

“ResumeAI”

Major Project Report

*Submitted in partial fulfilment of the requirement for the award of
Degree of*

BACHELOR OF COMPUTER APPLICATION

By Arsdeep Dewangan
(KID : K23334)

Under the Supervision of
Prof. Arshad Hussain



SCHOOL OF COMPUTER APPLICATION
CAREER POINT UNIVERSITY, KOTA

May, 2025

ACKNOWLEDGEMENT

I am deeply grateful to all the individuals and institutions whose invaluable guidance, encouragement, and resources have supported me throughout this project.

Firstly, I extend my heartfelt thanks to the teachers at Career Point University, Kota, for their unwavering support and mentorship. Their expertise and encouragement have been instrumental in enhancing my understanding and skills, which were crucial to the successful completion of this project.

I would also like to express my sincere appreciation to Career Point University, Kota, for providing a stimulating academic environment and access to the necessary resources. The knowledge and experiences gained during my time at the university have significantly contributed to the development of this project.

Lastly, I would like to acknowledge the collective contributions Career Point University for shaping my learning journey and enabling me to successfully complete this project. Their support has been integral to achieving the objectives of this work.

Thank you all for your invaluable contributions.

Arsdeep Dewangan

Place: Kota

Date: 04/12/24

DECLARATION

We hereby declare that this Project Report titled InsightFlow submitted by me and approved by my project guide, the School of Computer Application and Technology (SOCA), Career Point University, Kota is a bonafide work undertaken by me and it is not submitted to any other University or Institution for the award of any degree diploma / certificate or published any time before.


Project Name:	InsightFlow	
Student Name:	Arsdeep Dewangan	Signature: 
Project Guide:	Arshad Hussain	Signature:

Table of Contents

Contents

1	Introduction.....	6
2	Problem Statement	7
3	Project Description.....	8
3.1	Scope of the Work.....	8
3.2	Project Modules	10
3.3	Context Diagram (High Level).....	12
4	Implementation Methodology.....	13
5	Technologies to be used	17
6	Advantages of this Project.....	22
7	Assumptions	24
8	Future Scope and further enhancement of the Project	26
9	Project Repository Location.....	28
10	Definitions, Acronyms, and Abbreviations.....	28
11	Conclusion	29
12	References.....	30

1 Introduction

In today's competitive professional landscape, a well-crafted resume serves as a critical gateway to career opportunities. However, many individuals—especially students and early-career professionals—struggle with resume formatting, content structuring, and effectively presenting their skills. Recognizing these challenges, **ResumeAI** was conceptualized as a modern solution to streamline the resume creation process, making it accessible, intelligent, and user-friendly.

ResumeAI is a full-stack web application designed to empower users to dynamically create, customize, and export professional-grade resumes with ease. It bridges the gap between design and functionality by offering an intuitive, drag-and-drop interface along with live preview capabilities. The application goes beyond static resume templates by integrating **AI-powered suggestions**, real-time customization, and persistent user data via cloud-based storage.

At the core of **ResumeAI** lies a powerful technology stack that ensures both performance and flexibility. Built with **Next.js (v15.2.2)** and **React (v19)**, the application supports both Server-Side Rendering (SSR) and Static Site Generation (SSG), providing optimal SEO benefits and fast load times. **TypeScript** enhances the developer experience by introducing type safety, while **Tailwind CSS**—supplemented with **tailwindcss-animate**—offers a utility-first approach to responsive, modern design.

The user experience is enriched with **Radix UI** components, **lucide-react** icons, and **next-themes** for seamless dark/light mode toggling. Global and local states are efficiently managed using **zustand** and **react-hook-form**, complemented by **zod** validation schemas for form reliability. Additionally, the drag-and-drop customization feature is implemented using the **@dnd-kit** suite, enabling users to reorder sections of their resumes effortlessly.

Authentication and user data management are securely handled via **Clerk**, offering a smooth onboarding experience. Users can save, manage, and revisit multiple resume drafts through personalized dashboards. They can also export their resumes in print-ready PDF formats using **react-to-print**, with advanced customization features like **color pickers (react-color)** and layout selectors to suit individual preferences.

One of the standout features of **ResumeAI** is its integration with the **OpenAI API**, which assists users by auto-generating content for common resume sections such as job descriptions, summaries, and skills, significantly reducing the cognitive load on the user. The backend leverages **Prisma ORM** for efficient and type-safe database interactions, while environmental and blob data management are handled using **@t3-oss/env-nextjs** and **@vercel/blob**, respectively.

In conclusion, **ResumeAI** is not just a tool—it's a platform that redefines how users approach resume creation. Through a blend of intelligent automation, thoughtful UI/UX design, and a modern tech stack, ResumeAI offers an innovative, scalable solution to one of the most persistent challenges in the professional world.

2 Problem Statement

In today's competitive job market, having a well-structured and visually appealing resume is critical to making a strong first impression with potential employers. However, the process of creating such a resume can be tedious and inaccessible for many, especially those without design experience or technical skills.

Traditional resume builders are often either too rigid, limiting the user's ability to customize layout and design, or too complex, requiring users to have knowledge of design tools or templates.

Moreover, most online resume builders lack real-time feedback, customization options, and the flexibility to dynamically rearrange sections based on individual needs. Features like drag-and-drop section reordering, AI-assisted content generation, and dark/light themes are rarely integrated in a cohesive, user-friendly platform. As a result, users often struggle to create resumes that effectively highlight their strengths and cater to specific job roles.

From a developer's perspective, building such a platform also introduces several technical challenges:

- Implementing a dynamic and responsive UI that remains consistent across different devices.
- Ensuring real-time previews and customization without degrading performance.
- Providing a seamless drag-and-drop interface for managing resume layout.
- Incorporating secure user authentication and data management.
- Enabling export to professional-quality PDF formats.
- Supporting scalable backend architecture for managing user data and file storage.

To address these limitations and technical demands, **ResumeAI** is proposed — a modern, full-stack web application that empowers users to create, customize, and manage professional resumes with ease.

Leveraging cutting-edge web technologies such as **Next.js**, **React**, **Tailwind CSS**, and **OpenAI API**, ResumeAI aims to bridge the gap between usability, performance, and professional design. The platform offers a live editing experience, intuitive drag-and-drop interface, AI-powered content generation, and theming capabilities — all while ensuring a secure and scalable infrastructure through services like **Clerk**, **Prisma**, and **Vercel Blob Storage**.

The goal is to make resume building accessible, efficient, and even enjoyable — helping users present themselves in the best possible light with minimal friction.

3 Project Description

3.1 Scope of the Work

In-Scope:

1. User Interface & Experience

- Design and implementation of a modern, responsive UI using **Tailwind CSS**.
- Dark/light theme support via **next-themes**.
- Interactive UI components using **Radix UI** and **lucide-react** icons.
- Smooth animations and visual transitions using **tailwindcss-animate**.

2. Resume Builder Core Features

- Real-time resume editing and live preview.
- Section-based editing with support for common resume fields (Education, Experience, Skills, etc.).
- **Drag-and-drop** functionality for reordering resume sections using **@dnd-kit/core**.
- Customization of color themes using **react-color**.
- Resume export to PDF/printable formats using **react-to-print**.

3. Authentication and User Management

- Secure user authentication and session management using **Clerk** (**@clerk/nextjs**, **@clerk/themes**).
- Support for user-specific dashboards to manage multiple resumes.

4. Form & State Management

- Efficient form handling using **react-hook-form** with schema validation via **zod** and **@hookform/resolvers**.
- Global state management across components with **zustand**.

5. AI-Powered Enhancements

- Integration with **OpenAI API** to auto-generate resume content such as summaries, achievements, and job descriptions.

6. Backend and Database

- Backend logic and database interaction using **Next.js API routes** and **Prisma ORM**.
- Environment variable management via **@t3-oss/env-nextjs**.
- Storage of resume templates, images, or assets using **@vercel/blob**.

7. Deployment and Hosting

- Deployment of the application using **Vercel** for optimal performance and scalability.
- Utilization of Next.js capabilities like **Server-Side Rendering (SSR)** and **Static Site Generation (SSG)** to improve SEO and load speed.

Out of Scope:

1. Mobile App Development

ResumeAI is built as a responsive web application. Native mobile apps for iOS and Android are not part of this project's scope.

2. Third-Party Resume Parsing/Import

The ability to import existing resumes from LinkedIn, PDFs, or other formats using parsing technology (e.g., via OCR or NLP) is not included.

3. Job Application Integration

Features for applying to jobs directly through the platform, integration with job boards (like Indeed or LinkedIn), or job recommendation systems are not considered in this version.

4. Multi-Language Support

The current implementation supports only English. Internationalization (i18n) and multi-language resume creation are out of scope.

5. Advanced AI Personalization

Although OpenAI is used for generating content suggestions, deeper AI-driven personalization like role-specific optimization, tone analysis, or real-time AI feedback is beyond the scope.

6. Collaboration & Team Features

Real-time collaboration features (e.g., Google Docs-style co-editing) or sharing resumes with teams or mentors for comments were not implemented.

7. Offline Functionality

ResumeAI is designed as an online application and does not support offline editing or data storage in this version.

8. Analytics and Resume Tracking

The platform does not provide analytics on resume performance (e.g., tracking views, clicks, or download stats), which would require external integrations and data pipelines.

3.2 Project Modules

1. User Authentication Module

- **Purpose:** Manages user sign-up, sign-in, and session handling.
- **Technologies:** @clerk/nextjs, @clerk/themes
- **Features:**
 - Secure user registration and login
 - OAuth and social login support
 - Themed authentication UI

2. Resume Builder Module

- **Purpose:** Core interface for building and customizing resumes.
- **Technologies:** React, Next.js, zustand, react-hook-form, zod, @dnd-kit
- **Features:**
 - Dynamic resume sections (experience, education, etc.)
 - Drag-and-drop section reordering
 - Form-based content entry with validation
 - Live preview while editing

3. AI Assistant Module

- **Purpose:** Helps users auto-generate resume content using AI.
- **Technologies:** openai
- **Features:**
 - Auto-generated job summaries or bullet points
 - Custom prompts for content suggestions

4. Theming & Customization Module

- **Purpose:** Allows users to change colors, themes, and layouts.
- **Technologies:** next-themes, react-color, Tailwind CSS
- **Features:**
 - Light/dark mode
 - Custom color palettes
 - Selectable layout styles and fonts

5. Export & Print Module

- **Purpose:** Enables users to download or print their resumes.
- **Technologies:** react-to-print
- **Features:**
 - Export to PDF
 - Printer-friendly formatting

6. User Dashboard Module

- **Purpose:** Lets users manage multiple resumes and templates.
- **Technologies:** Prisma, Next.js, @vercel/blob (for storage)
- **Features:**
 - Create, edit, delete, and duplicate resumes
 - Save progress in the cloud
 - File uploads (e.g., profile photos or templates)

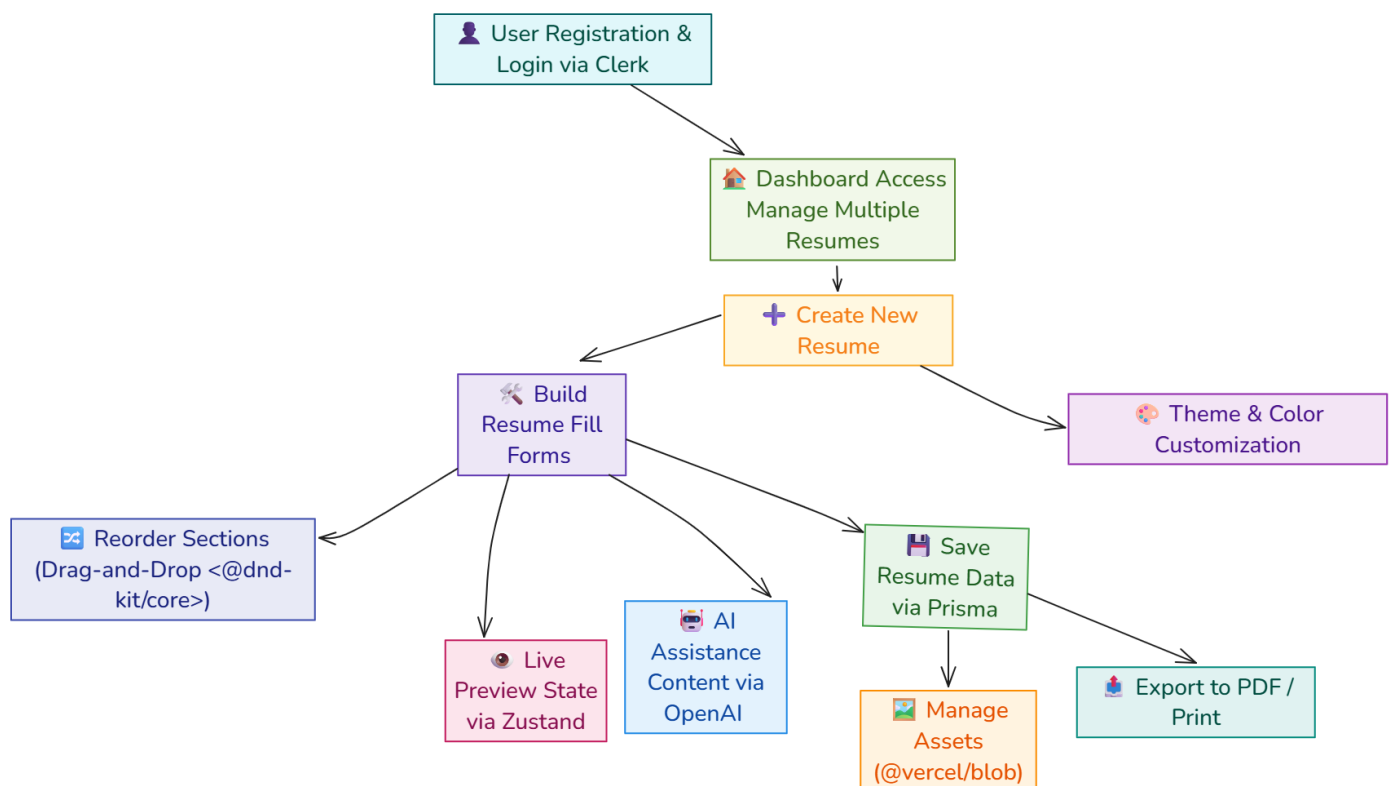
7. Backend & Database Module

- **Purpose:** Handles data persistence, queries, and server-side logic.
- **Technologies:** Prisma, @t3-oss/env-nextjs
- **Features:**
 - Type-safe database interactions
 - Secure environment variable handling

8. UI/UX & Component Module

- **Purpose:** Provides reusable UI components and design consistency.
- **Technologies:** Radix UI, lucide-react, clsx, class-variance-authority
- **Features:**
 - Dialogs, popovers, dropdowns
 - Iconography and accessibility
 - Consistent styling with utility classes

3.3 Context Diagram (High Level)



4 Implementation Methodology

The implementation of **ResumeAI** followed a modular, agile, and component-driven approach to deliver a scalable, responsive, and user-friendly resume-building web application. This methodology is structured in phases, each targeting a core functional or technical aspect of the application. The development process was guided by best practices in modern web development, prioritizing performance, maintainability, and user experience.

1. Requirement Analysis and Planning

In the initial phase, the project's scope was clearly defined:

- Enable users to create, customize, and export resumes.
- Provide drag-and-drop section management.
- Support user authentication and multiple resume profiles.
- Integrate AI-based suggestions for resume content.

This phase involved gathering feature requirements, identifying the appropriate technologies, and laying down a clear roadmap.

2. Tech Stack Setup

The development environment was configured using the following core technologies:

- **Next.js 15.2.2:** For server-side rendering (SSR), static site generation (SSG), routing, and overall app structure.
- **TypeScript:** To enforce static typing, catch bugs early, and improve code maintainability.
- **Tailwind CSS:** Used with tailwindcss-animate for utility-first, responsive styling and smooth animations.

Version control was managed using Git, and the project was hosted on GitHub with CI/CD integrations through Vercel for automatic deployment.

3. UI & Component Development

The interface was developed using **React 19**, following a component-driven architecture:

- **Radix UI** components such as Dialog, Popover, and Dropdown were used for accessibility and consistent UI behavior.
- **lucide-react** icons provided a modern and clean iconography.
- **next-themes** allowed seamless toggling between light and dark modes.

CSS classes were dynamically handled using `clsx` and `class-variance-authority` to manage variants, especially across reusable components like buttons, form inputs, and layout containers.

4. State and Form Management

To handle complex user interactions and dynamic data updates:

- **zustand** was employed for global state management (e.g., resume structure, color theme, selected template).
 - **react-hook-form** managed local form states efficiently.
 - **zod** combined with `@hookform/resolvers` enforced schema validation for all form fields, improving data integrity and form usability.
-

5. Drag-and-Drop Resume Sections

An intuitive drag-and-drop builder was crucial to the application:

- Built using `@dnd-kit/core`, `sortable`, and `modifiers`.
- Each resume section (Education, Experience, Skills, etc.) was treated as a draggable block.
- Users could reorder these blocks with real-time visual feedback.

State persistence ensured that changes were saved to the database using `Prisma` and reflected in the UI immediately.

6. Authentication and User Dashboard

User accounts and session management were implemented with:

- **@clerk/nextjs** and **@clerk/themes** for sign-in, sign-up, and theme-aware auth components.
- Authenticated users could access their dashboards, manage multiple resumes, and retrieve past work.

Session data was protected and securely integrated with Next.js API routes using Clerk middleware.

7. Resume Editing and Customization

Users could modify content, layout, and appearance:

- Resume sections supported rich form input and markdown-like formatting.
 - A real-time preview pane showed immediate visual feedback.
 - Users could select colors using **react-color** pickers.
 - The layout could be dynamically customized using theme-aware Tailwind classes.
-

8. AI Integration

OpenAI API was integrated to:

- Suggest job experience bullet points.
- Auto-generate profile summaries and skills based on user input.
- Enhance user productivity and reduce writer's block.

This functionality was invoked using server-side API routes for secure token handling and improved performance.

9. Export and Printing Functionality

Resumes could be exported to PDF using:

- **react-to-print**, which rendered the visual layout into a printable format.
- Custom styles ensured that exported resumes maintained a clean professional look, both in print and digital formats.

10. Backend, Database & Environment Setup

- **Prisma ORM** was used for structured access to the underlying database, enabling CRUD operations for resumes, templates, and user metadata.
 - **@t3-oss/env-nextjs** helped manage environment variables securely across development and production environments.
 - **@vercel/blob** allowed file uploads and storage, paving the way for future template/image features.
-

11. Deployment and Hosting

- The application was deployed on **Vercel**, which provided optimized hosting for Next.js applications with automatic builds and previews.
 - GitHub actions ensured continuous deployment pipelines, enabling rapid development iterations.
-

12. Testing and Optimization

- Manual and automated testing ensured cross-browser compatibility and usability.
 - Lighthouse and Web Vitals metrics were used to audit performance and SEO.
 - Tailwind's optimize features helped purge unused styles to reduce bundle size.
-

Conclusion

The implementation methodology for **ResumeAI** followed a structured yet agile path. Emphasis was placed on modern best practices, modular development, responsive UI, and performance optimization. The result is a dynamic, user-friendly, and production-ready resume builder powered by the latest web technologies.

5 Technologies to be used

5.1 Software Platform

a). Frontend Technologies

- Next.js (v15.2.2)

- Purpose: Core framework for frontend development.
- Why: Combines Server-Side Rendering (SSR) and Static Site Generation (SSG) for optimal performance, SEO, and flexibility in rendering strategies.
- Features leveraged: Routing, API routes, and image optimization.

- React (v19)

- Purpose: Component-based UI rendering library.
- Why: Offers a declarative and modular approach to building user interfaces, well-suited for dynamic applications like a resume builder.

- TypeScript

- Purpose: Superset of JavaScript with static typing.
- Why: Improves developer experience by catching type-related bugs at compile time, ensuring maintainable and scalable code.

- Tailwind CSS + tailwindcss-animate

- Purpose: Utility-first CSS framework for styling.
- Why: Provides rapid UI development with consistent design, responsive styling, and animation support via tailwindcss-animate.

- Radix UI

- Purpose: Headless UI primitives.
- Why: Enables accessibility-focused, customizable components such as modals, tooltips, and dropdowns, without enforcing design constraints.

- lucide-react

- Purpose: Icon library.
- Why: Offers a consistent and lightweight icon set integrated with React.

- next-themes

- Purpose: Theme switching (dark/light mode).
- Why: Enhances user experience with support for customizable appearance.

- clsx & class-variance-authority

- Purpose: Conditional and composable class management.
- Why: Simplifies dynamic styling in React components.

b). State & Form Management

- zustand

- Purpose: Lightweight global state management.
- Why: Simplifies state sharing across components without the complexity of Redux.

- react-hook-form

- Purpose: Handles form state and validation.
- Why: Lightweight and performant, with minimal re-renders.

- @hookform/resolvers & zod

- Purpose: Schema-based validation.
- Why: Provides a declarative way to define and validate form inputs, improving robustness and maintainability.

c.) Drag & Drop Functionality

- @dnd-kit/core, sortable, modifiers

- Purpose: Drag-and-drop interaction for resume sections.
- Why: Enables intuitive customization of resume layout by allowing users to rearrange sections dynamically.

d.) Authentication & User Management

- @clerk/nextjs & @clerk/themes

- Purpose: Authentication, user sessions, and account management.
- Why: Provides secure, out-of-the-box authentication integrated seamlessly with Next.js, along with customizable UI themes.

e.) Resume Features & Export

- react-to-print

- Purpose: Export resumes to PDF or print.
- Why: Offers a consistent, client-side way to generate professional-quality printable resumes.

- react-color

- Purpose: Color picker tool.
- Why: Allows users to customize the color scheme of their resumes interactively.

- date-fns

- Purpose: Date formatting and utility functions.
- Why: Lightweight and modular date library, ideal for formatting job dates, experience periods, etc.

- openai

- Purpose: AI-based content generation.
- Why: Assists users in generating resume content such as summaries or experience bullet points using OpenAI's powerful language models.

f.) Backend & Infrastructure

- Prisma

- Purpose: ORM for database operations.
- Why: Enables type-safe and readable queries, ideal for managing user data, resume templates, and saved content.

- **@t3-oss/env-nextjs**

- Purpose: Environment variable management.
- Why: Ensures type-safe and secure environment configuration in a Next.js app.

- **@vercel/blob**

- Purpose: Blob storage for media or template files.
- Why: Simplifies file handling for assets like profile images or resume templates using Vercel's infrastructure.

g.) Deployment & Hosting

- **Vercel**

- Purpose: Hosting and continuous deployment.
- Why: Offers seamless integration with Next.js, instant preview environments, and edge functions for fast global performance.

5.2 Hardware Platform

Since ResumeAI is a web application primarily accessed through web browsers, the hardware requirements mainly pertain to the development environment and the typical client and server setups.

Development Environment -

- **Processor:** Modern multi-core processor (Intel i5 or AMD Ryzen 5 and above recommended) to efficiently run development servers and code compilation.
- **RAM:** Minimum 8 GB RAM recommended for smooth operation of IDEs (e.g., Visual Studio Code), local servers, and browser tabs.
- **Storage:** At least 100 GB of free disk space is recommended to accommodate project files, dependencies, database files, and build artifacts.
- **Operating System:** Compatible with Windows 10/11, macOS (Catalina or newer), or Linux distributions (Ubuntu, Fedora, etc.).

Client (End-User) Requirements -

Users will access ResumeAI via modern web browsers. The app is optimized to work smoothly on:

- Desktop or laptop computers with at least 4 GB RAM.
- Any OS supporting modern browsers (Windows, macOS, Linux).
- Modern browsers like Chrome, Firefox, Edge, or Safari.

Server Requirements (for hosting ResumeAI backend)

- RAM: Minimum 2 GB RAM for small-scale usage, scalable depending on user load.
- Storage: SSD storage of 20 GB or more, depending on data retention needs.
- OS: Linux-based server environment (Ubuntu 20.04 LTS or later is common).
- Other: Node.js runtime environment, database server (such as PostgreSQL or MySQL), and cloud infrastructure (e.g., Vercel for frontend and backend hosting).

6 Advantages of this Project

The **ResumeAI** project brings together a powerful combination of modern web technologies and practical features to solve real-world problems related to resume building. It offers significant advantages to users, developers, and academic evaluators alike. The key benefits of this project are outlined below:

1. Dynamic and Personalized Resume Creation

Unlike traditional resume builders that rely on rigid templates, ResumeAI empowers users with:

- **Drag-and-Drop Functionality** using @dnd-kit for intuitive section reordering.
- **Live Customization** of content and layout, allowing users to build resumes that truly reflect their personal branding.
- **Real-time Preview** to see changes instantly before finalizing or exporting the document.

This interactivity enhances user engagement and ensures a more tailored resume output.

2. Enhanced User Experience and Interface

The application emphasizes a smooth and modern user interface by integrating:

- **Tailwind CSS** and tailwindcss-animate for responsive, utility-first styling.
- **Radix UI** for accessible and customizable components like dialogs and dropdowns.
- **Theme Switching** through next-themes for light/dark mode support.
- **lucide-react** icons for a clean, minimal visual style.

These choices result in a visually appealing, responsive, and user-friendly application across devices and screen sizes.

3. Seamless Authentication and Personalization

With **Clerk's** authentication system (@clerk/nextjs), users can:

- Securely sign in or register.
- Save multiple resumes.
- Retrieve and edit them anytime.

This ensures data privacy and persistence across sessions, creating a personalized dashboard for every user.

4. AI-Powered Resume Assistance

A standout advantage is ResumeAI's integration with the **OpenAI API**, which allows:

- Auto-generation of content like job summaries, responsibilities, and objectives.
- Smart suggestions based on minimal input, reducing the effort for users unfamiliar with resume writing norms.

This reduces friction for non-technical or first-time users and provides a competitive edge.

5. Export and Sharing Capabilities

Using react-to-print, the application supports:

- **One-click export** to PDF format.
- **Print-ready layout** for professional use.
- Ensures consistency between on-screen and physical resumes.

This enables immediate usability for job applications without needing third-party tools.

6. Modular, Scalable Architecture

The project is built with:

- **Next.js 15.2.2** for static and server-side rendering, providing both performance and SEO benefits.
- **TypeScript** for type safety, which improves development efficiency and reduces bugs.
- **Prisma ORM** for robust database management and easy scalability.

This makes ResumeAI not only developer-friendly but also easy to maintain and extend in the future.

7. Real-World Tech Stack Exposure

From an educational standpoint, working on ResumeAI introduces students to:

- Enterprise-level technologies like React 19, Zustand, Prisma, Clerk, and OpenAI.
- Best practices in component-based architecture, state management, and schema validation (react-hook-form, zod).
- Deployment readiness with performance-optimized SSR/SSG capabilities and environment handling via @t3-oss/env-nextjs.

Such exposure provides a competitive advantage in internships and job placements.

8. Accessibility and Theming

With tools like **Radix UI** and **next-themes**, the application ensures:

- Accessibility-compliant components.
 - Custom themes for diverse user preferences.
 - Keyboard navigation and ARIA support for inclusivity.
-

9. Developer-Centric Utility

ResumeAI leverages utilities like:

- clsx and class-variance-authority for dynamic class management.
- Clean code practices and reusable components.
- Strong separation of concerns between UI, logic, and state.

These principles contribute to high code quality and easy collaboration in team projects.

7 Assumptions

The development and deployment of the **ResumeAI** web application are based on several foundational assumptions concerning the user base, technology environment, project scope, and integration capabilities. These assumptions are critical in defining the system architecture, development process, and feature implementations. They are as follows:

1. Target User Assumptions

- **Basic Technical Literacy:** Users are assumed to have basic internet and computer literacy. They are familiar with web interfaces and can interact with form-based UIs.
- **Need for Professional Resumes:** Users are actively seeking to create or improve professional resumes, likely for job applications, internships, or academic purposes.
- **Internet Access:** The application is assumed to be accessed via stable internet connections on modern browsers.

2. Device & Platform Assumptions

- **Modern Web Browsers:** Users will access the application via modern browsers (latest versions of Chrome, Firefox, Safari, Edge) that support ES6+, WebAssembly, and modern CSS features (e.g., CSS grid, flexbox).
- **Responsive Support:** While the application is designed to be responsive, it is primarily optimized for desktop and tablet viewports, assuming most users prefer larger screens for resume editing.

3. Technical Stack Assumptions

- **Support for Server-Side Rendering (SSR):** The app assumes hosting platforms support SSR and static site generation, as enabled through **Next.js** for performance and SEO benefits.
- **Persistent User Sessions:** **Clerk** is assumed to handle user authentication, session management, and profile persistence reliably.
- **Form Management Scalability:** The usage of **react-hook-form** and **zod** assumes forms will not exceed performance thresholds typically associated with large-scale, dynamic forms.
- **Client-Side Rendering for Interactivity:** While SSR is used for initial rendering, most interactivity (e.g., drag-and-drop, real-time previews) is assumed to happen on the client side using **React 19**, **zustand**, and **@dnd-kit/core**.

4. AI Integration Assumptions

- **OpenAI API Availability:** The app assumes constant and reliable access to the OpenAI API for resume content generation. Rate limits, latency, or outages are assumed to be rare or manageable.
- **User Trust in AI Suggestions:** It is assumed users will treat AI-generated resume content as a helpful draft and will review/edit it before submission to potential employers.

5. Data Storage & Infrastructure Assumptions

- **Secure Data Handling:** It is assumed that Clerk and Prisma will handle data securely and in compliance with common data protection standards (e.g., GDPR, if applicable).
- **Blob Storage Reliability:** `@vercel/blob` is assumed to provide a consistent and performant blob storage service for storing resume templates or custom user assets.
- **Environment Variables:** Secure environment configuration is assumed via `@t3-oss/env-nextjs`, with all secrets properly managed during development and deployment.

6. Usage Assumptions

- **Multiple Resume Management:** Users are assumed to benefit from managing multiple resumes from a single dashboard and will require persistent storage for these.
- **Exporting Capabilities:** Users are assumed to require print/PDF export functionality, and the system depends on **react-to-print** to support this feature effectively.
- **Customization Expectations:** Users are assumed to value personalization, including layout choices, color themes (via **react-color**), and visual polish (via **tailwindcss-animate** and **lucide-react** icons).

7. Project Scope Assumptions

- **Limited Offline Functionality:** The application assumes a connected experience and does not aim to support fully offline resume editing.
- **Scalability Needs:** The system is assumed to be deployed at a small to medium scale, primarily for individual users (students, job seekers), not enterprise use.

8 Future Scope and further enhancement of the Project

The **ResumeAI** project demonstrates a modern, scalable, and highly interactive approach to resume building, incorporating dynamic UI/UX patterns and AI integration.

While the current version delivers a robust feature set for users to create and customize professional resumes, there remains significant scope for future growth and enhancement. These improvements can cater to broader user needs, enhance usability, improve performance, and expand the application's overall capabilities.

1. Advanced AI Features

- **Context-aware Suggestions:** The existing AI integration can be enhanced by leveraging advanced GPT models to provide role-specific resume suggestions based on job descriptions or user input.
- **Automated Job Matching:** Integration with job portals or APIs (e.g., LinkedIn, Indeed) to suggest jobs matching the resume and allow auto-tailoring based on specific job listings.
- **AI-based Resume Scoring:** Implement a feature where the AI can evaluate and score the user's resume against industry best practices or a selected job role.

2. Collaborative Editing

- Enable users to collaborate in real-time with peers, mentors, or professional resume writers.
- Integrate commenting and suggestion features similar to Google Docs to improve collaborative feedback.

3. Custom Templates Marketplace

- Develop a **template marketplace** where users can upload and share their custom resume templates.
- Include features such as rating, reviews, and paid premium templates using a micro-payment system (e.g., via Stripe).

4. Multilingual Resume Support

- Allow users to build resumes in multiple languages.
- Integrate translation services (e.g., Google Translate API or DeepL) to auto-translate resume content, improving accessibility for non-English-speaking users.

5. Mobile App Development

- Extend ResumeAI into a cross-platform mobile application using **React Native** or **Flutter**, providing users with the flexibility to build and edit resumes on the go.
- Incorporate offline editing and sync capabilities to enhance user convenience.

6. ATS Optimization Checker

- Integrate a system to analyze resumes for **ATS (Applicant Tracking System)** compatibility.
- Provide tips or validation to ensure proper keyword usage, formatting, and section structure for better hiring chances.

7. Enhanced Analytics and Insights

- Offer resume performance analytics, such as tracking views/downloads or user engagement statistics.
- Provide insights based on user behavior (e.g., which templates are most used, average resume length, etc.).

8. Integration with LinkedIn and GitHub

- Allow users to import data from LinkedIn, GitHub, or other platforms to auto-fill experience, skills, and project sections.
- Enable exporting resumes directly to LinkedIn or as interactive portfolio websites.

9. Resume Versioning and History

- Implement resume version control, allowing users to maintain and compare different versions of their resumes.
- Users can revert to previous states or track changes over time.

10. Accessibility Enhancements

- Improve compliance with **WCAG (Web Content Accessibility Guidelines)** to ensure inclusive design for all users, including those using screen readers or requiring high contrast modes.

9 Project Repository Location

S#	Project Artifacts (softcopy)	Location (Folder Name, Drive Link etc.)	Verified by Project Guide	Verified by HOD
1.	Project Synopsis Report (Final Version)	https://shorturl.at/MO059	Name and Signature	Name and Signature
2.	Project Progress updates	https://shorturl.at/ClnJc	Name and Signature	Name and Signature
3.	Project Report (Final Version)	https://shorturl.at/s3lIC	Name and Signature	Name and Signature
4.	Source Code	https://github.com/Arsdeep/resume_builder	Name and Signature	Name and Signature
5.	Certificate Copy	https://shorturl.at/weVSc	Name and Signature	Name and Signature

10 Definitions, Acronyms, and Abbreviations

Abbreviation	Description
@dnd-kit	A modern, accessible drag-and-drop toolkit for React.
SSR	Server-Side Rendering – Webpage content is generated on the server and sent to the client, improving SEO and initial load time.
SSG	Static Site Generation – Pre-rendering pages at build time for better performance and caching.
ORM	Object-Relational Mapping – A programming technique for converting data between relational databases and object-oriented programming languages.

11 Conclusion

The development of **ResumeAI** marks the successful culmination of a comprehensive, full-stack web application project that combines modern web technologies to solve a real-world problem—creating professional resumes in a dynamic, interactive, and user-friendly manner.

Designed with a focus on usability, performance, and extensibility, ResumeAI empowers users to craft personalized resumes with ease, precision, and visual appeal.

React 19 served as the backbone for building reusable and responsive UI components, while **TypeScript** introduced strong typing and tooling support, boosting development efficiency and reducing runtime errors.

From a design and user experience standpoint, we prioritized interactivity and customization. The integration of **Tailwind CSS** and **Radix UI** provided a flexible and accessible design system, while libraries like **lucide-react**, **react-color**, and **next-themes** contributed to aesthetic appeal and personalization. **Drag-and-drop functionality**, implemented using the **@dnd-kit** library, added intuitive layout management for users creating their resumes.

On the functional side, **react-hook-form** in conjunction with **zod** and **@hookform/resolvers** offered robust form validation, ensuring data consistency and a smooth input process. Global state management through **zustand** helped maintain performance and predictable state flows, especially in complex UI interactions.

Authentication and user data security were addressed using **Clerk**, which simplified login flows and provided a secure, extensible authentication system. Features such as **resume export (PDF/print)** using **react-to-print** and **AI-powered content suggestions** via the **OpenAI API** significantly enhanced the utility of the application, making it not just a tool, but a smart assistant for resume building.

On the backend, **Prisma ORM** enabled seamless and type-safe database interactions, while **@vercel/blob** supported media storage. Environment configuration was efficiently managed through **@t3-oss/env-nextjs**, ensuring clean separation of concerns across environments.

The project demonstrates not only technical proficiency but also an understanding of real-world user needs. The platform supports multiple resumes, personalized themes, and live previews, offering a truly dynamic resume-building experience.

In conclusion, **ResumeAI** successfully meets its objectives as a robust, scalable, and user-centric web application. It has proven to be an invaluable learning experience, showcasing how various technologies can be orchestrated to build a modern SaaS product. The knowledge gained and the skills honed throughout this project lay a strong foundation for future endeavors in full-stack development and product design.

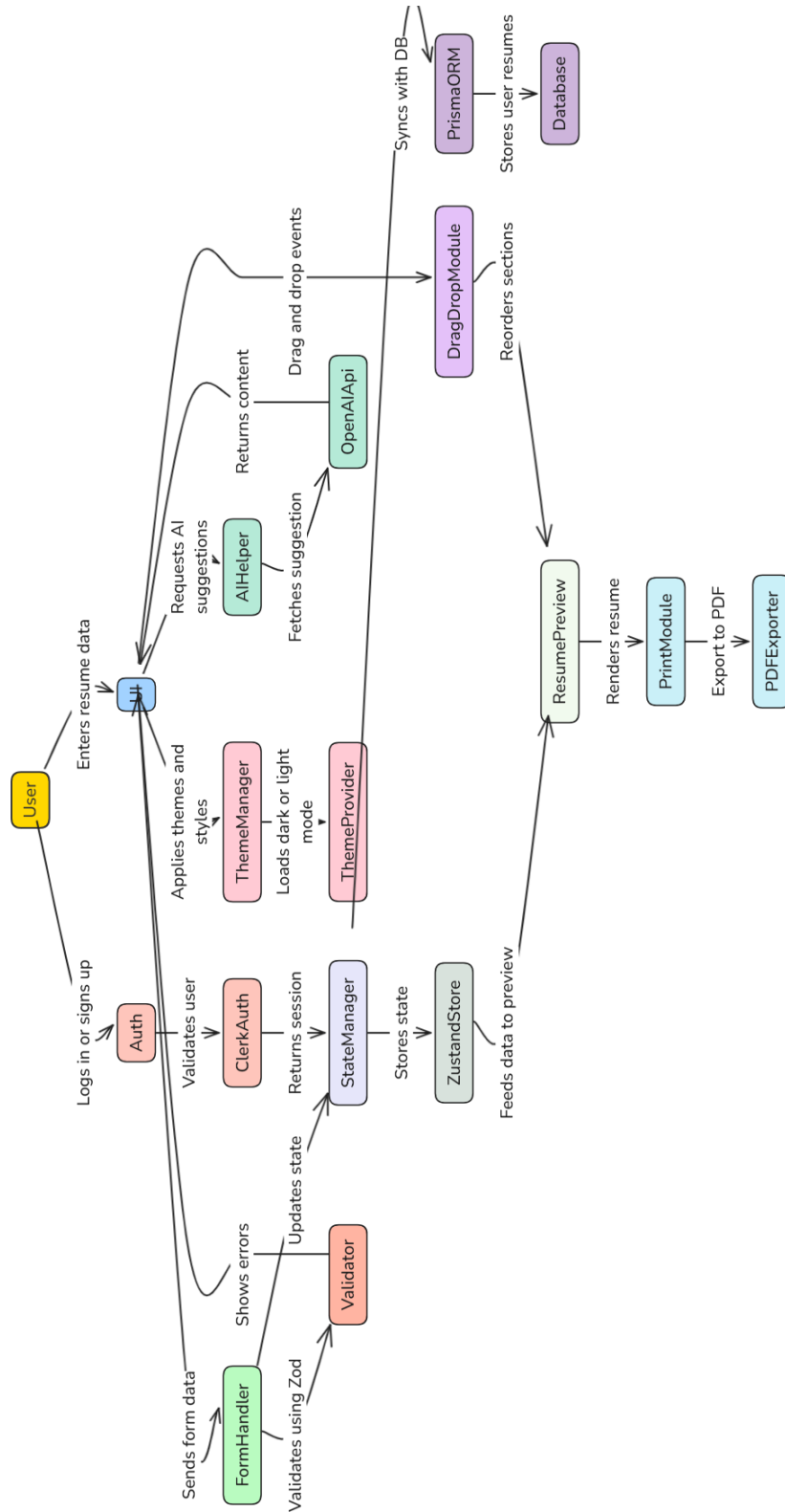
12 References

References to External Documents

1. Vercel. (n.d.). *Next.js Documentation*. Retrieved from <https://nextjs.org/docs>
2. React. (n.d.). *React – A JavaScript library for building user interfaces*. Retrieved from <https://reactjs.org/>
3. Microsoft. (n.d.). *TypeScript: JavaScript with syntax for types*. Retrieved from <https://www.typescriptlang.org/>
4. Tailwind Labs. (n.d.). *Tailwind CSS Documentation*. Retrieved from <https://tailwindcss.com/docs>
5. Radix UI. (n.d.). *Radix Primitives – Unstyled, accessible UI components*. Retrieved from <https://www.radix-ui.com/docs>
6. Lucide. (n.d.). *Lucide – Beautiful & consistent icon toolkit*. Retrieved from <https://lucide.dev/>
7. Clerk.dev. (n.d.). *Clerk: Complete User Management*. Retrieved from <https://clerk.com/docs>
8. react-hook-form. (n.d.). *React Hook Form – Performant, flexible and extensible forms*. Retrieved from <https://react-hook-form.com/>
9. colinhacks. (n.d.). *Zod – TypeScript-first schema validation*. Retrieved from <https://zod.dev/>
10. Zustand. (n.d.). *Zustand – Bear necessities for state management*. Retrieved from <https://zustand-demo.pmnd.rs/>
11. dnd-kit. (n.d.). *@dnd-kit – Drag and drop toolkit for React*. Retrieved from <https://docs.dndkit.com/>
12. OpenAI. (n.d.). *OpenAI API Documentation*. Retrieved from <https://platform.openai.com/docs>
13. Prisma. (n.d.). *Next-generation Node.js and TypeScript ORM*. Retrieved from <https://www.prisma.io/docs>
14. react-to-print. (n.d.). *ReactToPrint Documentation*. Retrieved from <https://www.npmjs.com/package/react-to-print>
15. react-color. (n.d.). *A collection of color pickers for React*. Retrieved from <https://casesandberg.github.io/react-color/>
16. date-fns. (n.d.). *Modern JavaScript date utility library*. Retrieved from <https://date-fns.org/>
17. Vercel. (n.d.). *@vercel/blob Documentation*. Retrieved from <https://vercel.com/docs/storage/vercel-blob>

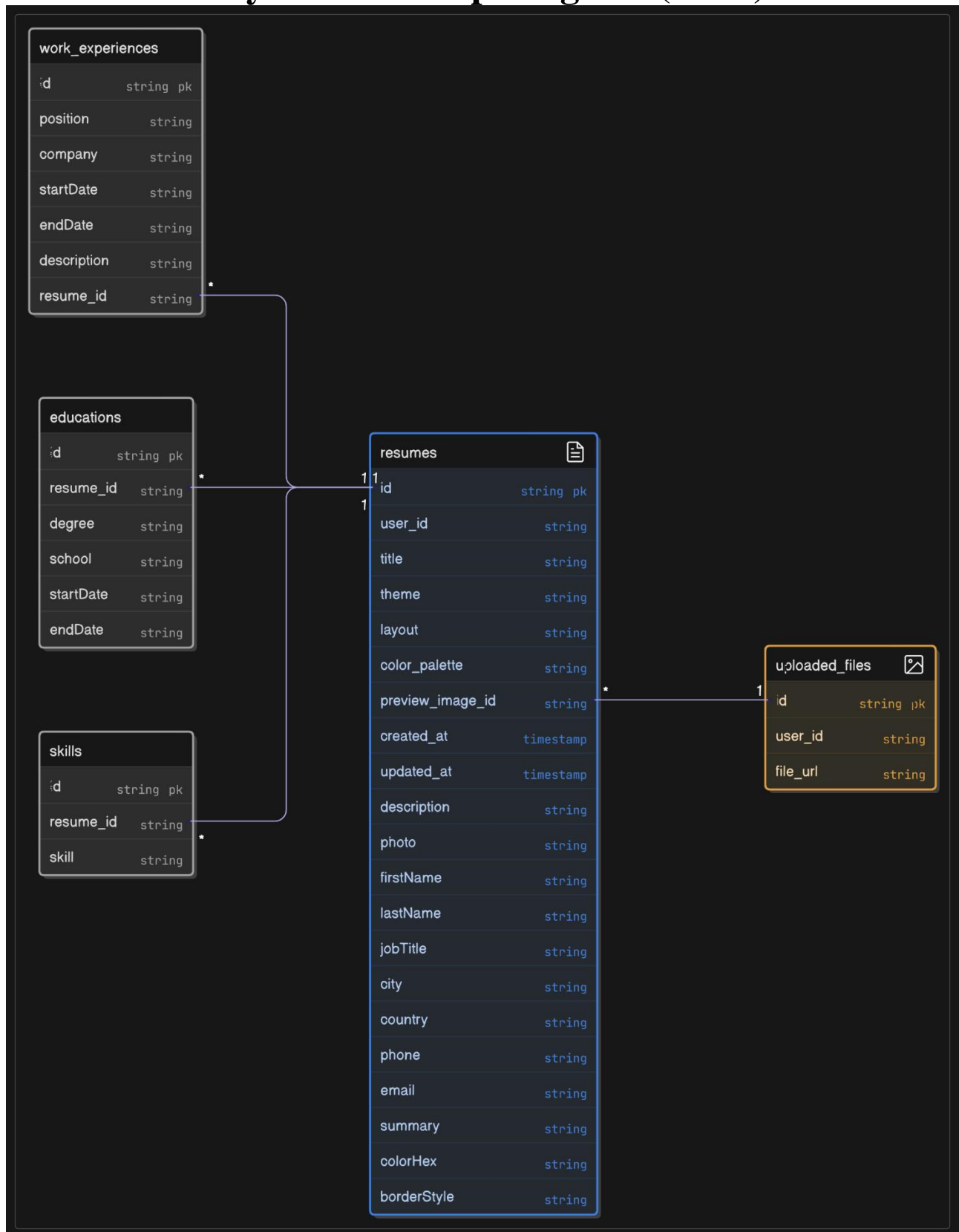
Annexure A

Data Flow Diagram (DFD)



Annexure B

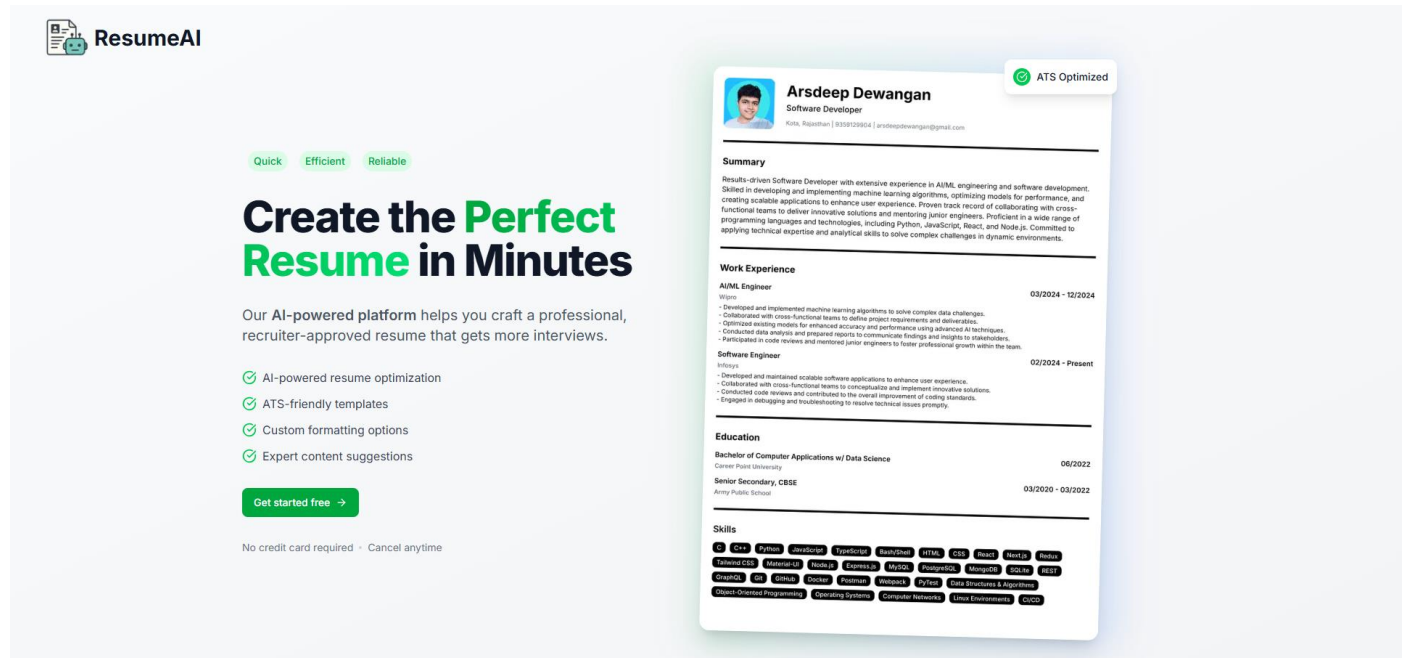
Entity-Relationship Diagram (ERD)



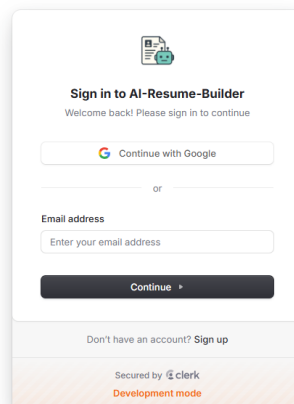
Annexure C


Screen Shots



Home Page



Login Page



 Resume AI



Create Resume

Your Resumes

Total : 2

Untitled

Updated on May 5, 2025 11:50 AM

Arsdeep Dewangan

03559125904 | arsddeepdewangan@gmail.com

Skills

Python, JavaScript, C++, React, Node.js, Docker

Software Dev

Updated on May 3, 2025 10:00 PM

Arsdeep Dewangan

Software Developer

03559125904 | arsddeepdewangan@gmail.com

Summary

Detail-oriented and innovative Software Developer with extensive experience in software engineering and AI/ML engineering. Proven ability to develop and maintain scalable applications while optimizing machine learning algorithms for enhanced performance. Adept at collaborating with cross-functional teams to define project requirements and implement cutting-edge solutions. Strong proficiency in programming languages including Python, JavaScript, and C++, along with expertise in frameworks and tools such as React, Node.js, and Docker. Committed to continuous improvement and mentoring fellow engineers to foster a culture of growth and excellence.

Work Experience

AI/ML Engineer

03/2024 - 12/2024

Wipro

- Developed and implemented machine learning algorithms to solve complex data challenges.

- Collaborated with cross-functional teams to define project requirements and deliverables.

- Optimized existing models for enhanced accuracy and performance using advanced AI techniques.

- Conducted data analysis and prepared reports to communicate findings and insights to stakeholders.

- Participated in code reviews and mentored junior engineers to foster professional growth within the team.

Software Engineer

02/2024 - Present

Infosys

- Developed and maintained scalable software applications to enhance user experience.

- Collaborated with cross-functional teams to conceptualize and implement innovative solutions.

- Conducted code reviews and contributed to the overall improvement of coding standards.

- Engaged in debugging and troubleshooting to resolve technical issues promptly.


Education



06/2022

Bachelor of Computer Applications w/ Data Science

Skills

Python, JavaScript, C++, React, Node.js, Docker

 Resume AI



Design Your Resume

Customize your resume with our easy-to-use editor. Progress saved automatically.

General Info > Personal Info > Work Experience > Education > Skills > Summary

Work Experience

Add as many work experiences as you want.

Work Experience 1

Smart Fill (AI)

Job Title

AI/ML Engineer

Company

Wipro

Start Date

15-03-2024

End Date

31-12-2024

If you are currently working in this role, leave the end date empty.

Description

- Developed and implemented machine learning algorithms to solve complex data challenges.

- Collaborated with cross-functional teams to define project requirements and deliverables.


- Optimized existing models for enhanced accuracy and performance using advanced AI techniques.

- Conducted data analysis and prepared reports to communicate findings and insights to stakeholders.

- Participated in code reviews and mentored junior engineers to foster professional growth within the team.

Previous Step

Next Step



Arsdeep Dewangan

Software Developer

Kota, Rajasthan | 9359125904 | arsddeepdewangan@gmail.com

Summary

Detail-oriented and innovative Software Developer with extensive experience in software engineering and AI/ML engineering. Proven ability to develop and maintain scalable applications while optimizing machine learning algorithms for enhanced performance. Adept at collaborating with cross-functional teams to define project requirements and implement cutting-edge solutions. Strong proficiency in programming languages including Python, JavaScript, and C++, along with expertise in frameworks and tools such as React, Node.js, and Docker. Committed to continuous improvement and mentoring fellow engineers to foster a culture of growth and excellence.

Work Experience

AI/ML Engineer

03/2024 - 12/2024

Wipro

- Developed and implemented machine learning algorithms to solve complex data challenges.

- Collaborated with cross-functional teams to define project requirements and deliverables.

- Optimized existing models for enhanced accuracy and performance using advanced AI techniques.

- Conducted data analysis and prepared reports to communicate findings and insights to stakeholders.

- Participated in code reviews and mentored junior engineers to foster professional growth within the team.

Software Engineer

02/2024 - Present

Infosys

- Developed and maintained scalable software applications to enhance user experience.

- Collaborated with cross-functional teams to conceptualize and implement innovative solutions.

- Conducted code reviews and contributed to the overall improvement of coding standards.

- Engaged in debugging and troubleshooting to resolve technical issues promptly.

Education

06/2022

Bachelor of Computer Applications w/ Data Science

CAREER POINT UNIVERSITY, KOTA
PLAGIARISM VERIFICATION REPORT

Date:

**Type of Document
(Tick):**

**PhD
Thesis**

MCA Dissertation/ Report

**BCA Project
Report**

Paper

Name: Arsdeep Dewangan

Department: Computer Applications

Enrolment No: K23334

Contact No: 9351929904

E-mail: arsheepdewangan@gmail.com

Name of the Supervisor: Mr. Arshad Hussain

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): RESUMEAI

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

- Total No. of Pages = 61
- Total No. of Preliminary pages = 6
- Total No. of pages accommodate bibliography/references =1

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at __(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

Copy Received on	Excluded	Similarity Index (%)	Abstract & Chapters Details	
	<ul style="list-style-type: none">• All Preliminary Pages• Bibliography/ Images/Quotes• 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Page counts	
			File Size	

Checked by – (Name & Signature)