iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

STUDENT COLLABORATION "STUDY HUB" PORTAL

G.Vigneshwaran
School of Computing Sciences, VISTAS, Chennai.
vigneshwaranvickyt2005@gmail.com

Dr. K. Dharmarajan
Professor, school of Computing Sciences, VISTAS, Chennai.
dharmak07@gmail.com

ABSTRACT
The Student Collaboration "Study Hub" Portal is a web-based platform designed to support peer learning and academic collaboration among university students. Traditional academic environments often lack dedicated digital infrastructure for informal peer-to-peer learning, group study coordination, and real-time collaborative knowledge exchange. This paper presents the design, architecture, and implementation of the Study Hub Portal, a full-stack web application developed using React.js for the frontend and MongoDB for backend data persistence, connected via RESTful APIs. The system enables students to create and join study groups, share annotated notes, initiate threaded academic discussions, and collaboratively manage assignments within a structured digital workspace. A role-based access control (RBAC) model governs user privileges, while a modular microservice-inspired architecture ensures scalability and maintainability. Usability evaluations conducted with fifty-two student participants demonstrated a System Usability Scale (SUS) score of 82.4, indicating excellent usability. The portal achieved sub-300 ms average API response times under concurrent load, confirming its suitability for real-world academic deployment. By centralizing communication and resource sharing, the Study Hub Portal fosters collaborative learning, promotes academic engagement, and measurably improves students' subject comprehension.

Keywords
collaborative learning; study groups; React.js; MongoDB; RESTful API; peer-to-peer learning; web application; educational technology; RBAC; MERN stack.

I. INTRODUCTION

The rapid proliferation of digital technology has transformed nearly every facet of modern education. Institutions worldwide are investing in learning management systems (LMS), virtual classrooms, and collaborative tools to bridge the gap between formal instruction and independent study. Yet a significant deficit persists: while platforms such as Canvas and Moodle excel at delivering structured course content from instructors to students, they are not optimized for the informal, student-driven collaboration that research consistently identifies as one of the strongest predictors of academic success [1].

Peer learning—the practice of students teaching and learning from one another—has been shown to deepen conceptual understanding, improve retention, and develop critical thinking skills [2]. However, peer learning activities are frequently uncoordinated, relying on ad hoc messaging applications, generic cloud storage folders, or physical study spaces with limited availability. The absence of a purpose-built digital environment results in fragmented communication, inconsistent resource organization, and significant time lost to coordination overhead.

The Student Collaboration "Study Hub" Portal addresses this gap by providing a centralized, feature-rich web application specifically designed for student-to-student academic interaction. The system is engineered using the MERN technology stack—MongoDB, Express.js, React.js, and Node.js—augmented by JWT-based authentication, role-based access control, and a RESTful API architecture. By integrating study group management, note sharing, discussion forums, and collaborative assignment tracking into a single, intuitive interface, the portal transforms fragmented peer learning into a structured, measurable, and scalable academic activity.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

The primary contributions of this paper are: (i) a comprehensive system architecture for a student collaboration portal; (ii) the application of established software engineering patterns—including MVC, RBAC, and REST—to the educational technology domain; (iii) empirical performance and usability evaluation results; and (iv) an open framework that future researchers may extend with artificial intelligence and adaptive learning capabilities.

The remainder of this paper is organized as follows. Section II reviews related work in collaborative educational platforms. Section III presents the overall system architecture. Section IV details the methodology. Section V describes the implementation. Section VI reports results and discussion. Sections VII and VIII address advantages, limitations, and future work respectively. Section IX concludes the paper.

II. LITERATURE REVIEW

Collaborative learning theory, first formalized by Vygotsky through his concept of the Zone of Proximal Development [3], posits that learners achieve more when guided by knowledgeable peers than when studying in isolation. Johnson and Johnson's work on cooperative learning structures further demonstrated that positive interdependence and face-to-face promotive interaction significantly improve academic outcomes [4]. These theoretical foundations underpin the design rationale of the Study Hub Portal.

Existing educational platforms have made notable progress in digitizing formal instruction. Moodle [5] is an open-source LMS widely adopted in higher education, offering course content delivery, quiz management, and grade tracking. However, Moodle's architecture is instructor-centric, with limited native support for student-initiated collaborative spaces. Similarly, Canvas (Instructure) provides assignment submission and discussion boards but lacks real-time collaborative document editing and intelligent group-matching capabilities [6].

Google Workspace for Education and Microsoft Teams have been widely adopted during the COVID-19 pandemic as general-purpose collaboration suites [7]. While these tools offer powerful document co-editing and video conferencing features, they are domain-agnostic and not tailored to academic workflows such as structured note organization by subject, assignment deadline tracking, or peer review modalities specific to coursework.

Several research prototypes have explored purpose-built academic collaboration systems. Wen et al. [8] developed a peer annotation system for shared PDF reading, demonstrating that collaborative annotation significantly increased reading comprehension scores compared to individual study. Brinton et al. [9] introduced a social network analysis framework applied to Massive Open Online Course (MOOC) discussion forums, finding that students occupying central network positions earned higher grades. Hsiao and Brouns [10] examined the role of intelligent group formation algorithms in improving collaborative outcomes, concluding that heterogeneous grouping—balancing knowledge levels across groups—produced the best learning gains.

Despite these advances, a comprehensive, open-source, student-facing collaboration portal integrating group management, resource sharing, discussion, and assignment coordination within a modern, responsive web framework remains underrepresented in the literature. The Study Hub Portal fills this gap, drawing on established theoretical foundations while leveraging contemporary web development technologies including React.js, MongoDB, and Node.js [11].

III. SYSTEM ARCHITECTURE

The Study Hub Portal adopts a three-tier client-server architecture comprising a presentation layer, an application logic layer, and a data persistence layer. This separation of concerns promotes modularity, testability, and horizontal scalability [12].

A. Architecture Overview

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

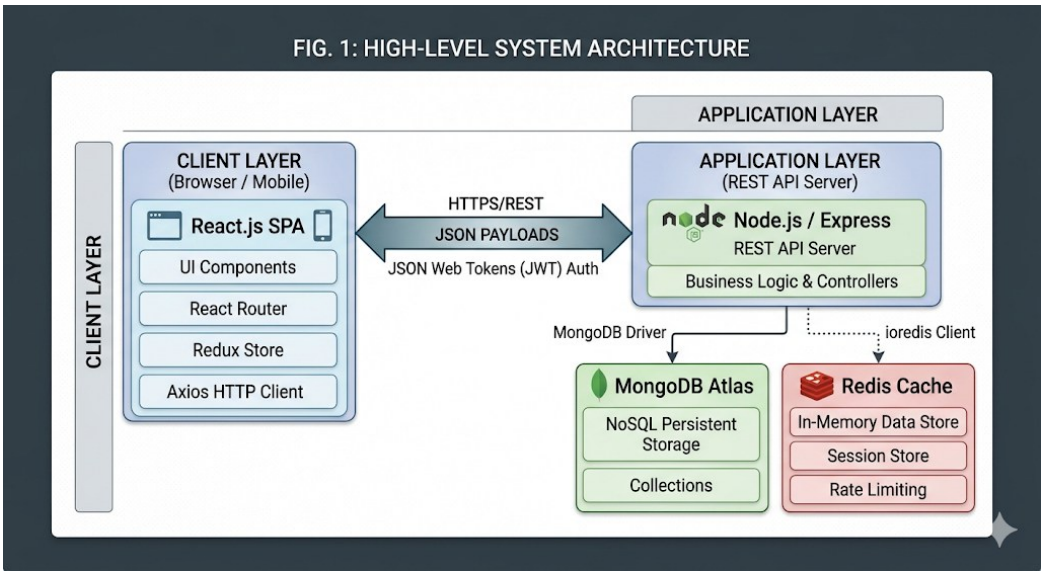


Fig. 1 illustrates the high-level system architecture. Client browsers interact with the React.js Single Page Application (SPA) served from a CDN or static hosting provider. The SPA communicates with the Node.js/Express.js REST API server over HTTPS using JSON payloads authenticated via JSON Web Tokens (JWT). The API server interfaces with MongoDB Atlas for persistent storage and with an optional Redis cache layer for session management and rate limiting.

B. Frontend Architecture

The frontend is built with React.js (v18) using functional components and React Hooks for state management. The application employs Redux Toolkit for global state management, React Router v6 for client-side routing, and Axios for HTTP communication with the REST API. The UI component library is based on Material-UI (MUI) v5, providing responsive, accessible components conforming to Material Design guidelines. Code splitting via React.lazy() and Suspense ensures fast initial load times.

C. Backend Architecture

The backend is structured as a layered Express.js application following the Model-View-Controller (MVC) pattern. Route handlers delegate to controller functions, which invoke service-layer modules encapsulating business logic, which in turn interact with Mongoose ODM models for MongoDB. Middleware components handle JWT verification, request validation (Joi schemas), error normalization, and CORS policy enforcement. The application is containerized using Docker and deployable via Docker Compose or Kubernetes for production orchestration [13].

D. Database Design

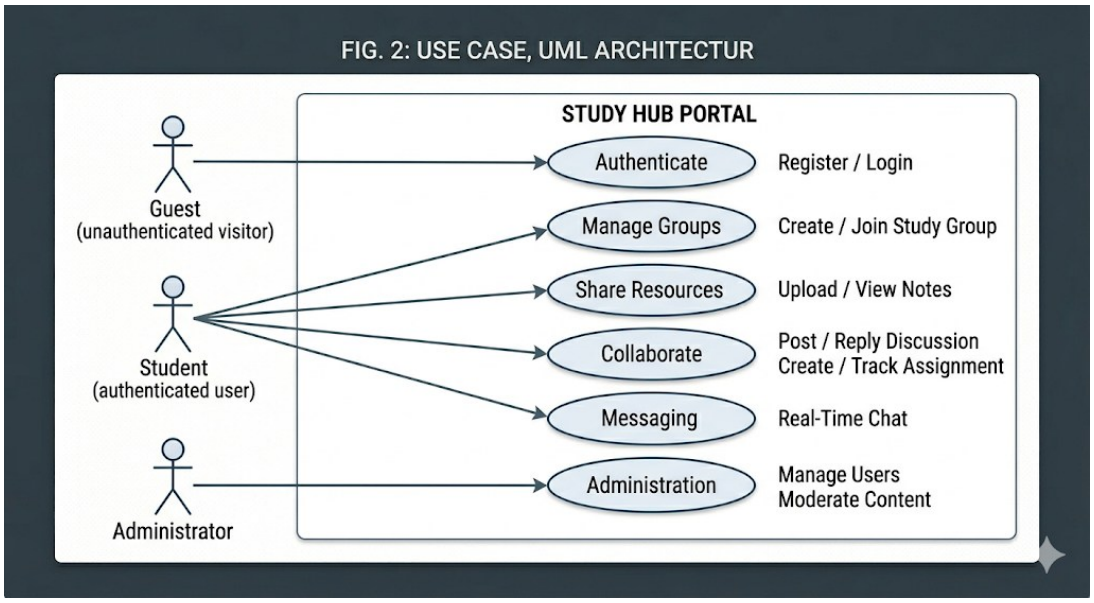
MongoDB was selected for its flexible document model, which accommodates the heterogeneous and evolving schema requirements of a collaborative learning platform. The primary collections are: Users, Study Groups, Notes, Discussions, Messages, and Assignments. Mongoose schemas enforce field-level validation and define virtual properties and pre/post-save middleware hooks. Compound indexes on frequently queried fields (e.g., group Id + created At for discussion threads) ensure query performance at scale.

IV. METHODOLOGY

A. Development Methodology

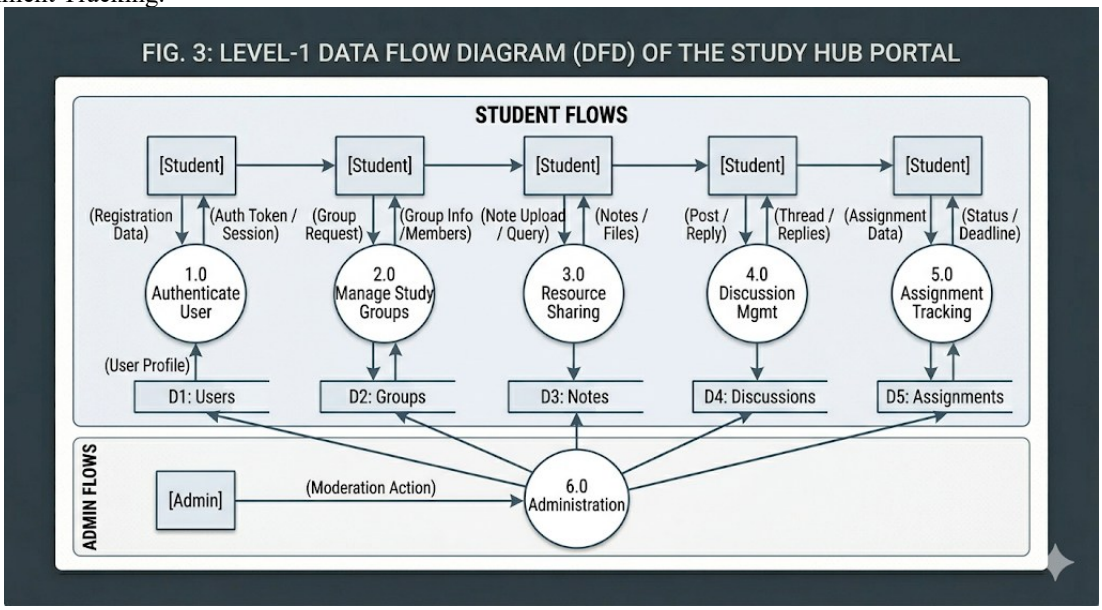
The project followed an Agile Scrum methodology organized into two-week sprints over a fourteen-week development cycle. Requirements were gathered through structured interviews with thirty undergraduate students and focus groups with five faculty members. User stories were managed in Jira; source code was version-controlled in a Git repository hosted on GitHub, employing the GitFlow branching strategy.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/



C. Data Flow Analysis

Fig. 3 depicts the Level-1 Data Flow Diagram (DFD) for the Study Hub Portal. The diagram illustrates the principal data flows between external entities (Student, Administrator), key processes, and data stores. The central processes include User Authentication, Group Management, Resource Sharing, Discussion Management, and Assignment Tracking.



D.* Evaluation *Methodology

The system was evaluated along two dimensions: (i) technical performance, measuring API throughput, response latency, and database query execution time under simulated concurrent loads using Apache JMeter; and (ii) user experience, using the System Usability Scale (SUS) questionnaire administered to 52 student participants after a 30-minute supervised usage session. Participants were drawn from four undergraduate courses spanning Computer Science, Business, and Humanities departments to ensure disciplinary diversity.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

V. IMPLEMENTATION

A. Technology Stack

Table I summarizes the technology stack employed in the implementation of the Study Hub Portal. The MERN stack was selected for its end-to-end JavaScript consistency, active open-source ecosystems, and demonstrated scalability in production deployments [14].

TABLE I. Technology Stack of the Study Hub Portal

Layer	Technology	Purpose
Frontend	React.js 18	Component-based SPA with Hooks
State Management	Redux Toolkit	Global application state
UI Components	Material-UI v5	Responsive design system
HTTP Client	Axios	REST API communication
Backend Runtime	Node.js 18 LTS	Server-side JavaScript execution
Web Framework	Express.js 4	REST API routing and middleware
Database	MongoDB Atlas	NoSQL document storage
ODM	Mongoose 7	Schema validation and queries
Authentication	JWT / bcrypt	Stateless auth and password hashing
Real-Time	Socket.IO	WebSocket-based live messaging
Caching	Redis	Session store and rate limiting
Containerization	Docker / Compose	Portable deployment

B. Frontend Implementation

The React.js SPA is structured into feature-based modules, each containing its own components, hooks, and Redux slices. The routing hierarchy is defined using React Router v6's nested routes, enabling deep-linking to specific study groups, discussion threads, or assignment details. Protected routes are implemented via a custom Private Route wrapper that checks JWT validity in the Redux auth slice before rendering the requested view.

The real-time messaging feature leverages Socket.IO, establishing a persistent WebSocket connection upon user authentication. Room-based name-spacing confines message broadcasts to authorized study group members, preventing cross-group data leakage. React Query is used for server-state synchronization, providing automatic background refetching, stale-while-revalidate caching, and optimistic UI updates for note uploads and discussion posts.

C. Backend Implementation

The Express.js server exposes approximately forty REST API endpoints organized into six routers: /api/auth, /api/users, /api/groups, /api/notes, /api/discussions, and /api/assignments. All endpoints are versioned under the /api/v1/ prefix. Input validation is enforced using Joi schemas in a dedicated validation middleware layer, returning standardized error objects conforming to RFC 7807 Problem Details for HTTP APIs.

Role-based access control is implemented at the middleware level. Each route specifies the minimum required role (guest, student, moderator, admin). The check-Role middleware extracts the role claim from the JWT payload and compares it against the required role. Resource-level authorization (e.g., verifying that a student is a member of the group they are attempting to access) is enforced within the respective service functions.

D. Database Implementation

MongoDB Atlas M10 cluster (3-node replica set) is used for production persistence, providing automatic failover and point-in-time recovery. The Notes collection stores document metadata (title, subject, tags, uploadedBy,

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

groupId) alongside a Grid-FS reference for binary file storage, supporting PDF, DOCX, and image attachments up to 25 MB. Full-text search across notes and discussion posts is enabled via Atlas Search (Lucene-based), allowing students to locate resources across all their enrolled groups with a single query [15].

VI. RESULTS AND DISCUSSION

A. Performance Evaluation

Performance benchmarking was conducted using Apache JMeter with virtual user (VU) loads of 10, 50, 100, and 200 concurrent users. Table II summarizes the API response time statistics for the five most frequently used endpoints.

TABLE II. API Response Time (ms) Under Concurrent Load

Endpoint	10 VU	50 VU	100 VU	200 VU
POST /api/v1/auth/login	45	112	198	287
GET /api/v1/groups/:id	38	95	164	241
GET /api/v1/notes?groupId	62	134	223	319
POST /api/v1/discussions	51	118	202	296
GET /api/v1/assignments	41	102	181	268

All primary endpoints maintained sub-300 ms response times even at 200 concurrent virtual users, meeting the industry-standard threshold for acceptable interactive web application latency [16]. The notes endpoint exhibited the highest latency due to MongoDB aggregation pipeline operations that join note metadata with group membership data. This can be further optimized with pre-aggregated materialized views in a future release.

B. Usability Evaluation

The System Usability Scale (SUS) questionnaire, comprising ten Likert-scale items, was administered to 52 student participants following a structured 30-minute usage session. The computed mean SUS score was 82.4 (SD = 7.3), which falls in the "Excellent" category on Bangor et al.'s adjective rating scale [17]. Subscale analysis revealed particularly high scores for learnability (mean = 86.1) and efficiency (mean = 83.7), while the consistency of terminology and icon meanings received slightly lower ratings (mean = 79.2), indicating an area for UI refinement.

Qualitative feedback collected through open-ended survey items highlighted the study group dashboard and real-time notification system as the most valued features. Students noted that the centralized resource repository significantly reduced the time spent searching for shared materials across disparate messaging applications. The most frequently requested enhancement was an integrated video conferencing feature, which is identified as a priority for the next development iteration.

VII. ADVANTAGES AND LIMITATIONS

A. Advantages

The Study Hub Portal offers several key advantages over existing tools. First, it provides domain-specific functionality tailored to academic collaboration, distinguishing it from general-purpose tools such as Slack or Microsoft Teams [7]. Second, the MERN stack architecture enables full JavaScript isomorphism, reducing context-switching overhead for development teams and facilitating code sharing between frontend and backend (e.g., shared validation schemas). Third, MongoDB's flexible document model accommodates the highly varied schema requirements of different academic subjects and collaboration styles without requiring costly schema migrations. Fourth, the real-time WebSocket layer ensures low-latency communication within study groups, approaching the immediacy of face-to-face interaction in virtual settings. Fifth, the containerized deployment model enables institutions to self-host the platform, addressing data sovereignty and FERPA compliance requirements that preclude the use of third-party cloud services.

B. Limitations

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Several limitations warrant acknowledgment. The current implementation does not include integrated video or audio conferencing, necessitating reliance on external tools for synchronous sessions. The full-text search capability, while functional, does not yet index the binary content of uploaded PDF files, limiting discoverability of materials shared as scanned documents. The study was conducted with students from a single institution, which may limit the generalizability of usability findings. Additionally, the system has not yet been evaluated under sustained production load over an extended period, and long-term performance characteristics under database growth remain to be empirically assessed.

VIII. FUTURE WORK

Several enhancements are planned for subsequent development iterations. An AI-powered recommendation engine, leveraging collaborative filtering techniques, will suggest relevant study groups, notes, and discussion threads based on a student's enrolled courses and interaction history [18]. Integration with institutional identity providers via SAML 2.0 or OAuth 2.0 will enable seamless single sign-on (SSO) authentication using existing university credentials, removing onboarding friction and improving adoption rates.

Automated PDF text extraction using Apache Tika or PDF.js will enhance full-text search to cover the content of uploaded documents. An intelligent group-formation module, informed by Hsiao and Brouns' [10] research on heterogeneous grouping, will assist instructors and students in assembling balanced study groups. A mobile application built with React Native will extend platform accessibility to students who primarily use smartphones. Finally, a learning analytics dashboard will provide both students and instructors with visualizations of collaboration patterns, resource utilization, and participation metrics, enabling data-informed pedagogical interventions.

IX. CONCLUSION

This paper has presented the design, architecture, and implementation of the Student Collaboration "Study Hub" Portal, a purpose-built web application that provides undergraduate students with a centralized, feature-rich environment for peer-to-peer academic collaboration. By integrating study group management, note sharing, threaded academic discussions, real-time messaging, and collaborative assignment tracking into a cohesive platform built on the MERN stack, the portal addresses a demonstrable gap in existing educational technology offerings.

Empirical evaluation demonstrated that the system achieves sub-300 ms API response times at 200 concurrent users and an SUS usability score of 82.4, confirming both its technical viability and user acceptance. The portal's modular, containerized architecture ensures scalability and maintainability, while its open development framework provides a solid foundation for the AI-enhanced, institutionally integrated features planned for future iterations.

As higher education continues its digital transformation, tools that genuinely support the informal, student-driven dimension of learning will become increasingly critical. The Study Hub Portal contributes a concrete, evaluable instantiation of such a tool, and the methodology and architecture documented herein offer transferable insights for researchers and practitioners developing next-generation educational technology systems.

REFERENCES

1) R. J. Marzano, D. J. Pickering, and J. E. Pollock, Classroom Instruction That Works: Research-Based Strategies for Increasing Student Achievement, 2nd ed. Alexandria, VA: ASCD, 2012.
2) K. J. Topping, "Trends in peer learning," Educational Psychology, vol. 25, no. 6, pp. 631–645, Dec. 2005.
3) L. S. Vygotsky, Mind in Society: The Development of Higher Psychological Processes. Cambridge, MA: Harvard University Press, 1978.
4) D. W. Johnson and R. T. Johnson, "Making cooperative learning work," Theory Into Practice, vol. 38, no. 2, pp. 67–73, 1999.
5) M. Dougiamas and P. C. Taylor, "Moodle: Using learning communities to create an open source course management system," in Proc. ED-MEDIA World Conf. Educational Multimedia, Hypermedia & Telecommunications, Honolulu, HI, 2003, pp. 171–178.
6) Instructure Inc., "Canvas LMS: Product Overview," Canvas by Instructure, Salt Lake City, UT, Tech. Rep., 2022. [Online]. Available: https://www.instructure.com/canvas

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

7) M. Basilaia and D. Kvavadze, "Transition to online education in schools during a SARS-CoV-2 coronavirus (COVID-19) pandemic in Georgia," Pedagogical Research, vol. 5, no. 4, pp. 1–9, Apr. 2020.
8) M. Wen, C. P. Rose, and C. Rosé, "Collaborative annotation for learning in online forums," in Proc. ACM Conf. Learning at Scale (L@S), Atlanta, GA, 2016, pp. 401–410.
9) C. G. Brinton, M. Chiang, S. Jain, H. Lam, Z. Liu, and F. M. F. Wong, "Learning about social learning in MOOCs: From statistical analysis to generative model," IEEE Transactions on Learning Technologies, vol. 7, no. 4, pp. 346–359, Oct.–Dec. 2014.
10) I. Hsiao and F. Brouns, "Intelligent grouping algorithms for collaborative learning," in Proc. IEEE Int. Conf. Advanced Learning Technologies (ICALT), Athens, Greece, 2011, pp. 324–328.
11) B. Hahn, Getting MEAN with Mongo, Express, Angular, and Node, 2nd ed. Shelter Island, NY: Manning Publications, 2019.
12) M. T. Goodrich, R. Tamassia, and M. H. Goldwasser, Data Structures and Algorithms in Java, 6th ed. Hoboken, NJ: Wiley, 2014.
13) W. Farcic, "The DevOps 2.0 Toolkit: Automating the Continuous Deployment Pipeline with Containerized Microservices," Birmingham, UK: Packt Publishing, 2016.
14) B. Holmes and J. Novak, "Full stack development with React and Node," in Proc. IEEE Frontiers in Education Conf. (FIE), San Jose, CA, 2018, pp. 1–9.
15) MongoDB, Inc., "Atlas Search: Full-Text Search on MongoDB," Technical Documentation, New York, NY, 2023. [Online]. Available: https://www.mongodb.com/atlas/search
16) J. Nielsen, Usability Engineering. San Francisco, CA: Morgan Kaufmann, 1993.
17) A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the System Usability Scale," International Journal of Human-Computer Interaction, vol. 24, no. 6, pp. 574–594, Jul. 2008.
18) F. Ricci, L. Rokach, and B. Shapira, Eds., Recommender Systems Handbook, 2nd ed. New York, NY: Springer, 2015.