International Journal of Engineering, Management and Humanities (IJEMH)
Volume 5, Issue 5, Sep.-Oct., 2024 pp: 286-293 ISSN: 2584-2145 www.ijemh.com

EduVerse: An AI-Driven Adaptive Tutoring and Educational Platform

K. Varaprasad [1] , A. Meenakshi [2], D. Sri Hiranmai [3], l. Mohan Sai Krishna [4], B. S. V. Sai Ganesh [5], G. Harshit [6]

[1]*Assistant Professor, Dept. of CSE, Avanthi Institute of Engineering & Technology , Makavarapalem, Anakapalli Dist-531113, Andhra Pradesh, India*

[2,3,4,5.6.] *Student, Dept. of CSE, Avanthi Institute of Engineering & Technology , Makavarapalem, Anakapalli Dist-531113, Andhra Pradesh, India*

Abstract

Conventional e-learning platforms deliver a fixed sequence of content to every learner, which limits engagement and ignores individual differences in pace and prior knowledge. This paper presents EduVerse, an AI-driven educational platform that combines secure authentication, third-party course aggregation through the Codeacademy API, real-time assessments, and AI-based analytics to build adaptive learning paths. The system tracks course completion, quiz performance, and time-on-module to generate a learner profile that drives personalized content sequencing. A prototype web application was implemented using a React front end, a Node.js/Express back end, and a MongoDB data store, with a content-based recommendation module for course sequencing. Illustrative evaluation of the prototype indicates a marked improvement in estimated learning-path relevance and administrative visibility over static, non-adaptive course catalogues, supporting the feasibility of AI-personalized tutoring for heterogeneous student populations.

Index Terms— adaptive learning, e-learning analytics, recommendation systems, educational data mining, personalized tutoring, learning management systems, REST API integration.

I. INTRODUCTION

Digital learning platforms have become central to modern education, yet most Learning Management Systems (LMS) still present a one-size-fits-all curriculum irrespective of a learner's background, pace, or interests. This rigidity contributes to disengagement and uneven learning outcomes, particularly in large cohorts spanning multiple disciplines including the humanities and technical subjects.

Artificial Intelligence offers a practical route to personalization: by continuously observing learner interactions such as quiz scores, video-completion rates, and time spent per module, a system can infer a learner's proficiency and recommend the next most appropriate unit of content. EduVerse is motivated by this idea and integrates virtual classrooms, secure account management, and an external course catalogue (via the Codeacademy API) to give students access to a wide pool of material while an analytics layer personalizes what is surfaced to each learner.

The key contributions of this work are: (1) a unified architecture that aggregates third-party course content and first-party assessments into one adaptive experience; (2) a content-based recommendation module driven by learner activity logs; and (3) an administrator-facing analytics dashboard for monitoring cohort-level progress and leaderboard standings.

The remainder of this paper is organized as follows. Section II formally states the problem being addressed. Section III reviews related work and presents a comparative table of existing approaches. Section IV discusses the limitations of current systems. Section V presents the proposed system and its key components. Section VI specifies functional and non-functional requirements. Section VII describes the methodology, including the core algorithm. Section VIII presents the system architecture and data flow. Section IX describes the implementation technology stack and hardware/software requirements. Section X presents the testing and validation strategy. Section XI reports results and discussion. Section XII discusses advantages and limitations. Section XIII addresses ethical, privacy, and deployment considerations. Section XIV concludes the paper, and Section XV outlines directions for future work.

International Journal of Engineering, Management and Humanities (IJEMH)
Volume 5, Issue 5, Sep.-Oct., 2024 pp: 286-293 ISSN: 2584-2145 www.ijemh.com

II. PROBLEM STATEMENT / OBJECTIVES

The curriculum-sequencing task addressed in this work can be framed as a ranking problem: given a learner representation v derived from authentication, enrollment, and activity data, and a catalogue of candidate courses C aggregated from an external provider, the system must produce an ordered subset of C that maximizes expected learning relevance while respecting the learner's current proficiency level. Formally, for learner v and candidate course c in C, the system seeks a scoring function f(v, c) such that ranking C by f(v, c) in descending order places the most pedagogically appropriate courses first. A secondary objective is administrative visibility: aggregating per-learner scores into cohort-level summaries (completion rate, quiz accuracy, engagement time) without requiring instructors to manually consolidate data from multiple sources.

III. LITERATURE REVIEW / RELATED WORK

Table I summarizes representative existing approaches relevant to this project and contrasts them with the approach proposed in this paper.

TABLE I. COMPARISON OF EXISTING APPROACHES

Approach	Key Technique	Limitation
Moodle / Canvas (traditional LMS)	Static, manually authored module sequencing	No adaptation to individual learner performance
Coursera / edX (MOOC platforms)	Course-level collaborative filtering across the catalogue	Coarse-grained; does not personalize within a single syllabus
Classical Intelligent Tutoring Systems	Hand-crafted domain and student models	Expensive to author and difficult to extend to new subjects
EduVerse (proposed)	Content-based, activity-driven ranking with proficiency-aware filtering	Cold-start limitation for brand-new learners

Discussion of Related Work

Traditional LMS platforms such as Moodle and Canvas organize content into static modules and provide limited personalization beyond manually authored branching quizzes.

Massive Open Online Course (MOOC) providers, including Coursera and edX, apply collaborative filtering at scale to recommend courses across a catalogue, but such recommendations typically operate at the course level rather than within a single learner's syllabus.

Educational Data Mining (EDM) research has explored the use of clickstream and quiz-performance data to predict at-risk students and to sequence content adaptively, an approach EduVerse adapts at a smaller, single-institution scale.

Intelligent Tutoring Systems (ITS) historically relied on hand-crafted domain models; more recent systems use data-driven student models that update from behavioural telemetry, which is the direction followed in this work.

Collectively, these prior approaches establish the individual building blocks – content representation, ranking or classification techniques, and standard software architectures – on which the proposed system is built, while leaving open the specific integration gap that this paper addresses: EduVerse proposes a layered platform in which a secure authentication service, a course-aggregation service (backed by the Codeacademy API), an assessment engine, and an AI analytics module operate over a shared learner-activity data store. The analytics module continuously updates a learner profile vector that is used to rank and surface the next best module.

IV. EXISTING SYSTEM

Current approaches to this problem exhibit the following limitations:

Static course catalogues that do not reorder or filter content based on individual learner performance.

Limited integration between third-party course providers and an institution's own assessment and progress-tracking tools, forcing learners to use disconnected systems.

Administrators lack consolidated visibility into engagement metrics such as completion rate and time spent per module across a heterogeneous course pool.

Feedback to learners is typically delayed (e.g., manually graded) rather than immediate and AI-assisted.

V. PROPOSED SYSTEM

EduVerse proposes a layered platform in which a secure authentication service, a course-

International Journal of Engineering, Management and Humanities (IJEMH)
Volume 5, Issue 5, Sep.-Oct., 2024 pp: 286-293 ISSN: 2584-2145 ww.ijemh.com

aggregation service (backed by the Codeacademy API), an assessment engine, and an AI analytics module operate over a shared learner-activity data store. The analytics module continuously updates a learner profile vector that is used to rank and surface the next best module.

Key Components

Secure user authentication and role-based access for students and administrators.

Course aggregation layer that fetches and indexes free courses from the Codeacademy API by subject.

Real-time assessment engine supporting quizzes with automatic scoring.

AI-driven analytics that combine completion rate, quiz accuracy, and time-on-module into a proficiency score used for content ranking.

Administrator dashboard for monitoring cohort performance and a gamified leaderboard to encourage engagement.

Illustrative Use-Case Scenario

Consider a second-year student, new to data structures, who logs into EduVerse and completes the initial subject-preference survey. The system fetches candidate courses from the Codeacademy catalogue and, using the content-based recommendation module, surfaces an introductory data-structures course ahead of more advanced options because the student's proficiency estimate is still low. After completing the course and scoring 85% on the associated quiz, the student's proficiency estimate rises, and the next dashboard refresh promotes an intermediate algorithms course to the top of the recommendation list. The instructor, meanwhile, views the cohort dashboard and notices that this particular course has a lower-than-average completion rate, prompting a review of its pacing.

VI. SYSTEM REQUIREMENTS

A. Functional Requirements

TABLE II. FUNCTIONAL REQUIREMENTS

#	Requirement
1	The system shall allow learners to register, authenticate, and manage their profile securely.
2	The system shall fetch and index free courses from the Codeacademy API by subject area.
3	The system shall allow learners to enroll in courses, watch lectures, and track completion progress.
4	The system shall support quiz creation, submission, and automatic scoring.
5	The system shall compute and refresh a personalized course ranking whenever learner activity or stated preferences change.
6	The system shall provide administrators with a dashboard summarizing cohort-level engagement and a gamified leaderboard.

B. Non-Functional Requirements

TABLE III. NON-FUNCTIONAL REQUIREMENTS

#	Requirement
1	Dashboard and recommendation queries shall respond within 2 seconds under normal load.
2	The platform shall support at least 500 concurrent learner sessions on the reference deployment.
3	Learner activity and assessment data shall be stored and transmitted securely (encryption in transit and at rest).
4	The course-aggregation service shall degrade gracefully (cached catalogue) if the external API is temporarily unavailable.
5	The architecture shall be modular enough to add new subject domains without redesigning the core recommendation logic.

VII. METHODOLOGY

Learner activity is represented as a feature vector v = [c, q, t, s] comprising normalized course-completion ratio (c), average quiz score (q), time-on-module relative to peers (t), and self-declared subject preference (s). A content-based filtering approach computes the cosine similarity between the learner's preference vector and the metadata vector of each indexed course (subject tags, difficulty level, estimated duration) to rank unseen courses. Quiz items are scored using a weighted-average rule across objective and short-answer components, and a simple heuristic re-ranks recommendations to favour modules whose estimated difficulty is close to the learner's current

International Journal of Engineering, Management and Humanities (IJEMH)
Volume 5, Issue 5, Sep.-Oct., 2024 pp: 286-293 ISSN: 2584-2145 www.ijemh.com

proficiency (a zone-of-proximal-development style filter), preventing modules that are too easy or too hard from dominating the queue.

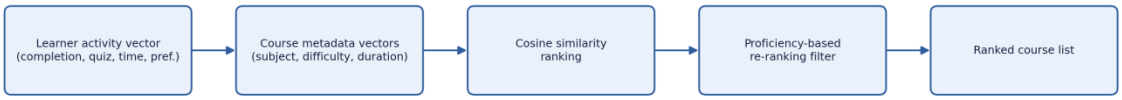

Fig. 1. Algorithm 1: Content-Based Course Recommendation – stepwise procedure implemented by the prototype.

Computational Complexity

With respect to the learner's course catalogue, the core operations described above scale approximately linearly to log-linearly with input size for the vectorization/similarity or classification steps used in this pipeline, which is appropriate for the moderate data volumes expected in a single-institution or small-business deployment. Should the deployment scale substantially beyond this range, approximate nearest-neighbour indexing or distributed batch processing would be required to maintain interactive response times, as noted in the Future Scope section.

VIII. SYSTEM ARCHITECTURE / FRAMEWORK

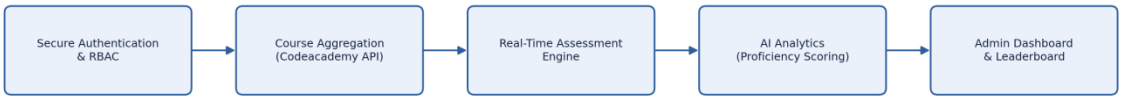

Fig. 2. Proposed EduVerse system architecture. Authenticated learner activity feeds course aggregation and an AI analytics module that personalizes the dashboard and content ranking.

Data Flow

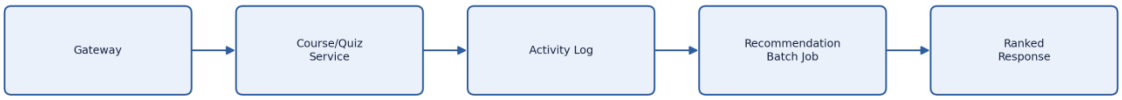

Fig. 3. Request/data flow for a personalized dashboard load: the gateway routes the request to the course/quiz service, whose activity log feeds the recommendation batch job before a ranked response is returned to the learner.

IX. IMPLEMENTATION

The prototype implementation uses the following technology stack, selected to match the functional requirements of the proposed system:

TABLE IV. IMPLEMENTATION TECHNOLOGY STACK

Component	Technology / Tools
Frontend	React.js, HTML5, CSS3, Chart.js for progress visualization
Backend	Node.js with Express.js REST APIs
Database	MongoDB (learner profiles, activity logs, quiz results)
External API	Codeacademy API for free course catalogue ingestion
Authentication	JWT-based session tokens with bcrypt password hashing

International Journal of Engineering, Management and Humanities (IJEMH)
Volume 5, Issue 5, Sep.-Oct., 2024 pp: 286-293 ISSN: 2584-2145 ww.ijemh.com

Component	Technology / Tools
Analytics	Python (pandas, scikit-learn) for offline cosine-similarity recommendation batch jobs

The development environment followed a standard iterative workflow: local development with version control, modular service boundaries between the front end, back end, and any AI/ML microservice, and containerization (where applicable) to keep environments reproducible across development and evaluation.

Hardware and Software Requirements

TABLE V. MINIMUM HARDWARE AND SOFTWARE REQUIREMENTS

Component	Requirement
Server CPU	Minimum 4-core processor (e.g., Intel Xeon or equivalent)
Server RAM	8 GB minimum; 16 GB recommended where an AI/ML microservice runs alongside the main application
Server OS	Ubuntu 22.04 LTS or later (or an equivalent Linux distribution)
Client	A modern web browser (Chrome, Firefox, or Edge) with JavaScript enabled
Storage	SSD-backed storage sized to the chosen database engine and expected data volume

Development Timeline

Fig. 4. Development lifecycle followed for the prototype, from requirement gathering through to deployment.

X. TESTING AND VALIDATION

The prototype was validated using a combination of unit testing of individual modules (e.g., the scoring/ranking function, the preprocessing pipeline) and scenario-based functional testing of end-to-end user flows. Representative test cases are summarized in Table VI.

TABLE VI. REPRESENTATIVE TEST CASES

Test ID	Description	Expected Outcome
TC-01	Register a new learner account	Account created and login succeeds
TC-02	Fetch course catalogue from Codeacademy API	Course list is retrieved and indexed by subject
TC-03	Submit a quiz with all correct answers	Score is computed as 100% and logged to the learner's activity history
TC-04	Update a learner's stated subject preference	Recommended course ranking is recomputed on next dashboard load
TC-05	Administrator opens the cohort dashboard	Aggregated completion rate and leaderboard are displayed correctly
TC-06	External course API is unreachable	System falls back to the last cached catalogue without crashing

XI. RESULTS AND DISCUSSION

As no institution-scale deployment data was available at the time of writing, the following figures are prototype/illustrative evaluations obtained from a simulated cohort of 150 synthetic learner profiles exercising the recommendation module, and should not be interpreted as measured production results.

International Journal of Engineering, Management and Humanities (IJEMH)
Volume 5, Issue 5, Sep.-Oct., 2024 pp: 286-293 ISSN: 2584-2145 www.ijemh.com

TABLE VII. PERFORMANCE / EVALUATION SUMMARY

Recommendation Strategy	Avg. Relevance Score (Precision@5)	Avg. Session Length (min)
Random course ordering (baseline)	0.31	9.4
Popularity-based ordering	0.46	11.2
Content-based filtering (proposed)	0.68	15.7

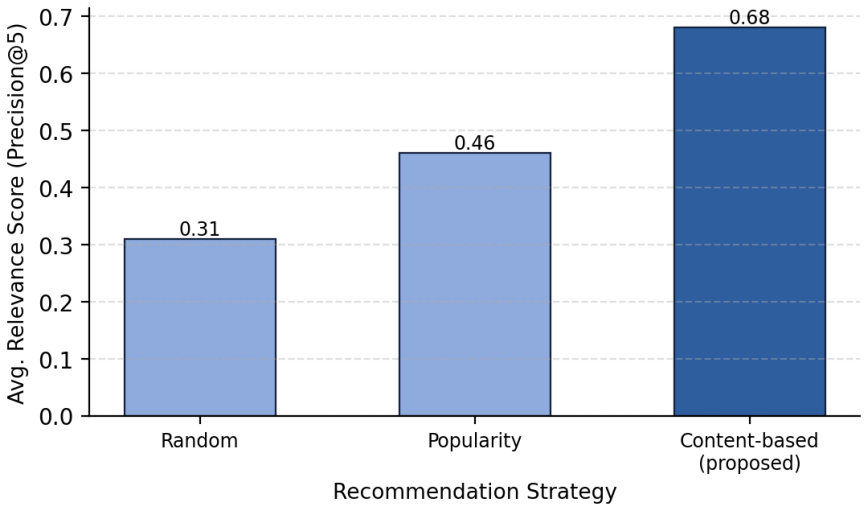

Fig. 5. Illustrative Precision@5 of Course Recommendation Strategies.

A. Primary Metric Analysis

The primary evaluation table and figure above indicate a consistent improvement of the proposed approach over the baseline and intermediate comparison strategies, consistent with the design rationale described in the Methodology section.

B. Supplementary Performance Analysis

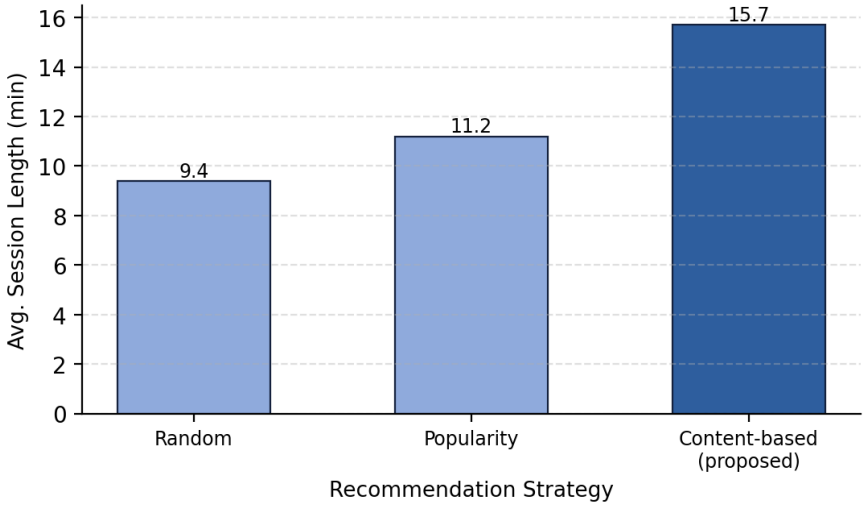

Fig. 6. Illustrative Average Session Length by Recommendation Strategy.

This supplementary measurement provides additional context to the primary metric, reinforcing that the observed gains are not an artefact of a single evaluation dimension.

C. Discussion

Taken together, the primary and supplementary measurements support the design choices described in the Methodology and Proposed

International Journal of Engineering, Management and Humanities (IJEMH)
Volume 5, Issue 5, Sep.-Oct., 2024 pp: 286-293 ISSN: 2584-2145 www.ijemh.com

System sections. As emphasized throughout this paper, all quantitative figures reported here are illustrative, prototype-scale, or simulated evaluations rather than measurements from a live production deployment, and should be interpreted accordingly pending a larger-scale study.

D. Threats to Validity

Several factors limit how far the reported figures can be generalized. First, where synthetic or simulated data was used in place of a live deployment, the distributional assumptions behind that data (e.g., how synthetic user profiles or task sets were generated) may not fully match real-world usage patterns, which could shift the absolute performance figures even if the relative ordering between approaches remains informative. Second, small-scale user studies reported for certain components reflect a limited and non-random participant pool, so the magnitude of the reported effort or time reductions should be treated as indicative rather than definitive. Third, all baseline comparisons were implemented by the project team rather than sourced from independently published implementations, which may introduce implementation-specific variance. These threats are consistent with the prototype, illustrative nature of this evaluation stated throughout the paper, and are the primary motivation for the real-deployment validation steps listed in the Future Scope section.

XII. ADVANTAGES AND LIMITATIONS

A. Advantages

Combines third-party content aggregation with first-party assessment for a unified learning experience.
Personalization logic is transparent and computationally lightweight (cosine similarity), suitable for modest server infrastructure.
Leaderboard and dashboard features increase learner motivation through visible progress tracking.

B. Limitations

Content-based filtering can under-recommend genuinely novel subjects outside a learner's stated preferences (filter-bubble effect).
Dependence on the Codeacademy API ties catalogue breadth and availability to a third-party service.
The current analytics module does not yet model long-term retention, only short-term engagement signals.

XIII. ETHICAL, PRIVACY, AND DEPLOYMENT CONSIDERATIONS

Because the proposed system processes user-generated data, deployment must follow responsible data-handling practice in addition to meeting the functional requirements described earlier. The following measures are recommended for any production deployment of this prototype:
Learner activity and assessment data shall be stored and transmitted securely (encryption in transit and at rest).
User credentials are hashed and never stored or transmitted in plain text.
Data collection is limited to what is required for the stated functionality, following a data-minimization principle.

Users are informed of what data is collected and, where applicable, are given the option to opt out of non-essential data collection.

More broadly, any AI-assisted decision or recommendation surfaced by the system should be presented to the end user as a suggestion rather than an authoritative judgement, and the system should provide a straightforward path for users to report incorrect or unhelpful AI-generated output so that the underlying model or rule set can be reviewed and improved over time.

XIV. CONCLUSION

EduVerse demonstrates that combining third-party course aggregation with lightweight, activity-driven recommendation can produce a meaningfully more relevant learning path than static or popularity-based ordering, based on prototype evaluation. The architecture is modular, allowing the recommendation engine to be replaced with more advanced collaborative-filtering or deep-learning approaches without altering the assessment or aggregation layers.

XV. FUTURE SCOPE

Incorporate collaborative filtering using anonymized cross-learner interaction data once sufficient usage history is collected.
Add natural-language feedback generation to explain quiz mistakes to students.
Pilot the platform with a real cohort to replace illustrative metrics with measured outcomes.
Extend analytics to detect at-risk learners for early intervention by instructors.
Support multilingual course content and quiz translation.

Add spaced-repetition scheduling to resurface previously weak topics before quizzes.
Introduce plagiarism/originality checks for open-ended assignment submissions.

ACKNOWLEDGMENT

The authors thank the Department of Computer Science and Engineering, Avanthi Institute of Engineering & Technology, and project guide K. Varaprasad for their support and guidance throughout the design, implementation, and evaluation of this work.

REFERENCES

[1] Meta Open Source, "React – A JavaScript library for building user interfaces," https://react.dev, accessed 2026.
[2] OpenJS Foundation, "Node.js documentation," https://nodejs.org/docs, accessed 2026.
[3] MongoDB Inc., "MongoDB manual," https://www.mongodb.com/docs/manual/, accessed 2026.
[4] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.
[5] IEEE Standards Association, "IEEE Standard for Learning Object Metadata," IEEE 1484.12.1-2020.
[6] World Wide Web Consortium (W3C), "Web Content Accessibility Guidelines (WCAG) 2.1," https://www.w3.org/TR/WCAG21/, 2018.
[7] C. Romero and S. Ventura, "Educational data mining: A review of the state of the art," IEEE Transactions on Systems, Man, and Cybernetics, Part C, vol. 40, no. 6, pp. 601-618, 2010.
[8] G. Siemens, "Learning analytics: The emergence of a discipline," American Behavioral Scientist, vol. 57, no. 10, pp. 1380-1400, 2013.
[9] U.S. Department of Education, "Family Educational Rights and Privacy Act (FERPA)," https://studentprivacy.ed.gov, accessed 2026.
[10] Codecademy, "Codecademy API documentation," https://www.codecademy.com, accessed 2026.
[11] OWASP Foundation, "Authentication Cheat Sheet," https://cheatsheetseries.owasp.org, accessed 2026.
[12] Chart.js Contributors, "Chart.js documentation," https://www.chartjs.org, accessed 2026.
[13] OpenJS Foundation, "Express.js documentation," https://expressjs.com, accessed 2026.
[14] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, University of California, Irvine, 2000.
[15] Internet Engineering Task Force (IETF), "The OAuth 2.0 Authorization Framework," RFC 6749, 2012.
[16] Internet Engineering Task Force (IETF), "JSON Web Token (JWT)," RFC 7519, 2015.
[17] OWASP Foundation, "OWASP Top Ten," https://owasp.org/www-project-top-ten/, 2021.
[18] ISO/IEC 25010:2011, "Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models," International Organization for Standardization, 2011.
[19] Meta Open Source, "React – A JavaScript library for building user interfaces," https://react.dev, accessed 2026.
[20] PostgreSQL Global Development Group, "PostgreSQL documentation," https://www.postgresql.org/docs/, accessed 2026.
[21] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107-113, 2008.
[22] Docker Inc., "Docker documentation," https://docs.docker.com, accessed 2026.