iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.81452

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

LuminaCode: A Review of a SaaS Platform for Collaborative Code Editing and Sharing

Tejaswani Singh[1], Er. Harish Shukla[2]
Computer Science & Engineering, Shri Ramswaroop Memorial College of Engineering and Mangement , Lucknow, India

Abstract: ***Software development evolution has underscored the growing relevance of collaborative programming environments in real time to facilitate accelerated teamwork, productivity, and learning. This paper consolidates and synthesizes key findings of some research. They utilize technologies such as ReactJS, Node.js, Docker, Socket.IO, and Eclipse plug-ins in providing seamless synchronization of the code base, live communications, and multi-user programming experiences. Distributed teams are also catered for with real-time collaboration with the inclusion of features like dynamic software visualization, context-based operation merging (COMA), and shared code cities for improved program understanding and editing cohesion. Educational platforms also expand their features with the inclusion of AI-powered learning support and collaborative awareness features for students. The collective analysis provides insight into how the approaches eliminate geographical and technical boundaries to construct intelligent, web-centric, and interactive coding ecosystems that transform software development and programming education.***

Keywords: ***Real-time collaboration, Collaborative programming, Software visualization, Program comprehension, Code synchronization, Context-based operation merging, Eclipse plug-in, ReactJS, Node.js, Docker, Socket.IO, AI chatbot, DSA learning platform, Education technology, Distributed systems, Awareness, Synchronization, Open-source development, User interface, Knowledge transfer, Teamwork, Learning enhancement.***

I. INTRODUCTION

The accelerated growth of technology has dramatically impacted the schooling sector in the creation of collaborative, intelligent, and cloud-infused learning spaces. The implementation of real-time communications, remote lab activities, and artificial intelligence-augmented systems has transformed the mode in which students and builders collaborate, interact, and create knowledge within digital platforms.

These earlier collaborative innovations involved supporting distributed editing and knowledge sharing with the assistance of low-bandwidth and high-latency systems such as the Jupiter collaboration system [38] and fundamental work in operational transformation (OT) for the maintenance of the environment of real-time collaboration being consistent. Later research then extended the frameworks to encompass handling semantic consistency and synchronization in multi-user settings [34].

Due to the necessity for collective programming and real-time collaborative work in the software development area, software such as CollabEd [22], COLLEGE-2.0 [35] and CodeFlow [42] having features like synchronized code editing, debugging, and role-based interfaces [20] came into existence. Other innovations such as CodeQuest and Denicek [44] enabled collaborative code learning in interactive, document-oriented programming styles. Research [32] showed how data visualization and user analytics may enhance initiative and performance in such scenarios.

The education sector also adopted collaborative learning with the help of cloud technologies that facilitate collaborative learning experiences. Studies by Al-Samarraie & Saeed (2018) and Kumar & Bhardwaj (2020) studied the implementation of cloud computing in blended and schooling education, with [36] underscoring collaborative learning's social and cognitive implications. Some showed large-scale participation patterns in massive open online courses (MOOCs), also testifying to the scalability of collaborative schools. Non-prescribed learning modalities with the aid of the Internet and social media and systematic reviews regarding cloud tools the flexibility and inclusivity of contemporary learning ecologies.

Simultaneously emerging remote laboratory infrastructure and virtual laboratory platform technologies [3] [37] have closed the distance between theoretical and practical learning by providing real-time instrumental and simulation access. They have especially become highly useful in electronics and IoT training [26] [33] [41], developing technical skills through interactive learning.

The blending of chatbots and AI has gone one step further in realizing the potential for personalized and adaptive learning. Tools like NajahniBot [5], Digital Professor and other conversational learning chatbots [24] [28] portend the future of contextually intelligent conversational learning support that can shepherd students through personalized interactions. Usability research [24] and systematic reviews indicate the expansion in maturity and legitimacy of chatbot tech in education.

From an engineering and computational perspective, programming assistance based on AI has attracted attention, with research examining the possibilities of machine learning being used for smart generation and correction of code [25]. In the same vein, collaborative and browser-based coding platforms like Replit [21] and CodePen [34] have made it easier for novices and professional programmers. All in all, the intersection between real-time collaboration software, cloud technologies, AI-powered customization, and remote experimentation has reimagined the contours of the contemporary education and software development space. As aggregated in these papers [36] [44] [31], the new generation of programming and learning platforms requires emphasis on inclusivity, interactivity, and flexibility such that the digital innovation space remains centered on collaboration.

II. LITERATURE REVIEW

A. Evolution of Collaborative Coding Systems

The development of collaborative software environments has gone through some technological turning points. Initial work (1995-2004) like the Jupiter collaborative system and the development of Operational Transformation (OT) algorithms provided the basic principles behind concurrent editing and real-time synchronization in the presence of high latency [35], [40]. They established the trade-off between responsiveness and consistency which remains the kernel of contemporary collaborative systems. They excel in formal algorithmic design but operated almost entirely in the domain of plain-text documents rather than semantically enriched code.

2003–2009 saw decentralized, peer-to-peer systems such as CollabEd and COE with provisions for positive replication and editing of the ontology. These allowed for offline assistance and extensibility but did not scale to global consistency in large projects [14] [3].

2010–2015 saw the emergence of online IDEs (Collabode, Repl it, CodePen) with real-time compilation, error tracking, and remote debugging, moving the emphasis towards code-aware platforms [10] [15] [22]. Usability increased but scalability and commercial potential remained restricted.

2015–2023 focused on network performance, efficiency, and visualization. Such techniques as COMA and code-proximal visualization facilitated coordination and understanding, though domain-specific sophistication remained [7] [23] [37].

Since 2019, the cloud-native, SaaS programs (CodeFlow, CodeQuest) merge collaborative editing in real time with chat and adaptive feedback, framing education and professional coding environments [4] [16] [32] [33].

B. Thematic Contributions

1) *Consistency and Synchronization Models:* Semantic models and OT continue to be cornerstone, dictating hybrid synchronization in contemporary instruments [35] [36].
2) *IDEs in the Browser & Collaborative Programming: Browser IDEs did demonstrate real-time-code aware collaborative programming but didn't scale economically [10] [14] [15].*
3) *Operation Merging & Network Resilience:* COMA and low-bandwidth strategies improve efficiency for distributed users [7] [40].
4) *Visualization & Analytics:* Integrated visualization combinations with collaborative logs facilitate better learning and knowledge measures [8] [23] [37].
5) *Learning Platforms and Educational Collaboration:* Sites with the combination of peer-to-peer collaboration and awareness mechanisms demonstrate educative potential [1] [2] [24] [18].
6) *Integration of AI and Chat Support:* Chatbot-enabled adaptive learning [4] [16] [29] [39] suggests intelligent chat agents may reduce cognitive burden and maximize learner investment. Concerns with bias, plagiarism, and explainability persist, however.
7) *Architecting SaaS and Monetization:* New platforms embrace subscription-oriented scalable solutions with cloud orchestration and payment APIs [32] [33].

C. Patterns and Insights

During the decades of research, four trends dominate the history of the development:

1) From text to semantic code understanding, moving collaboration from vanilla editing to syntactic and runtime-aware operations [10] [15] [23].

2) From centralized to hybrid synchronization, blending OT/CRDT reliability with compression and operation-merging for scale [7] [35].
3) From visualization as an add-on to visualization as an embedded context enhancing understanding and awareness [23].
4) From ground-level software to SaaS platforms with the infusion of collaboration, AI, analytics, and monetization in single platforms [32] [33].
5) These patterns together also substantiate the combined design of LuminaCode, bringing together real-time editing, visualization, community, and AI under a viable SaaS architecture.

D. Critical Appraisal & Significance

Previous work is solid in theory, prototypes, and experimentation but scattered, with little combination of collaboration, AI, analytics, and monetization at scale.

III. APPLICATIONS OF COLLABORATIVE CODE EDITOR

1) *Real-Time Code Collaboration*: Users edit the code at the same time with real-time error and conflict feedback. Platforms: Replit, LuminaCode, Collabode.
2) *Education and Learning:* It provides for peer mentoring, adaptive feedbacking, and gamified learning spaces. These also track students' progress and give prompts or explanations of mistakes.
3) *Enterprise Development:* Distributed teams use cloud IDEs with chat, versioning, and task management in one.
4) *Visualization and Analytics:* Dashboards monitor collaboration metrics, runtime behavior, and learning achievements. Embedded visualizations improve program comprehension and decision-making.
5) *AI-Assisted Programming:* Intelligent agents also provide recommendations for code completion, error detection, and automated guidance.

Algorithm 1. Operational Transformer (OT):

It ensures consistency among distributed replicas by transforming con-current operations.

Input:

- *O_local* (local operation).
- *O_remote* (remote operation).
- *D* (document state).

Output:

D' (consistent document state).

Steps:

- Recieves *O_remote*.
- If *O_local* & *O_remote* are con-current, then proceed to transform.
- Compute the Transform Operation:

  $$O'_{remote} = T(O_{remote}, O_{local})$$

  Where *T* is the transformation function which ensures intension preservation.
- Now apply the transformed operation:

  $$D' = Apply(D, O'_{remote})$$

- Update the document by merging *O_local* and *O'_{remote}* into operation history ***H***.
- Lastly broadcast the updated document to all replicas.

Algorithm 2. Conflict-free Replicated Data Type (CRDT):

It automatically converges all replicas without requiring OT.

Input:

- *Ri* (replica identifier).
- *u* (local update).
- *U_remote* (remote updates from other replicas).

Output:

S_final (converged state).

Steps:

- Initializing the CRDT object:
  $S_0 = initial\ state$
- Applying update *u* using merge function:
  $S_i = Merge(S_i, u)$
- To receive remote updates for each *U_remote* from other replicas, we need to apply:
  $S_i = Merge(S_i, U_{remote})$
- The merge function guarantees to ensure commutativity, associativity and idempotence:
  $Merge(S_a, S_b) = Merge(S_b, S_a)$

$Merge\ (Merge\ (S_a, S_b), S_c) = Merge\ (S_a, Merge\ (S_b, S_c))$

- Lastly convergence is done once all the updates are applied:

$$S_1 = S_2 = \ldots = S_n = S_{final}$$

IV. ADVANTAGES OF OF COLLABORATIVE CODE EDITOR

1) Collaborative code-editor provides real-time editing.
2) This reduces the development time and duplication.
3) Semantic code-awareness is also possible where errors are detected and syntactic correctness is enforced.
4) Integration with AI provides adaptive learning, coding recommendations, and feedback.
5) Cloud & SaaS platforms are seamless, scalable and accessible across.
6) Visualization & Analytics allows user awareness and progress tracking.
7) Comprehension Peer-to-peer and offline support manages productivity within network limitations.

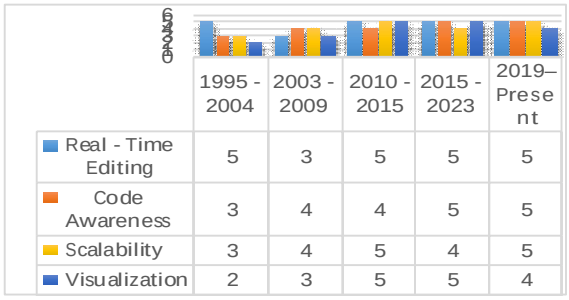
V. CHALLENGES AND CONSIDERATIONS

1) Consistency and managing OT/CRDT-based synchronization across nodes.
2) Network efficiency, optimizing bandwidth and operation merging in real-time collaboration.
3) Scalability and managing big, widespread teams without performance loss.
4) Security and privacy concerns i.e. securing code, data, outputs of the AI, and intellectual property.
5) Integrating by combining Artificial Intelligence, visualization, cloud infrastructure, and Monet.
6) User Experience where more natural interfaces with less cognitive overhead.
7) AI-Ethics and managing automation over-reliance, plagiarism, and bias.

VI. COMPARISON GRAPH OF PLATFORMS

Representation: 0 → Poor and 5 → Excellent

VII. SUMMARY



	1995 - 2004	2003 - 2009	2010 - 2015	2015 - 2023	2019– Present
Real - Time Editing	5	3	5	5	5
Code Awareness	3	4	4	5	5
Scalability	3	4	5	4	5
Visualization	2	3	5	5	4

Era	Summary of key developments		
	Key Innovations	*Strengths*	*limitations*
1995 - 2004	OT algorithm, Jupiter	Concurrency & responsiveness	Text focused, limited functionality
2003 - 2009	Peer-to-peer frameworks	Offline support, extensibility	Global consistency issues
2010 - 2015	Web IDEs, compilation-aware editing	Usability, code aware collaboration	Limited scalability, monetization gaps
2015 - 2023	COMA, network optimization, visualizations	Efficient networks, improved comprehension	Domain specific, complex integration
2019 - Present	AI-assisted SaaS platforms	Unified collaboration, analytics, monetization	Security, AI bias, explainability

VIII. EMERGING TRENDS

1) *Text→Semantic Code Understanding:*
   The Collaboration now provides syntactic and runtime awareness.
2) *Centralized→Hybrid Synchronization:*
   OT/CRDT with merge of operations reduces latency and maximizes correctness.
3) *Visualization Integrated in IDEs:*
   Helps in awareness, learning, and debugging.
4) *Local Tools→SaaS Ecosystems:*
   Integration of AI, analytics, and monetization.
5) *AI augmented Learning & Development:*
   Allows for lowered cognitive burden, enables adaptive guidance and feedback.

IX. FUTURE SCOPE

The future of real-time collaborative code environments, especially represented by LuminaCode, is very promising and has enormous potential for spurring innovation and making a notable large-scale impact upon the industry. Specifically, developments representing the area of artificial intelligence have the potential for further refining the level of assistance that is afforded coders by integrating features such as intelligent code completion, automated debugging protocols, and context-sensitive suggestions that are uniquely designed to conform to the individual coder's distinct style of coding. Moreover, the infusion of natural language interfaces will represent new frontiers that will enable voice-controlled manipulation of the code and will represent a type of conversational programming that will benefit the overall process of collaboration by rendering it far easier for all participants involved.

Moreover, by proliferating the scalability for these systems by integrating the use of distributed cloud infrastructure and efficient container orchestration methods, such as those represented by the name of Kubernetes, we may witness a noted expansion in the levels of performance and resilience when subjected to the conditions of a high workload involving several concurrent users functioning all at once.

In the end, the future version of LuminaCode will transition from its present role of simply being a vehicle for the dissemination of code. It will become a wide-based ecosystem for the purpose of facilitating collaborative innovation. The new version will integrate the current frontier AI technologies together with state-of-the-art visualization tools and the strength of powerful distributed computation. It will seek to dramatically alter the way by which developers, educators, and students collaborate to create and share knowledge in real time and enhance their overall experience and involvement in the learning process.

X. CONCLUSION

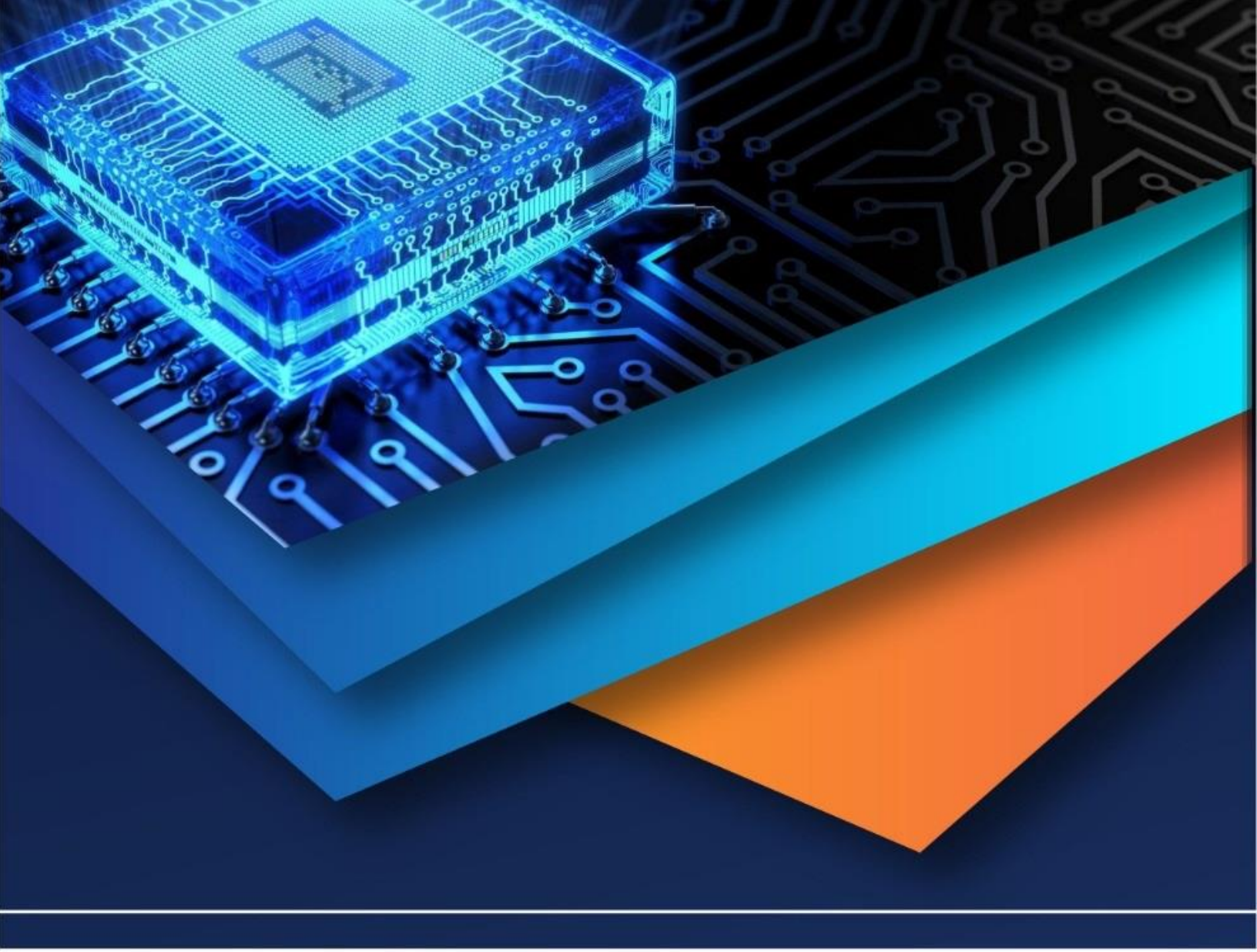
The history of collaborative programming environments has truly revolutionized the manner in which software is created, learned, and eventually propagated across dispersed groups that are positioned in diverse geographical regions. This extensive work embarked on a rigorous survey of key developments in this area from the early operational transformation models and peer-to-peer configurations to the current cloud-native and AI-infused platforms that together have served a key role in informing the underlying design principles of LuminaCode. The careful analysis showed that while previous work has achieved great and notable advances in aspects of synchronization, visualization, and building collaborative awareness, current systems continued to be disjointed and often existed in the form of mere academic prototype or single-purpose tools.

LuminaCode fills these gaps by consolidating real-time code editing, AI-enabled interaction, visualization, authentication, and billing within a single SaaS package. It utilizes state-of-the-art web technology and hybrid synchronization methods to boost performance, scalability, and end-user interaction. The system thereby closes the gap between theoretical research and real implementation and advances the software engineering and educational technology fields. Overall, LuminaCode marks a significant leap towards the fulfillment of a future collaborative ecosystem. It is a novel platform that will not only enable co-editing between users but also offer intelligent help that is bespoke for their requirements. It also values ethical scalability so that its functionalities are accessible to developers and students from every nook and cranny of the planet.

REFERENCES

[1] Aung, L., et al.,(2024). “an implementation of web-based answer platform in the flutter programming learning assistant system using docker compose,” electronics, Vol. 13, no. 24, pp. 4878.
[2] Bagheri, M., and Movahed, S. H., (2016). “the effect of the IoT on education business model,” in proc. 12th Int. Conf. Signal-Image Technology & Internet-Based Systems (SITIS), pp. 435 - 441.
[3] Bagnasco, A., et al., (2008). “a learning resources centre for simulation & remote experimentation in electronics,” in proc. 1st Int. Conf. Pervasive Technologies Related to Assistive Environments, pp. 1 - 7.
[4] Bani-Salameh, H., Jeffery, C., Al-Sharif, Z., and Abu Doush, I., (2008). “integrating collaborative program development and debugging within a virtual environment,” in 14th collaboration researchers’ Int. workshop on Groupware, vol. 5411, pp. 107 - 120.
[5] Bayounes, W., Saadi, I. B., and Hamroun, L., (2023). “NajahniBot: an intelligent chatbot aware of educational context for adaptive-learning,” in int. conf. innovations in intelligent systems & applications (INISTA), Hammamet, Tunisia, pp. 1 - 5.
[6] Bergstrom, R., and Bernstein, M., (2022). “glitch: an online platform for collaborative webD,” J. internet technologies, vol. 14, no. 4, pp. 75 - 88.
[7] Brinckman, A., et al., (2019). “collaborative circuit designs using the craft repository,” : future generation computer systems, vol. 94, pp. 841 - 853.
[8] Brown, M. G., (2016).“blended instructional practice: a review of the empirical literature on instructors adoption & use of online tools in face-to-face teaching,” internet higher education, vol. 31, pp. 1 - 10.
[9] Buitrago, P. A., et al., (2020). “mobile arduino robot programming using a remote laboratory in UNAD: pedagogic & technical aspects,” in International Conference of Remote Engineering & Virtual Instrumentation, pp. 171 - 183.
[10] Christopher, R. P., & Cormack, G. V., (1998). “operation transform for distributed shared readsheet,” in proc. ACM Conf. Computer Supported Cooperative Work (CSCW '98), pp. 69 - 78.
[11] Doernhoefer, M., (1999). “surfing the net for software engineering notes,” ACM SIGSOFT s/w engineering notes, vol. 24, no. 3, pp. 15 - 24.
[12] El-Medany, W. M., (2008). “FPGA remote laboratory for hardware e-learning courses,” in IEEE Region 8 Int. Conf. Computational Technologies in Electrical and Electronics Engineering, pp. 106 - 109.
[13] El-Medany, W. M., and Ismail, Y., (2013). “mobile learning laboratory for h/w courses,”. in 4th Int. Conf. E-Learning Best Practices in Management, Design & Development of E-Courses, pp. 51 - 54.
[14] Ericsson, K. A., Krampe, R. T., and Tesch-Romer, C., (1993). “ the role of deliberate practice in the acquisition of expert performance,”:- psychological review, vol. 100, no. 3, pp. 363 - 406.
[15] Ellis, C. A., and Gibbs, S. J., (1989). “concurrency control in groupware systems,” in proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 399 - 407.
[16] Fan, H., Sun, C., and Shen, H., (2012). “ATCoPE: any-time collaborative programming environment for seamless integration of real-time & non-real-time teamwork in software development,” in 17th ACM Int. Conf. Supporting Group Work, pp. 107 - 116.
[17] Fourie, C. J., (2020). “electronic design automation tools for super-conducting circuits,” J. Phys.: Conf. Ser., Vol. 1590, p. 012040.
[18] Frößler, F., (2008). “a practice theoretical analysis of real-time collaboration technology: skype & sametime in s/w development projects”. Göttingen: Cuvillier.
[19] Goldman, M., (2011). “role based interfaces for collaborative software development,” in proc. 24th Annual ACM Symp. Adjunct on UI S/w & Technology, p. 23.

[20] Goldman, M., Little, G., and Miller, R. C., (2011). " real time collaborative coding in a web IDE,":- in proc. 24th Annual ACM Symp. on UI s/w & Technology, pp. 151 - 164.
[21] Gonzalez, M., and Hill, (2019). "replit: facilitating collaborative coding & learning," Advances in Computer Science, vol. 10, no. 1, pp. 22 - 40.
[22] Granville, K. G., and Hickey, T. J., (2009), "collabEd: a platform for collaboratizing existing editors," in Int. Conf. Mobile, Hybrid & On-line Learning, pp. 90 - 96.
[23] Gusev, M., Ristov, S., and Armenski, G., (2013), "adaptive interactive online course on computer architecture and organization," in 12th Int. Conf. Information Technology Based Higher Education & Training (ITHET), pp. 1 - 8.
[24] Hidayat, A., Nugroho, A., and Nurfa'izin, S., (2022). "usability evaluation on educational chatbot using the system usability scale (SUS)," in 7th Int. Conf. Informatics and Computing (ICIC), pp. 1 - 5.
[25] Huang, L., Zhang, H., Li, R., Ge, Y., and Wang, J., (2020). "AI coding: learning to construct error correction codes," IEEE Trans. Commun., vol. 68, no. 1, pp. 26 - 39.
[26] Hussein, A., et al., (2019). "crowdsourced peer learning activity for IoT education- a case study," IEEE Internet Things Mag., vol. 2, pp. 26 - 31.
[27] Jaldén, J., Moreno, X. C., and Skog, I., (2018). "using the arduino due for teaching digital signal processing," in ICASSP 2018 IEEE Int. Conf. Acoustics, Speech & Signal Processing, pp. 6468 - 6472.
[28] Jain, V., Singh, I., Syed, M., Mondal, S., and Palai, D. R., (2014). "enhancing educational interactions- a comprehensive review of AI chatbots in learning environments," in 11th Int. Conf. Reliability, Infocom Technologies & Optimization (ICRITO), pp. 1 - 5.
[29] Kats, L. C. L., et al., (2012). "s/w development environments on the web," in proc. ACM Int. Symp. New Ideas, paradigms & reflections on programming & s/w - onward '12, p. 99.
[30] Klein, S., Vehring, N., and Kramer, M., (2010). "introducing real time communication: frames, modes & rules," in proc. 23rd Bled eConference eTrust: implications for the individual, pp. 591 - 606.
[31] Krause-Glau, A., and Hasselbring, W., (2023). "collaborative, code proximal dynamic software visualization within code editors," in IEEE Working Conf. Software Visualization (VISSOFT), pp. 50 - 63.
[32] Kumar, N. A., and Chandra, A. J., (2019). "development of remote instrumentation & control for laboratory experiments using smart phone application," in Int. Conf. Remote Engineering & Virtual Instrumentation, pp. 465 - 476.
[33] Kumar, V., and Bhardwaj, A., (2020). " the role of cloud computing in: school education,":-- in handbook of research on diverse teaching strategies for the technology rich classroom, Pp. 98 - 108, IGI Global.
[34] Kusuma, P., and Luxton-Reilly, A., (2021). "codePen: an online environment for frontend development," Int. J. Web Development, vol. 8, no. 2, pp. 15 - 30.
[35] Lacave, C., García, et al., (2019). "COLLECE-2.0: a real time collaborative programming system on eclipse," in Int. Symp. Computers in Education (SIIE), pp. 1 - 6.
[36] Laal, M., and Ghodsi, S. M., (2012) "benefits of collaborative learning," procedia social & behavioral sciences, vol. 31, pp. 486 - 490.
[37] Mayoo, C. A., et al., (2020). " FPGA remote laboratory: experience in UPNA & UNIFESP,":- in Int. Conf. Remote Engineering & Virtual Instrumentation, pp. 112 - 127.
[38] Nichols, D., Curtis, P., Dixon, M., & Lamping, J., (1995). " high latency, low bandwidth windowing in the jupiter collaboration system,":-- unpublished.
[39] Naji, K., and Ibriz, A., (2019). "adaptive MOOC supports the elicitation of learners' preferences," in Int. Conf. Advanced Intelligent Systems for Sustainable Development, pp. 68 - 73.
[40] Oztank, F., Bahrami, M., and Balcisoy, S., (2025). "let me share my groupware: an analysis of groupware as a remote collaboration medium," Learning & Collaboration Technologies.
[41] Oballe-Peinado, Ó., et al., (2020). "FPGA based remote laboratory for digital electronics," in XIV Technologies Applied to Electronics Teaching Conf. (TAEE), pp. 1 - 5.
[42] Pathak, H., Ritik, H., Pawar, R., and Pingle, Y., (2024). "codeFlow: real time collaborative code editor," in Int. Conf. Knowledge Engineering & Communication Systems (ICKECS), pp. 1 - 5.
[43] Palmer, C. R., and Cormack, G. V., (1998). "operation transforms for distributed shared readsheet".
[44] Petricek, T., and Edwards, J., (2025). "denicek: computational substrate for document oriented end user programming," in proc. 38th Annual ACM Symp. UI s/w & Technology, pp. 1 - 19.
[45] Quiroga Perez, J., Daradoumis, T., and Puig, J., (2020). "re-discovering the use of chatbots in education: a systematic literature review," Computer Applications in Engineering Education, vol. 28.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY