LAB PRACTICAL REPORT

# ON

BACKEND ENGINEERING 22CS008

# BACHELOR OF ENGINEERING

**in**

# COMPUTER SCIENCE AND ENGINEERING

****

### Submitted by: Supervised By:

**Asmita Sharma (2310990307) Mr. Venkatesh**

**Balreet Singh (2310090310) Mentor/Trainer**

**Bhavisha Ahuja (2310990312)**

**Dhruv Kumar (2310990325)**

### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**CHITKARA UNIVERSITY**

**CHANDIGARH-PATIALA NATIONAL HIGHWAY, RAJPURA, PUNJAB,**

# INDEX

|  |  |  |
| --- | --- | --- |
| Sr. No. | Topics | Page Number |
| 1) | Abstract | 1 |
| 2) | Introduction   * Background and Significance * Objectives * Features & Functionality * Technology Stack | 2-8 |
| 3) | Problem Definition and Requirements   * Problem Statement * Software and hardware requirements | 9 |
| 4) | Proposed Design and Methodology | 10-12 |
| 5) | Results | 13-22 |

1. **ABSTRACT**

In today’s fast-paced digital era, the job market has become increasingly competitive, and both job seekers and employers face numerous challenges in the recruitment process. Traditional recruitment methods, such as newspaper advertisements, walk-in interviews, or reliance on personal networks, are often time-consuming, inefficient, and unable to scale to meet the needs of a rapidly evolving workforce. Online job portals have therefore emerged as essential tools, serving as centralized platforms where employers can post vacancies and candidates can discover and apply for opportunities in a streamlined, structured manner.

With the proliferation of technology and the increasing digitization of services, there is a growing demand for platforms that not only connect job seekers and employers but also provide efficient, interactive, and user-friendly experiences. Modern job seekers expect portals to offer more than just a listing of job opportunities—they seek personalized features such as resume building, interview preparation, application tracking, and real-time notifications. Similarly, employers require tools to manage applications efficiently, communicate quickly with candidates, and access data-driven insights about the recruitment process.

The system is designed with **role-based functionality**, ensuring that job seekers and employers are presented with separate dashboards tailored to their specific needs. Job seekers can explore vacancies, apply for jobs, build professional resumes, and practice technical or aptitude assessments through the Interview Preparation module. Employers can post job opportunities, review applications, and take decisions such as acceptance or rejection, all of which are tracked and stored securely in MongoDB. The notifications system ensures real-time communication between parties, improving efficiency and user engagement.

This project presents the **development of a Job Portal Web Application using the MERN stack**— comprising **MongoDB** for database management, **Express.js** and **Node.js** for back-end services, and **React.js** for the front-end interface.

# INTRODUCTION

In today’s competitive world, finding the right job or the right candidate has become increasingly challenging for both job seekers and employers. Traditional methods of recruitment, such as newspaper advertisements and offline applications, are time-consuming, less efficient, and limited in reach. With the rapid growth of technology and the internet, **online job portals** have emerged as a powerful solution to bridge this gap by providing a centralized, user-friendly, and interactive platform.

A job portal acts as a digital marketplace where job seekers can create profiles, upload resumes, and apply for relevant opportunities, while recruiters can post job openings, search for candidates, and manage applications seamlessly. Unlike traditional systems, job portals offer advanced features such as keyword-based search, filtering, instant notifications, and secure authentication, making the hiring process faster and more transparent.

The proposed **Job Portal Web Application** is designed to simplify recruitment by bringing employers and job seekers to a common platform. It provides features such as user authentication, profile management, job posting, resume uploading, and application tracking. The system is built using modern web technologies to ensure scalability, security, and ease of use. By leveraging a structured backend and an intuitive frontend, the portal ensures that users can interact with the system efficiently without technical barriers.

This project not only reduces the time and effort required in the hiring process but also enhances the chances of successful employment by connecting the right talent with the right opportunities. Ultimately, the Job Portal aims to improve the overall recruitment experience, making it more reliable, accessible, and effective for both parties.

### Background and Significance

Recruitment has always been a crucial part of organizational growth. Traditionally, companies relied on offline methods such as newspaper advertisements, employment agencies, and walk-in interviews. While effective to an extent, these methods were often slow, costly, and limited in reach. With the rise of the internet, digital platforms started transforming how job seekers and employers connect. Online job portals like Naukri, Indeed, and LinkedIn have revolutionized the hiring process by offering faster, broader, and more efficient recruitment solutions.

The significance of the Job Portal Web Application lies in its ability to simplify and modernize the recruitment process by addressing the limitations of traditional methods and existing online platforms. For job seekers, it provides a centralized, user-friendly space to create profiles, upload resumes, and apply for relevant opportunities while saving time through features like search filters and application tracking. For employers, the system reduces recruitment costs and effort by enabling quick job postings, efficient candidate management, and streamlined shortlisting.

### Objectives

* + - * To provide a secure and user-friendly platform for job seekers and employers.
      * To enable job seekers to register, create profiles, upload resumes, and apply for jobs.
      * To allow recruiters to register, post job openings, and manage candidate applications.
      * To implement search and filter options for quick access to relevant job opportunities.
      * To ensure secure authentication and protect user data with encryption methods.
      * To provide real-time notifications and application tracking for users.
      * To reduce the overall time and cost involved in the recruitment process.
      * To design a responsive, scalable, and interactive web application accessible across devices.

### Features and Functionality

To achieve these objectives, Job portal incorporates the following features:

#### General Features

* + **Landing Page:** Entry point of the application offering two roles — *User* or *Employer*.
  + **Role-Based Login:** Secure login system distinguishing between job seekers (users) and employers.
  + **MongoDB Integration:** All user, employer, job, and application data is stored and managed in the database.

#### Employer Features

* + **Employer Dashboard:** Personalized dashboard after login with easy navigation.
  + **Post a Job:** Employers can create job postings with details such as:
    - Job Title
    - Job Description
    - Company Name
    - Salary
    - Location
    - Job Type (Full-time, Part-time, Internship, etc.)

#### Manage Applicants

* + - View all applications submitted by users.
    - Accept or reject applications.
    - Store applicant responses in MongoDB.
    - Decisions are reflected in real-time on the user’s interface.
  + **Notifications:** Employers receive updates when users apply for jobs.

#### User (Job Seeker) Features

* + **User Dashboard:** Personalized dashboard after login showing available options.

#### Job Section:

* + - View jobs posted by employers.
    - Filter/search jobs based on role, location, or type.
    - Apply to jobs directly from the dashboard.

#### Resume Builder:

* + - Create a professional resume in a standard format.
    - Input personal, educational, and professional details.
    - Save and update resumes for future use.

#### Interview Preparation Module:

* + - Practice questions in key subjects like DSA, Aptitude, etc.
    - Attempt assessments to evaluate readiness.

#### Application Tracking:

* + - View the status of applied jobs.
    - Receive notifications if the application is **accepted** or **rejected** by employers.

#### System Functionalities

* + **Database Management:**
    - MongoDB stores and manages user profiles, employer profiles, job postings, and applications.

#### Notification System:

* + - Real-time updates on application status sent to users.

#### Scalable Architecture:

* + - Built using Node.js and Express for handling multiple user/employer requests.

#### Responsive Design:

* + - EJS templates and frontend ensure a clean, accessible interface across devices.

#### Security:

* + - Role-based access and data protection for users and employers.

### Technology Stack

The Job Portal Web Application is developed using the MERN stack — MongoDB, Express.js, React.js, and Node.js. The technologies used are as follows:

#### Frontend

* + React.js: For building dynamic and responsive user interfaces with reusable components.
  + HTML5 & CSS3: For structuring and styling the web pages.
  + JavaScript (ES6+): Provides interactivity and client-side logic.
  + Bootstrap / Material-UI: For responsive and mobile-friendly design.

#### Backend

* + Node.js: Server-side runtime environment that executes JavaScript code outside the browser.
  + Express.js: Web application framework for building RESTful APIs, handling routes, middleware, and server logic.

#### Database

* + MongoDB: NoSQL database used to store structured documents like user details, employer profiles, job postings, resumes, and applications.
  + Mongoose (ODM): Provides schema modeling and simplifies database queries and interactions with MongoDB.

#### Authentication & Security

* + bcrypt.js: For hashing and securing user and employer passwords.
  + JWT (JSON Web Tokens): For handling sessions and role-based authentication.

#### Development & Deployment Tools

* + Git & GitHub: For version control and collaborative development.
  + Nodemon: For automatically restarting the server during development.
  + Postman: For API testing and debugging backend endpoints.
  + Visual Studio Code (VS Code): Main code editor.
  + Cloud Deployment (Optional): AWS, Render, or Heroku for hosting and scalability.

## Problem Definition and Requirements

### Problem Statement:

In today’s competitive job market, both job seekers and employers face significant challenges in the recruitment process. Job seekers often struggle to find relevant opportunities that match their skills, while employers find it time-consuming and costly to identify suitable candidates through traditional methods. Existing job portals, although popular, are often cluttered, lack user-friendly interfaces, and fail to provide interactive features such as resume building, interview preparation, and real-time application tracking. Additionally, many platforms do not ensure personalized experiences for different roles (job seekers and employers), leading to inefficiency and dissatisfaction.

There is a need for a centralized, secure, and role-based web application that streamlines the hiring process by offering tailored dashboards for both employers and job seekers. Such a system should allow employers to easily post jobs, manage applicants, and communicate hiring decisions, while enabling job seekers to build resumes, prepare for interviews, apply to jobs, and track application status seamlessly. By integrating these features into one platform and leveraging modern technologies like React.js, Node.js, Express.js, and MongoDB, the proposed system addresses the shortcomings of existing solutions and improves the overall recruitment experience.

### Software & Hardware Requirements:

Frontend: React.js (for building dynamic, responsive, and interactive user interfaces)

Backend: Node.js, Express.js (handling API requests, role-based authentication, job posting, and application management)

Database: MongoDB, Mongoose (storing user profiles, employer profiles, job postings, resumes, applications, and notifications)

Security: JWT, bcrypt.js (secure user authentication and password encryption) Development Tools:

* + Visual Studio Code (main code editor)
  + Git & GitHub (version control)
  + Postman (API testing and debugging)
  + Nodemon (automatic server restart during development) Hardware:
  + Server / Development Machine with minimum 8GB RAM and SSD storage
  + Stable internet connection for development, API testing, and deployment
  + Cloud deployment capability to handle multiple concurrent users and job postings efficiently

## Proposed Design / Methodology

#### Proposed Design / Methodology

The proposed Job Portal Web Application is designed to streamline the recruitment process for both job seekers and employers. The system uses a **MERN stack** architecture, enabling a responsive frontend, scalable backend, and secure database management. The methodology focuses on modular development, role-based access, and real-time interaction between users and employers.

#### System Architecture

* + **Frontend (React.js):** Provides dynamic and interactive user interfaces. Components like dashboards, job listings, resume builder, and interview prep are modular for scalability.
  + **Backend (Node.js + Express.js):** Handles API requests, authentication, job posting, application management, and notifications. RESTful APIs connect the frontend and backend.
  + **Database (MongoDB):** Stores structured documents including user profiles, employer profiles, job postings, applications, resumes, and notifications.

#### Role-Based Access

* + **User (Job Seeker):** Can create profiles, build resumes, apply for jobs, practice assessments, and track application status.
  + **Employer:** Can post jobs, view applicants, accept/reject applications, and manage job postings.
  + **Admin (Optional):** Manages users, employers, and system security to ensure smooth operation.

#### Methodology Steps

1. **Requirement Analysis:** Identify the needs of job seekers and employers and define system features.

#### System Design:

* + Create database schema with MongoDB collections for users, employers, jobs, applications, and notifications.
  + Design React components and routing for dashboards, forms, and job listings.

#### Implementation:

* + Develop frontend using React.js with responsive design.
  + Build backend APIs using Node.js and Express.js for CRUD operations.
  + Integrate MongoDB with Mongoose for data management.
  + Implement authentication and role-based authorization with JWT and bcrypt.js.

#### Testing:

* + Perform unit testing for frontend components and backend APIs.
  + Conduct integration testing to ensure seamless communication between frontend and backend.
  + Validate database operations and notification updates.

#### Deployment:

* + Host the application on a cloud service (AWS, Heroku, or Render).
  + Ensure scalability, performance optimization, and secure access.

#### Key Features Integrated in Design

* + Real-time notifications for job application status.
  + Resume builder for standardized professional resumes.
  + Interview preparation module with assessments (DSA, aptitude, etc.).
  + Employer dashboard with job posting and applicant management.

## Results

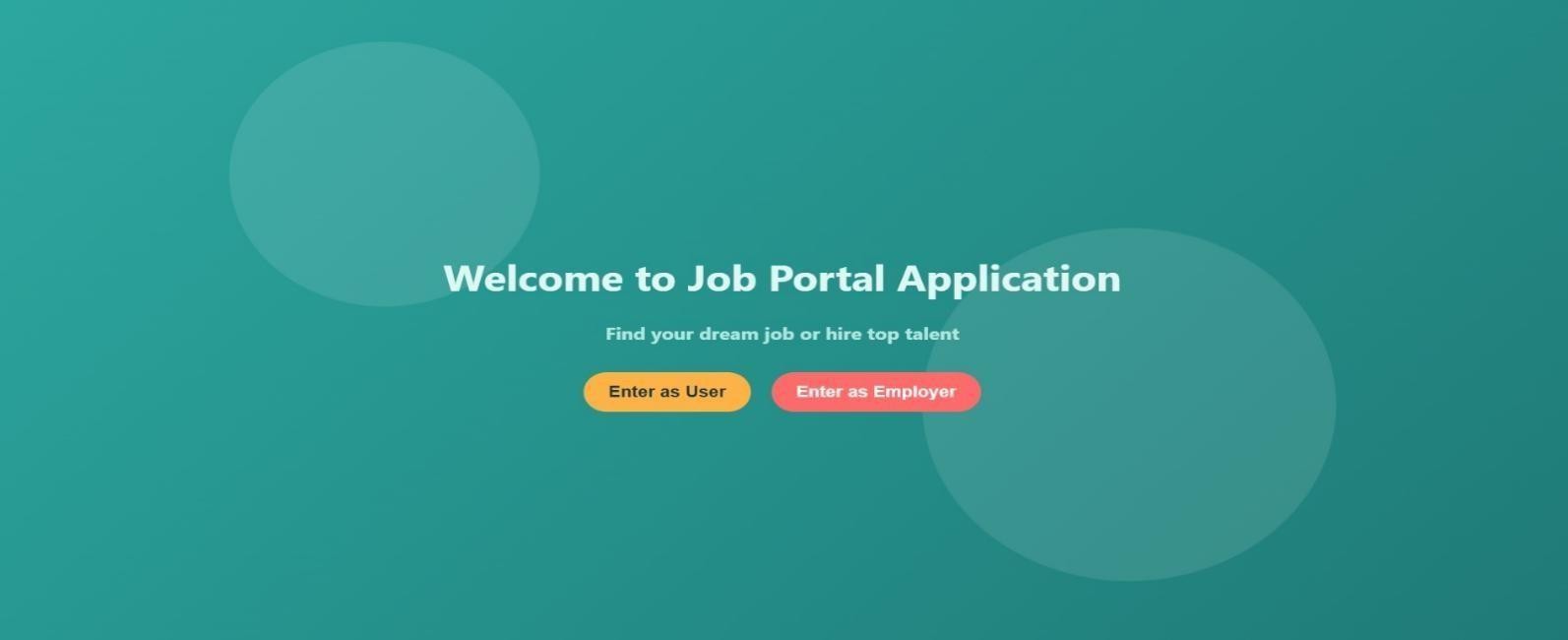
****

Fig 1. Home Page

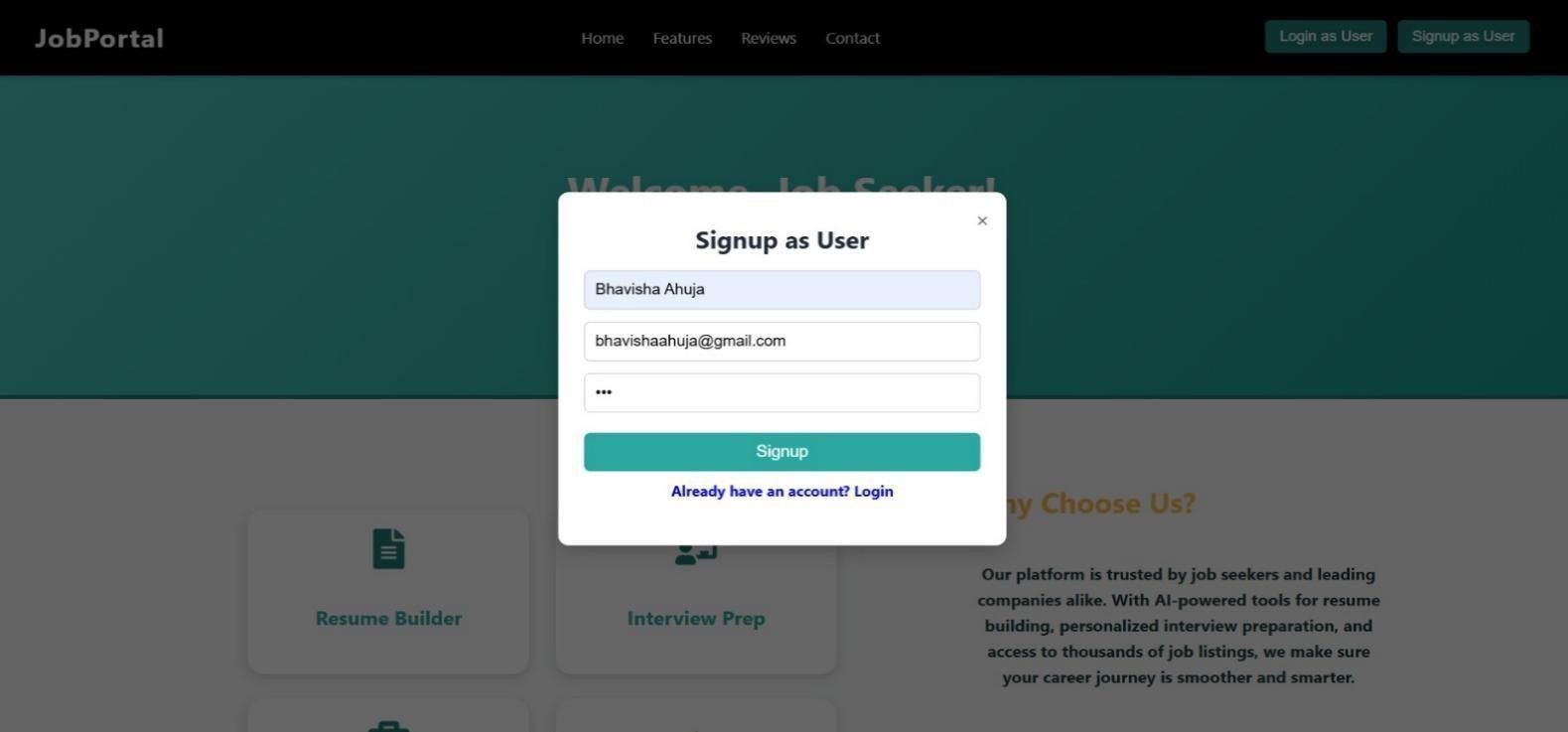


Fig 2. SignUp as User

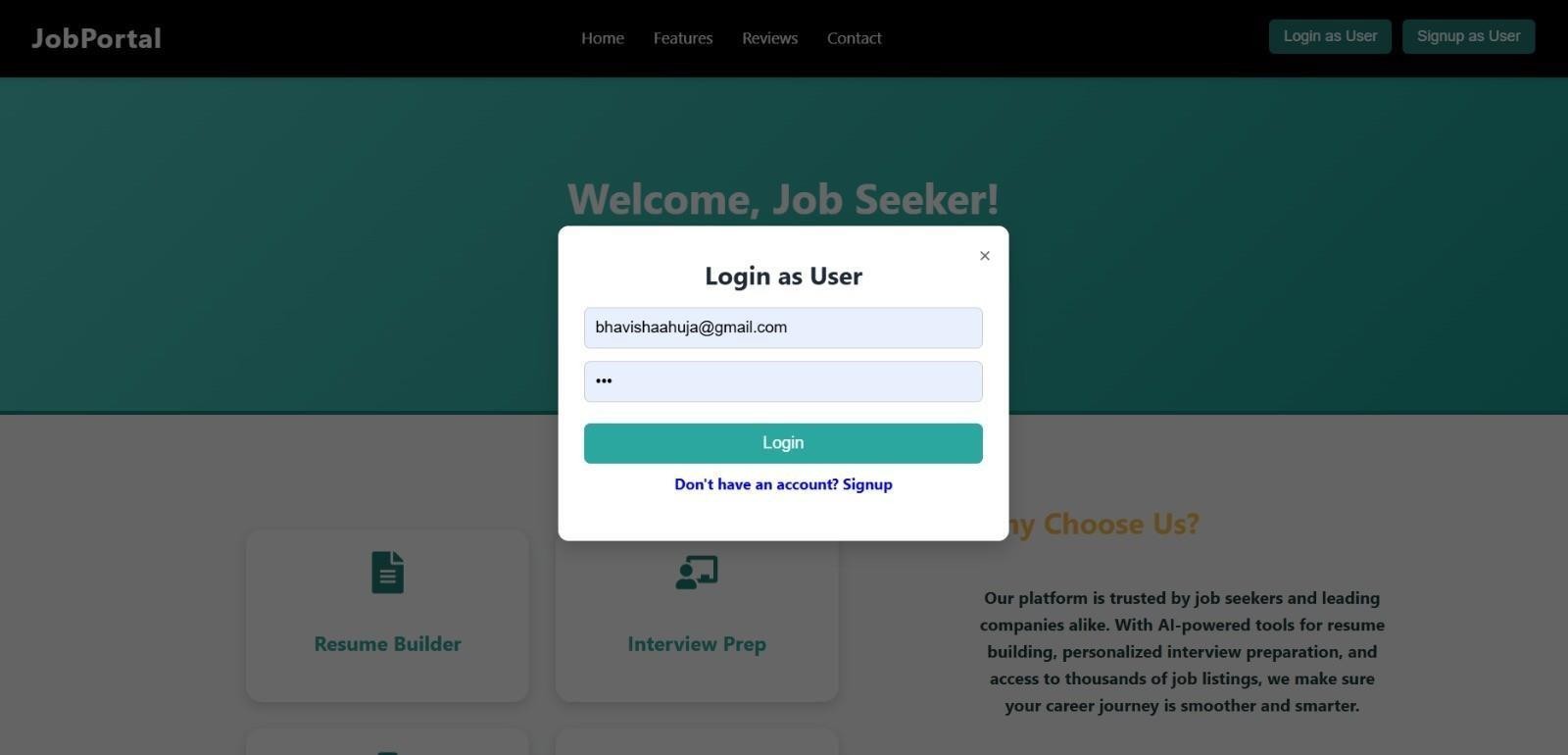


Fig 3. Login of User

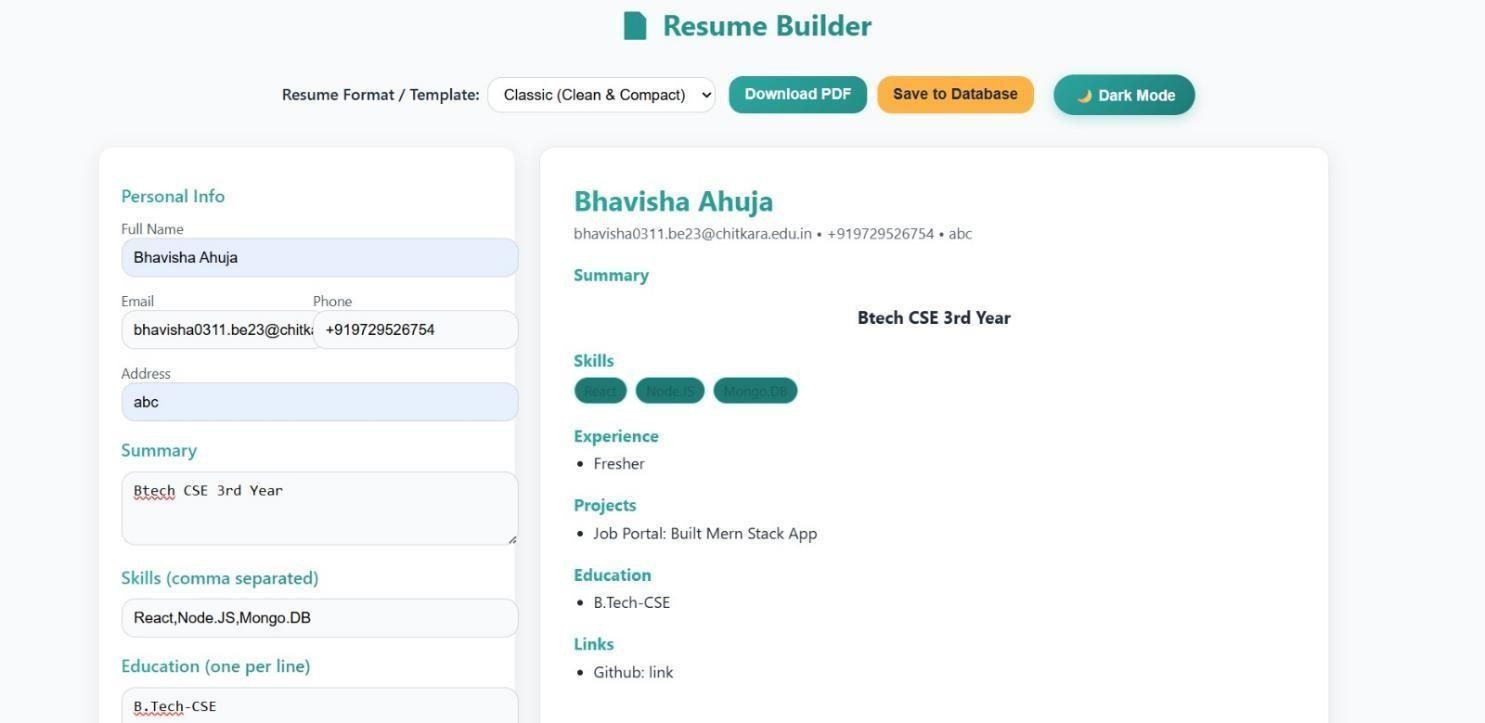


Fig 4. Resume Builder

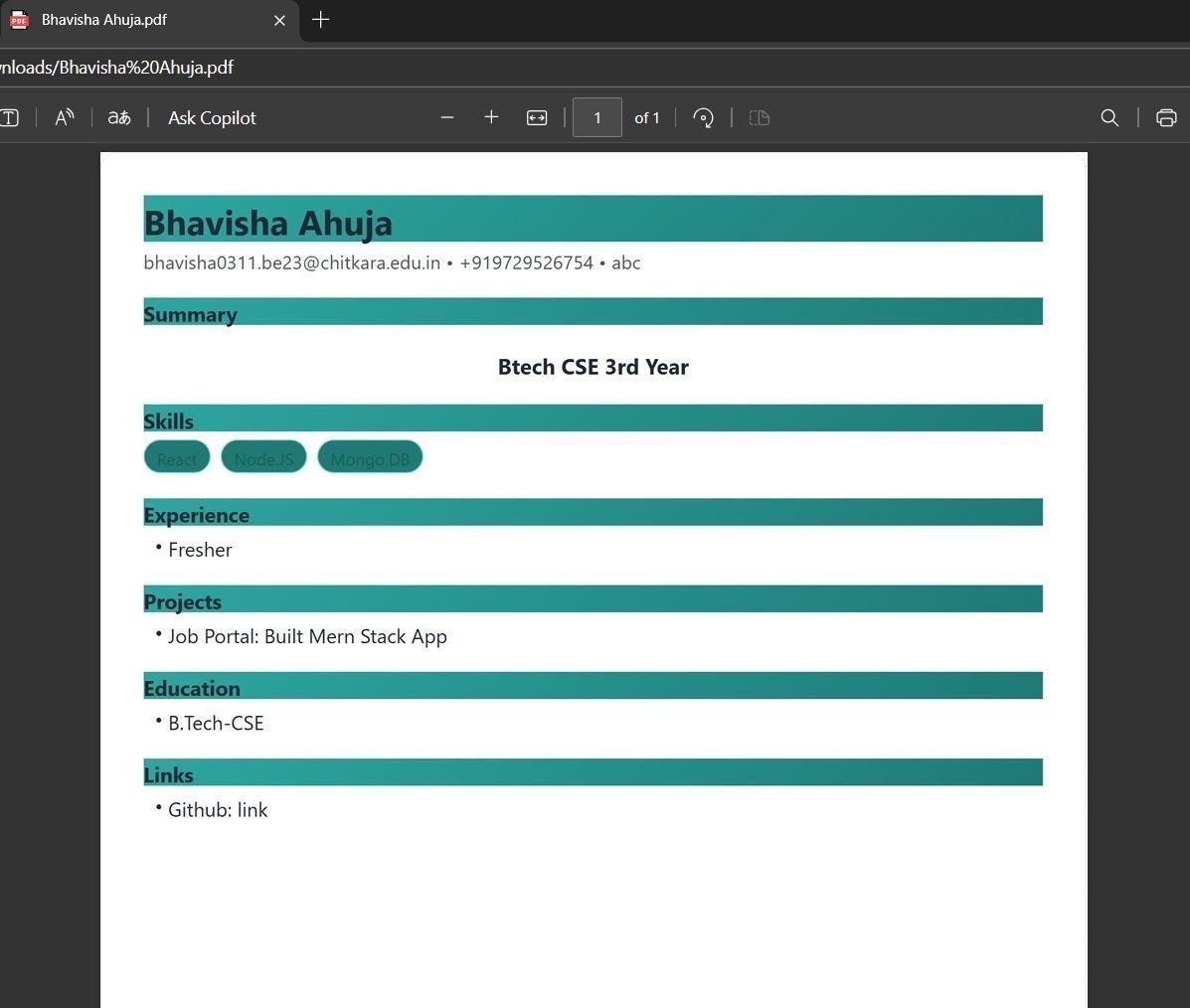


Fig 5. Summary of Resume

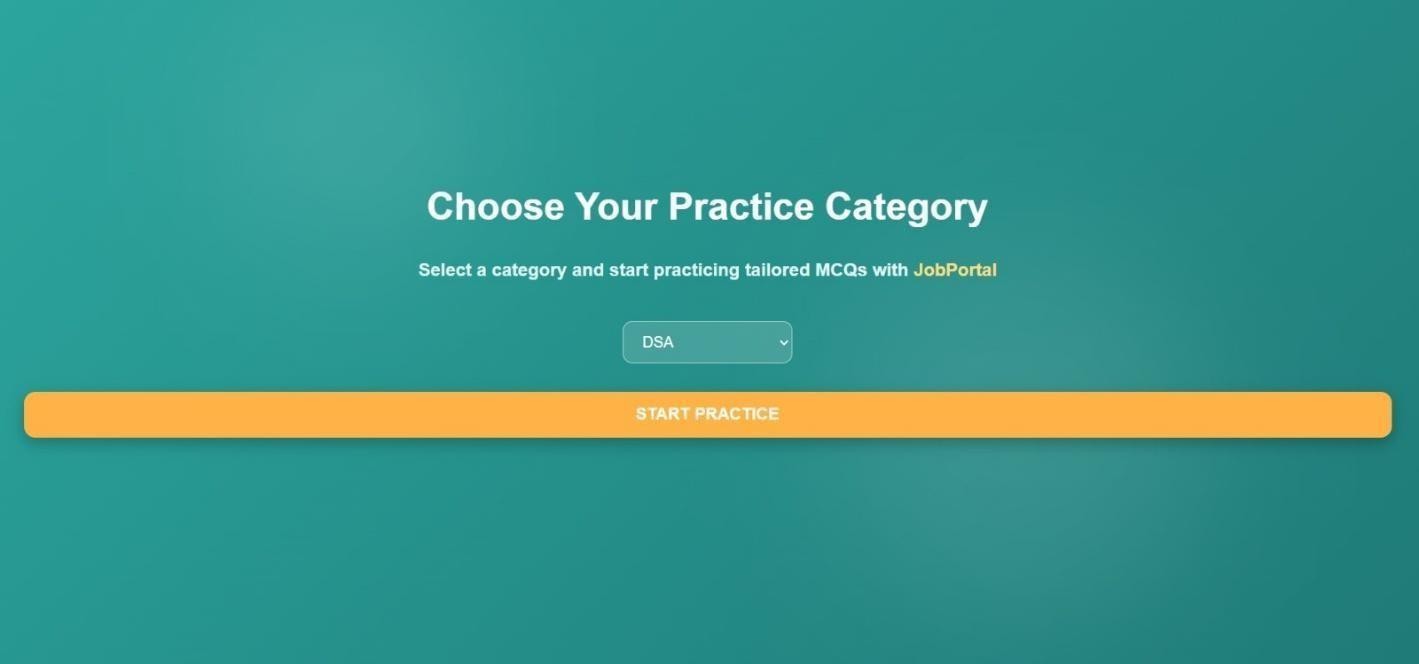


Fig 6. Practice Category Page

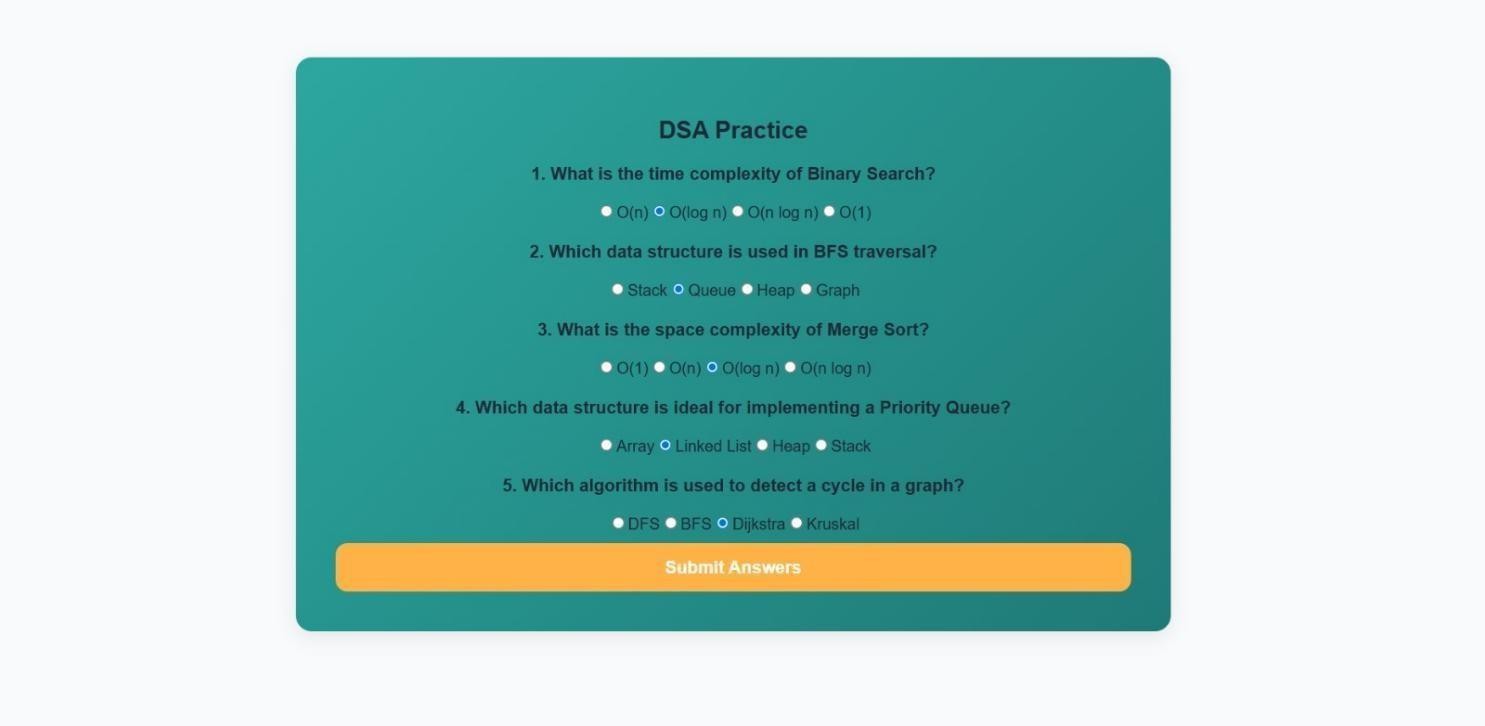


Fig 7. Practice Question Page



Fig 8. Score Page

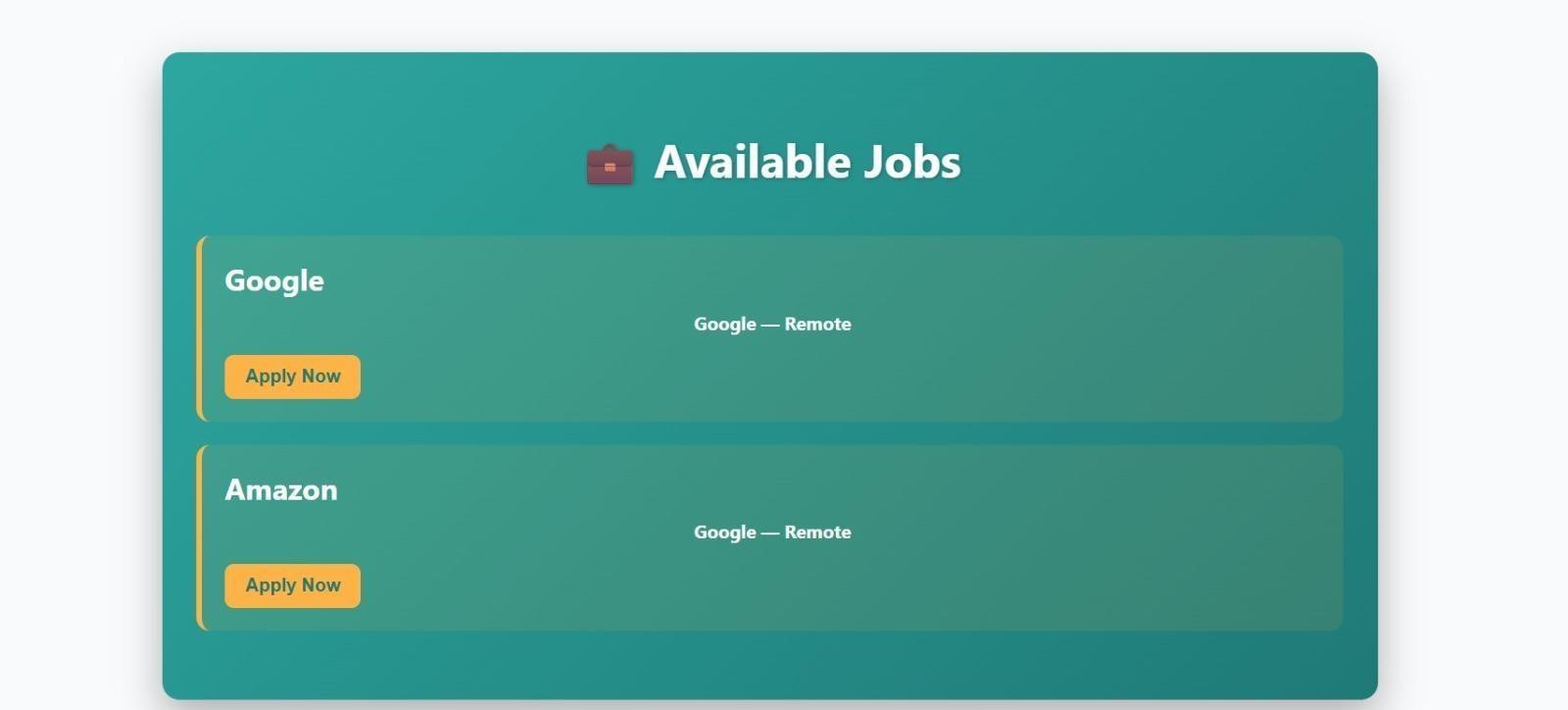


Fig 9. Available job page

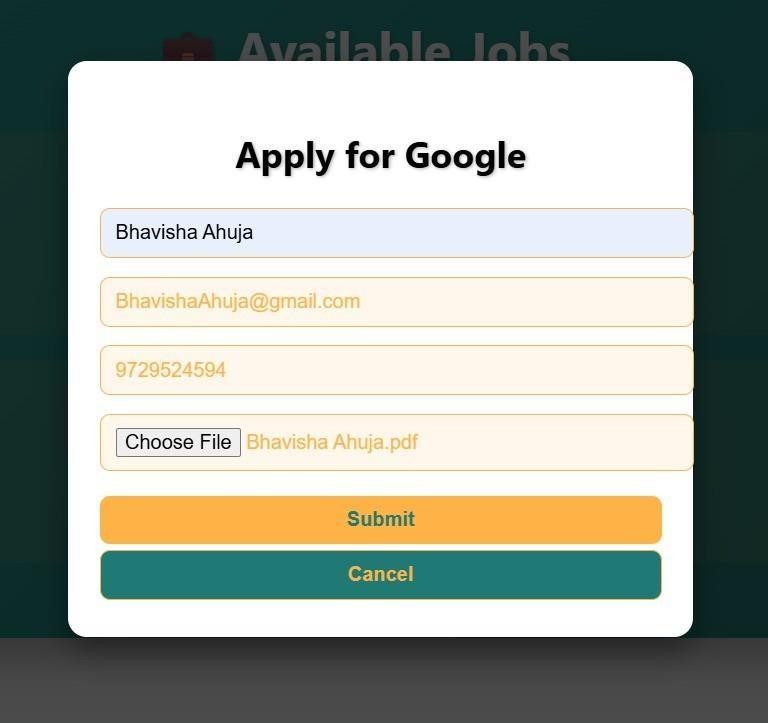


Fig 10. Company Apply page

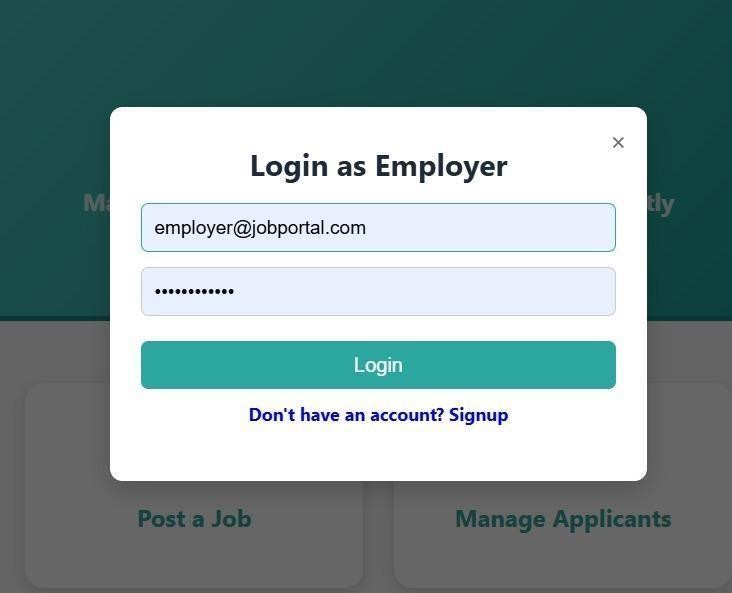


Fig 11. Login of Employer

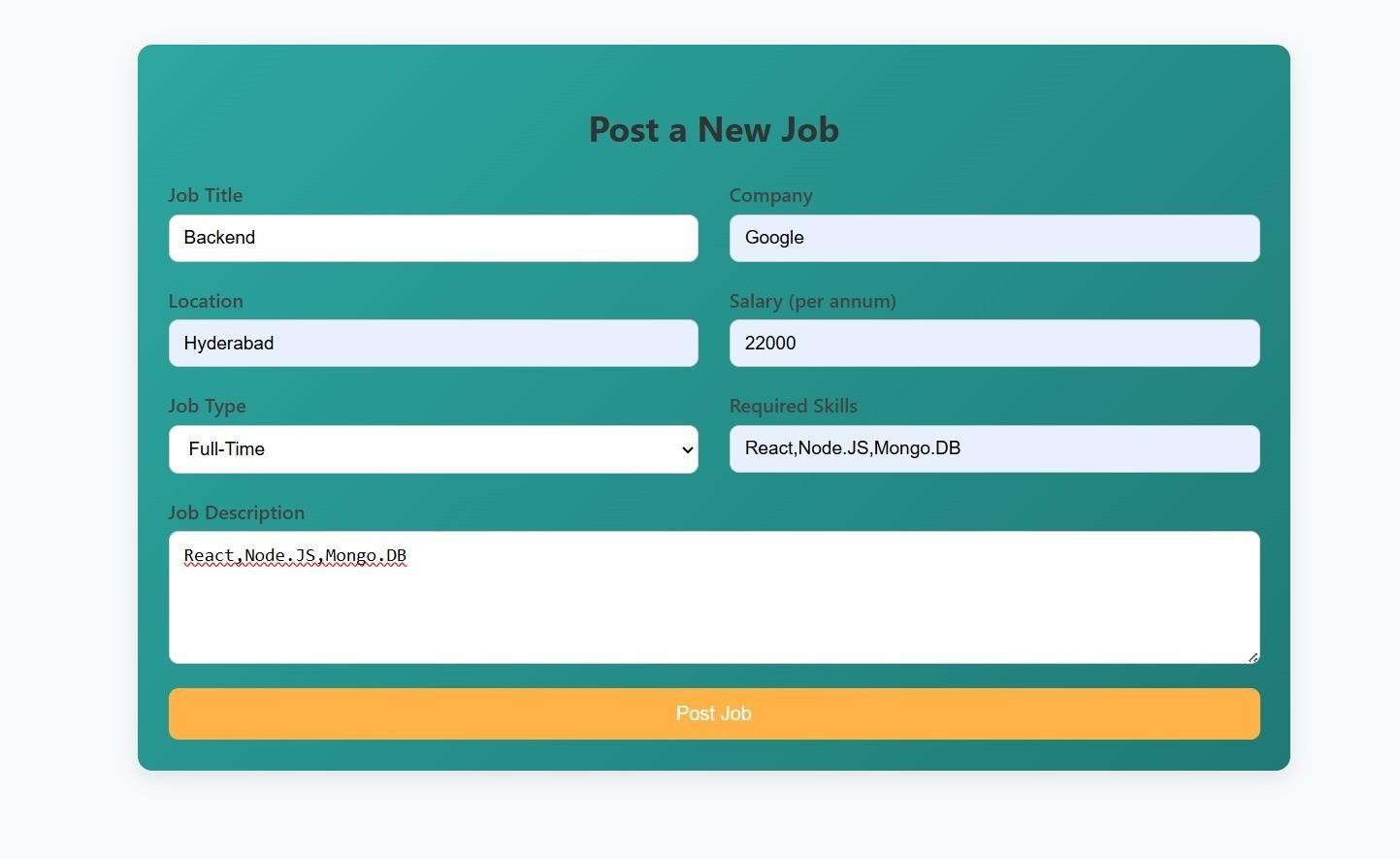


Fig 12. Job Creation Page

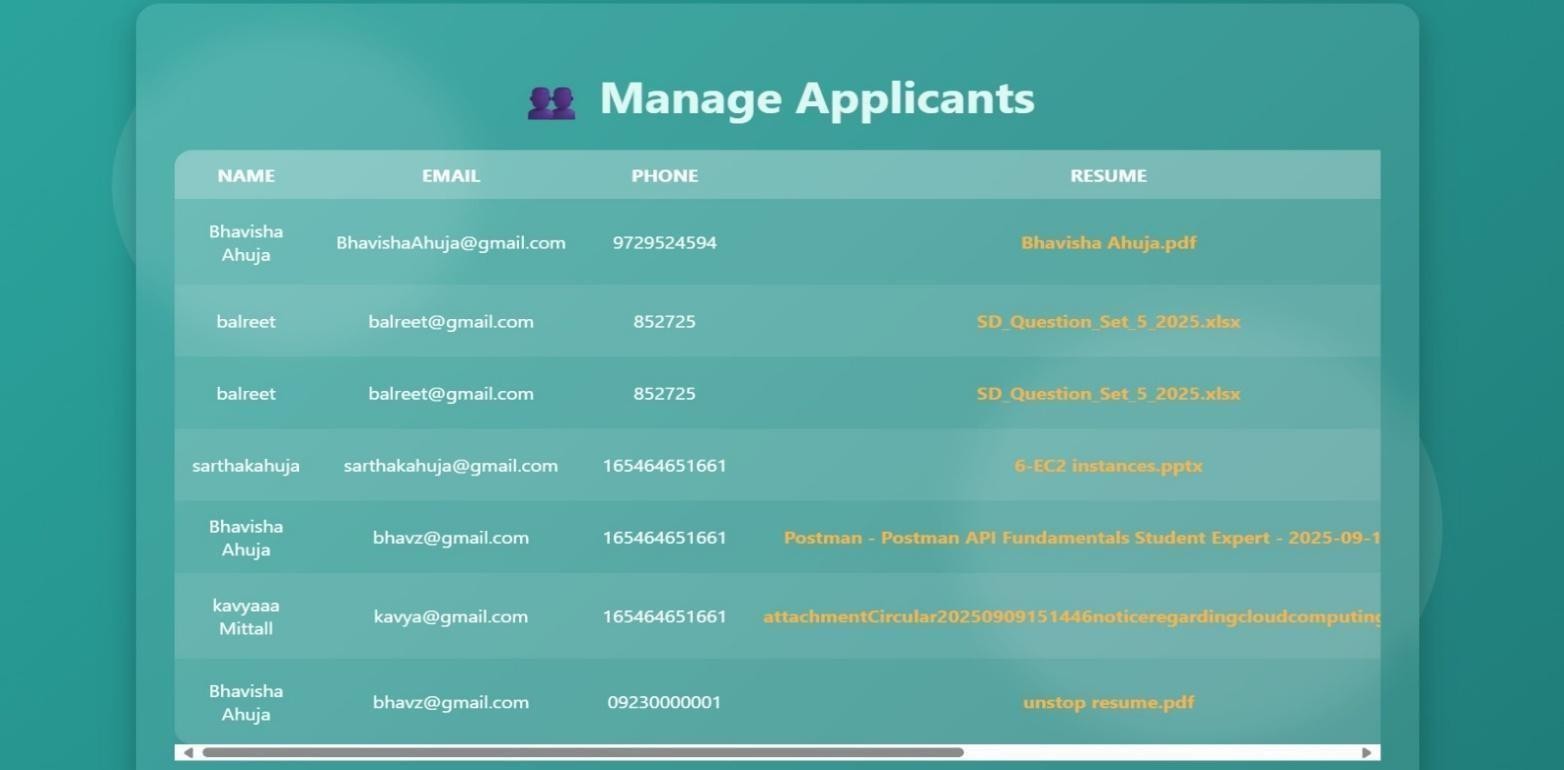


Fig 13. Manage Applicants Page



Fig 14. Accept/Reject Resume Pages

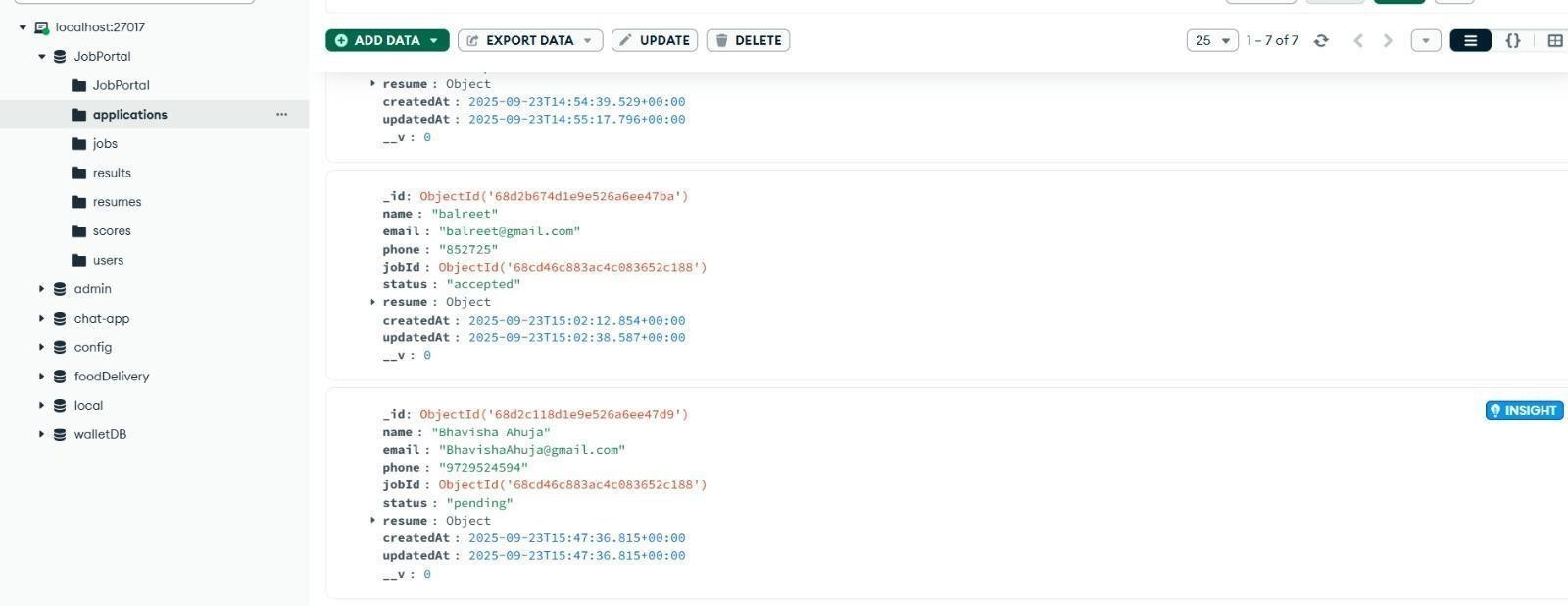


Fig 15. Application database

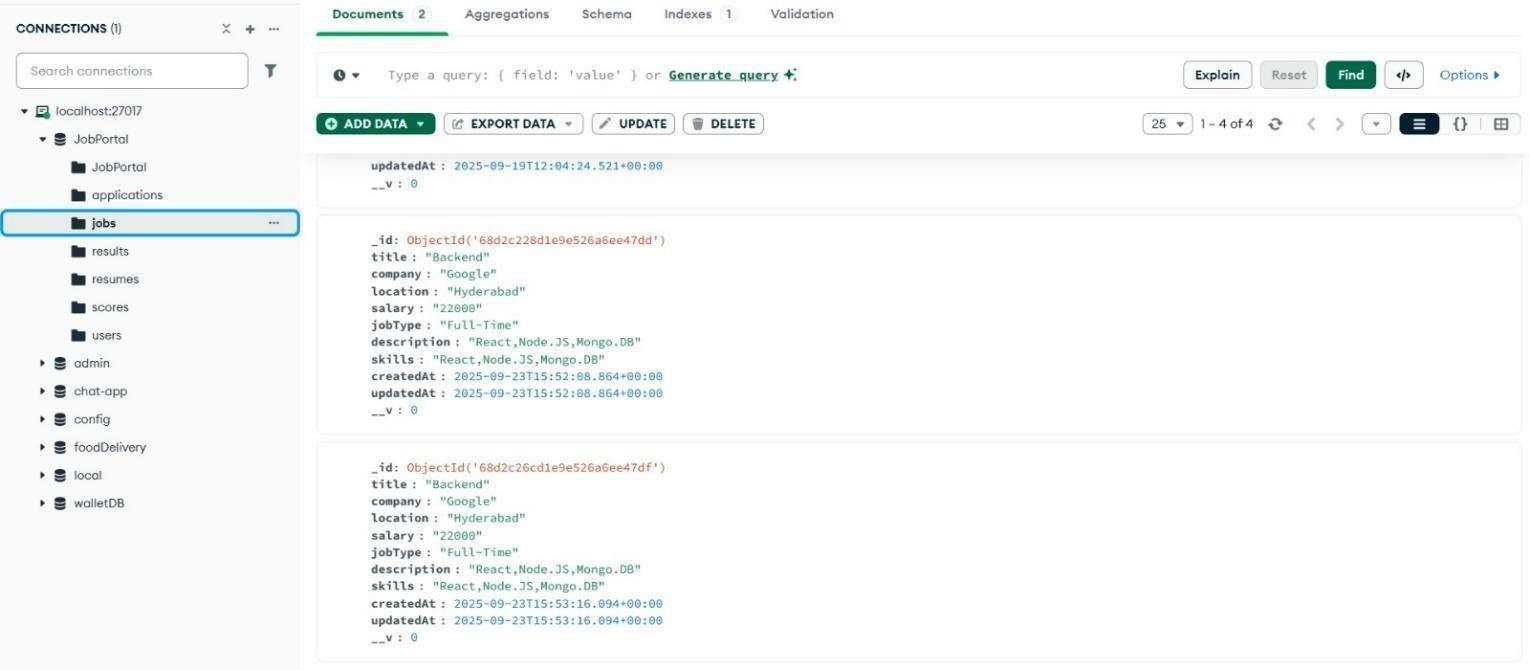


Fig 16. Jobs Database

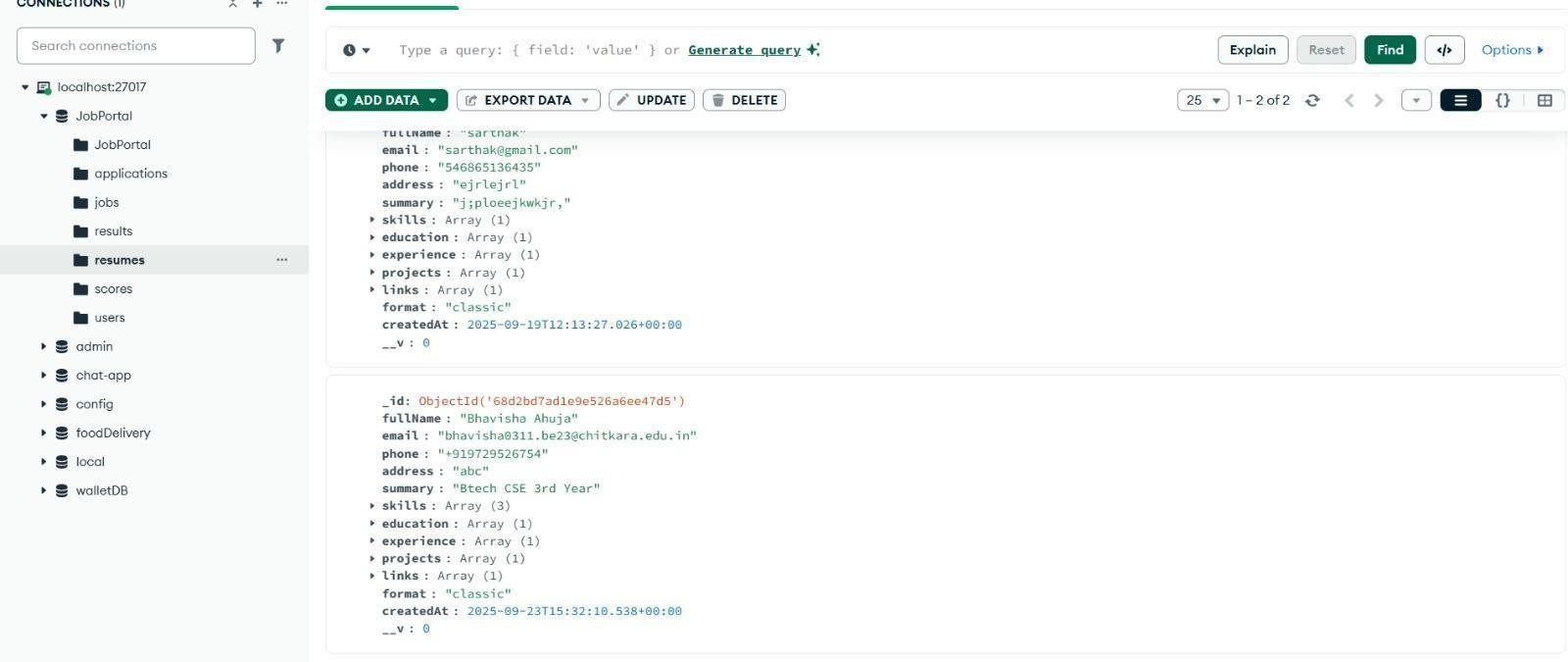


Fig 17. Resume database

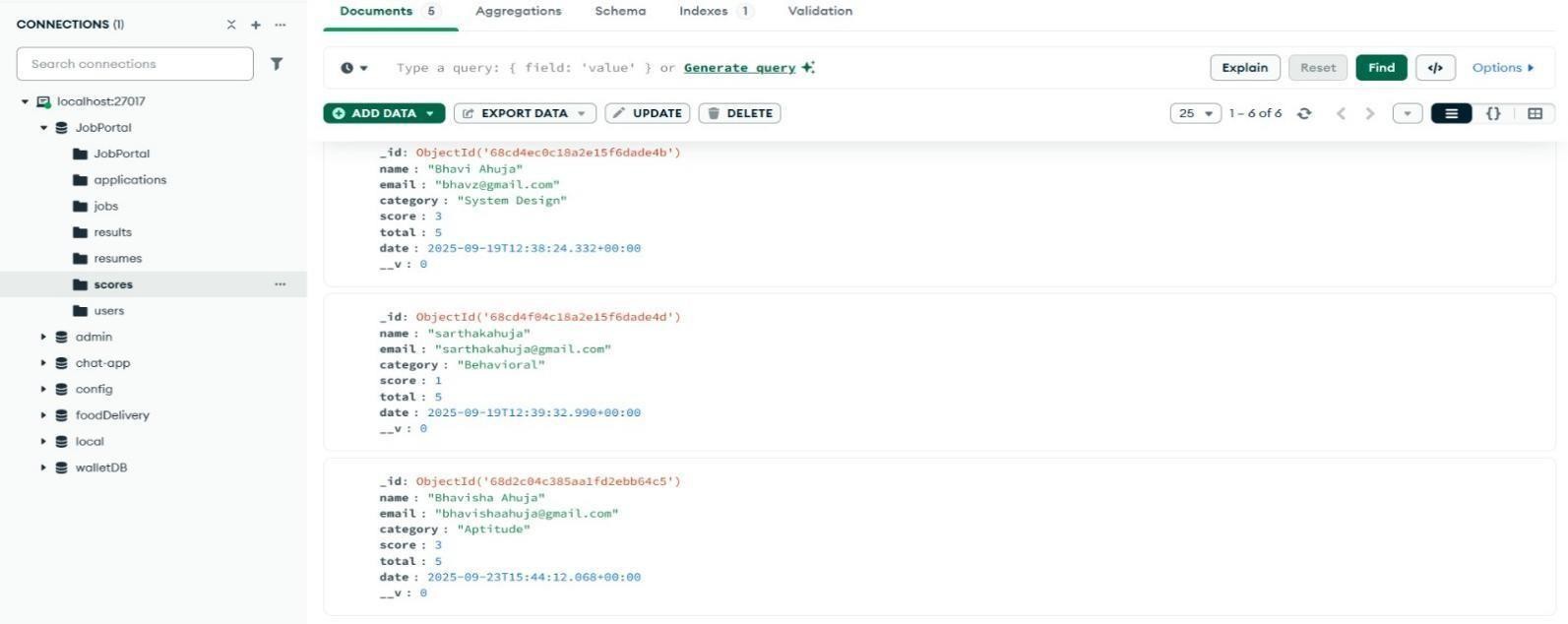


Fig 18. Score Database

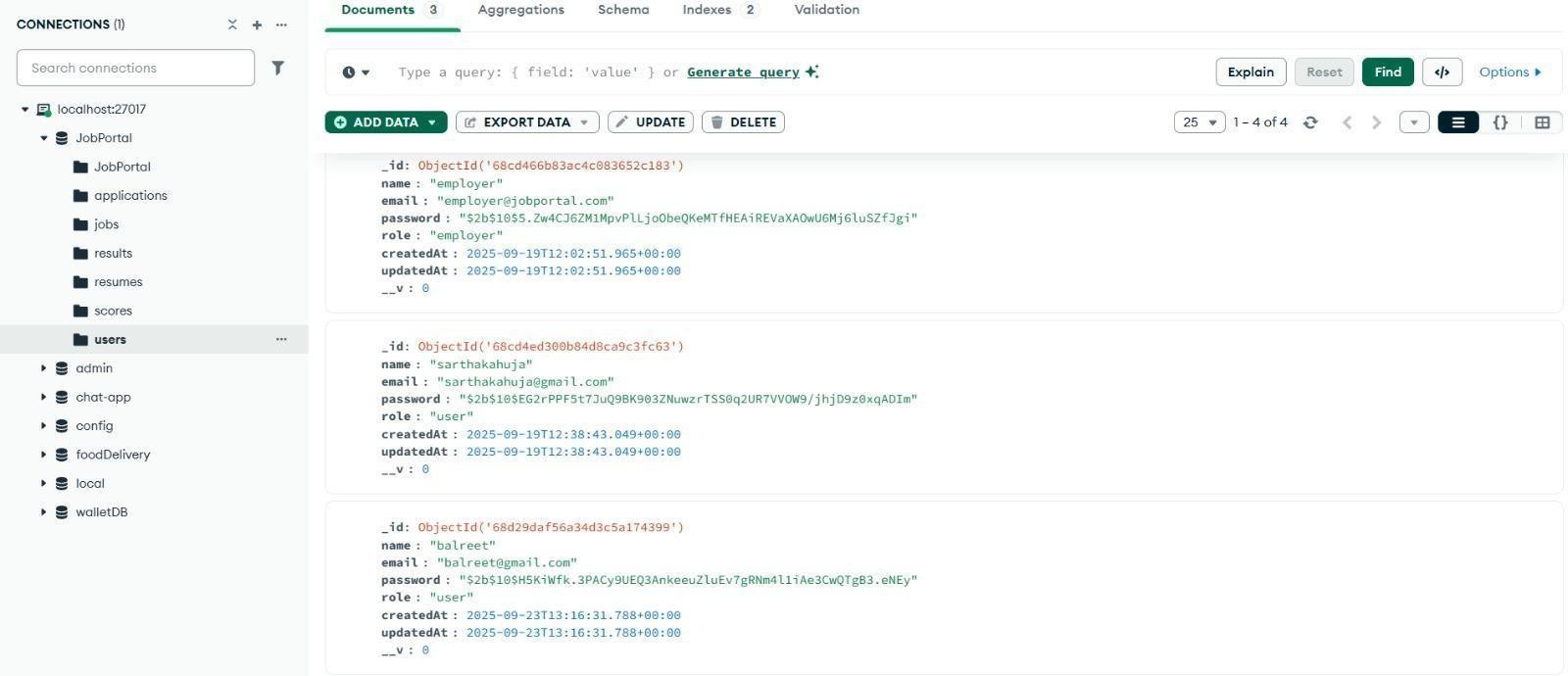


Fig 19. Users Database

## References:

**Official Documentation**: Documentation for libraries, frameworks, and tools used in the project, as well as APIs or services integrated.

**Tutorials and Guides**: Online tutorials, guides, blog posts, and educational videos that provided assistance or insights during development.

**Code Repositories**: GitHub repositories or other code repositories where code snippets, examples, or inspiration were found.

**Forums and Communities**: Online forums, such as Stack Overflow or Reddit, and developer communities where questions were asked, advice was sought, or discussions were participated in.

**Personal Communication**: Mentors, peers who provided guidance, feedback, or support during development.