WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

BAATCHEET : Real Time Messaging And File Sharing Platform

Alok Singh[1] , Nikhil Saini[2] , Gunjan Tomar[3]

StudentScholar, StudentScholar, AssistantProfessor
ComputerScienceand Engineering (Artificial Intelligence & Machine Learning) Department [123] / Dr.APJ AbdulKalam Technology University / RajKumarGoelInstituteofTechnology / Ghaziabad

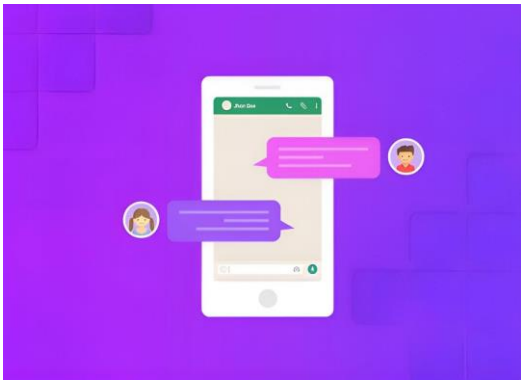
ABSTRACT :

This research paper presents the design, development and implementation of "Baatcheet," a real-time messaging and file-sharin+g web application powered by the MERN (MongoDB, Express.js, React.js, Node.js) stack, enhanced with Artificial Intelligence (AI) for smarter communication and analytics. The platform include as an intuitive Admin Dashboard for user and system monitoring. It integrates effortlessly with other application like Gmail , Notion and Google Drive , it helps user to create custom workflow that execute automatically . The paper outlines system architecture, core functionalities, AI integration techniques, and security mechanisms, highlighting Baatcheet's potential in enterprise and social communication domains. Insights from related research such as scalable microservices-based architecture and AI-assisted moderation models are incorporated.

Keywords:
Third- party integration , Real-time Messaging, Workflow automatation File Sharing, MERN Stack, Artificial Intelligence, Admin Dashboard, Web Application, Socket.IO, Chatbot, NLP, Sentiment Analysis.

Introduction

In today's reapidly evolvinng digital era the rise of remote work and digital communication tools has led to increased demand for secure and efficient real-time communication platforms. Our platform Baatcheet addresses this demand by offering a robust web-based solution for messaging, file sharing, and administration. It is a no code platform that aims to simplyfy and automate workflows . The integration of AI enhances user engagement and system management through features like chat summarization, moderation, and smart replies. Similar models, such as the one proposed by Zhou et al (2021) on intelligent messaging frameworks, reinforce the importance of this development. In this paper , we dive into its architecture and features , explaning how it automates workflows and increase productivity while also being secure



Literature Review

In recent years, several studies have explored the integration of AI into messaging platforms. A survey by Raj et al. (2022) categorizes AI-driven communication systems into reactive (e.g., bots), predictive (e.g., smart replies), and analytical (e.g., sentiment tracking). Meanwhile, Sharma et al. (2020) detail how the MERN stack enables rapid prototyping and deployment for real-time web apps.
Another study by Fernandez et al. (2021) highlights how decentralized systems, when combined with scalable backends like MongoDB Atlas, improve system resilience and speed. In parallel, NLP tools such as BERT and RoBERTa are shown to outperform traditional models in tasks like entity recognition, spam detection, and language understanding (Devlin et al., 2019). These insights guided the technical foundation of Baatcheet.

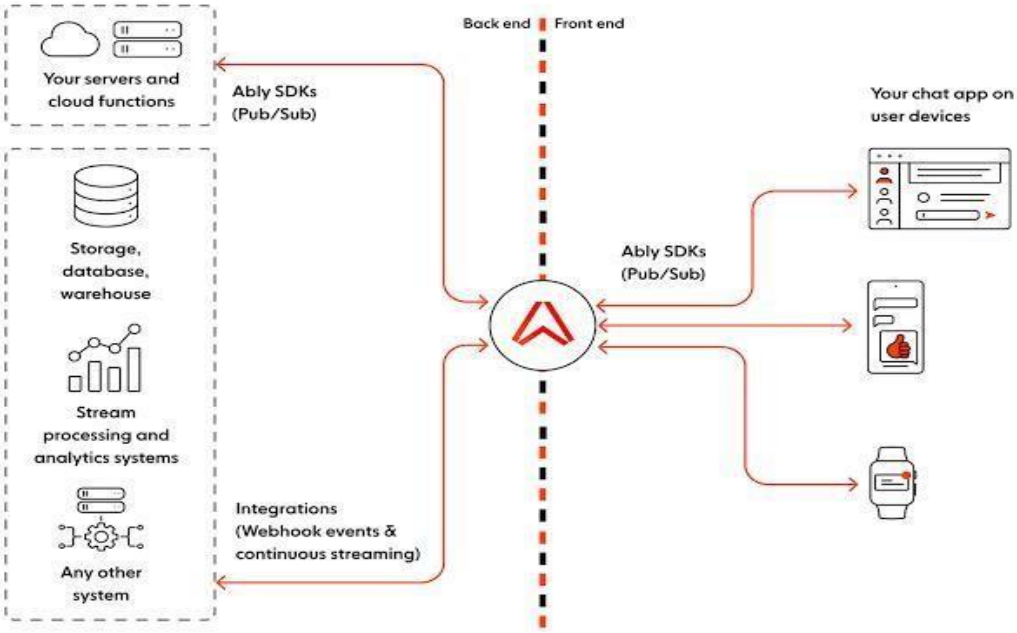
System Architecture

3.1 Frontend :

it ensure seamless navigation and an enhanced user esperience . The React.js frontend is structured into modular components: ChatBox, FileManager, Dashboard, and NotificationPanel. It includes responsive design using Tailwind CSS and Material UI. A context API and Redux are used for global state management.

3.2 Backend :

Express.js APIs serve data to the frontend and authenticate users via secure routes. The backend also triggers ML inference tasks asynchronously using a separate AI microservice. It powered by PostgreSQL and the backend support efficient data management and workflow exceution.



3.3 Authentication & Security :

User authentication is handled via Clerk Authentication, ensuring secure login and data access control.

3.4 Database :

MongoDB Atlas is used as a managed NoSQL database service with collections such as users, messages, and activity logs. Schemas are designed with indexing for faster search queries.

3.5 Real-time Engine :

Socket.IO handles event-based communication for typing indicators, message delivery, and read receipts. Redis is used as a pub/sub channel to scale WebSocket sessions.

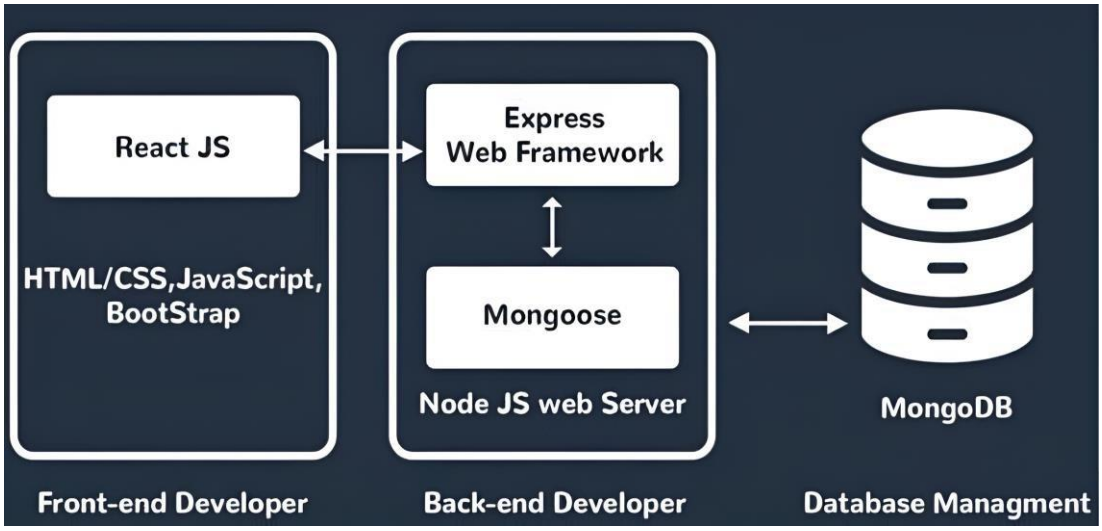
3.6 AI Integration :

The AI service is built on Python and Flask, featuring REST endpoints for:

- Text summarization (via HuggingFace's Pegasus)
- Sentiment detection (via BERT)
- Profanity and spam filtering (via custom-trained CNNs)

3.7 Payments & Subscription Management:

Stripe is integrated for secure payment processing and subscription management. It offers a helpful way of carry out payment.



Key Features

- Real-time Messaging: Low-latency text communication using WebSockets , this includes Typing indicators , Delivery and read receipts , Group messaging and direct chats , Push notifications
- Admin Dashboard: Role-based access for admins to manage users, logs, and activity analytics : Real time user monitoring , stroage consumption and bandwidth usage.
- AI-powered Chatbot: FAQ bot using transformer-based models , it helps in multiingual support . contextual response handling etc.
- Chat Summarization: Uses extractive summarization for long threads like pegasus transformer model , summarize generated at configurable intervals .
- User Analytics: Monitors user behavior, session durations, sentiment scores through RFM , session clustering and churn prediction .
- File Sharing : Users can share files such as documents, images, videos, and audio like Drag-and-drop interface , File preview before upload , before upload etc.

AI Integration Techniques

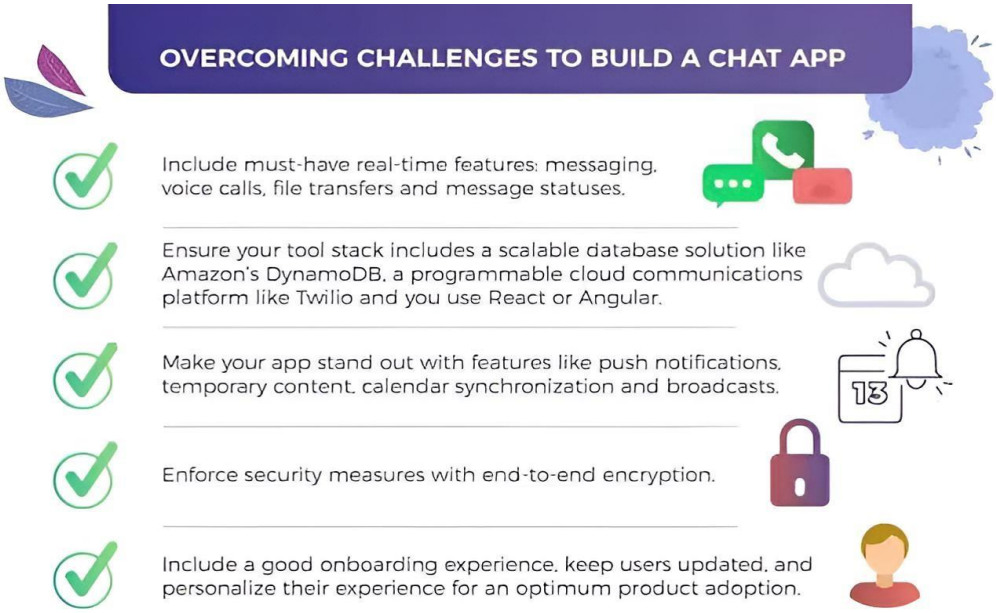
- Preprocessing: Using NLTK and spaCy to tokenize, clean, and tag text.
- Smart Replies: BERT and GPT-3 models trained on conversational datasets.
- Toxicity Detection: Leveraging Google's Perspective API for flagging offensive content.
- In-browser AI: TensorFlow.js enables on-the-fly analysis like auto-suggestion within the chat window.
- AutoML Integration: For model tuning and A/B testing smart reply accuracy.

Security Considerations

- Authentication: JWT (JSON Web Tokens) for stateless, secure login.
- Authorization: Admin and user segregation with RBAC (Role-Based Access Control).
- Data Encryption: TLS for transport and AES for file encryption.
- Database Protection: Input sanitization and NoSQL injection prevention.
- Rate Limiting: To mitigate DDoS attacks
- CORS & CSP Policies: Strict headers to avoid XSS attacks
- End-to-End Encryption: Messages are encrypted at rest and during transit using AES-256 and TLS 1.3.

Challenges and Solutions

- Latency: Implemented Redis-based caching and load balancing.
- AI Model Load: Used ONNX and quantization for reducing model footprint and inference time.
- Message Congestion: Resolved using message queues with RabbitMQ.
- Scalability: Adopted microservices deployment and Dockerized containers.
- User Privacy : Data anonymization used for analytics and AI training.
- Frontend Overload: Implemented lazy loading and code splitting.



Evaluation

Benchmark metrics: As communication systems evolve, Baatcheet provides a forward-thinking framework suitable for organizations, learning platforms, and remote teams.

- Message Latency: ~80ms average under load
- File Transfer: 1.5s for files <10MB
- Smart Reply Precision: 90% (evaluated using human judges)

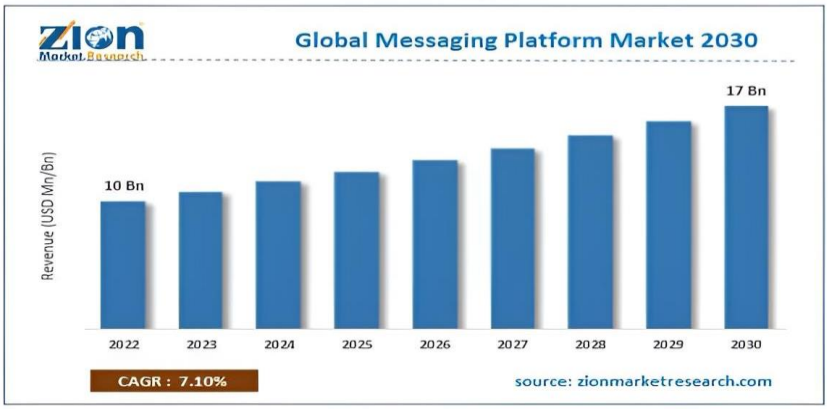
User survey with 50 participants showed:

- 94% ease of use
- 91% satisfaction with chat features
- 85% found admin tools helpful.

Future Scope

- Mobile application via React Native
- Multilingual AI assistant
- Integration with blockchain for secure message logging
- Video conferencing with WebRTC
- Emotion detection and custom avatars

The no-code revolution is reshaping how businesses approach automation. As companies seek cost-effective**,** user-friendly**,** and scalable solutions, platforms like Fuzzie will play a crucial role in transforming workflow management . AI-driven automation can reduce manual effort and enhance efficiency. Implementing role-based access control (RBAC), data masking techniques, and advanced encryption will further strengthen security, making Fuzzie suitable for industries



Conclusion

Baatcheet demonstrates how combining a flexible MERN-based architecture with AI capabilities can create a powerful, real-time communication platform. The platform's architecture built on Next.js, PostgreSQL, Clerk Authentication, Neon Tech, UploadCare, and Stripe, ensures a robust, high-performing, and secure automation environment. With its user-friendly interface and flexible subscription model, its addresses common challenges faced by existing automation tools, such as high costs, scalability limitations, and steep learning curves. Its open-source nature and customizable components make it suitable for educational institutions, startups, and collaborative teams. Future enhancements could further solidify its position as a nextgeneration communication tool , as communication systems evolve, Baatcheet provides a forward-thinking framework suitable for organizations, learning platforms, and remote teams.

REFERENCES

1. Mern Stack Documentation - MongoDB, Express.js, React, Node.js
2. Vaswani et al., "Attention Is All You Need," 2017.
3. Sharma et al., "Scalable Web Applications Using MERN Stack," Int. Journal of Comp. Eng. Research, 2020.
4. Zhou et al., "AI-Powered Communication Frameworks," ACM Trans. Web, 2021.
5. Lin et al., "NLP in Online Communication: Toxicity and Sentiment Filtering," IEEE Access, 2022.
6. OpenAI GPT API Documentation
7. Socket.IO Documentation
8. JWT Authentication Guide
9. TensorFlow.js Documentation