CS3532 – FullStack and Web Development

STREAMIFY

By

Gayatri Mishra 1RVU23CSE167
Anirudh Anand 1RVU23CSE059
Bhavani Mishra 1RVU23CSE109
Anmol B Rao 1RVU23CSE062

School of Computer Science and Engineering

RV University, Bangalore

2025-2026

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

School of Computer Science and Engineering

CERTIFICATE

Certified that the CS3532 Advanced AWS Cloud computing Project Case Study titled Streamify is carried out by Gayatri Mishra (1RVU23CSE167), Anirudh Anand (1RVU23CSE059), Bhavani Mishra (1RVU23CSE109) and Anmol B Rao (1RVU23CSE062) who is a bonafide student of the School of Computer Science and Engineering, RV University, Bengaluru, during the year 2025–26. It is certified that all corrections/ suggestions from all the continuous internal evaluations have been incorporated in the project and in this report.

Faculty Guide   Program Director

INTRODUCTION

In recent years, the demand for real-time communication platforms has grown exponentially due to increased remote collaboration, social networking, and online communities. Integrating chat, video calling, and social features into a single web application allows users to interact seamlessly and enhances the user experience.

This project focuses on developing an interactive language exchange platform designed to help users improve their speaking skills in a target language through real-time chat and video calling. The application enables users to sign up, select their native language as well as the language they want to practice, and connect with other learners around the world. By combining text messaging and peer-to-peer video communication, the platform creates an immersive environment where users can practice conversation, receive instant feedback, and build confidence in their language abilities. The system offers extensive personalization with 32 user interface themes, allowing each user to tailor their experience to their preferences.

The application is a full-stack web platform built using the MERN stack (MongoDB, Express.js, React.js, Node.js) alongside WebRTC and Stream SDK for seamless real-time communication. Users create accounts and complete an onboarding process to define their language preferences. After registration, they are directed to a home screen where they can search for and add friends who share similar language goals. Connection requests appear in a dedicated notification panel, and once accepted, users can begin chatting or start video calls to practice speaking together. The platform not only facilitates meaningful cultural exchange but also provides a modern, user-friendly interface with customizable themes to enhance the learning experience.

FEATURES

User Authentication & Secure Login (JWT)

- Users can create accounts using a secure registration system.
- Login is protected using JSON Web Tokens (JWT), ensuring only authenticated users access the platform.
- Credentials and sensitive data are stored securely in MongoDB.

Language Preference Onboarding

- After signing up, users select their native language and target language.
- These preferences help match users with suitable conversation partners.
- The onboarding process customizes the user's learning path.

Friend System & Social Networking

- Users can search for friends based on language goals.
- Can send, accept, or decline friend requests.
- A dedicated notifications panel shows incoming requests in real time.
- Enhances the sense of community and collaborative learning.

Real-Time Text Messaging (Stream SDK)

- Instant messaging between connected users.
- Messages appear in real time using Stream SDK's WebSocket-based channels.
- Includes timestamps and smooth scrolling for seamless conversation flow.
- Supports one-on-one chat environments designed for language practice.

Peer-to-Peer Video Calling (WebRTC + Simple-Peer)

- High-quality audio/video calls between two users.
- Uses WebRTC for low-latency, encrypted peer-to-peer connections.
- Simple-Peer handles connection negotiation and media stream setup.
- Users can mute, unmute, and end calls within a responsive video interface.
- Ideal for speaking practice and real-time pronunciation feedback.

Customizable UI Themes (32 Options)

- The application supports 32 user-selectable themes.
- Users can personalize the UI according to preference.
- Enhances engagement and provides a modern, interactive feel.

Responsive & Modern User Interface (React + Tailwind CSS)

- Clean, flexible interface built using React components.

- Tailwind CSS ensures fast, aesthetic UI customization and responsiveness.
- Works smoothly on desktops, tablets, and smartphones.

Real-Time Notifications

- Users receive immediate alerts for:
    - Friend requests
    - Accepted connections
    - Incoming calls
- Notifications appear without refreshing the page.

Scalable Backend Architecture (Node.js + Express)

- RESTful APIs handle user authentication, friends management, and chat metadata.
- The backend manages call signaling and room coordination.
- Built to handle multiple users concurrently.

Persistent Data Storage (MongoDB + Mongoose)

- Stores all essential user data:
    - Profiles
    - Languages
    - Chat metadata
    - Friend relationships
    - Theme preferences
- Mongoose schemas enforce structure and data integrity.

IMPLEMENTATION DETAILS

Project Setup & Environment Configuration

- Created a full-stack project structure with separate frontend and backend folders.

- Initialized Node.js backend using npm init and installed core packages (Express.js, Mongoose, dotenv, Stream SDK, CORS).
- Created the React frontend using create-react-app and installed libraries like Zustand, Tailwind CSS, Simple-Peer, and Stream SDK-client.
- Configured environment variables for sensitive data (MongoDB URI, JWT secret, STUN/TURN credentials).

MongoDB Database Setup

- Set up MongoDB as the primary database for storing user profiles and preferences.
- Designed Mongoose schemas for:
  - User details
  - Language preferences
  - Friends list
  - Chat metadata (optional)
- Established a reliable connection to MongoDB using mongoose.connect().

Backend API Development (Node.js + Express)

- Developed RESTful APIs for:
  - User registration
  - Login and JWT authentication
  - Updating user language preferences
  - Sending and receiving friend requests
  - Notification retrieval
- Implemented middleware for authentication and input validation.
- Tested all backend routes with Postman.

Frontend UI Development (React + Tailwind CSS)

- Built interactive pages including:
  - Signup, Login, Onboarding
  - Home screen (friend list + suggestions)
  - Notification panel
  - Chat interface
  - Video calling page

- Implemented Zustand for managing global state (active call state, user data, media streams).
- Ensured responsive design across different devices.

Real-Time Messaging Integration (Stream SDK)

- Connected the backend to Stream SDK to enable WebSocket-based real-time events.
- Implemented messaging channels to deliver instant chat updates.
- Configured join-room, send-message, and presence events.
- Linked the frontend to Stream SDK for subscribing to channels and rendering messages in real time.

WebRTC Video Calling Implementation (Simple-Peer + WebRTC APIs)

- Integrated WebRTC for peer-to-peer video/audio communication.

- Used navigator.mediaDevices.getUserMedia() to access the camera and microphone.
- Used Simple-Peer to simplify offer/answer creation and ICE handling.
- Implemented signaling over Stream SDK to exchange:
  - SDP offers
  - SDP answers
  - ICE candidates
- Successfully established a low-latency video call between two peers.

Signaling Workflow Implementation

- Backend acts as a mediator for call initiation events.
- Stream SDK transports signaling data using WebSockets.
- Ensured proper synchronization for:

  - Call initiation
  - Answering
  - Disconnect events
  - Re-negotiation (if needed)

Theme & Personalization System

- Added 32 pre-defined Tailwind theme configurations.
- Saved user-selected themes in MongoDB.
- Applied dynamic theming using React context/state.

System Testing & Debugging

- Tested all features using multiple browser instances and devices.
- Verified:

  - Real-time chat performance
  - Video call quality
  - User authentication
  - Friend request workflow
  - Notifications
  - UI responsiveness

- Logged WebRTC events and ICE candidate flows to debug connection issues.

Deployment (Optional Step)

- Generated production build for React frontend.
- Served frontend and backend through Express in production mode.
- Configured environment variables for live deployment.
- Ensured database and Stream SDK credentials are secured.

RESULT AND CONCLUSION

The Streamify platform was successfully implemented as a real-time language exchange application that integrates secure authentication, language-based onboarding, instant messaging, and peer-to-peer video calling into a cohesive system. Testing demonstrated that user registration and login functions operated reliably with JWT authentication, and language preferences were stored accurately for effective partner matching. The friend request system and notifications worked in real time, enabling smooth social interaction between users. Stream SDK ensured instant message delivery with minimal latency, while WebRTC provided stable, low-lag audio and video communication across devices. The frontend remained responsive and customizable through 32 user-selectable themes, and the backend handled API requests and database operations efficiently using Node.js and MongoDB. Overall, the system performed as intended and delivered a seamless, interactive experience suitable for language practice.

In conclusion, Streamify achieves its goal of creating a modern, user-centered platform for language exchange by combining real-time chat, video calling, and social connectivity within a scalable MERN-based architecture. The integration of WebRTC and Stream SDK enables smooth, low-latency communication, while features such as onboarding, friend management, and theme customization enhance the overall usability and engagement of the platform. Despite challenges related to signaling flow, NAT traversal, and multi-client testing, the project successfully delivers a functional and reliable environment for practicing spoken languages. The work establishes a strong foundation for future enhancements, such as AI-driven language feedback, group calls, mobile applications, and improved analytics, demonstrating the system’s potential to evolve into a comprehensive learning and communication platform.