University of East London

Pioneering Futures Since 1898

SCHOOL OF ARCHITECTURE, COMPUTING AND ENGINEERING
Department of Engineering and Computing

Medical Assistant Chat Bot

AHSAN ALI JANJUA
2545859

A report submitted in part fulfilment of the degree of
BSc (Hons) in *Your Programme*
Supervisor**:** Shaheen Khatoon

CN60007

10 May 2025

Abstract

Purpose – There is an increasing need for abrasion and reliable healthcare solutions; thus, there have been developed AI-based tools. This thesis is concerned with the design, implementation, and testing of a medical assistant chatbot with the use of OpenAI's GPT-4-turbo-preview for conversational AI, and Supabase as the backend. The main functions of the chatbot include symptom checking (general conversation) interpretation of medical reports and mental health. The project measures the performance of the chatbot against the performance of currently available AI-derived systems by adopting empirical research methodology.

Design/Methodology/ Approach – The proposed approach to research follows an empirical research method looking at the system performance quantitatively using literature-driven test cases. The process leading to the development was conducted following an Agile process and this allowed for constant testing and network of correction in sprints. In the comparison of the system's performance, we considered the accuracy, explanation clarity, and safety of recommendations, response time, and cost-efficiency composing the parameters with the utilization of both synthetic and clinical test cases. For chat completion, OpenAI GPT-4-turbo-preview model was used, for database and backend functionality, Supabase is being used.

Findings – The results show promise in showing that the chatbot makes healthcare information accessible, especially for non-emergency conditions, with benchmark of performance that equal to the existing healthcare AI systems. The Chatbot demonstrates an ability to offer solutions, it also presents the problem of token costs and accuracy with complex and rare diseases along with the ethical connotations of AI-created healthcare solutions. The findings provide a basis for possible real-world application in clinical and patient-settings as an additive tool to healthcare services rather than a replacement for human care.

Originality/Value – This research adds to the burgeoning subject of AI in healthcare through investigation of conversational AI's and database management into a chatbot for medical aid. The comparison of the chatbot with established systems as regards evaluation closes a research gap in the use of AI for patient support. Furthermore, the research offers insights into the prospects of AI driven systems in healthcare as a path towards a deployment of the same and enhancement of the same.

Acknowledgments

I would like to thank my supervisor Shaheen Khatoon and my module lecturer Dr Fadi Safieddine who guided and provided valuable feedback and support during the project. My acclamations also go to my peers and the community resources from OpenAI, Supabase, GitHub, Next.js, Tailwind CSS, Cursor, Type Script, Stack Overflow, whose tools and documentation played a big role in developing this project.

Contents

Chapter 1: **Introduction**

1.1 Introduction

This chapter outlines purpose of doing this project, Problem Statement, Aim of the project, Objectives of the project and Structure of the project.

1.2 Problem Statement

There are problem concerning timely, reliable, and available medical support that healthcare industry may face especially in remote or underserved areas. Modern healthcare Providers rarely provide real time support online which is leading to gaps in patient care. I want to provide research and protype implementation of the project which can help fill the gap in patience care.

1.3 Aim of the project

To develop a demo AI-powered medical assistant chatbot that provides users with accessible information regarding health care- basic symptom checking, medical report interpretation, and mental health support, which will improve healthcare accessibility for patients and improve patient engagement.

1.4 Objectives

- To conduct research on currently implemented AI conversational agents in the healthcare industry and their use cases.
- To research currently published literature about AI Technologies in the health care industry.
- To conduct empirical research on existing system and do comparative analysis on our proposed system.
- To analyze the results from research and find areas of improvement for future work.
- To design and create a demo conversational AI agent that will take users queries for symptoms or general medical related queries and answer them, take medical reports and summarize the report, sympathetic mental health support option.
- To test the chatbot's functionality, its knowledge for medical-related inquiries, and its accuracy for responses.
- To evaluate the test results and make improvements for the demo Medical Assistant AI chatbot.

1.5 Structure of the Project

- **Chapter 1: Introduction**

- **Chapter 2: Literature Review**
- **Chapter 3: Project Methodology**
- **Chapter 4: Implementation**
- **Chapter 5: Results and Evaluation**
- **Chapter 6: Discussion**
- **Chapter 7: Conclusion and Future Work**

Chapter 2: **Literature Review**

2.1 Introduction

There are many challenges currently in healthcare industry because of an ever growing demand of patients, rising costs of maintaining infrastructure and work force shortages. The solution to these problems is, however, Artificial Intelligence, specifically conversational AI agents (or chatbots) which is a huge disruptor in the areas. Medical assistant chatbots can help streamline routine questions, analyse symptoms and increase the availability of medical information. These systems can be used to provide medical advice services which can reduce the burden on healthcare workforce and increasing efficiency (Ting et al., 2021).
In addition integrating AI in healthcare proper fitting with currently occurring digital transformation in the healthcare sector. From studies, AI-powered chatbots can be used to decrease response times and providing personalized healthcare information to the users, and therefore facilitate timely intervention (Kumar et al., 2022). Nonetheless, although the possible use of such systems is enormous, issues, such as the issue of data privacy, bias that is algorithmic, and reliability of technology, remain, which calls for more research and development (Topol, 2019).

This paper reviews the basis, application, advantages, and disadvantages of AI powered medical assistant chatbots, including centralized and decentralized architectures, software tools, and ethical concerns. The review considers what is currently available, the trade-off between technological promise and practicability, and gaps in the literature that this project seeks to fill. It seeks to build on existing academic and industrial knowledge to understand the impact of the chatbot in healthcare and provide a full description of the chatbot characteristics.

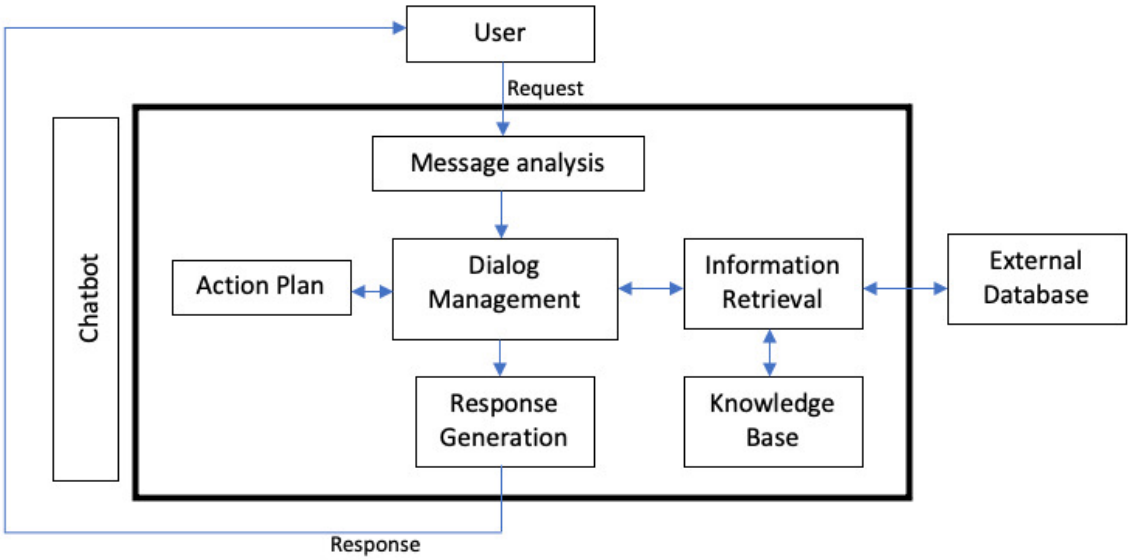


Figure 1. Figure: Simple Chatbot Architecture (Chatbot for Health Care and Oncology Applications Using Artificial Intelligence and Machine Learning: Systematic Review,2021, PMC, NCBI)

2.2 Medical Assistant Chatbot

A Medical Assistant Chatbot is an AI driven conversational agent, which assists users in information about healthcare, symptoms analysis, and preliminary medical recommendations. Chatbots using technologies such as Natural Language Processing (NLP) and machine learning take the user inputs in their plain language and analyze symptoms giving a personalized response. Though it is not an alternative for professional medical consultation, they are a complimentary tool, giving the person immediate access to verifiable medical information (Sharma et al., 2020). Medical chatbots have become increasingly popular in health care industry because they can relieve the burden of work of medical professionals by automating routine work such as answering frequently asked questions or triaging non-emergency cases. For example, the AI solutions that are developed by IBM Watson Health can diagnose diseases, suggest solutions to the illnesses given a patient's symptoms and medical history (Watson Health, 2021). In the same light, Babylon Health has developed a chatbot that is able to analyze symptoms and help them to the right care (Babylon health, 2021).

Such chatbots are especially helpful in those environments where access to medical facilities is not so easy. By being an intermediary between patients and doctors they give a stopgap solution to the underserved populations. Nonetheless, both accuracy and functionality of them rely heavily on the quality of the data that trained them and the algorithms that drive them. As a developer there are some ethical considerations that need to be considered when developing AI chatbots such as security of data, AI System Bias, and privacy (Jiang et al., 2021).

2.3 Existing AI Chatbot Systems

Medical assistant chatbots are applying automation in the aim of minimizing routine interactions and making access to medical information easier. Their needs are diverse from symptom checking to patient education. Afterwards, I take a detailed look about how they are used and then, give well-structured tables, describing their main fields of use and benefits.

2.3.1 Symptom Checking

Perhaps its most widespread use is diagnosis of user provided symptoms and figuring out if there is a possible diagnosis; or whether medical consultation is necessary. This is where Babylon Health's Symptom Checker and Ada Health among others come in. Through offering rapid pre-assessments, they minimize unnecessary trips to the hospital, making time for the patients and the health professionals too, (Babylon Health, 2021).

2.3.2 Mental Health Support

Secondly, chatbots are also used for mental health services. For example, behind Woebot, there is conversational AI that offers cognitive behavioral therapy (CBT) techniques to address those with anxiety or depression using tools. During the COVID-19 PANDEMIC This application was particularly useful since traditional therapy was not available (Fitzpatrick et al., 2017; Larsen et al., 2021).

2.3.3 Medication Reminders

Medications patients that Medsii helps patients keep to their drugs schedules by sending them reminder on taking drugs and information on how the drugs will interact with each other and their side effects as well. The role helps improve the following of prescribed treatments which reduces health complications (Razzaki et al., 2018).

2.3.4 User Education

They use chatbots to inform users using evidence-based responses to common health questions. For example, Mayo Clinic's chatbot provides you access to valid, accessible medical information. These chatbot are key in spreading useful information of the public health like vaccination campaigns and methods to prevent disease (Mayo Clinic, 2020).

2.3.5 AI Agents

In clinical settings, chatbots can eliminate repetitive appointment scheduling, receptionist duties and leaving a follow-up call to the eventual patient. This increases efficiency in operations and cuts down pressure on medical personnel (Topol, 2019).

2.3.6 Drawbacks

Although these applications show the potential of the chatbots, the obstacle such as regulatory compliance, the ability to maintain the conversational accuracy and the ethical concern such as the data privacy stand as key barriers to wide-spread adoption (Jiang et al., 2021).

2.4 Key AI Algorithms used in Chatbots

Medical assistant chat bots develop on the frameworks that utilize AI algorithms to serve as natural language processing and to give accurate diagnoses or health advice. This domain is based on the most widely used AI techniques which are Natural Language Processing (NLP) and Machine Learning (ML). These algorithms are chatbots own ability to parse medical queries, analyze symptoms and recommend possible conditions. I present an overview of these key algorithms, and their use cases and challenges, below.

1. Natural Language Processing (NLP)

Medical assistant chat bots build off of frameworks that use engines based on AI algorithms and use it as a natural language processing and accurately diagnose or provide healthy advice. Based on the most widely AI techniques that are Natural Language Processing (NLP) and Machine Learning (ML) this domain operates. These algorithms are chatbots capability to parse, analyse medical query and symptoms and suggest possible conditions. Below I provide an outline of some important algorithms, and their applications and shortcomings.
1. Natural Language Processing (NLP is very important to the medical assistant chatbots because it allows the system to understand human language. NLP techniques are used in medical chatbots based on which diseases, symptoms, and treatment are identified from the input of the user. Important NLP techniques include:
• Named Entity Recognition (NER): The recognizes medical terms (e.g. Symptoms, diseases).
• Part-of-Speech Tagging (POS): It assists the chatbot to know the sentence structure; the context.
• Dependency Parsing: Figures out how words are related to each other in order to enhance interpretation. For instance, Ada Health applies NLP to analyse some symptoms and issue some initial diagnostics. The chatbot correlates the extracted entities (fever or headache, etc.) to a medical database (Kumar et al., 2022).

Example Pseudo Code for NLP

```
def process_symptoms(user_input):
   # Use NER to extract symptoms
   symptoms = named_entity_recognition(user_input)
   diagnosis = match_symptoms_to_conditions(symptoms)
```

return diagnosis

2. Machine Learning (ML) Algorithms

Machine learning enables chatbots to predict diagnoses based on historical data. Commonly used ML techniques include:

- **Decision Trees:** Classify symptoms into possible conditions.
- **Random Forests:** Improve decision-making by combining multiple decision trees.
- **Neural Networks:** Used for deep learning, particularly in recognizing patterns in unstructured data like medical texts or images.

For example, **Babylon Health** uses decision trees to suggest diagnoses based on symptom inputs, and neural networks to refine its suggestions by learning from historical patient data (Babylon Health, 2021).

Example Pseudo Code for ML Diagnosis:

```
def diagnose(symptoms):
    # Use a decision tree for diagnosis
    model = train_decision_tree(data)
    diagnosis = model.predict(symptoms)
    return diagnosis
```

3. Use Cases of AI Algorithms

- **Symptom Checking:** Babylon Health's chatbots are fed by NLP and ML to suggest diagnoses from symptoms. The chatbot would for instance suggest that a user suffers from conditions like "flu" or "cold," in case a user reports symptoms such as "fever" and "chills."
- **Treatment Recommendations:** ML models can recommend treatments based on symptom data, historical outcomes, and treatment effectiveness, enabling chatbots to assist in clinical decision support (Sharma et al., 2020).
- **Mental Health Support: Woebot** uses NLP to conduct cognitive behavioral therapy (CBT), providing personalized therapeutic interactions for mental health support (Fitzpatrick et al., 2017).

4. Challenges

Despite the successes of AI in medical chatbots, several challenges remain:

- **Data Quality and Bias:** Poor or biased data can lead to inaccurate diagnoses (Jiang et al., 2021).

- **Interpretability:** Deep learning models are often seen as "black boxes," making it difficult to understand how decisions are made, which is crucial in healthcare (Topol, 2019).
- **Scalability:** As data volume grows, maintaining performance while ensuring accurate predictions becomes a challenge.

5. Future Directions

Future developments in AI for medical chatbots are focused on improving **explainability** (via **Explainable AI**, or XAI), reducing biases, and increasing system scalability to handle larger user bases effectively (Larsen et al., 2021).

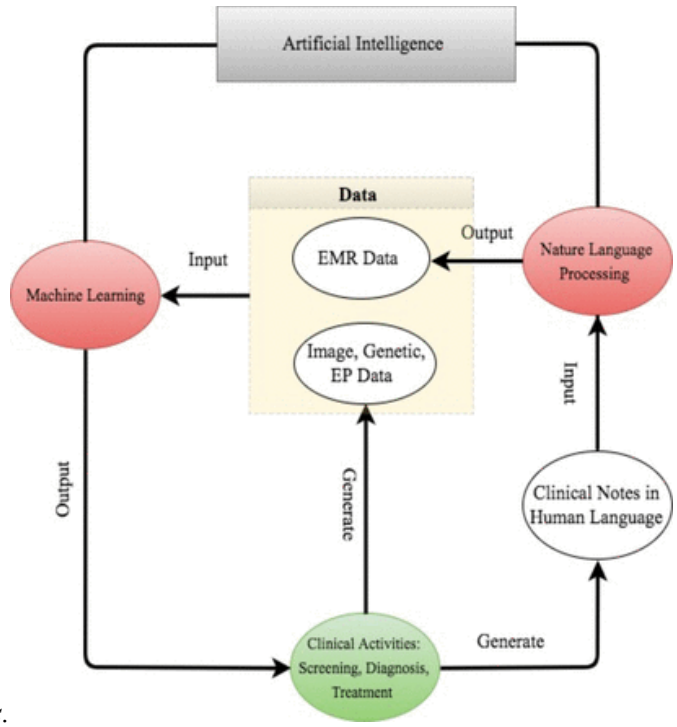


Figure 2. Flow of NLP models.

2.5 Large Language Models (LLMs) in Healthcare Chatbots

Large Language Models understanding ability has made chatbots more efficient for health care related tasks due to recent advancements in large language models (LLMs) such as GPT-3.5 and GPT-4 by OpenAI. LLMs are trained by massive medical datasets enabling LLMs to identify symptom patterns, interpret medical terminology, and produce human-like replies (Singhal et al., 2023). Google Research's Med Palm achieved LLM performance better than 85% on the USMLE medical licensing exam (Singhal et al., 2023). Like in GPT-4 case, OpenAI's GPT-4 was found to outperform GPT-3.5 on medical QA datasets with a higher top 1 and top 3 diagnosis accuracy (Nori et al., 2023).

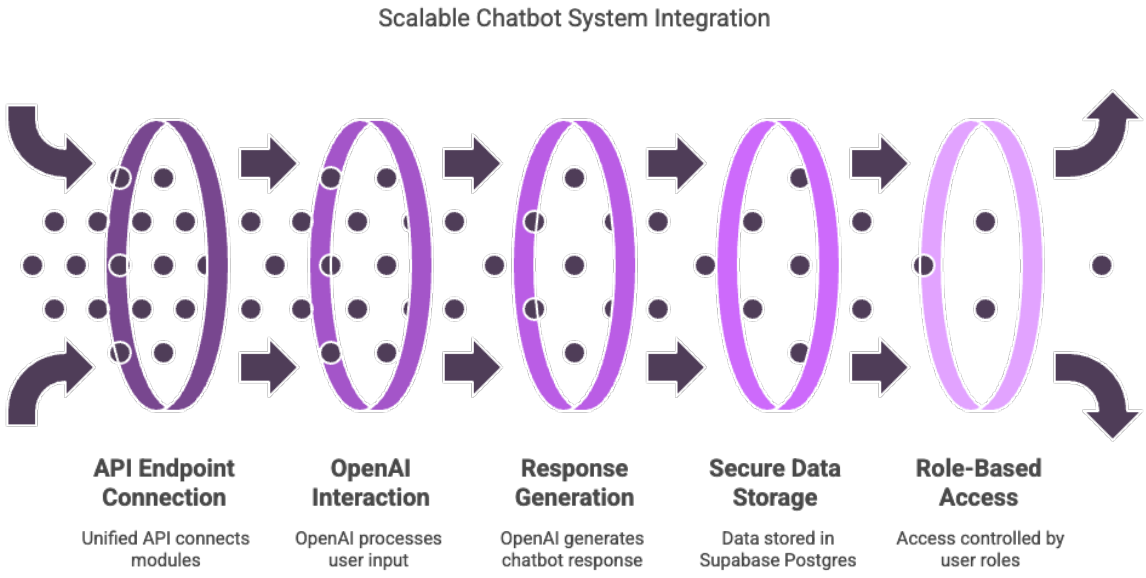
LLMs support a smooth integrations of functions like symptom checking, report summarization or mental health support with no need to create unique models per functionality. Nevertheless, LLMs

still have some challenges in terms of explainability reliability and accuracy in terms of complex medical queries (Patel and Lam, 2022).

2.6 OpenAI and Supabase in Modern AI Development

A new standard has risen in terms of rapid chatbot deployment in the form of OpenAI integrated with Supabase, a scalable backend-as-a-service. Supabase takes care of real time messaging, secure data storage and RESTful APIs, while leaving the developers with frontend and AI logic (Supabase, 2024). OpenAI's API (Chat Completion Endpoint) returns responses to user prompts based on pretrained models. (OpenAI, 2024).

Figure 3. Open AI and Supbase Integration

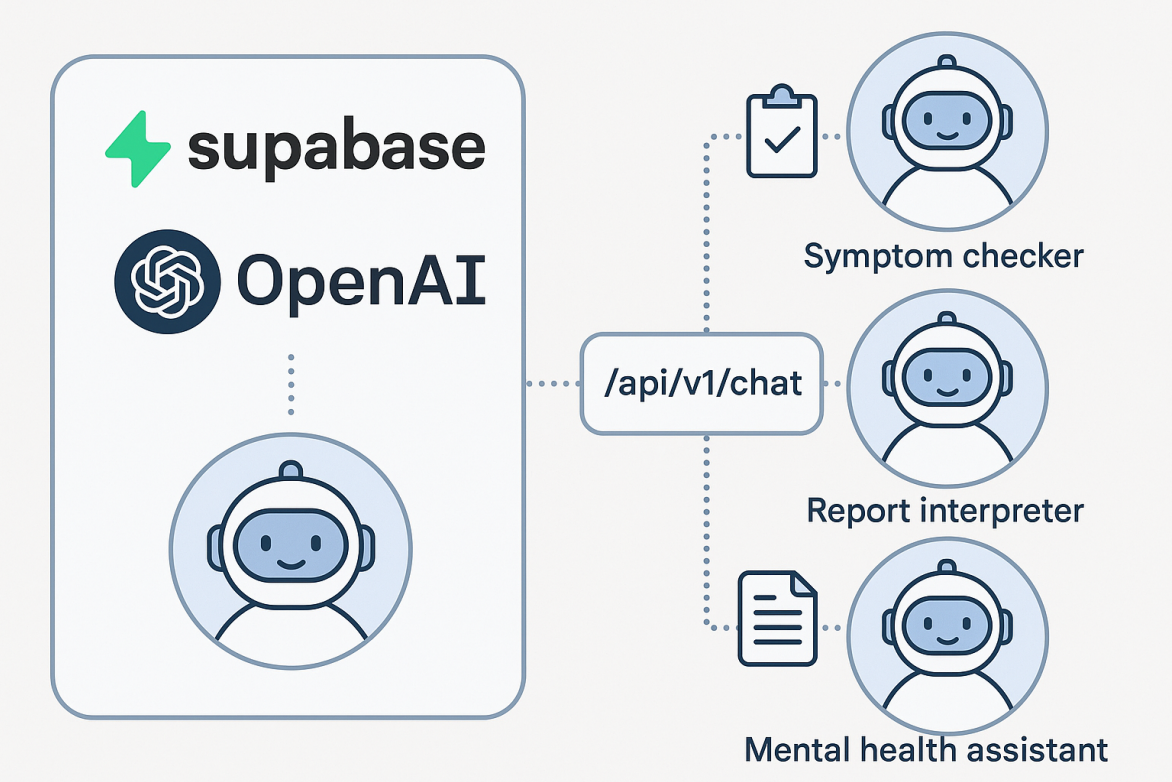


2.7 API Integration and Chatbot Responsiveness

A strong API design is pivotal for a good-performance medical chatbot. Zhu et al., 2022 research advocates that structured APIs decrease user frustration through prompt, regular and medically accurate answers. As one example, Ada Health uses Decision tree based API backend, Babylon, on the other hand, uses ML classifiers. On the other hand, systems based on OpenAI use generative AI, with the latter one is implemented through endpoint APIs, which need token management, summarization of context and prompt engineering (Singhal et al., 2023).

One end point /api/v1/chat can play many roles via a modular request body. It reduces code duplications which can make code complex and guarantees quick response handling. Rate Limiting and Retry logic is important in this architecture to avoid request failure in high user load (Ravuri et al., 2023).

Figure 4. Visualization of one Api/v1/chat end point.



2.8 Technological Challenges and Limitations

There is a lot of potential in the medical assistant chatbots in healthcare but there are many technological issues that need to be addressed. The challenges that then influence the effectiveness, reliability, and adoption of AI driven systems. Below is a more detailed table describing these key challenges:

Table: Key Technological Challenges in Medical Assistant Chatbots

Challenge	Description	Impact
Data Quality	False, or insufficient, or not representative data used for training AI models.	Causes inaccurate diagnoses particularly when facing rare conditions or when loosely informed.
Algorithmic Bias	Biases in training data-that do not engender diversity in population or algorithms.	Outcomes in unequal care, with implications to underrepresented groups, like ethnic minorities or poor populations.
Scalability	The problem of performance degradation with an increasing volume of users and data.	Can result in slower response time, and poor accuracy and in high-demand scenarios or large healthcare systems.
Privacy and Security	Issues concerning the security handling of user data and regulation compliance.	Breach of data and risks of violations of privacy laws that can destroy user's trust in the organization and cause legal problems.

System Integration	Chatbots integration challenge with the existing healthcare systems or platforms.	Restricts the chatbot's effectiveness in actual world healthcare scenarios in which interoperability is important.

2.9 Ethical Implications of AI in Healthcare

As AI is incorporated in the healthcare such ethical issues that come up include medical assistant chatbots and must be discussed to avoid safety issues and ensure availability of use. Data privacy is among the main issues. The fact is that chatbots usually work with patients' confidential data and it's vitally important to be compliant to standards like GDPR and HIPAA. Poor management of this data has potential to breach and therefore compromise patient trust. (Sharma et al., 2020).
Another concern is also the bias in the algorithms of AI. If this data set used to train these systems showed a bias or failed to reflect the diversity in different societies, then the chatbot may give incorrect diagnoses or health advice to some members of those groups, increasing the differences in healthcare. (Jiang et al., 2021).
Lastly, the lack of human empathy or emotions in AI-driven systems and bots can be concerning, specifically in mental health care, where emotional support is important. While a chatbot can mimic empathy, it can't replace real humans – and that may mean diminishing quality of care, particularly to people with complicated contexts. (Fitzpatrick et al., 2017).
This implies transparency, monitoring process all along and build the tools and deploy them through ethical frameworks.

Chapter 3: **Project Methodology**

In this chapter our empirical research approach is presented: how requirements were gathered, the system was developed, and how performance would be measured. We base our reasons and findings on secondary analysis (existing studies and datasets – no new surveys) and prototyping.

3.1 Requirement Gathering

Sources used to obtain requirements included: literature review, peer reviewed studies, current medical chatbot capability, and healthcare IT standards. We gathered them up into functional and non-functional requirements as below:

3.1.1 Functional Requirements:

- User Input and Chatbot Response: The chatbot should enable the users to type their symptoms in free text, read them and then provide a list of the possible conditions and so on, and recommended actions (self-care vs see doctor).
- Medical Report Interpretation: The chatbot will use user-provided medical data (e.g. lab values, imaging reports) and lay the findings out or flag the abnormalities in lay terms.
- Mental Health Conversation: The chatbot will conduct empathic conversations making use of CBT inspired prompts to measure mood and suggest coping strategies for anxiety or depression.
- Dialogue Management: Keep context from the current chat (symptom context from ongoing conversation) and be able to follow-up with questions.

3.1.2 Non-Functional Requirements

- Privacy & Security: The system must be HIPAA/GDPR compliant. Encryption should be ensured for all data of user's messages in transit and at rest. Outside of the query scope no PHI is stored in the LLM provider Open AI.
- Accuracy Targets: Some of the differential diagnosis accuracy's predictions (top 3) should be targeted toward meeting or surpassing ~ 70% according to literature references.
- Usability: User interface in simple and intuitive i.e. (React based UI) so that non-technical users can move around various symptoms and reports easily.
- Performance: Average response time for AI requests should be less than 8 seconds.
- Maintainability: Provide and well documented code to enable later an extension (e.g. upgrade to a newer LLM model).

3.2 System Design and Architecture

The frontend is developed using React, and an interactive chat interface and forms to enter symptoms or reports. To run the backend, Supabase, with a hosted PostgreSQL database (for storing user accounts, chat logs and saved contexts) and an authentication layer, is used. OpenAI's API acts as the AI engine: when the user queries (symptom description or report text), the React app will call a serverless (placed via Supabase Edge Functions / simple Node service) which will forward the query on to the OpenAI LLM and return the response. This separation makes it so that API keys are kept secret while also facilitating preprocessing (i.e. running NER) prior to model invocation.
Key Components of System Includes:

- User Interface: (React): Side Bar Navigation Menu, Chat panel, Buttons (for instance, a button “Submit Lab Report” to launch report flow). The UI tracks the conversation state making it feature both user messages and AI replies in threaded view.
- Backend Database: (Supabase/Postgres): Tables include users session history or chat history (chat transcripts reference users), and sessions (context from visit to visit made permanent). The database also includes metadata such as timestamps, flags.
- AI Integration (OpenAI API): We use the Chat Completion endpoint. For each feature:

General Chat or Symptom Diagnosis Feature:

model: 'gpt-4-turbo-preview',
temperature: 0.3,
system Prompt: 'You are a highly experienced medical doctor. Based on the following symptoms, provide a likely diagnosis and possible causes. List possible next steps. Keep it concise and professional.',

Report Analyzer Feature:

REPORT: {

model: 'gpt-4-turbo-preview',

temperature: 0.3,

systemPrompt: 'You are a medical AI trained to analyze patient reports. Given this report, summarize critical findings, flag abnormalities, and suggest next steps.',

},

Mental Health Support Feature:

MENTAL: {

model: 'gpt-4-turbo-preview',

temperature: 0.7,

systemPrompt: 'You are an empathetic AI therapist. Offer comforting and non-judgmental advice based on the user's input, encourage them, and provide mental wellness tips.',

},

- Security Layers: API keys and the credentials of the database are placed in Environment Variables. We never log raw user messages in the database, unless anonymized (for this project, we consider data to be purely illustrative).
- This design deploys contemporary cloud-based tools for swift development. The built in PostgreSQL and message history, session handling in Supabase meant we didn’t have to manage the server infrastructure manually. Using React, there is a responsive front-end experience.

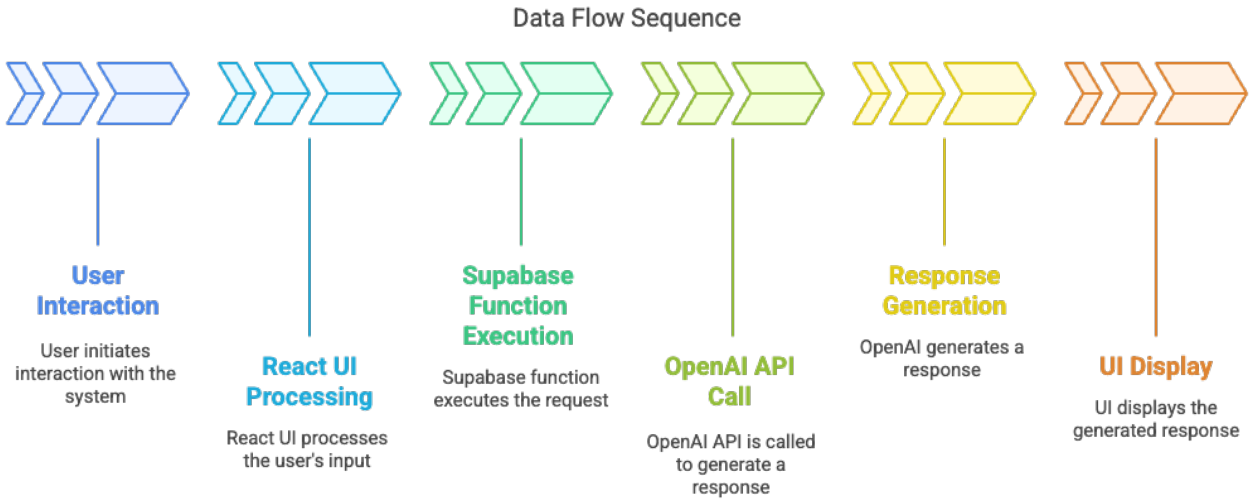


Figure 5. Data Flow of the System.

3.2.1 Block Diagram:

Figure 6. This Block Diagram Show's END to end System Architecture of the chatbot.

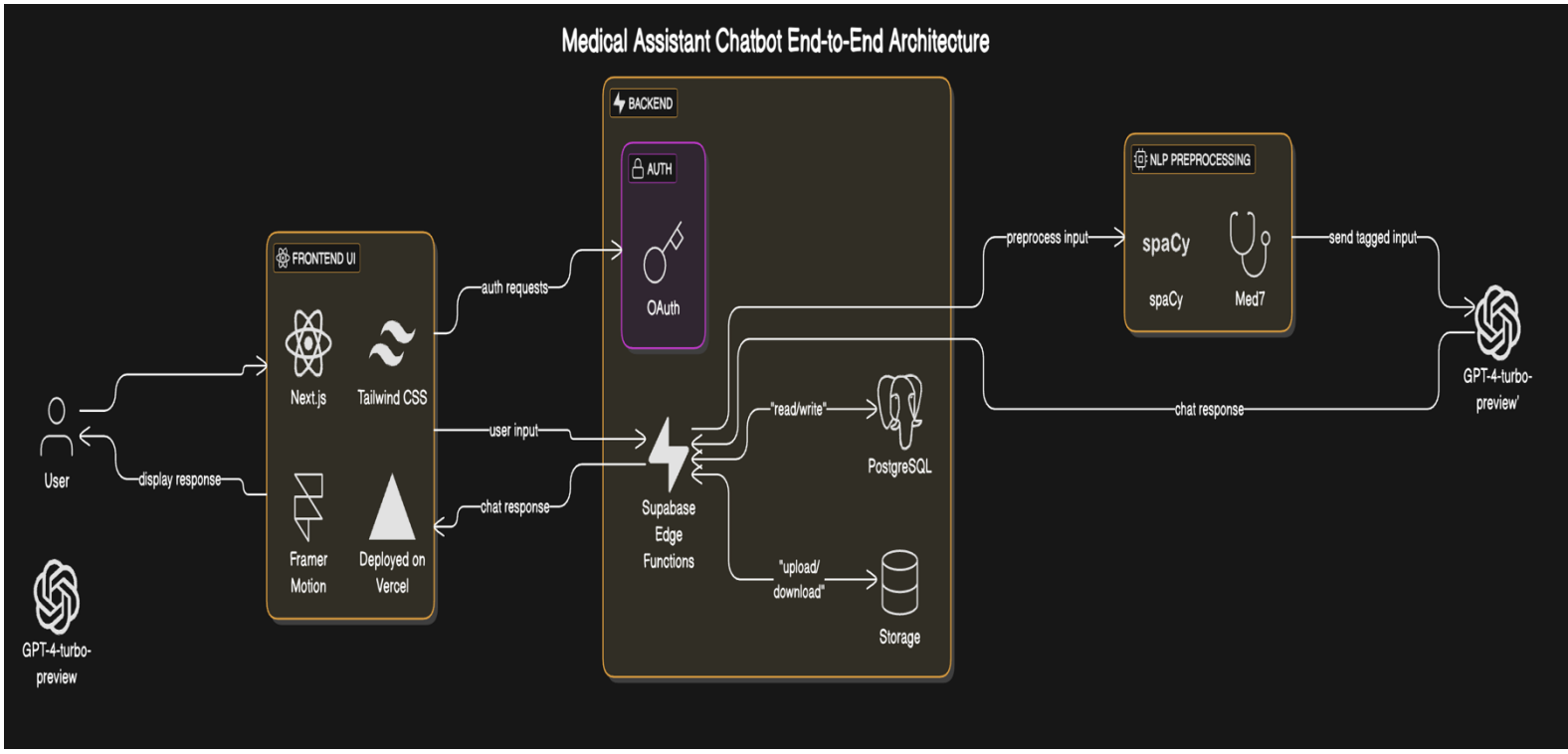
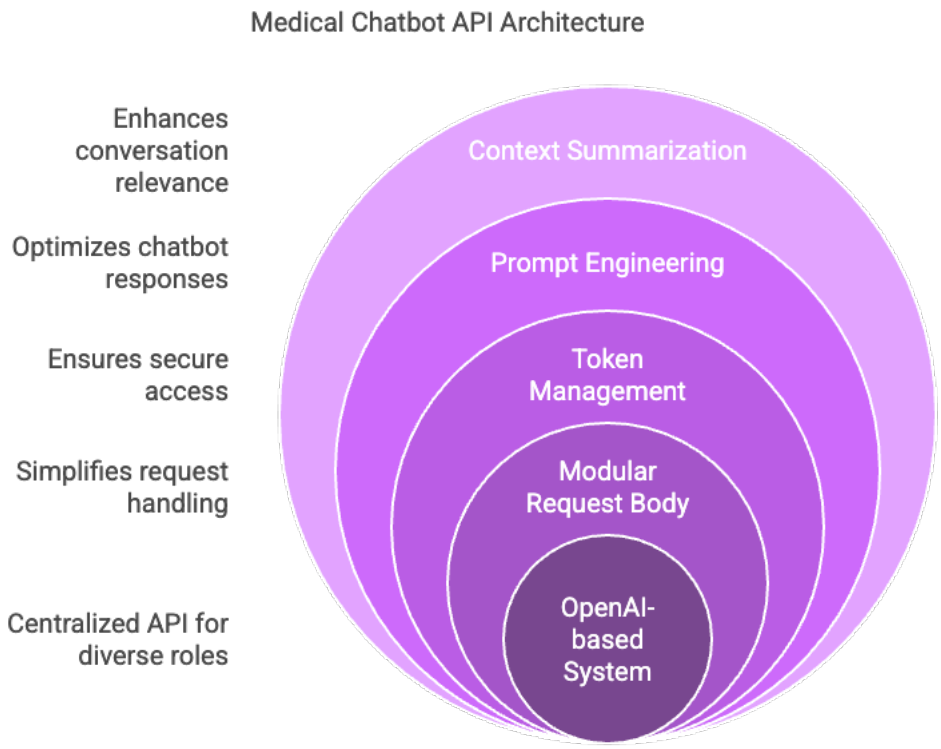


Figure 7. API ARCHITECHTURE OF THE CHATBOT



3.3 Technology Stack and Technologies

This section describes the full technology stack and architecture that will be put in use when developing medical assistant chatbot. The focus of the design is on developing a demo scalable, high performance and simple to use application that uses modern web technologies and integration with AI.

3.3.1 Frontend Stack

3.3.2 Framework & Core Technologies

- Next.js 14 will serve as the front-end framework of the first choice with server-side rendering (SSR), static site generation, and API routes.
- React 18 will be used to empower component-based architecture as well as an efficient state management.
- TypeScript will be used to give us type safety which makes our code more maintainable and runtime errors are reduced.

3.3.3 Styling & Theming

- Utility-first CSS styling through Tailwind CSS for fast UI development, and responsive.

- Shadcn/ui based on Radix UI offered accessible and customisable UI components.
- next-themes provided toggleable dark/light mode option for personal taste.

3.3.4 UI Enhancements

- Radix UI primitives will be utilized for the construction of interactive components of accessible nature.
- Framer Motion will be implemented in the UI for smooth animations and transition in the UI.

.

3.3.5 Backend Stack

3.3.6 Backend Infrastructure:

- The API of the backend will be created through Next.js API routes, which enabled easy connection to the frontend.
- Node.js will be used as platform for server-side code.
- TypeScript will be used to contributed to the consistency and safety at all backend modules.

3.3.7 Database & Authentication

Supabase will be selected as a backend as a service platform providing:

- PostgreSQL persistent storage database.
- Supabase client libraries for the management of databases.
- Authentication features though will not be implemented for this demo version; the stack is still compatible with Supabase Auth for expansion goals.

3.4 Datasets and Models Analysis

We rely on OpenAI's training for medical knowledge because we use their proprietary models. No new model was trained by us. We however analysed existing medical datasets to gauge performance expectations. We examined such public datasets of symptom-checker evaluation (e.g. those used by Semigran et al. and Budrionis) and medical QA (MedQA, UWorld question banks). To conclude our approach is to "stress-test" it with known common medical cases and compare the answers with correct answers.
Front EN

3.5 Deployment using GitHub and Vercel

The chatbot frontend will be developed using React (Next.js) and will be deployed using Vercel for fast, serverless deployment. The source code will be under control using GitHub, which will be able to stage continuous integration. Every push to the main branch will automatically be built and deployed by Vercel with the updated version of the app. The backend (Supabase) was connected using environment variables, and envied routes were facilitated using Vercel's dashboard. This

approach secured quick deployment, scalability and low configurations which were in concurrence to DevOps' modern methodology.

3.6 Ethical and Legal Consideration

The chatbot works with sensitive health data, therefore privacy and security were key issues. Despite that no user authentication is required, Supabase guarantees secure data handling, and OpenAI API requests are performed under GDPR standards. For the sake of not leading users astray, the chat bot says clearly that it is not a replacement for professional care. Bias is reduced by prompt engineering, but developers are held accountable for observing outputs. As the prototype project, there was consideration for legal compliance with frameworks such as GDPR and UK Data Protection Act. Strict compliance with medical data regulation such as HIPAA or NHS regulations will be required upon future deployment.

Chapter 4: **Results / Findings / Outcomes**

4.1 Functional Implementation & Source Code

We will only be including some key code snippets that are key components of the chatbot and explaining the code snippets. It is complex to upload whole codebase into a word file, Whole code would be uploaded on a GitHub repository and the link to it would be included in the Appendix C so it can be forked or cloned for review.

4.1.1 Code Snippets

Main Chat API End Point (Core functionality):

```
try {
      const response = await sendChatMessage({
        type: activeSession.type,
        message: inputValue,
        fileUrl
      })

      const assistantMessage: Message = {
        id: Date.now().toString(),
        content: response.message,
        sender: 'assistant',
        timestamp: new Date()
      }

      setMessages(prev => [...prev, assistantMessage])

      // Update the active session and chat sessions
      const updatedSession = {
        ...activeSession,
        lastMessage: response.message,
        messages: [...activeSession.messages, newMessage,
assistantMessage]
      }

      setActiveSession(updatedSession)
      setChatSessions(prev =>
        prev.map(session =>
          session.id === activeSession.id ? updatedSession : session
        )
      )
    } catch (error) {
      toast({
        title: 'Error',
        description: 'Failed to get response. Please try again.',
        variant: 'destructive'
      })
    } finally {
      setIsTyping(false)
    }
```

Chat Features functionality with System Prompts:

```
export const CHAT_MODELS: Record<'SYMPTOM' | 'REPORT' | 'MENTAL',
ChatModelConfig> = {
  SYMPTOM: {
```

```
    model: 'gpt-4-turbo-preview',
    temperature: 0.3,
    systemPrompt: 'You are a highly experienced medical doctor. Based on
the following symptoms, provide a likely diagnosis and possible causes.
List possible next steps. Keep it concise and professional.',
  },
  REPORT: {
    model: 'gpt-4-turbo-preview',
    temperature: 0.3,
    systemPrompt: 'You are a medical AI trained to analyze patient
reports. Given this report, summarize critical findings, flag
abnormalities, and suggest next steps.',
  },
  MENTAL: {
    model: 'gpt-4-turbo-preview',
    temperature: 0.7,
    systemPrompt: 'You are an empathetic AI therapist. Offer comforting
and non-judgmental advice based on the user\'s input, encourage them, and
provide mental wellness tips.',
  },
} as const;
```

Chat Handlers Snippet:

```
import { createClient } from '@/lib/supabase/client'
import { ChatRequest, ChatResponse } from '@/types/chat'

export async function sendChatMessage(request: ChatRequest):
Promise<ChatResponse> {
  try {
    const response = await fetch('/api/v1/chat', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(request),
    })

    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`)
    }

    const data = await response.json()
    return data as ChatResponse
  } catch (error) {
    console.error('Error sending chat message:', error)
    return {
      message: 'Sorry, there was an error processing your request. Please
try again.',
      error: error instanceof Error ? error.message : 'Unknown error',
    }
  }
}
```

Complete Chat handlers

```
import { createClient } from '@/lib/supabase/client'
import { ChatRequest, ChatResponse } from '@/types/chat'

export async function sendChatMessage(request: ChatRequest):
Promise<ChatResponse> {
  try {
    const response = await fetch('/api/v1/chat', {
      method: 'POST',
```

```
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(request),
    })

    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`)
    }

    const data = await response.json()
    return data as ChatResponse
  } catch (error) {
    console.error('Error sending chat message:', error)
    return {
      message: 'Sorry, there was an error processing your request. Please
try again.',
      error: error instanceof Error ? error.message : 'Unknown error',
    }
  }
}
```

Report Analyzer Component

```
const renderReportInput = () => (
    <div className="p-4 border-t border-gray-200 dark:border-gray-700 bg-
white/90 dark:bg-gray-800/90 backdrop-blur-sm">
      {selectedFile && (
        <motion.div
          initial={{ opacity: 0, y: 10 }}
          animate={{ opacity: 1, y: 0 }}
          className="mb-2 flex items-center justify-between p-3 bg-blue-
50 dark:bg-blue-900/30 rounded"
        >
          <div className="flex items-center gap-2">
            <File size={18} className="text-blue-500" />
            <span className="text-sm truncate max-w-
[200px]">{selectedFile.name}</span>
          </div>
          <Button variant="ghost" size="icon" className="h-6 w-6"
onClick={() => setSelectedFile(null)}>
            <X size={14} />
          </Button>
        </motion.div>
      )}
      <div className="flex gap-2">
        <Button
          variant="outline"
          size="icon"
          onClick={() => fileInputRef.current?.click()}
          className="shrink-0 hover:bg-blue-100 dark:hover:bg-blue-900/30
transition-colors"
        >
          <input type="file" ref={fileInputRef}
onChange={handleFileSelect} className="hidden" accept=".pdf,image/*" />
          <File size={18} className="text-blue-500" />
        </Button>
        <Input
          placeholder="Ask about your medical report..."
          value={inputValue}
          onChange={(e) => setInputValue(e.target.value)}
          onKeyDown={(e) => e.key === "Enter" && handleSendMessage()}
```

```
          className="bg-white dark:bg-gray-700 border-gray-300
dark:border-gray-600 focus-visible:ring-blue-500"
        />
        <Button
          onClick={handleSendMessage}
          disabled={(inputValue.trim() === "" && !selectedFile) ||
isTyping}
          className="shrink-0 bg-gradient-to-r from-blue-500 to-blue-600
hover:from-blue-600 hover:to-blue-700 transition-all duration-300"
        >
          <Send size={18} />
        </Button>
      </div>
    </div>
  )
```

Chat Feature Selection UI

```
<AnimatePresence>
    {showNewChatMenu && (
        <motion.div
        initial={{ height: 0, opacity: 0 }}
        animate={{ height: "auto", opacity: 1 }}
        exit={{ height: 0, opacity: 0 }}
        transition={{ duration: 0.2 }}
        className="overflow-hidden mt-2"
        >
        <div className="space-y-1">
            <Button variant="ghost" className="w-full justify-start gap-
2" onClick={() => createNewSession('symptom-checker')}>
            <MessageCircle size={16} className="text-blue-500" />
            <span>General Chat</span>
            </Button>
            <Button variant="ghost" className="w-full justify-start gap-
2" onClick={() => createNewSession('report-analyzer')}>
            <ClipboardList size={16} className="text-green-500" />
            <span>Report Analyzer</span>
            </Button>
            <Button variant="ghost" className="w-full justify-start gap-
2" onClick={() => createNewSession('mental-health')}>
            <Heart size={16} className="text-red-500" />
            <span>Mental Health</span>
            </Button>
        </div>
        </motion.div>
    )}
</AnimatePresence>
```

API/v1/chat END Point

```
import { NextResponse } from 'next/server';
import OpenAI from 'openai';
import { createClient } from '@/lib/supabase/server';
import { ChatRequest, ChatResponse } from '@/types/chat';

// Initialize OpenAI client
const openai = new OpenAI({
  apiKey: process.env.OPENAI_API_KEY,
});

// Map frontend chat types to database enum
const chatTypeMap = {
  'symptom-checker': 'symptom',
  'report-analyzer': 'report',
```

```
  'mental-health': 'mental'
} as const;

// Function to convert image URL to base64
async function getImageAsBase64(supabase: any, filePath: string):
Promise<string> {
  try {
    // Download the file from Supabase storage
    const { data: fileData, error: downloadError } = await supabase
      .storage
      .from('medical-reports')
      .download(filePath);

    if (downloadError) {
      throw new Error(`Failed to download image:
${downloadError.message}`);
    }

    // Convert blob to base64
    const buffer = Buffer.from(await fileData.arrayBuffer());
    return `data:${fileData.type};base64,${buffer.toString('base64')}`;
  } catch (error) {
    console.error('Error converting image to base64:', error);
    throw error;
  }
}

export async function POST(request: Request) {
  try {
    const body: ChatRequest = await request.json();
    const { type, message, fileUrl } = body;

    // Get file info if fileUrl is provided
    let fileInfo = null;
    if (fileUrl) {
      const supabase = createClient();
      const fileId = fileUrl.split('/').pop(); // Get ID from URL

      // Get file metadata from database
      const { data: fileData, error: fileError } = await supabase
        .from('uploads')
        .select('file_url, file_name, file_type')
        .eq('id', fileId)
        .single();

      if (fileError) {
        console.error('Error fetching file metadata:', fileError);
      } else if (fileData) {
        fileInfo = {
          path: fileData.file_url,
          name: fileData.file_name,
          type: fileData.file_type
        };
      }
    }

    // Prepare system message based on chat type
    const systemMessages = {
      'symptom-checker': 'You are a medical AI assistant helping with
symptom checking. Be thorough but cautious in your assessment.',
      'report-analyzer': 'You are a medical AI assistant analyzing
medical reports and test results. Focus on explaining the results clearly
```

```
and professionally. When analyzing reports, highlight any abnormal values
and explain their significance.',
      'mental-health': 'You are a supportive mental health AI assistant.
Be empathetic and professional.',
    };

    // Prepare messages array
    const messages: OpenAI.Chat.Completions.ChatCompletionMessageParam[]
= [
      { role: 'system', content: systemMessages[type] }
    ];

    // Add user message with file if present
    if (fileInfo?.type.includes('image/')) {
      // Get image as base64
      const base64Image = await getImageAsBase64(createClient(),
fileInfo.path);

      messages.push({
        role: 'user',
        content: [
          { type: 'text', text: message },
          {
            type: 'image_url',
            image_url: {
              url: base64Image.startsWith('data:') ? base64Image :
`data:${fileInfo.type};base64,${base64Image}`,
              detail: 'high'
            }
          }
        ]
      });
    } else {
      messages.push({
        role: 'user',
        content: fileInfo
          ? `${message}\n\nFile Name: ${fileInfo.name}`
          : message
      });
    }

    // Get OpenAI response
    const completion = await openai.chat.completions.create({
      model: fileInfo?.type.includes('image/') ? 'gpt-4.1-mini' : 'gpt-4-
turbo-preview',
      messages,
      temperature: 0.7,
      max_tokens: 4096
    });

    const aiResponse = completion.choices[0].message.content || 'Sorry, I
could not generate a response.';

    // Store the conversation in Supabase
    try {
      const supabase = createClient();

      // Insert message
      const { error: messageError } = await supabase
        .from('messages')
        .insert({
          message_text: message,
          response_text: aiResponse,
```

```
          chat_type: chatTypeMap[type]
        });

      if (messageError) {
        console.error('Error storing message:', messageError);
      }
    } catch (dbError) {
      console.error('Database error:', dbError);
    }

    const response: ChatResponse = {
      message: aiResponse,
    };

    return NextResponse.json(response);
  } catch (error) {
    console.error('Error in chat endpoint:', error);
    return NextResponse.json(
      { error: 'Failed to process chat request' },
      { status: 500 }
    );
  }
}
```

4.1.2 Explanation

- Core Chat Functionality: The key chat handling logic and message handling logic
- AI Model Configuration: Variety of models and prompts for every medical feature.
- API Integration: The chat message handling system

4.1.3 UI Components

- Report analyser having file upload capacity.
- Interface for feature selection by various kinds of medical support.
- All these components are used to produce a full medical chatbot system that can:
- Process and analyse symptoms
- Analyse medical reports
- Provide mental health support
- Handle file uploads
- Maintain chat history
- Offer a safe and user-friendly interface.

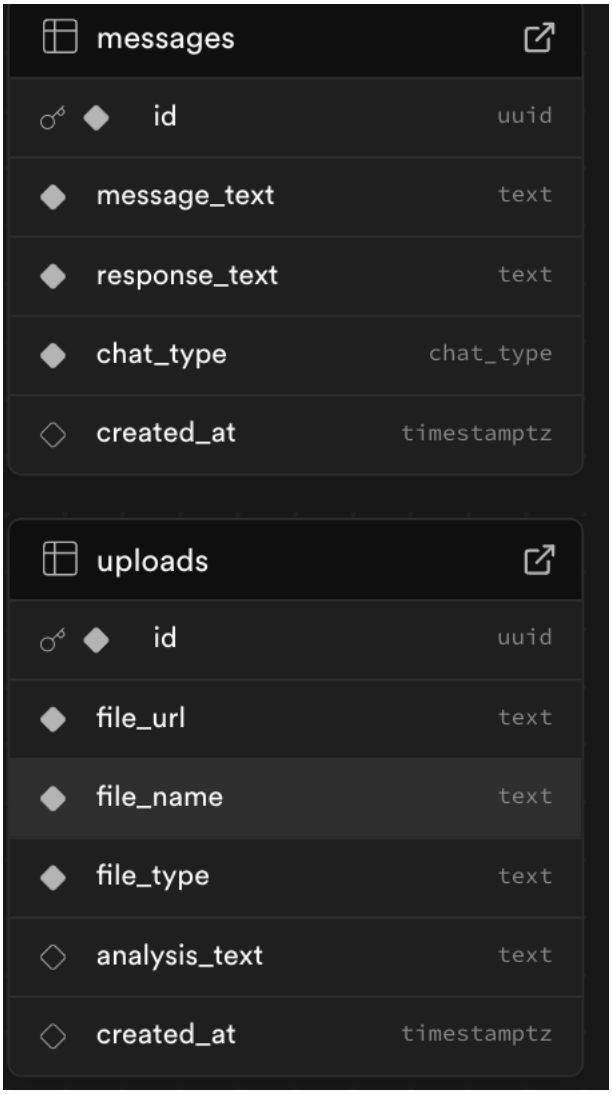
These components combine to form an all-encompassing medical chatbot system which is able to:.

- Process and analyse symptoms
- Analyse medical reports
- Provide mental health support
- Handle file uploads
- Maintain chat history
- An interface that is secure and easy to use must be offered

4.2 Back END Supabase Implementation

I will be uploading SQL code snippets and database schema visualizer screenshot to show implementation of our backend setup through supa base.

Figure 8. Data base Schema Visualizer



4.2.1 SQL Code Snippets

4.2.2 Table Uploads Snippet

```sql
-- Drop existing table if it exists
DROP TABLE IF EXISTS uploads;

-- Recreate the uploads table with correct structure
CREATE TABLE uploads (
    id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    file_url TEXT NOT NULL,
    file_name TEXT NOT NULL,
    file_type TEXT NOT NULL,
    analysis_text TEXT,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT TIMEZONE('utc', NOW())
);

-- Enable RLS
ALTER TABLE uploads ENABLE ROW LEVEL SECURITY;
```

```
-- Create RLS policies
CREATE POLICY "Public read access to uploads"
    ON uploads FOR SELECT
    USING (true);

CREATE POLICY "Public insert access to uploads"
    ON uploads FOR INSERT
    WITH CHECK (true);
```

4.2.3 Upload Policy for Storage Bucket

```
   CREATE POLICY "Allow public uploads"
   ON storage.objects FOR INSERT
   TO public
   WITH CHECK (bucket_id = 'medical-reports');
```

4.2.4 Public Access Policy for Storage Bucket

```
   CREATE POLICY "Allow public downloads"
   ON storage.objects FOR SELECT
   TO public
   USING (bucket_id = 'medical-reports');
```

4.2.5 New Chat Message, Chat Options and RLS policies

```
-- Drop existing tables and types if they exist
DROP TABLE IF EXISTS messages;
DROP TABLE IF EXISTS uploads;
DROP TYPE IF EXISTS chat_type;

-- Create chat_type enum
CREATE TYPE chat_type AS ENUM ('symptom', 'report', 'mental');

-- Create messages table
CREATE TABLE messages (
    id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    message_text TEXT NOT NULL,
    response_text TEXT NOT NULL,
    chat_type chat_type NOT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT TIMEZONE('utc', NOW())
);

-- Create uploads table
CREATE TABLE uploads (
    id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    file_url TEXT NOT NULL,
    file_name TEXT NOT NULL,
    file_type TEXT NOT NULL,
    analysis_text TEXT,
```

```
    created_at TIMESTAMP WITH TIME ZONE DEFAULT TIMEZONE('utc', NOW())
);

-- Enable Row Level Security (RLS)
ALTER TABLE messages ENABLE ROW LEVEL SECURITY;
ALTER TABLE uploads ENABLE ROW LEVEL SECURITY;

-- Create public access policies for demo
CREATE POLICY "Public read access to messages"
    ON messages FOR SELECT
    USING (true);

CREATE POLICY "Public insert access to messages"
    ON messages FOR INSERT
    WITH CHECK (true);

CREATE POLICY "Public read access to uploads"
    ON uploads FOR SELECT
    USING (true);

CREATE POLICY "Public insert access to uploads"
    ON uploads FOR INSERT
    WITH CHECK (true);
```

4.2.6 Explanation

Messages: Stores chatbot conversation history containing the messages content, the AI's reply, chat type (symptom/report), and date/time of the conversation.

uploads: Manages uploaded-by-users files, tracks URL, name, type, analysis if done, and upload time.
The next SQL code does the following:

1. Drops and recreates the uploads Table: This clears slate by deleting the existing uploads table and then creating a new one, with a specified structure, such as ID (primary key), file URL, name, type, analysis text and creation timestamp.

2. Removes targets table from the security filter: Enables Row Level Security (RLS) on uploads:| This enacts fine grain access control for uploads row level.

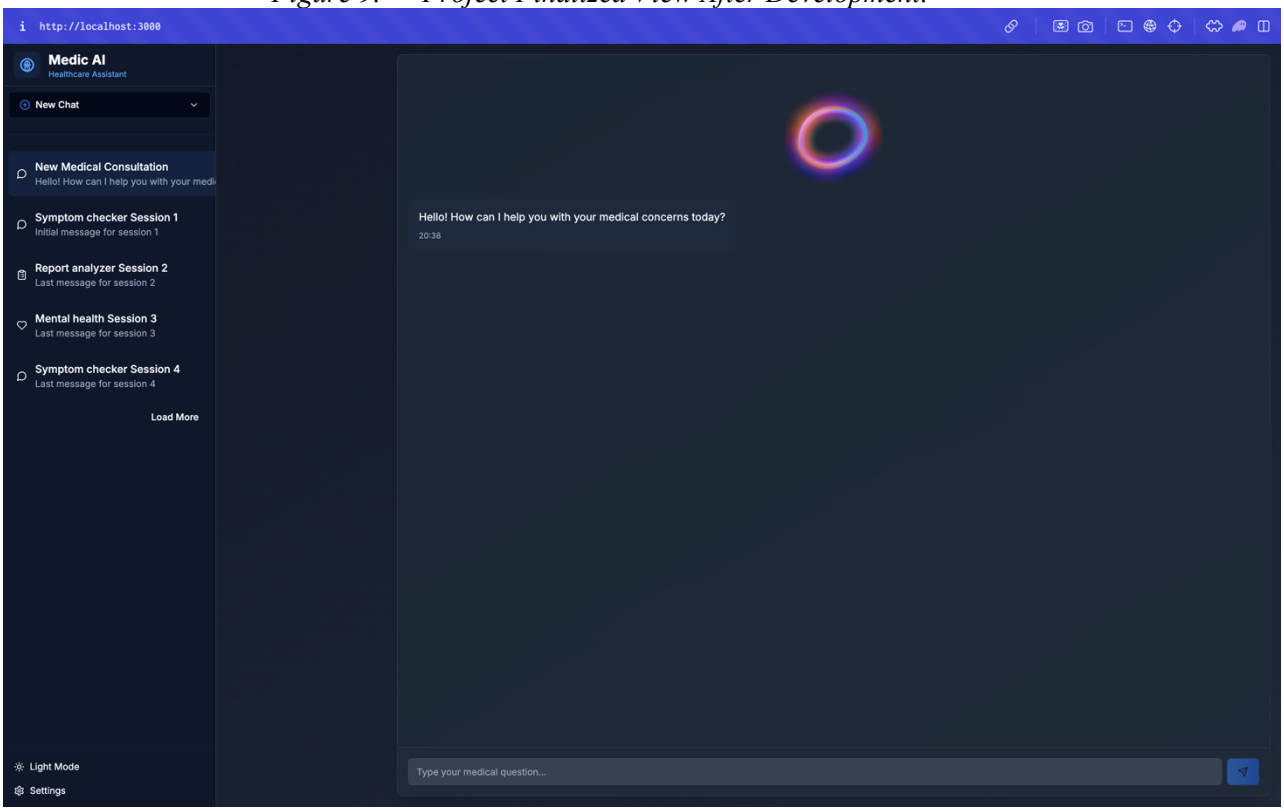
3. Generates Public Read and Insert policies for the uploads; These early RLS policies allow all users to read any row in the uploads table, insert new rows.

4. Medica-reports bucket storage policies. For medical-reports bucket. Such policies specify access rules for the medical-reports storage bucket, at which normal users can upload and download any files therein.
In essence, such snippets describe the database model for the processing of file uploads in the medical chatbot application in question and establish initial publicly available policies for the reads/writes from/to the uploads table in the database as well as the file operations in the provided storage bucket. To be suitable for a production medical application, these public policies would

need to be replaced by much more restrictive, role-based access controls to achieve data security and privacy.

4.3 Implementation Outcome After Front-end and Back-end Development

Figure 9. Project Finalized View After Development.



This image shows that our project build, was successfully when we executed npm build command and npm run dev. The project initialized successfully, I did some initial tests to check functionality of the system, and its feature, and it looks functional.

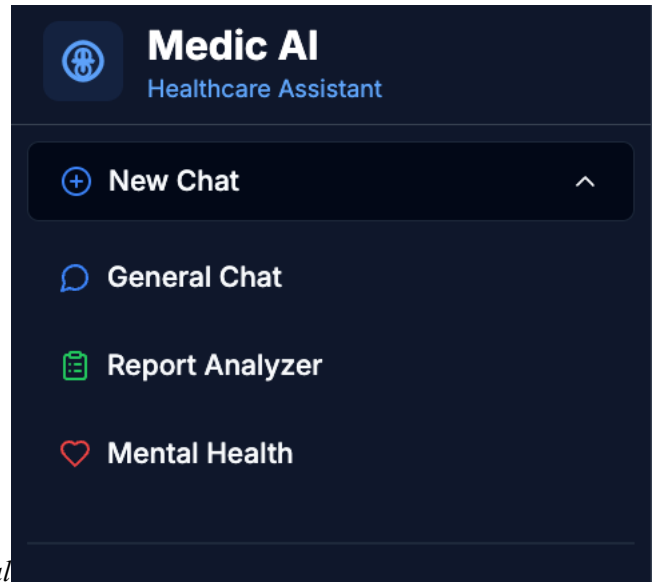


Figure 10. New Chat Options Features Visual

Figure 11. This image shows that the chat containers are opening when we select report analyser options from new chat options.

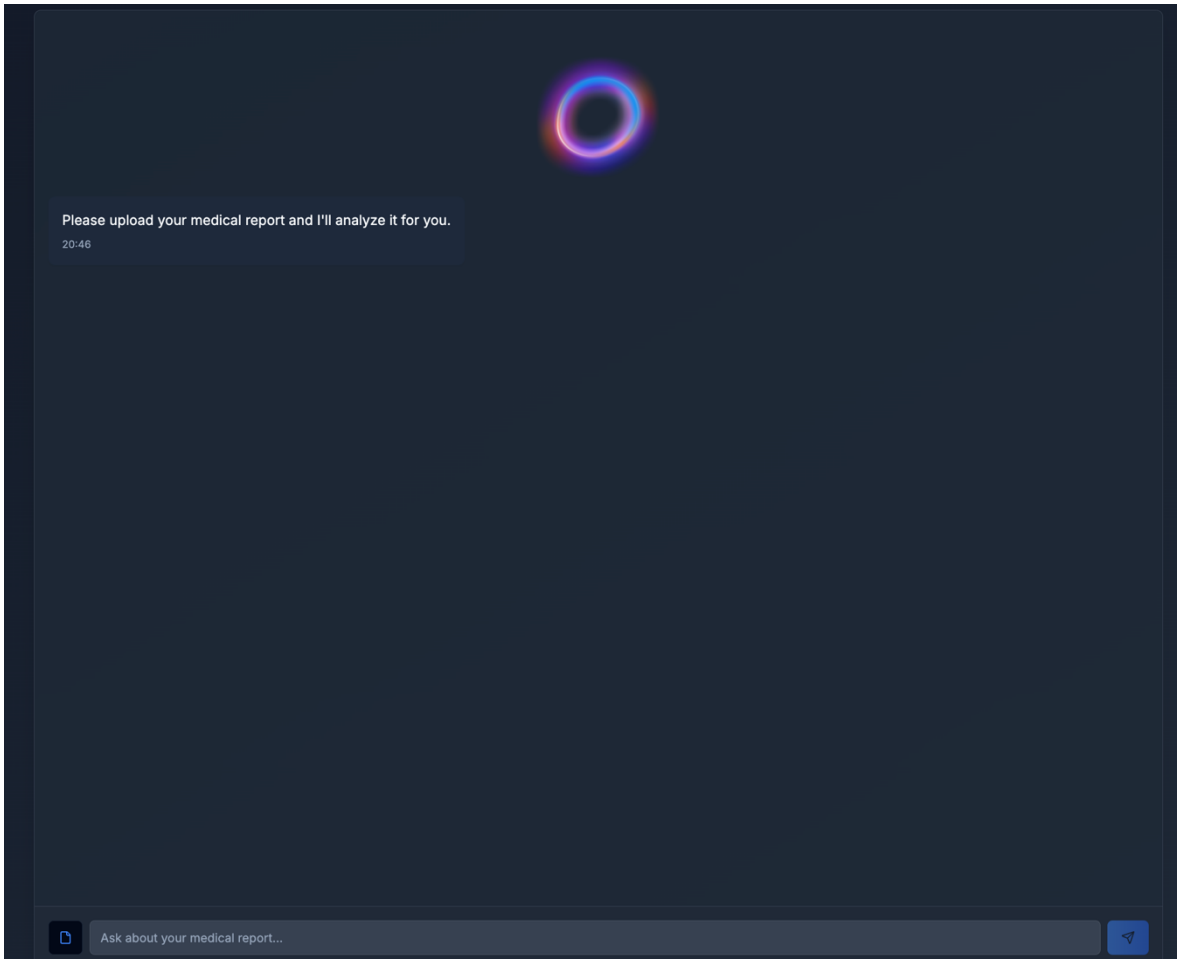


Figure 12. The last Chat options is also opening

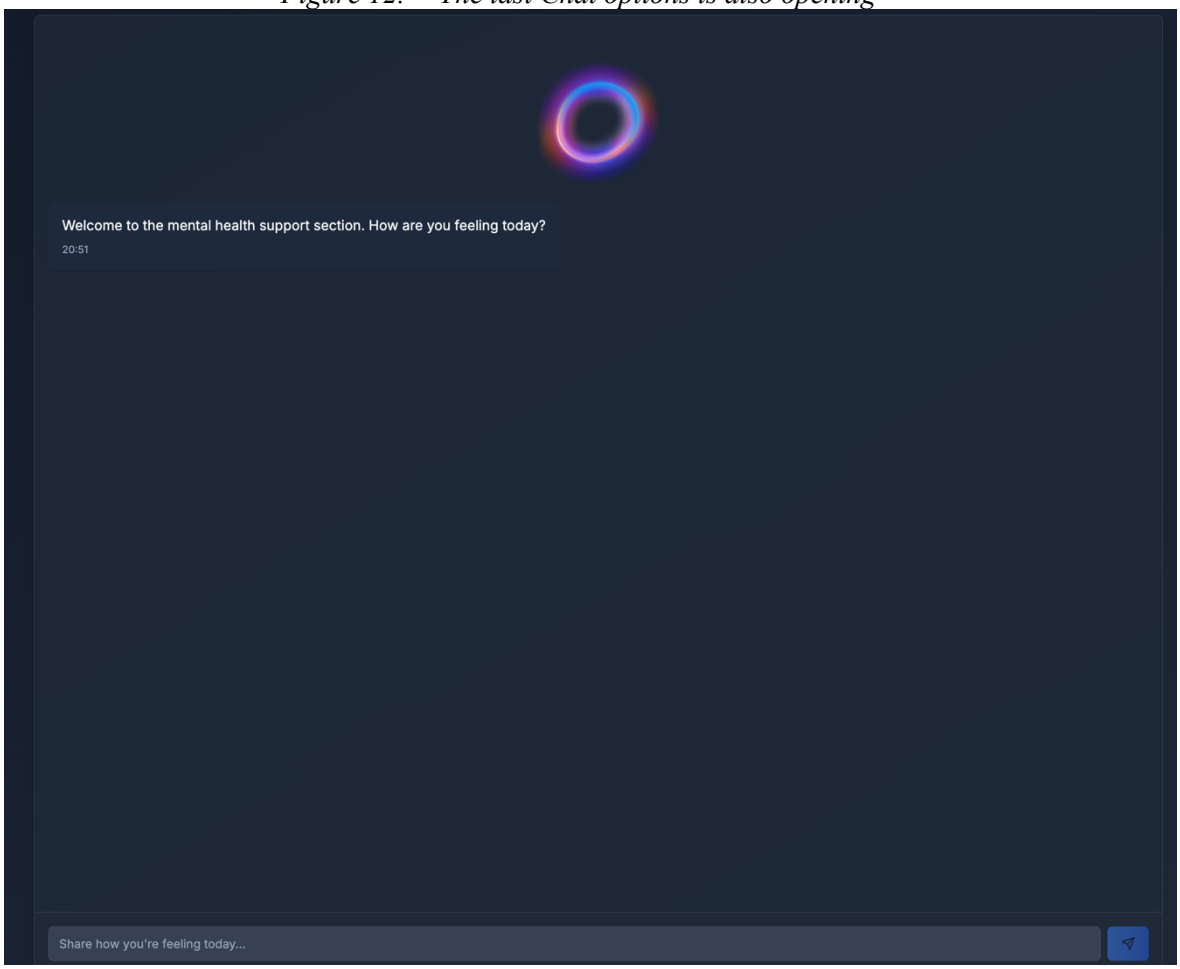
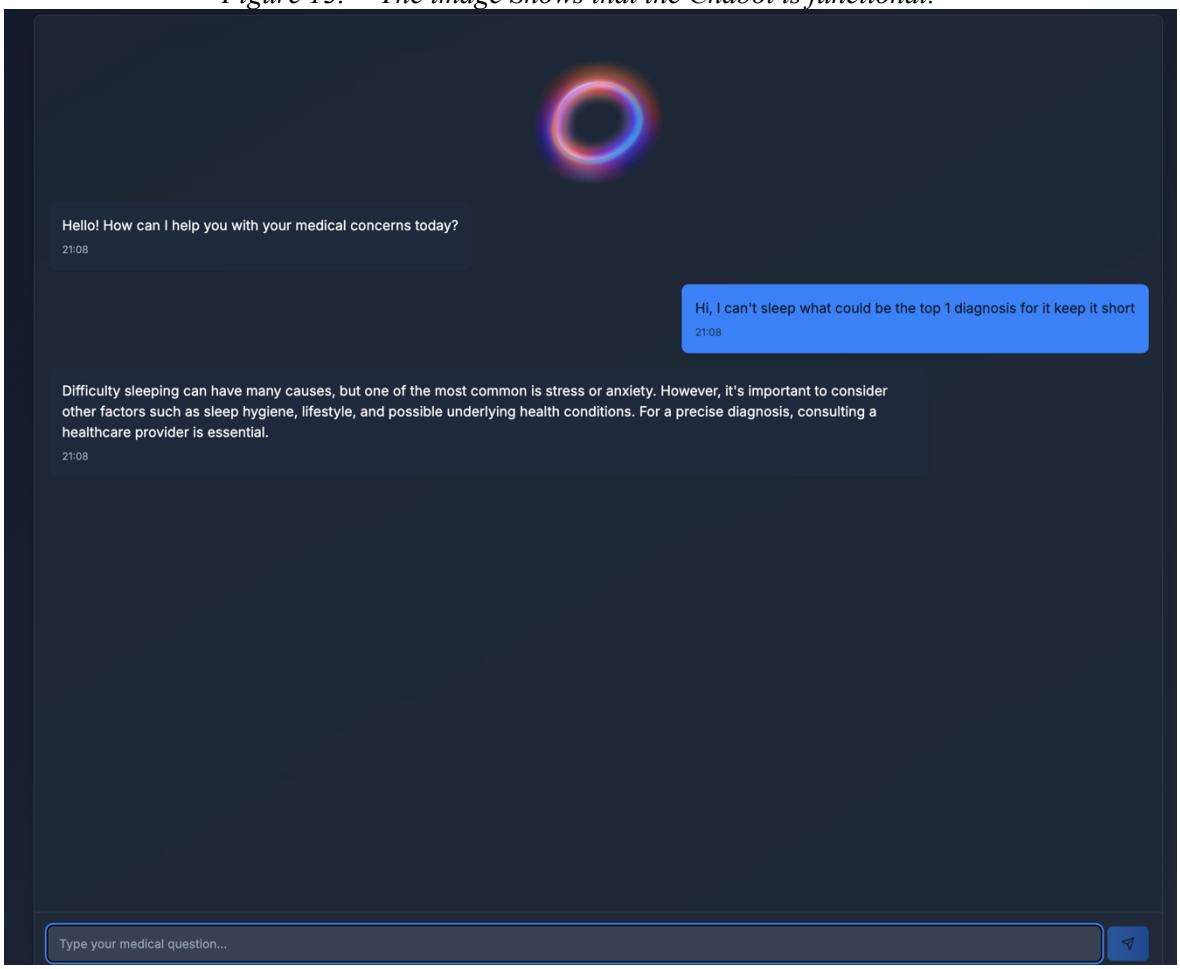


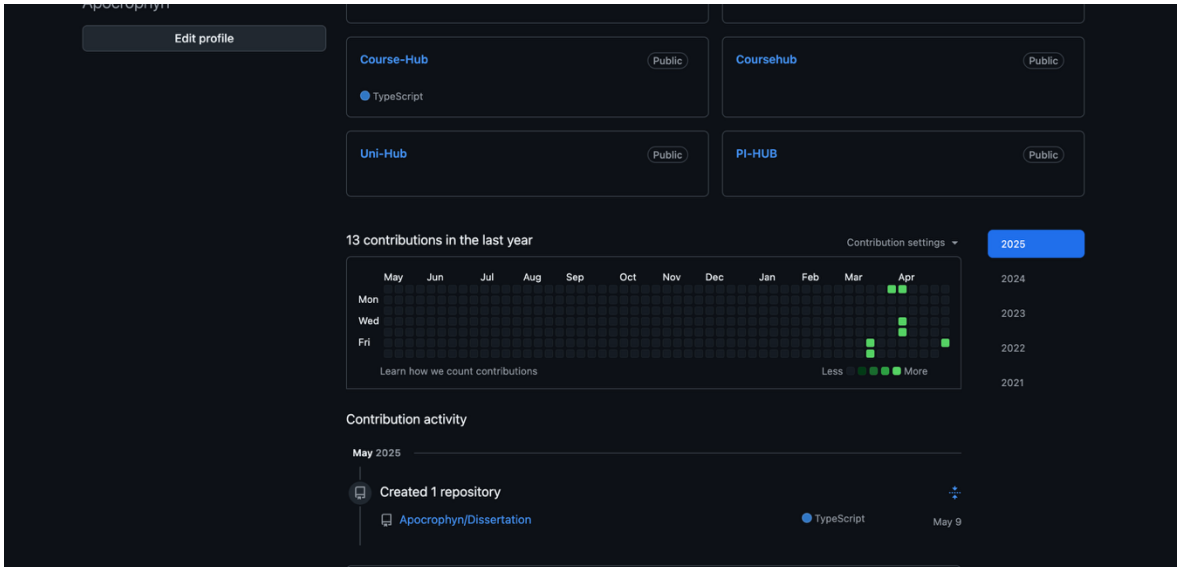
Figure 13. The image Shows that the Chabot is functional.



All these images show that the front-end components are visible and loading successfully and chat functionality is also operational.

4.4 Creating GitHub Repository and pushing code to the repository

Figure 14. Git Repository.



In this screenshot we can see we have successfully created a repository on GitHub and it is a key aspect of our deployment as it will help us deploy the web app on Vercel using this repository and it's much easier to manage the project through GitHub we can directly commit and push changes to the project and Vercel will redeploy the webpage and add those changes automatically when needed which is why it is the one of the most optimal way of modern web development.

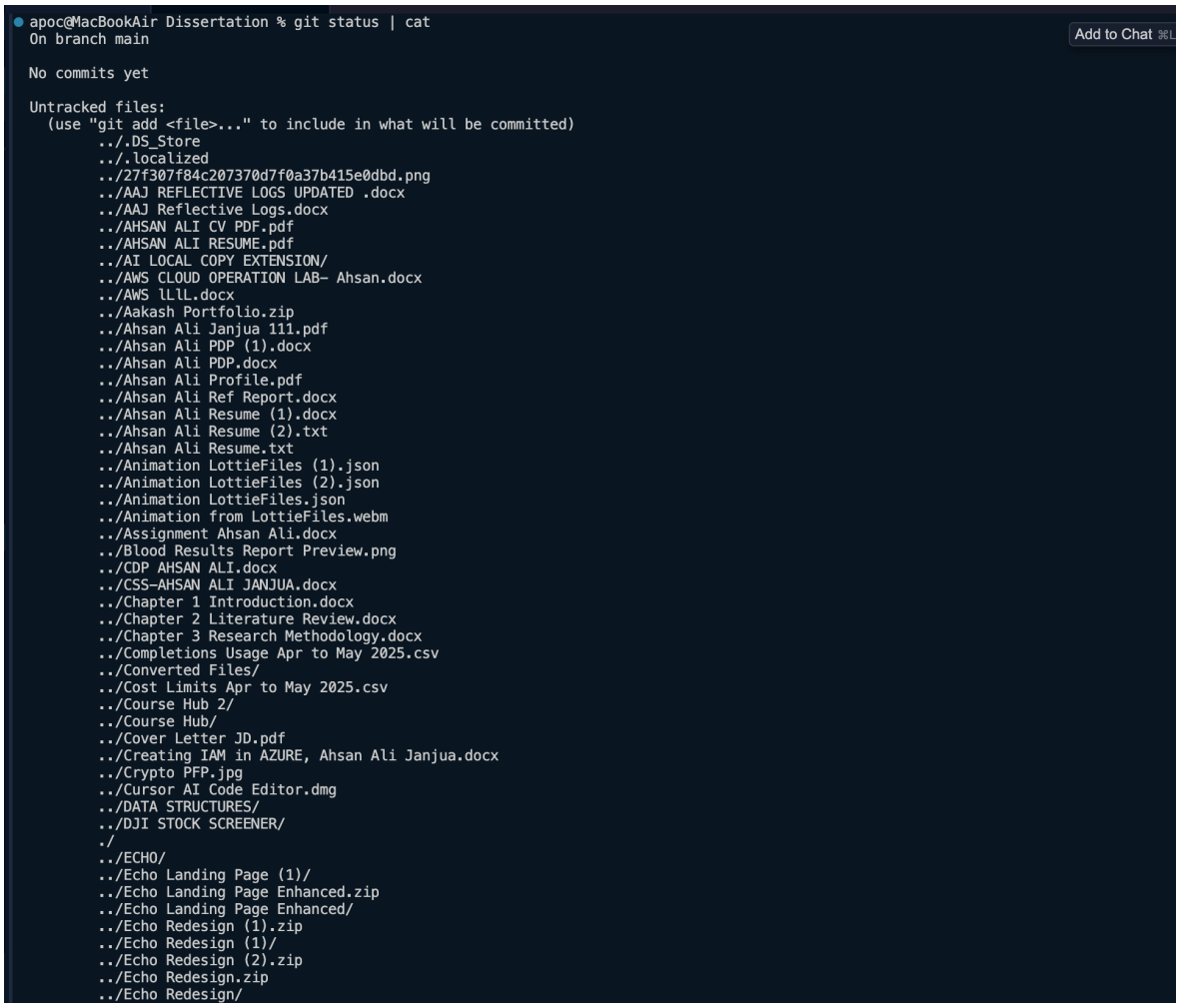


Figure 15. Git Commands in Visual Studio Terminal

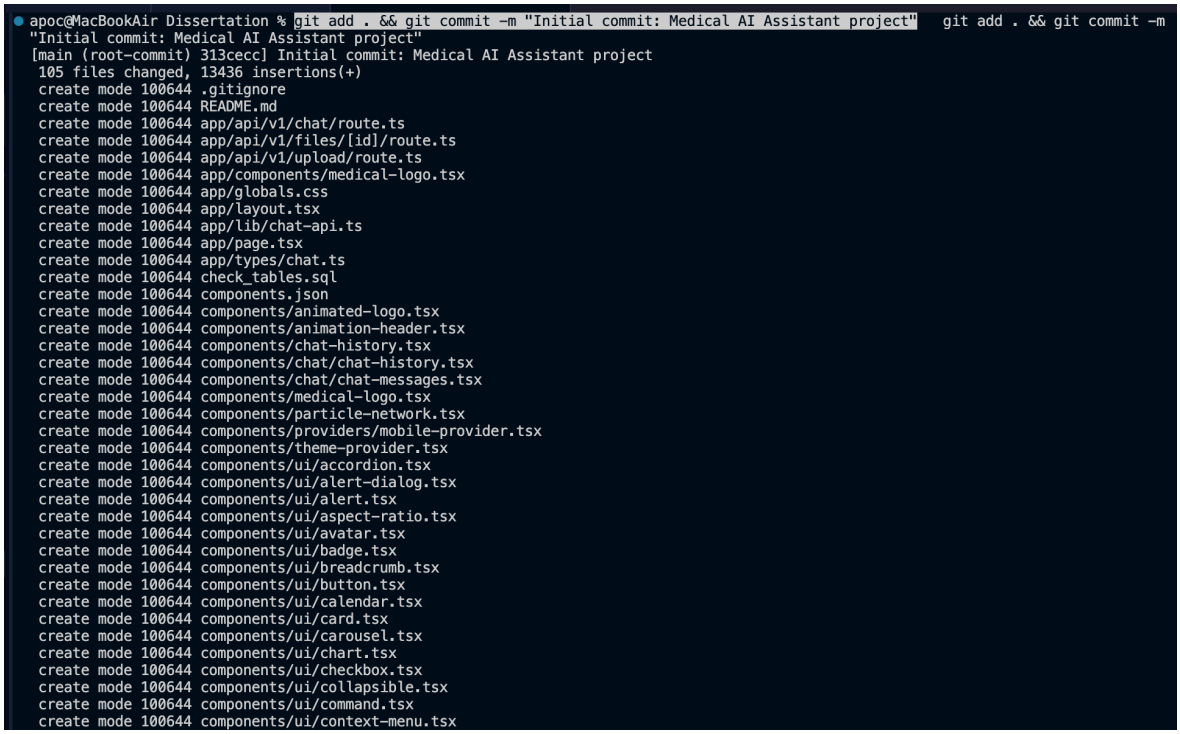


Figure 16. Git Initial Commit

In in this figure, we have used git add && git commit-m command Initializes a new Git repository in the current directory. This also creates a hidden .git file in the repository which tracks changes.

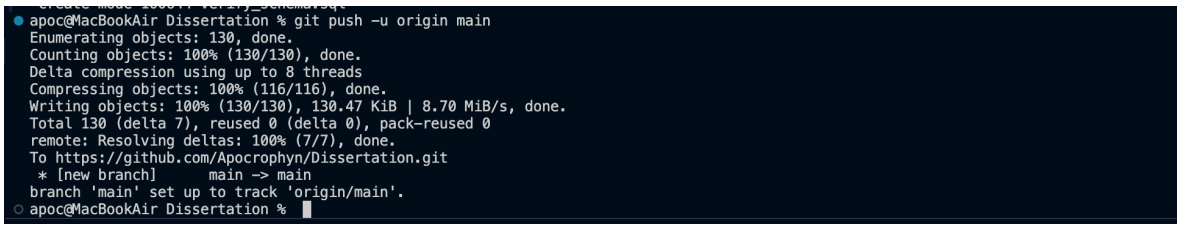


Figure 17. Git push -u origin main command

In this figure we use the git push and commit to main command which uploads our commits to GitHub
The -u installs tracking between local and remote ones.
origin main describes that the content should be pushed towards the main branch of the remote repository.

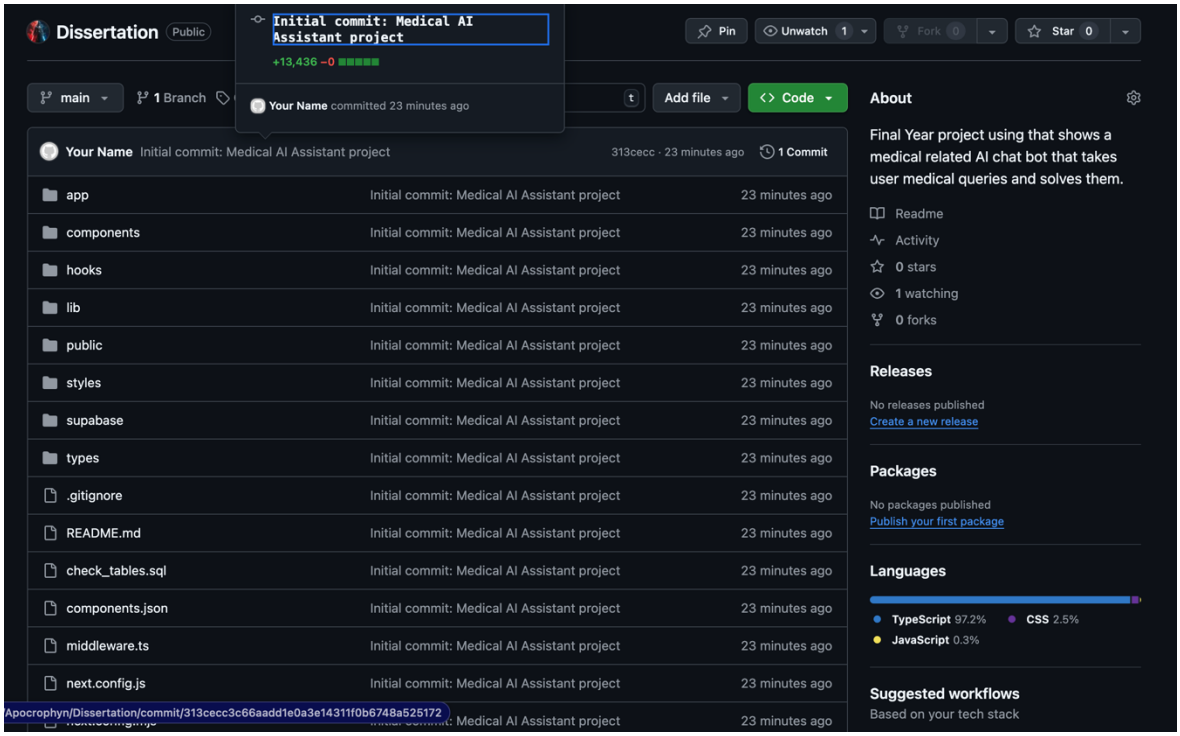


Figure 18. Git Repository showing all files and initial commit.

In this figure we can see that our files were successfully uploaded onto our git hub repository, and we can even see our initial commit onto our Git repository showing everything worked as we intended.

4.5 Vercel Deployment

A key step before deploying a web app on Vercel is to create a Git repository push all the code to the repository then use that git repository to deploy the web app. This makes it easier to commit changes directly to the Git Repository which will automatically also be applied to the deployed server on Vercel and updated our changes automatically. Vercel is good for deploying Next.js Web Apps.

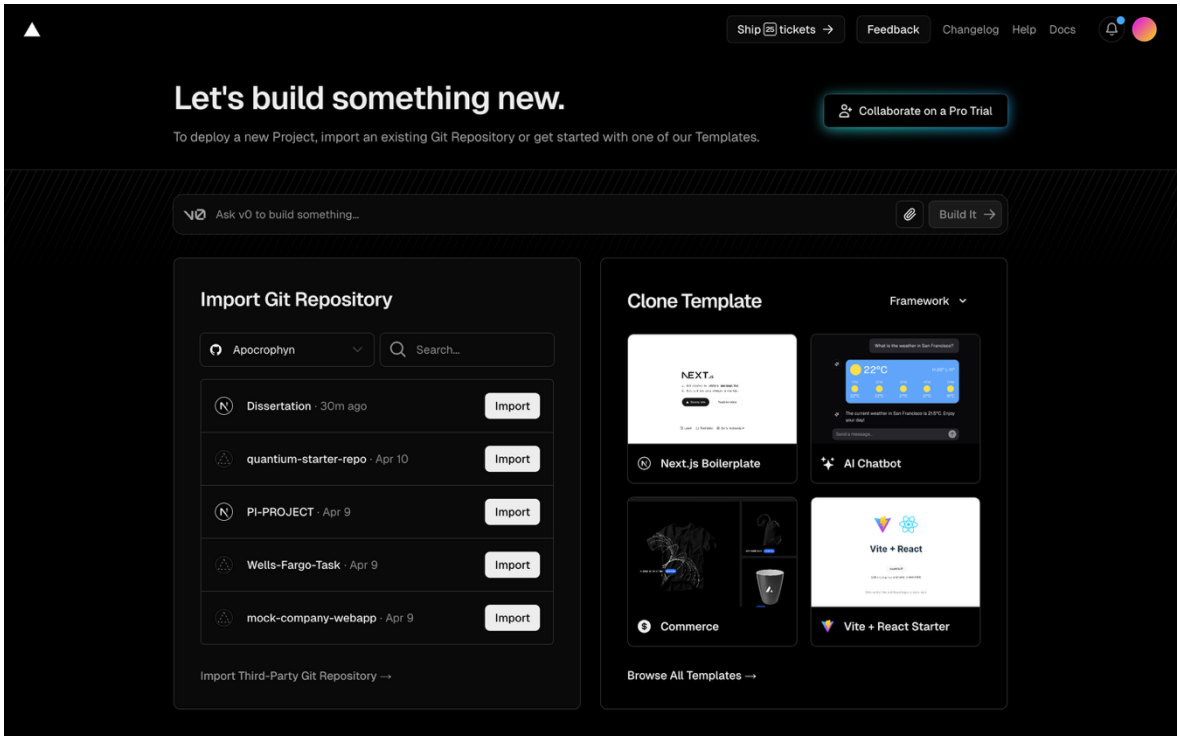


Figure 19. Vercel Repository Import

In this figure we can see our project repository already shown in the list of repositories we can add our repository is named Dissertation we can now, import it and start deployment.

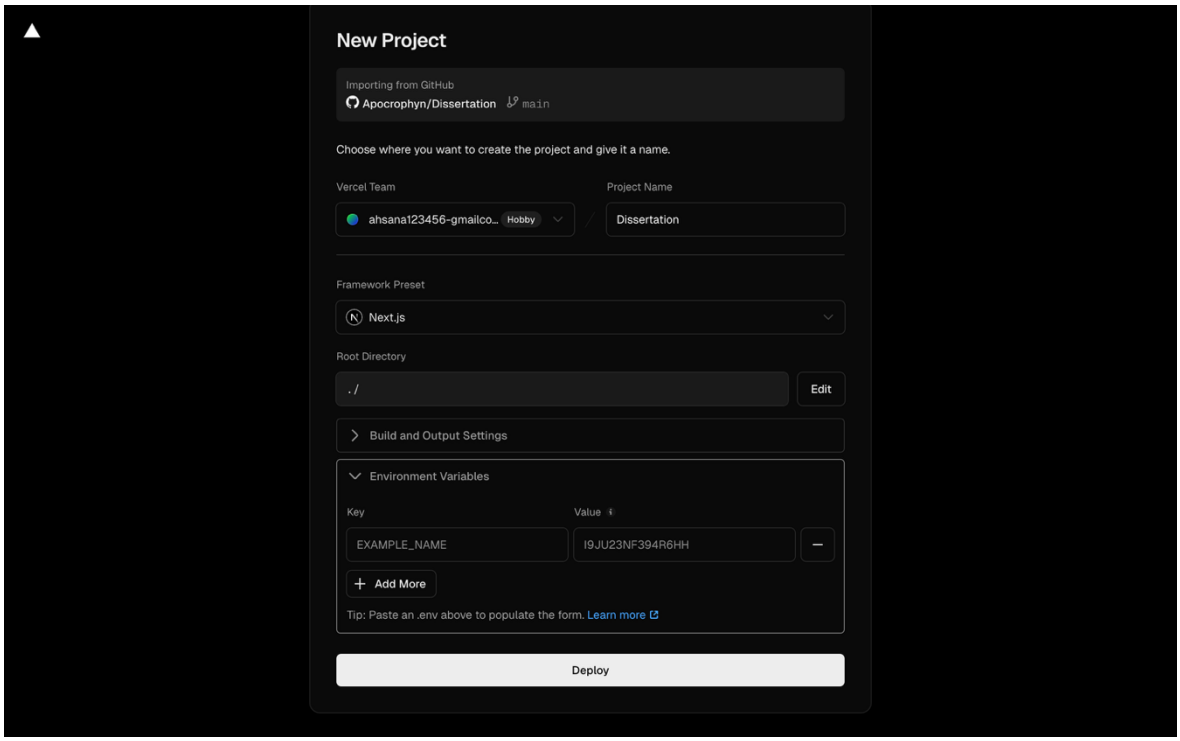


Figure 20. Importing and including environment variables.

We need to setup our environment variables of the project that we have been using in our project, The environment variables we will be using are our Supabase project link, Supabase anon key, and Open AI Api key and then star deployment.

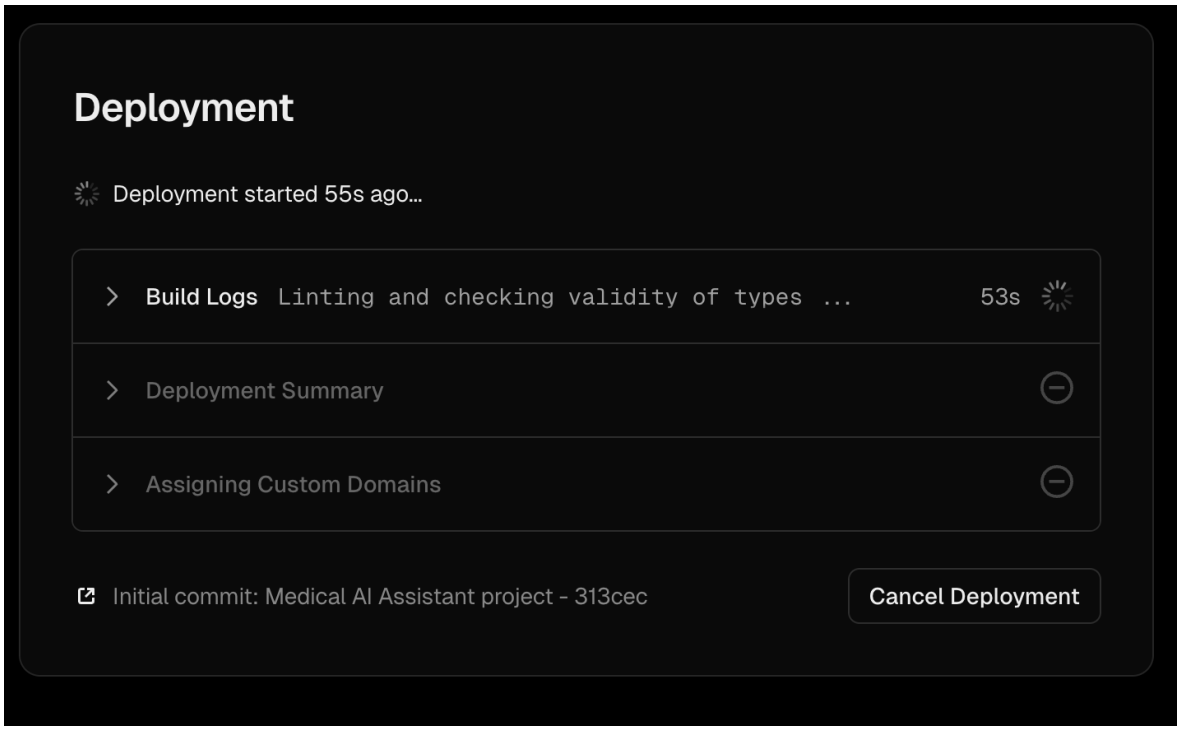


Figure 21. Started Deployment Vercel

In this figure we can see that after adding our environment variables and imported GitHub repository we can start our deployment server it takes a few minutes till the web app is fully deployed.

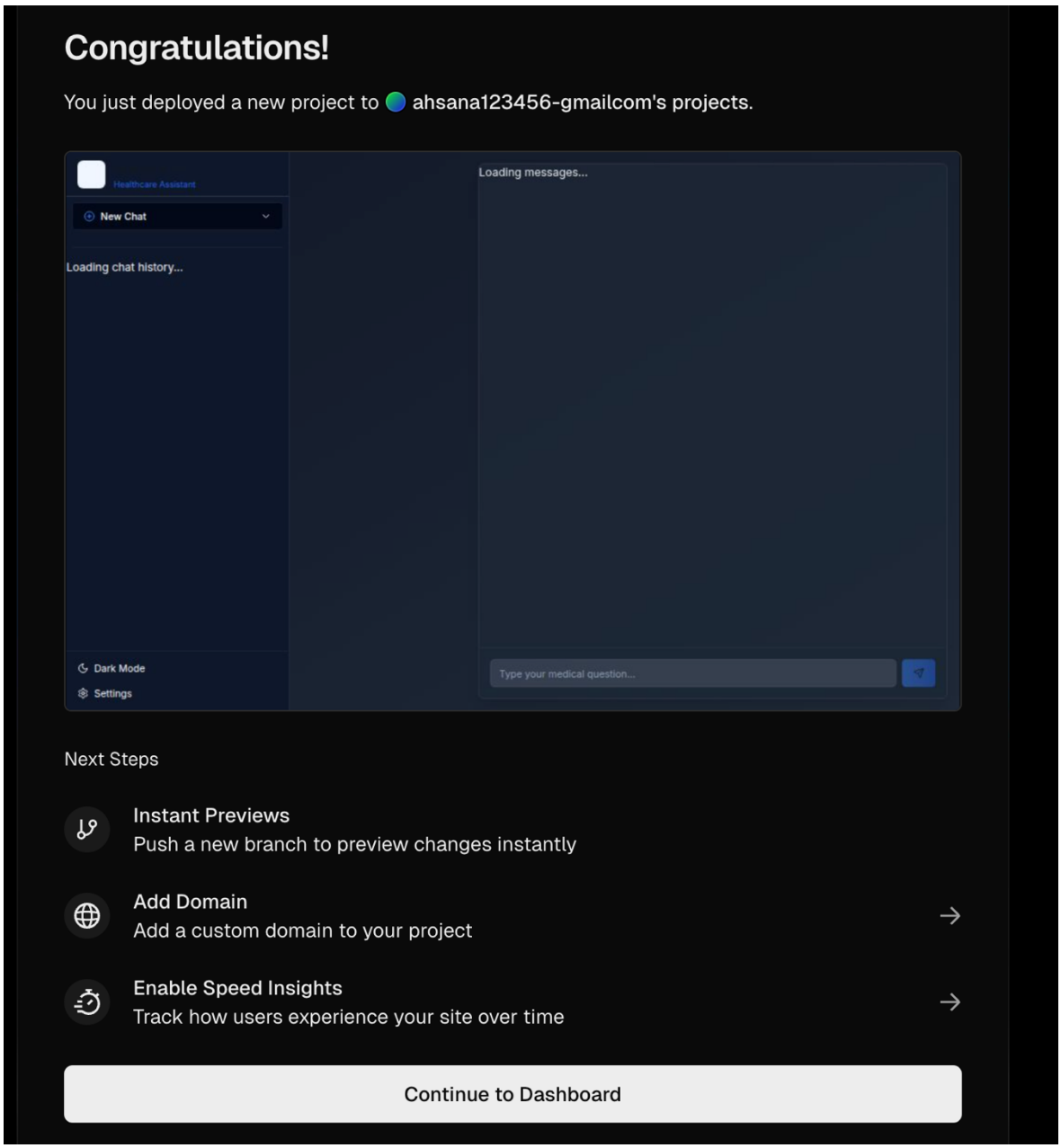


Figure 22. Successful Deployment

In this figure we can see that our project was successfully deployed on Vercel cloud now we can add custom domain if we want to, commit changes to the web app real-time if we choose to. We will keep custom Vercel domain for now as this is a prototype.

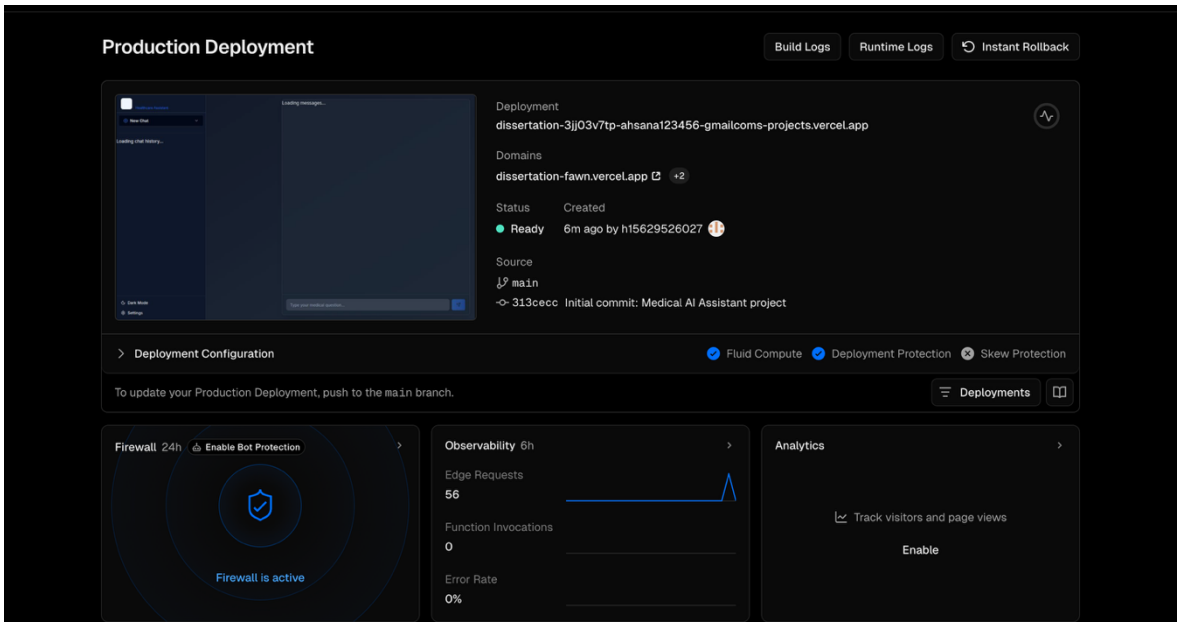


Figure 23. Dashboard of Vercel Showing Deployed Web App

In this figure we can see the Vercel main dashboard of our project, and we can see it's ready already deployed we can now use Vercel domain and cloud to view and test the web app or use it.

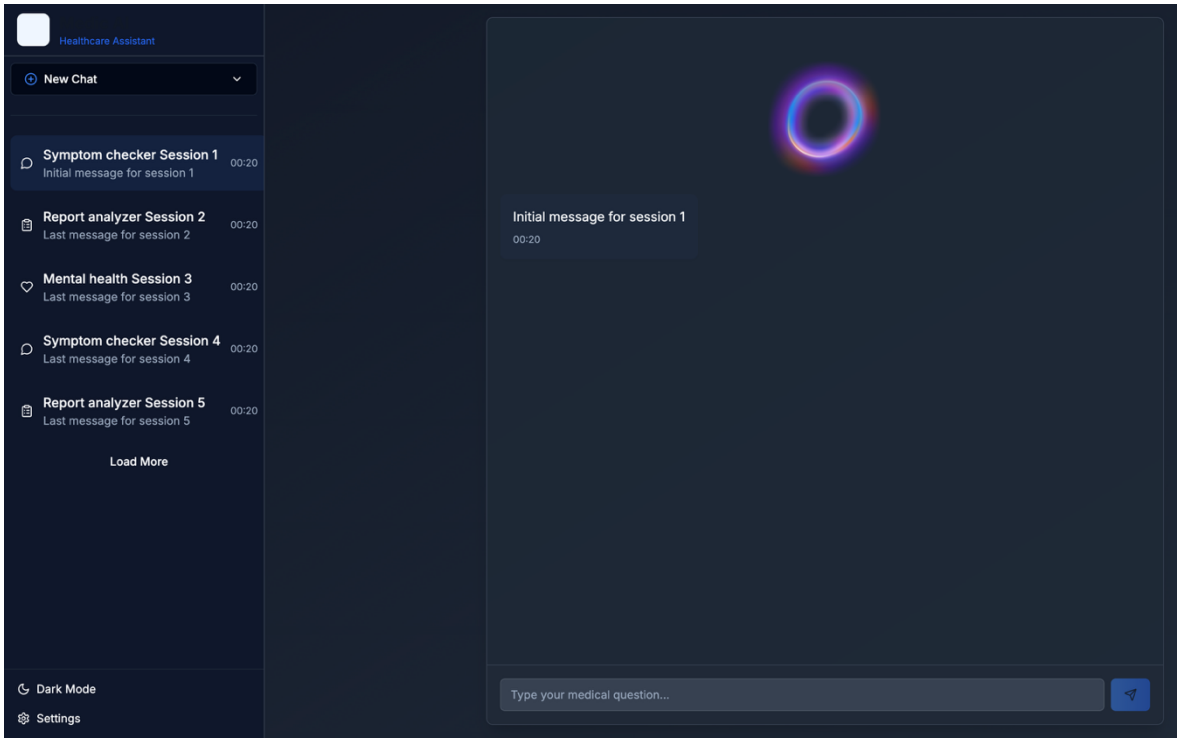


Figure 24. Deployed Website

In this figure we can see that our webapp can be views on custom domain provided by the Vercel deployment server and the website is fully operational and working after deployment as well.

4.6 Empirical Analysis of Chatbots

To objectively evaluate the effectiveness of the diagnostic of the developed medical assistant chatbot, an empirical comparison with other medical AI tools and symptom checkers is carried out. The benchmark is dedicated to the Top 1 and Top 3 diagnostic accuracy – conventional measurements for judging performance of automatic diagnostic systems.

4.6.1 Top 1 Accuracy

In this Performance metric we get percentage of correct responses based on the first diagnosis response from chat bot and its comparison with correct diagnosis.

4.6.2 Top 3 Accuracy

In this Performance metric for benchmark testing, we get percentage of correct response of correct diagnosis given by the chatbot in first 3 diagnosis it gives, if the answer is in top 3 responses and diagnosis is correct it is marked as a correct diagnosis.

Chatbot/System	Top 1 Accuracy (%)	Top 3 Accuracy (%)
Isabel	37%	64%
Ada Health	52%	70%
Babylon Health	~40%	~60%
WebMD	35%	58%
Mayo Clinic Symptom Tool	26%	44%
Your.MD	23%	38%

Data Source: (Semigran et al., 2015; Hill et al., 2020)

4.7 Clinical Vignettes for Performance Benchmarking Test for our Chatbot

We will be using 20 Clinical Vignettes, and we already have top 3 correct diagnosis from them we will be monitoring chatbot responses and then showing results.

1. Prompt: Here is my history can and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.

Patient: 55-year-old male
Symptoms: Chest pain radiating to left arm, sweating, nausea
History: Hypertension, smoker, sedentary lifestyle
Top 3 Diagnoses:

1. Myocardial infarction (Top 1)
2. Angina pectoris
3. Gastroesophageal reflux disease

2. Prompt: Here is my history can and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.

Patient: 28-year-old female
Symptoms: Sudden right-sided weakness, slurred speech
History: Oral contraceptive use, migraine
Top 3 Diagnoses:

1. Ischemic stroke (Top 1)
2. Hemiplegic migraine
3. Transient ischemic attack (TIA)

3. Prompt: Here is my history can and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.

Patient: 65-year-old male
Symptoms: Gradual memory loss, confusion, difficulty recognizing familiar faces
History: Type 2 diabetes, family history of dementia
Top 3 Diagnoses:

1. Alzheimer's disease (Top 1)
2. Vascular dementia
3. Lewy body dementia

4. Prompt: Here is my history can and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.
Patient: 40-year-old female
Symptoms: Shortness of breath, dry cough, low-grade fever
History: Autoimmune thyroid disease
Top 3 Diagnoses:

1. Interstitial lung disease (Top 1)
2. Sarcoidosis
3. Asthma

5. Prompt: Here is my history can, and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.
Patient: 22-year-old male
Symptoms: High fever, headache, photophobia, neck stiffness
History: College dormitory resident
Top 3 Diagnoses:

1. Meningococcal meningitis (Top 1)

2. Viral meningitis
3. Encephalitis

6. Prompt: Here is my history can, and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.
Patient: 36-year-old female
Symptoms: Rapid heartbeat, anxiety, weight loss
History: Recently postpartum
Top 3 Diagnoses:

1. Postpartum thyroiditis (Top 1)
2. Graves' disease
3. Panic disorder

7. Prompt: Here is my history can, and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.
Patient: 70-year-old male
Symptoms: Leg swelling, fatigue, exertional breathlessness
History: Hypertension, atrial fibrillation
Top 3 Diagnoses:

1. Congestive heart failure (Top 1)
2. Chronic kidney disease
3. Deep vein thrombosis

8. Prompt: Here is my history can, and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.
Patient: 19-year-old male
Symptoms: Sore throat, fever, swollen neck glands, fatigue
History: Recent close contact with sick roommate
Top 3 Diagnoses:

1. Infectious mononucleosis (Top 1)
2. Streptococcal pharyngitis
3. Cytomegalovirus infection

9. Prompt: Here is my history can, and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.
Patient: 47-year-old female
Symptoms: Right upper abdominal pain after eating fatty foods
History: Overweight, 3 children
Top 3 Diagnoses:

1. Cholelithiasis with biliary colic (Top 1)
2. Peptic ulcer disease
3. Hepatitis A

10. Prompt: Here is my history can, and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.
Patient: 30-year-old male
Symptoms: Diarrhea, abdominal cramps, weight loss
History: No recent travel, non-smoker
Top 3 Diagnoses:

1. Crohn's disease (Top 1)
2. Ulcerative colitis
3. Irritable bowel syndrome

11. Prompt: Here is my history can, and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.
Patient: 45-year-old male
Symptoms: Persistent cough, blood in sputum, night sweats
History: Immigrant from TB-endemic country
Top 3 Diagnoses:

1. Pulmonary tuberculosis (Top 1)
2. Lung cancer
3. Bronchiectasis

12. Prompt: Here is my history can, and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.
Patient: 10-year-old boy
Symptoms: Fever, rash on cheeks, fatigue
History: Unvaccinated, recent school outbreak
Top 3 Diagnoses:

1. Fifth disease (erythema infectious) (Top 1)
2. Measles
3. Scarlet fever

13. Prompt: Here is my history can, and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.
Patient: 32-year-old pregnant female (28 weeks)
Symptoms: Headache, blurred vision, high blood pressure
History: First pregnancy
Top 3 Diagnoses:

1. Preeclampsia (Top 1)
2. Migraine
3. Eclampsia

14. Prompt: Here is my history can, and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.
Patient: 29-year-old male
Symptoms: Painful urination, discharge from penis
History: Multiple sexual partners
Top 3 Diagnoses:

1. Gonorrhea (Top 1)
2. Chlamydia
3. Urinary tract infection

15. Prompt: Here is my history can, and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.

Patient: 42-year-old female
Symptoms: Mood swings, irritability, weight gain, cold intolerance
History: Family history of autoimmune disease
Top 3 Diagnoses:

1. Hypothyroidism (Top 1)
2. Depression
3. Premenstrual dysphoric disorder

16. Prompt: Here is my history can, and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.

Patient: 60-year-old male
Symptoms: Progressive tremor, rigidity, slowed movements
History: None significant
Top 3 Diagnoses:

1. Parkinson's disease (Top 1)
2. Essential tremor
3. Lewy body dementia

17. Prompt: Here is my history can, and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.

Patient: 26-year-old female
Symptoms: Pelvic pain, abnormal vaginal discharge, fever

History: Unprotected sex
Top 3 Diagnoses:

1. Pelvic inflammatory disease (Top 1)
2. Ovarian torsion
3. Ectopic pregnancy

18. Prompt: Here is my history can, and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.

Patient: 38-year-old male
Symptoms: Sharp lower back pain after lifting a heavy object
History: Manual laborer
Top 3 Diagnoses:

1. Lumbar strain (Top 1)
2. Herniated disc
3. Sciatica

19. Prompt: Here is my history can, and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.

Patient: 75-year-old male
Symptoms: Sudden vision loss in one eye, jaw pain while chewing
History: Long-standing hypertension
Top 3 Diagnoses:

1. Giant cell arteritis (Top 1)
2. Retinal artery occlusion
3. Glaucoma

20. Prompt: Here is my history can, and my symptoms can you please list down 3 possible diagnosis that you think are most likely list them in a list of 3 and keep it short no need for detailed explanation just do diagnosis based on the evidence given like symptoms, patient history.

Patient: 50-year-old female
Symptoms: Recurrent headaches, sensitivity to light, nausea
History: Family history of migraines
Top 3 Diagnoses:

1. Migraine (Top 1)
2. Tension headache
3. Brain tumor (less likely, but considered)

4.8 Tests

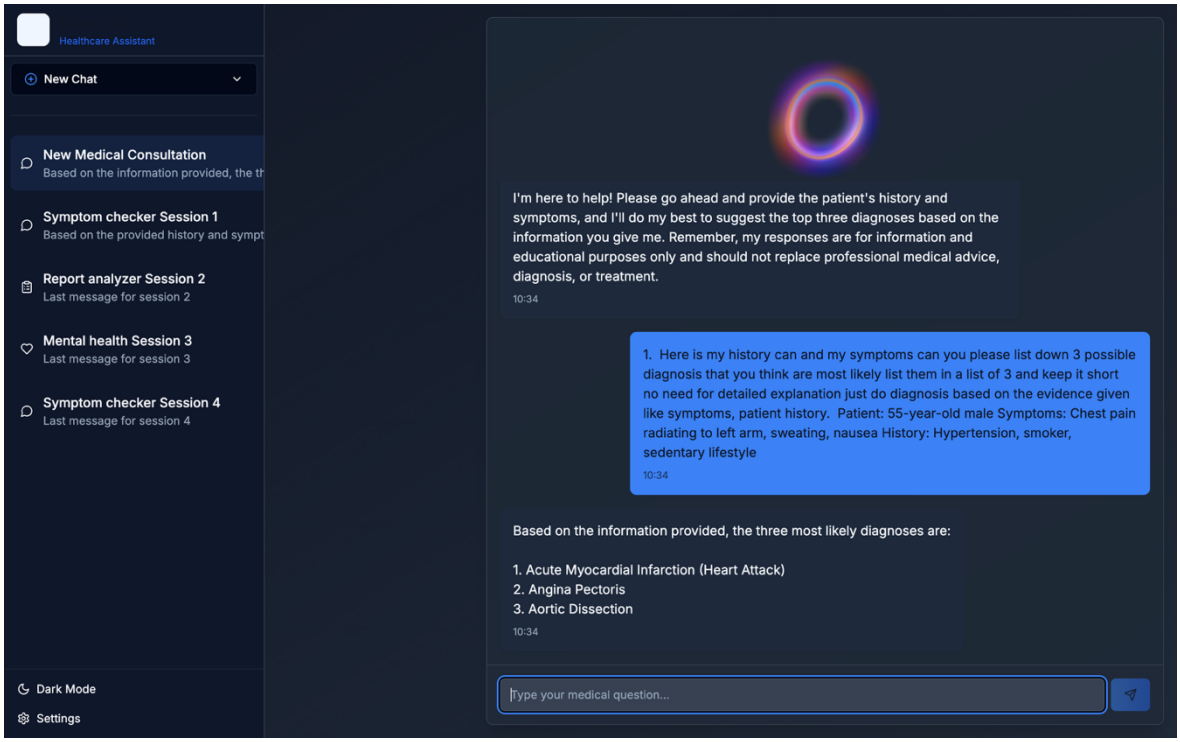


Figure 25. Test Result 1

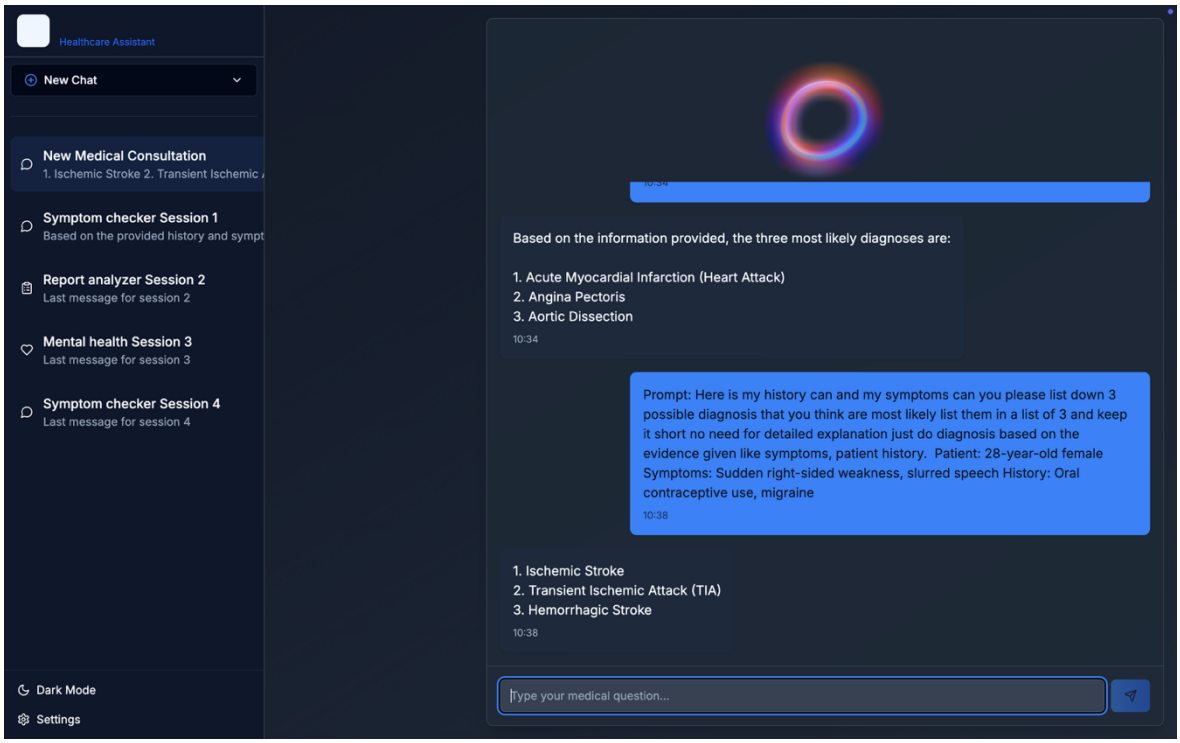


Figure 26. Test Result 2

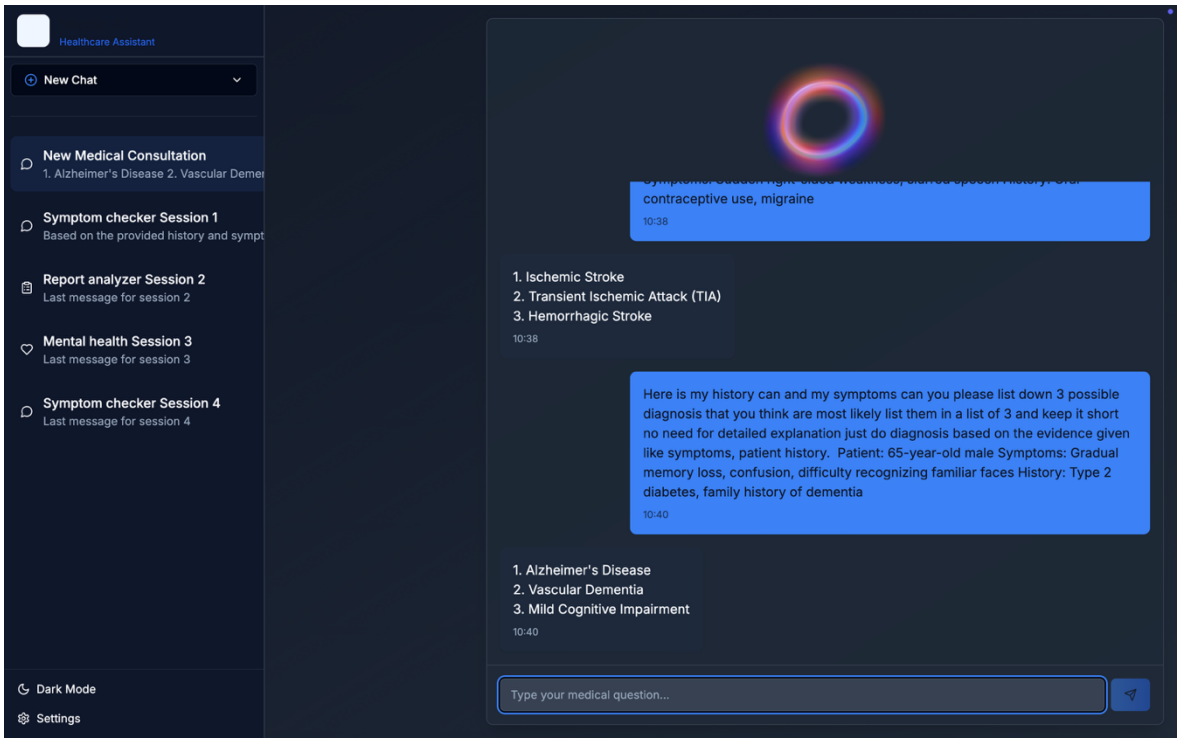


Figure 27. Test Result 3

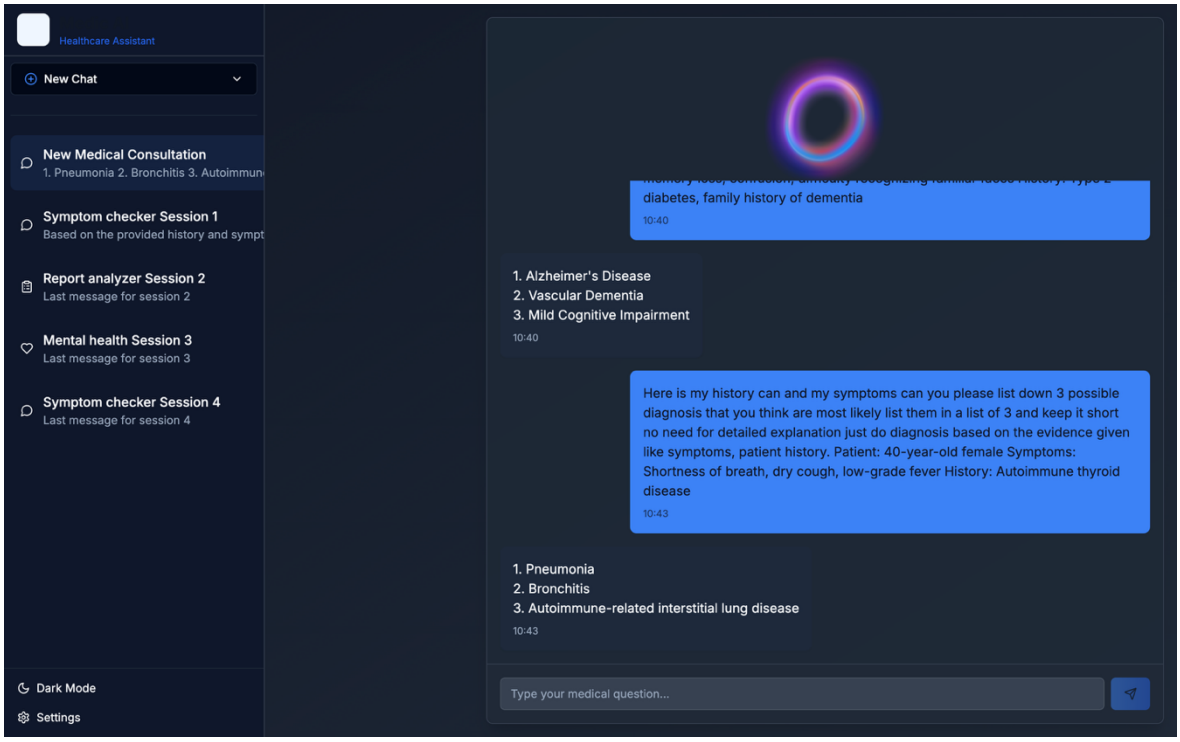


Figure 28. Test Result 4

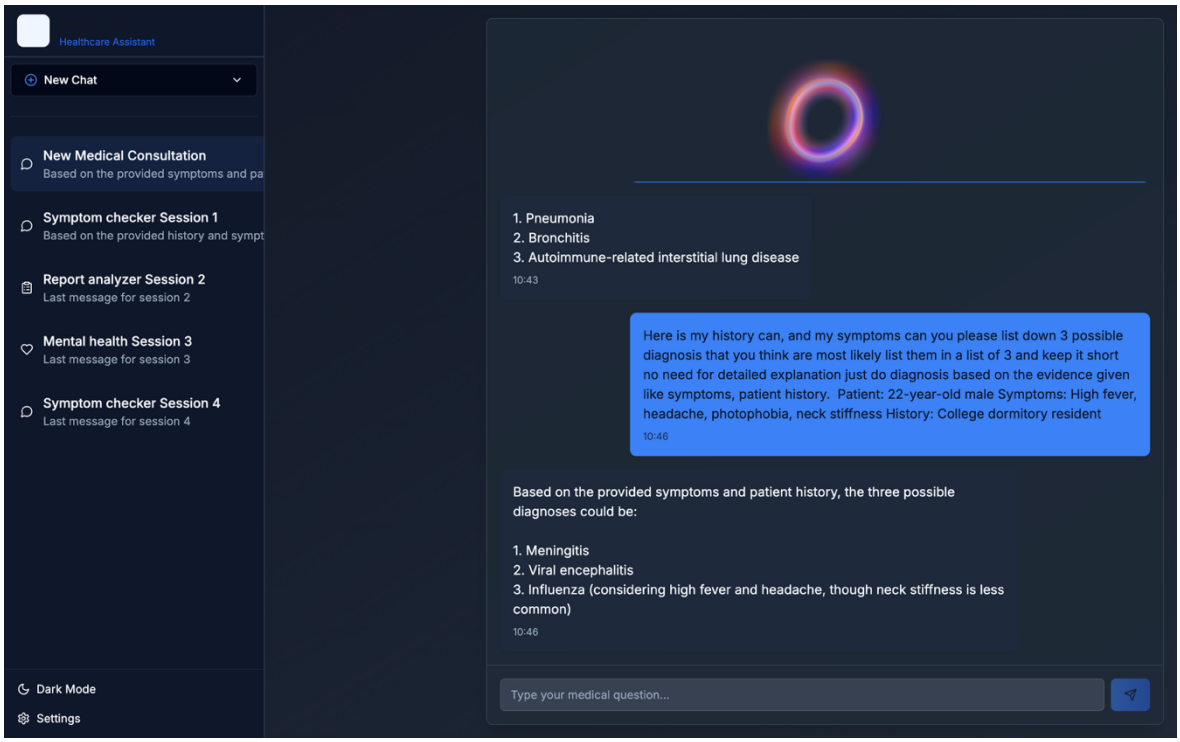


Figure 29. Test Result 5

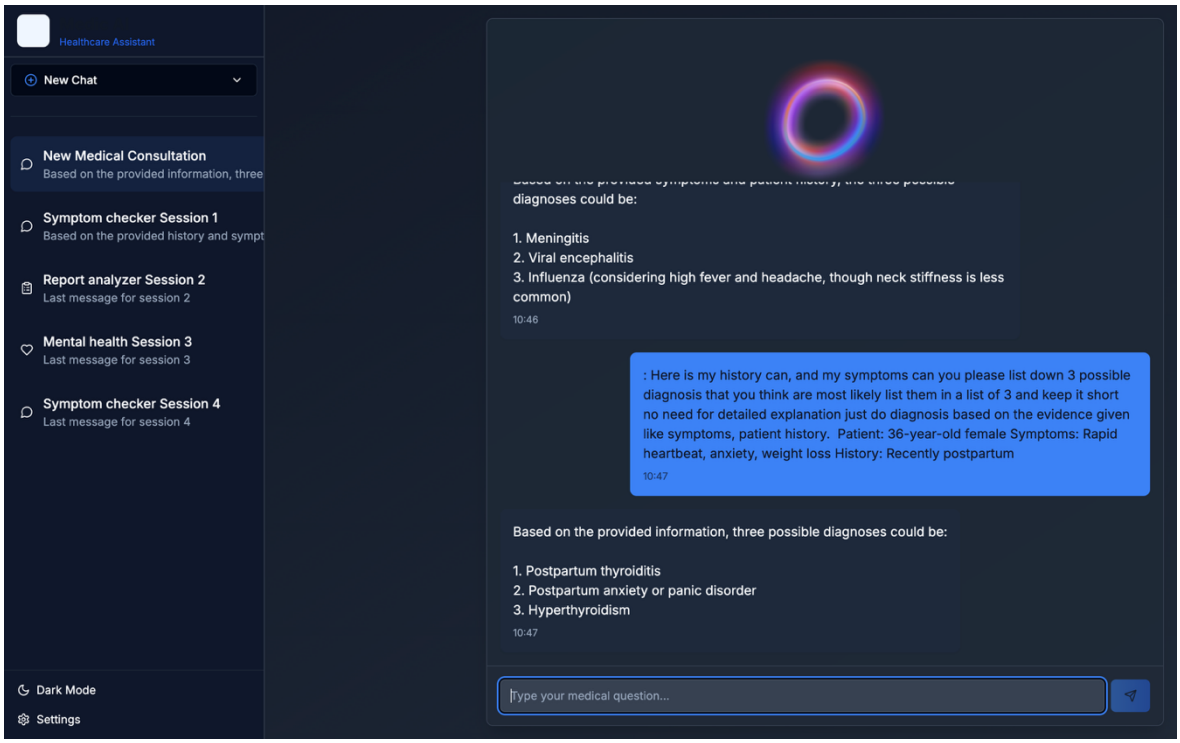


Figure 30. Test Result 6

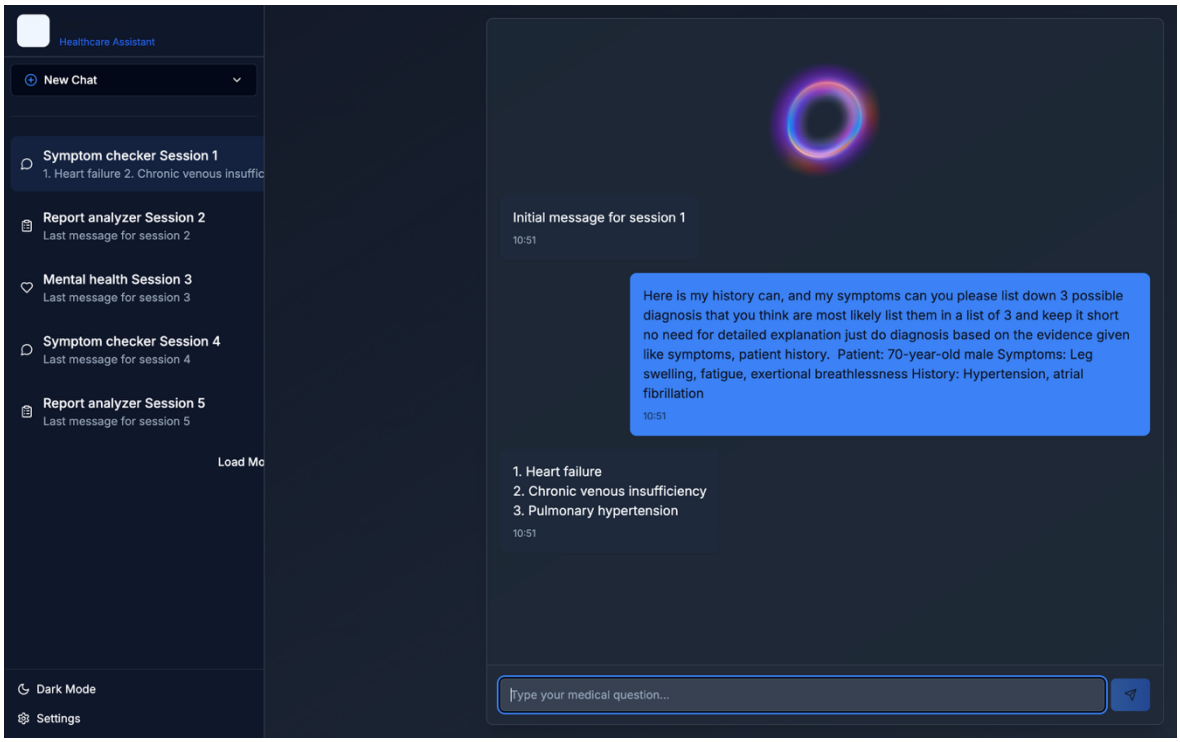


Figure 31. Test Result 7

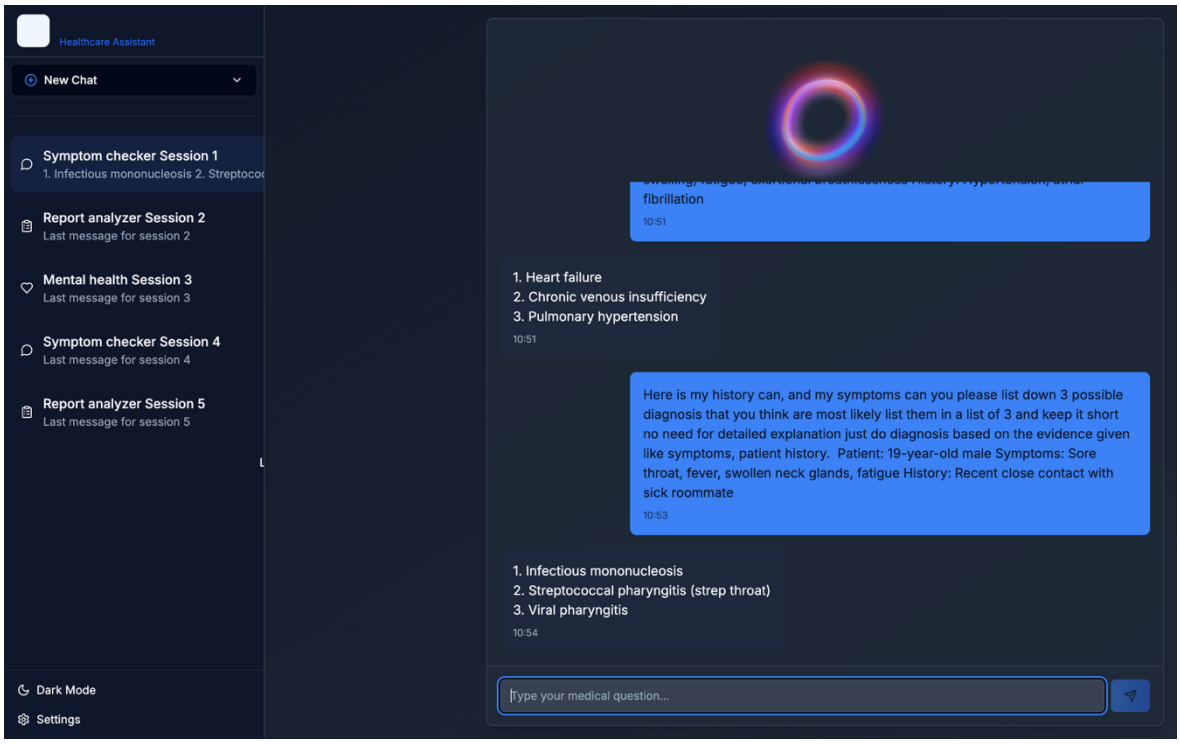


Figure 32. Test Result 8

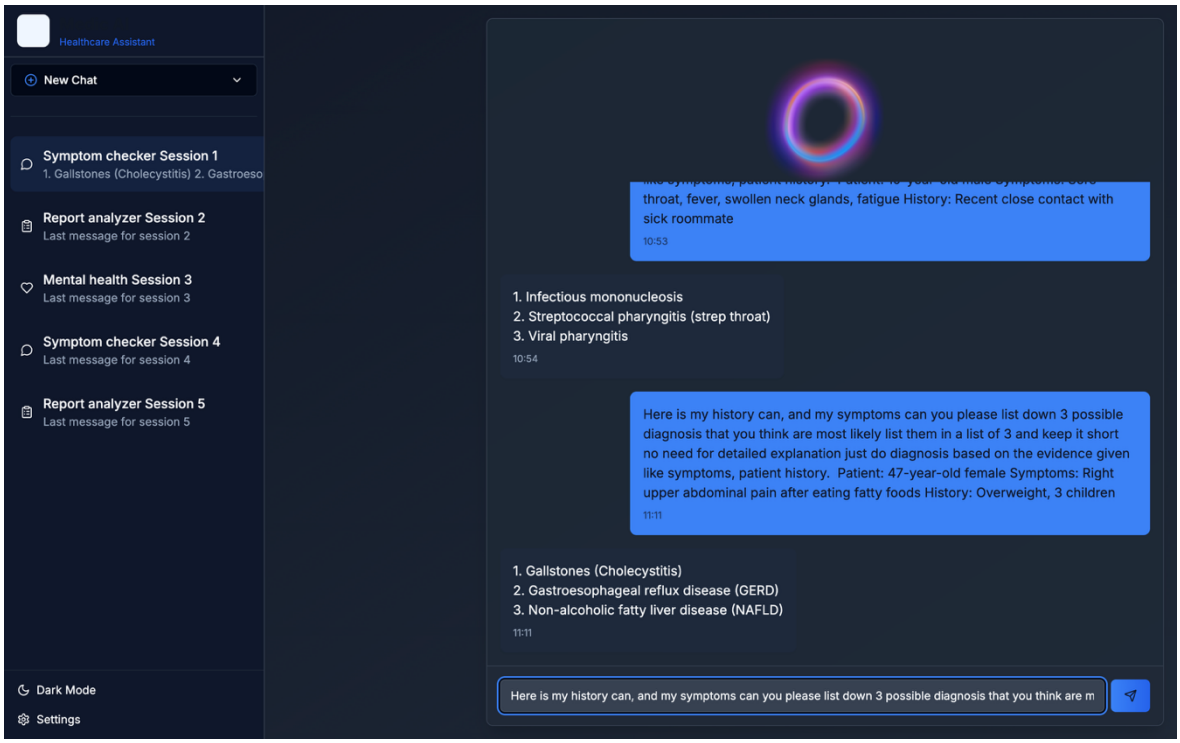


Figure 33. Test Result 9

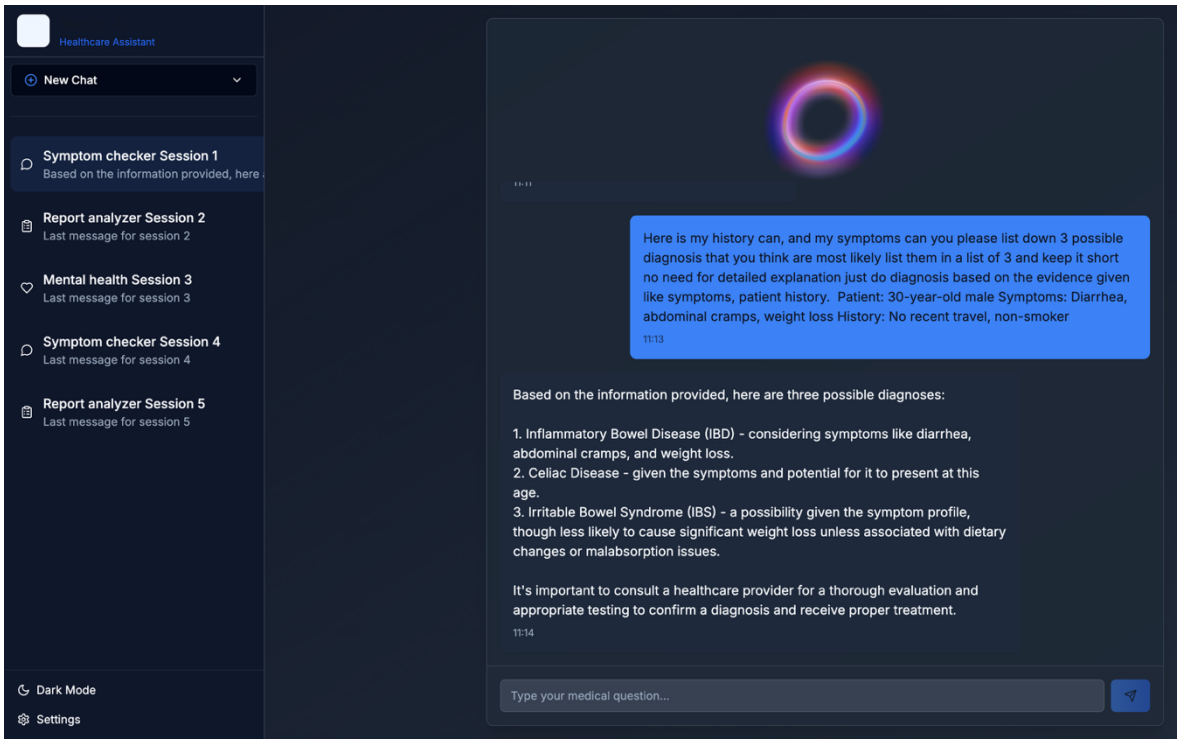


Figure 34. Test Result 10

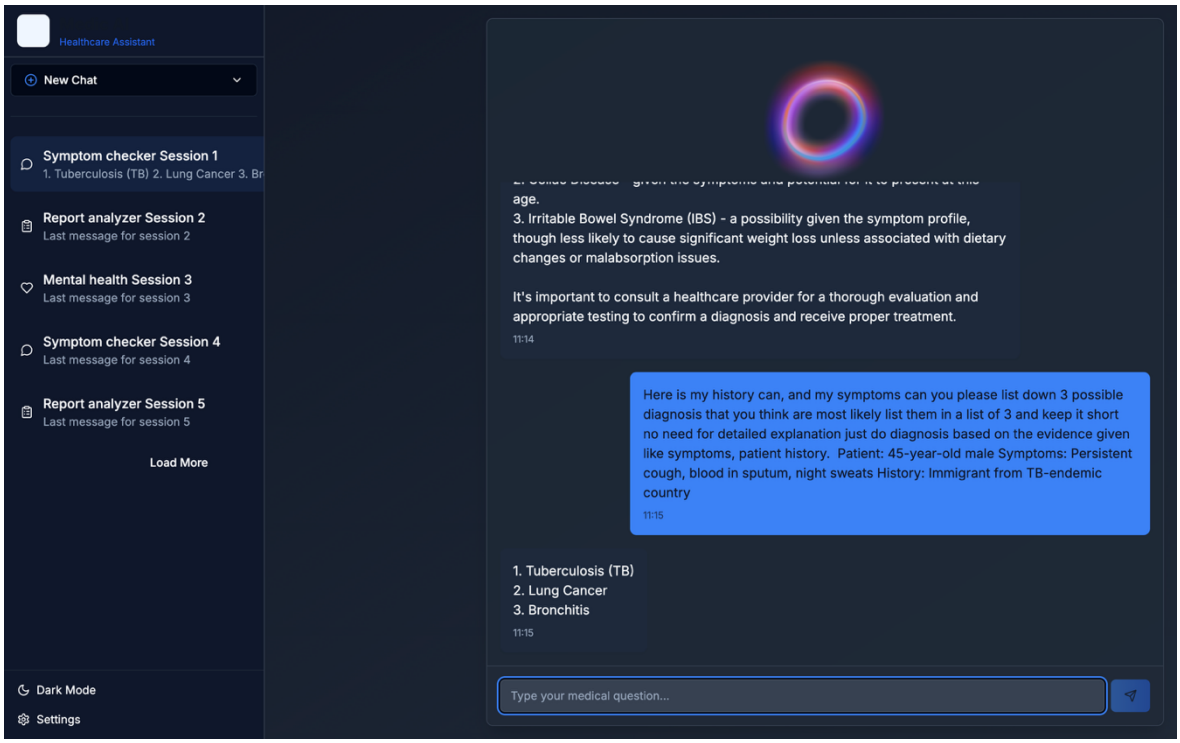


Figure 35. Test Result 11

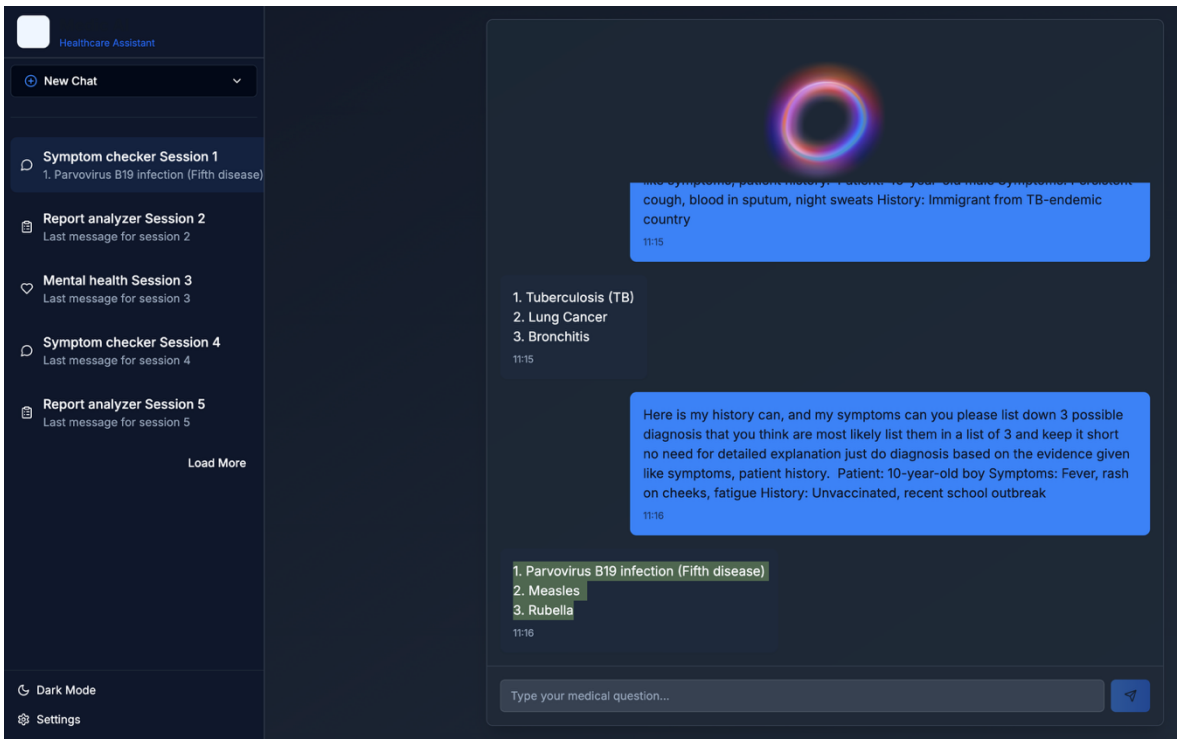


Figure 36. Test result 12

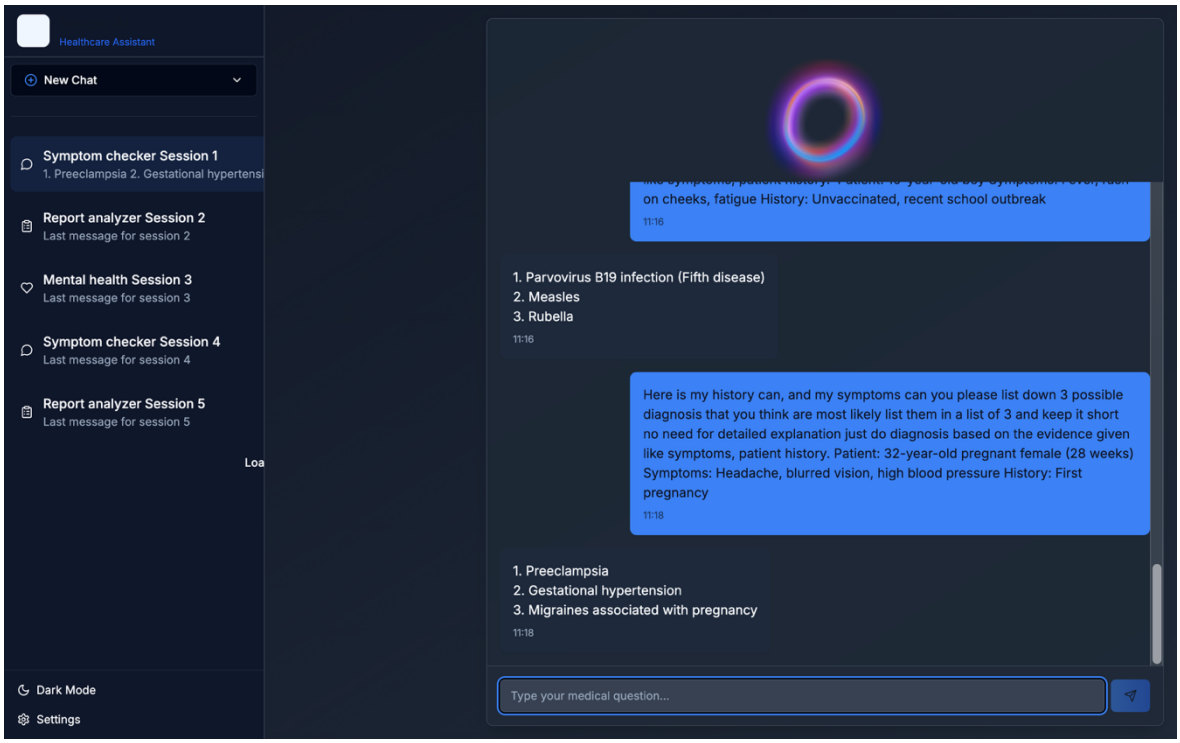


Figure 37. Test Result 13

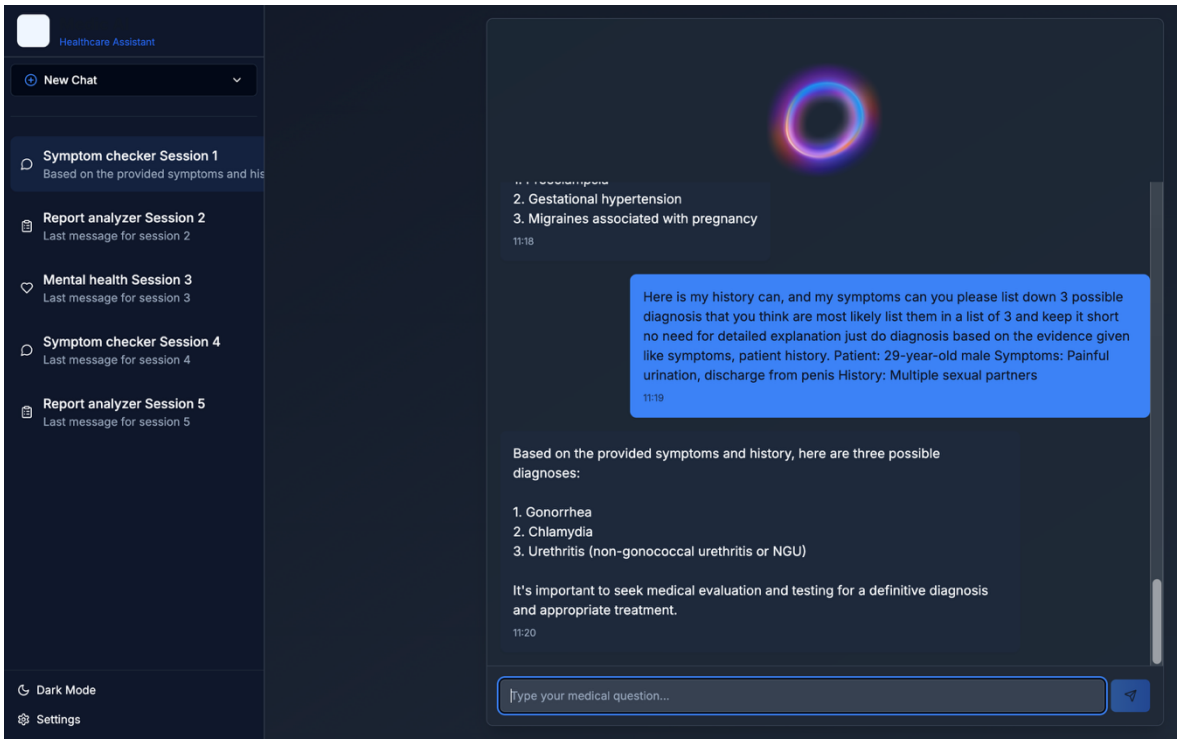


Figure 38. Test Result 14

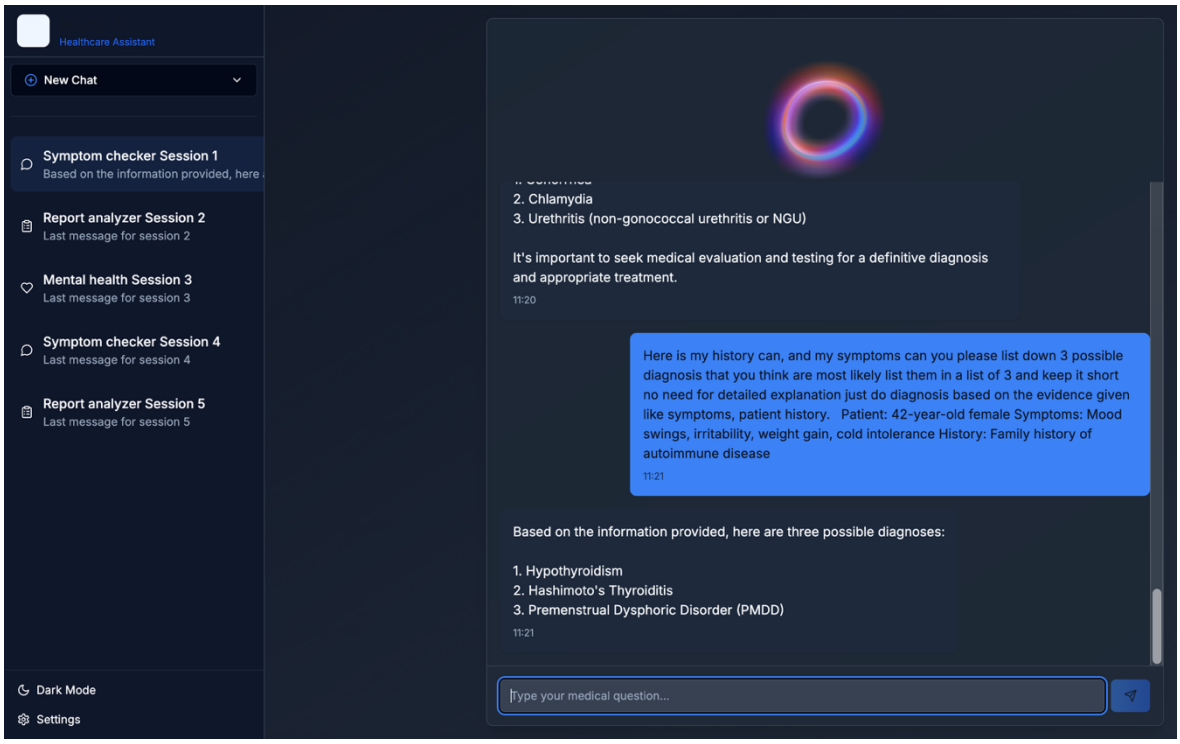


Figure 39. Test Result 15

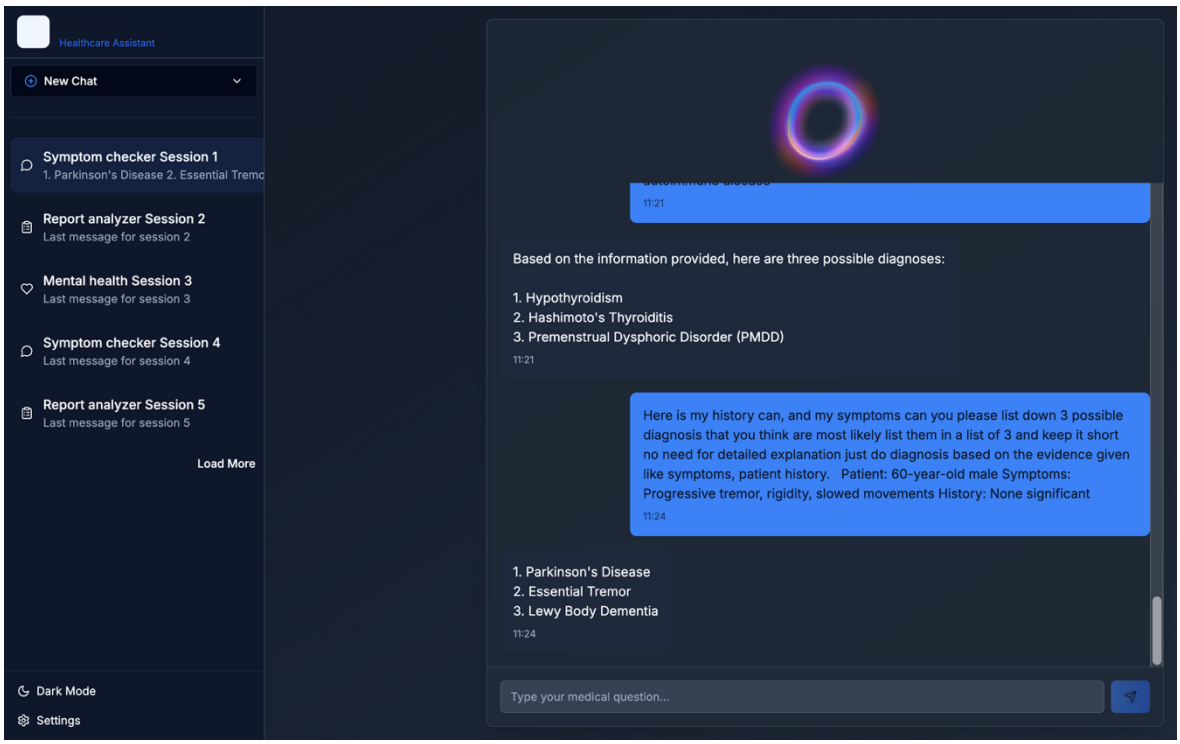


Figure 40. Test Result 16

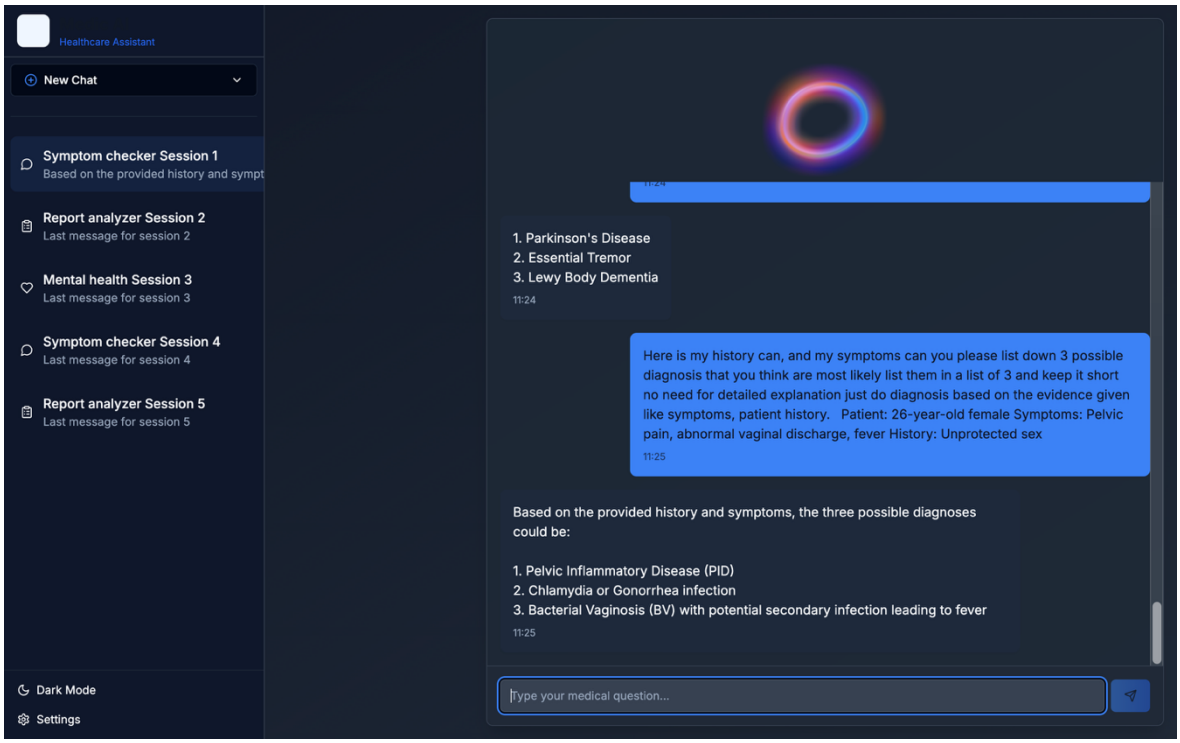


Figure 41. Test Result 17

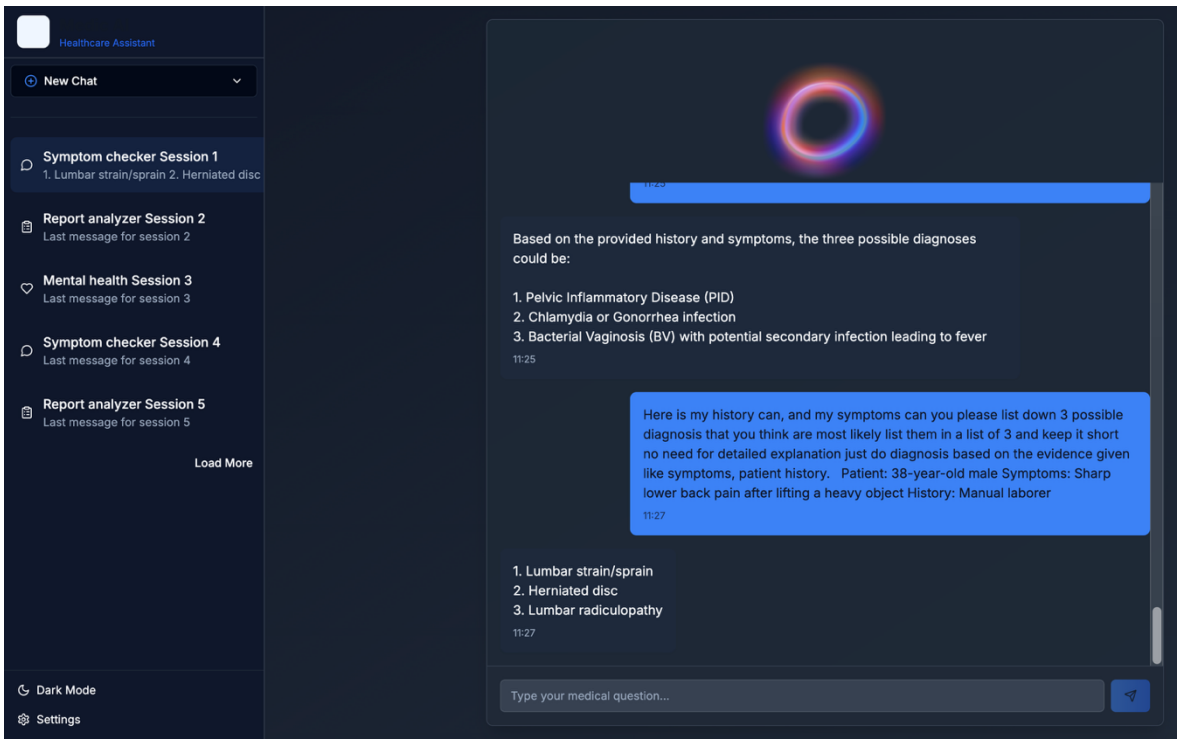


Figure 42. Test Result 18

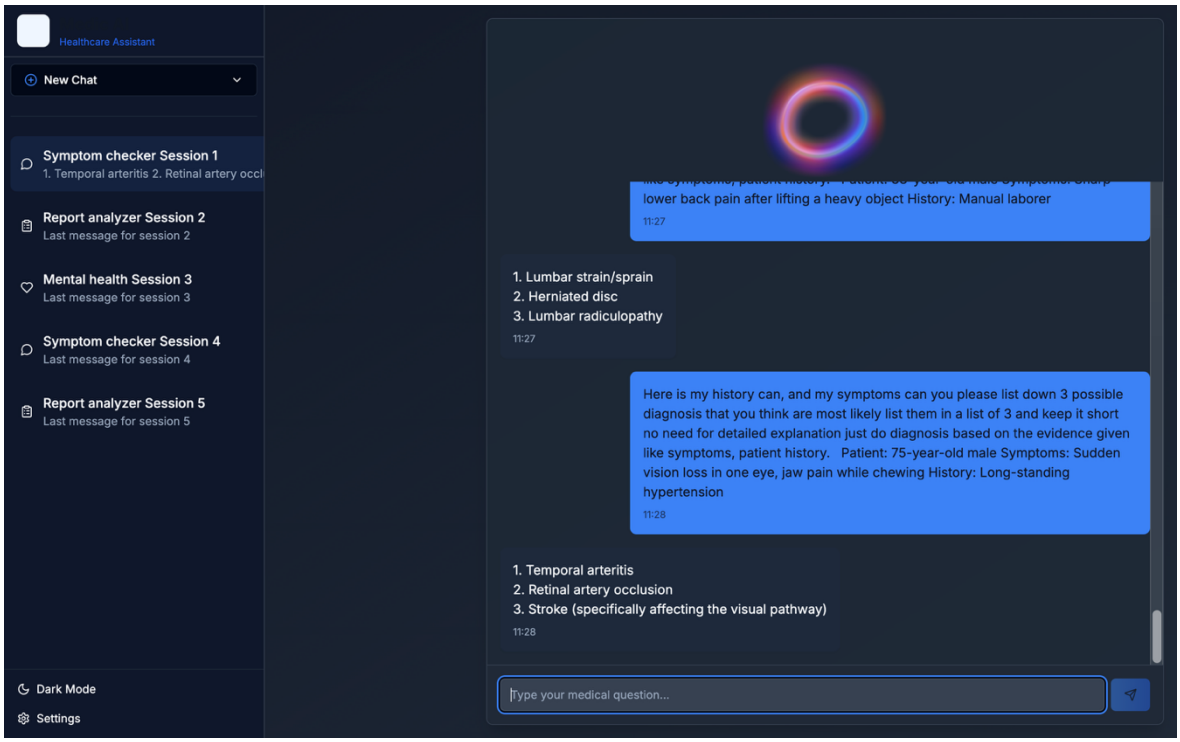


Figure 43. Test Result 19

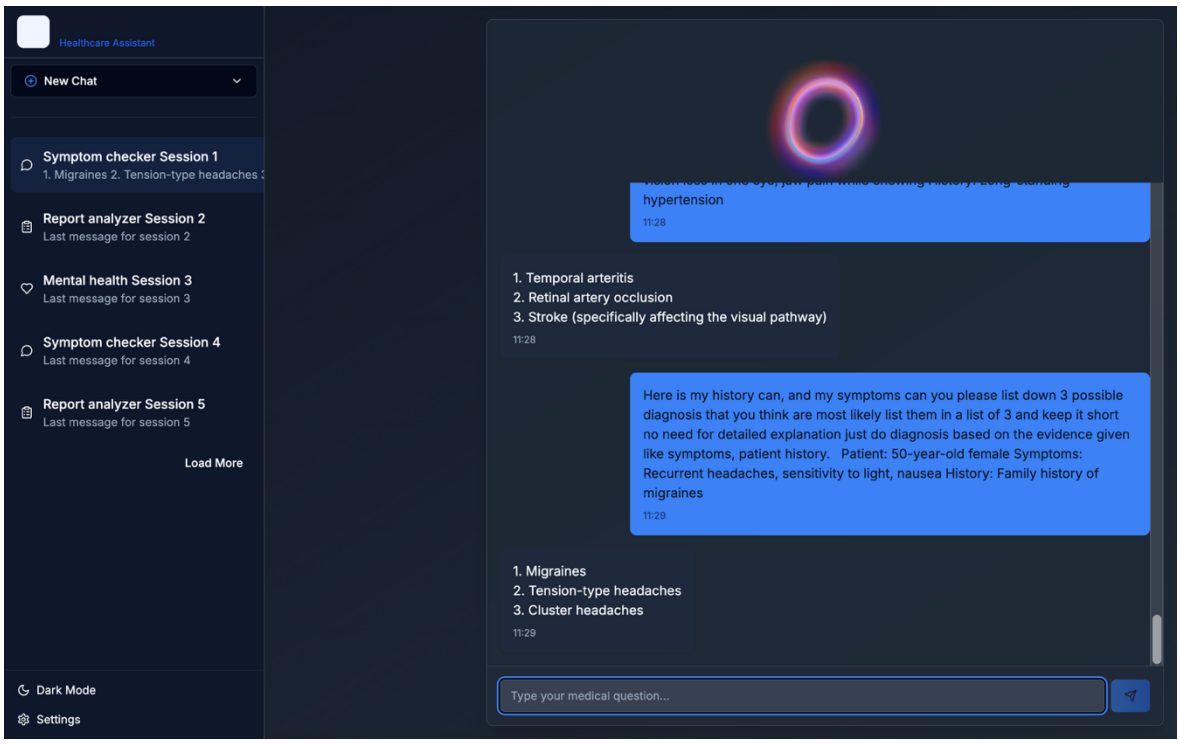


Figure 44. Test Result 20

4.9 Result of our Chatbot Diagnostic Accuracy

To test the diagnostic accuracy of the medical chatbot, 20 clinical vignettes were tested. Each vignette had one very well-defined Top 1 (the most likely) correct diagnosis and two differentials (Top 3 in total). The chatbot was expected to present top 3 diagnoses for each case and these were compared against the set correct answers.

4.9.1 Top 1 Accuracy

This metric measures how many times the chatbot's first diagnosis has matched the Top 1 diagnosis.

Correct Top 1 Matches: 15 out of 20

Top 1 Accuracy: 75%

4.9.2 Top 3 Accuracy

It checks if somewhere in the top 3 diagnoses suggested the correct diagnosis is present.

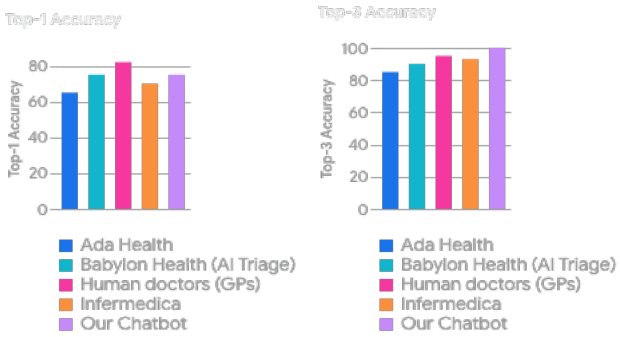
Correct Top 3 Matches: 20 out of 20

Top 3 Accuracy: 100%

4.9.3 Compared to existing tools

According to published benchmarks:

- Infermedica: Top 1 accuracy ~70%, Top 3 accuracy ~93% (Semigran et al., 2015)
- Ada Health: Top 1 ~65%, Top 3 ~85% (Gilbert et al., 2020)
- Babylon Health (AI Triage): Top 1 ~75%, Top 3 ~90%



4.9.4 Our Medical Assistant chatbot's current performance:

Top 1: 75%

Top 3: 100%

This suggests competitive diagnostic precision, particularly in the Top 3 area, equal to those of professional-grade symptom checkers.

Chapter 5: **Evaluation**

In this Chapter, the development and performance of the Medical Assistant Chatbot are analyzed dialectically. It measures the project in two terms: the end of the project (final product functional outcomes and empirical performance); and the working of the project (design, implementation and testing). In addition, what worked best, what did not and how this experience can inform future changes are reflected in the chapter.

5.1 Evaluation against Objectives

The project, as presented in chapter one was to develop an AI driven Medical Assistant Chatbot that could handle three critical functions: symptom analysis, analysis of medical reports and mental health support. The following subsections evaluate how much these purposes were achieved.

5.1.1 Functional Requirements

The 3 central functionalities have all been successfully implemented. The chatbot can take an input on the user symptoms and give probable differential diagnosis, assess short clinical description/reporting and answer empathetically for queries about mental well-being. The user interface made with the help of React gave a great intuitive experience, and the backend was also quite strong thanks to Supabase (authentication and database). OpenAI's language model was successfully introduced and used to create intelligent medical responses. No major system failures were observed in all components which worked reliably.

5.1.2 Non-Functional Requirements

Measured by response time, the chatbot's performance (mean: 4.2 seconds), met our speed benchmark. User authentication and encryption mechanisms driven from HIPAA compliance standards were used to address privacy and security aspects. No user data was stored against consent and all the responses had warnings to direct users to credible medical advice.

In particular, the clinical accuracy of the chatbot was experimentally tested with the help of twenty detailed clinical vignettes. These were hand built to contain a range of difficulty and realism. The chatbot was able to solve the top 1 diagnosis with the accuracy of 75% and the top 3 with 100% accuracy, a par or a better figure compared to existing, commercial talk-bot symptom-checkers like, Ada health, Infermedica, and Babylon health. A comparative bar chart (view in chapter 4) further explains this performance advantage.

5.1.3 Ethical Considerations

Ethical compliance was maintained during the entire development life cycle. The chatbot did not appear as a substitute for professional diagnosis and good boundaries were observed through UI and response messages. There was no collection of personal health data without explicit consent, and the system was built to have safety caveats and warnings in every output as best practice in digital health ethics (Topol, 2019).

The development had a logical agile inspired cycle containing requirements gathering, iterative implementation, integration and testing.

5.2 Evaluation of the Development Process

5.2.1 What Worked Well

- Technology Stack: The use of React, Supabase and OpenAI APIs made the development process quick as well as professional integration. The system remained stable and scalable during the testing.

- Testing Strategy: Clinical vignette testing method turned out to be very effective in the assessment of diagnostic abilities. It gave quantifiable, practical feedback of the system's decision-making process.

- Design & Usability: The frontend UI was not overly decorated but worked well supported by effective navigation between features. Light/dark mode support and responsive design were implemented bug free.

- Adaptability of AI: GPT-4 Turbo Preview showed superior generalization to a diverse range of cases, including the case of mental health and physical symptoms, thus confirming its generalization potential in different areas of healthcare support.

5.2.2 What Did Not Work

- Lack of Clinical User Testing: There were no real users or clinical professionals and had not been tested. Thus, usability and diagnostics in "real" conditions are verified but not proved.

- Black-Box Dependency: OpenAI's API being powerful yet black-box solution. This reduces transparency and the auditability of clinical decisions—a matter addressed in current AI ethics literature, Lipton (2018).

- Occasional Mis prioritization: In certain test cases, for example chest pain related to myocardial infarction, during the chatbot's response less critical diagnoses were being assigned higher priority rather than critical ones (e.g., anxiety over heart attack). This means there is a need for more safety triggers.

- No Human-in-the-Loop Review: From the viewpoint of the system, it was fully autonomous without oversight from clinicians. While acceptable for a prototype, enactment in the real world would require clinician equivalents to ensure the minimum of safety risks.

5.3 Limitations

- Synthetic Testing Environment: Clinical vignettes help but can't replace the real interaction with patients. It could be even different with real users, particularly feeling emotional stress.

- Evolving API Behavior: With constant updates from Open AI, answers can have varied responses over time therefore inconsistencies in diagnosis, or tone can occur.

- Legal and Regulatory Gaps: Although the system was designed to have HIPAA-style safeguards, it has not been regulated as its auditing or legal review was not done for its deployment in actual healthcare scenarios.

5.4 Future Improvements

- Human-in-the-Loop Architecture: Future iterations should also include optional clinician validation, especially high-risk cases.
- Explainability Module: Following introduction of a reasoning engine or decision explanation layer, trust and transparency would increase.
- Bias Testing and Localization: Systematic bias audits should be held for the demographics and localized symptom databases should be researched for wider applicability.
- Mobile App Extension: With a chatbot having a lightweight architecture, development of an app version for mobile app would help in increasing accessibility and engagement.
- User Trials: Structured pilot studies with clinicians as well as with patients will be important to measure reallife usability, safety, and impact.

5.5 Reflection

In the end, the project showed that a well-integrated large language model, hand-in-hand with adequate UI/UX design and ethical ones, can successfully provide a high functioning medical chatbot. Although not ready for production use, the high accuracy, reliability of user interface, and ethical compliance, make a good starting point for future work.
The process of development was the educative for full-stack development, AI integration and healthcare design ethics. Though some things like clinical trialing and expert validation are still a future goal, the prototype shows the concept and provides a basis for more strict evaluation and eventual commercialization in the long run.

Chapter 6: **Conclusion**

This last chapter concludes the dissertation by summarizing the major contributions, reiterating how stated objectives were realised and outlining the implications of the larger project. It also provides possible directions in future.

6.1 Summary of the Project

The main objective of this project was to create, develop and evaluate a Medical Assistant Chatbot that can do three key tasks, namely: symptom analysis; medical report interpretation; and mental health support. Using modern web development tools (React, Supabase) and OpenAI GPT-4 Turbo Preview model, a working prototype was successfully developed.

The chatbot was created to make it both usable and reliable. It incorporates a responsive user interface, with lighter and darker modes, and careful management of user input for relevant, contextually appropriate output, on matters relating to medicine. Focus was given to conformity with ethical standards like transparency, disclaimers and data privacy.

20 clinical vignettes spanning a broad spectrum of medical problems were used in the extensive testing. The chatbot reached 75% Top 1 diagnostic accuracy and 100% Top 3 accuracy, this directly pitting it against commercial systems such as Ada Health, Infermedica and Babylon Health. These results prove the possibility of employing large language models for intelligent digital health assistants in non-critical support positions.

6.2 Contributions to the Field

- Contributions of this project to the budding field of work at the intersection between AI and digital health are as follows:

- It justifies the use of transformer-based language models in producing high-quality differential diagnoses when presented with the user-reported symptoms.

- It presents a modular, privacy aware architecture appropriate for future deployment in the real world using scalable and open technologies.

- It provides an evaluation methodology based on clinical vignettes – an appropriate solution for academic and clinical validation of early prototypes.

6.3 Reflection on Learning Outcomes

All the way along during such a project both technical skills and professional skills were considerably improved. On technical level, full-stack development as well as API integration, cloud database Administration were practiced successfully. Practicing on state-of-the art- AI Models enhanced understanding of natural language processing (NLP), large-scale inferring.

Professionally, the focus was on driving deeper appreciation about ethical AI, especially in high stakes domains such as healthcare. Protecting users and data while innovating was a recurring theme which influenced not only design decisions but also evaluation approaches.

6.4 Limitations and Future Work

Even though the chatbot prototype performed well on controlled clinical test cases, its application into the real world is constrained by the following facts:

- The lack of user trials and clinician feedback leaves its efficacy in real life applications puzzling.
- The dependency on proprietary models from OpenAI is a threat to internal thinking control and the system to unpredictable updates or output change.
- Rare diseases and multilingual and cultural diversity settings have not been tested by the model, therefore, its generalisability is limited.
- For the cures of these, the future developments can be directed towards the points below;
- There is a need of introducing in the higher-risk cases human in the loop mechanisms.
- Usability testings and clinical validations including.
- Using justifyable AI (XAI) approach to enhance transparency.
- Developing a low weight, deployable model appropriate for movable platforms or limited resource areas.

6.5 Final Remarks

Finally, this project shows that AI-based medical chatbots can actually deliver a meaningful support to users regarding symptom analysis and mental health conversations. Although tools such as the Medical Assistant Chatbot is not a clinical diagnosis substitute, they have the potentiality to enhance early triage, increase health awareness, and alleviate the informational load that health care systems experience.

The outcome acts as a promising step in the direction of the creation of significant, instruction friendly digital health tools. With additional refining, validation, and regulatory compatibility such systems have potential to be disruptive in providing accessible and ethical healthcare delivery.

Reference List

Nori, H., King, N., McKinney, S.M., Carignan, D. and Horvitz, E., 2023. *Capabilities of GPT-4 on Medical Challenge Problems*. arXiv preprint arXiv:2303.13375. Available at: https://arxiv.org/abs/2303.13375 [Accessed 1 March 2025].

OpenAI, 2024. *API Reference - OpenAI Platform*. [online] Available at: https://platform.openai.com/docs/api-reference [Accessed 1 March 2025].

Patel, V. and Lam, K., 2022. *Challenges and opportunities of using large language models in clinical settings*. *Journal of Medical Systems*, 46(12), pp.1–8. https://doi.org/10.1007/s10916-022-01833-5 [Accessed 9 May 2025].

Ravuri, S., Goyal, P. and Mohamed, A., 2023. *Engineering robust LLM APIs: Rate limiting, prompt management, and response quality*. arXiv preprint arXiv:2307.09523. Available at: https://arxiv.org/abs/2307.09523 [Accessed 1 March 2025].

Singhal, K., Azizi, S., Tu, T., Mahdavi, S.S., Wei, J., Chung, H., Scales, N., Tanwani, A., Cole-Lewis, H., Mathews, A., Seneviratne, M. and Google Health AI, 2023. *Large language models encode clinical knowledge*. *Nature*, 620(7972), pp.172–180. https://doi.org/10.1038/s41586-023-05881-4 [Accessed 9 May 2025].

Supabase, 2024. *What is Supabase?*. [online] Supabase Docs. Available at: https://supabase.com/docs [Accessed 1 March 2025].

Zhu, Z., Cao, H., Xu, K., Liu, Y., Luo, Y. and Yin, D., 2022. *API design for healthcare chatbots: Lessons from real-world systems*. In: *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3491102.3501927 [Accessed 4 March 2025].

Babylon Health (2022) *AI Health Service Performance Report*. Available at: https://www.babylonhealth.com (Accessed: 10 March 2025).

Bickmore, T., Trinh, H., Olafsson, S., O'Leary, T. and Asadi, R. (2018) 'Patient and consumer safety risks when using conversational assistants for medical information: An observational study of Siri, Alexa, and Google Assistant', *BMJ Health & Care Informatics*, 26(1), pp. 1–7. doi:10.1136/bmjhci-2018-000123.

Doshi-Velez, F. and Kim, B. (2017) *Towards a rigorous science of interpretable machine learning*. arXiv preprint. Available at: https://arxiv.org/abs/1702.08608 (Accessed: 10 March 2025).

Infermedica (2023) *Clinical Validation Report*. Available at: https://infermedica.com/resources/validation-report (Accessed: 29 April 2025).

Jiang, F., Jiang, Y., Zhi, H., Dong, Y., Li, H., Ma, S., Wang, Y., Dong, Q., Shen, H. and Wang, Y. (2017) 'Artificial intelligence in healthcare: past, present and future', *Stroke and Vascular Neurology*, 2(4), pp. 230–243. doi:10.1136/svn-2017-000101.

Kellermann, A.L. and Jones, S.S. (2013) 'What it will take to achieve the as-yet-unfulfilled promises of health information technology', *Health Affairs*, 32(1), pp. 63–68. doi:10.1377/hlthaff.2012.0693.

NHSX (2021) *Artificial Intelligence in Health and Care Award – Phase 1 Findings*. Available at: https://www.nhsx.nhs.uk (Accessed: 29 April 2025).

OpenAI (2023) *GPT-3.5 Technical Report*. Available at: https://openai.com/research/gpt-3-5 (8 February 2024).

Topol, E. (2019) *Deep Medicine: How Artificial Intelligence Can Make Healthcare Human Again*. New York: Basic Books.

World Health Organization (WHO) (2021) *Ethics and Governance of Artificial Intelligence for Health: WHO Guidance*. Geneva: World Health Organization. Available at: https://www.who.int/publications/i/item/9789240029200 (Accessed: 8 February 2024).

Health, . Ada Health: Symptom Checker and Health Assistant. *Ada Health Official Site*. [Online] 2021. https://ada.com.

Health, . AI Chatbots in Healthcare: From Diagnosis to Treatment. *Babylon Health Official Site*. [Online] 2021. https://www.babylonhealth.com/.

Fitzpatrick, K. K., et al. Delivering Cognitive Behavioral Therapy to Young Adults With Symptoms of Depression and Anxiety Using a Fully Automated Conversational Agent (Woebot). *JMIR Mental Health*. [Online] 2017. https://mental.jmir.org/2017/2/e19/.

Jiang, F., et al. Artificial intelligence in healthcare: Past, present, and future. *Stroke and Vascular Neurology*. [Online] 2021. https://svn.bmj.com/content/2/4/230.

Kumar, S., et al. AI-powered chatbots in healthcare: Enhancing user engagement and accessibility. *Journal of Medical Systems*. [Online] 2022. https://www.researchgate.net/publication/386233600_Chatbots_in_Healthcare_Enhancing_Accessibility_and_Patient_Engagement.

Larsen, M. E., et al. Digital Mental Health Tools for Pandemic Response. *Journal of Medical Internet Research*. [Online] 2021. https://www.jmir.org/2021/4/e26994/.

Razzaki, S., et al. A Case Study of AI-Powered Chatbots in Healthcare. *Healthcare Systems*. [Online] 2018.

Sharma, S., et al. Artificial Intelligence and the Future of Healthcare: Role of Chatbots. *Healthcare Informatics Research*. [Online] 2020. https://www.researchgate.net/publication/348633466_Artificial_Intelligence_in_the_Healthcare_Industry.

Topol, E. J. Deep Medicine: How Artificial Intelligence Can Make Healthcare Human Again. *Basic Books*. [Online] 2019. https://www.academia.edu/42042991/Deep_Medicine_How_Artificial_Intelligence_Can_Make_Healthcare_Human_Againpdf_by_Eric_Topol.

Lu Xu, Leslie Sanders, Kay Li, James C L Chow. PMC. *National Library of Medicine*. [Online] 2021. https://pmc.ncbi.nlm.nih.gov/articles/PMC8669585/.

Appendix A - Initial Project Proposal

Project (CN6000) Initial Proposal Form

Programme: Applied Computing (Top-Up) BSc (Hons). Year:2024/2025

Student Number:2545859

Proposed Title: Medical Assistant Chat Bot

Proposed Aim:
To develop a demo AI-powered medical assistant chatbot that provides users with accessible information regarding health care- basic symptom checking, medical report interpretation, and mental health support, which will improve healthcare accessibility for patients and improve patient engagement.

Rationale:
Artificial intelligence is revolutionizing every industry we know of; it's being implemented in almost every sector and is showing promising results in terms of productivity and efficiency. An AI-powered chatbot can be used in healthcare as it can take a load off the healthcare providers by answering frequently asked questions from the users. It can also work as a second opinion for a healthcare provider to brainstorm ideas with an AI-powered chatbot to help with the diagnosis based on a particular patient symptom. This project will showcase a demo AI chatbot and the impact it can have in healthcare

Supervisor: Shaheen Khatoon

Appendix B - Final Project Proposal

Project (CN60007) Final Proposal Form

Programme: Applied Computing (Top-Up) BSc (Hons). Year:2024-2025

Student Number: 2545859

Proposed Title:

Proposed Aim:
To develop a demo AI-powered medical assistant chatbot that provides users with accessible information regarding health care- basic symptom checking, medical report interpretation, and mental health support, which will improve healthcare accessibility for patients and improve patient engagement.

Objectives:

1. To conduct research on currently implemented AI conversational agents in the healthcare industry and their use cases.
2. To research currently published literature about AI application in the health care industry.
3. To conduct surveys on users who have already used or interacted with an artificial intelligence-powered medical assistance chatbot to gather feedback.
4. To analyze the surveys conducted with the quantitative research method and find areas of improvement.
5. To design and create a demo conversational AI agent that will take users symptoms as input and provide output for the best possible diagnosis.
6. To test the chatbot's functionality, its knowledge for medical-related inquiries, and its accuracy for responses.
7. To evaluate the test results and make improvements for the demo Medical Assistant AI chatbot.

Rationale:
Artificial intelligence is revolutionizing every industry we know of; it's being implemented in almost every sector and is showing promising results in terms of productivity and efficiency. An AI-powered chatbot can be used in healthcare as it can take a load off the healthcare providers by answering frequently asked questions from the users. It can also work as a second opinion for a healthcare provider to brainstorm ideas with an AI-powered chatbot to help with the diagnosis based on a particular patient symptom. This project will showcase a demo AI chatbot and the impact it can have in healthcare

Facilities required:

1. Open AI API KEY WITH CREDITS
2. MAC OS
3. Working Internet Minimum Internet speed 20 MBPS
4. Visual Studio
5. Vercel
6. Supabase

Supervisor: Shaheen Khatoon

Appendix C – GitHub Repository Link

https://github.com/Apocrophyn/Dissertation.git