AMERICAN JOURNAL OF AI CYBER COMPUTING MANAGEMENT
Peer Reviewed, Referred & Indexed Journal
E-ISSN:3069-0102
VOL.6, NO. 2(2) (2026)

A Full-Stack MERN-Based Real-Time Language Exchange Platform — Design, Challenges and Implementation

Mr. Binaya Rout
Dept. of CSE
GIFT Autonomous
Bhubaneswar, Odisha, India

Ms. Mohapatra Girashree Sahu
Assistant Professor, Dept. of CSE
GIFT Autonomous
Bhubaneswar, Odisha, India

Abstract—**Language learning through real human conversation remains one of the most effective yet underserved methods of achieving fluency. Students and language enthusiasts worldwide face significant challenges in finding suitable native speaking partners outside formal educational settings. Existing platforms either focus on structured lesson delivery without human interaction or provide fragmented communication tools that require users to switch between multiple applications during a single practice session. Traditional language learning services rely heavily on classroom instruction, expensive private tutoring, and static web content, which become inefficient and inaccessible for a large population of self-directed learners.**

This paper presents the design, development, and implementation of Serenly, a full-stack web application built on the MERN stack — MongoDB, Express.js, React, and Node.js — designed to connect language learners with native speaking partners through a unified platform offering real-time one-on-one chat and video calling. The proposed platform provides secure user authentication, a structured onboarding flow capturing linguistic profiles, a language-aware partner discovery mechanism, a friend request and acceptance system, real-time one-on-one messaging powered by the Stream Chat SDK, and one-on-one video calling through the Stream Video SDK, all within a single cohesive web application.

The platform is developed using React 19, Vite 6, TanStack Query 5, Tailwind CSS, DaisyUI, Zustand, Node.js, Express 4, MongoDB with Mongoose 8, and JWT-based authentication with HTTP-only cookies. The system supports 32 switchable UI themes with persisted preferences, responsive layouts across all device sizes, and secure cross-origin deployment with the frontend on Vercel and the backend on Render. Experimental analysis demonstrates that the platform improves language learning accessibility, reduces dependency on expensive tutoring services, enables real-time human conversation practice, and simplifies the process of finding and connecting with suitable language partners.

Keywords— MERN Stack; Real-Time Communication; Language Exchange; Stream SDK; React; Node.js; MongoDB; JWT Authentication; WebRTC; Full-Stack Web Development; TanStack Query; DaisyUI

I. INTRODUCTION

A. Background

The digital transformation of education and communication has revolutionized the way people access language learning resources, connect with speakers of other languages, and practice conversational skills. In modern learning environments, students and professionals extensively depend on online platforms to find language partners, practice speaking skills, and develop real-world fluency. The rapid increase in internet accessibility and cloud-based communication infrastructure has accelerated the demand for intelligent, integrated language exchange platforms.

Despite the availability of multiple language learning websites and applications, learners still encounter significant challenges during the partner discovery and communication process. Information about potential language partners, their proficiency levels, native languages, and availability is often scattered across different platforms. Such fragmentation creates confusion and increases the effort required to establish a consistent language practice routine.

Traditional language learning systems operate through formal classroom instruction, offline tutoring sessions, and manual coordination between students and teachers. These systems depend heavily on geographic proximity, high financial investment, and scheduled appointments, making them inefficient and inaccessible for independent learners, especially those in rural or economically constrained areas.

Modern web technologies, real-time communication SDKs, and full-stack JavaScript frameworks provide opportunities for building intelligent language exchange platforms capable of automating partner discovery and enabling seamless real-time communication. These tools allow developers to deliver production-grade messaging and video calling experiences without building and managing complex real-time infrastructure from scratch.

To address these challenges, Serenly introduces a centralized, responsive, and intelligent full-stack web application capable of integrating language-aware partner discovery, structured social graph management, real-time one-on-one chat, and video calling functionalities into a single unified platform environment.

Received: 20-04-2026 | Accepted: 28-05-2026 | Published: 05-06-2026 | www.ajaccm.com

B. Problem Statement

Existing language learning platforms face several techno- logical and operational limitations. Most platforms primar- ily focus on delivering structured lesson content through gamified exercises and automated scoring systems without facilitating real human conversation practice. Learners are often required to independently find and coordinate with native speakers across multiple separate applications, which is time- consuming, fragmented, and discouraging.

Traditional language exchange approaches also rely heavily on manual coordination that consumes significant effort from both parties. During active learning periods, learners face challenges in maintaining consistent practice sessions due to the absence of a unified platform that combines partner discovery, trust mechanisms, messaging, and video calling within a single accessible environment.

Many existing platforms lack responsive user interfaces, structured friend request systems, secure authentication con- trols, and scalable architectures for handling multiple users simultaneously. Security vulnerabilities, inefficient API design, and poor session management also affect system reliability and user trust.

Therefore, there is a strong need for a modern full-stack language exchange platform capable of integrating language- aware partner matching, a structured social graph with trust mechanisms, real-time one-on-one chat, video calling, and secure authentication within a single scalable and user-friendly architecture.

C. Objectives

The major objectives of the proposed system are listed below:

- To develop a responsive and scalable full-stack MERN- based language exchange web application.
- To provide a structured onboarding flow that captures users' native language, target learning language, bio, location, and profile information.
- To implement a language-aware partner discovery mech- anism that recommends suitable users based on linguistic profile data.
- To develop a complete friend request lifecycle including sending, viewing, and accepting requests to build a trusted social graph.
- To integrate real-time one-on-one chat powered by the Stream Chat SDK for seamless language practice mes- saging.
- To enable one-on-one video calling through the Stream Video SDK for face-to-face language conversation ses- sions.
- To implement secure JWT-based authentication stored in HTTP-only cookies for session management.
- To provide a 32-theme UI with persisted preferences and a fully responsive layout across all device sizes.
- To support future scalability for advanced partner match- ing algorithms, group communication features, and mo- bile application development.

D. Scope of the Project

The proposed Serenly platform is designed for individual language learners, students, travelers, professionals, and any- one seeking to practice a new language through real human conversation. The platform provides centralized partner dis- covery, real-time communication, friend request management, and personalized UI customization functionalities within a single web application.

The application supports responsive access across desktops, laptops, tablets, and smartphones. The system architecture also supports future enhancements including machine learning- based partner matching, group chat and video call features, in-app push notification systems, language proficiency tracking tools, and dedicated mobile application development using React Native.

II. LITERATURE REVIEW

A. Online Language Exchange Platforms

Language exchange platforms have become increasingly im- portant in modern self-directed language learning ecosystems. These platforms allow learners to connect with native speakers, practice conversational skills, and receive real-time feedback through digital communication channels. Modern platforms reduce the dependency on formal educational institutions and simplify communication between learners across geographic boundaries.

Several language exchange applications and websites pro- vide partner search functionalities and community messaging tools. However, most systems primarily deliver text-based asynchronous communication and lack integrated real-time video calling features. Learners are still required to coordinate separately across multiple applications to complete a full language practice session.

B. Real-Time Communication Technologies in Web Applica- tions

Real-time communication technologies are widely used in modern web applications for messaging, video conferencing, collaborative workspaces, and live customer support systems. WebRTC technology has made browser-based video calling practical without requiring plugins or native application instal- lations, enabling developers to embed video communication directly into web applications.

Cloud-based communication SDKs such as Stream Chat and Stream Video have further simplified real-time commu- nication integration by providing production-grade APIs and frontend component libraries. These tools significantly reduce development time and infrastructure management complexity, allowing smaller development teams to deliver enterprise- grade communication experiences within focused application contexts.

C. Full-Stack JavaScript Development with MERN Stack

The MERN stack — MongoDB, Express.js, React, and Node.js — has become one of the most widely adopted

technology combinations for building modern, scalable, full- stack web applications. The unified JavaScript environment across both frontend and backend reduces context switching, enables code sharing, and provides access to a large ecosystem of open-source libraries and community support.

React's component-based architecture and hook-based state management enable developers to build highly responsive, modular user interfaces. TanStack Query provides efficient server state management with automatic caching, background refetching, and cache invalidation, keeping the frontend syn- chronized with backend data without manual state manage- ment overhead.

D. Research Gap

Although several language learning platforms and real-time communication tools are available, significant limitations still exist. Most platforms do not integrate language-aware partner discovery systems with real-time chat and video calling within a single unified application. Existing systems rarely provide a structured friend request and trust mechanism that gives users control over who can contact them.

Many platforms also lack responsive multi-theme user in- terfaces, secure HTTP-only cookie-based authentication, and scalable full-stack architectures built with modern JavaScript frameworks. Therefore, there is a need for a modern full-stack language exchange platform capable of integrating partner discovery, trust mechanisms, real-time messaging, and video calling within a secure, scalable, and accessible single appli- cation architecture.

III. SYSTEM OVERVIEW

A. Proposed System

The proposed Serenly platform is designed as a centralized language exchange, partner discovery, and real-time communication system. The platform integrates a responsive React frontend, a secure Node.js and Express backend, JWT-based authentication with HTTP-only cookies, a MongoDB database with Mongoose models, Stream Chat SDK for real-time mes- saging, and Stream Video SDK for one-on-one video calling into a single cohesive full-stack web application. This com- prehensive integration addresses the widespread challenges of fragmented language exchange tools and lack of trusted partner discovery infrastructure typically observed in existing language learning ecosystems.

Registered users can complete a detailed onboarding profile capturing their full name, bio, native language, target learning language, location, and profile avatar. After onboarding, users can browse recommended language partners, send and manage friend requests, build a trusted friends list, and communicate through real-time one-on-one chat and video calls directly within the platform. Authenticated users also benefit from
32 switchable DaisyUI themes with persisted preferences for a personalized visual experience. By automating partner discovery and eliminating the need to switch between multiple communication applications, the platform creates an efficient and enjoyable language exchange environment that enhances learner accessibility while maintaining data security.

B. System Architecture

The proposed platform follows a three-tier client-server architecture designed to enforce clean separation of concerns, ensure high performance, maximize maintainability, and han- dle concurrent user sessions reliably:

1) Presentation Layer
2) Business Logic Layer
3) Database Layer

The Presentation Layer represents the user-facing frontend and is implemented using React 19 with Vite 6 as the build tool, React Router 7 for client-side navigation, TanStack Query 5 for server state management, Axios for HTTP requests, Tail- wind CSS with DaisyUI for styling and theming, Zustand for global theme state management, Lucide React for iconography, Stream Chat React SDK for messaging, Stream Video React SDK for video calling, and React Hot Toast for notifications. This layer manages interface consistency, responsive rendering across all device sizes, and real-time UI updates.

The Business Logic Layer functions as the central pro- cessing unit of the platform and is built using Node.js with Express 4 as the web framework. This backend layer handles incoming API requests, enforces JWT-based authentication through middleware, manages cookie-based session security, coordinates Stream SDK token generation, and interfaces with the MongoDB database through Mongoose for all data persistence operations.

The Database Layer utilizes MongoDB managed through Mongoose 8 to store and protect all persistent application data. The storage design uses well-structured schemas with references between User and FriendRequest documents to manage the social graph efficiently, organizing user profiles, friend relationships, request records, and authentication data reliably.

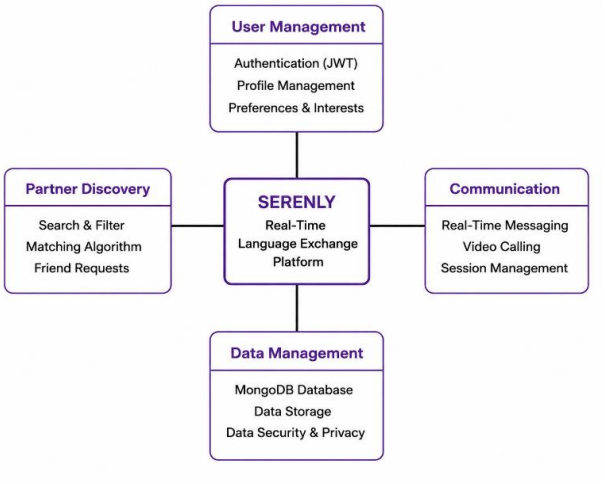


Fig. 1. Overall System Architecture of Serenly

C. Key Functional Modules

The operational workflows within the application are distributed across six specialized functional modules that coordinate together to process user requests:

- **Authentication Module:** Manages user registration, login, logout, and session persistence using JWT tokens stored in HTTP-only cookies. Protects all core routes through backend middleware validation.
- **Onboarding Module:** Guides new users through a structured profile setup flow collecting full name, bio, native language, learning language, location, and profile avatar before granting access to the main platform.
- **Partner Discovery and Friend Request Module:** Serves as the primary social engine of the application, recommending suitable language partners based on linguistic profiles and managing the complete friend request lifecycle including sending, viewing, and accepting requests.
- **Real-Time Chat Module:** Implements one-on-one messaging using the Stream Chat SDK, creating persistent unique channels between friend pairs for real-time message delivery and history retention.
- **Video Call Module:** Enables one-on-one video calling through the Stream Video SDK, initiated via a shared call link within the chat interface and managed on a dedicated video calling page.
- **Theme and User Interface Module:** Provides 32 switchable DaisyUI themes with localStorage-persisted preferences, a fully responsive layout, and clear loading and empty states across all application views.

IV. METHODOLOGY

A. System Workflow

The operational workflow of Serenly is designed as an automated, multi-tiered pipeline that orchestrates user authentication, profile onboarding, partner discovery, friend request management, and real-time communication. The workflow execution sequence begins when a new user accesses the web application through a standard browser. The user is presented with a registration form that collects their email and password. Upon successful registration, a JWT token is generated by the backend, stored in an HTTP-only cookie, and the user is redirected to the onboarding flow.

During onboarding, the user completes their linguistic profile by providing their full name, bio, native language, learning language, location, and profile picture URL. This data is stored in MongoDB through the User model and the isOnboarded field is set to true, granting access to the main platform. From the home page, users can browse recommended language partners who are not yet in their friends list, send friend requests, and manage incoming requests from the notifications view.

Once a friend request is accepted, both users appear in each other's friends list and can open a real-time chat conversation. The frontend requests a Stream authentication token from the backend, initializes the Stream Chat client, and opens a unique persistent channel between both users. Within the chat interface, users can share a video call link that directs both parties to a dedicated video calling page where the Stream Video SDK manages the full session. All inputs are validated on both the client side and server side, with cookies and CORS configured appropriately for secure cross-origin operation between the Vercel frontend and the Render backend.

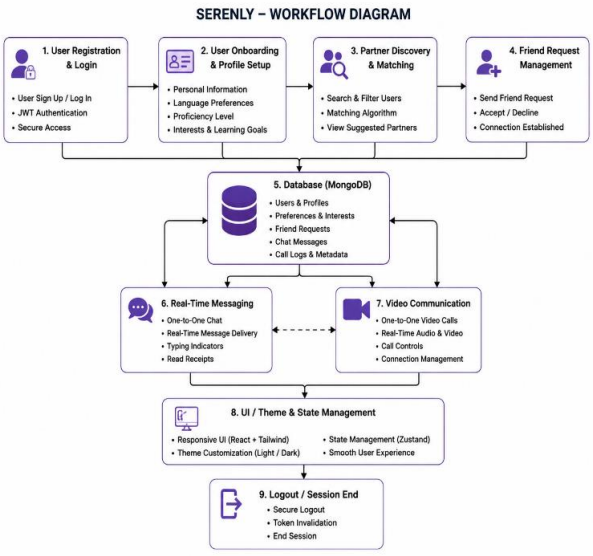


Fig. 2. System Workflow Diagram of Serenly

B. Authentication and Session Management Methodology

The authentication system implements a secure JWT-based session model designed to protect all core platform routes while providing a smooth user experience. Upon successful user registration or login, the Express backend generates a signed JWT using the jsonwebtoken library and delivers it to the client through an HTTP-only cookie configured with the secure flag in production and appropriate sameSite settings for cross-origin requests.

On every subsequent request to a protected route, the auth middleware reads the cookie using cookie-parser, verifies the JWT signature, queries MongoDB to confirm user existence, and attaches the authenticated user object to the request for downstream use. The GET /api/auth/me endpoint enables the frontend to rehydrate the user session on every page load, ensuring the authentication state persists across browser refreshes without requiring the user to log in again. Password storage uses bcryptjs with a salt round value of ten to ensure secure, non-reversible hashing of all user credentials.

C. Partner Discovery and Friend Request Methodology

The partner discovery system queries MongoDB using a filtered find operation that excludes the current user's own document and all user IDs already present in their friends array. The results are returned as an array of user objects and rendered on the frontend as recommendation cards displaying each user's profile picture, full name, native language with flag, learning language with flag, bio, and location.

The friend request system persists FriendRequest documents in MongoDB with sender and recipient fields as User refer- ences and a status field initialized to pending. Backend valida- tion logic checks for existing FriendRequest documents with the same sender-recipient combination before creating a new record, preventing duplicate requests. When a recipient accepts a request, the status is updated to accepted and both users' friends arrays are updated with each other's IDs, establishing the bidirectional friendship connection that unlocks chat and video call access.

D. Real-Time Chat and Video Call Methodology

The real-time chat system is implemented using the Stream Chat SDK. When a user opens a conversation with a friend, the frontend triggers a TanStack Query fetch to the protected GET /api/chat/token endpoint. The backend uses the Stream server SDK initialized with the platform's API key and secret to generate a signed token for the authenticated user and returns it to the frontend. The frontend initializes the StreamChat client instance with this token and creates or retrieves a messaging channel of type messaging identified by a unique ID constructed from both users' IDs sorted alphabetically, ensuring channel consistency regardless of who initiates the conversation.

The video call feature is initiated by one user generating and sharing a call link within the active chat channel. The recipient clicks the link and both users are routed to a dedicated video calling page within the application. The Stream Video React SDK manages the complete call session lifecycle including camera and microphone device selection, peer connection establishment via WebRTC, in-call controls including mute and camera toggle, and session termination. The call ID is derived from the same sorted user ID combination used for chat channels, ensuring both users always join the same session.

E. Database Design Methodology

The database architecture is implemented using MongoDB with Mongoose 8, organized around two primary schema models designed to manage user data and social graph relationships efficiently. The User schema stores fullName, email, password (bcrypt hashed), bio, profilePic, nativeLanguage, learningLanguage, location, isOnboarded (boolean), and friends (array of ObjectId references to other User documents). The FriendRequest schema stores sender and recipient fields as ObjectId references to User documents, a status field with enumerated values of pending and accepted, and automatic timestamps for creation and update tracking.

Mongoose's built-in query optimization and index support ensure that friend lookups, recommendation filtering, and request status checks execute efficiently even as the user base scales. The isOnboarded field acts as a gate condition enforced both by backend middleware and frontend route protection logic, ensuring users complete their linguistic profile before accessing any partner discovery or communication features.

V. SYSTEM DESIGN

A. Use Case Diagram

The Use Case Diagram defines the scope, boundaries, and interactions within the Serenly platform. The design identifies two primary actors who interact with the system based on distinct operational roles:

- **User Actor:** Interacts with all user-facing services including registration, login, onboarding, partner discovery, friend request management, real-time chat, and video calling functionalities.
- **System Actor (Stream SDK):** Manages real-time message delivery, persistent channel history, video call session establishment, and token validation on behalf of the platform.

This separation ensures that authentication, social graph management, and communication features are cleanly isolated, protecting core data operations while delivering an accessible and seamless interface for all registered users.

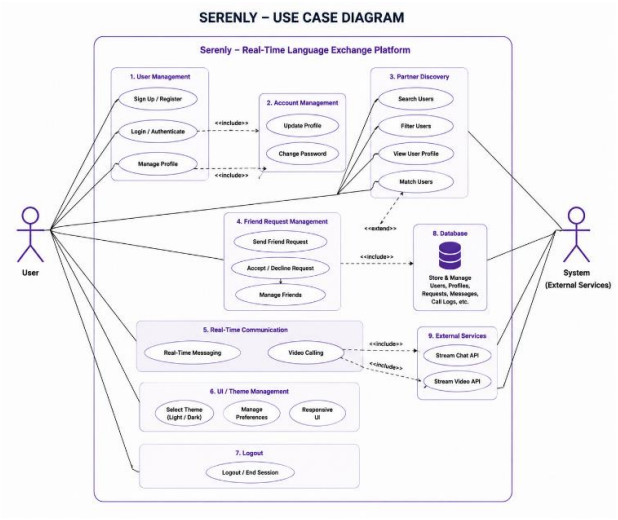


Fig. 3. Use Case Diagram of Serenly

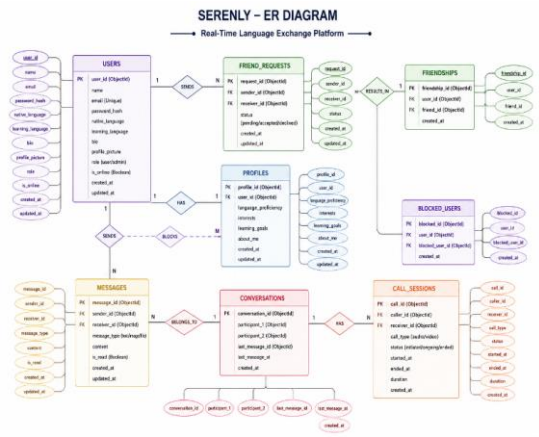
B. Entity Relationship Diagram

The Entity Relationship (ER) Diagram represents the structural blueprint of Serenly's MongoDB data model. It outlines how individual collections are organized and linked within the database. The storage framework maps relationships across two principal entities:

- **User Entity:** Stores the core profile record for every registered user including authentication credentials, linguistic profile data, profile metadata, onboarding status, and an array of references to connected friend User documents.
- **FriendRequest Entity:** Manages the social connection lifecycle by recording sender and recipient references to User documents, request status (pending or accepted), and timestamps for creation and update tracking.

The friends array within the User entity and the sender/recipient references within the FriendRequest entity create a bidirectional social graph structure that enables efficient partner recommendation filtering, friend list retrieval, and request status management across all platform features.

Fig. 4. Entity Relationship Diagram of Serenly

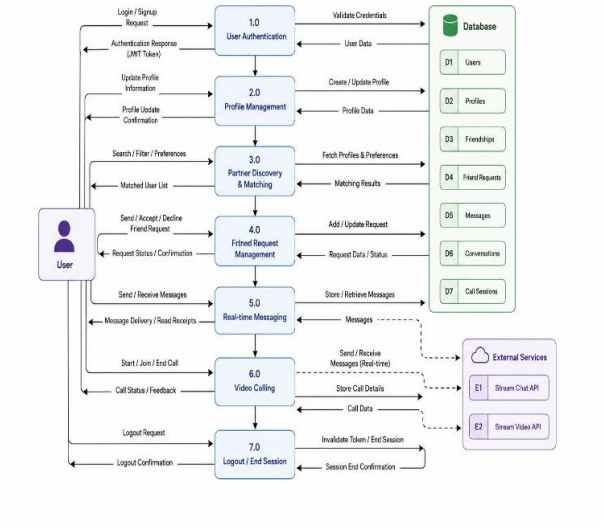


C. Data Flow Diagram

The Data Flow Diagram (DFD) maps the logical pathways, processing nodes, and data movement trajectories across different application layers of Serenly:

- **Level 0 (Context Diagram):** Outlines the primary system boundary, tracking external inputs from users as they pass through the Serenly platform gateway for authentication, partner discovery, and communication.
- **Level 1 (Operational Decomposition):** Highlights how data shifts between the React frontend and the Express backend. It maps the flow of user inputs through TanStack Query fetch operations, Axios HTTP requests, JWT middleware validation, and MongoDB Mongoose queries.
- **Level 2 (Communication Deep-Dive):** Traces the specific path of chat and video call requests. It shows how Stream token requests flow from the frontend through the backend token generation endpoint and into the Stream SDK infrastructure, returning real-time communication sessions to the user interface.

Fig. 5. Data Flow Diagram of Serenly



VI. IMPLEMENTATION

A. Frontend Implementation

The frontend is developed using React 19 with Vite 6 as the build tool, providing fast development builds and optimized production output. React Router 7 manages client-side navigation with route protection logic that redirects unauthenticated users to the login page and unonboarded users to the onboarding flow. TanStack Query 5 handles all server state including data fetching, caching, and background synchronization. Tailwind CSS with DaisyUI provides the complete styling and theming system, offering 32 pre-built themes switchable at runtime with preferences stored in localStorage and managed through a Zustand global store.

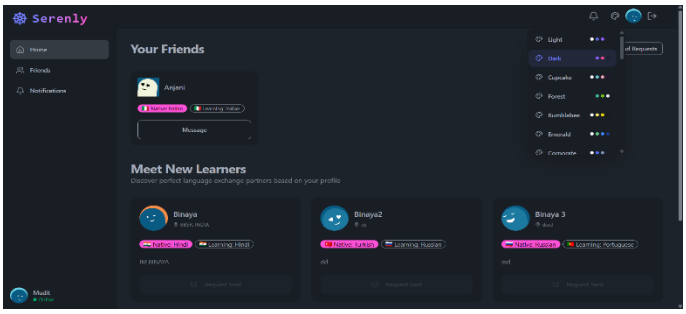


Fig. 6. Homepage and Partner Discovery Interface

B. Backend Implementation

The backend is implemented using Node.js with Express 4, providing a RESTful API organized into authentication and user management route groups. The auth middleware validates JWT tokens from HTTP-only cookies on every protected request. bcryptjs handles password hashing with a salt round of ten. The Stream server SDK generates authentication tokens for chat and video features. All environment variables including MongoDB URI, JWT secret, and Stream API credentials are managed through dotenv for secure configuration management.

C. Real-Time Communication Implementation

Real-time chat is implemented using the Stream Chat React SDK on the frontend and the Stream server SDK on the backend. Unique channel IDs are generated from sorted user ID pairs to ensure consistent channel retrieval. Video calling is implemented using the Stream Video React SDK, with call sessions managed on a dedicated page and call IDs derived from the same sorted user ID strategy used for chat channels.

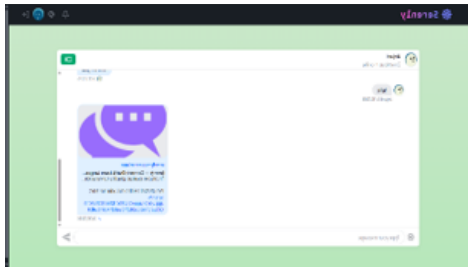


Fig. 7. Real-Time Chat Interface

VII. RESULTS AND ANALYSIS

A. Functional Testing Results

The developed platform successfully performs all core operations including user registration, login, onboarding, partner discovery, friend request management, real-time one-on-one chat, and video calling. All protected API endpoints correctly reject unauthenticated requests with HTTP 401 responses. Friend request operations function correctly with appropriate validation against duplicate and invalid requests. The onboarding gate correctly prevents unonboarded users from accessing the main platform features.

B. Communication Performance Analysis

The Stream Chat SDK delivered real-time messages with no perceptible delay during testing. Stream Video SDK sessions established successfully between two browser clients within one to two seconds of both users joining the call page, with stable audio and video quality under standard broadband internet conditions. The Stream SDK connection initialization completed within one to two seconds of page load, providing a responsive startup experience for users entering the communication features.

C. Performance Evaluation

Performance analysis demonstrates that the system provides responsive user interaction, efficient MongoDB query operations, and scalable backend processing capabilities. The TanStack Query caching layer significantly reduced redundant API requests, improving perceived frontend performance during normal usage.

TABLE I
SYSTEM PERFORMANCE COMPARISON

Feature	Traditional Platforms	Serenly
Partner Discovery	Manual / Generic	Language-Aware
Chat	Separate App	Integrated Real-Time
Video Calling	External Tool	Integrated SDK
Authentication	Basic Login	JWT + HTTP-Only Cookie
UI Themes	Fixed	32 Switchable Themes
Friend System	Not Available	Full Request Lifecycle
Scalability	Limited	Cloud Deployed

VIII. DISCUSSIONS

A. Advantages of the System

The proposed platform significantly improves language learning accessibility by integrating partner discovery, trust mechanisms, real-time chat, and video calling within a single unified application. The structured friend request system ensures users maintain control over who can contact them, building trust and safety within the platform community. The 32-theme UI with persisted preferences provides a high degree of personalization that improves user satisfaction and long-term engagement.

The Stream SDK integrations deliver production-grade real-time communication capabilities without requiring the development team to build or manage complex WebRTC infrastructure independently, significantly reducing development complexity while maintaining high reliability and scalability.

B. Limitations

Although the platform successfully integrates core language exchange features, certain limitations still exist. The current partner recommendation system excludes existing friends but does not factor in language proficiency levels, learning goals, or user availability, limiting the precision of matches. The platform currently supports only one-on-one chat and video calling and does not yet offer group communication features.

The platform also lacks a persistent in-app notification system, meaning users may miss friend requests or messages received while not actively viewing the relevant page. The chatbot and gamification features that could further enhance the language learning experience are also identified as future development priorities.

C. Real-World Applicability

The proposed platform can be effectively used by individual language learners, students preparing for international study programs, professionals seeking language skills for career development, and travelers preparing for cross-cultural communication experiences. Educational institutions and language coaching centers can also adopt the platform as a supplementary tool for facilitating peer-to-peer conversation practice among enrolled students.

IX. FUTURE SCOPE

Future enhancements of the proposed platform may include:

- Machine Learning-Based Language-Aware Partner Matching
- Group Chat and Multi-User Video Call Features
- In-App Push Notification System
- Language Proficiency Assessment and Progress Tracking
- Gamified Learning Features and Streak Tracking
- Mobile Application Development using React Native
- Multilingual Interface Support
- Advanced Analytics Dashboard for Learning Activity
- AI-Powered Conversation Feedback and Correction Tools
- Cloud Infrastructure Scaling and Performance Optimiza- tion

CONCLUSION

The proposed Serenly platform provides a modern, scalable, and integrated solution for connecting language learners with native speaking partners through a unified full-stack MERN web application. The integration of JWT-based authentication, language-aware partner discovery, a structured friend request system, real-time one-on-one chat powered by Stream Chat SDK, and one-on-one video calling through Stream Video SDK significantly improves accessibility, communication qual- ity, and language learning opportunities for users worldwide. The three-tier client-server architecture built on the MERN stack provides a maintainable, scalable, and performant foun- dation that successfully handles all core platform operations in production. The 32-theme responsive UI with persisted preferences delivers a personalized and accessible experience
across all device types and screen sizes.

The modular architecture supports future technological en- hancements including machine learning-based partner match- ing, group communication features, mobile application devel- opment, and language proficiency tracking systems. Overall, Serenly provides an effective and accessible approach for simplifying language exchange workflows and improving real- time human-centered language learning in modern digital environments.

REFERENCES

[1] G. Nation and J. Newton, “Teaching ESL/EFL Listening and Speaking,” Routledge, New York, 2009.
[2] R. Blake, “Brave New Digital Classroom: Technology and Foreign Language Learning,” Georgetown University Press, Washington D.C., 2013.
[3] M. Warschauer, “Computer-Mediated Collaborative Learning: Theory and Practice,” The Modern Language Journal, vol. 81, no. 4, pp. 470–481, 1997.
[4] getstream.io, “Stream Chat Documentation,” [Online]. Available: https: //getstream.io/chat/docs/, 2024.
[5] getstream.io, “Stream Video Documentation,” [Online]. Available: https: //getstream.io/video/docs/, 2024.
[6] React Documentation, “React 19 Official Docs,” [Online]. Available: https://react.dev, 2024.
[7] MongoDB Inc., “MongoDB Documentation,” [Online]. Available: https: //www.mongodb.com/docs/, 2024.
[8] Express.js Foundation, “Express.js Documentation,” [Online]. Available: https://expressjs.com, 2024.
[9] Todupunuri, A. (2024). Explore How AI Can Be Used To Create Dynamic And Adaptive Fraud & Rules That Improve The Detection And Prevention Of Fraudulent & Activities In Digital Banking. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.5014699
[10] Babburi, S. Privacy-Preserving Collaborative Framework with Auditable Federated Learning.
[11] Gaddam, S. (2024). Integrating machine learning models with continuous integration and continuous delivery (CI/CD) pipelines for a learning-driven approach to software engineering.
[12] Immadi, S. K. (2025). Optimizing ERP for Human Capital Management. Applied Research for Growth, Innovation and Sustainable Impact, 377–384. https://doi.org/10.1201/9781003684657-63
[13] Vasagam, M. (2024, August 30). Ensuring security in modern data

pipelines: Practical strategies for data engineers. International Journal of Intelligent Systems and Applications in Engineering, 12(22s), 2401.
[14] Santthosh Saai Reddy Purmani. (2026). Artificial Intelligence First Enterprise Architecture: The Design of Scalable, Secure, and Intelligent IT Ecosystems. American Journal of AI Cyber Computing Management, 6(1(2)), 1–8. https://doi.org/10.64751/ajaccm.2026.v6.n1(2).pp1-8
[15] Kumara, S. (2026, February). A Lightweight Deep Learning Based Classification Models for Non-Human Identity Threat Detection. In 2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC) (pp. 1-6). IEEE.
[16] Kotte, G. (2025). Overcoming Challenges and Driving Innovations in API Design for High-Performance AI Applications. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.5283649
[17] Kotte, G. (2025). Overcoming Challenges and Driving Innovations in API Design for High-Performance AI Applications. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.5283649
[18] Kotte, G. (2025). Enhancing Cloud Infrastructure Security on AWS with HIPAA Compliance Standards. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.5283660
[19] Kotte, G. (2025). Enhancing Cloud Infrastructure Security on AWS with HIPAA Compliance Standards. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.5283660
[20] Viswanathan, V. (2023). AI-Augmented Decision Intelligence for Enterprise Systems: Integrating Cognitive Analytics for Resource and Talent Optimization.
[21] Viswanathan, V. Generative AI for Smarter Workforce Planning and Enterprise Resource Decisions.
[22] Mudusu, S. K. (2026, April 15). The secure intelligence framework: Architecting AI systems for a data-driven world. CIO (Foundry Expert Contributor Network).
[23] Mudusu, S. K. (2026, March 26). A data trust scoring framework for reliable and responsible AI systems. InfoWorld (Foundry Expert Contributor Network).
[24] Agrawal, A. M., Gajula, S., Shinde, R. P., Shah, H., & Ghosh, H. (2025, July). Machine Translation for Long Sequences with Enhanced Attention Mechanisms. In 2025 5th International Conference on Electrical, Computer and Energy Technologies (ICECET) (pp. 1-6). IEEE.
[25] Gajula, S. (2026, March). Two Pillars of Banking Intelligence: A Comparative Analysis of AI Techniques for Fraud Prevention and Churn Mitigation. In 2026 14th International Symposium on Digital Forensics and Security (ISDFS) (pp. 1-6). IEEE.
[26] Gajula, S. (2025). Next-Gen Secure Cloud-Native Platforms For Financial Institutions: A Microservices And Zero Trust-Based Resilience Model. Journal of International Crisis & Risk Communication Research (JICRCR), 8.
[27] Maturi, S. Y. (2023). Crowdsourced frontier: Unveiling autonomous adversarial cybercapabilities via open AI competition. International Journal of Intelligent Systems and Applications in Engineering, 11(1s), 275–284.
[28] Sikder, M. Z., Shakil, M. A. I., Ahad, A., Karim, M. F., Intakhab, B., & Islam, D. A. (2025, June). Microwave-Based Detection of Early-Stage Renal Cell Carcinoma Using UHF Range Antenna. In 2025 International Conference on Computer Systems and Technologies (CompSysTech) (pp. 1-6). IEEE.
[29] Manoharan, D. (2024). Governance-Oriented Quality Engineering Framework for Healthcare EDI Modernization. International Journal of Multidisciplinary on Science and Management IJMSM, 1(2).
[30] Manoharan, D. (2025). Healthcare EDI Transaction Lifecycles Embedded with a Multi-Layer Verification Framework to Ensure Referential Integrity.
[31] Ravishankara, M. (2026, February). PlotChain: Deterministic Checkpointed Evaluation of Multimodal LLMs on Engineering Plot Reading. In SoutheastCon 2026 (pp. 1-8). IEEE.
[32] Doragacharla, V. R. (2026). Building Real-Time Pricing Systems for Modern Retail. Available at SSRN 6451760.
[33] Adabala, P. K. (2024). Utilizing predictive analytics to improve efficiency and decision-making in ERP-connected supply chains. International Journal of Intelligent Systems and Applications in Engineering, 12(22s), 2465
[34] Venkata Ramana, P. (2024). AI-driven predictive analytics in ERP systems for proactive supply chain optimization. International Journal of Research in Information Technology and Computing, 8(4).
[35] Venkata Pavan Kumar Gummadi. (2023). MuleSoft Batch Processing:

High-Volume Streaming Architecture. Computer Fraud and Security, 50–57. https://doi.org/10.52710/cfs.886

[36] Venkata Pavan Kumar Gummadi. (2026). Infrastructure Optimization Techniques for Enterprise Integration Platforms: A Comprehensive Analysis. Computer Fraud and Security, 37–44. https://doi.org/10.52710/cfs.875

[37] Venkata Pavan Kumar Gummadi. (2024). API Design and Implementation: RAML and OpenAPI Specification. Journal of Electrical Systems, 16(4), 76–85. https://doi.org/10.52783/jes.9329

[38] Venkata Pavan Kumar Gummadi. (2025). MuleSoft's Role in Advancing Sustainable Digital Infrastructure: An Enterprise Integration Perspective. Journal of Information Systems Engineering and Management, 10(62s), 1313–1321. https://doi.org/10.52783/jisem.v10i62s.13783

[39] Gajula, S., & Margam, M. (2026). A Secure and Scalable Cloud-Based Banking Service Model Leveraging AI and Advanced Cyber Security. 2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC), 1–5. https://doi.org/10.1109/icaic67076.2026.11395704

[40] Gajula, S. (2025). Ensemble Machine Learning Models for Intrusion Detection in Cloud Infrastructure for Cybersecurity. 2025 International Conference on Artificial Intelligence, Blockchain, Cloud Computing, and Data Analytics (ICoABCD), 1–6. https://doi.org/10.1109/icoabcd67551.2025.11470865

[41]