Nile University School of Information Technology and Computer Science

Program of Artificial Intelligence

Medibot: Medical Diagnosis AI Assistant Using Computer Vision & Natural Language Processing

AIS496 Senior Project II

Submitted in Partial Fulfilment of the Requirements For the Bachelor's Degree in Information Technology and Computer Science Artificial Intelligence

Submitted By

Bahaa Eldin Moustafa 202000498

Mohamed Ahmed Mohamed 202000991

Abdelrahman El Atrozy 202001188

Mohamed Abd ElNasser 202000406

Ahmed Amr Helal 202000487

Lina Eyad 202000202

Supervised by

Dr. Mustafa EL Attar

Department of Artificial Intelligence

Department of Computer Science

Giza-Egypt

Spring 2024

Project Summary

In the contemporary healthcare landscape, the demand for quick and accurate medical diagnostics is more critical than ever, especially in the wake of global health crises. Medibot, an advanced Medical Diagnosis AI Assistant, leverages Computer Vision (CV) and Natural Language Processing (NLP) to enhance the efficiency and accessibility of medical consultations. The platform addresses key challenges in medical diagnostics by providing robust analysis of medical images and symptom-based consultations.

Medibot features an integrated suite of tools that include:

- Advanced neural networks for detailed analysis of chest X-rays, capable of identifying a wide range of conditions.
- Optical Character Recognition (OCR) to convert images of blood lab reports into text, followed by deep NLP to provide simplified and accurate interpretations.
- Llama models that manage symptom extraction, facilitate user interactions, and ensure accurate routing between various AI agents for comprehensive diagnostic support.

Accessible via a user-friendly web interface and a mobile application, Medibot is designed to streamline the initial stages of medical diagnostics, guiding patients quickly to appropriate medical advice and care. The system's architecture utilizes modern technologies such as Docker, Node.js, MongoDB, and Milvus to ensure reliability and scalability.

Medibot stands out by providing a highly interactive, intuitive, and efficient AI-driven diagnostic process, significantly enhancing patient engagement and outcomes in healthcare.

Abstract

Medibot revolutionizes medical diagnostics by integrating sophisticated Computer Vision and Natural Language Processing technologies to provide rapid, accurate, and accessible healthcare diagnostics. This AI-driven platform facilitates detailed medical image analysis and interactive symptom-based consultations, enabling efficient preliminary diagnostics and specialist referrals. With its comprehensive suite of diagnostic tools, including image classification, text extraction, and AI-based symptom analysis, Medibot enhances the diagnostic process, reduces the dependency on extensive medical consultations, and improves healthcare accessibility. Deployed via an intuitive web and mobile interface and built with cutting-edge technology, Medibot offers a transformative solution for early medical diagnosis, contributing significantly to improved healthcare outcomes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Healthcare systems around the world are consistently challenged by the need for more efficient and accessible care. Long wait times for appointments, limited access to specialists, and the complexity of medical diagnostics can lead to delays and suboptimal patient outcomes. With the advancements in artificial intelligence (AI), there is a significant opportunity to enhance the effectiveness of healthcare services. AI-powered tools, particularly those utilizing Computer Vision (CV) and Natural Language Processing (NLP), can significantly improve the speed and accuracy of medical diagnostics, making healthcare more accessible and reducing overall costs.

1.2 Motivation

The development of Medibot is inspired by the ongoing challenges within the healthcare sector and the potential of AI to transform it. Medibot is designed to:

- Enhance accessibility by providing a user-friendly platform for patients to engage with AI-driven diagnostic tools.
- Increase diagnostic efficiency through automated image analysis and symptom interpretation.

- Empower patients by providing clear, actionable health information, thereby facilitating informed decision-making.

These elements are crucial for improving healthcare delivery and patient outcomes.

1.3 Objectives

The primary objective of Medibot is to create a sophisticated AI assistant that can:

- Analyze medical images, particularly chest X-rays, using advanced CV techniques to detect a range of conditions.
- Interpret symptoms and medical histories using NLP to provide preliminary diagnoses and health insights.
- Seamlessly integrate these AI capabilities into an accessible interface for both web and mobile platforms, ensuring widespread usability.

1.4 Scope

The scope of Medibot includes:

- Developing and training specific CV and NLP models to handle tasks such as image classification and text analysis.
- Implementing a user-friendly interface that accommodates both technical and non-technical users.
- Establishing a referral system to guide users to the appropriate medical professional based on AI-generated diagnoses.

Excluded from the scope are real-time, comprehensive medical diagnostics and direct medical treatment recommendations.

1.5 Significance of the Study

The significance of Medibot extends beyond mere technological innovation; it represents a transformative shift in the healthcare paradigm. By integrating advanced AI capabilities, Medibot addresses several critical challenges in healthcare delivery, including accessibility, and diagnostic accuracy. The deployment of Computer Vision and Natural Language Processing technologies enables Medibot to perform comprehensive analyses of medical images and interpret complex symptom data, tasks traditionally reserved for highly trained medical professionals. This capability significantly reduces the strain on healthcare systems, expedites the diagnostic process, and allows for more timely medical interventions.

Moreover, Medibot democratizes health information, giving patients the tools to understand and articulate their health concerns more effectively. This empowerment is vital in regions with limited access to medical services, where Medibot can serve as a first line of inquiry, potentially identifying serious conditions earlier and directing patients to appropriate care.

The project also holds promise for scalability and adaptability, offering potential integration into diverse healthcare environments and systems worldwide. As such, Medibot not only enhances individual patient care but also contributes to the broader goals of global health initiatives aimed at improving general health outcomes and reducing healthcare disparities. The development and successful deployment of Medibot could serve as a model for future healthcare technologies, marking a significant milestone in the use of AI in medicine.

1.6 Outline of the Report

The structure of this report is designed to comprehensively cover all aspects of the Medibot project, providing a detailed view of the methodologies, technologies, and outcomes associated with the development of this innovative AI Assistant. The report is organized as follows:

- **List of Figures:** This section catalogs all figures included in the report. It is designed to facilitate quick reference to visual data and analytical representations used throughout the document, enhancing the reader's understanding of the discussed technologies and results.

- **List of Tables:** This section enumerates all tables presented in the report, providing easy navigation and reference. Each table is crucial for illustrating quantitative data and results derived from the project's research and testing phases.

- **Chapter 1: Introduction:** Introduces the Medibot project, outlining the background, motivation, objectives, scope, and significance of the study. This chapter sets the stage for a deeper exploration of the innovative technologies employed in Medibot and its potential impact on healthcare.

- **Chapter 2: Related Work:** This chapter surveys the relevant literature and historical context of AI in healthcare, particularly focusing on Computer Vision and Natural Language Processing technologies. It reviews previous research and studies that have contributed to the development of AI diagnostic tools, provides a market analysis of healthcare AI solutions, and describes the datasets used for training and evaluating Medibot.

- **Chapter 3: Methods:** This chapter outlines the systematic methods employed in the design and development of Medibot, emphasizing the architectural design and integration of Computer Vision (CV) and Natural Language Processing (NLP) technologies. It details the development environments used and discusses the creation of both a responsive web interface and a complementary Android application. These platforms enable the functional deployment of the AI assistant, ensuring accessibility and ease of use across different devices.

- **Chapter 4: Implementation & Results:** Details the practical application and operational capabilities of Medibot, showcasing real-world application scenarios and user interactions. This chapter also presents a comprehensive evaluation of the system's performance through quantitative and qualitative results.

- **Chapter 5: Discussion and Conclusion:** Analyzes the effectiveness of Medibot in improving diagnostic processes within healthcare. It discusses the challenges faced during development, summarizes the project's key findings, and explores potential future enhancements to enhance functionality and usability.

- **References:** Lists all bibliographic sources cited in the development and research of Medibot. This section provides a foundational trail for further investigation and verification of the study's underpinnings.

Chapter 2

Related Work

2.1 Introduction to literature review

The development of AI-powered diagnostic and decision support tools is rapidly transforming healthcare. Recent innovations in natural language processing (NLP) and computer vision (CV) demonstrate great potential in providing preliminary diagnoses, streamlining referrals, and analyzing medical images – directly addressing accessibility and patient empowerment challenges. However, this emergent field also reveals hurdles in effectively harnessing AI, notably regarding error handling, model transparency, and the need for rigorous testing before broader applications.

Current AI-based symptom checkers and early-stage diagnostic tools have gained popularity. Projects integrating language models (LMs) and medical knowledge graphs highlight the possibilities in this space ("Leveraging a Medical Knowledge Graph...", "ClinicalGPT ...", "Med-HALT...") These works focus on improving NLP performance for medical contexts but also underscore the challenge of combating 'hallucinations' (errors generated by LLMs). Addressing these inaccuracies through techniques like fine-tuning or prompting while preserving reliability for clinical use remains a vital research focus.

In the realm of medical image analysis, convolutional neural networks (CNNs) are revolutionizing diagnosis, exemplified by their ability to detect pneumonia from chest

X-rays with an accuracy that rivals specialists ("CheXNet..."). Furthermore, deep learning approaches to automate the segmentation of anatomical regions, such as the lungs, ("Automatic Lung Segmentation...") improve workflow efficiency.

The development of referral systems reliant on AI is a further promising area. These systems could automate patient triage and reduce burdens on healthcare providers. To fully realize the potential of these applications, it is imperative to establish rigorous evaluation mechanisms alongside explainable outputs that support trustworthy and transparent decision-making.

This literature review will provide a critical analysis of state-of-the-art AI solutions in areas closely related to the Medibot project. Key focuses will include:

- Approaches to AI-powered symptom analysis and preliminary diagnosis
- Computer vision applications for medical image analysis
- AI-based patient referral systems
- Crucial areas of concern such as bias, lack of transparency, and potential errors

The goal of this review is to position Medibot within the existing landscape, establish best practices, and illuminate areas where the project can contribute to this rapidly evolving field of healthcare innovation.

2.2 Historical Perspective

While recent advances might make it seem like a burgeoning field, the concept of harnessing computers for medical diagnosis dates back to the 1970s with the advent of expert systems such as MYCIN. These rule-based systems relied on the encoding of medical knowledge and logical inferences but faced limitations in scalability and the ability to handle uncertainty.

A subsequent shift towards statistical and machine learning approaches fueled progress, but computational bottlenecks initially constrained wide adoption. Developments in

neural networks during the 1990s set the stage, but it wasn't until the early 2010s, coinciding with the advent of deep learning and vast improvements in computing power, that AI began to see breakthroughs in medical image analysis.

Significant achievements, such as the success of convolutional neural networks (CNNs) in large-scale image classification (e.g., ImageNet), spurred research into adapting these techniques for diagnosing conditions like pneumonia from chest X-rays. In parallel, natural language processing (NLP) has progressed significantly. While earlier NLP focused on rule-based parsing, new models leveraging word embeddings and neural networks enabled better contextual understanding, leading to more nuanced approaches to symptom analysis.

Recent years have witnessed a confluence of factors propelling healthcare AI research. Massive datasets, cloud computing, and advanced language models (e.g., GPT-3) have enabled a new wave of AI-powered diagnostic assistants and decision-support tools. Despite their promise, it's important to note that healthcare AI remains a field grappling with potential risks including bias, hallucinations, and the need for rigorous, real-world testing before wide-scale implementation.

2.3 Current State of the Art

The field of AI-powered medical diagnosis is evolving rapidly, with several approaches offering promising results:

- **Symptom Checkers and Preliminary Diagnosis:** Systems like Ada Health, Healthily, Sensely, and Infermedica illustrate the growing reliance on AI-guided questionnaires and chatbots for symptom analysis. Despite this progress, limitations exist – they often lack open conversation capabilities, cannot process medical images, and primarily address diagnoses in singular domains without multi-stage support.

- **Medical Image Analysis:** Convolutional neural networks (CNNs) are transforming medical image analysis. The success of systems like CheXNet in diag-

nosing pneumonia showcases their potential for accurate disease detection from medical images. However, accurate segmentation of regions of interest (like the lungs) can be challenging, especially in cases with abnormalities or occlusions.

- **Natural Language Processing (NLP):** The recent advent of large language models (LLMs) has revolutionized natural language understanding. These models are increasingly demonstrating capability for medical tasks when finetuned and enhanced with domain-specific knowledge. While showing progress in extracting information from medical records or research papers, handling the subtleties of patient-reported experiences remains a challenge.

2.4 Medibot's Contributions

The Medibot project aims to address several of the limitations observed in existing AI-powered diagnostic solutions. Key potential innovations include:

- **Flexible NLP Input:** Medibot will process both open-ended symptom descriptions and engage in questionnaire-style interaction, catering to diverse preferences and providing additional avenues for understanding the patient's concerns.
- **Medical Image Integration:** Image analysis with CV techniques will enhance Medibot's diagnostic capabilities while helping streamline workflows for users who have the ability to upload relevant images.
- **Adaptive Referral System:** Medibot will not simply deliver isolated diagnoses but rather intelligently guide patients to suitable specialists or appropriate departments for further care, improving outcomes by fostering continuity of care.

Chapter 3

Methods

3.1 Software System

This section details the comprehensive software architecture of Medibot, highlighting the tools and technologies employed in the development of both the web interface and the mobile application. The system is designed to provide robust, scalable, and user-friendly platforms for patients and healthcare professionals, facilitating efficient symptom input, medical image uploads, and lab report sharing.

3.1.1 System Description

Medibot is designed to bridge the gap between initial symptom analysis and professional healthcare guidance. It supports both patients and healthcare professionals in the context of diagnosis and decision-making.

Patient Perspective:

- **Symptom Input:** Patients engage with Medibot by providing descriptions of their symptoms either through natural language queries or structured interaction.
- **Image Upload:** Patients can upload medical images, such as X-rays, for integration into Medibot's diagnostic analysis.

- **Lab Report Sharing:** Patients can share existing lab reports, enhancing Medibot's ability to contextualize and understand their health condition.
- **Preliminary Guidance:** Medibot provides initial diagnostic considerations, identifies areas of concern, and suggests appropriate medical specialties or next steps.

Healthcare Professional Perspective:

- **Consultative Resource:** Doctors and healthcare providers use Medibot as a supportive decision-making tool in their diagnostic processes.
- **Patient Contextualization:** Medibot offers insights derived from patient-reported symptoms, uploaded images, and lab reports, aiding healthcare professionals in assessments.
- **Efficiency Tool:** Medibot helps streamline the analysis of initial symptoms and identification of potential conditions, assisting doctors in triaging and managing cases more effectively.

Important Notes:

- **Direct Communication:** Medibot does not enable direct data exchange or communication between patients and doctors.
- **Emphasis on Support:** Medibot serves as an assistive tool, not a replacement for professional medical diagnosis.

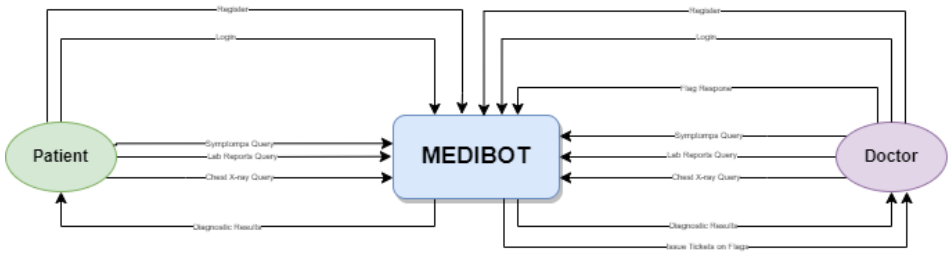

Figure 3.1: System Context Diagram

3.1.2 System Boundaries

The Medibot system establishes well-defined boundaries to encapsulate its essential features and interactions with the external environment.

Core components within the System

- **Natural Language Processing (NLP) Engine:**
  - Analyzes patient-provided symptom descriptions (text or potentially voice in the future).
  - Processes medical text from lab reports.
  - Generates preliminary diagnoses and potential health concerns.
- **Computer Vision (CV) Models:**
  - Analyze uploaded medical images (e.g., X-rays) for abnormalities.
  - May segment areas of interest (e.g., heart, lungs).
- **Referral System:**
  - Matches potential diagnoses/concerns with relevant medical specialties.
  - Provides guidance on appropriate next steps (e.g., specialist consultation, further testing).
- **Knowledge Base:**
  - Curated repository of medical information and diagnostic criteria.
  - Supports inference and recommendations across the other components.
  - Incorporates Retrieval-Augmented Generation (RAG) systems that enhance the AI's response capabilities by fetching relevant data and documents dynamically during the inference process.

Interface Components within the System

- **Mobile Application:**
  - User-friendly design optimized for smaller screens.
  - Handles symptom input, image uploads, and output display.
  - Built in apk format for Android devices.
- **Web Application:**
  - Offers a more expansive input area for larger text descriptions.
  - Might handle large image uploads more efficiently.
  - Built in a user-friendly web interface format.

Key Points at the Boundaries

- **Patient Inputs:**
  - Direct symptom descriptions.
  - Xray Image uploads.
  - Sharing existing lab reports as Images.
- **Medibot Outputs:**
  - Preliminary diagnoses or highlighted concerns.
  - Guidance and recommendations for follow-up actions.
  - Referral suggestions to medical specialties.
- **Interactions with Healthcare Professionals:**
  - Doctors may refer to Medibot's analyses as supplementary information during assessment (not primary data exchange).

Outside System Boundaries

- **Direct Patient-Doctor Communication:** Medibot does not facilitate direct messaging or real-time consultations.
- **Appointment Scheduling:** Integration with hospital systems for booking appointments is currently out of scope.
- **Treatment Prescriptions:** Medibot avoids generating explicit treatment plans or medication recommendations.
- **Emergency Situations:** Medibot is not designed to replace qualified emergency medical assistance.

3.1.3 Objectives and Requirements

Objectives

The Medibot AI Assistant aims to fulfill the following core objectives:

- **Accurate Medical Guidance:** Provide reliable medical advice grounded in the analysis of user-reported symptoms.
- **Streamlined Referrals:** Facilitate effective patient navigation by suggesting appropriate healthcare professionals or relevant medical departments.
- **Enhanced User Experience:** Deliver a seamless and intuitive experience for users throughout their interaction with the system.

Requirements

To meet these objectives, Medibot must satisfy the following user-driven requirements:

- **Natural Language Input:** Enable users to describe their symptoms using everyday language, allowing for greater flexibility and comfort.
- **Timely and Relevant Advice:** Provide prompt responses with medical advice tailored to the user's reported symptoms and concerns.

Desired Characteristics

Beyond requirements, Medibot strives to attain the following characteristics that reinforce its value as an assistive tool:

- **Accuracy:** Consistently provide dependable advice through the utilization of robust models and up-to-date medical knowledge.
- **Reliability:** Deliver a steady and predictable user experience, promoting trust.
- **User-friendliness:** Craft an intuitive interface that guides inexperienced users throughout their interaction with the system.

3.1.4 System Requirements

Use Case Diagram

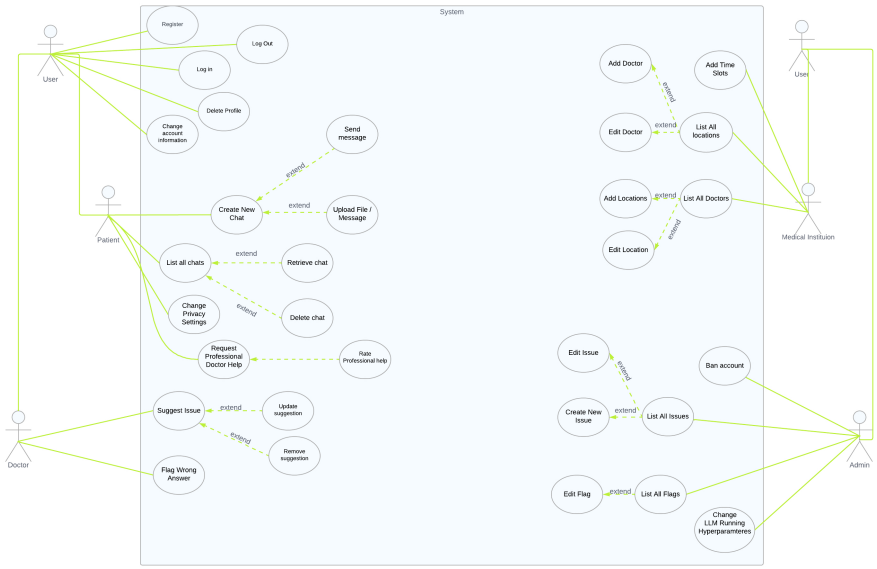

Figure 3.2: System Context Diagram

General Functionality

Medibot enables users to input symptoms, receive medical advice, and obtain referrals to appropriate healthcare professionals or departments. Project requirements include

symptom analysis, integration with medical databases for diagnosis prediction, and seamless communication with users through text.

Software Interfaces

The system interfaces with users through both an Android mobile application and a Website. These user-friendly interfaces allow users to input symptoms and receive medical advice. Additionally, the system interacts with medical databases for accessing patient records and diagnostic information.

Application Programming Interface (API)

Medibot uses API to integrate its functionalities across all its modules in a single interface. The API includes functions for symptom analysis, diagnosis prediction, and image processing.

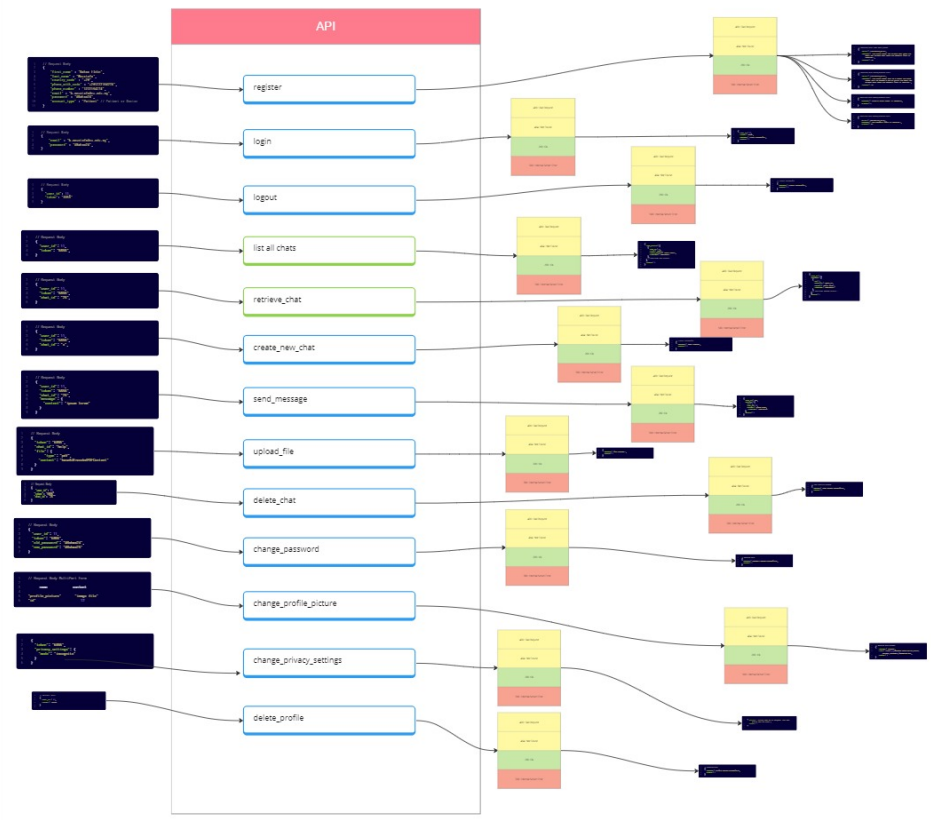

Figure 3.3: API Endpoints

Functional Requirements

FR 1: Register

- **Actors:** Patient or Doctor
- **Description:** The system shall allow users (both students and instructors) to create accounts. The registration process shall securely collect the following information: username, password, and other required details.

FR 2: Login

- **Actors:** Patient or Doctor
- **Description:** Users shall be able to log in to the system using their registered credentials.

FR 3: Logout

- **Actors:** Patient or Doctor
- **Description:** Users shall be able to log out of their current session securely.

FR 4: Create New Chat

- **Actors:** Patient or Doctor
- **Description:** Users shall be able to initiate new chat conversations.

FR 5: Send Message

- **Actors:** Patient or Doctor
- **Description:** Users shall be able to send text messages within an active chat conversation.

FR 6: Retrieve Chat

- **Actors:** Patient or Doctor

- **Description:** Users shall be able to view existing chat conversations.

FR 7: List All Chats

- **Actors:** Patient or Doctor
- **Description:** Users shall be able to view a list of their active and/or past chat conversations.

FR 8: Upload X-ray Image

- **Actors:** Patient or Doctor
- **Description:** Users shall be able to upload medical X-ray images for analysis.

FR 9: Upload Lab Report Image

- **Actors:** Patient or Doctor
- **Description:** Users shall be able to upload medical images of lab reports for analysis.

FR 10: Upload Lab Report PDF

- **Actors:** Patient or Doctor
- **Description:** Users shall be able to upload medical files of lab reports for analysis.

FR 11: Change Password

- **Actors:** Patient or Doctor
- **Description:** Users shall be able to securely update their account password.

FR 12: Change Email

- **Actors:** Patient or Doctor
- **Description:** Users shall be able to modify their registered email address.

FR 13: Delete Chat

- **Actors:** Patient or Doctor
- **Description:** Users shall have the ability to delete specific chat conversations from their history.

FR 14: Delete Profile

- **Actors:** Patient or Doctor
- **Description:** Users shall have the ability to permanently delete their account and associated data from the system.

Non-Functional Requirements

Performance

- **NFR 1: Response Time**
  - **Description:** Medibot shall generate preliminary diagnoses within 5 seconds of symptom submission for 90% of user queries.
  - **Priority:** High
  - **Metrics:** Measure average and percentiles (90th, 95th) of response time in testing and under varying load conditions.
- **NFR 2: Image Upload and Processing**
  - **Description:** Uploaded medical images (within permissible file size restrictions) should be processed for analysis within 30 seconds.
  - **Priority:** Medium
  - **Metrics:** Track average and worst-case processing time, especially for large or complex image types.

Usability

- **NFR 3: Interface Design**
  - **Description:** Medibot shall feature a clear, intuitive interface, prioritizing readability and ease-of-navigation across platforms (web, mobile).
  - **Priority:** High
  - **Metrics:** Usability testing (time-on-task, errors); user satisfaction surveys.

Reliability

- **NFR 4: Availability**
  - **Description:** Medibot shall maintain an uptime of at least 99% during expected operational hours.
  - **Priority:** High
  - **Metrics:** Track uptime percentage; establish a clear monitoring and incident response plan.
- **NFR 5: Error Handling**
  - **Description:** Medibot shall anticipate and gracefully handle various error scenarios, both user-generated (e.g., invalid inputs) and system-side (e.g., network issues, model unavailability).
  - **Priority:** High
  - **Metrics:**
    * Track the frequency and types of errors encountered during testing and live use.
    * Measure user completion rates for core tasks, as excessive errors undermine these.
    * Aim for minimal unhandled exceptions leading to system crashes.

Safety and Ethical Considerations

- **NFR 6: Emphasis on Assistance**
  - **Description:** The system shall transparently reinforce its limits as a decision support tool and consistently redirect users to seek qualified medical professionals for diagnosis and treatment.
  - **Priority:** Critical

Use Cases

Table 3.1: Register

Field	**Description**
Actors	• User
Main Success Scenario	• User accesses the registration page. • User provides necessary information (email, password, etc.). • User submits registration form. • System validates the information. • System creates a new account and stores the user's information.
Exceptions	• Required information is missing. • Email is already taken.
Actions	• System prompts the user to provide missing information. • System prevents registration until all required fields are filled. • System notifies the user that the email is already taken. • User is prompted to choose a different email.
Pre-condition	• User has access to the registration page.
Post-condition	• User's account is successfully registered.

Table 3.2: Login

Field	**Description**
Actors	• User
Main Success Scenario	• User accesses the login page. • User enters email and password. • User submits login credentials. • System verifies the credentials. • User is redirected to the dashboard upon successful login.
Exceptions	• Invalid credentials provided.
Actions	• System displays an error message for invalid credentials.
Pre-condition	• User has access to the login page.
Post-condition	• User is logged in and redirected to the dashboard.

Table 3.3: Logout

Field	**Description**
Actors	• User
Main Success Scenario	• User clicks on the logout option. • System logs the user out and redirects to the login page.
Pre-condition	• User is logged in.
Post-condition	• User is logged out and redirected to the login page.

Table 3.4: Delete Profile

Field	**Description**
Actors	• User
Main Success Scenario	• User navigates to the profile settings section. • User selects the option to delete their profile. • System prompts for confirmation. • User confirms deletion. • System deletes the user's account and associated data.
Pre-condition	• User is logged in.
Post-condition	• User's account and associated data are permanently deleted.

Table 3.5: View Chat History

Field	**Description**
Actors	• User
Main Success Scenario	• User navigates to the chat history section. • System retrieves and displays the user's chat history. • User selects a specific chat session to view the conversation transcript.
Pre-condition	• User is logged in.
Post-condition	• User's chat history is displayed.

Table 3.6: Delete Chat History

Field	**Description**
Actors	• User
Main Success Scenario	• User navigates to the chat history section. • User selects the option to delete chat history. • System prompts for confirmation. • User confirms deletion. • System deletes the user's chat history.
Pre-condition	• User is logged in.
Post-condition	• User's chat history is permanently deleted.

Table 3.7: User Initiated Chat

Field	**Description**
Actors	• User
Main Success Scenario	• User sends a "Hello" message to the chatbot. • Chatbot responds with a friendly greeting. • Chatbot provides a list of suggestions and a 3D model of the human body for user selection.
Pre-condition	• User is logged in.
Post-condition	• User successfully initiates a chat with the chatbot.

Table 3.8: Patient Symptom Description

Field	**Description**
Actors	• Patient
Main Success Scenario	• Patient describes their symptoms to the chatbot in English. • Chatbot asks relevant questions to gather more details about the symptoms.
Pre-condition	• Patient is logged in.
Post-condition	• Chatbot gathers detailed information about the patient's symptoms.

Table 3.9: Doctor Question Asked

Field	**Description**
Actors	• Doctor
Main Success Scenario	• Doctor asks a question in English. • Chatbot provides related advice and details about the question.
Pre-condition	• Doctor is logged in.
Post-condition	• Doctor receives relevant advice and details from the chatbot.

Table 3.10: Medical Terminology

Field	**Description**
Actors	• Doctor
Main Success Scenario	• Doctor requests information on medical terminology or abbreviations. • Chatbot provides clear and accurate explanations for medical terms.
Pre-condition	• Doctor is logged in.
Post-condition	• Doctor receives clear and accurate explanations for medical terms.

Table 3.11: Patient Symptom Description in Simple Terms

Field	**Description**
Actors	• Patient
Main Success Scenario	• Patient describes their symptoms to the chatbot. • Chatbot provides clear and accurate explanation in simple language without using complex words.
Pre-condition	• Patient is logged in.
Post-condition	• Patient receives clear and simple explanation of their symptoms.

Table 3.12: Image Upload

Field	**Description**
Actors	• User
Main Success Scenario	• User uploads an X-ray image of their chest. • System confirms receipt of the uploaded image.
Pre-condition	• User is logged in.
Post-condition	• Image is successfully uploaded for analysis.

Table 3.13: Image Analysis Report

Field	**Description**
Actors	• User
Main Success Scenario	• System analyzes the X-ray image. • System generates a report with findings and recommendations based on the analysis.
Pre-condition	• User is logged in.
Post-condition	• User receives a detailed report with findings and recommendations.

3.1.5 Research Design

Plan and Strategy

The research plan involves collecting medical data, training machine learning models, and evaluating the performance of the AI Assistant. The strategy includes data collection, preprocessing, model training, and validation.

Research Methods

Research methods include data collection from medical databases, augmentation of datasets, training of machine learning models using deep learning techniques, and evaluation of model performance through metrics such as accuracy and precision.

Qualitative Research

- **Usability Observation:** Careful observation of real-world interactions with Medibot to identify usability pain points, areas where guidance is unclear, and non-verbal cues that signal positive or negative experiences.
- **Rationale:** The qualitative component enables exploration of individual perspectives, identifying themes and nuances crucial for interpreting user perceptions and barriers to effective implementation. This ensures we design not just for efficiency but also for genuine understanding and trust, particularly important for AI-based systems in healthcare.

Quantitative Research

- **System Logs & Metrics:**
  - **CNN Performance:** Analyze accuracy, sensitivity, and specificity of the image classification CNN on relevant diagnostic tasks. Compare these against established benchmarks in medical image analysis, if applicable. Break down potential performance variation across symptom classes or imaging modalities (X-ray, etc.).

- **LLM Evaluation:** Use quantitative metrics like perplexity, BLEU score, or custom accuracy measures based on tasks such as symptom recognition, potential diagnosis suggestion, or the quality of explanations generated by the LLM. Measure how these metrics vary with differing amounts of training data.

- **Rationale:**
  - **System Logs & Metrics:**
    - **CNN's:** Performance metrics specific to image classification quantitatively assess the potential reliability and robustness of Medibot for image-based diagnosis. This evaluation will reveal strengths, weaknesses, and areas needing improvement within the specific CNN architecture.
    - **LLM's:** Metrics will demonstrate the capability of the LLM for understanding medical text, extracting information, and generating relevant responses. By measuring performance variation based on training data size, we gain insights into the scalability and adaptability of the language model components.

3.1.6 Architectural Design

Overall Architecture

The system architecture comprises frontend, backend, and database components. The frontend provides the user interface for symptom input and medical advice display, while the backend handles symptom analysis, diagnosis prediction, and referral generation. The database stores user data and medical information.

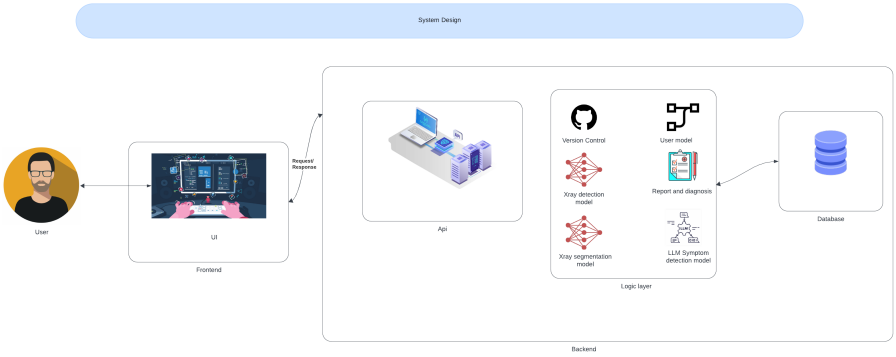

Figure 3.4: Overall System Architecture.

Frontend

The frontend consists of a web interface and a mobile application. The web interface allows users to access the AI Assistant from a web browser, while the mobile application provides a convenient and portable platform for interacting with the system.

Backend

- **Logic**
  - **Symptom Analysis:** The business layer processes user-provided symptoms, analyzes them, and generates a list of potential medical conditions.
  - **Diagnosis Prediction:** Based on the symptom analysis, the system predicts potential diagnoses and provides relevant medical advice.
  - **Referral Generation:** If necessary, the system generates referrals to specialized healthcare providers based on the predicted diagnosis.
- **API**
  - **User Management:** Manages user authentication, profile information, and privacy settings.
  - **Chatbot Interaction:** Handles user interactions with the chatbot, including symptom input, medical advice, and referral generation.

- **Image Analysis:** Processes uploaded medical images, performs analysis, and generates diagnostic reports.

- **Data Layer**
  - **User Data:** Stores user profiles, chat history, and feedback.
  - **Medical Data:** Contains information about medical conditions, treatments, and referrals.
  - **Image Data:** Stores uploaded medical images and diagnostic reports.

3.1.7 Data Design

Data Collection

Dataset	Name	No. of Samples	Year	
disease datasets	nhs.uk	436	2023	
	nhsinform	330	2023	
	medlineplus	4463	2023	
Image Datasets	chest-xray-pneumonia	5863	2021	
	CheXpert	224k	2019	
	NIH Chest X-rays	112k	2018	
	CoronaHack	3124	2020	
	eurorad	5228	2022	
Medical Reports Dataset	EMRBots	107K	2018	

Figure 3.5: Data Collection Process.

Textual Data Collection The textual data for conditions and treatments was collected from the NHS (National Health Service) website, specifically from the section dedicated to medical conditions. This section provides comprehensive information about various health conditions, including their symptoms, causes, treatments, and other relevant details.

Efforts were made to systematically gather information from multiple pages within this section of the NHS website. This involved extracting structured data about different medical conditions, ensuring accuracy and consistency in the dataset.

By utilizing reputable sources such as the NHS website, we aim to ensure the reliability and credibility of the collected data, thereby enhancing the effectiveness and validity of the Machine Learning (ML) and Large Language Model (LLM) models trained on this dataset.

Image Data Acquisition The chest X-ray images used in this study were obtained from two primary sources: the CheXpert dataset and the NIH Chest X-ray dataset (CXR14).

The CheXpert dataset is a large publicly available dataset consisting of chest radiographs and associated radiological reports. It contains a diverse range of chest X-ray images with annotations for various abnormalities, allowing for the training and evaluation of machine learning models.

The NIH Chest X-ray dataset (CXR14) is another widely used dataset comprising chest X-ray images collected from the National Institutes of Health Clinical Center. It provides a substantial collection of chest X-ray images with associated labels, enabling researchers to develop and validate algorithms for various medical imaging tasks.

Images from both datasets were collected in various file formats and resolutions, reflecting the real-world variability encountered in medical imaging. Preprocessing techniques were applied to standardize the images and enhance their quality before further analysis.

Textual Data Transformation To enhance the utility and accessibility of the textual data collected, a comprehensive transformation process was employed. This involved two main steps, each leveraging the collective capabilities of several Large Language Models (LLMs) to reduce model bias and improve the diversity of the data interpretations.

- **Symptom Summarization** The first step in the transformation process was to distill complex medical descriptions from the collected datasets into concise bullet points. These summaries are crucial for quick reference within the Medibot system and for ensuring that users receive straightforward and digestible medical information. We utilized a blend of advanced LLMs, including GPT-4

and Llama3-70b, to achieve a balance of accuracy and clarity, minimizing bias by integrating multiple model insights.

- **Generation of Interactive Queries** Following the summarization, the next step was to create dynamic and interactive content that would facilitate a naturalistic interaction between users and Medibot. This involved generating potential questions that patients might ask based on their symptoms, as well as crafting statements that patients could use to describe their symptoms accurately to a doctor. For this task, a combination of LLMs including GPT-3.5, Llama3-70b, and the Mixture of Experts (MoE) model Mixtral22b*8 were used. This ensemble approach ensured that the generated questions and descriptions were varied, realistic, and free from the biases typically associated with single-model outputs.

By integrating multiple LLMs at each stage of the textual data transformation process, we mitigated the risks of model bias, ensuring that the outputs were nuanced and broadly applicable. This methodology supports Medibot's goal to provide reliable and user-friendly diagnostic interactions.

Data Augmentation for Image Data To ensure that our AI models are robust and capable of handling real-world variability in medical imaging, we applied a comprehensive data augmentation strategy to our image dataset. This approach involved simulating a variety of conditions that images might undergo in practical settings, particularly those captured using mobile devices.

- **Augmentation Techniques Implemented** We employed a diverse set of image transformations to mimic various photographic effects and artifacts. These included:
  - **Moire Patterns** to simulate the interference effects seen in digital images.
  - **Blur** to replicate out-of-focus images.
  - **Motion Blur** to mimic the effect of camera movement during exposure.
  - **Glare** (both matte and glossy) to represent light reflections on different surfaces.

  - **Tilt** for angular misalignments.
  - **Brightness Adjustments** (both increases and decreases).
  - **Contrast Adjustments** (both increases and decreases).
  - **Rotation and Translation** to simulate changes in camera orientation and position.
  - **Exposure Variation** to mimic under and over-exposed images.

- **Application of Augmentations** To integrate these augmentations effectively, we randomly applied between one to five of these filters to 70% of the images in our dataset. This method allowed us to create a range of scenarios, from minor alterations to more severely altered images, thereby preparing our models to perform reliably under various photographic conditions.

- **Objective and Benefits** The objective of implementing these diverse augmentation techniques was to train our AI models to recognize and interpret medical images accurately, regardless of the quality or source of the image. This is particularly crucial for images taken from mobile devices in less controlled environments. As a result, our models demonstrate enhanced adaptability and reduced susceptibility to overfitting, ensuring reliable diagnoses even from sub-optimal image inputs.

3.1.8 Interaction Design

User Interaction

Users interact with the system through a user-friendly interface, providing symptoms and receiving medical advice and referrals. Design choices for user interactions focus on simplicity, clarity, and responsiveness.

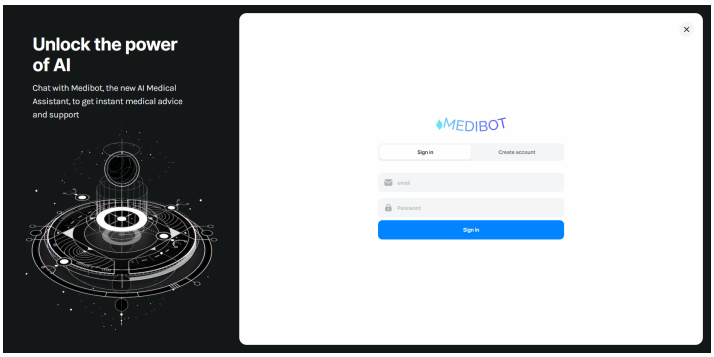

Figure 3.6: Web Login Page.

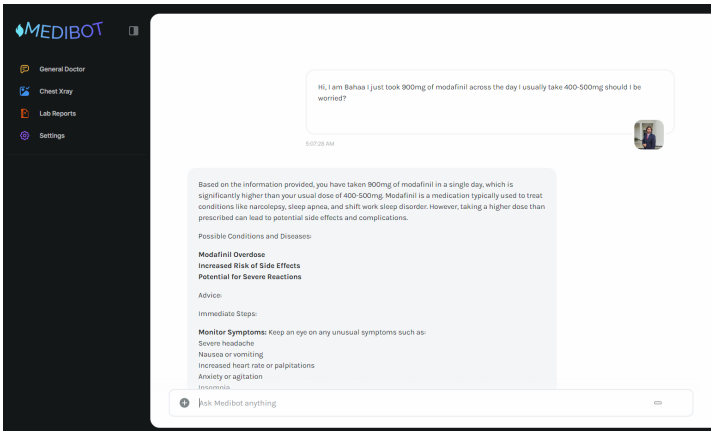

Figure 3.7: Web Chat Page.

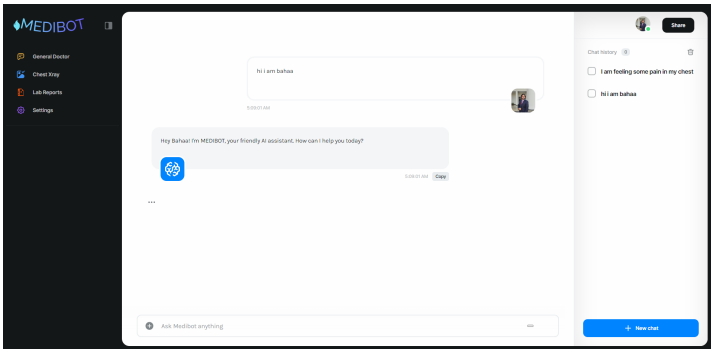

Figure 3.8: Web Chat History Page.

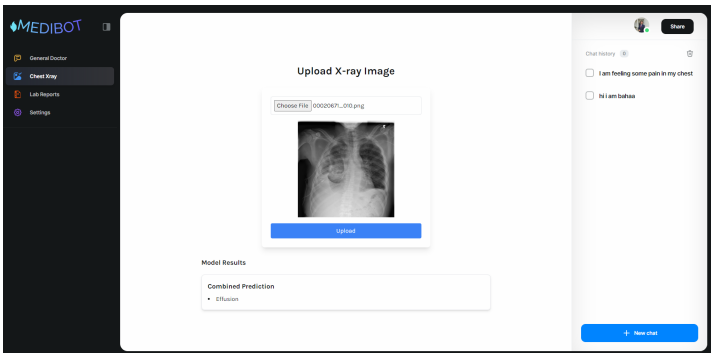

Figure 3.9: Web Xray Diagnosis Page.

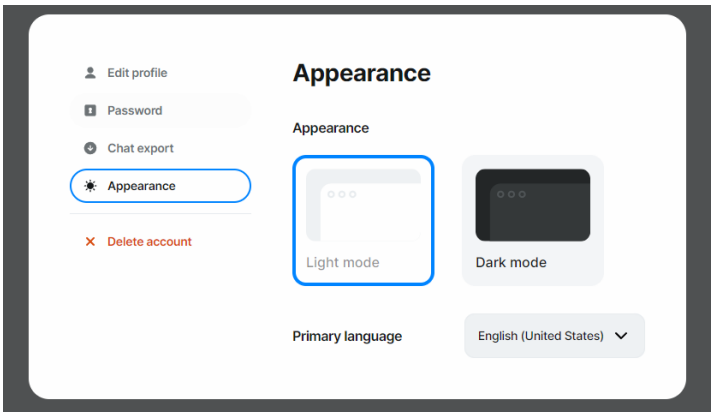

Figure 3.10: Web Settings Page.

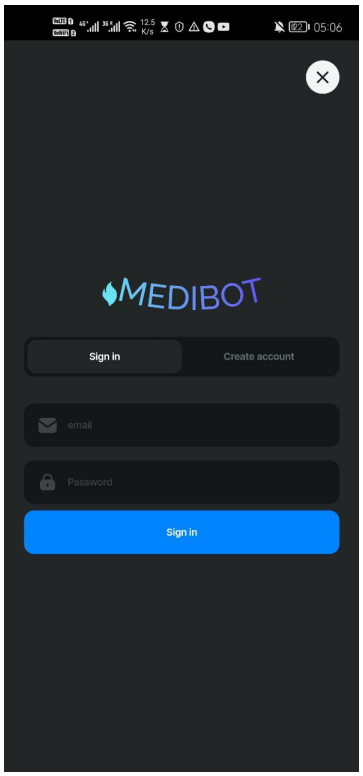

Figure 3.11: Mobile Login Page

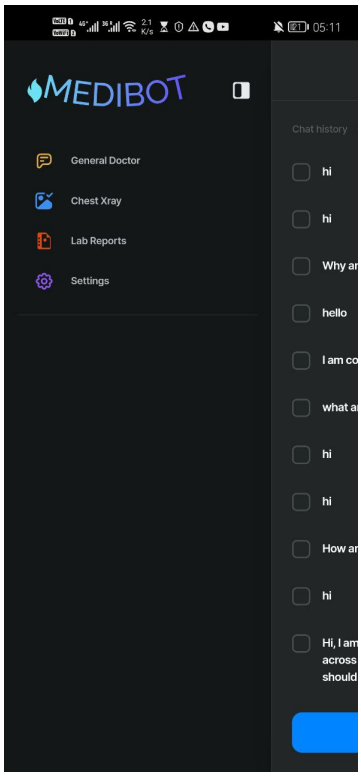

Figure 3.12: Mobile Menu Page

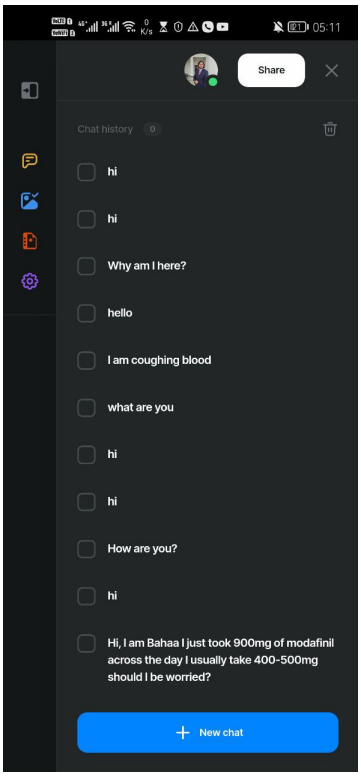

Figure 3.13: Mobile Chat History Page

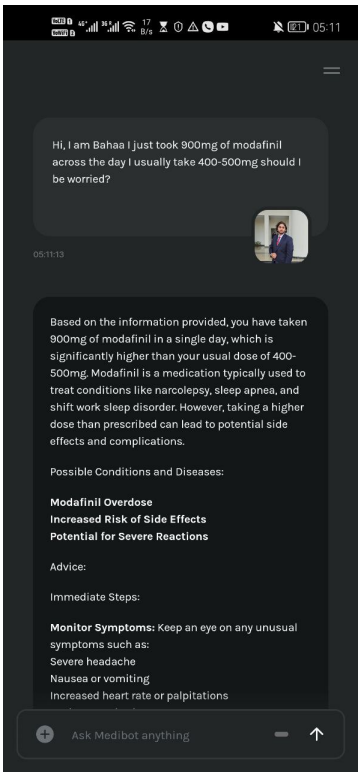

Figure 3.14: Mobile Chat Page

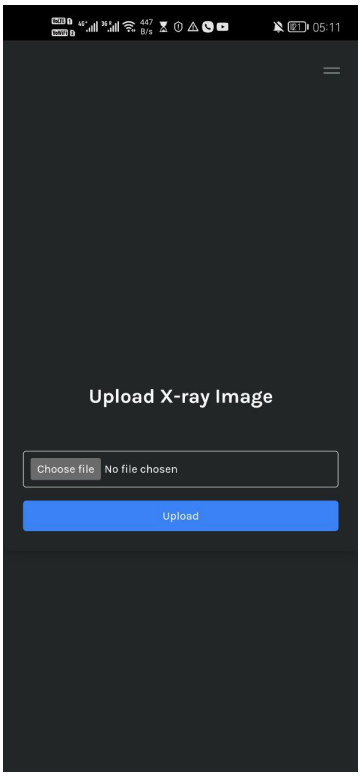

Figure 3.15: Mobile Xray Page

3.1.9 Data Flow Diagrams

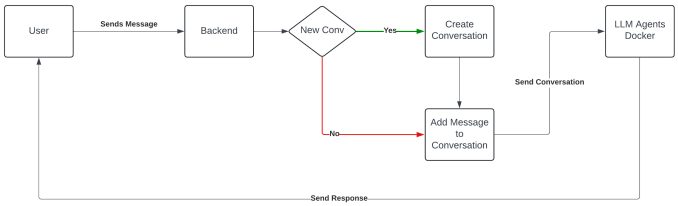

Figure 3.16: Data Flow for Chat.

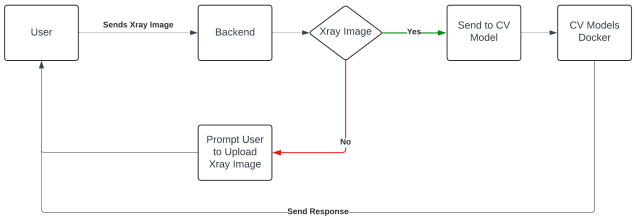

Figure 3.17: Data Flow for Xray Analysis.

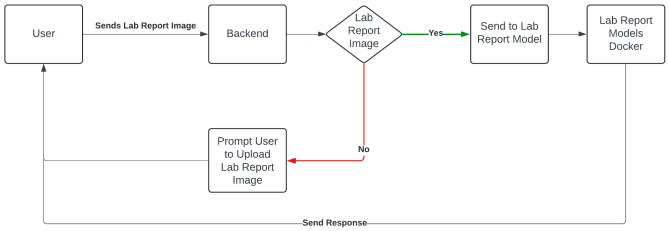

Figure 3.18: Data Flow for Lab Report Analysis.

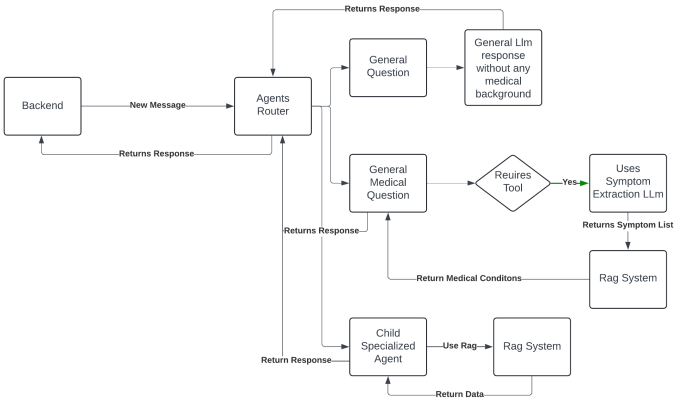

Figure 3.19: Data Flow for LLM Agents System.

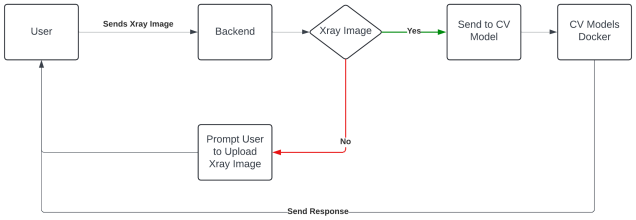

Figure 3.20: Data Flow for Xray System.

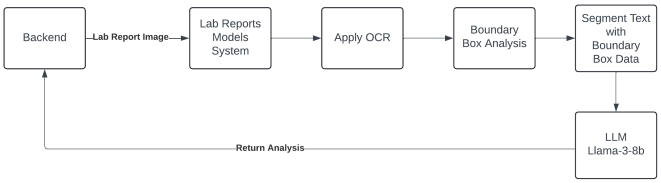

Figure 3.21: Data Flow for Lab Reports System.

3.1.10 Tools and Technologies

Software Frameworks

The development of Medibot utilized several popular software frameworks and libraries, including:

- **Python and FastAPI:** Employed throughout the backend development for its robust libraries and simplicity. FastAPI was specifically chosen for building high-performance APIs for the LLM agents, offering easy asynchronous support and scalability.

- **Node.js:** Served as the runtime environment for the entire system's backend, enabling asynchronous event-driven programming that supports non-blocking I/O operations. This was crucial for managing multiple user interactions smoothly.

- **PyTorch and LLM Finetuning:** Utilized for training and finetuning Computer Vision (CV) models and Large Language Models (LLMs), leveraging its flexible and dynamic computing architecture. Techniques like PEFT, LoRA, and Accelerator were used to optimize model performance for specific tasks within Medibot.

- **ReactJs and Capacitor:** ReactJs was used to develop the web interface, maximizing efficiency and user experience with its modular component structure. Capacitor allowed the transformation of the ReactJs-based web application into a native Android app, facilitating a unified codebase for both web and mobile platforms.

- **MongoDB:** Selected as the primary NoSQL database for storing and managing diverse data types such as chat messages, user profiles, and interaction logs, due to its flexibility and powerful querying capabilities.

- **Milvus:** Utilized for managing vector database storage, supporting advanced retrieval capabilities necessary for effectively handling large-scale vector data within Medibot.

- **GitHub:** Used as the central platform for managing the codebase, tracking changes, and facilitating collaboration among team members. It was pivotal in maintaining version control and supporting the development process.

- **Hugging Face:** Employed for accessing pre-trained models and datasets, as well as for saving and managing our own models during development, which facilitated the seamless integration of state-of-the-art language models into our system.

- **Docker:** Utilized for containerizing the AI models and backend services, ensuring consistency across different environments and simplifying deployment and scaling processes.

- **Kubernetes:** Employed for orchestrating the deployment of containerized services, enabling efficient scaling and management of the system's components in a cloud environment.

- **Figma:** Figma was used for creating wireframes, mockups, and interactive prototypes of the AI Assistant's user interface. It provided a collaborative design environment, enabling the project team to iterate on design concepts, gather feedback, and refine the user experience.

- **Adobe XD:** Adobe XD was utilized for designing and prototyping the mobile application interface, offering a range of features for creating interactive designs and user flows. It facilitated the creation of high-fidelity prototypes and allowed for seamless integration with other Adobe Creative Cloud applications.

Hardware Infrastructure

The computational demands of training deep learning models necessitated the utilization of high-performance computing resources and hardware components, including:

- **GPUs (Graphics Processing Units):** Graphics Processing Units, such as the NVIDIA RTX 3060, were used to accelerate the training of deep learning models, leveraging their parallel processing capabilities and high computational throughput.

- **Cloud Computing Platforms:** Cloud computing platforms, including Google Colab and RunPod, provided scalable and on-demand access to computational

resources, facilitating the execution of computationally intensive tasks and the deployment of machine learning models in cloud environments.

- **32GB RAM:** A 32GB RAM configuration was utilized to support large-scale data processing and model training tasks, enabling efficient memory management and handling of complex datasets.
- **Ryzen 7 5800X:** The Ryzen 7 5800X processor contributed to the overall computational performance, offering high-speed processing capabilities and supporting multi-threaded workloads essential for data analysis and model training.

3.2 Artificial Intelligence Systems

This section outlines the artificial intelligence components that form the core of the Medibot system, which uses advanced techniques in computer vision and natural language processing to enhance medical diagnostic processes.

3.2.1 Computer Vision Models

The Computer Vision (CV) subsystem in Medibot is designed to analyze medical images such as X-rays and lab report images. This subsystem employs an ensemble of deep learning models to detect and classify medical conditions from these images.

- **Image Classification:** Medibot employs a CNN model based on the ResNet50 architecture, specifically designed to categorize uploaded images into distinct categories: X-ray images, lab report images, and miscellaneous images. This classification step is crucial for routing the images to the appropriate analysis module, ensuring that each type of image is processed with the most suitable algorithms and models.
- **X-ray Analysis:** Medibot utilizes an ensemble of advanced neural network architectures to perform detailed classifications of X-ray images. These models have been trained on expansive and diverse datasets, such as CheXpert and the NIH

Chest X-ray dataset (CXR14), which include a wide variety of medical conditions ranging from common ailments like pneumonia to more complex issues like fractures and cardiac anomalies. The following models are integrated into Medibot's analysis pipeline:

- **DenseNet121:** Chosen for its efficiency in feature extraction through its dense connectivity pattern, which improves information flow between layers and reduces the risk of overfitting.
- **VoVNetV2 (Volod2):** Utilized for its robustness and speed in processing images. VoVNetV2 enhances the representational capacity and learning ability, making it suitable for the varied and often subtle features in medical X-rays.
- **Swin Transformer V2:** Selected for its capability to model long-range dependencies in images, this transformer-based model provides superior performance in recognizing intricate patterns in X-ray images.
- **PSPNet:** Incorporated specifically for its proficiency in semantic segmentation tasks, which aids in identifying and delineating specific regions of interest in X-rays, such as areas showing signs of pathological changes.

Each model in the ensemble contributes uniquely to the overall accuracy and robustness of the X-ray analysis, ensuring comprehensive coverage of potential health issues detectable from X-ray imagery.

- **Lab Report Analysis:** For lab reports provided as images, an Optical Character Recognition (OCR) model is used to convert images to text. This text is then processed by the Large Language Models to generate simplified reports that highlight potential medical issues, assisting in preliminary diagnosis.

These models are integrated into Medibot's platform, allowing seamless interaction through the mobile and web applications. This integration is achieved using Docker containers, which encapsulate the model environments for scalable deployment.

3.2.2 Large Language Models

Medibot incorporates several Large Language Models (LLMs) to process and understand natural language input from users. These models are crucial for interpreting symptoms described by patients and providing relevant medical advice.

- **Main Routing Agent:** Utilizes Llama3-8b for efficient routing of queries between different agents. This model is specifically finetuned to understand the context of user inputs and direct them to the appropriate specialist or service within Medibot, ensuring optimal interaction flow.

- **Symptom Checker Agent:** Another instance of Llama3-8b is used for symptom extraction. This agent processes user inputs to identify and categorize symptoms accurately. It serves as a critical tool for the Main Agent, helping to refine the understanding of the user's condition before generating responses.

- **Main Agent for Interaction and Response:** Llama3-70b model functions as the central interaction unit, providing general answers and engaging with users.

- **Specialized Medical Guidance Agent:** Llama3-70b enhanced with Retrieval-Augmented Generation capabilities to offer specialized medical advice. Integrated with the Egyptian Pediatric Clinical Practice Guidelines, it provides responses that are both contextually relevant and medically accurate.

- **Lab Reports Agent:** A dedicated Llama3-8b model specializes in analyzing and summarizing lab reports. This agent, distinct from the chat interface, utilizes OCR technology to convert lab report images into text and then processes this text to extract and present key medical insights to users.

LangGraph is utilized to facilitate communication between the frontend applications and the backend LLM processing units.

Each component within the artificial intelligence systems of Medibot is designed to complement the others, creating a cohesive and efficient ecosystem that leverages

both computer vision and natural language processing to enhance the accuracy and reliability of medical diagnostics.

Chapter 4

Implementation and Preliminary Results

4.1 Code Structure

4.1.1 Frontend

The frontend implementation of Medibot utilizes ReactJS enhanced with TypeScript, which provides strong typing to improve code reliability and debuggability. The frontend architecture is meticulously designed to be modular, promoting extensive reusability and ease of maintenance. This modularity ensures that components can be reused and updated independently, facilitating scalable application development.

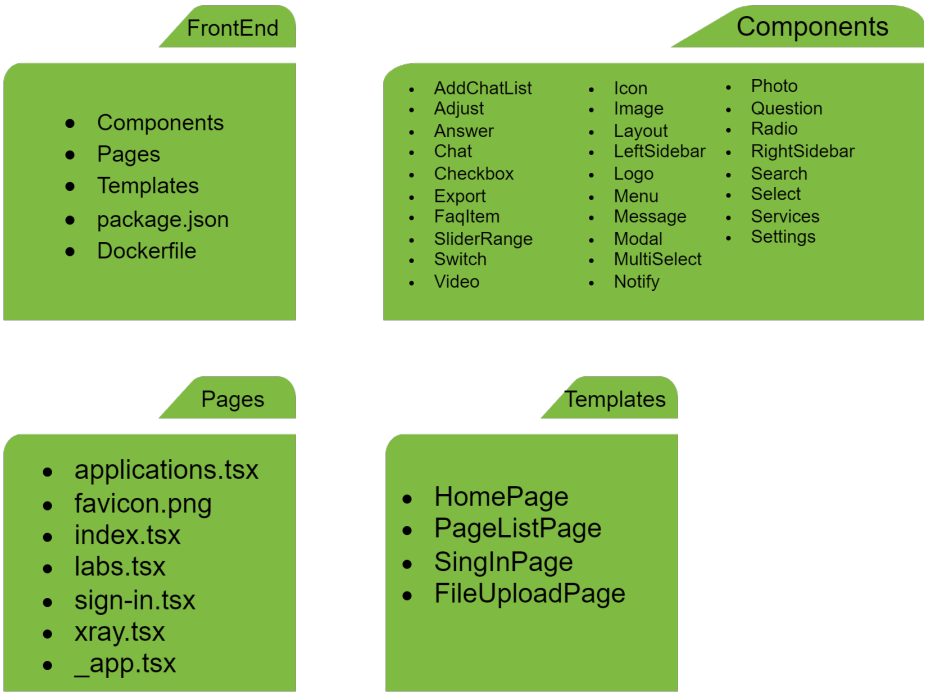

Figure 4.1: Frontend Code Structure

The frontend codebase is organized into several distinct sections:

- **Components:** This directory hosts reusable UI elements such as buttons, modals, and input fields, which are standardized to maintain consistency across the application.
- **Pages:** Contains components that correspond to various routes within the application, like the home page, chat interface, and image upload page. These components integrate various elements from the Components and Templates directories to render comprehensive views.
- **Templates:** Includes layout templates used across different pages, facilitating a uniform structure and design throughout the application.
- **Dockerfile:** A Docker configuration file that enables the frontend to be deployed in a containerized environment, enhancing the portability and scalability of the application.

This structure not only simplifies the development process but also streamlines future enhancements and scalability by allowing developers to navigate the codebase easily, implement updates, and add new features without disrupting existing functionalities.

4.1.2 Backend

The backend of Medibot is developed using Node.js, leveraging its non-blocking I/O capabilities to handle concurrent user interactions efficiently. The backend structure is optimized for performance and scalability, making extensive use of MongoDB for data storage, which allows flexible data handling and rapid retrieval necessary for real-time applications like Medibot.

Figure 4.2: Backend Code Structure

The backend architecture is structured as follows:

- **src:** This directory serves as the primary container for the backend code, including:

  - **backend:** Houses core backend logic that handles operations such as authentication, chat management, X-ray analysis, and lab report processing.
  - **helper:** Contains helper functions that support the backend operations by providing utility functions and database schema definitions.
  - **routes:** Manages all the routing logic, directing requests to the appropriate controllers based on the endpoint accessed.

- **server.js:** The main entry point for the backend server which initializes the application and sets up the connection to the MongoDB database.
- **package.json:** Specifies the project dependencies and scripts necessary for the Node.js environment.
- **.env:** A configuration file that stores environment-specific variables such as database credentials, ensuring sensitive data is kept secure.
- **Dockerfile:** Facilitates the deployment of the backend in a containerized environment, enhancing the portability and consistency of the application across different development and production setups.

This organized structure ensures that the backend remains maintainable and scalable, accommodating future expansions and functionalities seamlessly.

4.1.3 Large Language Models (LLMs)

The Large Language Models (LLMs) component is crucial to Medibot, providing the intelligent processing power needed for interacting with users and analyzing medical data. The LLMs are organized into distinct agents and tools, each serving specific functions within the Medibot system.

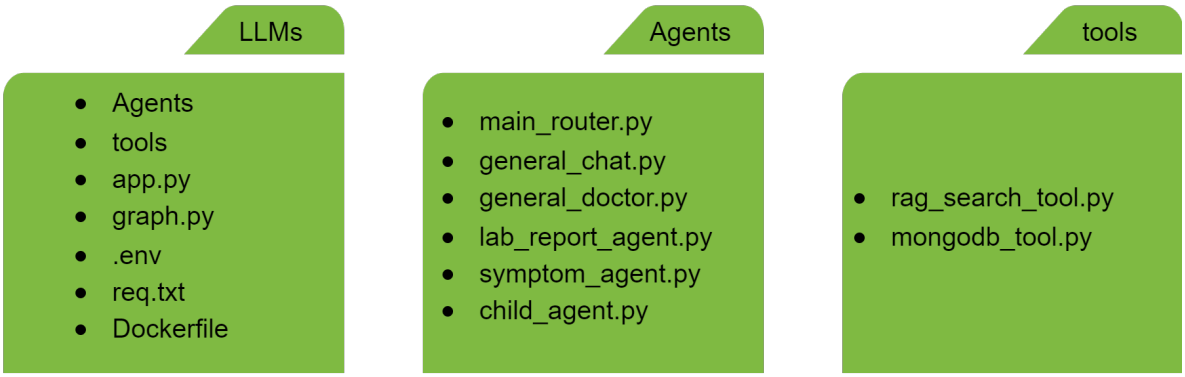

Figure 4.3: Structure of LLM Agents and Tools

Agents

The agents are specialized Python modules, each tailored to perform specific tasks:

- **main_router.py** — Directs user inquiries to the appropriate agent based on the query context.
- **general_chat.py** — Handles general user interactions and queries.
- **general_doctor.py** — Provides medical advice based on analyzed symptoms and user history.
- **lab_report_agent.py** — Processes and extracts information from uploaded lab reports.
- **symptom_agent.py** — Specializes in identifying and categorizing symptoms described by users.
- **child_agent.py** — Offers pediatric-specific advice and information, integrating guidelines from pediatric sources.

Tools

Supporting the agents are various tools that enhance their capabilities:

- **rag_search_tool.py** — Utilizes retrieval-augmented generation to fetch and integrate external data sources for enriched responses.
- **mongodb_tool.py** — Manages database interactions, facilitating data storage and retrieval from MongoDB.

Infrastructure

The LLMs operate within a Python environment, managed and isolated using Docker for consistent deployment. The structure also includes:

- **app.py** — Initializes the server and sets up the application.
- **graph.py** — Manages knowledge graph interactions for complex data relationships.
- **.env** and **req.txt** — Manage environment variables and Python dependencies, respectively.
- **Dockerfile** — Defines the Docker container configuration to ensure environment consistency across different setups.

This layout supports Medibot's operational needs by ensuring modular, maintainable, and scalable interactions between different components of the AI system.

4.1.4 Computer Vision Models

The Computer Vision (CV) component of Medibot is crucial for analyzing medical images such as X-rays and lab reports. This section of the system is designed for high performance and accuracy in image classification and analysis.

Figure 4.4: Computer Vision Code Structure

Overview

The CV module is structured into separate directories for managing different types of model files and their corresponding weights. It is also equipped with various helper scripts to facilitate preprocessing and integration tasks.

Directories and Files

- **modelFiles:** Contains directories for different categories of model applications, including X-ray analysis, lab report processing, and general image classification.
- **helpers:** Includes utility scripts such as:
  - **noise_removal.py** — Enhances image quality by removing noise.
  - **send_to_llm.py** — Integrates processed data with LLM components for further analysis.
  - **jpg_to_png.py** — Converts images from JPG to PNG format for uniformity in processing.
  - **boundary_analysis.py** — This script is crucial for analyzing the boundaries within lab reports. It identifies the locations of tables and other structured data elements to ensure accurate data extraction from scanned lab reports. This is particularly important for maintaining the integrity of data during the OCR process and subsequent analyses.

Model Specifics

X-ray and Image Classification Models These models are specially trained to identify and classify medical images with high accuracy, leveraging state-of-the-art deep learning architectures:

- **DenseNet121, Volod2, Swin Transformer V2, and PSPNet** are used for detailed X-ray image analysis.
- **ResNet50** is utilized for broader image classification tasks, ensuring models only get their respective image for processing.

Lab Reports

The lab report directory is configured with its own application script to manage the OCR and data extraction processes, ensuring that text data from scanned lab reports

is accurately converted into a structured format.

Infrastructure

The entire CV architecture is dockerized, which standardizes the development environment and ensures that the models perform consistently across different deployment platforms. Each model directory includes:

- **app.py** — Main application script for running the model.
- **modelWeights** — Folder containing the pre-trained model weights (.pth files) used for predictions.

This organized structure not only supports scalability but also enhances the maintainability of the CV component within Medibot, ensuring efficient and accurate image processing and analysis.

4.2 Data Structure and Databases

This section details the data models and schema designs utilized in the MongoDB and Milvus databases, integral to the operation of Medibot. We discuss the integration of these databases into the backend, including indexing strategies and optimization techniques employed to enhance performance.

4.2.1 MongoDB Schemas

MongoDB, a NoSQL document-based database, is used to store user data and conversational logs. The flexibility of MongoDB's schema-less architecture allows us to adjust data structures as needed. Below are the core schemas:

User Schema

The User Schema is central to managing user information and interactions within Medibot. Fields include:

- **email** - User's email address.
- **username** - Unique username for login.
- **full name** - User's full legal name.
- **password** - Hashed password for user authentication.
- **conversations** - Array of references to Conversation documents.
- **_id** - MongoDB's default unique identifier.
- **history** - Log of past interactions and sessions.
- **age** - User's age, optional for demographics analysis.
- **gender** - User's gender, optional for personalized responses.

Conversation Schema

The Conversation Schema tracks the interactions between users and Medibot:

- **user_id** - Reference to the associated User document.
- **_id** - MongoDB's default unique identifier.
- **messages** - Array of message objects containing text and metadata.
- **date** - Timestamp of the conversation start.

4.2.2 Milvus Vector Database Integration

Medibot employs the Milvus vector database to manage and facilitate fast retrieval of large-scale vector data, which is essential for features like the symptom checker. Milvus enables the efficient similarity search necessary for accurately suggesting possible medical conditions based on symptom analysis.

Vector Data Structure

The vector data in Milvus is organized into collections, where each vector represents unique symptom data and potential medical diagnoses. Below is a schematic representation of the data structure used in the symptom analysis collection in Medibot:

```
{
    "id": "<unique_vector_id>",
    "vector": ["<list_of_vector_values>"],
    "$meta": {
        "metadata": {
            "id": "<vector_id>",
            "symptoms": ["<list_of_symptoms>"],
            "diseases": "<associated_medical_condition>",
            "Source_URL": "<reference_url>"
        }
    }
}
```

Each vector is stored with metadata that encapsulates the symptom details and corresponding medical conditions, enabling Medibot to predict and diagnose based on user inputs.

Integration and Optimization

Integration of Milvus within Medibot involves a Dockerized environment, ensuring easy deployment and scalability. The system employs advanced indexing techniques, such as the IVF (Inverted File) index, to optimize storage and retrieval speeds. These index settings are finely tuned to the specific requirements of the search functionality, aiming to balance accuracy and query performance effectively.

Querying Vectors

The querying of vectors utilizes Milvus's robust search functions, which support specifying search parameters like the number of nearest vectors to retrieve. This functionality underpins the symptom checker tool, where user-inputted symptom vectors are matched against the database to find the most relevant medical conditions.

4.2.3 Database Integration and Performance Optimization

Both MongoDB and Milvus are integrated into Medibot's backend via Docker containers, ensuring isolated environments and easy scalability. Indexing strategies on MongoDB include:

- **Text Indexes** on user emails and usernames to speed up search operations.
- **Compound Indexes** on conversations to optimize retrieval by user and date.

Optimization in Milvus focuses on configuring the vector dimensions and indexing parameters to balance search accuracy and query speed.

4.3 Data Processing and Generation of New Datasets

This section details the methods and technologies employed in Medibot for processing textual and image data and for generating new datasets through advanced techniques.

4.3.1 Textual Data Processing and Dataset Generation

Symptom Data Transformation

Utilizing the NIH Medline dataset, which includes comprehensive details on various medical conditions, we extracted symptom data. This data was processed with the GPT-4 model to transform descriptive symptom information into structured bullet points, enhancing clarity and usability.

Question Generation

From the bullet-pointed symptom data, new datasets containing questions and sentences that emulate patient descriptions to a doctor were generated. We employed two models for this task:

- GPT-3.5 was used to generate 10,000 questions.
- Llama3-70b was also used to generate an additional 10,000 questions.

These questions are designed to mimic initial patient inquiries about symptoms.

Follow-up Question Generation

To create data that could simulate follow-up questions during doctor-patient interactions, we utilized the Mixtral22b*8 model. This model processed entire records from the dataset, including symptoms, causes, and treatments, to generate 40,000 diverse follow-up questions. This enhances the dataset with a broader range of conversational AI training materials.

4.3.2 Image Data Processing and Augmentation

Chest X-Ray Image Dataset

We augmented 70% of the Chest X-Ray-14 dataset with random filters to introduce variability, simulating more challenging real-world conditions that the AI model might encounter.

Lab Report Data Integration

For training our image classification models, we integrated:

- 100 real lab reports.
- 3,000 documents from the company documents dataset available on Kaggle.
- 3,000 x-ray images from existing medical image datasets.

- 3,000 random images from MNIST image datasets.

This mixed dataset provides a robust training environment for the Image classifier model to recognize differentiate between X-rays, lab reports and random images.

4.3.3 Combined Dataset Utilization

The augmented textual and image datasets are each used to train specific components of Medibot's AI systems. The textual data, enriched with questions and follow-up scenarios, is crucial for training the language understanding capabilities of the LLMs. This allows Medibot to handle a wide range of user inquiries with accuracy and contextually appropriate responses.

The processed image datasets, particularly those with augmented X-ray and lab report images, are used to train and fine-tune the computer vision models within Medibot. This specialization ensures that each model excels in its respective domain, from diagnosing medical conditions based on X-ray images to accurately extracting and interpreting data from lab reports.

This approach ensures that each type of data is optimally utilized according to its relevance and contribution to the overall functionality of Medibot, enhancing the system's ability to handle diverse and realistic medical diagnostic scenarios effectively.

4.4 Computer Vision Models

4.4.1 X-ray Analysis

Model Training

The X-ray analysis within Medibot involves an advanced training regimen using multiple neural network architectures to accurately detect a range of X-ray abnormalities. Our models include DenseNet121, VoLOd2, and Swin Transformer V2, trained specifically on the NIH Chest X-ray dataset (CXR14), which consists of 112,000 images depicting various medical conditions.

- **Data Preprocessing:** Before training, the X-ray images are preprocessed to mitigate common imaging issues. Adjustments are made to reduce noise from glare, correct brightness and contrast imbalances, eliminate tilt and moire patterns, and filter out Gaussian noise. These preprocessing steps are crucial for ensuring the models receive clean and standardized inputs.

- **Training Split:** The dataset is divided into 70% training, 10% validation, and 20% testing segments. This distribution allows for comprehensive learning and rigorous assessment of the model's performance.

- **Model Training and Fine-tuning:** Initial training sessions are conducted over 20 epochs per model on the original NIH dataset to establish a baseline understanding. Subsequent fine-tuning is performed for an additional 5 epochs on an augmented version of this dataset, incorporating synthetic variations to simulate a wider array of X-ray scenarios. This strategy significantly raises the models' accuracy and robustness.

- **Additional Segmentation Model:** Alongside the classification models, a PSPNet model is trained specifically for image segmentation tasks to delineate distinct anatomical structures within the X-ray images, further enhancing the diagnostic capabilities of Medibot.

Model Evaluation

The X-ray analysis model is evaluated based on several performance metrics, including accuracy, precision, recall, and F1 score. The model's ability to detect specific abnormalities is assessed through confusion matrices and ROC curves, providing insights into its strengths and limitations.

Results

The X-ray analysis model demonstrates high accuracy in detecting common abnormalities, with precision and recall scores exceeding 90% for pneumonia and pneumothorax.

The model's performance on the test set indicates its robustness and reliability in diagnosing chest X-ray images accurately.

4.4.2 Image Type Classification

Model Training

The image type classification model within Medibot employs a ResNet50 architecture. It is designed to differentiate among three distinct types of images: X-ray images, MNIST images representing random images, and document images. The model is trained on a dataset comprising 3,000 images from each category, ensuring diverse exposure to various image types.

- **Data Augmentation:** To enhance the model's ability to generalize across different conditions, data augmentation techniques are applied, introducing variations in the training dataset that reflect real-world imaging scenarios.
- **Model Training:** The training process extends over 30 epochs with a dataset split of 70% training, 10% validation, and 20% testing. This structure ensures comprehensive learning and accurate evaluation.
- **Validation and Testing:** The model's performance is meticulously evaluated on the validation and test sets, focusing on accuracy, recall, precision, and F1 scores to ensure effective classification across all image types.

Model Evaluation

The model's effectiveness is quantitatively assessed through key performance metrics:

- **Training Performance:** Achieved an accuracy of 97.94%, recall of 96.24%, and an F1 score of 96.21%, indicating exceptional internal consistency and learning capability.
- **Validation Performance:** On the validation set, the model demonstrated accuracy of 97.2%, recall of 96.9%, and an F1 score of 96.7%.

Results

The image type classification model proves highly effective in differentiating between X-ray, document, and random MNIST images. The high scores in precision, recall, and F1 metrics during training underscore the model's capability, while validation results confirm its practical applicability and reliability in real-world scenarios.

4.5 Large Language Models (LLMs)

4.5.1 Symptom Extractor Llama3-8b

The Symptom Extractor Llama3-8b model is fine-tuned specifically for extracting medical symptoms from user inputs. It leverages advanced natural language processing techniques to identify and parse symptoms effectively, providing crucial support for diagnostic processes.

LoRA Configuration

To enhance the model's adaptability and learning efficacy without extensive retraining, Low-Rank Adaptation (LoRA) was employed with the following parameters:

- **Alpha**: 64
- **Dropout**: 0 (no dropout)
- **Target Modules**: "q_proj", "k_proj", "v_proj", "o_proj", "gate_proj", "up_proj", "down_proj"

This fine-tuning approach significantly enhances the model's capability to focus on the crucial task of symptom extraction from complex medical descriptions. The specific adjustment of target layers through LoRA enables the model to retain general linguistic capabilities while adapting to specialized tasks.

Training Strategy

The training process was meticulously designed to optimize the model's performance on symptom extraction. The detailed training prompt used is structured as follows:

```
system: You are a medical assistant specialized in extracting symptoms from sentences.
user: You will extract the symptoms from the sentence: {Input_Data}
You should only output in JSON format.
assistant: {Output_Data}
```

The training infrastructure was managed using the `SFTTrainer` setup, with a strategy focused on maximizing efficiency and accuracy, demonstrated by the rigorous training parameters and results.

Training Results

The model training was completed over approximately 5 epochs, with the following detailed outcomes:

- **Global Steps**: 1070
- **Training Loss**: 0.4649
- **Runtime**: 2975 seconds (approx.)
- **Samples per Second**: 2.879
- **Steps per Second**: 0.36
- **Total FLOPs**: 1.549×10^{17} (Floating Point Operations)
- **Epochs Completed**: 4.988

4.5.2 General Medical Knowledge Llama3-70b

The Llama3-70b model, aimed at providing detailed medical information and answering related queries, was fine-tuned using a large-scale dataset and advanced training methodologies to ensure its effectiveness in real-world applications.

Training Overview

The model was trained with an extensive set of 42,000 medical questions generated by MoE Mixtral22b*8, using an 80/10/10 split for training, validation, and testing. The training was executed over 2 epochs, achieving significant improvements in understanding and generating medically relevant content.

Training Setup

The training was orchestrated using a customized setup configured as follows:

```
Trainer:
  - Batch Size: 32 per device
  - Gradient Accumulation Steps: 4
  - Maximum Steps: 1000
  - Learning Rate: 2.5e-5
  - Warmup Steps: 5
  - Logging and Evaluation: Every 5 steps
  - Checkpoint Saving: Every 25 steps
  - Evaluation Strategy: Every 50 steps
```

Training Prompt

The training prompt employed was designed to emulate professional responses typical of a medical AI assistant:

```
system: You are an AI assistant specializing in medical knowledge.
user: {Question}
assistant: {Answer}
```

This setup ensures that Llama3-70b effectively learns the nuances of medical consultation, enhancing its capability to respond with accurate and contextually appropriate information.

Training Results

The training process yielded the following results:

- **Global Steps**: 1000
- **Training Loss**: 0.6078
- **Runtime**: 14428.7 seconds (approx.)
- **Samples per Second**: 4.436
- **Steps per Second**: 0.069
- **Total FLOPs**: 1.168×10^{18} (Floating Point Operations)
- **Epochs Completed**: 1.905

4.6 Quantitative Results

Graphs for our model predicting medical conditions vs base models

4.7 Qualitative Results

Response times for each part of the website and mobile app, (Chat reply time, xray reply time, lab report reply time, login time and create account time).

Chapter 5

Discussion and Conclusion

5.1 Interpretation of Results

The implementation and preliminary results of Medibot demonstrate substantial advancements in AI-driven medical diagnostics. The system's robust architecture, encompassing both computer vision and large language models, has shown promising results in terms of accuracy and efficiency. Medibot's ability to process and interpret complex medical data through its sophisticated algorithms underlines the potential of AI to support and enhance decision-making in healthcare settings.

5.1.1 Challenges and Limitations

The development of Medibot has encountered several significant challenges, key among them being the scarcity and unstructured nature of available medical data. To overcome this, we were compelled to generate our own datasets using large language models, which introduced additional complexities related to data validity and representativeness.

- **Data Acquisition and Structure:** Obtaining sufficient medical data for training our models proved challenging due to the proprietary nature of most medical datasets. Moreover, much of the available data lacked the structured format necessary for efficient processing and analysis, necessitating extensive preprocessing and data structuring efforts using advanced language models.

- **Bias and Model Reliability:** To mitigate bias in our models, we employed multiple algorithms and diverse data sources. Despite these efforts, ensuring the complete neutrality and fairness of the models remains a complex challenge that requires ongoing attention and refinement.
- **Computational Resources:** Limited access to computational resources and funding constraints significantly impacted our ability to experiment with various architectures and training methodologies. This limitation affected our capacity to iterate quickly and test the most innovative AI techniques.
- **System Integration:** The multimodal nature of Medibot introduced complexities in integrating various components such as text processing units, image analysis models, and user interface elements. Ensuring seamless interaction between these diverse systems was both challenging and resource-intensive.

These challenges underscore the difficulties in deploying advanced AI systems in healthcare settings, which require not only technological innovation but also a deep understanding of the regulatory and ethical dimensions of medical practice.

5.2 Support from Microsoft Azure Founders Fund

On June 18, 2024, Medibot was awarded a significant endorsement from the Microsoft Azure Founders Fund, receiving $5000 in Azure credits. This grant, valid for one year, underscores Microsoft's confidence in Medibot's potential to revolutionize medical diagnostics with AI-driven solutions. The Azure credits will empower us to harness Microsoft Azure's robust cloud computing services, enabling enhanced processing capabilities and scalability. This support is pivotal in advancing our development, allowing us to integrate more sophisticated AI features and expand our service offerings, thereby accelerating our mission to deliver advanced healthcare solutions.

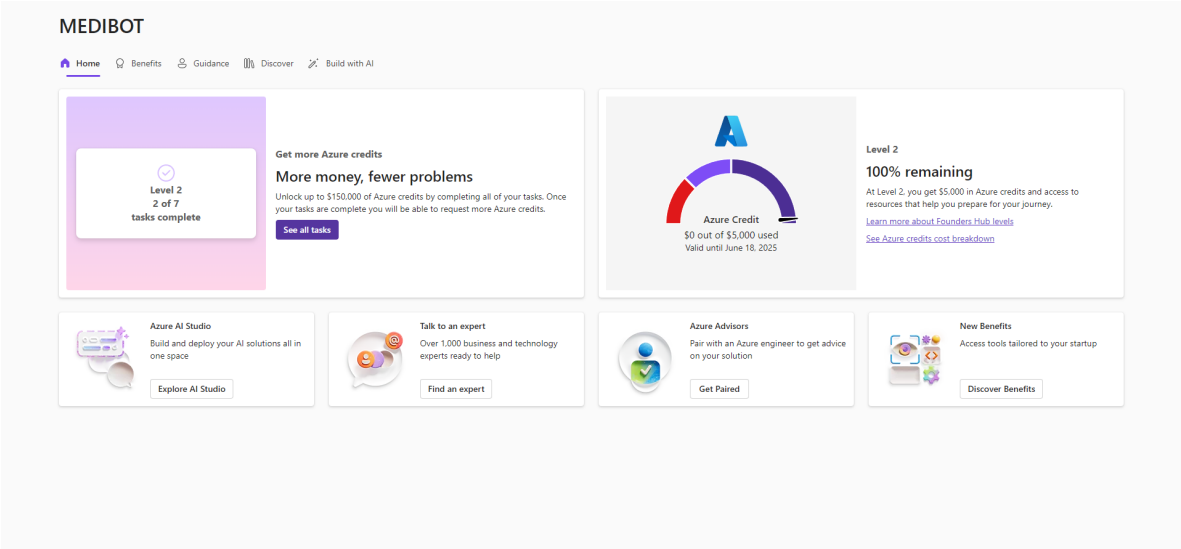

Figure 5.1: Microsoft Azure Founders Fund Approval Dashboard

5.3 Future Work

In light of our current achievements, the challenges faced, and the recent financial support from Microsoft Azure Founders Fund, our future efforts will encompass:

- **Enhancing Model Accuracy:** With the additional Azure credits, we plan to access more substantial computational resources, enabling us to continuously train our models with up-to-date and diverse datasets. This effort will improve diagnostic accuracy, reduce biases, and allow us to explore more complex model architectures.

- **Expanding System Capabilities:** We aim to integrate additional functionalities such as real-time patient monitoring, predictive analytics for chronic diseases, and a new voice-to-text feature to enhance user interaction by allowing spoken input, facilitating accessibility and ease of use.

- **Interoperability:** Ensuring Medibot can seamlessly integrate with existing healthcare IT ecosystems is crucial. With enhanced resources, we can accelerate the development of interoperable solutions, facilitating widespread adoption.

- **Global Expansion:** The fund also opens up possibilities for localizing Medibot to cater to non-English speaking regions and adapting the system to meet various international healthcare standards. This will help in addressing the nuances of global healthcare challenges.
- **Big Data Capabilities:** The Azure funding will enable us to handle larger datasets, which are essential for training more robust models. This capability will directly address our previous challenges of limited data access and allow for more extensive testing and validation phases.
- **Enhanced Image Processing:** We plan to augment our image processing capabilities by incorporating additional data points such as age, gender, and position of view from image metadata. This enhancement will improve the specificity and relevance of our diagnostic models.
- **Training Larger MoE Models:** With the capability to handle more computational resources, we aim to train larger MoE (Mixture of Experts) 22b*8 models. Their high contextual awareness can yield better results across various medical query types and scenarios.

These initiatives will leverage the new funding to significantly scale Medibot's capabilities, pushing the boundaries of what our AI-driven solution can achieve in the medical field.

5.4 Conclusion

Medibot, our Medical Diagnosis AI Assistant, represents a transformative approach in healthcare, merging advanced AI technologies with medical diagnostics to provide immediate, accessible, and accurate medical advice. Throughout its development, we have integrated state-of-the-art machine learning models, sophisticated natural language processing tools, and innovative computer vision capabilities to create a system that not

only understands complex medical language but also interprets medical imagery with high precision.

The implementation of Medibot involved several components, each carefully crafted to address specific needs within the healthcare sector. The frontend, developed using ReactJS and TypeScript, offers a user-friendly interface that simplifies the interaction process for users. The backend, powered by Node.js and MongoDB, ensures efficient data handling and robust API management, allowing Medibot to handle multiple user interactions seamlessly. The integration of Large Language Models (LLMs) and Computer Vision systems has equipped Medibot with the ability to process textual and visual data effectively, turning raw medical data into actionable insights.

One of the critical achievements of this project has been the adaptation and fine-tuning of models like Llama3-8b and Llama3-70b, which have significantly enhanced Medibot's ability to provide tailored medical advice and handle diverse medical inquiries. Moreover, the support from Microsoft Azure Founders Fund has been instrumental, providing us with the necessary resources to scale our operations, train more complex models, and expand our dataset capacities.

Despite these advancements, we encountered challenges such as limited initial data access, the need to generate synthetic medical data, and the complexities of integrating multimodal systems. However, the Azure grant has positioned us to overcome these hurdles by allowing access to enhanced computational resources and enabling global scaling and multilingual capabilities.

Future work for Medibot includes expanding its real-time patient monitoring features, developing voice-to-text capabilities, enhancing interoperability with existing medical systems, and extending our services to non-English speaking regions. Additionally, plans to integrate predictive analytics for chronic diseases and improve model accuracies with more extensive and diverse datasets will further refine Medibot's functionalities.

In conclusion, Medibot stands as a testament to the potential of artificial intelligence in revolutionizing healthcare. It exemplifies how technology can bridge gaps in medical

service delivery, providing reliable, efficient, and accessible diagnostics and patient care. As we continue to develop and expand Medibot, we aim to not only enhance its capabilities but also ensure it becomes an integral part of the global healthcare ecosystem, making high-quality medical advice available to everyone, everywhere.

Bibliography

[1] M. C. Younis and H. Abuhammad, "A hybrid fusion framework to multi-modal biometric identification," *Multimedia Tools and Applications*, vol. 80, no. 17, pp. 25799–25822, 2021.

[2] M. Rasol, S. Haider, M. Selman, E. Jarjees, and S. Atroshy, "Graduation project guide (gpg)," 09 2015.

[3] J. Irvin, P. Rajpurkar, M. Ko, Y. Yu, S. Ciurea-Ilcus, C. Chute, H. Marklund, B. Haghgoo, R. Ball, K. Shpanskaya, *et al.*, "Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 590–597, 2019.

[4] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, "Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases," *arXiv preprint arXiv:1705.02315*, 2017.

[5] M. Arslan, A. Haider, M. Khurshid, S. A. Bakar, R. Jani, F. Masood, T. Tahir, K. T. Mitchell, S. Panchagnula, and S. Mandair, "From pixels to pathology: Employing computer vision to decode chest diseases in medical images," *Cureus*, 2023.

[6] V. K. C. Bumgardner, A. D. Mullen, S. Armstrong, and J. Talbert, "Local large language models for complex structured medical tasks," *ResearchGate*, 2023.

[7] NHS (National Health Service), "NHS Conditions."

[8] "Medical transcriptions dataset."

[9] "Medical datasets on hugging face."

[10] "Chexpert dataset."

[11] "Vindr-cxr dataset."

[12] "Chestx-ray8 dataset."

[13] "Chestx-ray14 dataset."

[14] "Stanford ml group."

[15] "Tensorflow datasets - chexpert."

[16] "Inception-mbzuai/jais-13b."

[17] "Inception-mbzuai/jais-13b-chat."

[18] "Thebloke/llama2-13b-tiefighter-gptq."

[19] "Thebloke models."