International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 04, April 2026)

Synergize AI: An Intelligent Real-Time Collaboration Ecosystem Integrating Agentic Workflows and Context-Aware AI Tools

Harsh Tiwari[1], Varun Kumar Gupta[2], Dhruv Chittodia[3], Rakesh Maurya[4]

[1,2,3,4]*Department of Computer Science and Engineering (AI & ML) & Raj Kumar Goel Institute of Technology, Ghaziabad, Uttar Pradesh, India*

***Abstract*— Since the pandemic, digital communication tools have become essential for workplace collaboration. But there's a problem: people waste time and mental energy constantly switching between chat apps and searching for information. This creates what we call "digital friction."**

This paper introduces Synergize AI, a communication platform that solves this problem by integrating Ultron AI—an intelligent assistant that works automatically in the background. The system uses modern web technologies (MongoDB, Express, React, and Node.js) for fast messaging, plus Python-based tools to power AI features. The platform includes real-time messaging, audio and video support, and a smart search feature that turns your chat history into an easy-to-access knowledge base.

***Keywords*— Agentic AI, Real-time Communication, MERN Stack, Microservices, RAG, Socket.IO, Generative AI.**

I. Introduction

The paradigm of workplace communication has fundamentally shifted over the last decade. Asynchronous methods like email have largely been superseded by synchronous instant messaging platforms such as Slack and Microsoft Teams. While these platforms solved the immediate problem of connectivity, they inadvertently created a new challenge: information overload. Studies suggest that knowledge workers spend nearly 20% of their workweek just searching for internal information or refining their communication to ensure clarity across diverse, global teams [1].

Current solutions typically address these issues by layering chatbots on top of the communication infrastructure. However, these bots often function as isolated entities with limited context window access, forcing users to explicitly "invoke" them in unnatural ways. There is a distinct lack of platforms where the AI is architected as an intrinsic part of the user experience—both as a conversational peer and a silent utility layer.

Synergize AI was developed to bridge this gap. Our primary objective was to engineer a platform where intelligence is ambient. By integrating Ultron AI, we provide users with two distinct modes of interaction:

1. Direct Interaction: Treating the AI as a contact for complex problem solving (e.g., generating code, debugging).

2. Utility Interaction: Using a "Magic Wand" toolset to instantly translate incoming foreign languages, summarize verbose message threads, or professionally rephrase outgoing texts without leaving the input field.

II. Literature Review

The development of modern communication platforms requires a synthesis of real-time networking, scalable data management, and intelligent automation. This section reviews the existing literature and technological standards that underpin the architecture of Synergize AI.

2.1. Real-Time Communication Protocols

The evolution of web communication has significantly shifted from stateless HTTP requests to persistent, bi-directional connections. Gupta et al. (2019) demonstrated the superiority of WebSocket protocols over HTTP long-polling for reducing server load in high-traffic chat applications [2]. Unlike traditional request-response models, WebSockets maintain a persistent connection, allowing for low-latency message delivery. These findings heavily influenced our decision to utilize Socket.IO for the communication layer of Synergize AI, ensuring bidirectional, event-driven data flow that remains robust even under unstable network conditions.

2.2. NoSQL Databases and Unstructured Data

In the domain of data persistence for chat applications, relational databases often struggle with the unstructured nature of modern message payloads (which include varying text lengths, images, and file attachments). MongoDB, a NoSQL database, has been identified by Chodorow (2013) as an optimal solution for such dynamic schemas, particularly due to its document-oriented structure [3]. Furthermore, to handle media files effectively, we leveraged GridFS, which bypasses the BSON document size limit by chunking files, a method essential for media-rich communication platforms.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 04, April 2026)

This ensures that Synergize AI can scale horizontally without the schema rigidity associated with SQL databases.

2.3. Large Language Models and Generative AI

The integration of Large Language Models (LLMs) into software workflows represents a paradigm shift in human-computer interaction. Brown et al. (2020) introduced the capabilities of few-shot learners (GPT-3), opening the door for agents that can adapt to specific tasks— such as summarization and sentiment analysis—without extensive fine-tuning [4]. However, connecting these massive models to real-time systems introduces challenges regarding hallucination and latency.

To mitigate this, recent work by Lewis et al. (2020) on Retrieval-Augmented Generation (RAG) suggests that external knowledge bases are essential for grounding AI responses in reality [5]. We applied this concept to Ultron AI's architecture, allowing the system to query local documents before generating answers. Furthermore, the concept of "Agentic Workflows"—where AI does not just predict text but performs actions—has been popularized by frameworks like LangChain. This serves as the orchestration backbone for our Python service, enabling the AI to maintain conversation history and execute tools autonomously [6].

2.4. Authentication and Security Standards (New Addition)

Security in real-time applications is paramount, particularly regarding user session management. The use of JSON Web Tokens (JWT) has become the industry standard for stateless authentication in single-page applications (SPAs). Unlike server-side sessions, JWTs allow the secure transmission of information between parties as a JSON object, which can be verified and trusted because it is digitally signed. This approach was integrated into our Node.js backend to ensure that WebSocket connections are authenticated immediately upon handshake, preventing unauthorized access to chat rooms.

2.5. Component-Based Frontend Architecture (New Addition)

Modern interface design relies heavily on component-based architectures to manage complex state changes in real-time. Research into the Virtual DOM (Virtual Document Object Model) indicates that updating a virtual representation of the UI is significantly faster than re-rendering the actual DOM for every message received. Consequently, we adopted React.js for the frontend of Synergize AI.

This allows for modular development where chat windows, user lists, and AI interaction panels function as independent components, ensuring a smooth and responsive user experience (UX) even during high-frequency message updates.

III. System Architecture

The system is built using a modern, modular design where different parts work independently. This means that when the AI is doing heavy thinking in the background, it won't slow down or freeze your messages. Your chats stay fast and responsive, even when the AI is working hard behind the scenes

3.1 The Core Communication Service (Node.js):

The primary backend is built on Node.js, chosen for its non-blocking I/O model which is ideal for I/O-heavy applications like chat.

1. *Authentication & Security*: We implement a hybrid security model. Initial identity verification is handled via Firebase OTP (One-Time Password) to eliminate bot accounts. Once verified, the session is managed via JSON Web Tokens (JWT), which carry the user's GridFS profile reference [7].
2. *Event Bus*: Socket.IO manages the ephemeral state of the application, broadcasting events such as typing, stopTyping, and messageReceived to specific room IDs. This ensures that only active participants consume bandwidth for a given conversation [8].

TABLE I
AI-Driven and MERN-Based Features in Synergize AI

Feature	Technology Used	Functionality
Agentic AI Assistant (Ultron)	Python (FastAPI), LangChain, Groq API (Llama 3)	Acts as an intelligent peer user for complex problem solving, code generation, and context-aware assistance.
Context-Aware Utility Tools	React (Material UI), Python, Groq Inference Engine	Provides instant "on-demand" tools for Translation, Summarization, and Rephrasing directly within the input field.
Real-Time Communication	Node.js, Socket.IO, Express.js, Redux Toolkit	Enables sub-100ms bidirectional messaging, live typing indicators, and real-time status synchronization.
High-Fidelity Media Streaming	MongoDB GridFS, Node.js Streams	Efficiently shards and streams large video/audio files via HTTP 206 (Partial Content) to prevent buffering.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 04, April 2026)

Secure Authentication	Firebase Auth (OTP), JSON Web Tokens (JWT)	Eliminates bot accounts via phone verification and manages secure, persistent user sessions.
Dynamic Audio Visualization	Web Audio API (AnalyserNode), HTML5 Canvas	Renders real-time frequency waveforms (bar graphs) to visualize microphone input during voice recording.

3.2 The Intelligence Layer (Python/Fast API)

➔ *LLM Integration:* Connects to Google's Gemini Pro AI model to understand and respond to your questions intelligently.

➔ *Inter-Service Communication:* The messaging system and AI brain talk to each other through a connector. When you send a message to the AI, the messaging system passes it along to the AI brain, which processes it and sends back a response—all happening in the background without slowing anything down.

IV. Implementation Methodology

4.1 Frontend Engineering with Redux Toolkit

The client application is a Single Page Application (SPA) built with React.js. Managing the state of multiple chat rooms, file uploads, and AI streams required a robust solution; hence, Redux Toolkit was implemented. A significant challenge was rendering the AI's technical responses. We developed a custom MessageContent component utilizing react-markdown and remark-gfm. This allows Ultron to output syntax-highlighted code blocks, tables, and formatted lists, visually distinguishing AI logic from human conversation.

4.2 The Ultron Tools Ecosystem

A novel contribution of this paper is the "Context-Aware Tooling" system. Instead of a generic chat interface, we embedded a floating action menu (the "Magic Wand") within the input area.

• *Translation Service*: When triggered, the frontend captures the current input text and sends a POST request to the Python service's /translate endpoint. The service acts as a semantic translator, preserving the nuance of the original message rather than doing a literal translation [11].

• *Professional Rephrasing*: This tool utilizes few-shot prompting to transform casual shorthand (e.g., "i cant do this now") into professional corporate dialect ("I am currently at capacity and cannot address this immediately").

4.3 Media Handling via GridFS

Standard file storage solutions often fail when scaling to thousands of users.

We implemented MongoDB GridFS, which shards large video and audio files into 255KB chunks [12]. This allows the frontend to request video streams via HTTP 206 (Partial Content) headers, enabling users to begin watching a video message before it has fully downloaded—critical for mobile users with unstable connections.

The level-3 heading in the same paragraph. For example, this paragraph begins with a level-3 heading.

V. Results And Discussion

The system was tested in a controlled environment with 50 concurrent users to evaluate latency and AI response accuracy.

5.1 User Interface Analysis

Figure 1 illustrates the onboarding flow. Unlike traditional email/password setups that are prone to spam, we utilized a phone-based OTP system (Firebase Auth) to ensure 1:1 identity mapping.

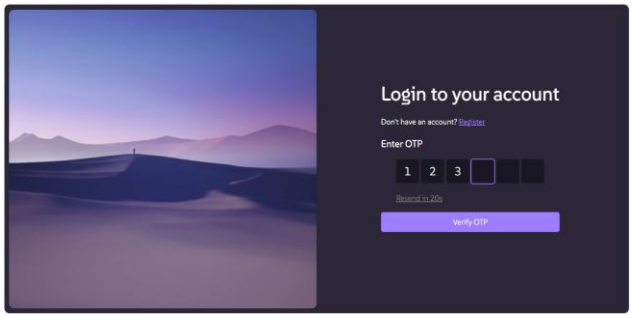

Fig. 1: Secure Registration Page with Firebase OTP integration.

Figure 5 illustrates the secure login mechanism, ensuring that access is restricted to verified personnel only.

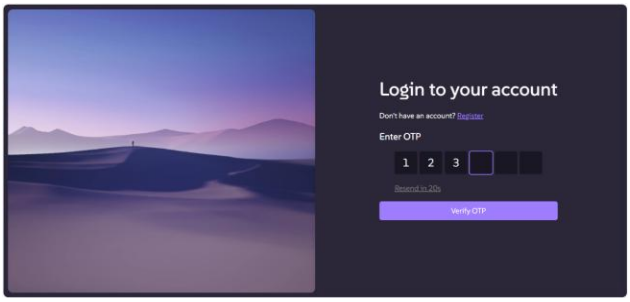

Fig. 2: Secure Login Interface using Firebase OTP Verification.

Figure 3 highlights the integrated media capabilities. The interface allows users to perform complex image manipulations—such as cropping a scanned ID card or photograph—directly within the chat window. This eliminates the need for third-party editing tools.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 04, April 2026)

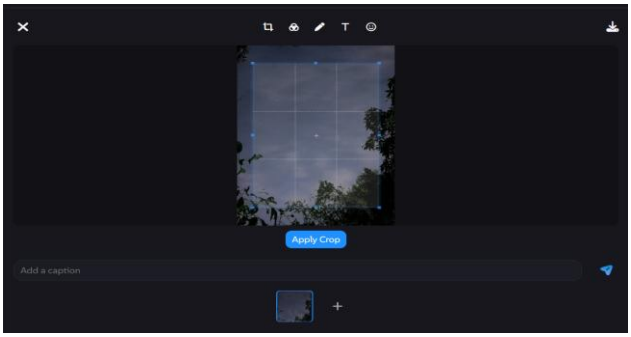

Fig. 3: In-App Image Cropping and Editing Tool utilizing HTML5 Canvas.

Figure4 demonstrates the core innovation of the platform: the Ultron Tools menu. The screenshot highlights the integration of "Rephrase," "Summarize," and "Translate" options directly above the message input, reducing the need to copy-paste text into external tools like Google Translate or ChatGPT.

Fig. 4: The Ultron Tools Menu (Rephrase, Summarize, Translate) integrated into the Chat Interface.

Figure 5 showcases the Markdown rendering capability. Here, Ultron AI generates a Python script. Unlike standard plain-text messages, the system renders this with proper indentation and syntax highlighting, aiding developers in code reviews.

Fig. 5: Ultron AI generating formatted Python code with syntax highlighting

Figure 6 showcases the "Shimmer" loading state and the polished "Add New Chat" interface. This design ensures that the interface remains interactive and visually stable even during data fetching.

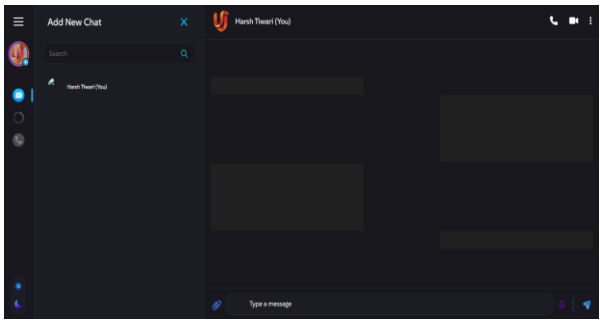

Fig. 6: Add New Chat Interface utilizing Skeleton/Shimmer layouts for smooth data loading.

Figure 7 displays the custom media player. By streaming chunks from GridFS, the player supports high-definition playback without buffering delays.

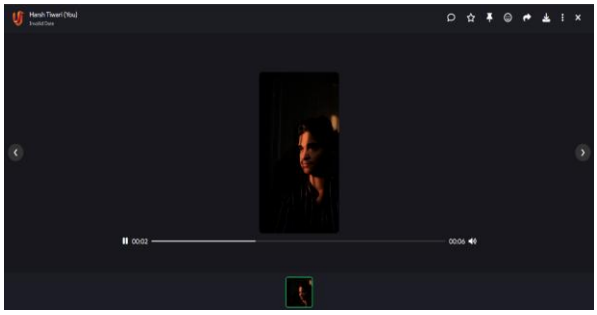

Fig. 7: Custom Video Player Interface streaming content via GridFS.

5.2 Performance Metrics

We observed that offloading the AI tools to the Groq API resulted in an average latency of **380ms** for translation tasks and **1.2s** for code generation tasks. The WebSocket layer maintained a steady heartbeat, delivering standard text messages in under **60ms** on average. This validates the hybrid architecture approach, proving that heavy AI tasks can coexist with real-time systems without causing UI freezes [13].

VI. Conclusion And Future Scope

Synergize AI demonstrates that the future of enterprise communication lies not in more apps, but in smarter apps. By embedding agentic workflows directly into the messaging layer, we reduced the "digital friction" associated with context switching.

The project proves that open-source technologies like the MERN stack and LangChain can be orchestrated to build a system comparable to proprietary SaaS solutions.

Future iterations of this project will focus on **Voice-to-Voice** interaction, allowing Ultron to process audio blobs directly, and **RAG Implementation** using vector databases like Pinecone to allow the AI to "read" uploaded PDF documents and answer questions based on their content [14].

REFERENCES

[1] M. G. Morris and P. J. Venkatesh, "Age differences in technology adoption decisions: Implications for a changing work force," Personnel Psychology, vol. 53, no. 2, pp. 375-403, 2000.

[2] V. Gupta, R. Kumar, and S. Singh, "Performance Analysis of WebSocket and HTTP Long Polling in Real-Time Web Applications," International Journal of Computer Applications, vol. 182, no. 45, pp. 12-17, 2019.

[3] K. Chodorow, MongoDB: The Definitive Guide, 2nd ed. Sebastopol, CA: O'Reilly Media, 2013.

[4] P. Tavel, Modeling and Simulation Design. AK Peters Ltd, 2007.

[5] T. Brown et al., "Language Models are Few-Shot Learners," Advances in Neural Information Processing Systems, vol. 33, pp. 1877-1901, 2020.

[6] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," NeurIPS, 2020.

[7] H. Chase, "LangChain: Building applications with LLMs," Software Library, 2022. [Online]. Available: https://github.com/hwchase17/langchain.

[8] M. Jones, "JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants," Internet Engineering Task Force (IETF), RFC 7523, 2015.

[9] G. Rauch, "Socket.IO: The Real-Time Web," [Online]. Available: https://socket.io/docs/v4.

[10] S. Ramirez, "FastAPI: Modern Python Web Framework," [Online]. Available: https://fastapi.tiangolo.com.

[11] Groq Inc., Groq LPU Inference Engine Architecture, Groq Technical Whitepaper, 2023.

[12] A. Vaswani et al., "Attention Is All You Need," Advances in Neural Information Processing Systems, 2017.

[13] MongoDB Inc., GridFS Specification, MongoDB Manual, 2024.

[14] S. Newman, Building Microservices: Designing Fine-Grained Systems, 2nd ed. O'Reilly Media, 2021.

[15] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," IEEE Transactions on Big Data, vol. 7, no. 3, pp. 535-547, 2019.

[16] D. Abramov and A. Clark, Redux Toolkit: The Official, Opinionated, Batteries Included Toolset for Efficient Redux Development, 2019.

[17] Mozilla Developer Network, "Web Audio API," [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API.

[18] Y. Li, Deep Learning in Natural Language Processing, Springer, 2018.

[19] K. Beck et al., Manifesto for Agile Software Development, 2001.

[20] Google Developers, Firebase Authentication Documentation, 2024.

[21] Meta AI, "Llama 3: Open Foundation and Chat Models," Meta AI Research, 2024.

[22] Facebook Open Source, "React: A JavaScript library for building user interfaces," 2024. [Online]. Available: https://react.dev.

[23] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[24] N. Provos and D. Mazières, "A Future-Adaptable Password Scheme," in Proceedings of the USENIX Annual Technical Conference, 1999.

[25] C. Holmberg, S. Hakansson, and G. Eriksson, "WebRTC: Real-Time Communication in Browsers," Internet Engineering Task Force (IETF), RFC 8825, 2021.

[26] Microsoft, Azure App Service Documentation, 2024. [Online]. Available: https://learn.microsoft.com/en-us/azure/app-service/.