Project Report: AI-Powered Resume Builder

Author: Rachana Dutta **Date:** September 8, 2025

1. Introduction

In today's competitive job market, a well-crafted, professional resume is the most critical tool for a job seeker. However, the process of creating, formatting, and tailoring a resume for different roles can be a tedious and challenging task. Many individuals struggle with organizing their information effectively and presenting it in a visually appealing format.

The AI-Powered Resume Builder is a web-based application designed to solve this problem by streamlining the entire resume creation process. It provides users with an intuitive, step-by-step interface to input their professional details, select from modern templates, and instantly generate a high-quality PDF. The project aims to empower users to build compelling resumes with minimal effort, saving them valuable time and helping them make a strong first impression on potential employers.

2. Abstract

This report outlines the development of the AI-Powered Resume Builder, a full-stack web application. The primary objective of this project was to create a user-friendly platform for building and downloading professional resumes. The application utilizes a modern technology stack, with a React.js frontend for a dynamic user experience and a Node.js/Express.js backend to handle data persistence, user authentication, and server-side operations.

Key features include a multi-step form for guided data entry, auto-saving of user progress, secure user authentication using JSON Web Tokens (JWT), and a powerful PDF generation engine using Puppeteer. Users can choose from multiple design templates, preview their resume in real-time, and download the final document in PDF format. The architecture is designed to be scalable, allowing for the future integration of AI-powered content suggestions to further assist users in crafting their resume content. The result is a robust, efficient, and practical tool for any job seeker.

3. Tools and Technologies Used

The project was developed using a combination of modern web technologies to ensure a robust and scalable application.

- **Frontend:**
    - **React.js:** A JavaScript library for building dynamic and responsive user interfaces.
    - **Tailwind CSS:** A utility-first CSS framework for rapid UI development and styling.
    - **Axios:** A promise-based HTTP client for making API requests from the browser to the backend.
- **Backend:**
    - **Node.js:** A JavaScript runtime environment for executing server-side code.
    - **Express.js:** A minimal and flexible Node.js web application framework used to build the REST API.
- **Database:**
    - **MongoDB:** A NoSQL database used to store user data and resume information, accessed via the Mongoose ODM.
- **PDF Generation:**
    - **Puppeteer:** A Node.js library that provides a high-level API to control a headless Chromium instance, used for generating PDFs from web page templates.
- **Authentication:**
    - **JSON Web Tokens (JWT):** Used for securely authenticating users and protecting API routes.
- **Deployment:**
    - **Render:** A cloud platform used for deploying the backend service.
    - **GitHub Pages:** Used for hosting the static HTML print templates.

4. Steps Involved in Building the Project

The development process was broken down into several logical phases:

1. **Project Scaffolding and Planning:** The initial phase involved setting up the project structure for both the frontend (using Create React App) and the backend. The application's features were defined, the user flow was mapped out, and the database schema for storing resume data was designed in MongoDB.
2. **Backend API Development:** The core of the backend was built using Node.js and Express. This involved creating REST API endpoints for:
    - User authentication (login/registration, though not shown in the snippet).
    - Saving resume progress (/resume/save) to the MongoDB database.
    - Loading existing resume data (/resume/load) for a logged-in user.
    - Handling the PDF download request (/resume/download).
3. **Frontend Component Creation:** The user interface was built with React. Reusable components were created for each section of the resume-building process, such as PersonalInfo, Experience, Skills, Education, and Projects. A main parent component (Resume.jsx) was used to manage the application's overall state, including the form data and the current step in the process.
4. **State Management and API Integration:** React's useState and useEffect hooks were used extensively to manage form data and component lifecycle. An auto-save feature was implemented using a setTimeout within a useEffect hook to create a debouncing effect, which saves user progress one second after they stop typing. Axios was integrated to facilitate communication between the frontend and backend for loading and saving data.
5. **PDF Generation and Download Implementation:** This was a critical feature. The process was designed as follows:
    - The frontend sends the complete formData and the selected template ID to the /resume/download endpoint.
    - The backend launches a headless instance of Chromium using Puppeteer.
    - It navigates to a static HTML template hosted on GitHub Pages, passing the formData as a URL-encoded JSON string in the query parameters.
    - The template page uses JavaScript to parse the data from the URL and dynamically render the resume.
    - Once the page is fully rendered (networkidle0), Puppeteer generates a PDF of the page.
    - The backend server sends this generated PDF back to the frontend as a blob, which is then triggered for download in the user's browser.
6. **Deployment and Finalization:** The backend server was deployed to Render, with environment variables configured for the database connection string and JWT secret. The static print templates were pushed to a GitHub repository and deployed using GitHub Pages. Final testing was conducted to ensure all parts of the application worked together seamlessly.

5. Conclusion

The AI-Powered Resume Builder project successfully achieved its goal of creating a functional and user-friendly platform for resume creation. It effectively combines a modern frontend, a robust backend, and a powerful document generation engine to provide a seamless user experience. The auto-save functionality and real-time preview significantly enhance usability.

The architecture is modular and allows for future expansion. Potential future work includes integrating a generative AI model (like a GPT API) to provide users with content suggestions, adding a wider variety of templates, and implementing features for sharing resumes via a public link. Overall, the project stands as a strong proof-of-concept and a valuable tool for job seekers.