

A CAPSTONE PROJECT REPORT ON

AI POWERED RESUME BUILDER

Submitted in partial fulfilment for the award of NGA-Pre Skilling Training in

WIPRO NGA – C#.NET FULL STACK REACT

Offered by

GREAT LEARNING

Submitted by

Vinay Kumar Rayudu

Email: vinayrayudu1777@gmail.com

Github: <https://github.com/vinayrayudu0710/Capstone/>

MARCH 2024

TABLE OF CONTENTS

1. INTRODUCTION.....	1-1
2. PROJECT REQUIREMENT.....	2-2
3. SYSTEM ARCHITECTURE OVERVIEW	3-3
4. BACKEND OVERVIEW	4-4
5. FRONTEND OVERVIEW	5-5
6. DATA FLOW	6-6
7. TEST CASES AND UNIT TESTING	7-7
8. DATABASE DESIGN.....	8-8
9. SCREENSHOTS AND DESCRIPTION	9-11
10 SWAGGER API'S.....	12
11 DATABASE SCHEMA AND SCRIPTS	13
12 DEPLOYMENT.....	14

1. INTRODUCTION

- The Introduction sets the stage for the **AI-Powered Resume Builder** project. It explains the purpose, scope, and the audience it serves.

- **Purpose:**

The AI-Powered Resume Builder is designed to help job seekers create compelling resumes that stand out to recruiters. The system provides intelligent suggestions to improve resume content, offers a user-friendly interface for creating and editing resumes, and enables secure PDF downloads.

- **Technology Advantage:**

The use of **ReactJS** ensures a highly responsive and dynamic frontend, while **ASP.NET Core** provides scalability and performance optimization on the backend.

- **Unique Features:**

- **AI-Powered Improvements :** Provides smart improvements to enhance resume content.

- **User-Friendly UI:** Allows users to easily create, edit, and format resumes.

- **Secure PDF Downloads:** Enables users to download their resumes as PDFs securely.

- **Role-Based Access Control:** Ensures secure access for guest users, registered users, and admins.

2. PROJECT REQUIREMENT

- **Problem Statement:**

Many job seekers struggle with creating a compelling resume that stands out to recruiters. Traditional resume builders offer basic templates but lack intelligent suggestions to improve content. This project aims to build an AI-powered resume builder that provides smart improvements to enhance resumes, allows users to create and edit resumes with a user-friendly UI, and enables secure PDF downloads.

- **User Story-Based Requirements:**

User Roles:

- **Guest User:** Can view the homepage and explore features but cannot create or download resumes.
- **Registered User:** Can create, edit, and download resumes.
- **Admin:** Manages users and monitors system performance.

User Stories:

- **User Registration & Authentication:**

- As a job seeker, I want to create an account using my email/password so that I can save my resumes.
- As a user, I want to securely log in using JWT authentication so that my data is protected.

- **Resume Creation :**

- As a user, I want to enter my personal and professional details to create a resume.
- As a user, I want AI-powered for improving my resume content to enhance its quality.

- **PDF Generation & Download:**

- As a user, I want to generate and download my resume as a PDF so that I can share it with recruiters.

2. SYSTEM ARCHITECTURE OVERVIEW

Blog CMS is designed with a modular, layered architecture for scalability and maintainability.

Three Layers of Architecture:

1. **Frontend:** The Frontend Layer forms the user interface of the Resume Builder, designed with **ReactJS**, a powerful library for building modern web applications.
2. **Backend:** The Backend Layer powers the business logic and acts as a bridge between the frontend and the database. Built with ASP.NET Core Web API, it handles requests, processes data, and sends responses.
3. **Database:** The Database Layer is where the data is stored, organized, and managed. Blog CMS uses SQL Server as the database, while Entity Framework Core (EF Core) acts as the Object Relational Mapper (ORM) for database interactions.

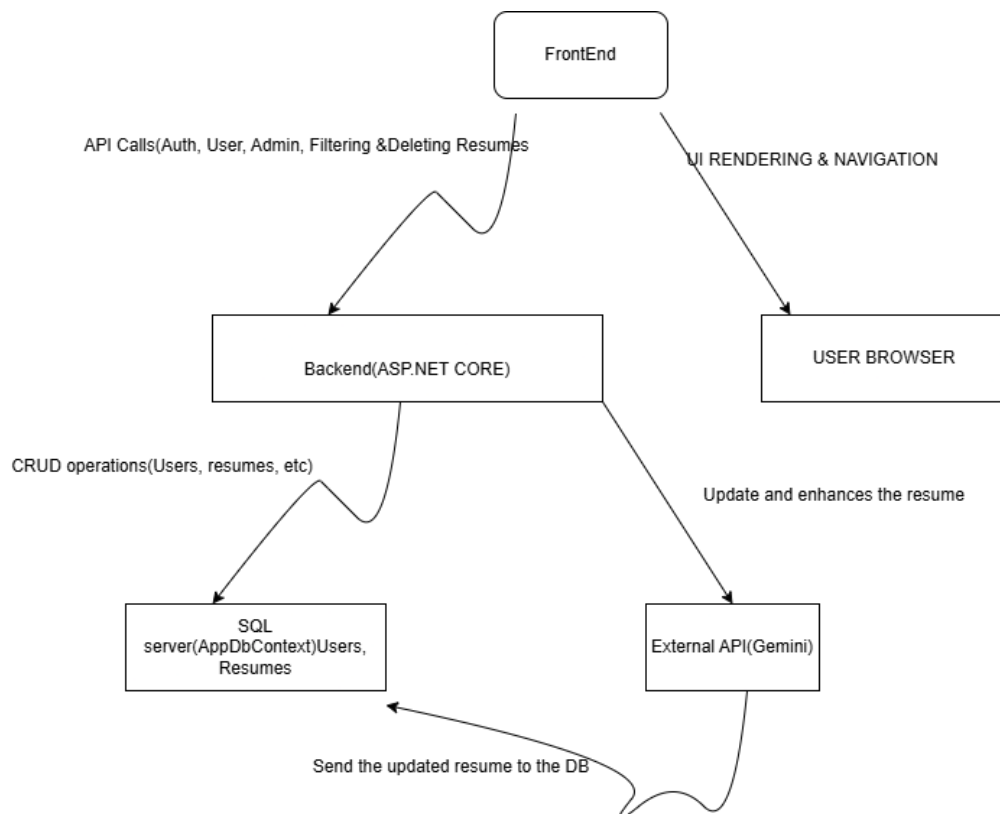


Fig.1: System Architecture Diagram

3. BACKEND OVERVIEW

This module focuses on backend implementation. It explains how APIs, controllers, and services are structured.

AuthController.cs:

- Endpoint: /api/Auth/signup – Registers users by encrypting passwords.
- Endpoint: /api/Auth/login – Authenticates and issues JWT tokens.

ResumeController.cs:

- Endpoint: /api/my-resumes – Allows bloggers to create, edit, and fetch posts.

AdminController.cs:

- Endpoint: /api/Admin/users – Fetches users.
- Endpoint: /api/Admin/ /users/{userId}/resumes – Fetches the resumes of a user
- Endpoint: /api/Admin/users/{userId} – Deletes the user by Id
- Endpoint: /api/Admin/ /users/{userId}/resumes/{resumeId} – Deletes the resumes of selected user

Business Logic and Database Integration:

- **Authentication Services:** These manage user sessions by validating JWT tokens and determining user roles (e.g., user, admin). They ensure that only authorized users access specific features, adding a layer of security.
- **Entity Framework Core for Database Interactions:** EF Core simplifies the communication between the application and the SQL Server database. It maps C# objects to database tables, enabling operations like data retrieval, updates, and deletion with ease.
- **Input Validation and Error Handling:** This ensures that only clean and valid data enters the system. Common input issues (like invalid email formats or exceeding character limits) are caught early to prevent runtime errors and maintain system stability

□

4. FRONTEND OVERVIEW:

The frontend ensures a seamless user experience using **ReactJS**. This module explains the components and pages in detail.

Key Components:

- **Navbar.js:** Displays navigation links used for travelling between pages like “Generate Resume”, and My Resumes page and logout.
-
- **ResumeEditor.js:** A specialized component that allows users to create and edit resumes with AI-powered suggestions.
- **AdminRoute:** Display a selected dashboard for the admin, while users will see general dashboard

Key Pages:

- **Home.js:** Displays the homepage with features for guest users.
- **SignUP.js:** Guest users can able to register from here.
- **Login.js:** Registered Users can be able to login from here
- **Resume Builder.js:** LoggedIn users can able to generate, edit and download the resumes from here.
- **MyResumes.js:** Allows registered users to manage their resume, and download them
- **AdminPanel.js:** Lists user details and the resumes generated by the users and should be able to delete them by admins.

5. DATA FLOW :

This module explains how information moves throughout the system.

Steps in the Data Flow:

1. **User Interaction:** Users interact via the frontend interface (e.g., submitting a resume).
2. **API Calls:** Actions trigger HTTP requests (e.g., POST /api/Resume) from the frontend to the backend.
3. **Backend Processing:** The backend validates the data, executes business logic, and interacts with the database.
4. **Database Update:** The SQL database is updated or queried (e.g., creating a new resume entry).
5. **Frontend Update:** The response data (e.g., new resume ID) is rendered on the frontend dynamically.

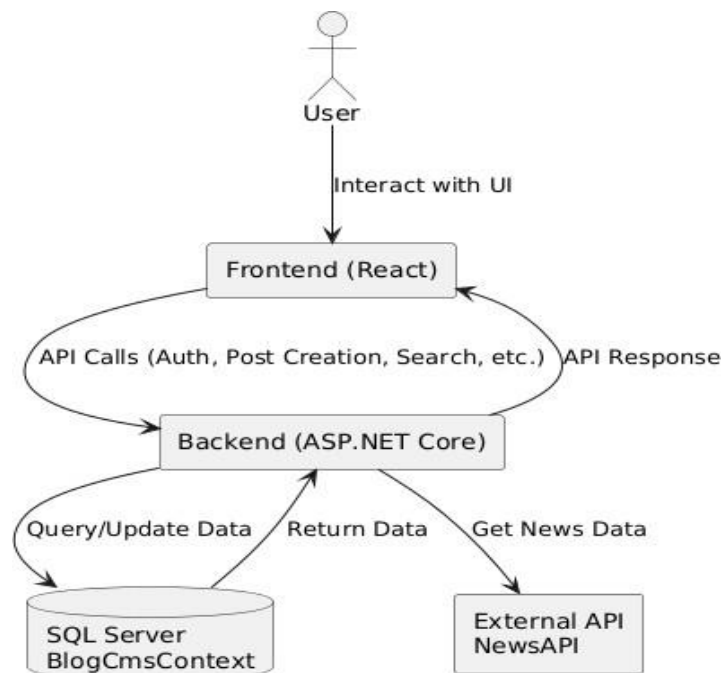


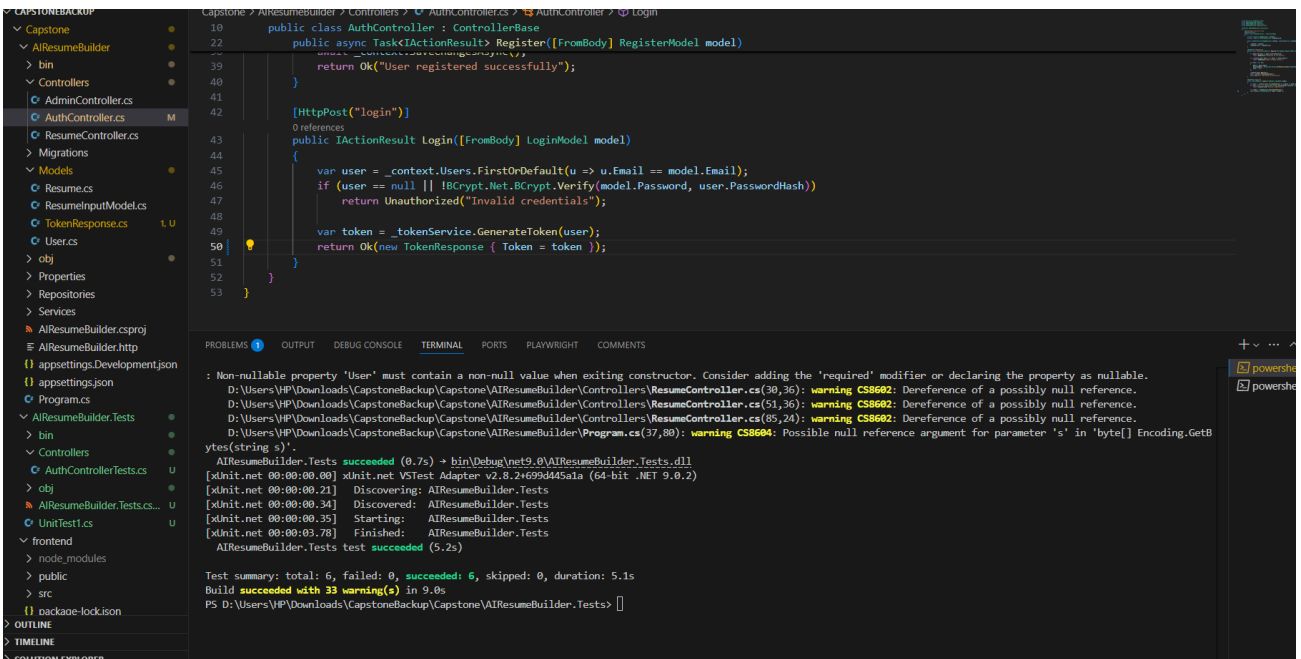
Fig.2: Data Flow Diagram

6. TEST CASES UNIT TESTING

To ensure reliability and functionality, both backend and frontend unit testing were implemented using industry-standard frameworks. This testing validates the correctness of controllers, business logic, database interactions, and UI components, ensuring error-free operation across all modules. Thorough testing ensures robust application performance and user satisfaction.

Backend Unit Testing:

- **Frameworks:** XUnit for ASP.NET Core.
- **Test Cases:**
 - Validated controller endpoints to ensure expected responses for valid and invalid inputs.
 - Test business logic for token validation and role-based access control.
 - Ensure EF Core queries perform CRUD operations correctly, maintaining database integrity.



The screenshot displays the Visual Studio IDE. On the left, the Solution Explorer shows the project structure for 'CapstoneBackup' and 'AIResumeBuilder'. The 'AuthController.cs' file is selected under 'Controllers'. The main editor shows the code for the 'AuthController' class, which inherits from 'ControllerBase'. It contains two methods: 'Register' and 'Login'. The 'Login' method is highlighted, showing it takes a 'LoginModel' and returns an 'ActionResult'. It uses 'context.Users' to find a user by email, verifies the password, and generates a token if successful.

Below the code editor, the 'TEST RESULTS' pane shows the output of the unit tests. It lists several warnings (CS8602) related to nullable references and a successful test result for 'AIResumeBuilder.Tests'. The test summary at the bottom indicates that all tests passed successfully.

```
public class AuthController : ControllerBase
{
    public async Task<ActionResult> Register([FromBody] RegisterModel model)
    {
        return Ok("User registered successfully");
    }

    [HttpPost("login")]
    public IActionResult Login([FromBody] LoginModel model)
    {
        var user = _context.Users.FirstOrDefault(u => u.Email == model.Email);
        if (user == null || !BCrypt.Net.BCrypt.Verify(model.Password, user.PasswordHash))
            return Unauthorized("Invalid credentials");
        var token = _tokenService.GenerateToken(user);
        return Ok(new TokenResponse { Token = token });
    }
}
```

Test summary: total: 6, failed: 0, succeeded: 6, skipped: 0, duration: 5.1s
Build succeeded with 33 warning(s) in 9.0s
PS D:\Users\VP\Downloads\CapstoneBackup\Capstone\AIResumeBuilder\Tests>

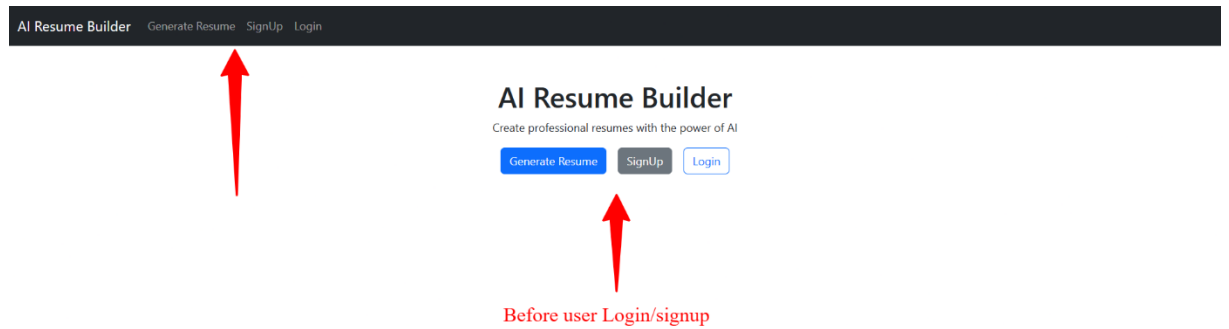
Fig.3: Backend Testcases

7.DATABASE DESIGN

Database Design:

- **Entities and Relationships:** The database is designed using a relational schema with key entities such as:
 - **Users Table:** Stores user details like UserID, Email, PasswordHash, and Role.
 - **Resumes Table:** Maintains resume-related data like ResumeID, Title, Content, and timestamps (CreatedAt, UpdatedAt).
 - **ResumeMedia Table:** Links media attachments (e.g., profile pictures) to their respective resumes through MediaID and ResumeID relationships
- **Normalization:** The schema follows database normalization principles to reduce redundancy and maintain consistency.

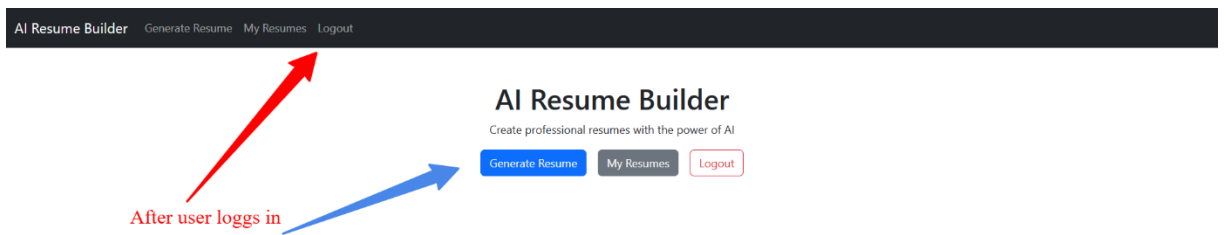
8. SCREENSHOTS AND DESCRIPTION



Guest User before Login



SignUp Page



After LoggedIn

The screenshot shows the 'Generate Resume' form. At the top, a dark navigation bar contains the text 'AI Resume Builder', 'Generate Resume', 'My Resumes', and 'Logout'. Below this, the heading 'Generate Resume' is centered. The form consists of several input fields: 'Name', 'Email', 'Phone', 'Education' (with sub-fields for 'Degree', 'Institution', and 'Year'), 'Skills', 'Experience' (with sub-fields for 'Title', 'Company', and 'Duration'), and 'Certifications'. Each section has a '+' button to add more entries. The 'Education' and 'Experience' sections have a '+' button below their respective sub-fields.

Generate Resume page

Diploma

Aditya Engineering College

2019

+

Skills

C#, JavaScript

Ms SQL, Asp.Net Core

+

Experience

Software Engineer

Sonic Solutions

1 year

+

Certifications

wipro's asp.net core, c# basics, react developer

+

GitHub

https://github.com/vinayrayudu

Languages

English

Telugu

+

Generating...

Cancel

When user clicks on generate button

AI Resume Builder Generate Resume My Resumes Logout

Generated Resume

Vinay
9876543210 | Rayudu@gmail.com | https://github.com/vinayrayudu

Summary

A highly motivated and skilled software engineer with one year of experience in developing and implementing solutions using C#, ASP.Net Core, and MS SQL. Possesses a strong understanding of software development principles and a proven ability to work effectively both independently and as part of a team. Eager to contribute to challenging projects and continue learning and growing within the field.

Education

- **B.Tech** - Pragati Engg College - 2022
- **Diploma** - Aditya Engineering College - 2019

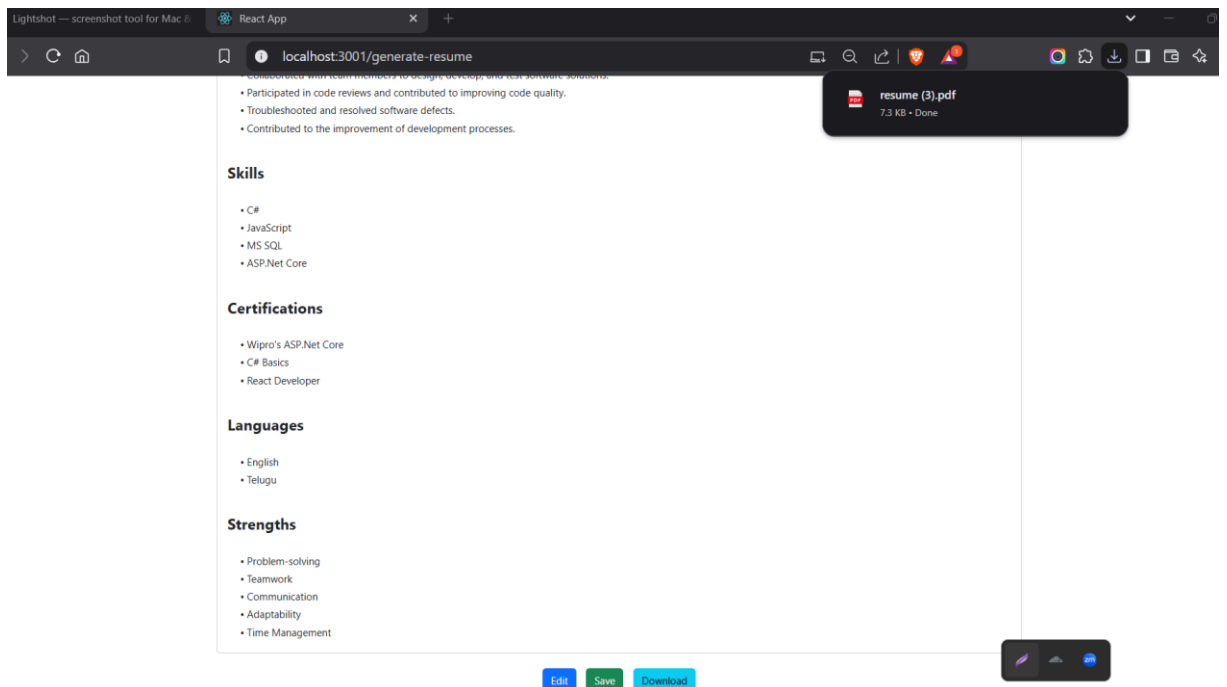
Experience

- **Software Engineer** - Sonic Solutions - 1 year
- Developed and maintained software applications using C#, ASP.Net Core, and MS SQL.
- Collaborated with team members to design, develop, and test software solutions.
- Participated in code reviews and contributed to improving code quality.
- Troubleshooted and resolved software defects.
- Contributed to the improvement of development processes.

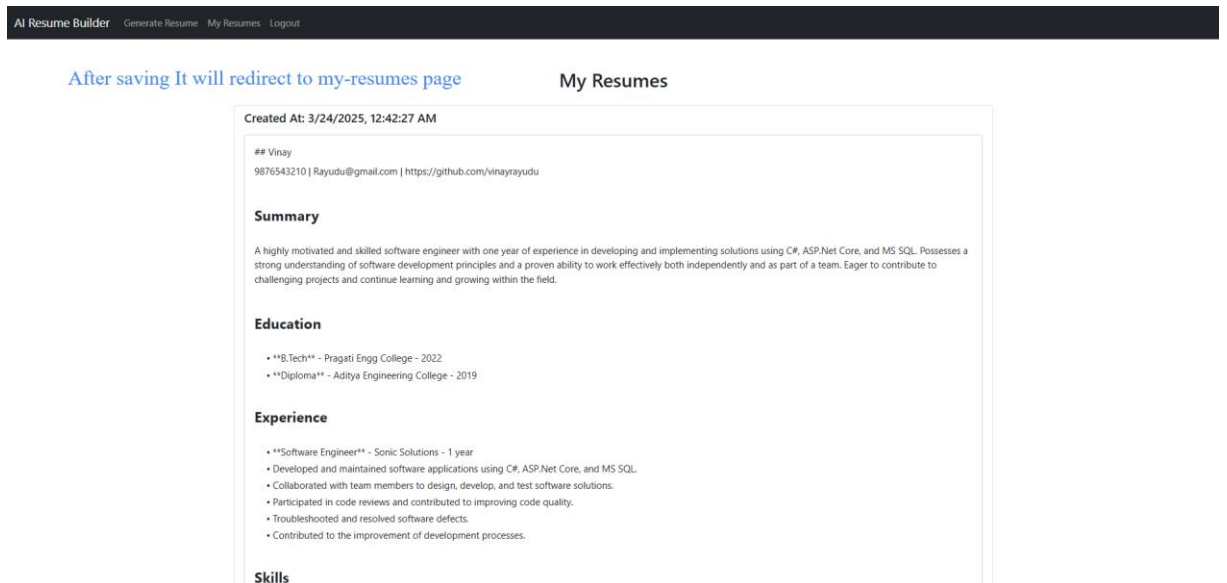
Skills

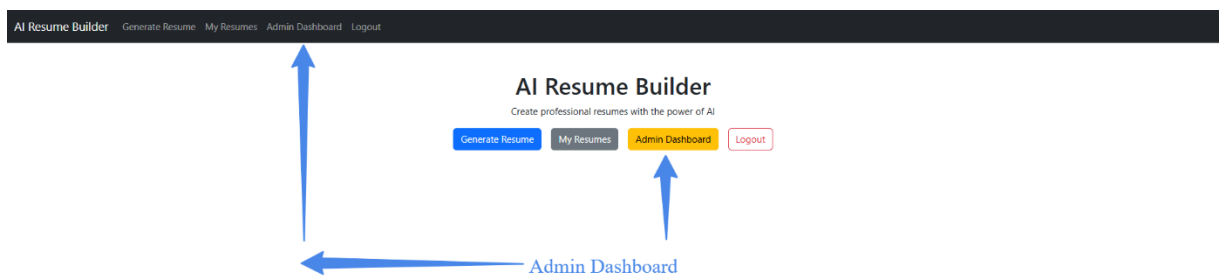
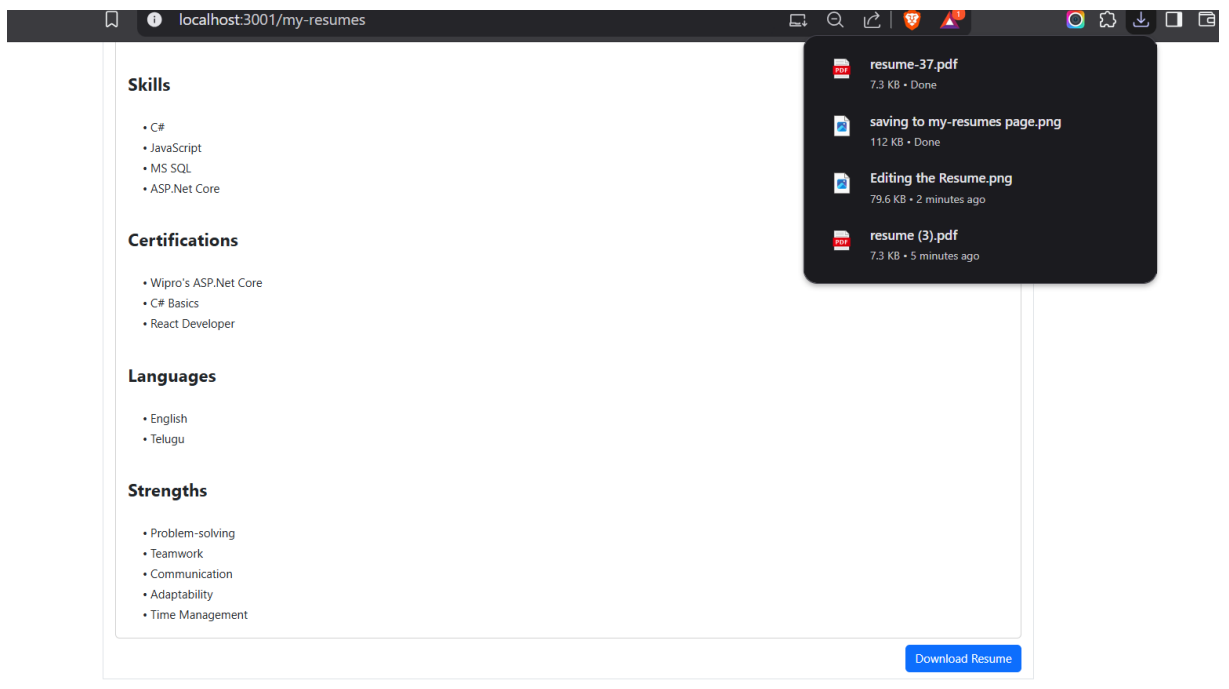
- C#

Resume Generated



Downloading Resume





Admin Dashboard

Admin Dashboard Users

Users Listing Page

Email	ID	Role	Actions	
admin@admin.com	4	Admin	View Resumes	Delete
4@gmail.com	16	User	View Resumes	Delete
dummy@dummy.com	17	User	View Resumes	Delete
2@dummy.com	18	User	View Resumes	Delete
Dummy@user.com	19	User	View Resumes	Delete
dummy1@user.com	20	User	View Resumes	Delete
fake@user.com	21	User	View Resumes	Delete

Users

Email	ID	Role	Actions	
admin@admin.com	4	Admin	View Resumes	Delete
4@gmail.com	16	User	View Resumes	Delete
dummy@dummy.com	17	User	View Resumes	Delete
2@dummy.com	18	User	View Resumes	Delete
Dummy@user.com	19	User	View Resumes	Delete
dummy1@user.com	20	User	View Resumes	Delete
fake@user.com	21	User	View Resumes	Delete

Resumes for User ID: 20

Resumes of specific Users

Created At: 3/23/2025, 11:47:28 PM

Vinay Rayudu

Vinay@gmail.com | 9876543210 | <https://github.com/vinayrayudu>

Summary

A highly motivated and skilled software developer with two years of experience at Sonic Solutions. Proficient in C#, JavaScript, ReactJS, and ASP.Net Core with a strong foundation in MS SQL. Eager to contribute to innovative projects and driven to continuously learn and improve technical skills.

Education

- **B.Tech, Pragati Engineering College** - 2022
- **Diploma, Aditya Engineering College** - 2019
- Developed and executed test plans and test cases for web applications.
- Identified, documented, and tracked software defects.
- Worked closely with developers to resolve defects and ensure software quality.
- Conducted regression testing to ensure that new code changes did not introduce new defects.

Skills

- C#, JavaScript
- ReactJS, Asp.Net Core
- Ms SQL

Certifications

- Wipro's ASP.Net Core
- C# Basics
- React Developer

Languages

- English
- Telugu

Strengths

- Adaptable
- Problem-solving skills
- Team player
- Detail-oriented
- Time Management

Deleting Resume option for admin

Delete

User's Resume showing and deleting

AI Resume Builder Generate Resume My Resumes Admin Dashboard Logout

Admin Dashboard

Users

Email	ID	Role	Actions
admin@admin.com	4	Admin	View Resumes Delete
4@gmail.com	16	User	View Resumes Delete
dummy@dummy.com	17	User	View Resumes Delete
2@dummy.com	18	User	View Resumes Delete
Dummy@user.com	19	User	View Resumes Delete
dummy1@user.com	20	User	View Resumes Delete
fake@user.com	21	User	View Resumes Delete

Resumes for User ID: 20

No resumes found

Resumes deleted for user with ID- 20

User Listing and Deleting

Swagger API'S

The AI-Powered Resume Builder project includes a well-structured set of RESTful APIs to manage authentication, resumes, user roles, and PDF generation. These APIs ensure smooth interaction between the frontend and

localhost/swagger/index.html

Swagger

AI Resume Builder API

Admin

Auth

Resume

Schemas

Education

Experience

LoginModel

backend.

9. DEPLOYMENT

The AI-Powered Resume Builder project has been successfully deployed on GitHub for both the backend and frontend repositories. This ensures easy accessibility, version control, and smooth collaboration

GitHub Repository Links

Backend Repository:

<https://github.com/vinayrayudu0710/Capstone/tree/master/AIRResumeBuilder>

Frontend Repository:

<https://github.com/vinayrayudu0710/Capstone/tree/master/frontend>

Deployment Details

1. The frontend is built with React.js and hosted via GitHub for a seamless user experience.
2. The backend is developed using ASP.NET Core Web API and is deployed on GitHub.
3. All necessary API configurations, database connections, and authentication mechanisms have been integrated properly for a fully functional system