IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

DOI: 10.17148/IARJSET.2026.13463

"INTERVIEW SCHEDULING TOOL"

Mrs.Alka Shrivastava[1], Mr. Tushar Awale[2], Mr. Sumit Deshmukha[3], Mr. Shantanu Bhambore[4], Ms. Punam Selwatkar[5], Ms. Priya Vaidya[6]

Assistant Professor, CT Department, Priyadarshini College of Engineering, Nagpur, Maharashtra, India, Asia, 440019[1]

Research Scholar, CT Department, Priyadarshini College of Engineering, Nagpur, Maharashtra, India, Asia, 440019[2-6]

Abstract: This presents the design and implementation of **Interview scheduling**, a monolithic web platform that unifies candidate onboarding, interview scheduling, live audiovisual collaboration, and structured evaluation. The system is implemented using **Spring Boot 3** on **Java 17**, with a **MySQL** database accessed through **Spring Data JPA**, and browser-based clients built from static HTML and JavaScript served by the same application. Candidates register, verify email, keep detailed profiles with document uploads, and reserve interview slots created by administrators. A **WebRTC** mesh topology carries peer media; signaling, chat, typing, and presence use **STOMP over WebSocket** (SockJS endpoint). Administrators control global meeting lifecycle through authenticated HTTP operations protected by an **opaque meeting administrator token**. Speech segments are transmitted to an external **Whisper-compatible HTTP** transcription service; transcripts feed **communication analytics** (e.g., speaking rate and filler-word statistics) combined with technical, behavioural, and profile-derived scores. **Weighted ranking** produces comparable outcomes across candidates. Optional **Judge0** integration evaluates coding submissions in a sandbox. Results can be published as **PDF reports** with embedded charts. The work explicitly documents security trade-offs: Spring Security is configured to allow all HTTP requests in the current codebase, and user identity for REST calls relies on client-supplied identifiers—acceptable for a controlled academic deployment but requiring hardening for production. The contribution is an end-to-end, implementation-grounded architecture description suitable for academic evaluation and further industrial extension.

I. INTRODUCTION

Modern hiring and campus placement processes require consistent scheduling, transparent communication, and fair comparison among a large number of candidates. Organizations and academic institutions often rely on fragmented tools such as online forms for data collection, separate video conferencing platforms for interviews, and spreadsheets for evaluation and ranking. While these tools individually serve specific purposes, their lack of integration introduces several operational challenges.

These challenges include:

- Increased administrative overhead due to manual coordination.
- Lack of traceability across various stages of the hiring process
- Inconsistent evaluation standards
- Difficulty in keeping structured candidate data.
- Limited analytical insights into candidate performance

As the scale of recruitment increases, these inefficiencies become more pronounced, leading to delays, miscommunication, and potential bias in decision-making. Therefore, there is a need for a unified system that integrates all stages of the recruitment lifecycle into a single, cohesive platform.

An integrated solution can significantly reduce friction by binding candidate identity management, scheduling, real-time interaction, and quantitative evaluation into one deployable application. Such a system not only improves efficiency but also enhances transparency, scalability, and fairness in the hiring process.

II. PROBLEM STATEMENT

The problem is to engineer a **single cohesive platform** that:

1. Let's **candidates** register safely, verify contact information, and keep structured profiles with evidence (certificates, documents).

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

DOI: 10.17148/IARJSET.2026.13463

2. Let's **administrators** define **interview slots**, see **bookings**, and run **live meetings** with clear modes (ad-hoc vs slot-bound).

3. Captures **behavioural signals** from speech (via transcription) and **technical signals** from coding tasks, then **aggregates** them with **configurable weights** into a defensible ranking and **reporting** artifact.

The challenge is not only UI but **internal architecture**: coordinating REST, WebSocket, file storage, external STT, optional remote code judging, and persistence without unmanageable coupling.

III. LITERATURE SURVEY

A. Interview Scheduling Systems

Interview scheduling has evolved from manual coordination to automated systems designed to reduce administrative effort and improve efficiency. Early approaches focused on optimization techniques to generate conflict-free schedules, particularly for large-scale interview events. While these methods ensure best allocation of time slots, they often lack flexibility and are difficult to adapt to dynamic real-world scenarios.

Modern online scheduling systems enable candidate self-scheduling, automated slot allocation, and notification mechanisms. Studies by Ethan Cohn and Eric D. Heggestad demonstrate that such systems significantly reduce manual coordination and improve user experience. However, these platforms are typically limited to scheduling and do not integrate interview execution or evaluation features.

B. Practical Scheduling Approaches

In real-world recruitment environments, scheduling must be efficient and adaptable. Practical systems often rely on heuristic-based approaches that prioritize speed and flexibility over strict optimality. These methods allow systems to oversee last-minute changes, varying availability, and real-time updates effectively.

Research by Harminder Kaur and Puneet Gupta highlights the use of automation in interview scheduling, combining application processing with scheduling workflows. While these approaches improve efficiency, they primarily focus on backend automation and lack integration with real-time communication and evaluation components.

C. Integration in Recruitment Systems

Modern recruitment systems emphasize integration with existing tools such as calendars, communication platforms, and applicant tracking systems. Integrated scheduling solutions reduce coordination effort, minimize delays, and streamline the recruitment pipeline.

Systems like Calendar. Help prove how workflow-based automation can improve scheduling efficiency by combining machine help with human input. However, such systems run as standalone scheduling tools and do not provide a unified platform that includes interview execution and candidate evaluation.

D. AI-Based Interview and Evaluation Systems

Recent advancements in recruitment technology include AI-based interview systems that support remote hiring and automated candidate assessment. Research by Byung Chul Lee and Byung Yong Kim presents systems capable of evaluating candidates using structured metrics and automated analysis.

While these systems enhance scalability and consistency, they often lack transparency and do not integrate seamlessly with scheduling and communication workflows. Additionally, many AI-based systems focus heavily on evaluation without addressing the complete recruitment lifecycle.

E. Real-Time Communication Technologies

Real-time communication is a critical part of modern interview platforms. Technologies such as WebRTC enable peer-to-peer audio and video communication directly within web browsers, cutting the need for external conferencing tools. Similarly, WebSocket supports low-latency, bidirectional communication needed for signaling, messaging, and real-time updates.

Despite their widespread use in communication platforms, these technologies are rarely integrated with scheduling and evaluation systems in a unified architecture.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

DOI: 10.17148/IARJSET.2026.13463

F. Research Gap and Motivation

Despite significant advancements, existing systems show several limitations:

- Lack of integration between scheduling, communication, and evaluation
- Limited support for real-time analytics during interviews
- Dependence on multiple disconnected tools
- Absence of unified platforms combining technical and behavioural assessment

Address these challenges, the proposed system introduces a **comprehensive interview scheduling and evaluation platform** that integrates:

- Automated scheduling and slot management
- Real-time communication using WebRTC and WebSocket
- Speech-to-text–based communication analysis
- Technical evaluation through coding assessment tools

Study	Focus	Key Features	Technologies Used	Findings / Contributions
Ethan Cohn & Eric D. Heggestad (2020)	Online Interview Scheduling Systems	Candidate self-scheduling, slot allocation, automated notifications	Web applications, database systems	Reduced manual coordination and improved user experience, but limited to scheduling only
Harminder Kaur & Puneet Gupta (2022)	Automated Scheduling with ML	Application processing, scheduling automation, workflow management	Machine learning models, backend systems	Improved efficiency and reduced administrative workload, but lacks real-time communication integration
Calendar. Help (2017)	Workflow-based Scheduling Automation	Human-in-loop scheduling, adaptive coordination, automated workflows	Multi-agent systems, AI negotiation frameworks	AI-assisted systems, workflow engines
Byung Chul Lee & Byung Yong Kim (2021)	AI-Based Interview Systems	Remote interview support, automated evaluation, structured scoring	AI models, web-based systems	Improved scalability and evaluation consistency, but lacks integration with scheduling and communication
Modern Recruitment Platforms	Integration in Recruitment Systems	Calendar sync, automated reminders, pipeline management	APIs, cloud-based systems	Improved recruitment efficiency but lacks unified evaluation and communication features
WebRTC	Real-Time Communication	Peer-to-peer audio/video communication	Browser-based RTC protocols	Enables live interviews without external tools but not integrated with evaluation systems

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

DOI: 10.17148/IARJSET.2026.13463

Study	Focus	Key Features	Technologies Used	Findings / Contributions
Proposed System (This Work)	Integrated Interview Platform	Scheduling, real-time communication, STT analysis, technical evaluation	Spring Boot, WebRTC, WebSocket, STT APIs, Judge0	Provides a unified platform integrating scheduling, communication, and evaluation, improving efficiency, transparency, and scalability
General Research (Heuristic Scheduling)	Practical Scheduling Approaches	Priority-based scheduling, dynamic slot adjustment, real-time updates	Algorithmic heuristics, backend logic	Provides fast and adaptable scheduling but sacrifices optimality

IV. OBJECTIVE

and implement an integrated, end-to-end recruitment and interview management platform that streamlines scheduling, enables real-time interaction, and supports structured, data-driven evaluation of candidates.
Achieve this goal, the specific aims are as follows:

1. System Design and Backend Implementation

- Develop a robust backend system using Spring Boot that exposes RESTful APIs for managing users, candidate profiles, interview scheduling, analytics, meeting resources, and interview lifecycle workflows.
- Maintain structured and consistent data storage using MySQL with Spring Data JPA for efficient data access and persistence.

2. Streamlining Scheduling and Recruitment Workflow

- Automate and simplify the interview scheduling process, reducing manual coordination between candidates, recruiters, and interviewers.
- Minimize scheduling conflicts, double-bookings, and missed interviews through structured slot management.
- Improve recruitment efficiency by accelerating the interview process and reducing administrative overhead.
- Ensure a fair and transparent system where all candidates have equal opportunity to take part in interviews.

3. Enhancing Candidate and Recruiter Experience

- Provide a user-friendly platform that allows candidates to register, manage profiles, upload documents, and select convenient interview slots.
- Improve overall candidate experience through flexibility, clarity, and reduced communication friction.
- Enable seamless coordination for recruiters and administrators within a unified system.

4. Real-Time Communication and Collaboration

- Implement live interview capabilities using WebRTC for audio and video communication.
- Enable signaling, messaging, and real-time interaction using STOMP over WebSocket with SockJS.
- Support collaborative interview environments including chat, presence tracking, and session control.

5. AI-Based Communication Analysis

- Integrate a configurable HTTP-based speech-to-text (STT) adapter for converting spoken communication into text.
- To analyses communication patterns such as speaking rate and filler-word usage.

- Incorporate these analytics into a structured scoring system for evaluating candidate performance.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

DOI: 10.17148/IARJSET.2026.13463

6. Technical and Coding Evaluation

- Support live coding assessments through integration with Judge0 when configured.
- Enable execution, result tracking, and status introspection of coding submissions within the platform.

7. Analytics, Reporting, and Decision Support

- Generate structured evaluation metrics combining technical, behavioural, and communication-based scores.
- Provide insights such as interview completion rates, candidate performance trends, and system usage statistics.
- Generate final evaluation reports in PDF format with embedded charts using server-side processing.

8. Integration and Extensibility

- Design the system in a way that allows integration with external tools such as calendar services (e.g., Google Calendar, Outlook), HRMS platforms, and applicant tracking systems.
- Support extensibility through modular design and external service integration.

9. Security Awareness and Future Improvements

- Document existing security limitations, including relaxed access control and client-dependent identity handling in the current implementation.
- Propose future enhancements such as token-based authentication, role-based authorization, and secure API design for production readiness.

V. PROJECT METHODOLOGY

1. System Design and Backend Implementation

- Design and implement a modular backend system using Spring Boot that exposes RESTful APIs for managing users, interviews, profiles, analytics, and meeting workflows.
- Follow a layered architecture consisting of controller, service, and repository layers to ensure separation of concerns.
- Use Spring Data JPA for efficient database interaction and abstraction of persistence logic.
- Maintain structured and consistent data storage using MySQL for reliability and scalability.

---

2. Layered Architecture and Maintainability

• Organize the application into distinct layers:

- Controller layer for handling client requests
- Service layer for business logic.
- Repository layer for database operations

• To improve maintainability, scalability, and testability through clear separation of responsibilities.
• To ensure reusability and centralized management of application logic.

---

3. Schema-First Database Design

- Adopt a schema-first approach by defining SQL scripts in the application resources for database initialization.
- Configure automatic schema setup using: spring. SQL. init. mode=always
- Ensure consistent database configuration across development and testing environments.
- Enable better control over schema evolution and versioning.

---

4. Incremental Feature Integration

Develop the system in multiple phases for better stability and reduced complexity:
Phase 1: Core REST APIs

- Implement user management, profile handling, and interview scheduling.

Phase 2: Real-Time Communication

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

DOI: 10.17148/IARJSET.2026.13463

- Integrate WebSocket communication using STOMP and SockJS
- Enable audio/video interaction using WebRTC.

Phase 3: AI and External Integration

- Integrate speech-to-text (STT) services for communication analysis.
- Use Judge0 for coding evaluation (optional)

Ensure each module is evaluated before adding new functionality.

---

5. Validation and Error Handling

- Implement input validation using Jakarta Bean Validation on DTOs.
- Enforce constraints such as required fields, formats, and value ranges.
- Implement centralized exception handling for consistent error processing.
- Return standardized JSON error responses for better debugging and client-side handling.

---

6. Streamlining Scheduling and Recruitment Workflow

- Automate interview scheduling and reduce manual coordination between candidates and recruiters.
- Prevent scheduling conflicts and double bookings using structured slot management.
- Improve recruitment efficiency and reduce administrative overhead.
- Ensure fairness and transparency in interview allocation.

---

7. Candidate Lifecycle Management

- Enable candidate registration and email verification for authentication.
- Provide profile management features including document uploads.
- Allow candidates to select and book interview slots based on availability.

---

8. Real-Time Interview Execution

- Enable live interview sessions started by administrators.
- Support communication using:
- WebRTC for audio/video streaming
- WebSocket (STOMP) for signaling and messaging.
- Provide real-time interaction including chat, presence tracking, and collaboration.

---

9. Communication Analysis and Evaluation

- Process audio streams and convert them into text using STT services.
- Generate real-time transcriptions for analysis.
- Compute communication metrics such as speaking rate and filler-word frequency.
- Use these metrics for evaluating candidate communication skills.

---

10. Technical Assessment and Coding Evaluation

- Allow candidates to take part in coding assessments during interviews.
- Evaluate code submissions using Judge0.
- Capture execution results and performance metrics for technical evaluation.

---

11. Final Evaluation and Reporting

• Aggregate scores from multiple evaluation dimensions:

- Technical performance
- Communication analysis
- Behavioural assessment

• To implement a weighted ranking algorithm for candidate evaluation.
• To generate structured reports and export results as PDF documents with charts.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

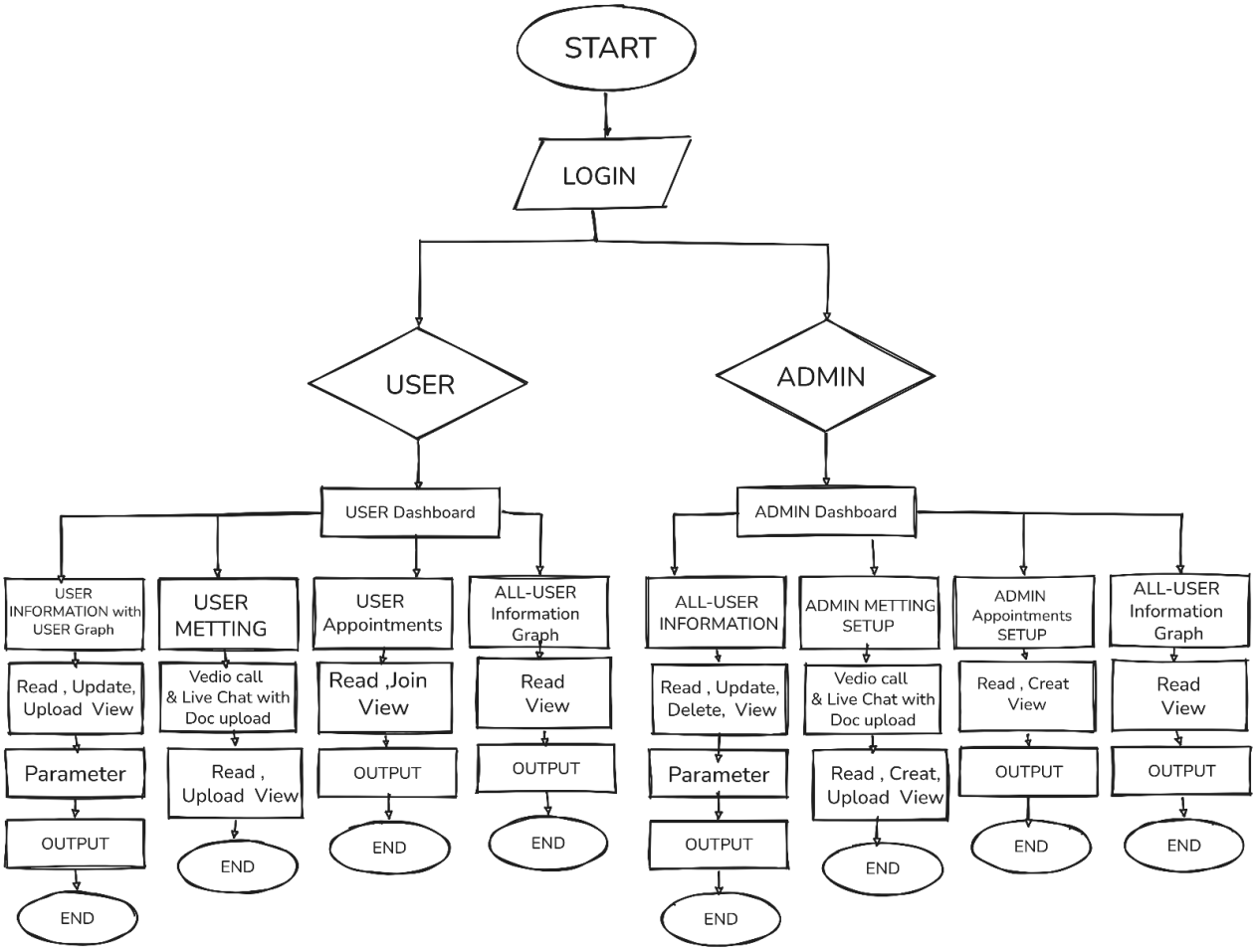
DOI: 10.17148/IARJSET.2026.13463

12. End-to-End Workflow Summary

The complete system workflow includes.

:

Register → Verify Email → Login → Profile Management → Interview Scheduling → Admin Initiates Meeting → WebRTC + WebSocket Communication → Audio Processing via STT → Communication Analysis → Coding Evaluation (Optional) → Score Aggregation → Ranking Generation → PDF Report Creation



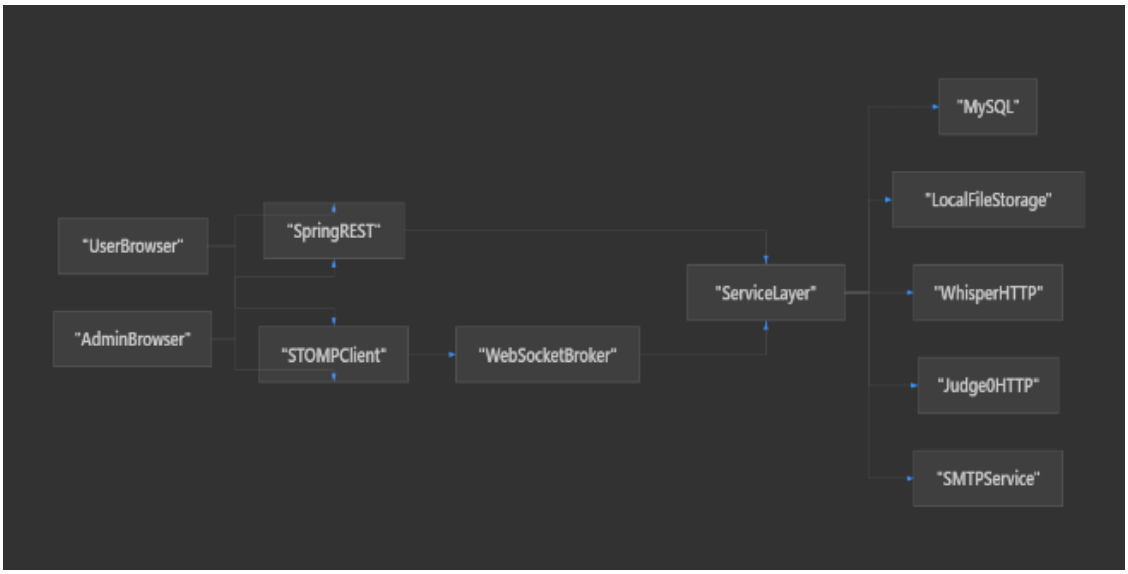
IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

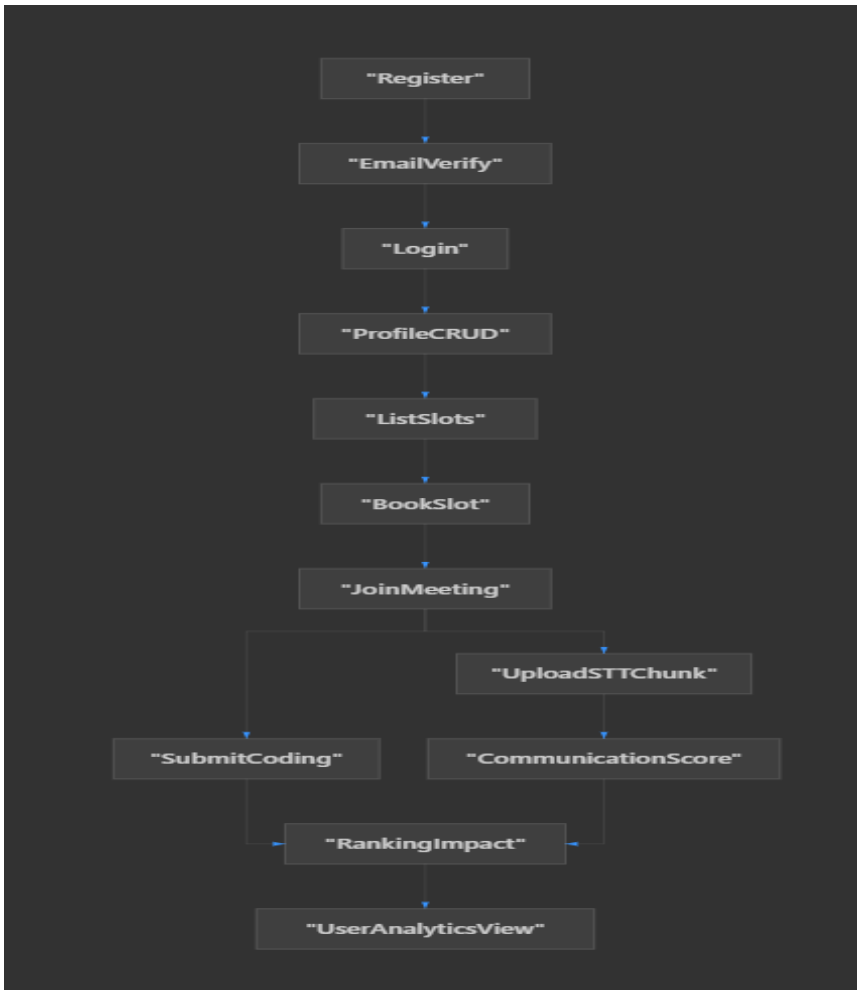
International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

DOI: 10.17148/IARJSET.2026.13463

V. USE CASE DIAGRAM





IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

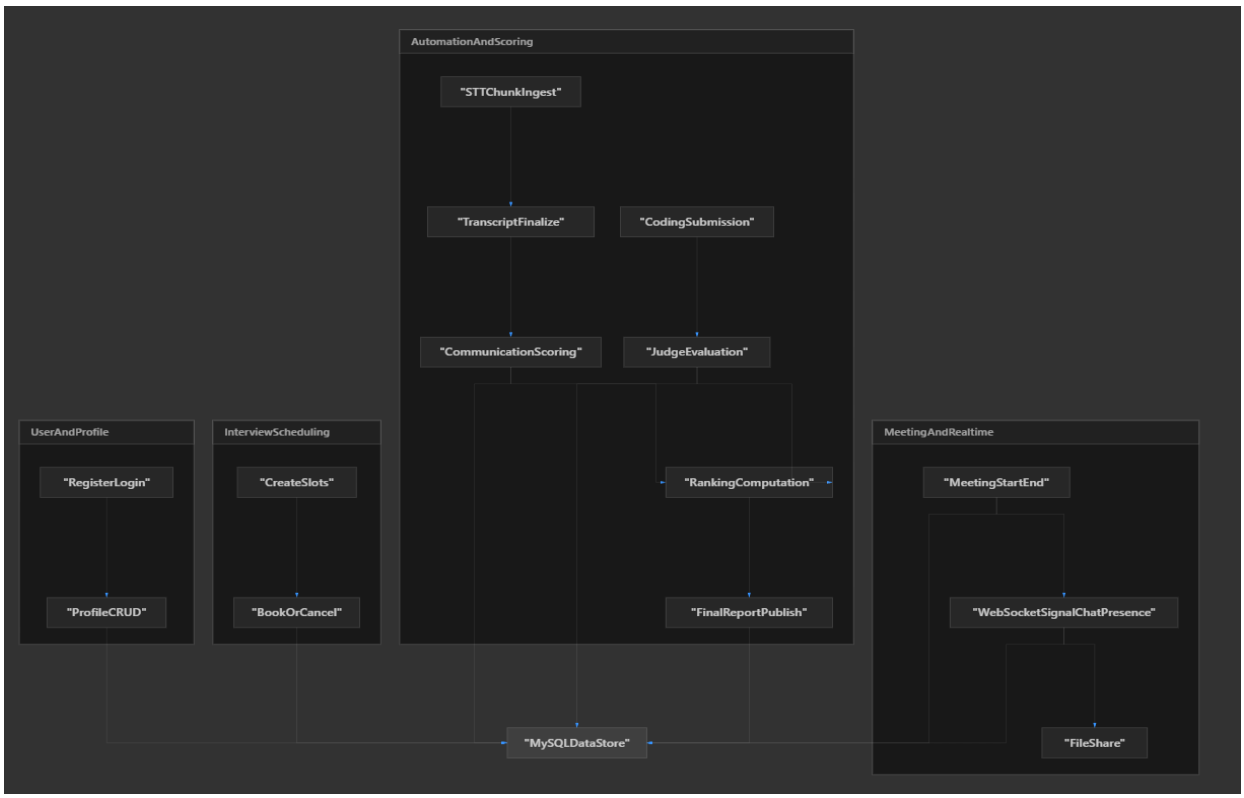
DOI: 10.17148/IARJSET.2026.13463

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

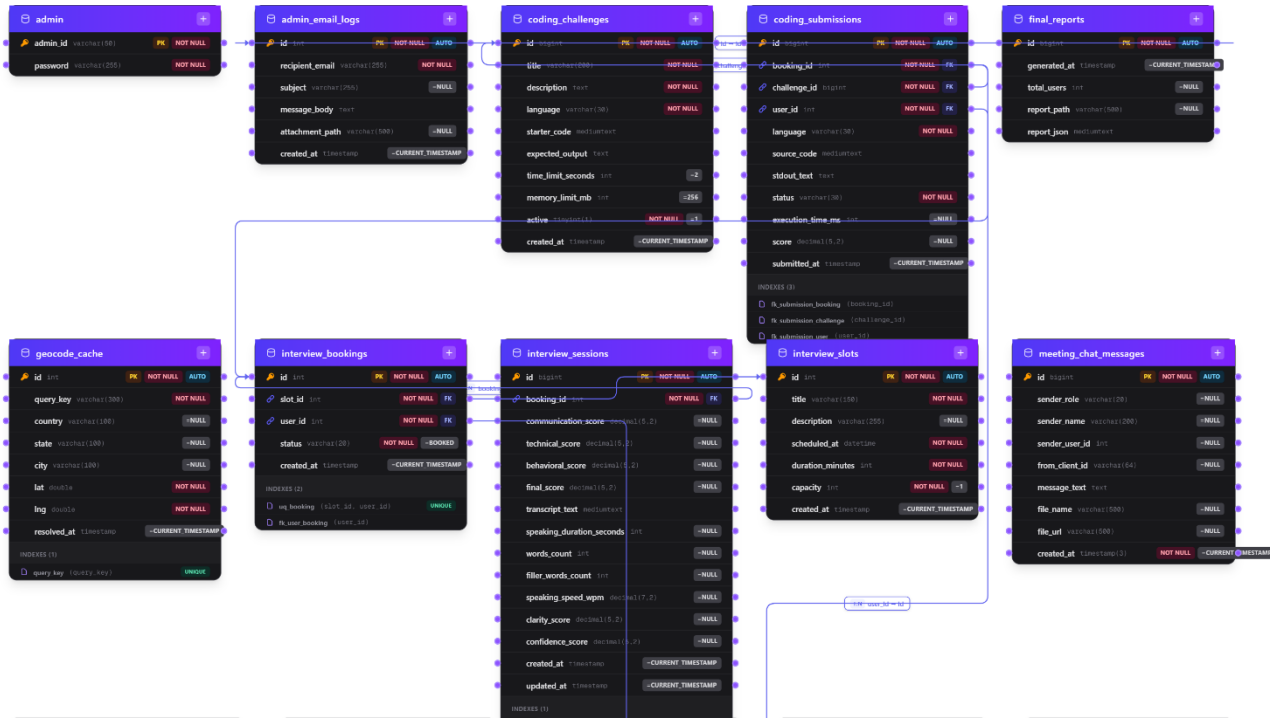
International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

DOI: 10.17148/IARJSET.2026.13463



ER-DIAGRAM

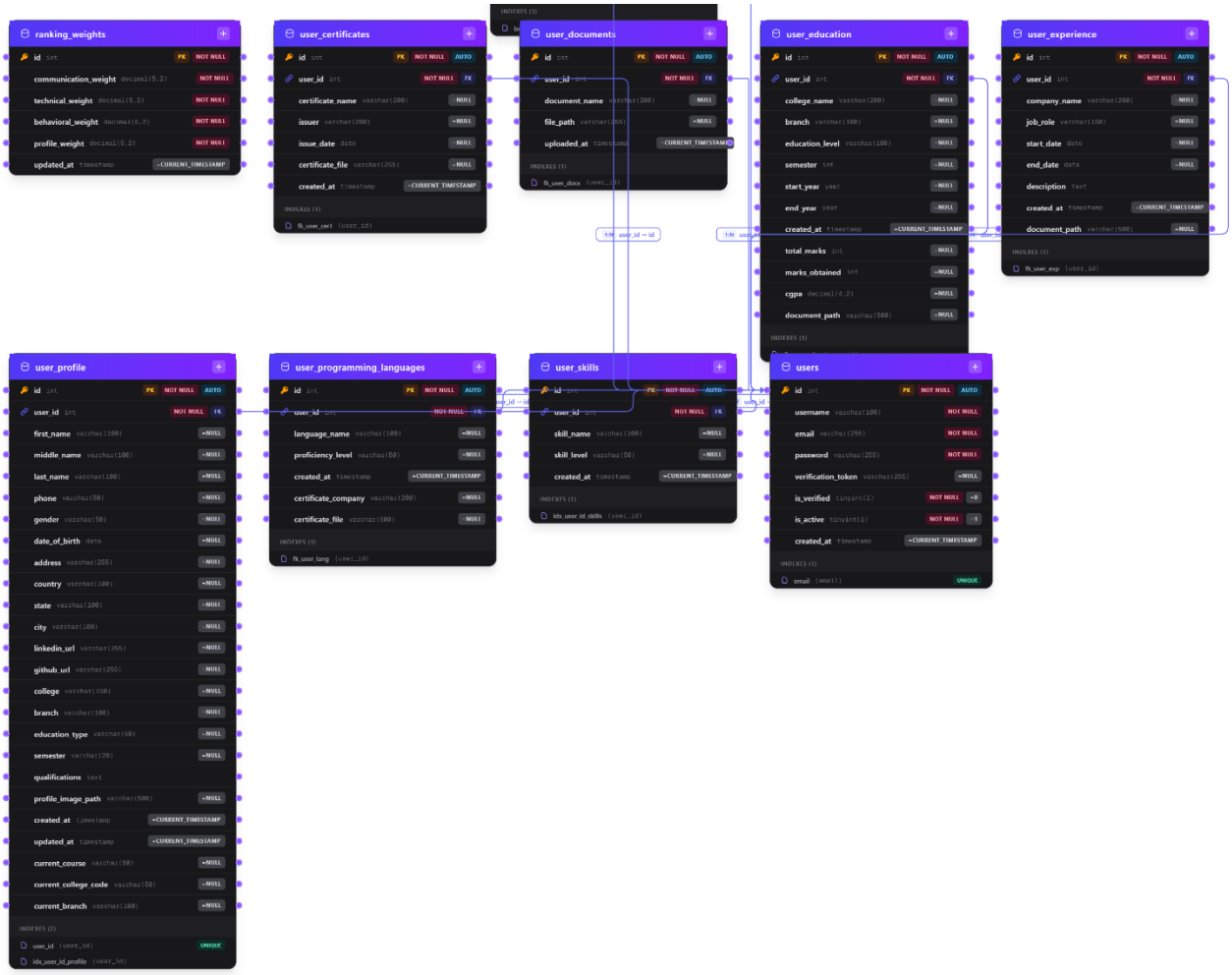


IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

DOI: 10.17148/IARJSET.2026.13463



VI. TESTING AND VALIDATION

ID	Scenario	Expected Result
T1	Register with duplicate email	409 Conflict or business validation error
T2	Login before email verification	401 Unauthorized
T3	Book slot twice (if restricted)	Error based on scheduling rules
T4	Start meeting without bookings	HTTP 400 with error message
T5	Upload STT data with invalid booking ID	404 Not Found

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

DOI: 10.17148/IARJSET.2026.13463

1. Testing Strategy

- The system is tested using a combination of unit testing, integration testing, and system-level validation to ensure reliability and correctness.
- Both automated and manual testing approaches are considered based on system complexity and environment dependencies.

2. Unit Testing

- The project supports unit testing through the Maven test lifecycle (mvn test).
- Core business logic components in the service layer are the primary focus for unit testing.
- Where automated test cases are limited, manual verification of service logic is performed.
- Recommended approach includes using JUnit 5 to confirm service-layer operations such as user registration, authentication, and scheduling logic.

3. Integration Testing

Integration testing ensures proper interaction between different modules and external interfaces.

- API Testing:

Endpoints are tested using tools like Postman or curl:

- /api/users/register
- /Login
- /api/interviews/slots
- WebSocket Testing**:**

STOMP-based clients are used to connect to /ws endpoint.
Real-time communication is verified through /topic message broadcasting.

- Security Testing:

Meeting authorization is confirmed:

- Without token → HTTP 403 (Forbidden)
- With valid token → HTTP 200 (Success)

4. System Testing

- End-to-end workflows are confirmed to ensure complete system functionality.

5. Manual Test Scenarios

• The following real-world workflows are tested manually:
Candidate Flow:
• Register → Verify Email → Login
• Update profile → Upload documents
Access analytics dashboard
Admin Flow:
Login → Create interview slot
Candidate books slot
Admin starts meeting
Multiple users join session
Communication Testing:
• Speech input is processed using STT service
• Transcription is generated in real time
• Communication metrics are evaluated

6. Results and Analysis

• The system successfully supports core functionalities such as registration, scheduling, and interview execution without critical failures when properly configured.
Performance:

DOI: 10.17148/IARJSET.2026.13463

• WebRTC-based communication follows a mesh architecture (N^2 connections), making it suitable for small-scale interview sessions.
• Scalability limitations are acknowledged and documented.
Evaluation Quality:
• Candidate scoring depends on:
- Accuracy of speech-to-text transcription
- Technical assessment inputs

• Weighted scoring allows flexible evaluation policies.

VII. CONCLUSION

This work presents the design and implementation of an integrated interview scheduling and evaluation platform that addresses the limitations of fragmented recruitment systems. By joining candidate onboarding, scheduling, real-time communication, and structured evaluation into a single application built on Spring Boot, the system proves how a unified architecture can significantly improve efficiency, transparency, and consistency in the hiring process.
The platform successfully enables end-to-end workflow management, including candidate registration, profile handling, interview slot booking, and live interview execution using WebRTC and WebSocket-based communication. The integration of communication analytics through speech-to-text processing and optional technical evaluation using Judge0 provides a multi-dimensional assessment framework. The use of a weighted scoring mechanism further ensures fair and comparable evaluation of candidates across different parameters.
From an implementation perspective, the layered architecture and use of Spring Data JPA with MySQL contribute to maintainability and scalability within a monolithic design. The system also highlights practical trade-offs, particularly in security configuration, where simplified access control is suitable for academic environments but requires enhancement for production deployment.
Despite its effectiveness, the system has certain limitations, including scalability constraints of WebRTC mesh topology and dependency on external services for transcription and evaluation. These limitations open opportunities for future enhancements.
Future work can focus on improving system scalability using media servers (e.g., SFU architecture), strengthening security through token-based authentication and role-based access control, and incorporating advanced analytics using machine learning techniques. Additionally, deployment in cloud environments and integration with enterprise tools can further extend the applicability of the system.
In conclusion, the proposed platform provides a practical and extensible solution for modern recruitment challenges, offering a foundation for both academic exploration and real-world implementation.

REFERENCES

[1]. *Residency applicant preferences of online systems for scheduling interviews.* **Western Journal of Emergency Medicine,** https://doi.org/10.5811/westjem.2016.4.29958

[2]. Cohn, E., & Heggestad, E. D. (2020). *An experimental evaluation of an online interview scheduler.* **Field Methods,** https://doi.org/10.1177/1525822X20921825

[3]. Bjørndal, M., & Johnston, K. (2022). *Interview scheduling: An integer programming approach.* **SSRN Electronic Journal.** https://doi.org/10.2139/ssrn.4028367

[4]. Kaur, H., & Gupta, P. (2022). *Automated application processing and interview scheduling using machine learning.* **arXiv preprint arXiv:2203.00456.** https://arxiv.org/abs/2203.00456

[5]. J. Cranshaw, E. Elwany, T. Newman, R. Kocielnik, B. Yu, S. Soni, and A. Monroy-Hernández, "Calendar. Help: Designing a Workflow-Based Scheduling Agent with Humans in the Loop," *arXiv preprint*,Mar.2017. Available: https://arxiv.org/abs/1703.08428

DOI: 10.17148/IARJSET.2026.13463

[6]. "An automated meeting scheduling system that utilizes user …" in *Proceedings of IAT '05: IEEE/WIC/ACM International Conference on Intelligent Agent Technology*. [DOI / Publication link via ACM / IEEE digital library] ACM Digital Library

[7]. B. C. Lee and B. Y. Kim, "Development of an AI-Based Interview System for Remote Hiring," *International Journal of Advanced Research in Engineering and Technology (IJARET)*, Vol. 12, Issue 3,2021. Available:https://www.researchgate.net/publication/357753685_Development_of_an_AI-Based_Interview_System_for_Remote_Hiring

[8]. Spring Team. *Spring Boot Reference Documentation.* https://spring.iprojects/spring-boot
[9]. Spring Team. *Spring Framework — Web on Servlet Stack (WebSocket/STOMP).* https://docs.spring.io/spring-framework/reference/web/websocket.html
[10]. WebRTC.org. *WebRTC API.* https://webrtc.org/
[11]. Oracle Corporation. *MySQL Documentation.* https://dev.mysql.com/doc/
[12]. Judge0. *CE & Extra CE API.* https://judge0.com/
[13]. OpenAI. *Whisper (speech recognition model).* https://github.com/openai/whisper