

"Mentesa: No - Code AI Chatbot Creation Platform"

Rakhi Punwatkar¹, Mayur Koli², Anirudha Kapurkar³, Niharika Wagh⁴

¹Rakhi Punwatkar Computer & Zeal College Of Engineering and Research,Pune

²Mayur Koli Computer & Zeal College Of Engineering and Research,Pune

³Anirudha Kapurkar Computer & Zeal College Of Engineering and Research,Pune

⁴Niharika Wagh Computer & Zeal College Of Engineering and Research,Pune

Abstract - This paper presents Mentesa v1.0, an improved version of the Mentesa no-code AI chatbot creation platform. The platform allows non-technical users to create, manage, and deploy personalized AI chatbots using their own documents as a knowledge source, without requiring any programming skills. The system continues to use Google's Gemini-2.5-Pro language model with a Retrieval-Augmented Generation (RAG) approach to provide accurate and context-aware responses. Firebase is used for secure data management and user authentication, while Render provides cloud-based deployment support. Compared to the previous version, Mentesa v1.0 introduces several key improvements focused on security, accessibility, and scalability. These enhancements include Firebase-based authentication for secure user accounts, individual API key generation for controlled chatbot access, an API-authenticated embeddable widget that allows chatbots to be integrated into external websites, a billing and usage quota management system, and an improved document processing pipeline that supports DOCX and plain text files in addition to PDFs. The frontend has also been redesigned and migrated from Streamlit to React, providing a more responsive, organized, and scalable user experience.

Keywords: No-Code AI, Retrieval-Augmented Generation, API Key Management, Firebase Authentication, Embeddable Widget, Multi-Format Ingestion, React, Conversational AI, Billing System

1.INTRODUCTION

Mentesa was initially developed to address a major challenge in the adoption of AI-based chatbot systems: although modern language models have become highly powerful, creating chatbots that can understand and respond based on custom

documents is still a complex process that usually requires technical expertise. To overcome this barrier, the platform introduced a Retrieval-Augmented Generation (RAG) pipeline, a multi-tenant Firebase-based architecture, and a no-code Streamlit interface that allowed users to upload documents, create knowledge bases, and deploy AI chatbots without writing any code. While the initial version successfully demonstrated the feasibility of this approach, real-world usage revealed several areas that required further improvement. The previous version had limited user access control, as it did not include a dedicated authentication system. Although data isolation was achieved at the architectural level, the absence of user authentication limited identity verification and prevented fine-grained access control. Additionally, chatbot access was restricted mainly to the Mentesa platform or a simple embedding method, preventing external applications from interacting with chatbots through APIs. The system also lacked usage tracking and resource management features, making it difficult to monitor chatbot activity or implement fair usage limits among users. Furthermore, the document processing pipeline supported only PDF files, limiting compatibility with commonly used formats such as DOCX and plain text files. Mentesa v1.0 focuses on addressing these limitations while maintaining the original goal of providing a simple, no-code AI chatbot creation experience. The core components, including the RAG pipeline, Firebase-based multi-tenant storage, and chatbot creation workflow, remain unchanged. The new version introduces improvements in security, accessibility, and scalability through user authentication, API key-based chatbot access, an authenticated embeddable widget system, usage monitoring and billing support, and expanded document ingestion capabilities for DOCX and TXT formats.

Additionally, the frontend has been migrated from Streamlit to React, creating a more structured, responsive, and maintainable user interface with better separation between frontend and backend components.

2. PROBLEM STATEMENT

Although the initial version of Mentesa successfully demonstrated the potential of a no-code AI chatbot platform, several limitations prevented it from being fully suitable for practical, real-world deployment.

The absence of a dedicated user authentication system was one of the major limitations. While the architecture maintained separation of user data, there was no mechanism to verify user identity or provide secure, user-specific access. This limitation also restricted the implementation of important features such as personalized dashboards, individual chatbot management, usage monitoring, and controlled access to user-created bots.

Another challenge was limited chatbot accessibility. Users could interact with their bots mainly through the Mentesa interface or a simple embedded script. However, the system did not provide an API-based method for external applications or services to communicate with chatbots. This reduced the platform's flexibility and made integration with existing business applications and workflows difficult.

The previous system also lacked usage monitoring and resource management capabilities. Since there was no quota or billing mechanism, the platform could not track chatbot usage, manage resource consumption, or support a scalable deployment model where resources could be allocated fairly among users.

Additionally, the document ingestion pipeline supported only PDF files, which limited the usability of the platform. Many organizations rely on formats such as DOCX and TXT for storing and sharing information. Requiring users to convert these files into PDFs before uploading added unnecessary steps and reduced the overall convenience of the system.

These limitations highlighted the need for improvements in authentication, integration capabilities, usage management, and document compatibility while maintaining Mentesa's core objective of providing an accessible no-code AI chatbot creation experience.

3. METHODOLOGY

3.1 User Authentication

Mentesa v1.0 uses Firebase Authentication as the primary mechanism for managing user identity and access control across the platform. Users must create an account and authenticate before accessing chatbot creation and management features. After successful login, Firebase generates a secure authentication token, which is verified by the backend for every incoming request. All user-related resources, including chatbot configurations, uploaded knowledge bases, and conversation records stored in Firebase Firestore, are associated with the authenticated user's unique identifier. This ensures that each user's data remains isolated and prevents unauthorized access to resources belonging to other users. The authentication system provides the foundation for implementing secure user-specific features and managing access across the platform.

3.2 API Key Generation and Management

Mentesa v1.0 introduces API key-based access for every chatbot created on the platform. When a new bot is created, the system automatically generates a unique API key in the format `mentesa_sk_XXXX...`, which acts as an authentication credential for external requests. Users can access their API keys through the bot management dashboard, use them for external integrations, and regenerate them whenever required. When a key is regenerated, the previous key becomes invalid immediately, preventing unauthorized usage. Each API key is connected with the user's allocated usage limits, allowing the platform to monitor requests and apply quota restrictions. The API key must be provided with every external API request or widget configuration. Requests containing missing or invalid keys are rejected by the backend. This approach extends Mentesa bots beyond the platform interface and enables

them to function as secure, externally accessible chatbot services.

3.3 Embeddable Widget with API Authentication

Mentesa v1.0 improves chatbot deployment by replacing the basic embedding approach from the previous version with an API-authenticated widget system. Users can generate an embed script directly from the bot management interface and add it to any external website to deploy their chatbot. The generated script includes three important configuration attributes: `data-api-key`, which verifies the request and provides authentication; `data-bot-name`, which identifies the chatbot being deployed; and `data-backend-url`, which connects the widget with the hosted backend service. Once added to a webpage, the widget enables visitors to interact with a document-based AI chatbot without requiring additional setup. Since every interaction is validated using the associated API key, unauthorized or incorrectly configured deployments are blocked by the backend. This provides a balance between easy deployment for non-technical users and secure access control for external chatbot usage.

3.4 Billing and Usage Quota Management

A billing layer has been introduced to track and manage resource usage across the platform. Each user account is associated with a message quota that governs how many bot interactions are permitted within a given period. Usage is incremented with each API key request and tracked in Firebase against the user's account. When a user approaches or reaches their quota, the platform surfaces usage information through the Billing section of the interface. This layer establishes the operational foundation for a sustainable deployment model and ensures that resource usage remains fair and accountable across the multi-tenant architecture.

4. SYSTEM FLOW

4.1 Registration and Authentication

The process begins when a user accesses Mentesa v1.0 and creates an account or logs in using Firebase Authentication. After successful authentication, Firebase generates a session token that is used to verify subsequent requests between the frontend and backend. The React frontend uses this session information to display only the resources associated with the logged-in user, including created bots, usage details, and billing information. Any request made without valid authentication credentials is blocked by the backend before accessing protected resources.

4.2 Bot Creation and Document Ingestion

After authentication, users can create a new chatbot by providing basic details such as the bot name, personality description, and uploading knowledge documents. The platform supports multiple document formats, including PDF, DOCX, and TXT files.

The backend performs file-type detection after upload and routes each document through its respective parsing pipeline, using PyMuPDF for PDF extraction, python-docx for DOCX processing, and native text handling for TXT files. The extracted content is divided into smaller text chunks and converted into vector embeddings. These embeddings are stored and indexed using FAISS to enable efficient similarity-based document retrieval during conversations.

The chatbot configuration, document metadata, and generated knowledge base are stored in Firebase under the respective user's account space. A unique API key is also generated during chatbot creation, allowing secure access to the bot through external requests.



4.3 Chat Interaction

Users can interact with their chatbot through the Mentesa interface or through an externally embedded widget. When a query is submitted, the backend first validates the provided API key and checks whether the user has available usage quota. After successful verification, the system retrieves relevant document segments from the FAISS index and combines them with recent conversation history stored in Firebase.

The collected information is used as context for Gemini-2.5-Pro, which generates a response based on the retrieved knowledge. The generated answer is returned to the user, while the conversation details and usage information are recorded in Firebase for future reference and quota tracking.



4.4 Bot Management and External Deployment

The bot management interface allows users to control and update their chatbot settings. Users can modify the bot's personality, change the bot name, clear conversation history, regenerate API keys, or remove the chatbot. All updates are synchronized with Firebase to maintain consistent data across the platform.

For external deployment, users can generate an embed script containing the required bot configuration, including the API key and backend connection details. This script can be added to external websites to make the chatbot available outside the Mentesa platform. The billing section provides users with information about their current usage and remaining quota, allowing better visibility and control over resource consumption.



4.5 Multi-Format Document Ingestion

The document ingestion pipeline has been extended with a file-type detection layer that routes uploaded files to format-specific parsers before they enter the existing chunking and embedding workflow. DOCX files are processed using the python-docx library, which extracts paragraph-level text content while preserving document structure. TXT files are ingested directly without any parsing overhead, making them the most lightweight ingestion path in the pipeline. PDF ingestion continues to function as in the original system using PyMuPDF. Once parsed, all three file types pass through the same downstream pipeline chunking into segments, embedding generation, vector indexing in FAISS, and storage in Firebase maintaining full architectural consistency regardless of source format.

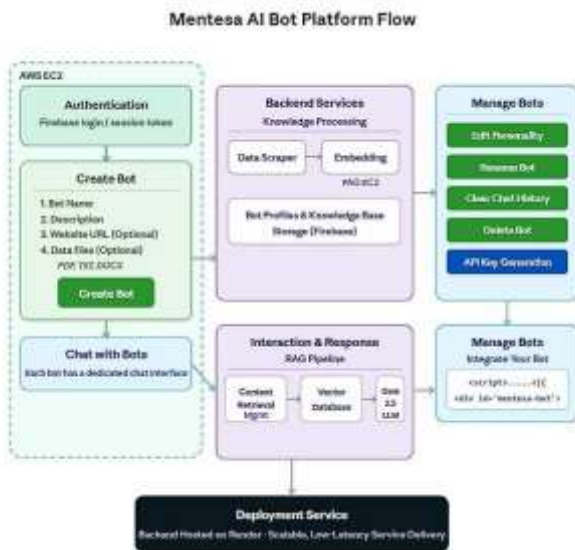


Figure -1: System Architecture

5. RESULTS / DISCUSSION

Mentesa v1.0 was evaluated based on five major improvement areas introduced in this version: security and access management, API-based chatbot accessibility, external deployment support, usage monitoring, and multi-format document processing. The evaluation focuses on how these enhancements address the limitations identified in the previous version while maintaining the platform's original no-code approach.

5.1 Security and Authentication

The integration of Firebase Authentication introduces a proper identity management layer that was not available in the earlier version of Mentesa. Each user session is verified using Firebase-generated authentication tokens, and resources stored in Firestore are associated with individual user identifiers. This ensures that users can access only their own bots, documents, and related data.

Testing showed that requests without valid authentication credentials are blocked before reaching the application's core processing components. This improves the platform's reliability for scenarios where user privacy and controlled access are important.

Additionally, authentication provides the foundation required for implementing user-specific capabilities such as API key ownership, usage tracking, and personalized account management.

5.2 API Key System and Bot Accessibility

The introduction of API keys improves the accessibility and integration capabilities of Mentesa bots. Previously, chatbot interaction was limited mainly to the Mentesa interface. With the new API key system, bots can now receive requests from external applications through authenticated API calls.

Each bot is assigned a unique API key, allowing users to connect their chatbots with external websites, business applications, and custom workflows. The key regeneration feature provides additional control by allowing users to invalidate previous keys when required. Furthermore, tracking usage through individual API keys provides better visibility into chatbot activity across both internal and external interactions.

5.3 Embeddable Widget Reliability

The authenticated widget system improves the deployment process by allowing users to integrate their chatbot into external websites using a simple embed script. This maintains the no-code objective of the platform by allowing deployment without requiring technical modifications.

During testing, the widget successfully loaded chatbot functionality on external webpages when configured with valid authentication details. Requests containing missing or incorrect API keys were rejected by the backend, ensuring that chatbot resources could not be accessed without proper authorization. The same authentication and quota rules are applied to embedded deployments, maintaining consistent security across different access methods.

5.4 Billing and Usage Visibility

The addition of usage monitoring and quota management introduces better control over resource consumption. The system records chatbot interactions through API key-based tracking and displays usage information through the billing interface.

This feature provides users with visibility into their chatbot activity and creates a structured approach for managing platform resources. By applying usage limits, the system can prevent excessive consumption by individual users and support a more scalable multi-user environment.

5.5 Multi-Format Document Ingestion

The updated document ingestion pipeline expands support beyond PDF files by adding compatibility with DOCX and TXT formats. Testing was performed using sample documents from all supported formats to verify successful extraction and processing.

TXT files showed reliable processing due to their simple structure, while DOCX files were successfully handled using python-docx for extracting document content. PDF processing continues to work with the existing pipeline. After extraction, all formats follow the same downstream process of text chunking, embedding generation, and FAISS indexing, allowing the RAG architecture to remain unchanged while improving user convenience.

5.6 Overall Discussion

Mentesa v1.0 demonstrates that a no-code AI chatbot platform can be extended with practical production-oriented features while preserving its original design philosophy. The existing RAG pipeline, Firebase-based data architecture, and chatbot creation workflow continue to serve as the foundation of the system.

The new improvements address important limitations of the previous version by introducing secure authentication, API-based chatbot access, external deployment support, usage monitoring, and wider document compatibility. These additions improve the platform's usability, scalability, and readiness for practical applications.

The migration from Streamlit to React further strengthens the system by providing a more structured frontend architecture and clearer separation between interface components and backend services. Overall, Mentesa v1.0 continues to focus on making AI chatbot creation accessible to non-technical users while improving the reliability and flexibility required for real-world deployment.

6. CONCLUSION

This paper presented Mentesa v1.0, an enhanced version of the original no-code AI chatbot creation platform. The system continues to build upon its core architecture, which includes a Retrieval-Augmented Generation (RAG) pipeline, multi-tenant Firebase-based data management, and the Gemini-2.5-Pro language model. This iteration focuses on improving the platform's readiness for practical usage by introducing important features such as Firebase Authentication for secure user management, API key-based chatbot access, an authenticated embeddable widget system, usage monitoring through billing and quota management, and expanded document support for PDF, DOCX, and TXT formats.

The introduced improvements directly address the limitations identified in the previous version and enhance the platform's security, flexibility, and usability. These additions allow Mentesa to support more realistic deployment scenarios while maintaining its original goal of enabling users without technical expertise to create and manage AI-powered chatbots. The migration from Streamlit to React further improves the platform's structure, maintainability, and ability to support future enhancements.

Mentesa v1.0 demonstrates that improving accessibility and improving deployment capabilities can progress together. By combining a user-friendly no-code approach with essential

production-oriented features, the platform moves closer to making advanced AI chatbot technology practical and accessible for a wider range of users.

REFERENCES

- [1] R. Yang et al., RAGVA: Engineering RetrievalAugmented Generation an Experience Report, Elsevier (ScienceDirect), 2025
- [2] Z. Li, Retrieval-Augmented Generation for Educational Application, Elsevier (ScienceDirect), 2025
- [3] Y. Liu, Reducing Hallucinations of Large Language Models via HSP, Springer, 2025
- [4] M. S. Swift, One Chatbot Safety Benchmark To Test Them All, IEEE Spectrum, 2024
- [5] C. Batista et al., Towards a Multi-Tenant Microservice Architecture: An Industrial Experience, IEEE, 2022
- [6] M. Arif, A Literature Review on Model Conversion, Inference, and Optimization for Efficient Deployment, Elsevier (ScienceDirect), 2025
- [7] R. K. Sharma & A. Dutta, AI-Powered Conversational Agents for Adaptive Learning Environments, IEEE (Conference), 2024
- [8] N. F. Mir, AI-Driven Management of Dynamic MultiTenant Cloud Networks, IEEE (Conference), 2023
- [9] G. Gamage et al., Multi-Agent RAG Chatbot Architecture for Decision Support in Net-Zero Emission Energy Systems, IEEE (Conference), 2024
- [10] S. U. Singh, A Survey on Chatbots and Large Language Models: Testing and Evaluation, IEEE (Conference), 2025
- [11] B. Suzic and M. Latinovic, "Rethinking Authorization Management of Web-APIs," *2020 IEEE International Conference on Pervasive Computing and Communications*, Austin, TX, USA, 2020, doi: 10.1109/PerCom45495.2020.9127384.