Eduvest – Journal of Universal Studies
Volume 6 Number 9, September, 2026
p- ISSN 2775-3735- e-ISSN 2775-3727

Implementation of a Real-Time Chat System Based on the MERN Stack and Socket.io with Natural Language Processing Integration for Automatic Sentiment Analysis

Tirta Patria Nandhika*, Adi Nugroho
Universitas Kristen Satya Wacana, Indonesia
Email: 672022208@student.uksw.edu*, adi.nugroho.sujarwo@gmail.com

Keywords
MERN Stack; Socket.io; Natural Language Processing; Sentiment Analysis; Lexicon-based.

abstract
Digital communication through chat platforms has become a primary medium for consultation services; however, the absence of non-verbal cues makes early detection of users' emotional states challenging. This study implemented a real-time chat system based on the MERN Stack (MongoDB, Express.js, React.js, and Node.js) integrated with a lexicon-based Natural Language Processing (NLP) module to automatically detect message sentiment and risk levels. The system utilized Socket.io for bidirectional communication, with the NLP module positioned within the Controller layer and executed inline in the sendMessage event handler without requiring external services or message queues. Functional testing of the analyzeText() function across ten scenarios showed that all risk-level classifications matched the expected results. Sentiment classification achieved 80% accuracy, with two misclassifications attributed to the inherent limitations of lexicon-based matching in handling negation and substring ambiguity. All eleven Socket.io communication scenarios and seven REST API endpoints successfully passed testing. The MVC pattern within the MERN Stack demonstrated sufficient flexibility to accommodate an inline NLP module without requiring modifications to the Model or View layers. Furthermore, the tiered priority logic, which placed risky-word detection above sentiment scoring, maintained system reliability in critical cases despite occasional sentiment classification errors.

INTRODUCTION

Digital communication has become the backbone of modern interactions in both personal and professional contexts, with chat-based interactions rapidly developing as a primary medium for consultation and collaboration services (Andzani & Irwansyah, 2023). The speed of information exchange is one of the main reasons instant messaging has transformed communication patterns (Widjaja, 2025). Although real-time chat interaction provides convenience, significant challenges remain in understanding users' emotional nuances conveyed through text messages. The absence of non-verbal cues, such as voice intonation and facial expressions, often makes it difficult to interpret the actual intent behind written messages (Widjaja, 2025). This limitation may result in miscommunication and missed opportunities to respond to users' emotional states at an early stage, while manual sentiment analysis of large-scale conversations is inefficient and susceptible to subjectivity (Pratama et al., 2025).

To address these challenges, the MERN Stack (MongoDB, Express.js, React.js, and Node.js) provides a robust and scalable foundation for developing real-time web applications based on Single Page Application (SPA) architecture (Za et al., 2023). Integration with

\

Socket.io enables bidirectional real-time communication using the WebSocket protocol as the primary transport mechanism (Riski et al., 2021). For sentiment analysis, Natural Language Processing (NLP) enables systems to process, understand, and extract meaning from human language (Amien, 2023), allowing applications to identify user emotions and provide more relevant responses proactively (Devlin et al., 2019; Fielding, 2000; Hidayatullah & Ma'arif, 2022; Jurafsky & Martin, 2023) .

Previous studies on MERN Stack-based chat applications have generally focused on communication features without integrating sentiment analysis capabilities (Tyagi et al., 2024). Meanwhile, lexicon-based sentiment analysis research for Indonesian texts has demonstrated effectiveness but continues to face limitations in handling negation and informal language patterns (Wikarsa et al., 2022). This research addresses this gap by integrating a lexicon-based NLP module inline within a Socket.io event handler in a MERN Stack architecture using the MVC pattern, enabling sentiment detection and risk-level classification within a single message-processing cycle without requiring additional external services (Putra et al., 2024) .

This study aimed to: (1) implement a real-time chat system based on the MERN Stack using the MVC pattern; (2) integrate Socket.io for bidirectional communication with room-based conversation isolation mechanisms; and (3) implement a lexicon-based NLP module to automatically detect sentiment and message risk levels in real-time communication flows. Previous relevant studies have supported the development of this system. (Tyagi et al., 2024) developed a real-time chat application called De-Link using MongoDB, Express.js, React.js, and Socket.io, which supported user authentication, password encryption using bcrypt, and group chat functionality; however, it did not integrate sentiment analysis or NLP technologies. (Gunawan & Prasetija, 2023) found that WebSocket as the primary transport mechanism in Socket.io achieved lower average latency than long polling, demonstrating its effectiveness for bidirectional communication. (Rivaldi & Wismarini, 2024) showed that NLP-based sentiment analysis could be applied to Indonesian texts through systematic processing stages with adequate accuracy. (Wikarsa et al., 2022) demonstrated that a lexicon-based approach was effective for classifying sentiment in Indonesian texts while acknowledging its limitations in handling negation and informal language. Za et al. (2023) confirmed that the MERN Stack could support the development of responsive web applications by utilizing SPA concepts and naturally accommodating MVC patterns, allowing additional modules such as NLP to be integrated within the Controller layer (Putri et al., 2024) .

METHOD

This study employed a systematic software development methodology consisting of four stages: (1) literature review and needs analysis of the MERN Stack, Socket.io, and lexicon-based NLP; (2) system architecture design, including frontend-backend interactions, Mongoose schemas for Patient, Doctor, and Message entities, and NLP integration within the Socket.io pipeline; (3) system implementation using React.js for the frontend and Node.js with Express.js and MongoDB for the backend (Nasrul & Izhar, 2023; Nurhayati, 2023; Pradigi et al., 2022); and (4) system testing focusing on REST API functionality, real-time bidirectional communication through Socket.io, and the NLP module's ability to classify sentiment and message risk levels.

The developed system was a consultation chat application called BubblePro, which enabled patients to communicate with doctors in real time while automatically monitoring message sentiment. The application context was selected as a representative scenario for validating the implementation of the three main components rather than as the primary focus of the research.

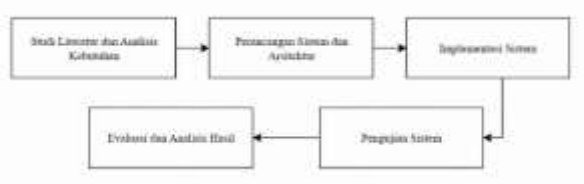

Figure 1. Research Stage – please insert images manually

System Architecture

The system architecture is designed in a layered architecture that clearly separates responsibilities between layers. The client layer is built with React.js that is divided into chat pages and analytics dashboards. The communication layer consists of Express.js that handles eight REST API endpoint groups and Socket.io servers. The logic layer (Controller) includes Controllers per entity, socket handlers for event join, joinRoom, and sendMessage, and inlineally called NLP modules. The data layer uses MongoDB to store the Patient, Doctor, and Message entities. The use of layered architecture is able to increase system modularity so that the addition of NLP modules to the logic layer does not affect the client layer or the data layer (Za et al., 2023).

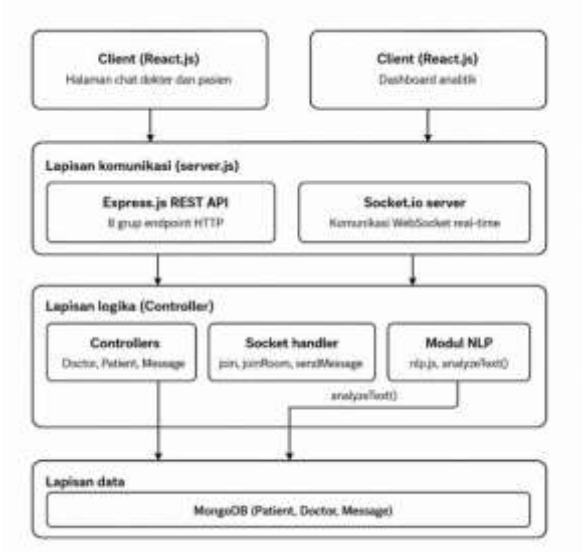

Figure 2. System Architecture – please insert images manually

Use Case Planning

The system consists of three main functional groups: authentication (registration, login, logout), real-time chat (send and receive messages, online status, typing indicators, and display of NLP analysis results), and analytics dashboard (physician statistics, patient management, and clinical records). Two actors are involved, namely the doctor and the patient, who each interact with different functionalities according to their role.

\

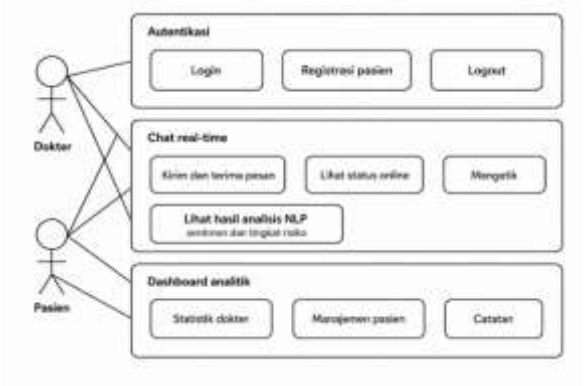

Figure 3. Use Case Diagram – please insert images manually

REST API implementation

Eight REST API endpoint groups are registered into the Express application, as mapped in Table 1. The Model layer defines the data schema using Mongoose ODM for three main entities: Patient (storing riskLevel and lastActive), Doctor (encrypted credentials), and Message (message history along with NLP analysis results in the form of sentiment, riskLevel, riskyWords, and emotions). MongoDB was chosen because of its flexibility in storing documents with diverse structures (Andrianto & Suyatno, 2024).

Table 1. Endpoint REST API Mapping

Basic Route	Controller	Main Functions
/api/Doctor	DoctorController	Registration, login, doctor profile
/api/Patient	PatientController	Account, login, patient status
/api/Message	MessageController	Message history per room
/api/Doctor-notes	DoctorNoteController	CRUD Doctor's Notes
/api/Patient-notes	PatientNoteController	CRUD patient records
/api/Patient-stats	PatientStatsController	Patient analytics dashboard
/api/Doctor-stats	DoctorStatsController	Doctor statistics dashboard
/api/Patient-management	PatientManagementController	Patient management

Real-time Communication and NLP Integration

Real-time communication is implemented using Socket.io on top of an HTTP server Node.js with the WebSocket protocol as the primary transport, allowing persistent connections and two-way (full-duplex) communication between the browser and the server (Gunawan & Prasetija, 2023) . User attendance management uses the onlineUsers object in the server's memory. The privacy of conversations is guaranteed through a user pair-based room mechanism, with roomIds deterministically formed using the formula [id_a, id_b].sort().join("_") so that roomIds are always identical from both sides without additional coordination. The core of the integration is in the sendMessage event, where NLP analysis, storage to MongoDB, patient risk profile updates, and messaging to all room members take place in a single, atomic processing pipeline. NLP analysis is only run when the field sender is valued as "Patient"; messages from the doctor get the default value without being processed by the NLP module.

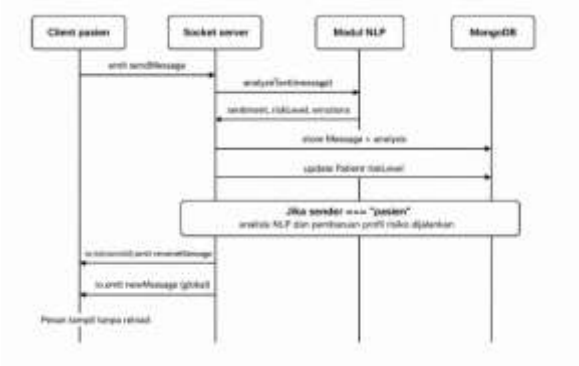

Figure 4. Sequence Diagram sendMessage – please insert image manually

Modul Natural Language Processing

The NLP module is implemented using a lexicon-based approach in nlp.js file. This approach was chosen because it does not require model training, the computation is synchronous and local so that it can be executed inline in a Socket.io flow without the invocation of external services, and is easily extended by simply adding words to the dictionary (Saputra et al., 2021; Wikarsa et al., 2022). The module uses four components of static vocabulary: positiveWords (7 words), negativeWords (7 words), riskyWords (5 high-risk phrases), and emotionsMap (34 words to 7 emotion categories). The analyzeText(text) function works in three stages: (1) the calculation of the sentiment score using String.includes() additively; (2) simultaneous detection of risky words and emotion mapping; and (3) classification with tiered priority logic—riskLevel is set to "high" if foundRisky.length is > 0 (highest priority, independent of sentiment score), "high" if negative sentiment is of high emotional intensity (anger, sad, or fear >= 2), "moderate" if sentiment is negative without meeting that threshold, and "safe" for other conditions.

Table 2. NLP Analysis Results Field on Message Schema

Field	Type	Remarks
sentiment	String	positive, negative, or neutral
riskLevel	String	safe, medium, or high
riskyWords	Array<String>	List of phrases at risk of detection
emotions	Object	Shoes 7 category emosi

RESULT AND DISCUSSION

NLP Module Functional Testing

Functional testing of the analyzeText() function was carried out using ten Indonesian-language input scenarios that included positive, neutral, low-intensity negative, high-intensity negative, and cases containing direct risk phrases. Full results are presented in Table 3.

Table 3. analyzeText() Functional Test Results

Yes	Text (concise)	Expected Sentiment	Expected Risk Level	Late Hour Aktual	Actual Risk	Status
1	happy, grateful	Positive	Safe	Positive	Safe	Pass
2	normal, normal	Neutral	Safe	Neutral	Safe	Pass
3	Feeling sad	Negative	moderate	Negative	moderate	Pass
4	tired, tired	Negative	moderate	Negative	moderate	Pass

\

5	angry, upset, hateful	Negative	height	Negative	height	Pass
6	Sad, desperate.	Negative	height	Negative	height	Pass
7	Fear, anxiety, panic	Negative	height	Negative	height	Pass
8	Suicidal Thoughts	Negative	height	Neutral*	height	Pass
9	die better	Negative	height	Positive*	height	Pass
10	happy, healed	Positive	Safe	Positive	Safe	Pass

From Table 3, it can be seen that all test scenarios produce a riskLevel classification that corresponds to the expected results. In scenarios 8 and 9, the actual sentiment is not as expected due to the limitations of the lexicon approach in dealing with negation and ambiguity of substrings. Scenario 8 generates a "neutral" sentiment because there are no words in the list of negativeWords that match exactly, while scenario 9 generates a "positive" sentiment due to matching the substring of the word "good" to the phrase "better". The accuracy of the sentiment classification obtained is 80% (8 out of 10 scenarios).

Nonetheless, both scenarios are still classified at a "high" risk level because the system successfully detects the risky phrases contained in the message. In scenario 8, the phrase "suicide" was detected, while in scenario 9, the phrase "just die" was detected. This shows that the riskyWords detection mechanism works independently of the sentiment score so that it is still able to identify high-risk conditions even if the sentiment classification results are not always accurate.

Real-time Communication Testing Socket.io

The test is done by verifying each event Socket.io using two browser clients connected at the same time. Eleven scenarios were tested, including initial connections, active user registration, room joining, sending and receiving messages without page reloading, isolation between rooms, typing indicators, and riskLevel and lastActive updates in MongoDB. All eleven scenarios were declared successful. The test results are presented in Table 4.

Table 4. Real-time Communication Test Results Socket.io

Yes	Skenario	Event Socket.io	Status
1	Initial client connection	connect	Lulus
2	Active user registration	registerUser	Lulus
3	Join a conversation room	joinRoom	Lulus
4	Patient messaging	sendMessage	Lulus
5	Receiving messages without reloading	receiveMessage	Lulus
6	Sending a doctor's message	sendMessage	Lulus
7	Isolation between rooms	sendMessage	Lulus
8	Typing indicator	typing	Lulus
9	Risk updatesPatient level	sendMessage	Lulus
10	Patient lastActive updates	sendMessage	Lulus
11	User Disconnection	disconnect	Lulus

Endpoint REST API Testing

Testing of REST APIs built using Express.js was conducted to verify that all HTTP endpoints function according to specifications (Nasrul & Izhar, 2023; Pradigi et al., 2022). Seven main endpoints were tested, including user registration and login, patient account

creation, message history retrieval with NLP data, and analytics dashboard endpoints. The test results are presented in Table 5.

Table 5. REST API Endpoint Test Results

No	Endpoint	Method	Expected Output	Code	Status
1	/api/Doctor/register	POST	Confirmation of registration	200	Lulus
2	/api/Doctor/login	POST	Success + doctor data	200	Lulus
3	/api/Patient/create	POST	PatientId PSN-YYYY-XXXX	200	Lulus
4	/api/Patient/login	POST	Success + patient data	200	Lulus
5	/api/Message/:roomId	GET	Array pesan + data NLP	200	Lulus
6	/api/Doctor-stats/overview	GET	Statistics dashboard	200	Lulus
7	/api/Patient-management/list	GET	Patient list + riskDots	200	Lulus

The test results showed that all test scenarios containing risky phrases were successfully classified at a risk level that corresponded to the expected outcome, while the sentiment classification achieved 80% accuracy with weaknesses in negation and substring ambiguity in accordance with the recognized limitations of the lexicon approach (Wikarsa et al., 2022). The design of a tiered priority logic that puts riskyWords detection above sentiment scores proved to be the right architectural decision: the system successfully identified all high-risk cases in the test scenario used even though the sentiment classification was not always accurate.

The implementation results also prove that lexicon-based NLP modules can be integrated inline within the sendMessage event handler without external asynchronous processing mechanisms. In comparison, deep learning-based approaches such as BERT require more complex inference processes and have the potential to add latency to Socket.io pipeline. Thus, the choice of a lexicon approach is not only a decision about accuracy, but also an architectural decision.

The MVC pattern on the MERN Stack architecture proved to be flexible enough to accommodate the integration of NLP modules without changing other components. The module nlp.js placed on the utility layer called from the Controller layer without touching the Model or View layers (Za et al., 2023). Compared to Tyagi et al. (2024) who do not have content analysis capabilities, this system adds an automated analysis layer without adding significant latency. The 80% accuracy obtained is also comparable to the results of Rivaldi and Wismarini (2024), corroborating the finding that limitations in negation and substring ambiguity are common characteristics of the lexical approach.

CONCLUSION

This study successfully implemented a real-time chat system based on the MERN Stack using an MVC pattern that accommodated the integration of an NLP module without requiring modifications to the Model or View layers. Socket.io was successfully integrated to support bidirectional communication through room-based messaging using a deterministic roomId mechanism, with all eleven test scenarios successfully completed.

The lexicon-based NLP module was successfully integrated directly into the sendMessage event handler. Across the ten testing scenarios, all riskLevel classifications matched the expected results, while sentiment classification achieved 80% accuracy. The two sentiment misclassifications did not affect the identification of risky conditions because riskyWords detection operated independently from the sentiment scoring process.

\

For further development, several improvements are recommended: (1) expanding the NLP lexicon vocabulary, particularly for handling negation and informal language patterns; (2) exploring deep learning models, such as IndoBERT, with asynchronous integration through message queue systems; (3) integrating external notification services to alert doctors when messages are classified as high-risk; and (4) conducting performance and scalability testing using tools such as Artillery or k6.

REFERENCE

Amien, M. (2023). Sejarah dan perkembangan teknik Natural Language Processing (NLP) Bahasa Indonesia: Tinjauan tentang sejarah, perkembangan teknologi, dan aplikasi NLP dalam bahasa Indonesia. *ELANG: Journal of Interdisciplinary Research*, *1*(01).

Andrianto, L. D., & Suyatno, D. F. (2024). Analisis performa load testing antara MySQL dan NoSQL MongoDB pada REST API Node.js menggunakan Postman. *JEISBI*, *5*(1), 18–26.

Andzani, D., & Irwansyah. (2023). Dinamika komunikasi digital: Tren, tantangan, dan prospek masa depan. *Jurnal Syntax Admiration*, *4*(11). https://doi.org/10.46799/jsa.v4i11.743

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 4171–4186.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*.

Gunawan, F., & Prasetija, A. B. (2023). Penerapan Websocket pada sistem live chat berbasis web. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer (J-PTIIK)*, *7*(2), 854–862.

Hidayatullah, A. F., & Ma'arif, M. R. (2022). Indonesian Sentiment Analysis Using Machine Learning and Natural Language Processing Approaches. *International Journal of Advanced Computer Science and Applications*.

Jurafsky, D., & Martin, J. H. (2023). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. *Online Manuscript*.

Nasrul, N., & Izhar, A. (2023). Pengembangan REST API menggunakan Express JS untuk mencari mentor pribadi. *Jurnal Informatika Terpadu*, *9*(2), 92–102.

Nurhayati. (2023). Rancang bangun back-end API pada aplikasi mobile AyamHub menggunakan framework Node JS Express. *JUSTIN*, *11*(3). https://doi.org/10.26418/justin.v11i3.66823

Pradigi, C. H., Harlina, T., & Solehatin. (2022). Implementasi Express JS untuk membangun REST API website STIKOM PGRI Banyuwangi. *JIKOM*, *9*(2), 118–122. https://doi.org/10.55794/jikom.v9i2.68

Pratama, R., Wijaya, A., & Lestari, D. (2025). Social impact orientation and sustainable business performance in SMEs. *Asian Journal of Business Research*, *15*(2), 89–105.

Putra, F. P. E., Muslim, F., Hasanah, N., Holipah, Paradina, R., & Alim, R. (2024). Analisis komparasi protokol Websocket dan MQTT dalam proses push notification. *Jurnal Sistim Informasi Dan Teknologi (JSISFOTEK)*, *5*(4), 63–72.

Putri, O. E., Pranatawijaya, V. H., & Kristianti, N. (2024). Analisis sentimen berbasis aspek dan deteksi emosi pada coffee shop Palangka Raya menggunakan deep learning. *JOINTECOMS*, *4*(3), 250–261.

Riski, R., Husaini, & Nasir, M. (2021). Implementasi socket programming pada aplikasi chat Uloen Messenger berbasis Android. *JAISE*, *2*(1), 70–76.

Rivaldi, R. C., & Wismarini, T. D. (2024). Analisis sentimen pada ulasan produk dengan metode Natural Language Processing (NLP) (Studi Kasus Zalika Store 88 Shopee). *Jurnal Ilmiah Elektronika Dan Komputer*, *17*(1), 120–128.

Saputra, N., Handayani, E., & Dwiretnani, A. (2021). Analisa Penjadwalan Proyek dengan Metode Critical Path Method (CPM) Studi Kasus Pembangunan Gedung Rawat Inap RSUD Abdul Manap Kota Jambi. *Jurnal Talenta Sipil*, *4*(1), 44. https://doi.org/10.33087/talentasipil.v4i1.48

Tyagi, K., Singh, D., & Sharma, B. (2024). De-Link chat application using MERN Stack and Socket.io. *International Journal of Research Publication and Reviews*, *5*(5), 8978–8983.

Widjaja, G. (2025). Perubahan pola komunikasi dalam masyarakat akibat penggunaan aplikasi pesan instan. *Prosiding Seminar Nasional Indonesia*, *3*(1), 10–16.

Wikarsa, L., Angdresey, A., & Kapantow, J. D. (2022). Implementasi metode Naïve Bayes dan lexicon-based approach untuk mengklasifikasi sentimen netizen pada tweet berbahasa Indonesia. *Jurnal Ilmiah Realtech*, *18*(1), 15–24.