, f, {
            headers: { "Content-Type": "multipart/form-data" }
        });
        e.currentTarget.reset();
        await loadAll();
        alert("Submitted!");
    } catch (e2) {
        alert(e2?.response?.data?.message || "Submit failed");
    } finally {
        setBusy(false);
    }
};
```

```
        }

    };

if (!hack) return <div className="max-w-6xl mx-auto p-6">Loading...</div>

return (
<div className="max-w-7xl mx-auto p-6 space-y-6">
  {/* Hero */}
  <div className="bg-white/80 backdrop-blur rounded-2xl border p-5">
    <div className="flex flex-col md:flex-row md:items-start gap-4">
      {hack.bannerUrl ? (
        <img src={hack.bannerUrl} alt="" className="h-28 w-full md:w-44 object-cover rounded-xl border" />
      ) : null}

      <div className="flex-1">
        <div className="flex items-start justify-between gap-3">
          <h1 className="text-xl font-semibold">{hack.title}</h1>
          <span className="text-xs px-2 py-1 rounded-full border bg-blue-50 text-blue-700">{hack.status}</span>
        </div>
      </div>
    </div>
  </div>
  <div className="mt-2 grid grid-cols-1 sm:grid-cols-3 gap-2 text-sm text-gray-700">
    <div className="inline-flex items-center gap-2">
      <Calendar size={16} /> {new Date(hack.startAt).toLocaleString()} → {new Date(hack.endAt).toLocaleString()}
    </div>
    <div className="inline-flex items-center gap-2">
      <Clock size={16} /> {started ? (ended ? "Ended" : "Ends in") : "Starts in"} {countdown}
    </div>
    {hack.prize ? (
      <div className="inline-flex items-center gap-2">
```

```
<Gift size={16} /> Prize: <b>{hack.prize}</b>
</div>
) : null}
</div>

{(hack.skills| |[]).length ? (
<div className="mt-2 flex flex-wrap gap-2 text-xs">
{(hack.skills| |[]).map((s, i) => (
<span key={i} className="px-2 py-0.5 rounded-full bg-gray-50 ring-1 ring-gray-200">
{typeof s === "string" ? s : s.name}
</span>
))}
</div>
) : null}

<p className="text-sm text-gray-700 mt-3 whitespace-pre-wrap">{hack.brief}</p>

<div className="mt-4 flex gap-2">
{reg ? (
<button onClick={()=>setShowUnreg(true)} disabled={busy} className="px-3 py-2 rounded-lg border">
Leave event
</button>
) : (
<button onClick={()=>setShowRegister(true)} disabled={busy} className="px-3 py-2 rounded-lg bg-blue-600 text-white">
Register
</button>
)}
</div>
</div>
</div>
```

```
</div>

{/* Rules & Submission */}

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

  <div className="bg-white/80 backdrop-blur rounded-2xl border p-5">

    <h3 className="font-semibold mb-2">Rules</h3>

    <div className="text-sm whitespace-pre-wrap">{hack.rules || "—"}</div>

    <h3 className="font-semibold mt-4 mb-2">Resources</h3>

    <div className="text-sm whitespace-pre-wrap">{hack.resources || "—"}</div>

  </div>

  <div className="bg-white/80 backdrop-blur rounded-2xl border p-5">

    <h3 className="font-semibold mb-3">Submit Project</h3>

    {new Date() > new Date(hack.endAt) ? (
      <div className="text-sm text-gray-500">Submission window closed.</div>
    ) : new Date() < new Date(hack.startAt) ? (
      <div className="text-sm text-gray-500">You can submit when the event starts.</div>
    ) : !reg ? (
      <div className="text-sm text-gray-500">Register to submit.</div>
    ) : (

      <form onSubmit={submit} className="space-y-3">

        <input name="repoUrl" placeholder="Repo URL (optional)" className="w-full border rounded-lg px-3 py-2" />

        <input name="demoUrl" placeholder="Demo URL (optional)" className="w-full border rounded-lg px-3 py-2" />

        <textarea name="notes" placeholder="Notes for judges (optional)" className="w-full border rounded-lg px-3 py-2" />

        <div>

          <label className="text-sm block mb-1">Upload ZIP/PDF (optional)</label>

          <input type="file" name="file" accept=".zip,.pdf" />

        </div>

    )}

  </div>

</div>
```

```

        <button disabled={busy} className="px-4 py-2 rounded-lg bg-blue-600 text-white">
          {busy ? "Submitting..." : "Submit / Update"}
        </button>
      </form>
    )}

{mySub && (
  <div className="mt-4 text-sm">
    <div className="font-medium mb-1">Your submission</div>
    <div className="flex items-center gap-2">
      <LinkIcon size={14} /> Repo:&nbsp;
      {mySub.repoUrl ? (
        <a className="text-blue-600 underline" href={mySub.repoUrl} target="_blank"
        rel="noreferrer">Open</a>
      ) : "—"}
    </div>
    <div className="flex items-center gap-2 mt-1">
      <LinkIcon size={14} /> Demo:&nbsp;
      {mySub.demoUrl ? (
        <a className="text-blue-600 underline" href={mySub.demoUrl} target="_blank"
        rel="noreferrer">Open</a>
      ) : "—"}
    </div>
    <div className="flex items-center gap-2 mt-1">
      <FileDown size={14} /> File:&nbsp;
      {mySub.fileUrl ? (
        <a className="text-blue-600 underline" href={toAbs(mySub.fileUrl)} target="_blank"
        rel="noreferrer" download>Download</a>
      ) : "—"}
    </div>
    <div className="mt-2">
      Score: {mySub.score ?? "—" } {mySub.rank ? `• Rank #${mySub.rank}` : ""}
    
```

```

        </div>

        {mySub.feedback ? <div className="mt-1 italic text-gray-700">"{mySub.feedback}"</div> :
      null}

      </div>
    )}

</div>

</div>

/* Leaderboard */

<div className="bg-white/80 backdrop-blur rounded-2xl border p-5">
  <h3 className="font-semibold mb-3">Leaderboard</h3>
  {leader.length === 0 ? (
    <div className="text-sm text-gray-500">Not published yet.</div>
  ) : (
    <table className="w-full text-sm">
      <thead className="bg-gray-50">
        <tr>
          <th className="p-2">Rank</th>
          <th className="p-2 text-left">Student</th>
          <th className="p-2">Score</th>
        </tr>
      </thead>
      <tbody>
        {leader.map((l)=>(
          <tr key={l._id} className="border-b">
            <td className="p-2 text-center">{l.rank}</td>
            <td className="p-2">{l.student?.name || "-"}</td>
            <td className="p-2 text-center">{l.score}</td>
          </tr>
        )))
      </tbody>
    
```

```

        </table>
    )}
</div>

/* Register modal */
{showRegister && (
    <div className="fixed inset-0 z-50 bg-black/40 grid place-items-center p-4">
        <div className="bg-white rounded-2xl w-full max-w-md p-6">
            <h3 className="font-semibold mb-3">Register for {hack.title}</h3>
            <input
                value={teamName}
                onChange={(e)=>setTeamName(e.target.value)}
                placeholder="Team name (optional)"
                className="w-full border rounded-lg px-3 py-2"
            />
            <div className="flex justify-end gap-2 mt-4">
                <button onClick={()=>setShowRegister(false)} className="px-3 py-2 rounded-lg border">Cancel</button>
                <button onClick={doRegister} disabled={busy} className="px-3 py-2 rounded-lg bg-blue-600 text-white">
                    {busy ? "Registering..." : "Register"}
                </button>
            </div>
        </div>
    </div>
)}

/* Unregister modal */
{showUnreg && (
    <div className="fixed inset-0 z-50 bg-black/40 grid place-items-center p-4">
        <div className="bg-white rounded-2xl w-full max-w-md p-6">
            <h3 className="font-semibold mb-3">Leave this hackathon?</h3>

```

```

<p className="text-sm text-gray-600">You can re-register before it ends.</p>

<div className="flex justify-end gap-2 mt-4">

  <button onClick={()=>setShowUnreg(false)} className="px-3 py-2 rounded-lg border">Cancel</button>

  <button onClick={doUnregister} disabled={busy} className="px-3 py-2 rounded-lg bg-red-600 text-white">
    {busy ? "Leaving..." : "Leave"}
  </button>
</div>
</div>
</div>
)}

</div>
);

}

```

---

```

import { useEffect, useState } from "react";
import API from "../api/axios";
import { Link } from "react-router-dom";
import { Trophy, MapPin, Calendar, Building2, ChevronRight } from "lucide-react";

export default function Hackathons() {
  const [rows, setRows] = useState([]);
  const [loading, setLoading] = useState(true);

  const load = async () => {
    setLoading(true);
    try {
      const res = await API.get("/student/hackathons");
      setRows(Array.isArray(res.data) ? res.data : []);
    } catch (e) {
      setRows([]);
    }
  };
}
```

```
    } finally {
      setLoading(false);
    }
  };

useEffect(() => { load(); }, []);

return (
  <div className="max-w-7xl mx-auto p-6 space-y-6">
    <div className="flex items-center justify-between">
      <h1 className="text-2xl font-semibold flex items-center gap-2">
        <Trophy /> Hackathons
      </h1>
    </div>

  {loading ? (
    <div className="grid md:grid-cols-2 gap-4">
      {[1,2,3,4].map((i)=>(
        <div key={i} className="rounded-2xl border bg-white/70 p-5 animate-pulse">
          <div className="flex gap-3">
            <div className="h-16 w-24 rounded-lg bg-slate-200" />
            <div className="flex-1">
              <div className="h-4 w-40 rounded bg-slate-200" />
              <div className="mt-2 h-3 w-56 rounded bg-slate-200" />
              <div className="mt-2 h-3 w-32 rounded bg-slate-200" />
            </div>
          </div>
        </div>
      )))
    ) : rows.length === 0 ? (

```

```
<div className="rounded-2xl border bg-white/70 p-10 text-center text-sm text-gray-600">
  No hackathons yet. Check back soon!
</div>

) : (
  <div className="grid md:grid-cols-2 gap-4">
    {rows.map((h) => (
      <Link
        key={h._id}
        to={`/student/hackathons/${h._id}`}
        className="group bg-white/80 backdrop-blur rounded-2xl border p-4 hover:shadow-sm transition"
      >
        <div className="flex gap-3">
          {h.bannerUrl ? (
            <img src={h.bannerUrl} alt="" className="h-16 w-24 object-cover rounded-lg border" />
          ) : (
            <div className="h-16 w-24 rounded-lg bg-gray-100 border grid place-items-center text-gray-400">
              <Trophy size={18} />
            </div>
          )}
        </div>
      </Link>
    ))
  <div className="flex-1">
    <div className="flex items-start justify-between gap-3">
      <div>
        <div className="font-semibold">{h.title}</div>
        <div className="text-xs text-gray-500 flex items-center gap-1 mt-0.5">
          <Calendar size={14} />
          {new Date(h.startAt).toLocaleString()} → {new Date(h.endAt).toLocaleString()}
        </div>
      </div>
    </div>
    <span className="text-xs px-2 py-1 rounded-full border bg-blue-50 text-blue-700">
```

```
{h.status || "upcoming"}  
</span>  
</div>  
  
<div className="mt-2 text-xs text-gray-600 flex items-center gap-2">  
  {(h.skills || []).slice(0, 4).map((s, idx) => (  
    <span key={idx} className="px-2 py-0.5 rounded-full bg-gray-50 ring-1 ring-gray-200">  
      {typeof s === "string" ? s : s.name}  
    </span>  
  ))}  
</div>  
</div>  
</div>  
  
<div className="mt-3 flex items-center justify-between text-xs text-gray-600">  
  <span className="inline-flex items-center gap-1">  
    <MapPin size={14} />  
    {h.location || "Online"}  
  </span>  
  <span className="inline-flex items-center gap-1 text-blue-600">  
    View details <ChevronRight size={16} />  
  </span>  
</div>  
</Link>  
))}  
</div>  
})  
</div>  
);  
}
```

---

```
import InternshipRecommendations from "../../components/InternshipRecommendations";

function InternshipPage() {
  return (
    <div className="min-h-screen bg-gray-50 p-6">
      <InternshipRecommendations />
    </div>
  );
}

export default InternshipPage;
```

---

```
import { useEffect, useMemo, useState } from "react";
import API from "../../api/axios";
import useSmartState from "../../hooks/useSmartState";
import {
  Briefcase, MapPin, Filter, Search, ChevronDown, Building2, Sparkles, FileText
} from "lucide-react";

/* --- tiny UI helpers --- */
function Pill({ children, className = "" }) {
  return (
    <span className={`inline-flex items-center gap-1 px-2.5 py-1 rounded-full text-xs font-medium ring-1 ${className}`}>
      {children}
    </span>
  );
}

function SkeletonCard() {
  return (
```

```
<div className="rounded-2xl border bg-white/70 backdrop-blur p-5 animate-pulse">
  <div className="flex items-center gap-3">
    <div className="h-10 w-10 rounded-lg bg-slate-200" />
    <div className="flex-1">
      <div className="h-4 w-40 rounded bg-slate-200" />
      <div className="mt-2 h-3 w-24 rounded bg-slate-200" />
    </div>
    <div className="h-5 w-16 rounded-full bg-slate-200" />
  </div>
  <div className="mt-4 h-3 w-full rounded bg-slate-200" />
  <div className="mt-2 h-3 w-5/6 rounded bg-slate-200" />
  <div className="mt-3 flex gap-2">
    <div className="h-5 w-16 rounded-full bg-slate-200" />
    <div className="h-5 w-20 rounded-full bg-slate-200" />
    <div className="h-5 w-14 rounded-full bg-slate-200" />
  </div>
</div>
);
}

export default function JobsBoard() {
  const [jobs, setJobs] = useState([]);
  const [skills, setSkills] = useState([]);
  const [loading, setLoading] = useState(true);

  const [filter, setFilter] = useSmartState("student.jobs.filter", { q: "", skillId: "" });
  const [sortBy, setSortBy] = useSmartState("student.jobs.sort", "newest");

  const [selected, setSelected] = useState(null);
  const [applyOpen, setApplyOpen] = useState(false);
  const [busy, setBusy] = useState(false);
```

```

const [resumeName, setResumeName] = useState("");

// Persisted set of applied job IDs
const [appliedIds, setAppliedIds] = useSmartState("student.jobs.appliedIds", []);
const appliedSet = useMemo(() => new Set(appliedIds), [appliedIds]);
const addApplied = (id) => setAppliedIds((arr) => Array.from(new Set([...arr, id])));
const removeApplied = (id) => setAppliedIds((arr) => arr.filter((x) => x !== id));

// ----- data load -----
const load = async () => {
  setLoading(true);
  try {
    const [j, s] = await Promise.all([
      API.get("/student/jobs"),
      API.get("/admin/skills"),
    ]);
    setJobs(Array.isArray(j.data) ? j.data : []);
    setSkills(Array.isArray(s.data) ? s.data : []);
  }
  catch {/* ignore */}
  catch (e) {
    console.error("Load jobs failed:", e?.response?.data || e.message);
  }
}

```

```

} finally {
    setLoading(false);
}
};

useEffect(() => { load(); }, []);

// ----- filtering & sorting -----
const visible = useMemo(() => {
    const q = filter.q.trim().toLowerCase();

    let out = jobs.filter((j) => {
        const matchQ = q
            ? (j.title?.toLowerCase().includes(q) ||
                j.description?.toLowerCase().includes(q) ||
                j.company?.name?.toLowerCase().includes(q))
            : true;

        const matchS = filter.skillId
            ? (j.skills || []).some((s) =>
                (typeof s === "string" ? s === filter.skillId : s._id === filter.skillId)
            )
            : true;

        return matchQ && matchS;
    });
}

out.sort((a, b) => {
    if (sortBy === "newest") return new Date(b.createdAt) - new Date(a.createdAt);
    if (sortBy === "oldest") return new Date(a.createdAt) - new Date(b.createdAt);
});

```

```

    if (sortBy === "company") return (a.company?.name || "").localeCompare(b.company?.name || "");
  }

  if (sortBy === "title") return (a.title || "").localeCompare(b.title || "");

  return 0;
});

return out;
}, [jobs, filter, sortBy]);

// ----- modal helpers -----
const openApply = (job) => { setSelected(job); setResumeName(""); setApplyOpen(true); };

const closeApply = () => { setApplyOpen(false); setSelected(null); setBusy(false); setResumeName(""); };

// ----- submit application -----
const submitApplication = async (e) => {
  e.preventDefault();

  if (!selected?._id) return;

  setBusy(true);

  const form = new FormData(e.currentTarget);

  const coverLetter = (form.get("coverLetter") || "").toString().trim();
  const resumeFile = form.get("resume");

  try {
    if (resumeFile && resumeFile.name) {
      const fd = new FormData();
      fd.append("coverLetter", coverLetter);
      fd.append("resume", resumeFile);

      await API.post(`/student/jobs/${selected._id}/apply`, fd, {
        headers: { "Content-Type": "multipart/form-data" },
      });
    }
  } catch (err) {
    console.error(err);
  }
};

```

```

    } else {

        await API.post(`student/jobs/${selected._id}/apply`, { coverLetter });

    }

    addApplied(selected._id);
    closeApply();
}

} catch (err) {

    const status = err?.response?.status;

    const msg = err?.response?.data?.message || "Failed to apply";

    if (status === 409) addApplied(selected._id); // already applied
    alert(msg);
    setBusy(false);
}

};

// ----- withdraw application -----
const withdraw = async (jobId) => {

    try {

        await API.delete(`student/jobs/${jobId}/apply`);

        removeApplied(jobId);

    } catch (err) {

        alert(err?.response?.data?.message || "Withdraw failed");

    }

};

return (
    <div className="max-w-7xl mx-auto p-6 space-y-6">
        {/* header */}
        <div className="flex flex-col gap-3 md:flex-row md:items-center md:justify-between">
            <h1 className="text-2xl font-semibold tracking-tight flex items-center gap-2">
                <Briefcase size={20} /> Jobs & Internships

```

```
</h1>

<div className="flex flex-col sm:flex-row gap-2 sm:items-center">
  {/* search */}
  <div className="relative">
    <Search size={16} className="absolute left-3 top-2.5 text-gray-400" />
    <input
      className="pl-9 pr-3 py-2 w-full sm:w-72 rounded-xl border bg-white/70 backdrop-blur
      focus:outline-none focus:ring-2 focus:ring-blue-200"
      placeholder="Search role, company..."
      value={filter.q}
      onChange={(e) => setFilter({ ...filter, q: e.target.value })}>
    />
  </div>

  {/* skill filter */}
  <div className="relative">
    <select
      className="appearance-none pr-9 pl-3 py-2 rounded-xl border bg-white/70 backdrop-blur
      focus:outline-none focus:ring-2 focus:ring-blue-200"
      value={filter.skillId}
      onChange={(e) => setFilter({ ...filter, skillId: e.target.value })}>
      >
      <option value="">All skills</option>
      {skills.map((s) => (
        <option key={s._id} value={s._id}>{s.name}</option>
      ))}
    </select>
    <ChevronDown size={16} className="pointer-events-none absolute right-3 top-2.5 text-gray-400" />
  </div>
```

```

/* sort */

<div className="relative">
  <select
    className="appearance-none pr-9 pl-3 py-2 rounded-xl border bg-white/70 backdrop-blur
    focus:outline-none focus:ring-2 focus:ring-blue-200"
    value={sortBy}
    onChange={(e) => setSortBy(e.target.value)}
  >
    <option value="newest">Newest first</option>
    <option value="oldest">Oldest first</option>
    <option value="company">Company</option>
    <option value="title">Title</option>
  </select>
  <ChevronDown size={16} className="pointer-events-none absolute right-3 top-2.5 text-gray-
  400" />
</div>
</div>
</div>

/* content */

{loading ? (
  <div className="grid md:grid-cols-2 gap-4">
    <SkeletonCard /><SkeletonCard /><SkeletonCard /><SkeletonCard />
  </div>
) : visible.length === 0 ? (
  <div className="rounded-2xl border bg-white/70 backdrop-blur p-10 text-center">
    <div className="mx-auto h-10 w-10 rounded-xl bg-blue-50 text-blue-600 grid place-items-
    center">
      <Sparkles size={18} />
    </div>
    <h2 className="mt-3 font-semibold">No matching roles</h2>
    <p className="text-sm text-gray-600 mt-1">Try adjusting your search or skill filter.</p>

```

```

        </div>
    ) : (
        <div className="grid md:grid-cols-2 gap-4">
            {visible.map((j) => {
                const isApplied = appliedSet.has(j._id);
                const companyLogo = j.company?.logoUrl;
                const companyName = j.company?.name || "-";
                return (
                    <div key={j._id} className="group bg-white/80 backdrop-blur rounded-2xl border p-5 hover:shadow-sm transition">
                        <div className="flex items-start justify-between">
                            <div className="flex items-center gap-3">
                                {companyLogo ? (
                                    <img src={companyLogo} alt="logo" className="h-10 w-10 rounded-lg object-cover border" />
                                ) : (
                                    <div className="h-10 w-10 rounded-lg bg-gray-100 text-gray-500 grid place-items-center">
                                        <Building2 size={16} />
                                    </div>
                                )}
                            <div>
                                <p className="font-semibold leading-tight">{j.title}</p>
                                <p className="text-xs text-gray-500">{companyName}</p>
                            </div>
                        </div>
                    </div>
                    <Pill className={
                        j.type === "internship"
                            ? "bg-emerald-50 text-emerald-700 ring-emerald-200"
                            : "bg-blue-50 text-blue-700 ring-blue-200"
                    }>
                );
            });
        );
    );
}

```

```
>
{j.type}
</Pill>
</div>

<p className="text-sm text-gray-600 mt-3 line-clamp-3">{j.description}</p>

<div className="mt-3 flex items-center gap-2 text-xs text-gray-600">
  <MapPin size={14} /> {j.location || "Remote"}
</div>

<div className="mt-3 flex flex-wrap gap-2">
  {(j.skills || []).map((s) => {
    const name = typeof s === "string" ? s : s.name;
    const id = typeof s === "string" ? s : s._id;
    return (
      <Pill key={id} className="bg-gray-50 text-gray-700 ring-gray-200">{name}</Pill>
    );
  })}
</div>

<div className="mt-4 flex justify-end items-center gap-3">
  {isApplied && (
    <Pill className="bg-slate-100 text-slate-700 ring-slate-200">Applied</Pill>
  )}
  {isApplied ? (
    <button
      onClick={() => withdraw(j._id)}
      className="px-3 py-2 rounded-lg border hover:bg-gray-50"
    >
```

```
        Withdraw
      </button>
    ) : (
      <button
        onClick={() => openApply(j)}
        className="px-3 py-2 rounded-lg bg-blue-600 text-white hover:bg-blue-700"
      >
        Apply
      </button>
    )}

</div>

<div className="mt-2 h-1 w-full rounded-b-2xl bg-gradient-to-r from-blue-100 via-transparent to-indigo-100 opacity-0 group-hover:opacity-100 transition" />
</div>
);

})}
</div>
)}

/* Apply Drawer/Modal */
{applyOpen && selected && (
  <div className="fixed inset-0 z-50 grid md:place-items-center">
    <div className="absolute inset-0 bg-black/40" onClick={closeApply} />
    <div className="relative bg-white w-full md:max-w-lg md:rounded-2xl p-6 shadow-xl">
      <div className="flex items-center justify-between mb-2">
        <h3 className="font-semibold">Apply — {selected.title}</h3>
        <button onClick={closeApply} className="text-sm text-gray-500">Close</button>
      </div>
      <div className="text-xs text-gray-500 mb-3">{selected.company?.name}</div>
```

```

<form onSubmit={submitApplication} className="space-y-4">
  <div>
    <label className="text-sm font-medium">Cover letter (optional)</label>
    <textarea
      name="coverLetter"
      rows={5}
      className="mt-1 w-full border rounded-lg px-3 py-2 focus:outline-none focus:ring-2
      focus:ring-blue-200"
      placeholder="Introduce yourself, highlight your fit...">
    />
  </div>

  <div>
    <label className="text-sm font-medium flex items-center gap-2">
      <FileText size={16} /> Resume (PDF/DOC) — optional
    </label>
    <input
      name="resume"
      type="file"
      accept=".pdf,.doc,.docx"
      className="mt-2 block w-full text-sm file:mr-3 file:rounded-md file:border file:px-3 file:py-1.5 file:bg-gray-50 hover:file:bg-gray-100"
      onChange={(e) => setResumeName(e.target.files?[0]?.name || "")}>
    />
    {resumeName && (
      <div className="mt-1 text-xs text-gray-600">Selected: {resumeName}</div>
    )}
  </div>

  <div className="flex justify-end gap-2">
    <button type="button" onClick={closeApply} className="px-4 py-2 rounded-lg border">

```

```

        Cancel

    </button>

    <button disabled={busy} className="px-4 py-2 rounded-lg bg-blue-600 text-white hover:bg-blue-700">

        {busy ? "Submitting..." : "Submit Application"}

    </button>

</div>

</form>

</div>

</div>

) {}

</div>

);

}

}

```

---

```

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

import {

    Calendar,
    Video,
    FileText,
    Briefcase,
    Building2,
    Search,
    ChevronDown,
} from "lucide-react";

import API from "../../api/axios";

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

/* ----- helpers ----- */

const apiOrigin =

```

```
window.location.origin;

const toAbsolute = (p) =>
!p ? null : p.startsWith("http") ? p : `${apiOrigin}${p.startsWith("/") ? p : `/${p}`}`;

/* status badge tones */
const tone = {
  applied: { bg: "bg-slate-100", text: "text-slate-700", ring: "ring-slate-200" },
  shortlisted: { bg: "bg-amber-50", text: "text-amber-700", ring: "ring-amber-200" },
  interview: { bg: "bg-indigo-50", text: "text-indigo-700", ring: "ring-indigo-200" },
  offer: { bg: "bg-emerald-50", text: "text-emerald-700", ring: "ring-emerald-200" },
  rejected: { bg: "bg-rose-50", text: "text-rose-700", ring: "ring-rose-200" },
  default: { bg: "bg-gray-100", text: "text-gray-700", ring: "ring-gray-200" },
};

function StatusChip({ value }) {
  const key = (value || "").toLowerCase();
  const t = tone[key] || tone.default;
  return (
    <span className={`inline-flex items-center gap-1 px-2.5 py-1 rounded-full text-xs font-medium ${t.bg} ${t.text} ring-1 ${t.ring}>
      {value || "--"}
    </span>
  );
}

function SkeletonRow() {
  return (
    <div className="rounded-2xl border bg-white/60 backdrop-blur-[2px] p-4 animate-pulse">
      <div className="flex items-center justify-between">
        <div className="h-4 w-40 rounded bg-slate-200" />
    
```

```
<div className="h-5 w-20 rounded-full bg-slate-200" />
</div>

<div className="mt-3 flex flex-wrap items-center gap-4">
  <div className="h-4 w-24 rounded bg-slate-200" />
  <div className="h-4 w-40 rounded bg-slate-200" />
</div>
</div>
);

}

/* ----- page ----- */

export default function MyApplications() {
  const [apps, setApps] = useState([]);
  const [loading, setLoading] = useState(true);
  const [q, setQ] = useState("");
  const [sortBy, setSortBy] = useState("newest");

  const load = async () => {
    setLoading(true);
    try {
      const res = await API.get("/student/jobs/mine/applications");
      setApps(Array.isArray(res.data) ? res.data : []);
    } catch (e) {
      console.error("Load apps failed:", e?.response?.data || e.message);
    } finally {
      setLoading(false);
    }
  };

  useEffect(() => { load(); }, []);
}
```

```

const visible = useMemo(() => {
  const term = q.trim().toLowerCase();
  let out = apps.filter((a) => {
    if (!term) return true;
    const title = a.job?.title?.toLowerCase() || "";
    const company = a.job?.company?.name?.toLowerCase() || "";
    const status = a.status?.toLowerCase() || "";
    return title.includes(term) || company.includes(term) || status.includes(term);
  });
  out.sort((a, b) => {
    if (sortBy === "newest") return new Date(b.createdAt) - new Date(a.createdAt);
    if (sortBy === "oldest") return new Date(a.createdAt) - new Date(b.createdAt);
    if (sortBy === "company") return (a.job?.company?.name || "").localeCompare(b.job?.company?.name || "");
    if (sortBy === "status") return (a.status || "").localeCompare(b.status || "");
    return 0;
  });
  return out;
}, [apps, q, sortBy]);

return (
  <div className="max-w-6xl mx-auto p-6 space-y-6">
    {/* header */}
    <div className="flex flex-col gap-3 sm:flex-row sm:items-center sm:justify-between">
      <h1 className="text-2xl font-semibold tracking-tight flex items-center gap-2">
        <Briefcase size={20} /> My Applications
      </h1>
      <div className="flex flex-col sm:flex-row gap-2 sm:items-center">

```

```
<div className="relative">
  <Search size={16} className="absolute left-3 top-2.5 text-gray-400" />
  <input
    value={q}
    onChange={(e) => setQ(e.target.value)}
    placeholder="Search by role, company, or status..."
    className="pl-9 pr-3 py-2 w-full sm:w-72 rounded-xl border bg-white/70 backdrop-blur
    focus:outline-none focus:ring-2 focus:ring-blue-200"
  />
</div>

<div className="relative">
  <select
    value={sortBy}
    onChange={(e) => setSortBy(e.target.value)}
    className="appearance-none pr-9 pl-3 py-2 rounded-xl border bg-white/70 backdrop-blur
    focus:outline-none focus:ring-2 focus:ring-blue-200"
  >
    <option value="newest">Newest first</option>
    <option value="oldest">Oldest first</option>
    <option value="company">Company</option>
    <option value="status">Status</option>
  </select>
  <ChevronDown size={16} className="pointer-events-none absolute right-3 top-2.5 text-gray-400" />
</div>
</div>
</div>

/* content */
{loading ? (
  <div className="space-y-3">
    <SkeletonRow /><SkeletonRow /><SkeletonRow />
)
```

```
</div>

) : visible.length === 0 ? (
  <div className="rounded-2xl border bg-white/60 backdrop-blur-[2px] p-10 text-center">
    <div className="mx-auto h-10 w-10 rounded-xl bg-blue-50 text-blue-600 grid place-items-center">
      <Briefcase size={18} />
    </div>
    <h2 className="mt-3 font-semibold">No applications yet</h2>
    <p className="text-sm text-gray-600 mt-1">When you apply, they'll appear here with live status and interview links.</p>
    <Link
      to="/student/jobs"
      className="inline-flex mt-4 px-4 py-2 rounded-xl bg-blue-600 text-white hover:bg-blue-700">
      >
      Browse jobs
    </Link>
  </div>
) : (
  <div className="grid gap-3">
    {visible.map((a) => {
      const iv = a.interview;
      const companyLogo = a.job?.company?.logoUrl;
      const companyName = a.job?.company?.name || "--";
      return (
        <div
          key={a._id}
          className="group rounded-2xl border bg-white/70 backdrop-blur transition hover:shadow-sm">
        >
        <div className="p-4">
          {/* top row */}

```

```
<div className="flex items-start justify-between gap-3">
  <div className="flex items-center gap-3">
    {companyLogo ? (
      <img
        src={companyLogo}
        alt="logo"
        className="h-10 w-10 rounded-lg object-cover border"
      />
    ) : (
      <div className="h-10 w-10 rounded-lg bg-gray-100 text-gray-500 grid place-items-center">
        <Building2 size={16} />
      </div>
    )}
  <div>
    <div className="font-medium leading-tight">{a.job?.title}</div>
    <div className="text-xs text-gray-500">{companyName}</div>
  </div>
</div>

<StatusChip value={a.status} />
</div>

/* meta row */
<div className="mt-4 flex flex-wrap items-center gap-4 text-sm">
  <span className="inline-flex items-center gap-1.5 text-gray-700">
    <FileText size={16} />
    {a.cvUrl ? (
      <a
        className="underline text-blue-600 hover:text-blue-700"
        href={toAbsolute(a.cvUrl)}
    ) : (
      <span>{a.cvUrl}</span>
    )}
  </span>
</div>
```

```

        target="_blank"
        rel="noreferrer"
        download
      >
      Resume
    </a>
  ) : (
  <span className="text-gray-500">No resume</span>
)
</span>

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

<span className="inline-flex items-center gap-1.5 text-gray-700">
  <Calendar size={16} />
  Applied on {new Date(a.createdAt).toLocaleDateString()}
</span>
</div>

/* interview row */
<div className="mt-4">
  {iv ? (
    <div className="flex flex-wrap items-center gap-3">
      <span className="inline-flex items-center gap-1.5 rounded-lg border px-2.5 py-1 text-xs text-indigo-700 bg-indigo-50 ring-1 ring-indigo-200">
        <Calendar size={14} />
        {new Date(iv.startsAt).toLocaleString()} • {iv.durationMins} mins • {iv.stage}
      </span>
      <Link
        to={`/student/webinar/${iv.roomId}`}

```

```

        className="inline-flex items-center gap-1.5 rounded-lg bg-indigo-600 text-white px-3
py-1.5 text-sm hover:bg-indigo-700"
        title="Join interview"
      >
  <Video size={16} /> Join Interview
</Link>
</div>
): (
<div className="text-sm text-gray-500">
  No interview scheduled yet. You'll see the link here once it's booked.
</div>
)
</div>
</div>

<div className="h-1 w-full rounded-b-2xl bg-gradient-to-r from-blue-100 via-transparent
to-indigo-100 opacity-0 group-hover:opacity-100 transition" />
</div>
);
}}}
</div>
)
</div>
);
}

```

```

import { useEffect, useMemo, useState } from "react";
import API from "../api/axios";
import { Line } from "react-chartjs-2";
import {
  Chart as ChartJS,
  LineElement,

```

```
CategoryScale,  
LinearScale,  
PointElement,  
Tooltip,  
Legend,  
} from "chart.js";  
  
import { Sparkles } from "lucide-react";  
  
ChartJS.register(LineElement, CategoryScale, LinearScale, PointElement, Tooltip, Legend);  
  
function Card({ children, className = "" }) {  
  return (  
    <div className={`bg-white/80 backdrop-blur rounded-2xl border p-6 ${className}`}>  
      {children}  
    </div>  
  );  
}  
  
function SkeletonCard() {  
  return (  
    <Card className="animate-pulse">  
      <div className="h-5 w-40 rounded bg-slate-200" />  
      <div className="mt-4 h-56 w-full rounded bg-slate-200" />  
    </Card>  
  );  
}  
  
export default function SkillProgress() {  
  const [rows, setRows] = useState([]);  
  const [loading, setLoading] = useState(true);
```

```

useEffect(() => {
  const fetchProgress = async () => {
    try {
      const res = await API.get("/student/skill-progress");
      setRows(Array.isArray(res.data) ? res.data : []);
    } catch (err) {
      console.error("Failed to fetch skill progress", err);
      setRows([]);
    } finally {
      setLoading(false);
    }
  };
  fetchProgress();
}, []);
}

const colors = [
  "#2563eb", "#059669", "#f59e0b", "#dc2626", "#7c3aed", "#0ea5e9", "#10b981", "#f43f5e",
];

```

  

```

const { labels, datasets } = useMemo(() => {
  const allLabels = [];
  const ds = rows.map((skillData, idx) => {
    const reversed = [...(skillData.history || [])].reverse();
    const scores = reversed.map((h) => h.score);
    if (reversed.length > allLabels.length) {
      for (let i = allLabels.length; i < reversed.length; i++) allLabels.push(`#${i + 1}`);
    }
    const c = colors[idx % colors.length];
    return {
      label: skillData.skill?.name || skillData.skill,
      data: scores,
    };
  });
  return { labels, datasets };
});

```

```
borderColor: c,  
backgroundColor: c + "33",  
pointRadius: 3,  
fill: false,  
tension: 0.3,  
};  
});  
return { labels: allLabels, datasets: ds };  
, [rows]);  
  
return (  
<div className="max-w-7xl mx-auto p-6 space-y-6">  
  <div className="flex items-center gap-2">  
    <h1 className="text-2xl font-semibold tracking-tight">Your Skill Progress</h1>  
  </div>  
  
  {/* Combined chart */}  
  {loading ? (  
    <SkeletonCard />  
  ) : rows.length === 0 ? (  
    <Card className="text-center">  
      <div className="mx-auto h-10 w-10 rounded-xl bg-blue-50 text-blue-600 grid place-items-center">  
        <Sparkles size={18} />  
      </div>  
      <h2 className="mt-3 font-semibold">No test records yet</h2>  
      <p className="text-sm text-gray-600 mt-1">  
        As you take tests, your progress will appear here.  
      </p>  
    </Card>  
  ) : (
```

```

<Card>

  <h2 className="text-lg font-semibold text-blue-700">Combined Progress</h2>
  <p className="text-sm text-gray-600 mt-1">
    Each line shows your score trend across attempts (0–10).
  </p>
  <div className="h-80 mt-4">
    <Line
      data={{ labels, datasets }}
      options={{
        responsive: true,
        maintainAspectRatio: false,
        scales: {
          y: { beginAtZero: true, max: 10, ticks: { stepSize: 1 } },
        },
        plugins: {
          legend: { position: "bottom" },
          tooltip: { intersect: false, mode: "index" },
        },
      }}
    />
  </div>
</Card>
)}

```

```

/* Per-skill summary */
{rows.length > 0 && (
  <div className="grid gap-4 md:grid-cols-2">
    {rows.map((skill, i) => {
      const history = Array.isArray(skill.history) ? skill.history : [];
      if (!history.length) return null;
      const attempts = history.length;

```

```

const latest = [...history].sort(
  (a, b) => new Date(b.createdAt) - new Date(a.createdAt)
)[0];

const average = (history.reduce((s, h) => s + h.score, 0) / attempts).toFixed(1);

let badge = "Beginner";
let badgeTone = "bg-rose-50 text-rose-700 ring-rose-200";
if (average >= 7) { badge = "Pro"; badgeTone = "bg-emerald-50 text-emerald-700 ring-emerald-200"; }
else if (average >= 4) { badge = "Intermediate"; badgeTone = "bg-amber-50 text-amber-700 ring-amber-200"; }

const barPct = Math.min(100, Math.max(0, (average / 10) * 100));
const lineColor = colors[i % colors.length];

return (
  <Card key={skill.skill?._id || skill.skill}>
    <div className="flex items-center justify-between">
      <h3 className="text-lg font-semibold">{skill.skill?.name || skill.skill}</h3>
      <span className={`text-xs px-2.5 py-1 rounded-full ring-1 ${badgeTone}`}>{badge}</span>
    </div>

    <div className="mt-3 text-sm text-gray-700">
      Attempts: <b>{attempts}</b>
      <span className="mx-2 text-gray-300">•</span>
      Latest: <b>{latest.score}</b> / {latest.total}
      <span className="mx-2 text-gray-300">•</span>
      Average: <b>{average}</b>
    </div>

    {/* pastel progress bar */}
    <div className="mt-3 h-2 w-full bg-gray-100 rounded-full overflow-hidden">

```

```
<div
  className="h-full"
  style={{
    width: `${barPct}`,
    background: `${lineColor}55`,
    outline: `1px solid ${lineColor}55`,
  }}
/>
</div>
</Card>
);
)}
)
}
</div>
);
}
);
}
```

---

```
import { motion } from "framer-motion";
import { useEffect, useMemo, useState } from "react";
import {
  CalendarDays,
  Sparkles,
  Target,
  ArrowRight,
  ArrowUpRight,
  GraduationCap,
} from "lucide-react";
import { Link } from "react-router-dom";

import SkillsProgressChart from "../../components/SkillsProgressChart";
```

```
import UniversityBadges from "../../components/UniversityBadges";
import WebinarsSchedule from "../../components/WebinarsSchedule";
import NotificationPanel from "../../components/NotificationPanel";
import CareerRoadmap from "../../components/CareerRoadmap";

/* ----- tiny helpers ----- */
const fade = (delay = 0) => ({
  initial: { opacity: 0, y: 18 },
  animate: {
    opacity: 1,
    y: 0,
    transition: { duration: 0.4, ease: "easeOut", delay },
  },
});

function Card({ children, className = "" }) {
  // pastel/minimal card
  return (
    <div
      className={
        "rounded-2xl border border-slate-200/60 bg-white/80 shadow-sm hover:shadow-md transition-
        shadow " +
        className
      }
    >
      {children}
    </div>
  );
}

export default function StudentDashboard() {
```

```
const [studentName, setStudentName] = useState("Student");

useEffect(() => {
  document.title = "Student Dashboard | StudentConnect";
  const stored = localStorage.getItem("studentName");
  if (stored) setStudentName(stored);
}, []);

const greeting = useMemo(() => {
  const hr = new Date().getHours();
  if (hr < 12) return "Good morning";
  if (hr < 18) return "Good afternoon";
  return "Good evening";
}, []);

return (
  <div className="w-full">
    <div className="mx-auto max-w-screen-2xl px-4 sm:px-6 lg:px-8 py-6 space-y-6">
      {/* Header */}
      <motion.div {...fade(0)}>
        <div className="flex flex-col gap-4 md:flex-row md:items-end md:justify-between">
          <div>
            <h1 className="text-2xl sm:text-3xl font-semibold tracking-tight text-slate-900">
              {greeting}, {studentName} <span className="inline-block">👋</span>
            </h1>
            <p className="text-sm text-slate-600 mt-1">
              Your roadmap, skills and upcoming events — all in one place.
            </p>
          </div>
        </div>
      </motion.div>
    </div>
  </div>
)
```

```
<div className="rounded-xl border border-indigo-200/60 bg-gradient-to-tr from-indigo-50 via-blue-50 to-sky-50 px-4 py-3">

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

    <Sparkles className="text-indigo-600" size={16} />

    <span className="font-medium text-slate-800">Tip:</span>

    <span className="text-slate-600">

      Ship one micro-project this week to level up your portfolio.

    </span>

  </div>

</div>

</div>

</motion.div>
```

```
{/* AI Career Guidance compact tile */}

<motion.div {...fade(0.03)}>

  <Card className="p-4 bg-gradient-to-br from-sky-50/70 to-white">

    <div className="flex flex-col sm:flex-row sm:items-center sm:justify-between gap-3">

      <div>

        <h2 className="text-lg font-semibold text-slate-900">

          AI Career Guidance

        </h2>

        <p className="text-sm text-slate-600 mt-0.5">

          Personalized strengths, gaps, job matches, and a weekly plan tailored for you.

        </p>

      </div>

      <Link

        to="/student/guidance"

        className="inline-flex items-center gap-2 rounded-xl border border-sky-200 bg-white px-3 py-2 text-sky-700 hover:bg-sky-50">

        >

        Open Guidance <ArrowUpRight size={16} />

      </Link>
```

```
</div>
</Card>
</motion.div>

/* Pastel stat row */

<motion.div {...fade(0.05)} className="grid grid-cols-12 gap-4">
  <Card className="p-4 col-span-12 sm:col-span-6 lg:col-span-4 bg-gradient-to-br from-blue-50/60 to-white">
    <div className="flex items-center justify-between">
      <div>
        <p className="text-xs text-slate-500">Roadmap Milestones</p>
        <p className="text-xl font-semibold leading-tight text-slate-900">
          Next 3 steps
        </p>
      </div>
      <Target className="text-blue-600" size={20} />
    </div>
    <p className="text-xs text-slate-500 mt-1">
      Keep momentum: complete one this week.
    </p>
  </Card>

  <Card className="p-4 col-span-12 sm:col-span-6 lg:col-span-4 bg-gradient-to-br from-indigo-50/60 to-white">
    <div className="flex items-center justify-between">
      <div>
        <p className="text-xs text-slate-500">Upcoming</p>
        <p className="text-xl font-semibold leading-tight text-slate-900">
          Webinars
        </p>
      </div>
      <CalendarDays className="text-indigo-600" size={20} />
    </div>
  </Card>
</motion.div>
```

```
</div>

<p className="text-xs text-slate-500 mt-1">
  Reserve your spot & set reminders.
</p>

</Card>
```

```
<Card className="p-4 col-span-12 sm:col-span-6 lg:col-span-4 bg-gradient-to-br from-emerald-50/60 to-white">

  <div className="flex items-center justify-between">
    <div>
      <p className="text-xs text-slate-500">Academics</p>
      <p className="text-xl font-semibold leading-tight text-slate-900">
        Badges & Orgs
      </p>
    </div>
    <GraduationCap className="text-emerald-600" size={20} />
  </div>

  <p className="text-xs text-slate-500 mt-1">
    Showcase your wins on applications.
  </p>
</Card>

</motion.div>
```

```
{/* Main Grid — perfectly symmetrical (12 columns) */}

<div className="grid grid-cols-12 gap-6 auto-rows-fr">
  {/* Career Roadmap */}
  <motion.div {...fade(0.08)} className="col-span-12 xl:col-span-4">
    <Card className="p-5 h-full">
      <div className="flex items-center justify-between">
        <h2 className="text-lg font-semibold text-slate-900">
          AI Career Roadmap
        </h2>
      </div>
    </Card>
  </motion.div>
</div>
```

```
</h2>

<button className="text-sm text-indigo-600 hover:text-indigo-700 inline-flex items-center gap-1">
    View detail <ArrowRight size={16} />
</button>
</div>

<p className="text-sm text-slate-600 mt-1">
    Curated steps for courses, projects & certifications —
    tailored for you.
</p>

<div className="mt-4 overflow-y-auto max-h-[360px] pr-1">
    <CareerRoadmap />
</div>
</Card>
</motion.div>

/* Skills */

<motion.div {...fade(0.1)} className="col-span-12 md:col-span-6 xl:col-span-4">
    <Card className="p-5 h-full">
        <div className="flex items-center justify-between">
            <h2 className="text-lg font-semibold text-slate-900">
                My Skills
            </h2>
            <span className="text-xs px-2 py-1 rounded-full border bg-slate-50 text-slate-700">
                Keep building
            </span>
        </div>
        <p className="text-sm text-slate-600 mt-1">
            Track growth across your chosen domains and see where to
            practice next.
        </p>
```

```
<div className="mt-4">
  <SkillsProgressChart />
</div>
</Card>
</motion.div>

{/* Internships */}

<motion.div {...fade(0.12)} className="col-span-12 md:col-span-6 xl:col-span-4">
  <Card className="p-5 h-full">
    <div className="flex items-center justify-between">
      <h2 className="text-lg font-semibold text-slate-900">
        Recommended Internships
      </h2>
      <button className="text-sm text-indigo-600 hover:text-indigo-700 inline-flex items-center gap-1">
        See more <ArrowRight size={16} />
      </button>
    </div>
    <ul className="text-sm text-slate-800 mt-3 space-y-2">
      {"Frontend Intern — TechNova", "ML Intern — AIWorks", "Backend Intern — CodeCraft"}.map(
        (t) => (
          <li
            key={t}
            className="rounded-xl border border-slate-200/70 bg-white p-3 hover:bg-slate-50 transition">
            >
              {t}
            </li>
          )
      )
    </ul>
  </Card>
</motion.div>
```

```
</Card>

</motion.div>

{/* Badges */}

<motion.div {...fade(0.14)} className="col-span-12 md:col-span-6 xl:col-span-6">
  <Card className="p-5 h-full">
    <h2 className="text-lg font-semibold text-slate-900 mb-2">
      University Badges
    </h2>
    <UniversityBadges />
  </Card>
</motion.div>

{/* Webinars */}

<motion.div {...fade(0.16)} className="col-span-12 md:col-span-6 xl:col-span-6">
  <Card className="p-5 h-full">
    <div className="flex items-center justify-between">
      <h2 className="text-lg font-semibold text-slate-900">
        Upcoming Webinars
      </h2>
      <span className="text-xs px-2 py-1 rounded-full border bg-indigo-50 text-indigo-700">
        Live & Recorded
      </span>
    </div>
    <div className="mt-2">
      <WebinarsSchedule />
    </div>
  </Card>
</motion.div>

{/* Notifications */}
```

```

<motion.div {...fade(0.18)} className="col-span-12">
  <Card className="p-5">
    <h2 className="text-lg font-semibold text-slate-900 mb-2">
      Notifications
    </h2>
    <NotificationPanel />
  </Card>
</motion.div>
</div>
</div>
</div>
);
}

```

---

```

import { useEffect, useRef, useState } from "react";
import { useParams, Link } from "react-router-dom";
import io from "socket.io-client";

const socket = io(import.meta.env.VITE_API_BASE?.replace("/api", "") || "http://localhost:5000", {
  transports: ["websocket"]
});

export default function StudentInterviewRoom() {
  const { roomId } = useParams();
  const [connected, setConnected] = useState(false);
  const videoMine = useRef(null);
  const videoPeer = useRef(null);
  const pcRef = useRef(null);
  const localStreamRef = useRef(null);

  useEffect(() => {
    socket.on("connect", () => setConnected(true));

```

```
socket.on("disconnect", () => setConnected(false));
return () => { socket.off("connect"); socket.off("disconnect"); };
}, []);
```

```
useEffect(() => {
  if (!roomId) return;
  let mounted = true;

  (async () => {
    localStreamRef.current = await navigator.mediaDevices.getUserMedia({ video: true, audio: true });
    if (videoMine.current) videoMine.current.srcObject = localStreamRef.current;

    pcRef.current = new RTCPeerConnection();
    localStreamRef.current.getTracks().forEach(t => pcRef.current.addTrack(t, localStreamRef.current));
    pcRef.current.ontrack = (e) => { if (videoPeer.current) videoPeer.current.srcObject = e.streams[0]; };
  });
})
```

```
socket.emit("requestJoin", { roomId, username: "Candidate" });

socket.on("admitted", async () => {
  const offer = await pcRef.current.createOffer();
  await pcRef.current.setLocalDescription(offer);
  socket.emit("signal-offer", { target: "host", sdp: offer }); // server routes to host
});
```

```
socket.on("signal-answer", async ({ sdp }) => {
  await pcRef.current.setRemoteDescription(new RTCSessionDescription(sdp));
});
```

```
socket.on("signal-ice", async ({ candidate }) => {
  if (candidate) await pcRef.current.addIceCandidate(new RTCIceCandidate(candidate));
```

```

    });

pcRef.current.onicecandidate = (e) => {
  if (e.candidate) socket.emit("signal-ice", { target: "host", candidate: e.candidate });
};

})();

return () => {
  mounted = false;
  socket.off("admitted");
  socket.off("signal-answer");
  socket.off("signal-ice");
  pcRef.current?.close();
  localStreamRef.current?.getTracks().forEach(t => t.stop());
};

}, [roomId]);

return (
  <div className="min-h-screen bg-gray-50 p-4">
    <div className="max-w-5xl mx-auto">
      <div className="flex items-center justify-between mb-3">
        <Link to="/student/dashboard" className="text-blue-600 underline">Back</Link>
        <span className={`text-xs px-2 py-1 rounded ${connected ? "bg-green-100 text-green-700": "bg-gray-200 text-gray-600"}`}>{connected?"Connected":"Disconnected"}</span>
      </div>

      <div className="grid md:grid-cols-2 gap-3">
        <video ref={videoPeer} autoPlay playsInline className="w-full bg-black rounded-xl aspect-video" />
        <video ref={videoMine} autoPlay playsInline muted className="w-full bg-black rounded-xl aspect-video" />
      </div>
    </div>
  </div>
);

```

```
        </div>
    </div>
);
}
```

---

```
import { useEffect, useState } from "react";
import { motion } from "framer-motion";
import { UserCircle, Mail, Phone, GraduationCap, Building2, FileText } from "lucide-react";
import API from "../../api/axios";
```

```
function Field({ label, children }) {
    return (
        <label className="block text-sm">
            <span className="text-gray-600">{label}</span>
            <div className="mt-1">{children}</div>
        </label>
    );
}
```

```
export default function StudentProfile() {
    const [formData, setFormData] = useState({
        name: "",
        username: "",
        email: "",
        phone: "",
        university: "",
        education: "UG",
        skills: "",
        resume: null,
    });
    const [busy, setBusy] = useState(false);
```

```
useEffect(() => {
  document.title = "My Profile | StudentConnect";
  fetchProfile();
}, []);

const fetchProfile = async () => {
  try {
    const res = await API.get("/student/profile");
    const data = res.data || {};
    if (!data.name) throw new Error("Invalid user data");
    setFormData({
      name: data.name || "",
      username: data.username || "",
      email: data.email || "",
      phone: data.phone || "",
      university: data.university || "",
      education: data.education || "UG",
      skills: Array.isArray(data.skills) ? data.skills.join(", ") : (data.skills || ""),
      resume: null,
    });
    localStorage.setItem("studentName", data.name);
  } catch (err) {
    console.error("Failed to load profile:", err.message || err);
  }
};

const handleChange = (e) => {
  const { name, value, files } = e.target;
  if (name === "resume") {
    setFormData((prev) => ({ ...prev, resume: files?.[0] || null }));
  }
};
```

```

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

const handleSubmit = async (e) => {
  e.preventDefault();
  if (busy) return;
  setBusy(true);
  try {
    const form = new FormData();
    Object.entries(formData).forEach(([k, v]) => {
      if (v !== null && v !== undefined && v !== "") form.append(k, v);
    });
    await API.post("/student/profile/update", form); // use POST per your backend
    localStorage.setItem("studentName", formData.name);
    alert("Profile updated successfully");
  } catch (err) {
    console.error("Failed to update profile:", err.response || err.message);
    alert("Update failed. Check console for details.");
  } finally {
    setBusy(false);
  }
};

return (
  <div className="max-w-5xl mx-auto p-6">
    <motion.div
      initial={{ opacity: 0, y: 24 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ duration: 0.35 }}>

```

```
  className="bg-white/80 backdrop-blur rounded-2xl border p-6 md:p-8"

>

 {/* header */}

<div className="flex flex-col md:flex-row md:items-center md:justify-between gap-4">

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

    <div className="h-14 w-14 rounded-xl bg-blue-50 text-blue-600 grid place-items-center">
      <UserCircle size={28} />
    </div>

    <div>

      <h2 className="text-xl font-semibold leading-tight">
        {formData.name || "Student"}
      </h2>

      <p className="text-xs text-gray-500">@{formData.username || "username"}</p>
    </div>
  </div>

</div>

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

  <label className="text-sm font-medium flex items-center gap-2">
    <FileText size={16} /> Upload Resume (PDF)
  </label>

  <input
    type="file"
    name="resume"
    accept=".pdf"
    onChange={handleChange}
    className="mt-2 w-full md:w-72 text-sm file:mr-3 file:rounded-md file:border file:px-3
    file:py-1.5 file:bg-gray-50 hover:file:bg-gray-100"
  />

  {formData.resume && (
    <p className="text-xs text-gray-600 mt-1 truncate">Selected: {formData.resume.name}</p>
  )}
</div>
```

```
</div>

</div>

/* form */

<form onSubmit={handleSubmit} className="mt-6 grid gap-5">

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

    <Field label="Full Name">

      <input
        type="text"
        name="name"
        value={formData.name}
        onChange={handleChange}
        required
        className="w-full px-3 py-2 border rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-200"
      />

    </Field>

    <Field label="Username">
      <input
        type="text"
        name="username"
        value={formData.username}
        disabled
        className="w-full px-3 py-2 border rounded-lg bg-gray-50 text-gray-500"
      />
    </Field>
  </div>

  <div className="grid md:grid-cols-2 gap-4">
    <Field label="Email">
```

```
<div className="relative">
  <Mail size={16} className="absolute left-3 top-2.5 text-gray-400" />
  <input
    type="email"
    name="email"
    value={formData.email}
    onChange={handleChange}
    required
    className="w-full pl-9 pr-3 py-2 border rounded-lg focus:outline-none focus:ring-2
    focus:ring-blue-200"
  />
</div>
</Field>

<Field label="Phone">
  <div className="relative">
    <Phone size={16} className="absolute left-3 top-2.5 text-gray-400" />
    <input
      type="text"
      name="phone"
      value={formData.phone}
      onChange={handleChange}
      className="w-full pl-9 pr-3 py-2 border rounded-lg focus:outline-none focus:ring-2
      focus:ring-blue-200"
    />
  </div>
</Field>
</div>

<div className="grid md:grid-cols-2 gap-4">
  <Field label="University">
    <div className="relative">
```

```
<Building2 size={16} className="absolute left-3 top-2.5 text-gray-400" />

<input
  type="text"
  name="university"
  value={formData.university}
  onChange={handleChange}
  className="w-full pl-9 pr-3 py-2 border rounded-lg focus:outline-none focus:ring-2
  focus:ring-blue-200"
/>
```

```
</div>
```

```
</Field>
```

```
<Field label="Education Level">
  <div className="relative">
    <GraduationCap size={16} className="absolute left-3 top-2.5 text-gray-400" />
    <select
      name="education"
      value={formData.education}
      onChange={handleChange}
      className="w-full pl-9 pr-3 py-2 border rounded-lg appearance-none focus:outline-none
      focus:ring-2 focus:ring-blue-200"
    >
      <option value="UG">Undergraduate</option>
      <option value="PG">Postgraduate</option>
      <option value="Diploma">Diploma</option>
      <option value="Other">Other</option>
    </select>
  </div>
</Field>
</div>
```

```
<Field label="Skills (comma-separated)">
```

```

<input
  type="text"
  name="skills"
  value={formData.skills}
  onChange={handleChange}
  className="w-full px-3 py-2 border rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-200"
/>
</Field>

<div className="text-right">
  <button
    type="submit"
    disabled={busy}
    className="inline-flex items-center px-5 py-2 rounded-lg bg-blue-600 text-white hover:bg-blue-700 disabled:opacity-60">
    >
    {busy ? "Saving..." : "Save Changes"}
  </button>
</div>
</form>
</motion.div>
</div>
);

}


```

---

```

import { useEffect, useState } from "react";
import { Link } from "react-router-dom";
import API from "../api/axios";

export default function StudentWebinars() {
  const [loading, setLoading] = useState(true);

```

```
const [webinars, setWebinars] = useState([]);

useEffect(() => {
  (async () => {
    try {
      const res = await API.get("/student/webinars");
      setWebinars(res.data || []);
    } catch (e) {
      console.error("Fetch webinars failed:", e?.response?.data || e.message);
    } finally {
      setLoading(false);
    }
  })();
}, []);

if (loading) {
  return (
    <div className="p-6">
      <div className="animate-pulse h-8 w-48 bg-gray-200 rounded mb-4" />
      <div className="space-y-3">
        <div className="h-16 bg-gray-100 rounded" />
        <div className="h-16 bg-gray-100 rounded" />
      </div>
    </div>
  );
}

return (
  <div className="p-6">
    <div className="flex items-center justify-between mb-4">
      <h1 className="text-xl font-semibold">Live & Upcoming Webinars</h1>
```

```

</div>

{webinars.length === 0 ? (
  <p className="text-sm text-gray-500">No webinars available right now.</p>
) : (
  <ul className="space-y-3">
    {webinars.map((w) => (
      <li key={w.roomId} className="border rounded-lg p-4 flex items-center justify-between bg-white">
        <div>
          <p className="font-medium">{w.title}</p>
          <p className="text-xs text-gray-500">
            {w.startsAt ? new Date(w.startsAt).toLocaleString() : "No start time"} • {w.durationMins || 60} mins
          </p>
        </div>
        {/* Relative to /student route so it stays inside the student layout */}
        <Link to={`/webinar/${w.roomId}`} className="text-blue-600 text-sm underline">
          Join
        </Link>
      </li>
    )));
  </ul>
)
);
}

```

```

import { useEffect, useMemo, useRef, useState } from "react";
import { useParams, Link } from "react-router-dom";
import io from "socket.io-client";
import {

```

```
ArrowLeft, MessageCircle, Info, Smile, Maximize2, Minimize2,
MicOff, VideoOff, ScreenShare, PhoneOff
} from "lucide-react";
import useLocalMedia from "../../hooks/useLocalMedia";

const socket = io(import.meta.env.VITE_API_BASE || "http://localhost:5000", { transports: ["websocket"] });
const PANEL_W = 360;

const Pill = ({ active, children, onClick, title }) => (
  <button type="button" title={title} onClick={onClick}
    className={`px-3 h-9 rounded-full border text-sm flex items-center gap-2 transition
    ${active ? "bg-blue-600 text-white border-blue-600" : "bg-white/90 hover:bg-white border-black/10 text-gray-800"}`}
  >
    {children}
  </button>
);

const ChatPanel = ({ messages, me, onSend }) => {
  const inputRef = useRef(null);
  return (
    <div className="h-full flex flex-col">
      <div className="px-4 h-12 border-b border-black/10 flex items-center font-semibold">In-call
        messages</div>
      <div className="flex-1 overflow-y-auto p-4 space-y-2 text-sm">
        {messages.length === 0 ? (
          <p className="text-gray-500">No messages yet.</p>
        ) : messages.map((m, i) => (
          <div key={i}>{m.system ? <span className="italic text-gray-500">{m.message}</span> : (<><b
            className={m.username === me ? "text-blue-600" : ""}>{m.username}:</b><span>{m.message}</span></>)}</div>
        )));
      </div>
    </div>
  );
}
```

```

<div className="p-3 border-t border-black/10 flex gap-2">
  <input ref={inputRef} className="flex-1 border rounded-md px-3 h-10" placeholder="Send a message"
    onKeyDown={(e)=>{ if(e.key==="Enter"){ const v=inputRef.current?.value?.trim(); if(v){ onSend(v); inputRef.current.value="" }}}} />
  <button onClick={()=>{ const v=inputRef.current?.value?.trim(); if(v){ onSend(v); inputRef.current.value="" } }} className="h-10 px-4 rounded-md bg-blue-600 text-white">Send</button>
</div>
</div>
);
};

const StudentInfoPanel = () => (
<div className="h-full flex flex-col">
  <div className="px-4 h-12 border-b border-black/10 flex items-center font-semibold">Info</div>
  <div className="p-4 text-sm space-y-3">
    <p>Welcome! The host controls the meeting.</p>
    <div>
      <p className="font-medium mb-1">Shortcuts</p>
      <ul className="list-disc ml-5 space-y-1">
        <li><kbd className="px-1 py-0.5 bg-gray-100 rounded">C</kbd> — Toggle chat</li>
        <li><kbd className="px-1 py-0.5 bg-gray-100 rounded">I</kbd> — Toggle info</li>
        <li><kbd className="px-1 py-0.5 bg-gray-100 rounded">M</kbd> — Mute</li>
        <li><kbd className="px-1 py-0.5 bg-gray-100 rounded">V</kbd> — Camera</li>
        <li><kbd className="px-1 py-0.5 bg-gray-100 rounded">P</kbd> — Present</li>
        <li><kbd className="px-1 py-0.5 bg-gray-100 rounded">F</kbd> — Fullscreen</li>
        <li><kbd className="px-1 py-0.5 bg-gray-100 rounded">R</kbd> — React 😊 </li>
      </ul>
    </div>
  </div>
</div>

```

```
);

export default function StudentWebinarViewer() {
  const { roomId } = useParams();
  const me = "Student";
  const stageRef = useRef(null);

  // media hook
  const {
    videoRef, muted, cameraOff, presenting,
    toggleMute, toggleCamera, togglePresent, stopAll, ensureStream
  } = useLocalMedia();

  const [connected, setConnected] = useState(false);
  const [panel, setPanel] = useState(null); // "chat" | "info" | null
  const [messages, setMessages] = useState([]);
  const [reactions, setReactions] = useState([]);
  const containerMinH = useMemo(() => "min-h-[560px]", []);

  useEffect(() => {
    if (!roomId) return;
    // join + basic listeners
    socket.emit("joinWebinar", { roomId, username: me });
    const onMsg = (payload) => setMessages((m) => [...m, payload]);
    const onEnded = () => setMessages((m) => [...m, { system: true, message: "Webinar ended." }]);
    socket.on("chatMessage", onMsg);
    socket.on("webinarEnded", onEnded);
    socket.on("connect", () => setConnected(true));
    socket.on("disconnect", () => setConnected(false));
    return () => { socket.off("chatMessage", onMsg); socket.off("webinarEnded", onEnded);
      socket.off("connect"); socket.off("disconnect"); };
  }, [roomId, socket, setConnected, setPanel, setMessages, onMsg, onEnded, containerMinH]);
}
```

```
}, [roomId]);

// start devices when the user first interacts (click any control) OR when page loads (optional)
useEffect(() => { ensureStream().catch(() => {}); }, [ensureStream]);

// reactions auto-fade
useEffect(() => {
  if (!reactions.length) return;
  const t = setTimeout(() => setReactions((r) => r.slice(1)), 1200);
  return () => clearTimeout(t);
}, [reactions]);

const sendChat = (text) => socket.emit("chatMessage", { roomId, username: me, message: text });
const togglePanel = (next) => setPanel((p) => (p === next ? null : next));

const requestFullscreen = () => {
  const el = stageRef.current;
  if (!el) return;
  if (document.fullscreenElement) document.exitFullscreen?.();
  else el.requestFullscreen?.();
};

// keyboard shortcuts
useEffect(() => {
  const h = (e) => {
    const k = e.key.toLowerCase();
    if (k === "c") setPanel(p => (p === "chat" ? null : "chat"));
    if (k === "i") setPanel(p => (p === "info" ? null : "info"));
    if (k === "f") requestFullscreen();
    if (k === "m") toggleMute();
    if (k === "v") toggleCamera();
  };
});
```

```

    if (k === "p") togglePresent();

    if (k === "r") setReactions(r => [...r, { id: Date.now(), emoji: "👉" }]);

};

window.addEventListener("keydown", h);

return () => window.removeEventListener("keydown", h);

}, [toggleMute, toggleCamera, togglePresent]);



return (
  <div className="min-h-screen bg-gray-50">
    <div className="max-w-7xl mx-auto p-4 space-y-4">
      <div className="flex items-center justify-between">
        <Link to="/student/dashboard" className="inline-flex items-center gap-2 text-sm text-blue-600">
          <ArrowLeft size={16} /> Back to dashboard
        </Link>
        <span className={`text-xs px-2 py-1 rounded ${connected ? "bg-green-100 text-green-700" : "bg-gray-200 text-gray-600"}`}>
          {connected ? "Connected" : "Disconnected"}
        </span>
      </div>
    <div className="rounded-3xl bg-gradient-to-b from-slate-800 to-slate-900 border border-white/10 overflow-hidden">
      <div ref={stageRef} className={`relative flex w-full ${containerMinH}`}>
        /* STAGE */
        <div className="relative flex-1 bg-slate-900">
          <video ref={videoRef} className="w-full h-full object-cover" autoPlay playsInline />
        </div>
      </div>
      /* reactions */
      <div className="pointer-events-none absolute inset-0 flex items-end justify-center pb-24 gap-3">
        {reactions.map((r) => <div key={r.id} className="animate-bounce text-3xl drop-shadow">{r.emoji}</div>)}
      </div>
    </div>
  </div>
)

```

```

        </div>

/* controls */

<div className="absolute bottom-4 left-0 right-0 flex justify-center" style={{ paddingRight: panel ? PANEL_W : 0 }}>
  <div className="backdrop-blur bg-black/40 border border-white/10 rounded-full px-3 py-2 flex items-center gap-2 text-white">
    <button onClick={toggleMute} className="w-9 h-9 grid place-items-center rounded-full hover:bg-white/10" title={'Mute (M)'>
      <MicOff size={18} className={muted ? "opacity-100" : "opacity-50"} />
    </button>
    <button onClick={toggleCamera} className="w-9 h-9 grid place-items-center rounded-full hover:bg-white/10" title={'Camera (V)'>
      <VideoOff size={18} className={cameraOff ? "opacity-100" : "opacity-50"} />
    </button>
    <button onClick={togglePresent} className="w-9 h-9 grid place-items-center rounded-full hover:bg-white/10" title={'Present (P)'>
      <ScreenShare size={18} className={presenting ? "opacity-100" : "opacity-50"} />
    </button>
    <button onClick={requestFullscreen} className="w-9 h-9 grid place-items-center rounded-full hover:bg-white/10" title="Fullscreen (F)">
      {document.fullscreenElement ? <Minimize2 size={18} /> : <Maximize2 size={18} />}
    </button>
    <button onClick={stopAll} className="w-9 h-9 grid place-items-center rounded-full bg-red-600 hover:bg-red-700 ml-1" title="Leave">
      <PhoneOff size={18} />
    </button>
  </div>
</div>

/* dock */

<div className="absolute bottom-5 flex gap-2" style={{ right: (panel ? PANEL_W : 0) + 20 }}>
  <Pill active={panel === "info"} onClick={() => togglePanel("info")}>Info (I)</Pill>

```

```

        <Info size={16} /> Info
      </Pill>
      <Pill active={panel === "chat"} onClick={() => togglePanel("chat")}>
        <MessageCircle size={16} /> Chat
      </Pill>
      <Pill onClick={() => setReactions((r) => [...r, { id: Date.now(), emoji: "👍" }])}>
        title="React (R)">
          <Smile size={16} /> React
        </Pill>
      </div>
    </div>

    {/* PANEL */}
    <div className="h-full bg-white text-zinc-900 border-l border-black/10 transition-[width] duration-200 ease-out overflow-hidden shadow-[inset_8px_0_16px_rgba(0,0,0,0.05)]">
      style={{ width: panel ? PANEL_W : 0 }}>
      {panel === "chat" && <ChatPanel messages={messages} me={me} onSend={sendChat} />}
      {panel === "info" && <StudentInfoPanel />}
    </div>
    </div>
  </div>
</div>
);
}

```

```

import { useEffect, useState } from "react";
import API from "../api/axios";
import { BookOpen } from "lucide-react";

function TestSkill() {
  const [skills, setSkills] = useState([]);

```

```
const [selectedSkill, setSelectedSkill] = useState(""); // skill._id
const [questions, setQuestions] = useState([]);
const [currentIndex, setCurrentIndex] = useState(0);
const [answers, setAnswers] = useState({});
const [timeLeft, setTimeLeft] = useState(600);
const [submitted, setSubmitted] = useState(false);
const [score, setScore] = useState(null);
const [loading, setLoading] = useState(false);

useEffect(() => {
  fetchSkills();
}, []);

const fetchSkills = async () => {
  try {
    const res = await API.get("/admin/skills");
    setSkills(res.data);
  } catch (err) {
    console.error("Failed to fetch skills:", err.response?.data || err.message);
  }
};

useEffect(() => {
  let timer;
  if (!submitted && questions.length > 0) {
    timer = setInterval(() => {
      setTimeLeft((prev) => {
        if (prev <= 1) {
          clearInterval(timer);
          handleSubmit();
        }
        return 0;
      });
    }, 1000);
  }
});
```

```
        }

        return prev - 1;
    });

}, 1000);
}

return () => clearInterval(timer);

}, [submitted, questions]);

useEffect(() => {

    if (selectedSkill) fetchQuestions();

}, [selectedSkill]);

const fetchQuestions = async () => {

try {

    setLoading(true);

    const res = await API.get(`/admin/questions/skill/${selectedSkill}`);
    const data = res.data.questions || [];

    if (data.length === 0) {
        alert("No questions found for this skill. Try another one.");
    }

    setQuestions(data);
    setAnswers({});

    setCurrentIndex(0);
    setTimeLeft(600);
    setSubmitted(false);
    setScore(null);
} catch (err) {
    console.error("Failed to load questions", err.response?.data || err.message);
} finally {
```

```
    setLoading(false);
}
};

const handleOptionChange = (qid, option) => {
    setAnswers((prev) => ({ ...prev, [qid]: option }));
};

const handleSubmit = async () => {
    if (submitted) return;
    let correct = 0;
    questions.forEach((q) => {
        if (answers[q._id] === q.answer) correct++;
    });
    setScore(correct);
    setSubmitted(true);

    try {
        await API.post("/student/skill-test/submit", {
            skill: selectedSkill,
            score: correct,
            total: questions.length,
        });
    } catch (err) {
        console.error("Failed to save test result:", err.response?.data || err.message);
    }
};

const formattedTime = `${Math.floor(timeLeft / 60)}:${String(timeLeft % 60).padStart(2, "0")}`;
```

```
if (!selectedSkill) {  
  return (  
    <div className="max-w-4xl mx-auto py-12 px-6">  
      <div className="text-center mb-10">  
        <h2 className="text-3xl font-bold text-gray-800 flex justify-center items-center gap-2">  
          <BookOpen className="text-blue-600" size={28} /> Choose a Skill to Start Test  
        </h2>  
        <p className="text-gray-500 mt-2">Select one of the following skills to begin your multiple choice test</p>  
      </div>  
      <div className="grid gap-6 sm:grid-cols-2 md:grid-cols-4">  
        {skills.map((skill) => (  
          <button  
            key={skill._id}  
            onClick={() => setSelectedSkill(skill._id)}  
            className="bg-gradient-to-r from-blue-500 to-indigo-600 text-white font-semibold py-3 px-4 rounded-xl shadow hover:shadow-lg transform transition hover:scale-105">  
            >  
            {skill.name}  
          </button>  
        ))}  
      </div>  
    </div>  
  );  
}  
  
if (loading) {  
  return (  
    <div className="text-center py-20 text-lg font-medium text-gray-600">  
      Loading questions for selected skill...  
    </div>  
  );  
}
```

```
}

if (submitted) {

  const percentage = Math.round((score / questions.length) * 100);

  return (
    <div className="max-w-xl mx-auto py-10 px-4 text-center space-y-6">
      <h2 className="text-2xl font-bold text-green-600">Test Submitted!</h2>
      <p className="text-lg">
        You scored {score} out of {questions.length} ({percentage}%)
      </p>

      <div className="mt-6">
        <div className="w-full bg-gray-200 rounded-full h-4 overflow-hidden">
          <div
            className="bg-blue-600 h-full rounded-full"
            style={{ width: `${percentage}` }}
          ></div>
        </div>
        <p className="mt-2 text-sm text-gray-600">Great job! Want to challenge yourself more?</p>
      </div>

      <div className="mt-6 space-y-3">
        <p className="text-gray-800 font-medium">Take another test and improve your progress:</p>
        {skills
          .filter((s) => s._id !== selectedSkill)
          .map((skill) => (
            <button
              key={skill._id}
              onClick={() => setSelectedSkill(skill._id)}
              className="block w-full bg-blue-500 text-white py-2 rounded hover:bg-blue-600"
            >

```

```

    Take {skill.name} Test
    </button>
  )})
</div>
</div>
);
}

const currentQ = Array.isArray(questions) && questions.length > 0 ? questions[currentIndex] : null;

return (
<div className="max-w-6xl mx-auto py-10 px-4 grid md:grid-cols-[1fr_250px] gap-6">
<div>
<div className="flex justify-between items-center mb-6">
<h2 className="text-xl font-semibold">Skill Test</h2>
<span className="font-mono text-sm text-gray-700">Time Left: {formattedTime}</span>
</div>

{currentQ && (
<div className="bg-white shadow-md p-6 rounded-xl space-y-4">
<h3 className="font-medium">
  Q{currentIndex + 1}. {currentQ.question}
</h3>
<div className="space-y-2">
  {currentQ.options.map((opt, idx) => (
    <label key={idx} className="block border px-3 py-2 rounded cursor-pointer">
      <input
        type="radio"
        name={`question_${currentQ._id}`}
        value={opt}
        checked={answers[currentQ._id] === opt}
      </input>
    </label>
  ))
</div>
)
)
</div>
)
)
);
}

```

```
        onChange={() => handleOptionChange(currentQ._id, opt)}
        className="mr-2"
      />
      {opt}
    </label>
  ))}
</div>
<div className="flex justify-between pt-4">
  <button
    disabled={currentIndex === 0}
    onClick={() => setCurrentIndex((prev) => prev - 1)}
    className="bg-gray-200 px-4 py-1 rounded disabled:opacity-50"
  >
    Prev
  </button>
  {currentIndex < questions.length - 1 ? (
    <button
      onClick={() => setCurrentIndex((prev) => prev + 1)}
      className="bg-blue-600 text-white px-4 py-1 rounded"
    >
      Next
    </button>
  ) : (
    <button
      onClick={handleSubmit}
      className="bg-green-600 text-white px-4 py-1 rounded"
    >
      Submit
    </button>
  )}
</div>
```

```

        </div>
    )}
</div>

/* Sidebar */

<div className="bg-white shadow-md rounded-xl p-4">
  <div className="flex justify-between items-center mb-3">
    <h3 className="text-md font-bold">Question Overview</h3>
    <p className="text-sm text-gray-600">
      {Object.keys(answers).length}/{questions.length} Attempted
    </p>
  </div>
  <div className="grid grid-cols-5 gap-2">
    {questions.map((q, idx) => {
      const isCurrent = idx === currentIndex;
      const isAttempted = !!answers[q._id];
      return (
        <button
          key={q._id}
          title={isCurrent ? "Current" : isAttempted ? "Attempted" : "Unattempted"}
          onClick={() => setCurrentIndex(idx)}
          className={`w-10 h-10 text-sm rounded-full font-semibold border flex items-center justify-center
          ${isCurrent ? "bg-blue-700 text-white border-blue-700" : ""}
          ${!isCurrent && isAttempted ? "bg-green-500 text-white border-green-600" : ""}
          ${!isCurrent && !isAttempted ? "bg-gray-200 text-gray-800 border-gray-300" : ""}
        `}
        >
          {idx + 1}
        </button>
      );
    })
  </div>
</div>

```

```
    })}
  </div>
  <div className="mt-4 text-sm">
    <div className="flex items-center gap-2 mb-1">
      <span className="w-4 h-4 bg-blue-700 rounded-full inline-block"></span>
      <span>Current</span>
    </div>
    <div className="flex items-center gap-2 mb-1">
      <span className="w-4 h-4 bg-green-500 rounded-full inline-block"></span>
      <span>Attempted</span>
    </div>
    <div className="flex items-center gap-2">
      <span className="w-4 h-4 bg-gray-300 rounded-full inline-block"></span>
      <span>Unattempted</span>
    </div>
  </div>
</div>
);
}
```

```
export default TestSkill;
```

---

```
// src/pages/university/UniversityDashboard.jsx
import { useState } from "react";
import { Tabs, TabsContent, TabsList, TabsTrigger } from "../../components/ui/Tabs";
import {
  Brain,
  GraduationCap,
  Building2,
  FolderCheck,
```

```
Star,  
Video,  
} from "lucide-react";  
  
import StudentAnalyticsCard from "../../components/university/StudentAnalyticsCard";  
  
export default function UniversityDashboard() {  
  const [tab, setTab] = useState("analytics");  
  
  return (  
    <div className="min-h-screen bg-gray-50 py-10 px-6 md:px-12">  
      <header className="mb-8">  
        <h1 className="text-3xl font-bold text-gray-800 mb-1">University Dashboard</h1>  
        <p className="text-gray-600">Monitor and manage student career success.</p>  
      </header>  
  
      <Tabs value={tab} onChange={setTab} className="w-full">  
        <TabsList className="bg-white shadow-md rounded-lg p-2 flex gap-2 flex-wrap">  
          <TabsTrigger value="analytics" className="flex items-center gap-2 text-sm font-medium">  
            <Brain className="w-4 h-4" /> Career Analytics  
          </TabsTrigger>  
          <TabsTrigger value="collaboration" className="flex items-center gap-2 text-sm font-medium">  
            <Building2 className="w-4 h-4" /> Industry Collaboration  
          </TabsTrigger>  
          <TabsTrigger value="placements" className="flex items-center gap-2 text-sm font-medium">  
            <FolderCheck className="w-4 h-4" /> Placement System  
          </TabsTrigger>  
          <TabsTrigger value="validation" className="flex items-center gap-2 text-sm font-medium">  
            <Star className="w-4 h-4" /> Skill Validation  
          </TabsTrigger>  
          <TabsTrigger value="webinars" className="flex items-center gap-2 text-sm font-medium">  
            <Video className="w-4 h-4" /> Webinars  
          </TabsTrigger>  
        </TabsList>  
      </Tabs>  
    </div>  
  );  
}  
;
```

```
</TabsTrigger>

</TabsList>

<TabsContent value="analytics" className="mt-6">
  <StudentAnalyticsCard />
</TabsContent>

<TabsContent value="collaboration" className="mt-6">
  <div className="bg-white rounded-xl p-6 shadow">
    <h2 className="text-lg font-bold text-gray-800 mb-3">University-Company Collaboration</h2>
    <ul className="list-disc list-inside text-gray-700 space-y-1">
      <li>Initiate research collaborations with industry leaders.</li>
      <li>Manage ongoing MoUs and project funding.</li>
      <li>Track student participation in joint programs.</li>
    </ul>
  </div>
</TabsContent>

<TabsContent value="placements" className="mt-6">
  <div className="bg-white rounded-xl p-6 shadow">
    <h2 className="text-lg font-bold text-gray-800 mb-3">Internship & Placement Management</h2>
    <ul className="list-disc list-inside text-gray-700 space-y-1">
      <li>Monitor student job applications in real-time.</li>
      <li>Connect recruiters with top-performing students.</li>
      <li>Export placement reports & analytics.</li>
    </ul>
  </div>
</TabsContent>

<TabsContent value="validation" className="mt-6">
```

```

<div className="bg-white rounded-xl p-6 shadow">
  <h2 className="text-lg font-bold text-gray-800 mb-3">Skill Validation & Recommendations</h2>
  <ul className="list-disc list-inside text-gray-700 space-y-1">
    <li>Professors can endorse verified student skills.</li>
    <li>AI suggests personalized learning pathways.</li>
    <li>Track gaps and recommend certified courses.</li>
  </ul>
</div>
</TabsContent>

<TabsContent value="webinars" className="mt-6">
  <div className="bg-white rounded-xl p-6 shadow">
    <h2 className="text-lg font-bold text-gray-800 mb-3">Live Webinars & Guest Lectures</h2>
    <ul className="list-disc list-inside text-gray-700 space-y-1">
      <li>Host and manage expert-led sessions.</li>
      <li>Track attendance and student engagement.</li>
      <li>Make recordings available for asynchronous viewing.</li>
    </ul>
  </div>
</TabsContent>
</Tabs>
</div>
);
}

```

---

```

import { Link } from "react-router-dom";
import { motion } from "framer-motion";
import { useState, useEffect } from "react";
import { FaArrowUp, FaUsers, FaUniversity, FaBuilding } from "react-icons/fa";
import {

```

```
MdStars,  
MdVerified,  
MdWork,  
MdAnalytics,  
MdBusinessCenter,  
MdDesignServices,  
} from "react-icons/md";  
  
import Header from "../components/Header";  
import Footer from "../components/Footer";  
  
function Home() {  
  const [showTopBtn, setShowTopBtn] = useState(false);  
  
  useEffect(() => {  
    const handleScroll = () => setShowTopBtn(window.scrollY > 300);  
  
    window.addEventListener("scroll", handleScroll);  
  
    return () => window.removeEventListener("scroll", handleScroll);  
  }, []);  
  
  return (  
    <>  
    <Header />  
  
    <main className="font-sans text-gray-900">  
  
      {/* HERO - Dark */}  
      <section className="relative min-h-screen flex items-center bg-gradient-to-br from-[#0b0f19]  
      via-[#111827] to-[#0b0f19] text-white overflow-hidden">  
        {/* Background Shapes */}  
        <div className="absolute inset-0">  
          <svg
```

```
    className="absolute top-0 right-0 w-[500px] opacity-20"
    viewBox="0 0 500 500"
  >
  <circle cx="250" cy="250" r="200" fill="#2563eb" />
</svg>
<svg
  className="absolute bottom-0 left-0 w-[300px] opacity-10"
  viewBox="0 0 500 500"
>
  <circle cx="250" cy="250" r="200" fill="#38bdf8" />
</svg>
</div>

<div className="max-w-7xl mx-auto px-6 grid lg:grid-cols-2 gap-12 items-center relative z-10">
  {/* LEFT CONTENT */}
  <motion.div
    initial={{ opacity: 0, y: 30 }}
    animate={{ opacity: 1, y: 0 }}
    transition={{ duration: 0.7 }}
    className="text-center lg:text-left"
  >
    <h1 className="text-5xl lg:text-6xl font-extrabold leading-snug mb-6 max-w-3xl">
      Launch Your <span className="text-blue-500">Career</span> with AI-Powered Confidence
    </h1>
    <p className="text-gray-300 text-lg mb-8 max-w-xl">
      StudentConnect is your one-stop platform for skills, mentorship, verified credentials, and
      career opportunities — all in one place.
    </p>
<div className="flex flex-wrap gap-4 justify-center lg:justify-start">
  <Link
```

```

        to="/register"

        className="bg-blue-600 hover:bg-blue-700 px-6 py-3 rounded-md font-semibold
transition"

    >

        Get Started

    </Link>

    <Link

        to="/login"

        className="border border-gray-400 hover:border-blue-400 px-6 py-3 rounded-md font-
semibold transition"

    >

        Learn More

    </Link>

</div>

</motion.div>

```

```

/* RIGHT VISUAL - Modern Illustration */

<motion.div

    initial={{ opacity: 0, x: 30 }}

    animate={{ opacity: 1, x: 0 }}

    transition={{ duration: 0.7, delay: 0.2 }}

    className="flex justify-center"

>

    /*  */

</motion.div>

</div>

</section>

```

```

/* WHO WE SERVE - Modern Flow Layout */

<section className="relative py-20 bg-white overflow-hidden">

/* Background Decorative Shapes */

<div className="absolute top-0 left-0 w-40 h-40 bg-blue-100 rounded-full opacity-30 -z-10 blur-3xl"></div>

<div className="absolute bottom-0 right-0 w-64 h-64 bg-blue-200 rounded-full opacity-20 -z-10 blur-3xl"></div>

<div className="max-w-7xl mx-auto px-6 text-center">
  <h2 className="text-3xl font-bold mb-6">Who We Serve</h2>
  <p className="text-gray-600 max-w-2xl mx-auto mb-14">
    Connecting students, universities, and companies in one seamless career platform.
  </p>

/* Flow Layout */

<div className="relative flex flex-col md:flex-row justify-between items-center gap-14 md:gap-0">
  /* Connecting Line */
  <div className="hidden md:block absolute top-10 left-1/2 transform -translate-x-1/2 w-[80%] h-[2px] bg-gradient-to-r from-blue-300 via-blue-500 to-blue-300 -z-10"></div>

  [
    { icon: <FaUsers size={36} />, title: "Students", desc: "Career paths, mentorship, and verified credentials." },
    { icon: <FaUniversity size={36} />, title: "Universities", desc: "Tools to certify, track, and boost outcomes." },
    { icon: <FaBuilding size={36} />, title: "Companies", desc: "Find pre-assessed talent and hire faster." },
  ].map((item, i) => (
    <motion.div
      key={i}
      initial={{ opacity: 0, y: 30 }}
      whileInView={{ opacity: 1, y: 0 }}
      transition={{ duration: 0.5, delay: i * 0.2 }}
    >
      <div>
        {item.icon}
        <h3>{item.title}</h3>
        <p>{item.desc}</p>
      </div>
    </motion.div>
  ))
</div>

```

```

viewport={{ once: true }}

className="flex flex-col items-center text-center max-w-xs"

>

/* Floating Circular Icon */

<div className="w-20 h-20 rounded-full flex items-center justify-center bg-gradient-to-br from-blue-500 to-blue-300 text-white shadow-lg mb-6">

{item.icon}

</div>

<h3 className="text-lg font-semibold mb-2">{item.title}</h3>

<p className="text-gray-600 text-sm">{item.desc}</p>

</motion.div>

)})

</div>

</div>

</section>

/* PLATFORM HIGHLIGHTS - Light Grey */

<section className="py-20 bg-[#f8f9fb]">

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

<h2 className="text-3xl font-bold mb-14">Platform Highlights</h2>

<div className="grid sm:grid-cols-2 md:grid-cols-3 gap-10 text-left">

{[

  { icon: <MdStars />, label: "AI Career Planning", desc: "Personalized roadmaps to reach your goals." },

  { icon: <MdVerified />, label: "Verified Credentials", desc: "University-approved skill validation." },

  { icon: <MdWork />, label: "Mentorship & Hackathons", desc: "Engage with real-world projects." },

  { icon: <MdBusinessCenter />, label: "Smart Job Matching", desc: "Get matched with relevant jobs." },

  { icon: <MdDesignServices />, label: "Freelance Projects", desc: "Earn while you learn." },

  { icon: <MdAnalytics />, label: "Institutional Insights", desc: "Data-driven performance tracking." },


```

```

].map((f, i) => (
  <motion.div
    key={i}
    whileHover={{ scale: 1.03 }}
    className="bg-white p-8 rounded-xl border-l-4 border-blue-600 shadow-sm
    hover:shadow-lg transition"
  >
    <div className="flex items-center gap-3 text-blue-500 mb-3">
      {f.icon}
      <h4 className="font-medium">{f.label}</h4>
    </div>
    <p className="text-gray-600 text-sm">{f.desc}</p>
  </motion.div>
))}

</div>
</div>
</section>

/* STATS - Dark */

<section className="bg-gradient-to-r from-[#0b0f19] to-[#1e293b] py-20 text-center text-
white">

<div className="max-w-6xl mx-auto grid sm:grid-cols-3 gap-12">
  [
    { label: "Students", count: "15,000+" },
    { label: "Universities", count: "120+" },
    { label: "Companies", count: "80+" },
  ].map((stat, i) => (
    <motion.div key={i} whileInView={{ opacity: 1 }} initial={{ opacity: 0 }}>
      <p className="text-5xl font-bold text-blue-400">{stat.count}</p>
      <p className="text-gray-300 mt-3 text-base">{stat.label}</p>
    </motion.div>
  )))

```

```
</div>

</section>

/* TESTIMONIALS - White */

<section className="py-20 bg-white text-center">

  <div className="max-w-7xl mx-auto px-6">

    <h2 className="text-3xl font-bold mb-14">What People Say</h2>

    <div className="grid md:grid-cols-3 gap-10">

      {[

        { name: "Ravi (Student)", quote: "StudentConnect helped me land my first internship easily." },

        { name: "Dr. Sharma (University)", quote: "We now track student outcomes better than ever." },

        { name: "TechCorp HR", quote: "Access to pre-verified talent sped up our hiring." },

      ].map((t, i) => (

        <motion.div

          key={i}

          whileHover={{ scale: 1.02 }}

          className="bg-white p-8 rounded-xl border border-gray-200 shadow-sm hover:border-blue-500 transition"

        >

          <p className="text-gray-600 italic mb-4">{t.quote}</p>

          <p className="font-semibold text-blue-500">{t.name}</p>

        </motion.div>

      ))}

    </div>

  </div>

</section>

/* CTA - Dark */

<section className="bg-[#0b0f19] py-20 text-center text-white">

  <h3 className="text-3xl font-semibold mb-5">Start Your Career Journey Today</h3>
```

```

<p className="mb-8 text-gray-300 max-w-xl mx-auto">
  Join thousands already using StudentConnect to achieve their goals.
</p>

<Link
  to="/register"
  className="bg-blue-600 hover:bg-blue-700 px-8 py-3 rounded-md font-semibold text-white transition">
  >
  Get Started →
</Link>
</section>
</main>

{/* BACK TO TOP */}
{showTopBtn && (
  <button
    onClick={() => window.scrollTo({ top: 0, behavior: "smooth" })}
    className="fixed bottom-6 right-6 bg-blue-600 text-white p-3 rounded-full shadow-lg
    hover:bg-blue-700 transition z-50"
    aria-label="Scroll to Top"
  >
    <FaArrowUp />
  </button>
)};

<Footer />
</>
);
}

export default Home;

```

---

```
import { useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import API from "../api/axios";

function Login() {
  const navigate = useNavigate();
  const [role, setRole] = useState("student");
  const [showPass, setShowPass] = useState(false);
  const [loginData, setLoginData] = useState({ identifier: "", password: "" });
  const [error, setError] = useState("");

  const handleChange = (e) => setLoginData({ ...loginData, [e.target.name]: e.target.value });

  const validateLogin = () => {
    if (!loginData.identifier || !loginData.password) {
      setError("Please fill in all fields.");
      return false;
    }
    const isEmail = loginData.identifier.includes("@");
    const emailRegex = /^[^@\s]+@[^\s]+\.[^\s]+$/;
    if (isEmail && !emailRegex.test(loginData.identifier)) {
      setError("Please enter a valid email address.");
      return false;
    }
    setError("");
    return true;
  };

  const handleLogin = async (e) => {
    e.preventDefault();
    if (!validateLogin()) return;
    try {
      const response = await API.post("/login", loginData);
      if (response.data.error) {
        setError(response.data.error);
      } else {
        setRole(response.data.role);
        navigate("/");
      }
    } catch (error) {
      setError("An error occurred during login. Please try again.");
    }
  };
}

export default Login;
```

```

try {

  const res = await API.post("/auth/login", { ...loginData, role });

  localStorage.setItem("userToken", res.data.token);
  localStorage.setItem("userRole", res.data.user.role);

  if (res.data.user.role === "student") navigate("/student/dashboard");
  else if (res.data.user.role === "university") navigate("/university");
  else if (res.data.user.role === "company") navigate("/company/dashboard");

} catch (err) {
  setError(err.response?.data?.message || "Login failed");
}

};

const roles = ["student", "university", "company"];


return (
  <div className="min-h-screen bg-white flex items-center justify-center px-4 py-12">
    <div className="w-full max-w-7xl grid md:grid-cols-2 gap-10 items-center">
      {/* LEFT CONTENT */}
      <div className="space-y-6 md:pr-8">
        <h1 className="text-4xl font-bold leading-tight text-gray-900">
          Welcome Back to <span className="text-blue-600">StudentConnect</span>
        </h1>
        <p className="text-gray-600 max-w-md">
          Login to access your dashboard, manage your profile, and continue your journey with our AI-powered tools.
        </p>
        <ul className="space-y-2 text-gray-700">
          <li>✓ View and apply for internships</li>
          <li>✓ Update and showcase your skills</li>
          <li>✓ Track your career roadmap</li>
        </ul>
      </div>
    </div>
  </div>
);

```

```

        </ul>
    </div>

/* RIGHT FORM */

<div className="bg-white border border-gray-200 p-8 rounded-xl shadow-md w-full">
    /* Role Selector */
    <div className="flex gap-3 mb-6">
        {roles.map((r) => (
            <button
                key={r}
                onClick={() => setRole(r)}
                className={`px-5 py-2 rounded-full text-sm font-medium transition ${role === r ? "bg-blue-600 text-white" : "bg-gray-100 text-gray-700 hover:bg-gray-200"}`}
            >
                {r.charAt(0).toUpperCase() + r.slice(1)}
            </button>
        )));
    </div>

<form onSubmit={handleLogin} className="space-y-5">
    <input
        type="text"
        name="identifier"
        placeholder="Email or Username"
        className="w-full px-4 py-3 border border-gray-300 rounded-md focus:outline-none focus:border-blue-500"
        value={loginData.identifier}
        onChange={handleChange}
    />
    <div className="relative">

```

```
<input
  type={showPass ? "text" : "password"}
  name="password"
  placeholder="Password"
  className="w-full px-4 py-3 border border-gray-300 rounded-md focus:outline-none focus:border-blue-500"
  value={loginData.password}
  onChange={handleChange}
/>

<span
  onClick={() => setShowPass((prev) => !prev)}
  className="absolute right-3 top-3 text-sm text-blue-600 cursor-pointer"
>
  {showPass ? "Hide" : "Show"}
</span>
</div>

{error && <p className="text-red-600 text-sm">{error}</p>}

<button
  type="submit"
  className="w-full py-3 bg-blue-600 hover:bg-blue-700 transition font-semibold text-white rounded-md"
>
  Login
</button>
</form>

<p className="mt-6 text-center text-sm text-gray-500">
  Don't have an account?{" "}
<a href="/register" className="text-blue-600 font-semibold hover:underline">
  Register here
</a>
</p>
```

```

        </Link>
      </p>
    </div>
  </div>
</div>
);

}

export default Login;

```

---

```

// src/pages/Pricing.jsx

import { useMemo, useState } from "react";
import { Link } from "react-router-dom";
import {
  Check,
  X,
  Sparkles,
  ShieldCheck,
  Users2,
  GraduationCap,
  Building2,
  Stars,
  ChevronDown,
  ChevronUp,
  ArrowRight,
  Briefcase,
  Trophy,
  Video,
  ClipboardList,
  Binary,
  Route as Routelcon,
  Bell,
  BadgeCheck
} from "lucide-react";

/* tiny UI helpers */
const Pill = ({ children, className = "" }) => (
  <span className={`inline-flex items-center gap-1 px-2.5 py-1 rounded-full text-xs font-medium ring-1 ${className}`}>
    {children}
  </span>
);
const Feature = ({ children }) => (
  <li className="flex items-start gap-2">
    <Check className="mt-0.5 text-emerald-600" size={16} />
    <span>{children}</span>
  </li>
)

```

```

);

const NoFeature = ({ children }) => (
  <li className="flex items-start gap-2 text-slate-500">
    <X className="mt-0.5" size={16} />
    <span>{children}</span>
  </li>
);

export default function Pricing() {
  const [cycle, setCycle] = useState("monthly"); // "monthly" | "annual"
  const [faqOpen, setFaqOpen] = useState(null);
  const price = (m, y) => (cycle === "monthly" ? m : y);

  /* Plans reflect your current build + room to grow */
  const tiers = useMemo(
    () => [
      {
        key: "student_free",
        label: "Student — Free",
        icon: <GraduationCap className="text-sky-600" size={18} />,
        badge: "Best for starters",
        color: "from-sky-50 to-indigo-50",
        price: "₹0",
        period: "/forever",
        cta: { to: "/signup?role=student&plan=free", text: "Get Started" },
        features: [
          "Profile & basic networking",
          "Apply to limited jobs / month",
          "Skill tests (limited)",
          "Join public hackathons",
          "Join webinars (viewer)",
        ],
      }
    ]
  );
}

```

```
        "Notifications & badges (basic)",  
    ],  
    missing: [  
        "AI résumé optimizer",  
        "Interview simulator",  
        "Advanced analytics",  
    ],  
},  
{  
    key: "student_pro",  
    label: "Student — Pro",  
    icon: <Stars className="text-indigo-600" size={18} />,  
    badge: "Most popular",  
    color: "from-indigo-50 to-violet-50",  
    price: `₹${price(299, 249)}`,  
    period: cycle === "monthly" ? "/month" : "/month (billed annually)",  
    highlight: true,  
    cta: { to: "/checkout?role=student&plan=pro", text: "Upgrade to Pro" },  
    features: [  
        "Unlimited job applications",  
        "AI résumé & portfolio optimization",  
        "Interview simulator & tips",  
        "Advanced profile analytics",  
        "Priority support",  
    ],  
    missing: [],  
},  
{  
    key: "recruiter",  
    label: "Recruiter / Company",  
    icon: <Users2 className="text-emerald-600" size={18} />,
```

```
badge: "For hiring teams",
color: "from-emerald-50 to-teal-50",
price: `₹${price(3999, 3499)}`,
period: cycle === "monthly" ? "/month" : "/month (billed annually)",
cta: { to: "/company/subscribe?plan=recruiter", text: "Start Hiring" },
features: [
  "Unlimited job postings",
  "Smart candidate screening (resume/test/fit)",
  "Interview scheduling + host join link",
  "Talent search & filters",
  "Company branding page",
  "Webinars (Studio host)",
  "Hackathons: create, judge, leaderboard",
],
missing: [],
},
{
key: "university",
label: "University Suite",
icon: <Building2 className="text-rose-600" size={18} />,
badge: "For placement cells",
color: "from-rose-50 to-orange-50",
price: `₹${price(14999, 12999)}`,
period: cycle === "monthly" ? "/month" : "/month (billed annually)",
cta: { to: "/contact?type=university", text: "Talk to Sales" },
features: [
  "Cohort analytics & dashboards",
  "Skill-gap & curriculum insights",
  "Placement tracking & reports",
  "Private talent network",
  "Multi-admin roles & SSO",
]
```

```

        ],
        note: "Custom onboarding & SLAs available.",
        missing: [],
    },
]),
[cycle]
);

const addons = [
{
    title: "AI Tools Pack",
    price: `₹${price(199, 149)}`,
    period: cycle === "monthly" ? "/month per user" : "/month per user (annual)",
    features: ["Résumé rewrite & score", "Cover-letter generator", "Job match predictor"],
},
{
    title: "Event & Hackathon Sponsorship",
    price: "Custom",
    period: "",
    features: ["Branding across events", "Talent leads & insights", "Judging & stage time"],
},
];
/* Comparison focuses on features you already have live */
const compare = [
{ label: "Unlimited applications", free: false, student: true, recruiter: "—", uni: "—" },
{ label: "Skill tests & progress", free: true, student: true, recruiter: "—", uni: "—" },
{ label: "Career roadmap", free: true, student: true, recruiter: "—", uni: "—" },
{ label: "Interview scheduling (join links)", free: "—", student: "—", recruiter: true, uni: true },
{ label: "Smart screening (resume/test/fit)", free: "—", student: "—", recruiter: true, uni: true },
{ label: "Talent search", free: "—", student: "—", recruiter: true, uni: true },

```

```
{ label: "Webinars (viewer/studio)", free: "viewer", student: "viewer", recruiter: "studio", uni: "studio" },  
{ label: "Hackathons (join/create/judge)", free: "join", student: "join", recruiter: "create/judge", uni: "create/judge" },  
{ label: "Cohort analytics", free: "—", student: "—", recruiter: "—", uni: true },  
];  
  
const faqs = [  
{  
q: "Does Student Pro include AI résumé tools and interview simulator?",  
a: "Yes. Student Pro unlocks AI résumé optimization, portfolio suggestions, and an interview simulator with feedback.",  
},  
{  
q: "How do company interviews work?",  
a: "Recruiters schedule interviews (stage, time, duration). Both company and student get join links; company joins the Studio, student joins the viewer.",  
},  
{  
q: "What's included in hackathons?",  
a: "Registration, team name, submissions (repo/demo/file), judging with scores & feedback, and leaderboard publishing are all supported.",  
},  
{  
q: "Can universities get custom onboarding and SLAs?",  
a: "Yes. The University Suite includes custom onboarding, SSO, cohort analytics, and optional SLAs.",  
},  
{  
q: "Is there a free trial?",  
a: "We offer a 7-day trial for Student Pro with limited AI credits. Companies can request a pilot via Sales.",  
},
```

```

    ];

const modules = [
  { to: "/student/jobs", icon: <Briefcase size={18} />, title: "Jobs & Applications", desc: "Modern board, apply/withdraw, status, résumé links." },
  { to: "/student/status", icon: <ClipboardList size={18} />, title: "Application Tracking", desc: "Live statuses, interview window & join buttons." },
  { to: "/student/hackathons", icon: <Trophy size={18} />, title: "Hackathons", desc: "Register, submit, see feedback & leaderboard." },
  { to: "/student/webinars", icon: <Video size={18} />, title: "Webinars", desc: "Attend live sessions; companies host via Studio." },
  { to: "/student/testskill", icon: <Binary size={18} />, title: "Skill Tests", desc: "Timed MCQs, overview grid, auto-scored, progress." },
  { to: "/student/roadmap", icon: <Routelcon size={18} />, title: "Career Roadmap", desc: "AI-guided steps, topics, resources per skill." },
  { to: "/student/badges", icon: <BadgeCheck size={18} />, title: "Badges", desc: "Earn achievements; show credibility on profile." },
  { to: "/student/dashboard", icon: <Bell size={18} />, title: "Notifications", desc: "Clean, minimal updates across modules." },
];

return (
  <div className="min-h-screen bg-gradient-to-b from-white to-slate-50">
    {/* Hero */}
    <header className="border-b bg-white/70 backdrop-blur">
      <div className="max-w-7xl mx-auto px-4 py-12 md:py-16">
        <div className="flex items-start justify-between gap-6 flex-col md:flex-row">
          <div className="max-w-2xl">
            <div className="inline-flex items-center gap-2 px-3 py-1 rounded-full bg-indigo-50 text-indigo-700 ring-1 ring-indigo-200 text-xs font-medium">
              <Sparkles size={14} /> Built around your live features
            </div>
          <h1 className="mt-3 text-3xl md:text-4xl font-bold tracking-tight text-slate-900">
            Pricing that scales with students, recruiters, and universities
          </h1>
        </div>
      </div>
    </header>
  </div>
);

```

```
</h1>

<p className="mt-3 text-slate-600">
  Freemium for learners, Pro tools for career acceleration, and powerful suites for hiring
  teams

  & universities. Clean, pastel, responsive — ready for today's UX.

</p>

<div className="mt-5 inline-flex items-center gap-2 text-sm text-slate-600">
  <ShieldCheck className="text-emerald-600" size={16} />

  14-day refund on first purchase • Cancel anytime

</div>

</div>

/* Billing cycle toggle */

<div className="shrink-0 rounded-xl border bg-white p-2 ring-1 ring-slate-200">
  <div className="flex items-center">
    <button
      onClick={() => setCycle("monthly")}
      className={`${`px-4 py-2 rounded-lg text-sm font-medium ${cycle === "monthly" ? "bg-slate-900 text-white" : "text-slate-700"}`}`}
    >
      Monthly
    </button>
    <button
      onClick={() => setCycle("annual")}
      className={`${`ml-1 px-4 py-2 rounded-lg text-sm font-medium ${cycle === "annual" ? "bg-slate-900 text-white" : "text-slate-700"}`}`}
    >
      Annual <span className="ml-1 text-emerald-600 font-semibold">(Save ~15%)</span>
    </button>
  </div>
</div>
</div>
```

```

        </div>
    </header>

    <main className="max-w-7xl mx-auto px-4 py-10 space-y-12">
        {/* Your live modules (quick links) */}
        <section className="space-y-3">
            <h2 className="text-xl font-semibold">Everything live in your build</h2>
            <div className="grid sm:grid-cols-2 lg:grid-cols-4 gap-4">
                {modules.map((m, i) => (
                    <Link key={i} to={m.to} className="group rounded-2xl border bg-white/80 backdrop-blur p-4
                    hover:shadow-sm transition">
                        <div className="flex items-start justify-between">
                            <div className="flex items-center gap-2">
                                <div className="h-8 w-8 rounded-xl bg-white border grid place-items-center">{m.icon}</div>
                                <div className="font-semibold">{m.title}</div>
                            </div>
                            <ArrowRight size={16} className="opacity-0 group-hover:opacity-100 transition" />
                        </div>
                        <p className="mt-2 text-sm text-slate-600">{m.desc}</p>
                    </Link>
                )))
            </div>
        </section>

        {/* Tiers */}
        <section className="grid md:grid-cols-2 lg:grid-cols-4 gap-4">
            {tiers.map((t) => (
                <div key={t.key} className={`group relative rounded-2xl border bg-gradient-to-b ${t.color} p-1
                hover:shadow-sm transition`}>
                    <div className="rounded-2xl bg-white/80 backdrop-blur p-5 h-full">
                        <div className="flex items-center justify-between">

```

```

<div className="flex items-center gap-2">
  <div className="h-9 w-9 rounded-xl bg-white border grid place-items-center">{t.icon}</div>
  <div className="font-semibold">{t.label}</div>
</div>
{t.badge && <Pill className="bg-indigo-50 text-indigo-700 ring-indigo-200">{t.badge}</Pill>}
</div>

<div className="mt-4">
  <div className="text-3xl font-bold">{t.price}</div>
  <div className="text-xs text-slate-600">{t.period}</div>
  {t.note && <div className="text-xs text-slate-500 mt-1">{t.note}</div>}
</div>

<ul className="mt-4 space-y-2 text-sm">
  {t.features.map((f, i) => <Feature key={i}>{f}</Feature>)}
  {t.missing?.map((m, i) => <NoFeature key={`m-${i}`}>{m}</NoFeature>)}
</ul>

<Link
  to={t.cta.to}
  className={`${`mt-5 inline-flex w-full items-center justify-center gap-2 rounded-lg px-4 py-2`}`}
  style={{`background-color: ${t.highlight ? "bg-slate-900 text-white hover:bg-black" : "bg-white border hover:bg-slate-50"}`}}
>
  {t.cta.text} <ArrowRight size={16} />
</Link>
</div>
</div>
)})}

```

```
</section>

{/* Add-ons */}

<section className="grid lg:grid-cols-3 gap-4">
  <div className="lg:col-span-1">
    <h2 className="text-xl font-semibold">Add-ons</h2>
    <p className="text-slate-600 text-sm mt-1">Enhance any plan with AI tools and sponsored events.</p>
  </div>
  <div className="lg:col-span-2 grid md:grid-cols-2 gap-4">
    {[...addons,
      {
        title: "Custom Integrations",
        price: "Quote",
        period: "",
        features: ["SSO/SAML", "ATS/HRIS bridges", "Data exports & APIs"],
      },
    ].map((a, i) => (
      <div key={i} className="rounded-2xl border bg-white p-5">
        <div className="flex items-center justify-between">
          <div className="font-semibold">{a.title}</div>
          <div className="text-right">
            <div className="text-xl font-bold">{a.price}</div>
            <div className="text-xs text-slate-600">{a.period}</div>
          </div>
        </div>
      </div>
      <ul className="mt-3 space-y-2 text-sm">
        {a.features.map((f, k) => <Feature key={k}>{f}</Feature>)}
      </ul>
      <div className="mt-4">
```

```

<Link
  to={i === 0 ? "/checkout?addon=ai_tools" : "/contact?type=sponsorship"}
  className="inline-flex items-center gap-2 rounded-lg border px-3 py-1.5 text-sm
  hover:bg-slate-50"
>
  Learn more <ArrowRight size={16} />
</Link>
</div>
</div>
))}

</div>
</section>

```

```

{/* Comparison table */}
<section className="space-y-3">
  <h2 className="text-xl font-semibold">Compare plans</h2>
  <div className="overflow-x-auto">
    <table className="w-full text-sm overflow-hidden rounded-2xl border border-slate-200">
      <thead className="bg-slate-50/80">
        <tr>
          <th className="p-3 text-left w-1/3">Features</th>
          <th className="p-3 text-center">Student Free</th>
          <th className="p-3 text-center">Student Pro</th>
          <th className="p-3 text-center">Recruiter</th>
          <th className="p-3 text-center">University</th>
        </tr>
      </thead>
      <tbody>
        {compare.map((row, i) => (
          <tr key={i} className="border-t">
            <td className="p-3">{row.label}</td>

```

```

{[row.free, row.student, row.recruiter, row.uni].map((v, idx) => (
  <td key={idx} className="p-3 text-center">
    {v === true ? (
      <Check className="inline text-emerald-600" size={18} />
    ) : v === "—" ? (
      <span className="text-slate-400">—</span>
    ) : v === "viewer" ? (
      <span className="text-slate-700">Viewer</span>
    ) : v === "studio" ? (
      <span className="text-slate-700">Studio</span>
    ) : v === "join" ? (
      <span className="text-slate-700">Join</span>
    ) : v === "create/judge" ? (
      <span className="text-slate-700">Create/Judge</span>
    ) : (
      <X className="inline text-slate-400" size={18} />
    )
  )})
  </td>
))
</tr>
))
</tbody>
</table>
</div>
</section>

/* FAQ */

<section className="grid lg:grid-cols-3 gap-4">
  <div>
    <h2 className="text-xl font-semibold">FAQs</h2>
    <p className="text-slate-600 text-sm mt-1">

```

Everything you need to know before choosing a plan.

```
</p>
</div>

<div className="lg:col-span-2 space-y-2">
  {faqs.map((f, i) => {
    const open = faqOpen === i;
    return (
      <div key={i} className="rounded-2xl border bg-white">
        <button
          onClick={() => setFaqOpen(open ? null : i)}
          className="w-full flex items-center justify-between px-4 py-3"
        >
          <span className="text-sm font-medium text-left">{f.q}</span>
          {open ? <ChevronUp size={18} /> : <ChevronDown size={18} />}
        </button>
        {open && <div className="px-4 pb-4 text-sm text-slate-600">{f.a}</div>}
      </div>
    );
  })}
</div>
</section>

/* CTA */

<section className="rounded-2xl border bg-white p-6 flex flex-col md:flex-row items-center justify-between gap-4">
  <div>
    <h3 className="text-lg font-semibold">Ready to list your first role or join a hackathon?</h3>
    <p className="text-slate-600 text-sm">Students grow careers. Recruiters hire faster. Universities get better outcomes.</p>
  </div>
  <div className="flex items-center gap-2">
```

```

    <Link to="/signup?role=student&plan=pro" className="rounded-lg bg-slate-900 text-white px-4 py-2 text-sm">
      Try Pro free
    </Link>

    <Link to="/company/subscribe?plan=recuriter" className="rounded-lg border px-4 py-2 text-sm hover:bg-slate-50">
      Start Hiring
    </Link>

    <Link to="/contact?type=university" className="rounded-lg border px-4 py-2 text-sm hover:bg-slate-50">
      Talk to Sales
    </Link>
  </div>
</section>

<p className="text-center text-xs text-slate-500">
  Prices shown are indicative. Currency and tax are finalized at checkout.
</p>
</main>
</div>
);
}


```

---

```

import { useState, useEffect } from "react";
import { Link, useNavigate } from "react-router-dom";
import debounce from "lodash.debounce";
import API from "../api/axios";

function Register() {
  const navigate = useNavigate();
  const [formData, setFormData] = useState({
    // common

```

```
name: "",  
username: "",  
email: "",  
password: "",  
confirmPassword: "",  
role: "student",  
phone: "",  
// student  
university: "",  
education: "UG",  
skills: "",  
// company  
companyName: "",  
companyWebsite: "",  
companySize: "1-10",  
companyContactEmail: "",  
companyContactPhone: "",  
companyLocations: "",  
companyDomains: "",  
});  
  
const [errors, setErrors] = useState({});  
const [usernameAvailable, setUsernameAvailable] = useState(null);  
  
useEffect(() => {  
  if (formData.username.length < 4) {  
    setUsernameAvailable(null);  
    return;  
  }  
  const checkAvailability = debounce(async () => {  
    try {
```

```

    const res = await API.get(`/auth/check-username?username=${formData.username}`);
    setUsernameAvailable(res.data.available);
} catch {
    setUsernameAvailable(null);
}
}, 500);
checkAvailability();
return () => checkAvailability.cancel();
}, [formData.username]);

const validateForm = () => {
    const newErrors = {};
    const emailRegex = /^[^@\s]+@[^\s]+\.[^\s@]+$/;
    const passRegex = /^(?=.*[A-Z])(?=.*![#@%$^&*])(?=.*\d).{8,}$/;

    if (!emailRegex.test(formData.email)) newErrors.email = "Invalid email format";
    if (!passRegex.test(formData.password))
        newErrors.password = "Password must be 8+ chars, 1 number, 1 symbol, 1 uppercase";
    if (formData.password !== formData.confirmPassword)
        newErrors.confirmPassword = "Passwords do not match";
    if (usernameAvailable === false) newErrors.username = "Username already taken";

    // minimal company field checks
    if (formData.role === "company") {
        if (!formData.companyName.trim()) newErrors.companyName = "Company name is required";
        if (!formData.companyContactEmail.trim()) newErrors.companyContactEmail = "Company contact email is required";
    }

    setErrors(newErrors);
    return Object.keys(newErrors).length === 0;
}

```

```
};

const handleChange = (e) => {
  setFormData((prev) => ({ ...prev, [e.target.name]: e.target.value }));
};

const handleRegister = async (e) => {
  e.preventDefault();
  if (!validateForm()) return;
  try {
    // 1) Create user
    const res = await API.post("/auth/register", {
      name: formData.name,
      username: formData.username,
      email: formData.email,
      password: formData.password,
      role: formData.role,
      phone: formData.phone,
      // student extras (backend can ignore for non-students)
      university: formData.university,
      education: formData.education,
      skills: formData.skills,
    });
    // 2) Save auth
    localStorage.setItem("userToken", res.data.token);
    localStorage.setItem("userRole", res.data.user.role);
    // 3) If company, bootstrap their Company Profile immediately
    if (res.data.user.role === "company") {
      try {
```

```

await API.post("/company/profile", {
  name: formData.companyName || res.data.user.name,
  website: formData.companyWebsite || "",
  logoUrl: "", // optional now; can be edited later
  description: "",
  size: formData.companySize || "1-10",
  locations: (formData.companyLocations || "")  

    .split(",")
    .map((v) => v.trim())
    .filter(Boolean),
  domains: (formData.companyDomains || "")  

    .split(",")
    .map((v) => v.trim())
    .filter(Boolean),
  contactEmail: formData.companyContactEmail || formData.email,
  contactPhone: formData.companyContactPhone || formData.phone,
});

} catch (e2) {
  // not fatal — they can finish profile later
  // console.warn("Auto-create company profile failed:", e2?.response?.data || e2.message);
}

}

// 4) Redirect based on role
if (res.data.user.role === "student") navigate("/student/dashboard");
else if (res.data.user.role === "university") navigate("/university");
else if (res.data.user.role === "company") navigate("/company");
} catch (err) {
  alert(err.response?.data?.message || "Registration failed");
}
};


```

```

const roles = [
  { value: "student", label: "Student" },
  { value: "university", label: "University" },
  { value: "company", label: "Company" },
];

return (
  <div className="min-h-screen bg-white flex items-center justify-center px-4 py-12">
    <div className="w-full max-w-7xl grid md:grid-cols-2 gap-10 items-center">
      {/* LEFT CONTENT */}
      <div className="space-y-6 md:pr-8">
        <h1 className="text-4xl font-bold leading-tight text-gray-900">
          Create Your <span className="text-blue-600">StudentConnect</span> Account
        </h1>
        <p className="text-gray-600 max-w-md">
          Join our AI-powered platform to showcase your skills, connect with companies,
          and get real career opportunities. It's free and only takes a minute to sign up.
        </p>
        <ul className="space-y-2 text-gray-700">
          <li>✓ Build your verified skill profile</li>
          <li>✓ Connect with universities & companies</li>
          <li>✓ Access internships & job opportunities</li>
        </ul>
      </div>

      {/* RIGHT FORM */}
      <div className="bg-white border border-gray-200 p-8 rounded-xl shadow-md w-full">
        {/* Role Selector */}
        <div className="flex gap-3 mb-6">
          {roles.map((role) => (

```

```
<button
  key={role.value}
  type="button"
  onClick={() => setFormData((prev) => ({ ...prev, role: role.value }))}
  className={`px-5 py-2 rounded-full text-sm font-medium transition ${{
    formData.role === role.value
    ? "bg-blue-600 text-white"
    : "bg-gray-100 text-gray-700 hover:bg-gray-200"
  }}`}
>
  {role.label}
</button>
)})}
</div>
```

```
<form onSubmit={handleRegister} className="space-y-5">
  {/* Common fields */}
  <div className="grid md:grid-cols-2 gap-4">
    <input
      type="text"
      name="name"
      placeholder="Full Name"
      required
      className="w-full px-4 py-3 border border-gray-300 rounded-md focus:outline-none focus:border-blue-500"
      value={formData.name}
      onChange={handleChange}
    />
    <div>
      <input
        type="text"
```

```
        name="username"
        placeholder="Username"
        required
        className={`w-full px-4 py-3 border ${(
          usernameAvailable === true
            ? "border-green-500"
            : usernameAvailable === false
            ? "border-red-500"
            : "border-gray-300"
        )} rounded-md focus:outline-none focus:border-blue-500`}
        value={formData.username}
        onChange={handleChange}
      />
      {formData.username.length >= 4 && (
        <p
          className={`text-xs mt-1 ${(
            usernameAvailable === true
              ? "text-green-600"
              : usernameAvailable === false
              ? "text-red-600"
              : "text-yellow-600"
          )}`}
        >
          {usernameAvailable === true
            ? "✓ Available"
            : usernameAvailable === false
            ? "✗ Taken"
            : "⌚ Checking..."}
        </p>
      )}
    </div>
```

```
</div>

<div className="grid md:grid-cols-2 gap-4">
  <input
    type="email"
    name="email"
    placeholder="Email Address"
    required
    className={`w-full px-4 py-3 border ${(
      errors.email ? "border-red-500" : "border-gray-300"
    )} rounded-md focus:outline-none focus:border-blue-500`}
    value={formData.email}
    onChange={handleChange}
  />
  <input
    type="text"
    name="phone"
    placeholder="Phone Number"
    required
    className="w-full px-4 py-3 border border-gray-300 rounded-md focus:outline-none focus:border-blue-500"
    value={formData.phone}
    onChange={handleChange}
  />
</div>

<div className="grid md:grid-cols-2 gap-4">
  <input
    type="password"
    name="password"
    placeholder="Password"
  />
```

```
required

className={`w-full px-4 py-3 border ${

  errors.password ? "border-red-500" : "border-gray-300"

} rounded-md focus:outline-none focus:border-blue-500`}

value={formData.password}

onChange={handleChange}

/>>

<input

  type="password"

  name="confirmPassword"

  placeholder="Confirm Password"

  required

  className={`w-full px-4 py-3 border ${

    errors.confirmPassword ? "border-red-500" : "border-gray-300"

} rounded-md focus:outline-none focus:border-blue-500`}

  value={formData.confirmPassword}

  onChange={handleChange}

/>

</div>

/* Student-only fields */

{formData.role === "student" && (

  <>

    <input

      type="text"

      name="university"

      placeholder="University / College Name"

      className="w-full px-4 py-3 border border-gray-300 rounded-md focus:outline-none

      focus:border-blue-500"

      value={formData.university}

      onChange={handleChange}


```

```

        />

        <select
            name="education"
            className="w-full px-4 py-3 border border-gray-300 rounded-md focus:outline-none
focus:border-blue-500"
            value={formData.education}
            onChange={handleChange}
        >
            <option value="UG">Undergraduate</option>
            <option value="PG">Postgraduate</option>
            <option value="Diploma">Diploma</option>
            <option value="Other">Other</option>
        </select>

        <input
            type="text"
            name="skills"
            placeholder="Skills (comma separated)"
            className="w-full px-4 py-3 border border-gray-300 rounded-md focus:outline-none
focus:border-blue-500"
            value={formData.skills}
            onChange={handleChange}
        />
    </>
}

/* Company-only fields */
{formData.role === "company" && (
    <div className="space-y-4 border-t pt-4">
        <div className="grid md:grid-cols-2 gap-4">
            <div>
                <input
                    type="text"

```

```
        name="companyName"
        placeholder="Company Name"
        className={`w-full px-4 py-3 border ${(
          errors.companyName ? "border-red-500" : "border-gray-300"
        ) rounded-md focus:outline-none focus:border-blue-500`}
        value={formData.companyName}
        onChange={handleChange}
      />
      {errors.companyName && (
        <p className="text-xs text-red-600 mt-1">{errors.companyName}</p>
      )}
    </div>
    <select
      name="companySize"
      className="w-full px-4 py-3 border border-gray-300 rounded-md focus:outline-none focus:border-blue-500"
      value={formData.companySize}
      onChange={handleChange}
    >
      {[{"1-10", "11-50", "51-200", "201-500", "500+"}.map(s => (
        <option key={s} value={s}>{s} employees</option>
      ))}
    </select>
  </div>

<div className="grid md:grid-cols-2 gap-4">
  <input
    type="text"
    name="companyWebsite"
    placeholder="Company Website"
    className="w-full px-4 py-3 border border-gray-300 rounded-md focus:outline-none focus:border-blue-500"
  >
</div>
```

```
        value={formData.companyWebsite}
        onChange={handleChange}
      />
    <div>
      <input
        type="email"
        name="companyContactEmail"
        placeholder="Company Contact Email"
        className={`w-full px-4 py-3 border ${(
          errors.companyContactEmail ? "border-red-500" : "border-gray-300"
        ) rounded-md focus:outline-none focus:border-blue-500`}
        value={formData.companyContactEmail}
        onChange={handleChange}
      />
      {errors.companyContactEmail && (
        <p className="text-xs text-red-600 mt-1">{errors.companyContactEmail}</p>
      )}
    </div>
  </div>

<div className="grid md:grid-cols-2 gap-4">
  <input
    type="text"
    name="companyContactPhone"
    placeholder="Company Contact Phone"
    className="w-full px-4 py-3 border border-gray-300 rounded-md focus:outline-none focus:border-blue-500"
    value={formData.companyContactPhone}
    onChange={handleChange}
  />
  <input
```

```
        type="text"
        name="companyLocations"
        placeholder="Locations (comma separated)"
        className="w-full px-4 py-3 border border-gray-300 rounded-md focus:outline-none
focus:border-blue-500"
        value={formData.companyLocations}
        onChange={handleChange}
      />
    </div>

    <input
      type="text"
      name="companyDomains"
      placeholder="Domains / Industry (comma separated)"
      className="w-full px-4 py-3 border border-gray-300 rounded-md focus:outline-none
focus:border-blue-500"
      value={formData.companyDomains}
      onChange={handleChange}
    />
  </div>
}

<button
  type="submit"
  className="w-full py-3 bg-blue-600 hover:bg-blue-700 transition font-semibold text-white
rounded-md"
>
  Sign Up
</button>
</form>

<p className="mt-6 text-center text-sm text-gray-500">
```

```
    Already have an account?" "}

    <Link to="/login" className="text-blue-600 font-semibold hover:underline">
        Login here
    </Link>
</p>
</div>
</div>
</div>

);

}

export default Register;
```

---

```
// App.jsx

import { BrowserRouter as Router, Routes, Route, Navigate } from "react-router-dom";
import { ErrorBoundary } from "react-error-boundary";

import Home from "./pages/Home";
import Login from "./pages/Login";
import Register from "./pages/Register";

import StudentLayout from "./layouts/StudentLayout";
import StudentDashboard from "./pages/student/StudentDashboard";
import CareerRoadmapPage from "./pages/student/CareerRoadmapPage";
import InternshipPage from "./pages/student/InternshipPage";
import StudentProfile from "./pages/student/StudentProfile";
import TestSkill from "./pages/student/TestSkill";
import SkillProgress from "./pages/student/SkillProgress";
import JobsBoard from "./pages/student/JobsBoard";
import StudentWebinars from "./pages/student/StudentWebinars";
import StudentWebinarViewer from "./pages/student/StudentWebinarViewer";
```

```
import Admin from "./pages/admin/Admin";
import UniversityDashboard from "./pages/university/UniversityDashboard";
import CompanyDashboard from "./pages/company/CompanyDashboard";
import WebinarStudio from "./pages/company/WebinarStudio";

import RequireRole from "./components/RequireRole";
import StudentInterviewRoom from "./pages/student/StudentInterviewRoom";
import CompanyInterviewRoom from "./pages/company/CompanyInterviewRoom";
// import MyApplicationsWidget from "./pages/student/MyApplicationsWidget";
import MyApplications from "./pages/student/MyApplications";
import HackathonManager from "./pages/company/HackathonManager";
import Hackathons from "./pages/student/Hackathons";
import HackathonDetail from "./pages/student/HackathonDetail";
import Pricing from "./pages/Pricing";
import AIGuidance from "./pages/student/AIGuidance";
import AdminLogin from "./pages/admin/AdminLogin";
import AdminRoute from "./components/AdminRoute"; // the guard above

function ErrorFallback({ error, resetErrorBoundary }) {
  return (
    <div role="alert" className="text-center py-20 px-4">
      <h2 className="text-2xl font-bold text-red-600 mb-2">Something went wrong!</h2>
      <pre className="text-gray-800 mb-4">{error.message}</pre>
      <button onClick={resetErrorBoundary} className="bg-blue-600 text-white px-4 py-2 rounded">
        Try Again
      </button>
    </div>
  );
}

export default function App() {
```

```

return (
  <ErrorBoundary FallbackComponent={ErrorFallback}>
    <Router>
      <Routes>
        {/* Public */}
        <Route path="/" element={<Home />} />
        <Route path="/login" element={<Login />} />
        <Route path="/register" element={<Register />} />

        {/* Student (nested) */}
        // App.jsx — replace the student parent route

<Route
  path="/student"
  element={
    <RequireRole role="student">
      <StudentLayout />
    </RequireRole>
  }
>
  <Route index element={<StudentDashboard />} />
  <Route path="dashboard" element={<StudentDashboard />} />
  <Route path="roadmap" element={<CareerRoadmapPage />} />
  <Route path="internships" element={<InternshipPage />} />
  <Route path="profile" element={<StudentProfile />} />
  <Route path="testskill" element={<TestSkill />} />
  <Route path="progress" element={<SkillProgress />} />
  <Route path="jobs" element={<JobsBoard />} />
  <Route path="webinars" element={<StudentWebinars />} />
  <Route path="webinars/webinar/:roomId" element={<StudentWebinarViewer />} />
  <Route path="interview/:roomId" element={<StudentInterviewRoom />} />
  <Route path="status" element={<MyApplications />} />

```

```

<Route path="hackathons" element={<Hackathons />} />
<Route path="hackathons/:id" element={<HackathonDetail />} />
<Route path="guidance" element={<AIGuidance />} />

</Route>
<Route path="/pricing" element={<Pricing />} />

    <Route path="/admin/login" element={<AdminLogin />} />

<Route
  path="/admin"
  element={
    <AdminRoute>
      <Admin />
    </AdminRoute>
  }
/>

{/* University */}
<Route path="/university" element={<UniversityDashboard />} />

{/* Company (guarded) */}
<Route path="/company" element={<Navigate to="/company/dashboard" replace />} />
<Route
  path="/company/dashboard"
  element={
    <RequireRole role="company">
      <CompanyDashboard />
    </RequireRole>
  }
/>
<Route

```

```

    path="/company/webinar/:roomId"
    element={

      <RequireRole role="company">
        <WebinarStudio />
      </RequireRole>
    }
  />

  <Route path="/company/interview/:roomId" element={<RequireRole
role="company"><CompanyInterviewRoom /></RequireRole>} />

  <Route path="/company/hackathons/:id" element={<RequireRole
role="company"><HackathonManager /></RequireRole>} />

  /* 404 fallback */

  <Route path="*" element={<Navigate to="/" replace />} />
</Routes>
</Router>
</ErrorBoundary>
);

}

```

#### SERVER:

```

const mongoose = require("mongoose");

const connectDB = async () => {
  try {
    await mongoose.connect(process.env.MONGO_URI);
    console.log("✅ MongoDB Atlas connected");
  } catch (err) {
    console.error("❌ MongoDB connection error:", err.message);
    process.exit(1);
  }
};

```

```
module.exports = connectDB;



---



```
const jwt = require("jsonwebtoken");
const bcrypt = require("bcryptjs");
const User = require("../models/User");

const sign = (user) =>
  jwt.sign({ id: user._id, role: user.role }, process.env.JWT_SECRET, { expiresIn: "7d" });

exports.registerUser = async (req, res) => {
  try {
    const { name, username, email, password, phone, role, university, education, skills } = req.body;

    const exists = await User.findOne({ $or: [{ email: email.toLowerCase() }, { username: username.toLowerCase() }] });
    if (exists) return res.status(409).json({ message: "User already exists" });

    const hashed = await bcrypt.hash(password, 10);
    const user = await User.create({
      name,
      username: username.toLowerCase(),
      email: email.toLowerCase(),
      password: hashed,
      phone,
      role: role || "student",
      university: university || "",
      education: education || "UG",
      skills: (skills || "").split(",").map(s => s.trim()).filter(Boolean),
    });
  }
}
```


```

```

const token = sign(user);

res.json({ token, user: { id: user._id, role: user.role, name: user.name } });

} catch (e) {
    console.error(e);
    res.status(500).json({ message: "Registration failed" });
}

};

exports.loginUser = async (req, res) => {

try {

    const { identifier, password, role } = req.body;

    const query = identifier.includes("@")
        ? { email: identifier.toLowerCase() }
        : { username: identifier.toLowerCase() };

    const user = await User.findOne(query);

    if (!user) return res.status(400).json({ message: "Invalid credentials" });

    if (role && role !== user.role) return res.status(403).json({ message: "Role mismatch" });

    const ok = await bcrypt.compare(password, user.password);

    if (!ok) return res.status(400).json({ message: "Invalid credentials" });

    const token = sign(user);

    res.json({ token, user: { id: user._id, role: user.role, name: user.name } });

} catch (e) {
    console.error(e);
    res.status(500).json({ message: "Login failed" });
}

};



---


const Skill = require("../models/Skill");

```

```

const SkillTestResult = require("../models/SkillTestResult");
const Company = require("../models/Company");
const Job = require("../models/Job");
const Application = require("../models/Application");
const Hackathon = require("../models/Hackathon");
const Webinar = require("../models/Webinar");
const Partnership = require("../models/Partnership");
const User = require("../models/User");

// ----- Helpers

const avgScoreBySkillMap = async (studentId) => {
  const results = await SkillTestResult.find({ user: studentId })
    .populate("skill", "name")
    .lean();
  const map = {};
  results.forEach(r => {
    const name = r.skill?.name;
    if (!name) return;
    if (!map[name]) map[name] = [];
    map[name].push(r.score / (r.total || 10) * 10); // normalize to /10
  });
  const avg = {};
  Object.keys(map).forEach(k => {
    const arr = map[k];
    avg[k] = arr.reduce((a, b) => a + b, 0) / arr.length;
  });
  return avg;
};

const computeFit = (student, requiredSkills, avgScoreMap) => {
  // skill coverage

```

```
const studentSkillNames = (student.skills || []).map(s => s.toLowerCase());  
let coverage = 0;  
requiredSkills.forEach(rs => {  
  if (studentSkillNames.includes(rs.name?.toLowerCase())) coverage++;  
});  
const coveragePct = requiredSkills.length ? (coverage / requiredSkills.length) : 1;  
  
// test score avg for those skills  
const testScores = requiredSkills.map(rs => avgScoreMap[rs.name] || 0);  
const meanTest = testScores.length  
? testScores.reduce((a, b) => a + b, 0) / testScores.length  
: 0;  
  
// quick heuristic 0..100  
return Math.round((coveragePct * 0.6 + (meanTest / 10) * 0.4) * 100);  
};  
  
// ----- Profile  
exports.getCompanyProfile = async (req, res) => {  
  const owner = req.user.id;  
  const profile = await Company.findOne({ owner }).lean();  
  res.json(profile || null);  
};  
  
exports.upsertCompanyProfile = async (req, res) => {  
  const owner = req.user.id;  
  const { name, website, logoUrl, hq, about } = req.body;  
  const profile = await Company.findOneAndUpdate(  
    { owner },  
    { name, website, logoUrl, hq, about, owner },  
    { upsert: true, new: true }  
  );
```

```

};

res.json(profile);
};

// ----- Jobs

exports.listJobs = async (req, res) => {
  const owner = req.user.id;

  const company = await Company.findOne({ owner }).lean();
  if (!company) return res.json([]);

  const jobs = await Job.find({ company: company._id })
    .populate("skills", "name")
    .sort({ createdAt: -1 });

  res.json(jobs);
};

exports.createJob = async (req, res) => {
  const owner = req.user.id;

  const company = await Company.findOne({ owner });
  if (!company) return res.status(400).json({ message: "Create company profile first" });

  const { title, type, location, skills = [], description, minScore = 0, isFeatured = false } = req.body;

  const job = await Job.create({
    company: company._id,
    title, type, location, skills, description, minScore, isFeatured,
  });

  const populated = await job.populate("skills", "name");
  res.status(201).json(populated);
};

exports.toggleJobStatus = async (req, res) => {

```

```

const job = await Job.findById(req.params.id);

if (!job) return res.status(404).json({ message: "Job not found" });

job.status = job.status === "open" ? "closed" : "open";

await job.save();

res.json(job);

};

// ----- Applications (basic)

exports.listApplications = async (req, res) => {

const owner = req.user.id;

const company = await Company.findOne({ owner }).lean();

if (!company) return res.json([]);

const jobIds = (await Job.find({ company: company._id }).select("_id")).map(j => j._id);

const apps = await Application.find({ job: { $in: jobIds } })

.populate("job", "title")

.populate("student", "name email skills")

.sort({ createdAt: -1 })

.lean();

res.json(apps);

};

// ----- Talent Search (AI-ish ranking)

exports.searchTalent = async (req, res) => {

const { skillIds = [], minScore = 0 } = req.query;

const ids = Array.isArray(skillIds) ? skillIds : (skillIds ? [skillIds] : []);

const requiredSkills = await Skill.find({ _id: { $in: ids } }).lean();

// Pull student users with any skills overlap first (simple heuristic)

const students = await User.find({ role: "student" }).select("name email university skills").lean();

```

```

// prefetch skill average scores for each student

const cache = {};
const results = [];
for (const s of students) {
  if (!cache[s._id]) cache[s._id] = await avgScoreBySkillMap(s._id);

  const fit = computeFit(s, requiredSkills, cache[s._id]);

  // minScore filter: mean of required skills should be >= minScore
  const meanReq = requiredSkills.length
    ? requiredSkills
      .map(r => cache[s._id][r.name] || 0)
      .reduce((a, b) => a + b, 0) / requiredSkills.length
    : 0;

  if (meanReq >= Number(minScore)) {
    results.push({
      student: s,
      fitScore: fit,
      avgRequiredSkillScore: Math.round(meanReq * 10) / 10,
    });
  }
}

results.sort((a, b) => b.fitScore - a.fitScore);
res.json(results.slice(0, 100));
};

// ----- Hackathons

exports.listHackathons = async (req, res) => {
  const owner = req.user.id;
  const company = await Company.findOne({ owner }).lean();

```

```

if (!company) return res.json([]);

const items = await Hackathon.find({ company: company._id }).populate("skills", "name").sort({
createdAt: -1 });

res.json(items);

};

exports.createHackathon = async (req, res) => {
  const owner = req.user.id;
  const company = await Company.findOne({ owner });
  if (!company) return res.status(400).json({ message: "Create company profile first" });

  const { title, brief, skills = [], startAt, endAt, prize } = req.body;
  const item = await Hackathon.create({ company: company._id, title, brief, skills, startAt, endAt, prize });
  res.status(201).json(await item.populate("skills", "name"));
};

// ----- Webinars

exports.listWebinars = async (req, res) => {
  const owner = req.user.id;
  const company = await Company.findOne({ owner }).lean();
  if (!company) return res.json([]);
  const items = await Webinar.find({ company: company._id }).sort({ startsAt: 1 });
  res.json(items);
};

exports.createWebinar = async (req, res) => {
  const owner = req.user.id;
  const company = await Company.findOne({ owner });
  if (!company) return res.status(400).json({ message: "Create company profile first" });

  const { title, speaker, description, url, startsAt, durationMins } = req.body;

```

```

    const item = await Webinar.create({ company: company._id, title, speaker, description, url, startsAt, durationMins });

    res.status(201).json(item);

};

// ----- Partnerships

exports.listPartnerships = async (req, res) => {
    const owner = req.user.id;
    const company = await Company.findOne({ owner }).lean();
    if (!company) return res.json([]);
    const items = await Partnership.find({ company: company._id }).sort({ createdAt: -1 });
    res.json(items);
};

exports.createPartnership = async (req, res) => {
    const owner = req.user.id;
    const company = await Company.findOne({ owner });
    if (!company) return res.status(400).json({ message: "Create company profile first" });

    const { university, title, details, status } = req.body;
    const item = await Partnership.create({ company: company._id, university, title, details, status });
    res.status(201).json(item);
};

exports.updateApplicationStatus = async (req, res) => {
    const { id } = req.params; // application id
    const { status = "applied", notes = "" } = req.body;

    const app = await Application.findById(id)
        .populate({ path: "job", select: "company" });
    if (!app) return res.status(404).json({ message: "Application not found" });

```

```

// ensure this company owns the job

const owner = req.user.id;

const company = await Company.findOne({ owner }).lean();

if (!company || String(app.job.company) !== String(company._id)) {
    return res.status(403).json({ message: "Not allowed" });
}

};

app.status = status;
app.screening.notes = notes;
await app.save();
res.json({ message: "Updated", app });
};

```

---

```

const SkillTestResult = require("../models/SkillTestResult");

exports.getSkillProgress = async (req, res) => {
    try {
        const userId = req.user?.id;
        if (!userId) return res.status(401).json({ message: "Unauthorized access" });

        const results = await SkillTestResult.find({ user: userId })
            .populate("skill", "name")
            .sort({ createdAt: -1 });

        const grouped = {};

        results.forEach((result) => {
            const skillId = result.skill._id.toString();
            if (!grouped[skillId]) {
                grouped[skillId] = {

```

```

    skill: result.skill,
    history: [],
  );
}

grouped[skillId].history.push({
  score: result.score,
  total: result.total,
  createdAt: result.createdAt,
});
});

const response = Object.values(grouped);
res.json(response);
} catch (err) {
  console.error("Error fetching skill progress:", err.message);
  res.status(500).json({ message: "Failed to fetch skill progress" });
}
};


```

---

```

const path = require("path");
const fs = require("fs");
const Job = require("../models/Job");
const Application = require("../models/Application");
const SkillTestResult = require("../models/SkillTestResult");
const Skill = require("../models/Skill");

// quick resume scoring (very simple keyword coverage)
const keywordScore = (text = "", keywords = []) => {
  if (!text || !keywords?.length) return 0;
  const lower = text.toLowerCase();

```

```

let hits = 0;

keywords.forEach(k => {
  if (lower.includes(k.toLowerCase())) hits++;
});

return Math.min(100, Math.round((hits / keywords.length) * 100));
};

const meanRequiredSkillScore = async (userId, skillIds) => {
  if (!skillIds?.length) return 0;

  const skills = await Skill.find({ _id: { $in: skillIds } }).lean();
  const results = await SkillTestResult.find({ user: userId }).populate("skill", "name").lean();
  const map = {};

  results.forEach(r => { map[r.skill?.name] = (r.score / (r.total || 10)) * 10; });

  const arr = skills.map(s => map[s.name] || 0);
  return arr.length ? arr.reduce((a, b) => a + b, 0) / arr.length : 0;
};

exports.listOpenJobs = async (req, res) => {
  const jobs = await Job.find({ status: "open" })
    .populate("company", "name logoUrl")
    .populate("skills", "name")
    .sort({ createdAt: -1 })
    .lean();

  res.json(jobs);
};

exports.getJob = async (req, res) => {
  const job = await Job.findById(req.params.id)
    .populate("company", "name logoUrl website")
    .populate("skills", "name")
};

```

```
.lean();

if (!job) return res.status(404).json({ message: "Job not found" });

res.json(job);

};

exports.applyToJob = async (req, res) => {

const userId = req.user.id;
const jobId = req.params.id;
const { coverLetter = "" } = req.body;

const job = await Job.findById(jobId).populate("skills", "name").lean();

if (!job || job.status !== "open") {

return res.status(400).json({ message: "Job is not open or not found" });

}

// resume file (optional)

let cvUrl = "";
if (req.file) {

cvUrl = `/uploads/resumes/${req.file.filename}`;

}

// smart screening (heuristics you can evolve later)

const resumeText = (req.body.resumeText || "").slice(0, 20000); // if you embed text
const resumeScore = keywordScore(resumeText || coverLetter, job.skills.map(s => s.name));
const testScore = await meanRequiredSkillScore(userId, job.skills.map(s => s._id));

// combine 0..100

const fitScore = Math.round(resumeScore * 0.5 + (testScore / 10) * 50);

const existing = await Application.findOne({ job: jobId, student: userId });

if (existing) return res.status(409).json({ message: "Already applied" });

}
```

```
const app = await Application.create({
  job: jobId,
  student: userId,
  cvUrl,
  coverLetter,
  screening: { resumeScore, testScore, fitScore, notes: "" },
  status: "applied",
});

res.status(201).json({ message: "Application submitted", applicationId: app._id });
};
```

---

```
// server/middleware/adminAuth.js
const jwt = require("jsonwebtoken");
const User = require("../models/User");

const getSecrets = () =>
  [process.env.ADMIN_JWT_SECRET, process.env.JWT_SECRET].filter(Boolean);

module.exports = async function adminAuth(req, res, next) {
  try {
    const auth = req.headers.authorization || "";
    const token = auth.startsWith("Bearer ") ? auth.slice(7) : null;
    if (!token) return res.status(401).json({ message: "No token provided" });

    let decoded = null;
    for (const sec of getSecrets()) {
      try { decoded = jwt.verify(token, sec); break; } catch {}
    }
    if (!decoded) return res.status(401).json({ message: "Invalid or expired token" });
  }
```

```
const user = await User.findById(decoded.id).select("role");

if (!user || user.role !== "admin") {
    return res.status(403).json({ message: "Forbidden (admin only)" });
}

req.user = { id: String(user._id), role: user.role };

next();
} catch {
    return res.status(401).json({ message: "Invalid or expired token" });
}
};
```

---

```
const jwt = require("jsonwebtoken");

const authMiddleware = (req, res, next) => {
    const authHeader = req.headers.authorization;

    if (!authHeader?.startsWith("Bearer ")) {
        return res.status(401).json({ message: "Unauthorized: No token provided" });
    }

    const token = authHeader.split(" ")[1];

    try {
        // ✅ make sure your LOGIN/REGISTER signs both id + role in the token
        const decoded = jwt.verify(token, process.env.JWT_SECRET);

        if (!decoded?.id) {
            return res.status(401).json({ message: "Unauthorized: Invalid token payload" });
        }

        // expose both id and role
        req.user = { id: decoded.id, role: decoded.role || null };
    }
}
```

```
next();

} catch (err) {
    console.error("Token verification failed:", err.message);
    return res.status(403).json({ message: "Invalid or expired token" });
}

};

module.exports = authMiddleware;
```

---

```
// middleware/companyAuth.js

const jwt = require("jsonwebtoken");
const User = require("../models/User");

/**
 * Accepts tokens signed by either JWT_SECRET (company/student) or ADMIN_JWT_SECRET (admin).
 * Allows role "company" OR "admin" to access /api/company/*.
 */
module.exports = async function companyOrAdminAuth(req, res, next) {
    try {
        const auth = req.headers.authorization || "";
        const token = auth.startsWith("Bearer ") ? auth.slice(7) : null;
        if (!token) return res.status(401).json({ message: "No token provided" });

        const secrets = [process.env.JWT_SECRET, process.env.ADMIN_JWT_SECRET].filter(Boolean);
        if (!secrets.length) return res.status(500).json({ message: "Server misconfigured (no JWT secrets)" });

        let decoded = null;
        for (const sec of secrets) {
            try {
                decoded = jwt.verify(token, sec);

```

```

        break;
    } catch (_e) {}
}

if (!decoded) return res.status(401).json({ message: "Invalid or expired token" });

const user = await User.findById(decoded.id).select("role");
if (!user) return res.status(401).json({ message: "User not found" });

if (user.role !== "company" && user.role !== "admin") {
    return res.status(403).json({ message: "Forbidden (company/admin only)" });
}

req.user = { id: String(user._id), role: user.role };
next();
} catch (err) {
    return res.status(401).json({ message: "Invalid token" });
}
};


```

---

```

// server/middleware/requireRole.js

module.exports = function requireRole(required) {
    const allowed = new Set(Array.isArray(required) ? required : [required]);
    return function (req, res, next) {
        const role = req.user?.role;
        if (role === "admin" || allowed.has(role)) return next(); // ✓ admin passes all gates
        return res.status(403).json({ message: "Forbidden" });
    };
};


```

---

```

// middleware/upload.js

const path = require("path");

```

```
const fs = require("fs");

const multer = require("multer");

const dir = path.join(__dirname, "..", "uploads", "resumes");

fs.mkdirSync(dir, { recursive: true });

const storage = multer.diskStorage({  
  destination: (_req, _file, cb) => cb(null, dir),  
  filename: (_req, file, cb) => {  
    const ts = Date.now();  
    const safe = file.originalname.replace(/\s+/g, "_");  
    cb(null, `${ts}_${safe}`);  
  },  
});  
  
const upload = multer({  
  storage,  
  limits: { fileSize: 20 * 1024 * 1024 }, // 20MB  
  fileFilter: (_req, file, cb) => {  
    const ok = /\.(pdf|doc|docx)$/i.test(file.originalname);  
    cb(ok ? null : new Error("Only PDF/DOC/DOCX"), ok);  
  },  
});  
  
module.exports = upload;  


---

  
const mongoose = require("mongoose");  
  
const AdminSchema = new mongoose.Schema(  
{  
  email: { type: String, required: true, unique: true, lowercase: true, trim: true },  
});
```

```
username: { type: String, required: true, unique: true, trim: true },  
phone: { type: String, required: true, trim: true },  
password: { type: String, required: true }, // hashed  
role: { type: String, default: "admin" }, // keep it simple, no enum collision with User  
name: { type: String, default: "Admin" }  
,  
{ timestamps: true }  
);
```

```
module.exports = mongoose.model("Admin", AdminSchema);
```

---

```
// models/Application.js  
const mongoose = require("mongoose");  
  
const applicationSchema = new mongoose.Schema(  
{  
    job: { type: mongoose.Schema.Types.ObjectId, ref: "Job", required: true },  
    company: { type: mongoose.Schema.Types.ObjectId, ref: "CompanyProfile", required: true },  
    student: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },  
  
    // Keep statuses aligned with your UI  
    status: { type: String, enum: ["applied", "shortlisted", "interview", "offer", "rejected"], default: "applied", index: true },  
  
    cvUrl: { type: String, default: null },  
  
    screening: {  
        resumeScore: { type: Number, default: null },  
        testScore: { type: Number, default: null },  
        fitScore: { type: Number, default: null },  
    },
```

```
// Link one upcoming interview (simple flow). History stays in Interview collection.  
interview: { type: mongoose.Schema.Types.ObjectId, ref: "Interview", default: null },  
,  
{ timestamps: true }  
);  
  
// One application per student per job  
applicationSchema.index({ job: 1, student: 1 }, { unique: true });  
  
module.exports = mongoose.model("Application", applicationSchema);
```

---

```
const mongoose = require("mongoose");  
  
const companySchema = new mongoose.Schema(  
{  
  name: { type: String, required: true, trim: true },  
  website: { type: String, default: "" },  
  logoUrl: { type: String, default: "" },  
  hq: { type: String, default: "" },  
  about: { type: String, default: "" },  
  // optional: link to user account that owns this company profile  
  owner: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },  
,  
  { timestamps: true }  
);
```

```
module.exports = mongoose.model("Company", companySchema);
```

---

```
// models/CompanyProfile.js  
const mongoose = require("mongoose");
```

```

const companyProfileSchema = new mongoose.Schema(
  {
    owner: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "User",
      required: true,
      index: true,
      unique: true, // one profile per company account
    },
    user: { type: mongoose.Schema.Types.ObjectId, ref: "User", index: true },
    name: { type: String, required: true, trim: true },
    website: { type: String, trim: true },
    logoUrl: { type: String, trim: true },
    description: { type: String, trim: true },
    size: { type: String, enum: ["1-10", "11-50", "51-200", "201-500", "500+"], default: "1-10" },
    locations: { type: [String], default: [] },
    domains: { type: [String], default: [] },
    contactEmail: { type: String, trim: true },
    contactPhone: { type: String, trim: true },
  },
  { timestamps: true }
);

module.exports = mongoose.model("CompanyProfile", companyProfileSchema);



---


// models/Hackathon.js
const mongoose = require("mongoose");

const hackathonSchema = new mongoose.Schema(
  {

```

```

company: { type: mongoose.Schema.Types.ObjectId, ref: "CompanyProfile", required: true },
title: { type: String, required: true },
brief: { type: String, default: "" },
skills: [{ type: mongoose.Schema.Types.ObjectId, ref: "Skill" }],
prize: { type: String, default: "" },
rules: { type: String, default: "" },
resources: { type: String, default: "" }, // links, repo templates, dataset links
startAt: { type: Date, required: true },
endAt: { type: Date, required: true },
visibility: { type: String, enum: ["public", "private"], default: "public" },
status: { type: String, enum: ["upcoming", "live", "ended"], default: "upcoming" },
bannerUrl: { type: String, default: null },
},
{ timestamps: true }
);

```

```
module.exports = mongoose.model("Hackathon", hackathonSchema);
```

---

```

// models/HackathonRegistration.js
const mongoose = require("mongoose");

const regSchema = new mongoose.Schema(
{
  hackathon: { type: mongoose.Schema.Types.ObjectId, ref: "Hackathon", required: true },
  student: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },
  teamName: { type: String, default: "" },
},
{ timestamps: true }
);

regSchema.index({ hackathon: 1, student: 1 }, { unique: true });

```

```
regSchema.index({ hackathon: 1, student: 1 }, { unique: true });
```

```
module.exports = mongoose.model("HackathonRegistration", regSchema);



---



```
// models/HackathonSubmission.js
const mongoose = require("mongoose");

const submissionSchema = new mongoose.Schema(
{
  hackathon: { type: mongoose.Schema.Types.ObjectId, ref: "Hackathon", required: true },
  student: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },
  repoUrl: { type: String, default: "" },
  demoUrl: { type: String, default: "" },
  fileUrl: { type: String, default: "" }, // uploaded zip/pdf etc.
  notes: { type: String, default: "" },

  // judging
  score: { type: Number, default: null }, // 0..100
  feedback: { type: String, default: "" },
  rank: { type: Number, default: null },
},
{ timestamps: true });

```


```

```
submissionSchema.index({ hackathon: 1, student: 1 }, { unique: true });

module.exports = mongoose.model("HackathonSubmission", submissionSchema);



---



```
const mongoose = require("mongoose");

const scoreItemSchema = new mongoose.Schema({
  rubricKey: String, // matches Hackathon.rubric.key

```


```

```

score: { type: Number, default: 0 },
}, { _id: false });

const judgeScoreSchema = new mongoose.Schema({
judge: { type: mongoose.Schema.Types.ObjectId, ref: "User" },
items: [scoreItemSchema],
total: { type: Number, default: 0 },
}, { _id: false });

const hackSubmissionSchema = new mongoose.Schema({
hackathon: { type: mongoose.Schema.Types.ObjectId, ref: "Hackathon", required: true },
team: { type: mongoose.Schema.Types.ObjectId, ref: "HackTeam", required: true },
title: { type: String, required: true },
repoUrl: { type: String, default: "" },
demoUrl: { type: String, default: "" },
notes: { type: String, default: "" },
files: [{ name: String, url: String }], // if you store uploaded assets

judgeScores: [judgeScoreSchema],
finalScore: { type: Number, default: 0 }, // average of judge totals
rank: { type: Number, default: null },
}, { timestamps: true });

hackSubmissionSchema.index({ hackathon: 1, team: 1 }, { unique: true });

module.exports = mongoose.model("HackSubmission", hackSubmissionSchema);



---


const mongoose = require("mongoose");

const hackTeamSchema = new mongoose.Schema({
hackathon: { type: mongoose.Schema.Types.ObjectId, ref: "Hackathon", required: true },

```

```
name: { type: String, required: true },  
members: [{ type: mongoose.Schema.Types.ObjectId, ref: "User", required: true }],  
createdBy: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },  
, { timestamps: true });
```

```
hackTeamSchema.index({ hackathon: 1, name: 1 }, { unique: true });
```

```
module.exports = mongoose.model("HackTeam", hackTeamSchema);
```

---

```
// models/Interview.js
```

```
const mongoose = require("mongoose");
```

```
const interviewSchema = new mongoose.Schema(
```

```
{
```

```
    application: { type: mongoose.Schema.Types.ObjectId, ref: "Application", required: true, index: true },
```

```
    company: { type: mongoose.Schema.Types.ObjectId, ref: "CompanyProfile", required: true, index: true },
```

```
    student: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true, index: true },
```

```
    stage: { type: String, enum: ["screening", "technical", "hr", "final"], default: "technical" },
```

```
    startsAt: { type: Date, required: true },
```

```
    durationMins: { type: Number, default: 45 },
```

```
// Reuse your in-platform video infra (same as webinars)
```

```
    roomId: { type: String, required: true, unique: true },
```

```
    notes: { type: String, default: "" },
```

```
    status: { type: String, enum: ["scheduled", "completed", "cancelled"], default: "scheduled" },
```

```
},
```

```
    { timestamps: true }
```

```
);
```

```
module.exports = mongoose.model("Interview", interviewSchema);
```

---

```
const mongoose = require("mongoose");

const jobSchema = new mongoose.Schema(
{
    company: { type: mongoose.Schema.Types.ObjectId, ref: "CompanyProfile", required: true },
    title: { type: String, required: true },
    type: { type: String, enum: ["job", "internship"], default: "job" },
    location: String,
    description: String,
    skills: [{ type: mongoose.Schema.Types.ObjectId, ref: "Skill" }],
    minScore: { type: Number, default: 0 }, // 0..10
    isFeatured: { type: Boolean, default: false },
    status: { type: String, enum: ["open", "closed", "paused"], default: "open" },
},
{ timestamps: true }
);
```

```
module.exports = mongoose.model("Job", jobSchema);
```

---

```
const mongoose = require("mongoose");

const partnershipSchema = new mongoose.Schema({
    company: { type: mongoose.Schema.Types.ObjectId, ref: "CompanyProfile", required: true },
    university: String,
    title: String,
    details: String,
    status: { type: String, enum: ["proposal", "active", "completed"], default: "proposal" },
},
{ timestamps: true });
```

```
module.exports = mongoose.model("Partnership", partnershipSchema);
```

---

```
const mongoose = require("mongoose");

const questionSchema = new mongoose.Schema(
{
  set: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "QuestionSet",
    required: true,
  },
  skill: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Skill",
    required: true, // recommended for filtering and indexing
    index: true,
  },
  question: {
    type: String,
    required: true,
    trim: true,
  },
  options: {
    type: [String],
    validate: {
      validator: (arr) => Array.isArray(arr) && arr.length === 4,
      message: "Exactly 4 options are required.",
    },
  },
  answer: {
    type: String,
  }
}
```

```
    required: true,
    trim: true,
  },
},
{ timestamps: true }
);

module.exports = mongoose.model("Question", questionSchema);
```

---

```
const mongoose = require("mongoose");

const questionSetSchema = new mongoose.Schema(
{
  title: {
    type: String,
    required: true,
    trim: true,
  },
  skill: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Skill",
    required: true,
    index: true, // improves query performance
  },
  description: {
    type: String,
    default: "",
    trim: true,
  },
  status: {
    type: String,
```

```
    enum: ["active", "inactive"],
    default: "active",
  },
},
{ timestamps: true }
);
```

```
module.exports = mongoose.model("QuestionSet", questionSetSchema);
```

---

```
const mongoose = require("mongoose");

const roadmapStepSchema = new mongoose.Schema({
  level: {
    type: String,
    enum: ["Beginner", "Intermediate", "Advanced"],
    required: true,
  },
  title: { type: String, required: true },
  description: String,
  topics: [String],
  resources: [
    {
      title: String,
      url: String,
    },
  ],
});
```

```
const skillSchema = new mongoose.Schema({
  name: {
```

```
    type: String,
    required: true,
    unique: true,
    trim: true,
    lowercase: true,
  },
  roadmap: [roadmapStepSchema], // ✅ optional, dynamic, and extendable
},
{ timestamps: true
);

// Optional: enable text search on name
skillSchema.index({ name: "text" });

module.exports = mongoose.model("Skill", skillSchema);
```

---

```
const mongoose = require("mongoose");

const skillTestResultSchema = new mongoose.Schema({
  user: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },
  skill: { type: mongoose.Schema.Types.ObjectId, ref: "Skill", required: true },
  score: Number,
  total: Number,
}, { timestamps: true });

module.exports = mongoose.model("SkillTestResult", skillTestResultSchema);
```

---

```
const mongoose = require("mongoose");

const testResultSchema = new mongoose.Schema(
{
```

```
studentId: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },
skill: { type: mongoose.Schema.Types.ObjectId, ref: "Skill", required: true },
score: { type: Number, required: true },
total: { type: Number, required: true },
percentage: { type: Number },
},
{ timestamps: true }
);
```

```
module.exports = mongoose.model("TestResult", testResultSchema);
```

---

```
const mongoose = require("mongoose");
```

```
const userSchema = new mongoose.Schema(
{
  name: {
    type: String,
    required: true,
    trim: true,
  },
  username: {
    type: String,
    required: true,
    unique: true,
    lowercase: true,
    trim: true,
  },
  email: {
    type: String,
    required: true,
    unique: true,
  }
});
```

```
    lowercase: true,  
    trim: true,  
,  
  password: {  
    type: String,  
    required: true,  
,  
  phone: {  
    type: String,  
    required: true,  
    trim: true,  
,  
  role: {  
    type: String,  
    enum: ["student", "university", "company", "admin"],  
    default: "student",  
,  
  university: {  
    type: String,  
    default: "",  
    trim: true,  
,  
  education: {  
    type: String,  
    enum: ["UG", "PG", "Diploma", "Other"],  
    default: "UG",  
,  
  skills: {  
    type: [String],  
    default: [],  
,  
  },
```

```
    },
    { timestamps: true }
);

module.exports = mongoose.model("User", userSchema);



---



```
// models/Webinar.js
const mongoose = require("mongoose");

function genId(prefix = "wb") {
  // e.g. wb_1712589123456_p8x2qf
  return `${prefix}_${Date.now()}_${Math.random().toString(36).slice(2, 8)}`;
}

const webinarSchema = new mongoose.Schema(
{
  title: { type: String, required: true, trim: true },
  description: { type: String, default: "" },
  speaker: { type: String, default: "" },

  // When it starts and for how long
  startsAt: { type: Date },
  durationMins: { type: Number, default: 60, min: 1 },

  // Visibility: public | private | unlisted
  visibility: { type: String, enum: ["public", "private", "unlisted"], default: "public" },

  // Unique roomId for joining/viewing
  roomId: {
    type: String,
    unique: true,
```


```

```
index: true,  
default: () => genId("room"),  
},  
  
// Meeting credentials (your internal signaling layer)  
meeting: {  
  meetingId: { type: String, default: () => genId("mtg") },  
  hostToken: { type: String, default: () => genId("host") },  
  attendeeToken: { type: String, default: () => genId("join") },  
},  
  
// The company profile that hosts this webinar (REQUIRED)  
hostCompany: {  
  type: mongoose.Schema.Types.ObjectId,  
  ref: "CompanyProfile",  
  required: true,  
  index: true,  
},  
  
// Optional: who created it (User)  
createdBy: {  
  type: mongoose.Schema.Types.ObjectId,  
  ref: "User",  
},  
,  
{ timestamps: true }  
);  
  
module.exports = mongoose.model("Webinar", webinarSchema);
```

---

```
const express = require("express");
```

```
const router = express.Router();

const Question = require("../models/Question");
const QuestionSet = require("../models/QuestionSet");
const authMiddleware = require("../middleware/auth");

// GET: All questions for a specific set
router.get("/set/:setId", async (req, res) => {
  try {
    const questions = await Question.find({ set: req.params.setId });
    res.status(200).json(questions);
  } catch (err) {
    console.error("Error fetching questions:", err.message);
    res.status(500).json({ message: "Failed to fetch questions", error: err.message });
  }
});

// POST: Add a question to a set
router.post("/", async (req, res) => {
  const { set, question, options, answer } = req.body;

  if (!set || !question || !options?.length || !answer) {
    return res.status(400).json({ message: "All fields are required" });
  }

  try {
    const setExists = await QuestionSet.findById(set);
    if (!setExists) {
      return res.status(404).json({ message: "Question set not found" });
    }

    const newQuestion = new Question({

```

```

    set,
    skill: setExists.skill.toString(),
    question: question.trim(),
    options: options.map((opt) => opt.trim()),
    answer: answer.trim(),
  });

  await newQuestion.save();
  res.status(201).json(newQuestion);
} catch (err) {
  console.error("Error adding question:", err.message);
  res.status(500).json({ message: "Failed to add question", error: err.message });
}
});

// Add this in routes/api/admin/questions.js
router.get("/skill/:id", authMiddleware, async (req, res) => {
  try {
    const skillId = req.params.id;
    const latestSet = await QuestionSet.findOne({ skill: skillId }).sort({ createdAt: -1 });
    if (!latestSet) return res.status(404).json({ message: "No question set found" });

    const questions = await Question.find({ set: latestSet._id }).limit(15);
    res.json({ setId: latestSet._id, skillId, questions });
  } catch (err) {
    console.error("Error fetching questions by skill:", err.message);
    res.status(500).json({ message: "Server error" });
  }
});

// PUT: Update a question

```

```
router.put("/:id", async (req, res) => {
  const { question, options, answer } = req.body;

  if (!question || !options?.length || !answer) {
    return res.status(400).json({ message: "Incomplete update fields" });
  }

  try {
    const updated = await Question.findByIdAndUpdate(
      req.params.id,
      {
        question: question.trim(),
        options: options.map((opt) => opt.trim()),
        answer: answer.trim(),
      },
      { new: true }
    );

    if (!updated) {
      return res.status(404).json({ message: "Question not found" });
    }

    res.status(200).json(updated);
  } catch (err) {
    console.error("Error updating question:", err.message);
    res.status(500).json({ message: "Failed to update", error: err.message });
  }
});

// DELETE: Remove a question
router.delete("/:id", async (req, res) => {
```

```

try {

  const deleted = await Question.findByIdAndDelete(req.params.id);

  if (!deleted) {
    return res.status(404).json({ message: "Question not found" });
  }

  res.status(200).json({ message: "Deleted successfully" });

} catch (err) {
  console.error("Error deleting question:", err.message);
  res.status(500).json({ message: "Failed to delete", error: err.message });
}

});

module.exports = router;

```

```

const express = require("express");
const router = express.Router();
const SkillTestResult = require("../models/SkillTestResult");
const User = require("../models/User");
const Skill = require("../models/Skill");

// ✅ GET: Admin fetch all skill test results with populated user and skill
router.get("/admin/skill-test-results", async (req, res) => {
  try {
    const results = await SkillTestResult.find()
      .populate("user", "name email")
      .populate("skill", "name") // 🔥 THIS is what makes skill.name work
      .sort({ createdAt: -1 });

    res.status(200).json(results);
  }
});

```

```
    } catch (err) {
      console.error("Error fetching results:", err.message);
      res.status(500).json({ message: "Failed to fetch results", error: err.message });
    }
  });

module.exports = router;
```

---

```
const express = require("express");
const router = express.Router();
const QuestionSet = require("../..../models/QuestionSet");
const Skill = require("../..../models/Skill");

// GET: Fetch all question sets with skill names
router.get("/", async (req, res) => {
  try {
    const sets = await QuestionSet.find()
      .populate("skill", "name")
      .sort({ createdAt: -1 });

    res.status(200).json(sets);
  } catch (err) {
    console.error("Error fetching sets:", err.message);
    res.status(500).json({ message: "Failed to fetch sets", error: err.message });
  }
});

// POST: Create a new question set
router.post("/", async (req, res) => {
  try {
    const { title, skill, description } = req.body;
```

```
if (!title || !skill) {
    return res.status(400).json({ message: "Title and skill are required" });
}

const skillExists = await Skill.findById(skill);
if (!skillExists) {
    return res.status(404).json({ message: "Skill not found" });
}

const newSet = new QuestionSet({
    title: title.trim(),
    skill,
    description: description?.trim() || "",
});

await newSet.save();
res.status(201).json(newSet);
} catch (err) {
    console.error("Error adding set:", err.message);
    res.status(500).json({ message: "Error adding set", error: err.message });
}
});
```

  

```
// PUT: Update question set
router.put("/:id", async (req, res) => {
    try {
        const { title, skill, description } = req.body;

        if (!title || !skill) {
            return res.status(400).json({ message: "Title and skill are required" });
        }
```

```
        }

const skillExists = await Skill.findById(skill);

if (!skillExists) {
    return res.status(404).json({ message: "Skill not found" });
}

const updatedSet = await QuestionSet.findByIdAndUpdateAndUpdate(
    req.params.id,
    {
        title: title.trim(),
        skill,
        description: description?.trim() || "",
    },
    { new: true }
);

if (!updatedSet) {
    return res.status(404).json({ message: "Set not found" });
}

res.status(200).json(updatedSet);
} catch (err) {
    console.error("Error updating set:", err.message);
    res.status(500).json({ message: "Error updating set", error: err.message });
}
});

// DELETE: Remove a question set
router.delete("/:id", async (req, res) => {
    try {
```

```
const deletedSet = await QuestionSet.findByIdAndDelete(req.params.id);

if (!deletedSet) {
    return res.status(404).json({ message: "Set not found" });
}

res.status(200).json({ message: "Set deleted successfully" });

} catch (err) {
    console.error("Error deleting set:", err.message);
    res.status(500).json({ message: "Error deleting set", error: err.message });
}

});

module.exports = router;
```

---

```
const express = require("express");
const router = express.Router();
const Skill = require("../models/Skill");

// GET: Fetch all skills
router.get("/", async (req, res) => {
    try {
        const skills = await Skill.find().sort({ createdAt: -1 });
        res.status(200).json(skills);
    } catch (err) {
        console.error("Error fetching skills:", err.message);
        res.status(500).json({ message: "Error fetching skills", error: err.message });
    }
});

// POST: Add a new skill
router.post("/", async (req, res) => {
```

```
try {

    const { name } = req.body;

    if (!name || !name.trim()) {

        return res.status(400).json({ message: "Skill name is required" });

    }

}

const existing = await Skill.findOne({ name: name.trim() });

if (existing) {

    return res.status(409).json({ message: "Skill already exists" });

}

const newSkill = new Skill({ name: name.trim() });

await newSkill.save();

res.status(201).json(newSkill);

} catch (err) {

    console.error("Error adding skill:", err.message);

    res.status(500).json({ message: "Error adding skill", error: err.message });

}

});

// PUT: Update an existing skill

router.put("/:id", async (req, res) => {

    try {

        const { name } = req.body;

        if (!name || !name.trim()) {

            return res.status(400).json({ message: "Skill name is required" });

        }

    }

    const skill = await Skill.findByIdAndUpdate(
        req.params.id,
        { name: name.trim() },
        { new: true }
    );

    res.json(skill);

});
```

```
        { new: true }

    );

if (!skill) {
    return res.status(404).json({ message: "Skill not found" });
}

res.status(200).json(skill);

} catch (err) {
    console.error("Error updating skill:", err.message);
    res.status(500).json({ message: "Error updating skill", error: err.message });
}

});

// DELETE: Remove a skill
router.delete("/:id", async (req, res) => {
    try {
        const skill = await Skill.findByIdAndDelete(req.params.id);
        if (!skill) {
            return res.status(404).json({ message: "Skill not found" });
        }

        res.status(200).json({ message: "Skill deleted successfully" });
    } catch (err) {
        console.error("Error deleting skill:", err.message);
        res.status(500).json({ message: "Error deleting skill", error: err.message });
    }

});

module.exports = router;
```

---

```
// routes/company/applications.js

const express = require("express");
const router = express.Router();
const auth = require("../middleware/auth");
const User = require("../models/User");
const CompanyProfile = require("../models/CompanyProfile");
const Application = require("../models/Application");
const Interview = require("../models/Interview");
const companyAuth = require("../middleware/companyAuth");

// company auth

async function requireCompany(req, res, next) {
  try {
    const u = await User.findById(req.user.id).select("role");
    if (!u) return res.status(401).json({ message: "Unauthorized" });
    if (u.role !== "company") return res.status(403).json({ message: "Companies only" });
    next();
  } catch (e) {
    res.status(500).json({ message: "Server error" });
  }
}

// load company profile by owner

async function loadCompany(req, res, next) {
  try {
    const c = await CompanyProfile.findOne({ owner: req.user.id }).select("_id");
    if (!c) return res.status(400).json({ message: "Create company profile first" });
    req.companyId = c._id;
    next();
  } catch (e) {
    res.status(500).json({ message: "Server error" });
  }
}
```

```

    }

}

router.use(companyAuth);

/** 
 * GET /api/company/applications
 */
router.get("/", auth, requireCompany, loadCompany, async (req, res) => {
try {

  const list = await Application.find({ company: req.companyId })
    .sort({ createdAt: -1 })
    .populate({ path: "student", select: "name email" })
    .populate({ path: "job", select: "title type location" })
    .populate({ path: "interview" })
    .lean();

  res.json(list);
} catch (e) {
  console.error("List company apps failed:", e);
  res.status(500).json({ message: "Server error" });
}
});

/** 
 * PATCH /api/company/applications/:id
 * Body: { status }
*/
router.patch("/:id", auth, requireCompany, loadCompany, async (req, res) => {
try {

  const updated = await Application.findOneAndUpdate(
    { _id: req.params.id, company: req.companyId },

```

```

    { $set: { status: req.body.status } },
    { new: true }
  ).lean();

  if (!updated) return res.status(404).json({ message: "Application not found" });
  res.json(updated);
} catch (e) {
  res.status(400).json({ message: e.message || "Bad request" });
}
});

/** 
 * POST /api/company/applications/:id/schedule-interview
 * Body: { startsAt (ISO), stage, durationMins, notes }
 */
router.post("/:id/schedule-interview", auth, requireCompany, loadCompany, async (req, res) => {
  try {
    const app = await Application.findOne({ _id: req.params.id, company: req.companyId })
      .populate("student", "name email")
      .populate("job", "title");

    if (!app) return res.status(404).json({ message: "Application not found" });

    const startsAt = new Date(req.body.startsAt);
    if (isNaN(startsAt.getTime())) return res.status(400).json({ message: "Invalid startsAt" });

    const stage = req.body.stage || "technical";
    const durationMins = Number(req.body.durationMins || 45);
    const notes = (req.body.notes || "").toString();

    // roomId compatible with your webinar infra
  }
}
);

```

```
const roomId = `iv_${app._id.toString()}`;

const interview = await Interview.create({
    application: app._id,
    company: req.companyId,
    student: app.student,
    stage,
    startsAt,
    durationMins,
    roomId,
    notes,
});

app.status = "interview";
app.interview = interview._id;
await app.save();

res.status(201).json({ success: true, interview });
} catch (e) {
    console.error("Schedule interview failed:", e);
    res.status(500).json({ message: "Server error" });
}

};

/** 
 * PATCH /api/company/applications/:id/reschedule-interview
 * Body: { startsAt (ISO), stage, durationMins, notes }
 */
router.patch("/:id/reschedule-interview", auth, requireCompany, loadCompany, async (req, res) => {
    try {
```

```

    const app = await Application.findOne({ _id: req.params.id, company: req.companyId
}).populate("interview");

    if (!app) return res.status(404).json({ message: "Application not found" });

    if (!app.interview) return res.status(400).json({ message: "No interview to reschedule" });

    const patch = {};
    if (req.body.startsAt) {
        const d = new Date(req.body.startsAt);
        if (isNaN(d.getTime())) return res.status(400).json({ message: "Invalid startsAt" });
        patch.startsAt = d;
    }
    if (req.body.durationMins != null) patch.durationMins = Number(req.body.durationMins);
    if (req.body.stage) patch.stage = req.body.stage;
    if (req.body.notes != null) patch.notes = (req.body.notes || "").toString();

    const updated = await Interview.findOneAndUpdate(
        { _id: app.interview._id, company: req.companyId },
        { $set: patch },
        { new: true }
    );

    res.json({ success: true, interview: updated });
} catch (e) {
    console.error("Reschedule interview failed:", e);
    res.status(500).json({ message: "Server error" });
}
});

/**
 * DELETE /api/company/applications/:id/cancel-interview
 */

```

```

router.delete("/:id/cancel-interview", auth, requireCompany, loadCompany, async (req, res) => {
  try {
    const app = await Application.findOne({ _id: req.params.id, company: req.companyId })
      .populate("interview");

    if (!app) return res.status(404).json({ message: "Application not found" });

    if (!app.interview) return res.status(400).json({ message: "No scheduled interview" });

    await Interview.deleteOne({ _id: app.interview._id, company: req.companyId });

    app.interview = null;

    // Only revert status if it was "interview"
    if (app.status === "interview") app.status = "shortlisted";

    await app.save();

    res.json({ success: true });
  } catch (e) {
    console.error("Cancel interview failed:", e);
    res.status(500).json({ message: "Server error" });
  }
});

module.exports = router;

```

```

// routes/company/hackathons.js
const express = require("express");
const router = express.Router();
const auth = require("../middleware/auth");
const User = require("../models/User");
const CompanyProfile = require("../models/CompanyProfile");
const Hackathon = require("../models/Hackathon");

// guards

```

```

async function requireCompany(req, res, next) {
  try {
    const u = await User.findById(req.user.id).select("role");
    if (!u) return res.status(401).json({ message: "Unauthorized" });
    if (u.role !== "company") return res.status(403).json({ message: "Companies only" });
    next();
  } catch (e) {
    res.status(500).json({ message: "Server error" });
  }
}

async function loadCompany(req, res, next) {
  try {
    const c = await CompanyProfile.findOne({ owner: req.user.id }).select("_id");
    if (!c) return res.status(400).json({ message: "Create company profile first" });
    req.companyId = c._id;
    next();
  } catch (e) {
    res.status(500).json({ message: "Server error" });
  }
}

// list (mine)
router.get("/", auth, requireCompany, loadCompany, async (_req, res) => {
  const list = await Hackathon.find({ company: _req.companyId })
    .sort({ createdAt: -1 })
    .populate("skills", "name")
    .lean();
  res.json(list);
});

// create

```

```

router.post("/", auth, requireCompany, loadCompany, async (req, res) => {
  const { title, brief, skills = [], prize, rules, resources, startAt, endAt, visibility = "public", bannerUrl } =
    req.body || {};
  if (!title || !startAt || !endAt) return res.status(400).json({ message: "title, startAt, endAt required" });
}

const now = new Date();
let status = "upcoming";
if (new Date(startAt) <= now && now <= new Date(endAt)) status = "live";
if (now > new Date(endAt)) status = "ended";

const doc = await Hackathon.create({
  company: req.companyId, title, brief, skills, prize, rules, resources,
  startAt, endAt, visibility, status, bannerUrl: bannerUrl || null
});
const pop = await doc.populate("skills", "name");
res.status(201).json(pop);
});

// update
router.put("/:id", auth, requireCompany, loadCompany, async (req, res) => {
  const id = req.params.id;
  const body = req.body || {};
  const doc = await Hackathon.findOneAndUpdate({ _id: id, company: req.companyId }, body, { new: true })
    .populate("skills", "name");
  if (!doc) return res.status(404).json({ message: "Not found" });
  res.json(doc);
});

// set status
router.patch("/:id/status", auth, requireCompany, loadCompany, async (req, res) => {

```

```

const doc = await Hackathon.findOneAndUpdate(
  { _id: req.params.id, company: req.companyId },
  { $set: { status: req.body.status } },
  { new: true }
);
if (!doc) return res.status(404).json({ message: "Not found" });
res.json(doc);
});

// registrations (view)
router.get("/:id/registrations", auth, requireCompany, loadCompany, async (req, res) => {
  const HackathonRegistration = require("../models/HackathonRegistration");
  const regs = await HackathonRegistration.find({ hackathon: req.params.id })
    .populate("student", "name email")
    .sort({ createdAt: -1 })
    .lean();
  res.json(regs);
});

// submissions (view)
router.get("/:id/submissions", auth, requireCompany, loadCompany, async (req, res) => {
  const Submission = require("../models/HackathonSubmission");
  const subs = await Submission.find({ hackathon: req.params.id })
    .populate("student", "name email")
    .sort({ createdAt: -1 })
    .lean();
  res.json(subs);
});

// judge / score
router.post("/:id/judge/:submissionId", auth, requireCompany, loadCompany, async (req, res) => {

```

```

const Submission = require("../models/HackathonSubmission");

const { score, feedback } = req.body || {};

const sub = await Submission.findOne({ _id: req.params.submissionId, hackathon: req.params.id });

if (!sub) return res.status(404).json({ message: "Submission not found" });

sub.score = Number(score ?? 0);

sub.feedback = feedback || "";

await sub.save();

res.json(sub);

});

// publish leaderboard

router.post("/:id/publish-leaderboard", auth, requireCompany, loadCompany, async (req, res) => {
  const Submission = require("../models/HackathonSubmission");

  const subs = await Submission.find({ hackathon: req.params.id, score: { $ne: null } }).sort({ score: -1, createdAt: 1 });

  for (let i = 0; i < subs.length; i++) {
    subs[i].rank = i + 1;

    await subs[i].save();
  }

  res.json({ success: true, count: subs.length });
});

module.exports = router;

```

---

```

const express = require("express");

const router = express.Router();

const auth = require("../middleware/auth");

const CompanyProfile = require("../models/CompanyProfile");

const companyAuth = require("../middleware/companyAuth");

```

```

// ✅ guard: only for feature routes (NOT for /company/profile)

async function ensureCompanyProfile(req, res, next) {
  try {
    const profile = await CompanyProfile.findOne({ owner: req.user.id });

    if (!profile) {
      return res.status(428).json({ message: "Create company profile first" }); // 428 Precondition Required
    }

    req.companyProfile = profile; // <-- pass to downstream handlers
    next();
  } catch (e) {
    next(e);
  }
}

router.use(companyAuth); // ✅ every route under /api/company/* is auth'd

// ---- FEATURE ROUTES (guarded) ----

const webinars = require("./webinars");
router.get("/webinars", auth, ensureCompanyProfile, webinars.list);
router.post("/webinars", auth, ensureCompanyProfile, webinars.create);

module.exports = router;

```

```

// routes/company/jobs.js

const express = require("express");
const router = express.Router();
const auth = require("../middleware/auth");
const Job = require("../models/Job");
const Skill = require("../models/Skill");
const User = require("../models/User");
const CompanyProfile = require("../models/CompanyProfile");

```

```

// All company job routes require auth
router.use(auth);

// ensure the caller is a company & load their CompanyProfile _id
async function loadCompany(req, res, next) {
  try {
    const u = await User.findById(req.user.id).select("role");
    if (!u) return res.status(401).json({ message: "Unauthorized" });
    if (u.role !== "company") {
      return res.status(403).json({ message: "Companies only" });
    }
    const cp = await CompanyProfile.findOne({ owner: req.user.id }).select("_id");
    if (!cp) return res.status(400).json({ message: "Create company profile first" });
    req.companyId = cp._id; // <-- THIS is what Job.company expects
    next();
  } catch (e) {
    console.error("loadCompany failed:", e);
    res.status(500).json({ message: "Server error" });
  }
}

// GET /api/company/jobs
router.get("/", loadCompany, async (req, res) => {
  try {
    const jobs = await Job.find({ company: req.companyId })
      .populate("skills", "name")
      .sort({ createdAt: -1 });
    res.json(jobs);
  } catch (e) {
    console.error("List jobs error:", e);
  }
}

```

```
    res.status(500).json({ message: "Failed to fetch jobs" });

}

});

// POST /api/company/jobs

router.post("/", loadCompany, async (req, res) => {
  try {
    const { title, type, location, description, minScore, isFeatured, skills } = req.body;
    if (!title) return res.status(400).json({ message: "Title is required" });

    // ensure skills is an array of ObjectId strings
    const skillIds = Array.isArray(skills) ? skills.filter(Boolean) : [];

    // (Optional) validate skills exist
    if (skillIds.length) {
      const count = await Skill.countDocuments({ _id: { $in: skillIds } });
      if (count !== skillIds.length) {
        return res.status(400).json({ message: "One or more skills are invalid" });
      }
    }

    const job = await Job.create({
      company: req.companyId,           // <-- correct CompanyProfile _id
      title: title.trim(),
      type: type || "job",
      location: location || "",
      description: description || "",
      minScore: Number(minScore || 0),
      isFeatured: !!isFeatured,
      skills: skillIds,
    });
  }
});
```

```

const populated = await job.populate("skills", "name");
res.status(201).json(populated);
} catch (e) {
  console.error("Create job error:", e);
  res.status(400).json({ message: e.message || "Invalid payload" });
}
});

// PUT /api/company/jobs/:id
router.put("/:id", loadCompany, async (req, res) => {
try {
  const id = req.params.id;
  const { title, type, location, description, minScore, isFeatured, skills, status } = req.body;

  const update = {
    ...(title != null ? { title } : {}),
    ...(type ? { type } : {}),
    ...(location != null ? { location } : {}),
    ...(description != null ? { description } : {}),
    ...(minScore != null ? { minScore: Number(minScore) } : {}),
    ...(isFeatured != null ? { isFeatured: !!isFeatured } : {}),
    ...(status ? { status } : {}),
    ...(Array.isArray(skills) ? { skills } : {}),
  };
}

const job = await Job.findOneAndUpdate(
  { _id: id, company: req.companyId }, // <-- filter by CompanyProfile _id
  update,
  { new: true }
).populate("skills", "name");

```

```

if (!job) return res.status(404).json({ message: "Job not found" });

res.json(job);

} catch (e) {
  console.error("Update job error:", e);
  res.status(400).json({ message: e.message || "Invalid payload" });
}

});

// PATCH /api/company/jobs/:id/toggle
router.patch("/:id/toggle", loadCompany, async (req, res) => {
  try {
    const id = req.params.id;

    const job = await Job.findOne({ _id: id, company: req.companyId });

    if (!job) return res.status(404).json({ message: "Job not found" });

    job.status = job.status === "open" ? "closed" : "open";
    await job.save();

    res.json({ status: job.status });

  } catch (e) {
    console.error("Toggle job error:", e);
    res.status(400).json({ message: "Toggle failed" });
  }

});

// DELETE /api/company/jobs/:id
router.delete("/:id", loadCompany, async (req, res) => {
  try {
    const id = req.params.id;

    const del = await Job.findOneAndDelete({ _id: id, company: req.companyId });

    if (!del) return res.status(404).json({ message: "Job not found" });

    res.json({ success: true });

  }
});

```

```
    } catch (e) {
        console.error("Delete job error:", e);
        res.status(400).json({ message: "Delete failed" });
    }
});

module.exports = router;
```

---

```
// routes/company/profile.js
const express = require("express");
const router = express.Router();
const auth = require("../middleware/auth");
const CompanyProfile = require("../models/CompanyProfile");
const User = require("../models/User");
const companyAuth = require("../middleware/companyAuth");

// Ensure logged-in account is a company
async function requireCompanyRole(req, res, next) {
    try {
        const user = await User.findById(req.user.id).select("role");
        if (!user) return res.status(401).json({ message: "Unauthorized" });
        if (user.role !== "company") {
            return res.status(403).json({ message: "Only company accounts can access this resource" });
        }
        next();
    } catch (e) {
        console.error("requireCompanyRole failed:", e);
        res.status(500).json({ message: "Server error" });
    }
}
```

```

// Helpers

function asArray(v) {
  if (!v) return [];
  if (Array.isArray(v)) return v.map(String).map(s => s.trim()).filter(Boolean);
  return String(v).split(",").map(s => s.trim()).filter(Boolean);
}

router.use(companyAuth);

// GET current profile

router.get("/", auth, requireCompanyRole, async (req, res) => {
  try {
    const profile = await CompanyProfile.findOne({ owner: req.user.id }).lean();
    if (!profile) return res.status(404).json({ message: "No company profile yet" });
    res.json(profile);
  } catch (e) {
    console.error("Get profile failed:", e);
    res.status(500).json({ message: "Server error" });
  }
});

/** 
 * POST create or update profile (two-step to avoid Mongo modifier conflicts).
 * - If not exists: create with { owner, user } set.
 * - If exists: update fields; set legacy `user` ONLY if missing.
 */
router.post("/", auth, requireCompanyRole, async (req, res) => {
  try {
    const body = req.body || {};
    const base = {
      name: body.name,
      website: body.website,

```

```
  logoUrl: body.logoUrl,
  description: body.description,
  size: body.size || "1-10",
  locations: asArray(body.locations),
  domains: asArray(body.domains),
  contactEmail: body.contactEmail,
  contactPhone: body.contactPhone,
};

let profile = await CompanyProfile.findOne({ owner: req.user.id });

if (!profile) {
  // CREATE
  profile = await CompanyProfile.create({
    ...base,
    owner: req.user.id, // canonical
    user: req.user.id, // legacy (to satisfy old unique index on user)
  });
  return res.status(201).json(profile);
}

// UPDATE (set legacy field only if missing)
const update = { ...base };
if (!profile.user) update.user = req.user.id;

profile = await CompanyProfile.findByIdAndUpdate(
  profile._id,
  { $set: update },
  { new: true, runValidators: true }
);
```

```
res.json(profile);

} catch (e) {
    console.error("Upsert (two-step) profile failed:", e);
    res.status(400).json({ message: e.message || "Bad request" });
}

});

// PUT strict update (must already exist)

router.put("/", auth, requireCompanyRole, async (req, res) => {
    try {
        const body = req.body || {};
        const update = {
            name: body.name,
            website: body.website,
            logoUrl: body.logoUrl,
            description: body.description,
            size: body.size,
            locations: asArray(body.locations),
            domains: asArray(body.domains),
            contactEmail: body.contactEmail,
            contactPhone: body.contactPhone,
        };
        const existing = await CompanyProfile.findOne({ owner: req.user.id });
        if (!existing) return res.status(404).json({ message: "Create company profile first" });

        // keep legacy `user` non-null if it was missing
        if (!existing.user) update.user = req.user.id;

        const updated = await CompanyProfile.findByIdAndUpdate(
            existing._id,
```

```

    { $set: update },
    { new: true, runValidators: true }
);

res.json(updated);

} catch (e) {
  console.error("Update profile failed:", e);
  res.status(400).json({ message: e.message || "Bad request" });
}

});

// Debug

router.get("/me", auth, requireCompanyRole, async (req, res) => {
  try {
    const profile = await CompanyProfile.findOne({ owner: req.user.id }).lean();
    res.json({
      userIdFromToken: req.user.id,
      hasProfile: !!profile,
      profileOwner: profile?.owner || null,
      profile,
    });
  } catch (e) {
    res.status(500).json({ message: "Server error" });
  }
});

module.exports = router;

```

---

```

// routes/company/webinars.js
const express = require("express");
const router = express.Router();

```

```

const auth = require("../middleware/auth");
const Webinar = require("../models/Webinar");
const CompanyProfile = require("../models/CompanyProfile");
const companyAuth = require("../middleware/companyAuth");

// Attach the company profile for the logged-in user
async function loadCompany(req, res, next) {
  try {
    const company = await CompanyProfile.findOne({ owner: req.user.id }).lean();
    if (!company) return res.status(400).json({ message: "Create company profile first" });
    req.company = company; // { _id, ... }
    next();
  } catch (e) {
    console.error("loadCompany failed:", e);
    res.status(500).json({ message: "Server error" });
  }
}

router.use(companyAuth);

// GET /api/company/webinars -> list webinars for this company
router.get("/", auth, loadCompany, async (req, res) => {
  try {
    const list = await Webinar.find({ hostCompany: req.company._id }).sort({ startsAt: 1 }).lean();
    res.json(list);
  } catch (e) {
    console.error("List webinars failed:", e);
    res.status(500).json({ message: "Server error" });
  }
});

// POST /api/company/webinars -> create a webinar (roomId + meeting auto-generated by model)

```

```

router.post("/", auth, loadCompany, async (req, res) => {
  try {
    const { title, description, speaker, startsAt, durationMins, visibility } = req.body;

    if (!title || !title.trim()) {
      return res.status(400).json({ message: "Title is required" });
    }

    let startsAtDate = null;
    if (startsAt) {
      const d = new Date(startsAt);
      if (isNaN(d.getTime())) return res.status(400).json({ message: "Invalid startsAt datetime" });
      startsAtDate = d;
    }

    const webinar = await Webinar.create({
      title: title.trim(),
      description: description || "",
      speaker: speaker || "",
      startsAt: startsAtDate,
      durationMins: Number.isFinite(Number(durationMins)) ? Number(durationMins) : 60,
      visibility: ["public", "private", "unlisted"].includes(visibility) ? visibility : "public",
      hostCompany: req.company._id, // ✅ REQUIRED
      createdBy: req.user.id, // optional
      // roomId + meeting.* are auto-generated by model defaults
    });

    // Helpful join URLs for your frontend
    res.status(201).json({
      ...webinar.toObject(),
      joinUrl: `/student/webinar/${webinar.roomId}`, // students view here
    });
  }
});

```

```

        hostUrl: `/company/webinar/${webinar.roomId}`,    // company studio
    });
}

} catch (e) {
    // duplicate key safety for roomId collisions (very unlikely)
    if (e && e.code === 11000 && e.keyPattern && e.keyPattern.roomId) {
        console.error("RoomId collision, retrying once...");
        try {
            // simple retry: let the model re-generate defaults via a new create
            const again = await Webinar.create({
                title: (req.body.title || "").trim(),
                description: req.body.description || "",
                speaker: req.body.speaker || "",
                startsAt: req.body.startsAt ? new Date(req.body.startsAt) : null,
                durationMins: Number.isFinite(Number(req.body.durationMins)) ?
                    Number(req.body.durationMins) : 60,
                visibility: ["public", "private", "unlisted"].includes(req.body.visibility) ? req.body.visibility :
                    "public",
                hostCompany: req.company._id,
                createdBy: req.user.id,
            });
            return res.status(201).json({
                ...again.toObject(),
                joinUrl: `/student/webinar/${again.roomId}`,
                hostUrl: `/company/webinar/${again.roomId}`,
            });
        } catch (e2) {
            console.error("Create webinar retry failed:", e2);
        }
    }
}

console.error("Create webinar failed:", e);
res.status(400).json({ message: e.message || "Bad request" });

```

```
}

});

// PUT /api/company/webinars/:id -> update (only your own)
router.put("/:id", auth, loadCompany, async (req, res) => {
  try {
    const payload = { ...req.body };

    // Never allow roomId/hostCompany to be overwritten by clients
    delete payload.roomId;
    delete payload.hostCompany;
    delete payload.meeting;

    const updated = await Webinar.findOneAndUpdate(
      { _id: req.params.id, hostCompany: req.company._id },
      { $set: payload },
      { new: true, runValidators: true }
    );

    if (!updated) return res.status(404).json({ message: "Webinar not found" });
    res.json(updated);
  } catch (e) {
    console.error("Update webinar failed:", e);
    res.status(400).json({ message: e.message || "Bad request" });
  }
});

// DELETE /api/company/webinars/:id -> delete (only your own)
router.delete("/:id", auth, loadCompany, async (req, res) => {
  try {
    const removed = await Webinar.findOneAndDelete({
      _id: req.params.id,
    });
  }
});
```

```
    hostCompany: req.company._id,  
});  
  
if (!removed) return res.status(404).json({ message: "Webinar not found" });  
  
res.json({ success: true });  
} catch (e) {  
  
    console.error("Delete webinar failed:", e);  
  
    res.status(500).json({ message: "Server error" });  
}  
};  
  
module.exports = router;
```

---

```
// routes/student/hackathons.js  
  
const express = require("express");  
  
const router = express.Router();  
  
const path = require("path");  
  
const fs = require("fs");  
  
const multer = require("multer");  
  
const auth = require("../middleware/auth");  
  
const User = require("../models/User");  
  
const Hackathon = require("../models/Hackathon");  
  
const HackathonRegistration = require("../models/HackathonRegistration");  
  
const HackathonSubmission = require("../models/HackathonSubmission");  
  
  
// uploads for submissions  
  
const uploadDir = path.join(__dirname, "../uploads/hackathons");  
  
if (!fs.existsSync(uploadDir)) fs.mkdirSync(uploadDir, { recursive: true });  
  
const storage = multer.diskStorage({  
  
    destination: (_req, _file, cb) => cb(null, uploadDir),  
  
    filename: (_req, file, cb) => {  
  
        const ts = Date.now();
```

```

const safe = file.originalname.replace(/[^\\w\\.]+/g, "_");
cb(null, `${ts}_${safe}`);
},
});

const upload = multer({ storage });

// list public + live/upcoming
router.get("/", async (_req, res) => {
  const now = new Date();
  const list = await Hackathon.find({
    visibility: "public",
    endAt: { $gte: new Date(now.getTime() - 1000 * 60 * 60 * 24) } // show recent ended too if you want tweak
  })
    .sort({ startAt: 1 })
    .populate("skills", "name")
    .lean();
  res.json(list);
});

// detail
router.get("/:id", async (req, res) => {
  const doc = await Hackathon.findById(req.params.id).populate("skills", "name").lean();
  if (!doc) return res.status(404).json({ message: "Not found" });
  res.json(doc);
});

// my registration
router.get("/:id/registration", auth, async (req, res) => {
  const reg = await HackathonRegistration.findOne({ hackathon: req.params.id, student: req.user.id }).lean();
  res.json(reg || null);
}

```

```

});

// register

router.post("/:id/register", auth, async (req, res) => {
  const doc = await Hackathon.findById(req.params.id);
  if (!doc) return res.status(404).json({ message: "Hackathon not found" });
  const now = new Date();
  if (now > doc.endAt) return res.status(400).json({ message: "Registration closed" });

  const reg = await HackathonRegistration.findOneAndUpdate(
    { hackathon: doc._id, student: req.user.id },
    { $setOnInsert: { teamName: (req.body?.teamName || "").trim() } },
    { upsert: true, new: true }
  );
  res.json(reg);
});

// unregister

router.delete("/:id/register", auth, async (req, res) => {
  await HackathonRegistration.findOneAndDelete({ hackathon: req.params.id, student: req.user.id });
  res.json({ success: true });
});

// submit (multipart optional)

router.post("/:id/submit", auth, upload.single("file"), async (req, res) => {
  const hack = await Hackathon.findById(req.params.id);
  if (!hack) return res.status(404).json({ message: "Hackathon not found" });

  const now = new Date();
  if (now < hack.startAt) return res.status(400).json({ message: "Hackathon not started" });
  if (now > hack.endAt) return res.status(400).json({ message: "Submission window closed" });
});

```

```

const fileUrl = req.file ? `/uploads/hackathons/${req.file.filename}` : "";

```

```

const sub = await HackathonSubmission.findOneAndUpdate(
  { hackathon: hack._id, student: req.user.id },
  {
    $set: {
      repoUrl: (req.body?.repoUrl || "").trim(),
      demoUrl: (req.body?.demoUrl || "").trim(),
      notes: (req.body?.notes || "").trim(),
      ...(fileUrl ? { fileUrl } : {}),
    },
  },
  { upsert: true, new: true }
);
res.json(sub);
});

// my submission
router.get("/:id/my-submission", auth, async (req, res) => {
  const sub = await HackathonSubmission.findOne({ hackathon: req.params.id, student: req.user.id }).lean();
  res.json(sub || null);
});

// leaderboard (public once ranked)
router.get("/:id/leaderboard", async (req, res) => {
  const subs = await HackathonSubmission.find({ hackathon: req.params.id, rank: { $ne: null } })
    .populate("student", "name")
    .sort({ rank: 1 })
    .lean();

```

```
res.json(subs);

});

module.exports = router;



---



```
const express = require("express");
const router = express.Router();
const auth = require("../middleware/auth");
const User = require("../models/User");
const Interview = require("../models/Interview");

async function requireStudent(req, res, next) {
  try {
    const u = await User.findById(req.user.id).select("role");
    if (!u) return res.status(401).json({ message: "Unauthorized" });
    if (u.role !== "student") return res.status(403).json({ message: "Students only" });
    next();
  } catch (e) {
    res.status(500).json({ message: "Server error" });
  }
}

// GET /api/student/interviews (my upcoming)
router.get("/", auth, requireStudent, async (req, res) => {
  try {
    const list = await Interview.find({ student: req.user.id, status: "scheduled" })
      .sort({ startsAt: 1 })
      .populate({ path: "application", select: "job", populate: { path: "job", select: "title" } })
      .lean();
    res.json(list);
  } catch (e) {
```


```

```
    res.status(500).json({ message: "Server error" });

}

});

module.exports = router;



---



```
// routes/student/jobs.js

const express = require("express");
const router = express.Router();
const auth = require("../middleware/auth");
const upload = require("../middleware/upload");

const User = require("../models/User");
const Job = require("../models/Job");
const Application = require("../models/Application");
const companyAuth = require("../middleware/companyAuth");

// Guard: only students for write operations
async function requireStudentRole(req, res, next) {
  try {
    const u = await User.findById(req.user.id).select("role");
    if (!u) return res.status(401).json({ message: "Unauthorized" });
    if (u.role !== "student") {
      return res.status(403).json({ message: "Students only" });
    }
    next();
  } catch (e) {
    console.error("requireStudentRole failed:", e);
    res.status(500).json({ message: "Server error" });
  }
}
```


```

```

}

router.use(companyAuth);

/** 
 * GET /api/student/jobs
 * Public list of open jobs (company + skills populated)
 */
router.get("/", async (_req, res) => {
  try {
    const list = await Job.find({ status: "open" })
      .sort({ createdAt: -1 })
      .populate("company", "name logoUrl")
      .populate("skills", "name");
    res.json(list);
  } catch (e) {
    console.error("List student jobs failed:", e);
    res.status(500).json({ message: "Server error" });
  }
});

/** 
 * POST /api/student/jobs/: jobId/apply
 * Multipart accepted: coverLetter + resume
 */
router.post("/: jobId/apply", auth, requireStudentRole, upload.single("resume"), async (req, res) => {
  try {
    const { jobId } = req.params;

    // Ensure job exists and is open
    const job = await Job.findById(jobId).select("_id status company");
    if (!job) return res.status(404).json({ message: "Job not found" });
  }
});

```

```

    if (job.status !== "open") return res.status(400).json({ message: "Job is not accepting applications" });
});

// Has this student already applied?

const existing = await Application.findOne({ student: req.user.id, job: jobId }).select("_id");
if (existing) return res.status(409).json({ message: "Already applied" });

const coverLetter = (req.body?.coverLetter || "").toString();
const cvUrl = req.file ? `/uploads/resumes/${req.file.filename}` : null;

const app = await Application.create({
  student: req.user.id,
  job: jobId,
  company: job.company, // <- critical for company views/screening
  status: "applied",
  cvUrl,
  screening: { resumeScore: null, testScore: null, fitScore: null },
});

res.status(201).json({ success: true, applicationId: app._id, cvUrl });
} catch (e) {
  console.error("Apply failed:", e);
  res.status(500).json({ message: "Server error" });
}
});

/** 
 * DELETE /api/student/jobs/:jobId/apply
 * Withdraw an application (if still not past interview/offer).
 */
router.delete("/:jobId/apply", auth, requireStudentRole, async (req, res) => {

```

```

try {

  const { jobId } = req.params;

  const app = await Application.findOne({ job: jobId, student: req.user.id });

  if (!app) return res.status(404).json({ message: "Application not found" });

  if (["offer", "interview"].includes(app.status)) {
    return res.status(400).json({ message: "Cannot withdraw at this stage" });
  }

  await app.deleteOne();
  res.json({ success: true });
} catch (e) {
  console.error("Withdraw failed:", e);
  res.status(500).json({ message: "Server error" });
}

});

/** 
 * GET /api/student/jobs/mine/applications
 * This student's apps (with interview populated)
 */
router.get("/mine/applications", auth, requireStudentRole, async (req, res) => {
  try {
    const apps = await Application.find({ student: req.user.id })
      .sort({ createdAt: -1 })
      .populate({
        path: "job",
        select: "title type location skills company",
        populate: [
          { path: "company", select: "name logoUrl" },
          { path: "skills", select: "name" }
        ]
      })
  }
}

```

```

    ]
})

.populate("interview"); // <- bring interview (if scheduled)

res.json(apps);

} catch (e) {
  console.error("List my apps failed:", e);
  res.status(500).json({ message: "Server error" });
}

});

module.exports = router;

```

---

```

const express = require("express");
const router = express.Router();
const Skill = require("../models/Skill");
const QuestionSet = require("../models/QuestionSet");
const Question = require("../models/Question");
const authMiddleware = require("../middleware/auth");
const TestResult = require("../models/TestResult");

router.get("/", authMiddleware, async (req, res) => {
  const skillName = req.query.skill;

  if (!skillName) {
    return res.status(400).json({ message: "Skill name is required" });
  }

  try {
    const skill = await Skill.findOne({ name: skillName });
    if (!skill) {

```

```
        return res.status(404).json({ message: "Skill not found" });

    }

const latestSet = await QuestionSet.findOne({ skill: skill._id }).sort({ createdAt: -1 });

if (!latestSet) {
    return res.status(404).json({ message: "No question set found for this skill" });
}

const questions = await Question.find({ questionSet: latestSet._id }).limit(15);

res.json({ setId: latestSet._id, skillId: skill._id, questions });
} catch (err) {
    console.error("Skill test fetch error:", err.message);
    res.status(500).json({ message: "Failed to fetch skill test questions" });
}
});

router.post("/submit", async (req, res) => {
    const { skill, score, total } = req.body;
    const studentId = req.user.id; // assuming middleware sets req.user

    if (!skill || score == null || !total) {
        return res.status(400).json({ message: "Missing fields in result submission." });
    }

    try {
        const percentage = (score / total) * 100;
        const result = await TestResult.create({
            studentId,
            skill,
```

```
    score,
    total,
    percentage,
});

res.status(200).json({ message: "Result saved", result });
} catch (err) {
  console.error(err);
  res.status(500).json({ message: "Error saving result" });
}
});

module.exports = router;
```

---

```
// routes/student/webinars.js
const express = require("express");
const router = express.Router();
const auth = require("../middleware/auth");
const User = require("../models/User");
const Webinar = require("../models/Webinar");

// Optional: ensure student role
async function requireStudent(req, res, next) {
  try {
    const user = await User.findById(req.user.id).select("role");
    if (!user) return res.status(401).json({ message: "Unauthorized" });
    if (user.role !== "student") return res.status(403).json({ message: "Students only" });
    next();
  } catch (e) {
    console.error("requireStudent failed:", e);
    res.status(500).json({ message: "Server error" });
  }
}
```

```

        }
    }

// GET /api/student/webinars -> public, upcoming or ongoing
router.get("/", auth, requireStudent, async (req, res) => {
    try {
        const now = new Date();
        const list = await Webinar.find({
            visibility: "public",
            startsAt: { $gte: new Date(now.getTime() - 60 * 60 * 1000) }, // started within last hour or future
        })
        .sort({ startsAt: 1 })
        .select("title speaker startsAt durationMins roomId hostCompany");
        res.json(list);
    } catch (e) {
        console.error("List student webinars failed:", e);
        res.status(500).json({ message: "Server error" });
    }
});

// GET /api/student/webinars/:roomId -> details for viewer
router.get("/:roomId", auth, requireStudent, async (req, res) => {
    try {
        const w = await Webinar.findOne({ roomId: req.params.roomId, visibility: "public" })
        .select("-meeting.hostToken"); // hide host token
        if (!w) return res.status(404).json({ message: "Webinar not found" });
        res.json(w);
    } catch (e) {
        console.error("Get webinar failed:", e);
        res.status(500).json({ message: "Server error" });
    }
});

```

```
});
```

```
module.exports = router;
```

---

```
const express = require("express");
const router = express.Router();
const SkillTestResult = require("../models/SkillTestResult");
const User = require("../models/User");

router.get("/skill-test-results", async (req, res) => {
  try {
    const results = await SkillTestResult.find()
      .populate("user", "name email")
      .populate("skill", "name") // 🔥 THIS is what makes skill.name work
      .sort({ createdAt: -1 });

    res.status(200).json(results);
  } catch (err) {
    console.error("Error fetching results:", err.message);
    res.status(500).json({ message: "Failed to fetch results", error: err.message });
  }
});
```

```
module.exports = router;
```

---

```
// server/routes/auth.admin.js
const express = require("express");
const jwt = require("jsonwebtoken");
const bcrypt = require("bcryptjs");
const adminAuth = require("../middleware/adminAuth");
const User = require("../models/User");
```

```
const router = express.Router();

const getAdminSecret = () => process.env.ADMIN_JWT_SECRET || process.env.JWT_SECRET;

// POST /api/auth/admin/login
router.post("/admin/login", async (req, res) => {
  try {
    const SECRET = getAdminSecret();
    if (!SECRET) return res.status(500).json({ message: "Missing ADMIN_JWT_SECRET/JWT_SECRET" });

    const { email, password } = req.body || {};
    if (!email || !password) return res.status(400).json({ message: "Email & password required" });

    const user = await User.findOne({ email: String(email).toLowerCase().trim() });
    if (!user || user.role !== "admin") return res.status(401).json({ message: "Invalid credentials" });

    const ok = await bcrypt.compare(password, user.password || "");
    if (!ok) return res.status(401).json({ message: "Invalid credentials" });

    const token = jwt.sign({ id: user._id, role: user.role }, SECRET, { expiresIn: "7d" });
    res.json({ token, user: { id: user._id, name: user.name, email: user.email, role: user.role } });
  } catch (err) {
    res.status(500).json({ message: "Server error" });
  }
});

// GET /api/auth/admin/me
router.get("/admin/me", adminAuth, async (req, res) => {
  const me = await User.findById(req.user.id).select("name email role username phone");
  res.json({ user: me });
});
```

```
module.exports = router;



---



```
const express = require("express");
const router = express.Router();
const User = require("../models/User");
const { registerUser, loginUser } = require("../controllers/authController");

// @route POST /api/auth/register
// @desc Register new user
router.post("/register", registerUser);

// @route POST /api/auth/login
// @desc Login user
router.post("/login", loginUser);

// @route GET /api/auth/check-username?username=xyz
// @desc Check if a username is available
router.get("/check-username", async (req, res) => {
    const { username } = req.query;

    if (!username) {
        return res.status(400).json({ message: "Username is required" });
    }

    try {
        const existingUser = await User.findOne({ username: username.toLowerCase() });
        res.json({ available: !existingUser });
    } catch (err) {
        console.error("Error checking username:", err.message);
        res.status(500).json({ message: "Server error" });
    }
})
```


```

```
    }

});

module.exports = router;



---



```
const express = require("express");
const router = express.Router();
const mongoose = require("mongoose");
const auth = require("../middleware/auth");
const requireRole = require("../middleware/requireRole");

const CompanyProfile = require("../models/CompanyProfile");
const Job = require("../models/Job");
const Webinar = require("../models/Webinar");
const Hackathon = require("../models/Hackathon");
const Partnership = require("../models/Partnership");
const Skill = require("../models/Skill");
const SkillTestResult = require("../models/SkillTestResult");

// helper: ensure user has company profile
async function getOrCreateCompanyProfile(userId, payloadIfCreate) {
  let profile = await CompanyProfile.findOne({ user: userId });
  if (!profile && payloadIfCreate) {
    profile = await CompanyProfile.create({ user: userId, ...payloadIfCreate });
  }
  return profile;
}

/** ----- Company Profile ----- */
router.get("/profile", auth, requireRole("company"), async (req, res) => {
  const profile = await CompanyProfile.findOne({ user: req.user.id });

```


```

```
    res.json(profile || null);
  });

router.post("/profile", auth, async (req, res) => {
  const exists = await CompanyProfile.findOne({ user: req.user.id });
  if (exists) return res.status(400).json({ message: "Profile already exists" });
  const created = await CompanyProfile.create({ user: req.user.id, ...req.body });
  res.json(created);
});

router.put("/profile", auth, async (req, res) => {
  const updated = await CompanyProfile.findOneAndUpdate(
    { user: req.user.id },
    { $set: req.body },
    { new: true, upsert: false }
  );
  if (!updated) return res.status(400).json({ message: "Create company profile first" });
  res.json(updated);
});

/** ----- Jobs ----- */
router.get("/jobs", auth, async (req, res) => {
  const profile = await CompanyProfile.findOne({ user: req.user.id });
  if (!profile) return res.json([]);
  const jobs = await Job.find({ company: profile._id }).populate("skills", "name");
  res.json(jobs);
});

router.post("/jobs", auth, async (req, res) => {
  const profile = await CompanyProfile.findOne({ user: req.user.id });
  if (!profile) return res.status(400).json({ message: "Create company profile first" });
});
```

```
const payload = { ...req.body };

payload.company = profile._id;

payload.skills = (payload.skills || []).map((id) => new mongoose.Types.ObjectId(id));

const job = await Job.create(payload);

res.json(job);

});

router.put("/jobs/:id", auth, async (req, res) => {

const profile = await CompanyProfile.findOne({ user: req.user.id });

if (!profile) return res.status(400).json({ message: "Create company profile first" });

const payload = { ...req.body };

if (payload.skills) {

payload.skills = payload.skills.map((id) => new mongoose.Types.ObjectId(id));

}

const job = await Job.findOneAndUpdate(
{ _id: req.params.id, company: profile._id },
{ $set: payload },
{ new: true }
).populate("skills", "name");

if (!job) return res.status(404).json({ message: "Job not found" });

res.json(job);

});

router.delete("/jobs/:id", auth, async (req, res) => {

const profile = await CompanyProfile.findOne({ user: req.user.id });

if (!profile) return res.status(400).json({ message: "Create company profile first" });


```

```

    await Job.deleteOne({ _id: req.params.id, company: profile._id });

    res.json({ success: true });

});

router.patch("/jobs/:id/toggle", auth, async (req, res) => {
  const profile = await CompanyProfile.findOne({ user: req.user.id });

  if (!profile) return res.status(400).json({ message: "Create company profile first" });

  const job = await Job.findOne({ _id: req.params.id, company: profile._id });

  if (!job) return res.status(404).json({ message: "Job not found" });

  job.status = job.status === "open" ? "closed" : "open";

  await job.save();

  res.json(job);

});

/** ----- Webinars / Hackathons / Partnerships (used by UI) ----- */
router.get("/webinars", auth, async (req, res) => {
  const profile = await CompanyProfile.findOne({ user: req.user.id });

  if (!profile) return res.json([]);

  const items = await Webinar.find({ company: profile._id }).sort({ createdAt: -1 });

  res.json(items);

});

router.post("/webinars", auth, async (req, res) => {
  const profile = await CompanyProfile.findOne({ user: req.user.id });

  if (!profile) return res.status(400).json({ message: "Create company profile first" });

  const item = await Webinar.create({ company: profile._id, ...req.body });

  res.json(item);

});

router.get("/hackathons", auth, async (req, res) => {
  const profile = await CompanyProfile.findOne({ user: req.user.id });

```

```

if (!profile) return res.json([]);

const items = await Hackathon.find({ company: profile._id }).populate("skills", "name").sort({ createdAt: -1 });

res.json(items);
});

router.post("/hackathons", auth, async (req, res) => {
  const profile = await CompanyProfile.findOne({ user: req.user.id });

  if (!profile) return res.status(400).json({ message: "Create company profile first" });

  const payload = { ...req.body, company: profile._id };

  payload.skills = (payload.skills || []).map((id) => new mongoose.Types.ObjectId(id));

  const item = await Hackathon.create(payload);

  res.json(item);
});

router.get("/partnerships", auth, async (req, res) => {
  const profile = await CompanyProfile.findOne({ user: req.user.id });

  if (!profile) return res.json([]);

  const items = await Partnership.find({ company: profile._id }).sort({ createdAt: -1 });

  res.json(items);
});

router.post("/partnerships", auth, async (req, res) => {
  const profile = await CompanyProfile.findOne({ user: req.user.id });

  if (!profile) return res.status(400).json({ message: "Create company profile first" });

  const item = await Partnership.create({ company: profile._id, ...req.body });

  res.json(item);
});

/** ----- Applications (placeholder to match UI) ----- */
// Adjust this to your actual Application model if you have one

router.get("/applications", auth, async (req, res) => {
  // Return empty initially unless you have a schema for applications
}

```

```

    res.json([]);

});

router.patch("/applications/:id", auth, async (req, res) => {
    // Implement once you have Application model
    return res.status(400).json({ message: "Not implemented yet" });
});

/** ----- Talent Search (simple version using SkillTestResult) -----
 * Query: /company/talent-search?skillIds=...&skillIds=...&minScore=6
 */
router.get("/talent-search", auth, async (req, res) => {
    const skillIds = []
        .concat(req.query.skillIds || [])
        .map((id) => (mongoose.isValidObjectId(id) ? new mongoose.Types.ObjectId(id) : null))
        .filter(Boolean);

    const minScore = Number(req.query.minScore || 0);

    if (!skillIds.length) return res.json([]);

    // 1) Pull all test results for those skills
    const rows = await SkillTestResult.aggregate([
        { $match: { skill: { $in: skillIds } } },
        {
            $group: {
                _id: { user: "$user", skill: "$skill" },
                avgScore: { $avg: "$score" },
            },
        },
        {
            $group: {

```

```

        _id: "$_id.user",
        perSkill: { $push: { skill: "$_id.skill", avgScore: "$avgScore" } },
        avgRequiredSkillScore: { $avg: "$avgScore" },
    },
},
{ $match: { avgRequiredSkillScore: { $gte: minScore } } },
{ $limit: 100 },
]);

// 2) decorate with user & basic fitScore
const results = await Promise.all(
rows.map(async (row) => {
    const user = await mongoose.model("User").findById(row._id).select("name email university").lean();
    return {
        student: user,
        avgRequiredSkillScore: Math.round(row.avgRequiredSkillScore * 10) / 10,
        fitScore: Math.round(row.avgRequiredSkillScore * 10) / 10, // simple; replace with custom logic
    };
})
);

res.json(results);
});

module.exports = router;

```

---

```

const express = require("express");
const router = express.Router();

const fetch = (...args) => import("node-fetch").then(({default: f}) => f(...args)); // only if you need LLM APIs

```

```

// Helpers (replace with your models if needed)

async function getSkillProgress(userId, api) {
  // You likely already have a controller; this just proxies your existing route.
  const r = await api.get("/student/skill-progress");
  return Array.isArray(r.data) ? r.data : [];
}

async function getJobs(api) {
  const r = await api.get("/student/jobs");
  return Array.isArray(r.data) ? r.data : [];
}

async function getCareerRoadmap(api) {
  const r = await api.get("/student/career-roadmap");
  return r.data?.roadmap || {};
}

async function getHackathons(api) {
  const r = await api.get("/student/hackathons");
  return Array.isArray(r.data) ? r.data : [];
}

// ---- Simple scoring utilities ----

function avg(arr) { return arr.length ? arr.reduce((a,b)=>a+b,0)/arr.length : 0; }

function normalizeScore(n, min=0, max=10) { return Math.max(min, Math.min(max, +n|0)); }

// Make a shallow rule-based guidance scaffold

function makeGuidance({progress, jobs, roadmap, hackathons}) {
  // Skill baselines
  const skills = progress.map(s => ({
    id: s.skill?._id || s.skill,

```

```

name: s.skill?.name || s.skill,
latest: s.history?.length ? s.history.slice().sort((a,b)=>new Date(b.createdAt)-new
Date(a.createdAt))[0].score : 0,
average: normalizeScore(avg((s.history || []).map(h=>h.score)))
}).sort((a,b)=>b.average-a.average);

const strengths = skills.filter(s=>s.average>=7).map(s=>s.name);
const gaps    = skills.filter(s=>s.average<6).map(s=>s.name);

// Job fit: rank by overlap of skills + minimum score heuristics
const rankedJobs = jobs.map(job=>{
  const jobSkills = (job.skills || []).map(x=>typeof x==="string"?x:(x.name || x._id));
  const matchCount = jobSkills.filter(js => skills.some(s => s.name==js)).length;
  const avgRequired = avg(
    jobSkills.map(js => (skills.find(s=>s.name==js)?.average ?? 0))
  );
  const fit = Math.round((matchCount*2 + avgRequired) * 10) / 10; // simple heuristic
  return { job, fit, matchCount, avgRequired };
}).sort((a,b)=>b.fit-a.fit).slice(0,6);

// Weekly plan (lightweight)
const weeklyPlan = [
  { day: "Mon", task: "Revise core weak topic", minutes: 40 },
  { day: "Tue", task: "1 mock interview (technical)", minutes: 45 },
  { day: "Wed", task: "Build/extend mini project", minutes: 60 },
  { day: "Thu", task: "Apply to 3 matched roles", minutes: 30 },
  { day: "Fri", task: "Timed quiz on weak skill", minutes: 30 },
  { day: "Sat", task: "Open-source or challenge issue", minutes: 45 },
  { day: "Sun", task: "Reflect & plan next week", minutes: 20 },
];

```

```

// Resources from roadmap if available

const gapResources = (gaps || []).map(g => ({
  skill: g,
  suggestions: (roadmap[g]?.[0]?.resources || []).slice(0,3) // first step's resources
}));


// Hackathons to join (upcoming)

const upcomingH = hackathons
  .filter(h=>new Date(h.startAt)>new Date())
  .sort((a,b)=>new Date(a.startAt)-new Date(b.startAt))
  .slice(0,3);

const readiness = Math.round(avg.skills.map(s=>s.average))*10)/10;

return {
  metrics: { readiness, strengths, gaps },
  jobMatches: rankedJobs.map(r=>({
    _id: r.job._id,
    title: r.job.title,
    company: r.job.company?.name || "-",
    type: r.job.type,
    location: r.job.location || "Remote",
    fit: r.fit,
    skills: (r.job.skills || []).map(x=>typeof x==="string"?x:(x.name || x._id))
  })),
  weeklyPlan,
  gapResources,
  upcomingHackathons: upcomingH.map(h=>({
    _id: h._id, title: h.title, startAt: h.startAt, endAt: h.endAt
  }))
};

```

```

}

// Optional LLM enrich (only if you have OPENAI_API_KEY in env)

async function llmEnrich(summaryInput) {
  try {
    if (!process.env.OPENAI_API_KEY) return null;

    const prompt = `You are a career coach. Given the student profile below (strengths, gaps, job matches, weekly plan), provide a short, encouraging 120-150 word guidance note and 3 tactical tips.

JSON input:\n${JSON.stringify(summaryInput, null, 2)}`;

    const resp = await fetch("https://api.openai.com/v1/chat/completions", {
      method: "POST",
      headers: { "Content-Type": "application/json", "Authorization": `Bearer ${process.env.OPENAI_API_KEY}` },
      body: JSON.stringify({
        model: "gpt-4o-mini",
        messages: [{role: "user", content: prompt}],
        temperature: 0.7
      })
    });

    const json = await resp.json();
    const text = json?.choices?.[0]?.message?.content?.trim();
    return text || null;
  } catch {
    return null;
  }
}

router.get("/guidance", async (req, res) => {
  try {
    // If you already wrap Axios for internal server-to-server calls, use that here.

    const api = req.api || req; // adapt to your server structure
  
```

```

const [progress, jobs, roadmap, hackathons] = await Promise.all([
  getSkillProgress(req.user?._id, api),
  getJobs(api),
  getCareerRoadmap(api),
  getHackathons(api),
]);

const guidance = makeGuidance({progress, jobs, roadmap, hackathons});

const enriched = await llmEnrich({
  strengths: guidance.metrics.strengths,
  gaps: guidance.metrics.gaps,
  jobMatches: guidance.jobMatches.slice(0,3),
  weeklyPlan: guidance.weeklyPlan
});

res.json({ ...guidance, coachNote: enriched || null });

} catch (e) {
  console.error("AI guidance failed:", e);
  res.status(500).json({ message: "Guidance generation failed" });
}

});

// Minimal chat endpoint (optional)

router.post("/ask", async (req, res) => {
  const q = (req.body?.q || "").toString().trim();
  if (!q) return res.status(400).json({ message: "Missing question" });

  // If no LLM key, provide a safe, non-empty fallback
  if (!process.env.OPENAI_API_KEY) {
    return res.json({ answer: "AI coach is offline. Tip: focus on one gap at a time, apply to roles where your average skill score ≥6, and ship a small weekly project to build signal." });
  }

  try {
    const resp = await fetch("https://api.openai.com/v1/chat/completions", {

```

```

    method:"POST",
    headers:{ "Content-Type":"application/json", "Authorization":`Bearer
${process.env.OPENAI_API_KEY}` },
    body: JSON.stringify({
        model: "gpt-4o-mini",
        messages: [{role:"system", content:"You are a concise career coach for students in
tech."},{role:"user", content:q}],
        temperature: 0.6
    })
});

const json = await resp.json();

const text = json?.choices?.[0]?.message?.content?.trim() || "No answer.";
res.json({ answer: text });

} catch (e) {
    res.json({ answer: "Couldn't reach AI right now. Try again in a bit." });
}

});

```

module.exports = router;

---

```

const express = require("express");
const router = express.Router();
const mongoose = require("mongoose");

const User = require("../models/User");
const SkillTestResult = require("../models/SkillTestResult");
const authMiddleware = require("../middleware/auth");
const multer = require("multer");
const upload = multer({ dest: "uploads/" });
const Skill = require("../models/Skill");

```

// ✅ 1. Get Skill Progress

```

const { getSkillProgress } = require("../controllers/studentController");
router.get("/skill-progress", authMiddleware, getSkillProgress);

// ✅ 2. Career Roadmap Generator

router.get("/career-roadmap", authMiddleware, async (req, res) => {
  try {
    const results = await SkillTestResult.find({ user: req.user.id }).populate("skill", "name").lean();

    if (!results.length) {
      return res.json({ message: "No roadmap available", roadmap: [] });
    }

    // Group scores by skill name
    const skillScores = {};
    results.forEach(({ skill, score }) => {
      const skillName = skill.name || skill; // fallback
      if (!skillScores[skillName]) skillScores[skillName] = [];
      skillScores[skillName].push(score);
    });

    const roadmap = {};

    for (const skill in skillScores) {
      const scores = skillScores[skill];
      const avg = scores.reduce((a, b) => a + b, 0) / scores.length;
      const steps = [];

      if (avg < 4) {
        steps.push(
          {
            id: "start",

```

```
        type: "Start",
        title: `Start Learning ${skill}`,
        description: `Build a solid foundation in ${skill}.`,
        parent: null,
        link: null,
    },
{
    id: "course1",
    type: "Course",
    title: `Beginner Course for ${skill}`,
    description: `Understand core concepts of ${skill} from scratch.`,
    parent: "start",
    link: `https://www.coursera.org/search?query=${encodeURIComponent(skill)}`,
},
{
    id: "basics",
    type: "Practice",
    title: `Hands-on Practice for ${skill}`,
    description: `Practice with beginner-level problems or mini projects.`,
    parent: "course1",
    link: null,
}
);
}

} else if (avg < 7) {
    steps.push(
    {
        id: "project",
        type: "Project",
        title: `Intermediate ${skill} Projects`,
        description: `Apply your knowledge in real-world use cases.`,
        parent: null,
```

```
link: `https://github.com/search?q=${encodeURIComponent(skill)}+intermediate+project`,
},
{
id: "mentor",
type: "Mentorship",
title: `Join a ${skill} Community`,
description: `Get feedback and collaborate with peers.`,
parent: "project",
link: `https://discord.com/search?q=${encodeURIComponent(skill)}&type=public`,
}
);
} else {
steps.push(
{
id: "certification",
type: "Certification",
title: `Get Certified in ${skill}`,
description: `Showcase your proficiency in ${skill}.`,
parent: null,
link: `https://www.edx.org/search?q=${encodeURIComponent(skill)}`,
},
{
id: "oss",
type: "Contribute",
title: `Open Source ${skill} Projects`,
description: `Give back to the community by contributing to codebases.`,
parent: "certification",
link: `https://github.com/search?q=${encodeURIComponent(skill)}+good+first+issue`,
}
);
}
```

```
    roadmap[skill] = steps;
}

res.json({ roadmap });
} catch (err) {
  console.error("Failed to generate roadmap:", err);
  res.status(500).json({ message: "Server error" });
}
});

// ✅ 3. Submit Skill Test Result
router.post("/skill-test/submit", authMiddleware, async (req, res) => {
  const { skill, score, total } = req.body;

  if (!skill || score == null || total == null) {
    return res.status(400).json({ message: "Incomplete data" });
  }

  try {
    const newResult = new SkillTestResult({
      user: req.user.id,
      skill: new mongoose.Types.ObjectId(skill),
      score: Number(score),
      total: Number(total),
    });

    await newResult.save();
    res.status(201).json({ message: "Result saved" });
  } catch (err) {
    console.error("Error saving test result:", err);
  }
});
```

```
res.status(500).json({ message: "Server error while saving result" });

}

});

// ✅ 4. Get Student Profile

router.get("/profile", authMiddleware, async (req, res) => {

try {

const user = await User.findById(req.user.id).select("-password").lean();

if (!user) return res.status(404).json({ message: "User not found" });

res.json(user);

} catch (err) {

console.error("Profile fetch error:", err);

res.status(500).json({ message: "Server error" });

}

});

// ✅ 5. Update Student Profile

router.post("/profile/update", authMiddleware, upload.single("resume"), async (req, res) => {

try {

const user = await User.findById(req.user.id);

if (!user) return res.status(404).json({ message: "User not found" });

user.name = req.body.name?.trim() || user.name;

user.email = req.body.email?.trim().toLowerCase() || user.email;

user.phone = req.body.phone?.trim() || user.phone;

user.university = req.body.university?.trim() || user.university;

user.education = req.body.education || user.education;

user.skills = req.body.skills?.split(",").map(s => s.trim()).filter(Boolean) || user.skills;

// Optional: handle uploaded resume if needed later

}

});
```

```
// if (req.file) user.resumePath = req.file.path;

await user.save();

res.json({ success: true, message: "Profile updated", user });

} catch (err) {

  console.error("Profile update error:", err);

  res.status(500).json({ message: "Update failed" });

}

});

module.exports = router;
```

---

PORT=5000

MONGO\_URI=mongodb+srv://thedigitalakr:9OlTvGfttDfMIdHM@cluster0.nvgmk5y.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0

JWT\_SECRET=4db26feed4877dab3fbb4f50600053b3410146e5e58f1d87f0389950d8e256849a5a78e17606cb8ea9bdb850793523c98233845f5ba34bcfb61d5179ab79dec8

ADMIN\_JWT\_SECRET=486eb36cd58a16fa9f5131bbe60ca632aa05787977e394f2451ef923cd3d341bf b7391b859483ce403c9c7aaceab1d781227478bdaaac662480860e44c1594eb

ADMIN\_EMAIL=admin@blyfto.com

ADMIN\_PASSWORD=Admin@123

ADMIN\_USERNAME=admin

ADMIN\_PHONE=0000000000

---

```
// server.js

const express = require("express");
const cors = require("cors");
const dotenv = require("dotenv");
const path = require("path");
const http = require("http");
const { Server } = require("socket.io");
```

```
const connectDB = require("./config/db");

const rooms = new Map();

dotenv.config();
connectDB();

const app = express();
app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// quick env visibility (safe, not printing secrets)
const show = (v) => (v ? "SET" : "MISSING");
console.log(`[env] JWT_SECRET: ${show(process.env.JWT_SECRET)}, ADMIN_JWT_SECRET: ${show(
  process.env.ADMIN_JWT_SECRET
)}\n`);

// ----- DEFAULT ADMIN BOOTSTRAP -----
const User = require("./models/User");
const bcrypt = require("bcryptjs");

async function bootstrapAdmin() {
  try {
    const {
      ADMIN_EMAIL,
      ADMIN_PASSWORD,
      ADMIN_USERNAME,
      ADMIN_PHONE,
    } = process.env;
  }
}
```

```
if (!ADMIN_EMAIL || !ADMIN_PASSWORD || !ADMIN_USERNAME || !ADMIN_PHONE) {
  console.warn(
    "[admin] Skipping default admin bootstrap (one or more envs missing: ADMIN_EMAIL,
ADMIN_PASSWORD, ADMIN_USERNAME, ADMIN_PHONE)"
  );
  return;
}

const email = ADMIN_EMAIL.toLowerCase().trim();
const username = ADMIN_USERNAME.trim();

let admin = await User.findOne({ email });

if (!admin) {
  const salt = await bcrypt.genSalt(10);
  const hash = await bcrypt.hash(ADMIN_PASSWORD, salt);
  admin = await User.create({
    name: "Platform Admin",
    username,
    email,
    phone: ADMIN_PHONE,
    password: hash,
    role: "admin",
  });
  console.log("[admin] Default admin created:", email);
} else {
  // ensure role is admin and password is up to date (only if you changed env)
  let toUpdate = {};
  if (admin.role !== "admin") toUpdate.role = "admin";
  if (admin.username !== username) toUpdate.username = username;
}
```

```

if (admin.phone !== ADMIN_PHONE) toUpdate.phone = ADMIN_PHONE;

// Optional: if you want to enforce env password on each boot, uncomment below:
const same = await bcrypt.compare(ADMIN_PASSWORD, admin.password || "");
if (!same) {
  const salt = await bcrypt.genSalt(10);
  toUpdate.password = await bcrypt.hash(ADMIN_PASSWORD, salt);
}

if (Object.keys(toUpdate).length) {
  await User.updateOne({ _id: admin._id }, { $set: toUpdate });
  console.log("[admin] Default admin ensured/updated:", email);
} else {
  console.log("[admin] Default admin already present:", email);
}
} catch (err) {
  console.error("[admin] Failed to bootstrap default admin:", err.message);
}

// Static (e.g., resumes)
app.use("/uploads", express.static(path.join(__dirname, "uploads")));

// ===== API Routes =====
app.use("/api/auth", require("./routes/auth.admin")); // admin auth
app.use("/api/auth", require("./routes/authRoutes")); // student/company auth
app.use("/api/student", require("./routes/student"));
app.use("/api/student/skill-test", require("./routes/student/skillTest"));

app.use("/api/admin/skills", require("./routes/admin/skills"));

```

```
app.use("/api/admin/question-sets", require("./routes/admin/sets"));

app.use("/api/admin/questions", require("./routes/admin/questions"));

app.use("/api/admin/results", require("./routes/admin/results"));

app.use("/api/admin", require("./routes/admin"));



app.use("/api/student/jobs", require("./routes/student/jobs"));

app.use("/api/student/webinars", require("./routes/student/webinars"));

app.use("/api/student/interviews", require("./routes/student/interviews"));

app.use("/api/student/hackathons", require("./routes/student/hackathons"));



app.use("/api/company/profile", require("./routes/company/profile"));

app.use("/api/company/jobs", require("./routes/company/jobs"));

app.use("/api/company/webinars", require("./routes/company/webinars"));

app.use("/api/company/applications", require("./routes/company/applications"));

app.use("/api/company/hackathons", require("./routes/company/hackathons"));

app.use("/api/company", require("./routes/company"));



app.use("/student/ai", require("./routes/student.ai"));



// ===== Socket.IO on SAME server =====

const server = http.createServer(app);

const io = new Server(server, {

  cors: { origin: "*", methods: ["GET", "POST"] },

});

app.set("io", io);



io.on("connection", (socket) => {

  const safeEmit = (id, evt, payload) => {

    try { io.to(id).emit(evt, payload); } catch {}

  };


```

```

socket.on("auth:identify", ({ userId }) => {
  if (!userId) return;
  socket.join(`user:${userId}`);
  socket.data.userId = userId;
});

// Host creates or reclaims a room

socket.on("createWebinar", ({ roomId, webinarName, hostName }) => {
  if (!roomId) roomId = `room_${Date.now()}`;
  if (!rooms.has(roomId)) {
    rooms.set(roomId, { host: socket.id, waiting: new Map(), participants: new Map() });
  } else {
    rooms.get(roomId).host = socket.id; // reclaim
  }
  socket.join(roomId);
  socket.data.roomId = roomId;
  socket.data.role = "host";
  socket.emit("webinarCreated", { roomId });

  // push initial lobby info to host
  const r = rooms.get(roomId);
  const waiting = [...r.waiting.values()];
  const participants = [...r.participants.values()];
  safeEmit(socket.id, "lobbyUpdate", { waiting, participants });
};

// Student requests to join a room (goes to waiting list first)

socket.on("requestJoin", ({ roomId, username }) => {
  const r = rooms.get(roomId);
  if (!r) return socket.emit("error", { message: "Webinar not found" });

  socket.join(roomId);
}

```

```

socket.data.roomId = roomId;
socket.data.role = "student";
socket.data.username = username || "Student";

r.waiting.set(socket.id, { socketId: socket.id, username: socket.data.username });

// notify host
safeEmit(r.host, "lobbyUpdate", {
  waiting: [...r.waiting.values()],
  participants: [...r.participants.values()],
});

});

// Host admits/denies
socket.on("admit", ({ roomId, socketIds }) => {
  const r = rooms.get(roomId);
  if (!r || socket.id !== r.host) return;
  const ids = Array.isArray(socketIds) ? socketIds : [socketIds];
  ids.forEach((sid) => {
    const u = r.waiting.get(sid);
    if (!u) return;
    r.waiting.delete(sid);
    r.participants.set(sid, u);
    safeEmit(sid, "admitted", { roomId });
  });
  safeEmit(r.host, "lobbyUpdate", {
    waiting: [...r.waiting.values()],
    participants: [...r.participants.values()],
  });
});

socket.on("deny", ({ roomId, socketIds }) => {

```

```

const r = rooms.get(roomId);
if (!r || socket.id === r.host) return;

const ids = Array.isArray(socketIds) ? socketIds : [socketIds];
ids.forEach((sid) => {
  r.waiting.delete(sid);
  safeEmit(sid, "denied", { roomId });
});

safeEmit(r.host, "lobbyUpdate", {
  waiting: [...r.waiting.values()],
  participants: [...r.participants.values()],
});

// --- WebRTC signaling (1:many; host is sender) ---
socket.on("signal-offer", ({ target, sdp }) => {
  safeEmit(target, "signal-offer", { from: socket.id, sdp });
});

socket.on("signal-answer", ({ target, sdp }) => {
  safeEmit(target, "signal-answer", { from: socket.id, sdp });
});

socket.on("signal-ice", ({ target, candidate }) => {
  safeEmit(target, "signal-ice", { from: socket.id, candidate });
});

// Chat (room-wide)
socket.on("chatMessage", ({ roomId, username, message }) => {
  io.to(roomId).emit("chatMessage", { username, message });
});

// Cleanup
socket.on("disconnect", () => {

```

```

const { roomId, role } = socket.data || {};
if (!roomId) return;
const r = rooms.get(roomId);
if (!r) return;

r.waiting.delete(socket.id);
r.participants.delete(socket.id);

if (role === "host") {
  io.to(roomId).emit("webinarEnded");
  rooms.delete(roomId);
} else {
  // notify host lobby change
  safeEmit(r.host, "lobbyUpdate", {
    waiting: [...r.waiting.values()],
    participants: [...r.participants.values()],
  });
}
});

// --- webinar signaling code unchanged (your existing handlers) ---
// ... (keep your createWebinar / requestJoin / admit / deny / signal-offer / signal-answer / signal-ice
// / chatMessage / disconnect exactly as you have)

// Start after bootstrapping admin
const PORT = process.env.PORT || 5000;
server.listen(PORT, async () => {
  await bootstrapAdmin();
  console.log(`Server + Socket.IO running on ${PORT}`);
});

```

