

**Tribhuvan University**

**Faculty of Humanities and Social Science**

**Orchid International College**

**A PROJECT REPORT**

**ON  
JOB PORTAL SYSTEM**

**USING RULE-BASED WEIGHTED SKILL MATCHING ALGORITHM**

**Submitted to**

**Department of Computer Application**

**Orchid International College**

***In partial fulfillment of the requirements for the Bachelors in Computer Application***

**Submitted by**

Smriti Khadka (93902090)

July 2025

Under the Supervision of

**Mr. Shivram Maharjan**



**Tribhuvan University**

**Faculty of Humanities and Social Science**

**Orchid International College**

# SUPERVISOR’S RECOMMENDATION

I hereby recommend that this project prepared under my supervision by Smriti Khadka entitled **“Job Portal System”** in partial fulfillment of the requirements for the degree of Bachelor of Computer Application is recommended for the final evaluation.

**SIGNATURE**

Mr. Shivram Maharjan

**SUPERVISOR**

Faculty of Humanities and Social Science

Bijayachowk, Gaushala, Kathmandu



**Tribhuvan University**

**Faculty of Humanities and Social Science**

**Orchid International College**

# LETTER OF APPROVAL

This is to certify that this project prepared by Smriti Khadka entitled “**Job portal System”** in partial fulfillment of the requirements for the degree of Bachelor in Computer Application has been evaluated. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

|  |  |
| --- | --- |
| **Signature of Supervisor**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Mr. Shivram Maharjan  Department of IT  Orchid International College | **Signature of HOD/ Coordinator**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Mr. Dhiraj Kumar Jha  Head of Department  Department of IT |
| **Signature of Internal Examiner    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Shrilata Wagle  Full-time Faculty  Orchid International College | **Signature of External Examiner    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

# ABSTRACT

A robust and scalable Job Portal System was developed to streamline digital recruitment, providing comprehensive support for job seekers, employers, and administrators through role based access, profile management, job postings, applications, and interview scheduling. The platform featured dynamic dashboards for tracking application progress, job analytics, and career growth, alongside real time messaging, secure document uploads, and workspace-based collaboration to facilitate seamless communication and workflow management. Built using Next.js, React, Node.js, TypeScript, and MongoDB, it utilized RESTful APIs for authentication and data handling, JWT for secure user verification, and state management for real time updates and profile tracking. The system delivered a responsive, modular, and user-friendly solution capable of managing collaborative recruitment at scale, while demonstrating expertise in full stack development, secure communication protocols, role based access control, and efficient real time data handling.

***Keywords****: Job Portal System, RESTful API, MongoDB, Next.js, React, Node.js, TypeScript, JWT Authentication, Role-based Access Control, Document Management System*

# ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to everyone who supported me throughout this project, with my deepest appreciation going to Er. Dhiraj Kumar Jha for his unwavering guidance, encouragement, and invaluable mentorship, which greatly shaped the direction and quality of my work. I am equally thankful to Mr. Shivram Maharjan for his continuous support and motivation, whose guidance and assistance provided me with the confidence and clarity I needed at every stage. To both of them, I remain sincerely grateful for their inspiration and support.

**Smriti Khadka (93902090)**

# TABLE OF CONTENT

[SUPERVISOR’S RECOMMENDATION i](#_Toc214350849)

[LETTER OF APPROVAL ii](#_Toc214350850)

[ABSTRACT iii](#_Toc214350851)

[ACKNOWLEDGEMENT iv](#_Toc214350852)

[TABLE OF CONTENT v](#_Toc214350853)

[LIST OF FIGURES vii](#_Toc214350854)

[LIST OF TABLES viii](#_Toc214350855)

[LIST OF ABBREVIATIONS ix](#_Toc214350856)

[Chapter 1: Introduction 1](#_Toc214350857)

[1.1 Introduction 1](#_Toc214350858)

[1.2 Problem Statement 1](#_Toc214350859)

[1.3 Objectives 2](#_Toc214350860)

[1.4 Scope and limitation 2](#_Toc214350861)

[1.4.1 Scope of System 3](#_Toc214350862)

[1.4.2 Limitation of Existing System 3](#_Toc214350863)

[1.5 Report Organization 3](#_Toc214350864)

[Chapter 2: Background Study and Literature Review 4](#_Toc214350865)

[2.1 Background Study 4](#_Toc214350866)

[2.2 Literature Review 4](#_Toc214350867)

[Chapter 3: System Analysis and Design 5](#_Toc214350868)

[3.1 System Analysis 5](#_Toc214350869)

[3.1.1 Requirement Analysis 6](#_Toc214350870)

[3.1.2 Feasibility Study 11](#_Toc214350871)

[3.1.3 Class Diagram 12](#_Toc214350872)

[3.1.4 Sequence Diagram 14](#_Toc214350873)

[3.1.5 Activity Diagram 16](#_Toc214350874)

[3.2 System Design 18](#_Toc214350875)

[3.2.1 Sequence Diagram 18](#_Toc214350876)

[3.2.2 Database Design 21](#_Toc214350877)

[3.2.3 Restful API Workflow 29](#_Toc214350878)

[3.3 Algorithm Details 30](#_Toc214350879)

[Chapter 4: Implementation and Testing 34](#_Toc214350880)

[4.1 Implementation 34](#_Toc214350881)

[4.1.1 Tools Used 34](#_Toc214350882)

[4.1.2 Implementation Detail of Modules 36](#_Toc214350883)

[4.2 Testing 37](#_Toc214350884)

[4.2.1 Test Cases for Unit Testing 37](#_Toc214350885)

[4.2.2 Test Cases for System Testing 40](#_Toc214350886)

[Chapter 5: Conclusion and Future Recommendation 43](#_Toc214350887)

[5.1 Lesson Learnt / Outcome 43](#_Toc214350888)

[5.2 Conclusion 43](#_Toc214350889)

[5.3 Future Enhancement 44](#_Toc214350890)

[REFERENCES 45](#_Toc214350891)

[APPENDICES 46](#_Toc214350892)

# LIST OF FIGURES

[Figure 3.1 Iterative Model 5](#_Toc214350837)

[Figure 3.2 Use Case Diagram of Job portal system 9](#_Toc214350838)

[Figure 3.3 Gantt Chart of Job Portal System 12](#_Toc214350839)

[Figure 3.4 Class Diagram of Job Portal System 13](#_Toc214350840)

[Figure 3.5 Sequence Diagram for User Registration and Job posting 14](#_Toc214350841)

[Figure 3.6 Activity Diagram 16](#_Toc214350842)

[Figure 3.7 Sequence Diagram for Admin User management 18](#_Toc214350843)

[Figure 3.8 Sequence Diagram for Chat and Conversation flow 20](#_Toc214350844)

[Figure 3.9 Database Design 29](#_Toc214350845)

[Figure 3.10 Restful API Workflow 30](#_Toc214350846)

[Figure 3.11 Working of Rule-Based Weighted Skill Matching Algorithm 31](#_Toc214350847)

[Figure 4.1 Implementation Detail of Chat Module 37](#_Toc214350848)

# LIST OF TABLES

[Table 3.1 Use Case Identifier - UC001 – Create and Publish Job Posting 10](#_Toc214350583)

[Table 3.2 Use Case Identifier - UC002 – Submit Job Application 10](#_Toc214350584)

[Table 4.1 Implementation Detail of Modules 36](#_Toc214350585)

[Table 4.2 Test Case 001 - Signup page 38](#_Toc214350586)

[Table 4.3 Test Case 002 - Login Page 39](#_Toc214350587)

[Table 4.4 Test Case 003 – Post Job 41](#_Toc214350588)

[Table 4.5 Test Case 004 – Apply Job 42](#_Toc214350589)

# LIST OF ABBREVIATIONS

**API** : Application Programming Interface

**AES** : Advanced Encryption Standard

**ATS** : Applicant Tracking System

**BSON** : Binary JSON (MongoDB format)

**CI/CD** : Continuous Integration / Continuous Deployment

**CL** : Cover Letter

**CRUD** : Create, Read, Update, Delete

**CV** : Curriculum Vitae

**E2E** : End-to-End (testing)

**ISR** : Incremental Static Regeneration (Next.js)

**JD** : Job Description

**ODM** : Object Document Mapper (MongoDB)

**RBAC** : Role-Based Access Control

**SGA** : Skill Gap Analysis (training recommendations feature)

**SSR** : Server-Side Rendering

**JWT** : JSON Web Token

# Introduction

## Introduction

A next-generation job portal system was developed to reimagine the recruitment process through a centralized, digital platform that connects job seekers, employers, and administrators. The solution was built using a modern technology stack, including Next.js, React, Node.js, Express.js, and MongoDB, ensuring high performance, responsiveness, and cross-device accessibility. The platform empowered candidates to create detailed profiles, discover suitable opportunities, and track their applications in real time, while enabling employers to post vacancies, manage applicant pipelines, and streamline decision-making.

Structured workflows supported core operations such as profile creation, job posting, application submission, role-based permissions, and communication between stakeholders. Role hierarchies comprising Admin, Employer, and Job Seeker were carefully designed to enforce secure and efficient access control. Job seekers could showcase their skills and monitor progress, employers could filter and evaluate candidates effectively, and administrators oversaw the platform with advanced analytics and management tools.

The system architecture was modular, with RESTful APIs managing authentication, job data, applications, and secure communication channels. Real-time updates and notifications were enabled through WebSockets and React state management, ensuring a seamless user experience. Security was a key priority, with robust encryption protocols safeguarding sensitive user data and ensuring compliance with privacy standards.

An interactive and mobile-friendly user interface enhanced usability through dashboards tailored to different user roles. Advanced search and filtering made job discovery efficient, while dynamic progress tracking offered transparency in the application process. Comprehensive error handling, system reliability, and scalability positioned the platform as a robust solution capable of adapting to the evolving demands of the digital recruitment landscape.

## Problem Statement

Many job seekers and employers face significant challenges in efficiently navigating the recruitment process. Traditional hiring methods often involve lengthy paperwork, fragmented workflows, and limited visibility into application statuses, resulting in delays, mismanagement, and missed opportunities. Job seekers frequently struggle to discover relevant job openings, present their qualifications effectively, and track multiple applications, while employers encounter difficulties in managing large volumes of applications, identifying qualified candidates, and maintaining organized recruitment workflows.

Existing recruitment platforms often lack integration, advanced filtering, and real-time updates, which can hinder efficiency and decision-making. Additionally, ensuring scalability, responsiveness, and user-friendly experiences across devices remains a pressing concern. There is a clear need for a centralized digital solution that allows job seekers and employers to interact effectively, supports structured workflows, and provides transparent tracking of applications and job postings.

This project aims to address these challenges by developing a scalable, secure, and user-centric JobPortal Pro platform. The platform is designed to streamline job discovery, application tracking, and profile management, providing an efficient and organized environment for all stakeholders involved in the recruitment process.

## Objectives

The objectives of the proposed system are:

* To create a dynamic online platform that efficiently connects employers with qualified job seekers.
* To simplify and automate the entire recruitment process, including job posting, application tracking, and interview scheduling.
* To ensure a secure, user-friendly experience with personalized recommendations and reliable data management.

## Scope and limitation

In today’s competitive job market, the process of recruitment and job searching has often relied on scattered methods such as newspaper listings, manual applications, and multiple job boards. These approaches lead to inefficiencies, lack of coordination, and difficulty in matching the right candidates with the right employers. Without a centralized system, tracking applications, scheduling interviews, and maintaining communication becomes challenging for both employers and job seekers. Therefore, the need for a unified and intelligent job portal platform has become essential one that simplifies recruitment, automates job matching, and enhances the overall hiring experience for all users.

### Scope of System

The Job Portal System aims to create a unified platform that efficiently connects employers and job seekers. It allows employers to post job vacancies, manage applications, and schedule interviews, while job seekers can register, build profiles, upload resumes, and apply for suitable jobs. The system is designed to simplify the recruitment process by centralizing all job-related activities, reducing manual effort, and enhancing communication between both parties.

The system includes three main user roles: Admin, Employer, and Job Seeker. The Admin oversees the platform’s operations, manages users, and ensures smooth functionality. Employers can post and manage job listings, review applicants, and communicate with potential hires, while Job Seekers can explore personalized job recommendations, track applications, and receive interview updates. Overall, the system provides a structured and efficient digital environment to improve the hiring experience and promote better career opportunities.

### Limitation of Existing System

 Recruitment processes are mostly manual and time-consuming.

 Employers find it difficult to manage and track multiple applications.

 Job seekers struggle to find suitable jobs matching their skills.

 Lack of centralized platform leads to poor communication.

 No real-time updates or automated job matching features.

 Limited transparency and difficulty in tracking application status.

## Report Organization

This report consists of five chapters, includingthe current one. **Chapter Two** discussesthe Background Studyand provides anOverview of related existing systems*,* outlining theirstrengths and weaknesses**. Chapter Three**focuses onSystem Analysis and Design, coveringRequirement AnalysisandFeasibility Analysis*.* **Chapter Four**explains th*e* Implementation*,* Testing*,* andDebuggingphases*.* **Chapter Five**concludes the report by summarizing theConclusion*,* Limitations*,* andFuture Enhancements*.* Overall, the report presents the system architecturealong with thetools and technologies used in developing the system.

# Background Study and Literature Review

## Background Study

In today’s competitive job market, traditional recruitment methods such as newspapers, emails, and manual application tracking are often inefficient and time-consuming for both employers and job seekers. These methods make it difficult to manage multiple applications, maintain communication, and match the right candidates with suitable jobs. To address these challenges, online job portals have become essential, providing a centralized platform where employers can post vacancies, review applications, and schedule interviews, while job seekers can create profiles, upload resumes, search for jobs, and track their applications in real-time. Such a system streamlines the recruitment process, improves communication, and enhances the overall efficiency and transparency of hiring.

## Literature Review

The advent of the internet has significantly transformed recruitment, giving rise to online job portals that connect employers and job seekers efficiently. The Internet has changed the way of looking for jobs, through the development of job portals. A job portal is a kind of web portal that provides an efficient ways for searching the Internet or the web for vacant job positions available. [1] Features such as user registration, profile creation, and employer dashboards have enhanced user engagement and streamlined the hiring process, making recruitment faster, more transparent, and cost-effective. It is a web-based recruiting system that can be described as a recruiting method that is carried out via web-primarily based gear, such as a company's intranet or the public internet. [2]

Recent advancements in technology have further improved job portal functionalities. Integration of backend databases like MySQL, coupled with web development frameworks such as PHP and Java, ensures efficient data management and smooth user interaction. Additionally, mobile accessibility, security protocols, and AI-driven job recommendations are shaping the future of recruitment platforms. When looking for a job, it's usually best to look online or get a suggestion from someone who has worked there before. [2] Research indicates that AI and analytics can personalize job suggestions, predict candidate suitability, and optimize the overall recruitment process. The purpose of this system is to ease up and effectively carry out the task of job announcements and application collection replacing the traditional method. [3]

# System Analysis and Design

## System Analysis

The Iterative Model is a cyclical approach to software development that emphasizes building the system through repeated refinement and gradual enhancement. For a Job Portal System, the iterative methodology will be employed to develop the platform in multiple cycles, where each cycle involves planning, designing, developing, testing, and evaluating selected features. Instead of completing all requirements upfront, each iteration delivers an improved version of key modules such as user registration, job posting, profile management, and intelligent job recommendations. Feedback from each cycle will guide the next, ensuring continuous refinement of functionality, usability, and performance. The iterative approach promotes close collaboration, frequent assessment, and adaptive improvements based on evolving project needs. This method enhances the ability to deliver a reliable, user-friendly job portal while allowing flexibility, early issue detection, and steady progress throughout the development lifecycle.

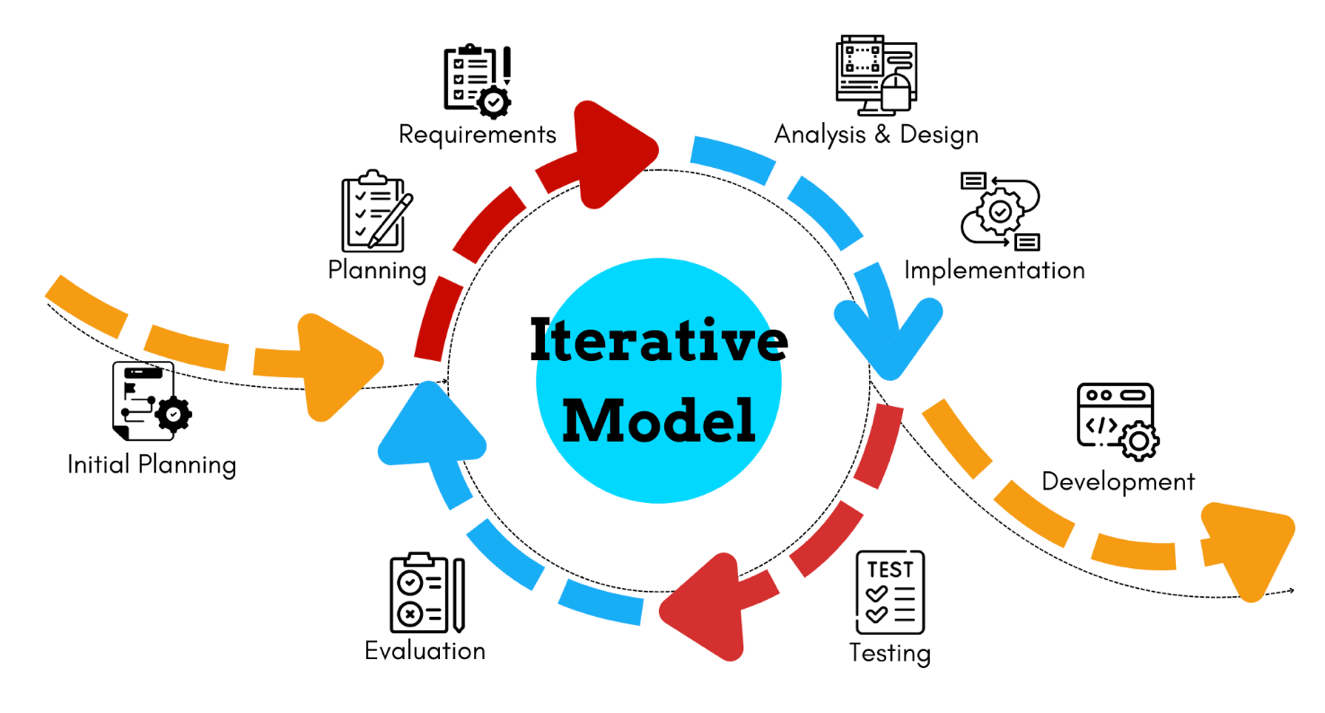


Figure 3.1 Iterative Model

**Iteration 1 –Core Job Portal Functionality**

In the first iteration, the primary focus will be on creating the essential components of the job portal. Features such as **user registration and login** for both job seekers and employers, **job posting** by employers, and **basic job browsing/search** for seekers will be implemented. The database will be structured to store users, job posts, and basic application details. This initial version acts as a working prototype, allowing users to perform fundamental operations and providing early insights into system requirements and usability issues.

**Iteration 2 – Enhanced User Interaction and Experience**

The second iteration will expand the portal’s functionality by improving user interaction and experience. Features such as **advanced job search filters** (location, category, salary), **job seeker profiles** with resume uploads and skill listings, and an **employer dashboard** to manage job posts will be introduced. Database structures will be optimized to support these enhancements. Feedback from users in the first iteration will inform adjustments to the UI and workflows, resulting in a more engaging and user-friendly platform.

**Iteration 3 – Intelligent Matching and Notifications**

The third iteration focuses on introducing **intelligent and personalized features** to the portal. This includes **resume screening and job matching**, **personalized job recommendations** based on user profiles and activity, and **email notifications** for new job postings or application updates. This iteration aims to make the portal more user-centric, helping job seekers find relevant opportunities more efficiently and enabling employers to attract qualified candidates. Continuous evaluation during this cycle ensures performance improvements and aligns the system more closely with user needs.

### Requirement Analysis

The **requirement analysis phase** for the Job Portal involved a thorough study of the expectations and needs of all stakeholders, including employers, job seekers, and administrators. The goal was to clearly define the system’s scope and essential features to ensure efficient job posting, application management, and candidate matching. This phase focused on identifying both functional and non-functional requirements, ensuring that the portal would be secure, user-friendly, reliable, and perform efficiently under various conditions. The documented requirements provide a strong foundation for system design and development and are generally divided into two categories:

1. Functional Requirements
2. Non-Functional Requirements
3. **Functional Requirements**

The functional requirements of the system to fulfill its purpose are listed below:

* **User Registration and Login:** Secure registration and login for Job Seekers, Employers, and Admin with profile management and password recovery.
* **Job Posting and Management:** Employers can create, update, and delete job listings with details such as title, skills, location, and salary.
* **Job Search and Application:** Job Seekers can search for jobs using filters and apply directly through the portal.
* **Application Tracking:** Employers can view, shortlist, or reject applications; Job Seekers can track their application status.
* **Interview Scheduling and Notifications:** Employers can schedule interviews; Job Seekers receive notifications for interview updates.
* **Role-Based Access Control:** Structured access ensures Admin, Employers, and Job Seekers have appropriate permissions for security and workflow management.
* **Secure Messaging:** Encrypted communication between Employers and Job Seekers to protect sensitive information.
* **Job Recommendations:** System provides personalized job suggestions to Job Seekers based on skills, experience, and preferences.
* **Admin Management and Reporting:** Admin can manage all users, job listings, and generate reports to monitor platform activity and performance.

.

1. **Non-Functional Requirements**

Non-functional requirements addressed the overall quality attributes and constraints of the platform beyond its basic functionality are:

* **Performance:** The system is designed to handle high volumes of concurrent users, job postings, and applications efficiently, ensuring fast load times and smooth interactions on both desktop and mobile platforms.
* **Security:** All user data, resumes, and communications are protected with strong encryption and secure authentication protocols to maintain privacy, prevent unauthorized access, and safeguard sensitive information.
* **Usability:** The portal provides a clear, intuitive, and consistent user interface, offering helpful feedback, guidance, and error messages to ensure a seamless experience for Job Seekers, Employers, and Admin.
* **Scalability:** The platform can easily accommodate growth in the number of users, job listings, and additional features, allowing for expansion without compromising performance or responsiveness.
* **Reliability & Maintainability:** Developed with a layered and modular architecture, the system ensures high availability, consistent operation, and simplified maintenance, enabling quick updates, bug fixes, and future feature enhancements.
* **Compatibility:** The portal is designed to work across different devices, screen sizes, and web browsers, ensuring accessibility and a consistent experience for all users.

**Use Case Diagram**

The use case diagram illustrates the overall interaction between three primary actors: **Employer, Job Seeker**, and **Admin** within the **Job Portal System**. Employers can create and publish job postings, review applications, manage their profiles, receive training recommendations, sign up or log in, and get real-time notifications about candidate activities. Job seekers can search and browse job listings, submit job applications, manage their profiles, receive personalized training recommendations, register or log in, and get notifications related to job updates or application statuses. The admin oversees the entire system, managing users, viewing analytics and reports, accessing audit logs, and moderating job postings and applications to maintain platform integrity. This diagram provides a comprehensive view of how each user type interacts with the system’s core functionalities, ensuring seamless communication and efficient management across all roles.

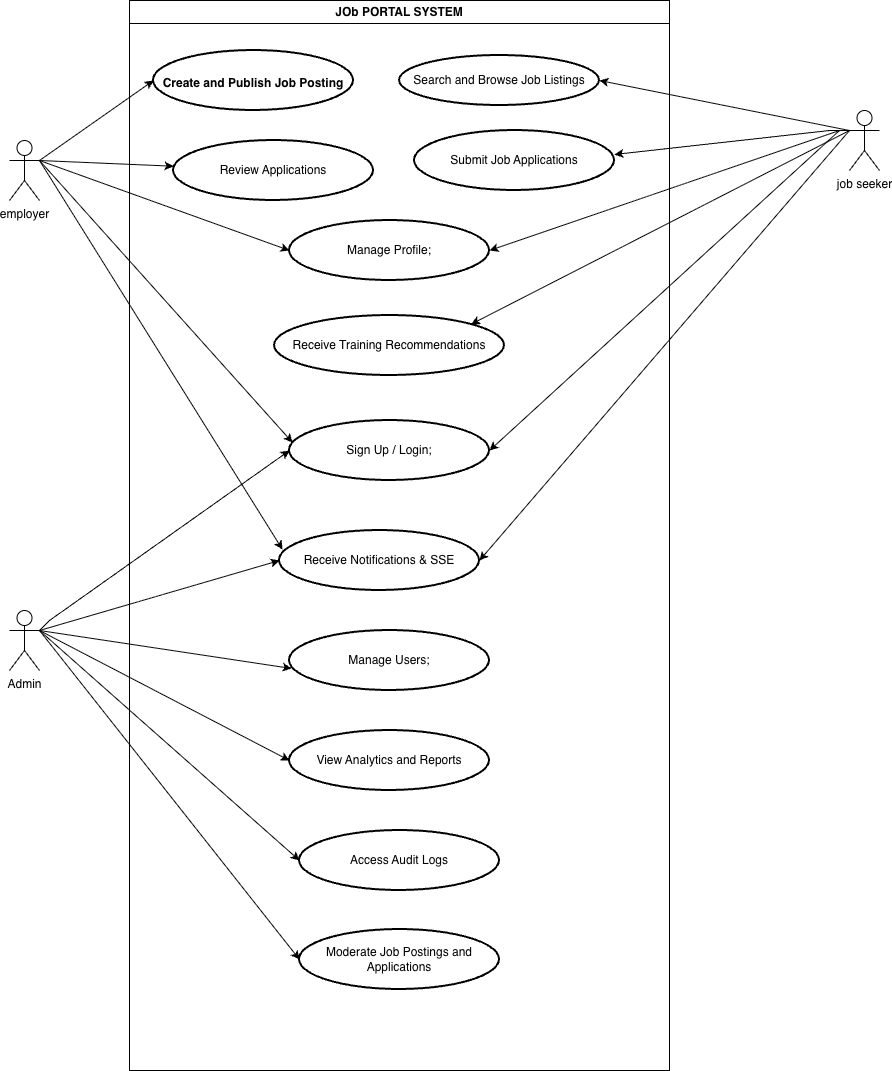


Figure . Use Case Diagram of Job portal system

Table . Use Case Identifier - UC001 – Create and Publish Job Posting

|  |  |
| --- | --- |
| Use Case Identifier | UC001 – Create and Publish Job Posting |
| Primary Actor | User (Employer) |
| Secondary Actor | None |
| Description | The employer creates a new job posting by entering details like title, description, skills, experience, and salary. Once submitted, the job becomes visible to job seekers. |
| Pre-Condition | Employer is logged in and authorized. |
| Post-Condition | Job is successfully published and visible to seekers. |
| Failure | Missing details or system error prevent posting. |

Table . Use Case Identifier - UC002 – Submit Job Application

|  |  |
| --- | --- |
| Use Case Identifier | UC002 – Submit Job Application |
| Primary Actor | User (Job Seeker) |
| Secondary Actor | Employer (Receives Application) |
| Description | The job seeker applies for a listed job by submitting required details and documents. The application is stored for employer review. |
| Pre-Condition | The job seeker is authenticated, has an active profile, and the job posting is open for applications. |
| Post-Condition | Application is successfully submitted and recorded. |
| Failure | Application submission fails due to missing documents, invalid job posting, or server error. |

### Feasibility Study

A feasibility study was performed to determine the practicality and sustainability of developing the Job Portal system across three primary dimensions: technical, operational, and economic.

1. **Technical Feasibility**

The Job Portal is technically feasible due to the adoption of a modern and reliable technology stack, including Next.js, React, Node.js, Express.js, and MongoDB. These technologies provide a strong foundation for building a fast, secure, and scalable platform capable of handling large volumes of user data and interactions. The system supports encrypted communication using AES-GCM encryption, while authentication and authorization are managed securely through JWT tokens. Real-time functionality such as live messaging, notifications, and status updates is implemented through Server-Sent Events (SSE), ensuring a seamless and dynamic user experience.

1. **Operational Feasibility**

Operationally, the platform is designed to streamline recruitment processes and provide an efficient medium for job seekers, employers, and administrators. Its user-friendly interface and well-structured navigation allow users to post jobs, search for opportunities, and manage applications with ease. The inclusion of real-time chat, application tracking, and personalized dashboards ensures that the system integrates smoothly into daily workflows. The design prioritizes accessibility and minimal training requirements, making it easy for users from diverse technical backgrounds to operate effectively.

1. **Schedule Feasibility**

The system is economically feasible due to its reliance on open-source technologies that significantly reduce development and maintenance costs. Since all major components: Next.js, React, Node.js, and MongoDB are free to use, the project avoids licensing expenses. The development and deployment processes can be executed using low-cost hosting solutions, while scalability can be achieved without major financial investment. As a result, the Job Portal offers a sustainable and cost-efficient solution for connecting employers and job seekers while minimizing long-term operational costs.

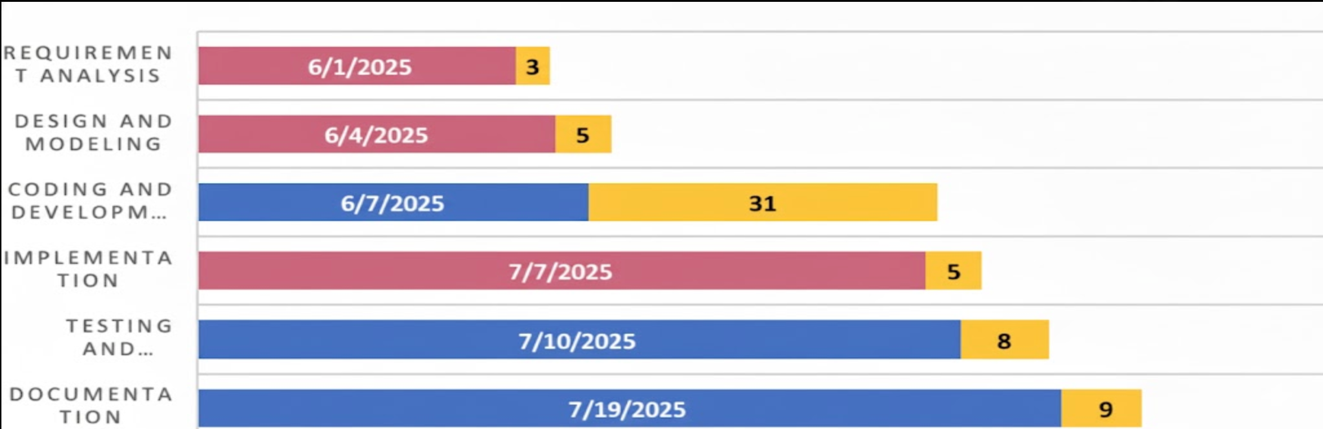


Figure . Gantt Chart of Job Portal System

The Gantt chart for the **Job Portal System** development project, spanning March to April 2025, presents a clear sequential workflow where each phase builds upon the completion of the previous one. The project started with **Requirement Analysis** on March 1, lasting 3 days, followed immediately by **Design and Modeling** from March 4 for 5 days, laying the foundation for development. The central and most time-consuming phase, **Coding and Development**, began on March 7 and extended for 32 days, representing the bulk of the project’s effort and highlighting its critical role in building the system. **Implementation** was scheduled from April 7 for 5 days, ensuring the developed modules were properly deployed, followed by **Testing and Deployment** from April 10 for 8 days to validate functionality and ensure system stability. The project concluded with **Documentation** from April 19 for 9 days, capturing all technical details and user instructions. In the chart, blue bars depict the duration of each task, while orange segments indicate the number of days allocated, visually emphasizing the sequential flow and the predominance of the coding phase within the overall timeline.

### Class Diagram

The class diagram illustrates the structure of a **Job Portal System**, consisting of eight primary classes and seven enumerations that define the relationships and interactions among key components. The **User** class functions as the central entity, storing essential user details such as email, name, role, and status. Employers can create multiple **Job** postings containing information like title, description, employment type, skills, and location. Job seekers can submit **Applications** that track submission details, documents, and status, which may lead to one or more **Interviews** and **Offers** managed through their respective classes.

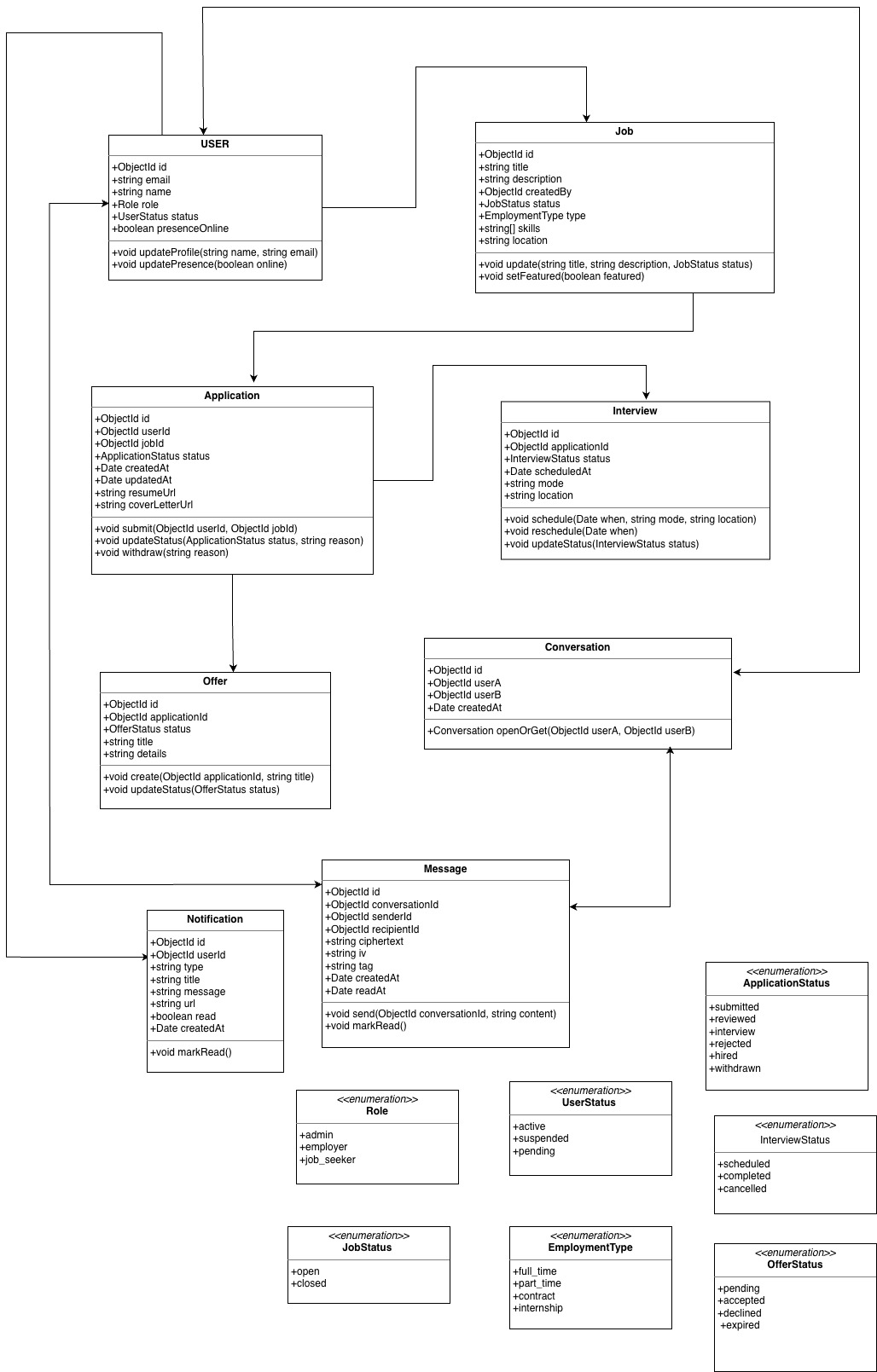


Figure 3.4 Class Diagram of Job Portal System

To support communication, the system includes Conversation and Message classes, enabling real-time chat between users, while Notification provides alerts about important system updates or actions. Enumerations like Role, User Status, Job Status, Employment Type, Application Status, Interview Status, and Offer Status define standardized states, roles, and employment types used throughout the system to maintain data consistency and clear process tracking.

### Sequence Diagram

Figure 3.5 Sequence Diagram for User Registration and Job posting

This sequence diagram illustrates the interaction flow for employer registration and jobposting in a job portal system. The process begins when the Employer UI sends a register() request to the Auth API with the employer’s details, prompting the Auth API to create a new user record in the Database and return a success response with a JWT token and user information. The Employer UI then stores the token locally and verifies the session by calling login() or GET /api/auth/me, confirming authentication. Once authenticated, the employer proceeds to create a job by sending a createJob() request to the Jobs API with job details and an authorization token. The Jobs API validates the token through the JWT service to ensure the requester’s role is “employer,” then stores the job information in the Database and returns a confirmation with the job ID. This sequence ensures that only authenticated employers can post jobs, maintaining security, data integrity, and a smooth end-to-end workflow within the job portal.

### Activity Diagram

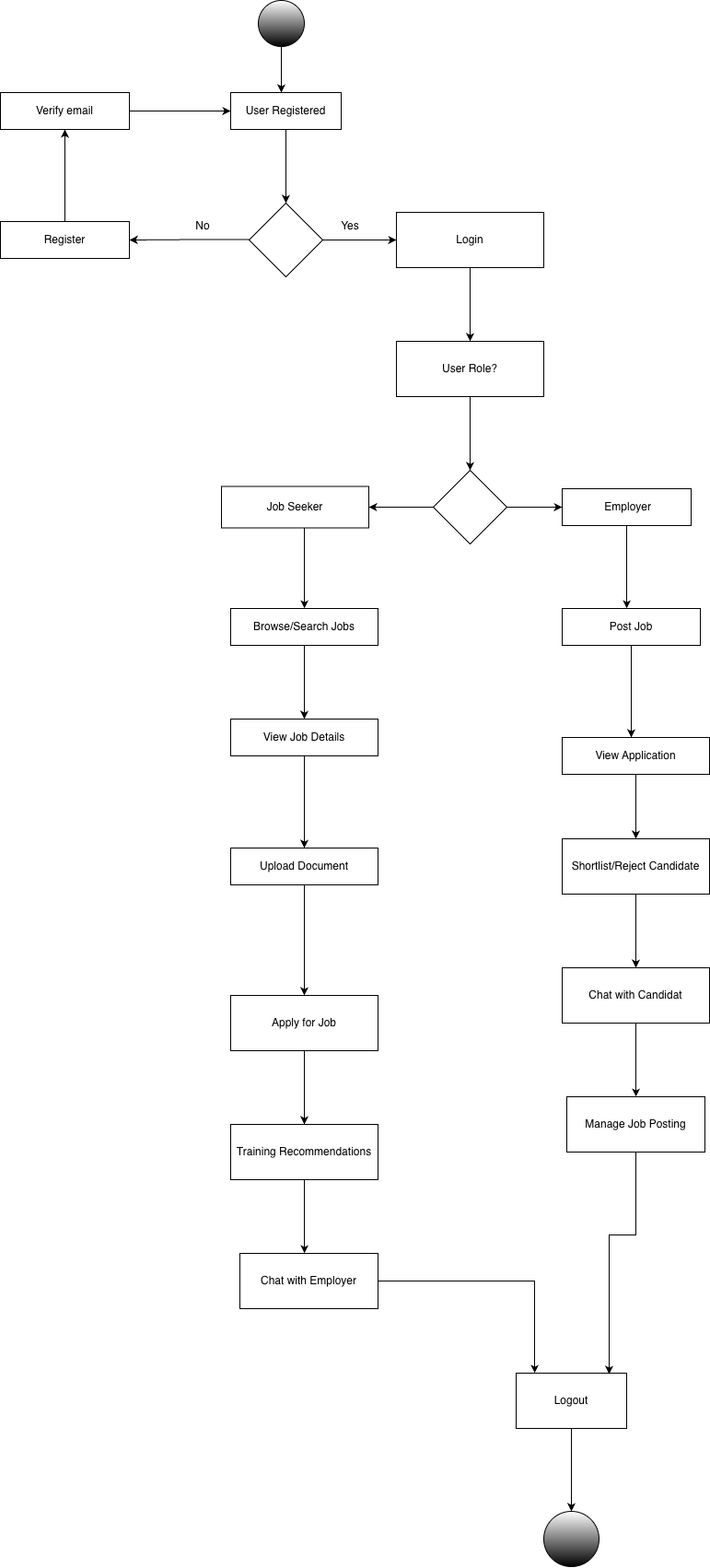


Figure . Activity Diagram

The activity diagram illustrates the workflow of the Job Portal System, showing how users interact with the platform from registration to logout. The process begins when a new user registers and verifies their email to gain access to the system. Once registered, the user logs in and is assigned a specific role, either as a job seeker or an employer. Job seekers can browse or search for job opportunities, view job details, upload documents, and apply for suitable positions. They can also receive training recommendations and communicate directly with employers through the chat feature. Employers can post job listings, review applications, shortlist or reject candidates, and chat with potential hires. Additionally, employers can manage and update their job postings as needed. The workflow concludes with both job seekers and employers logging out, completing a smooth and efficient interaction within the job portal system.

## System Design

### Sequence Diagram

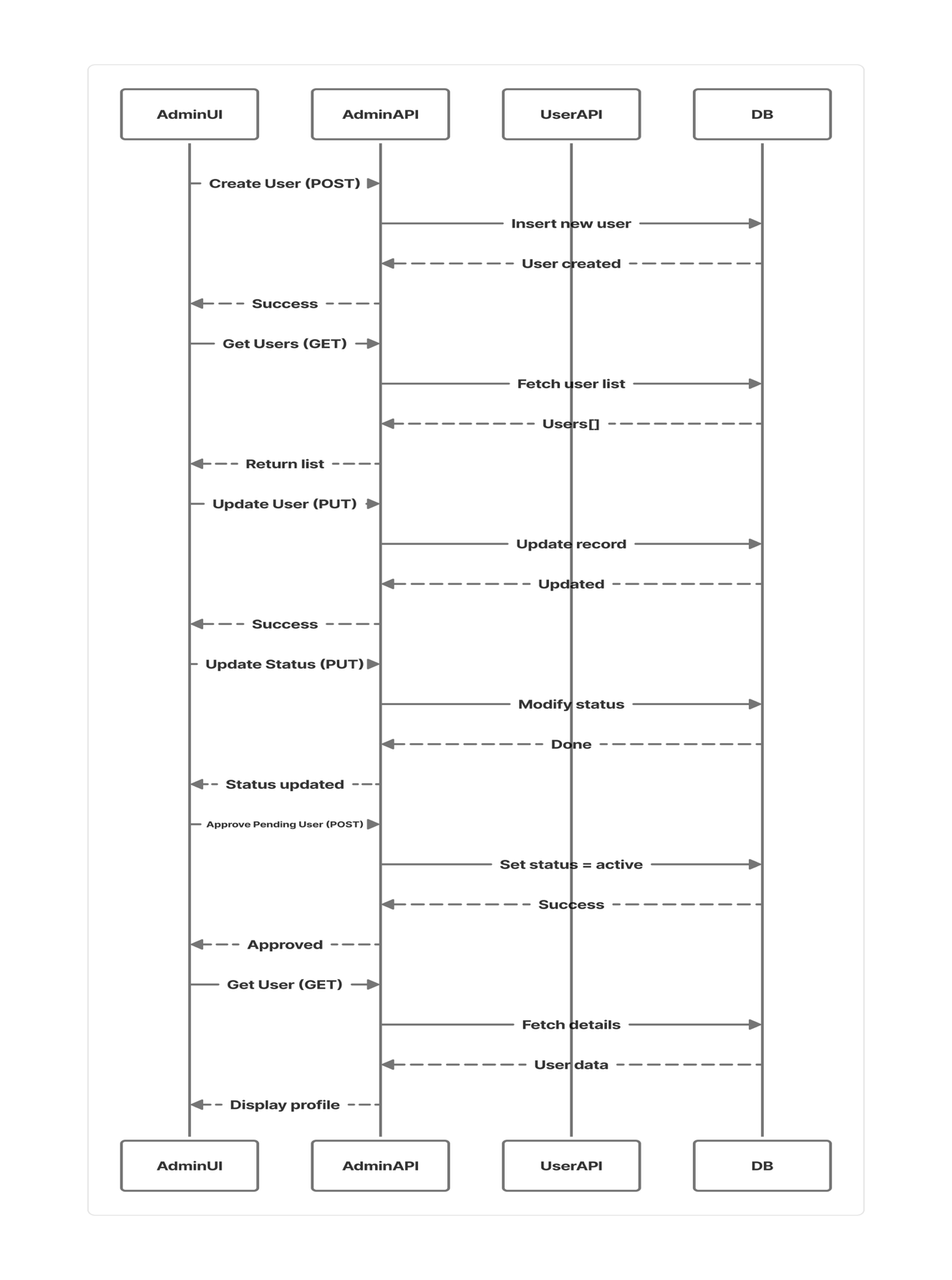


Figure . Sequence Diagram for Admin User management

This sequence diagram illustrates the end-to-end process of how an Admin manages users within the job portal system. The interaction begins when the Admin UI sends a request to the Admin Users API, which serves as the backend interface for user operations. When creating or inviting a new user, the Admin submits user details through a form, and the API validates the admin’s credentials using the authentication logic defined in lib/api-auth.ts. Once validated, the API creates a new record in the DB.users collection and returns a confirmation to the Admin UI. Similarly, when listing or searching users, the UI sends a GET request to retrieve user data, which is fetched from the database and displayed on the interface. For editing a user’s role or name, the Admin updates information via a PUT request, and the API modifies the corresponding database record. When the Admin suspends or reactivates a user, the API updates the user’s status in the database and may trigger a real-time update event (user.updated) to keep interfaces synchronized. Pending users can also be approved through a dedicated endpoint, activating their accounts. Finally, the Admin can view or override any user profile by fetching and updating individual records. This sequence ensures that all administrative actions are authenticated, securely executed, and consistently reflected across the system’s database and user interfaces.

.

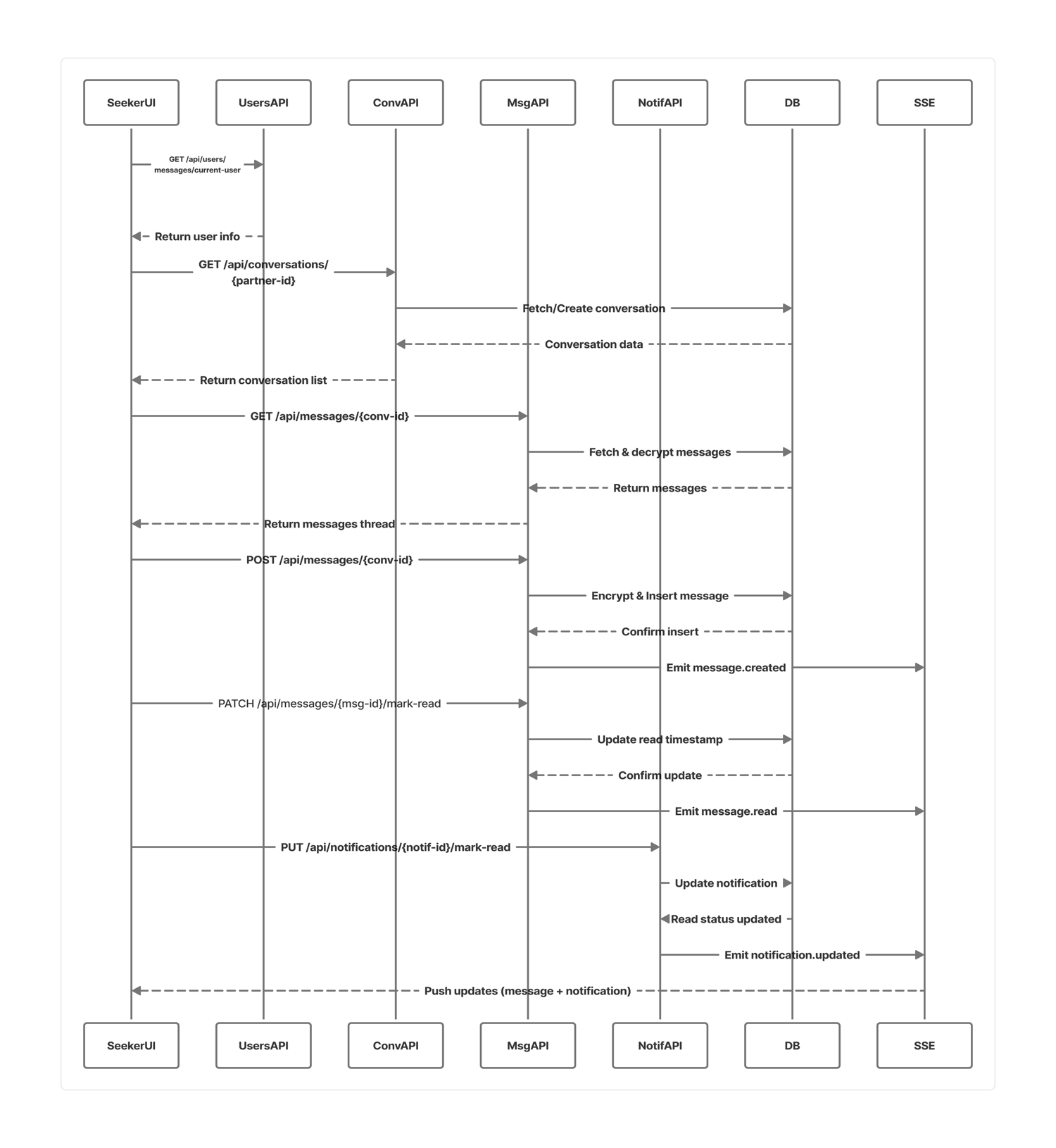


Figure . Sequence Diagram for Chat and Conversation flow

This sequence diagram represents how real-time chat and conversations function between job seekers and employers in the job portal. The process begins when a user (either a seeker or an employer) searches for another user by name or email through the **Users API**, which queries the database and returns the matching user details. The **Seeker/Employer UI** then initiates or retrieves a conversation via the **Conversations API**, which ensures that a conversation record exists in the database. Once a conversation is selected, the **Messages API** fetches and decrypts stored messages, returning the readable thread to the UI. When a user sends a new message, it is encrypted, stored in the database, and the conversation’s last message is updated. The system then emits a real-time message.created event through the **SSE bus**, allowing the recipient’s UI to refresh and display the new message instantly. When the recipient reads the message, a PATCH request updates the read status in the database, and a message.read event signals the sender’s UI to mark messages as seen. The presence system allows users to toggle online/offline status through the **Users API**, which updates the database and broadcasts user.updated events to reflect real-time availability. Finally, whenever a new message is created, the **Notifications API** logs a corresponding entry in the database and emits a notification.created event, prompting the recipient’s notification panel to update and route them to their messages. This end-to-end flow ensures secure, encrypted, and real-time communication between users within the platform.

### Database Design

The database design utilizes a NoSQL approach with MongoDB, employing a flexible, document-oriented structure that stores data in JSON-like formats. This schema-less design allows dynamic and adaptable data modeling, making it well-suited for applications that evolve over time. Data is organized into collections of related documents, with relationships managed through embedding for faster access or referencing for normalized data management. MongoDB’s architecture supports horizontal scalability, high-performance read and write operations, and efficient handling of complex, nested data structures, ensuring robustness and flexibility in managing diverse application data requirements.

{

"\_id": {

"$oid": "68fdc1d107678e1a1008b55b"

},

"email": "smritibca20@oic.edu.np",

"name": "Reshma Khadka",

"role": "job\_seeker",

"passwordHash": "$2a$10$cU7r5e1gNEwjcpjG4LsvH.B43jL2DpRQH3a6rFC.brzSqK5mEn1HS",

"createdAt": {

"$date": "2025-10-26T06:38:09.781Z"

},

"documents": {

"resumes": [

{

"\_id": {

"$oid": "69047cff06c8bf915824d156"

},

"url": "/uploads/68fdc1d107678e1a1008b55b/resume-1761901823287.pdf",

"name": "Resume.pdf",

"uploadedAt": {

"$date": "2025-10-31T09:10:23.290Z"

},

"size": 4030942,

"type": "resume"

}

],

"resume": {

"\_id": {

"$oid": "69047cff06c8bf915824d156"

},

"url": "/uploads/68fdc1d107678e1a1008b55b/resume-1761901823287.pdf",

"name": "Resume.pdf",

"uploadedAt": {

"$date": "2025-10-31T09:10:23.290Z"

},

"size": 4030942,

"type": "resume"

},

"coverLetter": {

"\_id": {

"$oid": "69047f0306c8bf915824d15a"

},

"url": "/uploads/68fdc1d107678e1a1008b55b/cover-letter-1761902339765.pdf",

"name": "Cover Letter.pdf",

"uploadedAt": {

"$date": "2025-10-31T09:18:59.767Z"

},

"size": 189730,

"type": "coverletter"

},

"coverLetters": [

{

"\_id": {

"$oid": "69047ec406c8bf915824d158"

},

"url": "/uploads/68fdc1d107678e1a1008b55b/cover-letter-1761902276850.pdf",

"name": "Cover Letter.pdf",

"uploadedAt": {

"$date": "2025-10-31T09:17:56.853Z"

},

"size": 1084003,

"type": "coverletter"

},

{

"\_id": {

"$oid": "69047f0306c8bf915824d15a"

},

"url": "/uploads/68fdc1d107678e1a1008b55b/cover-letter-1761902339765.pdf",

"name": "Cover Letter.pdf",

"uploadedAt": {

"$date": "2025-10-31T09:18:59.767Z"

},

"size": 189730,

"type": "coverletter"

}

]

},

"updatedAt": {

"$date": "2025-10-31T09:18:59.767Z"

},

"avatar": "/uploads/68fdc1d107678e1a1008b55b/avatar.jpg",

"availabilityStartDate": "0005-11-09",

"bio": "I am a passionate Data Scientist skilled in transforming complex data into actionable insights that drive smarter decisions and meaningful outcomes. With a strong foundation in statistical analysis, machine learning, and data visualization, I specialize in building data-driven solutions that enhance business performance and user experiences. I enjoy working across the full data lifecycle — from data collection and cleaning to model development and deployment — ensuring accuracy, scalability, and real-world impact. Curious by nature and motivated by innovation, I continuously explore new tools and techniques to deliver efficient, human-centered results in every project I take on.",

"certifications": [

{

"name": "AWS",

"issuer": "Amazon",

"date": "2024-05",

"credentialId": "",

"link": "https://aws.amazon.com/free/?trk=d6339f22-96ce-4ae0-80e5-6aa58733006a&sc\_channel=ps&ef\_id=EAIaIQobChMI5-GzvdvBkAMVZKhmAh3HbimKEAAYASAAEgKAjPD\_BwE:G:s&s\_kwcid=AL!4422!3!453325184782!e!!g!!aws!10712784856!111477279771&gad\_campaignid=10712784856&gbraid=0AAAAADjHtp9X9KTNuxSzGzXJoJOxwtkf2&gclid=EAIaIQobChMI5-GzvdvBkAMVZKhmAh3HbimKEAAYASAAEgKAjPD\_BwE"

}

],

"dreamJob": "Data Analayst",

"education": [

{

"degree": "Bachelor in computer Application",

"school": "Tribhuwan University",

"start": "2020",

"end": "2024",

"description": "ahvxgvgacxfcafvz"

}

],

"experiences": [

{

"title": "Data Analyst",

"company": "Softsaro",

"start": "2021",

"end": "Present",

"description": "i work on software development, testing etc."

}

],

"isFresher": false,

"location": "Kathmandu, Nepal",

"openToInternships": false,

"phone": "9761774426",

"preferredInternshipDuration": "",

"projects": [

{

"name": "Recipe app",

"role": "Developer",

"link": "https://github.com/smritiKha/Recipe\_app",

"start": "2023-05",

"end": "2023-08",

"description": "Chef’s Delights is a modern Android-based recipe recommendation application designed to help users explore delicious meals easily and cook confidently at home. The app provides a personalized cooking experience by recommending recipes based on user preferences and ingredients.",

"tech": [

"java",

"xml",

"android programming"

]

}

],

"salaryExpectation": "$80,000",

"skills": [

"JavaScript",

"Next.js",

"Node.js",

"MongoDB",

"SQL",

"Testing",

"System Design",

"REST APIs"

],

"title": "Data Anlyst",

"settings": {

"notifications": {

"applicationUpdates": true,

"jobAlerts": true,

"productNews": true

},

"presence": {

"online": false

}

}

}

Figure . Database Design

### Restful API Workflow

This diagram elegantly depicts the interaction between a client and a server in a RESTful API architecture, where communication occurs through structured HTTP requests and responses. Clients such as mobile, web, or desktop applications send requests (GET, POST, PUT, DELETE) formatted in JSON or XML to the server, which processes them via REST APIs and interacts with the database to retrieve or modify data as needed. The server then returns an appropriate HTTP response, including both data and status codes like “200 OK” or “404 Not Found,” ensuring clear communication of the outcome. This architecture provides a seamless, scalable, and efficient means of data exchange, maintaining a clean separation between client-side interaction and server-side logic while promoting flexibility, interoperability, and maintainability across diverse platforms.

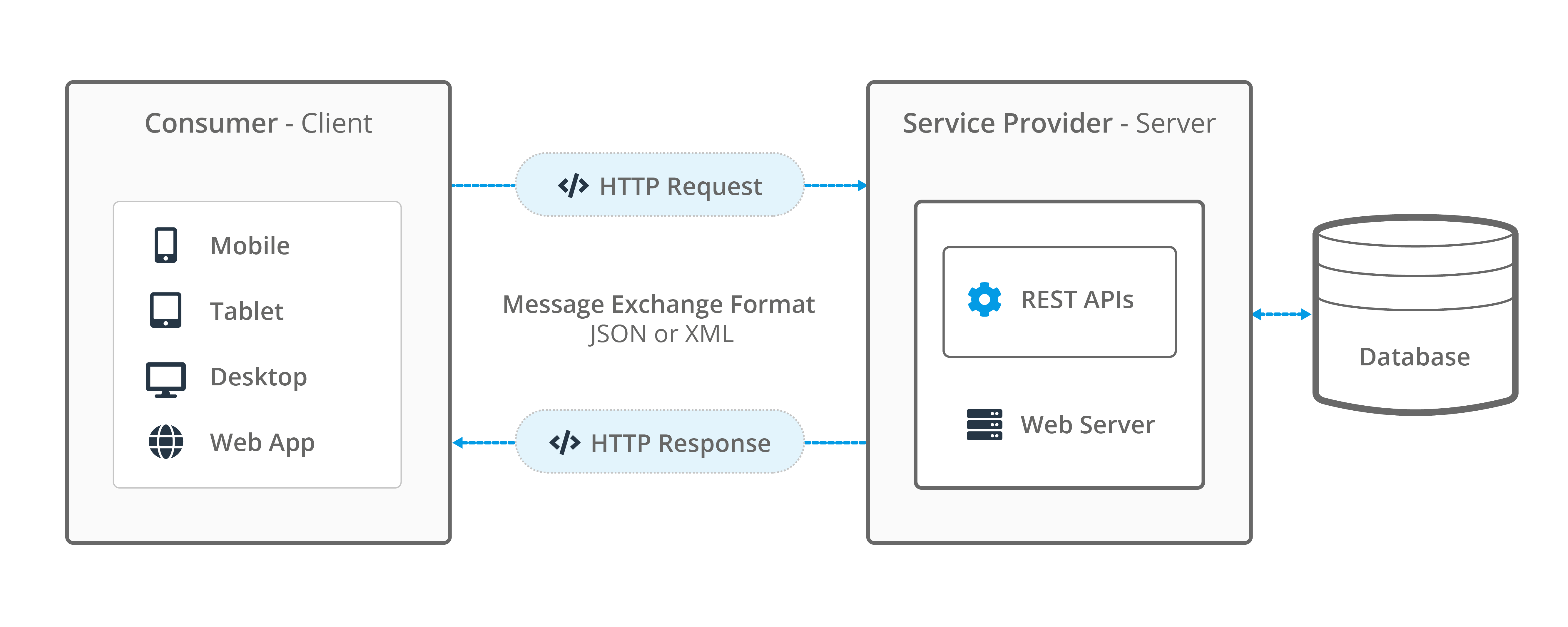
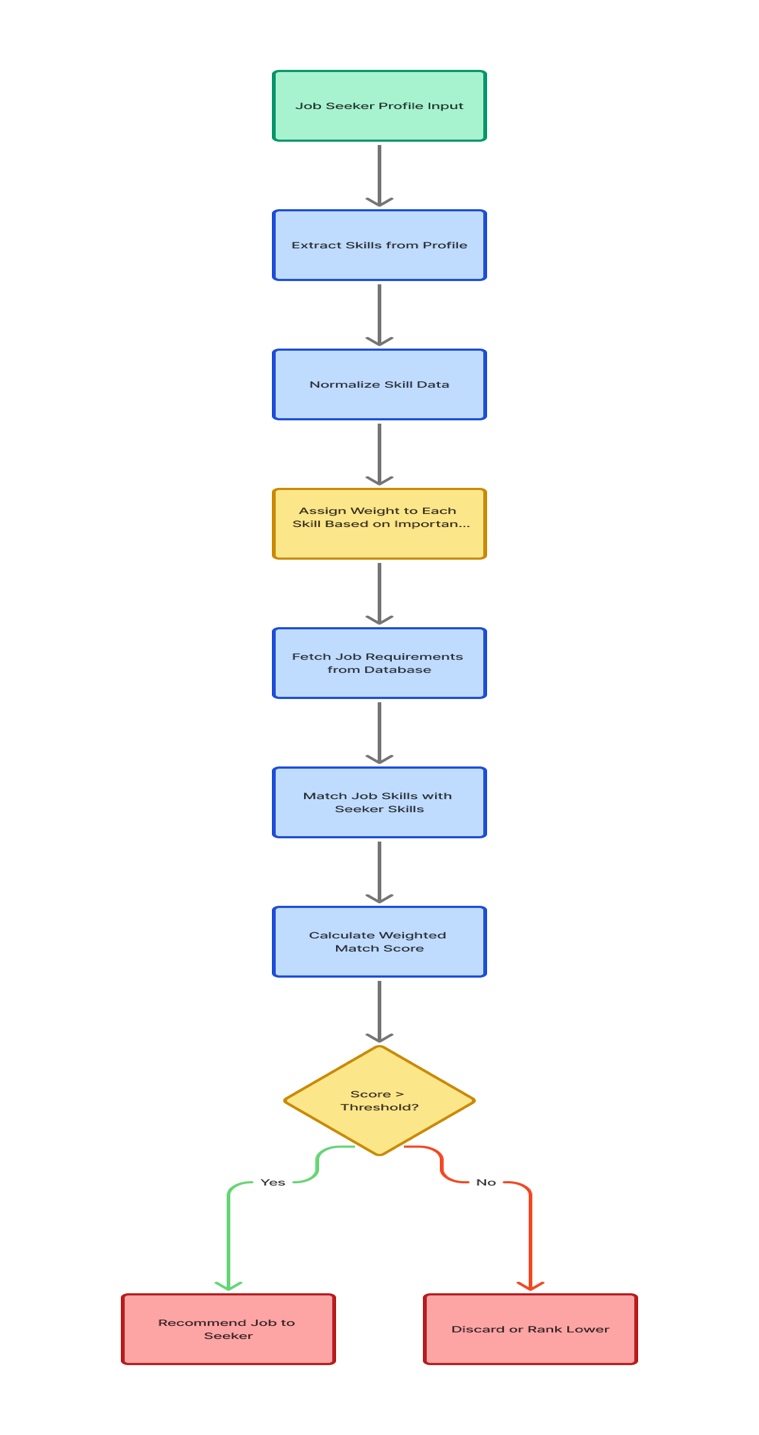


Figure . Restful API Workflow

## Algorithm Details

The **Rule-Based Weighted Skill Matching Algorithm** is a structured and transparent approach to evaluating compatibility between job seekers and job postings. It begins by processing two primary data sources: the **job seeker profile**, containing skills, experience, and education, and the **job post requirements**, which define desired skills and assign relative importance through weighted attributes. Within the **matching engine**, a rule-based system first validates essential criteria such as mandatory skills, experience thresholds, and qualification levels, ensuring only eligible candidates proceed. The **weight calculator** then normalizes the importance of each attribute so that highly valued competencies exert greater influence on the final outcome.

Figure . Working of ****Rule-Based Weighted Skill Matching Algorithm****

Once both data sets are ready, the algorithm performs skill matching by comparing the candidate’s skills with the required ones. Each matched skill contributes to a weighted match score, calculated by multiplying the match value by its assigned weight. The final score determines how well a candidate fits the job. Candidates with scores above a certain threshold are recommended to employers or displayed higher in search results, while others are ranked lower. This method ensures that matching decisions are fair, transparent, and data-driven, improving the quality and accuracy of job placements.

**Phase 1: Input Collection**

The system begins by collecting structured and semi-structured data from two primary sources: job seeker profiles and job postings. Job seeker data includes attributes such as skills, experience, education, and certifications, while job postings define required skills, experience levels, and weightage parameters. All collected data is stored in normalized relational database schemas (for example, DB.users and DB.jobs) to ensure consistency and efficient query processing.

**Phase 2: Data Preprocessing and Normalization**

This phase prepares the raw data for algorithmic matching. The system applies text parsing and Named Entity Recognition (NER) techniques to extract relevant skill entities and attributes. Synonymous or variant terms such as “JS” or “Java Script” are standardized to canonical forms like “JavaScript.” The processed data is then transformed into vectorized numerical representations, enabling compatibility with computational models and similarity computations.

**Phase 5: Threshold Evaluation**

The computed match score S is compared against a predefined threshold (T), typically set around 70 percent. Candidates with S ≥ T are categorized as strong matches, while those below the threshold are flagged as weak matches. This ensures that only high-relevance candidates advance to the recommendation phase.

**Phase 6: Output Generation and Ranking**

All evaluated candidates are sorted in descending order of their match scores (S). The system generates ranked outputs for employers and job seekers. Employers can view top-matching candidates in the job portal dashboard, while job seekers can see ranked job recommendations that align with their skill sets. This ranking mechanism enables data-driven candidate prioritization and efficient recruitment workflows.

**Phase 7: Feedback and Refinement**

After deployment, the system continuously captures feedback data from employer actions, application outcomes, and hiring decisions. This data is processed in a model refinement module that dynamically adjusts weight values and rule parameters. Over time, the algorithm evolves through iterative learning, improving precision, reducing bias, and increasing the accuracy of future match results.

**Security Advantages**

* Data Privacy: Candidate and job information is securely stored and protected within the system
* Role-Based Access: Users can access only the data relevant to their role (employer, candidate, admin)
* Secure Processing: Matching and scoring are performed internally without exposing raw personal data
* Integrity and Tamper Protection: Rule-based logic ensures that scores and rankings cannot be manipulated
* Auditability: All phases of the algorithm are traceable, enabling accountability and secure feedback handling

# Implementation and Testing

## Implementation

The development of the system was guided by the Agile Software Development Methodology, which facilitated an iterative and adaptive approach throughout the project lifecycle. The process was structured into a series of incremental sprints, each encompassing stages of requirement analysis, design, implementation, testing, and evaluation. Core system modules such as user authentication, workspace management, task coordination, and real-time communication were developed progressively and refined through continuous feedback and validation cycles. This methodological framework ensured flexibility in design, rapid resolution of issues, and effective incorporation of evolving user requirements, ultimately contributing to a robust, efficient, and user-centered system architecture.

### Tools Used

The technologies and tools employed in the design and development of the Job Portal system are outlined below:

**Next.js**: A full-stack React framework used for building server-rendered and statically generated web applications. It provides routing, API routes, and server-side rendering capabilities, making it ideal for building scalable and dynamic job portal platforms.

**React + TypeScript**: React is used for building interactive user interfaces, while TypeScript adds static typing, improving code reliability, maintainability, and reducing runtime errors. This combination ensures a robust frontend architecture for the job portal.

**Node.js**: A server runtime that executes the Next.js application and API routes, providing a reliable backend environment for handling requests and server-side logic.

**Tailwind CSS**: A utility-first CSS framework that allows rapid and consistent styling across components, supporting responsive and modern UI design.

**PostCSS + Autoprefixer**: CSS processing tools that transform styles and add vendor prefixes for cross-browser compatibility.

**shadcn/ui + Radix UI**: Prebuilt, accessible UI components and primitives that accelerate UI development and ensure accessibility compliance.

**Lucide React**: An icon library used throughout the frontend to provide consistent and customizable icons.

**JSON Web Tokens (JWT)**: Used for authentication and session management, enabling secure and stateless user login.

**bcryptjs**: A library for hashing passwords, ensuring secure storage of user credentials.

**AES-GCM (Web Crypto API)**: Provides encryption for sensitive messages and other confidential data.

**MongoDB**: A NoSQL database used for storing user profiles, job postings, messages, notifications, and other application data. Its flexible schema supports structured and unstructured data.

**Server-Sent Events (SSE)**: Enables real-time updates for notifications, messages, and user presence, keeping the UI interactive and responsive.

**pnpm**: A fast and reliable package manager used for dependency management and maintaining consistent builds.

**Docker**: Containerization tool used to package the application and its dependencies, ensuring consistent deployment across environments.

**GitHub Actions**: Provides continuous integration and deployment workflows, automating testing, building, and deployment processes.

**Environment Variables**: Securely store API keys, database URIs, and configuration settings, keeping sensitive information out of the codebase.

**Draw.io (diagrams.net)**: An online diagramming tool used to create flowcharts, system architecture diagrams, ER diagrams, and other technical schematics. It helps visualize complex workflows and supports planning and documentation for the job portal system.

### Implementation Detail of Modules

Table . Implementation Detail of Modules

|  |  |  |
| --- | --- | --- |
| Code | Module Name | Description |
| M1 | |  | | --- | |  |  |  | | --- | | Authentication Module | | Ensures secure access to the system by handling user login, registration, JWT-based session management, and role verification |
| M2 | |  | | --- | |  |  |  | | --- | | User Module | | Manages comprehensive user profiles, tracks online/offline presence, updates roles/statuses, and facilitates user lookup. |
| M3 | |  | | --- | |  |  |  | | --- | | Jobs Module | | Enables employers to create, update, filter, and close job postings while allowing job seekers to browse and apply efficiently. |
| M4 | Applications Module | Tracks job applications, monitors application statuses, and manages screening questions and candidate evaluation. |
| M5 | Messaging & Notifications Module | Provides secure, real-time communication between users and delivers timely notifications about messages, applications, and job activity. |
| M6 | File Uploads Module | Supports uploading and managing documents such as resumes, cover letters, portfolios, and certificates with proper metadata handling. |

**Implementation Detail of Chat Module**

This diagram illustrates the architecture of the chat module within the Job Portal system, built around React Context for authentication and page-local state management for conversations and UI. Server-Sent Events (SSE) provide real-time updates for new messages, read receipts, user presence, and notifications, ensuring the UI stays synchronized without full page reloads. The system includes three main UI groups: Conversation List, which shows seeker↔employer chats with unread counts and presence indicators; Chat Interface, which handles message display, sending, and read receipts; and User Presence, which tracks online/offline status for all participants. The components follow a unidirectional data flow, taking state from context/hooks and updating server-side data through REST APIs.

Security and backend handling are integral to the module. Messages are sent as plaintext from the client but are encrypted server-side using AES-GCM, stored securely, and only decrypted for authorized participants. JWT authentication ensures only valid users can access conversations, while SSE keeps read receipts and presence data live. Conversations are upserted when a peer is selected, threads are fetched via REST, and notifications integrate seamlessly, routing users to the correct messages page by role. This layered approach ;React Context, SSE, AES-GCM, JWT, and REST ensures the chat experience is secure, responsive, and easy to maintain across the job portal.

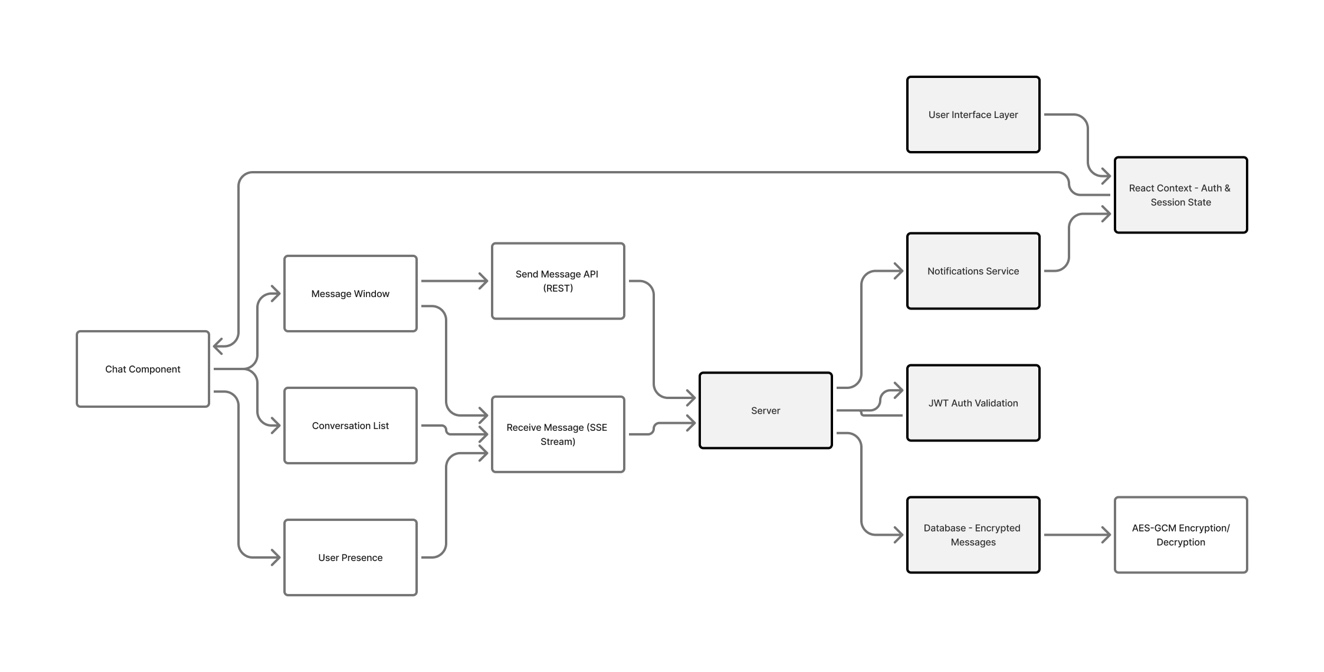


Figure . Implementation Detail of Chat Module

## Testing

The testing for Project Management System is done by testing the unit and system modules like login module, sign up module.

### Test Cases for Unit Testing

The test cases for the login and signup module are as follows:

Table . Test Case 001 - Signup page

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Name:** **Job Portal System** | | | | | | | |
| **Test Case** | | | | | | | |
| **Test Case ID:** **TC\_Signup\_001** | | | | **Test Design By:** **Smriti** | | | |
| **Module Name:** **Sign Up Page** | | | | **Test Design Date: 2025-07-25** | | | |
| **Test Title**: **Register New User** | | | | **Test Executed By: Smriti** | | | |
| **Description:** **Test the sign-up page of system** | | | | **Test execution date: 2025-07-25** | | | |
| **Pre-Conditions:** **User has all necessary details** | | | | | | | |
| **Dependencies:** | | | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | | **Actual Result** | **Status**  **(Pass/Fail)** | **Notes** |
| 1 | Navigate to sign up page |  | Sign up page should open | | As Expected, i.e., User is navigated to sign up page | Pass |  |
| 2 | Provide all required information | Email: [smriti@gmail.com](mailto:smriti@gmail.com)  Password: Smriti123 | Credential can be entered | | As Expected, | Pass |  |
| 3 | Click on Sign UP button |  | Customer should be able to sign up to the system and get verification email | | As Expected, i.e. Customer is able to register to the system and get verification email | Pass |  |
| **Post-conditions:**  **User credentials are validated and successfully registered to Job Portal System** | | | | | | | |

Table . Test Case 002 - Login Page

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Name:** **Job Portal System** | | | | | | | |
| **Test Case** | | | | | | | |
| **Test Case ID:** **TC\_Login\_002** | | | | **Test Design By:** **Smriti** | | | |
| **Module Name:** **Login Page** | | | | **Test Design Date: 2025-07-25** | | | |
| **Test Title**: **Login existing users** | | | | **Test Executed By: Smriti** | | | |
| **Description:** **Test the login page of system** | | | | **Test execution date: 2025-07-25** | | | |
| **Pre-Conditions:** **User has all necessary details and email should be verified** | | | | | | | |
| **Dependencies:** | | | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | | **Actual Result** | **Status**  **(Pass/Fail)** | **Notes** |
| 1 | Navigate to login page |  | Login page should open | | As Expected, i.e., User is navigated to log in page | Pass |  |
| 2 | Provide all required information | Email: smriti@gmail.com  Password: Smriti123 | Credential can be entered | | As Expected, | Pass |  |
| 3 | Click on Login button |  | Customer should be able to log in to the system | | As Expected, i.e., Customer is able to login to the system | Pass |  |
| **Post-conditions:**  **User is validated with database and successfully logged in to Job Portal System.**  **The account system details are logged in to the database.** | | | | | | | |

### Test Cases for System Testing

System testing was carried out to verify seamless integration across the different components of the Job Portal System, including user registration, authentication, job posting, application submission, messaging, notifications, and AI-powered training recommendations. The performance and interoperability of external systems such as MongoDB databases, real-time socket connections, and secure APIs were thoroughly assessed. This phase simulated real-world user interactions and potential system failures to evaluate system reliability under various conditions.

Common integration issues such as data inconsistencies, unauthorized access attempts, or miscommunication between modules were proactively mitigated through robust error-handling mechanisms and comprehensive test scenarios. Test cases were executed to evaluate system responses to both valid and invalid inputs, ensuring consistent behavior and system stability. All results were recorded to provide verifiable evidence of system performance and correctness.

Several categories of system testing were applied:

Functionality Testing: Key features including user registration, login/logout, role-based access control, profile and document management, job posting and filtering, application submission, messaging, notifications, and training recommendations were tested against the defined requirements. Each module was verified to ensure that user workflows, such as creating or updating profiles, posting or applying for jobs, and exchanging messages, operated as intended.

Usability Testing: The user interface was evaluated from an end-user perspective to ensure an intuitive and smooth experience. Tasks such as navigating dashboards, browsing jobs, applying for positions, managing profiles, and accessing training recommendations were examined to confirm that the platform facilitated efficient and user-friendly interactions.

The following test scenarios were executed during system testing:

Table . Test Case 003 – Post Job

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Name:** **Job Portal System** | | | | | | | |
| **Test Case** | | | | | | | |
| **Test Case ID:** **TC\_** **Post Job** **\_003** | | | | | **Test Design By:** **Smriti** | | |
| **Module Name:** **Job Posting** | | | | | **Test Design Date: 2025-7-18** | | |
| **Test Title**: **Create New Job (Employer)** | | | | | **Test Executed By: Smriti** | | |
| **Description:** **Validate that an employer can create and post a new job** | | | | | **Test execution date: 2025-7-18** | | |
| **Pre-Conditions:** **Logged in as employer** | | | | | | | |
| **Dependencies:** **Node.js backend API, MongoDB database, Internet connection, /api/jobs** | | | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | **Actual Result** | | **Status**  **(Pass/Fail)** | **Notes** |
| 1 | Navigate to Post Job |  | Job creation form opens | As expected | | Pass | Page loads correctly |
| 2 | Fill job details | Title: Frontend Engineer, Type: full-time, Skills: React, TS, Location: Remote, Salary: 80000–120000 | Fields accept input without validation errors | Form validations pass | | Pass |  |
| 3 | Submit Job |  | Job is created, success toast; job listed in employer jobs | Job created successfully | | Pass |  |
| **Post-conditions:**  **New job record exists with createdBy = employer; job visible in listing.** | | | | | | | |

Table . Test Case 004 – Apply Job

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Name:** **Job Portal System** | | | | | | | |
| **Test Case** | | | | | | | |
| **Test Case ID:** **TC\_ApplyJob\_004** | | | | **Test Design By:** **Smriti** | | | |
| **Module Name:** **Job Applications** | | | | **Test Design Date: 2025-7-18** | | | |
| **Test Title**: **Apply to Job (Seeker)** | | | | **Test Executed By: Smriti** | | | |
| **Description:** **Validate that a job seeker can apply to a posted job with a resume** | | | | **Test execution date: 2025-7-18** | | | |
| **Pre-Conditions:** **Job exists and is open; seeker logged in; resume uploaded** | | | | | | | |
| **Dependencies: Active backend API, MongoDB database, Internet connection, /api/applications, /api/upload** | | | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | | **Actual Result** | **Status**  **(Pass/Fail)** | **Notes** |
| 1 | Open Job Detail | Job: Frontend Engineer | Job details visible | | As Expected | Pass |  |
| 2 | Click Apply | Resume: resume.pdf Cover Letter: optional | Validation passes; application ready | | As Expected | Pass |  |
| 3 | Submit Application |  | Application created; success toast; status = submitted | | As Expected | Pass |  |
| 7 | |  | | --- | | Verify project  in project list |  |  | | --- | |  | |  | New project should appear in the workspace project list | | As Expected, i.e., newly created project is listed | Pass |  |
| **Post-conditions:**  Application exists; visible in seeker applications page | | | | | | | |

# Conclusion and Future Recommendation

## Lesson Learnt / Outcome

The development and implementation of the Job Portal System provided valuable insights into both technical and project management aspects of modern web applications. A key lesson was the importance of secure authentication and authorization, implemented using JWT-based sessions, robust password hashing, and role-based access control for admins, employers, and job seekers. Designing an intuitive and accessible user interface with Next.js and Tailwind CSS emphasized the need for clear navigation and a responsive user experience. Developing core modules such as job posting, profile management, job applications, notifications, messaging, and training recommendations reinforced the value of a well-structured MongoDB schema and consistent API design for smooth interaction between the front-end and back-end.

Implementing encrypted real-time chat with AES-GCM and Server-Sent Events highlighted the importance of securing data in transit while maintaining responsive communication. Thorough testing and optimization taught the value of database indexing, debounced updates, and idempotent operations to ensure reliability and scalability. Additionally, operational lessons such as suspending rather than deleting data, deep-linking notifications, and separating concerns with REST for CRUD operations, SSE for real-time updates, and JWT for authentication improved understanding of maintainable system architecture. Overall, the project strengthened technical, analytical, and collaborative skills, resulting in a secure, scalable, and user-friendly Job Portal System that meets the requirements of modern recruitment platforms.

## Conclusion

The Job Portal System effectively provides a secure, efficient, and user-friendly platform for job seekers and employers, combining robust back-end functionality with an intuitive front-end design. Key features such as secure authentication, role-based access, job management, notifications, messaging, and real-time communication ensure seamless interactions and support effective recruitment processes. The development process strengthened technical skills in database modeling, API design, secure communication, and responsive UI development, while also enhancing analytical thinking and project management abilities. Overall, the system delivers a scalable and maintainable solution that meets its objectives and provides a strong foundation for future enhancements, including AI-driven job recommendations, multi-language support, and third-party platform integration.

## Future Enhancement

The following features are recommended for implementation in future releases of the Job Portal System to enhance functionality, user experience, and efficiency:

* Introduce a built-in tool allowing users to create and customize professional resumes directly within the portal.
* Implement multiple language options to make the portal accessible to a wider audience globally.
* Connect with platforms like LinkedIn and Google Calendar for seamless profile updates and interview scheduling.
* Enable recruiters and candidates to conduct real-time interviews directly through the system.
* Strengthen data protection with two-factor authentication, encrypted communication, and regular security audits.

# REFERENCES

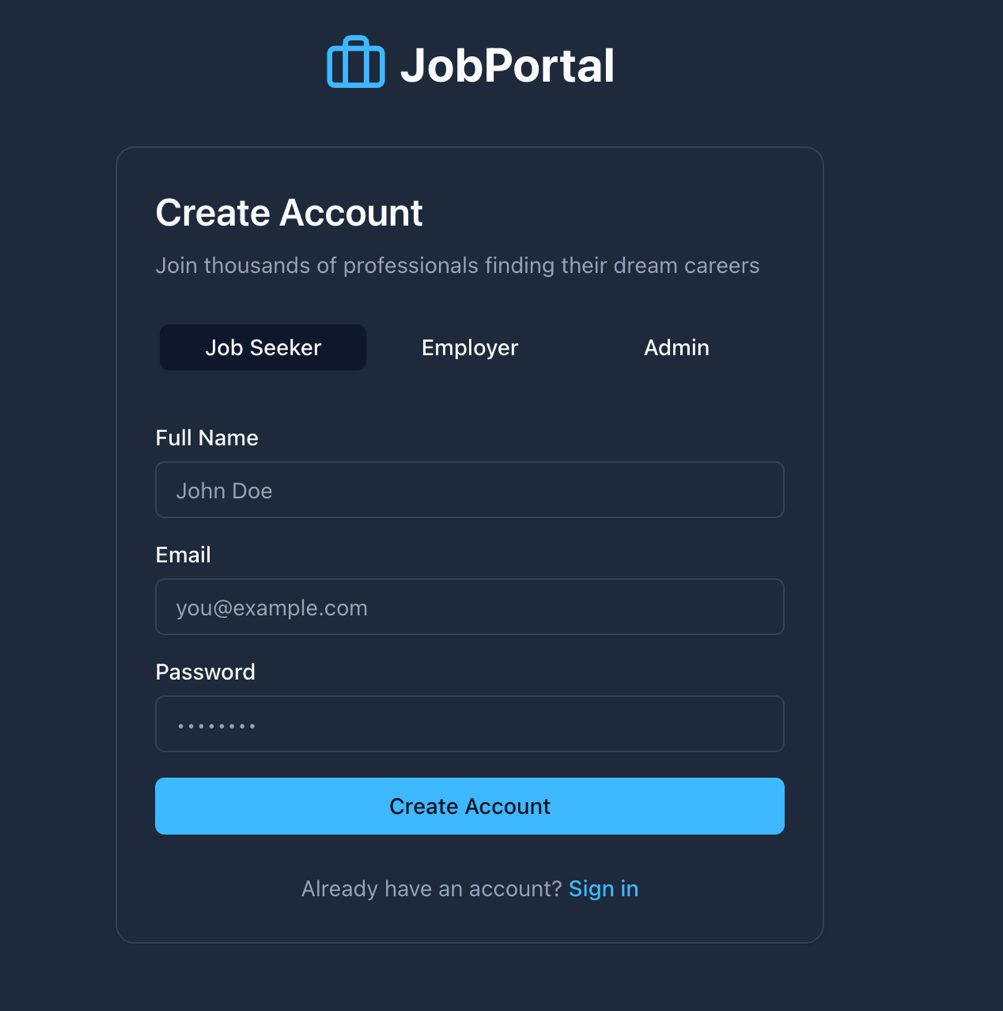
|  |  |
| --- | --- |
| [1] | A. J. M. S. A. S. D. M. Vivek Kumar Sehgal, Job Portal-A Web Application for Geographically Distributed Multiple Clients, Waknaghat, Solan, H.P (INDIA): First International Conference on Artificial Intelligence, Modelling & Simulation, 2013. |
| [2] | N. M. P. P. V. V. N. D. S. S. V Pavani, Feature Extraction based Online Job Portal, Guntur, Andhra Pradesh, India: International Conference on Electronics and Renewable Systems (ICEARS) , 2022. |
| [3] | U. G. S. K. S. R. R. A. S. Vijay Yadav, Smart Job Recruitment Automation: Bridging Industry and University, Pulchowk Campus Lalitpur, Kathmandu: Department of Electronics and Computer Engineering Pulchowk Campus, 2019. |

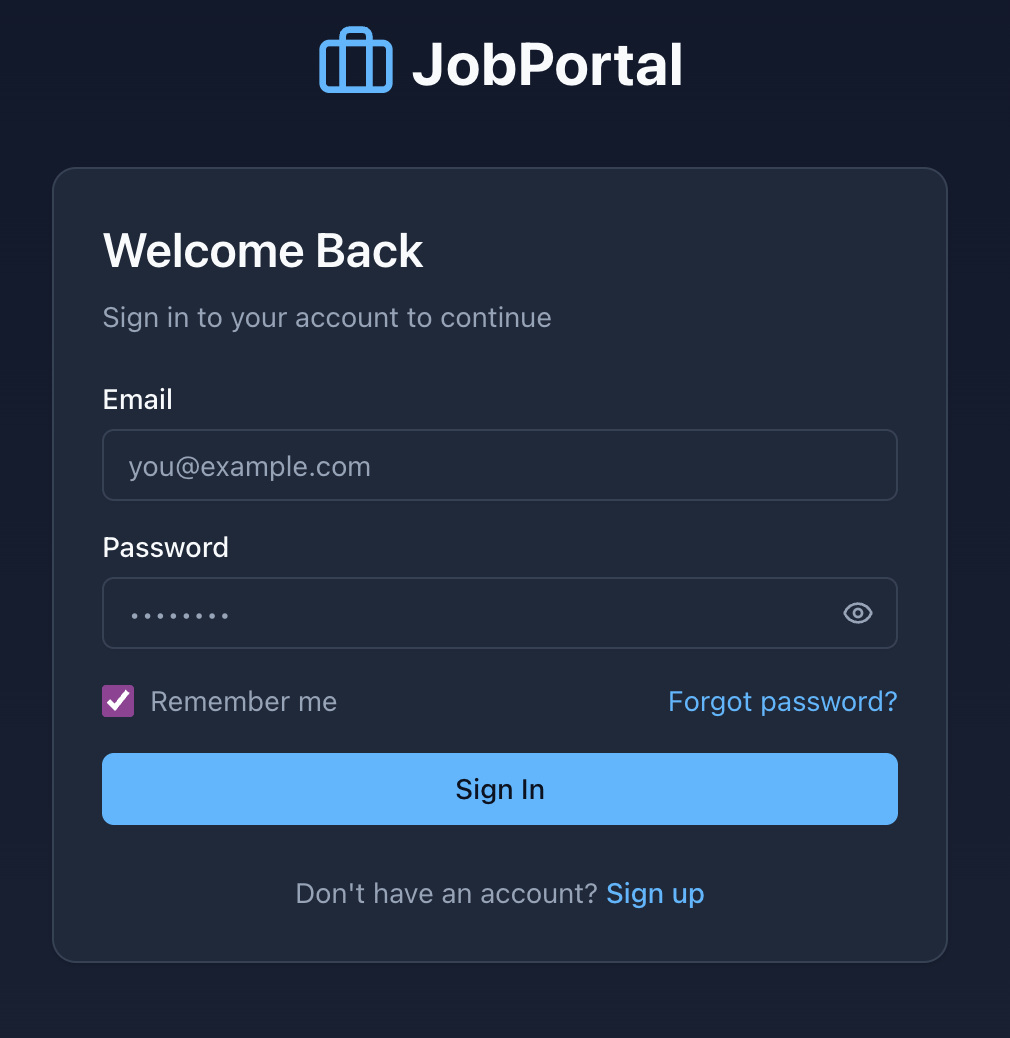
# APPENDICES

**Home Page UI**

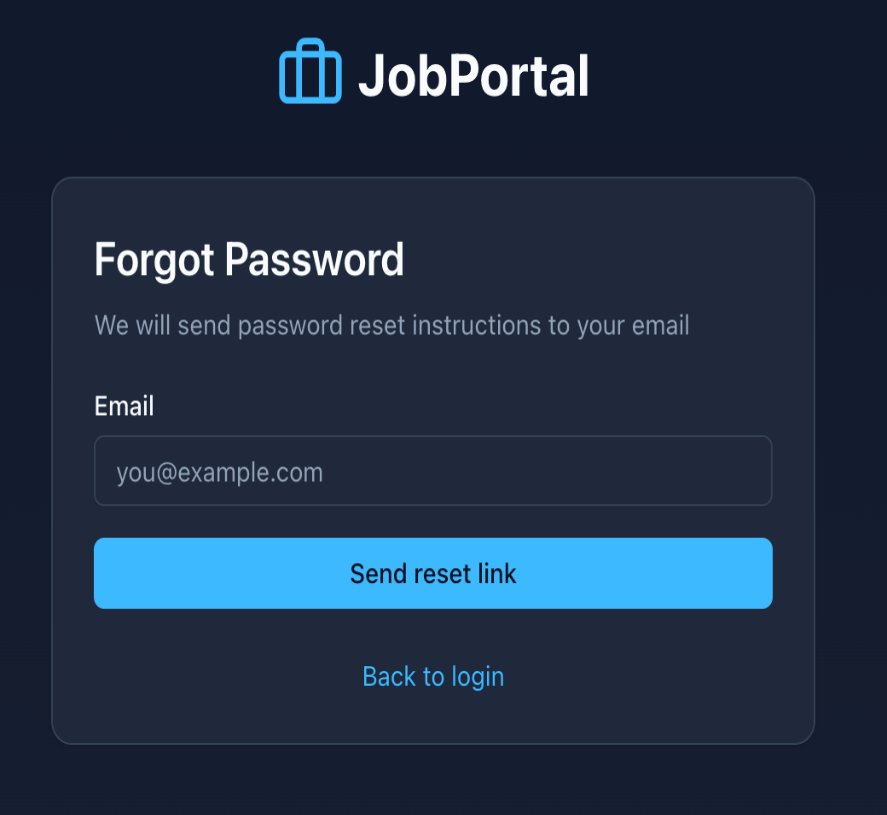
****

**Register Page UI**

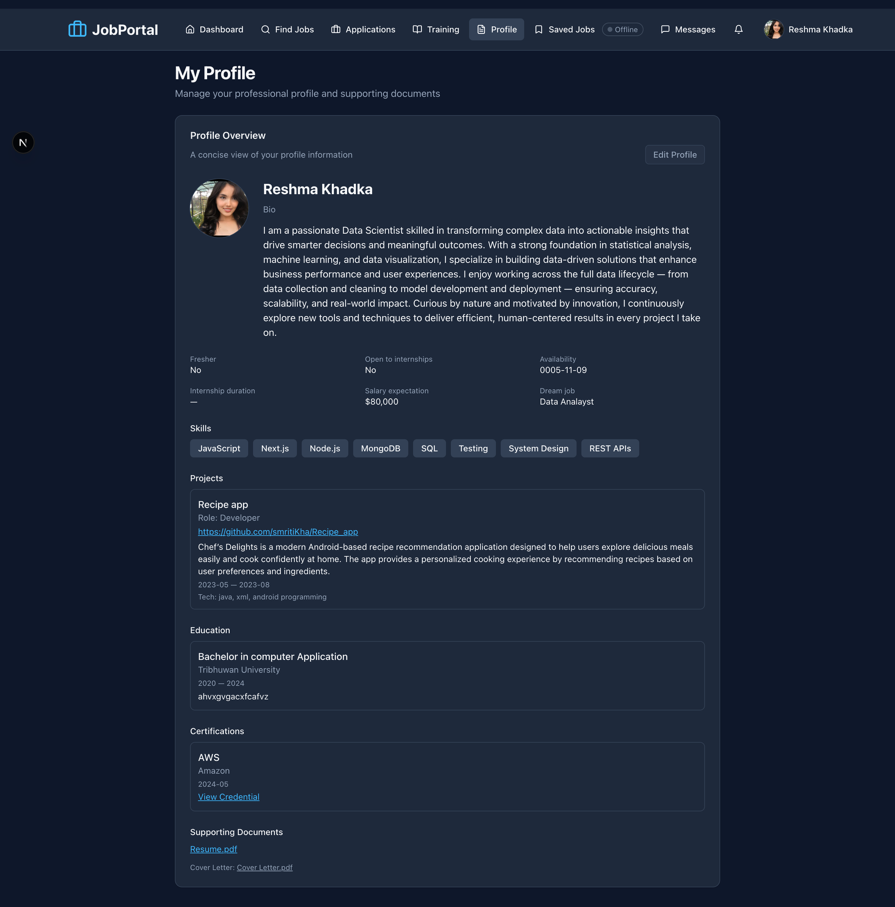
****

**Login Page UI**

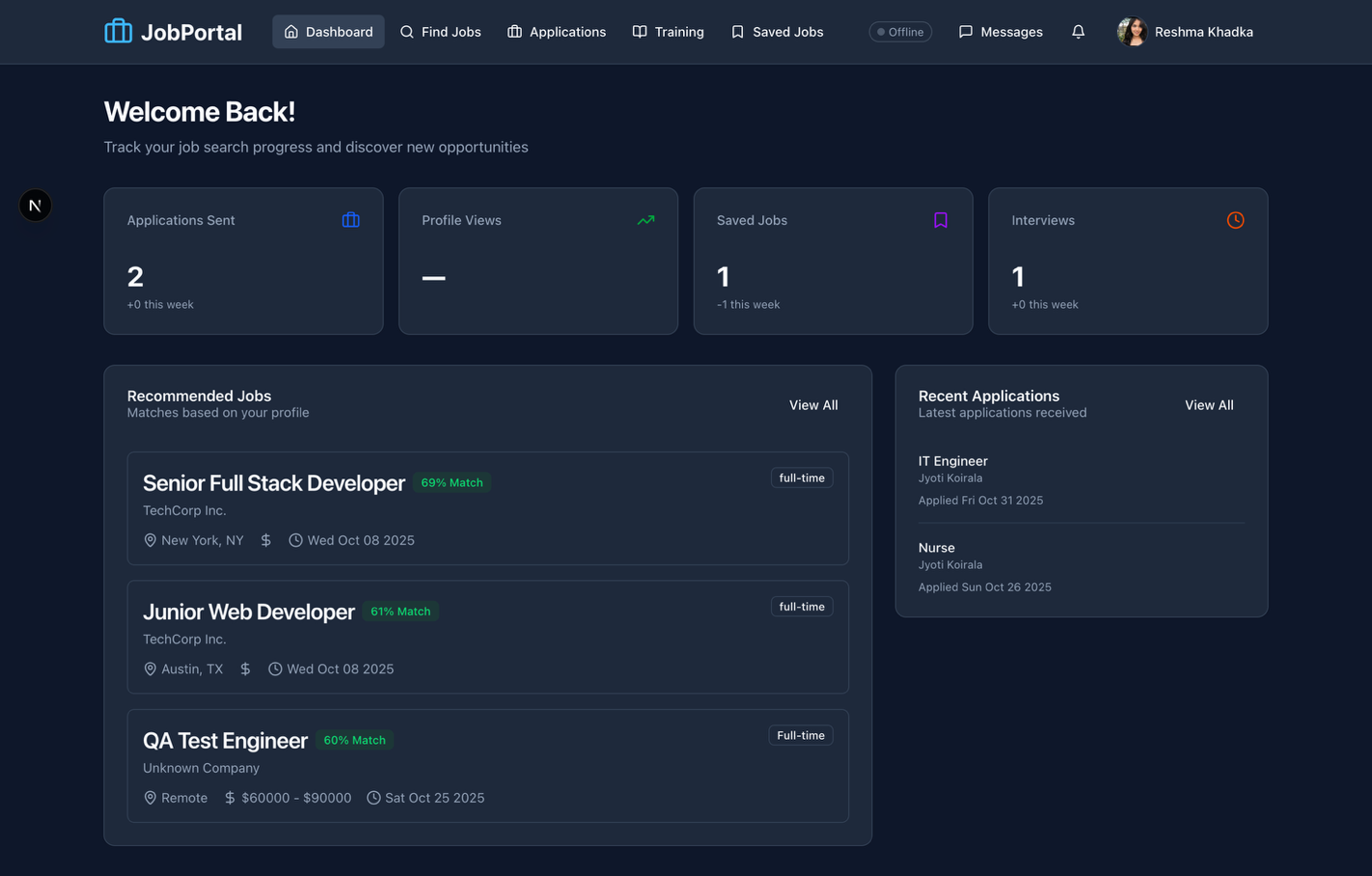
**Forget Password UI**

****

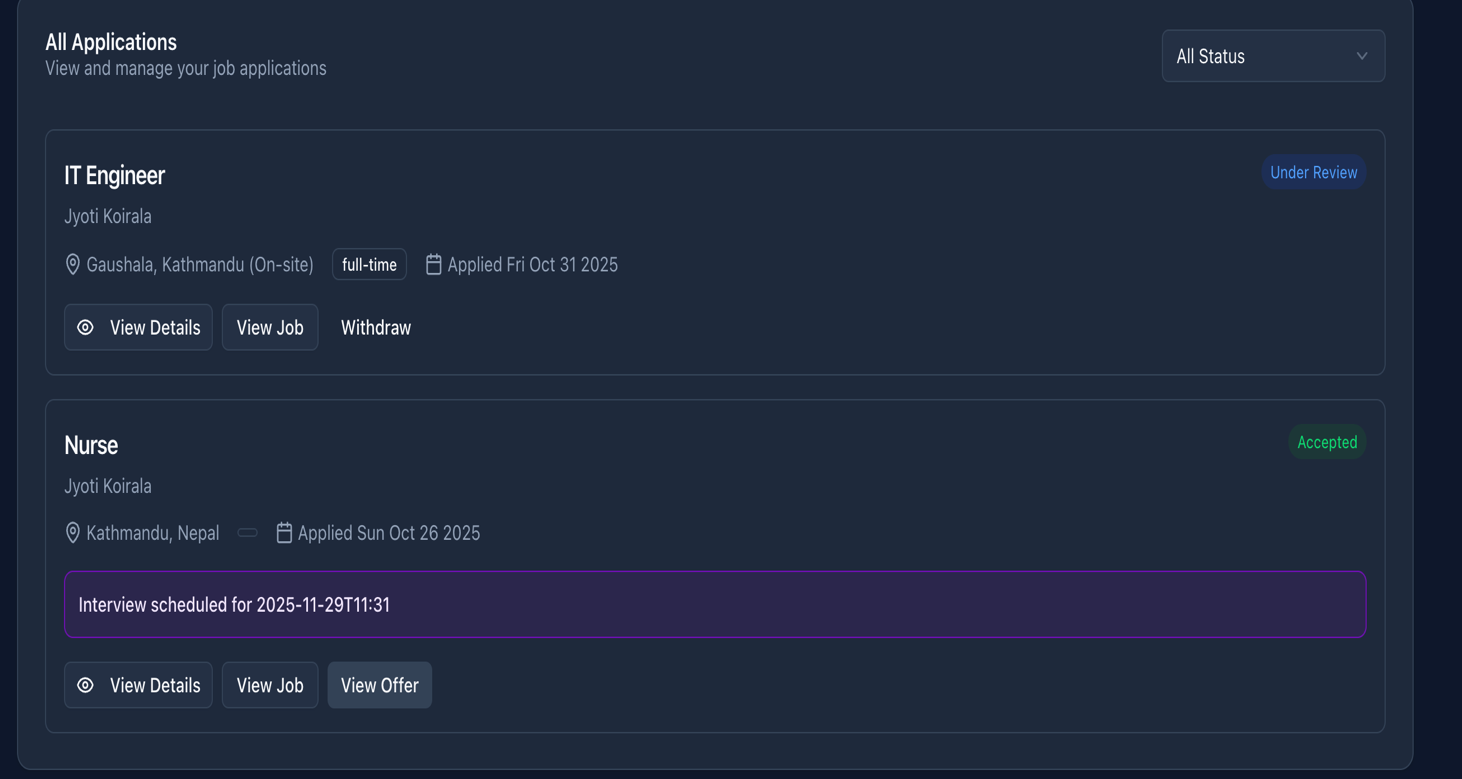
**Job Seeker Profile Page UI**

****

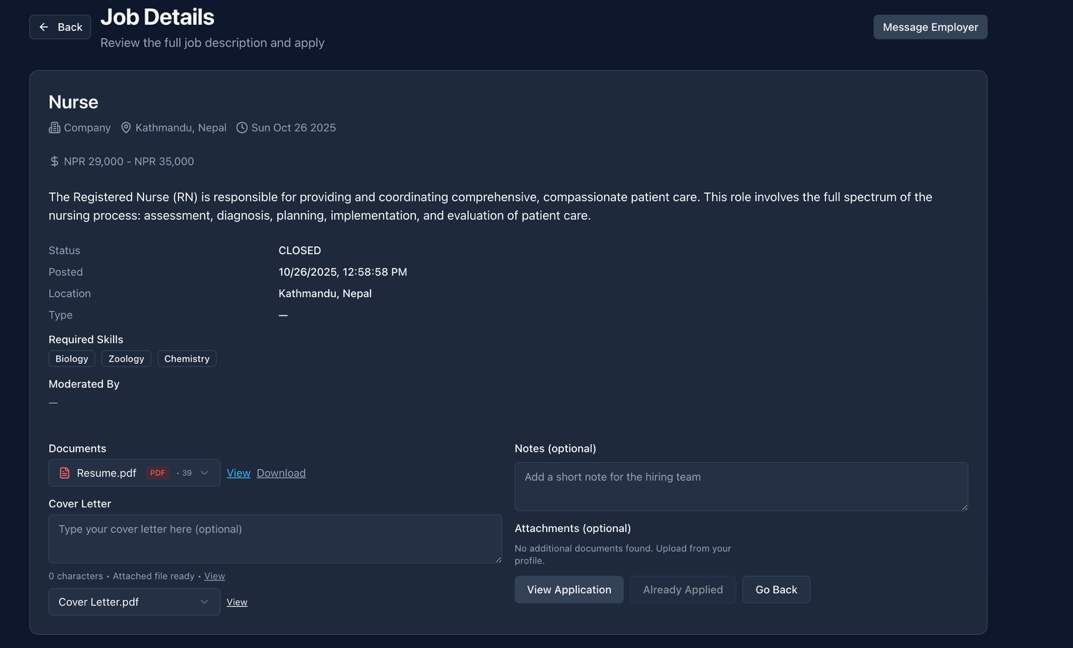
**Dashboard Page UI**



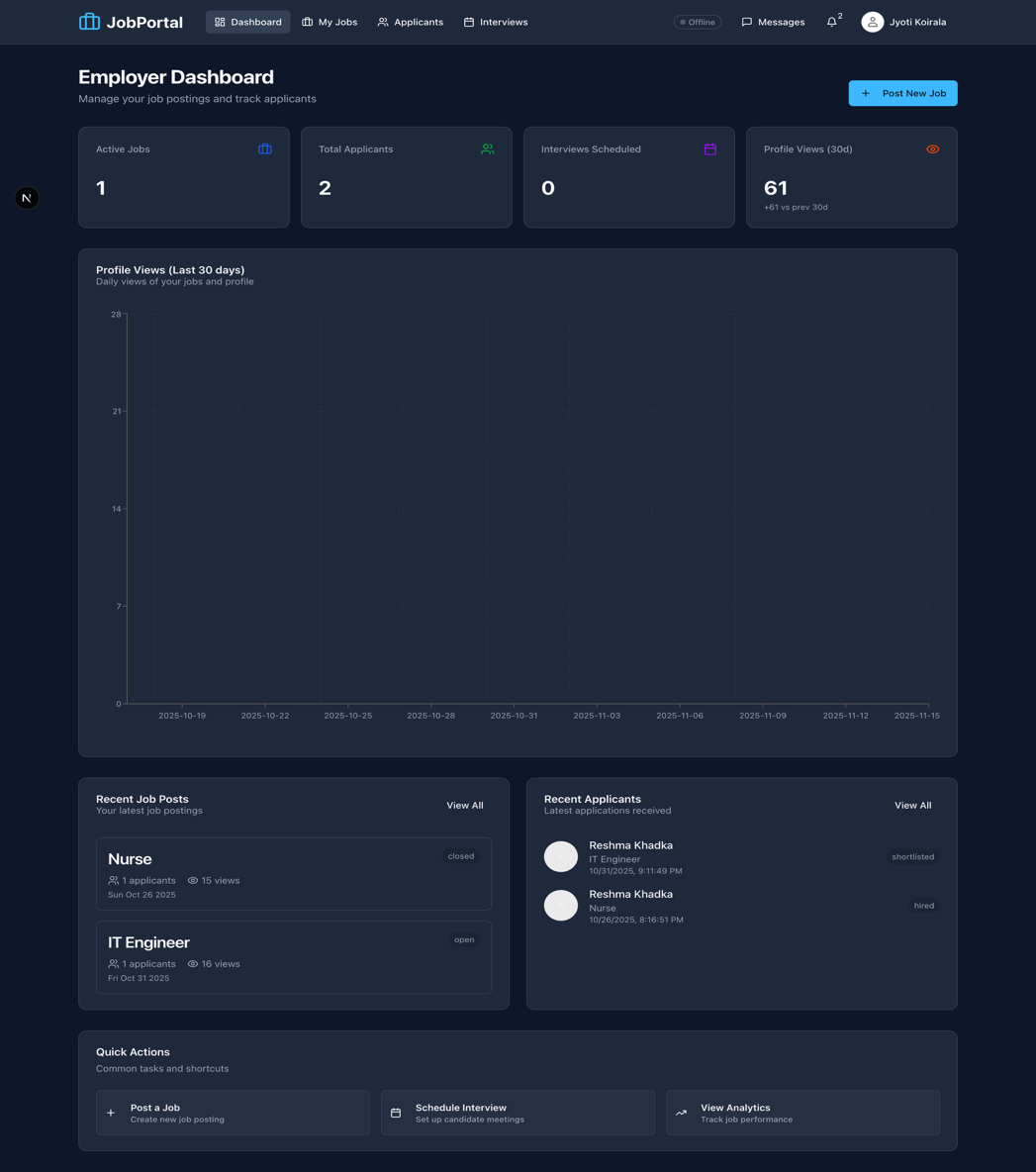
**Job Application Page UI**



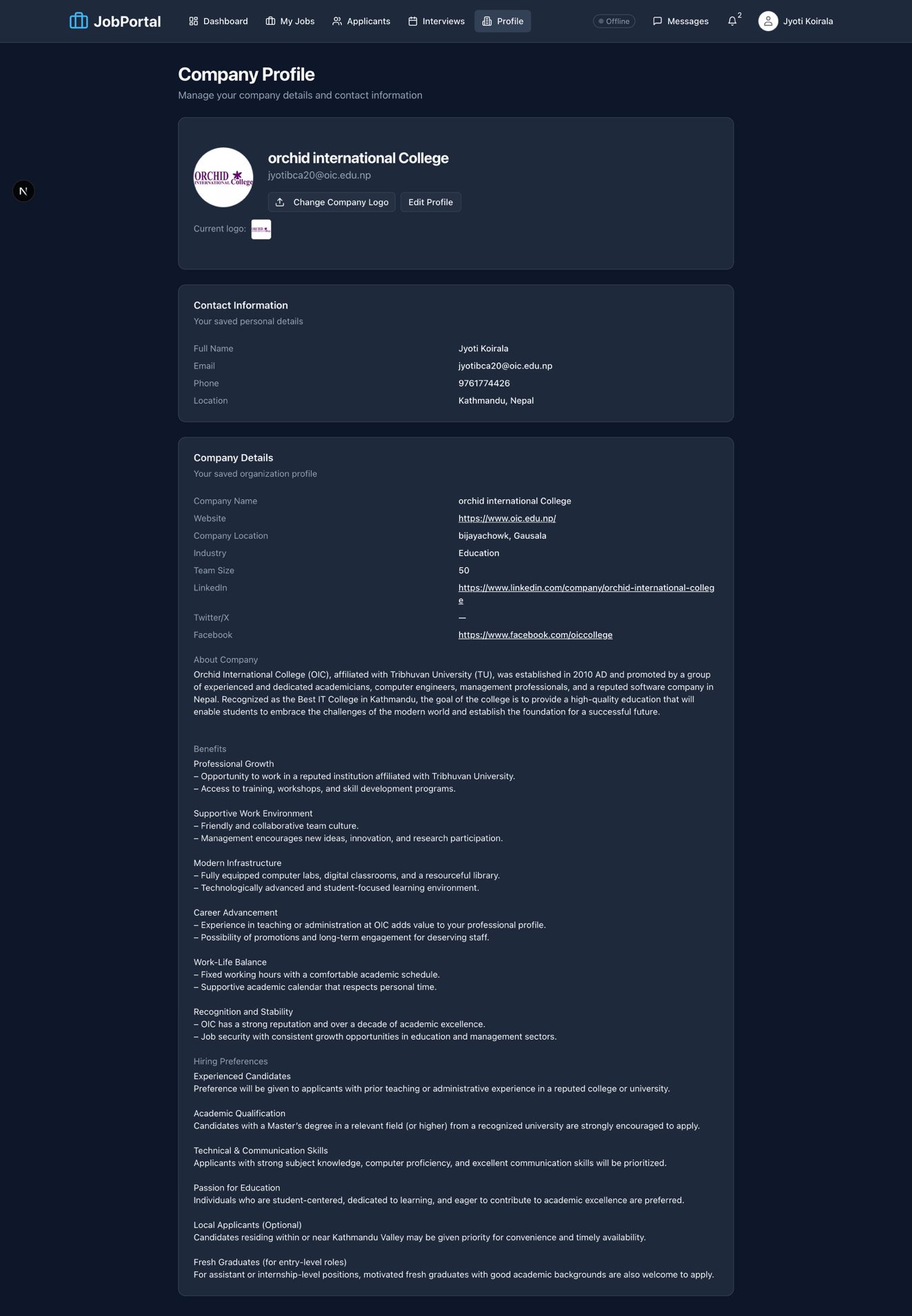
**Job Details Page UI**

****

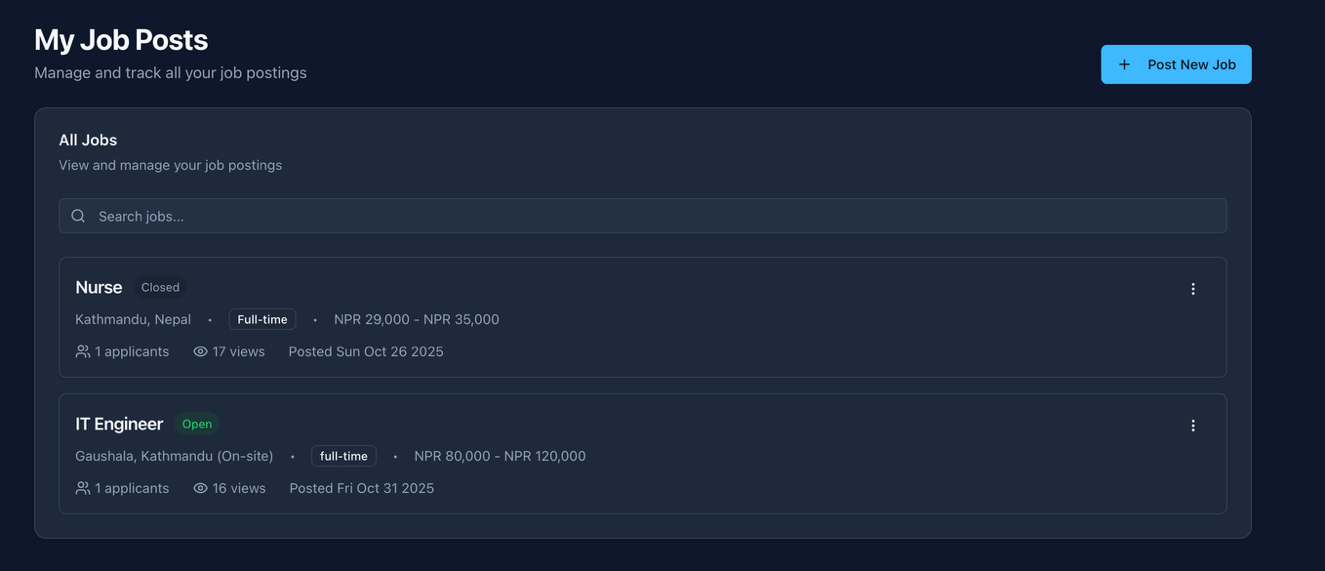
**Empoyer Dashboard UI**



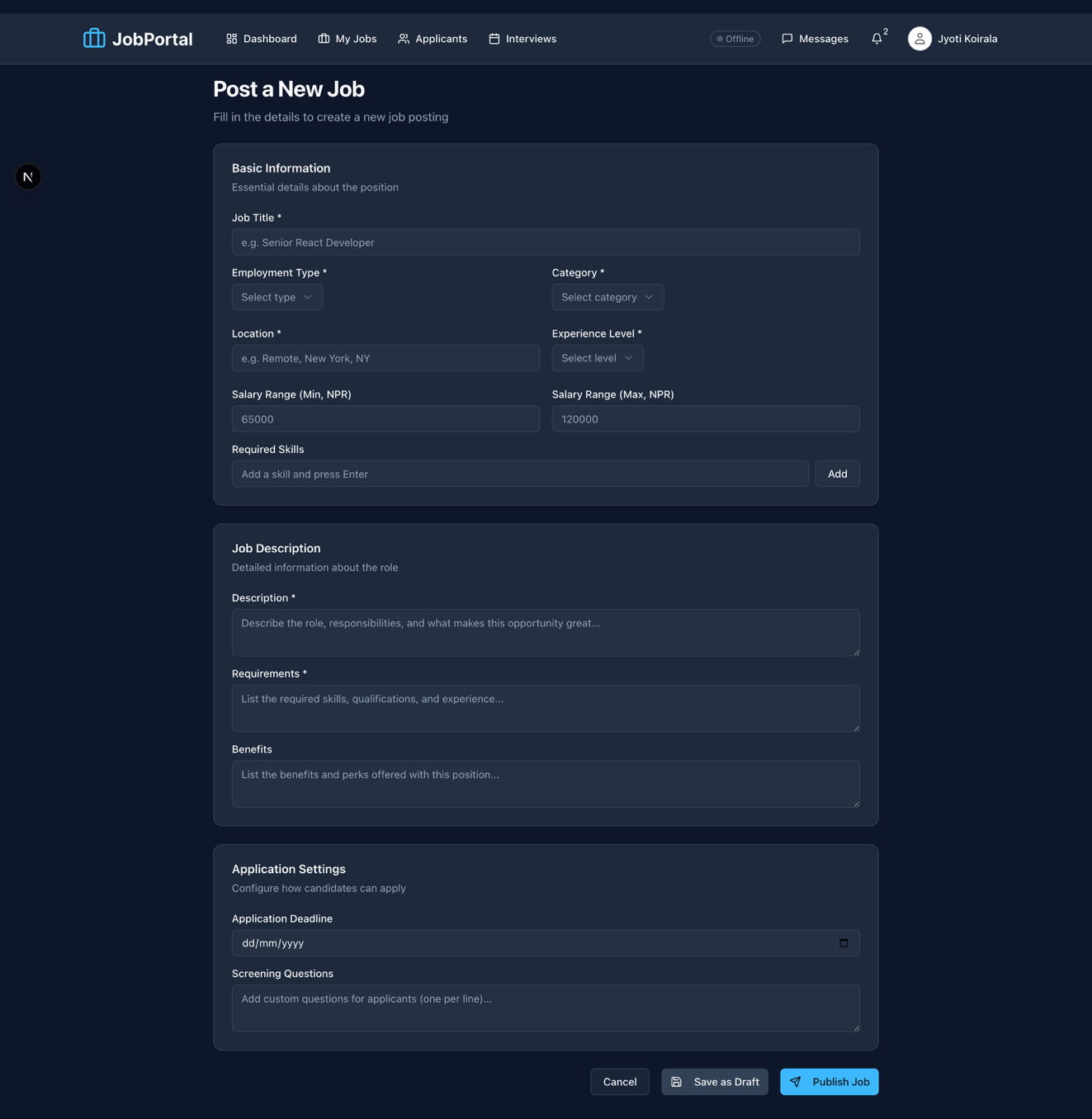
**Employer Profile Page UI**



**My Job Page UI**



**Post Job Page UI**

****