



iJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: <https://doi.org/10.22214/ijraset.2026.85309>

www.ijraset.com

Call: ☎ 08813907089

E-mail ID: ijraset@gmail.com

WaBot.ai: An AI-Powered SaaS Platform for Customer Engagement and Business Automation

Manthan Pande¹, Yasir Sheikh², Bhupendra Bhujade³, Chaitanya Nile⁴, Harshit Jangra⁵, Harshal Shahare⁶, Shyam Naik⁷

^{1, 2, 3, 4, 5, 6}Student, Department of Computer Science and Engineering, G H Rasoni University, Amravati, India

⁷Assistant Professor, Department of Computer Science and Engineering, G H Rasoni University, Amravati, India

Abstract: *Small businesses in India run much of their customer contact through WhatsApp, yet many of them still answer every message by hand. Enquiries arrive after hours, go unanswered, or sink in a crowded chat list, and what the business learns about each customer lives nowhere except the owner's phone. This paper describes WaBot.ai, a multi-tenant software-as-a-service platform that connects a business's WhatsApp number to a large language model (LLM), a business-specific knowledge base, a lightweight CRM, and an appointment system. Messages arrive through the WhatsApp Cloud API, are combined with organization context, and are passed to an LLM through the OpenRouter gateway. A distinguishing component is an action parser that separates ordinary conversational replies from structured business actions, such as booking, cancelling, or rescheduling an appointment, and executes the latter against Cloud Firestore. The parser also stops malformed model output from being forwarded to customers. We report a working prototype built with React, TypeScript, Express.js, and Firebase. Evaluation to date consists of functional and integration testing of the main workflows. Formal accuracy, latency, and user-study measurements have not been carried out and are therefore not reported.*

Keywords: *WhatsApp Cloud API, large language models, conversational AI, business process automation, CRM, appointment scheduling, SaaS, MSME*

I. INTRODUCTION

For many small businesses in India, WhatsApp works as the front desk. A clinic receives appointment requests on it, a salon answers price questions on it, and a coaching centre chases fee payments through it. The channel is convenient because customers already have it installed and know how to use it. It turns into a burden once the number of messages grows past what the owner, or one member of staff, can read in a working day.

The consequences are easy to list. Replies come late, or not at all after closing time. The same questions about price, timing, and location are typed out again and again. Appointments end up in a notebook or in someone's memory. Whatever the business learns about a customer stays inside a chat thread, where it cannot be searched, filtered, or handed to a colleague. Ordinary WhatsApp Business use offers none of the structure that a CRM or a booking system would give, and when a business does adopt such tools they sit apart from the conversation itself.

Large language models (LLMs) make it realistic to answer free-form customer messages helpfully [1], [2]. Answering is only part of the job, though. When a customer writes "I want to book an appointment tomorrow at 5 PM," the useful outcome is not a paragraph of text but a booking that exists in the system afterwards. This paper describes WaBot.ai, a software-as-a-service (SaaS) platform we built to close that gap for Indian micro, small and medium enterprises (MSMEs). Its design goal fits in one line: never miss a WhatsApp customer again.

The contributions of this work are:

- 1) An end-to-end architecture that joins the WhatsApp Cloud API, an LLM gateway, an organization-specific knowledge base, a CRM, and an appointment module in one multi-tenant platform;
- 2) An action-parsing layer that lets the model return structured business actions, which backend code executes, and that keeps malformed model output away from customers; and
- 3) A working prototype with functional test results for the main booking, AI, and CRM workflows, together with a plain statement of what has not yet been measured.

The rest of the paper is organized as follows. Section II reviews related work, Section III states the problem, Section IV presents the architecture, Section V explains the methodology, Section VI covers the implementation, Section VII reports testing, Section VIII compares and discusses, and Sections IX and X give future scope and conclusions.

II. RELATED WORK

We did not carry out a systematic literature survey for this prototype, so this section is short on purpose. It names the lines of work closest to the design of WaBot.ai.

A. Conversational Agents for Customer Support

Customer-facing chatbots have traditionally been built from rules, keyword matching, or static FAQ lists. They are predictable, but they fail on phrasing the author did not anticipate, and each new service a business offers means new rules. Transformer-based language models [1], trained at scale [2], cope with open phrasing far better and can follow instructions given in a prompt. That second property is the one WaBot.ai relies on.

B. Grounding Models in Business Knowledge

A general model does not know a particular clinic's timings or a particular salon's price list. Retrieval-augmented generation [3] handles this by giving the model relevant documents at inference time instead of retraining it. WaBot.ai borrows the idea loosely: business-provided information is stored per organization and supplied to the model as prompt context. We do not claim equivalence with the full method of [3].

C. Language Models That Act

Work on interleaving reasoning with actions [4], and on models that learn to call external tools [5], shows that model output can drive operations outside the text channel. WaBot.ai applies the same idea at a small scale. The model may return a structured action, but ordinary backend code, not the model, performs the operation.

D. Platform Services

The WhatsApp Business Platform exposes messaging to developers through the Cloud API and webhooks [10], and Firebase supplies authentication and a document database [6], [7]. These are building blocks rather than research, but they set the limits of what a system of this kind can do, so we cite them here.

III. PROBLEM STATEMENT AND OBJECTIVES

A. Problem Statement

Businesses that handle WhatsApp enquiries by hand run into a recurring set of problems: delayed replies, enquiries that are missed altogether, difficulty following many conversations at once, manual appointment scheduling, weak lead tracking, repeated questions, customer details scattered across chats, and no answers outside working hours. Existing options address parts of this. Staff can be hired to reply, a separate CRM can hold customer records, a separate tool can manage appointments, and a static FAQ or generic chatbot can deflect simple questions. What is missing is a system in which conversation, customer records, and bookings share one backend.

B. Objectives

The main objective is an AI-powered SaaS platform for MSMEs that can:

- automate customer communication on WhatsApp and generate answers to customer queries;
- capture and manage customer information, with CRM functions for leads;
- automate booking, cancellation, and rescheduling of appointments;
- keep a per-business knowledge base that shapes the AI's answers;
- give owners a central dashboard and allow hybrid AI-human handling of conversations; and
- reduce repetitive support work while remaining scalable across many businesses.

C. Target Users

The first target users are clinics, salons, coaching centres, and restaurants, that is, service businesses that take appointments or repeat enquiries over WhatsApp. The architecture is not tied to these categories.

IV. SYSTEM ARCHITECTURE

A. Overview

WaBot.ai is a multi-tenant SaaS platform. A business connects its WhatsApp Business number, and from then on every incoming customer message follows the path in Fig. 1. The Meta WhatsApp Cloud API [10] delivers the message to a webhook on an Express.js backend. A chat service manages the conversation, the organization service works out which business the message belongs to, and the customer record is found or created. The AI service then combines the business's knowledge with the conversation context into a prompt and sends it to an LLM through OpenRouter. The parsed output is a customer-facing reply, a business action, or both, and the results are written to Cloud Firestore [7] and appear in the dashboard.

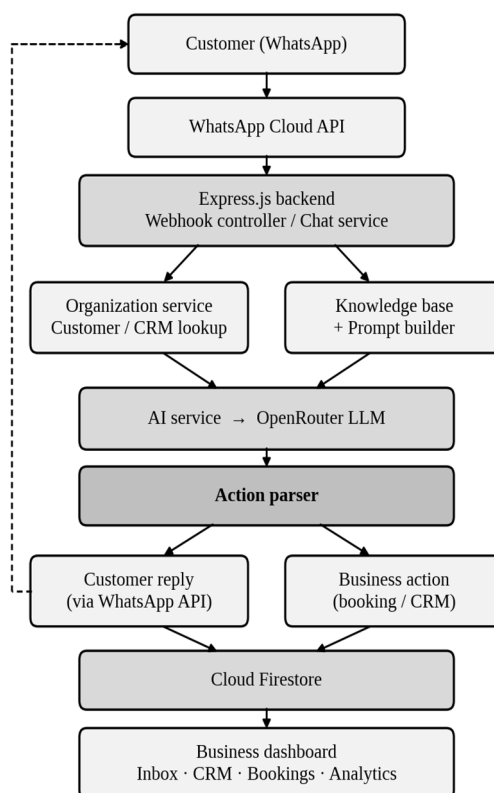


Fig. 1 Architecture of WaBot.ai. The dashed line shows the reply returning to the customer through the WhatsApp Cloud API

B. Technology Stack

Table I lists the technologies used. The AI component is an LLM reached through a gateway. No model was trained for this project.

TABLE I
TECHNOLOGY STACK OF WABOT.AI

Layer	Technology
Frontend	React, TypeScript, Vite, inline CSS (dark theme)
Backend	Node.js, TypeScript, Express.js
Database	Firebase Cloud Firestore
Authentication	Firebase Authentication with Google Sign-In; JWT middleware
Server SDK	Firebase Admin SDK
Messaging	Meta WhatsApp Cloud API
AI gateway	OpenRouter API
Dev tools	Visual Studio Code, Git/GitHub, ngrok (local webhook testing)

C. Backend Structure

The backend follows a service-oriented layout. Controllers handle authentication, the WhatsApp webhook, organizations, CRM, and bookings. Beneath them sit services for chat handling, AI, organizations, knowledge, and the WhatsApp connection. A JWT middleware protects the dashboard endpoints, and persistence goes through the Firebase Admin SDK to Firestore.

D. Organization-Based Data Model

Every record belongs to an organization, which is what makes the platform multi-tenant. Conceptually, an organization owns its users, customers, conversations, messages, bookings, knowledge base, and WhatsApp connection. A conversation keeps its messages in a subcollection in which user and AI turns alternate, and it stores inbox metadata, such as the latest message, on the conversation itself. Table II shows the customer record used by the CRM.

TABLE II
CUSTOMER RECORD IN THE CRM

Field	Purpose
id	Customer identifier
organizationId	Business that owns the record
name, phone	Contact details
status	NEW, CONTACTED, QUALIFIED, or CUSTOMER
tags, notes	Labels and owner's notes
lastMessage	Most recent message text
totalMessages	Count of messages exchanged
lastSeen	Time of last activity

E. Authentication and Data Separation

Business users sign in through Firebase Authentication with Google Sign-In [6], [8]. Protected backend routes sit behind JWT-based middleware, and organization identifiers keep one business's data apart from another's. We have not run a formal security assessment (Section VII-D).

V. METHODOLOGY

A. Message Processing Pipeline

A business is first onboarded with its details (name, location, time zone, working hours), signs in, and connects its WhatsApp account, after which webhook verification with Meta must succeed. For each incoming message the backend then does the following.

- 1) Receive the webhook event and identify the organization.
- 2) Find or create the customer record and add the message to the conversation.
- 3) Gather the conversation context and the business knowledge relevant to the message.
- 4) Build the prompt and send it to the LLM through OpenRouter.
- 5) Parse the output as either a general reply or a business action.
- 6) For an action, call the matching backend service (booking, cancellation, rescheduling, or another supported operation).
- 7) Store the outcome in Firestore.
- 8) Compose the final customer-facing message and send it through the WhatsApp Cloud API.
- 9) The dashboard reads the same Firestore data, so the owner sees the customer, conversation, and booking changes that result.

B. Prompt Construction and Knowledge Base

The knowledge base holds what the business enters itself: services, prices, timings, location, and similar facts. A prompt builder combines this material with the organization's details and the recent conversation. The aim is the property discussed in Section II-B, namely that answers reflect this business and not a generic one. The prototype does no conventional preprocessing or feature engineering. Relevant context is placed in the prompt, and the LLM is left to use it.

C. Action Parser

The model is asked to return structured output, and the action parser interprets it. A valid action, for instance a booking request with its details, is passed to the matching backend service. A valid general-response object is turned into a clean text message for the customer. The parser needed correction during development: a general reply that arrived as JSON could be forwarded to the customer as raw JSON, and so could malformed JSON. It now unwraps valid general-response JSON into plain text and handles malformed output safely instead of sending it to WhatsApp.

D. Appointment Operations

The booking module supports creating, cancelling, and rescheduling appointments and checking upcoming ones. Because the model only proposes an action, the backend is the component that reads and changes the booking data in Firestore. The owner can also cancel an appointment manually from the dashboard.

E. Human-AI Hybrid Handling

The Inbox is meant to let the owner step into a conversation. A human mode was designed for this, so that a person can take over from the AI on a given conversation. We describe it as a design, because it is not among the workflows tested so far.

VI. IMPLEMENTATION

A. Dashboard Modules

The dashboard is a React and TypeScript application built with Vite and styled with inline CSS in a dark theme. It has nine modules: Dashboard, Inbox, AI Builder, Analytics, CRM, Bookings, Knowledge Base, Billing, and Settings. The AI Builder is where a business configures and tests its agent, including a live test chat. The CRM supports customer listing, search, filters, summary statistics, a customer drawer, editing of name and status, notes, tags, and saving to the backend. A lead-assignment field (assigned To, holding a user ID) has been designed but is not finished.

VII. TESTING AND RESULTS

A. Testing Method

Testing was functional and integration testing done during development. The aim was to check that each service works and that complete customer workflows behave correctly, from a WhatsApp message, through Firestore, and back to the customer. Real WhatsApp messages and real Firestore operations were used.

B. Functional Results

Table IV lists the workflows exercised. Each one worked as intended in testing.

TABLE III
WORKFLOWS EXERCISED DURING TESTING

Area	Workflows	Outcome
WhatsApp	Webhook verification; incoming message reception; automatic reply delivery	Working
AI	Response generation; knowledge base use; AI agent; structured actions; malformed-output handling	Working
Booking	Book; cancel; reschedule; check upcoming; manual cancel from dashboard; check against Firestore state	Tested successfully
CRM	Create; retrieve; search; filter; edit; status; notes; tags; persistence	Working

VIII. FUTURE SCOPE

- 1) *Evaluation*: The first priority is measurement: action-recognition accuracy on a labelled set of messages in English, Hindi, and Marathi; end-to-end response time; the share of conversations resolved without human handover; and a pilot with a few real businesses.
- 2) *Real-time dashboard*: Firestore listeners [9] can keep the Inbox, Bookings, and CRM in step with the database.

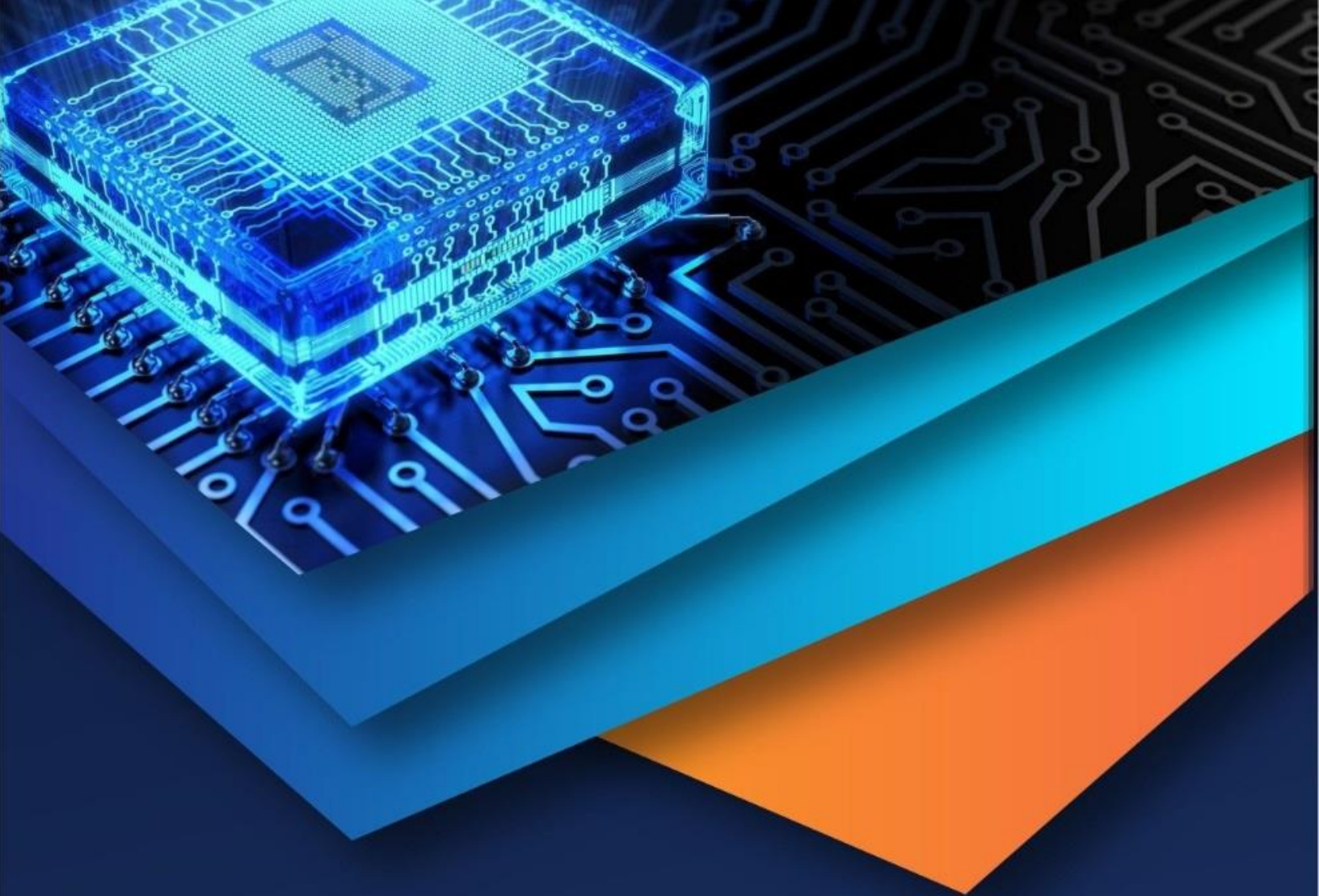
- 3) *Lead management*: Lead assignment, scoring, stages, a sales pipeline, and follow-up reminders would extend the CRM.
- 4) *Integrations*: Google Calendar, payment systems, email, SMS, and other CRM platforms are candidates. None has been settled.
- 5) *Richer agent*: The agent could take on order handling, payment status, reminders, lead qualification, and service recommendations, and could suggest replies to human operators.
- 6) *Scalability*: Queue-based message processing, database indexing, and horizontal scaling of the backend are the obvious routes. The production architecture has not been decided.

IX. CONCLUSIONS

We described WaBot.ai, a WhatsApp-based platform that pairs an LLM with a knowledge base, a CRM, and appointment workflows for Indian MSMEs. The prototype receives customer messages, replies through the WhatsApp Cloud API, and carries out booking operations through a parser that turns model output into backend actions. Functional testing of these workflows was successful. We have no measurements yet of accuracy, speed, or user satisfaction, and we have chosen not to report any. Producing those measurements and finishing the real-time dashboard are the next steps. If the results hold up, the central idea of the work, letting the model interpret and letting ordinary code act, looks like a practical route to automating small-business customer contact.

REFERENCES

- [1] A. Vaswani et al., "Attention is all you need," in Advances in Neural Information Processing Systems 30, 2017, pp. 5998–6008.
- [2] T. B. Brown et al., "Language models are few-shot learners," in Advances in Neural Information Processing Systems 33, 2020, pp. 1877–1901.
- [3] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Advances in Neural Information Processing Systems 33, 2020, pp. 9459–9474.
- [4] S. Yao et al., "ReAct: Synergizing reasoning and acting in language models," in Proc. Int. Conf. Learning Representations (ICLR), 2023.
- [5] T. Schick et al., "Toolformer: Language models can teach themselves to use tools," in Advances in Neural Information Processing Systems 36, 2023.
- [6] Google, Firebase Authentication. [Online]. Available: <https://firebase.google.com/docs/auth>
- [7] Google, Cloud Firestore. [Online]. Available: <https://firebase.google.com/docs/firestore>
- [8] Google, Authenticate Using Google Sign-In with JavaScript. [Online]. Available: <https://firebase.google.com/docs/auth/web/google-signin>
- [9] Google, Get Realtime Updates with Cloud Firestore. [Online]. Available: <https://firebase.google.com/docs/firestore/query-data/listen>
- [10] Meta, WhatsApp Business Platform Documentation. [Online]. Available: <https://developers.facebook.com/docs/whatsapp/>
- [11] OpenJS Foundation, Express.js: Fast, Unopinionated, Minimalist Web Framework for Node.js. [Online]. Available: <https://expressjs.com/>
- [12] Google Cloud, Quickstart: Build and Deploy a Node.js Web App to Cloud Run. [Online]. Available: <https://docs.cloud.google.com/run/docs/quickstarts/build-and-deploy/deploy-nodejs-service>
- [13] Google Cloud, The Node.js Runtime in Cloud Run. [Online]. Available: <https://docs.cloud.google.com/run/docs/runtimes/nodejs>



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)