A Project Report

On

"RESUME BUILDER"

by

RAJ MO FAHIM ZAKIR

23SS02IT161

under the mentorship of

Mr AYUSH DODIA

Assistant Professor, School of Engineering

NOVEMBER 2025

P P SAVANI SCHOOL OF ENGINEERING

P P SAVANI UNIVERSITY

NH NO.: 8, VILLAGE: DHAMDOD, TA. MANGROL, NEAR KOSAMBA, SURAT – 394 125. (GUJARAT).

CERTIFICATE

This is to certify that the Project Report submitted by **RAJ MO FAHIM ZAKIR (23SS02IT161)** to the **P P SAVANI UNIVERSITY** for the partial fulfilment of the subject credit requirements is a bonafied work carried out by the student.

This is to further certify that I have been supervising the Major/Minor Project of **RAJ MO FAHIM ZAKIR (23SS02IT161).**

The contents of this report, in full or in parts, have not been submitted to any other Institute or University for award of any degree, diploma or titles.

Sign of Faculty Mentor :

Name of Faculty Mentor: Mr Ayush Dodia

Date:

ACKNOWLEDGEMENT

This report would not have been possible without my teachers who were always there when I needed them the most. I take this chance to acknowledge them and extend my sincere gratitude for helping me make this Report possible.

I wish to thank my faculty mentor **Mr. Ayush Dodia,** Assistant Professor, School of Engineering. It has been an honor to learn under their mentorship.

As my mentor, he has constantly motivated me to remain focused on achieving my goal. Their observations and guidance helped me to establish the overall direction of the report and to move forward with learning in depth. Their vital support at each juncture, which culminated in successful completion of my project work. I express my sincere gratitude to them for constant support during the project work.

I am also thankful to faculty members of the department for constant support and guidance.

I am thankful to the Dean, School of Engineering for his initiative of imparting Project during tenure of your study making us learn new things and to help us in expanding our horizons.

Name of Student : **RAJ MO FAHIM ZAKIR**
Enrollment No : **23SS02IT161**

ABSTRACT

The Resume Builder project is a full-stack web application that makes creating professional resumes easier. In today's competitive job market, a resume is often the first contact point between job seekers and employers. This makes it an important tool for career advancement. However, many job seekers, especially students and recent grads, find it hard to format and design resumes that are both professional and impactful. This project aims to fill that gap by offering a simple, interactive, and customizable platform. It allows users to enter their information, choose from pre-designed templates, and generate a professional resume in PDF format.

The application uses modern web technologies. React.js powers the frontend, while Redux Toolkit manages the state. Node.js and Express.js handle backend services, and MongoDB manages the database. Tailwind CSS creates a responsive and visually appealing user interface. Libraries like jsPDF and react-pdf enable the downloading and exporting functions. The combination of these technologies helps ensure the application is scalable, maintainable, and performs well across devices.

The project used an iterative development approach. It focused on multiple milestones, including frontend design, backend integration, error handling, state management, and export features. Each phase improved both usability and stability. The outcome is a fully functional application that meets its initial goals and sets the stage for future innovations. Possible upgrades, such as AI-driven content suggestions, cloud synchronization, and support for multiple languages, could help the Resume Builder grow into a complete career-building tool.

Index

Diagram Index

CHAPTER 1

INTRODUCTION/OBJECTIVE

Introduction

Resume is the first meeting between you and a prospective employer more often now than ever. So, how do you want to be remembered ? Wrinkled and unorganized. Neat and structured. Long and boring. Precise and interesting. Companies do not have the time to interview every applicant that is interested in the job. If they did, there would not be a company to work for. They use an eliminating process. That's right - resumes. When a job seeker wants to apply for a job online then generally he/she needs to attach his/her resume with the email. Online Resume Builder System provides the users the popular resume formats & a better way to show their resumes to the employers. A job seeker does not need to attach a resume with every email, he/she just has to include the URL of his/her resume and the employer can view the resume online by clicking on the link and can download as well.

Objective

The Objective of Online Resume Builder is to provide a way to the customers to design their resumes according to their requirements.

The core functionalities of this system are designed to offer a comprehensive and user-friendly experience:

- **Creating resumes online :** The platform provides a streamlined process for users to build their resumes from scratch or by utilizing pre-designed templates. This includes guided input for personal details, work experience, education, skills, and other relevant sections, ensuring a complete and professional document.
- **Customizing the look and details:** Beyond basic data entry, the Online Resume Builder offers extensive customization options. Users can control the visual aesthetics of their resumes, including font styles, colors, layouts, and thematic designs.
- **Keeping track of the customers and their resumes:** A critical aspect of the system is its ability to efficiently manage both customer profiles and their associated resumes. This includes secure user accounts, enabling individuals to save, edit, and access multiple versions of their resumes at any time.

Scope

Online Resume Builder can be used in accordance with the requirements of the customers. Customers can customize their resumes with their choice of themes & details. The services are hard to be defeated by the competitors as the system is providing the customers exactly what they want.

Problem

Fresh graduates often face significant hurdles when attempting to build their first resume. The most common challenges include:

- **Lack of Content Knowledge:**Uncertainty about what sections to include and how to articulate their skills, projects, and internships effectively.
- **Formatting and Design Issues:** Difficulty in creating a layout that is both visually appealing and compliant with Applicant Tracking Systems (ATS), which are widely used by employers to screen candidates.
- **Generic and Ineffective Language:** Using weak or passive language instead of powerful action verbs and industry-specific keywords that capture an employer's attention.
- **One-Size-Fits-All Approach:** Creating a single, generic resume and using it for all job applications, which fails to target the specific requirements of different roles.

Existing tools like standard word processors are often too manual and lack the guidance required, while some online builders are overly complex or expensive. This creates a clear need for a specialized, user-friendly tool tailored for freshers.

CHAPTER 2

SYSTEM ANALYSIS

System analysis uses a combination of text and diagrammatic forms to depict requirements for data, function and behavior in a way that is relatively easy to understand, and more important, straightforward to review for correctness, completeness and consistency

1. Identification of Need

In today's competitive environment, having a professional and well-structured resume is essential for students, job seekers, and professionals. However, many individuals face difficulties in creating resumes that meet industry standards due to a lack of design skills, formatting knowledge, and understanding of how to present information effectively.

Traditional methods of resume preparation — using word processors or manual formatting — are often time-consuming, error-prone, and lack consistency. Moreover, updating or customizing a resume for different job roles can be tedious, especially when maintaining multiple versions.

The need for a Resume Builder System arises to solve these challenges by offering an automated, dynamic, and user-friendly platform that allows users to create, edit, and download resumes effortlessly. The system should provide customizable templates, ensure data accuracy, and support modern export options (like PDF), making it convenient for users to maintain and share their resumes.

2. Preliminary Investigation

The Preliminary Investigation phase is the first and one of the most crucial steps in system analysis. It involves identifying the problem, studying the current system (if any), and determining the feasibility of developing a new and improved system.

In the context of the Resume Builder Project, the investigation focused on understanding the challenges faced by users when creating resumes manually and analyzing existing tools to find improvement areas.

3. Feasibility Study

Depending on the results of the initial investigation the survey is now expanded to a more detailed feasibility study. FEASIBILITY STUDY is a test of system proposal according to its workability, impact of the organization, ability to meet needs and effective use of the resources. It focuses on these major questions: What are the users

demonstrable needs and how does a system meet them? What resources are available for a given system? What are the likely impacts of the system on the organization? Is it worth solving the problem? During feasibility analysis for this project, following primary areas of interest are to be considered. Investigation and generating ideas about a new system does this.

Technical Feasibility

Technical feasibility is the study of resource availability that may affect the ability to achieve an acceptable system. This evaluation determines whether the technology needed for the proposed system is available or not. Can the work for the project be done with current equipment, existing software technology & available personnel? Can the system be upgraded if developed? If new technology is needed then what can be developed?

This is concerned with specifying equipment and software that will successfully satisfy the user requirement. The technical needs of the system may include:

Front-end and back-end selection:

An important issue for the development of a project is the selection of suitable front-end and back-end. When we decided to develop the project we went through an extensive study to determine the most suitable platform that suits the needs of the academy as well as helps in development of the project. The aspects of our study included the following factors.

Front-end selection:

- It must have a graphical user interface that assists users that are not an advanced user of the computer.
- Scalability and extensibility.
- Flexibility.
- Robustness.

According to the organization requirement and the culture. Must provide excellent reporting features with good printing support. Platform independent. Easy to debug and maintain. Event driven programming facility.

Front-end must support some popular back end like Ms Access. According to the above stated features we selected Web Browser as the front-end for developing our project.

Back-end Selection:

- Multiple user support.
- Efficient data handling.
- Provide inherent features for security.
- Efficient data retrieval and maintenance.
- Stored procedures. Popularity.
- Operating System compatible.

- Easy to install.
- Various drivers must be available.
- Easy to implant with the Front-end.

It is essential that the process of analysis and definition be conducted in parallel with an assessment to technical feasibility. It centers on the existing computer system (hardware, software etc.) and to what extent it can support the proposed system.

Economical Feasibility

The development costs are minimal as the project relies entirely on open-source technologies. Hosting costs can be kept low using platforms like Vercel for the frontend and Render or MongoDB Atlas for the backend and database.

Operational Feasibility

It is mainly related to human organizations and political aspects. The points to be considered are:

- What changes will be brought with the system?
- What organization structures are disturbed?
- What new skills will be required? Do the existing staff members have these skills? If not, can they be trained in due course of time?

The system is operationally feasible as it is very easy for the End users to operate it.

Schedule Feasibility

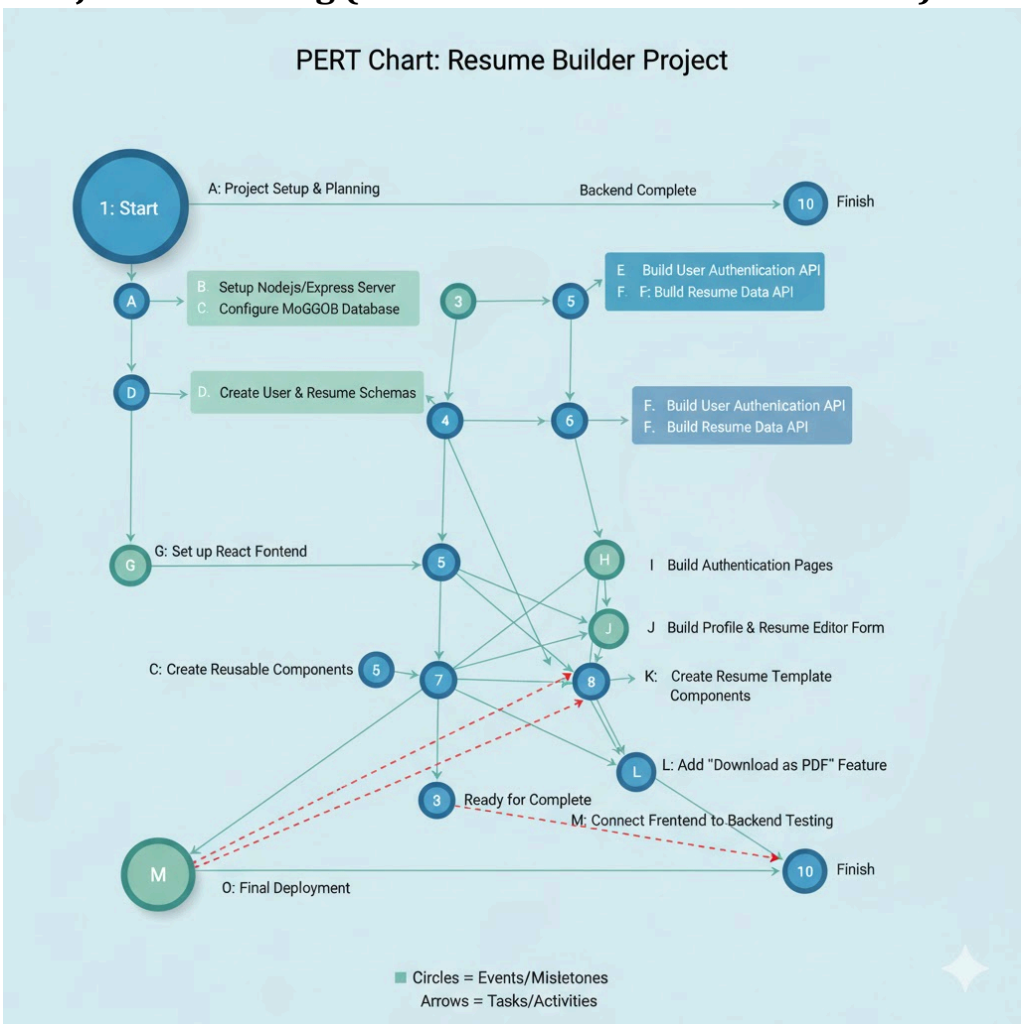
Time evaluation is the most important consideration in the development of a project. The time schedule required for the development of this project is very important since more development time affects machine time, cost and causes delay in the development of other systems. Online Resume Builder can be developed in a considerable amount of time.

4. Project Planning

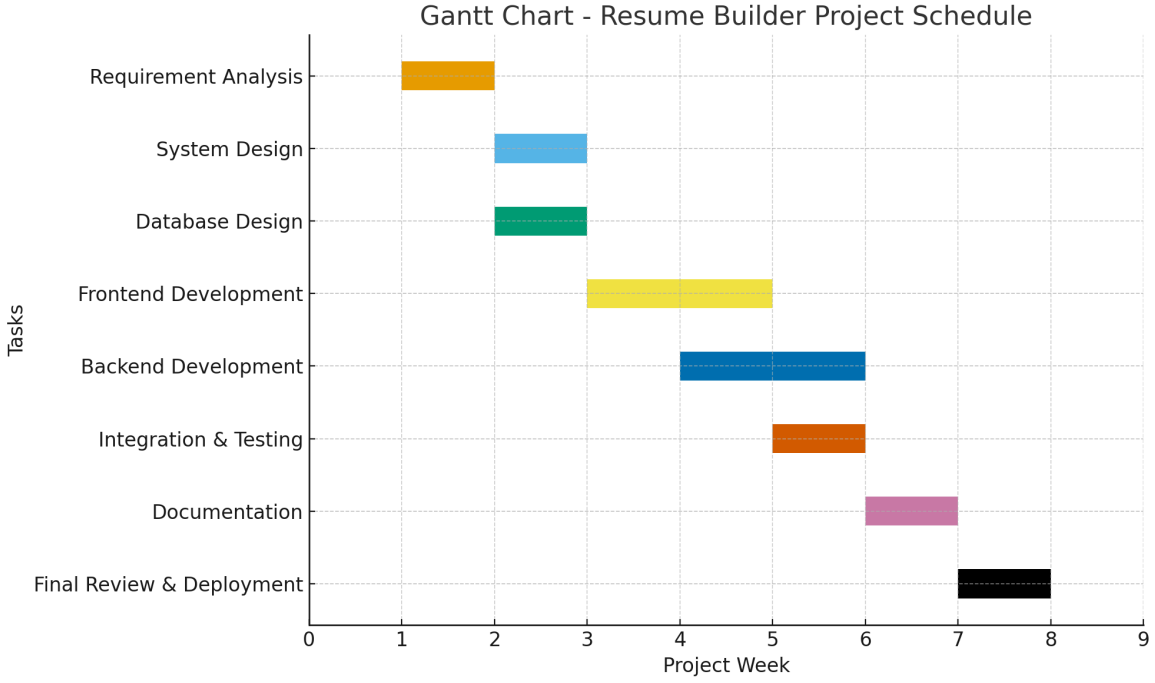
The Project Planning phase defines the overall roadmap, objectives, resources, and timeline required for the successful completion of the Resume Builder System. It ensures that the project is executed in a structured and controlled manner while meeting all functional and technical requirements.

Project planning also involves identifying tasks, setting milestones, assigning responsibilities, estimating resources, and determining deliverables to ensure timely completion of the project within scope and budget.

5. Project Scheduling (PERT Chart and Gantt Chart both)



1.1 PERT Chart



1.2 Gantt Chart

6. Software requirement specifications (SRS)

The Resume Builder project aims to provide users with an online platform to create professional resumes easily. Users can input their details, choose a template, preview the resume, and download it in PDF format.

Purpose

To simplify the resume-making process by automating formatting and providing customizable templates for students and job seekers.

Scope

The system allows user registration, dynamic addition of multiple sections (education, projects, skills), and real-time preview with secure cloud data storage.

Functional Requirements

- User registration and login
- Add, edit, and delete resume sections
- Select templates and preview resume
- Generate and download PDF
- Save data securely in the database

Non-Functional Requirements

- User-friendly and responsive UI
- Secure authentication using Firebase
- Fast resume generation (<5 seconds)
- Works across all browsers and devices

Hardware & Software Requirements

Software Requirements

Category	Details
Operating System	Win-98, Win-XP, Linux, or any other higher version
Web Browser	Internet Explorer, Mozilla Firefox, or any web browser

Hardware Requirements

Category	Details
Processor	Pentium II, Pentium III, Pentium IV or higher

Category	Details
RAM	64MB or Higher

Future Enhancements:

Add AI-based resume scoring

Multiple templates and color themes

Integration with LinkedIn

7. Software Engineering Paradigm applied

The Agile Software Development Model was applied for the development of the Resume Builder project. Agile focuses on iterative development, where requirements and solutions evolve through collaboration between the development team and stakeholders.

In this project, the development process was divided into short iterations or sprints, with each sprint delivering a functional part of the system — such as user authentication, resume creation, template design, and PDF generation. Continuous testing, feedback, and improvement were carried out at the end of each iteration to ensure quality and adaptability.

Key reasons for using the Agile Model:

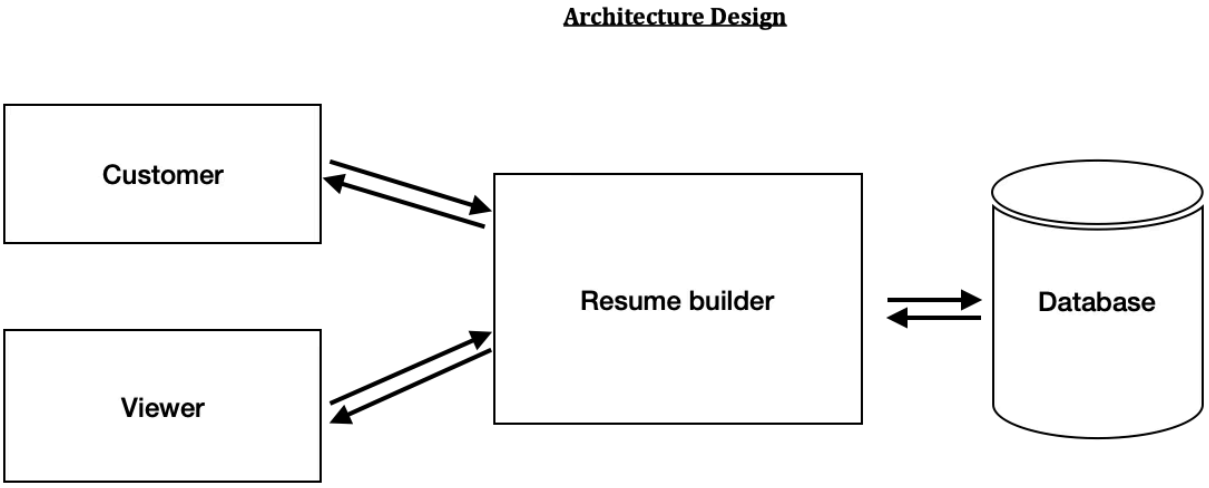
- It allows flexibility to make changes during development.
- Encourages regular communication and quick feedback.
- Ensures early and continuous delivery of working features.
- Reduces risks by testing and reviewing in every iteration.

Overall, the Agile Model helped maintain steady progress, ensured better team coordination, and resulted in a more reliable and user-centered Resume Builder system.

8. Diagrams

Architecture Design

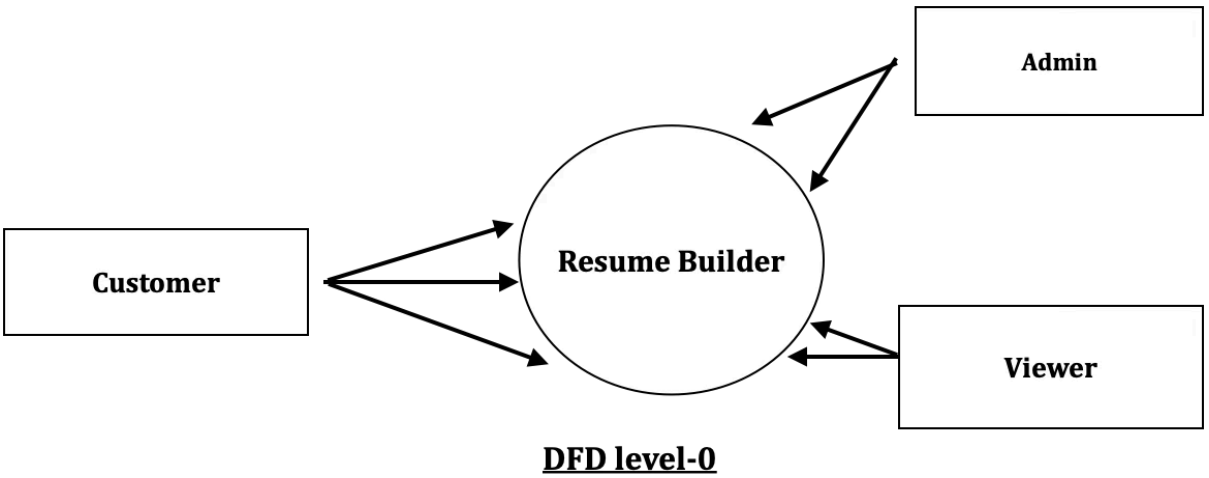
Architectural design represents the structure of data and program components that are required to build a computer-based system. It considers the architectural style that the system will take, the structure and properties of the components that constitute the system, and the interrelationships that occur among all architectural components of a system.



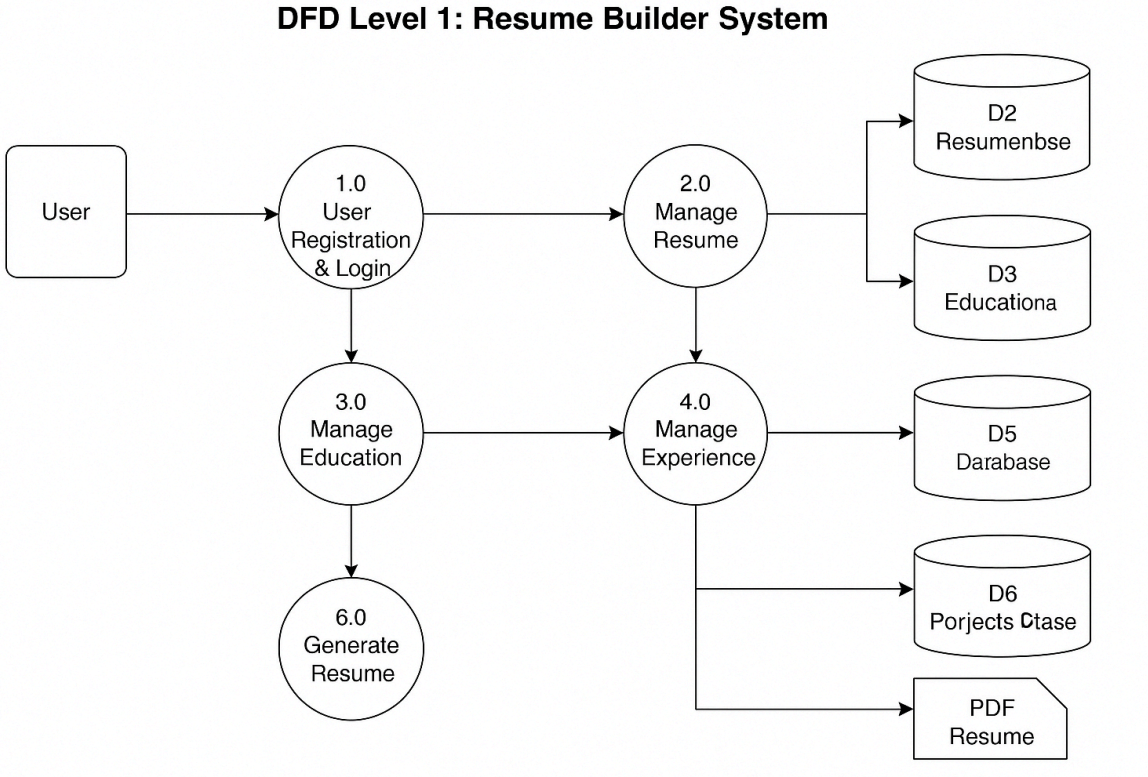
1.3 Architecture Diagram

Data Flow Diagram

The data flow diagram enables the software engineer to develop models of the information domain and functional domain at the same time. As the DFD is refined into greater levels of detail, the analyst performs an implicit functional decomposition of the system. At the same time, the DFD refinement results in corresponding refinement of data as it moves through the processes that embody the application.



1.4 DFD level 0

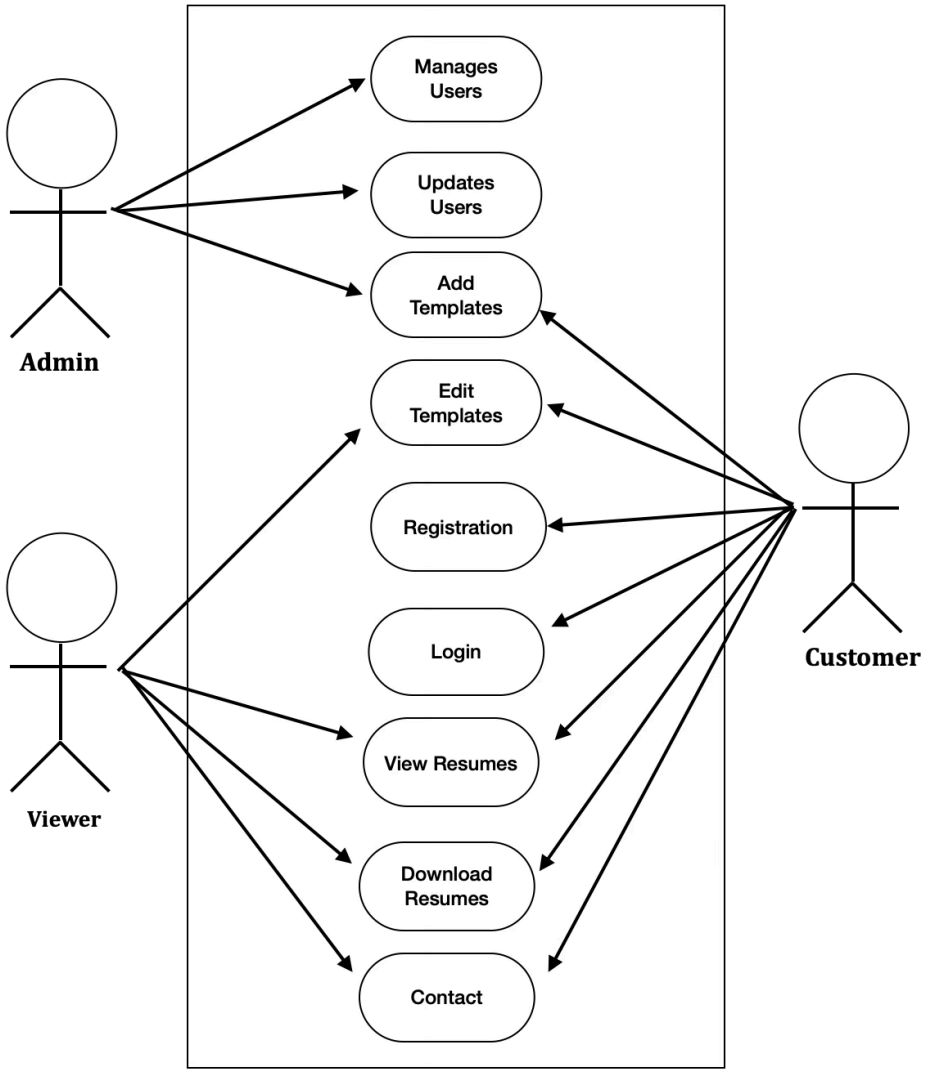


1.5 DFD level-1

UseCase Diagram

Use Case Model is an approach that is a combination of text and pictures in order to improve the understanding of requirements. A use case model describes the complete functionality of a system by identifying how everything that is outside the system interacts with it. A Use Case Diagram is given below that relates to this application.

This project is a web application that manages a system of building resumes online. Actors It has 3 actors.
1) Administrator
2) Customer
3) Viewer

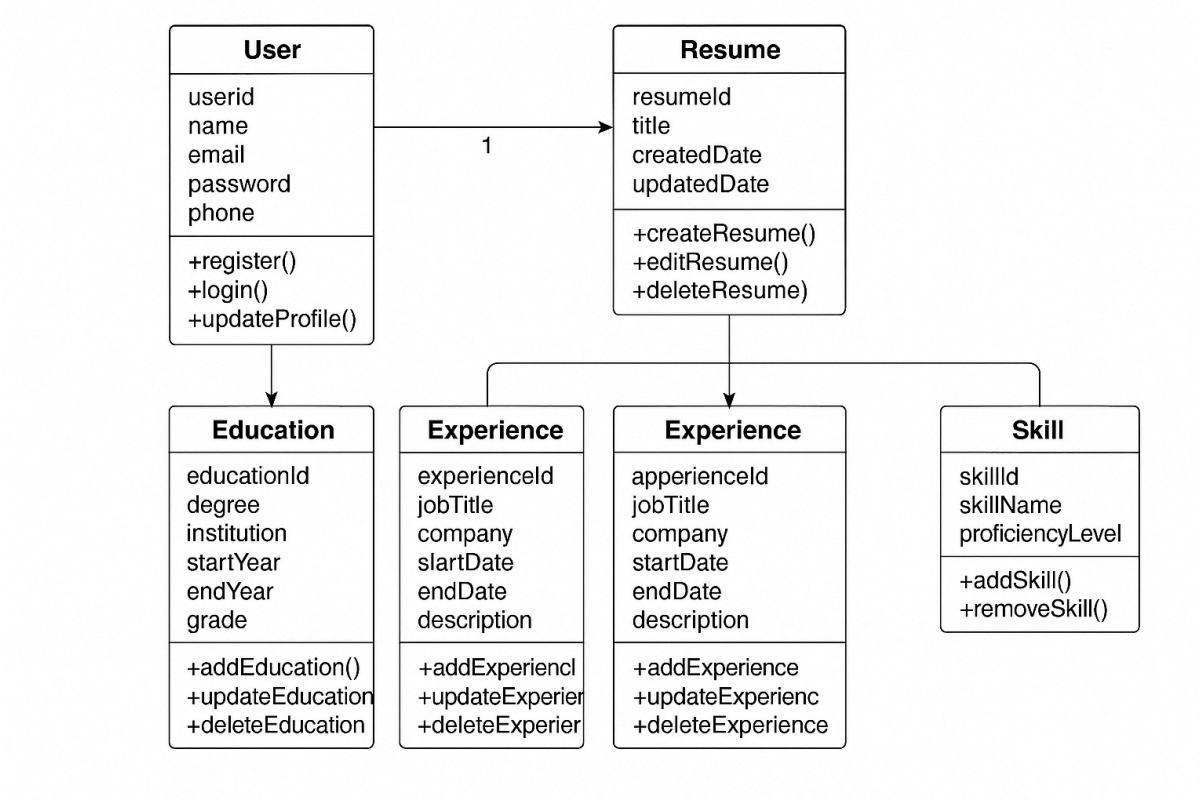


UseCase Diagram

1.6 UseCase Diagram

Class Diagram

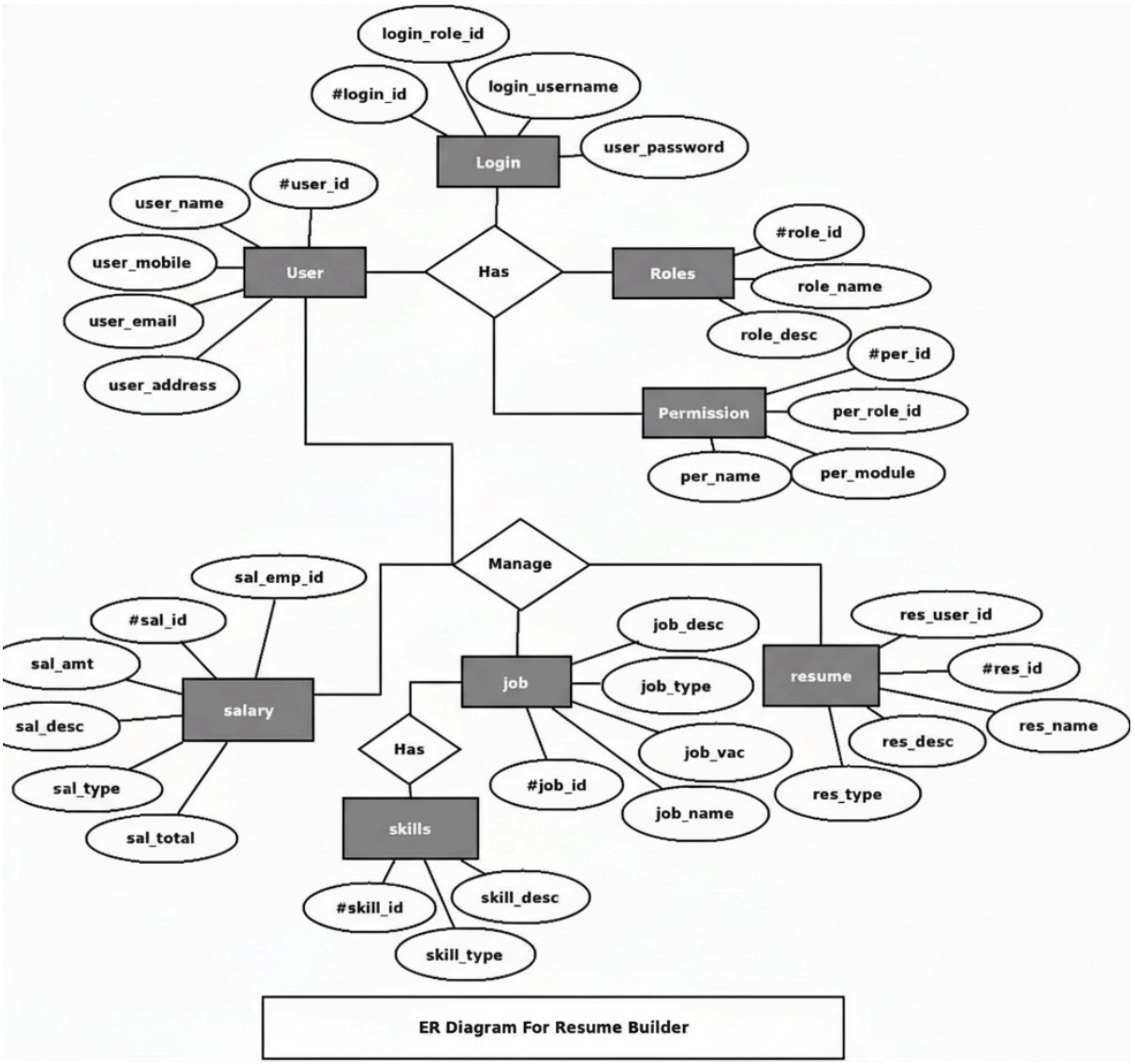
A class diagram shows relationships among classes used in the system. Every class has some contents, attributes and objects associated with it. Every class produces some outputs and uses some inputs. Outputs generated are consumed by other classes and inputs consumed are generated by other classes. The class diagram shows which class is producing what and consuming what.



1.7 Class Diagram

ER Diagram

The object/relationship pair is the cornerstone of the data model. These pairs are represented graphically using E-R diagrams. A set of primary components are identified for the ERD: data objects, attributes, relationships and various type indicators. The primary purpose of ERD is to represent data objects and their relationships.



1.8 ER Diagram

CHAPTER 3

SYSTEM DESIGN

System Design

System design is a solution, a way to approach the creation of a new system. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Emphasis is on translating the performance requirements into design specifications. Design goes through logical and physical stages of development. Logical design reviews the present physical system; prepared input and output specifications; details the implementation plan; and prepares a logical design walkthrough. The physical design maps out the details of the physical system, plans the system implementation, devises a test and implementation plan, and specifies any new hardware and software.

System Architecture

The application employs a three-tier architecture, which is standard for MERN applications:

1. **Client Tier (Presentation Layer):** A React-based Single Page Application (SPA) that runs in the user's browser. It handles the UI, manages frontend state, and communicates with the server via RESTful API calls.
2. **Application Tier (Logic Layer):** A Node.js and Express.js server that acts as the backend. It contains the business logic, handles API requests from the client, processes data, manages user authentication, and interacts with the database.
3. **Data Tier (Database Layer):** A MongoDB database that stores all application data, including user credentials and resume information, in a flexible, JSON-like BSON format.

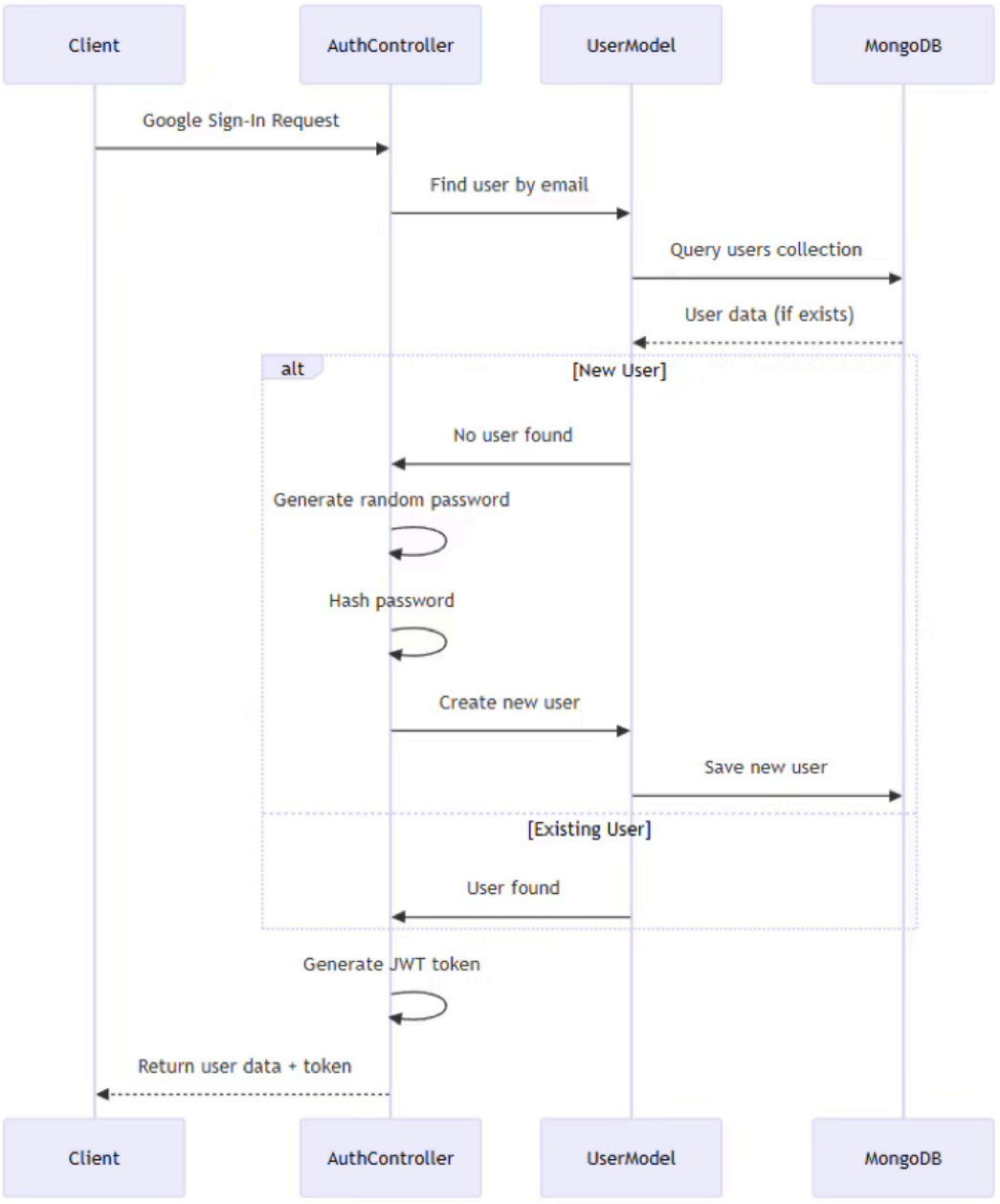
Database Design and Workflow

1. Request Flow

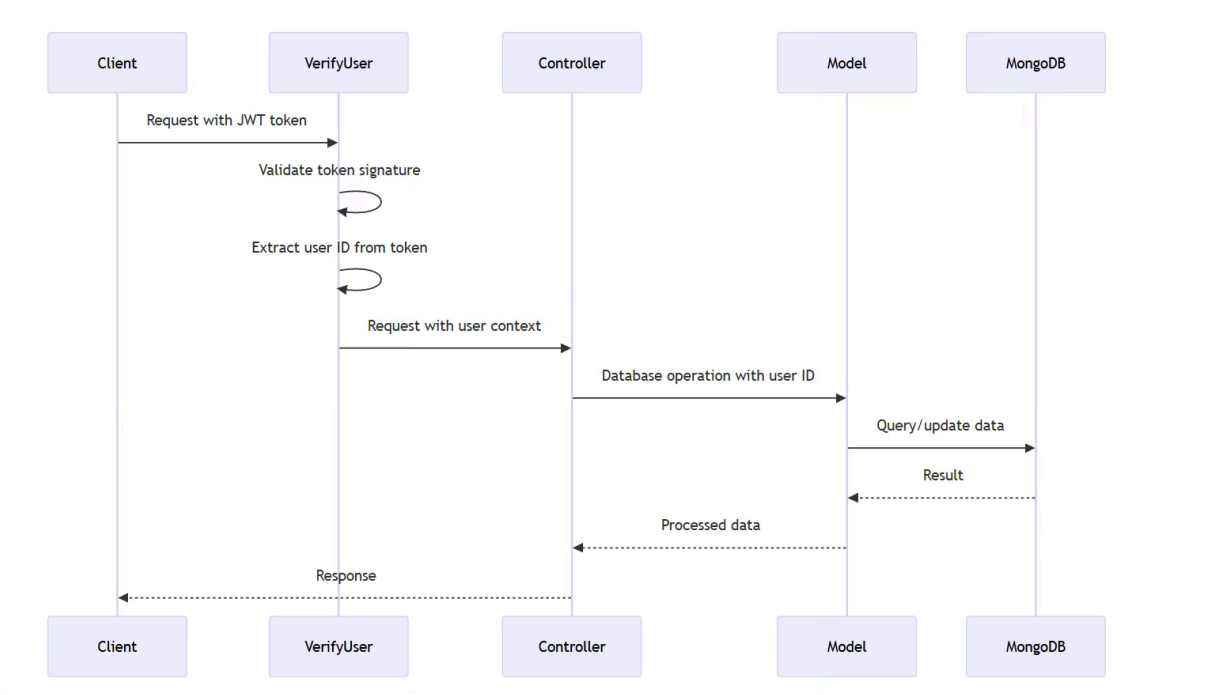
1.Client makes HTTP requests to the Express server
2.Requests pass through middleware stack:

CORS handling
JSON parsing
Authentication (for protected routes)
Logging

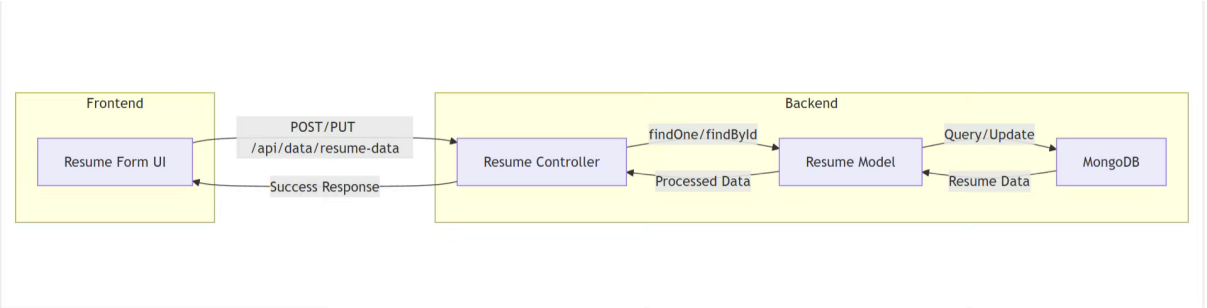
2.Authentication Process



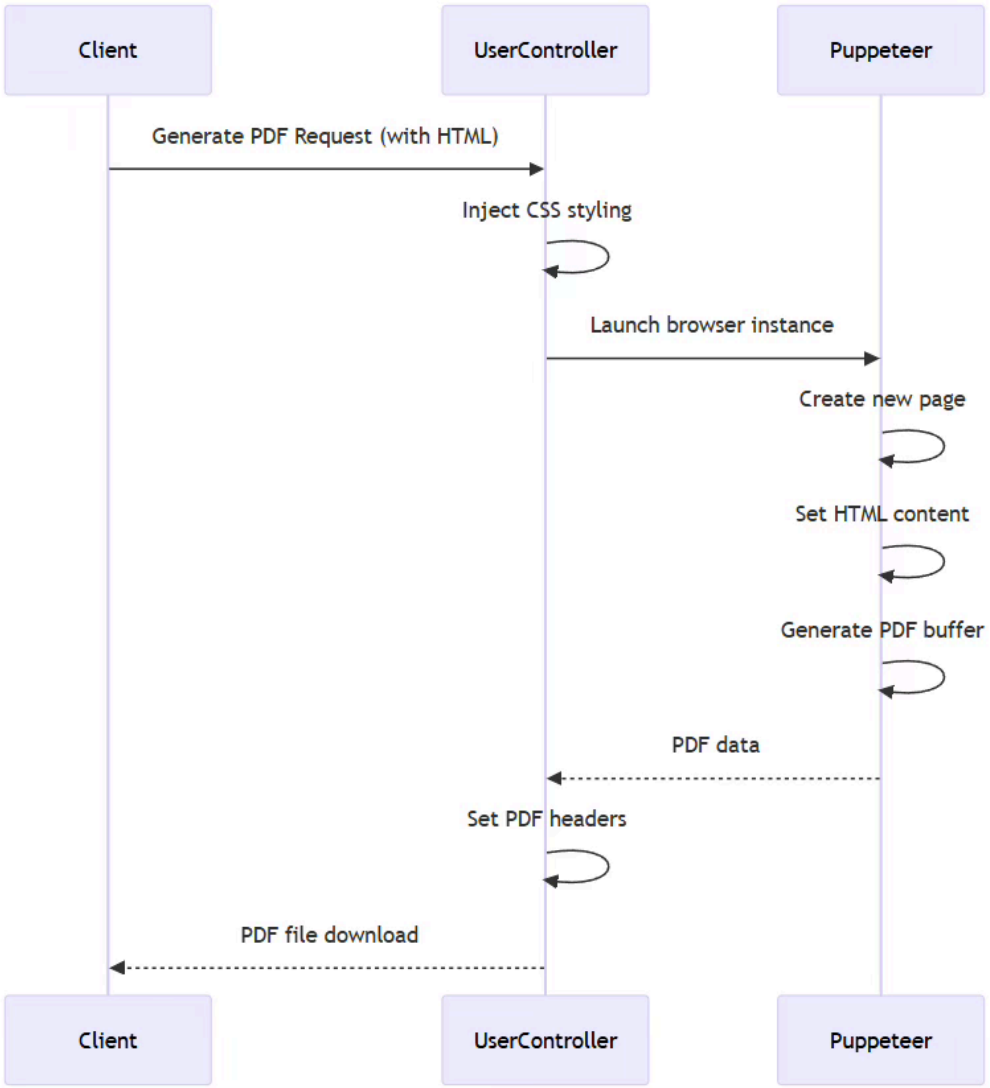
3.Protected Route Access



4.Resume Data Management



5.PDF Generation Process



Key Components Interaction

1.Database Models
- **User Model:** Stores user credentials and profile information
- Resume Model: Stores all resume data linked to users via userId

2.Security Layers
- **JWT Authentication:** Token-based authentication for all protected routes
- **Password Hashing:** BCrypt for secure password storage
- **Route Protection:** Middleware verifies tokens before allowing access

3.File Generation
- **Puppeteer:** Headless Chrome browser for converting HTML to PDF
- **CSS Injection:** Dynamic styling applied to resume templates.

Testing

A multi-level testing strategy was employed to ensure the application's quality and reliability.

- **Unit Testing:** Individual components (especially helper functions and API utility functions) were tested in isolation using frameworks like Jest.
- **Integration Testing:** The interaction between the frontend and backend was tested. API endpoints were tested using tools like Postman to verify that data was being correctly sent, received, and processed.
- **System Testing:** End-to-end testing was performed manually from a user's perspective. This involved testing the entire application workflow, from user registration to downloading a PDF, to ensure all modules work together as expected. Usability testing was also conducted to gather feedback on the user interface.

CHAPTER 4

CODING

This section details the technical implementation of the MERN-stack resume builder. The project is logically separated into two main parts: the frontend (client-side) and the backend (server-side).

Backend: Node.js, Express, and MongoDB

The backend is built with Node.js and the Express framework. It handles all business logic, database interactions, and user authentication.

Backend Folder Structure:

server/

```
├── config/
│   ├── User.js
│   └── db.js
├── controllers/
│   ├── auth.Controller.js
│   └── resumes.Controller.js
│   └── user.Controller.js
├── models/
│   ├── resume.models.js
│   └── user.model.js
├── routes/
│   └── auth.routes.js
│   └── resume.routes.js
│   └── user.routes.js
├── utils/
│   ├── authHelper.js
│   └── verify.js
```

```
└── index.js
```

Frontend: React.js

The frontend is a single-page application (SPA) built using React. It provides a dynamic and responsive user interface for creating and previewing resumes. It is designed to be state-driven and modular.

```
frontend/
│
├── public/
│   ├── index.html
│   ├── favicon.ico
│   └── manifest.json
│
├── src/
│   ├── api/
│   │   └── axiosClient.js         # Axios base setup for API calls
│   │
│   ├── app/
│   │   └── store.js               # Redux store configuration
│   │
│   ├── assets/
│   │   ├── logo.png
│   │   ├── icons/
│   │   └── images/
│   │
│   ├── components/
│   │   ├── Navbar.jsx             # Top navigation bar
│   │   ├── Footer.jsx             # Footer section
│   │   ├── ResumeEditor.jsx       # Dynamic resume form editor
│   │   ├── ResumePreview.jsx      # Preview resume before download
│   │   ├── TemplateSelector.jsx   # Choose resume design/template
│   │   ├── SectionInput.jsx       # Handles dynamic sections (project, education)
```

```
│   │   └── LoadingSpinner.jsx      # Loader component
│   │
│   ├── features/
│   │   ├── auth/
│   │   │   ├── authSlice.js        # Redux slice for login/logout
│   │   │   └── firebaseConfig.js   # Firebase initialization
│   │   │
│   │   └── resume/
│   │       ├── resumeSlice.js      # Redux slice for resume data
│   │       └── resumeThunks.js     # Async operations (fetch, save, delete)
│   │
│   ├── hooks/
│   │   └── useAuth.js              # Custom hook for Firebase user auth state
│   │
│   ├── layouts/
│   │   ├── MainLayout.jsx          # Layout with Navbar + Footer
│   │   └── AuthLayout.jsx          # Layout for login/register pages
│   │
│   ├── pages/
│   │   ├── Home.jsx                # Landing page
│   │   ├── Login.jsx               # Firebase login
│   │   ├── Register.jsx            # Firebase signup
│   │   ├── Dashboard.jsx           # User dashboard listing resumes
│   │   ├── CreateResume.jsx        # Resume creation screen
│   │   ├── EditResume.jsx          # Edit existing resume
│   │   ├── ViewResume.jsx          # Read-only view of resume
│   │   └── NotFound.jsx            # 404 page
│   │
│   ├── routes/
│   │   └── AppRouter.jsx           # All app routes using React Router DOM
│   │
│   ├── styles/
│   │   ├── theme.js                # MUI custom theme colors & typography
│   │   └── global.css              # Custom global CSS styles
```

```
│   │
│   ├── utils/
│   │   ├── formatDate.js          # Helper for displaying dates
│   │   ├── validators.js          # Form validation utilities
│   │   └── constants.js           # Reusable constants (API URLs, etc.)
│   │
│   ├── App.jsx                    # Root component (with Router + Redux Provider)
│   ├── main.jsx                   # App entry point
│   └── index.css                  # Global style imports
│
├── .env                           # API URLs, Firebase keys
├── package.json
├── vite.config.js / webpack.config.js
└── README.md
```

Database

```
// server/config/db.js
const mongoose = require("mongoose");
const mongoDB = async () => {
  try {
    // await mongoose.connect(process.env.MONGO_URL);
    const conn = await mongoose.connect(process.env.MONGO_URL, {
      useNewUrlParser: true,
      useUnifiedTopology: true,
      // Add these options to prevent connection issues
      serverSelectionTimeoutMS: 5000, // Timeout after 5s instead of 30s
      socketTimeoutMS: 45000, // Close sockets after 45s of inactivity
    });
    console.log("Connected to database");
    console.log("MongoDB Connected successfully");
    console.log(`Host: ${conn.connection.host}`);
    console.log(`Database: ${conn.connection.name}`);

    // List all collections to verify
    const collections = await conn.connection.db.listCollections().toArray();
    console.log("Available collections:", collections.map(c => c.name));

    return conn;
  } catch (error) {
    console.log(error.message);
    process.exit(1); // Exit process with failure
  }
};
module.exports = mongoDB
```

CHAPTER 5

STANDARDIZATION OF THE CODE

1. Code Efficiency & Error Handling

- The code has been developed following **modular programming principles**, where each module (component, slice, or service) performs a specific task.
- **Error boundaries** and try-catch blocks have been used in both frontend (React components) and backend (API routes) to prevent application crashes.
- API requests are handled using centralized Axios interceptors, ensuring **consistent error messages** and **automatic token injection** for authentication.
- Lazy loading and React memoization are used to enhance **performance** and **rendering speed**.

2. Parameter Calling and Passing

- Functions and React components use **props and hooks** for parameter passing, ensuring reusability and clear data flow.
- Redux Toolkit manages global state, reducing the need for excessive prop drilling.
- All API functions follow standardized parameter structures for uniformity

3. Validation Checks

- **Form validation** is applied using both client-side (React validation + MUI form control) and server-side (Express.js validation middleware).
- Before submitting data (like education, projects, or contact info), validation ensures no empty or invalid fields are allowed.
- Real-time error prompts and disabled buttons prevent users from submitting incomplete forms.
- Firebase Authentication adds an additional validation layer for login and registration.

4. Coding Style and Naming Conventions

- **CamelCase** is used for variables and functions .
- **PascalCase** is used for React components.
- Constants and environment variables use **UPPERCASE_WITH_UNDERSCORES**.
- Each file begins with a short **comment block** describing its purpose.
- Consistent **indentation (2 spaces)** and **ESLint/Prettier** tools are applied for automatic code formatting.

5. Reusability and Maintainability

- Common UI elements (buttons, text fields, modals) are placed inside a reusable **`/components`** folder.
- Helper functions like date formatting, string manipulation, and validation are centralized in the **`/utils`** directory.
- The codebase follows a **clear separation of concerns** — UI (React), state management (Redux), and backend communication (API services).

- Redux slices and Firebase setup are independent, making it easy to add new modules in the future.

6. Security and Data Handling

- Sensitive credentials (API keys, Firebase configs) are stored in `.env` files and never hard-coded.
- JWT tokens are securely managed in HTTP-only cookies or local storage.
- Input sanitization ensures protection against XSS and SQL injection attacks.

CHAPTER 6
TESTING

Testing Techniques and Strategies Used

To ensure the reliability and correctness of the Resume Builder application, both manual and automated testing techniques were applied:

- **Unit Testing –** Conducted on individual components such as form validation, login, and resume preview modules using Jest and React Testing Library.
- **Integration Testing –** Verified the interaction between frontend (React + Redux) and backend API (Node.js/Express) for seamless data flow.
- **System Testing –** Tested the complete application for functionality, performance, and compatibility across different browsers and devices.
- **User Acceptance Testing (UAT) –** Feedback was collected from test users to validate usability and user experience.

Testing Plan

Phase	Module/Feature Tested	Testing Type	Status
Phase 1	User Registration & Login	Unit + Integration	Passed
Phase 2	Resume Creation Form	Unit + Validation	Passed
Phase 3	Dynamic Section Handling (Projects, Education)	Integration	Passed
Phase 4	Resume Preview and Export (PDF)	System Testing	Minor formatting fixes
Phase 5	Overall UI Responsiveness	UAT + System Testing	Passed

Unit Test Cases

Test Scenario	Expected Result	Actual Result	Status
User logs in with valid credentials	Dashboard loads successfully	As expected	Pass
User submits empty form	“Required field” error shown	As expected	Pass
Add new education section	New entry appears dynamically	As expected	Pass
Resume download in PDF format	File exported correctly	Formatting mismatch fixed	Pass
Invalid email during registration	Validation error displayed	As expected	Pass

Debugging and Code Improvement

During testing, several issues were identified and resolved:

- PDF export formatting inconsistencies were fixed using a unified layout engine.
- Redux state synchronization issues were debugged to prevent stale data rendering.
- Authentication failures in Firebase were handled using async/await and improved error messages.
- UI rendering delays were minimized by optimizing component re-renders and applying lazy loading.

System Test Reports

- All modules were tested in Google Chrome, Edge, and Firefox.
- Tested for mobile responsiveness using Chrome DevTools (Pixel and iPhone simulators).
- End-to-end testing verified smooth navigation from login → resume creation → preview → PDF download.

CHAPTER 7

SYSTEM SECURITY MEASURES

Database / Data Security

- The database (MongoDB Atlas) is hosted on a secure cloud environment with encrypted connections (TLS/SSL) to prevent data interception.
- Authentication tokens (JWT) are used to verify every API request between the client and server.
- Sensitive credentials such as database URIs, API keys, and Firebase configuration details are stored securely in environment variables (.env) instead of being hardcoded.
- Input sanitization and validation are implemented to protect against SQL injection, XSS, and data tampering.
- Regular database backups ensure data recovery in case of accidental loss or system failure.

Creation of User Profiles and Access Rights

- Each user has a unique account created using Firebase Authentication, which securely manages login, logout, and password reset functions.
- The system follows role-based access control (RBAC) — users can only access their own profiles and resumes.
- Authorization checks are performed before accessing or modifying any user-specific data.
- Passwords are not stored in plain text; instead, they are securely handled through Firebase's encrypted authentication system.
- Inactive sessions are automatically logged out to prevent unauthorized access.
- The frontend and backend use token-based session validation, ensuring that only authenticated users can perform operations like editing or downloading resumes.

Application-Level Security

- HTTPS protocol is enforced during data transmission to prevent man-in-the-middle (MITM) attacks.
- CORS (Cross-Origin Resource Sharing) policies are properly configured to restrict API access to trusted domains only.
- Implemented error logging and monitoring to detect unauthorized activities or suspicious behavior.
- All uploaded or entered data is validated and sanitized to maintain data integrity and system reliability.

CHAPTER 8

COST ESTIMATION

The Resume Builder project was developed using modern web technologies like React.js, Node.js, Redux Toolkit, Firebase, and MongoDB Atlas.

Although it was developed in an academic environment, estimating the project cost helps in understanding the real-world software development budget, covering resources, tools, human effort, and infrastructure.

Cost Estimation Model Used – Basic COCOMO Model

The **Constructive Cost Model (COCOMO)** was used for estimating the project cost.
It provides a mathematical model based on the size of the project (in KLOC – Kilo Lines of Code).

The **Basic COCOMO formula** is:

Effort (E)=a×(KLOC)b

Development Time (D)=c×(Effort)d

Where constants for **Organic Type Projects** (like Resume Builder) are:

- a = 2.4
- b = 1.05
- c = 2.5
- d = 0.38

Effort and Duration Estimation

E=2.4×(4.5)1.05=11.3 Person-Months

D=2.5×(11.3)0.38=5.3 Months

Cost Calculation

Assuming 1 person works full-time for 5 months:

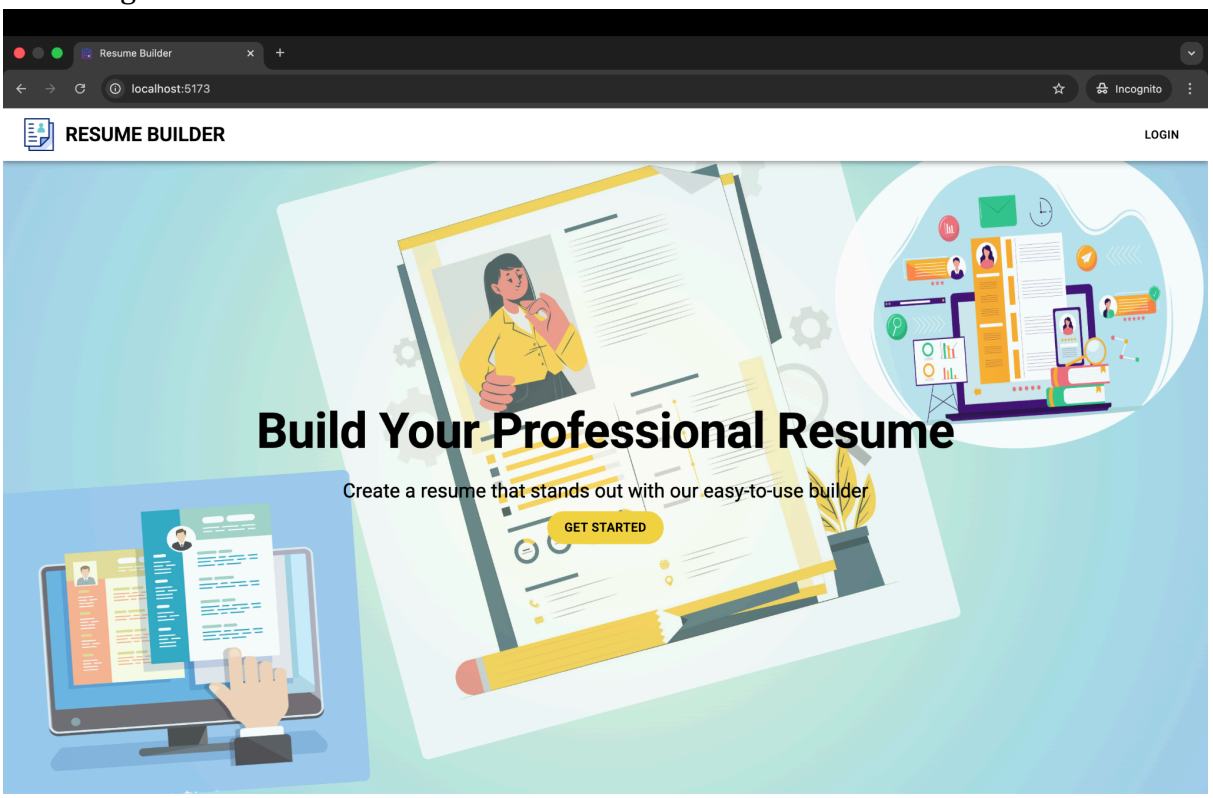
Total Cost=5 months×160 hours/month×₹500=₹4,00,000

Thus, the **estimated project cost ≈ ₹4,00,000** for development, testing, and deployment.

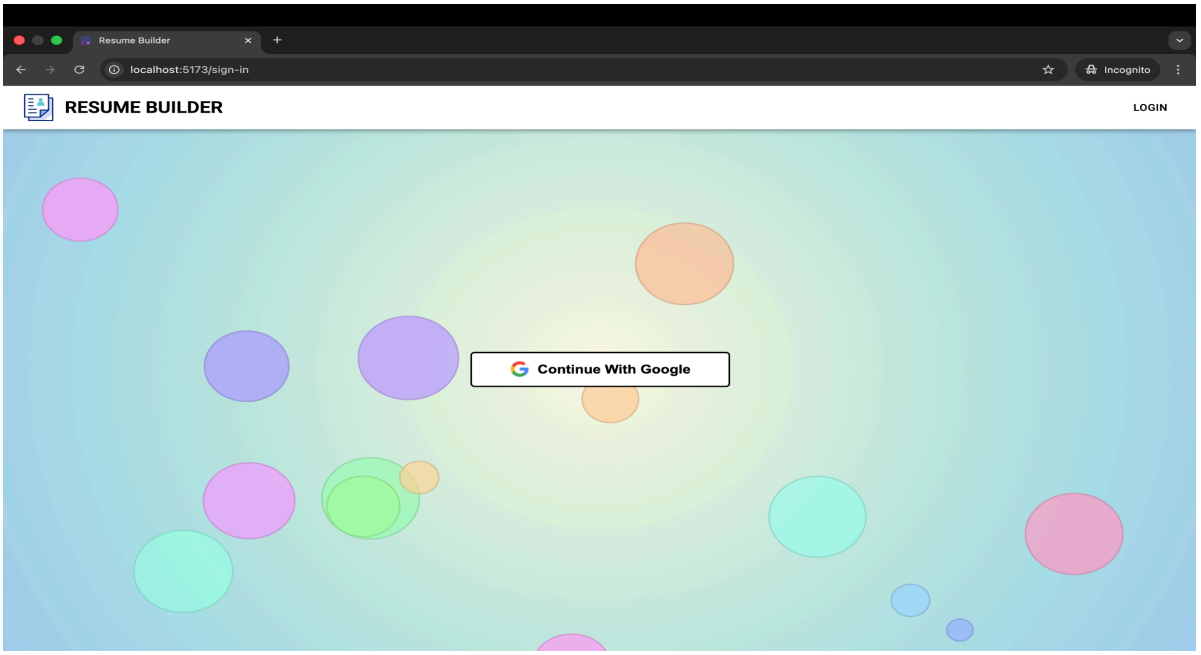
CHAPTER 9

SNAPSHOTS

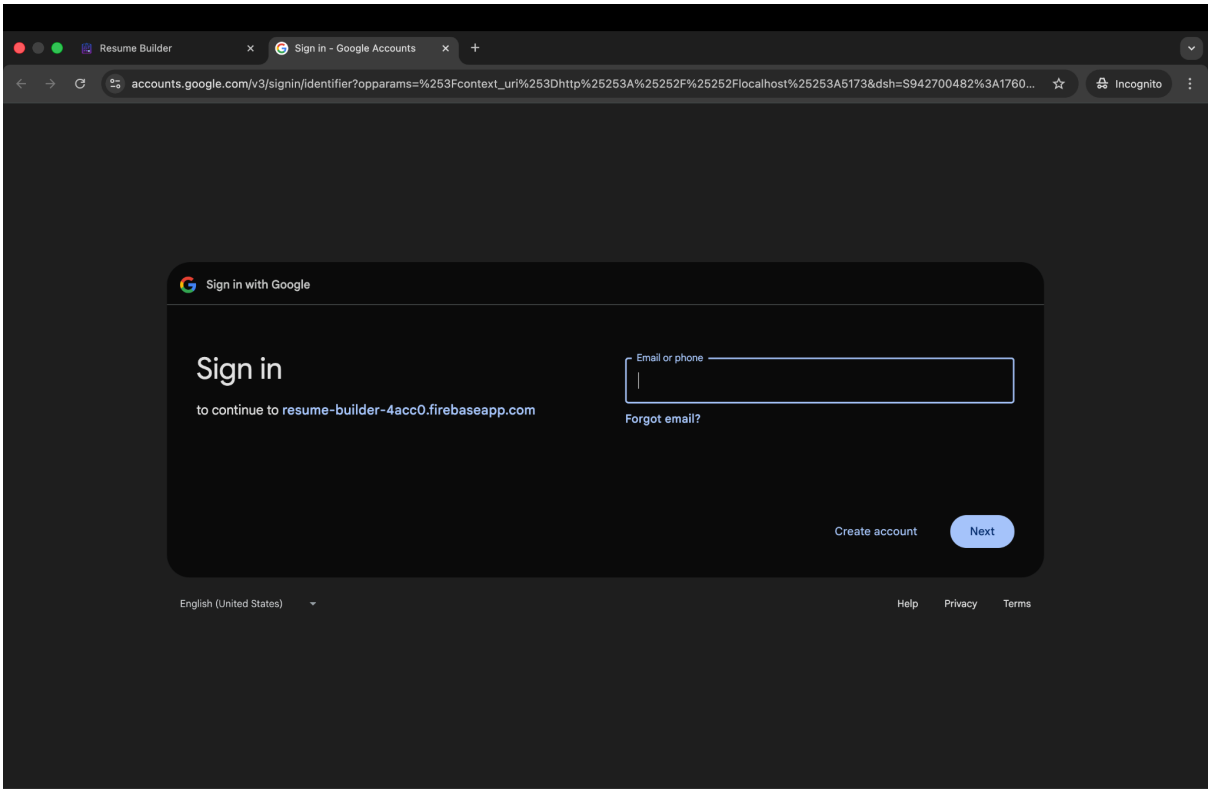
Home Page



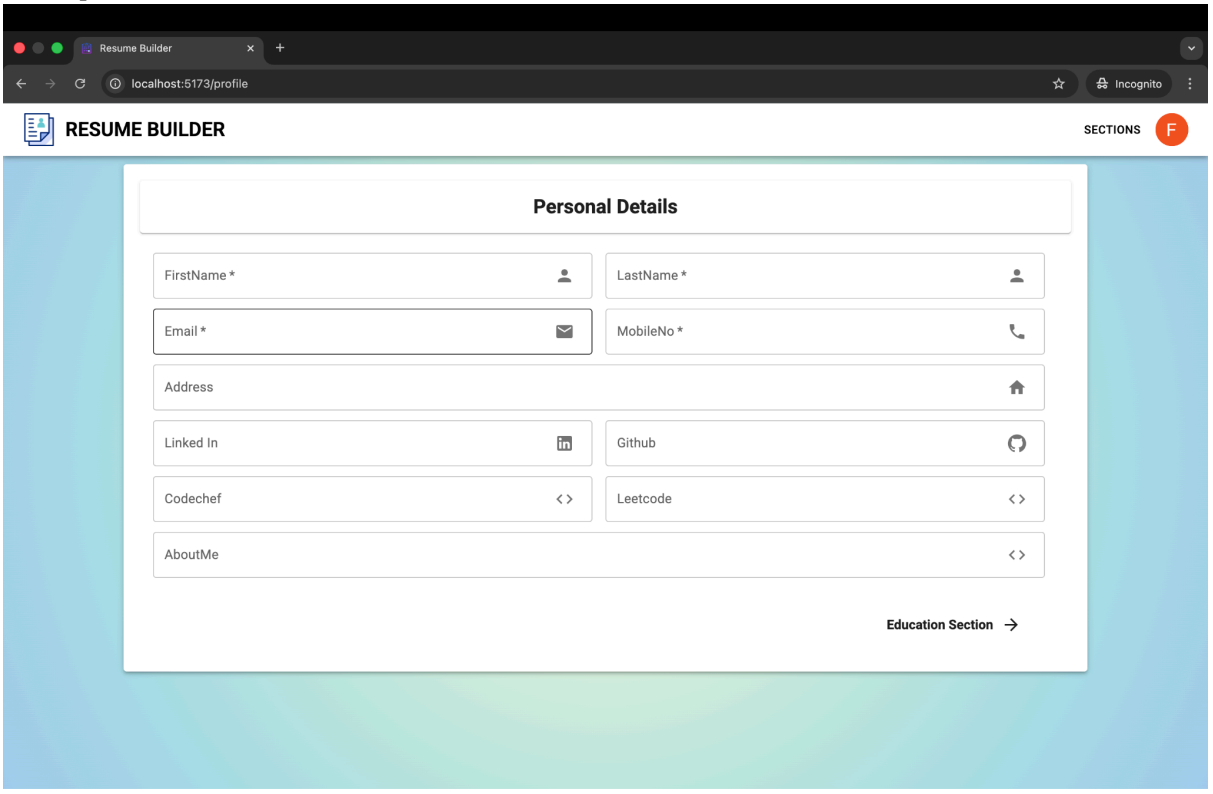
Login



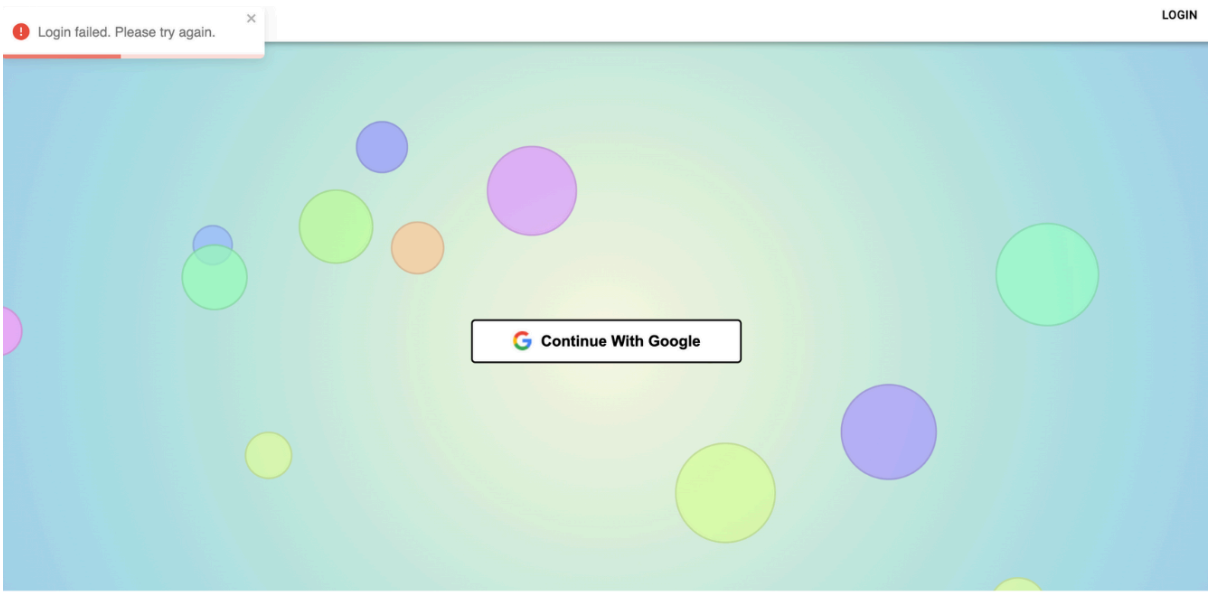
Sign in with Google ID

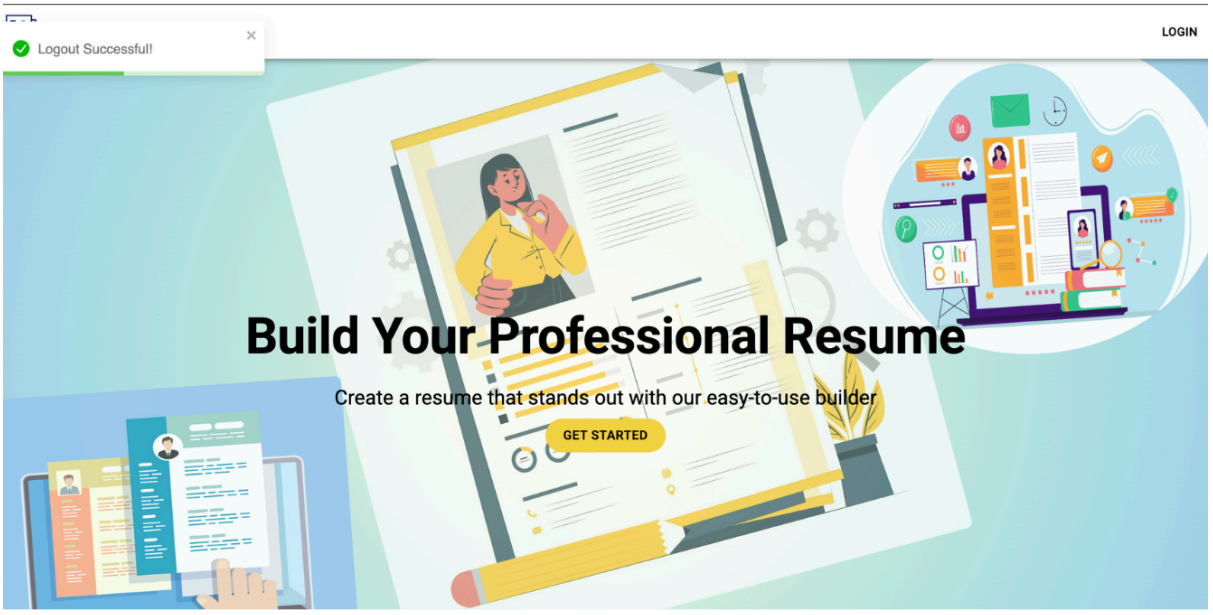


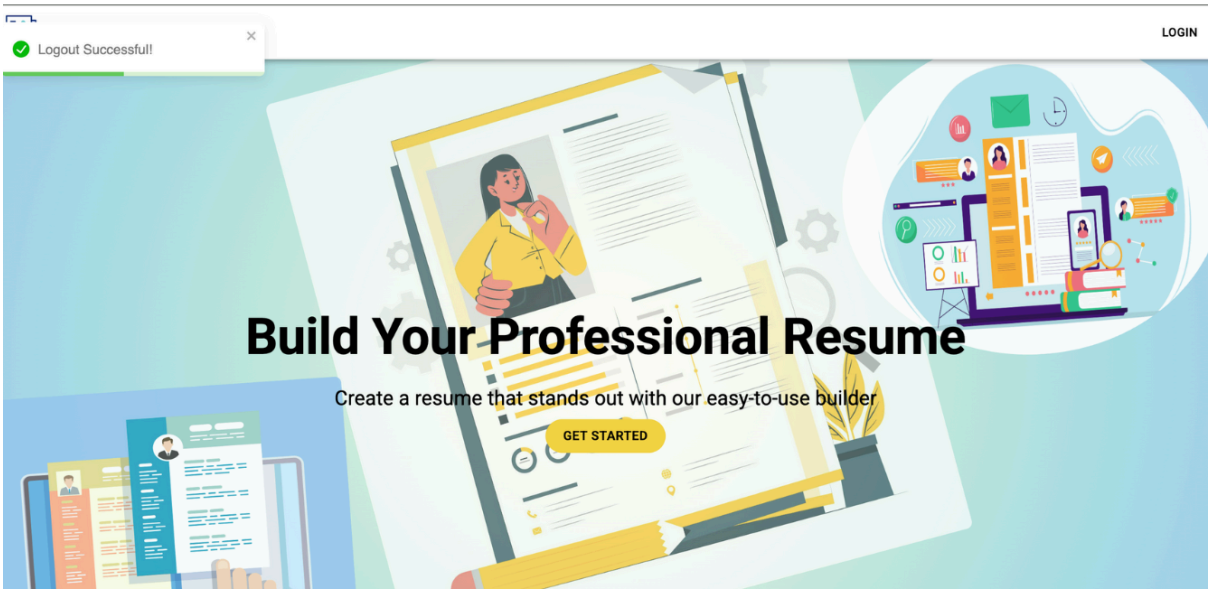
Form preview



Login Authentication


Login failed. Please try again.
LOGIN
Continue With Google


Logout Successful!
LOGIN
Build Your Professional Resume
Create a resume that stands out with our easy-to-use builder
GET STARTED


Logout Successful!
LOGIN
Build Your Professional Resume
Create a resume that stands out with our easy-to-use builder
GET STARTED

CHAPTER 10

CONCLUSION AND FUTURE SCOPE

Conclusion

The "Resume Builder" project successfully achieves its goal of providing freshers with a simple, effective, and modern tool for creating professional resumes. By leveraging the MERN stack, the application delivers a responsive, scalable, and feature-rich user experience. It effectively addresses the common pain points faced by fresh graduates, empowering them to craft compelling resumes that stand out to potential employers. The project demonstrates a practical application of modern web development technologies to solve a real-world problem.

Future Scope

While the current application is fully functional, there are several avenues for future enhancement:

- Enhanced AI Features: Integrate a more advanced NLP model to provide contextual suggestions, check for grammatical errors, and score the resume against specific job descriptions.
- Cover Letter Generator: Add a new module for creating cover letters that match the selected resume template and user data.
- Job Application Tracker: Implement a feature allowing users to track the jobs they have applied for using the resumes created on the platform.
- Expanded Template Library: Continuously add new and diverse templates to cater to different industries and roles.
- Sharing and Analytics: Allow users to generate a shareable link to their resume and provide basic analytics, such as the number of views.

BIBLIOGRAPHY

1. Sommerville, Ian. *Software Engineering*, 10th Edition, Pearson Education.
2. Pressman, R. S. *Software Engineering: A Practitioner's Approach*, McGraw-Hill.
3. Mozilla Developer Network (MDN) — https://developer.mozilla.org
4. ReactJS Documentation — https://react.dev/learn
5. MongoDB Documentation — https://www.mongodb.com/docs
6. Node.js Official Docs — https://nodejs.org/en/docs
7. Material UI Documentation— https://mui.com/material-ui/getting-started/
8. Firebase Documentation — https://firebase.google.com/docs

GLOSSARY

1. **Agile Model:** A flexible software development methodology that delivers projects in iterative cycles called sprints, allowing continuous feedback and improvement.
2. **Frontend:** The user-facing part of the web application — developed using React.js for interactive and responsive interfaces.
3. **Backend:** The server-side logic and data processing part of the application — developed using Node.js and Express.js.
4. **Database:** A structured collection of data. This project uses **MongoDB** to store user and resume details.
5. **API (Application Programming Interface):** A set of endpoints that allow communication between the frontend and backend.
6. **UI/UX (User Interface / User Experience):** The design and overall interaction quality of the web application for end users.
7. **Version Control:** Tracking and managing changes in code using tools like **Git** and **GitHub**.
8. **Template:** A predefined layout used to generate resumes in various designs and styles.
9. **Deployment:** The process of hosting the application online (e.g., on Vercel, Render, or Netlify).
10. **Validation:** Checking that user inputs (like name, email, etc.) are correct and formatted properly.
11. **Testing:** The process of verifying that the software meets the required functionality and quality standards.
12. **Authentication:** The process of verifying a user's identity before granting access to the system.
13. **Authorization:** Determines what actions or data a verified user is allowed to access.
14. **Component:** A reusable UI block in React.js that represents a part of the interface.
15. **Middleware:** Functions in Node.js that handle requests and responses between the client and server.
16. **Repository:** A storage location for project files managed by version control systems like GitHub.
17. **Resume Parser:** A tool or algorithm that extracts information (like skills and experience) from uploaded resumes.