Resume Builder

Project Submitted to the
SRM University AP, Andhra Pradesh
for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology in
Computer Science & Engineering
School of Engineering & Sciences submitted by

Praveen Ramisetti - AP23110011234
P. Dharma Teja - AP23110011255
P. V. Thapan - AP23110010044
B. Saketh - AP23110010789

Under the Guidance of
Mr. Yatharth Shahrawat

Department of Computer Science & Engineering
SRM University-Ap
Neerukonda, Mangalgiri, Guntur
Andhra Pradesh - 522 240
December – 2025

Resume Builder

Introduction:

In today's competitive job market, a well-structured and visually appealing resume plays a crucial role in creating a strong first impression with recruiters. However, many students and job seekers struggle with resume design, formatting, and presenting their information professionally. To address this challenge, digital resume-building tools have become increasingly popular, offering users an easy and efficient way to create polished resumes without requiring design skills.

The **Resume Builder Web Application** developed in this project aims to simplify and automate the resume creation process. Built using **React.js**, the application allows users to enter their personal information, skills, educational background, and work experience through intuitive and user-friendly forms. As users fill in the details, the resume preview updates in real time, ensuring accuracy and providing instant feedback.

The application also offers clean and modern templates, proper formatting, and the ability to download the final resume as a **PDF**, making it suitable for students, freshers, and job seekers who need professional resumes quickly. The project focuses on ease of use, responsive design, and efficient data handling, providing a seamless experience across devices.

Overall, this Resume Builder project demonstrates the importance of accessible digital tools in career development and showcases how modern frontend frameworks like React can be used to create powerful, interactive, and user-centric applications.

Scenario-Based Introduction:-

Imagine a final-year student preparing for campus placements. They have all their achievements, skills, and project details ready — but when it comes to creating a professional resume, they struggle with formatting, choosing a layout, and presenting information neatly. Each time they make a small edit, the entire document shifts, ruining the alignment. With deadlines approaching, the student needs a simple and reliable way to build a clean, professional resume without worrying about design skills.

The **Resume Builder Web Application** solves exactly this problem. Through an intuitive interface built with **React.js**, users can fill in their details step-by-step while

viewing a real-time preview of their resume. There is no need to adjust margins or worry about layout — the system automatically formats the content and provides a polished template. Once completed, the resume can be downloaded instantly as a **PDF**, making the entire process quick and stress-free.

In a world where first impressions matter, this application empowers students, freshers, and job seekers to present themselves professionally with minimal effort.

Target Audience:-

The Resume Builder web application is designed for users who need a quick and hassle-free way to create professional resumes. The primary target audience includes:

1. Students and Fresh Graduates:

- Individuals applying for internships, campus placements, or their first job.
- Users who may lack design skills or experience with resume formatting.

2. Job Seekers:

- Candidates looking to update or recreate their resumes in a clean and structured format.
- Users who need multiple resume versions for different job roles.

3. Professionals Switching Careers:

- Individuals transitioning to new roles or industries who require a modern, updated resume.

4. People with Limited Technical or Design Knowledge:

- Users who prefer simple forms and automated formatting instead of complex tools like MS Word or graphic design software.

5. Training Institutes & Career Services:

- Colleges, coaching centers, or career-guidance platforms helping students quickly produce polished resumes.

Project Goals and Objectives:-

Project Goals:

The main goal of the Resume Builder application is to provide users with a simple, efficient, and user-friendly tool to create professional resumes without needing design or technical skills. The project aims to streamline resume creation and offer clean, formatted templates that help users present their information effectively.

Objectives:

1. **Simplify Resume Creation**
   Provide an intuitive interface where users can enter their details easily through structured forms.

2. **Enable Real-Time Resume Preview**
   Update the resume layout instantly as users input or modify their information.

3. **Ensure Professional Formatting**
   Offer clean and visually appealing templates that automatically arrange content in a readable format.

4. **Support Export as PDF**
   Allow users to download their completed resume in PDF format for job applications.

5. **Provide Responsive and Accessible Design**
   Ensure the application works smoothly on various devices, including desktops and tablets.

6. **Reduce Manual Design Effort**
   Automate alignment, spacing, and layout so users do not need design tools or formatting skills.

7. **Create a Beginner-Friendly Workflow**
   Help students and job seekers quickly build a polished resume with minimal steps.

Key Features:-

1. User-Friendly Form Inputs

Users can easily enter their personal details, skills, education, and experience through simple and well-organized forms, making the process quick and intuitive.

2. Real-Time Resume Preview

As users fill in the information, the resume preview updates instantly. This helps users see how their final resume will look without needing additional formatting.

3. Clean and Professional Templates

The application offers neat and visually appealing templates that automatically format the content, ensuring a polished and recruiter-friendly resume.

4. PDF Download Option

Users can download the completed resume as a **PDF** with a single click, making it ready for job applications, email submissions, or printing.

5. Responsive Design

The layout adapts to different screen sizes, allowing users to create or edit resumes from desktops, laptops, or tablets comfortably.

6. Auto-Formatting and Alignment

The system automatically handles spacing, alignment, and layout structure, eliminating the need for design skills or manual adjustments.

7. Real-Time Editing

Users can modify any section at any time, and the changes reflect immediately in the preview, ensuring accuracy and flexibility.

8. Minimalistic and Fast Interface

The application loads quickly, uses lightweight components, and avoids unnecessary complexity, giving users a smooth experience.

9. Beginner-Friendly Workflow

The step-by-step approach helps first-time users easily understand and complete their resume without confusion.

10. Error-Free Resume Creation

Proper validation and form structure reduce chances of missing important details, ensuring the resume remains complete and well-organized.

Pre-Requisites:-

To develop, run, and test the Resume Builder web application, the following prerequisites are required:

1. Technical Knowledge:

- Basic understanding of **HTML, CSS, and JavaScript**
- Familiarity with **React.js** concepts such as components, props, state, and hooks
- Knowledge of **npm** (Node Package Manager) commands

2. Software Requirements:

- **Node.js** (v14 or above) installed on the system
- A code editor such as **Visual Studio Code**
- Any modern web browser (Chrome, Edge, Firefox)

3. Libraries & Dependencies:

- React.js and React DOM
- react-to-print (or equivalent library) for PDF generation
- CSS frameworks or custom styling (as used in the project)

4. System Requirements:

- Minimum 4 GB RAM (for smooth development)
- Stable internet connection for installing dependencies

5. Optional Tools:

- Git/GitHub for version control
- Browser developer tools for debugging
- PDF viewer to preview downloaded resumes

Project Structure:-

The Resume Builder project follows a simple React.js folder structure:

- **src/**
  - **components/** – Contains reusable components such as forms and resume preview.
  - **pages/** – Holds main screens/pages of the application.
  - **assets/** – Stores images, icons, and styling files.
  - **App.js** – Root component managing routing and layout.
  - **index.js** – Entry point that renders the React application.
- **public/** – Contains the base HTML template and static assets.
- **package.json** – Manages project dependencies and scripts.

PROJECT FLOW:

Project demo:

Before starting to work on this project, let's see the demo.

Demo Link:
https://drive.google.com/file/d/1UJBWwk0uH8r2iGkSng5Xn7uJQ6ZhHUrM/view?usp=sharing

Use the code in: https://github.com/Praveenramisetti76/Resume-Builder

Milestone 1: Project Setup and Configuration

1. Install Required Tools and Software

To begin building the Resume Builder web application, the necessary development tools and libraries must be installed.
Follow the steps below to set up the environment properly.

Installation of Required Tools

Open the project folder and install all required dependencies using **npm**.
This project uses the following technologies and libraries:

- **React.js**
  Used to create the overall user interface and manage dynamic resume updates.
- **React Icons**
  Provides icons for UI enhancement within forms and template display.
- **Material-UI (MUI)**
  Used for styling form components and layout elements to ensure a modern design.
- **react-to-print / PDF library**
  Enables downloading the formatted resume as a PDF document.

2. Reference Documentation

Use the following official resources for setup guidance and component usage:

- React Installation Guide
  https://react.dev/learn/installation
- Material UI Documentation
  https://mui.com/getting-started/installation/

Milestone 2: Web Development

1. Setup React Application

• Create React Application

- Initialize the project using npx create-react-app resume-builder.
- A default React folder structure is generated with src, public, and configuration files.

• Configure Routing

- Although the Resume Builder is single-page oriented, internal component routing is managed through App.js.
- Components such as Form Section and Resume Preview are rendered conditionally or side-by-side without page reloads.

• Install Required Libraries

- Libraries are installed to support UI, icons, and resume export

App.js Component:

```
import React from "react";
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import { ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import Home from "./pages/Home";
import CreateEditResume from "./pages/CreateEditResume";
import Dashboard from "./pages/Dashboard";
import ViewResume from "./pages/ViewResume";
import Navigation from "./components/Navigation";
import Footer from "./components/Footer";

export default function App() {
  return (
    <Router>
      <div className="flex flex-col min-h-screen">
        <ToastContainer
          position="top-right"
          autoClose={3000}
```

```
          hideProgressBar={false}
          newestOnTop={true}
          closeOnClick
          rtl={false}
          pauseOnFocusLoss
          draggable
          pauseOnHover
          theme="light"
        />
        <Navigation />
        <main className="grow">
          <Routes>
            <Route path="/" element={<Home />} />
            <Route path="/create" element={<CreateEditResume />} />
            <Route path="/edit/:id" element={<CreateEditResume />} />
            <Route path="/view/:id" element={<ViewResume />} />
            <Route path="/resumes" element={<Dashboard />} />
          </Routes>
        </main>
        <Footer />
      </div>
    </Router>
  );
}
```

Code Description:

- ./App.css: Styles the root layout and spacing.
- Home from ./components/Home: Main interface where users input details and preview resume.
- DataProvider from ./context/DataProvider: Used to manage user-entered resume data globally.
- App() returns React JSX with <DataProvider> wrapping <Home> so all components inside have state access.
- <DataProvider> ensures resume form inputs and preview remain synced live.
- <Home /> acts as the default landing and editing view.
- export default App makes App accessible for rendering in index.js.

2. Design UI Components

Create Components:

Personalinfo.js:

```
import React from "react";

export default function PersonalInfo({ data, setData }) {
  const handleChange = (e) => {
    const { name, value } = e.target;
    setData({ ...data, [name]: value });
  };

  return (
    <div className="bg-white rounded-2xl shadow-lg p-8 border border-gray-200
hover:shadow-xl transition-shadow">
      <h2 className="text-3xl font-bold text-gray-900 mb-8 flex items-center
gap-2">👤 Personal Information</h2>

      <div className="space-y-6">
        <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
          <div>
            <label className="block text-sm font-bold text-gray-700 mb-3
uppercase tracking-wide">Full Name</label>
            <input
              type="text"
              name="name"
              value={data.name}
              onChange={handleChange}
              placeholder="Enter your full name"
              className="w-full px-5 py-3 border-2 border-gray-200 rounded-lg
focus:outline-none focus:border-indigo-500 focus:ring-2 focus:ring-indigo-100
transition-all text-base"
            />
          </div>

          <div>
            <label className="block text-sm font-bold text-gray-700 mb-3
uppercase tracking-wide">Email</label>
            <input
              type="email"
              name="email"
              value={data.email}
              onChange={handleChange}
              placeholder="your.email@example.com"
              className="w-full px-5 py-3 border-2 border-gray-200 rounded-lg
focus:outline-none focus:border-indigo-500 focus:ring-2 focus:ring-indigo-100
transition-all text-base"
            />
          </div>
        </div>

        <div>
```

```
          <label className="block text-sm font-bold text-gray-700 mb-3
uppercase tracking-wide">Phone Number</label>
          <input
            type="tel"
            name="phone"
            value={data.phone}
            onChange={handleChange}
            placeholder="+1 (555) 123-4567"
            className="w-full px-5 py-3 border-2 border-gray-200 rounded-lg
focus:outline-none focus:border-indigo-500 focus:ring-2 focus:ring-indigo-100
transition-all text-base"
          />
        </div>

        <div>
          <label className="block text-sm font-bold text-gray-700 mb-3
uppercase tracking-wide">Professional Summary</label>
          <textarea
            name="summary"
            value={data.summary || ""}
            onChange={handleChange}
            placeholder="Brief summary of your professional background and key
achievements..."
            rows="5"
            className="w-full px-5 py-3 border-2 border-gray-200 rounded-lg
focus:outline-none focus:border-indigo-500 focus:ring-2 focus:ring-indigo-100
transition-all text-base resize-none"
          />
        </div>
      </div>
    </div>
  );
}
```

Code Description:

Code Description:

- **Education.js**: Component responsible for dynamically managing multiple education entries in the resume.
- **Props Used**:
    - data: Array holding all education objects (degree, institution, field, year, grade).
    - setData: Updates the education list in real-time whenever changes occur.
- **handleChange(index, key, value)**:
    - Updates a specific education entry without affecting others.

- Uses spread syntax to clone list and modify only targeted field:
- const newEdu = [...data];
- newEdu[index][key] = value;
- setData(newEdu);

- **addEducation()**:
  - Appends a new empty education object to the list.
  - Enables users to add multiple degrees.
- **removeEducation(index)**:
  - Deletes selected education entry from the array.
  - Uses filter to remove by index.
- **Dynamic Rendering**:
  - data.map() loops through each education entry and displays form fields.
  - Each entry includes Degree, Graduation Year, Institution, Field of Study, and Grade.
- **Remove Education Button**:
  - Displayed only if more than one entry exists (data.length > 1).
  - Styled with red gradient, hover transitions, and icon.
- **Add More Education Button**:
  - Positioned after all entries.
  - Uses react-icons (FiPlus).
  - Adds new blank education form with smooth UI feedback.
- **Styling**:
  - Tailwind CSS used heavily for spacing, borders, hover effects, shadows, gradients, and transitions.
  - Card-like section for each education entry with soft color gradients for visual clarity.
- **Component Export**:
- export default function Education
- Makes Education module reusable wherever academic details are needed.

Education.js :

```
import React from "react";
import { FiTrash2, FiPlus } from "react-icons/fi";

export default function Education({ data, setData }) {
  const handleChange = (index, key, value) => {
    const newEdu = [...data];
    newEdu[index][key] = value;
    setData(newEdu);
  };

  const addEducation = () => setData([...data, { degree: "", institution: "",
field: "", year: "", grade: "" }]);

  const removeEducation = (index) => {
    setData(data.filter((_, i) => i !== index));
  };

  return (
    <div className="bg-white rounded-2xl shadow-lg p-8 border border-gray-200
hover:shadow-xl transition-shadow">
      <h2 className="text-3xl font-bold text-gray-900 mb-8 flex items-center
gap-2">🎓 Education</h2>
      <div className="space-y-8">
        {data.map((edu, i) => (
          <div key={i} className="p-8 border-2 border-green-100 rounded-xl
space-y-6 bg-linear-to-br from-white to-green-50 hover:border-green-300
transition-colors shadow-md hover:shadow-lg">
            <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
              <div>
                <label className="block text-sm font-bold text-gray-700 mb-3
uppercase tracking-wide">Degree</label>
                <input
                  placeholder="e.g., Bachelor of Science"
                  value={edu.degree}
                  onChange={(e) => handleChange(i, "degree", e.target.value)}
                  className="w-full px-5 py-3 border-2 border-gray-200
rounded-lg focus:outline-none focus:border-indigo-500 focus:ring-2 focus:ring-
indigo-100 transition-all text-base"
                />
              </div>
              <div>
                <label className="block text-sm font-bold text-gray-700 mb-3
uppercase tracking-wide">Graduation Year</label>
                <input
                  placeholder="e.g., 2020"
                  value={edu.year}
                  onChange={(e) => handleChange(i, "year", e.target.value)}
                  className="w-full px-5 py-3 border-2 border-gray-200
rounded-lg focus:outline-none focus:border-indigo-500 focus:ring-2 focus:ring-
indigo-100 transition-all text-base"
                />
              </div>
```

```
            </div>
            <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
              <div>
                <label className="block text-sm font-bold text-gray-700 mb-3
uppercase tracking-wide">Institution</label>
                <input
                  placeholder="e.g., Tech University"
                  value={edu.institution}
                  onChange={(e) => handleChange(i, "institution",
e.target.value)}
                  className="w-full px-5 py-3 border-2 border-gray-200
rounded-lg focus:outline-none focus:border-indigo-500 focus:ring-2 focus:ring-
indigo-100 transition-all text-base"
                />
              </div>
              <div>
                <label className="block text-sm font-bold text-gray-700 mb-3
uppercase tracking-wide">Field of Study</label>
                <input
                  placeholder="e.g., Computer Science"
                  value={edu.field || ""}
                  onChange={(e) => handleChange(i, "field", e.target.value)}
                  className="w-full px-5 py-3 border-2 border-gray-200
rounded-lg focus:outline-none focus:border-indigo-500 focus:ring-2 focus:ring-
indigo-100 transition-all text-base"
                />
              </div>
              <div>
                <label className="block text-sm font-bold text-gray-700 mb-3
uppercase tracking-wide">Grade</label>
                <input
                  placeholder="e.g., 3.8 GPA"
                  value={edu.grade || ""}
                  onChange={(e) => handleChange(i, "grade", e.target.value)}
                  className="w-full px-5 py-3 border-2 border-gray-200
rounded-lg focus:outline-none focus:border-indigo-500 focus:ring-2 focus:ring-
indigo-100 transition-all text-base"
                />
              </div>
            </div>
            {data.length > 1 && (
              <button
                onClick={() => removeEducation(i)}
                className="w-full bg-linear-to-r from-red-500 to-red-600
hover:from-red-600 hover:to-red-700 text-white font-bold py-3 px-4 rounded-lg
flex items-center justify-center gap-2 transition-all duration-300 shadow-md
hover:shadow-lg"
              >
                <FiTrash2 size={18} /> Remove Education
              </button>
            )}
          </div>
        ))}
```

```
      </div>
      <button
        onClick={addEducation}
        className="w-full mt-8 bg-linear-to-r from-indigo-600 to-indigo-700
hover:from-indigo-700 hover:to-indigo-800 text-white font-bold py-3 px-4
rounded-lg flex items-center justify-center gap-2 transition-all duration-300
shadow-lg hover:shadow-xl transform hover:-translate-y-1"
      >
        <FiPlus size={18} /> Add More Education
      </button>
    </div>
  );
}
```

Code Description:

- **Education.js**: Component responsible for dynamically managing multiple education entries in the resume.
- **Props Used**:
  - data: Array holding all education objects (degree, institution, field, year, grade).
  - setData: Updates the education list in real-time whenever changes occur.
- **handleChange(index, key, value)**:
  - Updates a specific education entry without affecting others.
  - Uses spread syntax to clone list and modify only targeted field:
  - const newEdu = [...data];
  - newEdu[index][key] = value;
  - setData(newEdu);
- **addEducation()**:
  - Appends a new empty education object to the list.
  - Enables users to add multiple degrees.
- **removeEducation(index)**:
  - Deletes selected education entry from the array.
  - Uses filter to remove by index.
- **Dynamic Rendering**:
  - data.map() loops through each education entry and displays form fields.
  - Each entry includes Degree, Graduation Year, Institution, Field of Study, and Grade.

- **Remove Education Button**:
    - Displayed only if more than one entry exists (data.length > 1).
    - Styled with red gradient, hover transitions, and icon.
- **Add More Education Button**:
    - Positioned after all entries.
    - Uses react-icons (FiPlus).
    - Adds new blank education form with smooth UI feedback.
- **Styling**:
    - Tailwind CSS used heavily for spacing, borders, hover effects, shadows, gradients, and transitions.
    - Card-like section for each education entry with soft color gradients for visual clarity.
- **Component Export**:
- export default function Education
- Makes Education module reusable wherever academic details are needed.

Skills.js :

```
import React from "react";
import { FiTrash2, FiPlus } from "react-icons/fi";

export default function Skills({ data, setData }) {
  const handleChange = (index, value) => {
    const newSkills = [...data];
    newSkills[index] = value;
    setData(newSkills);
  };

  const addSkill = () => setData([...data, ""]);

  const removeSkill = (index) => {
    setData(data.filter((_, i) => i !== index));
  };

  return (
    <div className="bg-white rounded-2xl shadow-lg p-8 border border-gray-200
hover:shadow-xl transition-shadow">
```

```
      <h2 className="text-3xl font-bold text-gray-900 mb-8 flex items-center
gap-2">⭐ Skills</h2>
      <div className="space-y-5">
        {data.map((skill, i) => (
          <div key={i} className="flex gap-4 items-center group p-4 rounded-lg
bg-linear-to-r from-gray-50 to-indigo-50 border border-gray-200 hover:border-
indigo-300 transition-all shadow-sm hover:shadow-md">
            <input
              placeholder="e.g., JavaScript, React, Node.js"
              value={skill}
              onChange={(e) => handleChange(i, e.target.value)}
              className="flex-1 px-5 py-3 border-2 border-gray-200 rounded-lg
focus:outline-none focus:border-indigo-500 focus:ring-2 focus:ring-indigo-100
transition-all text-base bg-white"
            />
            {data.length > 1 && (
              <button
                onClick={() => removeSkill(i)}
                className="bg-red-500 hover:bg-red-600 text-white font-bold
py-3 px-4 rounded-lg flex items-center gap-2 transition-all duration-300
shadow-md hover:shadow-lg opacity-100 group-hover:opacity-100"
              >
                <FiTrash2 size={18} />
              </button>
            )}
          </div>
        ))}
      </div>
      <button
        onClick={addSkill}
        className="w-full mt-8 bg-linear-to-r from-indigo-600 to-indigo-700
hover:from-indigo-700 hover:to-indigo-800 text-white font-bold py-3 px-4
rounded-lg flex items-center justify-center gap-2 transition-all duration-300
shadow-lg hover:shadow-xl transform hover:-translate-y-1"
      >
        <FiPlus size={18} /> Add Skill
      </button>
    </div>
  );
}
```

Code Description:

- **Skills.js**: Component responsible for displaying and managing the list of user skills.
- **Props Used**:
  - `data`: Array of skill strings entered by the user.
  - `setData`: Updates the skills array whenever a skill is modified, added, or removed.

- **handleChange(index, value)**:
    - Updates a specific skill based on index.
    - Uses spread syntax to maintain immutability:
    - `const newSkills = [...data];`
    - `newSkills[index] = value;`
    - `setData(newSkills);`
- **addSkill()**:
    - Adds a new empty string to the skills array.
    - Enables dynamic addition of multiple skills.
- **removeSkill(index)**:
    - Removes selected skill using `filter`.
    - Button only shows if there is more than one skill entry.
- **Dynamic Rendering**:
    - `data.map()` loops through all skill values.
    - Creates an input field for each skill with responsive styling.
- **Input Features**:
    - Displays placeholder examples like “JavaScript, React, Node.js”.
    - Updates resume preview instantly via two-way binding.
- **Delete Button**:
    - Uses `FiTrash2` icon.
    - Only visible when more than 1 skill exists.
    - Styled with red background, hover effects, and shadow.
- **Add Skill Button**:
    - Styled with gradient colors and elevation effects.
    - Uses `FiPlus` icon.
    - Adds a new skill row on click.
- **Styling**:
    - Utilizes Tailwind CSS for:
        - Gradient backgrounds
        - Smooth hover transitions
        - Rounded edges and soft shadows
        - Focus ring and border highlights on input
- **Component Export**:

```
export default function Skills
```

Allows the Skills component to be imported and used across the project.

Experience.js :

```
import React from "react";
import { FiTrash2, FiPlus } from "react-icons/fi";

export default function Experience({ data, setData }) {
  const handleChange = (index, key, value) => {
    const newExp = [...data];
    newExp[index][key] = value;
    setData(newExp);
  };

  const addExperience = () => setData([...data, { company: "", role: "",
duration: "", description: "" }]);

  const removeExperience = (index) => {
    setData(data.filter((_, i) => i !== index));
  };

  return (
    <div className="bg-white rounded-2xl shadow-lg p-8 border border-gray-200
hover:shadow-xl transition-shadow">
      <h2 className="text-3xl font-bold text-gray-900 mb-8 flex items-center
gap-2">💼 Work Experience</h2>
      <div className="space-y-8">
        {data.map((exp, i) => (
          <div key={i} className="p-8 border-2 border-indigo-100 rounded-xl
space-y-6 bg-linear-to-br from-white to-indigo-50 hover:border-indigo-300
transition-colors shadow-md hover:shadow-lg">
            <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
              <div>
                <label className="block text-sm font-bold text-gray-700 mb-3
uppercase tracking-wide">Company</label>
                <input
                  placeholder="e.g., Tech Company Inc."
                  value={exp.company}
                  onChange={(e) => handleChange(i, "company", e.target.value)}
                  className="w-full px-5 py-3 border-2 border-gray-200
rounded-lg focus:outline-none focus:border-indigo-500 focus:ring-2 focus:ring-
indigo-100 transition-all text-base"
                />
              </div>
              <div>
                <label className="block text-sm font-bold text-gray-700 mb-3
uppercase tracking-wide">Job Title</label>
                <input
                  placeholder="e.g., Senior Developer"
                  value={exp.role}
                  onChange={(e) => handleChange(i, "role", e.target.value)}
```

```
                  className="w-full px-5 py-3 border-2 border-gray-200
rounded-lg focus:outline-none focus:border-indigo-500 focus:ring-2 focus:ring-
indigo-100 transition-all text-base"
                />
              </div>
            </div>
            <div>
              <label className="block text-sm font-bold text-gray-700 mb-3
uppercase tracking-wide">Duration</label>
              <input
                placeholder="e.g., Jan 2022 - Present"
                value={exp.duration}
                onChange={(e) => handleChange(i, "duration", e.target.value)}
                className="w-full px-5 py-3 border-2 border-gray-200 rounded-
lg focus:outline-none focus:border-indigo-500 focus:ring-2 focus:ring-indigo-
100 transition-all text-base"
              />
            </div>
            <div>
              <label className="block text-sm font-bold text-gray-700 mb-3
uppercase tracking-wide">Description</label>
              <textarea
                placeholder="Describe your responsibilities and key
achievements..."
                value={exp.description || ""}
                onChange={(e) => handleChange(i, "description",
e.target.value)}
                rows="4"
                className="w-full px-5 py-3 border-2 border-gray-200 rounded-
lg focus:outline-none focus:border-indigo-500 focus:ring-2 focus:ring-indigo-
100 transition-all text-base resize-none"
              />
            </div>
            {data.length > 1 && (
              <button
                onClick={() => removeExperience(i)}
                className="w-full bg-linear-to-r from-red-500 to-red-600
hover:from-red-600 hover:to-red-700 text-white font-bold py-3 px-4 rounded-lg
flex items-center justify-center gap-2 transition-all duration-300 shadow-md
hover:shadow-lg"
              >
                <FiTrash2 size={18} /> Remove Experience
              </button>
            )}
          </div>
        ))}
      </div>
      <button
        onClick={addExperience}
        className="w-full mt-8 bg-linear-to-r from-indigo-600 to-indigo-700
hover:from-indigo-700 hover:to-indigo-800 text-white font-bold py-3 px-4
rounded-lg flex items-center justify-center gap-2 transition-all duration-300
shadow-lg hover:shadow-xl transform hover:-translate-y-1"
```

```
      >
        <FiPlus size={18} /> Add More Experience
      </button>
    </div>
  );
}
```

Code Description:

- **Experience.js**: Component responsible for capturing and managing multiple work experience entries.

- **Props Used**:

  - `data`: Array of experience objects (company, role, duration, description).
  - `setData`: Updates the experience list in real-time when user edits, adds, or deletes entries.

- **handleChange(index, key, value)**:

  - Updates only the targeted field of a specific experience entry.
  - Uses spread cloning to avoid mutating original state:
  - `const newExp = [...data];`
  - `newExp[index][key] = value;`
  - `setData(newExp);`

- **addExperience()**:

  - Appends a new blank work experience structure:
    `{ company: "", role: "", duration: "", description: "" }`
  - Enables users to add multiple job entries dynamically.

- **removeExperience(index)**:

  - Removes selected entry using `filter`.
  - Remove button appears only if more than one experience exists.

- **Dynamic Mapping**:

  - `data.map()` renders each experience block with professional styling.
  - Each block includes: Company, Job Title, Duration, and Description.

- **Description Field**:

  - Uses `<textarea>` to capture responsibilities and achievements.
  - Supports multi-line input; auto-updates in resume preview.

- **Delete Button**:

  - Red gradient button with `FiTrash2` icon.
  - Visible only if `data.length > 1`.
  - Smooth hover transition and shadow enhancements.

- **Add More Experience Button**:
  - Styled with indigo gradient and elevation effects.
  - Uses `FiPlus` icon.
  - Adds additional job entry with a single click.
- **Styling**:
  - Tailwind CSS used for:
    - Rounded cards, borders, soft gradients
    - Responsive grid layout
    - Hover transitions and shadow effects
    - Spacing and typography consistency
- **User Feedback Enhancements**:
  - Smooth transitions on interaction
  - Elevated hover effects on containers and buttons
- **Component Export**:

`export default function Experience`

Allows reuse of the Experience component across the resume builder.

ResumePreview.js :

```
import React from "react";

export default function ResumePreview({
  personal,
  education,
  experience,
  skills,
  projects,
  achievements,
}) {
  return (
    <div
      id="resume-preview-area"
      style={{
        backgroundColor: "#ffffff",
        color: "#000000",
        fontFamily: "Arial, sans-serif",
        padding: "40px",
        lineHeight: "1.2",
      }}
    >
      {/* Header */}
      <div
        style={{
```

```
          textAlign: "center",
          borderBottom: "2px solid black",
          paddingBottom: "10px",
          marginBottom: "10px",
        }}
      >
        <h1
          style={{
            fontSize: "32px",
            fontWeight: "bold",
            color: "#1e1b4b",
            margin: "0 0 5px 0",
          }}
        >
          {personal.name || "Your Name"}
        </h1>
        <div
          style={{
            display: "flex",
            justifyContent: "center",
            gap: "5px",
            flexWrap: "wrap",
            fontSize: "12px",
            color: "#404040",
          }}
        >
          <span>{personal.email || "your@email.com"}</span>
          <span>•</span>
          <span>{personal.phone || "1234567890"}</span>
        </div>
      </div>

      {/* Professional Summary */}
      {personal.summary && (
        <section style={{ marginBottom: "10px" }}>
          <h3
            style={{
              fontSize: "14px",
              fontWeight: "bold",
              color: "#1e1b4b",
              borderBottom: "1.5px solid black",
              paddingBottom: "6px",
              marginBottom: "10px",
            }}
          >
            Professional Summary
          </h3>
          <p
            style={{
              color:  "#404040",
              fontSize:  "10px",
              lineHeight: "1.2",
              margin: "0",
```

```
          }}
        >
          {personal.summary}
        </p>
      </section>
    )}

    {/* Experience */}
    {experience.length > 0 && experience[0].company && (
      <section style={{ marginBottom: "10px" }}>
        <h3
          style={{
            fontSize: "14px",
            fontWeight: "bold",
            color: "#1e1b4b",
            borderBottom: "1.5px solid black",
            paddingBottom: "6px",
            marginBottom: "10px",
          }}
        >
          Work Experience
        </h3>
        <div>
          {experience.map((exp, i) => (
            <div key={i} style={{ marginBottom: "10px" }}>
              <div
                style={{
                  display: "flex",
                  justifyContent: "space-between",
                  alignItems: "flex-start",
                  marginBottom: "3px",
                }}
              >
                <div>
                  <p
                    style={{
                      fontWeight: "bold",
                      color: "#000000",
                      fontSize: "12px",
                      margin: "0 0 3px 0",
                    }}
                  >
                    {exp.role || "Position"}
                  </p>
                  <p
                    style={{
                      color: "#404040",
                      fontSize: "10px",
                      margin: "0",
                    }}
                  >
                    {exp.company || "Company"}
                  </p>
```

```
                </div>
                <span
                  style={{
                    color: "#404040",
                    fontSize: "10px",
                    fontStyle: "italic",
                  }}
                >
                  {exp.duration || "Duration"}
                </span>
              </div>
              {exp.description && (
                <p
                  style={{
                    color: "#404040",
                    fontSize: "10px",
                    lineHeight: "1.2",
                    marginTop: "3px",
                  }}
                >
                  {exp.description}
                </p>
              )}
            </div>
          ))}
        </div>
      </section>
    )}

    {/* Education */}
    {education.length > 0 && education[0].degree && (
      <section style={{ marginBottom: "10px" }}>
        <h3
          style={{
            fontSize: "14px",
            fontWeight: "bold",
            color: "#1e1b4b",
            borderBottom: "1.5px solid black",
            paddingBottom: "6px",
            marginBottom: "10px",
          }}
        >
          Education
        </h3>
        <div>
          {education.map((edu, i) => (
            <div key={i} style={{ marginBottom: "10px" }}>
              <div
                style={{
                  display: "flex",
                  justifyContent: "space-between",
                  alignItems: "flex-start",
                }}
```

```
            >
              <div>
                <p
                  style={{
                    fontWeight: "bold",
                    color: "#000000",
                    fontSize: "12px",
                    margin: "0 0 3px 0",
                  }}
                >
                  {edu.degree}
                </p>
                <p
                  style={{
                    color: "#404040",
                    fontSize: "10px",
                    margin: "0",
                  }}
                >
                  {edu.institution}
                  {edu.field && ` • ${edu.field}`}
                </p>
                <p
                  style={{
                    color: "#404040",
                    fontSize: "10px",
                    margin: "0",
                  }}
                >
                  {edu.grade && `Grade: ${edu.grade}`}
                </p>
              </div>
              <span
                style={{
                  color: "#404040",
                  fontSize: "10px",
                  fontStyle: "italic",
                }}
              >
                {edu.year}
              </span>
            </div>
          </div>
        ))}
      </div>
    </section>
  )}

  {/* Skills */}
  {skills.length > 0 && skills[0] && (
    <section style={{ marginBottom: "10px" }}>
      <h3
        style={{
```

```
            fontSize: "14px",
            fontWeight: "bold",
            color: "#1e1b4b",
            borderBottom: "1.5px solid black",
            paddingBottom: "6px",
            marginBottom: "10px",
          }}
        >
          Skills
        </h3>

        <div
          style={{
            display: "flex",
            flexWrap: "wrap",
            gap: "6px 20px",
            fontSize: "10px",
            color: "#444",
            paddingLeft: "10px",
          }}
        >
          {skills.filter(Boolean).map((skill, i) => (
            <span
              key={i}
              style={{
                width: "22%", // 4 per row
                minWidth: "22%",
              }}
            >
              • {skill}
            </span>
          ))}
        </div>
      </section>
    )}

    {/* Projects */}
    {projects.length > 0 && projects[0].title && (
      <section style={{ marginBottom: "10px" }}>
        <h3
          style={{
            fontSize: "14px",
            fontWeight: "bold",
            color: "#1e1b4b",
            borderBottom: "1.5px solid black",
            paddingBottom: "6px",
            marginBottom: "10px",
          }}
        >
          Projects
        </h3>

        {projects.map((proj, i) => (
```

```
          <div key={i} style={{ marginBottom: "10px" }}>
            <p
              style={{
                fontWeight: "bold",
                fontSize: "12px",
                margin: 0,
                color: "#000",
              }}
            >
              {proj.title}
            </p>

            {proj.description && (
              <p
                style={{
                  fontSize:  "10px",
                  color:  "#404040",
                  marginTop:  "3px",
                  lineHeight: "1.2",
                }}
              >
                {proj.description}
              </p>
            )}
          </div>
        ))}
      </section>
    )}

    {/* Achievements */}
    {achievements.length > 0 && achievements[0] && (
      <section style={{ marginBottom: "10px" }}>
        <h3
          style={{
            fontSize: "14px",
            fontWeight: "bold",
            color: "#1e1b4b",
            borderBottom: "1.5px solid black",
            paddingBottom: "6px",
            marginBottom: "10px",
          }}
        >
          Achievements
        </h3>

        <ul
          style={{
            margin: 0,
            paddingLeft: "20px",
            listStyleType: "disc",
          }}
        >
          {achievements.map((item, i) => (
```

```
              <li
                key={i}
                style={{
                  fontSize: "10px",
                  color: "#404040",
                  marginBottom: "4px",
                  lineHeight: "1.2",
                }}
              >
                {item}
              </li>
            ))}
          </ul>
        </section>
      )}
    </div>
  );
}
```

Code Description:

- **ResumePreview.js**: Component responsible for displaying the formatted resume layout in real-time.

- **Props Received**:

  - `personal`: Contains name, email, phone, and summary details.
  - `education`: List of academic entries with degree, institution, field, year, grade.
  - `experience`: Work history entries with company, role, duration, description.
  - `skills`: List of skills provided by the user.
  - `projects`: Array of project title and description.
  - `achievements`: Array of achievements strings.

- **`resume-preview-area` (Main Container)**:

  - Inline styling applied for print-ready clean layout:
    - White background, black text, Arial font, padding, and readable line spacing.

- **Header Section**:

  - Displays user name in large bold font.
  - Shows email and phone under name with small typography.
  - Includes bottom border separation for visual clarity.
  - Default placeholders shown if no data provided.

- **Professional Summary Rendering**:

  - Conditional: displayed only if summary text exists.
  - Uses small font and spacing for concise visual output.

- **Experience Section**:
    - Rendered only if at least one experience entry exists and contains company data.
    - Each entry includes:
        - Role (bold)
        - Company (normal text)
        - Duration (italic, right aligned)
        - Optional description block.
- **Education Section**:
    - Displayed only when degree exists in the first array entry.
    - Each academic entry shows:
        - Degree (bold)
        - Institution and field (normal text)
        - Grade (if available)
        - Year aligned right.
- **Skills Section**:
    - Rendered only if skill list contains at least one non-empty value.
    - Skills shown as bullet-style entries arranged in multi-column flex layout.
- **Projects Section**:
    - Displayed only if project title exists.
    - Each project lists:
        - Bold title
        - Optional description in smaller text.
- **Achievements Section**:
    - Rendered only if achievement list contains non-null items.
    - Achievements shown as bullet list with smaller typography.
- **Conditional Rendering**:
    - Ensures empty resume sections do not appear.
    - Produces clean, professional output without blank headings.
- **Styling**:
    - Inline CSS used to maintain consistent formatting especially during PDF export.
    - Professional resume typography: bold headings, fine separators, right-aligned dates.
- **Component Export**:

```
export default function ResumePreview
```

Makes this preview accessible for download/print integration.

DownloadButton.js :

```jsx
import React, { useState } from "react";
import { FiDownload } from "react-icons/fi";
import { toast } from "react-toastify";
import html2canvas from "html2canvas";
import jsPDF from "jspdf";

export default function DownloadButton({ fileName = "resume" }) {
  const [isDownloading, setIsDownloading] = useState(false);

  const handleDownloadPDF = async () => {
    const element = document.getElementById("resume-preview-area");

    if (!element) {
      toast.error("❌ Resume preview not found!");
      return;
    }

    try {
      setIsDownloading(true);
      toast.info("Generating PDF... Please wait");

      // Convert HTML to Canvas
      const canvas = await html2canvas(element, {
        scale: 2,
        useCORS: true,
        backgroundColor: "#ffffff",
        logging: false,
      });

      const imgData = canvas.toDataURL("image/png");
      const pdfWidth = 210; // A4 width in mm
      const pdfHeight = (canvas.height * pdfWidth) / canvas.width;

      const pdf = new jsPDF({
        orientation: "portrait",
        unit: "mm",
        format: "a4",
      });

      const pageHeight = 297; // A4 height in mm
      let heightLeft = pdfHeight;
      let position = 0;

      // First page
      pdf.addImage(imgData, "PNG", 0, 0, pdfWidth, pdfHeight);
      heightLeft -= pageHeight;

      // Additional pages if needed
      while (heightLeft > 0) {
        position = heightLeft - pdfHeight;
        pdf.addPage();
        pdf.addImage(imgData, "PNG", 0, position, pdfWidth, pdfHeight);
        heightLeft -= pageHeight;
```

```
      }

      // Clean file name
      const cleanFileName =
        fileName
          .trim()
          .replace(/[^a-z0-9\s]/gi, "")
          .replace(/\s+/g, "_")
          .toLowerCase() || "resume";

      pdf.save(`${cleanFileName}.pdf`);

      toast.success("☑ Resume downloaded successfully!");
    } catch (error) {
      console.error(error);
      toast.error("❌ Failed to download PDF.");
    } finally {
      setIsDownloading(false);
    }
  };

  return (
    <button
      onClick={handleDownloadPDF}
      disabled={isDownloading}
      className="w-full bg-green-500 hover:bg-green-600 disabled:bg-gray-400
text-white font-bold py-4 px-6 rounded-lg flex items-center justify-center
gap-2 transition-all duration-300 text-lg shadow-lg hover:shadow-xl cursor-
pointer"
    >
      <FiDownload size={20} />
      {isDownloading ? "Generating..." : "Download PDF"}
    </button>
  );
}
```

Code Description:

- **DownloadButton.js**: Component responsible for exporting the formatted resume as a downloadable PDF.

- **Props Used**:
  - `fileName`: Optional prop to customize the downloaded PDF name (default: `"resume"`).
- **Local State**:
  - `isDownloading`: Boolean state managed with `useState` to disable button and show loading text during PDF generation.

- **handleDownloadPDF()**:
    - Main function triggered on click that converts the resume preview into a multi-page PDF.
    - Retrieves resume content using:
    - `document.getElementById("resume-preview-area");`
    - Displays error toast if preview is missing.
- **User Notifications**:
    - Uses `react-toastify` to show status alerts:
        - `"Generating PDF..."` info toast
        - `"Resume downloaded successfully!"` success toast
        - Error notification if PDF creation fails
- **html2canvas Integration**:
    - Converts the HTML resume preview into high-quality canvas image.
    - Options include:
        - `scale: 2` for HD clarity
        - `useCORS: true` for embedded assets
        - `backgroundColor: "#ffffff"` for clean printing
- **jsPDF Integration**:
    - Creates A4-size PDF document.
    - Adds the canvas image to PDF using:
    - `pdf.addImage(imgData, "PNG", 0, 0, pdfWidth, pdfHeight);`
    - Handles multi-page splitting if resume exceeds one page height.
- **File Naming**:
    - Cleans user-entered file name by:
        - Removing symbols, replacing spaces with underscores, converting to lowercase.
    - Ensures valid file output: `resume.pdf` or custom cleaned name.
- **Button Rendering**:
    - Styled using Tailwind classes with green theme, rounded corners, shadow, and transitions.
    - Button text changes dynamically:
        - `"Download PDF"` → `"Generating..."` when downloading
- **Disabled State**:
    - Button disabled during PDF generation to prevent duplicate downloads.
- **Component Export**:

```
export default function DownloadButton
```

Makes this component reusable across app modules.

Implement Layout & Styling:

- UI built using **Material UI** grid system & cards.
- Professional typography applied for readability.
- Side-by-side layout:
    - Left side: form inputs
    - Right side: live resume preview panel

Add Navigation:

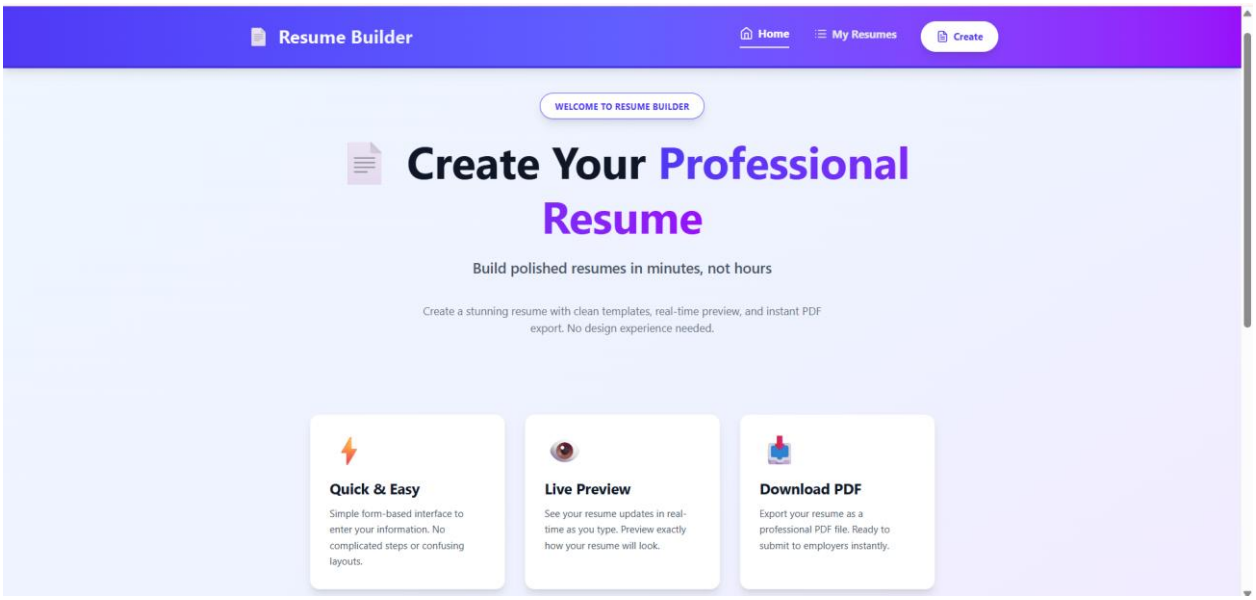
Navigation is minimal: primary view is divided into:

- Input Form Panel
- Resume Output Panel
- Download Button

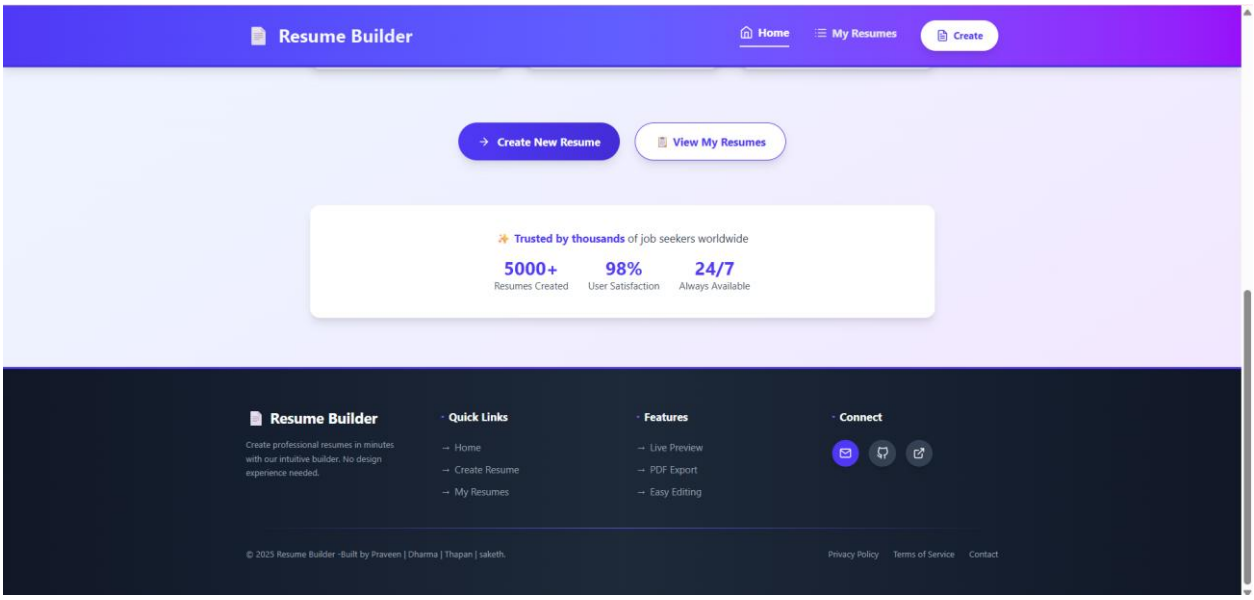
No routing pages required — all actions happen within a single view.

OUTPUTS:

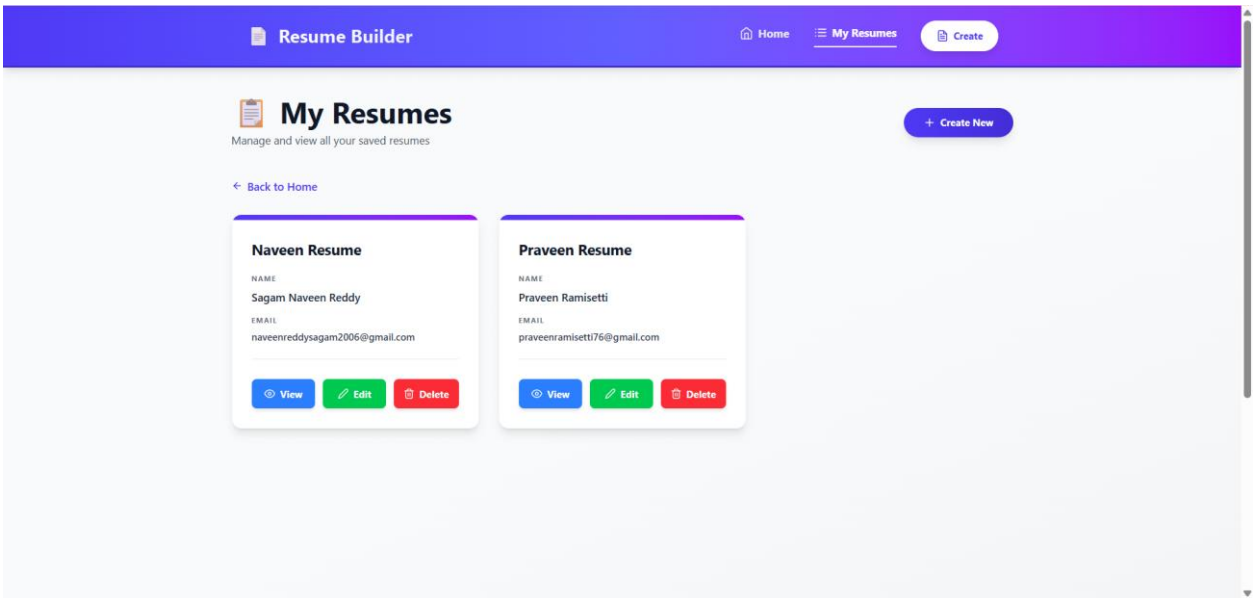
Home page:



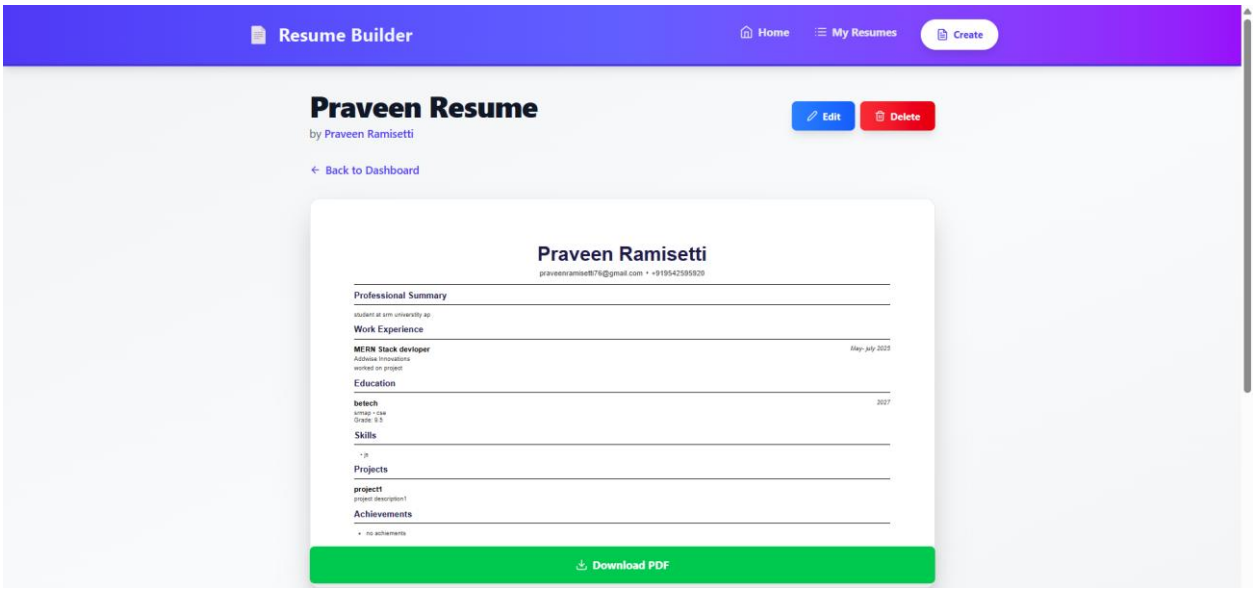
footer:



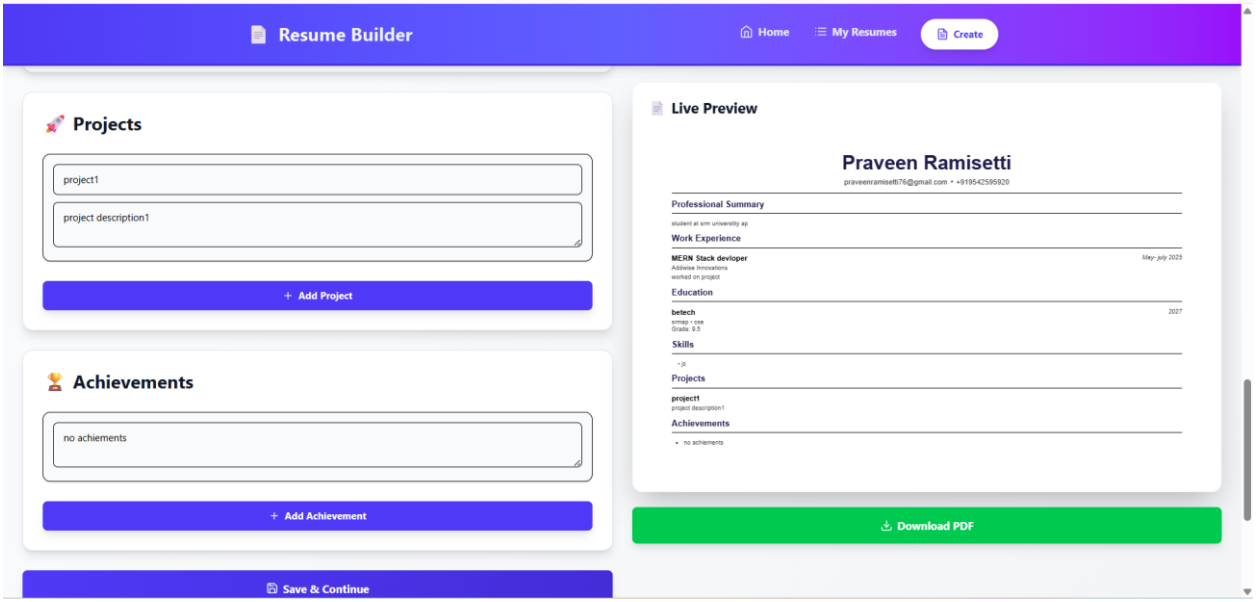
My resumes page :



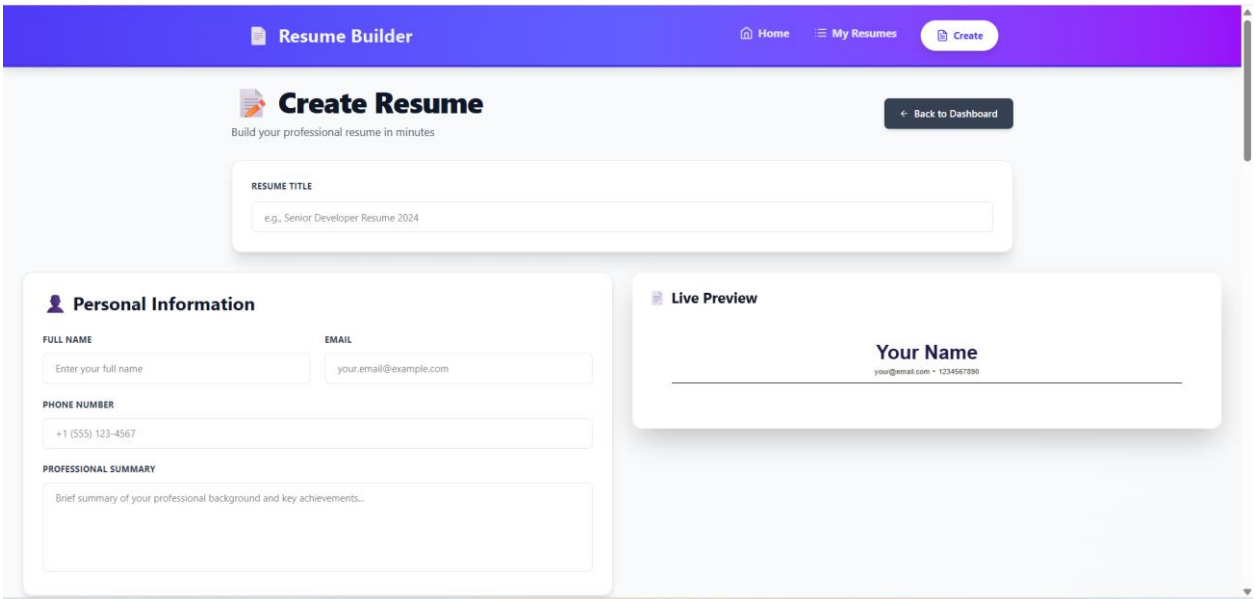
View resume option:

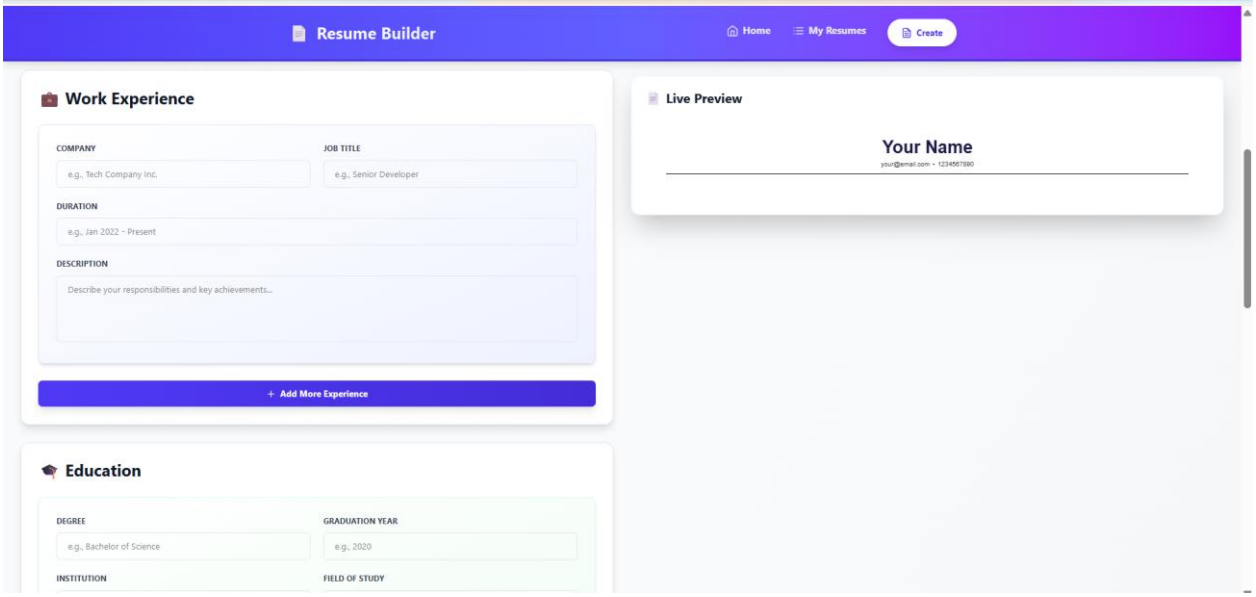


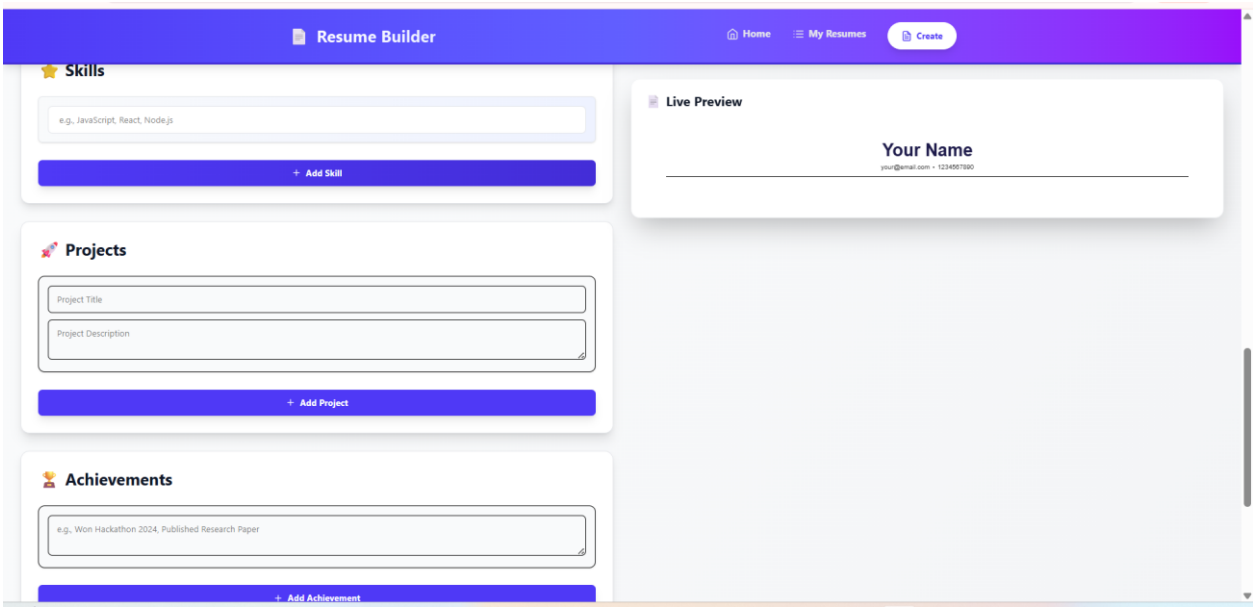
Editing the resume :



Creating the resume :







Project Demo Link:
https://drive.google.com/file/d/1UJBWwk0uH8r2iGkSng5Xn7uJQ6ZhHUrM/view?usp=sharing

Project Code Explanation Video Link:
https://drive.google.com/file/d/1WBBTfBsIVZL1B8dS9MzCx7xD-Fuw-mdF/view?usp=sharing