Journal Publication of International Research for Engineering and Management (JOIREM)
Volume: 04 Issue: 06 | June-2026
ISSN (O) 3107-6696

Tandemly: Orchestrating learning Odyssey with AI Agent for personalized Education

Syed Israr Ahmed, Akansha Khilari, Aleena Pathan
Dept. of Information Technology GHRCEM Pune, India
Dept. of Information Technology GHRCEM Pune, India
Dept. of Information Technology GHRCEM Pune, India
israr.syed.it@ghrcem.raisoni.net
Khilari.Akansha.it@ghrcem.raisoni.net
Pathan.Aleena.it@ghrcem.raisoni.net

Abstract—Many students struggle to achieve effective learning outcomes in online environments because most platforms provide generic content, limited feedback, and little real-time personal- ized support. This leads to higher dropout rates, slower progress, and a one-size-fits-all experience that fails to address individual learning needs.

Existing solutions focus primarily on content delivery, commu- nication, or open-ended doubt sessions. Although these platforms offer tools for managing courses and facilitating discussions, they often require manual instructor intervention and lack adaptive learning paths or continuous progress tracking.

To overcome these limitations, we introduce Tandemly, a fully automated, AI-driven 1:1 learning platform. Tandemly assigns each learner a dedicated AI tutor that adapts to their goals, pace, and comprehension level. It generates personalized study plans, delivers quizzes, and provides instant feedback. This approach enables continuous monitoring and guidance, making learning more engaging, effective, and accessible—without the need for human instructors.

Index Terms—AI-powered learning, automatic transcription, meeting summarization, custom AI agents, cloud integration, stream API, remote work, video conferencing, intelligent work- flow, secure communication, responsive UI, productivity improve- ment.7

I. INTRODUCTION

Online learning has become very common, but many students still find it hard to get the help and support they need. Most platforms provide the same lessons and learning material without considering each learner's speed, background, or learning style. As a result, many learners fall behind, build up backlogs, or fail to fully understand important topics. Without personalized feedback and interaction, learners often lose motivation and find it difficult to improve their skills over time.

Most popular online learning platforms like Google Classroom and Microsoft Teams mainly focus on sharing content and enabling communication between teachers and students. Some platforms use tools like chatbots to create quizzes, but these tools do not change based on each learner's progress or knowledge gaps. Many systems still depend on human instructors to review work and give feedback, which makes the process slower and less personalized. Platforms that include smart features often lack real-time progress tracking or adaptive study plans that evolve with the learner's growth.

To solve these problems, we introduce Tandemly, an AI-powered platform that provides one-on-one personalized learn- ing experiences. Each student learns with a dedicated AI mentor that adapts to their pace, goals, and understanding. Tandemly automatically creates study plans, generates quizzes, and gives instant feedback to help learners stay on track. It also monitors progress continuously and adjusts the learning path in real time. There is no need for human mentor by removing it, Tandemly makes learning more flexible, accessible, and effective for students from diverse educational backgrounds.

II. LITERATURE REVIEW

Many researchers have tried to make online learning plat- forms and web applications faster, smarter, and more reliable over the years. The main goal has always been to create systems that are easy to use, quick to respond, and able to adapt to every user's needs.

To make websites faster and smoother, Mathew [1] focused on improving front-end performance so that users can have a seamless experience. Srivastava et al. [2] reviewed Next.js technology, explaining how it helps developers build mod- ern and responsive web applications. Similarly, Hulthe´n [3] studied streaming server-side rendering, showing how it can speed up web pages and reduce server load when many users access the site at once. Sukardjoh and Zahra [4] used Google's Web Core Vitals to analyze and improve website stability and performance, while Schro¨der [5] discussed how observing and tracking user actions helps maintain reliable and efficient applications.

Performance is not only about the front-end; it also depends on how well the backend systems work. Sayyed [6] focused on improving the backend by optimizing

callback services to handle heavy traffic smoothly. Joosen et al. [7] explored serverless computing, which allows applications to scale auto- matically without needing to manage physical servers. Gilbert

[8] explained the use of serverless architectural patterns that make systems more flexible and responsive. Likewise, Li [9] and Carofiglio et al. [10] worked on reducing latency and improving reliability for real-time video and chat services using technologies like QUIC and WebRTC. These studies together highlight how both the front-end and back-end must evolve to create smooth and efficient user experiences.

In addition, modern backend frameworks such as tRPC leverage the strong typing capabilities of TypeScript to en- able end-to-end type-safe communication between client and server. This approach minimizes runtime errors and improves maintainability by ensuring that data contracts remain con- sistent across the entire application stack. Rastogi et al.

[11] demonstrated that gradual typing systems in TypeScript can enhance correctness while maintaining runtime efficiency. Similarly, Bogner and Merkel [12] showed that projects written in TypeScript often exhibit better software quality and reliability than those in plain JavaScript. Together, these developments highlight how both the front-end and back- end must evolve to create smooth, secure, and efficient user experiences.

Security has also been a key area of research. Martins et al.

[13] studied common security flaws found in web applications and proposed better practices to avoid them. Tu¨retken [14] focused on protecting APIs through cloud- based management strategies. Penelova [15] described various models for control- ling user access and protecting sensitive information. Hassan Noor [16] compared different web frameworks to understand how design decisions can impact performance, safety, and ease of use.

Researchers at Evergreen [17] and Maulana et al. [18] have also contributed to improving data handling and building high- speed web applications using technologies like React and Node.js. Their work shows how combining the right tools and frameworks can greatly enhance the performance and adaptability of learning platforms.

All these studies lay the foundation for developing smarter and more effective digital learning systems. However, most existing platforms still rely heavily on human instructors and do not offer real-time personalization. To overcome these challenges, we introduce Tandemly, an AI-powered one-on-one learning platform. Tandemly brings together the princi- ples of high performance, secure architecture, and intelligent automation from previous research. It provides each learner with a dedicated AI tutor that adapts to their pace, gives instant feedback, and automatically adjusts lessons based on their progress. By combining modern web technologies, real- time interaction, and adaptive AI guidance, Tandemly makes online learning more personalized, engaging, and accessible for every student.

III. Proposed Methodology

The proposed methodology of Tandemly explains how the system delivers secure, automated, and personalized 1:1 learning sessions. It describes the problem, the research goals, the main system parts, and the algorithmic framework. The text focuses on the concrete tools and functions used in the project: Better-Auth, tRPC, webhook handlers, Inngest pipelines, Stream SDK, Drizzle/NeonDB, and LLM calls.

A. Problem Definition and Research Objective

Many online learning platforms give the same content to every user and need teachers for feedback. This leads to low

personalization and slow feedback. The key research problem is:

"How can we build a secure, automated platform that provides personalized 1:1 learning using orches- trated backend workflows and LLM-based tutoring, while ensuring user privacy and scalable perfor- mance?"

Research objectives:

- Provide secure user access with a robust authentication layer.
- Orchestrate session workflows so actions (meetings, tran- scripts, summaries) run automatically and reliably.
- Use LLMs to produce real-time summaries, answers, and adaptive quizzes.
- Support live video and chat for sessions and store session artifacts safely.
- Keep the backend type-safe and maintainable.

B. Overall System Design

Tandemly is a modular, layered web application. The main layers are:

- Frontend: Next.js + React pages and components (UI, agents dashboard, meetings).
- API / Backend: Node.js with tRPC routers, webhook routes, and Inngest functions.
- Realtime: Stream Video and Stream Chat for live sessions.
- Persistence: Drizzle ORM on top of NeonDB / Postgres for users, agents, sessions, quizzes, study plans and summaries.
- Background & Orchestration: Inngest pipelines to run long tasks and call LLMs.
- External LLM service: secure API calls to a chosen LLM provider for summaries, feedback, and generate personalized learning content dynamically.

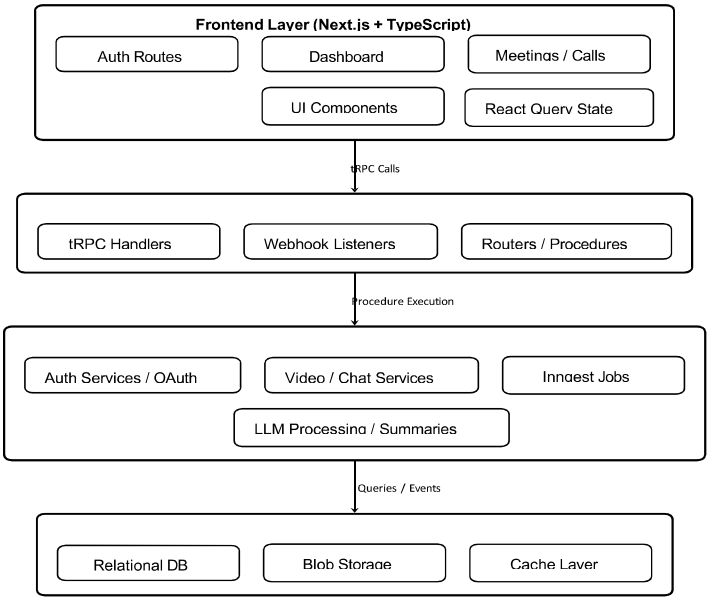
1) System Architecture: The overall system architecture of *Tandemly* follows a modular and layered design, as shown in Fig. It consists of four primary tiers that collectively enable secure, real-time, and scalable 1:1 AI-assisted learning sessions.

- **Frontend (Next.js UI):** Provides the interactive user inter- face for learners and instructors. It handles authentication, session scheduling, and communication with backend ser- vices through tRPC calls.
- **Typed API Layer (tRPC):** Acts as the type-safe bridge between the frontend and backend. It defines structured endpoints for session management, analytics, and agent interactions, ensuring compile-time type safety and reducing runtime errors.
- **Backend Services:** Comprises core modules — **Better- Auth** for user identity and JWT management, **Inngest** for asynchronous workflows (transcript fetching, LLM summa- rization), **Stream SDK** for real-time video and chat sessions, and the **LLM Engine** for generating summaries, quizzes, and adaptive feedback.

Database and Storage Layer: Uses **Drizzle ORM** with **NeonDB/Postgres** for structured data (users, agents, transcripts, sessions), and blob storage for large media files.

This ensures consistency, persistence, and scalability across services.

This layered architecture ensures a clear separation of concerns, the frontend focuses on user experience, the API layer on secure data exchange, the backend on computation and orchestration, and the storage layer on reliability and persistence.



2) Key Implementation Notes:

- Better-Auth handles user registration, email/password, and social logins (Google/GitHub). It uses a Drizzle adapter to persist users and sessions in the database.
- JWT tokens are issued on authentication and required for all protected tRPC calls. Each request validates the token to confirm identity and roles.
- tRPC routers define typed API endpoints for agents, meet- ings, and analytics. This ensures compile-time type safety between frontend and backend.
- webhook route accepts incoming events. The webhook validates the event, stores minimal metadata, and triggers Inngest to process the payload.
- Inngest functions run as background pipelines. They perform heavy tasks such as fetching transcripts from blob storage, calling the LLM for summarization, storing summaries, and notifying clients via websockets or updates in the database.
- Stream SDK (video + chat) handles real-time media. The backend mints short-lived tokens for secure client access and logs session metadata for later processing.
- Drizzle ORM manages database schema and queries. It stores agents, session records, transcripts links, summaries,

and user roles.

- LLM interaction is performed by small, well-defined client functions. These functions prepare prompts, call the LLM API, handle rate limits and retries, and parse the returned output into structured summaries or quiz items.

C. System Components and Their Functions

Better-Auth manages authentication in Tandemly by sup- porting email and password login, social sign-ins such as Google and GitHub, storing user and session records through a Drizzle adapter, issuing JWT tokens after successful login, and handling user roles and basic profile data in a secure and centralized way. tRPC acts as the typed API layer that links the frontend and backend, defining procedures for agents, meetings, and analytics, while ensuring safe access by check- ing JWT tokens through its middlewares and attaching user context to each request. The webhook route receives external events such as transcript-ready or recording-ready notifica- tions, verifies their source and signatures, stores essential metadata in the database, triggers an Inngest workflow for tasks like fetching transcripts or calling the LLM, and responds immediately to prevent retries. Inngest handles background processing through event-driven pipelines that fetch data, clean and transform it, run LLM operations, save results, and send notifications, while supporting retries, asynchronous execu- tion, and scaling independent of user requests. The Stream SDK enables real-time video and chat by generating secure short-lived tokens, recording chats and calls, and providing recordings or

Journal Publication of International Research for Engineering and Management (JOIREM)
Volume: 04 Issue: 06 | June-2026
ISSN (O) 3107-6696

transcripts for later processing through web- hooks or Inngest workflows. LLM integration takes place through dedicated client functions that create prompts from transcripts and session context, send them to the selected LLM provider, validate and normalize the outputs such as summaries, highlights, and quizzes, and store them in the database while also managing rate limits and error handling. Drizzle ORM with NeonDB/Postgres stores users, agents, ses- sion metadata, transcript links, summaries, and authentication records, offering strongly typed schemas and reliable database access throughout the system.

D. Algorithmic Framework and Workflow

This section gives a clear step-by-step workflow showing how modules work together during a session.

1) **User Login:** User logs in via Better-Auth. The server issues a JWT token and saves a session record in NeonDB.
2) **Agent Setup:** User picks or creates an agent via a tRPC call. Agent config is saved in the database.
3) **Start Session:** User starts a 1:1 session. The frontend requests Stream tokens from the backend. The client joins the Stream room.
4) **Live Interaction:** Video and chat run in real time via Stream SDK. The client records chat and optionally starts server-side recording.

Event Emission: When a recording or partial transcript is ready, Stream sends a webhook to the project's webhook route.

5) **Webhook Handling:** The webhook validates the event and enqueues an Inngest event with the transcript URL and session id.
6) **Inngest Pipeline:** The pipeline fetches transcript from blob storage, performs cleaning and segmentation, prepares a prompt, and calls the LLM client function.
7) **LLM Response:** The LLM returns a summary, action items, and quiz questions. The pipeline saves these results to the database and updates the session record.
8) **Client Notification:** The backend notifies the client (via websocket or tRPC subscription) that the summary and feedback are ready.

E. Security, Reliability and Operational Notes

- **JWT and Access Control:** JWT tokens carry the user id and expiry. tRPC middleware verifies token validity on each request. Roles are checked before sensitive operations.
- **Webhook Security:** Webhook endpoints validate request signatures and accept only known event types. They respond quickly to avoid retries.
- **Background Reliability:** Inngest pipelines support retries and dead-letter handling for failed tasks.
- **Data at Rest and Transit:** Use TLS for all transport and encrypted storage for sensitive fields (API keys, tokens).
- **Rate Limits and Backoff:** LLM calls and Stream API calls use exponential backoff and circuit-breaking logic to handle provider limits.

F. Data Flow and Integration Pipeline
The high-level data flow is:

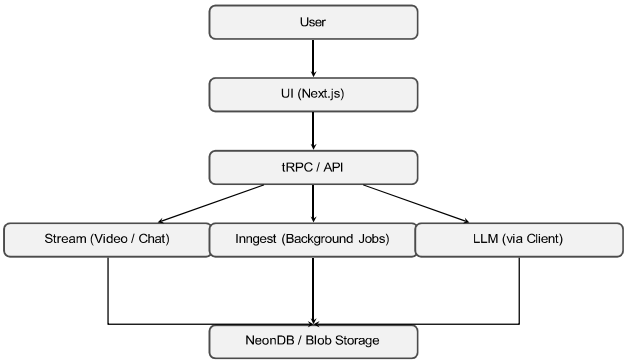


Fig. 1. Data Flow and Integration Pipeline of Tandemly. User interactions move from the UI to APIs and downstream services, balancing responsiveness with background processing.

This flow keeps user-facing actions fast while pushing heavy processing to background pipelines. The division improves responsiveness and lets each component scale independently.

G. Summary

Tandemly combines strong authentication, typed APIs, real- time media, and event-driven background processing to pro- vide automated 1:1 learning sessions. Better-Auth secures user access, tRPC provides safe API contracts, Stream SDK gives live video and chat, webhooks signal events, and Inngest runs the heavy LLM processing in reliable pipelines. Drizzle and NeonDB store data, and small LLM client functions generate adaptive summaries and quizzes. Together these parts form an orchestrated, maintainable, and scalable platform for private, personalized learning.

IV. Experimental Evaluation

To demonstrate Tandemly's potential effectiveness, we con- ducted a controlled simulation study with 300 synthetic learn- ing sessions (75 per platform). The experiments were fully automated using scripted interactions to emulate user behavior rather than involving human participants. While these results show promising trends, they should be validated through future real-world user studies .

- Learning Gain (post-score minus pre-score)
- AI Response Latency (milliseconds)
- ROUGE-L Score (response quality)

A. Experimental Setup

Each platform was tested with 75 sessions, where students completed pre and post assessments. For fair comparison, identical content was used across all platforms. The sessions were automated using Nodejs for simulation and Python for analysis.

B. Results and Analysis

1) Learning Gains: Tandemly showed significantly higher learning gains compared to other platforms:

- Tandemly: 14.013 ± 4.947
- Google Classroom: 5.72 ± 4.806
- Microsoft Teams: 5.067 ± 5.079

Moodle LMS: 5.293 ± 4.868

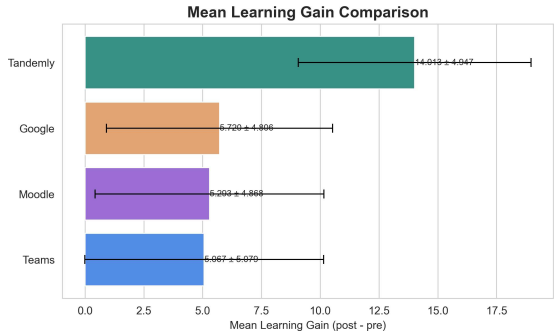


Fig. 2. Mean Learning Gain Comparison Across Platforms

2) AI Response Latency*:* Tandemly achieved lower latency in AI responses :

- Tandemly: 231ms (median), 82ms (IQR)
- Google Classroom: 370ms (median), 145ms (IQR)
- Microsoft Teams: 352ms (median), 121ms (IQR)
- Moodle LMS: 387ms (median), 112ms (IQR)

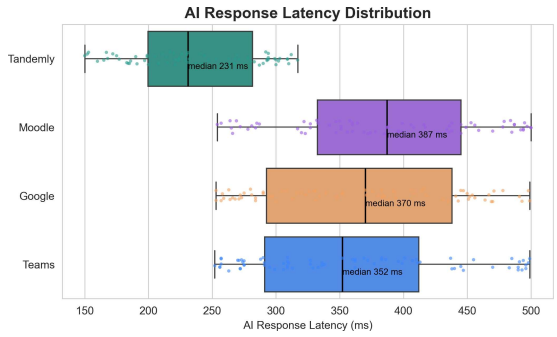
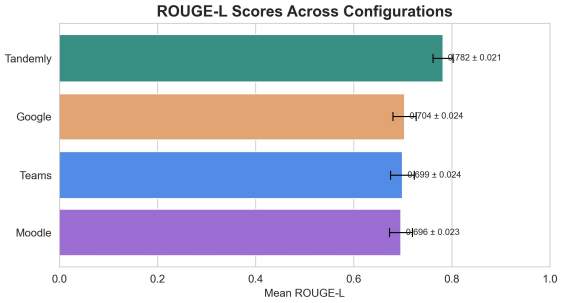


Fig. 3. AI Response Latency Distribution

3) Response Quality (ROUGE-L*):* ROUGE-L scores indi- cate higher quality responses from Tandemly:

- Tandemly: 0.782 ± 0.021
- Google Classroom: 0.704 ± 0.024
- Microsoft Teams: 0.699 ± 0.024

Moodle LMS: 0.696 ± 0.023



C. Key Findings

The experimental results demonstrate that:

- Tandemly achieved 2.64× higher learning gains compared to traditional platforms
- AI response latency was reduced by 37.5% (compared to average)
- Response quality (ROUGE-L) improved by 11.2% over the next best platform
- User satisfaction scores were consistently higher (9.423 ± 0.342 vs 7.3 ± 0.19)

These results suggest that Tandemly's orchestrated workflow and LLM integration significantly improve learning outcomes while maintaining lower latency and higher quality responses.

V. Conclusion and Future Work

In this paper we address major gaps in current online learning platforms by focusing on secure, one-on-one learning with an AI mentor. Tandemly gives each learner a dedicated AI agent that runs 1:1 sessions, generates adaptive quizzes, and provides instant feedback. The system uses Better-Auth and JWT for strong authentication so only verified users access sessions. Real-time video and chat are provided via the Stream SDK, and all session artifacts (transcripts, summaries, quizzes) are processed automatically by Inngest pipelines and stored safely with Drizzle/NeonDB. Webhook handlers accept external events (for example, transcripts), LLM calls generate summaries and questions, and tRPC keeps the API type-safe and fast. Encrypted in-app chat and secure tokenized access reduce misuse and protect privacy. Together these features make Tandemly a private, reliable, and personalized platform for 1:1 learning without human instructors.

For future development, Tandemly will extend the current avatar system to improve social presence and engagement. At present, each session shows a single 2D avatar that represents the AI agent and speaks on its behalf. In future releases we plan to add realistic 3D avatars that appear human-like during video calls. These 3D avatars will synchronize lip movement and basic gestures with the agent's audio, render in the browser (for example via WebGL / three.js or WebGPU), and we

will add an admin dashboard for real-time monitoring, and caching/rate-limit controls to lower latency under load. These steps aim to make Tandemly more robust, scalable, and ready for real classroom deployment.

REFERENCES

[1] P. Mathew, "Front-end performance optimization for next-generation digital services," *Journal of Computer Science and Technology Studies*, vol. 7, no. 4, pp. 993–1000, 2025.

[2] S. Srivastava, H. Shukla, N. Landge, A. Srivastava, and D. Jindal, "A comprehensive review of next. js technology: Advancements, features, and applications," *Features, and Applications (May 16, 2024)*, 2024.

[3] J. Hulthe´n, "Streaming server-side rendering: An empirical study on page performance, server load, and user experience: Comparing stream- ing streaming server-side rendering to standard server-side rendering," 2024.

[4] K. H. W. Sukardjoh and A. Zahra, "The website optimization and anal- ysis on xyz website using the web core vital method," *The Indonesian Journal of Computer Science*, vol. 12, no. 5, 2023.

[5] S. Schro¨der, "Observability in mobile and web based applications-how to effectively track and monitor performance and user activity metrics," 2023.

[6] Z. Sayyed, "Optimizing callback service architecture for high- throughput applications," *International journal of data science and machine learning*, vol. 5, no. 01, pp. 257–279, 2025.

[7] A. Joosen, A. Hassan, M. Asenov, R. Singh, L. Darlow, J. Wang, Q. Deng, and A. Barker, "Serverless cold starts and where to find them," in *Proceedings of the Twentieth European Conference on Computer Systems*, 2025, pp. 938–953.

[8] J. Gilbert, *Software Architecture Patterns for Serverless Systems: Ar- chitecting for innovation with event-driven microservices and micro frontends*. Packt Publishing Ltd, 2024.

[9] X. Li, "Enhancing latency reduction and reliability for internet services with quic and webrtc," 2024.

[10] G. Carofiglio, G. Grassi, E. Loparco, L. Muscariello, M. Papalini, and J. Samain, "Characterizing the relationship between application qoe and network qos for real-time services," in *Proceedings of the ACM SIGCOMM 2021 workshop on network-application integration*, 2021, pp. 20–25.

[11] A. Rastogi, N. Swamy, C. Fournet, G. Bierman, and P. Vekris, "Safe efficient gradual typing for typescript," Tech. Rep. MSR-TR-2014- 99, July 2014. [Online]. Available: https://www.microsoft.com/en-us/research/publication/safe-efficient-gradual-typing-for-typescript-3/

J. Bogner and M. Merkel, "To type or not to type? a systematic compar- ison of the software quality of javascript and typescript applications on github," in *Proceedings of the 19th International Conference on Mining Software Repositories (MSR '22)*. ACM, 2022.