LAB PRACTICAL REPORT
ON
BACKEND ENGINEERING
22CS008

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Asmita Sharma (2310990307)

Balreet Singh (2310090310)

Bhavisha Ahuja (2310990312)

Dhruv Kumar (2310990325)

Supervised By:

Mr. Rahul

Mentor/Trainer

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CHITKARA UNIVERSITY

CHANDIGARH-PATIALA NATIONAL HIGHWAY, RAJPURA, PUNJAB,

INDEX

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.

1) INTRODUCTION

In today's highly competitive and rapidly evolving job market, both job seekers and employers face significant challenges in finding the right match. For job seekers, identifying opportunities that align with their skills, qualifications, and career goals can be overwhelming due to the vast number of available options. Similarly, employers often struggle to attract and shortlist suitable candidates in an efficient and timely manner. Traditional recruitment methods, such as newspaper advertisements, job fairs, and walk-in interviews, are not only time-consuming but also limited in reach and scope. These methods may work at a local level but fail to leverage the global reach and efficiency provided by digital technologies.

With the advancement of technology and the widespread availability of the internet, online job portals have emerged as a transformative solution to bridge this gap. A job portal serves as a digital marketplace where job seekers and recruiters can interact seamlessly. Job seekers can register, create detailed profiles, upload resumes, and browse available opportunities based on their preferences. On the other hand, recruiters can post job vacancies, filter through applications, and manage candidates from a centralized platform. Unlike traditional systems, job portals offer features such as keyword-based search, advanced filtering, real-time notifications, and secure authentication, which significantly improve the speed, reliability, and effectiveness of the recruitment process.

The proposed **Job Portal Web Application** has been developed to simplify and modernize recruitment by bringing job seekers and employers together on one unified platform. Built using **React** for the front end, **Node.js** and **Express.js** for the back end, and **MongoDB** as the database, this system provides a dynamic and interactive interface, robust server-side logic, and scalable data storage. The application incorporates essential features such as user authentication, profile management, job posting, resume uploading, application submission, and applicant tracking. Additionally, a **notification system** has been implemented to keep candidates informed about job updates, application status changes, and employer responses in real time, enhancing communication and engagement.

To assist employers in making data-driven decisions, an **analytics dashboard** has been added, providing visual insights into recruitment performance, such as total applicants per job, accepted and rejected applications, and overall hiring trends. These analytical tools enable employers to evaluate their job postings effectively and improve their hiring strategies based on real-time data.

By leveraging modern frameworks and libraries, the platform ensures a smooth, secure, and responsive user experience across devices. This project ultimately aims to reduce the time and effort required in recruitment while increasing the chances of successful employment by connecting the right talent with the right opportunities. For job seekers, it eliminates the hassle of searching through fragmented sources, while for employers, it streamlines the hiring process by providing quick and effective access to suitable candidates. In the long run, the Job Portal Web Application enhances accessibility, transparency, and reliability, making the overall recruitment process more effective and impactful.

1.1 Background and Significance

Recruitment has always been an integral part of organizational growth and workforce management. Historically, companies relied on offline methods such as newspaper advertisements, employment exchanges, third-party recruitment agencies, and walk-in interviews. While these methods provided some level of visibility, they were often slow, costly, and restricted to a specific geographic location. Furthermore, offline recruitment lacked tools for tracking applicants, comparing profiles, and maintaining centralized records. As organizations grew in scale, these limitations became increasingly evident.

The rise of the internet and digital technologies revolutionized how recruitment is conducted. Online job portals such as Naukri, Indeed, Glassdoor, and LinkedIn have changed the landscape by offering employers and job seekers broader access and faster processes. Job seekers can now apply to multiple opportunities with just a few clicks, while recruiters can instantly reach thousands of potential candidates worldwide. The introduction of digital profiles, resume uploads, and search filters has significantly increased the efficiency of matching candidates with job openings.

However, even with these advancements, challenges remain. Many existing platforms are either overcrowded, difficult to navigate, or expensive for small and medium-sized enterprises (SMEs). Job seekers often face difficulties in creating standardized resumes, preparing for interviews, and tracking their applications. Employers, on the other hand, may struggle with filtering out irrelevant applications and effectively managing the large volume of candidate data.

The proposed **Job Portal Web Application** addresses these gaps by providing:

• For Job Seekers:

- A centralized and user-friendly platform to create profiles and upload resumes.
- A built-in **resume builder** to generate professional resumes in standardized formats.
- Access to **personalized job recommendations** based on skills, location, and preferences.
- An **interview preparation module** with assessments on technical and aptitude topics to enhance job readiness.
- **Real-time application tracking** to monitor the status of submitted applications.
- A **notification system** that alerts candidates about job updates, interview calls, and employer responses, ensuring timely communication and engagement.

• For Employers:

- A **cost-effective solution** for posting jobs and reaching suitable candidates efficiently.
- A **structured React-based dashboard** to manage job postings, view applicant profiles, and track their responses.
- The ability to **accept or reject applications**, automatically generating notifications to inform candidates of their application status.

- A **centralized MongoDB-powered database** to manage and store candidate information securely and efficiently.
- An **interactive analytics dashboard** that provides visual insights through charts and graphs, helping employers track total applicants per job, application outcomes, and overall hiring performance. This enables data-driven decision-making and improves recruitment strategy.

The significance of this project lies in its ability to combine modern technologies and user-centric features into a cohesive and scalable platform. The use of **React** for the front end ensures a dynamic, responsive, and interactive user interface that provides a smooth experience across devices. **Node.js** and **Express.js** efficiently handle the server-side logic, managing APIs for authentication, job posting, notifications, and application management. **MongoDB** serves as a robust backend database capable of handling large and flexible datasets while ensuring performance, scalability, and security.

By addressing the limitations of both traditional and existing online recruitment methods, the **Job Portal Web Application** delivers a balanced, efficient, and reliable recruitment ecosystem. It simplifies hiring for employers, enhances accessibility and engagement for job seekers, and introduces data-driven insights for improved decision-making. Ultimately, this project demonstrates how the integration of **modern web technologies like React, Node.js, and MongoDB** can transform recruitment into a smarter, faster, and more transparent process — benefiting both organizations and aspiring professionals.

1.2 Objectives

The primary objective of the **Job Portal Web Application** is to design and implement a modern, user-friendly, and scalable recruitment platform that bridges the gap between job seekers and employers. Unlike traditional hiring methods, this portal seeks to create a **digital ecosystem** that simplifies every stage of the recruitment cycle — from creating a resume and posting jobs to applying for positions, managing candidates, and analyzing recruitment performance. The system has been developed using **React** for the front end, **Node.js** and **Express.js** for the back end, and **MongoDB** for the database, ensuring responsiveness, high performance, and secure data handling.

The specific objectives of the project include:

- **Providing a Secure and User-Friendly Platform**
  The portal aims to deliver an intuitive and visually appealing interface where both job seekers and employers can interact with minimal technical barriers. Role-based access, robust authentication, and encrypted data storage ensure the platform remains reliable and secure.
- **Empowering Job Seekers**
  Job seekers can register on the platform, create professional profiles, upload or build resumes, and apply directly to job postings. This feature ensures accessibility for candidates from diverse backgrounds while reducing the complexity of finding suitable opportunities. A **real-time notification system** keeps candidates informed about application updates, interview calls, and employer feedback, ensuring better engagement and communication.

- **Facilitating Employers and Recruiters**
  Employers can register, create detailed job postings, and efficiently manage incoming applications from potential candidates. They can **accept or reject applications** directly, triggering **automated notifications** to inform candidates of their status. This streamlines the recruitment process while reducing manual follow-ups and response delays.

- **Implementing Search and Filter Options**
  To save time and effort for both parties, advanced search and filtering functionalities are integrated into the platform. Job seekers can refine job opportunities based on role, experience level, salary range, and location, while employers can quickly access and filter through relevant candidate profiles.

- **Providing Real-Time Notifications**
  The system includes a **notification module** that delivers instant updates to both employers and candidates. Employers receive alerts for new applications or status changes, while candidates are notified when their applications are viewed, accepted, or rejected.

- **Delivering Data Insights and Analytics**
  An integrated analytics dashboard provides employers with visual insights into key recruitment metrics such as total applicants per job, acceptance and rejection rates, and job performance trends. This enables data-driven decision-making and helps organizations evaluate the effectiveness of their hiring strategies.

- **Ensuring Data Security and Encryption**
  All sensitive user data, including login credentials and application information, is encrypted and securely stored in MongoDB. Authentication and authorization are handled through secure APIs to ensure that only verified users can access specific functionalities.

- **Reducing Recruitment Time and Cost**
  By centralizing the entire hiring process into a digital platform, the system eliminates manual effort, paperwork, and delays. It reduces overall recruitment time and operational costs, providing an efficient alternative to traditional hiring practices.

1.3 Features and Functionality

To achieve these objectives, the **Job Portal Web Application** integrates a comprehensive set of features divided into platform-wide functions and role-specific functionalities for **employers** and **job seekers**. These features not only support essential recruitment operations but also introduce modern tools to enhance efficiency, user experience, and decision-making.

General Features

- **Landing Page**
  The application opens with a responsive and visually engaging landing page, built using **React**, that introduces the two primary roles — **Job Seeker (User)** and **Employer**. From this page, users can easily navigate to login or registration, ensuring an intuitive first impression.

- **Role-Based Login and Authentication**
  A secure authentication system ensures that only registered users can log in. The login module distinguishes between job seekers and employers, redirecting each to their personalized dashboards. Authentication is handled using **JWT (JSON Web Tokens)**, ensuring secure session management and protection of user data.
- **Database Integration with MongoDB**
  MongoDB serves as the backbone of the platform's data management, storing user information, job postings, applications, notifications, and analytics data. Using **Mongoose** schemas, the system maintains structure, validation, and consistency across all records.

Employer Features

Employers have access to a feature-rich environment that simplifies the hiring process and provides actionable insights.

- **Employer Dashboard**
  After logging in, employers are directed to a **React-based dashboard** that provides an overview of their recruitment activities. It displays active job postings, received applications, analytics insights, and recent notifications for quick access.
- **Post a Job**
  Employers can create detailed job postings with the following fields:
  - Job Title
  - Job Description
  - Company Name
  - Salary
  - Location
  - Job Type (Full-Time, Part-Time, Internship, etc.)

These postings are stored in the **Jobs collection** of MongoDB and dynamically displayed to job seekers through the frontend interface.

- **Manage Applicants**
  Employers can view and manage all applications received for their job postings. Each entry includes the applicant's profile and resume. Employers can:
  - Accept or reject applications.
  - Store and track responses securely in MongoDB.
  - Trigger real-time updates that reflect instantly in the user's dashboard.
- **Notifications**
  Employers receive **real-time notifications** whenever a new application is submitted or

an existing application is updated. This enables prompt action and smoother communication throughout the hiring process.

- **Analytics Dashboard**

  A major enhancement to the system is the inclusion of an Analytics Dashboard, providing employers with data-driven insights into recruitment activities. Employers can view graphical reports and metrics such as:

  - Number of applicants per job posting.
  - Acceptance and rejection rates.
  - Overall hiring performance and trends.

  This analytical capability empowers employers to make informed decisions and continuously improve their recruitment strategies.

User (Job Seeker) Features

Job seekers are provided with tools that simplify job searching, enhance readiness, and improve visibility to potential employers.

- **User Dashboard**
  Upon logging in, users are taken to a personalized dashboard that displays relevant job postings, application statuses, and access to career tools like the **Resume Builder** and **Interview Preparation Module**.
- **Job Section**
  Job seekers can:
  - Browse all available job postings from registered employers.
  - Use **search and filter options** to refine results based on job title, type, salary range, and location.
  - Apply directly for positions, with the system storing applications in MongoDB and linking them to the user's profile.
- **Resume Builder**
  An integrated **Resume Builder** allows users to create professional resumes by entering their personal, educational, and professional details. The generated resume follows standardized formatting and can be updated or used for applications within the portal.
- **Interview Preparation Module**
  To improve job readiness, candidates can access an **interview preparation section** featuring practice questions and quizzes covering core areas such as **Data Structures & Algorithms (DSA)**, **Aptitude**, and **Logical Reasoning**. Results from these assessments help users gauge their strengths and areas for improvement.
- **Application Tracking**
  Users can monitor the **real-time status** of every submitted application. Any updates

made by employers — such as acceptance or rejection — are immediately reflected in the candidate's dashboard.

- **Notifications**
  Candidates receive **instant notifications** regarding job application updates, interview calls, and employer feedback. This ensures transparency and keeps users engaged throughout the recruitment process.

System Functionalities

Beyond role-specific tools, the application includes several **core system functionalities** that ensure smooth, reliable, and secure operation.

- **Database Management**
  MongoDB efficiently handles data storage for users, employers, job listings, applications, notifications, and analytics records. Its flexible NoSQL structure enables scalability and fast querying, even as the user base expands.
- **Notification System**
  MongoDB efficiently handles data storage for users, employers, job listings, applications, notifications, and analytics records. Its flexible NoSQL structure enables scalability and fast querying, even as the user base expands..
- **Scalable Architecture**
  Built using **Node.js** and **Express.js**, the backend supports multiple concurrent requests, providing stability, responsiveness, and scalability even under high traffic conditions.
- **Responsive Design**
  The use of **React** ensures that all pages and dashboards are fully responsive, delivering a seamless experience on desktops, laptops, tablets, and smartphones.
- **Security**
  The application employs encryption, **JWT-based authentication**, and **role-based access control** to safeguard sensitive data and prevent unauthorized access. User information and credentials remain protected at all times.
- **Analytics and Reporting**
  The analytics module uses MongoDB's aggregation capabilities and React visual components (charts/graphs) to present recruitment insights clearly to employers. It enhances decision-making and strategic planning.

By combining these robust features with modern technologies, the **Job Portal Web Application** delivers a complete, scalable, and data-driven recruitment ecosystem. It simplifies the hiring process for employers, empowers job seekers with career tools and insights, and ensures that all operations—from job posting to hiring—are secure, efficient, and transparent.

Technology Stack

The Job Portal Web Application has been developed using the **MERN (MongoDB, Express.js, React.js, Node.js)** stack—one of the most widely adopted full-stack development architectures for building scalable, responsive, and modern web applications. MERN provides a unified JavaScript-based development environment where the frontend, backend, and database interactions are handled seamlessly, eliminating the need to switch between multiple languages. This improves developer productivity, reduces complexity, and ensures consistency across all layers of the system.

In a job portal ecosystem that must simultaneously serve **job seekers, employers, and administrators**, the MERN stack offers real-time performance, high scalability, secure data handling, and smooth user experiences. Whether processing thousands of resumes, enabling employers to post job vacancies instantly, supporting analytics dashboards, or delivering real-time notifications, MERN's non-blocking architecture makes it an ideal choice.

The stack also supports **modern application needs** such as:

- **Single Page Applications (SPA)** for fast and fluid user interactions
- **RESTful APIs** for efficient communication between frontend and backend.
- **Cloud deployment** for handling heavy traffic during peak hiring seasons
- **Secure authentication** for protecting resumes, company details, and applicant credentials.
- **Real-time notifications** for application updates and employer alerts
- **Analytics processing** using MongoDB aggregations for insights such as applicant counts, job posting performance, and hiring trends

Each technology in the stack plays a specialized role while working in harmony with the others. Below is a **layer-by-layer breakdown** of the stack used in this project:

1. Frontend – React.js

- The frontend interface is built entirely using **React.js**, a powerful library known for its modular, scalable, and component-driven architecture. Rather than relying on traditional server-rendered templates, the application uses **JSX** to build dynamic UI components efficiently.
- **React.js (with JSX):**
  React powers all user-facing features, including dashboards, job listings, application tracking, the resume builder, notifications panel, and analytics visualizations.
    - **Component-based architecture** ensures reusability and easier maintenance
    - **Virtual DOM** enables high performance during job searches, filtering, or profile updates
    - **React Hooks** (useState, useEffect, useContext) simplify state managemen

- **JavaScript (ES6+):**
  Modern JavaScript ensures clean, efficient logic for:
    - API communication (using fetch or axios)
    - Asynchronous operations such as resume uploads, job posting updates, and real-time status changes
    - Improved readability with ES6 features like arrow functions, async/await, modules, and destructuring
- **Styling (CSS-in-JS, CSS Modules, and UI Libraries):**
  Instead of raw CSS, the project makes use of **component-scoped CSS modules** and **CSS-in-JS approaches** for encapsulated styling that avoids conflicts between components. Additionally, **Bootstrap** and **Material-UI (MUI)** are used for building responsive and modern UIs. Bootstrap's grid system ensures layouts adjust seamlessly across devices, while Material-UI provides pre-designed, customizable components such as buttons, cards, forms, and navigation bars that align with Google's Material Design standards. This combination ensures that the interface is not only functional but also visually appealing and consistent.
- **Routing and Navigation (React Router):**
  React Router enables smooth transitions between pages—job listings, resume builder, employer dashboard, and analytics—without full page reloads, delivering a seamless SPA experience.

In summary, the frontend leverages **React.js with JSX, modern JavaScript (ES6+), and advanced styling solutions** to deliver a highly interactive, responsive, and scalable interface. By combining these tools, the Job Portal ensures an engaging user experience that meets the needs of both job seekers and employers.

2. Backend

The backend handles core business logic, authentication, data processing, and communication with the database.

Node.js

Node.js provides a **JavaScript runtime environment** outside the browser. Its **event-driven, non-blocking I/O model** makes it ideal for handling concurrent requests. This is critical in a job portal where potentially hundreds of users might simultaneously submit applications, upload resumes, or search for jobs.

Key benefits in this project:

- **Scalability**: As the number of employers and job seekers increases, Node.js ensures the system scales without major performance drops.
- **Real-time support**: Features like instant notifications (e.g., "Your application has been received") are enabled.

- **High throughput**: Multiple tasks (e.g., database queries, API calls, file uploads) can be processed without blocking one another.

Express.js

Express.js sits on top of Node.js and acts as the **backend framework**. It is used for:

- **Routing**: Directing requests like /login, /post-job, or /apply-job to their respective controllers.
- **Middleware**: Enforcing authentication, error handling, and logging. For example, only authenticated employers can access the job posting route.
- **RESTful APIs**: Creating endpoints that the frontend consumes. For instance, when the React app requests job listings, Express fetches them from MongoDB and sends them back as JSON.

Express is lightweight yet flexible, making it ideal for this project's needs.

The database ensures **persistent storage of user data**. For a job portal, this includes job postings, resumes, user profiles, and application records.

MongoDB

MongoDB is a **NoSQL database** that stores data in a flexible, JSON-like structure. This makes it suitable for job postings that may not always follow a strict schema (e.g., some jobs may have salary fields, others may not).

Advantages:

- **Scalability**: Can handle large datasets as the portal grows.
- **Indexing**: Enables fast queries such as "jobs in New Delhi with salary above 50k."
- **Aggregation pipelines**: Employers can analyze applicant data (e.g., average applications per job).

Mongoose

Mongoose provides an **ODM (Object Data Modeling)** layer over MongoDB. It enforces structure by requiring fields such as jobTitle, companyName, and location. It also simplifies CRUD operations and provides built-in validation, preventing faulty data from being stored.

4. Authentication & Security

Since sensitive data is involved (passwords, resumes, and employer information), **security is paramount**.

- **bcrypt.js**: Used to hash and salt passwords before storing. This ensures that even if the database is compromised, raw passwords cannot be retrieved.

- **JWT (JSON Web Tokens)**: Provides stateless authentication. Once a user logs in, they receive a token, which is then sent with every request. This avoids repeated database lookups and improves performance. JWT also supports **role-based access control**, ensuring employers cannot access seeker-only features and vice versa.

5. Development & Deployment Tools

To streamline development and make the system production-ready, several tools were used:

- **Git & GitHub**: Version control and team collaboration.
- **Nodemon**: Automatically restarts the server during development.
- **Postman**: For API testing before frontend integration.
- **VS Code**: IDE with extensions for linting, debugging, and productivity.
- **Cloud Platforms (AWS/Heroku/Render)**: Enable live deployment and scalability. CI/CD pipelines can be integrated for **automated testing and deployment**.
- **PM2 / Docker** (optional) for production process management and containerization

6. Integration within the MERN Stack

The power of MERN lies in **seamless integration**:

- **React.js** handles the UI.
- **Express.js + Node.js** manage backend logic and APIs.
- **MongoDB** stores data, accessed via Mongoose.
- **JWT & bcrypt.js** ensure secure transactions.
- **Real-time notifications** → Enabled through asynchronous backend processing.
- **Analytics reporting** → Powered by MongoDB Aggregation Framework.

This **JavaScript-only ecosystem** reduces complexity, eases debugging, and ensures faster development cycles. For the job portal, this integration means the system is **functional, secure, scalable, and future-ready**.

3) Problem Definition and Requirements

Problem Definition

In the present era of globalization, digital transformation, and a rapidly evolving job market, the recruitment ecosystem has grown increasingly complex and highly competitive. Both job seekers and employers face multiple challenges that slow hiring cycles, create inefficiencies, and often result in mismatched outcomes. The rise of online job portals has modernized parts of the recruitment process, but significant gaps remain unaddressed, creating frustration for all stakeholders.

Challenges Faced by Job Seekers

For job seekers, the primary challenge lies not just in finding job opportunities but in identifying the **right opportunities** that align with their educational qualifications, career aspirations, and workplace preferences. Today's candidates must navigate through cluttered platforms filled with irrelevant postings, leading to information overload. Some key issues include:

- **Repetitive Application Process:** Candidates frequently need to upload resumes and fill out application forms across multiple portals, wasting considerable time on repetitive data entry.
- **Poor Relevance Matching:** Many portals lack intelligent recommendation systems. This often results in candidates being shown jobs that are mismatched with their skills or career trajectory, leading to wasted effort.
- **Lack of Real-Time Updates:** Job seekers often face long periods of uncertainty after applying. They may wait weeks or months without receiving any update on whether their application has been reviewed, accepted, or rejected.
- **Fragmented Tools:** A typical job seeker uses multiple platforms simultaneously — one for searching jobs, another for resume building, and yet another for practicing interview questions. This fragmented approach reduces efficiency and creates a disconnected experience.
- **Missed Opportunities:** Due to ineffective search filters, outdated postings, or slow response times, candidates may miss out on relevant openings, lowering their chances of securing suitable employment.

Challenges Faced by Employers

On the employer's side, the recruitment process is equally fraught with inefficiencies. Recruiters often face the daunting task of reviewing hundreds or even thousands of resumes, a significant percentage of which may not even meet the basic requirements of the advertised role. Specific challenges include:

- **Volume Overload:** The large number of irrelevant or poorly structured applications makes it difficult for recruiters to identify the most promising candidates quickly.

- **Unstructured Data:** Many resumes submitted online are inconsistent in format, lacking standardized structures. This makes automated parsing and comparison more difficult.
- **Communication Gaps:** Employers often struggle to maintain timely communication with candidates. For example, rejected candidates may not be notified at all, leading to dissatisfaction and a negative impression of the company.
- **Traditional Recruitment Costs:** Outdated hiring methods such as newspaper advertisements, offline walk-in interviews, or expensive campus placements are not only costly but also fail to tap into diverse and geographically distributed talent pools.
- **Time-to-Hire Issues:** The inefficiencies in sorting, filtering, and communicating extend recruitment cycles significantly, which is detrimental in a competitive market where talented candidates are quickly snapped up by other organizations.

Limitations of Existing Online Portals

While portals like LinkedIn, Indeed, and Naukri have undeniably revolutionized recruitment by moving processes online, they come with their own shortcomings:

- **Information Overload:** Pages are cluttered with ads, irrelevant recommendations, and sponsored job posts that often overshadow genuine openings.
- **Ineffective Filters:** Advanced search features may exist but are often poorly implemented, forcing users to browse manually through large lists of jobs.
- **Lack of Role-Centric Dashboards:** Many portals treat employers and job seekers alike without offering tailored dashboards that address the unique needs of each.
- **Absence of Integrated Tools:** Very few portals combine resume building, interview preparation, and application tracking into a single ecosystem. Instead, job seekers are left to use third-party tools outside the system, increasing inefficiency.
- **Weak Employer Support:** Applicant management tools are often underdeveloped, forcing employers to export data manually or use external spreadsheets to track candidates.

Identified Need

There is a strong need for a **centralized, secure, and role-based web application** that consolidates recruitment activities into a single platform. The proposed **Job Portal Web Application** addresses these gaps:

- **Employer Dashboard:** Recruiters will have access to tools for creating job postings, managing applicants, reviewing resumes, and making hiring decisions in real time.
- **Job Seeker Dashboard:** Resume builder (stored in database), job search, interview preparation, application tracking, and **real-time notifications**.

By combining these functions into one seamless system, the application aims to make recruitment **faster, more transparent, less costly, and more user-friendly** for all stakeholders.

Software & Hardware Requirements

To implement the proposed solution, a robust set of software frameworks, tools, and hardware resources is required. These components are carefully chosen not only for ease of development but also for ensuring **security, scalability, efficiency, and long-term maintainability**.

Frontend Requirements

- **React.js (with JSX):**
    - Forms the foundation of the user interface.
    - Offers component-based architecture for building reusable UI elements such as login forms, dashboards, job postings, and resume builders.
    - Virtual DOM ensures fast rendering and real-time updates when data changes.
    - Hooks (useState, useEffect) and Context API streamline state management across the application.
- **JavaScript (ES6+):**
    - Provides dynamic interactivity and asynchronous communication with the backend via fetch/axios.
    - Features like async/await, promises, and arrow functions improve code clarity and performance.
- **Styling Solutions (CSS Modules, Bootstrap, Material-UI):**
    - Instead of raw CSS, component-scoped modules prevent style conflicts.
    - Bootstrap provides a responsive grid system, while Material-UI offers a modern design system aligned with Google's Material Design.
    - Together, these ensure responsive and professional-looking interfaces across devices.
- **React Router:**
    - Enables smooth navigation between application modules without reloading the page, ensuring SPA-like performance.

Backend Requirements

- **Node.js:**
    - Serves as the server-side runtime environment.
    - Non-blocking, event-driven model allows handling thousands of concurrent requests efficiently.
    - Ideal for scaling the application as traffic grows.
- **Express.js:**
    - Provides a lightweight, fast framework for building RESTful APIs.

- Middleware ensures validation, authentication, logging, and error handling.
- Simplifies backend code structure while supporting modularity and scalability.

Database Requirements

- **MongoDB:**
  - NoSQL database designed for flexibility.
  - JSON-like schema makes it ideal for evolving entities like resumes, applications, or job postings.
  - Built-in features like indexing, aggregation pipelines, and sharding improve query speed and scalability.
- **Mongoose (ODM):**
  - Provides schema enforcement, validation, and simplified CRUD operations.
  - Ensures data consistency across users, employers, and applications.
- **Resume Storage:**
  - Resumes are now stored in the database for centralized access and management.

Authentication & Security

- **JWT (JSON Web Tokens):**
  - Enables secure, stateless session management.
  - Tokens are role-based, ensuring employers and job seekers have restricted access according to their roles.
  - Reduces server load by avoiding repeated database lookups for authentication.
- **bcrypt.js:**
  - Hashes and salts passwords before storing them in the database.
  - Protects sensitive data, even in the case of a database breach.

Development Tools

- **Visual Studio Code (VS Code):** Lightweight yet powerful IDE with debugging and extension support.
- **Git & GitHub:** Essential for version control and team collaboration.
- **Postman:** Used for testing and validating API endpoints before frontend integration.
- **Nodemon:** Improves development efficiency by restarting the server automatically when changes are made.

Hardware Requirements

- **Development Machine / Server:** Minimum 8GB RAM with SSD storage for smooth multitasking.
- **Processor:** A modern multi-core processor (Intel i5 / AMD Ryzen 5 or above) to handle server processes.
- **Stable Internet Connection:** Required for API testing, dependency downloads, and deployment.
- **Cloud Deployment Capability:** Services like AWS, Heroku, or Render for scalability and global accessibility.

Significance of the Requirements

The above requirements are not arbitrary but directly contribute to the success of the system in real-world deployment. Their importance can be understood in the following dimensions:

- **Efficiency:** Tools like Postman, VS Code, and Nodemon accelerate the development cycle, reduce errors, and shorten time-to-market.
- **Scalability:** MongoDB's schema-less design and Node.js's event-driven model enable the system to scale as the number of job seekers, employers, and postings grow.
- **Security:** With JWT and bcrypt.js, sensitive user data (e.g., resumes, contact information, and employer credentials) is safeguarded against breaches.
- **User Experience:** React.js, combined with modern UI libraries, ensures a smooth, interactive, and responsive interface tailored to both job seekers and employers.
- **Cost-Effectiveness:** Cloud deployment eliminates the need for expensive on-premises infrastructure, allowing the system to grow flexibly with demand.
- **Industry Alignment:** By using the MERN stack, the application remains aligned with current industry standards, making it easy to upgrade or integrate with other services in the future.

In conclusion, the problem definition highlights the **pressing gaps in existing recruitment systems**, while the requirements define a **robust roadmap for building a secure, scalable, and user-centric Job Portal Web Application** that addresses these gaps effectively

The Job Portal Web Application has been conceived as a modern recruitment solution designed to bridge the gap between job seekers and employers by combining cutting-edge technology, a user-centric design philosophy, and robust backend logic. In contrast to traditional methods such as campus drives, offline walk-ins, and newspaper advertisements, which are often slow, resource-intensive, and geographically limited, this portal provides a centralized, accessible, and scalable digital ecosystem.

Online recruitment platforms like LinkedIn, Indeed, or Naukri have already digitized hiring, but their shortcomings—including cluttered interfaces, limited personalization, poor role separation, and lack of advanced features like resume builders or interview preparation tools—create friction in the hiring journey. Therefore, the proposed system is designed to

combine all the missing elements into a single unified platform using the MERN stack (MongoDB, Express.js, React.js, Node.js).

At its core, the system follows Software Development Life Cycle (SDLC) principles, combined with modular architecture and role-based access control (RBAC). This ensures scalability, maintainability, and future extensibility. The primary design objectives can be summarized as:

- **Responsiveness:** Smooth navigation and consistent performance across desktops, tablets, and smartphones.
- **Scalability**: Ability to support thousands of concurrent users and job postings.
- **Security:** Protection of sensitive user and employer data with modern encryption and authentication practices.
- **Personalization:** Role-specific dashboards and workflows for job seekers, employers, and (optionally) administrators.
- **Transparency:** Real-time updates and notifications to keep all stakeholders informed.

4) Proposed Design and Methodology

1. System Architecture

The application adopts a three-tier layered architecture consisting of the frontend, backend, and database layers, with an optional cloud deployment environment as the infrastructure tier. Each component is designed independently but communicates seamlessly with others, ensuring both flexibility and robustness.

Frontend Layer (React.js)

The client-side portion is implemented using React.js, a powerful JavaScript library for building interactive UIs.

- **Component-Based Development:**
  Every feature (such as job seeker dashboards, employer dashboards, resume builder, and interview preparation module) is encapsulated as a standalone React component. This modularity ensures reusability, scalability, and easier debugging. For example, the "JobCard" component can be reused across job listings, saved jobs, and employer dashboards with minor modifications.
- **Virtual DOM:**
  The Virtual DOM ensures that only the components with updated data are re-rendered, significantly improving performance, particularly during frequent state changes (e.g., updating job application statuses in real time).
- **Routing with React Router:**
  Provides smooth navigation between different views (home, login, dashboards, job details) without full-page reloads, maintaining a single-page application (SPA) experience.
- **Responsive Design:**
  The UI leverages Material-UI and Bootstrap for consistent layouts and responsive elements. Media queries and grid systems allow the interface to adapt across desktops, tablets, and smartphones.
- **Advantages of React over Alternatives:**
  Compared to Angular (which is heavy and opinionated) and Vue (lightweight but less enterprise adoption), React was selected for its balance of performance, large community support, and ease of integration with third-party libraries.

Backend Layer (Node.js + Express.js)

The backend functions as the business logic and API layer.

- **Event-Driven Non-Blocking I/O:**
  Node.js allows simultaneous handling of multiple requests, ensuring scalability when many users apply for jobs or employers post openings at the same time.
- **RESTful APIs with Express.js:**
  Express.js manages routing for APIs such as:

- POST /api/jobs → Create a new job posting.
- GET /api/jobs → Retrieve job listings.
- POST /api/applications → Submit a job application.
- POST /api/auth/login → User authentication.

- **Middleware Architecture:**
  Middleware handles critical tasks such as:
  - Authentication using JWT tokens.
  - Validation of form inputs.
  - Logging API requests for debugging.
  - Error handling to return proper status codes.
- **Business Logic Examples:**
  - Employers can only view applications for jobs they posted.
  - Job seekers cannot access employer-only features like applicant management.
  - Notifications are automatically generated when application statuses are updated.
- **Employer Analytics Dashboard:**
  Middleware handles critical tasks such as:
  - Backend aggregates data for insights on applications, candidate trends, and hiring metrics.

Database Layer (MongoDB + Mongoose)

The database is the foundation of persistence for storing all user and job data.

- **Collections Designed:**
  - **Users:** Stores job seeker data (name, email, skills, resume link).
  - **Employers:** Stores company profiles, job posts, and hiring activity.
  - **Jobs:** Stores job details like title, description, salary, and requirements.
  - **Applications:** Links users to jobs, with statuses like *applied, shortlisted, rejected, accepted*.
  - **Notifications:** Stores messages sent to users and employers.
- **Schema Flexibility:**
  MongoDB's document-oriented approach is better suited than relational SQL databases because job postings and resumes often have varying fields. For example, some resumes may include certifications or portfolio links, while others may not.

- **Mongoose ODM:**
  Adds validation and schema enforcement. For instance, an "Application" entry cannot be stored without userId, jobId, and status.
- **Indexing & Aggregation:**
  MongoDB indexes ensure fast search queries (e.g., jobs by location, salary range). Aggregation pipelines allow analytics such as the number of applicants per employer or most applied job roles.
- **Resume Storage & Analytics:**
  Resumes stored in database; analytics queries for employer dashboard

Deployment Architecture

The system is designed for cloud-native deployment using platforms like AWS, Render, or Heroku.

- **Frontend:** Hosted on Netlify or Vercel.
- **Backend APIs:** Deployed on cloud VMs (e.g., AWS EC2).
- **Database:** Managed on MongoDB Atlas, ensuring automatic backups, replication, and scalability.
- **CI/CD Pipelines:** GitHub Actions or GitLab CI/CD automate code testing, building, and deployment.

2. **Role-Based Access Control (RBAC)**

A critical part of the design is RBAC, ensuring different user groups have tailored experiences.

- **Job Seeker (User):**
  - Build a profile and upload/create resumes.
  - Browse/search/filter jobs by multiple criteria.
  - Apply for jobs and track progress in real time.
  - Access the Interview Preparation Module with quizzes, aptitude tests, and coding practice.
  - See real -time Notifications.
- **Employer:**
  - Create company profiles and job postings.
  - Review candidate applications and resumes.
  - Update application statuses (shortlisted, rejected, hired).

- Manage all postings in a central Employer Dashboard.
- Receive instant notifications for new applications.
- Manage employer Analytics dashboard.

- **Admin (Optional Role):**
  - Monitor activity across the system.
  - Remove fraudulent accounts or job postings.
  - Enforce moderation and maintain platform integrity.

3. **Methodology (SDLC Approach)**

The system was developed using a structured SDLC-inspired methodology:

1. **Requirement Analysis**
   - **Studied job seekers' struggles:** redundant applications, lack of updates, and poor personalization.
   - **Identified employer pain points:** resume overload, communication gaps, and lack of filtering.
   - **Finalized key features:** role-based login, dashboards, resume builder, interview preparation, notifications.
2. **System Design**
   - **Database Schema:** Defined MongoDB collections with fields for jobs, applications, resumes, etc.
   - **Frontend Design:** Created modular React components, e.g., JobCard, ApplicationTracker ,ResumeForm.
   - **Backend Design:** Established APIs and middleware for CRUD operations, validation, and authentication.
3. **Implementation**
   - Built frontend with React + Material-UI.
   - Developed backend APIs with Express.js.
   - Integrated MongoDB via Mongoose.
   - Implemented JWT authentication and bcrypt password hashing.
4. **Testing**
   - Unit Testing: React components (JobCard, LoginForm) and API routes tested individually.

- Integration Testing: Verified seamless interaction between frontend forms and backend APIs.
- Database Testing: Ensured application statuses updated correctly and notifications triggered reliably.
- System Testing: Tested the platform as a whole with multiple roles simultaneously.

5. **Deployment**
   - Frontend deployed to Netlify, backend on AWS EC2, database on MongoDB Atlas.
   - Scalability ensured with database sharding, API load balancing, and frontend lazy loading.

4. Key Features Integrated

- **Real-Time Notifications:** Immediate updates for candidates and employers.
- **Resume Builder:** Provides a simple interface to generate professional resumes.
- **Interview Preparation Module:** Includes coding assessments, aptitude tests, and logical reasoning exercises.
- **Employer Dashboard & Analytics:** Track applications, analyze trends, optimize hiring.
- **Scalability & Security:** Designed to scale for thousands of users while ensuring data integrity with JWT and bcrypt.

5) Results

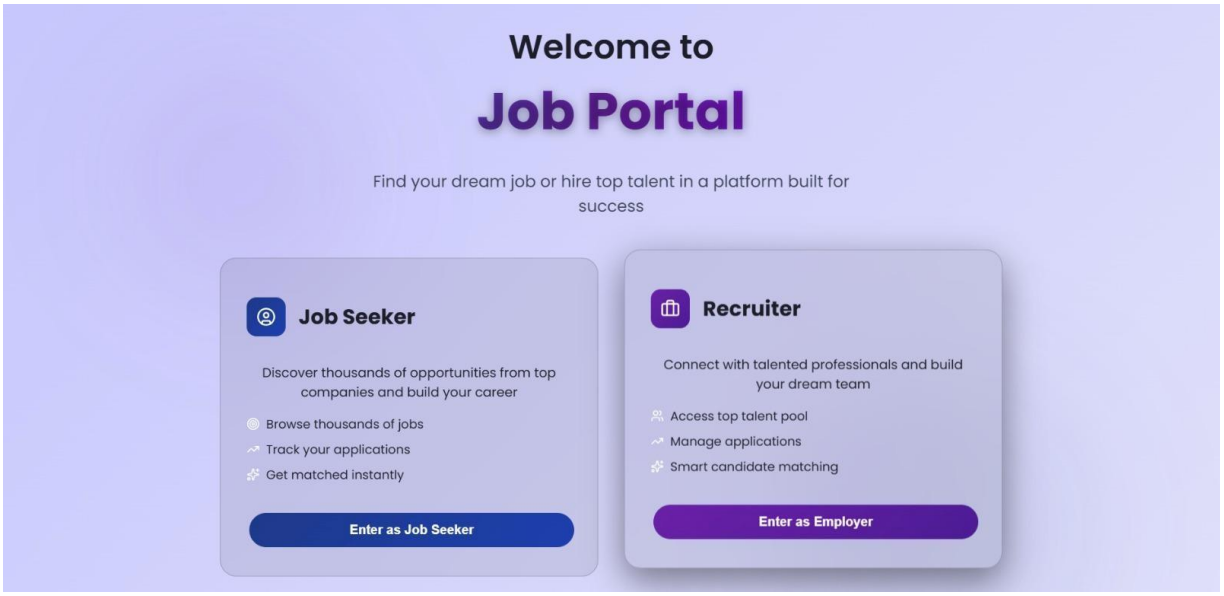


Fig 1. Landing pages welcomes with the role based registration of User and Recruiter

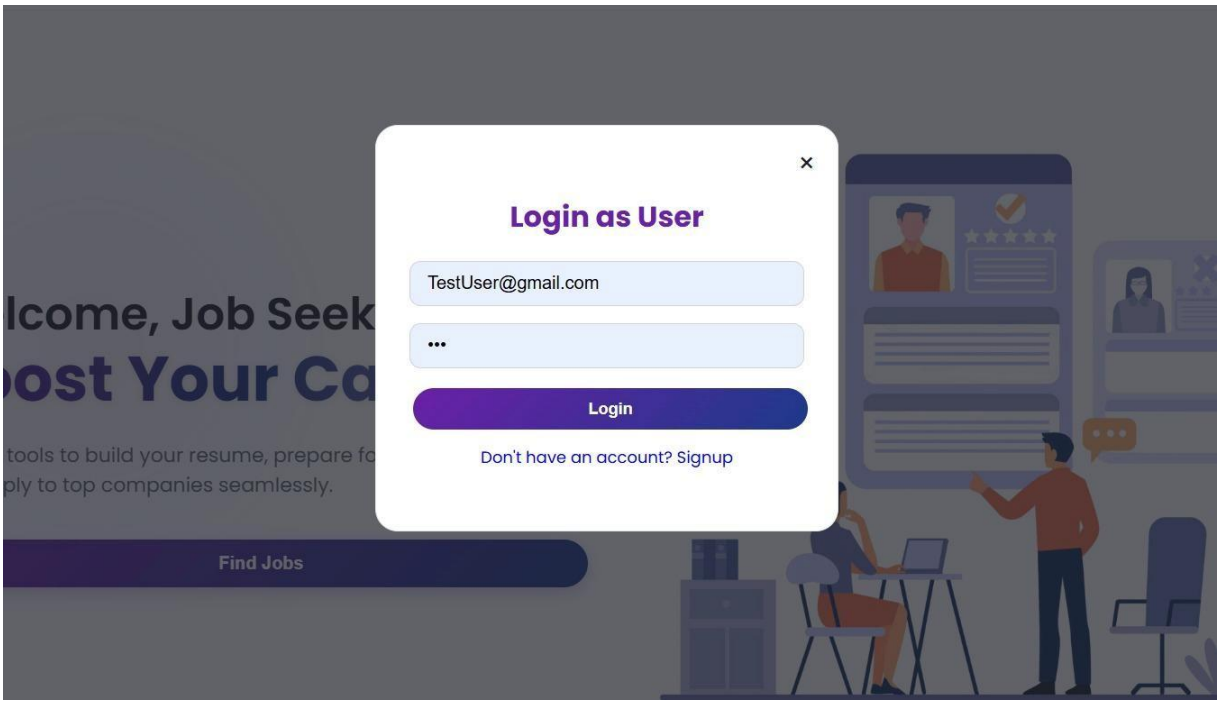


Fig 2. User logs in with his/her credentials

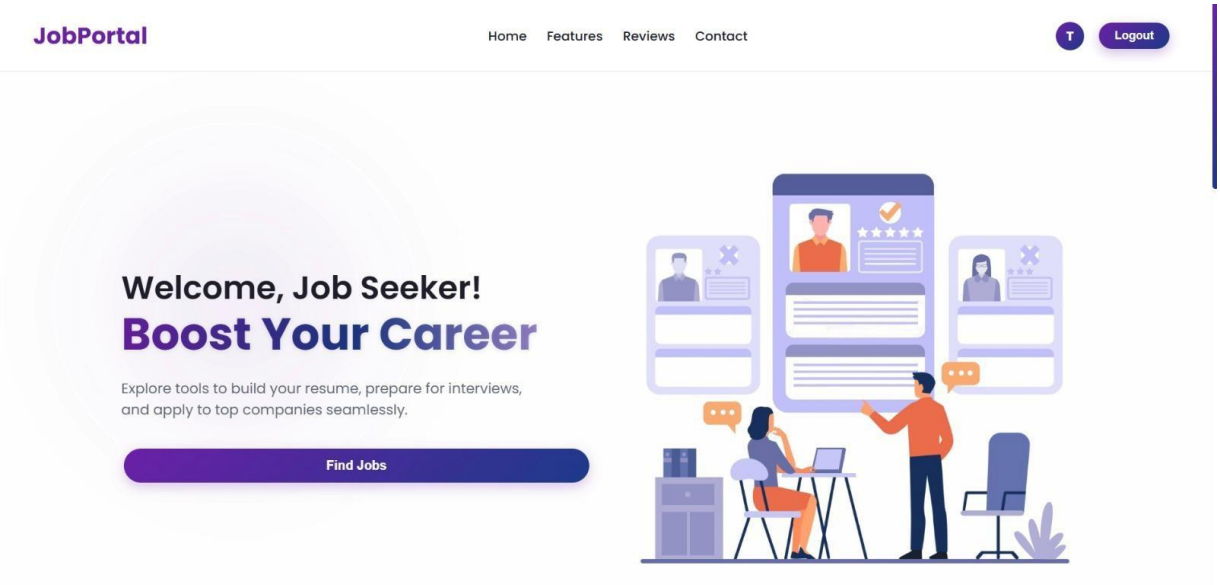


Fig 3. User can find jobs listed by Recruiter

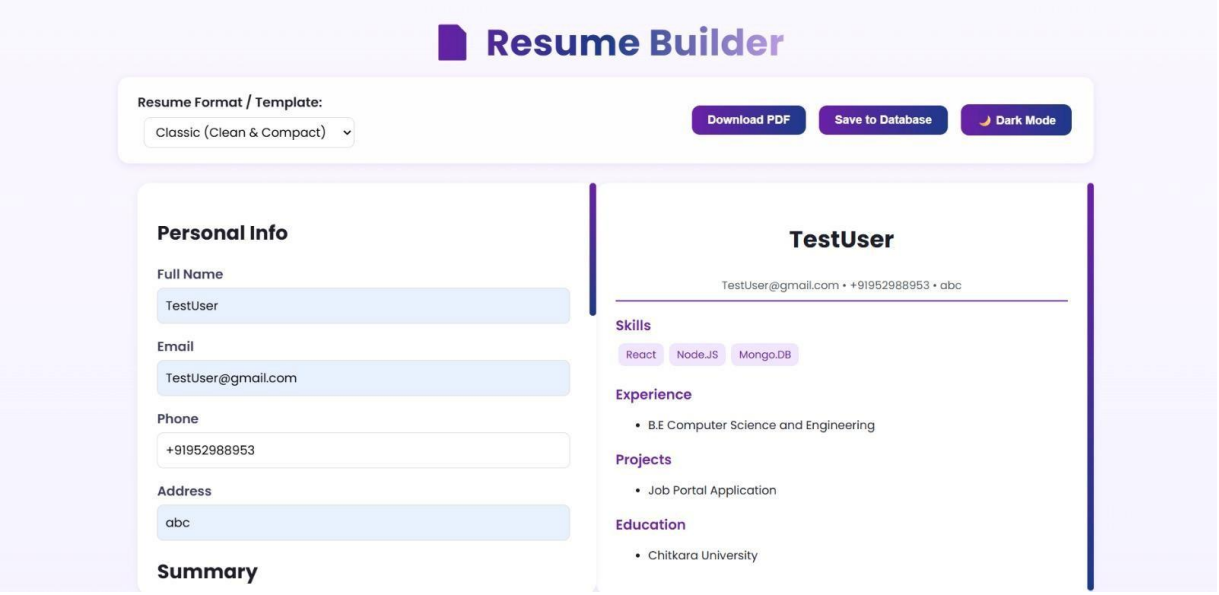


Fig 4. User can access the feature of resume builder, which automatically generates the resume's pdf by formatting them

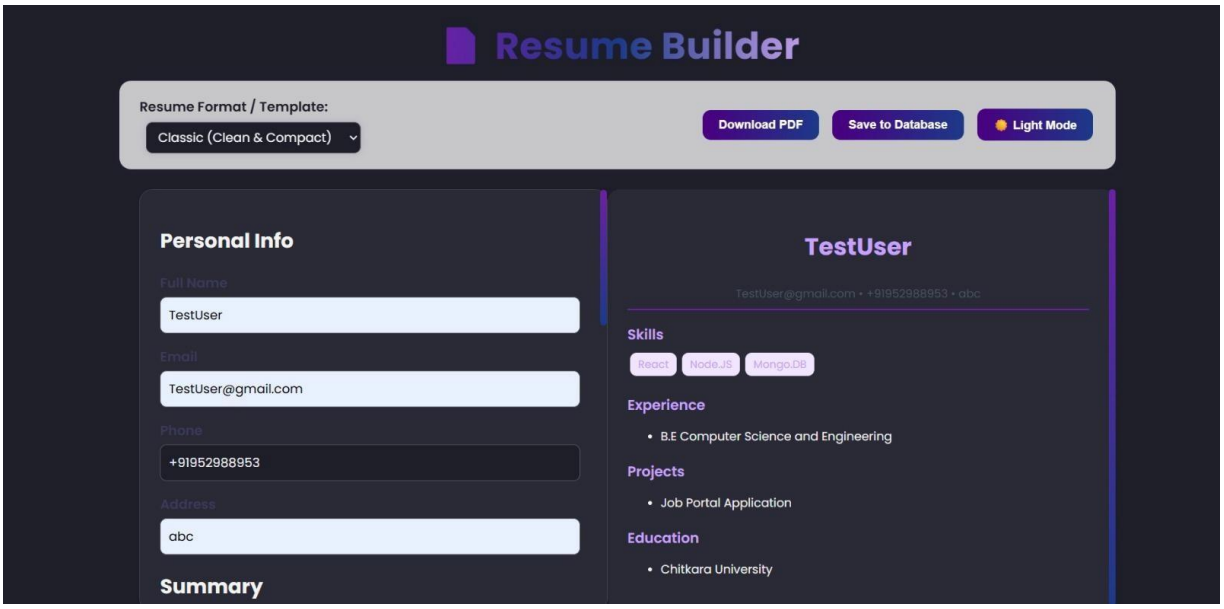


Fig 5. Light and dark modes for better UI experience

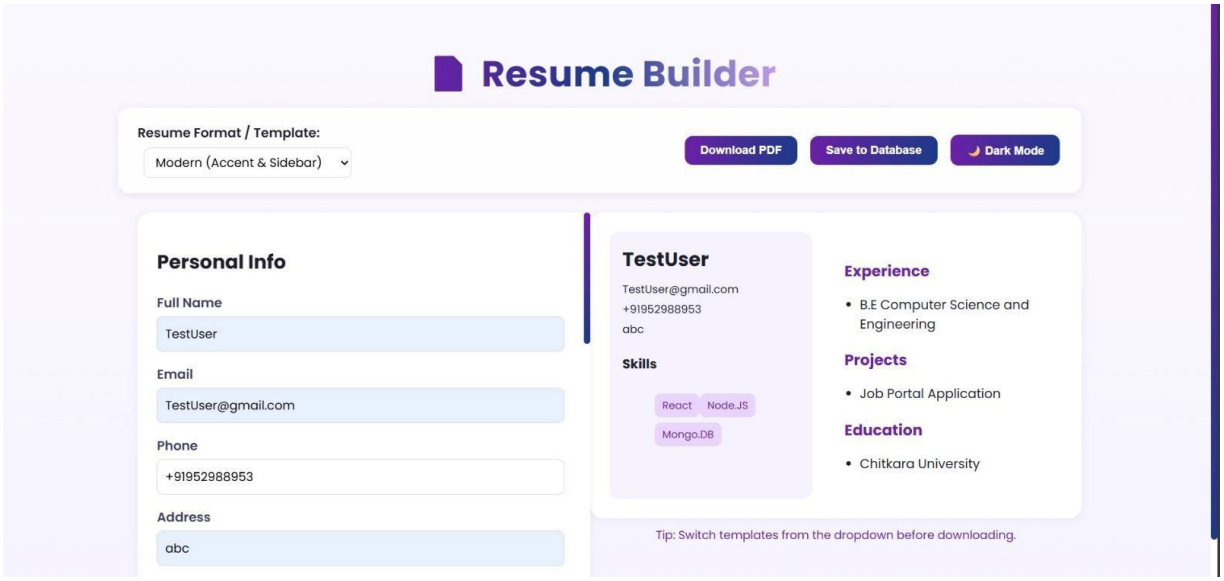


Fig 6. A modern ATS-friendly template, can also be accessed by user along with the classic format

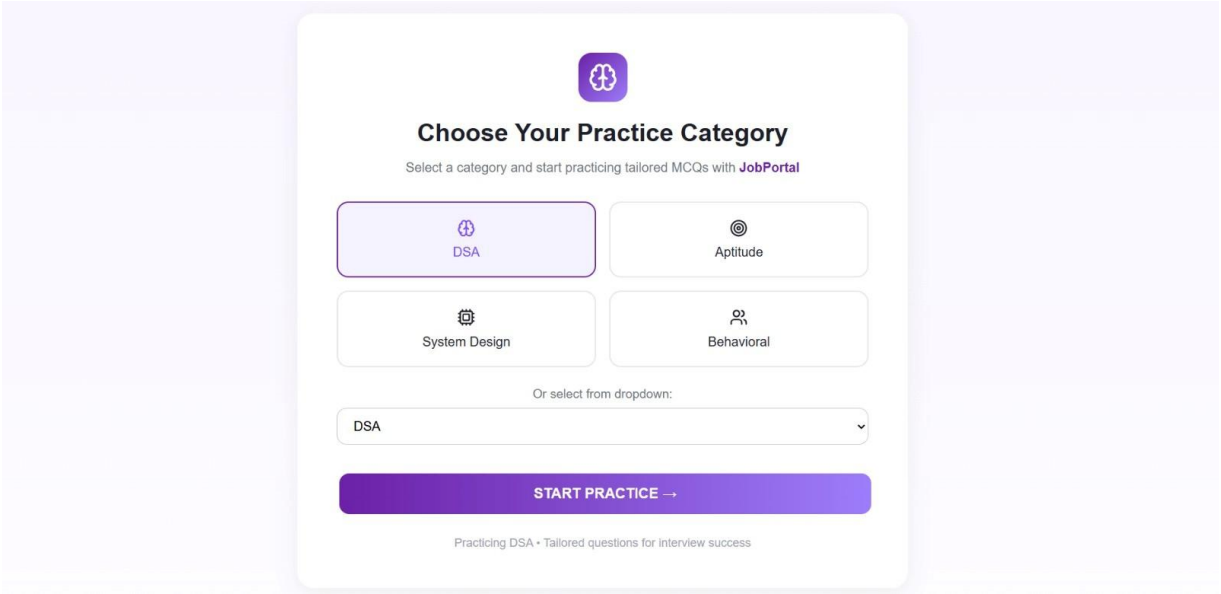


Fig 7. Practice assessments of various categories can be attempted to be prepared

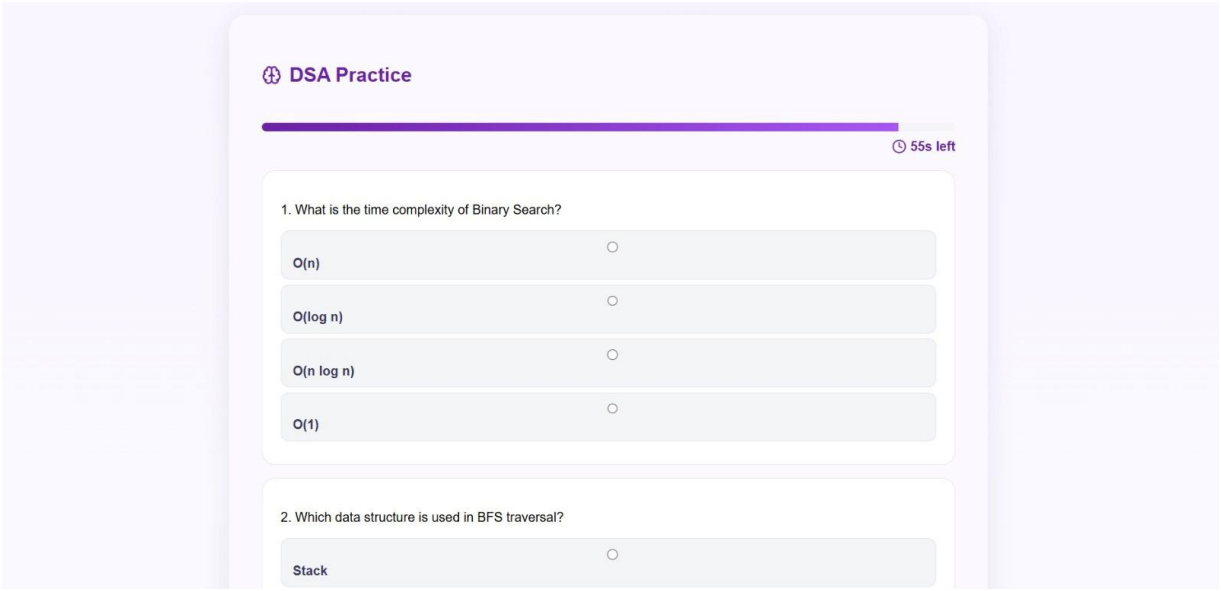


Fig 8. A short quiz with a timer is given to brush up the concepts

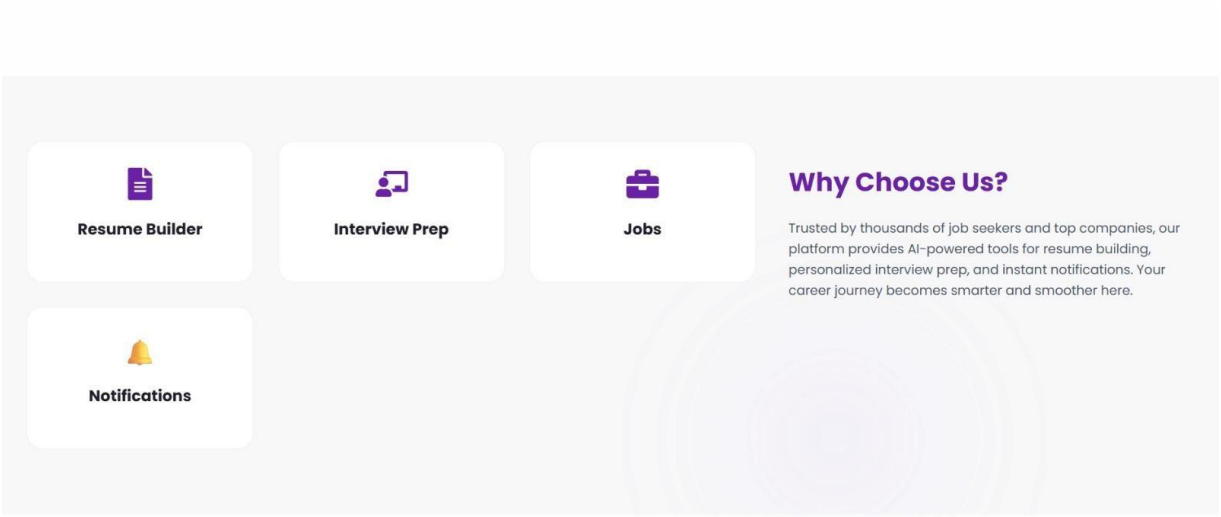


Fig 9. Key features of Job Seeker

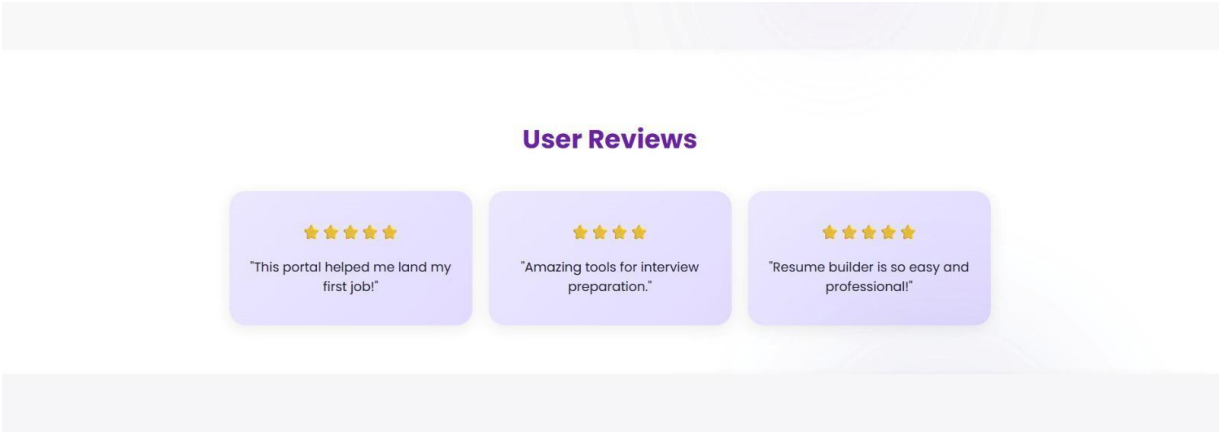


Fig 10. Review Section

Fig 11. Footer Section

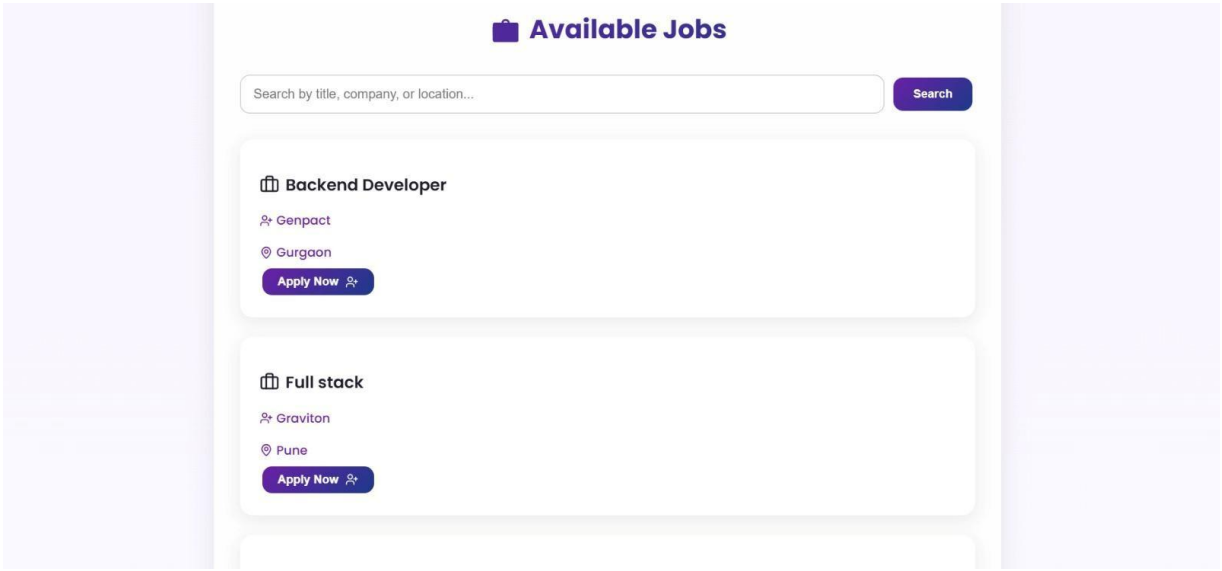


Fig 12. User can seek available jobs that are listed by the recruiter

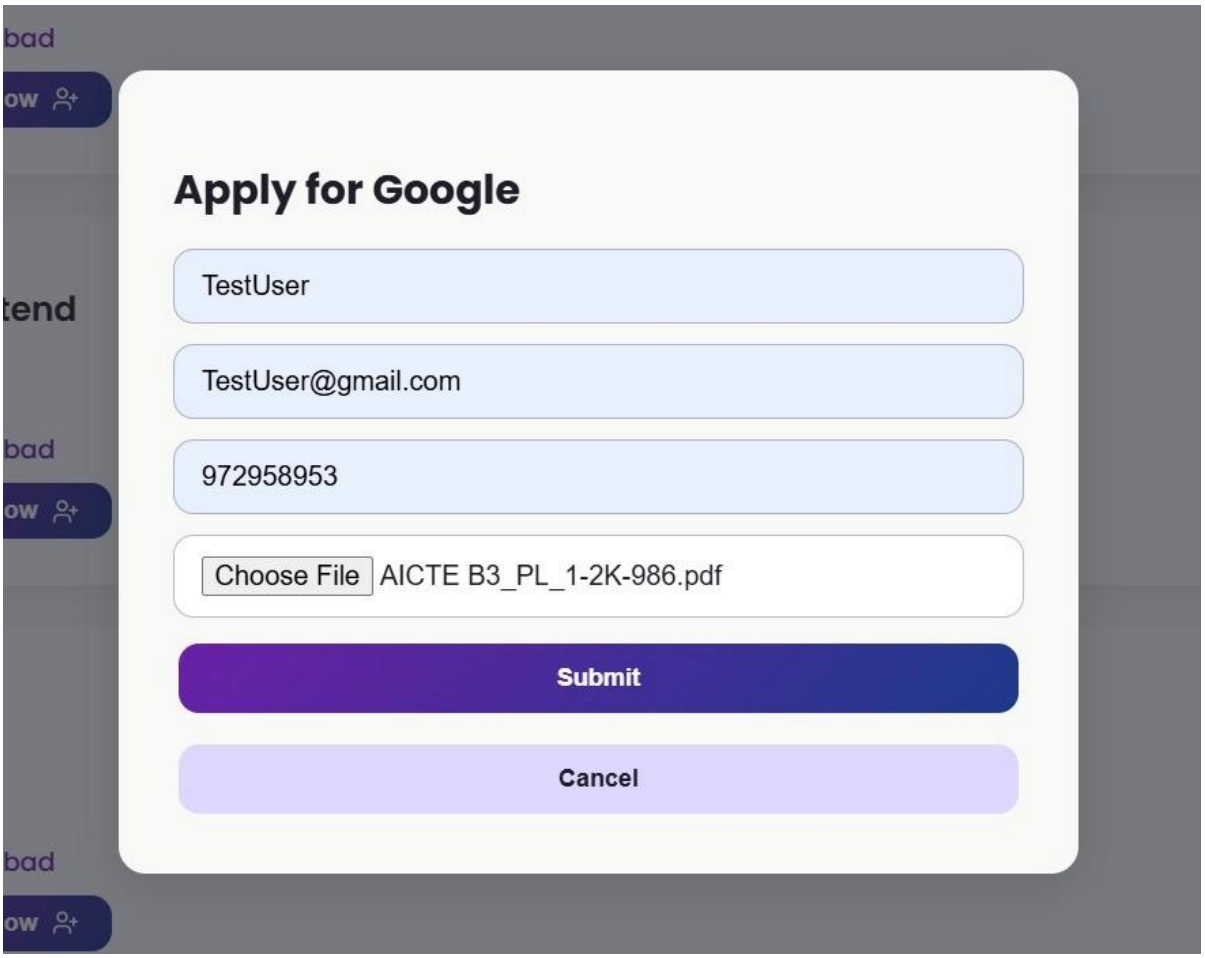


Fig 13. A pop-up form for submission of personal details and resume for Job Seeker

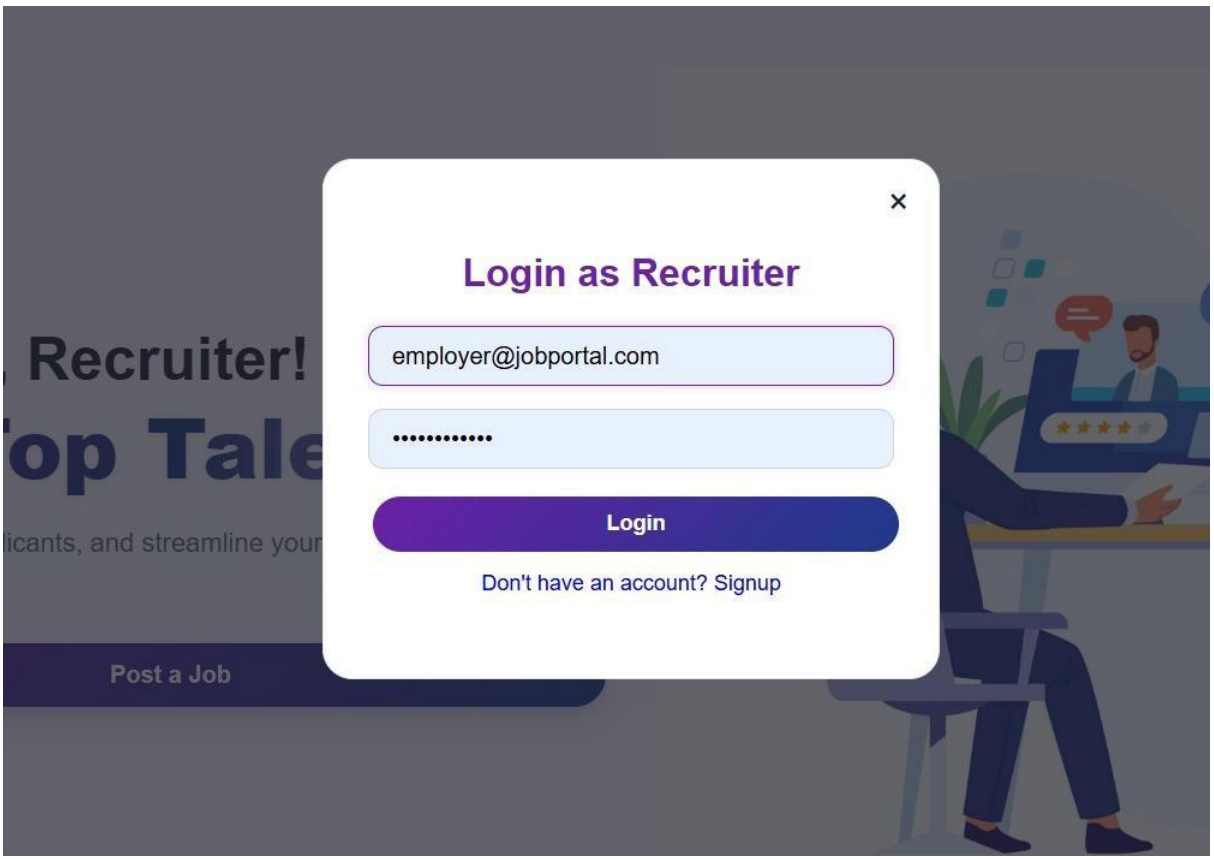


Fig 14. Logging in as Recruiter

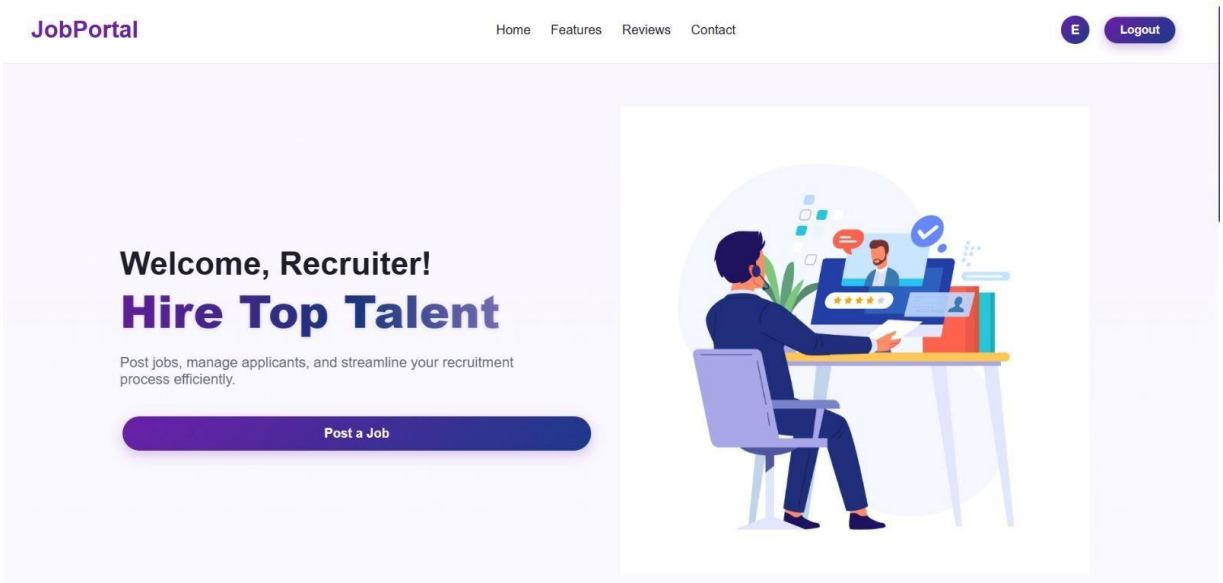


Fig 15. Home page of Recruiter

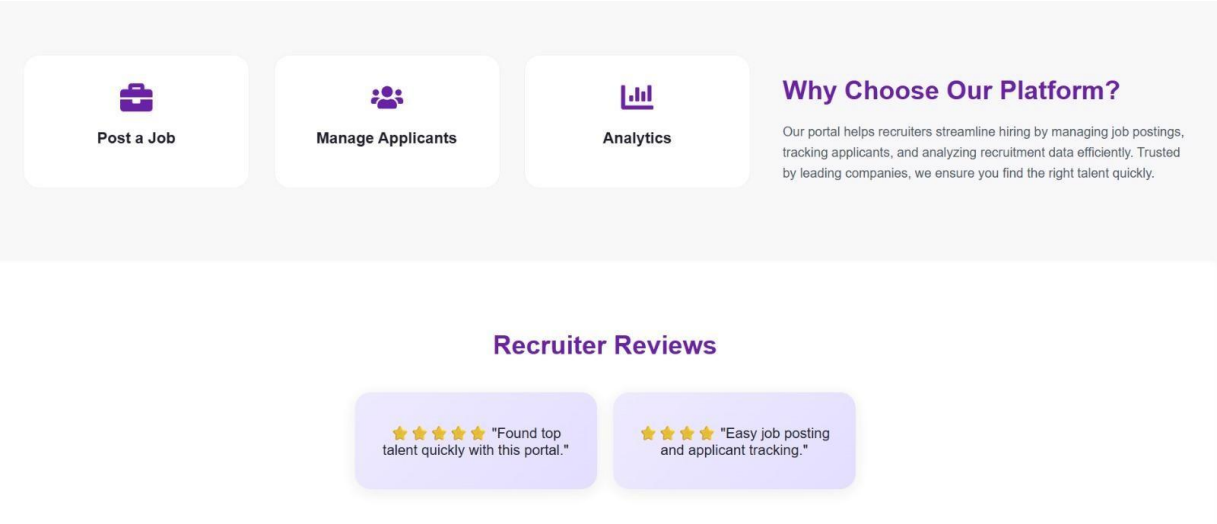


Fig 16. Reviews listed by recruiters as per their experience

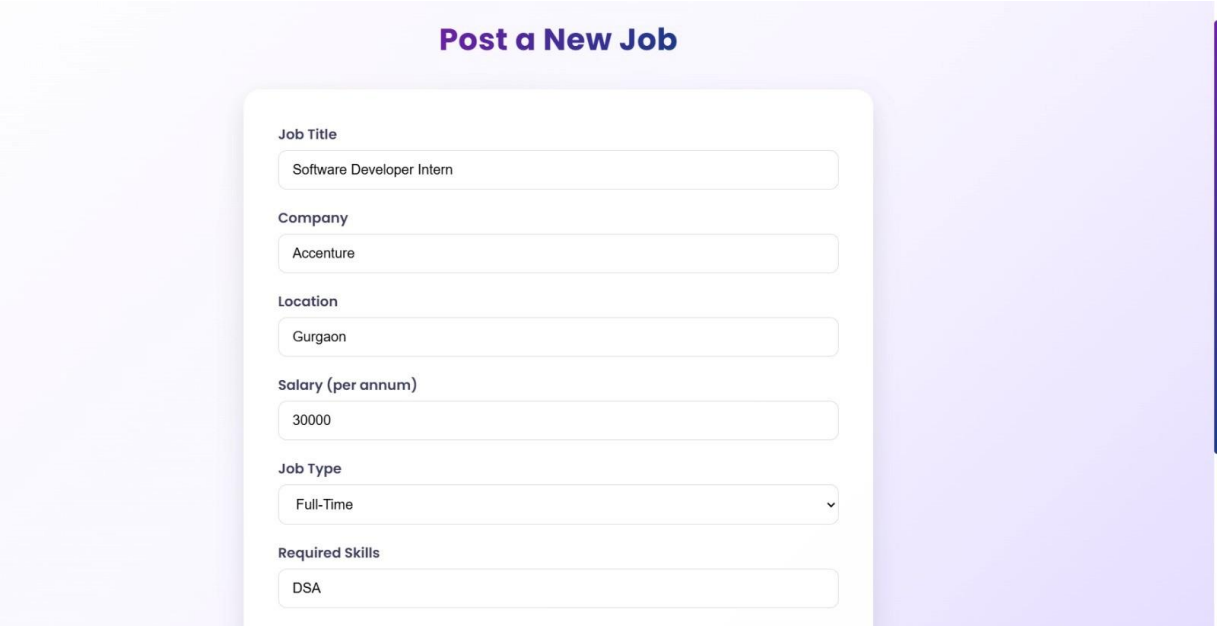


Fig 17. Recruiter can post jobs with job details and other essentials

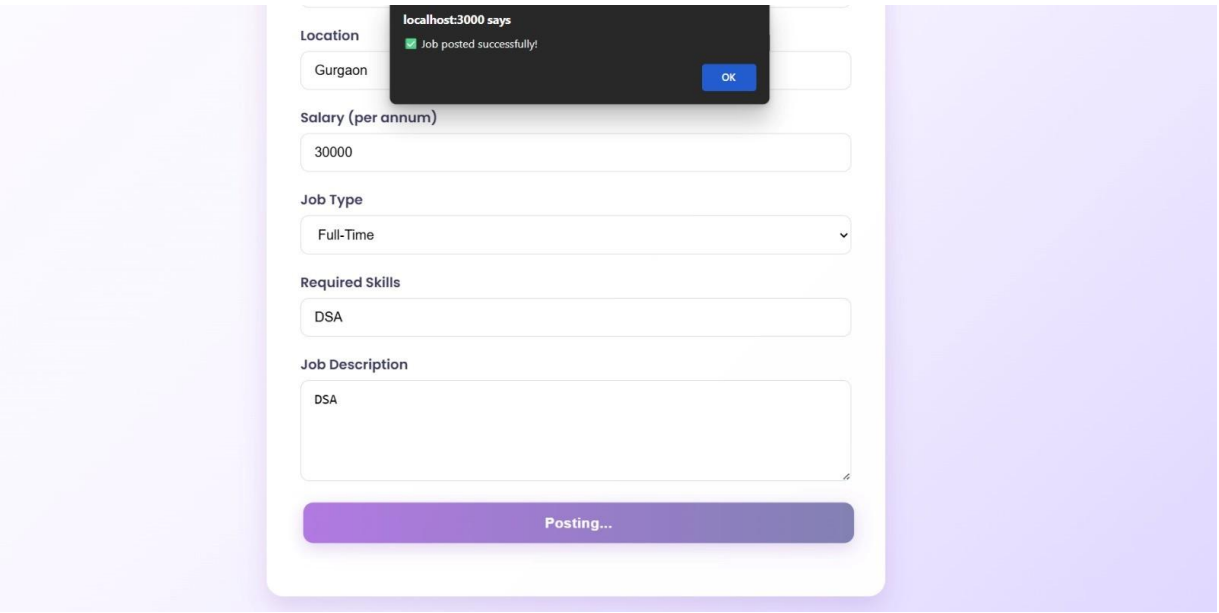


Fig 18. Form for posting jobs

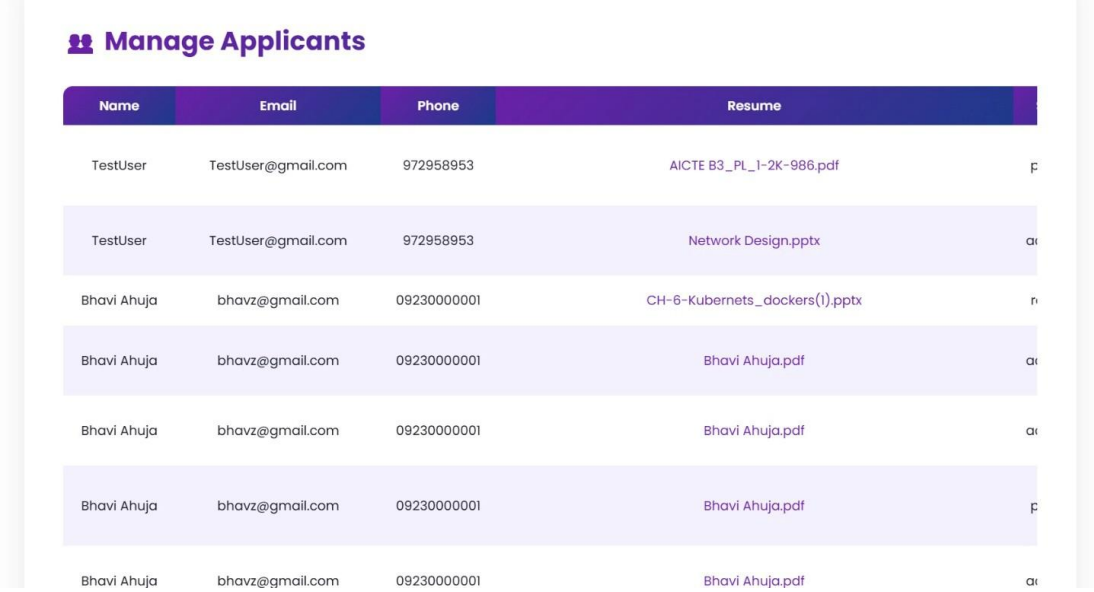


Fig 19. List of all the applicants that applied for a specific job

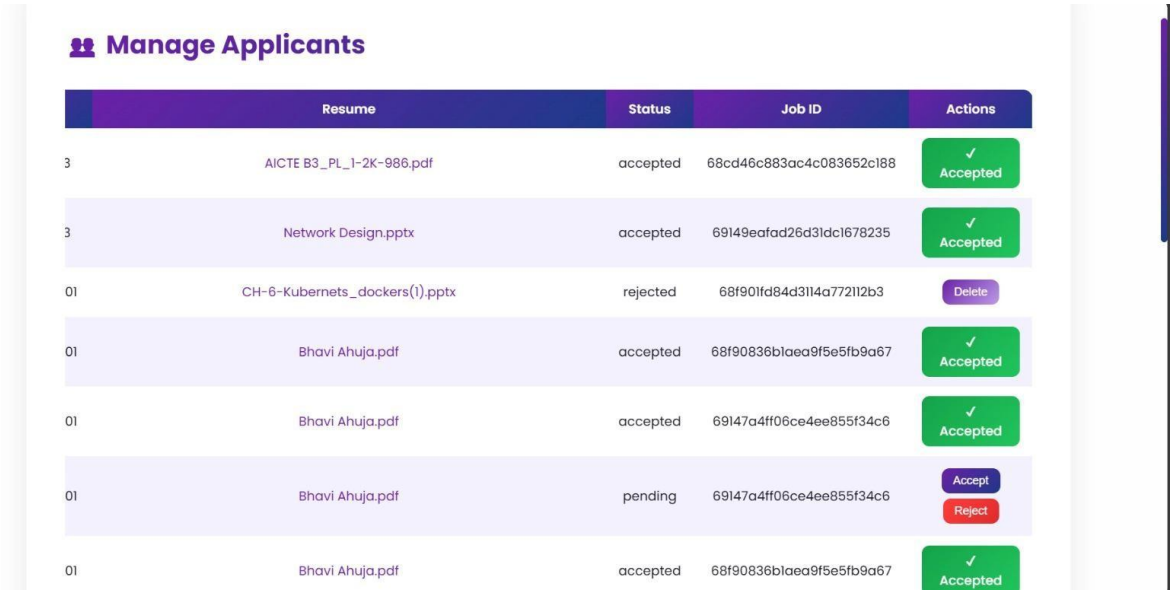


Fig 20. Recruiter can accept/reject the listed requests.

Fig 21. Analytics Page with bar graph and pie chart for Recruiter to view all the data of listed, accepted/rejected users and job details too

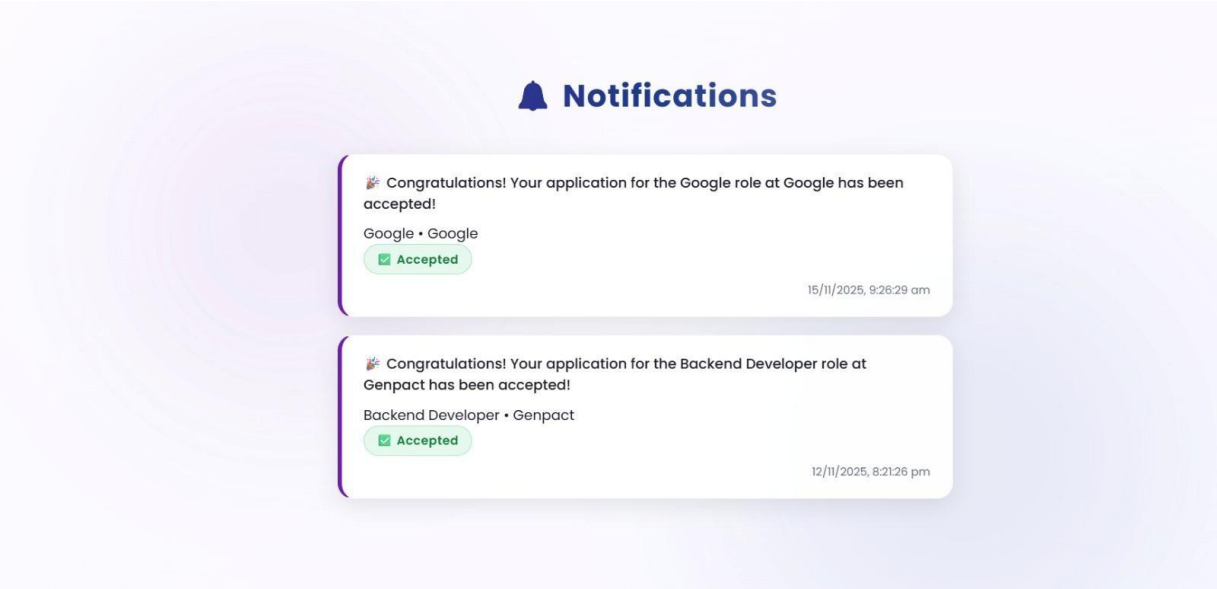
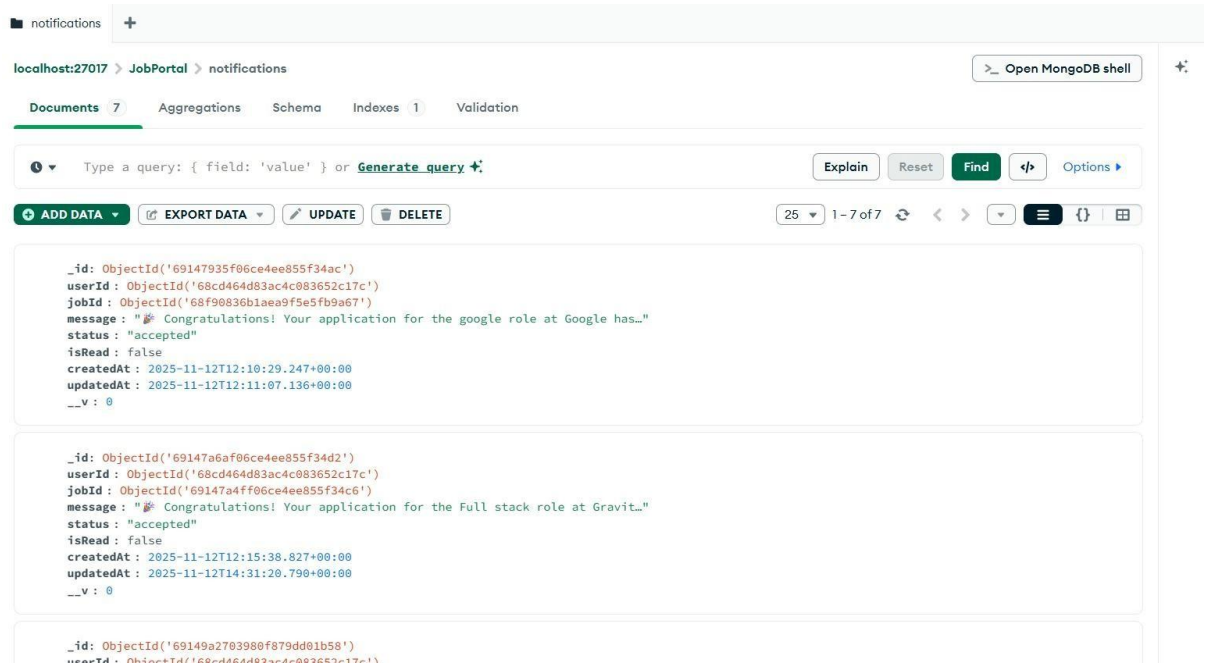
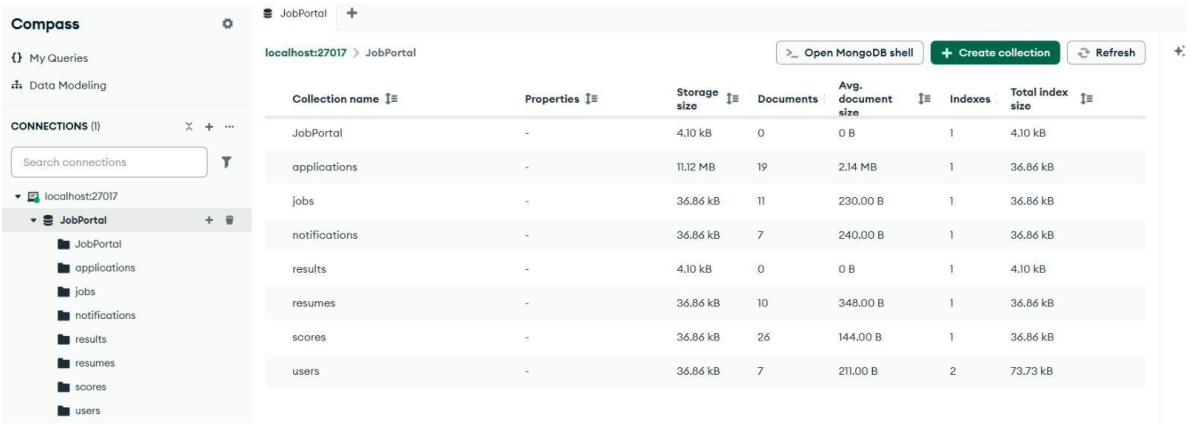
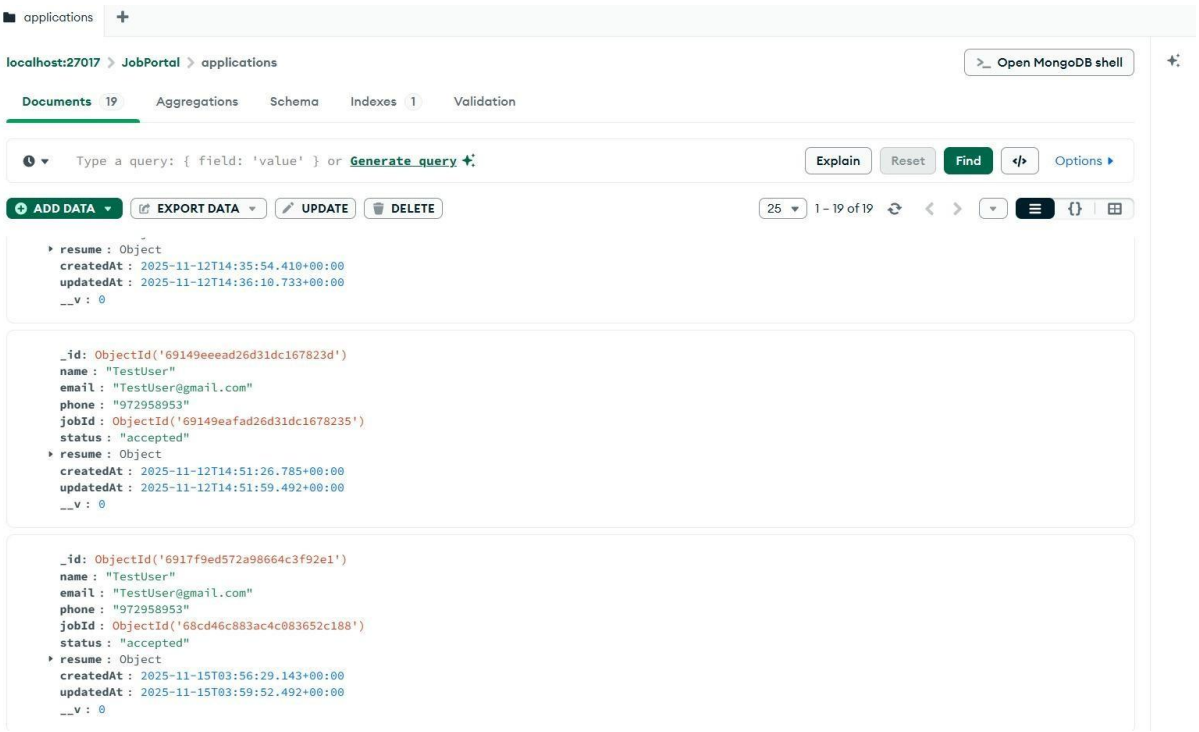


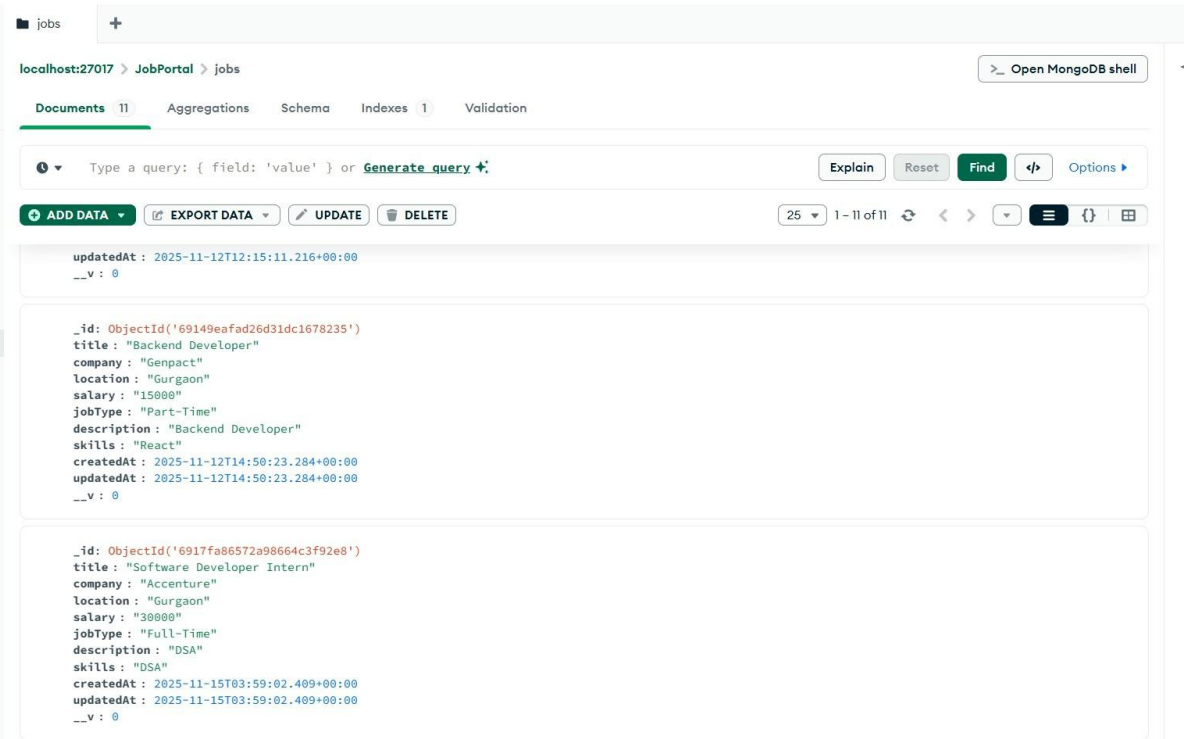
Fig 22. User can view the notifications as per the recruiter's response

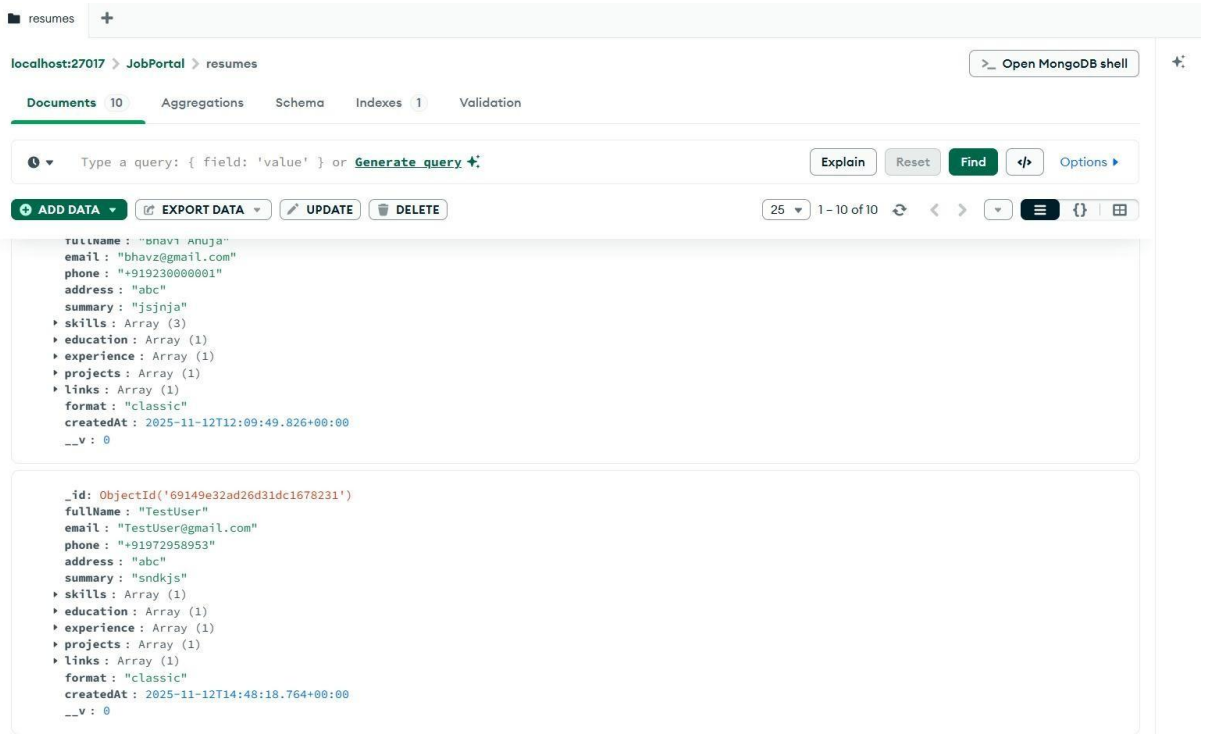
Database Details

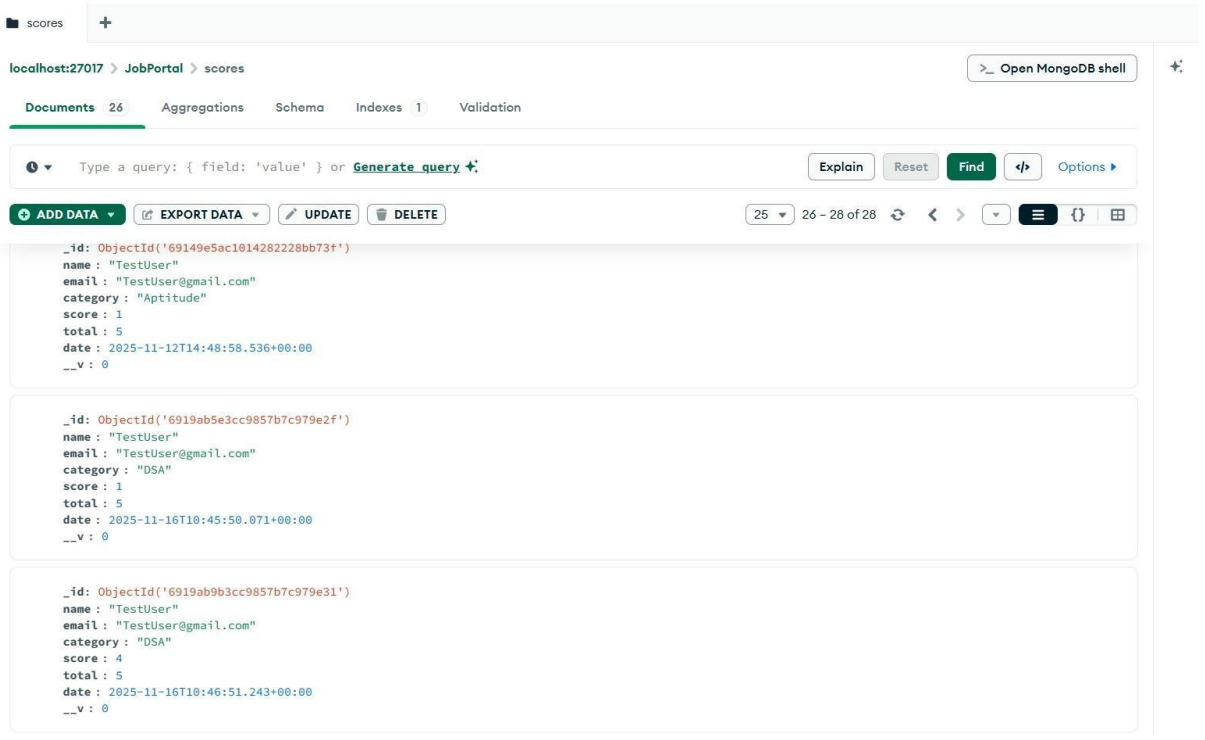


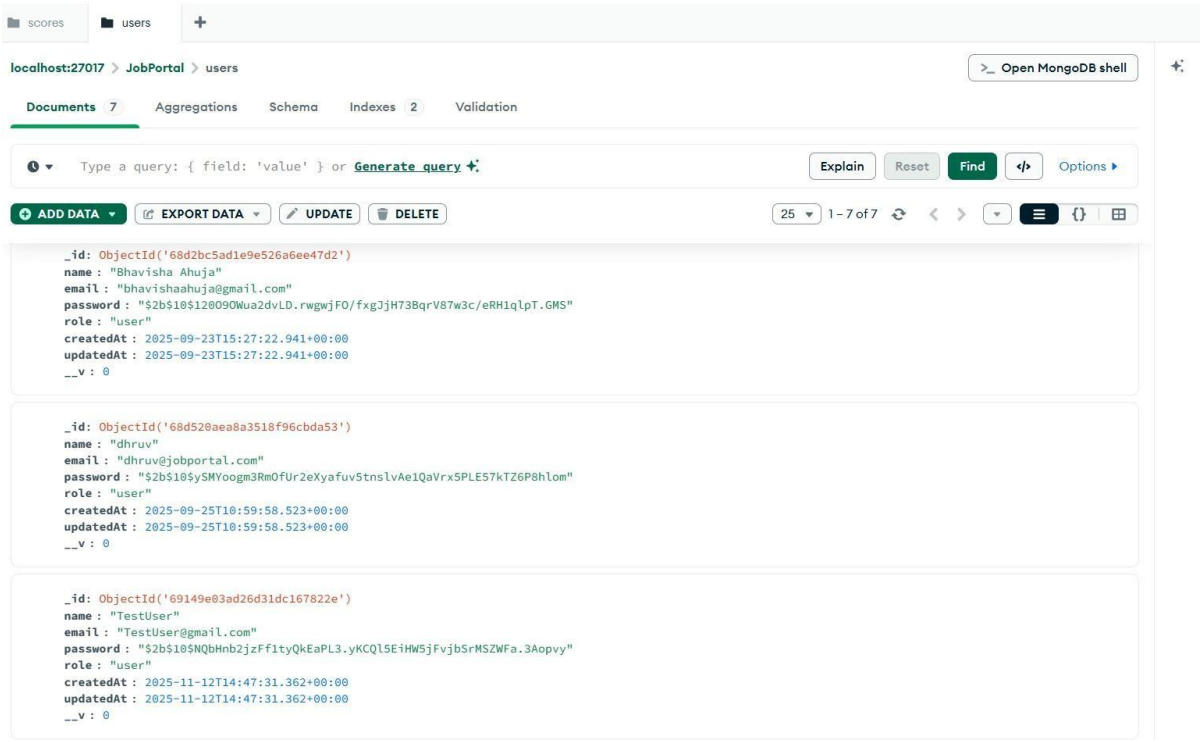



applications
localhost:27017 > JobPortal > applications
Open MongoDB shell
Documents 19
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 - 19 of 19
resume : Object
createdAt : 2025-11-12T14:35:54.410+00:00
updatedAt : 2025-11-12T14:36:10.733+00:00
__v : 0
_id: ObjectId('69149eeead26d31dc167823d')
name : "TestUser"
email : "TestUser@gmail.com"
phone : "972958953"
jobId : ObjectId('69149eafad26d31dc1678235')
status : "accepted"
resume : Object
createdAt : 2025-11-12T14:51:26.785+00:00
updatedAt : 2025-11-12T14:51:59.492+00:00
__v : 0
_id: ObjectId('6917f9ed572a98664c3f92e1')
name : "TestUser"
email : "TestUser@gmail.com"
phone : "972958953"
jobId : ObjectId('68cd46c883ac4c083652c188')
status : "accepted"
resume : Object
createdAt : 2025-11-15T03:56:29.143+00:00
updatedAt : 2025-11-15T03:59:52.492+00:00
__v : 0


jobs
localhost:27017 > JobPortal > jobs
Open MongoDB shell
Documents 11
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 - 11 of 11
updatedAt : 2025-11-12T12:15:11.216+00:00
__v : 0
_id: ObjectId('69149eafad26d31dc1678235')
title : "Backend Developer"
company : "Genpact"
location : "Gurgaon"
salary : "15000"
jobType : "Part-Time"
description : "Backend Developer"
skills : "React"
createdAt : 2025-11-12T14:50:23.284+00:00
updatedAt : 2025-11-12T14:50:23.284+00:00
__v : 0
_id: ObjectId('6917fa86572a98664c3f92e8')
title : "Software Developer Intern"
company : "Accenture"
location : "Gurgaon"
salary : "30000"
jobType : "Full-Time"
description : "DSA"
skills : "DSA"
createdAt : 2025-11-15T03:59:02.409+00:00
updatedAt : 2025-11-15T03:59:02.409+00:00
__v : 0


resumes
localhost:27017 > JobPortal > resumes
Open MongoDB shell
Documents 10
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 – 10 of 10
fullName : "Bhavi Anuja"
email : "bhavz@gmail.com"
phone : "+919230000001"
address : "abc"
summary : "jsjnja"
skills : Array (3)
education : Array (1)
experience : Array (1)
projects : Array (1)
links : Array (1)
format : "classic"
createdAt : 2025-11-12T12:09:49.826+00:00
__v : 0
_id: ObjectId('69149e32ad26d31dc1678231')
fullName : "TestUser"
email : "TestUser@gmail.com"
phone : "+91972958953"
address : "abc"
summary : "sndkjs"
skills : Array (1)
education : Array (1)
experience : Array (1)
projects : Array (1)
links : Array (1)
format : "classic"
createdAt : 2025-11-12T14:48:18.764+00:00
__v : 0


scores
localhost:27017 > JobPortal > scores
Open MongoDB shell
Documents 26
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
26 – 28 of 28
_id: ObjectId('69149e5ac1014282228bb73f')
name : "TestUser"
email : "TestUser@gmail.com"
category : "Aptitude"
score : 1
total : 5
date : 2025-11-12T14:48:58.536+00:00
__v : 0
_id: ObjectId('6919ab5e3cc9857b7c979e2f')
name : "TestUser"
email : "TestUser@gmail.com"
category : "DSA"
score : 1
total : 5
date : 2025-11-16T10:45:50.071+00:00
__v : 0
_id: ObjectId('6919ab9b3cc9857b7c979e31')
name : "TestUser"
email : "TestUser@gmail.com"
category : "DSA"
score : 4
total : 5
date : 2025-11-16T10:46:51.243+00:00
__v : 0


scores
users
localhost:27017 > JobPortal > users
Open MongoDB shell
Documents 7
Aggregations
Schema
Indexes 2
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 - 7 of 7
_id: ObjectId('68d2bc5ad1e9e526a6ee47d2')
name : "Bhavisha Ahuja"
email : "bhavishaahuja@gmail.com"
password : "$2b$10$120O9OWua2dvLD.rwgwjFO/fxgJjH73BqrV87w3c/eRH1qlpT.GMS"
role : "user"
createdAt : 2025-09-23T15:27:22.941+00:00
updatedAt : 2025-09-23T15:27:22.941+00:00
__v : 0
_id: ObjectId('68d520aea8a3518f96cbda53')
name : "dhruv"
email : "dhruv@jobportal.com"
password : "$2b$10$ySMYoogm3RmOfUr2eXyafuv5tnslvAe1QaVrx5PLE57kTZ6P8hlom"
role : "user"
createdAt : 2025-09-25T10:59:58.523+00:00
updatedAt : 2025-09-25T10:59:58.523+00:00
__v : 0
_id: ObjectId('69149e03ad26d31dc167822e')
name : "TestUser"
email : "TestUser@gmail.com"
password : "$2b$10$NQbHnb2jzFf1tyQkEaPL3.yKCQl5EiHW5jFvjbSrMSZWFa.3Aopvy"
role : "user"
createdAt : 2025-11-12T14:47:31.362+00:00
updatedAt : 2025-11-12T14:47:31.362+00:00
__v : 0