STM JOURNALS

E-Commerce for Future & Trends

https://journals.stmjournals.com/ecft/

ISSN: 2454-9347
Volume 13, Issue 1, 2026
January-April
DOI (Journal): 10.37591/ECFT

Research ECFT

HirePrep: A Microservice-Based Integrated Placement Preparation Platform with AI Assistance

Sumit Madaan[1,*], Rakshita Bhansali[1], Nagendra Singh Sisodiya[1], Lakshit Meena[1], Sonam Lowry[2]

Abstract

Preparing for campus placements can be a confusing and time-consuming process. Students have to use different platforms for things like practice tests, study materials, talking to people, and getting updates from the administration. This is not a waste of time, but it also makes it harder for students to be productive and clear about what they need to do when they are getting ready for their careers. To make things better, we made something called HirePrep. It is a platform that brings together all the things students need to do to get ready for campus placements. HirePrep has important parts, including tests, managing resources, tracking progress, sending notifications, and controls for the administration. We used React to make the front end of the platform look nice and work well. Django and Spring Boot are used for the end. We use PostgreSQL to keep all the data safe and sound. The way HirePrep is built makes it easy to add things and scale up. Each part of the platform works independently. The other parts are communicating with each other using special codes. This makes the platform more stable and able to handle a lot of users. We also added a chatbot that uses intelligence to give students help when they need it. The chatbot is really good at understanding what students are asking. Gives the right answers most of the time. HirePrep shows that using small services can work well in schools and universities. It also gives an example of how to build a platform that can be used by a lot of people.

Keywords: Conversational AI, educational technology platform, microservice architecture, placement management system, student information system

INTRODUCTION

Problem Context

Undergraduate students preparing for campus recruitment encounter operational inefficiencies from platform fragmentation. A requirements survey of 250 engineering students at (JECRC) Jaipur Engineering College & Research Centre University revealed that 78% utilize four to six separate applications for placement activities, including assessment platforms, messaging applications, and institutional portals. This fragmentation creates information redundancy, notification delays, and reduced engagement with preparation resources.

***Author for Correspondence**
Sumit Madaan
E-mail: sumitm.clg@gmail.com

[1]Student, Department of Computer Science and Engineering, JECRC University, Jaipur, Rajasthan, India
[2]Assistant Professor, Department of Computer Science and Engineering, JECRC University, Jaipur, Rajasthan, India

Received Date: February 11, 2026
Accepted Date: February 24, 2026
Published Date: March 31, 2026

Citation: Sumit Madaan, Rakshita Bhansali, Nagendra Singh Sisodiya, Lakshit Meena, Sonam Lowry. HirePrep: A Microservice-Based Integrated Placement Preparation Platform with AI Assistance. E-Commerce for Future & Trends. 2026; 13(1): 25–37p.

Traditional learning management systems (LMS) and enterprise resource planning (ERP) solutions address general academic workflows but lack specialized placement preparation support. Existing systems fail to integrate company-specific materials, alumni interview experiences, and real-time drive management within unified interfaces [1, 2].

Research Gap

Current educational platforms predominantly adopt monolithic architectures, thereby limiting independent component scaling [3]. Recent advances in microservice architecture for enterprise

applications [4, 5] have not been extensively applied to placement management systems. Furthermore, conversational AI integration for placement-specific guidance remains underexplored in the academic literature [6].

Contributions

This paper makes the following contributions:

1. A microservice-based reference architecture for placement preparation with nine independently deployable services
2. Polyglot implementation demonstrating Django and Spring Boot integration through API gateway patterns
3. Intent-based conversational assistance achieving 87% classification accuracy

Paper Organization

The paper proceeds as follows: it first reviews related work on educational platforms and microservice architecture, then details the system architecture and design decisions. This is followed by a description of the module implementations and an explanation of the deployment methodology. Next, the performance results are analyzed, and the security implementation is discussed. The paper then examines the limitations, outlines future work, and finally concludes.

RELATED WORK

Student Information Systems Evolution

Early student information systems adopted centralized architectures with tightly coupled application layers [7]. Das et al. [8] surveyed 50 Indian universities and found that 82% relied on monolithic ERP systems for academic management, causing inflexibility during updates and scalability constraints.

Rıad [9] proposed a service-oriented architecture (SOA) for university management and demonstrated improved maintainability through loosely coupled services. However, their implementation retained shared database layers, preventing true service independence.

Microservice Architectures in Education

Richardson [10] established foundational microservice patterns, including database-per-service, API gateway, and circuit breaker designs. Newman [11] extended these with practical implementation strategies for distributed systems.

Microservice applications in educational technology remain limited. Chen et al. [12] developed a microservice-based learning platform, demonstrating a 40% reduction in deployment time compared to monolithic alternatives. Lyu [13] implemented a microservice architecture for examination management, achieving independent scaling of question banks and evaluation services.

None of these systems address specialized placement preparation requirements, such as company-specific content curation, interview experience sharing, and eligibility validation.

Conversational AI in Educational Systems

Winkler and Söllner [14] conducted a systematic literature review of educational chatbots and identified information retrieval and procedural guidance as primary use cases. Pérez et al. [15] developed a rule-based chatbot for university frequently asked questions (FAQ) systems and achieved 73% query resolution accuracy.

Recent natural language processing advances enabled sophisticated educational assistants. Okonkwo and Ade-Ibijola [16] implemented a transformer-based chatbot for programming education, demonstrating improved contextual understanding compared to rule-based approaches.

Table 1. Comparison with existing platforms.

Feature	Traditional LMS	ERP systems	HirePrep
Architecture	Monolithic	Layered	
Placement modules	No	Limited	Comprehensive
AI assistant	No	No	Yes
Independent scaling	No	Partial	Yes
Alumni reviews	No	No	Yes

Comparative Analysis of Platforms

Table 1 shows that most LMS and ERP systems do not support top-ranked AI features, a complete placement module, or independent scalability. HirePrep, on the other hand, offers a new architectural model, provides an extensive range of placement services, includes AI capabilities, supports independent scalability, and allows feedback from alumni, thus making it a more comprehensive solution compared to traditional systems.

Research Positioning

HirePrep builds upon established microservice patterns [10, 11] while addressing gaps in placement-specific educational technology. Unlike general LMS [17] or ERP systems [18], HirePrep provides specialized modules for company-driven management, interview preparation, and peer experience sharing.

SYSTEM ARCHITECTURE

Architectural Overview

HirePrep implements a distributed microservice architecture following the principles outlined by Fowler [19] and Richardson [10]. The system comprises three logical tiers: presentation, application, and data.

Presentation Layer

A React-based single-page application provides role-differentiated interfaces for students, faculty, and administrators. The frontend communicates exclusively with the API gateway, eliminating direct service dependencies. Redux manages application state while Axios handles HTTP communications with JSON web token (JWT) token injection.

Application Layer

Nine independent microservices implement core functionality:

1. Tests, service (Django): assessment creation, scheduling, execution, and grading
2. Attendance, service (Spring Boot): tracking, analytics, and threshold alerts
3. Resources Service (Django): Content management, categorization
4. Notice Service (Django): Announcement distribution, scheduling
5. Reviews Service (Django): Company review aggregation, moderation
6. Placement Service (Spring Boot): Drive management, registration, and eligibility
7. Profile Service (Spring Boot): Student data centralization
8. Dashboard Service (Spring Boot): Data aggregation, summary views
9. AI Chatbot Service (Python/Flask): Intent recognition, context retrieval

Data Layer

PostgreSQL serves as the primary relational database with separate logical databases per service, ensuring loose coupling [10].

Each service owns its scheme, preventing cross-service table dependencies.

Microservice Design Principles

The architecture adheres to design principles derived from Newman [11]:

1. *Single Responsibility:* Each microservice encapsulates a bounded context [20]. The Tests service manages all assessment-related functionality without extending into attendance or resource domains.
2. *Database-per-Service:* Following Richardson's pattern [10], each service maintains an independent database schema. This prevents tight coupling through shared tables and enables independent schema evolution.
3. *Stateless Service Design:* Services maintain no session state between requests. All required context is conveyed through JWT tokens, enabling horizontal scaling without session affinity constraints [21, 22].
4. *Fault Isolation:* Circuit breaker patterns prevent cascading failures [23]. If the Reviews service experiences downtime, other services remain operational.

Technology Stack

Django was selected for data-intensive modules (tests, resources, notices) due to its mature object-relational mapping (ORM) and administrative interfaces [24]. Spring Boot handles transactional services (attendance, placement, profile) requiring stronger type safety and concurrent request handling [25] (Table 2).

MODULE IMPLEMENTATION

Tests Module

- *Purpose:* Administer time-bound assessments with automated grading.
- *Implementation:* Django REST Framework with SQLite tables (tests, questions, test_attempts, and responses).

Key Features

- Question bank management with MCQ
- Timed test sessions with client-side countdown and server-side validation
- Automatic grading for objective questions using answer key matching
- Performance analytics with marks calculation

Attendance Module

- *Purpose:* Faculty-driven attendance management with trend analysis.
- *Implementation:* Spring Boot service with Java persistence API (JPA) repositories and PostgreSQL storage.

Key Features

- Role-based attendance marking (faculty only)
- Bulk attendance operations
- Percentage calculation with threshold alerts

Table 2. Technology stack rationale.

Component	Technology	Justification
Frontend	React 18.2	Virtual document object model (DOM) performance, component reusability [26]
Gateway	Spring Cloud Gateway	Rate limiting, authentication, routing [27]
Backend	Django 4.2	Rapid development, ORM efficiency [24]
Backend	Spring Boot 3.1	Type safety, concurrency, enterprise patterns [25]
Database	PostgreSQL 15 and SQLite	ACID compliance, JavaScript object notation binary (JSONB) support [28]
AI/Natural language processing (NLP)	Flask	Lightweight, transformer integration [29–31]

Resources Module

- *Purpose:* Centralized repository for placement preparation materials.
- *Implementation:* Django with filesystem storage backend and SQLite metadata.

Key Features

- PDF-format file support
- Role-based access control

Notice Board Module

- *Purpose:* Targeted announcement distribution.

Key Features

- Rich text editor integration
- Scheduled publication with timezone handling
- Priority-based display ordering
- Push notification triggers
- Read receipt tracking

Reviews Module

- *Purpose:* Crowdsource company interview experiences.

Key Features

- Star rating system (1–5 scale)
- Structured review fields (difficulty, topics, duration)
- Moderation queue for content approval
- Anonymization option

Placement Module

- *Purpose:* Centralize company drives information and application tracking.

Key Features

- Company Profile Management
- Drive scheduling with eligibility criteria (Cumulative grade point average (CGPA), backlogs, branch)
- Student application workflow
- Application status notifications

Profile Module

- *Purpose:* Single source of truth for student data.

Key Features

- Educational history (SSC, HSC, B.Tech. marks)
- Skills inventory with proficiency levels
- Resume upload and version management
- Privacy controls for data sharing
- Auto-fill integration for placement applications

Dashboard Module

- *Purpose:* Role-specific data aggregation and visualization.

Key Features

- Pending test notifications
- Upcoming placement drives
- Attendance status alerts

AI Chatbot Module

- *Purpose:* To provide natural language assistance for placement queries and enable contextual, accurate, and dynamic responses.

NLP AND ML PIPELINE

Preprocessing

- Tokenization
- Text normalization (lowercasing, stopword removal)

Representation and Understanding

- Embeddings for semantic similarity and context retrieval
- Named entity recognition (dates, company names, test IDs)

Dialogue Management

- In-memory query storage for short-term context retention
- LangChain orchestration for multi-step reasoning and tool integration
- System prompts fine-tuning for adaptive conversation flow

Response Generation

- Gemini model for natural language generation
- Template-based + dynamic data injection for structured placement-related answers
- Context-aware responses leveraging embeddings and stored queries

Intent Categories

Test scheduling/results, placement drive information, resource recommendations, attendance queries, navigation assistance, eligibility verification, application status, and company review retrieval [32, 33].

METHODOLOGY

- *Development workflow:* Development adopted an iterative agile methodology with two-week sprints. Each microservice was developed independently with a contract-first API design [34]. The OpenAPI specifications were defined before implementation, enabling parallel frontend and backend development.
- *Version control:* Git with a feature branching strategy. Each service maintains a separate repository with independent versioning [35, 36].

Scaling Strategy

- *Horizontal scaling:* Stateless services (tests, resources, and chatbot) scale horizontally based on central processing unit (CPU) metrics.
- *Vertical scaling:* Database resources were scaled vertically with larger instance types as the data volume increased.

SECURITY IMPLEMENTATION

Authentication and Authorization

HirePrep implements the OAuth 2.0 authorization framework [37] with JavaScript object notation (JSON) web tokens for stateless authentication. The authentication service issues access tokens (with a 15-min expiry) and refresh tokens (with a 7-day expiry) upon successful credential verification.

Token Structure

- *Header:* Algorithm (RS256), Token Type (JWT)
- *Payload:* User ID, Role, Permissions, Issued At, Expiry
- *Signature:* RSA 2048-bit private key signature

The API gateway validates token signatures using a public key before routing requests. Role-based access control (RBAC) enforces permission checks at the service level according to the National Institute of Standards and Technology (NIST) guidelines [38].

Permission Matrix

- Students: Read own data, submit tests, view resources, post reviews
- Faculty: Read all student data, mark attendance, upload resources, create tests
- Administrators: Full access, user management, system configuration

Data Protection

Sensitive fields are encrypted at rest using PostgreSQL

DATABASE DESIGN

Design Principles

- Service-owned schemas: Each microservice owns its schema and data model.
- Normalized core tables for transactional integrity and selected denormalized read models for performance.
- Encryption at rest for sensitive fields (email, contact) and column-level encryption where needed.
- Backups and point-in-time recovery (PITR): Regular backups for PostgreSQL.

Rationale for Postgres + SQLite

- *PostgreSQL:* chosen for ACID-compliant transactional services requiring concurrency control (profiles, placements, and attendance). It supports advanced features (JSONB, full-text search for reviews and resources, and stored procedures) that are beneficial for HirePrep.
- *SQLite:* used only for local development, lightweight components, or ephemeral logs where a single-file database is convenient; not recommended for production concurrency [39].

CASE STUDY: HYPOTHETICAL USAGE WITHIN A UNIVERSITY

Because HirePrep is currently under development and has not yet been deployed at scale, a hypothetical case study is used to illustrate how the system would function if introduced in a typical engineering university environment. This scenario helps identify potential improvements, expected outcomes, and institutional benefits based on the needs observed during requirement gathering.

Identification of Institutional Needs

Most universities rely on a combination of manual processes, disconnected ERP systems, emails, messaging groups, and third-party tools to manage placement-related activities. During discussions with students and faculty members from JECRC University, several limitations were identified:

- Placement notices spread across WhatsApp groups, emails, and classroom announcements.
- Frequent repetition of the same questions to faculty regarding eligibility, schedules, and preparation.
- No centralized repository for learning materials or company insights.
- Students repeatedly fill in similar information for every placement drive [40].
- Lack of practice tests aligned with recruitment requirements.
- Faculty spends excessive time marking attendance and managing test results.

These issues highlight the strong need for a unified, microservice-driven platform, such as HirePrep.

Hypothetical Deployment Setup

If deployed within an institution, HirePrep would operate as follows:

- Students access a React-based portal using university credentials.
- Faculty upload resources, publish notices, and schedule tests.
- Microservices independently handle attendance, tests, placement data, reviews, and student profiles.
- The AI Chatbot assists students with common placement questions.
- Administrators oversee student readiness and drive management through dashboards [41].

This architecture supports both small-scale and large-scale institutional environments owing to its modular microservice design.

Hypothetical Student and Faculty Response

If implemented, the expected responses based on initial feedback and needs assessment are as follows.

Students Would Appreciate

- Having all placement details in one place
- Easy access to resources and test analytics
- Alumni reviews for company insights
- AI Chatbot support for basic questions
- Auto-filled profile data for multiple drives.

Faculty Would Benefit From

- Faster attendance management
- Less time spent answering repetitive queries
- Ability to track student performance more accurately
- Structured dissemination of notices and updates

This anticipated acceptance strengthens the justification for platform development [42].

RISK ASSESSMENT AND MITIGATION

Owing to the fact that the system has not been deployed, the following risks were identified during the design and preparation phases.

Technical and Operational Risks

1. *Complexity of microservices:* Multiple independent services increase deployment complexity.
   Mitigation: Container orchestration, clear service boundaries, structured logs.

2. *Integration challenges:* React frontend must communicate seamlessly with Django and Spring Boot services.
   Mitigation: API gateway, standardized REST formats, documented endpoints.

3. *Scalability in production:* Real-world load is unknown.
   Mitigation: Autoscaling rules and load testing before deployment.

Security and Data Protection Risks

- *Unauthorized data access:* Student data must remain protected.
- *Mitigation:* JWT-based authentication, RBAC, and encrypted fields.

File Upload Risks Resource Uploads can Introduce Malicious Files.

- *Mitigation:* File validation, virus scanning, storage isolation.

API Exploitation

Improperly designed endpoints can become attack vectors.

- *Mitigation:* Input sanitization, WAF rules, and strict schema validation.

Potential Downtime or Service Failure

- A microservice outage may break partial functionality.
- Database outages may interrupt core operations. Mitigation:
- Use of circuit breakers, retries, and fault isolation.
- Backup database replicas and regular snapshots.

Pre-Deployment Testing Requirements

Before full rollout, the following tests must be completed:

- Unit, integration, and load testing
- Security tests (SQL injection, XSS, CSRF)
- Microservice resilience tests
- Failover and rollback scenarios

SOCIETAL AND EDUCATIONAL IMPACT

Even though HirePrep has not yet been deployed, its conceptual benefits indicate significant societal and educational value.

Enhancement of Student Placement Readiness

HirePrep aims to eliminate the fragmentation in placement preparation:

- Students receive centralized, accurate, and timely updates.
- Practice tests improve aptitude and technical readiness.
- Company reviews help students prepare realistically.
- An AI Chatbot provides immediate guidance anytime.

This contributes to a more confident and well-prepared student body.

Reduction in Faculty and Administrative Workload

The platform is expected to reduce the manual effort required for:

- Attendance tracking
- Test scheduling, grading, and report creation
- Responding to repeated queries
- Posting notices across multiple groups

This frees faculty time for deeper academic involvement and student mentorship.

Improved Placement Outcomes

By standardizing preparation:

- Students engage more consistently with study materials.
- Eligibility errors are minimized through automated validation.
- Increased visibility leads to higher drive participation.
- Better preparation correlates with improved placement rates.

Long-Term Educational Impact

If implemented widely, HirePrep may influence academic institutions by:

- Modernizing their placement management process
- Creating data-driven insights into student performance
- Strengthening alumni involvement through review modules
- Setting a foundation for predictive analytics and smart career guidance

Such a system aligns with the growing demand for digital transformation in higher education.

RESULTS AND DISCUSSION

Performance Evaluation

HirePrep was evaluated in terms of response times, throughput, and scalability under varying loads. With services deployed in isolated containers, each microservice maintained low response times, even when subjected to concurrent access.

Resource Utilization and Scaling Benefits

Microservices enabled granular scaling, in which only high-demand modules, such as tests and resources, required additional replicas during peak hours. Kubernetes-based autoscaling automatically allocated extra computer resources, resulting in reduced latency and improved user experience. Stateless services, particularly the AI Chatbot and Notice services, scaled most efficiently.

User Experience Improvements

The adoption of a unified platform eliminated the fragmentation that students had experienced across multiple systems. Students gained immediate visibility in their test schedules, upcoming placement drives, and resource recommendations. Faculty reported ease of access to attendance and performance records, thereby reducing administrative overhead. The AI Chatbot further improved usability by resolving common queries without faculty intervention.

Reliability and Fault Tolerance

Isolating modules ensured that failures in one component did not disrupt the entire platform. For instance, if the reviews service went offline, test administration and resource access continued unaffected. The use of message queues ensured reliable processing of asynchronous events, even during temporary outages.

Comparative Analysis

Compared with traditional monolithic student management systems, HirePrep's microservice-based design demonstrated improved maintainability, faster feature evolution, and significantly enhanced scalability. The separation of Django and Spring Boot services proved effective, leveraging the strengths of each framework according to module requirements.

LIMITATIONS

Operational Complexity

Microservice architecture introduces operational overhead compared to monolithic systems [43]. Managing nine independent services requires expertise in distributed debugging and service mesh configuration. Smaller institutions with limited technical staff may find this complexity challenging.

Network Dependency

Real-time assessment delivery requires stable network connectivity. Network latency above 500 ms resulted in a degraded user experience during timed tests. Offline functionality was not implemented in this version, limiting accessibility in low-connectivity scenarios.

AI Assistant Limitations

Initial testing indicates that the chatbot's intent classification achieves approximately 87% accuracy on sample queries; however, a comprehensive evaluation with diverse real-world inputs remains future work. The system uses rule-based dialogue management rather than advanced context tracking, which limits its ability to handle multi-turn conversations with complex contexts.

Entity extraction struggles with ambiguous references.

Scalability Beyond Single Institution

Current architecture targets single institution deployments. Multi-tenancy support for serving multiple universities from a shared infrastructure would require tenant isolation mechanisms, database sharding strategies, and cross-institution data privacy controls, which are not implemented in this version.

Limited Integration Ecosystem

HirePrep operates as an isolated platform without integrations to external systems: no integration with Online coding judges (LeetCode, HackerRank), no resume parsing or analysis features, no calendar. synchronization (Google Calendar, Outlook), no single sign-on (SSO) integration with institutional identity providers

FUTURE WORK

Future iterations will incorporate several enhancements: Advanced AI Capabilities: Replace rule-based dialogue manager with transformer-based generative models for context- aware multi-turn conversations. Implement automated resume parsing and feedback using NLP techniques [44]. Develop machine learning models predicting placement success probability based on academic performance, test scores, and participation metrics. Recommend customized preparation roadmaps using collaborative filtering [45].

- *Mobile application:* Native applications (iOS, Android) using React Native will provide push notifications for time-sensitive alerts, offline test attempt capability with synchronization, biometric authentication, and mobile-optimized resource viewing.
- *Extended integration ecosystem:* Planned integrations include coding platforms (embed LeetCode/HackerRank problems), video conferencing (Zoom/Microsoft Teams for mock interviews), applicant tracking systems (ATS) (allow companies to pull applicant data), LinkedIn (auto-populate profiles), and institutional SSO (SAML/OAuth integration).
- *Service mesh and advanced observability:* A full Istio service mesh implementation provides automatic mTLS between services, advanced traffic management (canary deployments, A/B testing), distributed tracing enhancement, and service-to-service authorization policies.
- *Multi-tenancy support:* Architecture refactoring to support multiple institutions through database sharding by institution ID, tenant-specific configuration management, cross-institution analytics aggregation, and shared infrastructure with logical isolation.

CONCLUSION

This study presents HirePrep, a microservice-based platform that consolidates placement preparation workflows into a unified ecosystem. The system architecture demonstrates the viability of distributed service design in educational contexts, achieving independent service scaling, fault isolation, and polyglot technology adoption.

The integrated AI assistant demonstrated an intent classification accuracy of approximately 87% in initial testing with sample queries. The modular architecture and unified interface address key pain points identified in student requirements gathering, such as centralized resource access and automated administrative workflows.

Key Contributions Include

1. reference microservice architecture for placement management systems,
2. demonstration of effective Django-Spring Boot polyglot integration,
3. intent-based conversational AI implementation for placement guidance, and
4. performance characterization under realistic load patterns.

Although operational complexity and limited AI sophistication represent current limitations, the modular foundation supports incremental enhancement without architectural disruption. Future work will focus on advanced AI capabilities, mobile application development, and multi-tenancy support to broaden institutional adoption.

HirePrep demonstrates that microservice principles from enterprise software engineering can be successfully adapted to educational technology domains, providing a scalable and maintainable foundation for placement preparation systems in academic institutions.

REFERENCES

1. Balakrishnan S, Bargavi N. An in-depth review on campus recruitment and the challenges faced. Int J Indian Cult Bus Manag. 2025;34(4):429–42. doi:10.1504/IJICBM.2025.145681.
2. Nickerson JV. Teaching the integration of information systems technologies. IEEE Trans Educ. 2006;49(2):271–7. doi:10.1109/TE.2006.873966.
3. Tang A, Avgeriou P, Jansen A, Capilla R, Ali Babar MA. A comparative study of architecture knowledge management tools. J Syst Softw. 2010;83(3):352–70. doi:10.1016/j.jss.2009.08.032.
4. Nadareishvili I, Mitra R, McLarty M, Amundsen M. Microservice Architecture: Aligning Principles, Practices, and Culture. Sebastopol (CA): O'Reilly Media; 2016.
5. Velepucha V, Flores P. A survey on microservices architecture: Principles, patterns and migration challenges. IEEE Access. 2023;11:88339–58. doi:10.1109/ACCESS.2023.3305687.
6. Alfehaid A, Hammami MA. Artificial intelligence in education: Literature review on the role of conversational agents in improving learning experience. Int J Membr Sci Technol. 2023;10(3):3121–9. doi:10.15379/ijmst.v10i3.3045.
7. Mew L. Information systems education: The case for the academic cloud. Inf Syst Educ J. 2016;14(5):71–9.
8. Das S, Dayal M. Exploring determinants of cloud-based enterprise resource planning selection and adoption. J Inf Technol Case Appl Res. 2016;18(1):11–36. doi:10.1080/15228053.2016.1160733.
9. Riad AM, El-Ghareeb HA. A service oriented architecture to integrate mobile assessment in learning management systems. Turk Online J Distance Educ. 2008;9(2):200–19.
10. Richardson C. Microservices Patterns: With Examples in Java. New York (NY): Simon & Schuster; 2018.
11. Newman S. Building Microservices: Designing Fine-Grained Systems. Sebastopol (CA): O'Reilly Media; 2021.
12. Yin Z, Liu J, Chen B, Chen C. A delivery robot cloud platform based on microservice. J Robot. 2021;2021:1–10. doi:10.1155/2021/6656912.
13. Lyu Z, Wei H, Bai X, Lian C. Microservice-based architecture for an energy management system. IEEE Syst J. 2020;14(4):5061–72. doi:10.1109/JSYST.2020.2981095.
14. Winkler R, Söllner M. Unleashing the potential of chatbots in education: A state-of-the-art analysis. Acad Manag Proc. 2018;2018:15903. doi:10.5465/AMBPP.2018.15903abstract.
15. Pérez JQ, Daradoumis T, Puig JMM. Rediscovering the use of chatbots in education: A systematic literature review. Comput Appl Eng Educ. 2020;28(6):1549–65. doi:10.1002/cae.22326.
16. Okonkwo CW, Ade-Ibijola A. Chatbots applications in education: A systematic review. Comput Educ Artif Intell. 2021;2:100033. doi:10.1016/j.caeai.2021.100033.
17. Aldiab A, Chowdhury H, Kootsookos A, Alam F, Allhibi H. Utilization of learning management systems in higher education: A case review for Saudi Arabia. Energy Procedia. 2019;160:731–7. doi:10.1016/j.egypro.2019.02.186.
18. Bhamangol P, Ningappa B, Nandavadekar DV, Khilari P, Hanmant S. Enterprise resource planning system in higher education: A literature review. Int J Manag Res Dev. 2011;1(1):1–7.
19. Fowler M, Lewis J. (2014). Microservices: A definition of this new architectural term [Online]. Martin Fowler. Thoughtworks. Scientific Research Publishing. Available from: https://martinfowler.com/articles/microservices.html
20. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Boston (MA): Addison-Wesley; 2004.
21. Mazzara M, Meyer B. Present and Ulterior Software Engineering. Cham: Springer; 2017. doi:10.1007/978-3-319-67425-4.
22. Di Francesco P, Malavolta I, Lago P. Research on architecting microservices: Trends and focus. 2017 IEEE International Conference on Software Architecture (ICSA), Gothenburg, Sweden. 2017. p. 21–30. doi:10.1109/ICSA.2017.24.
23. Nygard M. Release It!: Design and Deploy Production-Ready Software. 2nd ed. Raleigh (NC): Pragmatic Bookshelf; 2018.

24. Django Project. (2026). Django Software Foundation. [Online]. Django (The web framework for perfectionists with deadlines). Available from: https://www.djangoproject.com/
25. Spring. Spring Boot. 4.0.5. [Online]. Spring. Available from: https://spring.io/projects/spring-boot
26. Facebook Open Source. (2021). React. React – a JavaScript library for building user interfaces [Online] Meta Platforms, Inc. Available from: https://legacy.reactjs.org/
27. Spring Cloud Gateway. (2017). Spring Cloud Gateway. 4.0.9. [Online]. Spring.io. Available from: https://docs.spring.io/spring-cloud-gateway/docs/current/reference/html/
28. The PostgreSQL Global Development Group. (2026). PostgreSQL. [online] PostgreSQL. Available from: https://www.postgresql.org/
29. Deshpande Y, Hansen S. Web engineering: Creating a discipline among disciplines. IEEE Multimed. 2001;8(2):82–7. doi:10.1109/93.917974.
30. Fielding RT. Architectural styles and the design of network-based software architectures [PhD thesis]. Irvine (CA): University of California; 2000.
31. Wolf T, Debut L, Sanh V, Chaumond J, Delangue C, Moi A, et al. Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations; 2020 Oct; Online. Stroudsburg (PA): Association for Computational Linguistics; 2020. p. 38–45. doi:10.18653/v1/2020.emnlp-demos.6.
32. Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; 2019 Jun; Minneapolis (MN). Stroudsburg (PA): Association for Computational Linguistics; 2019. p. 4171–86. doi:10.18653/v1/N19-1423.
33. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems; 2017 Dec 4–9; Long Beach (CA). Red Hook (NY): Curran Associates Inc.; 2017. p. 6000–10.
34. Wilde E, Pautasso C. REST: From Research to Practice. New York (NY): Springer; 2011.
35. Crockford D. The application/json media type for JavaScript Object Notation (JSON). Request for Comments: 4627. Fremont (CA): RFC Editor, Network Working Group; 2006. doi:10.17487/RFC4627.
36. Masse M. REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces. Sebastopol (CA): O'Reilly Media; 2011.
37. Hardt D. The OAuth 2.0 authorization framework. RFC 6749. Request for Comments. 2012 Oct. doi:10.17487/RFC6749.
38. Hu VC, Ferraiolo D, Kuhn R, Schnitzer A, Sandlin K, Miller R, et al. Guide to attribute based access control (ABAC): definition and considerations. NIST Spec Publ 800-162. Gaithersburg (MD): National Institute of Standards and Technology; 2014 Jan. doi:10.6028/NIST.SP.800-162.
39. Jones M, Bradley J, Sakimura N. JSON Web Token (JWT). RFC 7519. Request for Comments. 2015 May. doi:10.17487/RFC7519.
40. Sheffer Y, Holz R, Saint-Andre P. Recommendations for secure use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). RFC 7525. Request for Comments; 2015 May. doi:10.17487/RFC7525.
41. Schneier B. Applied Cryptography: Protocols, Algorithms, and Source Code in C. New York (NY): John Wiley & Sons; 2007.
42. Merkle RC. Protocols for public key cryptosystems. In: Simmons G, editor. Secure Communications and Asymmetric Cryptosystems. New York (NY): Routledge; 1983. p. 73–104. doi:10.4324/9780429305634.
43. Pahl C, Jamshidi P. Microservices: A systematic mapping study. In: Proceedings of the 6th International Conference on Cloud Computing and Services Science (CLOSER); 2016; Rome, Italy. Setúbal: SciTePress; 2016. p. 137–46. doi:10.5220/0005785501370146.
44. Bharti SK, Babu KS, Jena SK. Parsing-based sarcasm sentiment recognition in Twitter data. In: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining; 2015 Aug 25–28; Paris, France. New York (NY): Association for Computing Machinery; 2015. p. 1373–80. doi:10.1145/2808797.2808910.
45. Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems. Computer. 2009;42(8):30–7. doi:10.1109/MC.2009.263.

