Course Final Project – Medical AI Assistant

Final Written Report

Phuong Pham

CSCI 3124 – Intro to Cloud Computing

Project Introduction

The Medical AI Assistant is a lightweight AI-powered chatbot application designed to assist clinicians and clinic staff by answering common administrative questions related to a medical centre – such as FAQ about appointments, billing, hours, services, etc. using Retrieval-Augmented Generation (RAG). It does not provide medical advice, treatment advice or diagnosis for safety and responsibility.

The project includes both a backend server and a simple frontend chat interface:

- A Node.js/Express backend that handles chat requests, performs semantic search over stored FAQ documents, and returns AI-generated responses based only on those docs.
- A frontend UI (HTML/JS/CSS) that lets users interact with the chatbot via a browser.

The system uses Google's Gemini API to generate answers informed by the FAQ context, and PostgreSQL with pgvector to store embeddings for semantic search.

What is it supposed to do?

The purpose of this application is to:

1. Reduce clinician administrative burden by answering routine questions that patients or staff would otherwise have to respond to manually.
2. Provide consistent, context-aware responses based only on ingested FAQ documents; it's not intended to generate free-form medical information outside that context.
3. Serve as a demo/early prototype of how RAG techniques (semantic search + LLM generation) can be used in a healthcare administrative setting.

It achieves this by ingesting FAQ documents, embedding them, and then at runtime, retrieving the most relevant chunks for each query and using those as grounding for AI responses.

Who are the users?

The primary users of the application are:

- Medical clinic administrators and staff: who may deploy the assistant to handle frequently asked administrative questions from patients.
- Clinicians and support teams: who may use it internally to retrieve accurate answers to routine operational queries.
- Patients or visitors: by using a simple front-end chat interface, it allows them to ask standard questions about clinic logistics without requiring live human support.

What should its performance targets be?

Accuracy and Relevance:

- Retrieve the most relevant FAQ or administrative documents for each user query.
- Generate responses that are factually correct and strictly grounded in the retrieved context.

Response Quality:

- Maintain consistent answers when similar questions are asked multiple times.
- Use professional and appropriate language for a healthcare administrative setting.

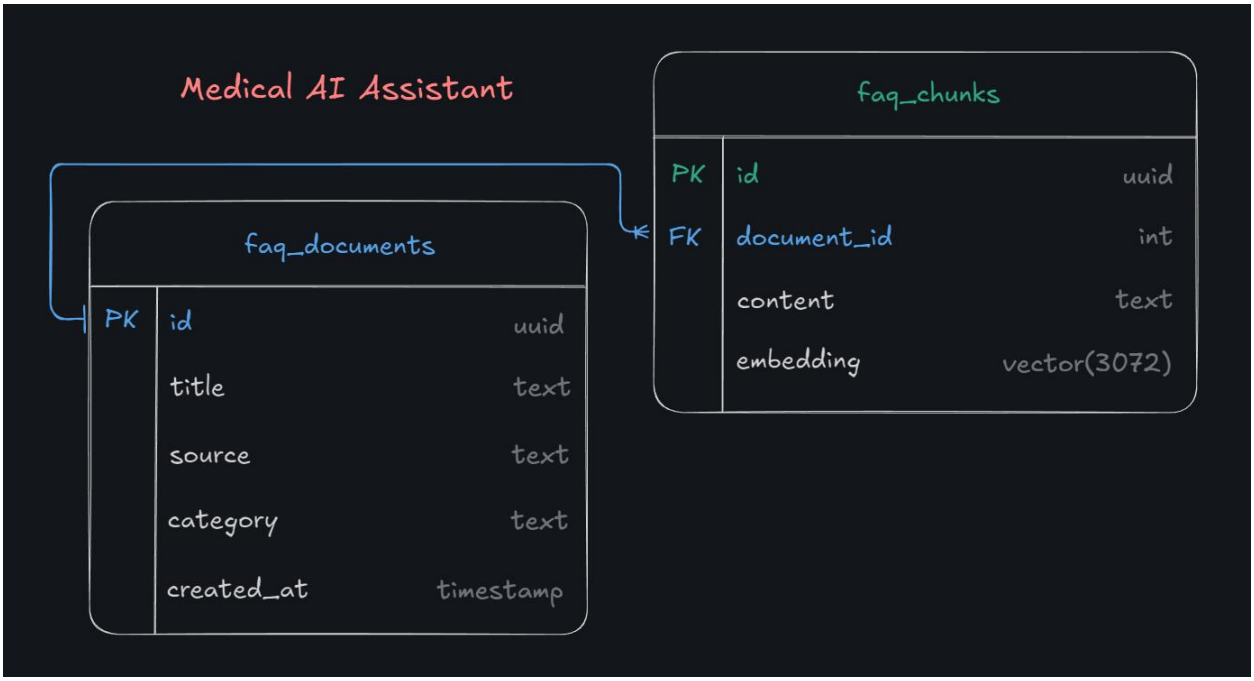
Safety and Scope Control:

- Avoid providing medical diagnoses, treatment advice, or clinical recommendations.
- Redirect users to human staff when medical advice is requested.

AWS Service Category Requirements

- **Compute:** For compute services, common options include Amazon EC2 and AWS Lambda, as they are widely used or general-purpose workloads. However in this project, I chose AWS Elastic Beanstalk to simplify deployment and infrastructure management. Elastic Beanstalk provides a Platform as a Service (PaaS), allowing the application to be deployed without manually configuring servers, load balancers, or scaling policies. This makes it well-suited for a small-scale academic project where ease of setup and maintainability are prioritized over fine-grained infrastructure control.
- **Storage:** Amazon Simple Storage Service (S3) was selected for storage due to its low cost, high durability, and ease of use. S3 is appropriate for storing static assets required by my application. Comparing to other storage options, S3 has minimal operational overhead and is sufficient for this application.
- **Networking and Content Delivery:** Amazon Virtual Private Cloud (VPC) is used to provide basic network isolation and control over resources. Amazon Route 53 was not used, as this project is for academic purposes and does no require a custom domain name. in a production deployment, Route 53 would be appropriate for DNS management. Amazon CloudFront was also one of the options, but it seems unnecessary, since the application does not have global performance requirements, or a large user base. Low-latency worldwide content delivery is not a priority for this use case.
- **Database:** Amazon RDS was selected as the database service because the application uses PostgreSQL, a relational database. Amazon DynamoDB was not chosen because it's a NoSQL database and does not align well with the structured relational data model used in this project.

Data



faq_documents: This is the master documents table that stores the original FAQ content from the medical facility, the FAQs here are not taken from any medical centres out there, they're fully generated by ChatGPT, but still resembles the questions that people would commonly ask most. You can take a look at the sample data in the picture below:

	id [PK] uuid	title text	source text
1	863526...	How to Book an Appointment	To book an appointment, you can: (1) Call our front desk at (555) 123-4567 during business hours, (2) Use our online patient portal at patient.example.com and click "
2	d36672...	Canceling or Rescheduling A...	To cancel or reschedule: Call us at least 24 hours in advance at (555) 123-4567, or use the patient portal. Late cancellations (less than 24 hours) may incur a $25 fee.
3	ac28ea...	Walk-in vs Scheduled Appoint...	We accept walk-ins Monday through Friday 8 AM to 11 AM for urgent matters only. Scheduled appointments are preferred and guarantee shorter wait times. Walk-in p
4	4a600c...	Booking Appointments for Fa...	You can book appointments for minor children (under 18) using your account. Adult family members must create their own patient portal account or call directly.
5	f6ffab7...	Regular Clinic Hours	Our clinic is open Monday through Friday 8:00 AM to 5:00 PM. Saturday hours are 9:00 AM to 1:00 PM. We are closed on Sundays. Last appointments are scheduled 3
6	49b868...	Holiday Hours and Closures	We are closed on major federal holidays: New Year's Day, Memorial Day, Independence Day, Labor Day, Thanksgiving, and Christmas. Check our website or call for spe
7	c1fabb6...	Emergency After-Hours Cont...	For medical emergencies after hours, call 911 or go to the nearest emergency room. For urgent but non-emergency questions, call our after-hours nurse line at (555) 1
8	f3623dd...	Prescription Refill Process	Request prescription refills through the patient portal, mobile app, or by calling your pharmacy directly. Allow 48-72 business hours for processing. Controlled substar
9	edc6c2...	How Long Prescription Refills...	Standard prescription refills take 2-3 business days. Expedited requests may be available for urgent needs - call our office at (555) 123-4567 to request.
10	391712...	Transferring Prescriptions to ...	To transfer prescriptions: Contact your new pharmacy and provide them with our clinic information. They will coordinate with us. Allow 3-5 business days for transfer
11	b358de...	What to Do for Lost Prescript...	If you lose a prescription: Contact our office immediately. Non-controlled medications can usually be reissued. Controlled substances require an in-person appointme
12	d04a69...	How to Access Lab Results	Lab results are available in your patient portal 3-7 days after testing. You will receive an email notification when results are ready. Log in to patient.example.com to vie
13	201ea1...	Lab Result Turnaround Time	Standard lab results: 3-5 business days. Complex tests: 7-14 days. Urgent results: 24-48 hours. Your doctor will contact you directly if immediate action is needed.
14	9bac36...	Understanding Your Lab Resu...	Each lab result includes a normal reference range. Values outside this range will be flagged. If you have questions about your results, message your doctor through th
15	1226c9...	What If My Results Are Abnor...	If your results are abnormal, your doctor will contact you within 3-5 business days to discuss next steps. Urgent abnormal results will be communicated within 24 hou

faq_chunks: This is the embeddings table that stores processed chunks of the documents for semantic search. To expand more on the fields:

- Foreign Key Relationship: Each chunk is linked back to its parent document via document_id.
- Content Field: Contains text segments extracted from the original documents (chunked for optimal embedding generation).

- Embedding Vector: A 3072-dimensional vector representation generated by the Gemini API's text-embedding-001 model (which produces 3072-dimensional embeddings).

Sample data:

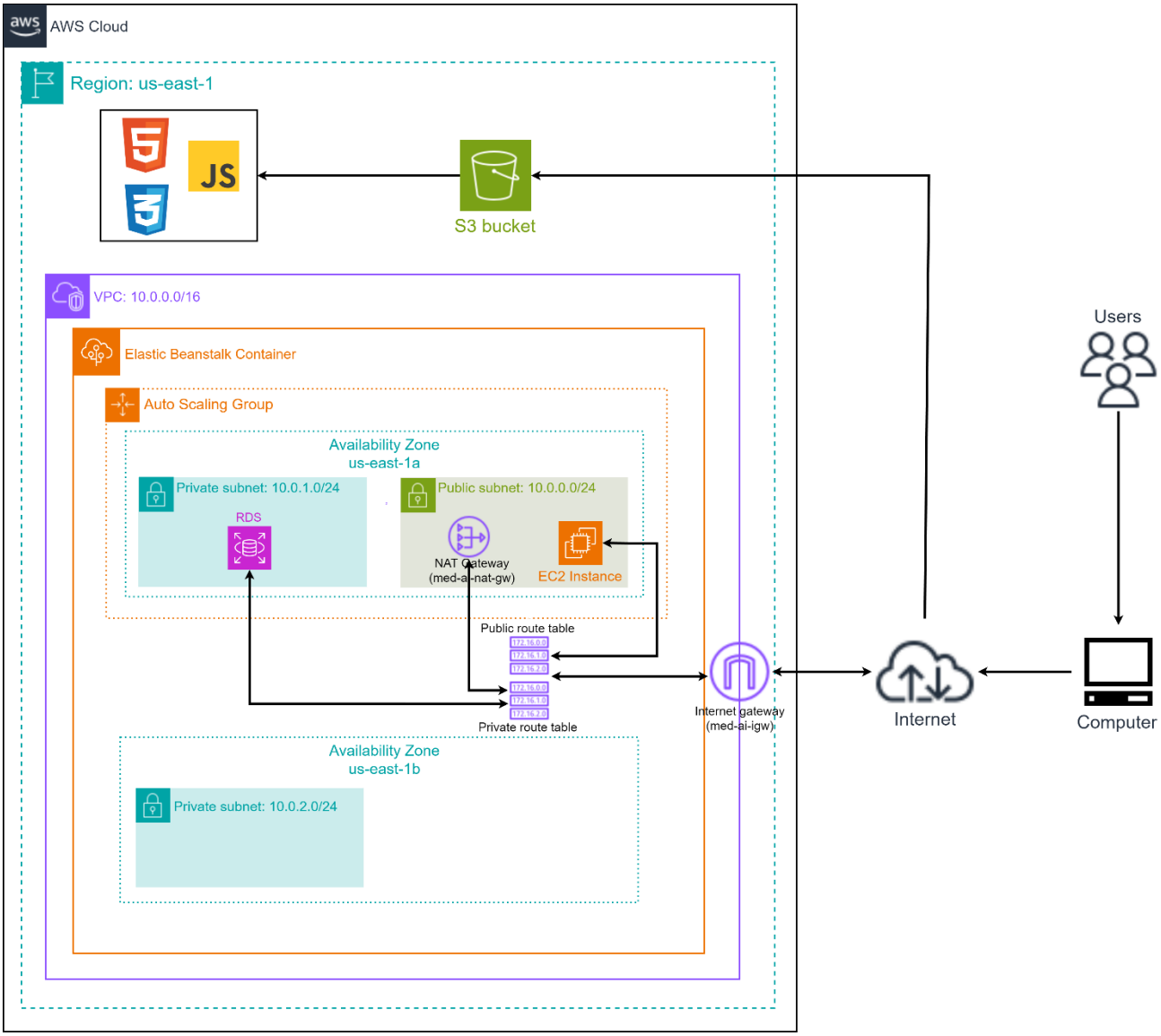
	id [PK] uuid	document_id uuid	content text	embedding vector
1	5c4ed33...	8635264...	How to Book an Appointment To book an appointment, you can: (1) Call our front desk a...	[0.004098613,-0.003980949,0.018746769,-0.04054365,-0.0037135556,-0.021686854,0.005
2	39a895...	8635264...	"Schedule Appointment", or (3) Use our mobile app. New patients should call first to regi...	[0.0015273156,-0.0013587141,0.019400336,-0.03705138,0.004012172,-0.013477473,0.012
3	2d4ba9...	d36672a...	Canceling or Rescheduling Appointments To cancel or reschedule: Call us at least 24 ho...	[0.0014721435,0.022425149,-0.022950461,-0.06545299,-0.0076685827,-0.0044867876,-0.0
4	d88347...	d36672a...	llations (less than 24 hours) may incur a $25 fee. Emergency cancellations are exempt fr...	[0.015946183,0.010655752,-0.01898972,-0.04600221,0.0057835598,-0.004350073,-0.0057
5	b4d6e6e...	ac28ead...	Walk-in vs Scheduled Appointments We accept walk-ins Monday through Friday 8 AM to ...	[-0.024672154,0.01904245,0.0060923533,-0.011032904,-0.025190722,-0.003152122,0.013
6	fa15623...	4a600cd...	Booking Appointments for Family Members	[-0.0062107365,-0.010658368,0.0008236459,-0.038092803,0.006842877,0.018054109,0.00
7	9cdcd3a...	f6ffab7a-...	Regular Clinic Hours Our clinic is open Monday through Friday 8:00 AM to 5:00 PM. Satu...	[-0.016256912,0.02032569,0.024123998,-0.011672722,0.01304492,-0.004817905,0.002576
8	2d25d3...	49b8680...	Holiday Hours and Closures We are closed on major federal holidays: New Year's Day, M...	[0.010060436,0.010762343,-0.00986075,-0.03179257,0.006318365,-0.006785301,-0.01254
9	525c115...	c1fabb6...	Emergency After-Hours Contact For medical emergencies after hours, call 911 or go to t...	[0.0060724085,0.02504428,-0.013294692,-0.039801497,0.004965363,-0.014574565,0.0186
10	218423...	f3623dd...	Prescription Refill Process	[0.02013894,-0.010953897,-0.0019810912,-0.0619561,-0.003894208,-0.009274271,0.01441
11	a98ea64...	f3623dd...	rectly. Allow 48-72 business hours for processing. Controlled substances require a docto...	[-0.0010675668,-0.029087521,-0.02940758,-0.07413287,0.010264762,0.0047961166,0.052
12	f788b64...	edc6c25f...	How Long Prescription Refills Take	[0.009283274,-0.011302121,0.004353673,-0.058587413,0.00745879,-0.026227409,0.01163
13	b05b52c...	3917122...	Transferring Prescriptions to Another Pharmacy To transfer prescriptions: Contact your ...	[0.0051995222,-0.014761065,0.004322338,-0.041738424,-0.011556546,-0.020432469,0.01
14	213cf8e...	b358de4...	What to Do for Lost Prescriptions If you lose a prescription: Contact our office immediat...	[-0.0009728398,-0.006256683,0.00074916444,-0.03123959,-0.0109943105,-0.010423704,0.
15	175224...	d04a694...	How to Access Lab Results Lab results are available in your patient portal 3-7 days after ...	[-0.0042147744,-0.004547489,-0.015193551,-0.05276027,-0.0066772,-0.007329131,0.0146

Why no User Tables?

There are no user tables in this database schema. This is an intentional design decision I've made for several important reasons:

1. Simplicity: Keeping the application minimal and focused solely on providing information without the complexity of user authentication, session management, or personalized data storage.
2. Privacy & Compliance: Avoiding the storage of any user-specific information, medical records, or personal health information (PHI) eliminates significant HIPAA compliance burdens and privacy concerns. The chatbot doesn't know who is asking questions or remember previous conversations.
3. Realistic Scope: Working with actual medical data, especially patient records and PHI, requires extensive legal, ethical, and technical considerations – determining what's appropriate to collect, how to secure it, consent requirements, and regulatory compliance would be extremely time-consuming and complex.
4. Common Use Case: Administrative and general information queries (appointment booking hours, prescription refills, lab result timelines, etc.) represent a realistic and valuable real-world problem that medical facilities face daily. These are the questions that flood phone lines and front desks, and an AI assistant can handle them effectively without needing to access patient-specific data.
5. Stateless Architecture: Each query is independent, users can ask questions anonymously through the web interface, receive answers based on the facility's FAQ knowledge base, and no conversation history is retained. This makes the system faster, simpler, and more secure.

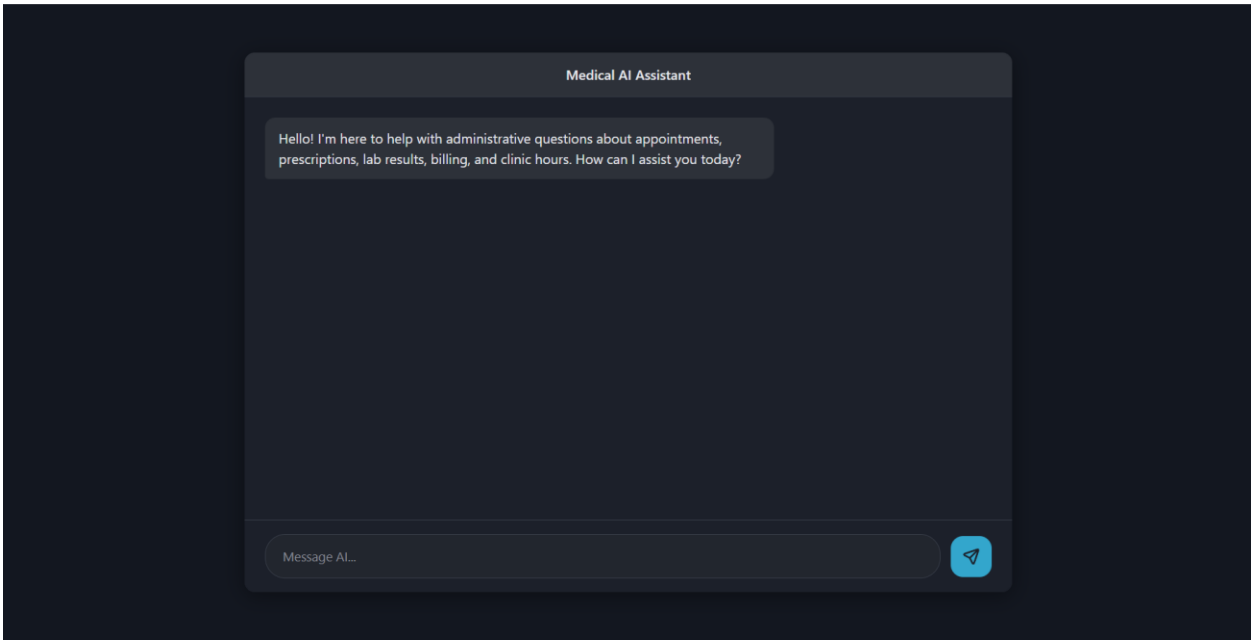
Architecture Diagram



Architecture Flow & Component Purposes

User Access & Frontend (Outside VPC)

Users access the chat web application through their browser via the **Internet**. When they visit the website URL, their browser requests the static files (HTML, CSS, JavaScript) from the **Amazon S3 bucket**, which hosts the frontend of the application. S3 is ideal for this because it provides cost effective. Once the files are downloaded, the browser renders the chat interface where users can ask questions like “How do I book an appointment?” or “What payment options do you offer for large medical bills?”, etc.



Entry into the VPC

When a user sends a message through the chat interface, the JavaScript running in their browser makes an API call to the backend. This request travels through the **Internet** and arrives at the **Internet Gateway**, which sits at the boundary between the Internet and resources within the **VPC (10.0.0.0/16)**. The Internet Gateway servers two critical functions:

- Routes incoming API requests from users' browsers into the VPC.
- Allows outbound requests from resources inside the VPC (like EC2 instances) to reach external services on the internet, such as the Gemini API.

Public Subnet & Backend Processing

The Internet Gateway routes the incoming requests to the **public route table**, which directs traffic with destination 0.0.0.0/0 to the Internet Gateway. This public route table is associated with the **public subnet (10.0.0.0/24)** in the **Availability Zone us-east-1a**, where the backend infrastructure resides.

In this public subnet, we have the **Elastic Beanstalk environment**, which manages the entire backend deployment. Elastic Beanstalk automatically handles the orchestration of several components:

- **Auto Scaling Group:** The Elastic Beanstalk is using a single-instance environment, so the ASG is here to ensure that the application has at least 1 instance running.
- **EC2 Instance:** This runs the **Node.js + Express backend application**, which contains the core RAG (Retrieval-Augmented Generation) logic.

When an EC2 instance receives the user's query (e.g. "How do I see my lab results?", etc.), it orchestrates the following workflow:

1. Semantic search: Using the embeddings stored in the database, the instance queries the **RDS PostgreSQL database** (located in the private subnet) to find the most relevant documents from the knowledge base.
2. Generate answer: The instance sends the retrieved context documents along with the user's original question back to the **Gemini API** to generate a natural language response.
3. Return response: The AI-generated answer is sent back through the Internet Gateway to the user's browser.

Now you might be wondering, this is a RAG, where's the Embedding process? Initially before starting this chat application, the embedding must be done manually by calling an endpoint. This endpoint will read all the rows in the faq_documents table in the PostgreSQL database, then it will put the text into chunks, and with those chunks, they will be embedded into vector embeddings and then stored into the vector database.

The EC2 instance need to be in the public subnet because they must accept incoming traffic from the Internet (user requests) and make outbound calls to external API (Gemini). The public route table ensures both inbound and outbound internet connectivity work.

NAT Gateway & Outbound Private Connectivity

Also located in the public subnet is the **NAT Gateway**. While our EC2 instance is in the public subnet and have direct internet access, the NAT Gateway serves a different purpose: it provides secure outbound internet access for resources in the **private subnets** without exposing them to inbound traffic from the internet.

The NAT Gateway connects to the **private route table**, which is associated with the private subnets. This private route table directs outbound traffic (0.0.0.0/0) to the NAT Gateway, which then forwards it to the Internet Gateway and out to the Internet. This setup is very important for scenarios where private resources (like RDS) might need to download patches, security updates, or access external services while remaining inaccessible from the public internet. However, in this specific architecture, the RDS database doesn't actually need internet access, it only communicates with the EC2 instance, but the NAT Gateway is there to follow AWS best practices.

Private Subnets & Database Layer

The backend EC2 instance communicate with the **Amazon RDS PostgreSQL database**, which resides in the private subnet. There are two private subnets:

- Private subnet 10.0.1.0/24 in us-east-1a
- Private subnet 10.0.2.0/24 in us-east-1b

If you notice, we're not using the subnet in us-east-1b to store RDS instance at all. This is because RDS requires that there's at least 2 Availability Zones, even if I'm just going with a Single AZ DB instance deployment.

The RDS database stores two critical types of data:

1. Document embeddings (vector representations of knowledge base articles)
2. Original documents (the actual content about appointments, billing, lab results, etc.)

RDS is placed in the private subnet as a security best practice, since it should never be directly accessible from the internet. Only the backend EC2 instances can communicate with it through tightly controlled **security group rules**. The EC2 instances' security group allows outbound connections to the RDS security group on port 5432 (PostgreSQL's default port), while the RDS security group only accepts inbound connections from the EC2 instances' security group.

What programming languages did you use (and why) and what parts of your application required code?

This application consists of both a frontend and a backend, each requiring custom code. For the frontend, I used HTML, CSS, and JavaScript rather than a more robust framework such as React. Since the frontend is not the core focus o this project, this priority was simplicity and rapid development. Additionally, static frontend assets built with HTML, CSS, and JavaScript can be easily hosted in an Amazon S3 bucket, which simplifies deployment to the cloud. This makes them a practical and efficient choice for the project.

For the backend, I used Node.js with Express. While Python is often preferred language for chatbot and Retrieval-Augmented Generation (RAG) applications, I chose Node.js because of my stronger experience with it. This allowed me to work more efficiently and avoid unnecessary time spent on syntax issues or debugging unfamiliar tooling. Furthermore, the Gemini API provides official support for Node.js, making it well-suited for implementing RAG techniques within this technology stack.

Security

Currently, the application implements several security measures across different architectural layers to protect data and system resources.

- At the network level, the application leverages **Amazon VPC** to create a logically isolated network environment where all backend infrastructure operates within a controlled boundary.

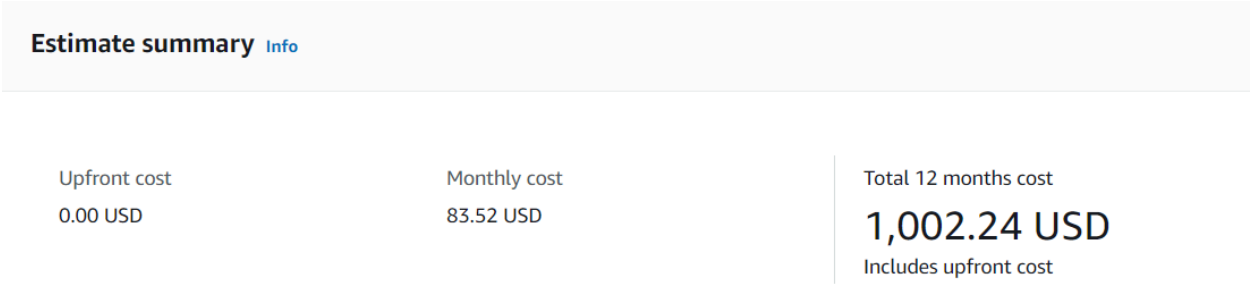
- The **database layer (RDS PostgreSQL instance)** is deployed in a private subnet. Which makes have no direct routes to the internet, and only giving EC2 the permission to communicate with the database.
- All credentials, **environment variables** are stored in the Elastic Beanstalk environment variables rather hardcoding them in the source code.

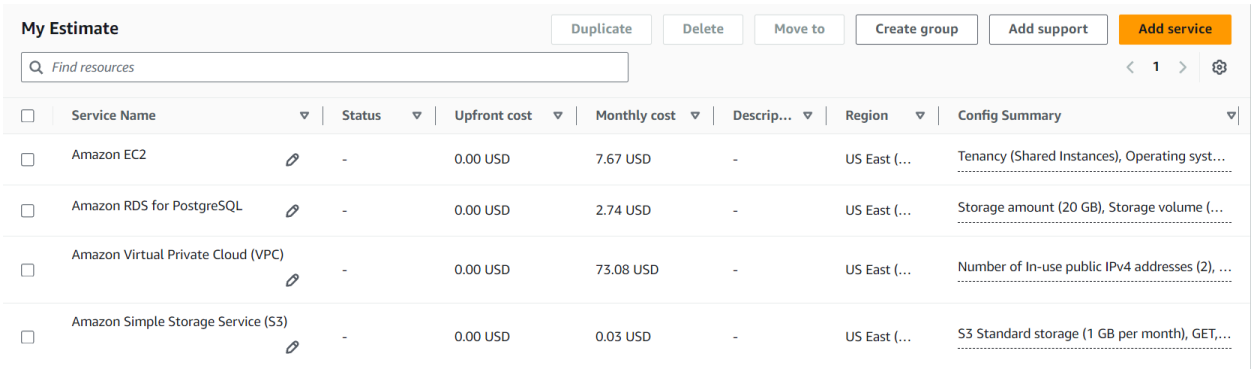
Despite the mentioned security measures, there are some several vulnerabilities.

- The **EC2 instance is currently being deployed in a public subnet with direct internet access**. While this simplifies the architecture and allows the instance to both receive user requests and make outbound calls to the Gemini API, it also means the instance have public IP addresses and are directly exposed to potential attacks from the Internet. The recommended solution would be to introduce an Application Load Balancer (ALB) in the public subnet and move the EC2 instance to a private subnet. The ALB here would be receiving all incoming traffic from the internet and forwarding it to the EC2 instance in the private subnet.
- **Lack of proper Identity and Access Management (IAM) implementation**. Due to the restrictions of the AWS Academy Learner Lab environment, IAM roles and policies cannot be fully configured for this project. However, in a production environment, IAM would be essential for implementing the principle of least privilege across all AWS resources.
- The **frontend is currently hosted on S3 with HTTP access**, meaning data transmitted between users' browsers and S3 is not encrypted. This exposes the static files and any initial requests to potential man-in-the-middle attacks. The solution is to implement Amazon CloudFront, AWS's content delivery network, as a layer in front of S3, CloudFront can serve the frontend over HTTPS, ensuring all communication with users is encrypted.

Cost

Here's the cost metrics to operate my current system.





As you can see, the estimate for VPC costs a lot, a lot of that comes from the NAT Gateway. While building this application, I noticed that my NAT Gateway is unnecessary and can be removed, since it's only used for outbound internet access from private subnets. Right now there's zero traffic but I'm still paying for the hourly cost of the NAT Gateway, so it would be best if I remove it, which ultimately will cut down the cost a lot.

Evolving the application

If I were to continue the development of this application, I would make the following improvements:

- **Remove the NAT Gateway**, as it does not serve a practical purpose in the current architecture. Since the EC2 instance resides in a public subnet, the NAT Gateway is unnecessary and only causes costs to increase rapidly without providing any tangible benefit.
- **Re-architect the networking layer** by placing the EC2 instances in private subnets and deploying an Application Load Balancer (ALB) in a public subnet. This would improve security by preventing direct public access to the instances while still allowing the application to be accessible to users.
- **Enhance the RAG chatbot implementation** if the application were to be used in a real-world setting. The current chatbot serves only as a proof of concept. A production-ready solution would require a more robust, reliable, and well-evaluated RAG pipeline to handle real users effectively, such as visitors accessing a medical center's website to ask frequently asked questions.
- **Introduce high availability and scalability**, as the current deployment focuses solely on achieving a successful cloud deployment. A more comprehensive architecture would be required to support scaling, fault tolerance, and potential global availability, for example by leveraging auto scaling groups, multi–Availability Zone deployments, and additional AWS managed services.