Contents

[**MedBotAssistRepo — AI-Powered Medical Assistant for Healthcare Secretaries** 2](#_Toc204777551)

[**Project Overview** 2](#_Toc204777552)

[**Core Features - Integration with Medical Systems** 2](#_Toc204777553)

[*Integration with Medical Systems* 2](#_Toc204777554)

[*Customizable Availability* 2](#_Toc204777555)

[*Clinical Summary Generation* 2](#_Toc204777556)

[*Security and Access Control* 2](#_Toc204777557)

[**System Architecture** 3](#_Toc204777558)

[**System Roles** 3](#_Toc204777559)

[**Project Status** 3](#_Toc204777560)

[**Delivered functionalities** 3](#_Toc204777561)

[*Login Interface – MedBotAssist* 3](#_Toc204777562)

[Doctor Registration Interface – MedBotAssist 4](#_Toc204777563)

[**Doctor Weekly Dashboard – MedBotAssist** 5](#_Toc204777564)

[Patient Management Dashboard – MedBotAssist 6](#_Toc204777565)

[Key Features: 11](#_Toc204777566)

[**Main Components:** 20](#_Toc204777567)

# **MedBotAssistRepo — AI-Powered Medical Assistant for Healthcare Secretaries**

**MedBotAssistRepo** is the official repository for the development of an intelligent conversational assistant powered by Artificial Intelligence, specifically designed to support medical secretaries in scheduling appointments and generating clinical summaries. This project is part of a broader vision of cognitive automation in the healthcare sector, where AI becomes a strategic ally to improve efficiency, security, and the administrative experience in medical environments.

## **Project Overview**

**MedBotAssist** leverages the power of **Large Language Models (LLMs)**, such as those provided by OpenAI, to automate and optimize routine workflows in clinical settings. Through a natural language interface, the assistant allows users (mainly administrative staff) to interact with medical systems without needing advanced technical knowledge.

This system enables:

* **Smart scheduling** of medical appointments based on specialty, availability, and priority
* **Automated generation** of clinical summaries from medical notes or conversations
* **Secure handling of sensitive data**, complying with privacy and audit standards
* **Context-aware conversations**, adapted to clinical settings and patient/professional history

## **Core Features - Integration with Medical Systems**

### *Integration with Medical Systems*

* Native connectivity with Electronic Health Records (**EHR**) and scheduling engines
* Interoperability with clinical databases to access, query, and update information in real time

### *Customizable Availability*

* Individual configuration per doctor or healthcare professional

### *Clinical Summary Generation*

* Automatic analysis of clinical notes to create human-readable summaries
* Capabilities for spellchecking, semantic cleanup, and extractive summarization

### *Security and Access Control*

* Authentication using JWT tokens and role-based access control (secretaries, doctors, auditors)

## **System Architecture**

The assistant is built with a modular and scalable architecture:

* **Frontend:** ReactJS (Web App) with possible mobile adaptations
* **Backend:** .NET 8 WebAPI + FastAPI for AI modules
* **AI Pipeline:** FastAPI + LangChain + Azure OpenAI (RAG or direct model consumption)
* **Database:** Azure SQL / CosmosDB
* **CI/CD:** GitHub Actions for continuous deployment to Azure Web Apps

Additional pipelines include:

* Clinical document vectorization (when using RAG)
* Medical tools orchestration (LangChain Tools)

## **System Roles**

* **Secretaries:** Access scheduling features, availability queries, and summaries
* **Doctors:** View and edit summaries, manage their own availability

## **Project Status**

* **Active:** development in progress
* **Unit testing** with AI tool mocking and code coverage
* **Functional deployment** pipelines for frontend and backend
* **MVP** planned for 1 August 2025

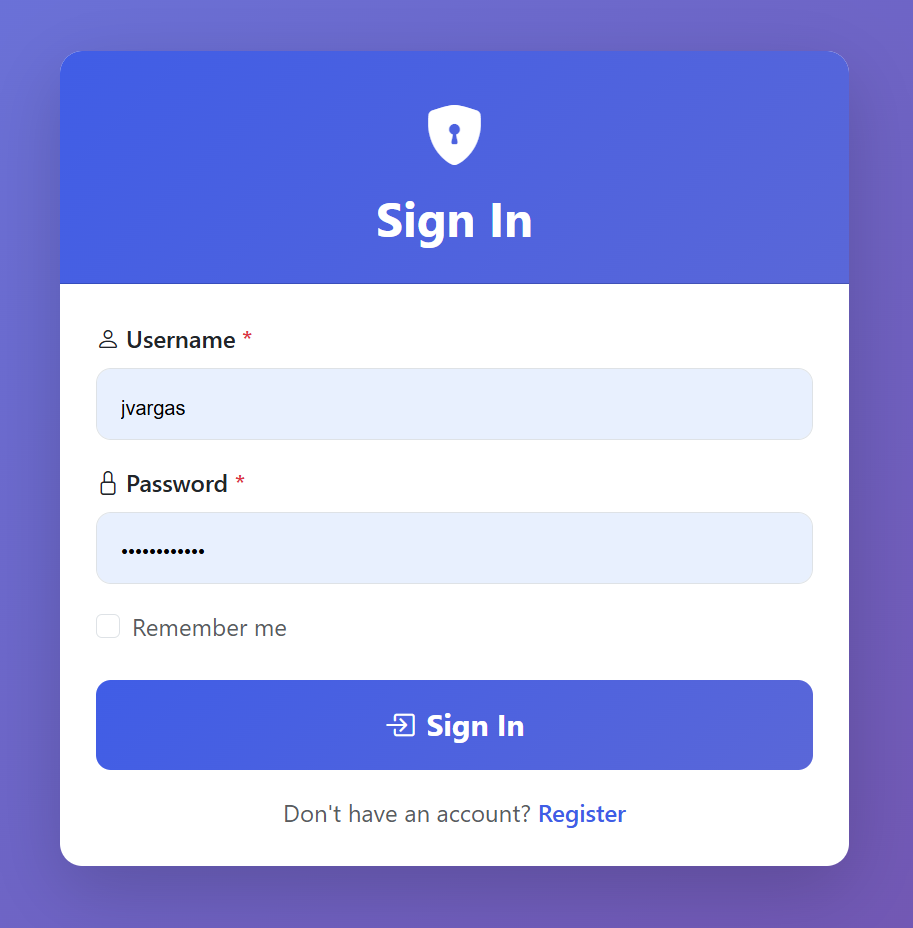
## **Delivered functionalities**

### *Login Interface – MedBotAssist*

This is the **secure login screen** for the MedBotAssist platform, designed to authenticate registered users such as medical secretaries, doctors, and administrators. The interface features a clean and modern design with a focus on usability and accessibility.

**Key Elements:**

* **Username Field:** Allows users to input their unique username (e.g., jvargas)
* **Password Field:** Secure input with obscured characters to protect sensitive credentials
* **"Remember Me"** Option: Enables persistent sessions on trusted devices
* **Sign In Button:** Triggers authentication against the system's backend (typically a .NET or Python API)
* **Registration Link:** Redirects users without an account to the registration page

****

### Doctor Registration Interface – MedBotAssist

This is the Doctor Registration form for the MedBotAssist platform, designed to onboard healthcare professionals securely and efficiently. The interface combines personal, professional, and access credential fields in a clear, structured layout, ensuring a smooth user experience for medical staff. Sections and Key Fields:

**Personal Information**

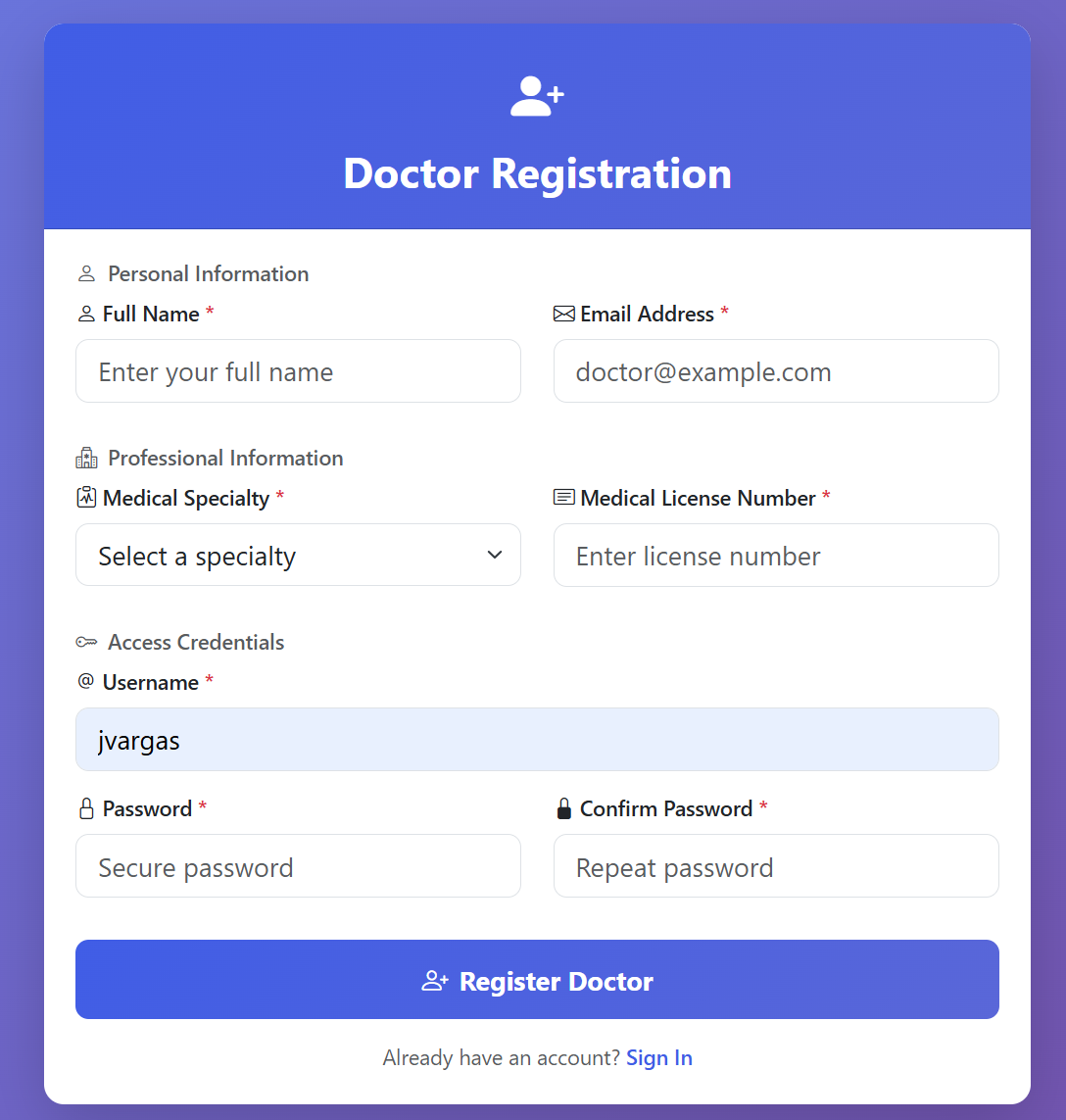
* **Full Name:** Text field for entering the doctor's full legal name
* **Email Address:** Required field for contact and login notifications (must follow a valid email format)

**Professional Information**

* **Medical Specialty:** Dropdown selector to choose the area of medical expertise (e.g., Cardiology, Pediatrics)
* **Medical License Number:** Mandatory field for inputting the official license or registration ID assigned to the physician

**Access Credentials**

* **Username:** Unique identifier that will be used during login (e.g., jvargas)
* **Password & Confirm Password:** Secure fields requiring strong password creation and confirmation, with masking enabled to prevent exposure

****

### **Doctor Weekly Dashboard – MedBotAssist**

This is the **Weekly Appointment Dashboard** for doctors within the MedBotAssist platform. It provides a clear and structured overview of the physician’s upcoming schedule and essential profile information, helping healthcare professionals manage their time and activities effectively.

#### Weekly Calendar View

* Displays the current week (e.g., July 27 – August 2, 2025) in a column-based layout.
* Each day shows appointment availability or the status *"No appointments"* if the doctor has no consultations scheduled.
* Navigation arrows allow switching between weeks to plan ahead or review past activity.
* The current day (Wednesday 30/7) is visually highlighted to guide daily focus.

#### Doctor Information Panel (Right Sidebar)

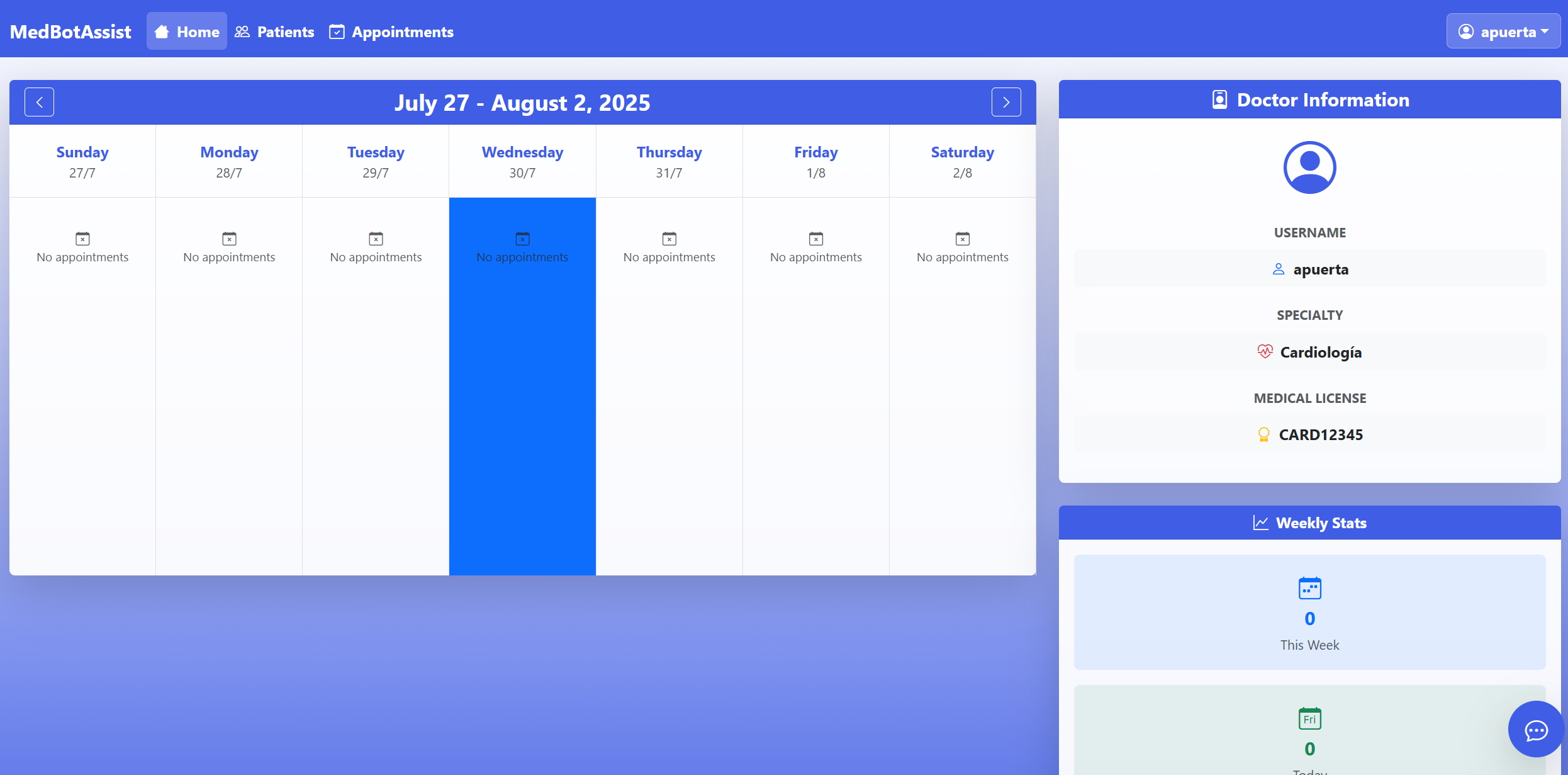
* Shows the authenticated doctor’s **username** (apuerta), **specialty** (Cardiología), and **medical license number** (CARD12345).
* This profile snapshot helps identify who is logged in and is especially useful in shared terminals or multi-user interfaces.

#### Weekly Statistics

* Provides quick performance metrics for the current week, including:
  + Total number of appointments scheduled for the week (0)
  + Daily appointment breakdown (e.g., by weekday like Friday)

#### Top Navigation Bar

* Contains links to primary modules:
  + **Home**, **Patients**, and **Appointments**
* Shows the active session user and account menu (apuerta) for quick profile or logout access.



### Patient Management Dashboard – MedBotAssist

This is the **Patient Management** interface in the MedBotAssist platform. It serves as a centralized control panel for viewing, searching, registering, and managing patient records within a healthcare facility. The interface is optimized for use by administrative staff and medical professionals, offering intuitive access to essential patient data.

#### Search and Filter Tools

* **Search Bar:** Allows keyword-based filtering using patient name, ID, or email.
* **Status Filter Dropdown:** Designed to filter patients by status (active, inactive, etc.) though all statuses are selected by default in this view.
* **Refresh Button:** Reloads the patient list to reflect the most up-to-date data.
* **Add Patient Button:** Opens a registration form to add new patients to the system.

#### Patient Records Table

A dynamic table listing registered patients with the following key fields:

|  |  |
| --- | --- |
| Column | Description |
| Patient ID | Sequential number assigned to the patient in the list |
| Full Name | Complete legal name of the patient |
| Identification | Government-issued ID or internal code (e.g., ID001, 800123456) |
| Date of Birth | Birthdate in MM/DD/YYYY format |
| Age | Automatically calculated from date of birth |
| Phone | Contact number of the patient |
| Email | Patient’s email address for communication |
| Actions | Buttons for: |
| View profile | Info of patient |
| Edit information | Edit info of patien |

#### **Patient Detail View – MedBotAssist**

A screenshot of a medical form

AI-generated content may be incorrect.

This screen corresponds to the **"View Patient"** interface within the MedBotAssist platform. It allows authorized medical staff (e.g., doctors, secretaries) to view detailed demographic and contact information about a specific patient. The layout is clean, organized in a two-column structure for fast readability, and focused on essential fields needed for clinical follow-up and administrative tasks.

|  |  |
| --- | --- |
| Field | Description |
| Patient ID | Internal unique identifier used by the system (e.g., 1) |
| Full Name | Complete name of the patient (e.g., *Carlos Sánchez Goez*) |
| Identification Number | Official ID or institutional code (e.g., ID001111) |
| Date of Birth | Patient’s birthdate in MM/DD/YYYY format (10/05/1980) |
| Phone Number | Contact number used for communication (555-1234-4789) |
| Email Address | Email for sending appointment confirmations or clinical summaries (carlos@example.com) |

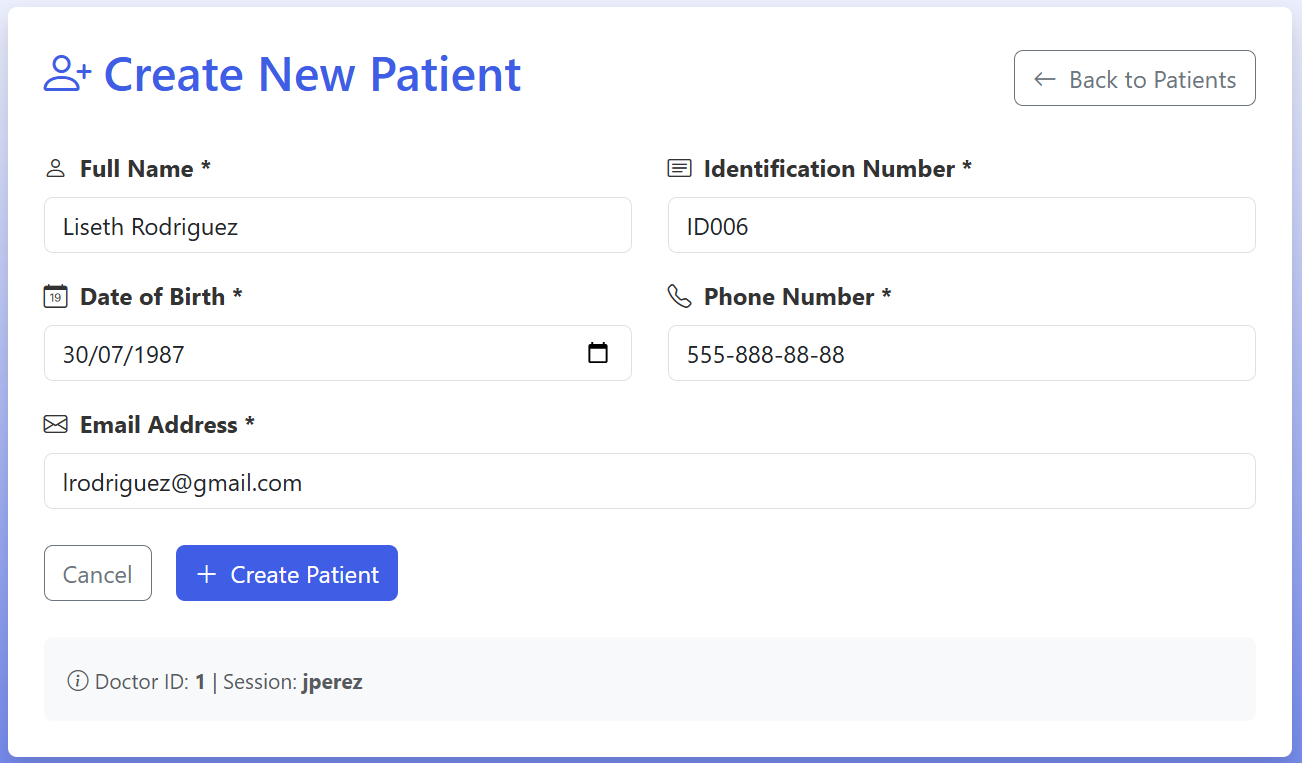
***Edit Patient Interface – MedBotAssist***

A screenshot of a medical form

AI-generated content may be incorrect.

This screen corresponds to the **“Edit Patient”** functionality within MedBotAssist, allowing authorized users (such as doctors or administrative personnel) to securely update patient records. It ensures data integrity by validating required fields and confirming updates with user feedback. A structured form composed of editable fields that reflect the current patient data.

#### **Create New Patient Interface – MedBotAssist**



This screen represents the **"Create New Patient"** form within the MedBotAssist platform. It enables authorized users (such as doctors or administrative personnel) to register a new patient into the system with all the required demographic and contact information. The layout is clean, responsive, and validation-aware, ensuring accurate data entry.

#### **Add Medical Note - MedBotAssist**

This interface is part of the **clinical workflow** within MedBotAssist, designed to support medical professionals in reviewing patient history and recording new clinical observations during a consultation.

Displays previously recorded clinical data, **automatically managed by the system**.

Example content:

* **Diagnosis:** Seasonal allergy
* **Treatment:** Antihistamines
* **Recommendations:** Avoid allergens
* **Next Steps:** Follow-up in 2 weeks

A structured entry from *July 25, 2025*, is shown, written in clean, professional English by the system.

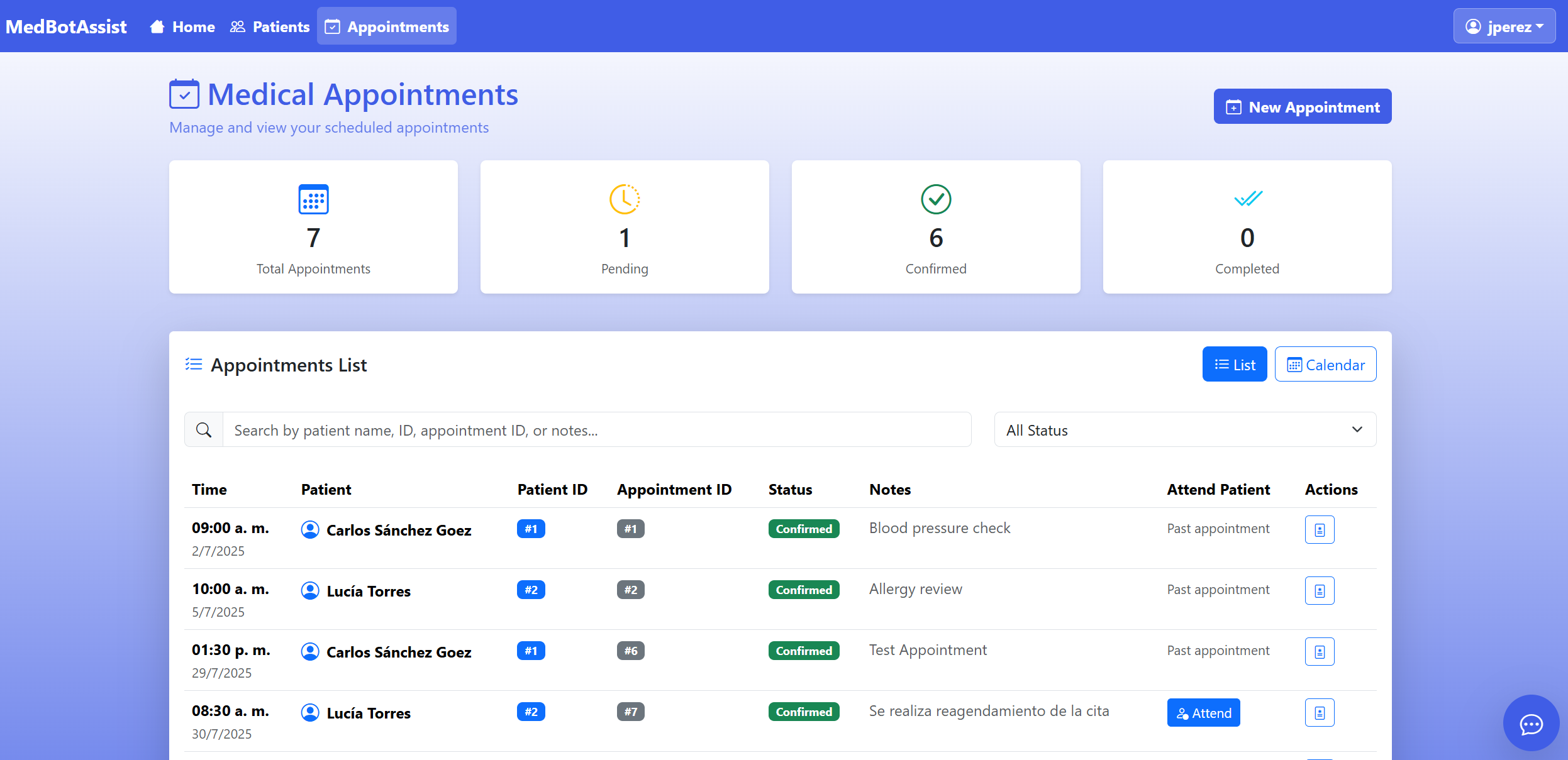
A screenshot of a medical history

AI-generated content may be incorrect.  
A screenshot of a computer

AI-generated content may be incorrect.

#### **Medical Appointments Dashboard – MedBotAssist**

This interface provides a centralized view for managing and monitoring patient appointments within the **MedBotAssist** platform. Designed for medical staff such as secretaries and physicians, it combines a real-time summary of appointment metrics with a searchable, filterable, and actionable appointment list.



**Top Overview Panel**

Four key statistics provide an immediate snapshot of the current appointment status:

* **Total Appointments**: 7 – Total number of scheduled consultations
* **Pending**: 1 – Appointments awaiting confirmation or action
* **Confirmed**: 6 – Appointments validated and confirmed with patients
* **Completed**: 0 – Appointments marked as fully attended and closed

A "New Appointment" button at the top right allows the user to create a new consultation entry through a guided form.

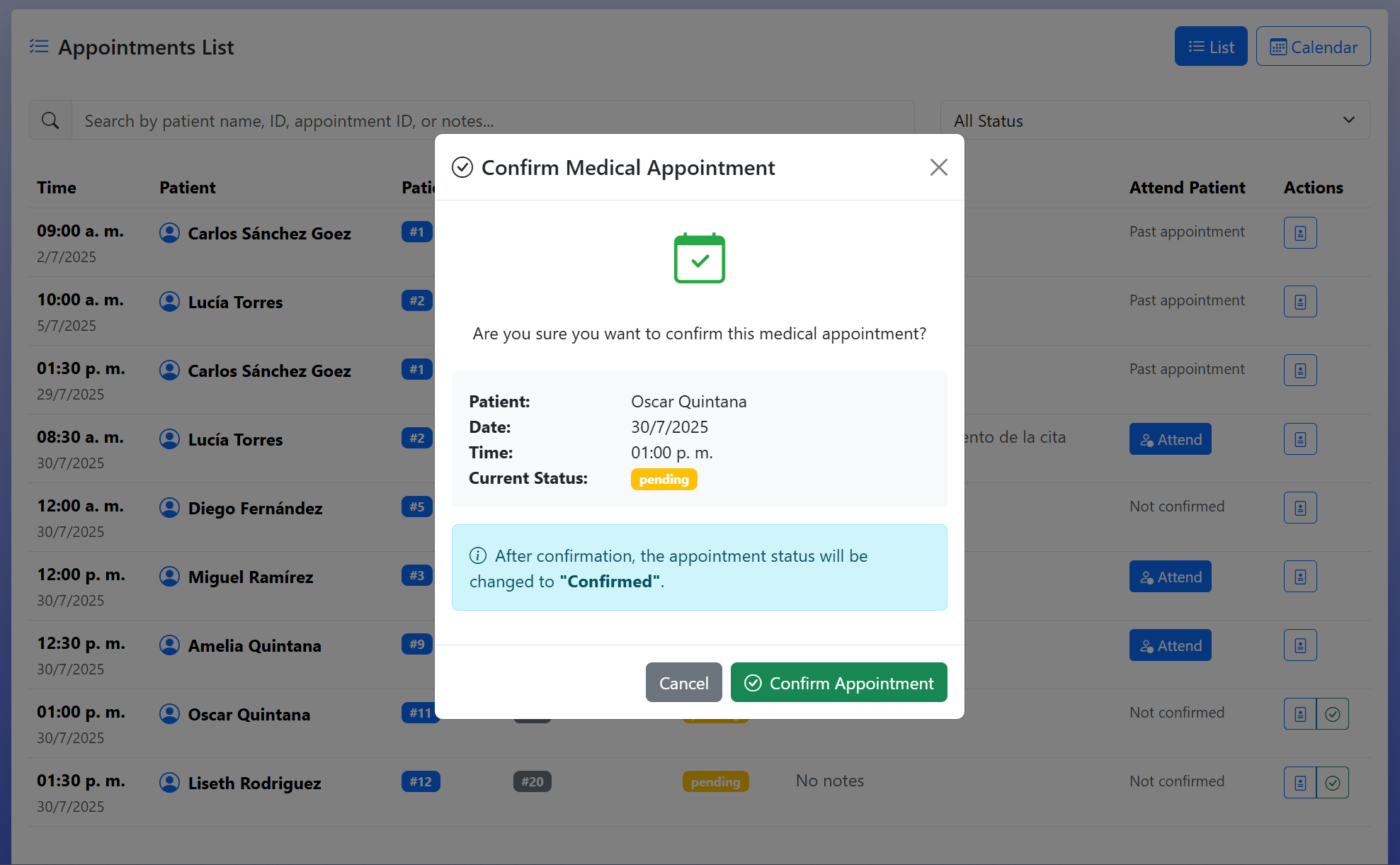
#### **Create New Appointment Modal – MedBotAssist**

This modal interface is designed to streamline the process of scheduling a new medical appointment within the **MedBotAssist** platform. It provides a simple and guided form to register appointment details in a structured and consistent way.

**Appointments can only be attended if they are confirmed and scheduled for today or a future date.**

#### **Confirm Medical Appointment**

The **Confirm Medical Appointment** feature allows healthcare staff or medical professionals to validate and finalize the scheduling of a patient consultation. This action changes the status of an appointment from **Pending** (or any provisional state) to **Confirmed**, signaling that the date and time have been officially reserved and acknowledged by the clinic and/or patient.



MedBot Assistant – Virtual Medical Chat Interface

This is the interactive chat widget for **MedBot Assistant**, a virtual medical assistant designed to engage users through natural language conversation. It allows patients or medical staff to ask questions, receive assistance, or initiate specific actions related to healthcare workflows (e.g., scheduling, clinical summaries, patient data review).

### Key Features:

**Welcome Message:**  
Displays a friendly greeting:  
*“Hello! I'm your virtual medical assistant. Write a message to get started.”*

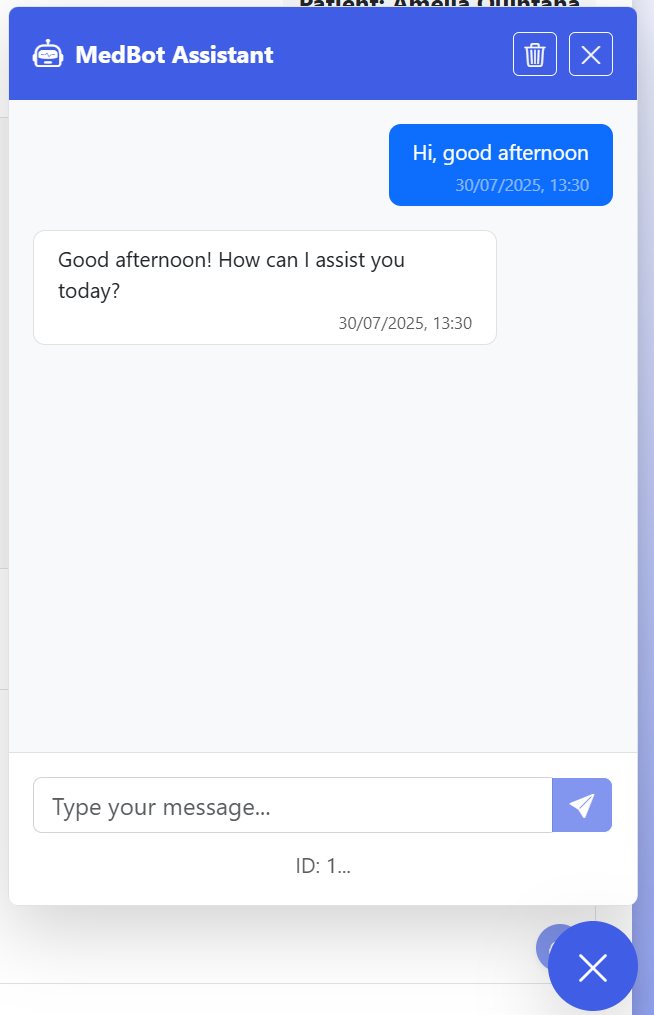
**Chat Input Field:**  
Users can type their message in the input box labeled *“Type your message...”* and send it using the **send icon** (paper plane).

**Session Info (ID):**  
Displays a conversation/session identifier at the bottom (e.g., ID: 1...), useful for tracking or maintaining context between interactions.

**Toolbar Actions (Top Right):**

* **Trash icon:** Likely used to clear the current conversation history
* **Close icon (X):** Minimizes or closes the chat interface

**UI Styling:**  
The chat box is compact, floating, and responsive—ideal for embedding into medical portals, dashboards, or patient-facing systems.



The medical agent is a virtual assistant equipped with a suite of tools that allow it to interact with clinical data through structured APIs. Based on the capabilities shown, this agent can process natural language questions and respond with relevant patient information, summaries, statistics, and actions—depending on the permissions granted and the tool invoked.

**Statistics**

* "How many patients are registered in total?"

A screenshot of a chat

AI-generated content may be incorrect.

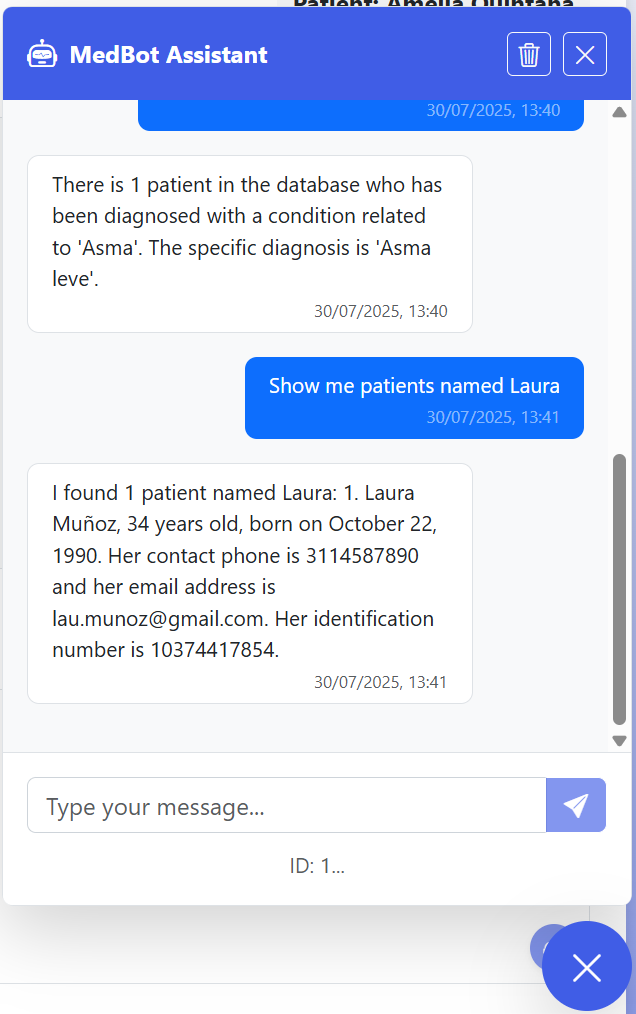
* "How many patients were diagnosed with Asma?"

A screenshot of a chat

AI-generated content may be incorrect.

**Patient Search**

* "Show me patients named Laura."

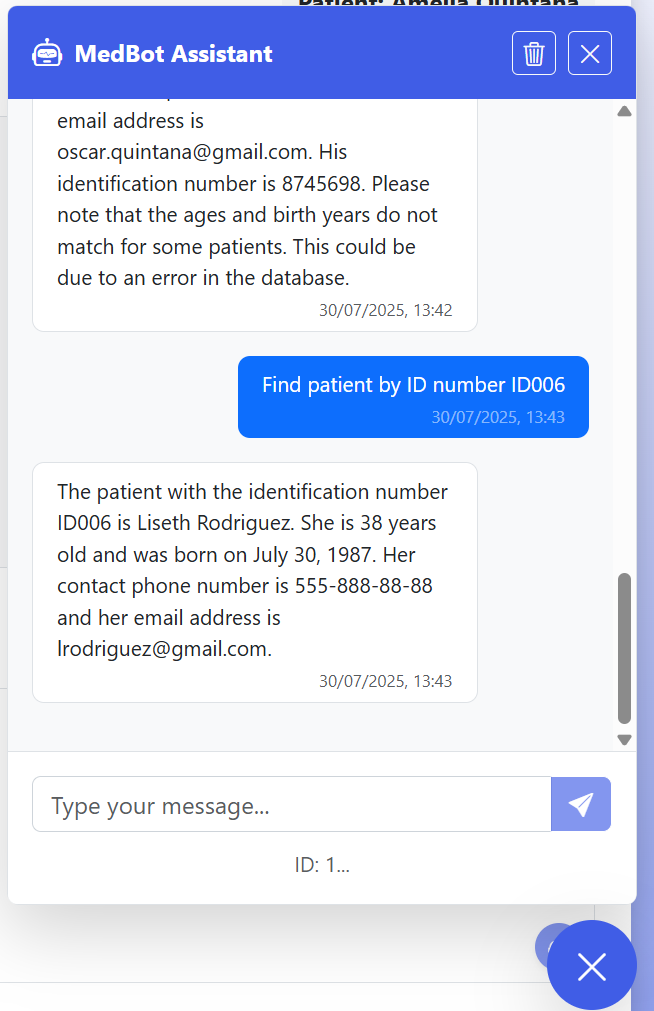


* "Search for all patients born in 1985."

A screenshot of a chat

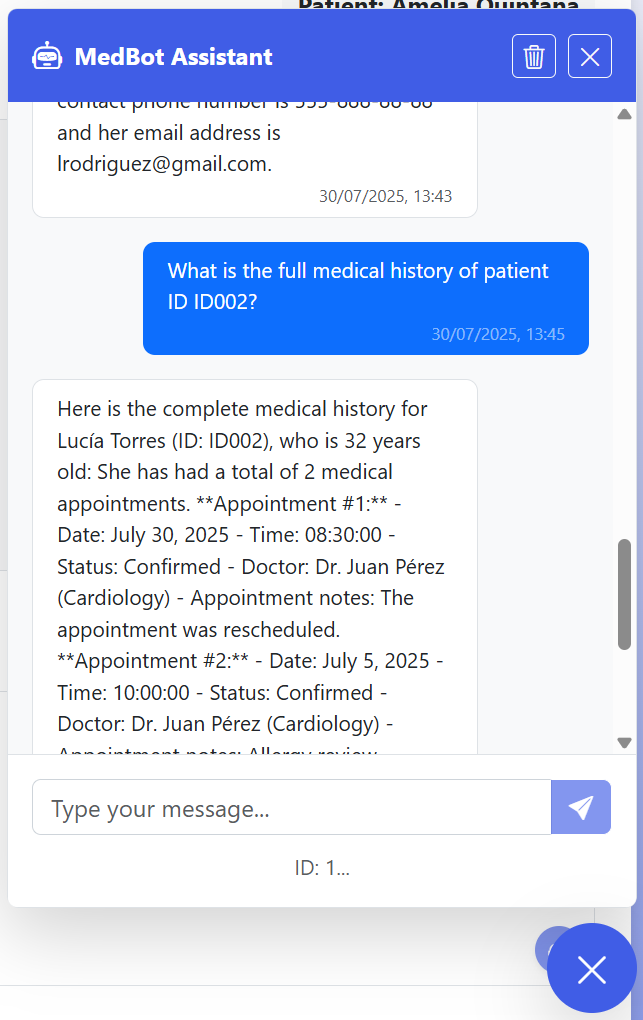
AI-generated content may be incorrect.

* "Find patient by ID number 123456."



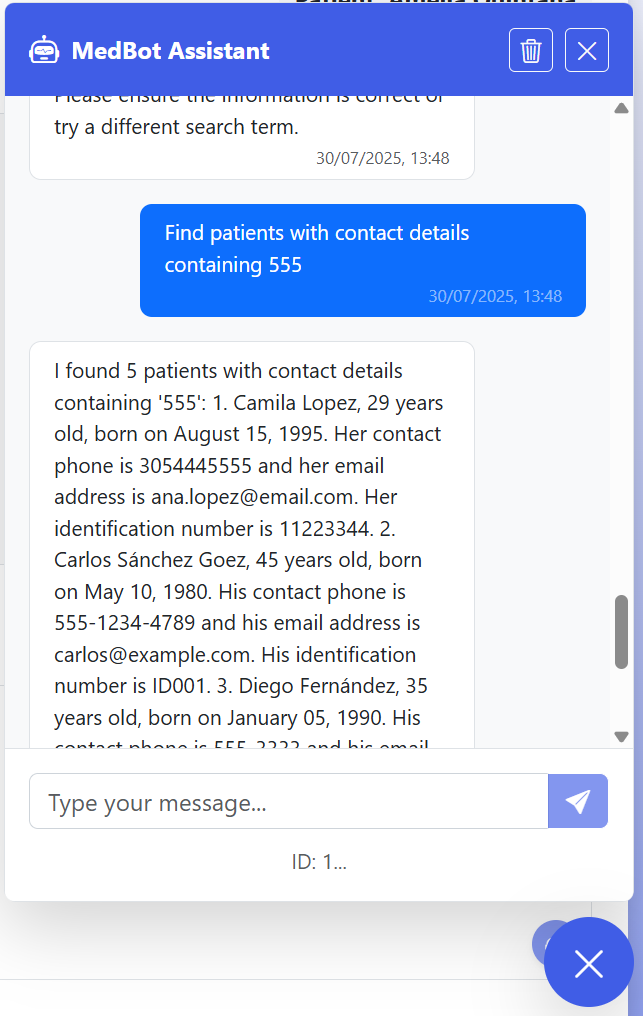
**Medical History**

* "What is the full medical history of patient ID ID002?"



**Contact-Based Search**

* "Find patients with contact details containing 555"



* "Who can I call from the patients diagnosed with flu?"

A screenshot of a chat

AI-generated content may be incorrect.

**Patient Management (if authorized)**

* "Create a new patient with the name 'Juana Pelaez', identification 'ID007', date of birth '1999-12-15', age 24, phone number '555-8888-7777', and email 'juana.pelaez@email.com'."

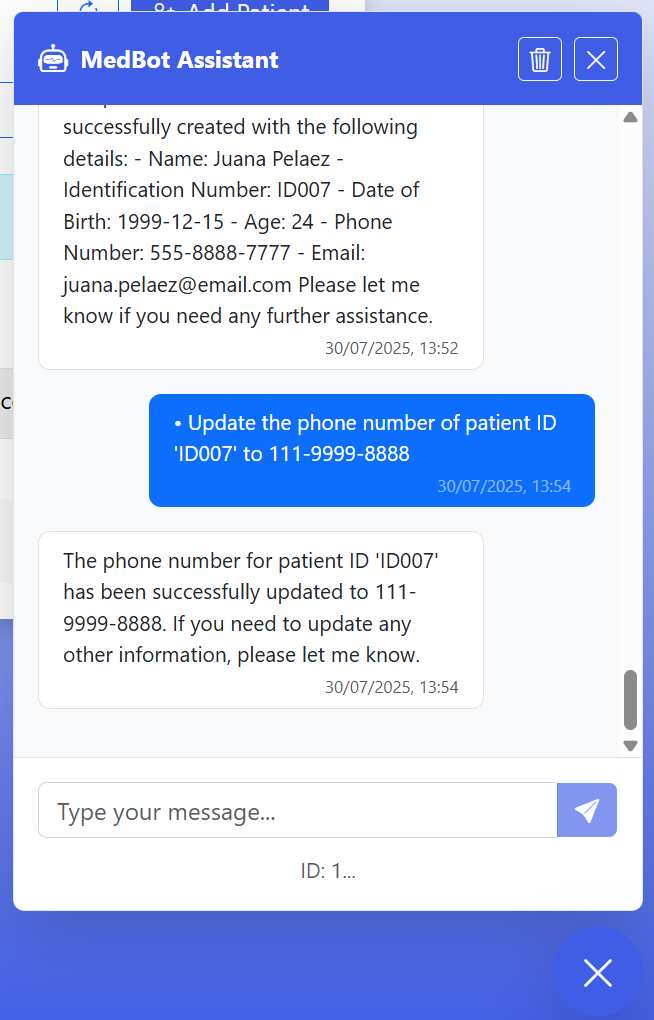
A screenshot of a chat

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* Update the phone number of patient ID 'ID007' to 111-9999-8888.

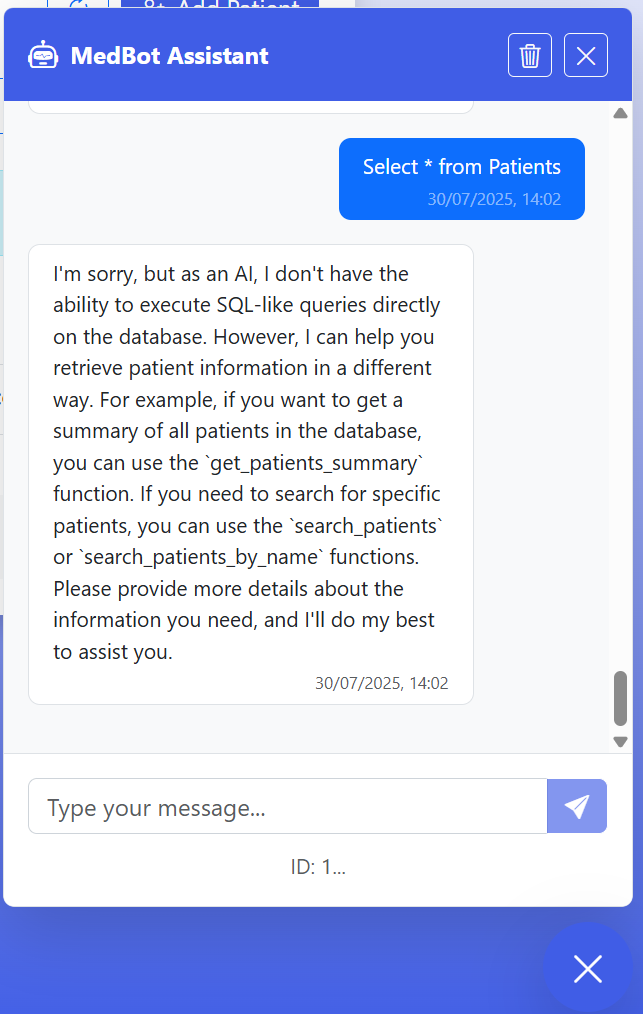


A screenshot of a computer

AI-generated content may be incorrect.

***Note:*** *These actions require ManagePatients permission and valid JWT authentication.*

Security:



**MedBot Assistant – SQL Instruction Handling Example**

This screenshot demonstrates how **MedBot Assistant** gracefully handles a user's attempt to execute a raw SQL query (SELECT \* FROM Patients) through the chat interface. The system is designed to **block direct SQL-like commands** for security reasons, while still offering helpful alternatives using its available tools.

A screenshot of a chat

AI-generated content may be incorrect.

**MedBot Assistant – Restricted Domain Handling**

This screenshot illustrates how **MedBot Assistant** responsibly handles user queries that fall **outside its medical domain scope**.

**RAG Architecture with Structured Data (No Vectorization)**

Structured query → Real-time retrieval → Contextual generation with LLM

### **Main Components:**

#### User / Frontend

Conversational interface or form (chatbot, portal, REST API)

* Sending a query in natural language

#### Language Processing Pipeline (NLP Preprocessing)

Optionally transforms user input (parsing, intent classification, etc.)

* Determines the type of query.  
  Example: *"What diagnoses does patient Juan have?"*

#### Structured Retrieval Module

Instead of searching vectorized documents, **real-time SQL (or NoSQL) queries** are generated

* Use a template or generation module to translate natural language → structured query
* Executes the query against the database
* Retrieves structured and readable information (tables, records)

#### Generator Model (LLM)

Receives the structured result as context

* Generates a final natural language response, clear and contextualized for the user  
  *(Example: "Patient Juan has been diagnosed with hypertension on two occasions...")*

#### Security and Access Control Layer

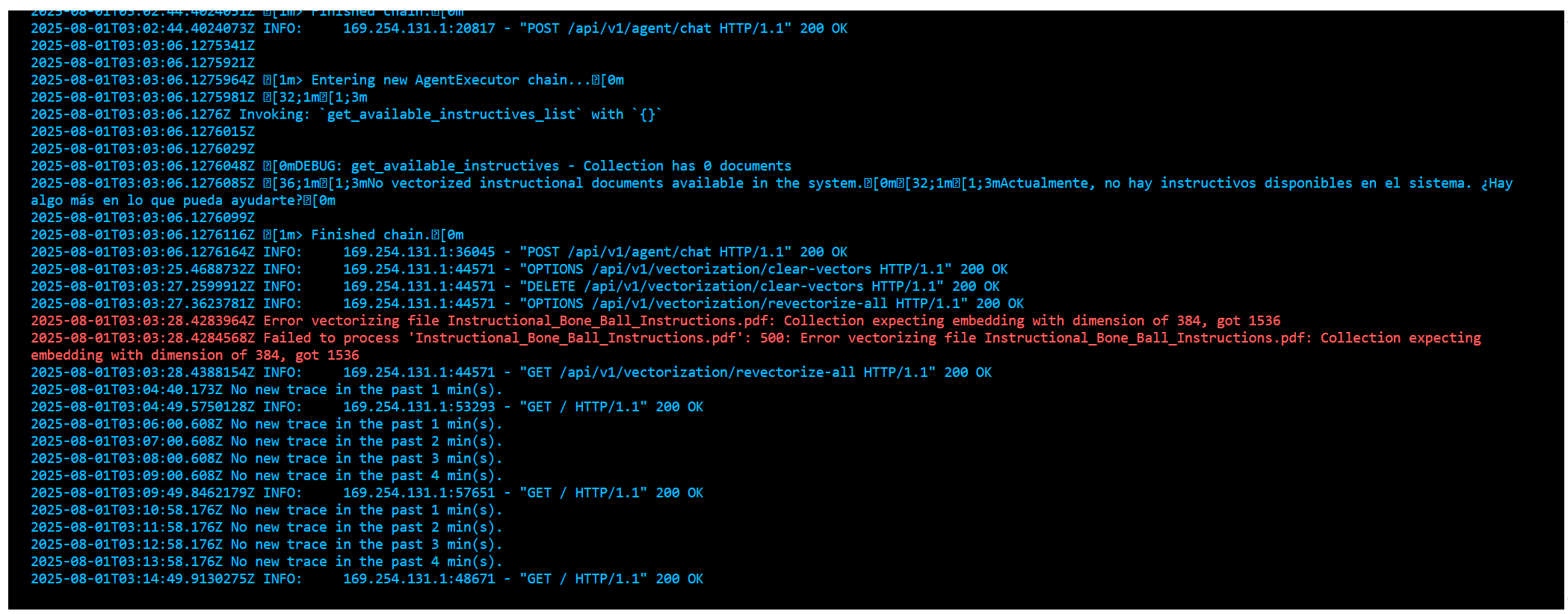
* JWT authentication, role-based permissions (e.g., secretary, doctor, auditor)
* Query logging and traceability

#### Backend or Orchestration Middleware

* Coordinates database and LLM calls
* Applies business rules, flow control, and input/output validation

### Conclusions and Lessons learned

### Chroma DataBase



Throughout multiple deployments, it became evident that **using Chroma as a vector store within Azure Web App environments introduced several operational limitations**. The main issues observed include:

**Slowness during initial load**, particularly due to the ONNX model downloads triggered when the container starts.

**Difficulties regenerating vectorized records** on demand, even when calling specific endpoints like revectorize-all, likely due to synchronization or temporary file storage issues.

**Predefined embeddings such as all-MiniLM-L6-v2 are loaded automatically during initialization**, which can lead to unnecessary startup delays and complicate resource management in scalable or low-consumption environments.

These observations suggest that while Chroma is convenient for prototypes and local development, **its behavior in cloud environments like Azure App Service may require fine-tuning or even the consideration of more robust and persistent alternatives**, such as Pinecone, Qdrant, or Azure Cognitive Search, especially for production-grade workloads that demand high availability and consistent performance.

**Why Chroma and not other vector databases?**  
Chroma was initially chosen for this implementation based on prior experiences in virtual courses and frequent community recommendations, particularly due to its ease of use and straightforward deployment in virtualized environments.

Although Chroma was used as the vector store, it’s important to note that more robust and production-ready alternatives—such as **Pinecone**, **Qdrant**, or **Supabase Vector**—could have offered superior performance, scalability, and reliability.

However, these alternatives were not feasible due to two main constraints:

* **Cost limitations**, as most of these platforms require paid plans for production-grade usage.
* **Data privacy concerns**, since these services operate outside the enterprise cloud environment, raising potential compliance and governance risks when handling sensitive medical or institutional data.

In summary, while external vector stores would have been the optimal choice in a production context, Chroma was retained to align with the constraints of budget, security, and local deployment requirements.

**High consumer prices**A screenshot of a computer

AI-generated content may be incorrect.

The cost analysis of the OpenAI API usage reveals **significant consumption spikes**, particularly on days when intensive interactions with the agent were performed. With over **607,000 tokens processed** and **335 requests made** in just a few days, it is evident that even moderate development and testing activities can lead to **rapid budget depletion**, especially when working with advanced models such as GPT-4.

This highlights the need for:

* **Careful monitoring of token usage**.
* **Token optimization strategies** (e.g., truncating input history, compressing prompts).
* **Usage caps or fine-tuned models** for sustainable development.
* Considering **on-premise LLMs or cost-efficient vector stores** for non-critical use cases.

In future enterprise-grade deployments, keeping these costs under control is essential to avoid unexpected overruns, especially when working within limited budgets or student environments.

### **Addressing Cost and Latency with Local Models (Planned in Future Learning Path)**

The high operational cost and latency observed during this Proof of Concept (POC), especially when relying on cloud-hosted LLMs (like OpenAI), could be **significantly mitigated by deploying local language models** using Docker containers.

Running models locally (e.g., via Ollama, LM Studio, or GPT4All) offers key advantages:

* **Cost efficiency**: No pay-per-token or API-based billing.
* **Improved privacy**: Data never leaves the organization’s virtual network.
* **Faster response times**: No outbound latency or rate limiting from cloud APIs.
* **Full control**: Model versioning, fine-tuning, and security managed internally.

However, for this initial POC, I made a conscious decision to **prioritize rapid development and functional demonstration** over infrastructure optimization.

My primary goal was to **showcase the possibilities of intelligent automation in healthcare workflows**, including chat-based interactions, document retrieval, vector search, and structured querying.

### **Architectural Learnings: Structured Data vs. Vectorized Storage**

Throughout the development of this POC, one of the most valuable insights was the **deep understanding of architectures driven by structured data**, particularly in the context of Retrieval-Augmented Generation (RAG) pipelines that operate without traditional vector stores.

This experience highlighted several key takeaways:

* **Structured databases** (e.g., SQL-based medical records) offer consistency, integrity, and auditability—making them ideal for real-time querying and compliance-sensitive environments.
* Querying structured data dynamically using language models enables **context-aware responses** without needing to maintain large vector indexes.

### **Challenges included:**

* Manual or scheduled re-vectorization to reflect updates.
* Risks of **serving stale embeddings** if pipelines fail silently.
* Operational overhead in containerized environments (e.g., memory usage, embedding caching, model downloads, concurrency limits).

While vector databases excel in use cases with **stable, long-lived content**, this POC demonstrated that **structured-query RAG pipelines** offer better control and agility for domains with **rapid data evolution**, such as healthcare.

#### **RAG architecture with structured data**

Esta es la arquitectura alcanzada en este challenge