International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 06 | June - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Design and Development of a Modern Hybrid E-Recruitment System with MERN Architecture

Ashay Tiwari[1], Prof. Ashish Tiwari[2]

[1]*Ashay Tiwari, Computer Science & Swami Vivekananda College of Engineering, Indore*
[2] *Prof. Ashish Tiwari, Computer Science & Swami Vivekananda College of Engineering, Indore*

Abstract **-** In the evolving world of digital hiring, the integration of modern technologies with recruitment processes has become essential for achieving operational excellence, improved user engagement, and creating a robust and scalable system. This paper presents the design and development of a hybrid e-recruitment drive system utilizing the MERN architecture, comprising MongoDB, Express JS, React JS, and Node JS. This system provides a unified interface for both Candidates (Job Seekers) and Recruiters (Employers), offering personalized dashboards, profile creation (showcasing portfolios), job posting, application tracking, an intelligent auto job matching system, ATS-friendly resume builders and many interactive features. Emphasis is placed on creating a modular, scalable and secure web-based solution for small to medium enterprises (SMEs). This paper discusses the underlying system architecture, design methodology, feature set and potential impact on recruitment workflows. The results highlights improved usability and performance in comparison to traditional recruitment portals. This implementation-oriented study contributes a practical, tech-driven perspective to e-recruitment system development.

Keywords— MERN stack, Microservices Architecture, E-recruitment, RESTful apis, Text-Index Searching.

I. INTRODUCTION

As technology continues to reshape industrial workflows/trends, the recruitment process has undergone significant innovations. E-recruitment has become a significant factor for modern human resource management, allowing faster and more accessible hiring processes through digital systems. Modern hiring process requires intelligent systems that can bridge the gap between candidates (job seekers) and companies (employers) while ensuring a seamless user experience. Implementing efficient recruitment strategies enables organizations to attract high-potential candidates and enhances the working way of talent management practices [1].

However, legacy systems often fall short in addressing the dynamic needs of both recruiters and job seekers. Existing solutions tend to focus on either company-centric or candidate-centric workflows, rarely offering a balanced, hybrid approach. There is a growing need for solutions that support role-based experiences while remaining easy to maintain and scale.

This research presents an innovative hybrid e-recruitment application design developed using the MERN stack that tries to overcome these limitations. The system integrates advanced dynamic frontend components, a non-relational database and RESTful apis to offer a responsive, scalable, and maintainable recruitment solution. It aims to redefine how e-recruitment systems are designed by merging better implementation strategies with user-centered design. The paper discusses the design rationale, development methodology, architecture, and system design.

II. LITERATURE REVIEW

Effective recruitment strategies can provide organizations with benefits by recognizing and attracting potential employees, as well as positively enhancing the organization's talent management efforts overall [1]. E-recruitment sites can significantly decrease organizations recruitment costs and increase the speed of hiring to make it more efficient by streamlining automated resume screening or application tracking processes, etc. These e-recruitment sites give organizations a central repository for candidate information and a user-centered interface to integrate recruitment workflows and make real-time decisions collectively. There are some challenges with these systems, particularly issues with users inputting incorrect data and resumes being duplicated but the technology and system design are updating with form validation, AI or algorithm-based screening, and some kind of layered verification [2].

E-recruitment has changed traditional hiring methods by enabling a faster, cheaper, and broader search for employees. These systems provide better distribution of job postings while also assisting with targeted audience selection, identifying candidates, and tracking applicants. Using e-recruitment platforms along with existing human resources systems enhances organizational scalability and efficiency in the operational strategies of talent acquisition [3]. Today's e-recruitment platforms combine job seekers and employers into ideal ecosystems for interaction. By enabling a consistent application process through the reuse of stored data, integrated testing, and identifying qualified candidates quickly, these systems ease the application process for all parties. Consequently, organizations experience shorter time-to-hire, cost savings, and a more accurate shortlist, ultimately resulting in a more reliable recruitment process overall [4].

The MERN stack, which consists of Mongodb, Express, React JS, and Node JS. It has become the most recognized and trendy architecture for full-stack development. The Node JS backend provides excellent performance for input-output concentrated applications, while React's virtual DOM and component-based approach help developing highly responsive user interfaces a straightforward task. Mongodb's flexibility and schemaless format allow for easy management of data. All of these components of the MERN stack provided us the high-performance solution for developing robust, scalable, and efficient applications [5].

III. PROPOSED METHODOLOGY

3.1 Overview of E-Recruitment System

The e-recruitment systems help organizations to ease and digitalize their process of hiring people by providing a common interface for candidates (job seekers) and companies (recruiters) to interact with each other. This helps the companies to hire the appropriate candidates for their vacant positions by excluding paperwork, providing faster time in their recruitment process, and increasing their threshold efficiency of recruiting by the automated process. This recruitment process can be categorized into two on the basis of line of enquiry: internal and external recruitment.

Internal recruitment refers to the replacement/promotion of existing employees in the organization or so to fill available vacancies. One of the most known internal recruitment strategies is known as "promoting from within", for example providing advanced opportunities for junior staff to progress into more responsible senior roles. This recruitment process seems as a time and cost-effective way as you don't have to recruit and assess candidates from scratch, as you already have a person with ground knowledge of a field with experience in that domain. Even it is considered that internal recruitment improves employee satisfaction and develops loyalty in employees.

On the other hand, external recruiting means the candidate will be hired or screened from outside of the organization. This is commonly achieved through posting and advertising jobs on job portals, social media, etc.

This method is a little expensive and time-consuming than the internal approach, but it provides opportunities to a broader and more skilled set of people. Although external hiring method can enhance the level of competition in the workplace, which helps in inviting new ideas and working methodology [6].

3.2 Technology and Tools used

To build an efficient and scalable e-recruitment platform, a modern, robust and scalable tech stack is required. The chosen stack for this application is MERN stack, which consists of Mongodb, Express, React JS and Node JS as we have already discussed. This fullstack javascript solution provides a seamless development cycle across the both frontend and backend, enabling faster application development and easier maintenance cycle.

Mongodb: Mongodb is a popular NoSQL document database that stores data in flexible, JSON-like documents called BSONs in the document format stored or clubbed in a collection. Unlike traditional relational databases, Mongodb doesn't use tables instead it stores data in flexible BSON documents that can vary from document to document. This makes it highly appropriate for storing and querying data with diverse structures. Mongodb documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents [7].

Express: Express JS is a widely used framework for Node JS, serving as the foundation for several other frameworks that ease the development lifecycle. It is not needed to write the code from scratch or reinvent the wheel. This framework helps developers to define api routes by specifying different HTTP methods (get, post, put, delete, patch etc.) and customized URL paths, making it easy to manage and segregate different request types. One of the greatest advantages of this framework is its middleware support, which helps us in writing our own custom validation logic for each or clubbed api requests without any need to without copying/pasting the code [8].

React JS: React is a javascript library designed by Meta for building user interfaces, which is mainly focused on the development of dynamic and interactive UIs by using component based approach. Although often referred to as a framework, React is technically a library and can be used beyond the web - for example, in mobile app development using React Native. React's modular structure helps reduce bugs and allows developers to focus more on the user experience while react handles much more complex rendering logic via Virtual DOM in the background[9].

Node: Node JS is an open-source, cross-platform runtime environment that enables developers to build server-side applications using javascript. Unlike traditional Javascript that runs in the browser, Node JS operates directly on the operating system, making it best choice for backend development. It doesn't includes browser-specific features and instead provides access to operating system-level APIs such as file handling, networking, and HTTP, allowing developers to build scalable and efficient server applications outside of the browser client-side environment [8].

The Google Trend report (figure 1) compares the world side interest in various web development stacks - MERN, MEAN, MEVN, PERN and LAMP over past 5 years. It clearly shows that the MERN stack has gained significant attention and popularity, reflecting developers growing preference for modern, javascript based technologies [10].

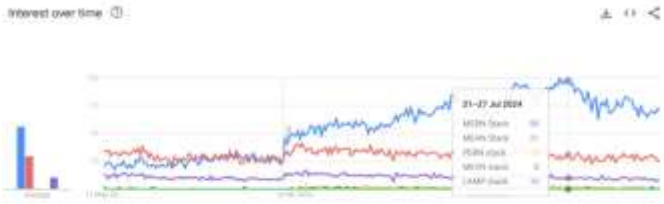

Fig- 1: Google Trends Report comparing different tech stacks.

During the actual development process, there are many tools that were utilized to help us make development journey smoother, enhance productivity, support for deployment and management of the application. These tools played an important role in both frontend and backend development, API handling, version control and cloud deployment. Below are some of the mentioned tools that we needs to use:

Visual Studio Code (VS Code): A Lightweight and powerful code editor with powerful support for javascript, typescript environments, a vast community for VS extensions, in-app debugging tools, source control and many more.

GitHub: A free platform for hosting and managing code repositories, handling version control and team collaborations.

Redux: One of the significant library to manage client-side states efficiently on Javascript apps, commonly used with React to manage application state more efficiently. Redux helps us in avoid props drilling practices in react. We had utilized Redux-Toolkit library, which provides a more advanced and simpler approach to manage redux workflows e.g., actions, states, reducers etc,. through the uses of redux slices.

Swagger: Swagger is a powerful tools designed to helps us document and test the REST APIs efficiently, while it also

provides a common user-centric interface to provide clarity and uniformity throughout the development journey.

Postman: An API testing tool for making requests to endpoints and interpreting responses during the backend process for development and debugging.

AWS S3: A cloud storage solution from Amazon that is used for the security and recovery of an application's assets like images, documents, videos and other media files.

AWS EC2: A scalable virtual server service from Amazon that can be used to host and run web applications with a reliable and flexible deployment option.

ngrok / Local Tunneling: A tunneling service useful for exposing a local development server to the internet, and can be used for webhook testing, or sharing a local site while it is under development [11].

Docker: A platform for developing, shipping and running applications in an isolated container, while achieving the same functionality independently of the environment stack.

Nginx: A high performance web server and reverse proxy used to serve static content, manage load balancing, and overall application performance.

MongoDB Atlas: A fully managed cloud database service for a MongoDB Database, that provides scalability, security, and global distribution while eliminating the need for infrastructure management.

3.3 User Roles and Features access

This recruitment system is designed to support and provide services to a diverse set of users, each having a specific set of access rights and responsibilities. The feature available in the platform completely depends on the role of the users. Below is an overview of the different user types and their core functionalities:

Super Admin (Administrator)

a. Has full control over the activities of the application.
b. Can create and manage sub-admins.
c. Monitors all user activities across the system.
d. Manages feature configurations, system-level settings, and reports.
e. Interact with Users via admin posts, newsletters, etc.

Sub Admin

a. Manages users, job listings, and reported content under supervision.
b. Can verify companies and candidates.
c. Onboard Event World and Benefit World Partners.
d. Handles moderation tasks and approves posts or events.
e. Generates and views limited analytics and reports.

Verified Candidate (Stream Member)

a. Has completed full profile verification.
b. Can view and apply to job postings.
c. Can access the application's ATS-friendly resume builder.
d. Manage profile portfolio.
e. Access to participate in different events organized and managed by Event World Partners.
f. Access to different benefits and offers by Benefit World Partners.

Unverified Candidate (Talentpool Member)

a. Has limited access until verification.
b. Manage profile portfolio.
c. Request for profile verification.
d. Interact with Admin Posts.

Verified Company (Stream Member)

a. Authorized to post jobs and manage applications.
b. Can view detailed candidate profiles.
c. Access to the in-app Applicant Tracking System.
d. Access to participate in different events organized and managed by Event World Partners.
e. Access to different benefits and offers by Benefit World Partners.
f. Manage the Organization's portfolio.
g. Access to the in-app Talent Admin Panel.

Unverified Company (New Company)

a. Has limited access until verification.
b. Manage the Organization's portfolio.
c. Request for profile verification.
d. Interact with Admin Posts.

Event World Partner (Event Affiliates)

a. Can create and manage events, webinars, and workshops.
b. Access to the in-app promotional tools to reach targeted user segments.
c. Can view attendee data and engagement insights.

Benefit World Partner (Benefit Affiliates)

a. Can showcase products or services on the platform.
b. Access to campaign performance metrics.
c. Engages with users via discount codes, offers, and third-party purchase links.

3.4 System Architecture

The system architecture (figure 2) of this e-recruitment platform is designed using a module and scalable three-tier architecture, which provides separation of concerns, it is easy to maintain and provide better future scalability. This architecture is broadly divided into three layers such as: Presentation tier (frontend), Application tier (backend), and Data Management tier (database). The frontend of the application is built using React JS which helps us in creating interactive and dynamic user interfaces. This layer communicates with the backend through API requests as we are using REST apis. This enables a decoupled architecture that supports flexibility and customizability in development and deployment.

As the main core or thinking brain of the system, the backend tier is mainly developed using Node JS and Express, which act as a gateway or bridge between the user interface and databases. This layer is responsible for handling routings, business logics, authentication, authorization, data processing and many more. It follows the MVC (Model View Controller) architecture. Integrating the backend system with tools like Swagger and Postman ensures efficient API documentation and testing throughout the application development lifecycle.

At the end, we have the Data Management layer, which consists of a document-oriented database, Mongodb, used for storing user records, application data, events, benefits, different metadata etc. In addition, AWS S3 is utilized for storing assets or media files such as resumes, profile images, post images etc., while an AWS EC2 instance will be responsible for hosting the deployed application over the cloud. Supporting tools such as Docker, ngrok, nginx etc will be used in development and deployment scalability. This well structured block architecture ensures a robust, secure and flexible environment for handling the complex and ever changing needs of a modern hybrid e-recruitment system.

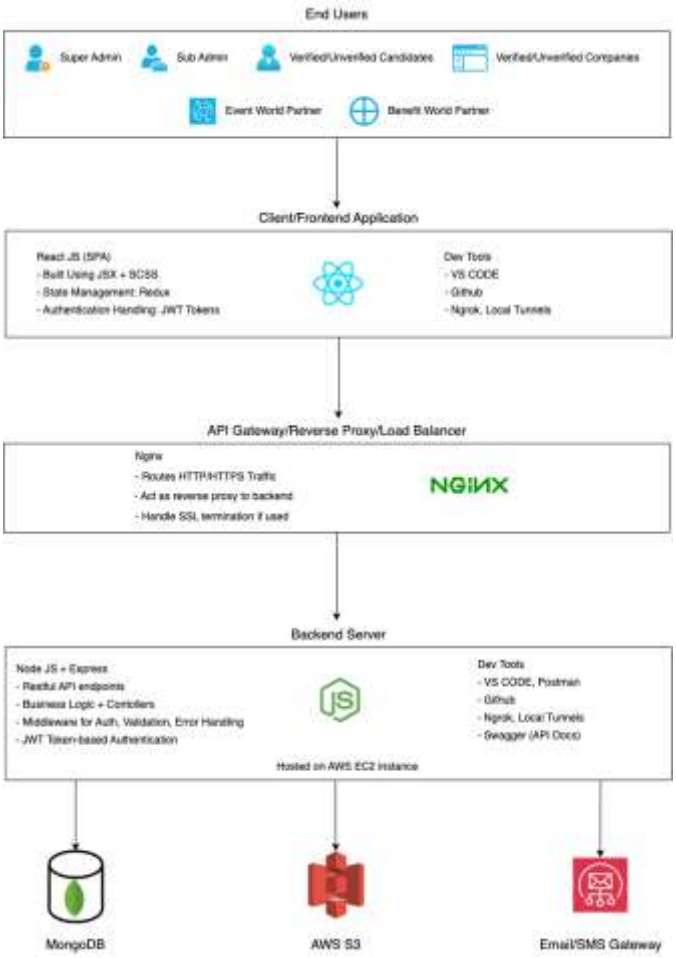

Fig- 2: Proposed Architecture of E-recruitment system

3.5 System Ontology

The E-recruitment system ontology serves as a structured framework that represents the various concepts, roles and relationships involved in an online recruitment system. It helps to standardize the terminology and processes used within the domain, enabling better data flow bindings, search accuracy and intelligent matching between job seekers and employers. Figure 3 presents a visual representation of the proposed ontology for the online recruitment system. The datatype properties are not included in the ontology diagram for better readability.

The ontology is composed mainly of fourteen entities: Candidate, Employee, Employer, Company, Job Offer (Job Application), Language, Skills, Work Experience, Education, Partners, Event World Partner, Benefit World Partner, Events and Benefits (Offers). In this ontology, we have the Employee entity, which is a subclass of Candidate. This Employee will be interacted with and correlated by the Employer which is a subclass of Company. Employee profile has different sets of sub-entities to make a proper profile portfolio as Skills, Language, Work Experience, Education and many more. Employer (Company) will be responsible for creating job applications (Job Offers) on the platform. These Job Offers will be intelligently linked with Candidate profiles using an index-based search approach.

To intelligently match the Job Applications with the Candidate profile, our system uses MongoDB's full-text indexing capabilities [12]. By creating indexes for key fields of Candidate profile, such as job titles, descriptions, required skills, work experience, education and many other meta details, we enable efficient and context-aware search operations at runtime, The used algorithm dynamically calculates the score for each job based on how closely it matches with candidate profiles, These scores are used to present the most recommended jobs at the top of the list. Additionally, the system allows Candidates with flexible filtering options, allowing them to customize their job preferences based on their needs.

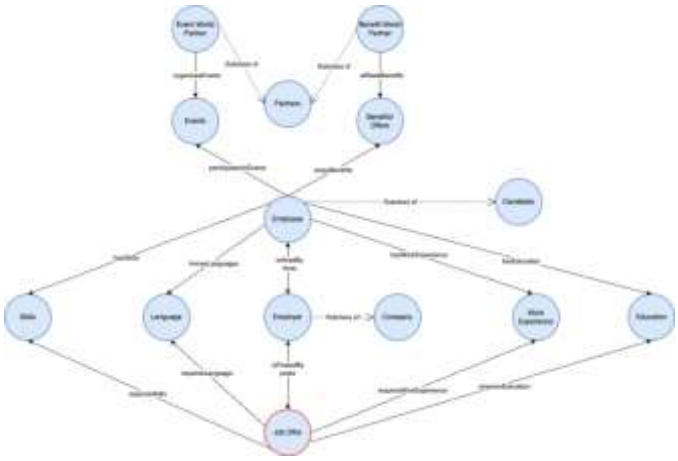

Fig- 3: Proposed System Ontology for E-recruitment system

IV. IMPLEMENTATION

In this section, we will look at how to use the proposed methodology and ontology to create a highly efficient, robust, and scalable e-recruitment system. The main thing that we need to keep in mind is that we are going to create an application that will help humans, not replace them. The application's goal is to provide valuable support in managing the recruitment process workflow for both job seekers and recruiters. As discussed, we are going to use the MERN stack (full-stack) based web application for the implementation of the e-recruitment system. We will further divide this section into different domains of the MERN stack-based web application to properly understand how to design the core of the application. We are using a three-tier architecture for our e-recruitment system. A three-tier architecture is a modular client-server architecture that provides three different layers: *Presentation tier, Application tier,* and *Data tier.*

We are going to use React SPA (Single Page Application) as a presentation layer, which will be responsible for providing end client-side interface. This presentation layer is going to interact with the other tiers using Application Programming Interface calls. At the Application tier, we have Node JS + Express backend system, which will be responsible for handling the core business logic. We are using MongoDB Atlas at the Data layer, which consists of a database and programs to manage internal reads and writes.

4.1 Frontend Implementation

The frontend of the e-recruitment system is created in React JS, a powerful javascript library for building SPA user interfaces. Styling is handled using SCSS modules, which allow for scoped and modular styles and avoid global style conflicts in React environment. To efficiently manage server-side state and handle data fetching operations, React Query has been used. TanStack Query (formerly known as React Query) is often described as the missing data-fetching library for web applications, but in more

technical terms, it makes fetching, caching, synchronizing and updating server state in your web applications a breeze [13].
Redux is our choice for global and shared state (for example, profile info, configuration flags for the UI, or user attributes tied to roles). React Query manages remote data (like fetching job listings or user applications), and Redux manages shared state that can be reused in multiple components and is not a direct result of server calls. This hybrid approach makes sure everything has its own responsibility and works efficiently.
To save on build size while preventing unwanted dependencies, we specifically avoided third-party UI libraries or component engines. Instead, we built our own custom UI components using advanced React features. We implemented modals using React Portals. React Portals allow a component to be rendered outside the parent DOM hierarchy while maintaining its state and behaviour. React Portals are useful for overlays in your UI, such as modals, tooltips, and dropdowns, where the current location of the DOM is important to the user experience.
The system features multiple frontend pages, bound with specific user roles. These include user authentication pages like login, registration, forgot password, dashboard views for candidates, companies, event and benefit partners, and super/sub-admins. Each role has access to its specific profile pages with additional sections like job listings, resume builders, talent admin panels, partner organizer pages etc. This frontend design will help us in creating a modular and scalable structure for the e-recruitment system (figure 4).

Fig- 4: E-recruitment system frontend hierarchy

4.2 Backend Implementation

The backend of this system is developed using Node JS and Express as discussed above, which ensures high performance and scalable server side architecture. To helps us modularize the development and scalability, backend system is structured into different microservices, each will be responsible for handling a specific business domain, logics and requirements. The main microservices includes:

Authentication Service: As the name of the service suggests, it will be responsible for handling all the authentication-related tasks such as user registration, login, forgot password workflow, token-based authentication and authorization using JWT (JSON Web Token). This ensures all endpoints are securely protected and accessible to only authorized users.
Master Service: Handles centralized data and configurations, such as roles, categories, filters, and lookup entities used across different modules.
Main Service: This service acts as the core business logic layer, which is responsible for managing job posts, user interactions, applications, resumes, dashboards, and other primary functionalities.
Shared Utility Module: This module **is** developed as a Git submodule, this reusable module service provides helper functions, constants, and common logic shared across all services to reduce duplication and increase maintainability.

While developing any application, it is very important to document the code workflows, any specific configurations, third party dependencies etc. For documenting the REST apis we are using Swagger Autogen which will be responsible for automatically generating the documentation with minimal amount of code. This helps us maintain up-to-date interactive documentation for all endpoints with different endpoints and structure, improving development productivity and communication between teams. For communication, alerts, notifications, the backend system is integrated with Nodemailer and Mailtrap for safe and efficient emails deliveries during development and testing phase. To implement security and access control over the system, we uses a middleware-based approach provided by Express. Rolebased restrictions are implemented through custom middlewares that validate user before granting access to specific resources. This combination of modular microservices, robust authentication and authorization capabilities, formal and inetractive documentation helps the backend to offer flexibility and security which is an essential factore for designing multirole e-recruitment system.

4.3 Database Implementation

For the database layer, the system utilizes Mongodb, a NoSQL document-oriented database that provides high performance, scalability and flexibility in handling a large volume of records. Given the constantly changing and heterogeneous nature of the data in an e-recruitment platform (user profiles, job profiles, job applications, event tracking), MongoDB is the best option.
To communicate with the database efficiently, the application uses Mongoose, which is an Object Data Modeling (ODM) library for MongoDB and Node JS. Mongoose provides a schema-based solution to modeling the application data, as well as validation, query building, and business logic at the data level. Each entity in the system—such as candidates, employers, jobs, and partners—is defined as a separate schema, resulting in neat models for our data and their relationships.

4.4 Deployment Strategy

To ensure high availability, scalability and cost-effective resource management, the e-recruitment system is deployed using a cloud-native approach. The frontend application is deployed using AWS S3 as a static website. S3 provides a highly durable and scalable storage service that supports fast content delivery. The application uses a Mongodb Atlas, an awfully managed cloud database service to host Mongodb instance. This ensures automated backups, monitoring, scaling and security controls such as IP whitelisting and TLS encryption. The backend architecture follows a microservices pattern, where services such as Authentication, Master and Main services are containerized using Docker. Each services run independently, allowing modular updates and better full isolation. These services are deployed on AWS EC2 instances, which provide full control over the server environment. Docker helps us in ensuring environment consistency, simplified deployment and isolation between services. To efficiently manage incoming HTTP requests and route them to the correct microservices, we are using nginx as a reverse proxy server. Nginx handles load balancing and URL-based routing, allowing all backend APIs to

be served through a unified domain (figure 5).

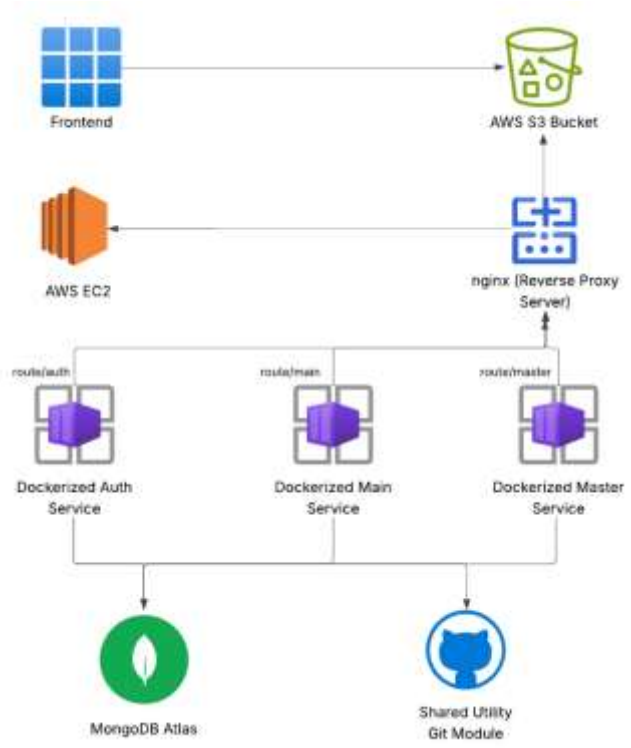

Fig- 5: E-recruitment system deployment strategy

V. RESULT

The proposed e-recruitment system was successfully implemented using modern, scalable architecture backed by microservices and intelligent matching logic. This system demonstrates robust performance in handling various user roles and responsibilities while ensuring a seamless user experience throughout the application. The frontend is created by React with modular SCSS and custom UI components, which provides clean and natural navigation for all user types: Super Admin, Sub Admin, Candidate, Company, Partners etc.

One of the key achievements of this system is the real-time job recommendation system, which uses Mongodb's full-text index based search to match candidates to jobs based on profile content. This allows the platform to dynamically calculate and assign a relevancy score based on the candidate's profile and activities.

The backend is organized into distinct microservices that allow independent development, scaling and deployment of core modules like authentication, main, master etc. By using docker and deploying services like AWS EC2 with nginx as reverse proxy server, we ensured high availability, load balancing and cleaner routing services. Below are selected screenshots (figures 6-9) that showcase different features and user interfaces of the system. All screenshots display placeholder or dummy data for demonstration purposes only.

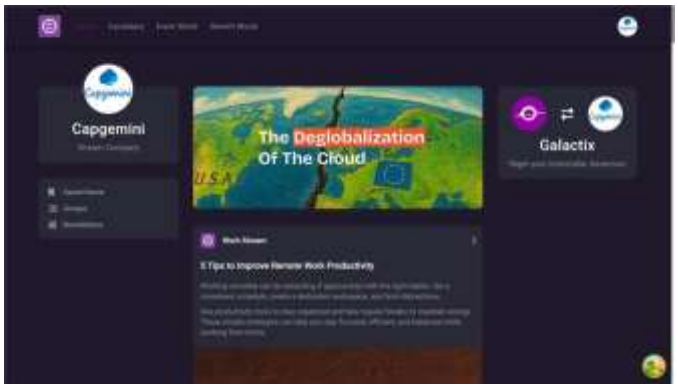

Fig- 6: Company's Community Page

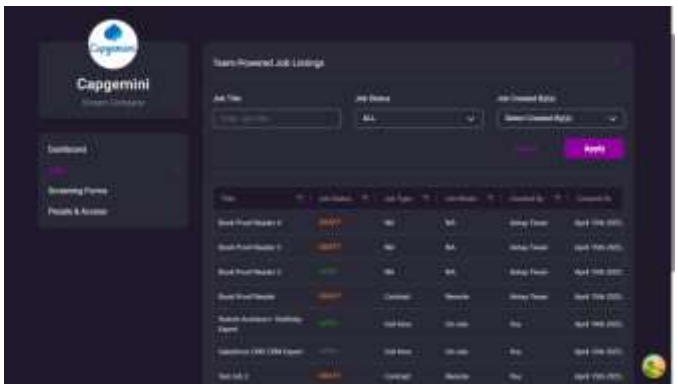

Fig- 7: Company's Job Editor Page

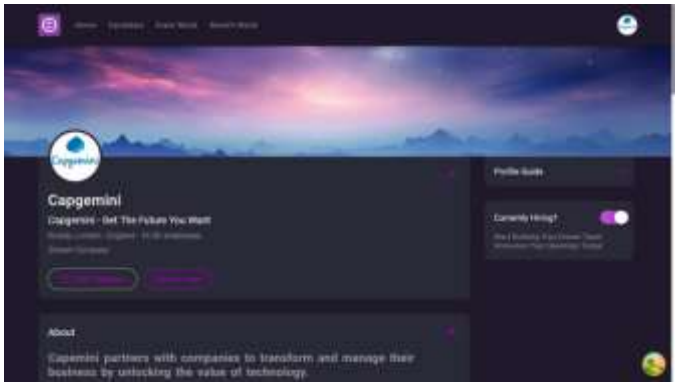

Fig- 8: Company's Profile Page

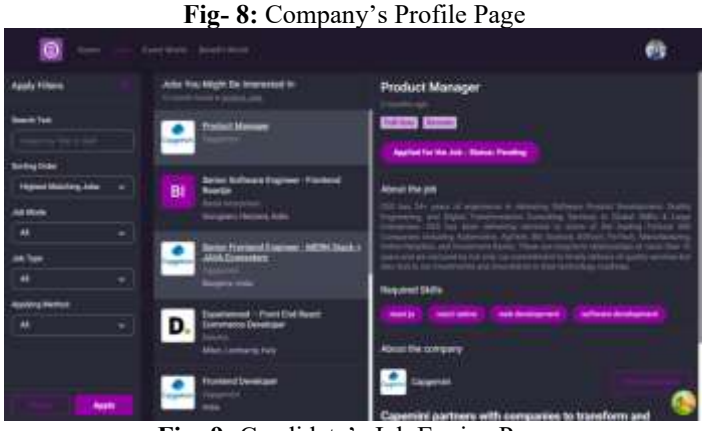

Fig- 9: Candidate's Job Engine Page

VI. CONCLUSION

The development and implementation of the proposed e-recruitment system represents a significant step towards modernizing and gap filler between candidates and recruiters in this digital era. By integrating intelligent features like real-time job matching, modular microservices and role-based access

controls, this proposed system not only enhances the e-recruitment system but also tries to overcome the gap found in traditional systems. This research demonstrates how proper architecture combined with the right set of technologies - React, Node JS, Express, Mongodb, Mongoose, Docker, AWS, and many more can deliver a robust, flexible, scalable and maintainable solution for complex use cases. This system is designed to scale with growing demand, accommodate diverse workflows, and maintain data integrity. As final words, the implemented system successfully meets the research and development goals and serves as a strong foundation for continuous innovation in the e-recruitment domain.

VII. FUTURE SCOPE

In the future, system can be significantly enhanced through the development of a cross-platform mobile application, by consuming the existing RESTful apis and modular backend architecture for seamless integration.

As we explore the future scope of the proposed system, it's very important to note that its scalability and direction may evolve based on market trends, technological shifts etc. Nevertheless there are several interactive features that can significantly enhance user experience, let's take a look at some of them.

Introducing new features like Kudos or Endorsement system will enable candidates to receive recognition from peer members and recruiters, which adds a layer of social credibility to their profiles. A skill assessment module consists of quizzes, coding/technical challenges which will further validate user competencies directly on their profiles. Implementing a built-in calendar and scheduling tools which will streamline interview coordination and availability tracking. Additionally, a secure, role-based in-app chat system will facilitate direct communication between candidates, companies, and partners. To optimize development workflows, setting up CI/CD pipelines is also a logical step forward. We can use many latest pipeline tools like Github actions, Gitlab CI/CD helpers etc. Lastly, integrating gamification elements such as progress tracking, badges, and leaderboards can help drive user engagement and long-term retention.

REFERENCES

1. N. Kumar, P. Garg, and A. C. S. Pvt, "Impact of Online-Recruitment on Recruitment Performance," 2010, pp. 327–336.
2. Dr. Bhupendra Singh Hada "Opportunities & Challenges of E-Recruitment" 2015, pp. 1–4.
3. Ugo Chuks Okolie and Ikechukwu Emmanuel Irabor "E-Recruitment: Practices, Opportunities and Challenges" 2017, pp. 116-122.
4. Fanny Ramadhani*, Muhammad Zarlis "Analysis of e-Recruitment System Design" Volume 9, Number 1, March 2019 pp. 38 - 45.
5. Mohanish Bawane, Ishali Gawande, Vaishnavi Joshi, Rujuta Nikam, Prof. Sudesh A. Bachwani "A Review on Technologies used in MERN stack" Volume 10 Issue I Jan 2022 pp. 479-488.
6. Abdulrahman Aljuaid and Maysam Abbod "Artificial Intelligence-Based E-Recruitments System" 2020, pp. 144-147
7. https://www.mongodb.com/docs/manual/introduction/
8. https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Express_Nodejs/Introduction
9. https://react.dev/learn
10. https://trends.google.com/trends/explore?date=today%205-y&q=MERN%20Stack,MEAN%20Stack,PERN%20stack,MEVN%20stack,LAMP%20stack
11. https://dev.to/ashay_tiwari_3658168ad5db/tunneling-made-simple-exposing-local-react-and-node-apps-with-ngrok-and-localtunnel-5g1g
12. https://www.mongodb.com/resources/basics/full-text-search
13. https://tanstack.com/query/latest/docs/framework/react/overview