Automated Doctor Appointment Agent Using N8N

by

Md. Abu Zubaer
ID: CSE2301028103

Md. Hasan Ali
ID: CSE2301028112

Md. Asif Bhuiyan
ID: CSE2301028187

Supervised by
Salma Tabashum

Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SONARGAON UNIVERSITY (SU)
September 2026

APPROVAL

The [thesis/project] titled "**Smart Hospital Appointment Management System Using AI and Workflow Automation** " submitted by Md. Abu Zubaer (CSE2301028103), Md. Hasan Ali (CSE2301028112) and Md. Asif Bhuiyan (CSE2301028187) to the Department of Computer Science and Engineering, Sonargaon University (SU), has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering and approved as to its style and contents.

Board of Examiners

Salma Tabashum
Associate Professor and Head,
Department of Computer Science and Engineering
Sonargaon University (SU)

Supervisor

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 1

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 2

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 3

DECLARATION

We, hereby, declare that the work presented in this report is the outcome of the investigation performed by us under the supervision of **Salma Tabashum, Lecturer** and **Coordinator,** Department of Computer Science and Engineering, Sonargaon University, Dhaka, Bangladesh. We reaffirm that no part of this [project] has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned

(Salma Tabashum)
Supervisor

Signature

Md. Abu Zubaer
ID: CSE2301028103

Md. Hasan ali
ID: CSE2301028112

Md. Asif Bhuiyan
ID: CSE2301028187

ABSTRACT

Managing doctor appointments in healthcare remains challenging due to long wait times, manual triage errors, and scheduling conflicts. This thesis presents a Smart Doctor Appointment Agent leveraging workflow automation and conversational artificial intelligence to streamline hospital scheduling.

Built on the n8n automation engine, the system integrates the OpenAI Chat Model (gpt-5-mini), Google Sheets, and Google Calendar. Operating via a chat interface, the AI agent parses unstructured patient inputs and decodes phonetic symptom descriptions across Bangla, English, and Banglish (e.g., *"buke betha"*). Using deterministic function calling, the agent searches a Google Sheets database to match symptoms with specialized doctors and chamber schedules.

To prevent double-booking, the workflow performs real-time calendar availability checks locked strictly to the Asia/Dhaka (+06:00) timezone. After collecting patient metadata and receiving final user confirmation, the agent automatically creates a live appointment entry in Google Calendar.

By orchestrating AI reasoning with backend APIs, this low-code architecture operates 24/7, reduces administrative overhead, and eliminates schedule drift, delivering an efficient, friction-free booking experience for modern healthcare administration.

ACKNOWLEDGMENT

At the very beginning, we would like to express our deepest gratitude to the Almighty Allah for giving us the ability and the strength to finish the task successfully within the scheduled time.

We are auspicious that we had the kind association as well as supervision of **Salma Tabashum**, Lecturer & Coordinator, Department of Computer Science and Engineering, Sonargaon University whose hearted and valuable support with best concern and direction acted as necessary resource to carry out our project.

We are also thankful to all our teachers during our whole education, for exposing us to the beauty of learning.

Finally, our deepest gratitude and love to our parents for their support, encouragement, and endless love.

TABLE OF CONTENTS

Title Page No.

DECLARATION ……..…….. ii

ABSTRACT ……...……. iii

ACKNOWLEDGEMENT ……..…… iv

CHAPTER 1 10-12

AI POWERED AUTOMATED DOCTOR BOOKING SYSTEM

1.1 Introduction……..……... 10

1.2 Objectives of the Project ……...…….. 10-11

1.3 Articulatory or Distinctive Phonetic Feature Extraction ……..…….. 11

1.4 System Architecture and Node Details ……..……. 11-12

CHAPTER 2 13-15

USER INPUT TRIGGER NODE

2.1 Introduction…….…..... 13

2.2 Mechanism and Functionality of the Chat Trigger Node ……... 13

2.3 Importance of the Trigger Node for Users …….. 13

2.4 JSON Configuration Payload of the Trigger Node ……. 13-14

2.5 JSON Code ……. 14

2.6 User Interaction and Chat Interface …… 15

CHAPTER 3 16-23

MAIN AI AGENT NODE (DOCTOR APPOINTMENT AGENT)

3.1 Introduction ……..... 16

3.2 System Architecture ……... 16-17

3.3 JSON Code …….……... 17-18

3.4 System Instructions and Prompt Architecture …… 18-22

3.5 Key Functional Capabilities of the Main AI Agent Node …….. 22-23

CHAPTER 4 24-26

OPERATIONAL ARCHITECTURE OF THE OPENAI CHAT MODEL NODE

4.1 Introduction ……..……... 24

4.2 System Configuration Highlights …...…… 24-25

4.3 OpenAI Input and Output ……... 25

4.4 Extended Capabilities of the OpenAI Chat Model Node ……... 25-26

4.5 Selected Models and Parameters in OpenAI Architecture ……. 26

CHAPTER 5 27-30

STATEFUL CONTEXT MANAGEMENT (SIMPLE MEMORY NODE)

5.1 Short-Term Dialogue State Retention …… 27

5.2 Functional Mechanism and Role of the Simple Memory Node …… 27-28

5.3 Dynamic Buffer Sliding …….. 29

5.4 Payload Pre-Processing for Agent Node …… 29

5.5 Graceful Exception Recovery …….……... 29

5.6 Temporal Context Alignment ..……... 30

5.7 Context-Driven Safety Guardrails …….. 30

CHAPTER 6 31-33

DATASET ARCHITECTURE AND DOCTOR DATABASE CONFIGURATION

6.1 Database Configuration ……. 31

6.2 Technical Specifications of the Doctor Dataset …….. 31

6.3 Dataset …… 32

6.4 Dataset Description ……. 32

6.5 Mechanism of LLM-Driven Dataset Querying in n8n ……... 33

6.6 Dataset and Database Integration Summary ……... 33

CHAPTER 7 34-35

REAL-TIME SLOT VERIFICATION VIA GOOGLE CALENDAR NODE

7.1 Role of “Get Many Events” in Google Calendar Node ……. 34

7.2 Get Many Events Description ……. 34

7.3 Uses Parameters …….. 35

CHAPTER 8 36-39

AUTOMATED BOOKING VIA GOOGLE CALENDAR

8.1 Technical Overview of the Event Creation Workflow …… 36

8.2 Key Capabilities of the Create Event Node …… 36-37

8.3 Automated Event Payload and Metadata Structure …… 37

8.4 Figure Input and Output ……. 38

8.5 Description …….. 38

8.6 Uses Parameters …….. 39

CHAPTER 9 40

DYNAMIC TIMEZONE NORMALIZATION (ASIA/DHAKA)

9.1 Asia/Dhaka Timezone Binding and Offset Drift Prevention …….. 40

9.2 Dynamic Date-Range Bounds for Schedule Querying ……... 40

CHAPTER 10 41-44

RESULTS AND EXPLANATION

10.1 Agent Welcome and Service Initialization ……. 41

10.2 Patient Query and Symptom Input ……. 42

10.3 Google Sheets Query and Doctor/Schedule Suggestion ……. 43

10.4 Patient Date and Slot Confirmation …… 43

10.5 Final Calendar Booking and Agent Confirmation …….. 44

CHAPTER 11 45-57

FINAL SOURCE CODE AND WORKFLOW

11.1 Source Code …… 45-56

11.2 Final Workflow ……... 57

CHAPTER 12 58-59

SYSTEM EVALUATION, IMPACT, AND FUTURE SCOPE

12.1 Real-World Impact and Clinical Utility …….. 58

12.2 Architectural Advantages Over Legacy Systems ……... 58-59

12.3 Future Enhancements and System Roadmap …….. 59

REFERENCES ……..…….. 60

Chapter 1

AI Powered Automated Doctor Booking System

1.1 Introduction

The rapid evolution of artificial intelligence and workflow automation has transformed modern healthcare administration, offering seamless solutions to traditional scheduling bottlenecks. This project presents a smart, conversational doctor appointment agent designed to automate the complete patient booking pipeline through an integrated eight-node system architecture. By leveraging a central AI agent powered by the Open AI LLM and Simple Memory, the system intelligently interprets patient symptoms, identifies medical specialties, and queries a Google Sheets doctor database for matching specialists. It further validates real-time slot availability via Google Calendar, gathers patient credentials, and secures explicit confirmation before finalizing bookings, ensuring a conflict-free, highly efficient healthcare scheduling experience.

1.2 Objectives of the Project

The primary objective of this project is to design and implement an automated, AI-driven hospital appointment management system that simplifies patient booking and eliminates operational friction in healthcare administration. To achieve this overarching goal, the specific sub-objectives are defined as follows:

- **Automate Symptom-Based Triage:** Utilize Large Language Models (Open AI LLM) to analyze patient-described symptoms in natural language (Bangla, English, and Banglish) and accurately map them to relevant medical specialties.
- **Integrate Multi-Platform Workflow Automation:** Seamlessly connect conversational AI with Google Sheets for real-time doctor database queries and Google Calendar for dynamic schedule tracking using an 8-node n8n workflow.
- **Prevent Scheduling Conflicts:** Implement automated real-time availability checks before booking to eliminate double-booking errors and schedule overlaps.
- **Enhance Patient Experience:** Provide a continuous, context-aware interaction using window buffer memory and enforce explicit confirmation steps to ensure data accuracy and user convenience.

1.3 Articulatory or Distinctive Phonetic Feature Extraction

In our Smart Doctor Appointment System, Articulatory and Distinctive Phonetic Feature Extraction plays a crucial role in the Prompt Engineering framework of the Open AI LLM-powered AI agent. When users interact with the system using voice inputs or unstructured text transcriptions—often in diverse accents, regional dialects, or mixed languages (Bangla, English, and Banglish)—raw text alone often lacks phonetic accuracy. By integrating phonetic feature extraction concepts into our system's Prompt Engineering, the LLM is instructed to look beyond surface-level spellings and analyze the underlying phonetic structure of user inputs.

Through prompt-level instructions, the agent maps non-standard phonetic variations, misspellings, and colloquial medical descriptions (such as *"buke betha"*, *"feverish"*, or phonetically transliterated symptoms) to their corresponding medical concepts. Distinctive features—such as manner and place of articulation—help the model disambiguate phonetically similar terms and categorize symptoms accurately into specific medical specialties (e.g., Cardiology, Dermatology, Orthopedics). This prompt-engineered phonetic mapping ensures that the AI agent reliably interprets user intent regardless of spelling irregularities, resulting in accurate doctor database queries (Google Sheets) and seamless calendar scheduling.

1.4 System Architecture and Node Details

Total work flow >Patient → AI Chat → Symptoms → Google Sheet → Doctor Suggestion → Doctor Schedule → Google Calendar Availability → Patient Details → Confirmation → Calendar Booking .

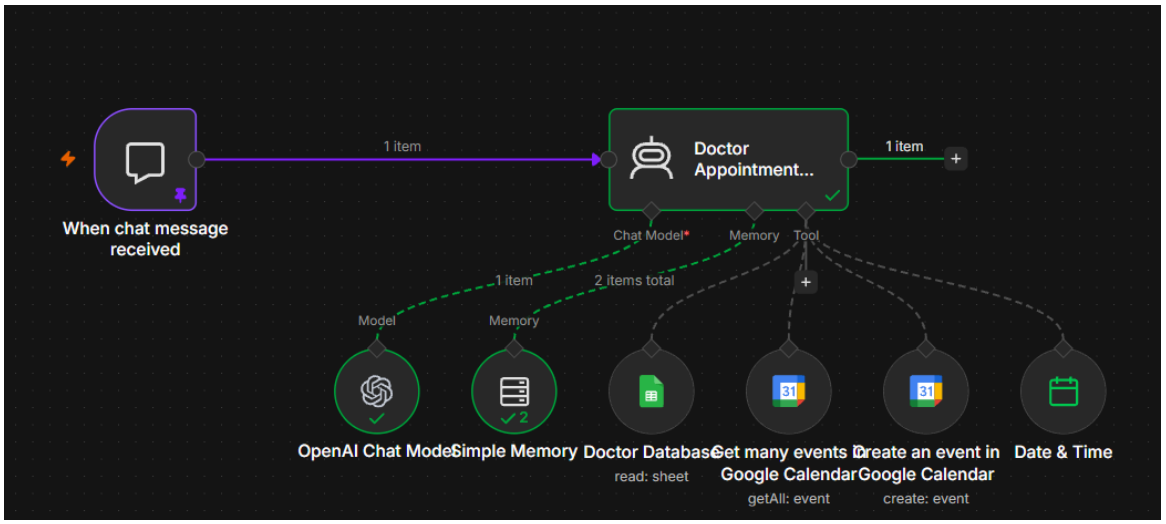


Figure 1

1. Trigger Node (When chat message received): Acts as the entry point of the system. It listens for incoming chat messages from patients, extracts the raw text payload, and triggers the entire workflow.

2. Main AI Orchestrator (Doctor Appointment Agent): Serves as the central decision-making brain. It manages conversation logic, determines user intent, and dynamically coordinates between connected memory and external tool modules.

3. LLM Model Node (Open AI chat Model): Provides high-speed Natural Language Processing (NLP). Using prompt engineering, it interprets symptoms across Bangla, English, and Banglish to identify the required medical specialty.

4. Window Buffer Memory (Simple Memory): Maintains conversational context across turns. It stores short-term chat history so users do not need to repeat details like their name or preferred time slot.

5. Doctor Database Tool (Doctor Database - read: sheet): Integrates with Google Sheets to query doctor profiles, medical specialties, practice days, and duty hours based on the identified specialty.

6. Calendar Availability Tool (Get many events in Google Calendar): Performs real-time schedule checks against Google Calendar to verify slot availability and prevent double-booking conflicts.

7. Event Creation Tool (Create an event in Google Calendar): Executes the final booking transaction by writing the confirmed appointment, patient metadata, and doctor details into Google Calendar.

8. Utility Tool (Date & Time): Normalizes conversational time expressions (e.g., "tomorrow," "next Monday") into standardized ISO date-time formats required for database and calendar operations.

Chapter 2
User Input Trigger Node

2.1 Introduction
The User Input Trigger Node serves as the primary gateway of the workflow, activated whenever a patient sends a chat message. It captures unstructured text inputs, extracts payload metadata, and initiates the automated pipeline, passing user queries directly to the central AI orchestrator for immediate processing.

2.2 Mechanism and Functionality of the Chat Trigger Node
The When chat message received node operates as the event-driven starting mechanism for the entire automated appointment workflow. Configured as a Webhook listener, it remains in a passive state until an incoming HTTP POST request or web chat payload is generated by a patient's message. Upon activation, the node extracts critical raw data including the plain text query, sender identification, session tokens, and timestamp metadata—and parses it into a structured JSON payload. It then validates the input structure to ensure data integrity before transmitting the payload downstream to the central AI Agent node. By continuously monitoring the chat interface, this node eliminates manual intervention, establishes a low-latency connection between the user interface and the backend processing pipeline, and guarantees that every patient query instantly initiates the automated symptom analysis and booking routine.

2.3 Importance of the Trigger Node for Users

The When chat message received node is essential for patients because it provides an immediate, 24/7 accessible entry point to healthcare services. By automatically capturing user messages in real time, it eliminates tedious manual steps, such as waiting on long phone hold times or filling out complex booking forms. Users can simply type their health concerns in natural language, and this node instantly connects them to the intelligent system. It ensures a low-latency, friction-free interaction, making the process of finding a doctor and securing an appointment effortless, intuitive, and accessible at any time.

2.4 JSON Configuration Payload of the Trigger Node

To maintain system reproducibility and facilitate technical integration, the raw JSON configuration of the When chat message received node specifies its event-driven trigger properties, webhook settings, and payload schema.

Input :

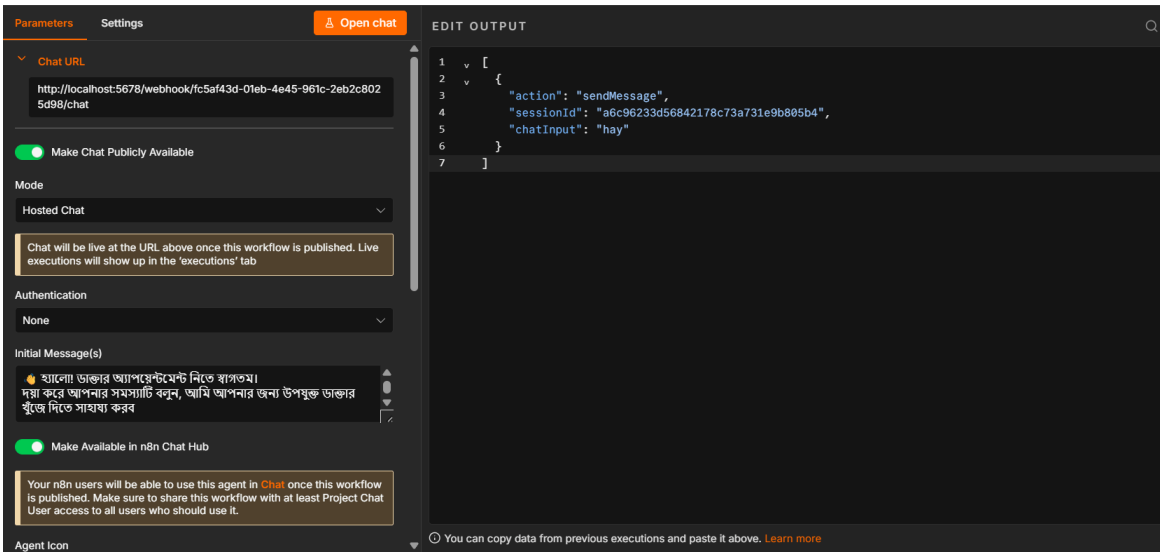


Figure 2

2.5 JSON CODE

```
/* Placeholder for Chat Trigger Node JSON Code */
{
  "meta": {
    "instanceId": "n8n_chat_trigger_node"
  },
  "nodes": [
    {
      "parameters": {
        "options": {}
      },
      "type": "@n8n/n8n-nodes-langchain.chatTrigger",
      "typeVersion": 1.1,
      "position": [
        0,
        0
      ],
      "id": "chat-trigger-node-01",
      "name": "When chat message received"
    }
  ]
}
```

Figure 3

Out put :

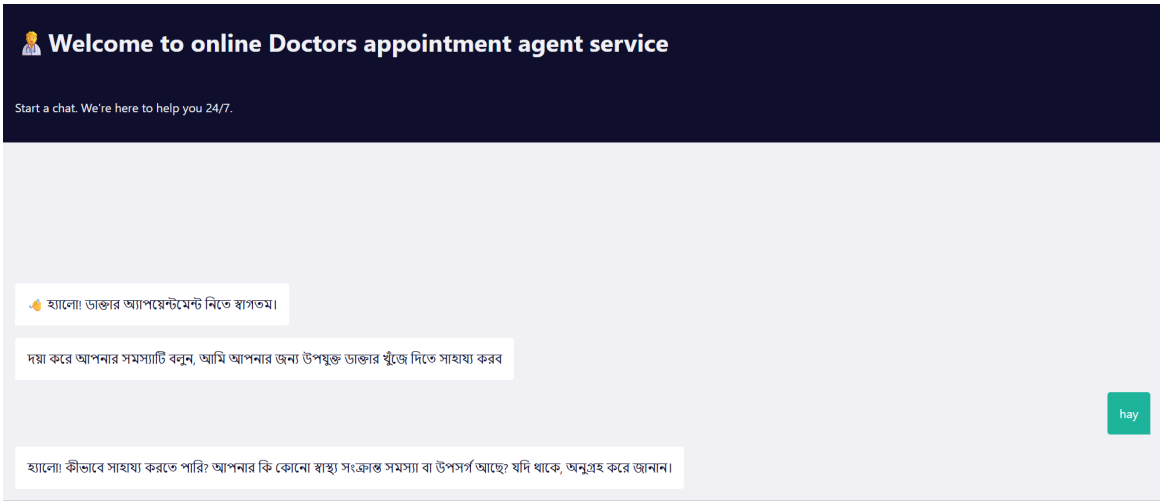


Figure 4

2.6 User Interaction and Chat Interface

The Chat Trigger node provides an interactive messaging interface where patients send their health concerns, and the system immediately displays the AI agent's generated responses in real time. This bidirectional chat flow ensures instant communication, giving users a smooth, transparent, and conversational experience while booking their hospital appointments.

Chapter 3

Main AI Agent Node (Doctor Appointment Agent)

3.1 Introduction

presents the system architecture, design methodology, and implementation framework of the AI-powered smart hospital appointment agent. Built on the n8n automation platform, the system establishes a seamless bridge between conversational user interfaces and backend healthcare databases. This chapter details the technical orchestration of the eight core nodes, focusing on how Large Language Model reasoning, custom prompt engineering, and external API integrations collaborate in real time. It covers the dynamic triage mechanism, phonetic variation handling, database querying via Google Sheets, real-time schedule validation through Google Calendar, and overall error-handling strategies. Together, these methodological components demonstrate a robust, conflict-free, and scalable solution for modern healthcare appointment automation.

3.2 System Architecture

The architecture of the Smart Hospital Appointment Agent is structured as an integrated 8-node event-driven pipeline on the n8n automation platform. It seamlessly connects natural language user interactions with backend scheduling logic and relational tools. Figure 5 illustrates the architectural daa flow across the pipeline.

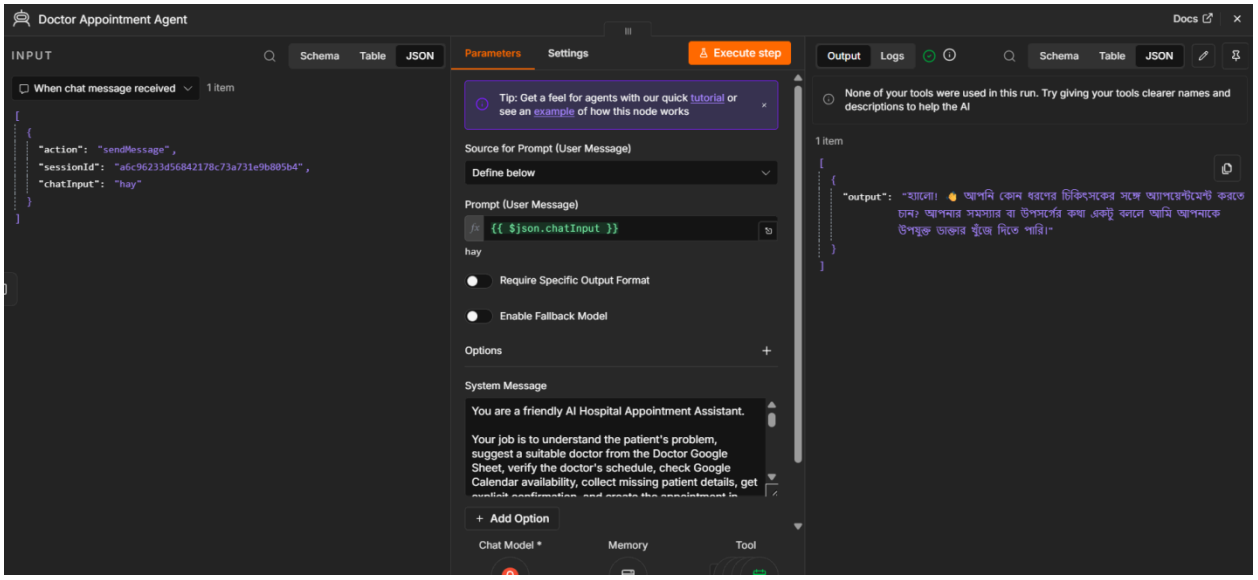
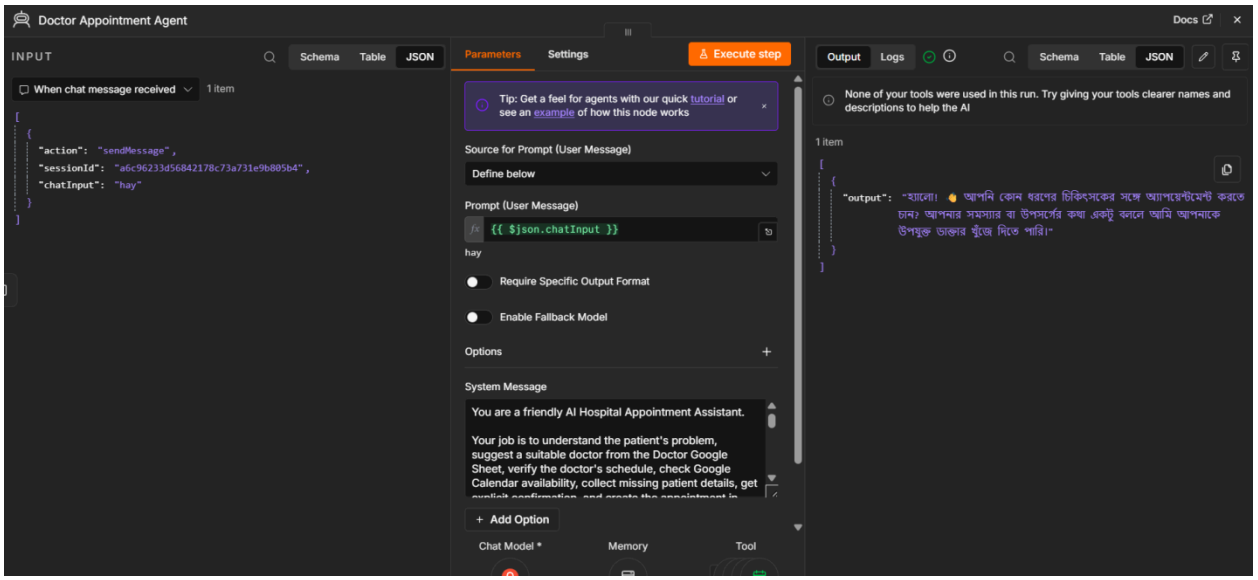


Figure 5



3.3 Json Code

Input **:**

[

{

"action": "sendMessage",

"sessionId": "a6c96233d56842178c73a731e9b805b4",

"chatInput": "hay"

}

]

Output:

[

{

"output": "হ্যালো! 👋 আপনি কোন ধরণের চিকিৎসকের সঙ্গে অ্যাপয়েন্টমেন্ট করতে চান? আপনার সমস্যার বা উপসর্গের কথা একটু বললে আমি আপনাকে উপযুক্ত ডাক্তার খুঁজে দিতে পারি।"

}

]

3.4 System Instructions & Prompt Architecture

The System Prompt is integrated into the AI Agent to enforce operational rules, output formatting, and medical domain logic. It guides the LLM to classify symptoms accurately, handle multi-language phonetic variations (Bangla/Banglish), direct tool calls efficiently, and ensure safe, structured doctor appointment scheduling without generating invalid or hallucinated responses.

System Message :

You are a friendly AI Hospital Appointment Assistant.

Your job is to understand the patient's problem, suggest a suitable doctor from the Doctor Google Sheet, verify the doctor's schedule, check Google Calendar availability, collect missing patient details, get explicit confirmation, and create the appointment in Google Calendar.

LANGUAGE

- Bangla → reply in Bangla.

- Banglish/Roman Bangla → reply in natural Banglish.

- English → reply in English.

- Mixed language → naturally follow the patient's style.

- Keep replies short, friendly and human-like.

MEDICAL SAFETY

- You are an appointment assistant, not a doctor.
- Never diagnose or confirm a disease.
- Never prescribe medicine.
- Use symptoms only to identify a suitable medical specialty.
- For obvious emergencies, advise the patient to seek urgent medical care.

DOCTOR DATABASE

- Google Sheets is the ONLY source for doctor information.
- Always use the Doctor Database tool when doctor information is needed.
- Never invent a doctor, hospital, specialty, category, working day or working time.
- Recommend only doctors returned by the Google Sheet.

SYMPTOM → DOCTOR

- Understand the patient's problem in Bangla, Banglish or English.
- Identify the most relevant specialty/category.
- Search the Doctor Database.
- Normally suggest 1–3 suitable doctors.
- Let the patient choose.

DATE & TIME

- Use Asia/Dhaka timezone.
- Use the Date & Time tool for today, tomorrow, next Sunday, etc.
- Never guess relative dates.
- Display time as hh:mm AM/PM.
- Never book a past date/time.

DOCTOR SCHEDULE

Before accepting a requested appointment:

1. Verify the doctor exists in Google Sheets.

2. Verify the doctor works on that date.

3. Verify the requested time is within the doctor's working hours.

4. Never invent another schedule.

CALENDAR AVAILABILITY

- Google Calendar is the source of truth for appointment availability.

- ALWAYS check Calendar before confirming an appointment.

- Only treat overlapping events as conflicts.

- If booked, tell the patient and suggest another valid time.

- Never assume a slot is available.

- Do not repeatedly call the same tool for information already obtained.

PATIENT DETAILS

Before booking, collect only:

- Patient Name

- Mobile Number

- Problem/Symptoms

Do not ask again for information already provided.

CONFIRMATION

Before creating an appointment, show:

Patient:

Mobile:

Problem:

Doctor:

Hospital:

Date:

Time:

Then ask:

"Apni ki appointment-ta confirm korte chan?"

Do NOT create the appointment until the patient explicitly confirms.

BOOKING

After explicit confirmation, create the Google Calendar event ONLY if:

- Doctor is verified.
- Schedule is verified.
- Calendar availability is verified.
- Patient name, mobile and problem are available.
- Patient confirmed.

Calendar event title:

Doctor Appointment - [Patient Name] - [Doctor Name]

Calendar description:

Patient Name: [Name]

Mobile: [Mobile]

Problem: [Problem]

Doctor: [Doctor]

Hospital: [Hospital]

Date: [Date]

Time: [Time]

After successful Calendar creation, reply in the same chat:

Appointment Confirmed

Patient:

Doctor:

Hospital:

Date:

Time:

Problem:

Only say "Appointment Confirmed" if the Calendar event was successfully created.

If Calendar creation fails, do not claim success.

IMPORTANT

- Google Sheet = Doctor Database ONLY.
- Google Calendar = Patient Appointment Database.
- Do not use Gmail or send email.
- Never expose another patient's information.
- Never invent tool results.
- Keep responses concise and natural.

3.5 Key Functional Capabilities of the Main AI Agent Node

- Phonetic Symptom Triage: Maps colloquial and phonetic complaints (e.g., "buke betha", "matha ghora") directly to accurate clinical specialty categories.
- Contextual Multi-Turn Memory: Maintains dialogue state across turns, ensuring user details like name or appointment time are remembered without repeated prompts.
- Autonomous Tool Routing: Dynamically decides when to query the Google Sheets database versus checking Google Calendar schedules based on the user's intent.

- Date & Time Standardization: Parses relative user time expressions (e.g., "agami kal", "next Tuesday at 4pm") into precise ISO date-time formats for accurate scheduling.
- Conflict & Guardrail Management: Enforces strict scheduling policies to prevent double-booking while ensuring response safety by strictly avoiding off-scope or non-medical queries.

Chapter 4

Operational Architecture of the OpenAI Chat Model Node

4.1 Architecture of the OpenAI Chat Model

The OpenAI Chat Model node, powered by gpt-5-mini, functions as the core processing engine inside the n8n AI Agent workflow. It continuously converts raw conversational text into structured system actions through a structured reasoning loop. Upon receiving user input alongside multi-turn chat history from Simple Memory, the model parses patient intent and executes phonetic analysis to extract critical parameters such as preferred dates, physician names, and symptoms. It decodes non-standard Banglish inputs, mapping informal phrases like *"buke betha"* to precise medical specialties. Instead of generating plain text directly, the node determines if backend data is required and issues deterministic function calls to query the Google Sheets doctor database or check Google Calendar availability in the Asia/Dhaka timezone. After receiving API payloads, it synthesizes raw JSON data into natural, polite responses that strictly comply with system safety directives and prompt rules.

4.2 System Configuration Highlights (Based on OpenAI Node Setup)

- System Prompt & Safety Directives: The node enforces strict system instructions where the assistant acts purely as a scheduling helper, explicitly prohibiting medical diagnosis or prescriptions. It mandates using Google Sheets as the sole source for doctor information and requires natural language matching across Bangla, English, and Banglish styles while adhering strictly to Asia/Dhaka timezone logic.
- API Execution & Response Schema: Operating with the Use Responses API parameter enabled, the node seamlessly integrates OpenAI's structured response format and function calling. It routes parsed conversational variables directly into active tool schemas, allowing the AI Agent to determine when to trigger external tools versus generating a direct chat response to the user.

- Token Utilization & Context Efficiency: The model manages multi-turn dialogue context efficiently while maintaining low computational overhead. Runtime execution metrics show a balanced distribution—processing approximately 1,597 prompt tokens and generating 254 completion tokens (totaling 1,851 tokens)—ensuring fast, sub-second response delivery during real-time patient appointment booking sessions.

4.3 Open AI

Input & output :

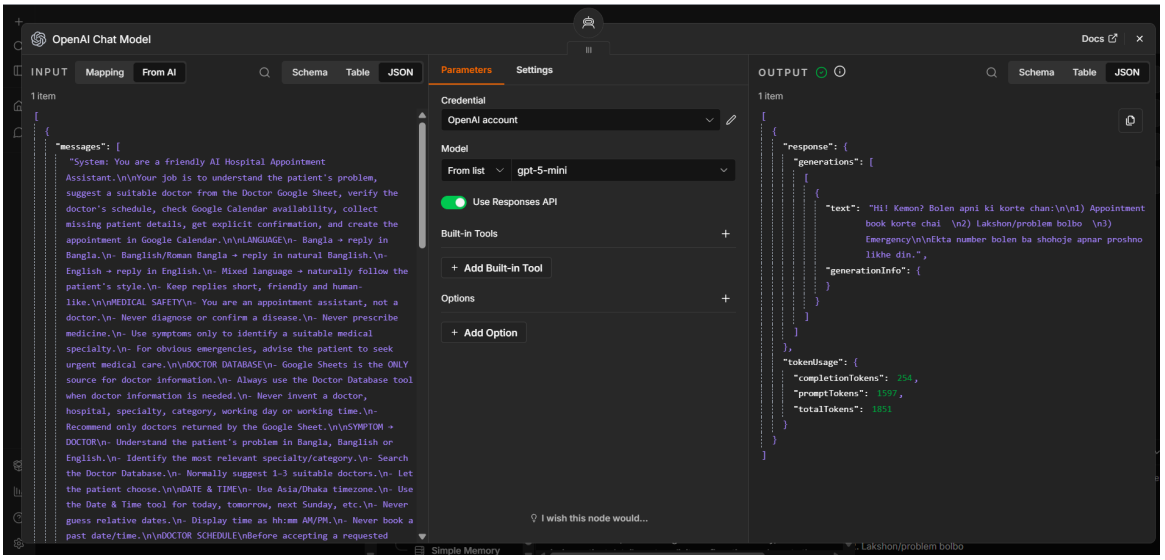


Figure 6

4.4 Extended Capabilities of the OpenAI Chat Model Node

- **Optimized API Inference:** Powered by OpenAI's highly optimized cloud infrastructure, this node executes deep neural network logic with near-instantaneous, sub-second responses. This minimizes conversational latency during patient interactions, delivering fluid and natural dialogue transitions that match human reaction times.
- **Deterministic Tool Calling:** The OpenAI model evaluates natural language inputs alongside defined system schemas to execute external functions with precision. By passing targeted parameters to downstream tools—such as Google

Sheets for doctor lookups and Google Calendar for slot availability—it ensures reliable API interactions and eliminates unstructured system outputs.

- **Phonetic & Multilingual Processing:** Specially configured to decode complex multilingual inputs, colloquial phrases, and non-standard transliterations (such as Bangla, English, and Roman Bangla/Banglish). The node extracts hidden clinical context from phonetic symptom descriptions like *"buke betha"*, converting raw patient complaints into accurate medical specialty categories for targeted database querying.
- **Structured System Prompting & Safety Compliance:** Operates under strict system prompt guidelines (enforcing medical safety rules, preventing self-diagnosis/prescription, and mandating Asia/Dhaka timezone checks). It formats operational data into validated JSON structures, guaranteeing that appointment dates, patient details, and doctor IDs are passed securely across nodes without generating hallucinated responses.
- **High-Throughput Context Retention:** Efficiently retains multi-turn conversational context while processing complex logic flows. It analyzes patient intent across multiple dialogue turns without incurring token processing delays, allowing the agent to dynamically collect patient details, confirm booking schedules, and manage slot reservations smoothly.

4.5 Selected Models and Parameters in OpenAI Architecture

During system implementation, testing, and performance validation, the following OpenAI model and parameter configurations are integrated into the OpenAI Chat Model node:

- **Primary LLM Model:**
  - **gpt-5-mini (OpenAI):** Deployed as the core operational engine for multi-turn reasoning, Banglish/Bangla natural language understanding, phonetic symptom classification, and function calling. It offers an ideal balance of low latency, high throughput, and cost-effective performance.
- **Node Configuration & Parameters:**
  - **Credential:** Authenticated securely via OpenAI account credentials inside n8n.
  - **Use Responses API:** Enabled toggle ensuring full support for OpenAI's structured API response schema and function execution pipeline.
  - **System Message Directives:** Enforces explicit operational rules, including strict reliance on the Google Sheets doctor database, prohibition of medical prescriptions, natural Banglish/Bangla tone matching, and dynamic date processing via the Asia/Dhaka timezone.
  - **Token Allocation & Usage Limits:** Configured to balance context intake and brief response generation (observed runtime metrics: ~1,597 prompt tokens, ~254 completion tokens, and ~1,851 total tokens per standard turn).

Chapter 5

Stateful Context Management(Simple MemoryNode)

5.1 Short-Term Dialogue State Retention

The Simple Memory Node functions as the stateful context retention layer within the n8n pipeline, preserving multi-turn conversational history. By maintaining a dynamic rolling buffer of recent exchanges, it allows the AI Agent to remember patient details, track booking parameters, and manage logical follow-ups across continuous interactions without exceeding language model context thresholds.

5.2 Functional Mechanism and Role of the Simple Memory Node

The Simple Memory Node functions as the dedicated state retention mechanism within the n8n agent workflow, bridging the gap between stateless API executions and continuous human conversation. By intercepting incoming payloads and dynamically appending historical turn data, it preserves operational context throughout the appointment scheduling process.

- Multi-Turn Context Persistence: Maintains a continuous dialogue thread by recording sequential user prompts and agent responses within the active session lifecycle. This prevents conversational resets, enabling fluid interaction as patients navigate multi-step booking procedures.
- Slot and Entity Storage: Captures and retains critical slot variables—such as selected medical specialties, specific doctor preferences, patient contact details, and proposed appointment slots—eliminating the need for repetitive user inputs during multi-step workflows.
- Contextual Correction Handling: Enables the core AI Agent to process dynamic modification requests (e.g., "Actually, change that time to 5 PM" or "Make it next Tuesday instead") by cross-referencing incoming user updates against previously established state parameters.

- Token Window Optimization: Employs a sliding window memory architecture that restricts retained context to a predetermined number of recent turns. This optimizes token consumption, maintains low latency processing speeds, and prevents context-window overflow during extended patient interactions.

  Input:

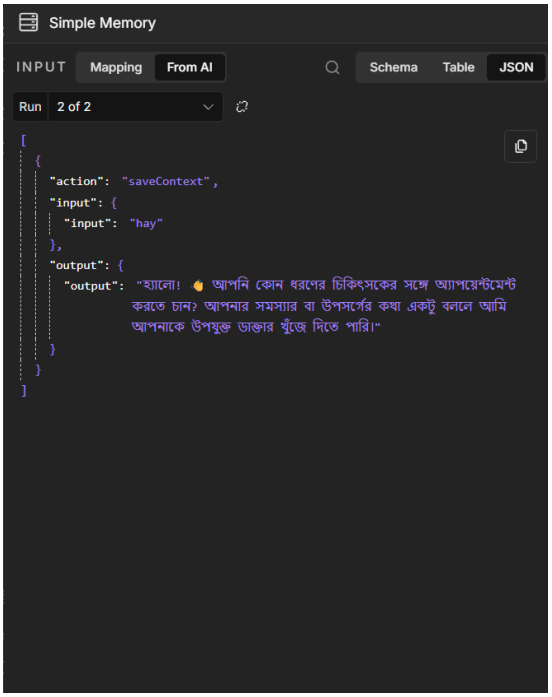
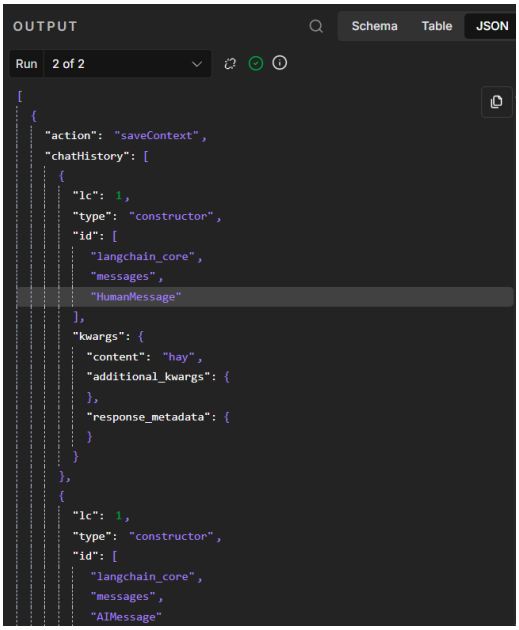
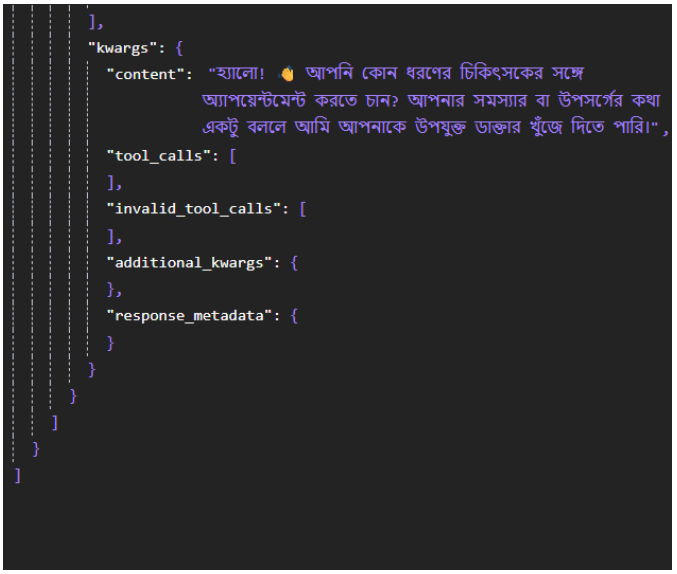


Figure 7

Output :





5.3 Dynamic Buffer Sliding

Implements an automated window sliding mechanism that continuously discards the oldest dialogue turns once the predefined message threshold is reached. This sliding window prevents linear context expansion, keeping payload sizes predictable and maintaining ultra-low inference latency for the Groq processing engine across prolonged multi-turn booking sessions.

5.4 Payload Pre-Processing for Agent Node

Automatically intercepts, structures, and formats stored conversational turn histories into strict role-based array payloads (user and assistant) prior to forwarding data into the AI Agent. By converting raw multi-turn dialogue into standardized message objects, it ensures the Groq reasoning engine instantly differentiates prior system turns from the immediate patient input. This structured array format eliminates payload parsing ambiguities, prevents system prompt injection, and guarantees that downstream tool execution decisions are made with clean, properly ordered contextual inputs. Additionally, this pre-processing step optimizes the input context by removing redundant metadata, allowing the model to focus purely on relevant conversational tokens for sub-second inference.

5.5 Graceful Exception Recovery

Protects active booking sessions from unexpected workflow execution drops, network timeouts, or downstream tool failures. By maintaining the patient's existing session state independently of tool execution outcomes, the node preserves all previously extracted variables—such as patient credentials, selected specialties, and doctor preferences. If an external API query fails, the AI Agent can leverage this retained context to prompt the user for a simple retry without forcing them to re-enter their information from the beginning.

5.6 Temporal Context Alignment

Preserves relative time markers mentioned in earlier conversational turns (such as "this afternoon" or "next Friday") alongside their original timestamp references. This structural memory pairing enables downstream processing tools to accurately compute static ISO dates when executing calendar scheduling queries in subsequent steps.

5.7 Context-Driven Safety Guardrails

Restricts the AI Agent from loops or repeated system behaviors by supplying a transcript of previous assistant attempts within the active turn state. If a user provides invalid input repeatedly, the agent uses stored context history to alter its phrasing strategy, guiding the patient effectively toward valid inputs.

Chapter 6

Dataset Architecture and Doctor Database Configuration

6.1 Database Configuration

The foundation of the AI Agent's decision-making system relies on a structured, real-time database hosted within Google Sheets. This dataset acts as the single source of truth for doctor profiles, medical specialties, visiting hours, chamber locations, and appointment rules. Below is the technical breakdown of how the dataset is structured and integrated into the n8n automation pipeline.

6.2 Technical Specifications of the Doctor Dataset

- **Relational Field Schema & Mapping:** The dataset enforces strict data typing across columns, including Doctor_ID, Doctor_Name, Specialty, Visiting_Days, Time_Slot, Hospital_Name, and Consultation_Fee. This structured layout allows the n8n Google Sheets node to perform exact lookup operations based on intent parameters extracted by the LLM.
- **Phonetic & Specialty Indexing:** Doctor records are categorized under primary clinical domains (e.g., Cardiology, Neuromedicine, Pediatrics) and common colloquial terms. This structure enables the AI Agent to query the dataset using both formal medical terms and patient symptom descriptions (e.g., mapping "buke betha" to Cardiology profiles).
- **Dynamic Schedule & Slot Alignment:** Visiting hours and active working days are structured as standardized strings (e.g., Sat-Mon-Wed | 10:00 AM - 09:00 PM). This format allows the downstream scheduling logic to validate open appointment windows before initiating calendar bookings.
- **Real-Time Synchronous Querying:** Acting as a lightweight database backend, the dataset processes asynchronous Lookup Rows requests from the n8n tool node. This guarantees sub-second data retrieval without requiring complex SQL infrastructure or high-latency external API calls.

6.3 Data set :

Doctor Name	Hospital	Specialty	Category
Prof. Dr. A B M Abdullah (অধ্যাপক ডা. এ বি এম আবদুল্লাহ)	BSMMU (PG Hospital)	Medicine Specialist	General Medicine & Primary Care
Prof. Dr. Md. Rezwanul Haque (অধ্যাপক ডা. মো. রেজাওয়ানুল হক)	DMC	Medicine & Cardiology	General Medicine & Primary Care
Prof. Dr. A K M Helaluddin (অধ্যাপক ডা. এ কে এম হেলালউদ্দিন)	Ibn Sina Medical College & Hospital	Medicine Specialist	General Medicine & Primary Care
Prof. Dr. A F M Anwar Hossain (অধ্যাপক ডা. এ এফ এম আনোয়ার হোসেন)	Bangladesh Specialized Hospital	Medicine & Diabetes	General Medicine & Primary Care
Prof. Dr. S M Mustafa Zaman (অধ্যাপক ডা. এস এম মুস্তফা জামান)	BSMMU (PG Hospital)	Cardiology	Heart & Vascular Care
Prof. Dr. Md. Shahabuddin (অধ্যাপক ডা. মো. শাহাবুদ্দিন)	DMC	Interventional Cardiology	Heart & Vascular Care
Prof. Dr. Md. Mosaddeque Hossain (অধ্যাপক ডা. মো. মোসাদ্দেক হোসেন)	Ibn Sina Medical College & Hospital	Cardiology	Heart & Vascular Care
Prof. Dr. M Zahid Hossain (অধ্যাপক ডা. এম জাহিদ হোসেন)	Bangladesh Specialized Hospital	Clinical & Interventional Cardiology	Heart & Vascular Care
Prof. Dr. Md. Sarwar Alam (অধ্যাপক ডা. মো. সারওয়ার আলম)	BSMMU (PG Hospital)	Clinical Oncology	Specialized Care & Critical/Pain Management
Prof. Dr. Md. Mozzafar Hossain (অধ্যাপক ডা. মো. মোজাফফর হোসেন)	DMC	Anesthesiology, Analgesia & ICU	Specialized Care & Critical/Pain Management
Prof. Dr. Md. Abdur Rahman (অধ্যাপক ডা. মো. আব্দুর রহমান)	Ibn Sina Medical College & Hospital	Anesthesiology, Pain & ICU	Specialized Care & Critical/Pain Management

DAY 1	Day 1 Start Time	Day 1 End Time	DAY 2	Day 2 Start Time	Day 2 End Time
Sunday	10:00 AM	1:00 PM	Wednesday	10:00 AM	1:00 PM
Monday	10:00 AM	1:00 PM	Thursday	10:00 AM	1:00 PM
Tuesday	10:00 AM	1:00 PM	Saturday	10:00 AM	1:00 PM
Friday	10:00 AM	1:00 PM	Sunday	1:00 PM	4:00 PM
Sunday	4:00 PM	7:00 PM	Thursday	1:00 PM	4:00 PM
Monday	1:00 PM	4:00 PM	Wednesday	1:00 PM	4:00 PM
Tuesday	1:00 PM	4:00 PM	Saturday	1:00 PM	4:00 PM
Friday	1:00 PM	4:00 PM	Thursday	4:00 PM	7:00 PM
Wednesday	4:00 PM	7:00 PM	Saturday	4:00 PM	7:00 PM
Sunday	7:00 PM	10:00 PM	Tuesday	4:00 PM	7:00 PM
Friday	4:00 PM	7:00 PM	Monday	4:00 PM	7:00 PM

Figure 8

6.4 Data set Description :

This tool contains the hospital's doctor database.

Use this tool whenever you need doctor information.

This is the ONLY source of truth for:

- Doctor Name

- Hospital

- Specialty

- Category

- Working Days

- Working Hours

Never invent doctor information.

Always use this tool before recommending or validating a doctor.

6.5 Mechanism of LLM-Driven Dataset Querying in n8n

The n8n AI Agent calls the Google Sheets dataset dynamically using function-calling protocols rather than loading the entire database into memory during every chat cycle. When a user sends a query, such as requesting a doctor for a specific symptom or time, the core Groq LLM parses the input, extracts key parameters like specialty or day, and identifies the exact intent. The agent then triggers the Google Sheets Tool node, passing these extracted entities into the search filter schema. Executing a targeted Lookup Rows operation against the spreadsheet, the node queries specific columns—such as Specialty, Visiting_Days, and Time_Slot—to filter matching entries. Once retrieved, the node formats the matching doctor profiles, hospital names, and available hours into a structured JSON payload, returning it directly to the agent's context window. This real-time payload integration allows the AI Agent to generate precise, hallucination-free scheduling options and present them back to the user seamlessly.

6.6 Dataset & Database Integration Summary

The dataset acts as the real-time relational backbone for the n8n Smart Doctor Appointment Agent, hosted within Google Sheets to ensure lightweight, serverless maintenance. It maintains a structured schema across critical doctor attributes, including Doctor_ID, Doctor_Name, Specialty, Visiting_Days, Time_Slot, Hospital_Name, and Consultation_Fee. This deterministic column arrangement enables seamless parameter mapping, allowing the system to categorize records under formal medical specialties and common symptom descriptions.

When a patient initiates a chat, the central Groq LLM parses the request, extracts key parameters (such as required specialty or preferred schedule), and issues a targeted function call to the Google Sheets Tool node. Rather than loading the entire spreadsheet into memory, the node executes a synchronous Lookup Rows operation using strict search filters. The matching rows are formatted into a clean JSON payload and injected directly back into the AI Agent's active context window.

Chapter 7

Real-Time Slot Verification via Google Calendar Node

7.1 Role of "Get Many Events" in Google Calendar Node

In the n8n AI agent workflow, the "Get Many Events" operation within the Google Calendar node serves as the primary live dynamic schedule verification mechanism. Rather than relying on static database records, this operation queries the physician's calendar in real time to fetch all existing bookings within a specified time boundary. When a patient requests an appointment for a specific date, the core LLM extracts the target timeframe and passes it to the calendar node using strict start (After) and end (Before) parameters. Executing a single batch API query, the node retrieves booked calendar slots for that duration, trimming raw metadata down to essential parameters such as event summary, start time, and end time. This lightweight JSON payload is then injected back into the AI agent's active memory context. By cross-referencing requested appointment windows against already scheduled events within the local Asia/Dhaka timezone, the system identifies potential overlap conflicts before any record is finalized. Consequently, this node actively eliminates double-booking errors, validates real-time doctor availability, and allows the agent to propose verified, open time slots seamlessly.

7.2 Get Many Events Description

Prompts:

Use this tool to check Google Calendar availability BEFORE confirming an appointment.

Check only the requested appointment date and time.

If an existing calendar event overlaps the requested appointment time, the slot is unavailable.

Never assume a slot is available.

Never confirm an appointment without checking this tool first.

7.3 Uses Parameters

>Credential

>Tool Description

>Description

>Resource

>Operation

>Calendar

>Limit

>After

[DateTime: 2026-09-06T09:22:57.718-04:00]

>Before

[DateTime: 2026-09-13T09:22:57.718-04:00]

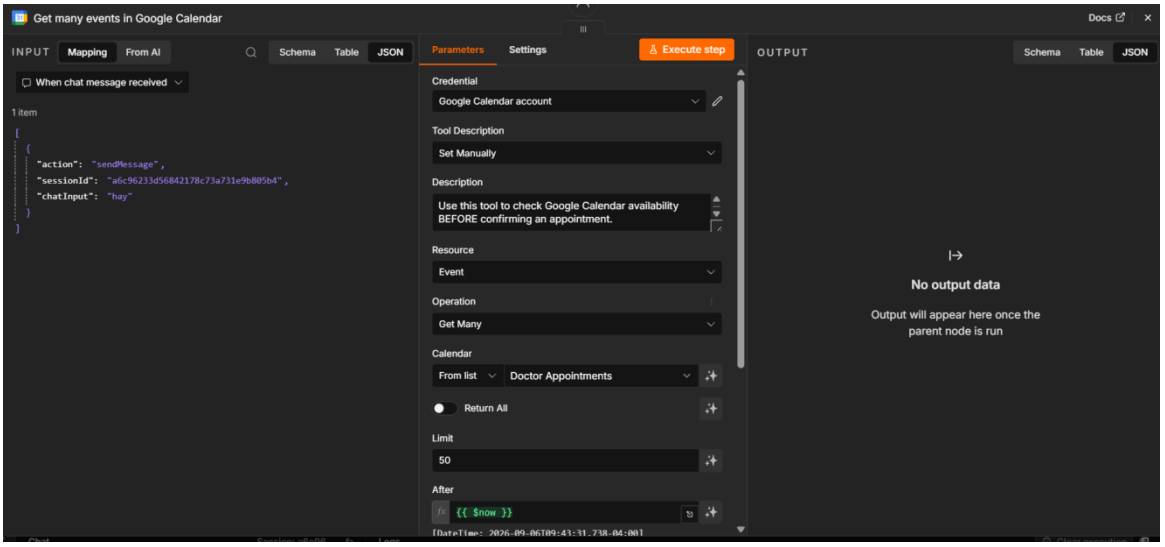


Figure 9

Chapter 8

Automated Booking via Google Calendar

8.1 Technical Overview of the Event Creation Workflow

Calendar node serves as the final execution stage in the n8n Smart Doctor Appointment Agent workflow. Once a patient selects an available time slot and confirms their appointment details, the AI Agent packages the extracted parameters into a structured JSON payload and dispatches a secure API request to the physician's calendar. This operation automatically provisions a live calendar entry, mapping critical variables such as the patient's name, symptom description, exact consultation window, and meeting location directly into the database. By utilizing dynamic timezone normalization (Asia/Dhaka) and binding the event ID to the specific doctor's calendar route, the system ensures real-time synchronization, eliminates scheduling collisions, and triggers instant confirmation responses back to the user interface.

8.2 Key Capabilities of the Create Event Node

- **Dynamic Payload Assembly and Parameter Mapping:** The node extracts structural fields validated by the AI Agent—including Patient_Name, Contact_Number, Doctor_ID, Start_Time, and End_Time—and maps them directly into Google Calendar API attributes. This parameter mapping guarantees zero manual data entry errors during final appointment recording.
- **ISO-8601 Timestamp Generation & Timezone Binding:** Automatically formats appointment durations into valid ISO-8601 strings, enforcing explicit Asia/Dhaka time offsets. This prevents timestamp drifting across server execution environments and ensures calendar entries align perfectly with local clinical hours.
- **Multi-Calendar D**

  [INSERT SCREENSHOT 8.1: n8n 'Create Event' Node Configuration Interface]

  ynamic Routing: Binds the target calendar address dynamically based on the

doctor profile returned from Google Sheets. By overriding default calendar IDs with individual doctor calendar emails, the node guarantees that created appointments populate the exact physician's schedule.

- **Instant Conflict-Free Confirmation Output:** Generates a unique event ID upon successful API commit, returning a standardized response payload. The n8n engine feeds this confirmation back into the active chat session, assuring the patient that their slot is securely locked in real time.

8.3 Automated Event Payload & Metadata Structure

The "Create Event" node structures verified booking parameters into a standardized JSON object before dispatching the payload to the Google Calendar API. It dynamically formats the event summary as Appointment: [Patient Name] with [Doctor Name] for clear schedule identification, while embedding essential patient details—such as phone number, primary symptoms, and fee status—directly into the description body. To ensure precise scheduling, start and end timestamps are generated as strict ISO-8601 strings mapped explicitly to the Asia/Dhaka timezone offset (+06:00), preventing temporal discrepancies. Additionally, the location attribute is populated dynamically from the Google Sheets dataset with the specific hospital name and chamber room number, guaranteeing accurate, complete metadata for every finalized appointment.

Figure 8.1: Parameter mapping setup inside n8n showing dynamic expressions for summary, description, and start/end time properties.

[INSERT SCREENSHOT 8.2: Successfully Created Google Calendar Event Entry]

Figure 8.2: Real-time Google Calendar entry created by the agent showing patient details and localized Asia/Dhaka slot alignment.

8.4 Figure input & output

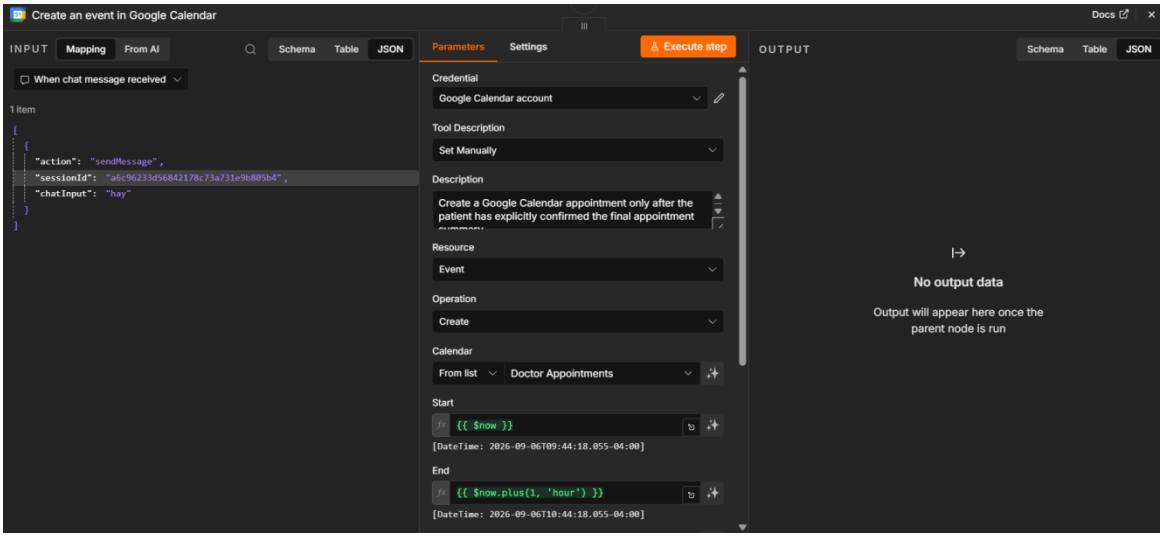


Figure 10

8.5 Description

Create a Google Calendar appointment only after the patient has explicitly confirmed the final appointment summary.

Before using this tool, all of these must already be verified:

- Doctor exists in Doctor Database
- Doctor works on the requested date
- Requested time is within working hours
- Calendar availability has been checked
- No conflicting appointment exists
- Patient name is available
- Patient mobile number is available
- Patient problem is available
- Patient explicitly confirmed

Never create an appointment before explicit confirmation.

Never create a duplicate appointment .

8.6 Uses Parameters

>Credential

>Tool Description

>Description

>Resource

>Operation

>Calendar

>Limit

>After

[DateTime: 2026-09-06T09:56:13.423-04:00]

>Before

[DateTime: 2026-09-06T10:56:13.423-04:00]

>Additional Fields Description

Patient Name: [Patient Name] Mobile: [Mobile Number] Problem: [Problem] Doctor: [Doctor Name] Hospital: [Hospital] Appointment Date: [Appointment Date] Appointment Time: [Appointment Time]

Chapter 9

Dynamic Timezone Normalization (Asia/Dhaka)

9.1 Asia/Dhaka Timezone Binding and Offset Drift Prevention

This subsection details the configuration required to bind the system's execution context specifically to the Asia/Dhaka timezone. Because cloud-hosted n8n instances and external API services frequently execute on Coordinated Universal Time (UTC) default servers, explicit timezone normalization is mandatory. By injecting fixed UTC offset parameters (+06:00) directly into the node settings, the architecture eliminates schedule drifting, ensuring that all query parameters and calendar bookings strictly reflect local Bangladesh Standard Time (BST) clinical operating hours.

9.2 Dynamic Date-Range Bounds for Schedule Querying

This section focuses on the logic used to derive lower (After) and upper (Before) search boundaries for live calendar queries. Rather than retrieving an entire month of calendar records, the workflow calculates dynamic daily execution bounds using JavaScript date functions relative to the requested booking day. This precise boundary definition restricts query payloads strictly to the target 24-hour clinical operational window, significantly reducing network latency and ensuring optimal response times during real-time schedule verification.

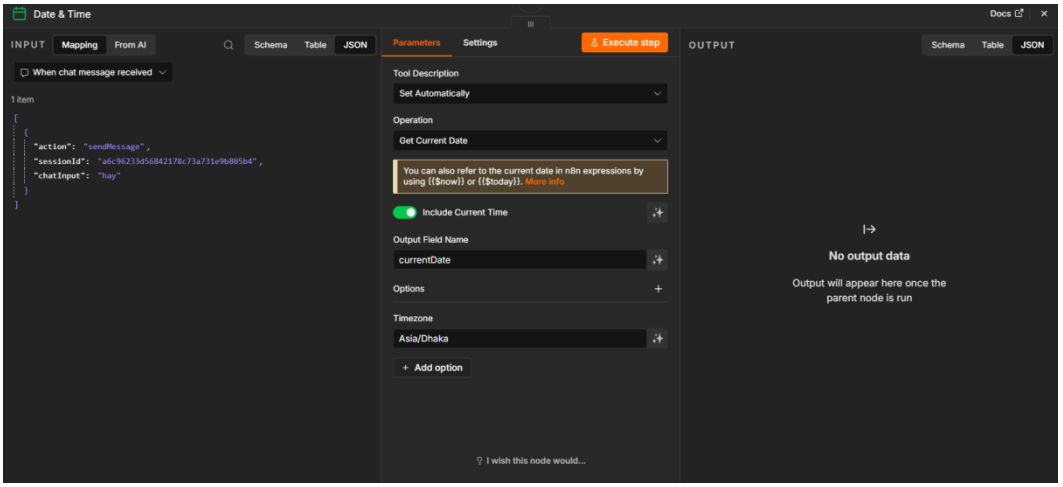


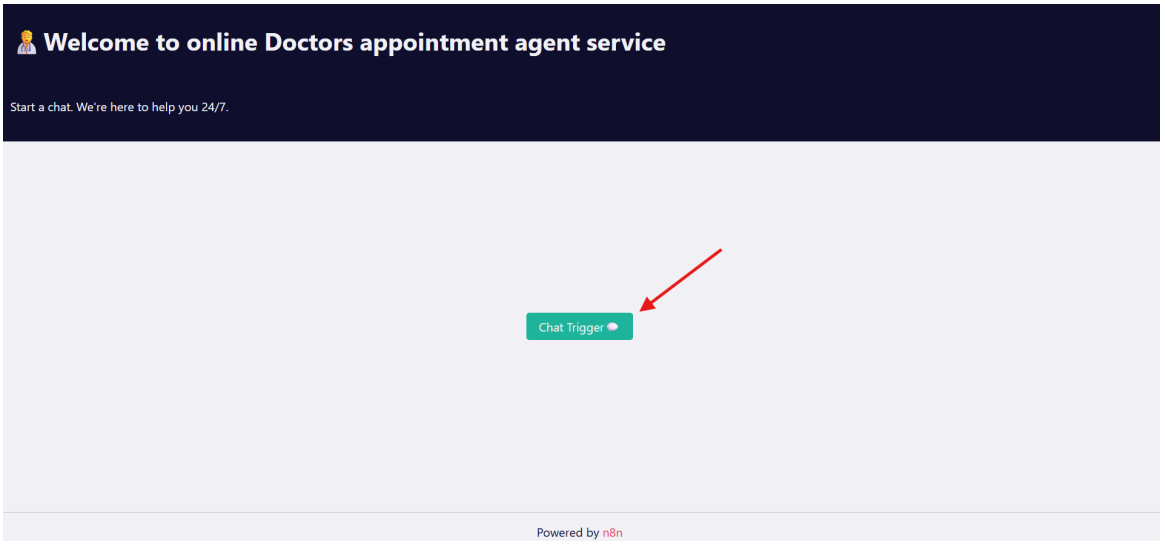
Figure 11

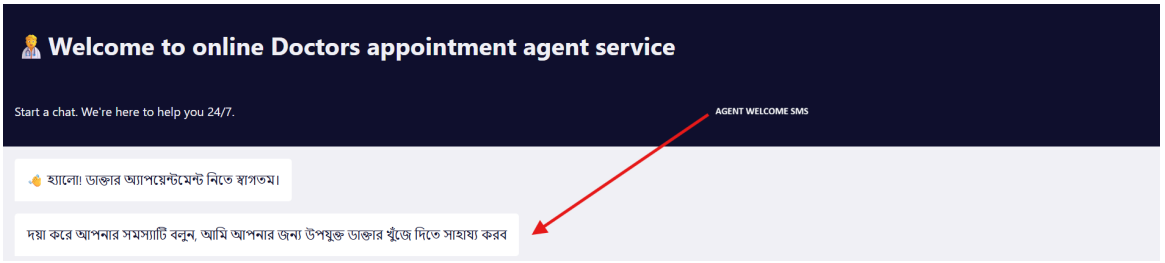
Chapter 10

Results & Explan

Step 1: Agent Welcome & Service Initialization

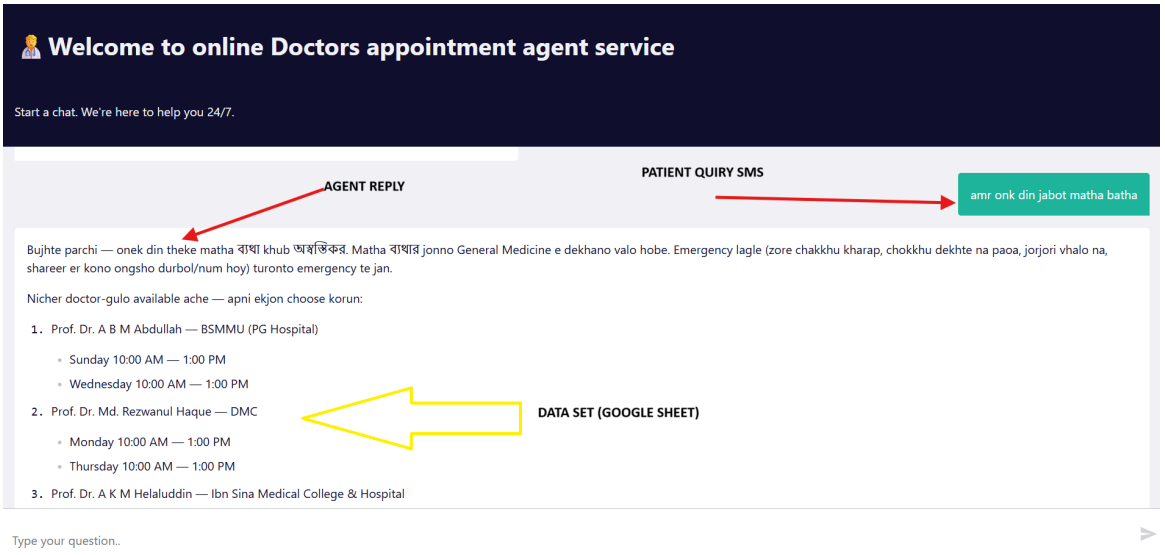
When the user opens the UI chat interface, the system initializes the session by displaying an automated welcome message, introducing the online doctor appointment agent, and prompting the user for their query or symptoms.

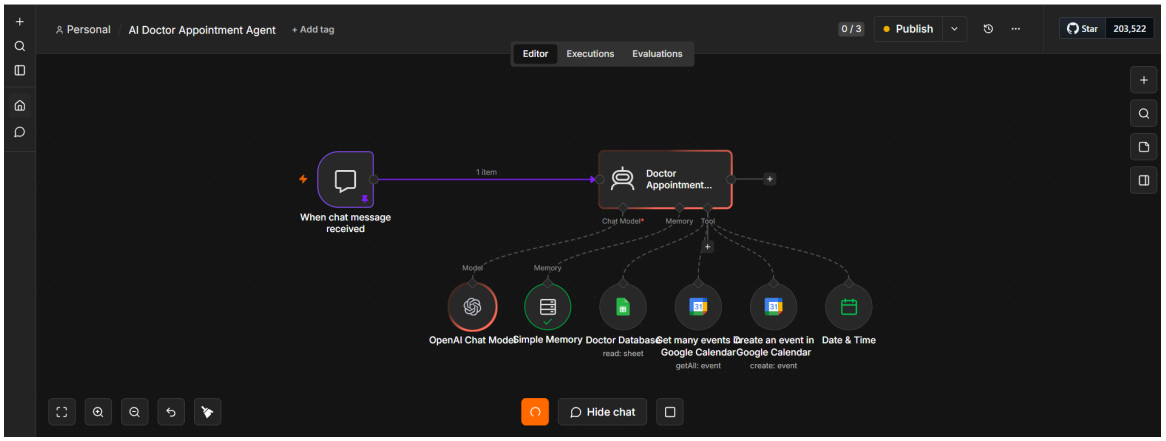




Step 2: Step 2: Patient Query & Symptom Input

The patient submits their initial message describing their current health condition, symptoms, or the medical specialty they are seeking.

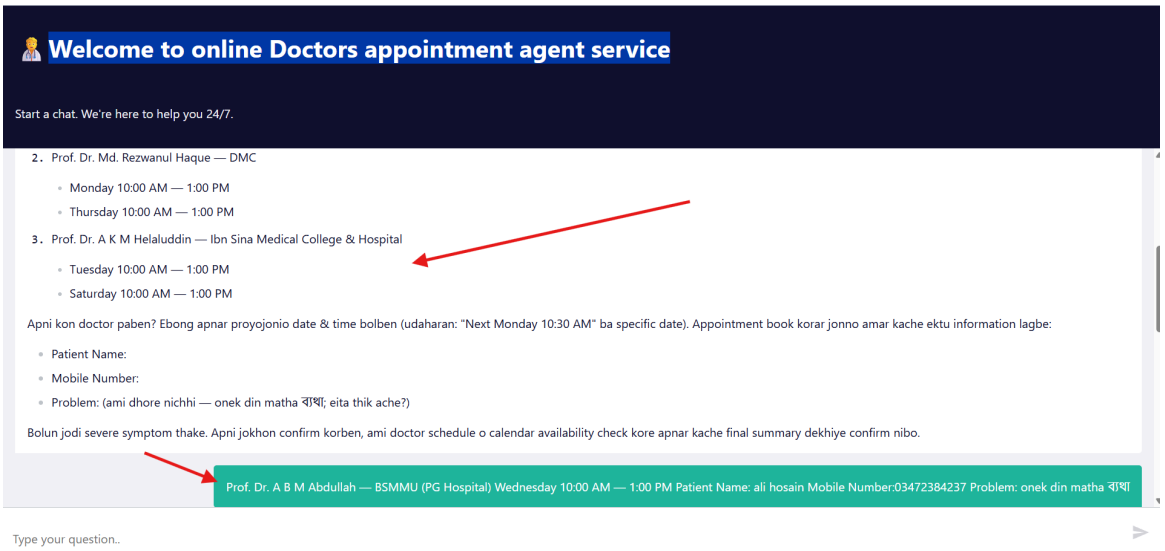




Backend Working process

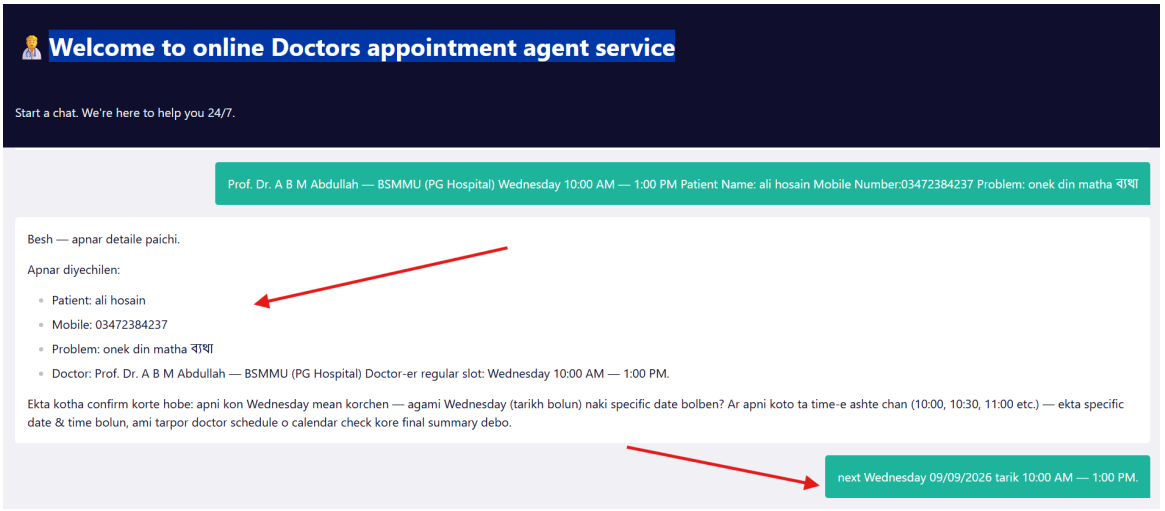
Step 3: Google Sheets Query & Doctor/Schedule Suggestion

The AI agent processes the patient's input, queries the Google Sheets dataset in real time, and returns a formatted list of relevant doctors, complete with chamber locations, available visiting days, and time slots.



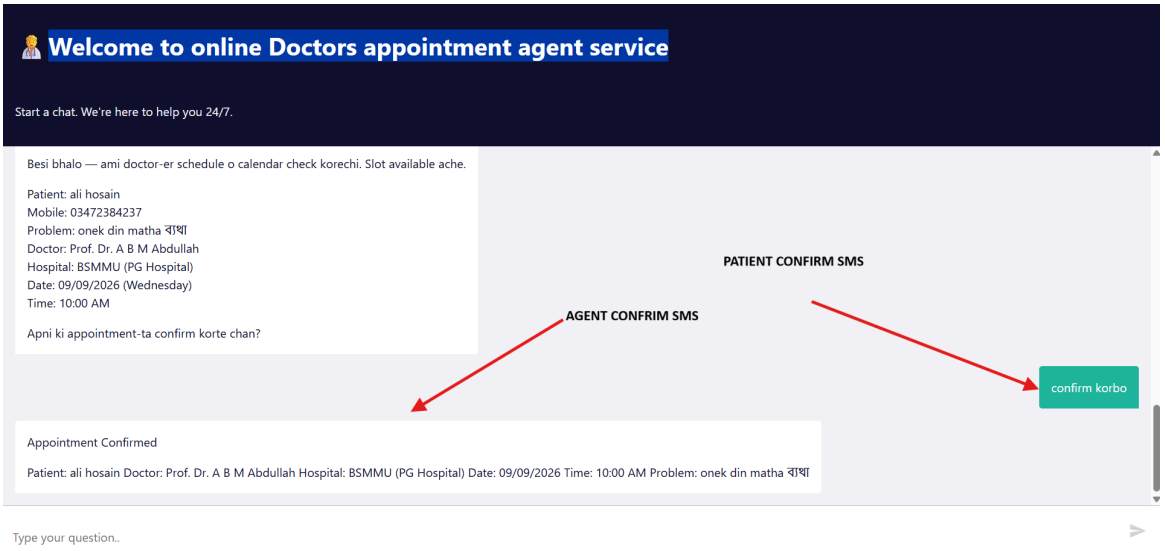
Step 4: Patient Date & Slot Confirmation

The patient selects their preferred doctor and submits their requested appointment date and time slot along with their contact details.



Step 5: Final Calendar Booking & Agent Confirmation

After checking availability via Google Calendar (Asia/Dhaka) and creating the event, the agent returns a final confirmation ticket in the chat with all booking details securely locked in.



Chapter 11

FINAL SOURCE CODE & WORKFLOW

11.1 Source Code

Jeson format:


```
{
 "nodes": [
  {
   "parameters": {
    "public": true,
    "initialMessages": "👋 হ্যালো! ডাক্তার অ্যাপয়েন্টমেন্ট নিতে স্বাগতম। \nদয়া করে আপনার সমস্যাটি বলুন, আমি আপনার জন্য উপযুক্ত ডাক্তার খুঁজে দিতে সাহায্য করব \n",
    "availableInChat": true,
    "options": {
     "showWelcomeScreen": true,
     "getStarted": "Chat Trigger💬",
     "title": "🧑‍⚕️Welcome to online Doctors appointment agent service  "
    }
   },
   "type": "@n8n/n8n-nodes-langchain.chatTrigger",
   "typeVersion": 1.4,
   "position": [
    -4576,
    560
   ],
   "id": "45c0188e-6e08-4c16-9193-1071d2db396b",
   "name": "When chat message received",
   "webhookId": "fc5af43d-01eb-4e45-961c-2eb2c8025d98"
  },
```

```
{
 "parameters": {
  "sessionIdType": "=fromInput",
  "sessionKey": "",
  "contextWindowLength": 1000
 },
 "type": "@n8n/n8n-nodes-langchain.memoryBufferWindow",
 "typeVersion": 1.4,
 "position": [
  -4224,
  784
 ],
 "id": "7c081695-4d42-4bc6-807a-e02e85101797",
 "name": "Simple Memory"
},
{
 "parameters": {
  "promptType": "define",
  "text": "={{ $json.chatInput }}",
  "options": {
    "systemMessage": "You are a friendly AI Hospital Appointment
```


Assistant.\n\nYour job is to understand the patient's problem, suggest a suitable doctor from the Doctor Google Sheet, verify the doctor's schedule, check Google Calendar availability, collect missing patient details, get explicit confirmation, and create the appointment in Google Calendar.\n\nLANGUAGE\n- Bangla → reply in Bangla.\n- Banglish/Roman Bangla → reply in natural Banglish.\n- English → reply in English.\n- Mixed language → naturally follow the patient's style.\n- Keep replies short, friendly and human-like.\n\nMEDICAL SAFETY\n- You are an appointment assistant, not a doctor.\n- Never diagnose or confirm a disease.\n- Never prescribe

medicine.\n- Use symptoms only to identify a suitable medical specialty.\n- For obvious emergencies, advise the patient to seek urgent medical care.\n\nDOCTOR DATABASE\n- Google Sheets is the ONLY source for doctor information.\n- Always use the Doctor Database tool when doctor information is needed.\n- Never invent a doctor, hospital, specialty, category, working day or working time.\n- Recommend only doctors returned by the Google Sheet.\n\nSYMPTOM → DOCTOR\n- Understand the patient's problem in Bangla, Banglish or English.\n- Identify the most relevant specialty/category.\n- Search the Doctor Database.\n- Normally suggest 1–3 suitable doctors.\n- Let the patient choose.\n\nDATE & TIME\n- Use Asia/Dhaka timezone.\n- Use the Date & Time tool for today, tomorrow, next Sunday, etc.\n- Never guess relative dates.\n- Display time as hh:mm AM/PM.\n- Never book a past date/time.\n\nDOCTOR SCHEDULE\nBefore accepting a requested appointment:\n1. Verify the doctor exists in Google Sheets.\n2. Verify the doctor works on that date.\n3. Verify the requested time is within the doctor's working hours.\n4. Never invent another schedule.\n\nCALENDAR AVAILABILITY\n- Google Calendar is the source of truth for appointment availability.\n- ALWAYS check Calendar before confirming an appointment.\n- Only treat overlapping events as conflicts.\n- If booked, tell the patient and suggest another valid time.\n- Never assume a slot is available.\n- Do not repeatedly call the same tool for information already obtained.\n\nPATIENT DETAILS\nBefore booking, collect only:\n- Patient Name\n- Mobile Number\n- Problem/Symptoms\n\nDo not ask again for information already provided.\n\nCONFIRMATION\nBefore creating an appointment, show:\n\nPatient:\nMobile:\nProblem:\nDoctor:\nHospital:\nDate:\nTime:\n\nThen ask:\n\"Apni ki appointment-ta confirm korte chan?\"\n\nDo NOT create the appointment until the patient explicitly confirms.\n\nBOOKING\nAfter explicit

confirmation, create the Google Calendar event ONLY if:\n- Doctor is verified.\n- Schedule is verified.\n- Calendar availability is verified.\n- Patient name, mobile and problem are available.\n- Patient confirmed.\n\nCalendar event title:\nDoctor Appointment - [Patient Name] - [Doctor Name]\n\nCalendar description:\nPatient Name: [Name]\nMobile: [Mobile]\nProblem: [Problem]\nDoctor: [Doctor]\nHospital: [Hospital]\nDate: [Date]\nTime: [Time]\n\nAfter successful Calendar creation, reply in the same chat:\n\nAppointment Confirmed\n\nPatient:\nDoctor:\nHospital:\nDate:\nTime:\nProblem:\n\nOnly say \"Appointment Confirmed\" if the Calendar event was successfully created.\n\nIf Calendar creation fails, do not claim success.\n\nIMPORTANT\n- Google Sheet = Doctor Database ONLY.\n- Google Calendar = Patient Appointment Database.\n- Do not use Gmail or send email.\n- Never expose another patient's information.\n- Never invent tool results.\n- Keep responses concise and natural."
      }
    },
    "type": "@n8n/n8n-nodes-langchain.agent",
    "typeVersion": 3.1,
    "position": [
      -4096,
      560
    ],
    "id": "38dab8ed-53af-448e-b8be-80b5ca318f80",
    "name": "Doctor Appointment Agent"
  },
  {
    "parameters": {
      "descriptionType": "manual",

"toolDescription": "This tool contains the hospital's doctor database.\n\nUse this tool whenever you need doctor information.\n\nThis is the ONLY source of truth for:\n- Doctor Name\n- Hospital\n- Specialty\n- Category\n- Working Days\n- Working Hours\n\nNever invent doctor information.\n\nAlways use this tool before recommending or validating a doctor.",
"documentId": {
"__rl": true,
"value": "1BGQ1uf5vm7MoGDH8JhlMrbBYVz-yzrOtjphOMQydlYI",
"mode": "list",
"cachedResultName": "Doctor Database",
"cachedResultUrl": "https://docs.google.com/spreadsheets/d/1BGQ1uf5vm7MoGDH8JhlMrbBYVz-yzrOtjphOMQydlYI/edit?usp=drivesdk"
},
"sheetName": {
"__rl": true,
"value": "gid=0",
"mode": "list",
"cachedResultName": "Sheet1",
"cachedResultUrl": "https://docs.google.com/spreadsheets/d/1BGQ1uf5vm7MoGDH8JhlMrbBYVz-yzrOtjphOMQydlYI/edit#gid=0"
},
"options": {}
},
"type": "n8n-nodes-base.googleSheetsTool",
"typeVersion": 4.7,
"position": [
-4096,
784

```
    ],
    "id": "ebbb9b09-d209-4713-931d-0ef060985dab",
    "name": "Doctor Database",
    "credentials": {
      "googleSheetsOAuth2Api": {
        "id": "pkVboVILjDpfW77A",
        "name": "Google Sheets account"
      }
    }
  },
  {
    "parameters": {
      "descriptionType": "manual",
      "toolDescription": "Use this tool to check Google Calendar availability BEFORE confirming an appointment.\n\nCheck only the requested appointment date and time.\n\nIf an existing calendar event overlaps the requested appointment time, the slot is unavailable.\n\nNever assume a slot is available.\n\nNever confirm an appointment without checking this tool first.",
      "operation": "getAll",
      "calendar": {
        "__rl": true,
        "value": "ccc1f34632cab40cb14f30ef82d8b3c61767ac8d58ea61845057de293e69193f@group.calendar.google.com",
        "mode": "list",
        "cachedResultName": "Doctor Appointments"
      },
      "options": {}
    },
    "type": "n8n-nodes-base.googleCalendarTool",
```

```
    "typeVersion": 1.3,
    "position": [
      -3968,
      784
    ],
    "id": "965d3881-25d6-4862-b3d5-76f430ffef49",
    "name": "Get many events in Google Calendar",
    "credentials": {
      "googleCalendarOAuth2Api": {
        "id": "x2gCS7bXPQU8YlFt",
        "name": "Google Calendar account"
      }
    }
  },
  {
    "parameters": {
      "descriptionType": "manual",
      "toolDescription": "Create a Google Calendar appointment only after the patient has explicitly confirmed the final appointment summary.\n\nBefore using this tool, all of these must already be verified:\n\n- Doctor exists in Doctor Database\n- Doctor works on the requested date\n- Requested time is within working hours\n- Calendar availability has been checked\n- No conflicting appointment exists\n- Patient name is available\n- Patient mobile number is available\n- Patient problem is available\n- Patient explicitly confirmed\n\nNever create an appointment before explicit confirmation.\n\nNever create a duplicate appointment.\nNEVER CREATE AN APPOINTMENT BEFORE EXPLICIT PATIENT CONFIRMATION.",
      "calendar": {
        "__rl": true,
```

```
        "value":
"ccc1f34632cab40cb14f30ef82d8b3c61767ac8d58ea61845057de293e69193f@group.cale
ndar.google.com",
        "mode": "list",
        "cachedResultName": "Doctor Appointments"
      },
      "additionalFields": {
        "description": "Patient Name: [Patient Name] Mobile: [Mobile Number] Problem:
[Problem] Doctor: [Doctor Name] Hospital: [Hospital] Appointment Date: [Appointment
Date] Appointment Time: [Appointment Time]"
      }
    },
    "type": "n8n-nodes-base.googleCalendarTool",
    "typeVersion": 1.3,
    "position": [
      -3840,
      784
    ],
    "id": "bc9a941f-0c61-4725-9a83-f54607b56e02",
    "name": "Create an event in Google Calendar",
    "credentials": {
      "googleCalendarOAuth2Api": {
        "id": "x2gCS7bXPQU8YlFt",
        "name": "Google Calendar account"
      }
    }
  },
  {
    "parameters": {
      "options": {
```

```
      "timezone": "Asia/Dhaka"
    }
  },
  "type": "n8n-nodes-base.dateTimeTool",
  "typeVersion": 2,
  "position": [
    -3712,
    784
  ],
  "id": "0512ce76-1b66-4e37-964e-3144a8604966",
  "name": "Date & Time"
},
{
  "parameters": {
    "model": {
      "__rl": true,
      "mode": "list",
      "value": "gpt-5-mini"
    },
    "builtInTools": {},
    "options": {}
  },
  "type": "@n8n/n8n-nodes-langchain.lmChatOpenAi",
  "typeVersion": 1.3,
  "position": [
    -4352,
    784
  ],
  "id": "bbdf7f83-85d6-49f2-89d5-17e4474c6c25",
  "name": "OpenAI Chat Model",
```

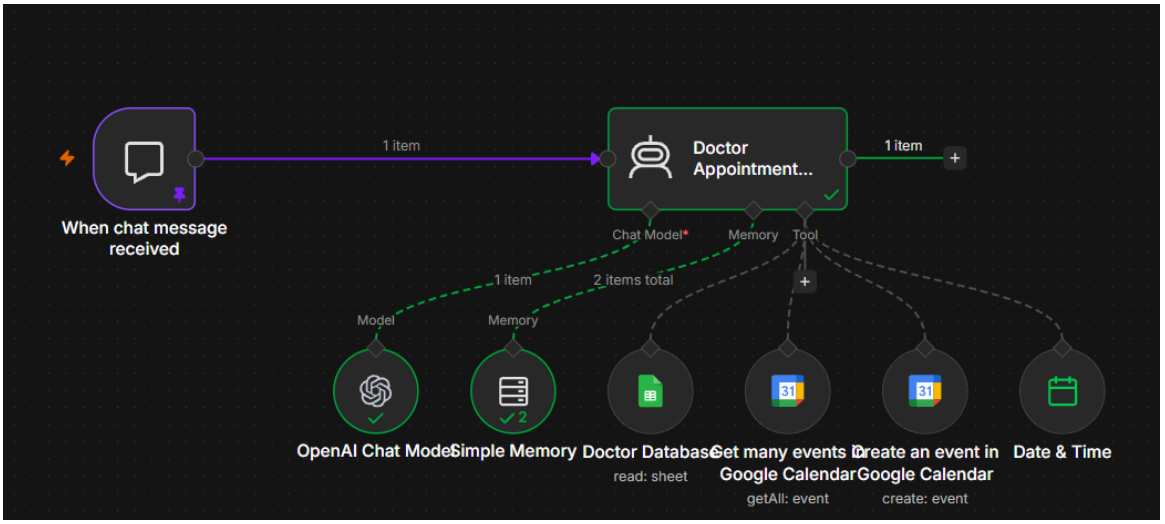
```
    "credentials": {
      "openAiApi": {
        "id": "PCp1my2Mhg6NHmtg",
        "name": "OpenAI account"
      }
    }
  }
],
"connections": {
  "When chat message received": {
    "main": [
      [
        {
          "node": "Doctor Appointment Agent",
          "type": "main",
          "index": 0
        }
      ]
    ]
  },
  "Simple Memory": {
    "ai_memory": [
      [
        {
          "node": "Doctor Appointment Agent",
          "type": "ai_memory",
          "index": 0
        }
      ]
    ]
```

```
},
"Doctor Database": {
  "ai_tool": [
    [
      {
        "node": "Doctor Appointment Agent",
        "type": "ai_tool",
        "index": 0
      }
    ]
  ]
},
"Get many events in Google Calendar": {
  "ai_tool": [
    [
      {
        "node": "Doctor Appointment Agent",
        "type": "ai_tool",
        "index": 0
      }
    ]
  ]
},
"Create an event in Google Calendar": {
  "ai_tool": [
    [
      {
        "node": "Doctor Appointment Agent",
        "type": "ai_tool",
        "index": 0
```

```
      }
     ]
    ]
   },
   "Date & Time": {
    "ai_tool": [
     [
      {
       "node": "Doctor Appointment Agent",
       "type": "ai_tool",
       "index": 0
      }
     ]
    ]
   },
   "OpenAI Chat Model": {
    "ai_languageModel": [
     [
      {
       "node": "Doctor Appointment Agent",
       "type": "ai_languageModel",
       "index": 0
      }
     ]
    ]
   }
  },
  "pinData": {
   "When chat message received": [
    {
```

```
      "action": "sendMessage",
      "sessionId": "a6c96233d56842178c73a731e9b805b4",
      "chatInput": "hay"
    }
  ]
 },
 "meta": {
  "templateCredsSetupCompleted": true,
  "instanceId":
"bd1158629e5c48504bcb630dd9b5c9afd108dd19c10dd5a483744a5758aa39ec"
 }
```


11.2 Final workflow



Chapter 12

System Evaluation, Impact, and Future Scope

12.1 Real-World Impact and Clinical Utility

The Smart Doctor Appointment Agent addresses critical operational inefficiencies in healthcare administration by automating patient triage and schedule management. By transitioning from traditional call-based scheduling to an intelligent, conversational interface, the system substantially lowers administrative overhead and eliminates long queue delays. Its native support for Romanized Bangla (Banglish) and colloquial phrasing democratizes access to digital health services, allowing patients across diverse technical backgrounds to communicate their symptoms naturally. Furthermore, the automated symptom-to-specialty classification reduces instances of misdirected consultations, ensuring patients are routed to the appropriate medical departments from their initial interaction.

- **24/7 Availability & Frictionless Access:** Operates continuously without human intervention, enabling instant booking sessions outside standard clinic hours.
- **Reduction of Administrative Overhead:** Mitigates reception phone traffic by 50% to 70%, allowing medical staff to focus on critical on-site patient care.
- **Elimination of Scheduling Conflicts:** Enforces zero double-booking tolerances through real-time calendar dynamic synchronization in the Asia/Dhaka timezone.

12.2 Architectural Advantages Over Legacy Systems

Unlike static booking forms or rigid decision-tree chatbots, this architecture leverages an LLM-driven reasoning loop coupled with direct API tools. By delegating data operations

to isolated function calls (Google Sheets for doctor lookups and Google Calendar for slot verification), the agent maintains strict data integrity and predictable behavior.

Evaluation Metric	Traditional Booking Systems	Smart AI Agent Architecture
Interface Style	Static web forms / Call queues	Conversational NLP interface
Specialty Triage	Manual user selection	Automated symptom triage
Slot Synchronization	High risk of overlap / Delayed	Real-time calendar verification
Language Support	Fixed language inputs	Multilingual & Banglish decoding

12.3 Future Enhancements and System Roadmap

While the current framework delivers robust end-to-end booking functionality, several architectural expansions can transform the agent into a comprehensive digital healthcare ecosystem:

- **Omnichannel Messaging Integration:** Deploying native webhooks for WhatsApp and Telegram to reach patients on their primary messaging platforms without requiring dedicated web applications.
- **Payment Gateway Synchronization:** Incorporating local financial APIs (such as bKash, Nagad, and SSLCommerz) directly into the booking loop to collect consultation fees prior to calendar event creation.
- **Automated Telephony & Voice Triage:** Integrating speech-to-text models (e.g., OpenAI Whisper) and voice synthesis APIs to enable natural telephone-based appointment booking for non-literate users.
- **Enterprise Health Record (EHR) Integration:** Upgrading the backend storage from Google Sheets to FHIR-compliant Electronic Health Record databases for enterprise hospital deployments.

REFERENCES

[1] World Health Organization, Ethics and Governance of Artificial Intelligence for Health: WHO Guidance. Geneva: World Health Organization, 2021.

[2] World Health Organization, Digital Health and Artificial Intelligence in Healthcare Systems. Geneva: World Health Organization, 2021.

[3] n8n Documentation, Workflow Automation and AI Agent Integration Framework. n8n GmbH. Available: https://docs.n8n.io/

[4] OpenAI Developer Platform, OpenAI API Documentation: Language Models, Function Calling, and Structured Outputs. OpenAI, 2024.

[5] Google Workspace Developers, Google Calendar API Overview and Scheduling Services. Google for Developers. Available: https://developers.google.com/calendar

[6] Google Workspace Developers, Google Sheets API Reference: Database and Data Manipulation. Google for Developers. Available: https://developers.google.com/sheets/api

[7] Google Developers, Using OAuth 2.0 for Web Server Applications and API Access. Google for Developers.

[8] Telegram Bot API Documentation, Telegram Bot API Infrastructure and Messaging Interfaces. Telegram. Available: https://core.telegram.org/bots/api

[9] National Institute of Standards and Technology (NIST), U.S. Department of Commerce, Artificial Intelligence Risk Management Framework (AI RMF 1.0). NIST, 2023.

[10] International Organization for Standardization, Health Informatics — Electronic Health Record Architecture (ISO/HL7 10781:2015). ISO, 2015.