Project Code Files

File: C:\Projects\skill lab\day5\webapp\src\app\globals.css

```css
@import "tailwindcss";

:root {
  --background: #ffffff;
  --foreground: #171717;
}

@theme inline {
  --color-background: var(--background);
  --color-foreground: var(--foreground);
  --font-sans: var(--font-geist-sans);
  --font-mono: var(--font-geist-mono);
}

@media (prefers-color-scheme: dark) {
  :root {
    --background: #0a0a0a;
    --foreground: #ededed;
  }
}

body {
  background: var(--background);
  color: var(--foreground);
  font-family: Arial, Helvetica, sans-serif;
}
```

File: C:\Projects\skill lab\day5\webapp\src\app\layout.js

```js
import { Geist, Geist_Mono } from "next/font/google";
import "./globals.css";

const geistSans = Geist({
  variable: "--font-geist-sans",
  subsets: ["latin"],
});

const geistMono = Geist_Mono({
  variable: "--font-geist-mono",
  subsets: ["latin"],
});

export const metadata = {
  title: "Create Next App",
  description: "Generated by create next app",
};

export default function RootLayout({ children }) {
  return (
```

Project Code Files

```
    <html lang="en">
      <body className={`${geistSans.variable} ${geistMono.variable} antialiased`}>
        {children}
      </body>
    </html>
  );
}
```

File: C:\Projects\skill lab\day5\webapp\src\app\page.js

```
"use client";
import React from "react";
import Link from "next/link";
import { Database, UserPlus, FileText } from "lucide-react";

export default function StudentManagementPage() {
  return (
    <div className="min-h-screen flex flex-col items-center justify-center p-4 bg-gray-900 text-white">
      <h1 className="text-2xl font-bold mb-6">Student Management</h1>
      <Link
        href="/student/create"
        className="p-3 w-64 bg-gray-800 rounded mb-2 flex items-center justify-center"
      >
        <UserPlus className="w-5 h-5 mr-2" /> Add Student
      </Link>
      <Link
        href="/student/output"
        className="p-3 w-64 bg-gray-800 rounded mb-2 flex items-center justify-center"
      >
        <Database className="w-5 h-5 mr-2" /> View Student
      </Link>

      <Link
        href="/student/resume"
        className="p-3 w-64 bg-gray-800 rounded flex items-center justify-center"
      >
        <FileText className="w-5 h-5 mr-2" /> Generate Resume
      </Link>
      <p className="mt-4 text-gray-400 text-sm">
        Choose an option to manage student data
      </p>
    </div>
  );
}
```

File: C:\Projects\skill lab\day5\webapp\src\app\component\InputField.js

```
import React from "react";

export default function InputField({ type = "text", value, placeholder, onChange }) {
  return (
```

Project Code Files

```
    <input
      type={type}
      value={value}
      placeholder={placeholder}
      onChange={onChange}
      className="w-full p-2 border border-gray-300 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500"
    />
  );
}
```

File: C:\Projects\skill lab\day5\webapp\src\app\link\lib.js

```
import { createClient } from "@supabase/supabase-js";

const supabaseUrl = process.env.NEXT_PUBLIC_SUPABASE_URL;
const supabaseAnonKey = process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY;
export const supabase = createClient(supabaseUrl, supabaseAnonKey);
```

File: C:\Projects\skill lab\day5\webapp\src\app\student\create\page.js

```
// File: CreateStudent.js
"use client";
import React, { useState } from "react";
import InputField from "@/app/component/InputField";
import { supabase } from "@/app/link/lib";

export default function CreateStudent() {
  const [name, setName] = useState("");
  const [usn, setUSN] = useState("");
  const [email, setEmail] = useState("");
  const [phone, setPhone] = useState("");
  const [address, setAddress] = useState("");
  const [gender, setGender] = useState("");
  const [age, setAge] = useState("");

  const handleAddStudent = async () => {
    if (!usn || !name || !email || !phone || !address || !gender || !age) {
      alert("All fields are mandatory!");
      return;
    }

    try {
      const { data: existingStudent } = await supabase
        .from("student")
        .select("usn")
        .eq("usn", usn);

      if (existingStudent.length > 0) {
        alert("USN already exists!");
        return;
      }

      // Create timestamp for created_at
      const created_at = new Date().toISOString();

      const { data, error } = await supabase.from("student").insert([
        {
          name,
          usn,
          email,
          phone,
          address,
          gender,
          age,
          created_at,
        },
      ]);

      if (error) {
        alert("Error inserting data: " + error.message);
      } else {
        alert(`Student added successfully! ?\n ${JSON.stringify(data)}`);
        setName("");
        setUSN("");
        setEmail("");
        setPhone("");
        setAddress("");
        setGender("");
        setAge("");
      }
    } catch (err) {
      console.error(`Unexpected error: ${JSON.stringify(err)}`);
    }
  };

  return (
    <div className="min-h-screen flex justify-center items-center bg-black text-white">
      <div className="bg-gray-800 p-6 rounded-lg shadow-lg w-160">
        <h1 className="text-2xl font-bold text-center mb-4 text-white">
          Create Student
        </h1>

        <div className="space-y-3">
          <InputField
            type="text"
            value={name}
            placeholder="Student Name"
            onChange={(e) => setName(e.target.value)}
            className="bg-white text-black"
          />
          <InputField
            type="text"
            value={usn}
            placeholder="Student USN"
            onChange={(e) => setUSN(e.target.value)}
            className="bg-white text-black"
          />
          <InputField
            type="email"
            value={email}
            placeholder="Student Email"
            onChange={(e) => setEmail(e.target.value)}
            className="bg-white text-black"
          />
          <InputField
            type="text"
            value={address}
            placeholder="Student Address"
            onChange={(e) => setAddress(e.target.value)}
            className="bg-white text-black"
          />
          <InputField
            type="number"
            value={phone}
            placeholder="Phone Number"
            onChange={(e) => setPhone(e.target.value)}
            className="bg-white text-black"
          />
          <InputField
            type="number"
            value={age}
            placeholder="Age"
            onChange={(e) => setAge(e.target.value)}
            className="bg-white text-black"
          />

          <select
            value={gender}
            onChange={(e) => setGender(e.target.value)}
            className="w-full p-2 border border-gray-300 rounded-md bg-white text-black"
          >
            <option value="">Select Gender</option>
            <option value="Male">Male</option>
            <option value="Female">Female</option>
            <option value="Other">Other</option>
          </select>

          <button
            onClick={handleAddStudent}
            className="w-full bg-gray-700 text-white p-2 rounded-md hover:bg-gray-600"
          >
            Add Student
```

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

File: C:\Projects\skill lab\day5\webapp\src\app\student\output\page.js

```
// File: FetchStudents.js
"use client";
import React, { useState, useEffect } from "react";
import { supabase } from "@/app/link/lib";

export default function FetchStudents() {
  const [students, setStudents] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [searchTerm, setSearchTerm] = useState("");
  const [viewMode, setViewMode] = useState("table"); // "table" or "card"

  useEffect(() => {
    fetchStudents();

    const handleResize = () => {
      if (window.innerWidth < 768) {
        setViewMode("card");
      } else {
        setViewMode("table");
      }
    };


    handleResize();

    // Add resize listener
    window.addEventListener("resize", handleResize);

    // Clean up
    return () => window.removeEventListener("resize", handleResize);
  }, []);

  const fetchStudents = async () => {
    try {
      setLoading(true);

      const { data, error } = await supabase
        .from("student")
        .select("*")
        .order("created_at", { ascending: false });
    if (error) {
      setError("Error fetching students: " + error.message);
      console.error("Error fetching students:", error);
    } else {
      setStudents(data || []);
    }
  } catch (err) {
    setError("Unexpected error occurred");
    console.error("Unexpected error:", err);
  } finally {
    setLoading(false);
  }
};

const handleDelete = async (usn) => {
  if (window.confirm("Are you sure you want to delete this student?")) {
    try {
      const { error } = await supabase
        .from("student")
        .delete()
        .eq("usn", usn);

      if (error) {
        alert("Error deleting student: " + error.message);
      } else {
        alert("Student deleted successfully!");
        fetchStudents(); // Refresh the list
      }
    } catch (err) {
      console.error("Unexpected error during deletion:", err);
      alert("An unexpected error occurred");
    }
  }
};

const filteredStudents = students.filter(
  (student) =>
    student.name?.toLowerCase().includes(searchTerm.toLowerCase()) ||
    student.usn?.toLowerCase().includes(searchTerm.toLowerCase())
);

// Card view component for each student
const StudentCard = ({ student }) => (
  <div className="bg-gray-800 p-4 rounded-lg shadow-md mb-4">
    <div className="flex justify-between items-center mb-2">
      <h3 className="text-lg font-semibold">{student.name}</h3>
      <span className="text-sm bg-gray-700 px-2 py-1 rounded">
        {student.usn}
      </span>
    </div>
    <div className="space-y-1 text-sm text-gray-300">
```

```
      <p>
        <span className="font-medium">Email:</span> {student.email}
      </p>
      <p>
        <span className="font-medium">Phone:</span> {student.phone}
      </p>
      <p>
        <span className="font-medium">Age:</span> {student.age}
      </p>
      <p>
        <span className="font-medium">Gender:</span> {student.gender}
      </p>
      <p>
        <span className="font-medium">Address:</span> {student.address}
      </p>
    </div>
    <div className="mt-3">
      <button
        onClick={() => handleDelete(student.usn)}
        className="bg-red-600 text-white px-3 py-1 rounded text-sm hover:bg-red-700 w-full"
      >
        Delete
      </button>
    </div>
  </div>
);

return (
  <div className="min-h-screen bg-black text-white p-3 md:p-6">
    <h1 className="text-xl md:text-2xl font-bold text-center mb-4 md:mb-6">
      Student Records
    </h1>

    <div className="mb-4">
      <div className="flex flex-col sm:flex-row sm:items-center gap-3">
        <div className="w-full">
          <input
            type="text"
            placeholder="Search by name or USN..."
            value={searchTerm}
            onChange={(e) => setSearchTerm(e.target.value)}
            className="w-full p-0.5 border border-gray-300 rounded-md bg-white text-black"
          />
        </div>
        <div className="flex-shrink-0 flex justify-end">
          <button
            onClick={() => setViewMode("table")}
            className={`px-3 py-1 rounded-l ${
              viewMode === "table" ? "bg-gray-600" : "bg-gray-800"
            }`}
          >
```

Project Code Files

```
        Table
      </button>
      <button
        onClick={() => setViewMode("card")}
        className={`px-3 py-1 rounded-r ${
          viewMode === "card" ? "bg-gray-600" : "bg-gray-800"
        }`}
      >
        Cards
      </button>
    </div>
  </div>
</div>

{loading ? (
  <div className="text-center py-10">
    <p className="text-xl">Loading students...</p>
  </div>
) : error ? (
  <div className="text-center py-10 text-red-500">
    <p>{error}</p>
  </div>
) : filteredStudents.length === 0 ? (
  <p className="text-center py-10">No students found</p>
) : viewMode === "table" ? (
  <div className="overflow-x-auto">
    <table className="min-w-full bg-gray-800 rounded-lg overflow-hidden">
      <thead className="bg-gray-700">
        <tr>
          <th className="px-3 py-2 text-left text-sm md:text-base">
            Name
          </th>
          <th className="px-3 py-2 text-left text-sm md:text-base">
            USN
          </th>
          <th className="px-3 py-2 text-left text-sm md:text-base hidden md:table-cell">
            Email
          </th>
          <th className="px-3 py-2 text-left text-sm md:text-base hidden md:table-cell">
            Phone
          </th>
          <th className="px-3 py-2 text-left text-sm md:text-base hidden md:table-cell">
            Age
          </th>
          <th className="px-3 py-2 text-left text-sm md:text-base hidden md:table-cell">
            Gender
          </th>
          <th className="px-3 py-2 text-center text-sm md:text-base">
            Actions
          </th>
        </tr>
```

```
            </thead>
            <tbody>
              {filteredStudents.map((student) => (
                <tr
                  key={student.usn}
                  className="border-t border-gray-600 hover:bg-gray-700"
                >
                  <td className="px-3 py-2 text-sm md:text-base">
                    {student.name}
                  </td>
                  <td className="px-3 py-2 text-sm md:text-base">
                    {student.usn}
                  </td>
                  <td className="px-3 py-2 text-sm md:text-base hidden md:table-cell">
                    {student.email}
                  </td>
                  <td className="px-3 py-2 text-sm md:text-base hidden md:table-cell">
                    {student.phone}
                  </td>
                  <td className="px-3 py-2 text-sm md:text-base hidden md:table-cell">
                    {student.age}
                  </td>
                  <td className="px-3 py-2 text-sm md:text-base hidden md:table-cell">
                    {student.gender}
                  </td>
                  <td className="px-3 py-2 text-center">
                    <button
                      onClick={() => handleDelete(student.usn)}
                      className="bg-red-600 text-white px-2 py-1 rounded text-sm hover:bg-red-700"
                    >
                      Delete
                    </button>
                  </td>
                </tr>
              ))}
            </tbody>
          </table>
        </div>
      ) : (
        // Card View
        <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-4">
          {filteredStudents.map((student) => (
            <StudentCard key={student.usn} student={student} />
          ))}
        </div>
      )}
    </div>
  );
}
```

Project Code Files

File: C:\Projects\skill lab\day5\webapp\src\app\student\resume\page.js

```
"use client";
import React, { useState, useEffect } from "react";
import { supabase } from "@/app/link/lib";
import { FileIcon, Download, RefreshCw, Loader } from "lucide-react";

export default function StudentResumeGenerator() {
  const [students, setStudents] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [searchTerm, setSearchTerm] = useState("");
  const [selectedStudent, setSelectedStudent] = useState(null);
  const [generatingResume, setGeneratingResume] = useState(false);
  const [resumeContent, setResumeContent] = useState("");
  const [displayText, setDisplayText] = useState("");
  const [isTyping, setIsTyping] = useState(false);
  const [typingIndex, setTypingIndex] = useState(0);

  const API_KEY = process.env.NEXT_PUBLIC_GOOGLE_API_KEY;

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

  useEffect(() => {
    if (resumeContent && !isTyping) {
      setDisplayText("");
      setTypingIndex(0);
      setIsTyping(true);
    }
  }, [resumeContent]);

  useEffect(() => {
    let timer;
    if (isTyping && typingIndex < resumeContent.length) {
      timer = setTimeout(() => {
        setDisplayText((prev) => prev + resumeContent.charAt(typingIndex));
        setTypingIndex((prev) => prev + 1);
      }, 30);
    } else if (isTyping && typingIndex >= resumeContent.length) {
      setIsTyping(false);
    }
    return () => {
      if (timer) clearTimeout(timer);
    };
  }, [isTyping, typingIndex, resumeContent]);

  const fetchStudents = async () => {
    try {
      setLoading(true);
```

Project Code Files

```
      const { data, error } = await supabase
        .from("student")
        .select("*")
        .order("created_at", { ascending: false });

      if (error) {
        setError("Error fetching students: " + error.message);
        console.error("Error fetching students:", error);
      } else {
        setStudents(data || []);
      }
    } catch (err) {
      setError("Unexpected error occurred");
      console.error("Unexpected error:", err);
    } finally {
      setLoading(false);
    }
  };

  const generateResume = async (student) => {
    if (!student) return;

    setSelectedStudent(student);
    setGeneratingResume(true);
    setResumeContent("");

    try {
      const response = await fetch(

`https://generativelanguage.googleapis.com/v1beta/models/gemini-2.0-flash:generateContent?key=${API_KEY}`,
        {
          method: "POST",
          headers: { "Content-Type": "application/json" },
          body: JSON.stringify({
            contents: [
              {
                parts: [
                  {
                    text: `Create a professional resume for a student with the following information:

                    Name: ${student.name}
                    USN: ${student.usn}
                    Email: ${student.email}
                    Phone: ${student.phone}
                    Age: ${student.age}
                    Gender: ${student.gender}
                    Address: ${student.address}

                       Please include sections for Education, Skills, Objective, and Contact Information in a
clean, professional format.
```

```
                  Use markdown formatting for the resume.`,
                },
              ],
            },
          ],
        }),
      }
    );

    const data = await response.json();

    if (!response.ok) {
      setError(data.error?.message || "Failed to generate resume");
      return;
    }

    const resumeText =
      data.candidates?.[0]?.content?.parts?.[0]?.text ||
      "No response from AI.";
    setResumeContent(resumeText);
  } catch (err) {
    setError("An error occurred while generating the resume");
    console.error(err);
  } finally {
    setGeneratingResume(false);
  }
};

const downloadResume = () => {
  if (!resumeContent || !selectedStudent) return;

  const blob = new Blob([resumeContent], { type: "text/markdown" });
  const url = URL.createObjectURL(blob);
  const a = document.createElement("a");
  a.href = url;
  a.download = `${selectedStudent.name.replace(/\s+/g, "_")}_Resume.md`;
  document.body.appendChild(a);
  a.click();
  document.body.removeChild(a);
  URL.revokeObjectURL(url);
};

const filteredStudents = students.filter(
  (student) =>
    student.name?.toLowerCase().includes(searchTerm.toLowerCase()) ||
    student.usn?.toLowerCase().includes(searchTerm.toLowerCase())
);

return (
  <div className="min-h-screen bg-black text-white p-3 md:p-6">
    <h1 className="text-xl md:text-2xl font-bold text-center mb-4 md:mb-6">
```

Project Code Files

```
  Student Resume Generator
</h1>

<div className="grid grid-cols-1 md:grid-cols-2 gap-6">
  <div className="bg-gray-900 p-4 rounded-lg shadow-md">
    <h2 className="text-lg font-semibold mb-4">Select Student</h2>

    <div className="mb-4">
      <input
        type="text"
        placeholder="Search by name or USN..."
        value={searchTerm}
        onChange={(e) => setSearchTerm(e.target.value)}
        className="w-full p-2 border border-gray-300 rounded-md bg-gray-800 text-white"
      />
    </div>

    {loading ? (
      <div className="text-center py-10">
        <p className="text-xl">Loading students...</p>
      </div>
    ) : error ? (
      <div className="text-center py-10 text-red-500">
        <p>{error}</p>
      </div>
    ) : filteredStudents.length === 0 ? (
      <p className="text-center py-10">No students found</p>
    ) : (
      <div className="max-h-96 overflow-y-auto">
        {filteredStudents.map((student) => (
          <div
            key={student.usn}
            className={`p-3 mb-2 rounded-lg cursor-pointer ${
              selectedStudent?.usn === student.usn
                ? "bg-blue-700"
                : "bg-gray-800 hover:bg-gray-700"
            }`}
            onClick={() => setSelectedStudent(student)}
          >
            <div className="flex justify-between items-center">
              <h3 className="font-medium">{student.name}</h3>
              <span className="text-sm bg-gray-900 px-2 py-1 rounded">
                {student.usn}
              </span>
            </div>
            <p className="text-sm text-gray-300 mt-1">{student.email}</p>
          </div>
        ))}
      </div>
    )}
  </div>
```

Project Code Files

```
<div className="bg-gray-900 p-4 rounded-lg shadow-md flex flex-col">
  <div className="flex justify-between items-center mb-4">
    <h2 className="text-lg font-semibold">Resume</h2>

    <div className="flex space-x-2">
      {selectedStudent && (
        <button
          onClick={() => generateResume(selectedStudent)}
          disabled={generatingResume}
          className={`flex items-center space-x-1 p-2 rounded-lg ${
            generatingResume
              ? "bg-gray-700 cursor-not-allowed"
              : "bg-blue-600 hover:bg-blue-700"
          }`}
        >
          {generatingResume ? (
            <Loader className="w-4 h-4 animate-spin" />
          ) : (
            <RefreshCw className="w-4 h-4" />
          )}
          <span>{generatingResume ? "Generating..." : "Generate"}</span>
        </button>
      )}

      {resumeContent && !isTyping && (
        <button
          onClick={downloadResume}
          className="flex items-center space-x-1 p-2 bg-green-600 hover:bg-green-700 rounded-lg"
        >
          <Download className="w-4 h-4" />
          <span>Download</span>
        </button>
      )}
    </div>
  </div>

  {selectedStudent ? (
    <div className="flex-grow flex flex-col">
      {!resumeContent && !generatingResume && (
        <div className="flex-grow flex items-center justify-center">
          <div className="text-center text-gray-400">
            <FileIcon className="w-12 h-12 mx-auto mb-2" />
            <p>
              Select a student and click Generate to create a resume
            </p>
          </div>
        </div>
      )}

      {generatingResume && !resumeContent && (
```

Project Code Files

```
                <div className="flex-grow flex items-center justify-center">
                  <div className="text-center">
                    <Loader className="w-12 h-12 mx-auto mb-2 animate-spin text-blue-500" />
                    <p>Generating resume for {selectedStudent.name}...</p>
                  </div>
                </div>
              )}

              {(resumeContent || isTyping) && (
                <div className="flex-grow bg-gray-800 rounded-lg p-4 overflow-y-auto whitespace-pre-wrap">
                  {isTyping ? (
                    <>
                      {displayText}
                      <span className="inline-block animate-pulse text-xl">
                        ?
                      </span>
                    </>
                  ) : (
                    resumeContent
                  )}
                </div>
              )}
            </div>
          ) : (
            <div className="flex-grow flex items-center justify-center text-gray-400">
              <p>Select a student to generate a resume</p>
            </div>
          )}
        </div>
      </div>
    </div>
  );
}
```