Mukesh Patel School of Technology & Management Engineering Department of Data Science Vile Parle (w), Mumbai- 400056

Software Requirement Specification Report

Topic: Resume Builder Web Application

By

Roll No.: S004 Aishwarya Nagchandi

Roll No.: S038 Om Lande

Roll No.: S072 Chetana Pawar

Course: IMAP

AY: 2024-25

1. Introduction

1.1 **Purpose**

This Software Requirements Specification (SRS) document provides a detailed description of the functionalities, design, and technical requirements for the "Resume Builder" web application. It is intended for developers, testers, and stakeholders involved in the project to understand the project scope, features, and objectives.

1.2 **Scope**

The Resume Builder is a web application aimed at simplifying the resume creation process by offering a user-friendly interface, real-time editing, customization options, and multiple export formats. The application provides a secure login system, allowing users to create, save, and manage multiple resumes.

1.3 **Definitions, Acronyms, and Abbreviations**

- JWT: JSON Web Tokens
- API: Application Programming Interface
- PDF: Portable Document Format
- CSS-in-JS: A styling technique where CSS is written directly in JavaScript code.

- CRUD: Create, Read, Update, Delete
- Strapi: An open-source headless CMS used for backend.

1.4 **References**

- React Documentation: https://reactjs.org/docs/getting-started.html
- Strapi Documentation: https://strapi.io/documentation
- PostgreSQL Documentation: https://www.postgresql.org/docs/

1.5 **Overview**

This document outlines the functional and non-functional requirements of the Resume Builder application. It covers the use cases, user interface requirements, and the technical stack required for the development and deployment of the application.

2. Overall Description

2.1 **Product Perspective**

The Resume Builder is a standalone web application that allows users to create professional resumes using predefined templates and customizations. It offers a responsive interface powered by React, allowing users to see real-time previews of their resumes.

2.2 **Product Features**

1. User Authentication:
   - Secure login and registration using Clerk.
   - Password recovery mechanism.
2. Resume Creation:
   - Users can input personal details, work experience, education, skills, and certifications.
   - Real-time editing and preview of resumes.
3. Resume Management:
   - Users can save multiple resumes and edit them later.
4. Export Options:
   - Export resumes in PDF and HTML formats.
5. Data Security:
   - Secure storage and processing of user data using PostgreSQL.
   - Compliance with data protection policies.
6. User Dashboard:

    - A dashboard where users can manage their saved resumes and profile information.

2.3 **User Classes and Characteristics**

- Registered User: Can create, save, and manage multiple resumes after logging in.

2.4 **Operating Environment**

- Frontend Hosting: Vercel
- Backend Hosting: Render
- Browsers Supported: Latest versions of Chrome, Firefox, Safari, and Edge.

2.5 **Design and Implementation Constraints**

- The system must maintain responsiveness across different devices (desktop, tablet, mobile).
- High traffic management, especially during peak hours.
- Data stored in PostgreSQL should be encrypted for security purposes.

2.6 **Assumptions and Dependencies**

- The user must have a stable internet connection to access and use the platform.
- Dependencies on third-party libraries such as React, Redux, Strapi, and PostgreSQL.
- The system will rely on external libraries and APIs for exporting resumes in PDF and Word formats.

3. **Functional Requirements**

3.1 **Authentication and Authorization**

- FR1: The system must allow users to register with a valid email and password.
- FR2: The system must allow registered users to log in using JWT-based authentication.
- FR3: The system must provide a password reset functionality.

3.2 **Resume Creation and Editing**

- FR4: The system must allow users to input personal details (name, contact, etc.), work experience, education, skills, and certifications.
- FR5: The system must provide a real-time preview of the resume as the user inputs information.
- FR6: The system must allow users to select different templates, fonts, and colours for their resumes.
- FR7: The system must save user data in PostgreSQL and allow users to edit their saved resumes.

3.3 **Resume Export Functionality**

- FR8: The system must allow users to export their resumes in PDF format.
- FR9: The system must allow users to export their resumes in HTML format.

3.4 **Data Security**

- FR10: The system must encrypt all user data stored in PostgreSQL.
- FR11: The system must ensure secure transfer of data between frontend and backend using HTTPS.

3.5 **Dashboard**

- FR12: The system must provide users with a dashboard where they can manage their resumes and profile information.
- FR13: The system must allow users to delete or update their saved resumes.

4. Non-Functional Requirements

4.1 **Performance Requirements**

- NFR1: The system should load resume previews within 1 second after a user inputs data.
- NFR2: The system should support up to 10,000 users concurrently.

4.2 **Security Requirements**

- NFR3: The system should comply with data protection standards, ensuring user privacy.

4.3 **Usability Requirements**

- NFR4: The system should have an intuitive UI, making it usable by individuals with basic computer literacy.
- NFR5: The system should be responsive and usable across devices (desktop, tablet, and mobile).

4.4 **Availability Requirements**

- NFR6: The system should have 99.9% uptime availability on both frontend and backend hosting environments (Vercel and Render).

4.5 **Maintainability and Support**

- NFR7: The system code should follow standard React/Strapi coding practices to ensure maintainability.

- NFR8: The system should be easily deployable with minimal downtime during updates.

5. **System Architecture**

5.1 **Technology Stack**

- Frontend: React, Redux, CSS-in-JS (e.g., styled-components)
- Backend: Strapi (Node.js-based REST API)
- Database: PostgreSQL
- Authentication: JWT (JSON Web Tokens), Clerk
- Hosting: Vercel (Frontend), Render (Backend)

5.2 **System Components**

- Frontend: Handles user interactions, data input, and real-time resume previews.
- Backend: Manages user authentication, CRUD operations for resumes, and handles secure storage of data.
- Database: PostgreSQL is used to store user data and resume templates.
- API Layer: Strapi will expose RESTful APIs for frontend communication.

6. **Components**

- Login Page
- SignUp Page
- Manage Account
- Home Page
- Templates-

  Editable Fields:

A) Personal Details:

  - ➔ Name
  - ➔ Job Title
  - ➔ Address
  - ➔ Phone
  - ➔ Email
  - ➔ Summary

B) Experience:

  - ➔ Id
  - ➔ Title
  - ➔ Company Name
  - ➔ City
  - ➔ Start Date
  - ➔ End Date
  - ➔ Currently Working
  - ➔ Summary
  - ➔ Add more

C) Education:

  - ➔ Id
  - ➔ University
  - ➔ Start date

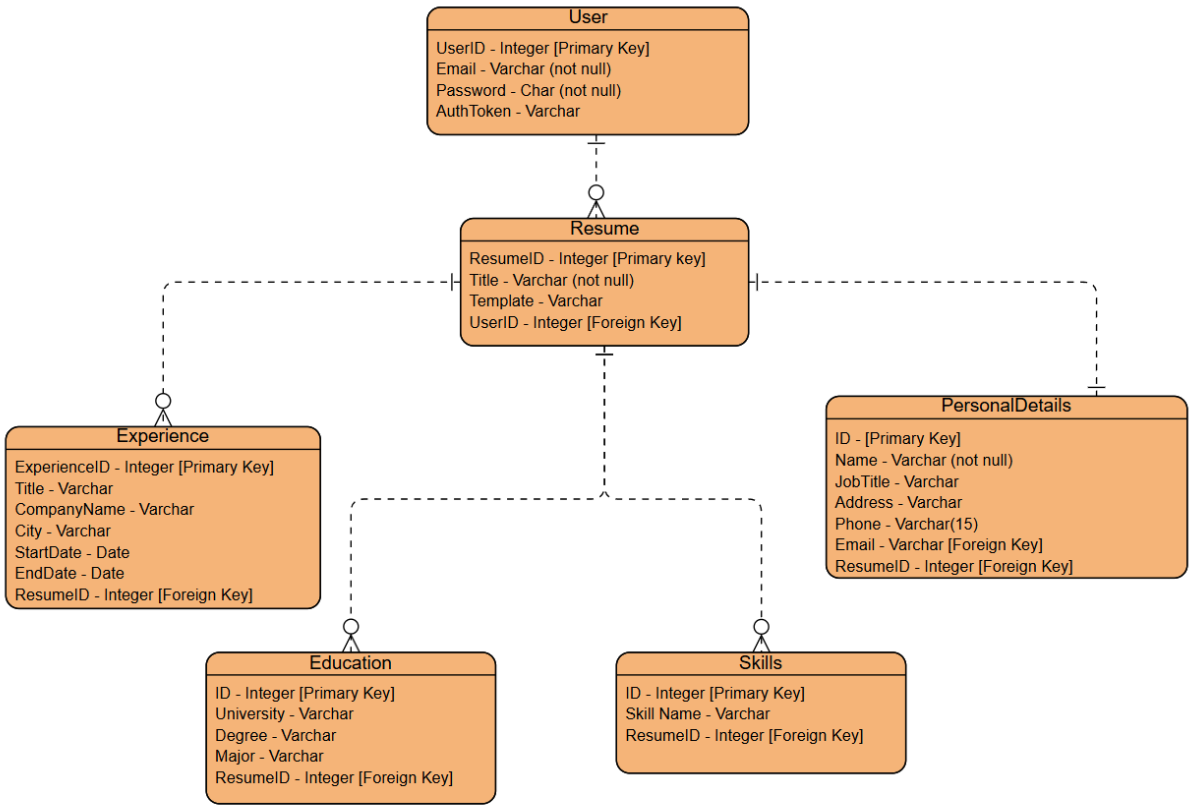
- ➔ End Date
- ➔ Degree
- ➔ Major
- ➔ Description
- ➔ Add More

D) Skills:

- ➔ Id
- ➔ Name
- ➔ Add More

- Download and Share Resume
- Change Theme
- Resume preview

7. **ERD Diagram**


User
UserID - Integer [Primary Key]
Email - Varchar (not null)
Password - Char (not null)
AuthToken - Varchar
Resume
ResumeID - Integer [Primary key]
Title - Varchar (not null)
Template - Varchar
UserID - Integer [Foreign Key]
Experience
ExperienceID - Integer [Primary Key]
Title - Varchar
CompanyName - Varchar
City - Varchar
StartDate - Date
EndDate - Date
ResumeID - Integer [Foreign Key]
PersonalDetails
ID - [Primary Key]
Name - Varchar (not null)
JobTitle - Varchar
Address - Varchar
Phone - Varchar(15)
Email - Varchar [Foreign Key]
ResumeID - Integer [Foreign Key]
Education
ID - Integer [Primary Key]
University - Varchar
Degree - Varchar
Major - Varchar
ResumeID - Integer [Foreign Key]
Skills
ID - Integer [Primary Key]
Skill Name - Varchar
ResumeID - Integer [Foreign Key]