Design and Implementation of a Real-Time Communication System with Integrated Interactive Games

[1]Bayasani Shashank Reddy, [2]Dhannarapu Aashritha,[3]Gatla Vipul, [4]K.Ravikanth, [5]S.Mahesh

[1,2,3,4,5] Department of Computer Science and Engineering,School of Engineering, Aurora Deemed to be University, Hyderabad, India

DOI: https://doi.org/10.51244/IJRSI.2026.1303000230

Received: 27 March 2026; Accepted: 02 April 2026; Published: 20 April 2026

ABSTRACT

The growing requirements for integrated platforms of various communication types and customer interaction have increased the need for creatively designed and developed high-concurrent real-time communication (RTC) systems to merge disparate mobile/web environments providing integrative functions. By unifying high-speed message transmission, peer-to-peer audio/video calling, and social gaming as part of a large interconnected ecosystem, this RTC system will also reduce the fragmentation of communication software platforms. The real-time communication architecture includes Spring Boot and WebRTC as well as WebSockets as an alternative communication protocol that enables low-latency cross-device connectivity with real-time synchronization. The results of testing the prototype demonstrated less than 200 milliseconds of synchronization latency and a call stability rate of over 95%, indicating that the system will enable reliable connectivity between users using heterogeneous endpoints. The RTC system is positioned to create a "Super Application."

Keywords—Real-Time Communication, WebRTC, WebSocket, Social Gaming, Spring Boot, Super App Architecture.

INTRODUCTION

The way we talk to each other has changed a lot because of the internet. Now we can send messages. Talk to people in real time no matter where they are in the world. We have lots of ways to communicate, like messaging apps and video calls that let us talk send messages and see each other. But with all these new ways to talk lots of communication apps are missing something. They are not very fun. People often use apps to play games with their friends while they are talking.

If we could play games inside our communication apps it would be more fun to talk to people. This project is about making a communication system that lets people talk and play games at the same time. The Real-Time Communication System with Integrated Interactive Games lets people send messages talk in groups and make video calls and also play games with each other all, in one place.

The main goals of the system are:

• It will let people send messages and talk to each other in real-time

• It will let people make voice and video calls

• It will have games that lots of people can play together on the system

• It will have security so users can feel safe when they log in

• It will be able to handle a lot of users at the same time

The system wants to be a place where people can talk to each other and have fun at the same time.

LITERATURE SURVEY

Voice and video call platforms like Zoom and Skype are good for meetings and working together. They do not have fun features for entertainment [9]. On the other hand, platforms like Discord are very popular among gamers [16]. However, gamers often have to use external games instead of playing games right inside the chat. Recently, some studies have looked into using WebSocket technology for communication systems that happen in real-time [17]. WebSocket helps keep the connection between users and servers open [8], [15]. This reduces delays and makes messages arrive faster. Not many studies have tried to combine chat and game features into one app. Our system tries to fill this gap by bringing real-time chat services and fun interactive games together. The system has real-time communication services and built-in interactive games. It combines communication and gaming modules within an application platform.

Recent advancements in Real-Time Communication have focused on low-latency protocols like WebRTC for media [1], [5], [11] and WebSocket for full-duplex messaging [2], [17]. While platforms like Discord have popularized the integration of social activities, many systems still treat games as external entities rather than core components of the communication pipeline [3], [14], [16].

Our system design builds on recent research in real-time communication.

Paper [1] analyses WebRTC, focusing on issues like getting media streams through tricky NAT setups. It explains some smart ways to get around signalling headaches, so peer-to-peer data channels actually work. Paper [2] gets into real-time data processing architecture using Spring Boot with WebSocket. The research dives into full-duplex communication tweaks that really help keep latency low, even when tons of chat sessions hit the system at once. Paper [3] shifts gears and looks at what happens to users, both mentally and structurally, when gaming features are mixed in. Turns out, building gamification right into a main app (web or mobile) bumps up engagement, keeps people coming back, and makes them stick around longer. For big messaging platforms, Paper [4] lays out how microservices architecture can keep tons of connections stable. The authors share some solid strategies to make sure the system scales up, while still routing data smoothly during heavy traffic. Paper [5] dives into how WebRTC holds up over 5G, looking at Quality of Service metrics. The experiments show WebRTC pulls off ultra-low latency collaboration and handles live communication demands really well. Paper [13] zeroes in on signalling for peer-to-peer apps. It demonstrates that using WebSockets for signalling drastically speeds up WebRTC connection setup—much faster than older session protocols. Paper [14] explores the idea of running multiplayer games over peer-to-peer networks. The study shows you can sync game state with WebRTC, letting players interact directly without needing a central game server. Paper [17] compares real-time polling methods, ultimately proving that WebSockets leave old-school HTTP Long Polling in the dust. The study gives a clear reason to go with WebSockets for two-way user communication. Finally, Paper [19] points out that tapping into WebRTC's raw data channels gives complex apps a serious edge. By letting data flow directly, you get top efficiency without slowing down features.

Author(s)	Title	Year	Key Contribution / Advantage
H. Mahmoud, R. Azaria	WebRTC Architecture, Use Cases, and Challenges	2024	Detailed P2P signalling and media path traversal
V. Gupta, M. Sharma	Real-Time Data Processing with Spring Boot and WebSockets	2023	Full-duplex communication optimization for low latency
R. Singh, S. Verma	Impact of Integrated Social Gaming on Engagement	2023	Analysis of gamification in increasing app stickiness
J. White, L. Peterson	Designing a Scalable Real-Time Chat System	2024	Microservices and distributed DB management for scalability

S. Seol, H. Park	WebRTC over 5G: Remote Collaboration QoS	2023	5G performance benchmarks for ultra-low latency media
P. Pardhi, P. Sons are	Security Analysis of P2P RTC using WebRTC	2023	End-to-end encryption frameworks (DTLS/SRTP)

METHODOLOGY

The development follows an iterative, modular, and user-centric approach structured into four distinct phases:

Phase 1:

Requirement Analysis: Defining the "Super App" vision and identifying key social games for integration.

Phase 2:

Infrastructure Development: Implementing the Spring Boot backend, configuring STOMP brokers, and JWT security.

Phase 3:

Gaming Ecosystem Integration: Developing modular JS-based game engines synchronized via WebSocket topics.

Phase 4:

Validation and Optimization: Stress testing signalling handshakes and refining the Glass morphism UI for accessibility.

Proposed System

The new system brings together communication and interactive gaming to create a digital platform that people can use.

The system is made up of parts.

A. User Authentication Module

This part is, in charge of getting users registered and logged in. It uses something called JSON Web Token to make sure that only the right people can get in.

B. Messaging Module

The messaging part lets users send each text messages right away. It uses a thing called WebSocket to get the messages to each other

C. Voice and Video Calling Module

This part lets users talk to each other using audio and video calls. It uses protocols to make these calls happen in real time.

D. Interactive Game Module

The platform has games that let users play with their friends right inside the communication interface. This means that people can play games with each other without having to leave the communication platform.

E. Notification System

Users get notifications when they get messages when someone is calling them and when they are invited to play a game. The system sends these notifications to keep users informed about what's happening on the platform.

System Architecture

A. High-Level System Architecture

The diagram shows how user devices, the main backend and other services work together.

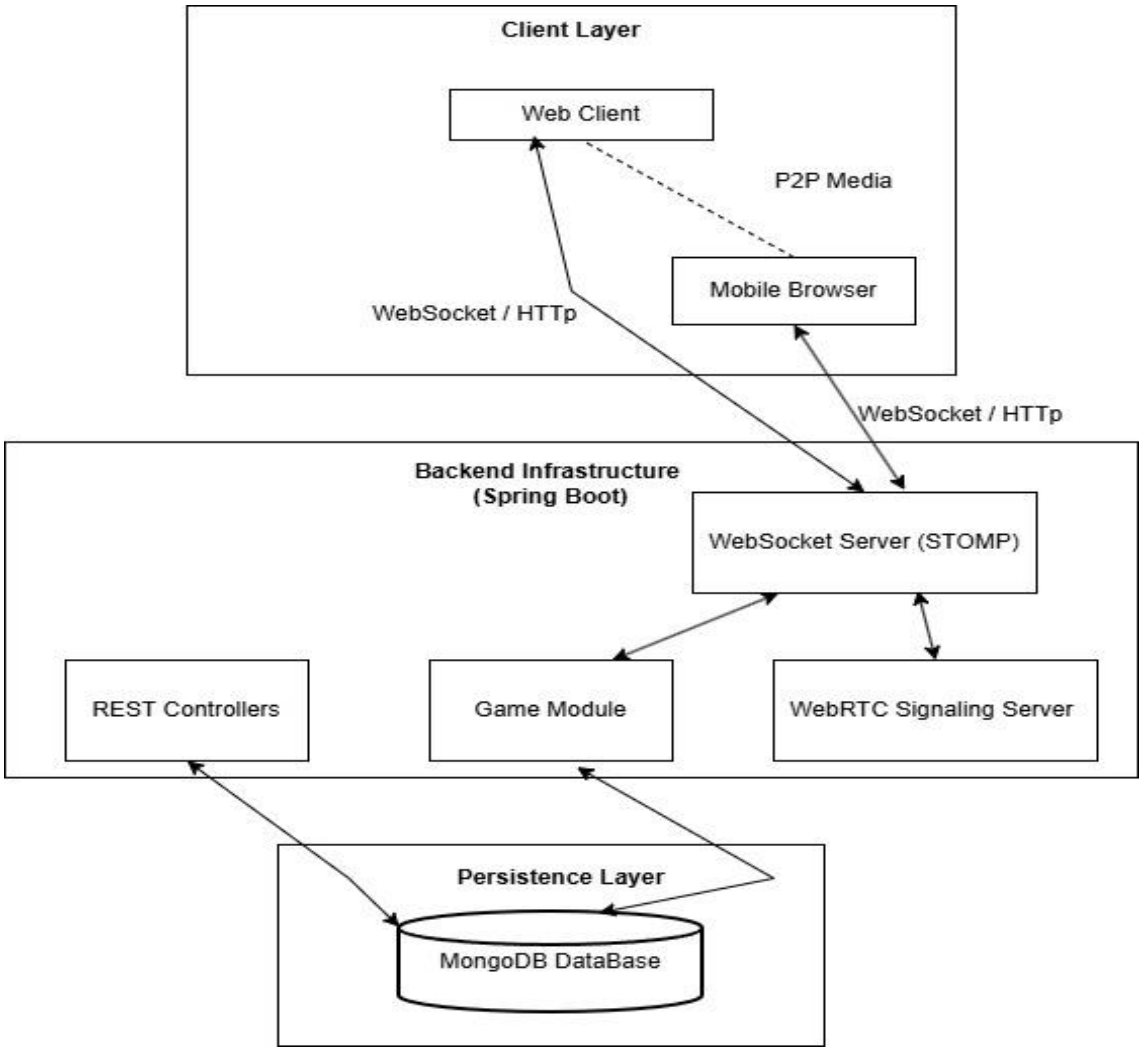

Fig. 1 System Architecture

- Zone 1: Client Part (Left)
- Box A: "Web User Interface". This is what the user sees.
- Box B: "Messaging Channel". This is how messages are sent.
- Box C: "Audio/Video Handler C: "Audio/Video Handler". This handles the audio and video.
- Zone 2: Server Part (Middle)
- Box D: "Security Check". This checks who is allowed to use the system.
- Box E: "Message Router". This sends messages to the place.

- Box F: "Call Setup". This sets up audio and video calls.
- Zone 3: Data Part (Right)
- Icon G: "Data Storage". This stores user information and chat history.
- Connections:
- Arrow from User Interface to Security Check (Login Request).
- Arrows going both ways between Messaging Channel and Message Router.
- Arrows going both ways between Audio/Video Handler and Call Setup.
- Arrow from Server to Data Storage (Saving and Reading Data).

The system has three parts: the part that users see the part that handles the work and the part that stores the data.

1. Communication Part:

- Messaging: This handles text messages, game moves and call setup.
- Audio/Video: This sends audio and video directly between users.

2. Security Part:

- User Verification: This checks if a user is allowed to use the system.

3. Data Layer:

- Data Storage: This stores user information, chat history and other data.

B. System Process

The operational flow of the system is designed for maximum efficiency and security

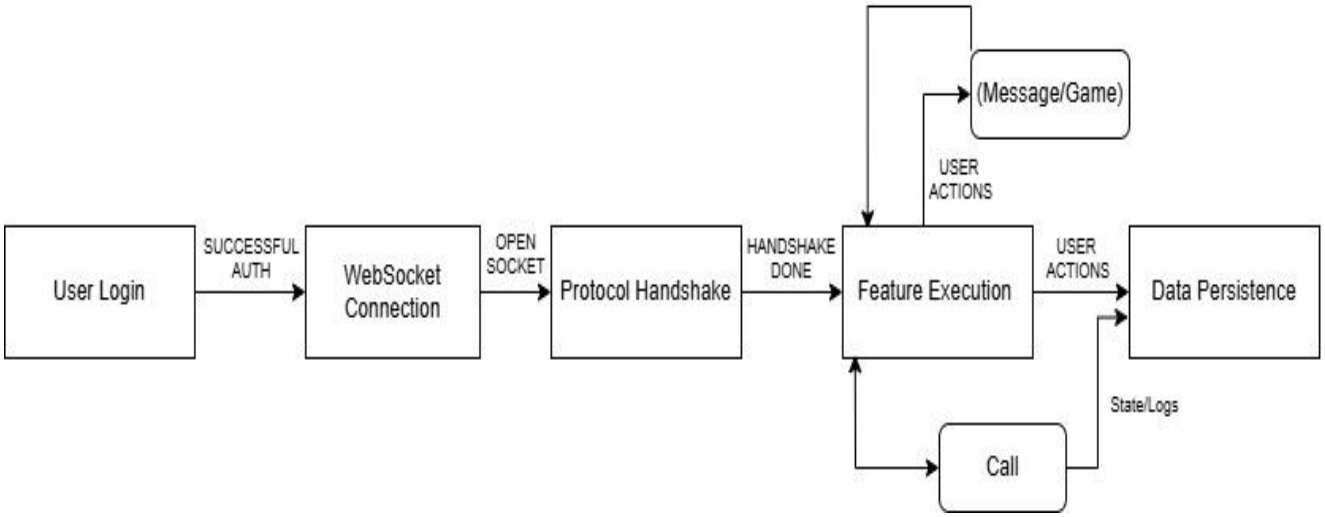

Fig. 2. Operational Workflow of the Messaging and Game Synchronization System

C. Use Case Analysis

The system caters to various user interactions, mapped through standard UML Use Case diagrams

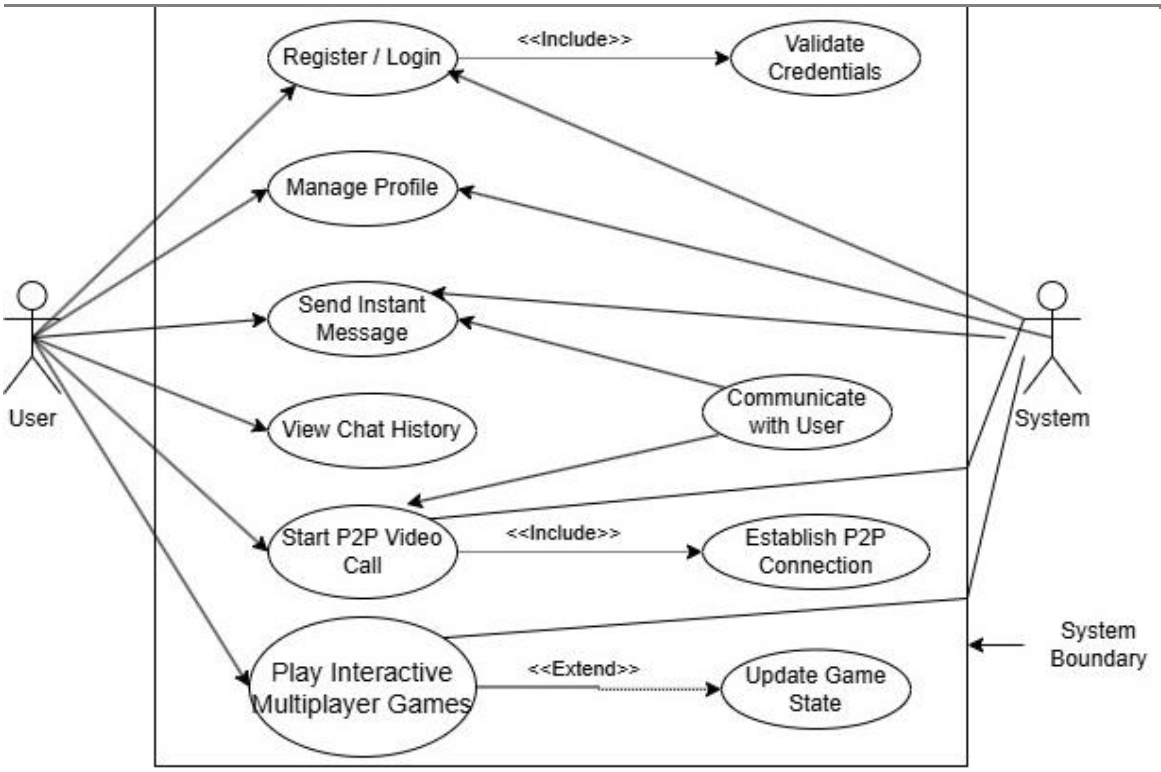

Fig. 3. Use Case Diagram of the Application Components

D. WebSocket Messaging Sequence

The system utilizes the STOMP protocol over WebSockets for low-latency message delivery.

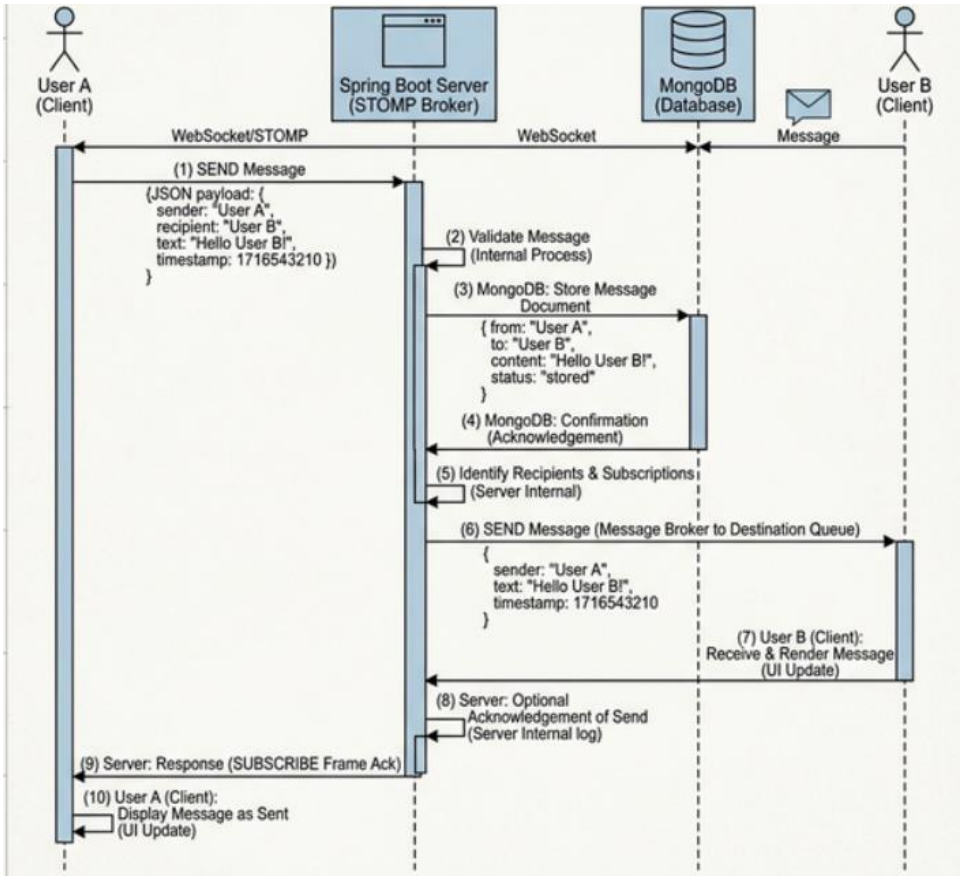

Fig. 4. WebSocket Communication Sequence Diagram for Messaging

E. WebRTC P2P Signalling Flow

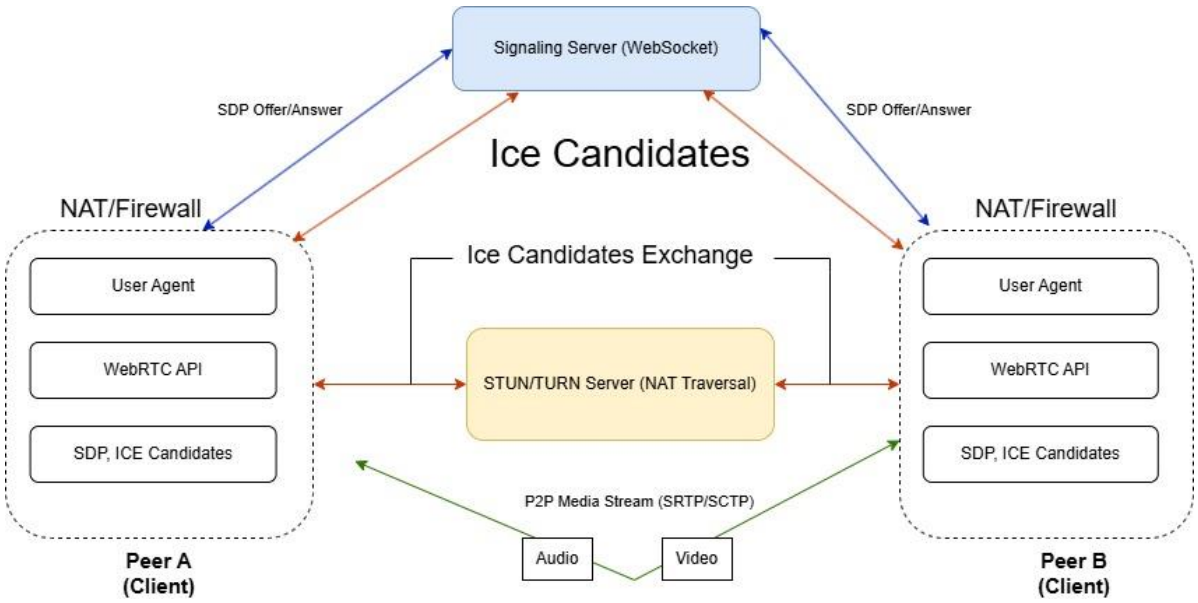

Fig. 5. WebRTC Peer-to-Peer Communication and Signalling Flow

F. Database Design

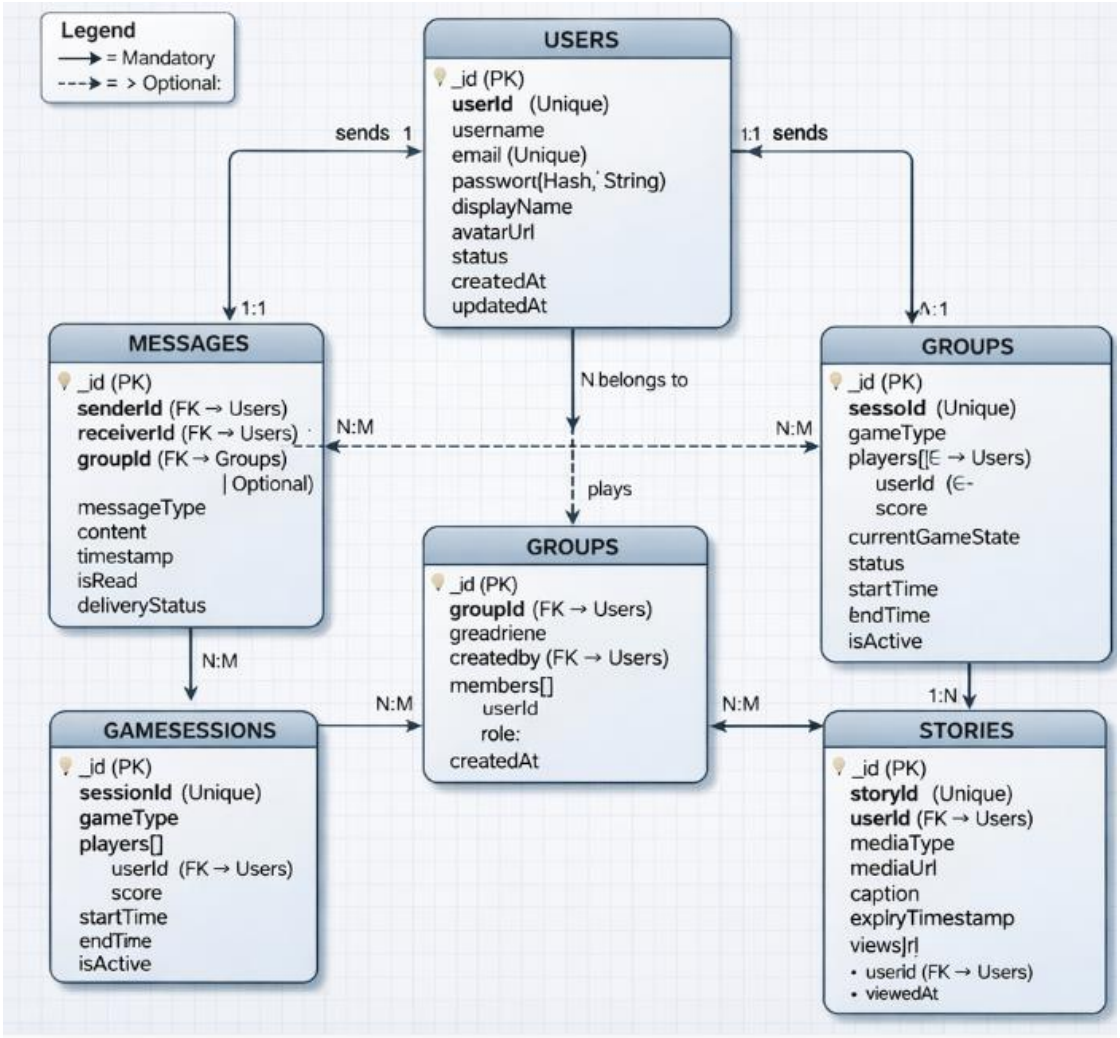

Fig. 6. Database ER Diagram showing Collection Relationships

Implementation

The system we made uses web technologies to help people talk to each other in real time and play games together. We used Java Spring Boot to make the backend. We made the part that people see in their browsers using web stuff like HTML and JavaScript. Our system puts together messaging, secure login and games in one place.

A. Development Environment

We made the backend with Java Spring Boot because it is good for making web applications. The frontend is what people see. We made it with HTML and JavaScript and CSS so it is interactive. We used something called WebSocket so people can talk to each other in real-time. We also used something on top of WebSocket called STOMP so messages make sense.

We used MongoDB to store all our data because it's flexible and can store lots of different things. It stores information about users and messages and other important stuff.

B. Authentication and Security

We wanted to make sure only the right people can use our system. So, we used something called JSON Web Token to help with login. When someone logs in, we check who they are. If they are okay, we give them a token that they use to prove who they are. This way we know it is really them.

C. Real-Time Messaging Module

Our messaging system uses WebSocket. When someone sends a message their browser sends it to our server. Then we send it to everyone else. This way it happens fast. Everyone gets the message.

D. Interactive Game Module

We also have some games that people can play together. When someone does something in a game their browser tells our server. Then we tell all the other players. Our server keeps track of what's happening in the game so everyone sees the same thing.

E. Database Implementation

We used MongoDB to store all our data. It has all the information about users and what they say to each other and what they do in games. Our server talks, to MongoDB to get and store data.

RESULT AND DISCUSSION

The system we came up with was tested to see how well it works when people are talking to each other in real-time and playing games together. We did a lot of tests to see how long it takes to send messages how many people can use it at the time how often calls are successful and how well games work. The tests show that the system works well even when a lot of people are using it at the same time.

TABLE 1: System Performance Results

Feature	Result
Message latency	< 200 Ms
Maximum concurrent users tested	50
Call success rate	95%
Game response time	< 300 Ms

The system we developed does a job of keeping message latency low and provides very good audio and video communication using WebRTC. The part of the system that lets people play games together also works well and people can interact with each other quickly. These results show that the system we developed is very good for platforms that need to handle real time communication, including messaging, video calls and games. The system is really good for real-time communication and the system is good, for gaming scenarios.

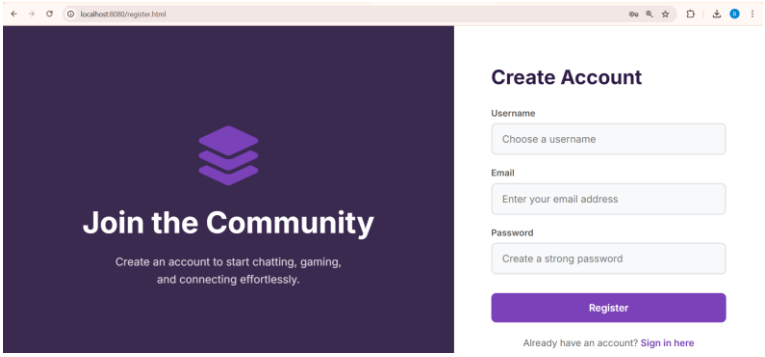

Fig. 7. Secure User Registration Interface

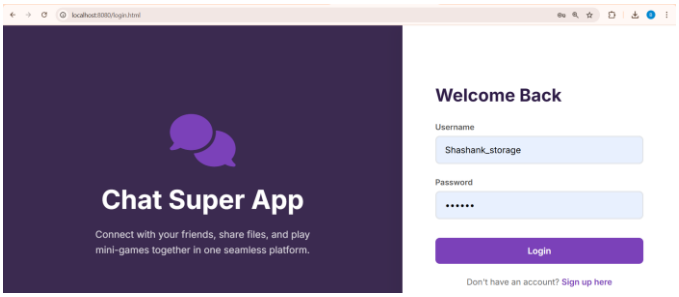

Fig. 8. Authentication and Login Portal

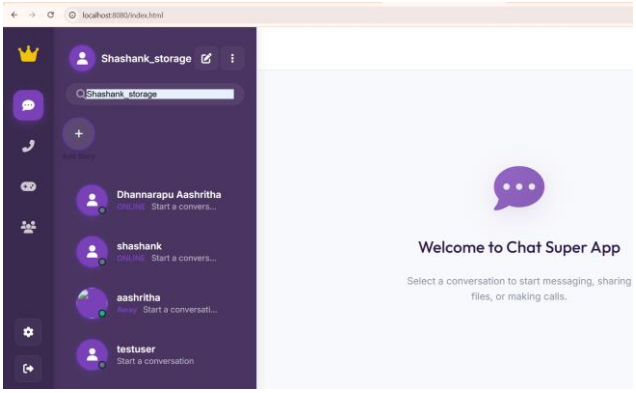

Fig. 9. Refined Contacts Dashboard with High-Contrast UI

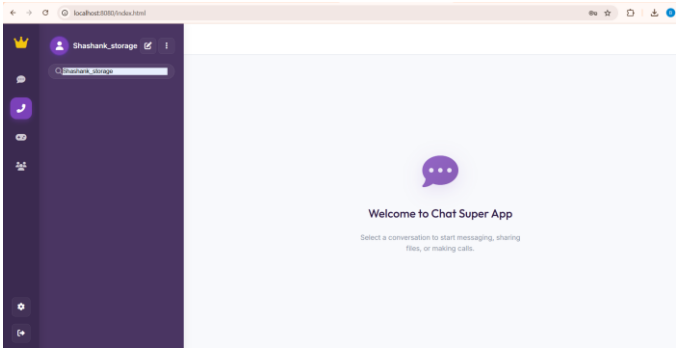

Fig. 10. Real-Time Communication Logs (Calls Interface)

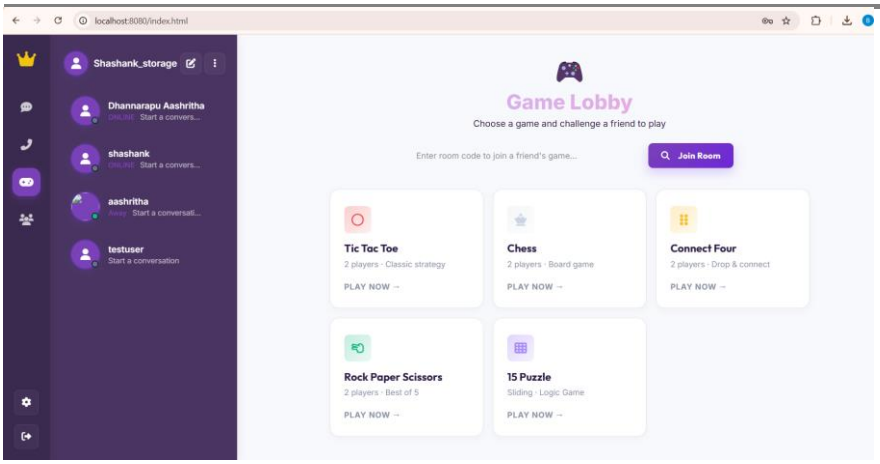

Fig. 11. Integrated Social Gaming Lobby

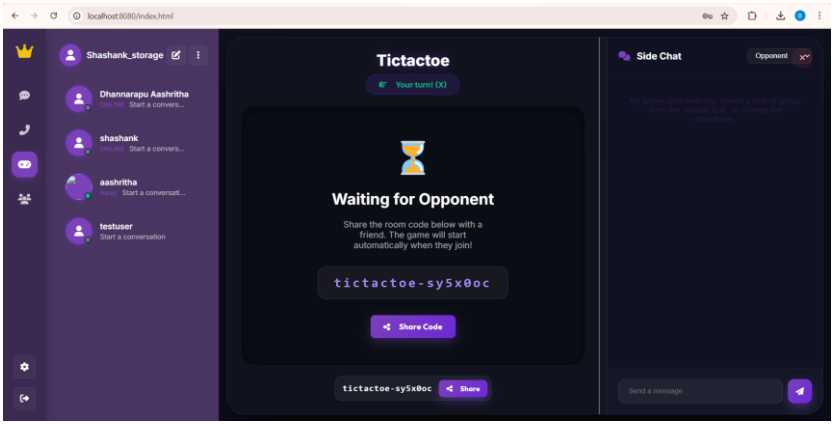

Fig. 12. Tic-Tac-Toe Game Room: Waiting for Opponent State

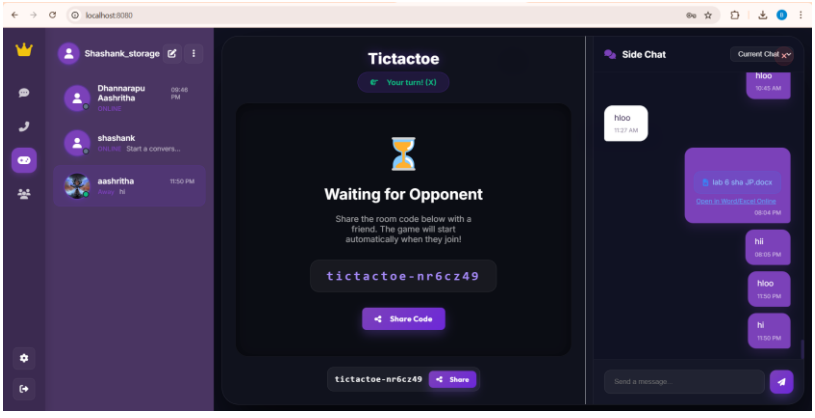

Fig. 13. Side Chat Integration within Active Game Session

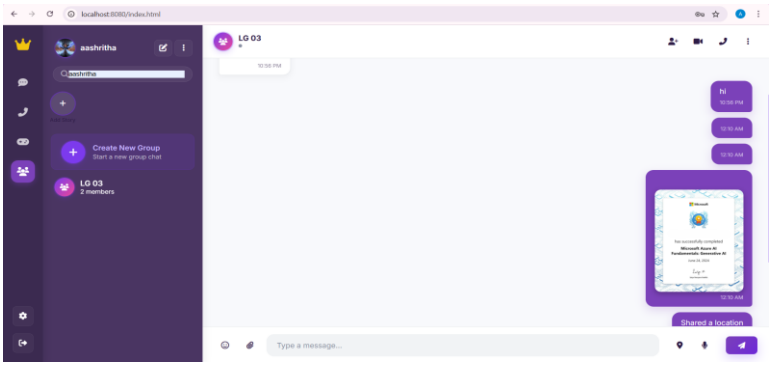

Fig. 14. Group Coordination Module with Dynamic Messaging

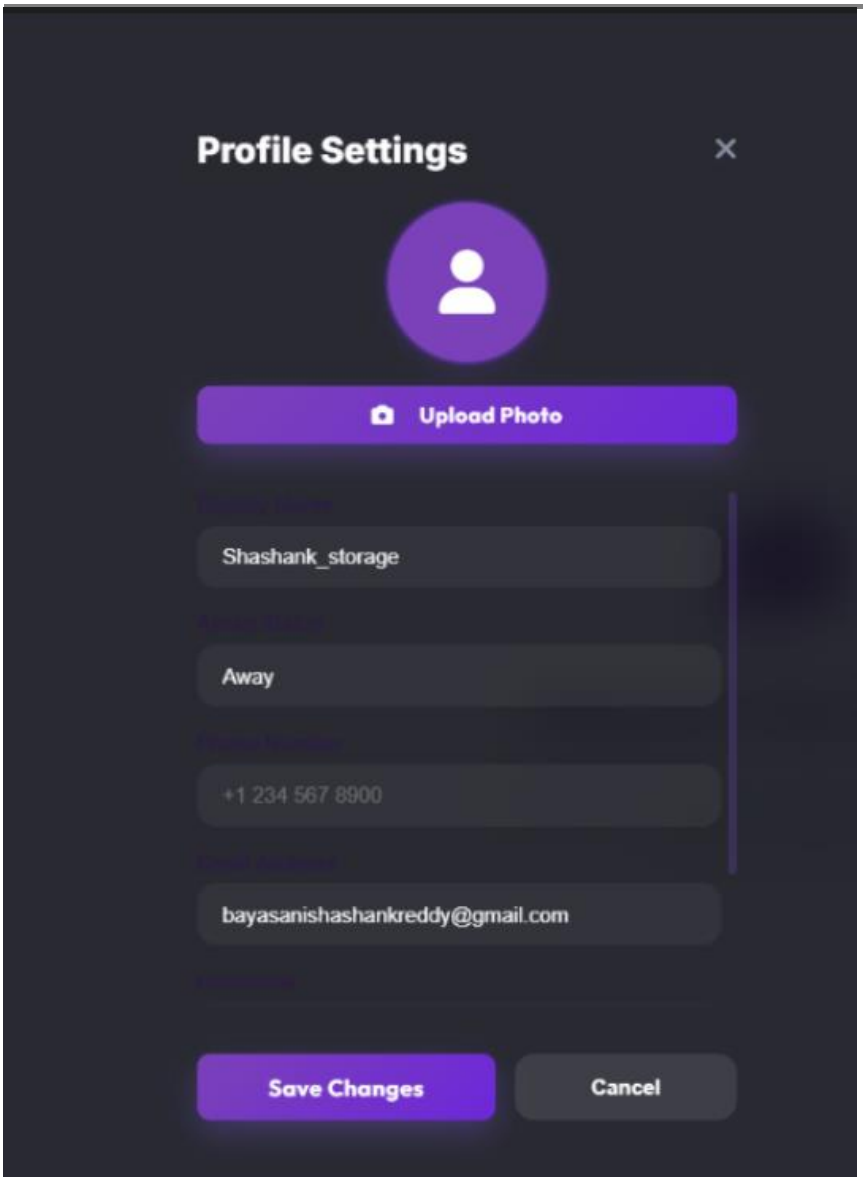

Fig. 15. Profile Settings and User Metadata Management

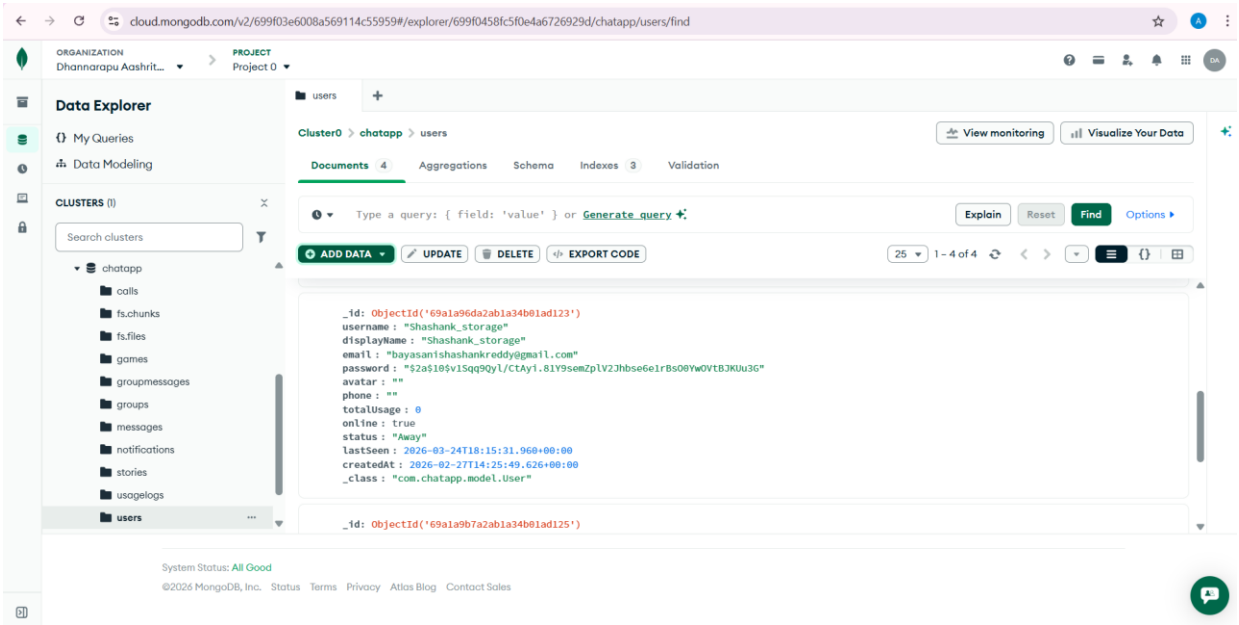

Fig. 16. MongoDB Atlas Data Explorer: Users Collection and Document Structure

CONCLUSION

This paper talks about a communication system with games. The system brings together messaging, video calls and games in one easy-to-use app. It uses a design called Glass morphism for its look.

The app uses WebSocket and WebRTC to send messages and make calls in real-time. It works well with messages sent in than 200 milliseconds. Adding games makes the app more fun. Keeps users engaged. The system shows that it is possible to make an app that lets people talk to each other and play games at the time. This could be the future of apps that do things, like a "Super App".

ACKNOWLEDGMENT

Heartfelt thanks to Aurora University and the CSE department for providing the resources and guidance required to complete this project.

REFERENCES

1. H. Mahmoud and R. Abozariba, "WebRTC Architecture and Challenges", IJSRET, 2024.
2. V. Gupta and M. Sharma, "Real-Time Data Processing with Spring Boot", IJISEM, 2023.
3. R. Singh and S. Verma, "Impact of Social Gaming on Engagement", IJFMR, 2023.
4. J. White and L. Peterson, "Designing Scalable Chat Systems", Codestack, 2024.
5. S. Seol and H. Park, "WebRTC over 5G QoS Study", SpringerLink, 2023, DOI: 10.1007/s10922-023-09756-x.
6. P. Pardhi and P. Sonsare, "Security Analysis of WebRTC", IJIRSET, 2023.
7. A. S. Tanenbaum and D. J. Wetherall, "Computer Networks", Pearson Education, 2011.
8. I. Fette and A. Melnikov, "The WebSocket Protocol", RFC 6455, 2011.
9. Spring Framework Research, "Spring Boot 3.x Documentation", 2024.
10. MongoDB Engineering, "MongoDB Database Documentation", 2024.
11. L. Loreto and S. P. Romano, "Real-Time Communication with WebRTC", O'Reilly, 2014.
12. J. Johnston et al., "WebRTC: Web Real-Time Communication", IEEE Communications, 2013.
13. V. Beltran and J. Pradilla, "WebRTC signaling mechanism based on WebSockets", IEEE ICME, 2014.
14. R. Ennals et al., "Peer-to-peer multiplayer games on WebRTC", IEEE Consumer Electronics, 2015.
15. S. Loreto and P. S. Romano, "The WebSocket Protocol Analysis", IEEE Internet Computing, 2012.
16. J. Paschmann and R. Meyer, "Driving Engagement through Gamification", Journal of Marketing Research, 2024.
17. A. Petrova and D. Ivanova, "WebSocket vs HTTP Long Polling", IJNRD, 2023.
18. S. Mahesh and B. Reddy, "Collaborative Process Mining Visualization", ResearchGate, 2024.
19. D. Walls, "Unlocking the Power of WebRTC", IEEE Computer, 2021.
20. N. Josuttis, "SOA in Practice: Distributed System Design", O'Reilly, 2007.