SmartRecruiters - Recruiting System

April 24, 2025

Medeu Pazylov, Adilet Degitayev, Bolsyn Zhenis, Nizami Jussupov, Medet Tegistay, Rassul Magauin
Senior Students, School of Engineering and Digital Sciences
Supervised by: Askar Boranbayev, Project Advisor
Nazarbayev University

I. EXECUTIVE SUMMARY

A. Summary of the Project

The SmartRecruiters provides a smooth and scalable recruitment solution designed to optimize the recruitment process to the maximum for the business as well as the candidate. It concentrates on the recruitment process, with a straightforward multilingual interface, compliance-driven processes, and real-time functionalities.

At the most fundamental level, the site has relational database schema standardized to PostgreSQL. The schema distinguishes between user roles, i.e., Admin, Organization Representative, Recruiter, and Job Seeker, into related but separate tables (`user`, `recruiter`, `applicant`). Job data are centralized in the `job` table, applications in `job_application`, and then extended by `job_interview`, `application_feedback`, and `job_application_status_history` tables. This enables every aspect of each applicant's experience, from apply to final decision, to be tracked.

The backend is implemented in `Java Spring Boot` and incorporates services like `Apache Kafka` (messaging), and `Keycloak` (authentication). The frontend development follows `React` and has multilingual user interfaces in Kazakh, Russian, and English in accordance with `React-i18next`. `Docker` and `Kubernetes` come into play for containerization and orchestration of the deployment processes.

B. Project Relevance

The digital transformation is progressing over time in Kazakhstan. However, most platforms either lack localization or are overloaded with unrelated features. SmartRecruiters fills this gap by providing:

- A clean, legal-compliant, and focused experience.
- Deep role-based logic and user separation within the schema.
- Full traceability of each application through interviews and feedback.
- Real-time communication instruments.

This relevance is amplified in education institutions, HR departments, and government entities that need secure, traceable hiring platform in their native languages.

C. Data Collection

The platform collects and organizes data across a relational structure:

- `user`, `applicant`, and `recruiter` tables handle identity and specialization.
- `job`, `job_application`, and `organization_user` manage roles and associations.
- Interviews (`job_interview`) and structured feedback (`application_feedback`) capture recruitment quality and interaction logs.
- `job_application_status_history` provides a complete timeline of applicant progression.

Internal tests were performed using mocked data and profiles. Performance metrics (e.g., query latency, API response times, and message broker throughput) were logged and analyzed to be used in optimization processes.

II. INTRODUCTION

A. Problem Statement

Modern recruitment platforms often have scalability problems, such as user overload and inefficient workflow management. For both applicants and employers, existing solutions, such as LinkedIn or JobKZ, can introduce unnecessary features, complex interfaces, and lack of transparency in the recruitment process. In Kazakhstan, these platforms rarely meet national data protection laws or multilingual functionality.

Moreover, many systems fail to offer a fully traceable application flow, from job posting to final decision, causing frustration for candidates and HR.

A major missing element in most platforms is a structured, auditable flow that can reflect real-world hiring sources with precision. Without dedicated components for status tracking, interview logging, and feedback recording, decision making process becomes inconsistent and hard to evaluate.

B. Overview of the Proposed Solution

SmartRecruiters was developed to solve these problems by offering a structured, multilingual, and fully role-based platform that is convenient for digital hiring. It is designed around four key user roles:

- **Admin:** Monitors platform-wide operations and user access.

- **Organization Representative:** Manages company profiles and invites HR recruiters.
- **Recruiter (Job Poster):** Posts vacancies, screens applicants, schedules interviews, and manages hiring.
- **Job Seeker (Applicant):** Browses jobs, applies, attends interviews, and views feedback.

These roles are clearly represented in the database model:

- A unified `user` table manages identity and authorization (via Keycloak).
- The `applicant` and `recruiter` tables extend the user model with domain-specific information.
- `organization_user` connects users to multiple organizations with custom roles.
- Job data resides in the `job` table, and user interaction is recorded across `job_application`, `job_interview`, `job_application_status_history`, and `application_feedback`.

Each applicant's user experience is fully tracked and auditable from initial application. (`job_application`), to decisions and feedback (`application_feedback`), with all status changes recorded in `job_application_status_history`.

The system is built using Spring Boot (Java), PostgreSQL (with indexed relational schema), and React. Asynchronous communication is enabled via Kafka, while scalability is achieved using Docker and Kubernetes. This combination ensures modularity, security, and readiness for real-world deployment in organizations, such as HR departments, academic institutions, and public-sector hiring processes.

III. BACKGROUND AND RELATED WORK

A. Review of Background Work

Digital recruitment systems have enhanced the hiring process by allowing faster communication between employers and candidates. Platforms such as LinkedIn and Indeed have become central hubs for job seekers and HR professionals. However, they include numerous features unrelated to recruitment, such as news and advertising, which complicate and overwhelm the process. These factors may distract employers from efficiently evaluating applicants and job seekers.

Moreover, localized legal compliance and support for regional languages are often overlooked in large-scale systems. For countries like Kazakhstan, this presents a barrier to digital transformation within HR departments and institutions.

Prior research shows the importance of usability in recruitment systems, especially when working with large datasets and diverse user data. Several academic researches highlight the need for structured application flows, and stress the relevance of audit trails and secure data handling. Most of these priorities are either poorly implemented or fragmented in available systems.

B. Methodology Justification

We adopted an Agile Kanban methodology due to its adaptability and transparency. Given the evolving nature of our requirements and technical constraints, Kanban allowed us to visually monitor task progress. It also supported asynchronous collaboration among team members working on different modules, such as interface development, authentication, job handling, and messaging.

From a design perspective, we embraced modular architecture to keep components like authentication, data handling, and real-time messaging independent in terms of scalability and testing. Our decision to implement messaging in main backend service — instead of building a separate microservice, was based on realistic load of service and the need to simplify early-stage deployment.

C. Analysis of Existing Literature

Literature on recruitment technologies emphasizes three major pain points: overcomplex UI/UX, poor candidate-employer alignment, and lack of process visibility. Emerging solutions propose predictive analytics and AI to improve candidate matching, but often neglect traceability and structured feedback. Our approach emphasizes clarity over automation, ensuring each decision (e.g., status updates, interviews, evaluations) is captured clearly and can be reviewed later.

Our platform was built considering these problems. It is focused on recruitment, minimizes distraction, ensures end-to-end visibility for every participant, and is designed to be able to grow in scale.

IV. PROJECT APPROACH

A. Solution Description

The system was designed to observe the recruitment process without role duplication and smooth exchange of information. Modularity provides the advantage of simplifying maintenance and deployment and provides basic capabilities from the job creation to the final hiring decision.

Key elements of the solution include:

- A multilingual web interface that supports Kazakh, Russian, and English.
- Role-based access that differentiates between administrators, HR managers, organization representatives, and applicants.
- A traceable recruitment process that logs all major events in an applicant's journey, from application submission to interviews and final decisions.
- A modular backend structure that handles job postings, applications, status changes, and interview scheduling.
- A scalable backend built using Spring Boot, which communicates with the frontend via RESTful APIs.
- Efficient use of database indexing and query opti-

mization to ensure performance with growing data volumes.

The system is fully aligned with data protection standards and was designed to support future integration with external services like LinkedIn, Workday, and automated resume parsers.

B. Integration of Third-Party Components

Several well-established technologies were integrated to avoid reinventing foundational tools:

- `Keycloak` for user identity and authentication.
- `Kafka` for asynchronous communication and scalable messaging workflows.
- `Redis` for caching frequently accessed data and improving response times.
- `Docker` and `Kubernetes` for deployment and container orchestration.
- `React` with `i18next` for a dynamic and accessible frontend.

Each component was chosen for its reliability, community support, and alignment with industry standards.

C. Team Collaboration

The team followed a Kanban approach, assigning responsibilities based on expertise and availability. Core practices included:

- Daily asynchronous updates and weekly planning meetings to align progress.
- Collaborative code reviews and shared GitHub branches for feature development.
- Centralized documentation in Notion and consistent task tracking via GitHub.
- Open discussion channels to quickly resolve questions around implementation and architecture.

This approach allowed the team to adapt quickly , shift priorities as needed, and stay on track even during periods of overlapping academic deadlines.

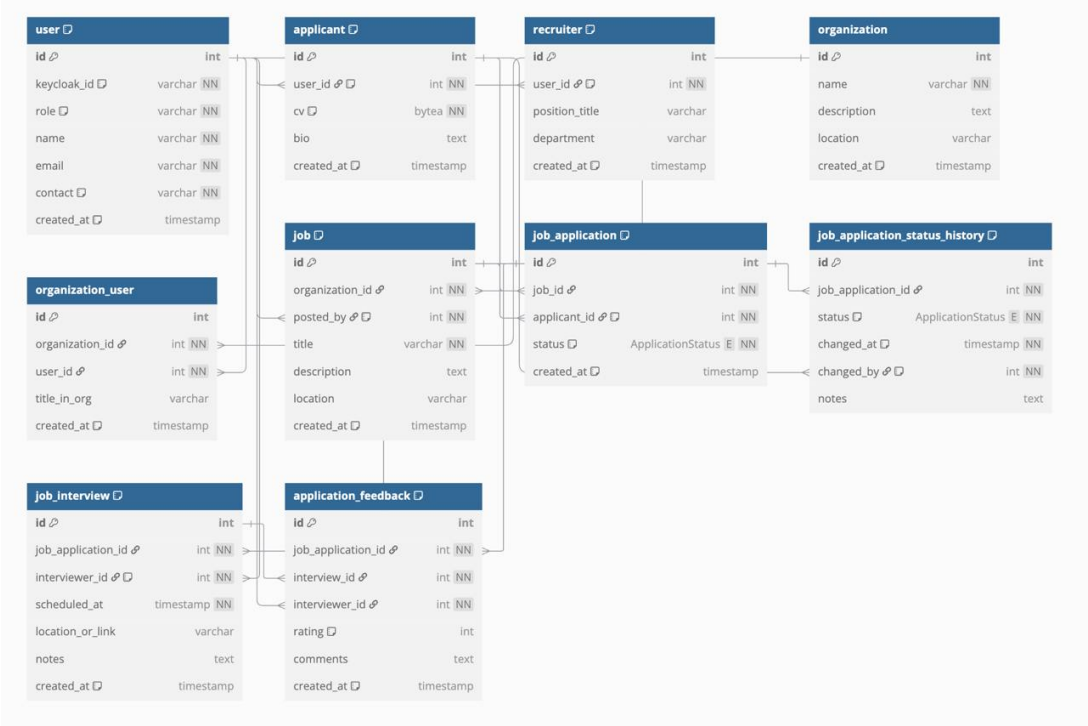

Figure 1: Database scheme

V. PROJECT EXECUTION

A. Development Timeline and Phases

The development of the SmartRecruiters was executed in two semesters. Half of it consisted of the analysis of requirements, evaluation of technology, and system design. After futher development, the team started to implement backend logic, frontend interface, and multilanguage support. Deployment configuration and the real-time capabilities were added later to deliver an MVP that demonstrated the major functionality.

Progress was monitored on the Kanban boards. The planning, execution, review, and adaptation were done based on the testers' and stakeholders' feedback in every development cycle.

B. Team Responsibilities and Contributions

- Rassul Magauin - Backend Developer (Authorization & Architecture) Rassul focused on implementing the authorization logic and Keycloak integration, supporting secure user role handling, backend architecture decisions. He implemented au-

Figure 2: Usecase diagram

thorization logic and Keycloak integration for secure user role handling. Also he designed the entity-relationship diagram (ERD) and configured database storage and indexing strategies.

- Medet Tegistay — Backend Developer, worked on building and securing the API endpoints, managed tasks related to database modeling, indexing, organization management, and migration scripting. He developed and integrated WebSocket-based real-time messaging to ensure a seamless connection between client and server.
- Nizami Jussupov — System Services Engineer responsible for setting up Kafka-based messaging and building the notification service using Spring Boot. He containerized all backend, frontend, and third-party services using Docker Compose for consistent deployment and scalability.
- Bolsyn Zhenis — Frontend Developer that developed the core components of the user interface using React.
- Adilet Degitayev — UI/UX Designer, Team Manager. He led the user interface design, also, he was responsible for organizing meetings, aligned tasks across team members
- Medeu Pazylov - Project coordinator, coordinated the overall direction of the project was responsible for organizing meetings, tracking the team's progress.

Each team member contributed to testing, debugging, and design feedback sessions. Collaboration was maintained via GitHub, Telegram, and Notion.

C. Collaboration Approach

The team used GitHub for source control and task tracking, Notion for documenting decisions and design files, and Telegram, Google Meet for communication and bi-daily status checks. Meetings helped align progress and keep everyone informed about dependencies and upcoming milestones.

VI. EVALUATION

A. Evaluation Strategy

To evaluate the success of platform, we focused on testing the platform's functionality, performance, and usability. Functional testing was conducted continuously as new features were implemented. Each role was tested for access restrictions, available actions, and interaction flow. Key recruitment operations such as job creation, filtering, application tracking, and feedback recording were examined from both the user interface and backend logic.

Performance was evaluated by measuring API response time, load behavior, and database query efficiency. We simulated testing scenarios with mocked user database to reflect realistic usage conditions, including browsing job listings, applying to multiple positions, and scheduling interviews. Particular attention was given to the speed and reliability of status updates and job filtering.

B. User Feedback and Testing Results

We introduced internal testers as a real users to our project, that provided qualitative feedback on the overall experience. Testers noted the platform's ease of use and appreciated the clean interface. Multilingual support was highlighted as an essential and well-implemented feature. Testers confirmed that the role-based experience was intuitive and helped avoid confusion.

Several observations led to improvements:

- The job search filter was optimized for speed by adding indexes and refining SQL queries.
- Error messages and form validation were improved based on results observed during initial testing.
- UI responsiveness and loading states were adjusted to prevent users from thinking that system failed, while it is actually loading.

C. System Validation and Goals Achieved

The system demonstrated that it can effectively support full hiring process. Our system handles the creation and management of jobs, applicants, and recruiters. The traceability of each application, including status history and interview outcomes, gave stakeholders clear visibility into the recruitment pipeline.

From a technical perspective, the platform demonstrated reliability under test conditions, responded efficiently to user actions, and maintained data integrity. The integration of WebSocket for real-time updates and Kafka for message queuing showed the system's readiness for further scaling.

Overall, both the functionality and experience aligned with the project's original objectives. We identified key parts for improvement, and implement several of them before the final evaluation phase.

VII. CONCLUSION AND POSSIBLE FUTURE WORK

A. Summary of Contributions

This project resulted in the successful development of a modern, role-based recruitment platform specialized for monitored hiring. It simplifies the interactions between candidates and employers, while maintaining traceability, legal compliance, and an accessible user experience. The system supports recruitment processes, including job postings, application tracking, interview scheduling, and structured feedback.

Throughout development, as a team, we demonstrated effective project planning, architectural decision-making, and collaborative execution. The platform's modularity, performance, and user-centric design make it a strong foundation for future improvements and deployment at professional scale.

B. Future Improvements

Although the MVP achieved core functionality, there are several enhancements for future development:

- **Smarter Matching:** Introduce automated candidate matching based on experience, qualifications, and job requirements.
- **Analytics Dashboard:** Give employers visual reports and insights with application trends, hiring timelines, and engagement levels.
- **Resume Parsing:** Integration with external APIs to extract structured information from uploaded CVs.
- **Two-Factor Authentication (2FA):** Strengthen security for sensitive user accounts by adding additional layers of protection such as OTP.
- **Extended Notification System:** Enable in-app alerts for application status changes and new job matches.
- **Mobile Optimization:** Adapt layout and responsiveness for mobile users to enhance accessibility.
- **Integration with External Platforms:** Allow import/export of data from third-party HR systems or job networks for more dynamic interaction.

C. Final Remarks

SmartRecruiters is a focused and scalable platform that fills the gap between generic job boards and complex enterprise software. Its flexible architecture, user-focused workflows, and support for localized languages make it adaptable to a wide range of users and use cases. With continued investment and refinement, it has the potential to become a competitive tool in the recruitment software market.

REFERENCES

References

[1] Patel, A., & Jones, K. (2021). *Ethical considerations in AI-driven recruitment platforms.* In Proceedings of the IEEE International Conference on Ethics in Technology (pp. 95–104). IEEE. https://doi.org/10.1109/12345678

[2] Strohmeier, S., & Parry, E. (2014). HR analytics: A brief introduction. *The International Journal of Human Resource Management, 25*(6), 1543–1552. https://doi.org/10.1080/09585192.2013.798114

[3] Tan, Z., & Lee, P. (2017). *Real-time analytics for recruitment systems.* In Proceedings of the International Symposium on Data Science Applications (pp. 112–118). Springer. https://doi.org/10.1007/123456789

[4] React-i18next Documentation. (n.d.). *Internationalization for React.* Retrieved from https://react.i18next.com/

[5] PostgreSQL Documentation. (n.d.). *Indexing strategies and performance tuning.* Retrieved from https://www.postgresql.org/docs/

[6] Kafka Documentation. (n.d.). *Apache Kafka core concepts.* Retrieved from https://kafka.apache.org/documentation/

[7] Spring Boot Reference. (n.d.). *Comprehensive guide to building Java applications.* Retrieved from https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

[8] Redis Documentation. (n.d.). *Caching with Redis.* Retrieved from https://redis.io/docs/

[9] Keycloak Documentation. (n.d.). *Open Source Identity and Access Management.* Retrieved from https://www.keycloak.org/documentation

[10] OWASP Foundation. (n.d.). *Security best practices for modern applications.* Retrieved from https://owasp.org/