IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154205

ScholarGrid: A Full-Stack Academic Collaboration Platform with Community-Driven Quality Layers

Ranvir Wadhawan[1], Swaroop Mandal[2], Aryan Yadav[3], and Meet Patil[4], Prof. K.L. More[5]

Department of Computer Engineering, A. C. Patil College of Engineering, Navi Mumbai, India[1-4]

Mentor, Department of Computer Engineering, A. C. Patil College of Engineering, Navi Mumbai, India[5]

Abstract: Academic collaboration in higher education is frequently hindered by fragmented resources across unorganized platforms. This paper introduces ScholarGrid, a centralized full-stack solution designed to streamline note-sharing and academic communication. Built using React 19 and FastAPI, the platform implements a community-driven quality layer featuring star ratings and peer reviews to ensure content reliability. Additionally, the integration of Socket.io enables real-time, subject-specific chat channels. Our methodology employs an iterative development model, resulting in a scalable architecture that bridges the gap between informal messaging and structured learning management systems.

Keywords: Note-Sharing Platform, Full-Stack Development, Academic Collaboration, React 19, FastAPI, Peer-Review Systems.

I. INTRODUCTION

Modern engineering education relies heavily on digital resources; however, students face significant "resource fragmentation" where materials are scattered across WhatsApp, Telegram, and unorganized Drive folders. This leads to substantial time loss as students hunt for relevant documents. Furthermore, the lack of a "Quality Filter" means students cannot easily judge the accuracy of a note before downloading. ScholarGrid addresses these challenges by providing a centralized, role-based repository designed specifically for the academic ecosystem.

II. RELATED WORK & GAP ANALYSIS

In recent years, digital platforms have significantly transformed the way students share academic resources and collaborate. Various tools such as messaging applications, cloud storage systems, and learning management platforms have been widely adopted in academic environments. However, despite their popularity, these platforms exhibit several limitations when evaluated in the context of structured academic resource sharing and collaborative learning.

Applications like WhatsApp and Telegram are extensively used by students for quick dissemination of notes, assignments, and important updates. Their primary advantage lies in real-time communication and ease of use. However, these platforms lack any form of structured organization. Academic materials are often shared in unorganized chat threads, making retrieval difficult and inefficient. Furthermore, there is no mechanism for validating the quality or accuracy of shared content, which can lead to the circulation of incorrect or low-quality study materials.

Cloud-based storage solutions such as Google Drive provide a more organized approach to storing and accessing academic files. They allow users to categorize and manage documents systematically using folders and sharing permissions. Despite these advantages, such platforms lack social and collaborative features. There is minimal support for peer interaction, feedback, or evaluation of resources. As a result, students are unable to assess the usefulness or credibility of shared materials based on collective input.

Learning management systems like Google Classroom attempt to bridge some of these gaps by offering structured course-based organization and teacher-student interaction. However, these systems are primarily instructor-driven and do not emphasize peer-to-peer knowledge sharing. The absence of community-driven feedback mechanisms and rating systems limits their effectiveness in promoting collaborative learning beyond the classroom hierarchy.

To address these limitations, the proposed system, ScholarGrid, introduces a unified platform that combines structured organization with social interaction. It incorporates subject-wise taxonomy for efficient categorization of academic

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154205

resources, enabling users to easily navigate and retrieve relevant materials. Additionally, the platform integrates peer-review mechanisms such as star ratings and feedback systems, allowing students to evaluate the quality and relevance of shared content. This ensures that high-quality resources are easily identifiable and widely accessible.

Thus, while existing solutions partially address the needs of academic content sharing, they fail to provide an integrated environment that supports both structured organization and collaborative evaluation. ScholarGrid aims to fill this gap by offering a comprehensive, student-centric platform that enhances both accessibility and quality of shared academic resources.

III. PROPOSED SYSTEM ARCHITECTURE

The proposed system adopts a decoupled full-stack architecture designed to achieve high performance, scalability, and maintainability. By separating the frontend, backend, and communication layers, the system ensures flexibility in development and efficient handling of user interactions and data processing. The architecture is structured into multiple layers, each responsible for specific functionalities, enabling modular design and easy future enhancements.

A. Frontend Layer

The frontend layer is developed using React 19 in combination with Vite, providing a high-performance and responsive user interface. React's component-based architecture enables the creation of reusable and modular UI elements such as subject cards, user profiles, and rating badges. Vite serves as a fast build tool and development server, significantly improving development speed and optimizing production builds.

The user interface is designed with a dark-theme palette to enhance visual comfort and modern aesthetics. Efficient state management techniques ensure seamless navigation and real-time updates across different screens. The frontend communicates with the backend through RESTful APIs and WebSocket connections, ensuring smooth data exchange and dynamic content rendering.

B. Backend Layer

The backend layer is implemented using FastAPI, a modern Python-based framework known for its high performance and asynchronous request handling capabilities. FastAPI enables the system to efficiently manage multiple concurrent requests, making it suitable for real-time and data-intensive applications.

MySQL is used as the primary database due to its reliability, scalability, and strong support for relational data integrity. It ensures structured storage of user data, academic resources, subject categories, and peer feedback. The backend also handles authentication, authorization, and data validation to maintain system security and consistency.

C. Real-Time Communication Layer

To facilitate instant communication and collaborative learning, the system integrates real-time messaging using Socket.io. This layer enables users to interact within dedicated subject-specific channels, ensuring that discussions remain organized and contextually relevant.

Unlike traditional messaging platforms where important discussions can get lost in continuous message streams, the subject-based channel structure ensures that conversations are categorized and easily retrievable. Real-time updates allow users to receive instant notifications for new messages, feedback, and content uploads, thereby enhancing engagement and collaboration.

D. Quality Verification Layer

A key feature of the proposed system is the integration of a quality verification mechanism to ensure the reliability of shared academic resources. This layer includes a 1–5 star rating system and threaded comment functionality, enabling users to evaluate and provide feedback on uploaded content.

The rating system helps in highlighting high-quality resources, while the threaded comments allow for detailed discussions and clarifications. This peer-driven moderation approach ensures that content is continuously reviewed and improved by the user community. Over time, this mechanism promotes the visibility of valuable resources and discourages the sharing of low-quality or irrelevant materials.

E. System Integration and Data Flow

All layers are interconnected through well-defined APIs and communication protocols. The frontend sends user requests to the backend, which processes the data, interacts with the database, and returns appropriate responses. For real-time interactions, WebSocket connections are established to enable instant data exchange without repeated HTTP requests.

This layered architecture ensures efficient data flow, reduced latency, and high system responsiveness. Additionally, the modular design allows independent scaling of different layers based on system load, making the platform suitable for large-scale deployment in academic environments.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154205

IV. METHODOLOGY

The development of the proposed system follows an iterative software development model, specifically the Build–Test–Iterate approach. This methodology was chosen to ensure continuous improvement, rapid prototyping, and effective incorporation of user feedback throughout the development lifecycle. The iterative nature of the model enables the identification and resolution of design and functional issues at early stages, thereby enhancing overall system reliability and user experience.

The process commenced with requirement analysis and system planning. During this phase, user needs were identified, and detailed user flows were designed to map interactions within the application. Special emphasis was placed on usability and accessibility, leading to the adoption of a dark-theme user interface. The dark theme not only enhances visual ergonomics but also aligns with modern UI/UX design trends, ensuring a more engaging and comfortable user experience during prolonged usage.

Following the planning phase, the system design focused on modular and reusable component development. Key interface elements such as user avatars, subject cards, and rating badges were designed as independent and reusable components. This modular approach improves code maintainability, scalability, and consistency across different sections of the application. Component reusability also reduces development time and ensures uniform behavior and appearance throughout the system.

The implementation phase involved the development of the front-end architecture using a component-based framework. State management was handled using a lifted state approach, where shared states were maintained at higher-level components and passed down to child components via props and callback functions. This ensured efficient cross-screen synchronization and seamless data flow between different parts of the application. React callbacks were extensively used to handle user interactions, update state dynamically, and maintain consistency in real-time.

Subsequently, the integration phase combined all individual components into a cohesive system. Functional modules such as content sharing, subject-wise categorization, and peer-review mechanisms were integrated and tested collectively. Special attention was given to ensuring that updates in one part of the application were accurately reflected across all relevant screens.

Testing was conducted at multiple levels, including unit testing for individual components and integration testing for complete workflows. Each iteration involved identifying bugs, optimizing performance, and refining the user interface based on observed issues and feedback. This continuous testing cycle ensured the stability and efficiency of the system. Overall, the adopted methodology facilitated a structured yet flexible development process, enabling the creation of a scalable, user-friendly, and efficient platform for academic resource sharing and peer collaboration.

IV. CONCLUSION AND FUTURE SCOPE

ScholarGrid provides a robust framework for academic exchange by combining the strengths of storage, communication, and peer reviews. Future development will focus on:

- **AI Integration**: Utilizing the Google Gemini API for automated document summarization.
- **Security**: Implementing ClamAV for automated file virus scanning to ensure platform safety.
- **Accessibility**: Support for Progressive Web Apps (PWA) to enable offline access to study materials.

REFERENCES

[1] Supabase Inc. (2024). Supabase Documentation: Authentication, Database, Storage, and Realtime. Retrieved April 2026 from https://supabase.com/docs
[2] React Core Team, Meta Open Source. (2024). React 19 Documentation. Retrieved April 2026 from https://react.dev
[3] PostgreSQL Global Development Group. (2024). PostgreSQL 15 Documentation. Retrieved April 2026 from https://www.postgresql.org/docs/15/
[4] Vitejs. (2024). Vite 8 Documentation. Retrieved April 2026 from https://vitejs.dev
[5] Tailwind CSS Documentation. (2024). Tailwind CSS v3 — Utility-First CSS Framework. Retrieved April 2026 from https://tailwindcss.com/docs
[6] Framer Motion. (2024). Framer Motion 12 API Reference. Retrieved April 2026 from https://www.framer.com/motion/

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154205

[7] Lucide. (2024). Lucide React — Icon Library Documentation. Retrieved April 2026 from https://lucide.dev
[8] Topping, K. J. (2009). Peer Assessment. Theory Into Practice, 48(1), 20–27. https://doi.org/10.1080/00405840802577569
[9] Falchikov, N., & Goldfinch, J. (2000). Student Peer Assessment in Higher Education: A Meta-Analysis. Review of Educational Research, 70(3), 287–322. https://doi.org/10.3102/00346543070003287
[10] Hamari, J., Koivisto, J., & Sarsa, H. (2014). Does Gamification Work? — A Literature Review of Empirical Studies on Gamification. 47th Hawaii International Conference on System Sciences (HICSS), pp. 3025–3034. https://doi.org/10.1109/HICSS.2014.377
[11] Denny, P., McDonald, F., Margulieux, L., & Kelly, M. (2018). Reducing the Effect of Confusion in Online Discussion Forums. Proceedings of the 2018 ACM Conference on International Computing Education Research
[12] Garrison, D. R., Anderson, T., & Archer, W. (2000). Critical Inquiry in a Text-Based Environment: Computer Conferencing in Higher Education. The Internet and Higher Education, 2(2-3), 87–105.
[13] Ware, M., & Mabe, M. (2015). The STM Report: An Overview of Scientific and Scholarly Journal Publishing (4th ed.). International Association of Scientific, Technical and Medical Publishers.
[14] Pressman, R. S., & Maxim, B. R. (2020). Software Engineering: A Practitioner's Approach (9th ed.). McGraw-Hill Education.
[15] Elmasri, R., & Navathe, S. B. (2016). Fundamentals of Database Systems (7th ed.). Pearson Education.
[16] Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). Database System Concepts (7th ed.). McGraw-Hill Education.
[17] OWASP Foundation. (2024). OWASP Top Ten Web Application Security Risks. Retrieved from https://owasp.org/www-project-top-ten/
[18] W3C Web Accessibility Initiative. (2023). Web Content Accessibility Guidelines (WCAG) 2.1. https://www.w3.org/TR/WCAG21/
[19] Mozilla Developer Network. (2024). WebSockets API. Retrieved April 2026 from https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
[20] react-router-dom v7. (2024). React Router Documentation. Retrieved April 2026 from https://reactrouter.com/en/main
[21] M. Gupta, "Digital Collaboration in Higher Education," Journal of Educational Technology, vol. 12, no. 3, pp. 45-50, 2024.
[22] FastAPI Documentation, "High performance, easy to learn, fast to code," [Online]. Available: https://fastapi.tiangolo.com/. [Accessed: April 14, 2026].
[23] React Documentation, "The library for web and native user interfaces," [Online]. Available: https://react.dev/. [Accessed: April 14, 2026].
[24] T. Sinha and A. Banka, "Leveraging user profile attributes for improving pedagogical accuracy of learning pathways," Vellore Institute of Technology, 2014. [Online]. Available: https://arxiv.org/abs/1407.7260.

