NeuraHealth

AHMAD SUBHAN
01-136221-001
MUSKAN
01-136221-024

Supervisor: Ms. Afrah Naeem

Bachelor of Science in Computer Science

Department of Computer Science
Bahria University, Islamabad

November 2025

Certificate

We accept the work contained in the report titled **NeuraHealth**, written by Mr. Ahmad Subhan and Ms. Muskan as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

Approved by:

Supervisor: Ms. Afrah Naeem

_____

Internal Examiner:

_____

External Examiner:

_____

Project Coordinator: Dr. Sohail Akhtar

_____

Head of the Department: Dr. Khurram Ehsan

_____

Abstract

NeuraHealth is an AI-driven healthcare system developed to improve patient experience and automate hospital workflows through patient-centric processes. The system addresses a common issue faced by patients who are often unsure about which specialist to consult for their symptoms. Using a custom-trained language model combined with domain-specific medical logic, NeuraHealth identifies symptoms, recommends suitable specialists, suggests precautionary steps, and advises basic diagnostic tests such as blood pressure, cholesterol, and sugar level assessments.

A conversational chatbot serves as the primary interaction channel, welcoming patients, collecting symptoms, generating structured summaries, and automating appointment bookings. These summaries are forwarded to the doctor's dashboard to support clinical preparation and decision-making. NeuraHealth also offers a comprehensive web-based platform that provides dedicated interfaces for doctors, patients, and administrators, enabling streamlined appointment management, record access, and workflow oversight.

Additional components include doctor dashboards, an appointment scheduler, and a web-scraping module that continuously enriches the medical knowledge base. NeuraHealth emphasizes iterative refinement, scalability, and adaptability for deployment across multiple hospitals. Unlike conventional solutions, the system adheres to university policies by relying solely on in-house trained models rather than pretrained large language models. Overall, NeuraHealth aims to reduce administrative burden, improve initial patient evaluation, and enhance operational efficiency within healthcare institutions.

Acknowledgments

We would like to express our sincere gratitude to all those who supported us throughout the development of this Final Year Project.

First and foremost, we are deeply thankful to our supervisor, **Ms. Afrah Naeem**, whose guidance, valuable feedback, and constant encouragement were instrumental in shaping the direction of this project. Her expertise and support made it possible to transform an ambitious concept into a functioning and meaningful system.

We are also grateful to the faculty members of the *Department of Computer Science, Bahria University Islamabad*, for creating an academic environment that fosters critical thinking, innovation, and problem-solving.

Special thanks to the hospital staff and medical professionals who provided insights into real-world healthcare challenges. Their input was crucial in aligning the system's goals with practical needs.

To our friends and colleagues who offered moral support, helped test the system, or simply believed in the idea—thank you. Your encouragement made a significant difference.

Lastly, we would like to thank our families for their unwavering patience, motivation, and understanding during the course of this project. Their faith in us has always been our greatest source of strength.

AHMAD SUBHAN
MUSKAN
Bahria University
E-8, Islamabad, Pakistan

November 2025

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
CNN	Convolutional Neural Network
DBMS	Database Management System
EMR	Electronic Medical Record
ERD	Entity Relationship Diagram
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
LLM	Large Language Model
ML	Machine Learning
NLP	Natural Language Processing
RAG	Retrieval-Augmented Generation
SQL	Structured Query Language
UML	Unified Modeling Language
UI	User Interface
UX	User Experience
WHO	World Health Organization

Chapter 1

Introduction

1.1 Project Background/Overview

In many healthcare settings, a common and often overlooked challenge patients face is not knowing which medical department or specialist to consult based on their symptoms.[1, 2] This uncertainty leads to delays in treatment, misdirection within hospital departments, and sometimes incorrect consultations. Such inefficiencies not only affect the quality of care but also put unnecessary strain on hospital administrative staff.[2]

To solve this problem, our project NeuraHealth: AI-Powered Smart Healthcare Automation, is designed as a modular hospital management platform featuring an intelligent chatbot system. The chatbot acts as a virtual receptionist and preliminary medical assistant [3] , collecting symptom data through natural conversation. It then analyzes the information using a custom-trained language model (LLM) and recommends the appropriate specialist [4] , possible precautionary measures, and basic diagnostic tests (e.g., blood pressure, cholesterol).

In addition, the system forwards a summarized case history to the doctor's dashboard before the patient's visit, improving preparedness and consultation efficiency. The platform follows a microservices-based architecture with API integrations, supporting both cloud and on-premise deployments.

While initially designed for a specific hospital for focused deployment and testing, NeuraHealth is built with scalability in mind. Its architecture allows for future expansion into a publicly available platform, supporting multi-hospital environments and remote teleconsultations.

1.2 Problem Description

Modern hospitals face operational overload due to high patient volumes, inefficient appointment management, and fragmented communication between patients and healthcare

providers. Reception staff are often overwhelmed by administrative tasks, resulting in increased patient waiting times and resource mismanagement.[1]

Moreover, many patients, especially first-time visitors or those without prior medical knowledge, struggle to identify which doctor or department to consult based on vague or overlapping symptoms.[2] This leads to:

- Wrong initial appointments or department referrals.
- Increased administrative back-and-forth.
- Inefficiencies in doctor-patient time management.

There is a pressing need for a smart automation system that improves patient guidance, enhances appointment flow, and provides doctors with context-aware summaries for better consultation.

1.3 Project Objectives

The primary objectives of the NeuraHealth system are:

- To design and develop an AI-powered conversational assistant [3, 4] that can:
  - Collect patient symptoms via chat.
  - Recommend suitable medical specialists.
  - Suggest basic medical tests and precautions.
- To automate appointment booking and reminders using natural language interaction.
- To generate concise medical summaries for doctors before appointments.
- To develop a scalable, modular platform deployable in both private and public hospital settings.
- To ensure patient data privacy and system reliability through best software engineering practices.

1.4 Project Scope

The scope of this project includes:

- Deployment in a single hospital environment for initial testing and real-time feedback.
- Integration with hospital-specific APIs for managing appointments, doctor availability, and patient history.

- Development of a custom-trained lightweight language model for natural language understanding (due to institutional restrictions on using pretrained LLMs).[3]
- Implementation of a dashboard interface for doctors to review summarized patient inputs before consultation.
- Web scraping modules to fetch relevant medical data to enrich chatbot responses where appropriate.
- Future-proof architecture to scale into a multi-hospital system or public healthcare platform.

The project does not aim to diagnose illnesses or replace medical professionals but rather serves as a pre-consultation and administrative aid to improve efficiency and user experience.

Table 1.1: **Tools and Technologies Used in System Implementation**

Tools and Technologies Used	
Frontend Technologies	• **Framework:** React.js • **Styling:** Tailwind CSS • **Data Visualization:** Recharts, Chart.js • **State Management:** Redux Toolkit • **Routing:** React Router • **Authentication:** JWT-based role login
Backend Technologies	• **Framework:** FastAPI (Python) • **Language:** Python 3.10 • **AI/ML Libraries:** scikit-learn, transformers, spaCy, NLTK • **Server:** Uvicorn ASGI Server • **Serialization:** Pydantic Models
Database and Storage	• **Database:** MongoDB Atlas • **Driver:** PyMongo • **Collections:** users, doctors, appointments, chat_history, medical_records, sessions • **Indexes:** email, doctor_id, appointment dates • **Data Security:** Encryption-at-rest + TLS • **Backup:** Automated cloud backups

Chapter 2

Literature Review

Literature review is a systematic method of identifying, evaluating, and interpreting existing work relevant to a specific topic. For this project, the focus lies in healthcare automation systems — specifically, AI-powered medical chatbots for symptom checking, specialist recommendation, and appointment scheduling.

This chapter contextualizes the NeuraHealth system by reviewing prior systems, identifying their limitations, and highlighting the opportunity for innovation through our proposed model.

2.1 Similar Pakistani Applications

Several health-tech applications have emerged in Pakistan to support digital healthcare transformation:

- **Marham** – A platform that allows patients to search for doctors, book appointments, and consult online. However, it lacks intelligent symptom analysis or test recommendation features [5].
- **Dawaai** – A pharmaceutical delivery app that expanded into telemedicine, offering doctor consultations. It doesn't assist users in selecting the right department based on symptoms [6].
- **Sehat Kahani** – Focuses on providing virtual consultations through qualified doctors, particularly targeting remote areas. While impactful, it lacks automated symptom-to-specialist routing [7].
- **Oladoc** – A directory-based doctor appointment app, providing scheduling and some patient reviews, but no AI-enabled interaction [8].

While these systems improve access to healthcare services, they primarily rely on manual input and user navigation. Patients still face challenges such as misinterpreting symptoms, selecting the wrong doctor, or manually navigating multiple platforms for tests, prescriptions, and follow-ups. This highlights a significant gap for intelligent automation that can guide patients end-to-end in a hospital workflow.

2.2 Global Systems and Academic References

Globally, AI has been increasingly applied to support healthcare decision-making:

- **Ada Health** – An AI-powered symptom checker used globally, allowing users to enter symptoms and get condition suggestions. Although robust, its backend is not optimized for localized healthcare needs, and it often lacks integration with hospital management systems [9].
- **Babylon Health** – Offers automated consultations based on patient medical history and symptoms using AI. While it provides strong triage functionality, privacy concerns and the lack of direct integration with local hospital workflows limit its applicability in certain regions [10].
- **Infermedica** – Provides APIs and SDKs for building medical decision support systems using probabilistic reasoning and medical knowledge graphs. Despite its flexibility, it requires significant customization for regional diseases and local hospital practices [11].

In addition, many international systems employ advanced AI techniques such as deep learning, transformer-based language models, and hybrid rule-based reasoning to improve accuracy. Studies show that integrating AI with structured clinical knowledge significantly improves triage accuracy and reduces misdiagnosis [12, 13].

2.3 Research Contributions

Academic research in AI-based healthcare support has highlighted several contributions relevant to NeuraHealth:

- **RAG-Based Systems in Healthcare** – Retrieval-Augmented Generation (RAG) models combine large language models with structured knowledge to provide contextually accurate responses, improving chatbot reliability [12].
- **Symptom Checkers: Accuracy and Impact** – Empirical studies indicate that AI-driven symptom checkers improve early diagnosis, reduce unnecessary appointments, and enhance patient satisfaction when integrated with triage systems [13].

- **AI-Driven Triage and Referrals** – Evidence suggests that AI-assisted triage improves patient throughput, reduces wait times, and provides doctors with structured preliminary information [14].
- **Natural Language Processing (NLP) in Healthcare** – NLP techniques, including entity recognition, intent classification, and semantic similarity, are critical for interpreting patient inputs accurately and mapping them to medical knowledge bases.

2.4 Identified Gaps and Opportunities

Despite the existence of digital healthcare solutions, no existing Pakistani system provides:

1. AI-based symptom interpretation tailored to local disease prevalence.
2. Test recommendation logic based on initial symptom clusters.
3. Specialist mapping with direct integration into appointment scheduling.
4. Seamless integration of pre-consultation patient summaries for doctors.
5. Lightweight, hospital-compatible AI models that comply with local data privacy regulations.

These gaps demonstrate a strong opportunity for NeuraHealth. By combining natural language understanding, AI classification algorithms, and workflow automation, it offers a holistic solution that can improve patient guidance, reduce administrative burden, and streamline hospital operations.

2.5 Summary

In conclusion, while local and global systems provide valuable functionalities, there is a clear lack of **comprehensive AI-driven healthcare automation tailored to regional contexts**. NeuraHealth addresses these gaps by offering an integrated, intelligent, and scalable platform capable of guiding patients accurately, recommending tests, and supporting hospital workflows efficiently.

Chapter 3

Requirement Specifications

This chapter outlines the key functional and non-functional requirements of the NeuraHealth system. It begins by evaluating the limitations of existing healthcare management systems and highlights how NeuraHealth addresses these issues through a smart, AI-powered chatbot-based solution. The system aims to assist patients, automate administrative workflows, and streamline hospital operations, with a special focus on helping patients identify the right specialist based on symptoms.

3.1 Existing System

Traditional hospital systems rely heavily on manual workflows [15] for patient registration, appointment scheduling, doctor consultations, and symptom intake. Common limitations include:

- Patients often don't know which specialist to consult based on their symptoms. [16]
- Long queues at the reception due to inefficient appointment processes.
- Repeated collection of the same patient information at different counters.
- No automated pre-checkup data collection and doctor briefing.[17]
- Lack of intelligent triage or test recommendations based on patient complaints.
- Limited or no integration of AI-powered systems in public sector hospitals.
- No predictive analytics for patient risk assessment and hospital resource planning.[18]

3.2 Proposed System

NeuraHealth is a hospital-specific, AI-powered smart healthcare automation system. The system is designed to integrate seamlessly into hospital infrastructure with future scalability in mind.[19] Its core component is a chatbot that interacts with patients to:

- Guide them to the right specialist based on symptom inputs using AI-powered triage.[16]
- Recommend potential tests and precautionary tips based on symptom analysis.
- Collect patient history and automatically generate comprehensive summaries for doctors.[17]
- Schedule appointments intelligently and issue automated reminders.
- Minimize front-desk workload by automating receptionist tasks.
- Provide predictive analytics for health risks and hospital operations.[18]
- Offer emotional support through sentiment-aware responses.

Modular Architecture: Built with modular APIs, supporting on-premise or cloud-based deployment, and integration via iframe for chatbot access across hospital terminals or websites.

AI-Powered Core: The system uses NLP, machine learning models, and custom LLM integration for symptom interpretation, risk prediction, and intelligent conversation management.

The NeuraHealth system is designed to enhance patient experience, reduce administrative burden, and improve clinical decision-making. By leveraging AI for symptom triage and predictive analytics, it supports faster, more accurate healthcare delivery. Its modular design ensures adaptability to various hospital setups, while its user-friendly interfaces provide seamless access for patients, doctors, and administrative staff. The system aims to not only streamline hospital operations but also contribute to better health outcomes through proactive monitoring and timely interventions.

Table 3.1: Functional and Non-Functional Requirements of NeuraHealth System

Functional Requirements	
FR ID	Functional Requirement Description
FR1	The system shall allow patients to input symptoms via AI chatbot with natural language processing.
FR2	The system shall identify the relevant department or specialist based on symptoms using AI triage.
FR3	The system shall suggest possible medical tests and health tips based on symptom analysis.
FR4	The system shall schedule appointments via chatbot interaction with real-time availability.
FR5	The system shall generate AI-powered patient summaries and send to doctor's dashboard.
FR6	The system shall allow doctors to view, update, and confirm patient summaries and appointments.
FR7	The system shall allow admin staff to monitor system logs, patient flow, and analytics.
FR8	The system shall provide predictive analytics for health risks and appointment no-shows.
FR9	The system shall analyze patient sentiment and adapt responses accordingly.
FR10	The system shall manage role-based access for patients, doctors, and administrators.
Non-Functional Requirements	
NFR ID	Non-Functional Requirement Description
NFR1	The system shall ensure response times of less than 2 seconds per chatbot interaction.
NFR2	The system shall comply with hospital data privacy and local security regulations.
NFR3	The system shall be accessible on both desktop and mobile terminals.
NFR4	The system shall be modular and support future integration with EHR systems.
NFR5	The system shall support both on-premise deployment and future cloud scalability.
NFR6	The system shall maintain 99.5% uptime for critical healthcare operations.
NFR7	The system shall support concurrent access by multiple users without performance degradation.

3.3 Requirement Specifications

3.4 Use Case Diagrams

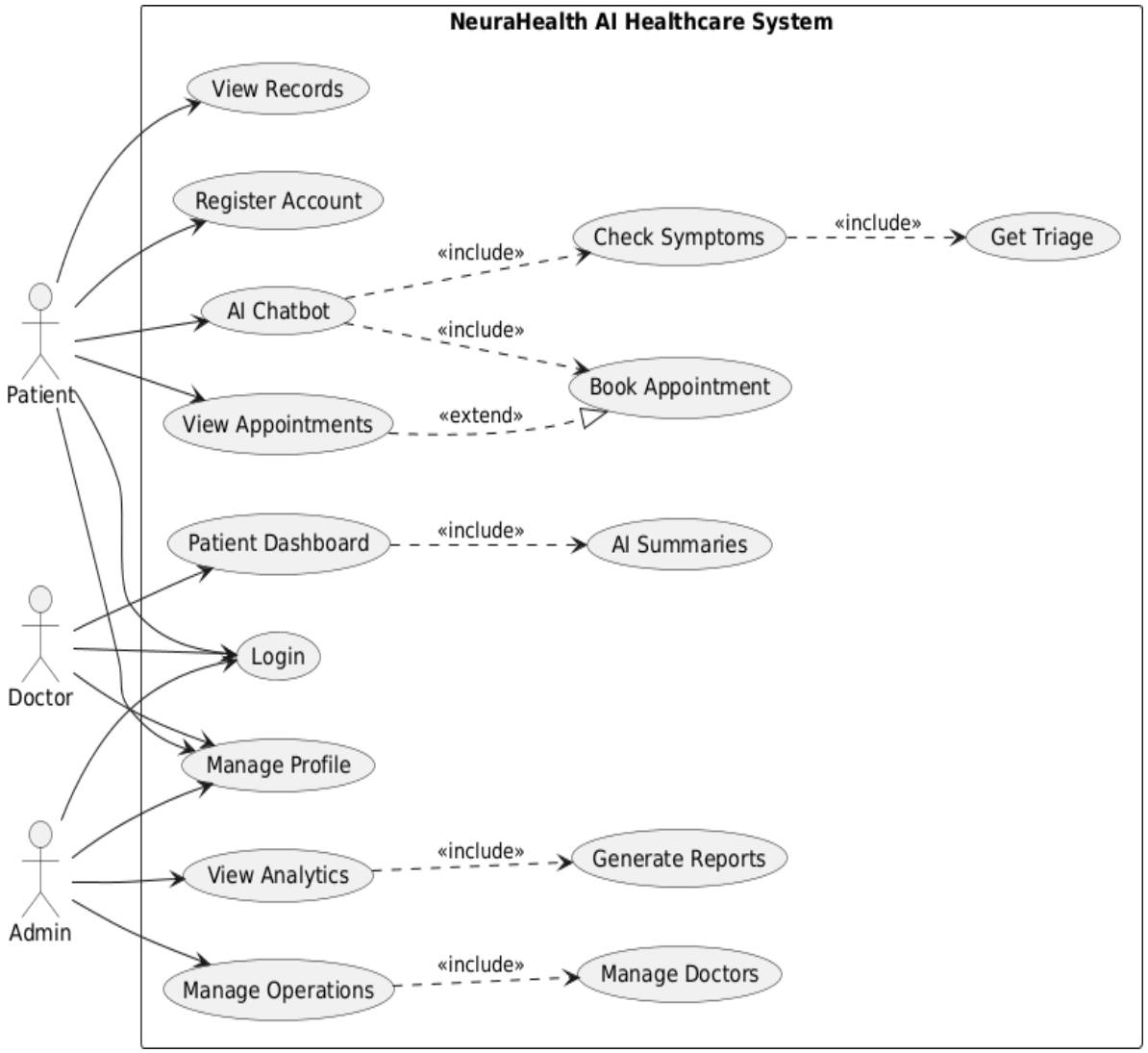


Figure 3.1: Overview of the NeuraHealth system's use cases, showing interactions among patients, doctors, and admins.

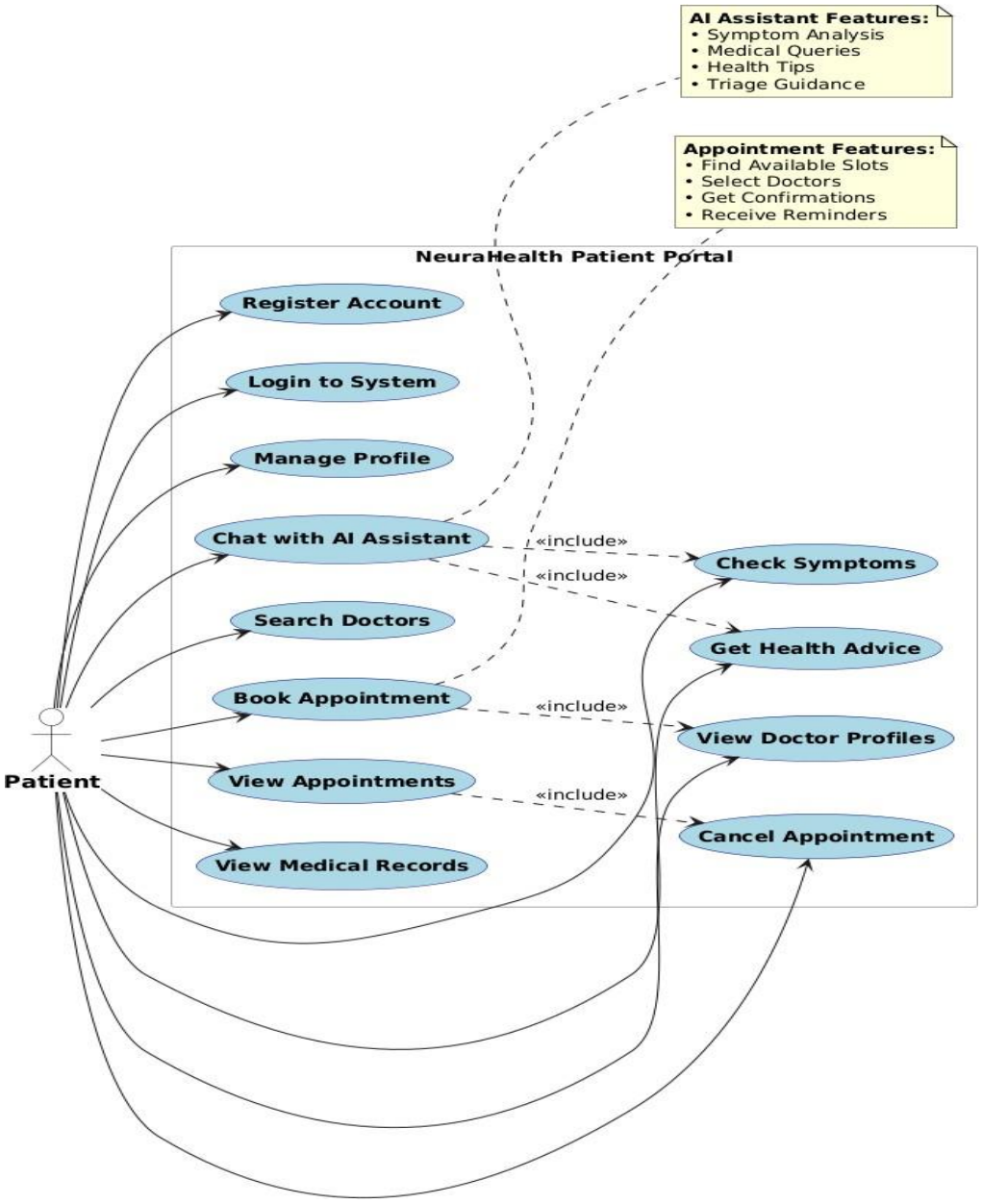


Figure 3.2: Detailed use cases for patients, including symptom input, chatbot interaction, and appointment booking.

Doctor Use Cases

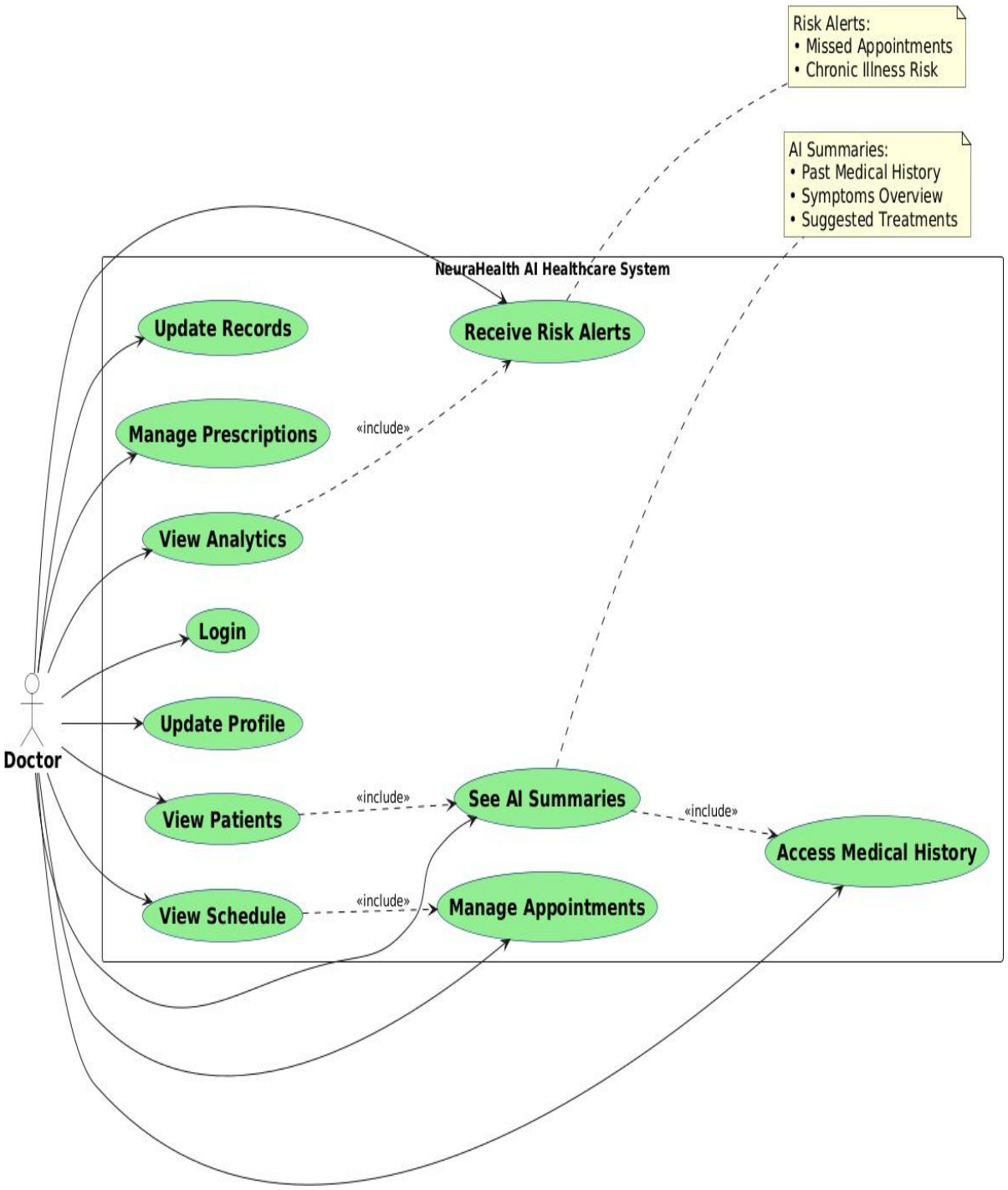


Figure 3.3: Doctor-specific use cases, covering patient monitoring, AI-assisted summaries, and appointment management.

Admin Use Cases

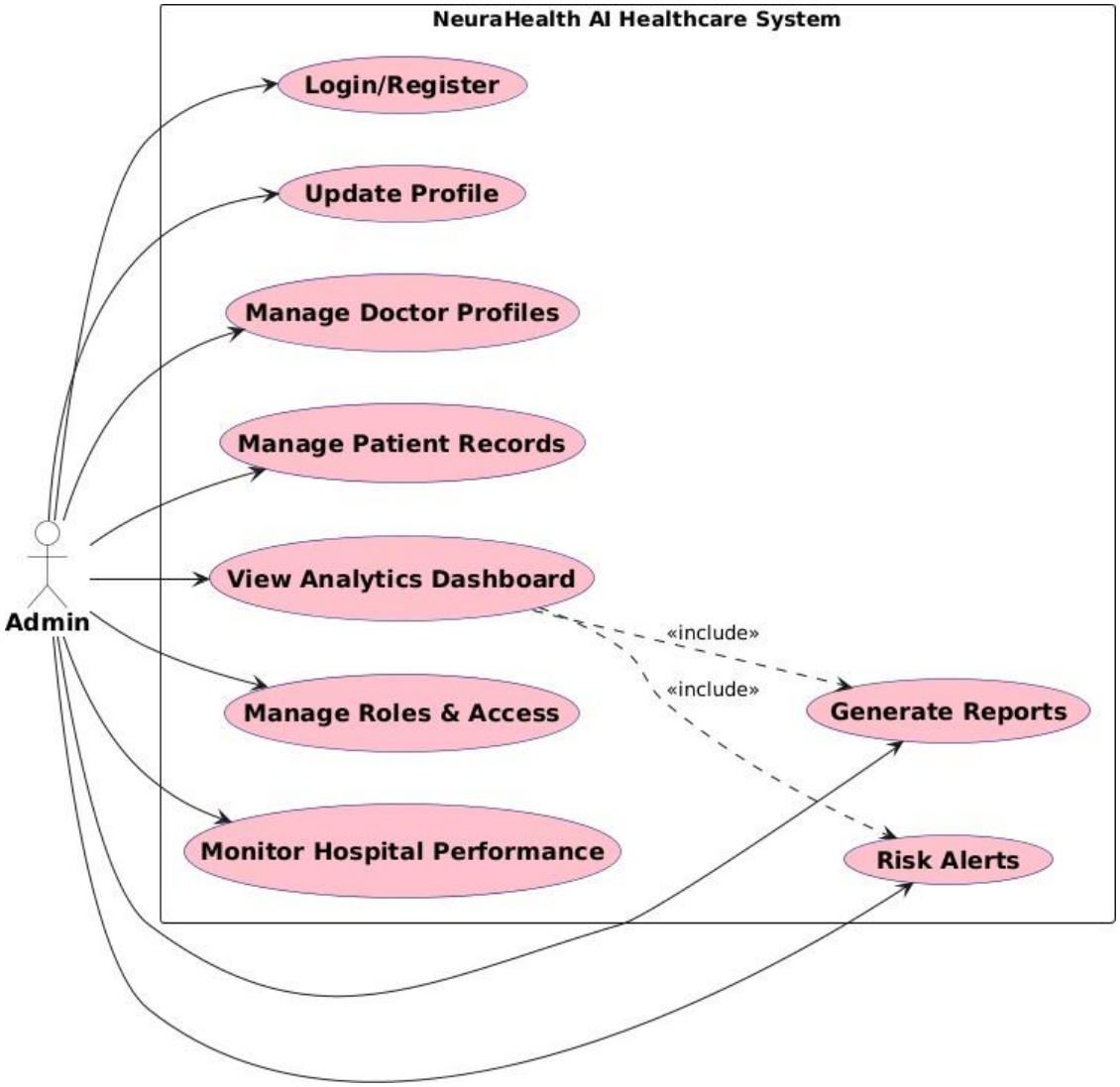


Figure 3.4: Admin use cases: managing doctors, viewing analytics, and handling role-based access.

3.5 Detailed Use Case Specifications (Condensed)

Table 3.2: **Use Case UC-1: User Registration**

Use Case ID	UC-1
Use Case Name	User Registration
Actors	Patient
Description	Patient creates an account by providing basic details.
Trigger	New patient signs up.
Preconditions	N/A
Post Conditions	Account created; confirmation sent.
Normal Flow	**User:** 1. User clicks on the "Register" button. 2. User enters username and password. 3. User clicks on the "Sign In" button.
	System: 4. System displays the Register page. 5. System saves the entered credentials. 6. System redirects the user to their respective dashboard.
Alternative Flows	N/A
Exceptions	Duplicate account; system unavailable
Assumptions	User has internet access

Table 3.3: **Use Case UC-2: User Login**

Use Case ID	UC-2
Use Case Name	User Login
Actors	Patient, Doctor, Admin
Description	Users log in to access dashboards.
Trigger	User enters credentials.
Preconditions	User must have registered account.
Post Conditions	User logged in; dashboard displayed.
Normal Flow	**User:** 1. User clicks on the “Login” button. 2. User enters username and password. 3. User clicks “Sign In”.
	System: 4. System displays the "login" page 5. System validates credentials. 6. System redirects the user to the dashboard.
Alternative Flows	N/A
Exceptions	Invalid login; account locked
Assumptions	User has internet access

Table 3.4: **Use Case UC-3: Chatbot Interaction**

Use Case ID	UC-3
Use Case Name	Chat with AI System
Actors	Patient, AI System
Description	AI chatbot provides guidance and symptom checks.
Trigger	Patient initiates chat.
Preconditions	Patient logged in; AI system online
Post Conditions	Response provided; conversation logged.
Normal Flow	**User:** 1. Patient opens chat interface. 2. Patient enters a message or symptom description.
	System: 3. System greets Patient. 4. System analyzes input and generates a response 5. System logs the conversation.
Alternative Flows	N/A
Exceptions	Input not understood; chatbot unavailable
Assumptions	Patient has internet access

Table 3.5: **Use Case UC-4: Appointment Booking**

Use Case ID	UC-4
Use Case Name	Book Appointment
Actors	Patient, Doctor
Description	Schedule an appointment with a doctor.
Trigger	Patient requests booking.
Preconditions	Patient logged in; doctor available.
Post Conditions	Appointment confirmed; notifications sent.
Normal Flow	**User:** 1. Patient selects "Book Appointment". 2. Patient selects a doctor and preferred time. 3. Patient confirms the booking.
	System: 4. System displays doctor availability. 5. System reserves the time slot. 6. System sends confirmation to both doctor and patient.
Alternative Flows	N/A
Exceptions	Doctor unavailable; system error
Assumptions	Patient has internet access

Table 3.6: **Use Case UC-5: View Appointments**

Use Case ID	UC-5
Use Case Name	View Appointments
Actors	Patient
Description	View upcoming appointments.
Trigger	Patient selects "View Appointments".
Preconditions	Patient logged in; appointments exist.
Post Conditions	Appointment list displayed.
Normal Flow	**User:** 1. Patient opens "View Appointments".
	System: 2. System retrieves scheduled appointments. 3. System displays list to the patient.
Alternative Flows	N/A
Exceptions	No appointments; system unavailable
Assumptions	Patient has internet access

Table 3.7: **Use Case UC-6: Cancel Appointment**

Use Case ID	UC-6
Use Case Name	Cancel Appointment
Actors	Patient, Doctor
Description	Cancel an existing appointment.
Trigger	Patient chooses to cancel.
Preconditions	Appointment exists; patient logged in.
Post Conditions	Appointment cancelled; notifications sent.
Normal Flow	**User:** 1. Patient opens appointment list. 2. Patient selects appointment to cancel. 3. Patient confirms cancellation.
	System: 4. System displays appointment list 5. System updates appointment status to "cancelled" 6. System updates doctor's availability 7. System notifies doctor and patient
Alternative Flows	N/A
Exceptions	Appointment not found; system error
Assumptions	Patient has internet access

Table 3.8: **Use Case UC-7: AI Symptom Analysis**

Use Case ID	UC-7
Use Case Name	AI Symptom Analysis
Actors	Patient, AI System
Description	AI suggests possible conditions based on symptoms.
Trigger	Patient inputs symptoms.
Preconditions	Patient logged in; AI system online.
Post Conditions	Suggestions displayed.
Normal Flow	**User:** 1. Patient enters symptoms in chatbot.
	System: 2. System extracts key symptoms using NLP. 3. System runs AI symptom analysis model. 4. System displays possible conditions and recommendations.
Alternative Flows	N/A
Exceptions	Symptoms unclear; AI unavailable
Assumptions	Patient has internet access

Table 3.9: **Use Case UC-8: View Patient List - Doctor**

Use Case ID	UC-8
Use Case Name	View Patient List
Actors	Doctor
Description	See assigned patients and appointments.
Trigger	Doctor logs in.
Preconditions	Doctor logged in; patients assigned.
Post Conditions	List displayed.
Normal Flow	**User:** 1. Doctor logs into system. 2. Doctor opens "Patient List".
	System: 3. System retrieves assigned patients and appointments. 4. System displays the complete list.
Alternative Flows	N/A
Exceptions	No patients; database error
Assumptions	Doctor has internet access

Table 3.10: **Use Case UC-9: Manage Doctor Profiles**

Use Case ID	UC-9
Use Case Name	Manage Doctor Profiles
Actors	Admin
Description	Add or update doctor information.
Trigger	Admin selects "Manage Profiles".
Preconditions	Admin logged in; doctor exists in system.
Post Conditions	Profiles updated.
Normal Flow	**User:** 1. Admin opens "Manage Doctor Profiles". 2. Admin selects doctor or adds new profile. 3. Admin submits updated information.
	System: 4. System displays profiles. 5. System updates doctor profile in database. 6. System confirms successful update.
Alternative Flows	N/A
Exceptions	Invalid data; system error
Assumptions	Admin has internet access

Table 3.11: **Use Case UC-10: View Analytics**

Use Case ID	UC-10
Use Case Name	View Analytics
Actors	Admin
Description	View usage, trends, and AI performance.
Trigger	Admin selects "Analytics".
Preconditions	Admin logged in; data available.
Post Conditions	Dashboard displayed.
Normal Flow	**User:** 1. Admin opens analytics dashboard.
	System: 2. System retrieves usage and performance data. 3. System displays charts, graphs, and analytics.
Alternative Flows	N/A
Exceptions	Data unavailable; system error
Assumptions	Admin has internet access

Table 3.12: **Use Case UC-11: AI Patient Summary**

Use Case ID	UC-11
Use Case Name	AI Patient Summary
Actors	AI System, Doctor
Description	Generates structured patient summary.
Trigger	Symptom analysis completed.
Preconditions	Doctor logged in; patient data exists.
Post Conditions	Summary displayed on dashboard.
Normal Flow	**System:** 1. System extracts patient history and chat data. 2. AI generates structured medical summary. 3. System updates doctor dashboard with summary.
	User (Doctor): 4. Doctor views generated summary.
Alternative Flows	N/A
Exceptions	Insufficient data; AI module unavailable
Assumptions	System online; patient data complete

Table 3.13: **Use Case UC-12: Predictive Health Risk**

Use Case ID	UC-12
Use Case Name	Predictive Health Risk
Actors	AI System, Doctor, Patient
Description	AI predicts health risks and appointment no-shows.
Trigger	New patient data or appointment scheduled.
Preconditions	Patient logged in; historical data available.
Post Conditions	Risk scores generated and alerts sent to doctor.
Normal Flow	**System:** 1. System retrieves patient history and appointment data. 2. AI model calculates risk scores. 3. System sends alerts to doctor if risk is high.
	User (Doctor/Patient): 4. Doctor/Patient views risk report.
Alternative Flows	N/A
Exceptions	Insufficient data; low confidence prediction
Assumptions	System online; patient history available

Table 3.14: **Use Case UC-13: Emotional Sentiment Analysis**

Use Case ID	UC-13
Use Case Name	Emotional Sentiment Analysis
Actors	AI System, Patient
Description	Detect patient emotion and adapt responses accordingly.
Trigger	Patient sends message in chat.
Preconditions	Patient logged in; AI system online.
Post Conditions	Emotional tone detected; chatbot responses adjusted.
Normal Flow	**User:** 1. Patient sends message in chat.
	System: 2. System performs sentiment detection. 3. System classifies emotion (e.g., anxious, neutral, upset). 4. System adjusts chatbot response tone.
Alternative Flows	N/A
Exceptions	Ambiguous input; AI module unavailable
Assumptions	Patient has internet access

Table 3.15: **Use Case UC-14: Manage Role Based Access**

Use Case ID	UC-14
Use Case Name	Manage Role-Based Access
Actors	Admin
Description	Assign and update user roles.
Trigger	Admin selects “Manage Roles”.
Preconditions	Admin logged in; users exist in system.
Post Conditions	Roles updated and access enforced.
Normal Flow	**User:** 1. Admin opens “Role Management”. 2. Admin selects user and assigns a role. 3. Admin saves changes.
	System: 4. System validates role permissions. 5. System updates access control settings. 6. System confirms update to admin.
Alternative Flows	N/A
Exceptions	Invalid role; system error
Assumptions	Admin has internet access

Chapter 4

Design

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements.[20] The NeuraHealth project leverages modular design principles, AI integration, and user-centric interfaces to build a scalable and intelligent hospital management system. This chapter details the design methodology, architecture, constraints, and interfaces of the proposed system.

4.1 System Architecture

NeuraHealth adopts a modular microservices-based architecture [21, 22], enabling separation of concerns and flexible deployment options. The system consists of the following core components: [23]

- **Receptionist Chatbot Module**: Collects symptoms and personal details, recommends departments, schedules appointments, and shares summaries with doctors.
- **Doctor Dashboard Module**: Allows doctors to view patient history, summaries, and update post-visit notes.
- **Admin Panel**: Tracks system usage, logs, and manages hospital operations.
- **Database Layer**: Stores patient data, doctor details, appointment logs, and summaries.
- **LLM Engine (Custom)**: A locally trained lightweight model performing symptom understanding and triage logic without external GPT APIs. *An external LLM (Groq) is used only as a fallback in rare cases where the custom model fails or times out.*

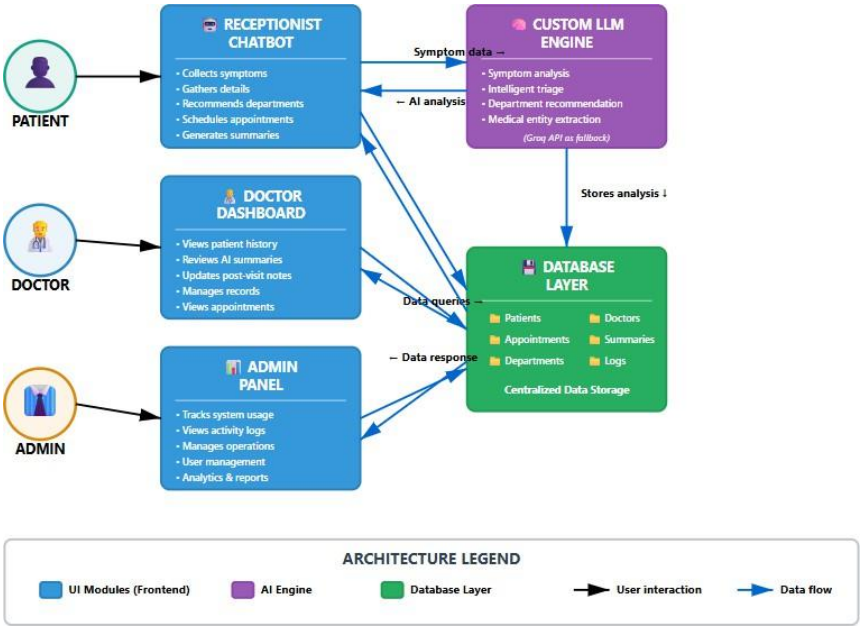


Figure 4.1: High-Level System Architecture of NeuraHealth

4.2 Design Constraints

The design of NeuraHealth is influenced by several key constraints:

- **No use of external pretrained LLMs**: Custom LLM is prioritized; external LLM is fallback only.
- **Offline/On-premise Deployment**: Operates within hospital intranet environments without cloud dependency.
- **Scalability**: System must support future integration with cloud services and EHR systems.[23]
- **Security**: Compliance with data privacy laws, end-to-end encryption of sensitive records, and role-based access control.[24]

4.3 Design Methodology

NeuraHealth follows an Agile methodology with iterative design cycles. The design approach is based on:

- **Object-Oriented Design (OOD)** using UML class diagrams and interaction models.[25]
- **Modular API design** for decoupled components.[21]

- **Component-driven GUI design** using modern frontend frameworks.
- **UML modeling**: Sequence diagrams, component diagrams, and class diagrams for clarity.

4.4 High Level Design

1. Conceptual View

- Patient interacts with chatbot → enters symptoms.
- System processes input → recommends department → offers appointment slot.
- Patient info stored → summary generated → shared with doctor.
- Doctor updates visit → system stores record.

2. Process View

- Real-time chatbot interaction using NLP engine.[26]
- Async background tasks for appointment management and notifications.

3. Physical View

- System runs on hospital's local server or intranet machine.
- Optionally dockerized modules for container-based deployment.[27]

4. Module View

- `frontend/` → Patient and doctor UIs
- `chatbot_engine/` → NLP processing + symptom logic
- `database/` → MongoDB or PostgreSQL schema [28]
- `admin/` → Analytics and logs

5. Security View

- Token-based authentication for all modules.
- Role-based access control for doctors, patients, and admin.
- Audit logs for each sensitive interaction.

4.5 Low Level Design

- Chatbot → Parser → Symptom Classifier → Specialist Mapper → Appointment Handler
- Dashboard → Patient Card Viewer → Summary Renderer → Visit Logger
- Admin Panel → User Monitor → Log Viewer → Test Recommender

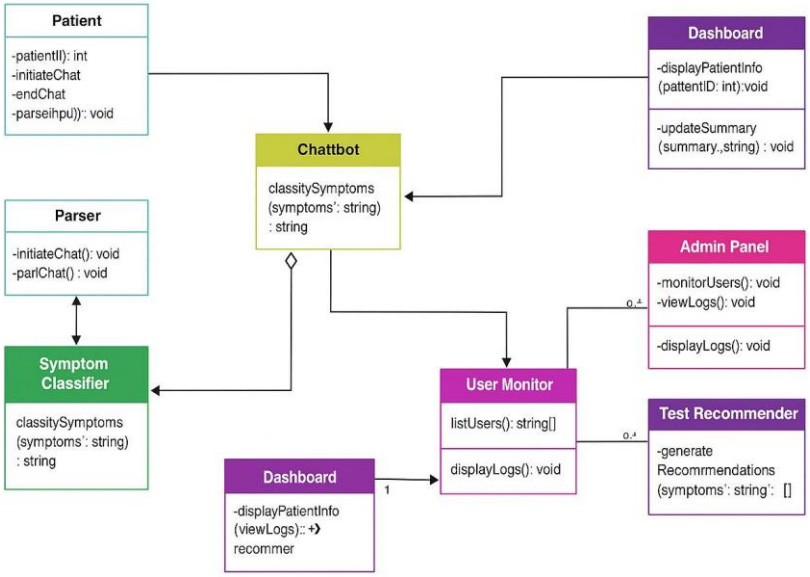


Figure 4.2: Core Class Diagram for Chatbot and Doctor Interaction Modules

4.6 Database Design

The backend database uses a hybrid schema [28] with relational storage for structured entities and document storage for summaries. Key tables:

- **Patients(id, name, age, gender, contact, history)**
- **Doctors(id, name, department, availability)**
- **Appointments(id, patient_id, doctor_id, datetime, status)**
- **Summaries(appointment_id, symptoms, tests, advice, timestamp)**

pdflscape

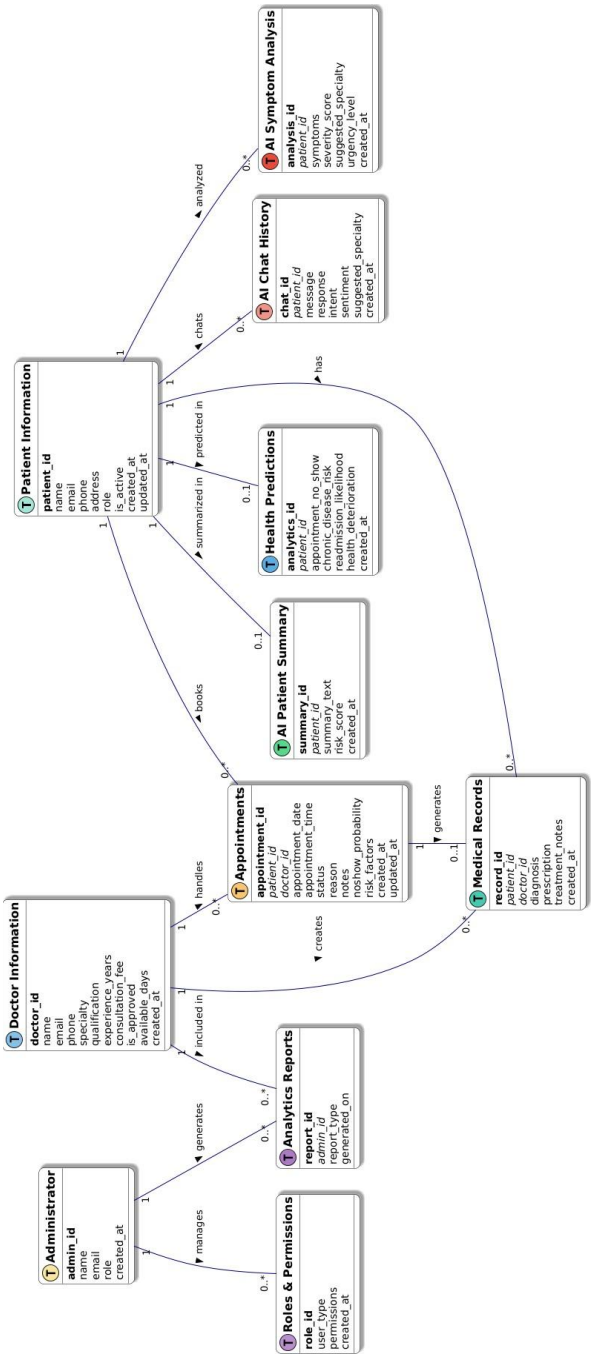


Figure 4.3: Entity–Relationship (ER) Diagram of NeuraHealth Database

4.7 GUI Design

- **Patient Interface:** Chatbot bubble UI, symptom entry, feedback loop, and confirmation card.
- **Doctor Interface:** Table view of upcoming patients, auto-generated summaries, and response panel.
- **Admin Dashboard:** Graphs, tables for system usage, login logs, and error tracking.
- **Core Functional Interfaces:** Includes chatbot, appointment booking, and analytics.

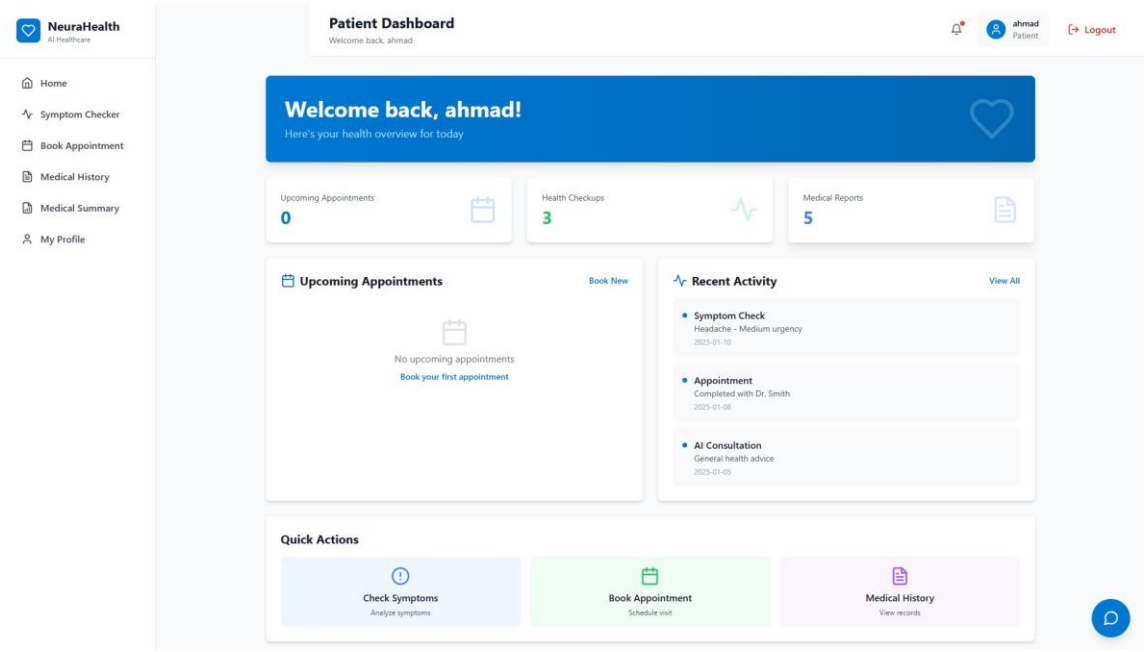


Figure 4.4: Patient Dashboard

Core Functional Interfaces: Includes chatbot, appointment booking, and analytics.

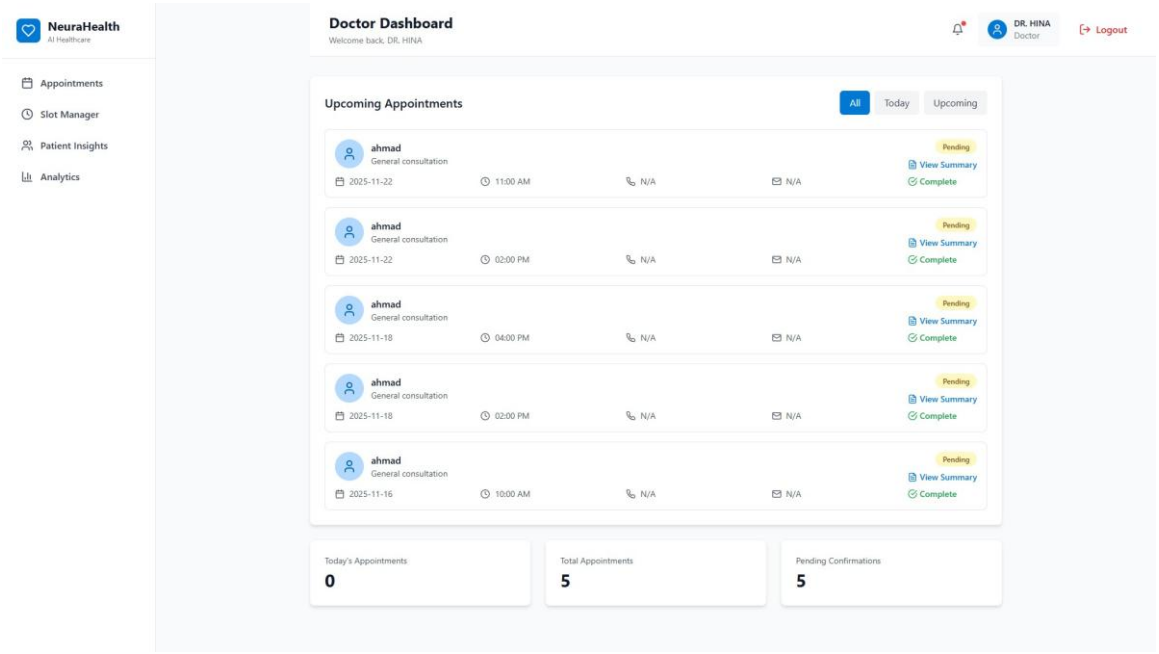


Figure 4.5: Doctor Dashboard

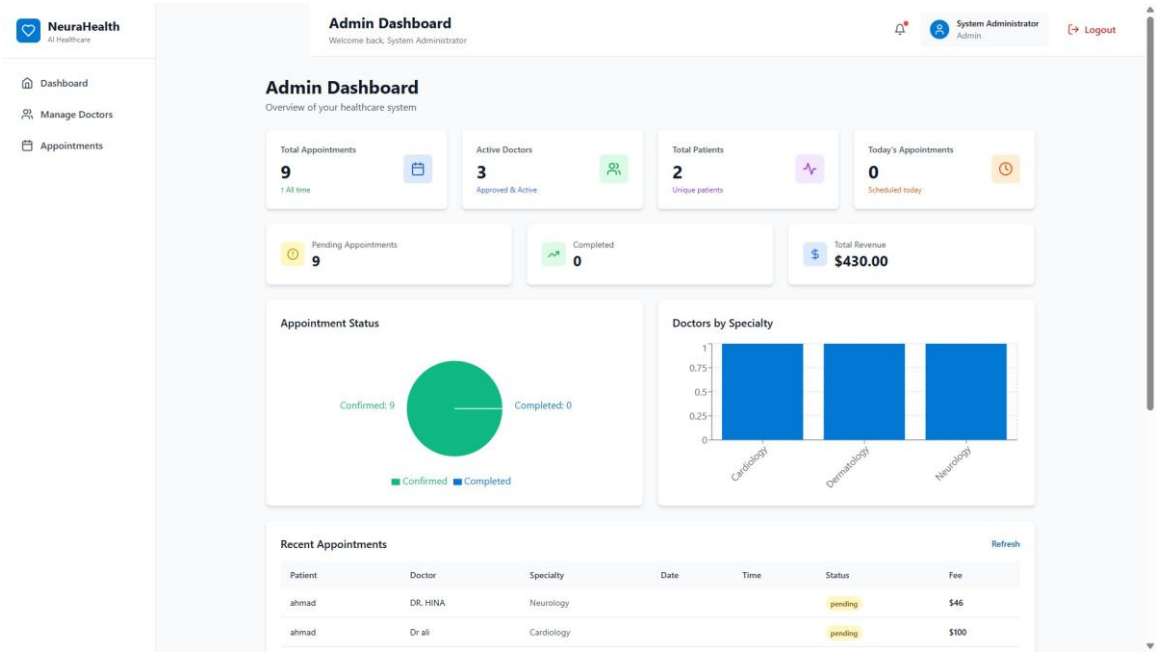


Figure 4.6: Admin Dashboard

0.45

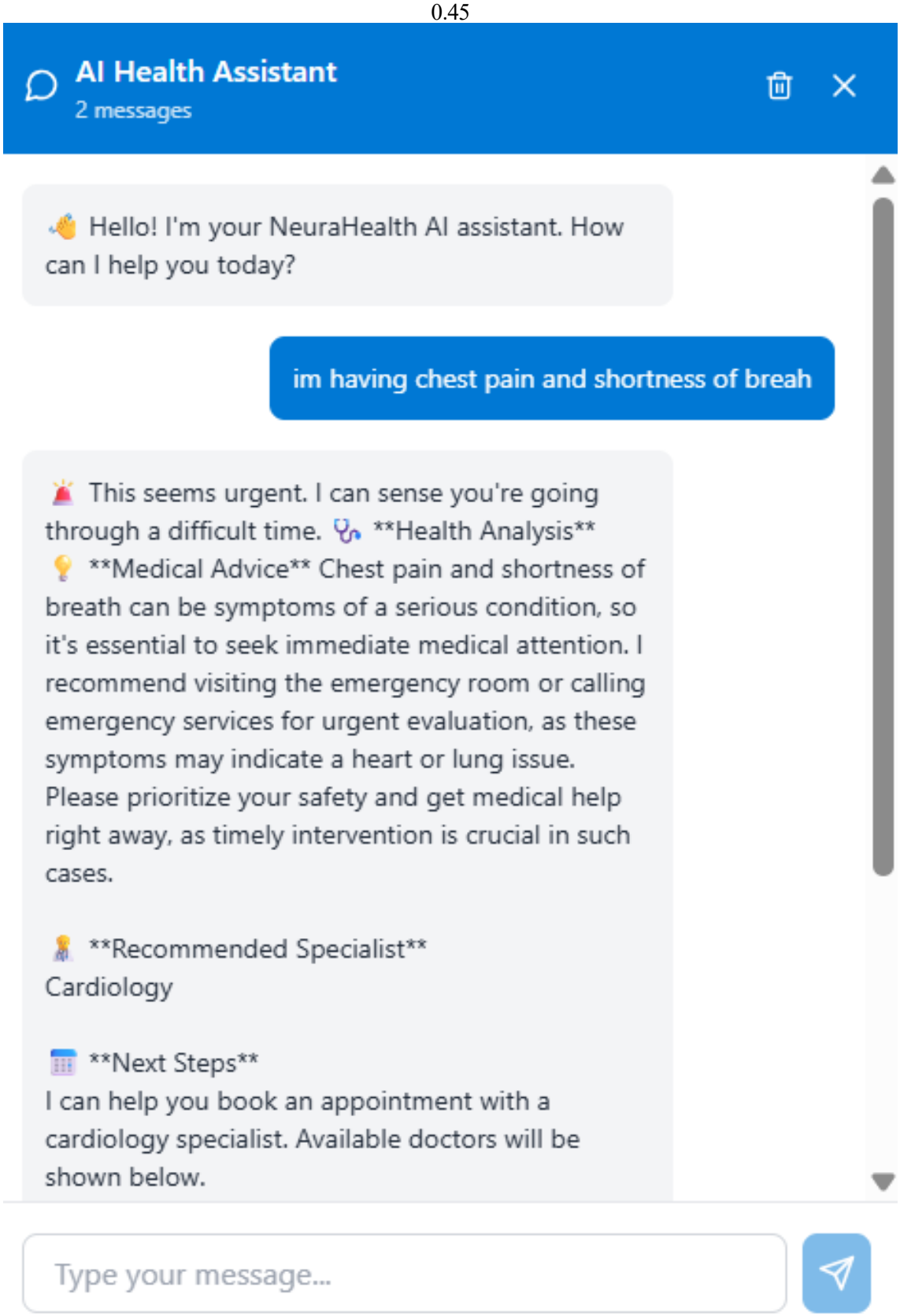


Figure 4.7: AI Chatbot Interface

0.45

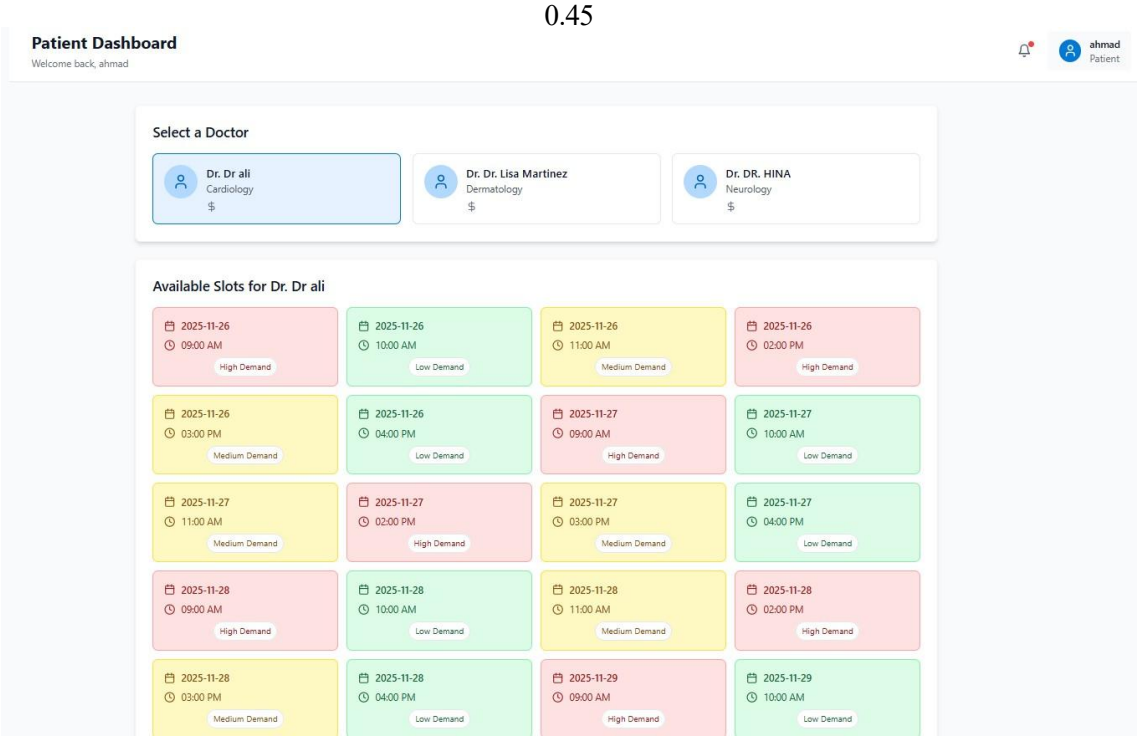


Figure 4.8: Appointment Booking Interface

0.45

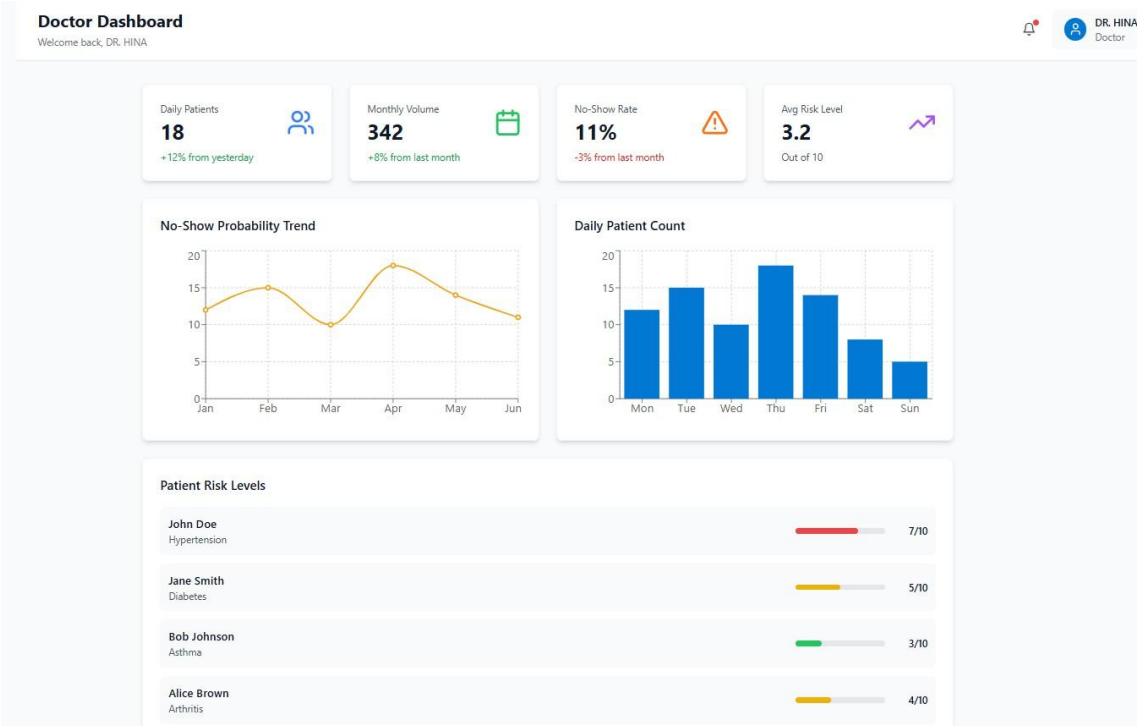


Figure 4.9: Appointment and Analytics Interfaces of NeuraHealth

Chapter 5

System Implementation

Implementation is the stage where the designed system is developed into a working product [20]. The "NeuraHealth – AI Enhanced Healthcare Platform" has been implemented as an intelligent, secure, and scalable web-based application connecting patients, doctors, and administrators through AI and predictive analytics. The system ensures efficient healthcare delivery via data-driven decision-making, intelligent triage, personalized patient interaction, and continuous monitoring of hospital operations.

5.1 System Architecture

The architecture of NeuraHealth follows a modular, service-oriented design [23, 21] separating presentation, business logic, AI intelligence, and data management. This approach supports scalability, maintainability, and real-time processing [23].

5.1.1 Component Layers

- **Frontend (Client Layer):** Responsive web interface for patients, doctors, and administrators.
- **Backend (Application Layer):** FastAPI-based RESTful APIs for AI functionalities, including smart symptom checking, predictive analytics, conversational AI, sentiment analysis, and secure data handling.
- **AI Intelligence Layer:** Multiple ML/NLP modules process inputs, predict risks, and generate recommendations.
- **Database Layer:** MongoDB Atlas stores all relevant user, medical, and interaction data securely.

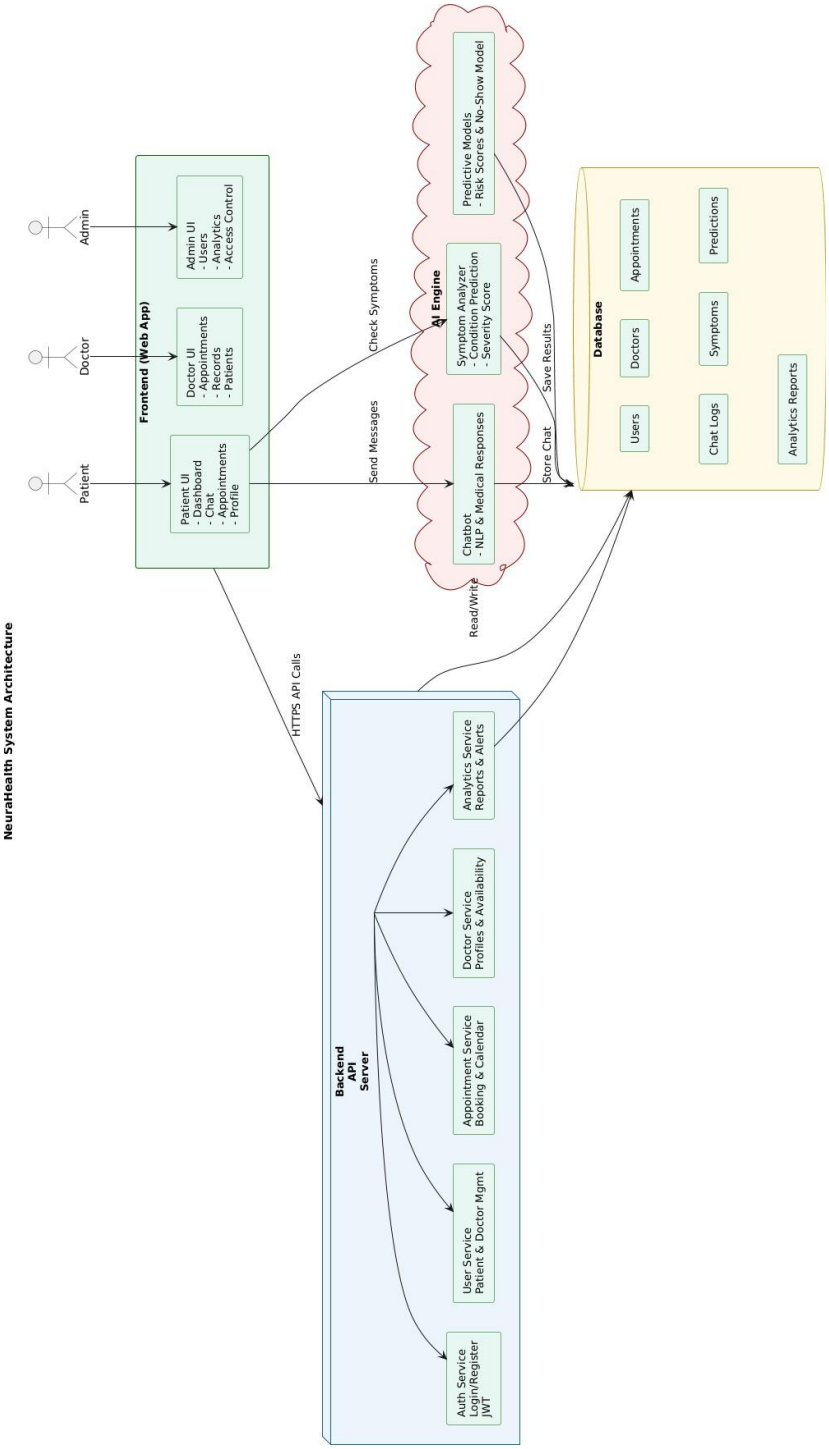


Figure 5.1: NeuraHealth System Architecture (Landscape, Full Page, Large Fonts)

5.2 Frontend Implementation

5.2.1 Patient Dashboard

- AI Chatbot for symptom consultation
- Appointment booking and cancellation interface
- Personal health risk visualization
- Sentiment-based mood and wellness tracking [29]

5.2.2 Doctor Dashboard

- Appointment calendar
- Patient profiles with AI-generated summaries
- Predictive insights for no-shows and high-risk patients
- Analytics charts for patient trends and workload

5.2.3 Administrator Dashboard

- Overview of all users
- Resource utilization and hospital performance analytics
- Doctor slot management and system configuration
- Audit logs for security and compliance

5.3 Application Access Security

- JWT-based authentication [24]
- Role-Based Access Control (RBAC)
- Password hashing with bcrypt
- Session tokens stored securely with expiration
- TLS encryption and MongoDB AES-256 at rest
- CORS policy enforcement
- Audit logging of all critical operations

5.4 Database Security

- IP whitelisting and role-based permissions
- SCRAM-SHA-256 authentication
- Server-side encryption with automatic key management
- Operation auditing for insert/update/delete
- Cloud backups with automatic failover [28]

5.5 Deployment

- Backend: FastAPI + Uvicorn
- Database: MongoDB Atlas Cluster
- Frontend: React PWA deployed on cloud
- Containerization: Docker + Docker Compose [27]
- Monitoring: MongoDB dashboards + server logs

5.6 Summary

The implementation of NeuraHealth demonstrates a seamless integration of AI and modern web technologies to build a secure, scalable, and intelligent healthcare management platform. Patients benefit from personalized interactions, intelligent symptom triage, and predictive risk assessments, while doctors gain actionable insights, AI-generated patient summaries, and predictive analytics for efficient decision-making. Administrators can **monitor hospital performance, manage resources, and ensure compliance** through real-time dashboards and audit logs.

The system is designed to operate on-premises or in hybrid cloud settings, supporting high availability, data privacy, and future expansion. Additionally, the architecture allows fallback to external LLMs like Groq in rare cases where the custom LLM fails or experiences a timeout, ensuring continuous system reliability. This implementation forms the foundation for data-driven, AI-enhanced hospital management, enabling better healthcare outcomes and optimized resource utilization.

Chapter 6

System Testing and Evaluation

System testing validates the complete and integrated NeuraHealth system to ensure it meets its functional and non-functional requirements [30, 31]. This chapter includes GUI Testing, Usability Testing, Performance Testing, Security Testing, and 10 essential functional test cases covering core system behavior.

6.1 Core Functional Test Cases

Below are the detailed real-user test cases for the NeuraHealth system, now including **screenshots for each test case**.

Table 6.1: Patient Registration

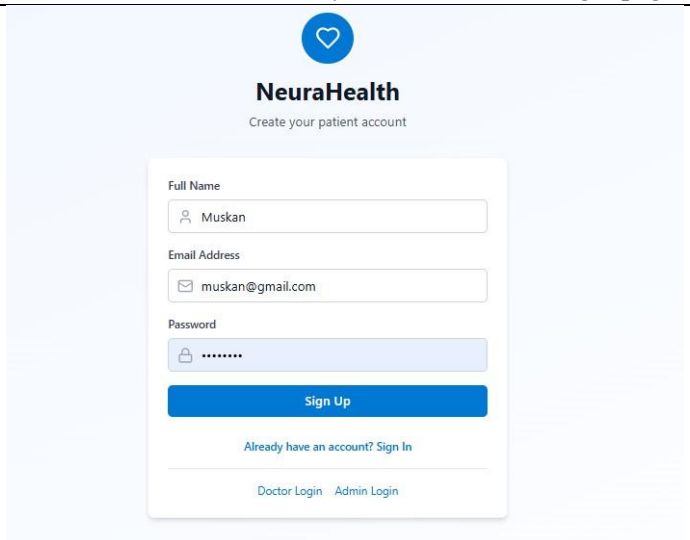
DATE	02-12-2025
TEST ID	TC-01
SYSTEM	NEURAHEALTH
TEST TYPE	Unit Testing
Objective	Register/Signup a new patient in the system
Status	Pass
Version	1.1
Input	Full Name: Muskan Email: muskan@gmail.com Password: muskan123 Press “Signup”
Expected Result	System should create the account and redirect to login page
Actual Result	Account created successfully and redirected to login page
Screenshot	

Table 6.2: Registration/login With Missing Field/Invalid credentials

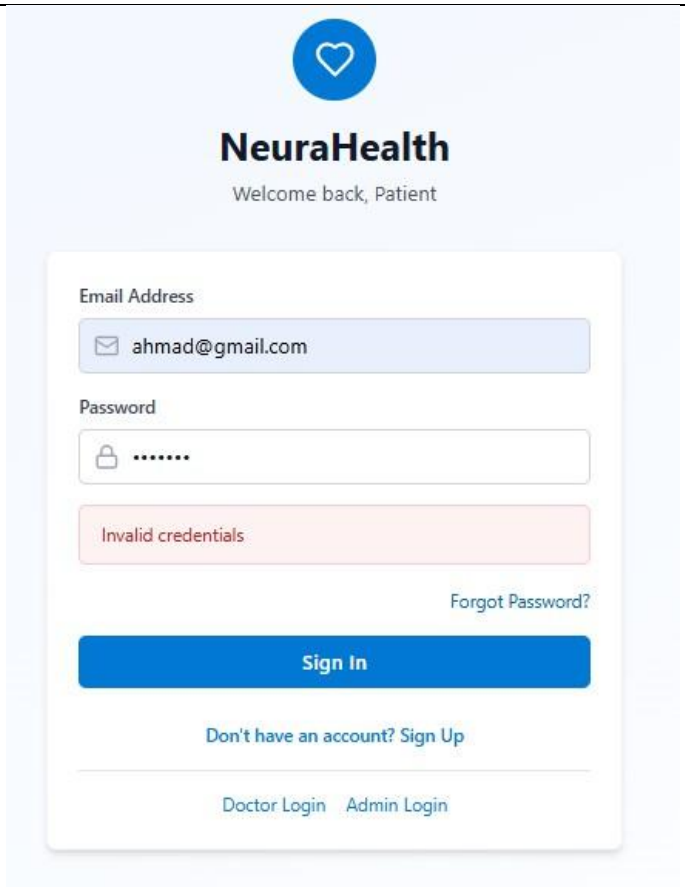
DATE	02-12-2025
TEST ID	TC-02
SYSTEM	NEURAHEALTH
TEST TYPE	Unit Testing
Objective	Validate registration with missing data
Status	Pass
Version	1.1
Input	Full Name: Ahmad Email: ahmad@gmail.com Password: wrong/missing Press “Sign in”
Expected Result	System shows error message for invalid credentials
Actual Result	Error displayed correctly
Screenshot	

Table 6.3: AI Chatbot Symptom Analysis

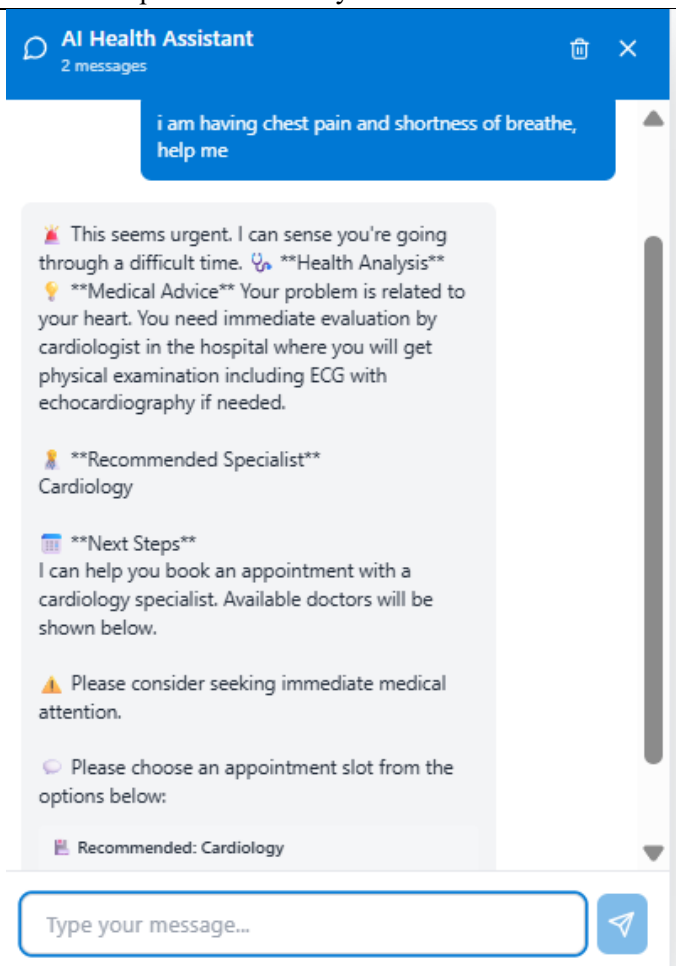
DATE	02-12-2025
TEST ID	TC-05
SYSTEM	NEURAHEALTH
TEST TYPE	Functional Testing
Objective	Check chatbot symptom analysis output
Status	Pass
Version	1.1
Input	Symptoms typed: “chest pain, shortness of breathe”
Expected Result	Chatbot returns risk level, suggested tests, and health tips and suggested speciality
Actual Result	Chatbot responded accurately
Screenshot	

Table 6.4: Table 5.6: Appointment Booking via Chatbot

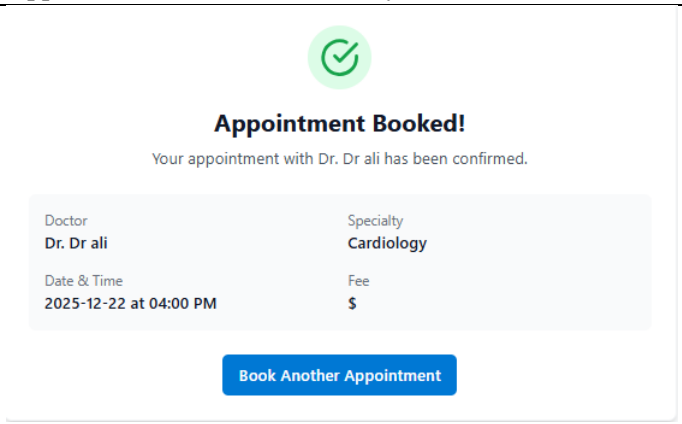
DATE	02-12-2025
TEST ID	TC-06
SYSTEM	NEURAHEALTH
TEST TYPE	Functional Testing
Objective	Validate appointment creation via chatbot
Status	Pass
Version	1.1
Input	Doctor: Dr. ali Date: 20-12-2025 Time: 02:00 PM
Expected Result	Appointment stored; confirmation displayed
Actual Result	Appointment booked successfully
Screenshot	

Table 6.5: Table 5.8: Admin Adding Doctor

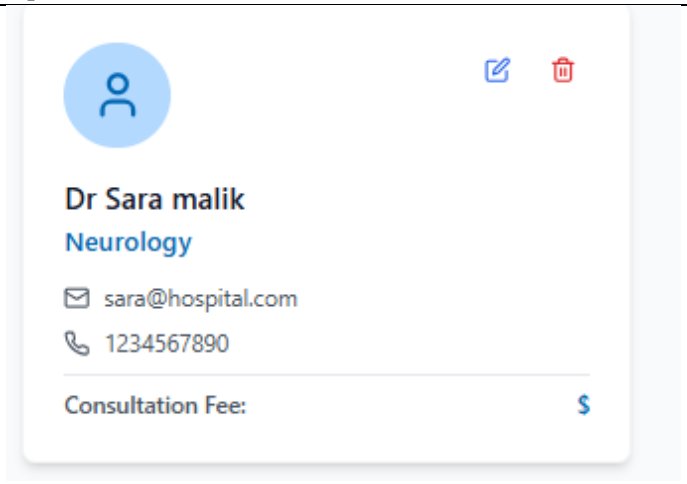
DATE	02-12-2025
TEST ID	TC-08
SYSTEM	NEURAHEALTH
TEST TYPE	Functional Testing
Objective	Add a new doctor profile
Status	Pass
Version	1.1
Input	Name: Dr. Sara Malik Specialty: Neurology Email: sara@hospital.com Qualification: MBBS Experience (years): 8 Consultation Fee ($): 45 Press “Add Doctor”
Expected Result	Doctor added to system
Actual Result	Operation successful
Screenshot	

Table 6.6: Patient Viewing Medical Summary

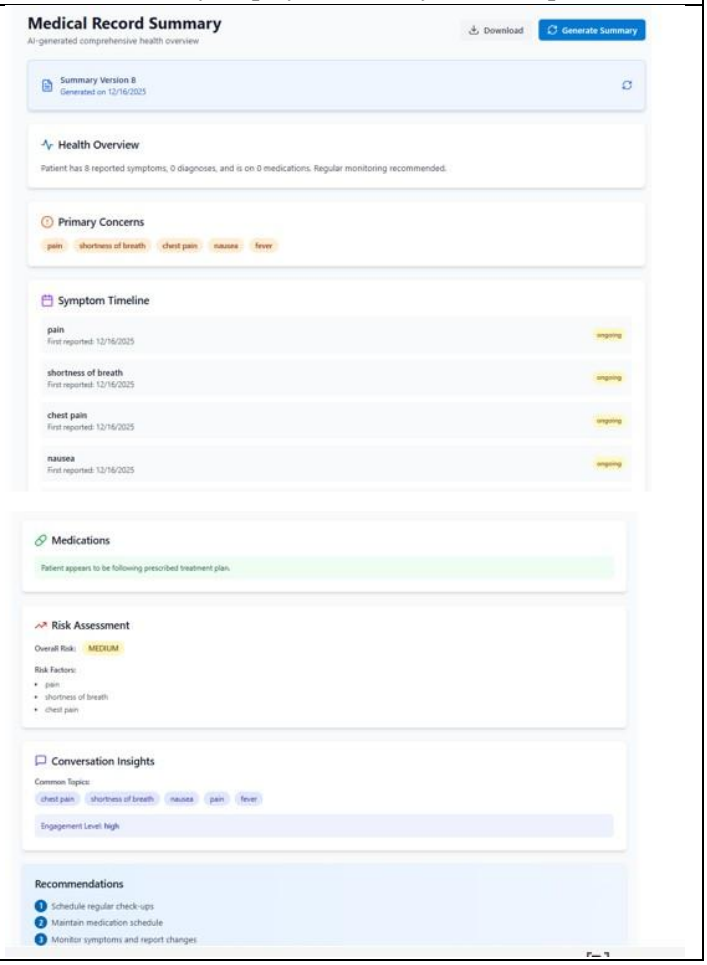
DATE	02-12-2025
TEST ID	TC-09
SYSTEM	NEURAHEALTH
TEST TYPE	Functional Testing
Objective	Verify that patient can view their medical summary
Status	Pass
Version	1.1
Input Click on “Medical Summary” from sidebar	Login as patient
Expected Result	System displays detailed patient medical summary including checkups, diagnosis, follow-ups, prescribed tests, recommendations, and doctor notes
Actual Result	Medical summary displayed correctly with complete details
Screenshot	

Table 6.7: Smart Symptom Analysis (AI-Assisted Triage)

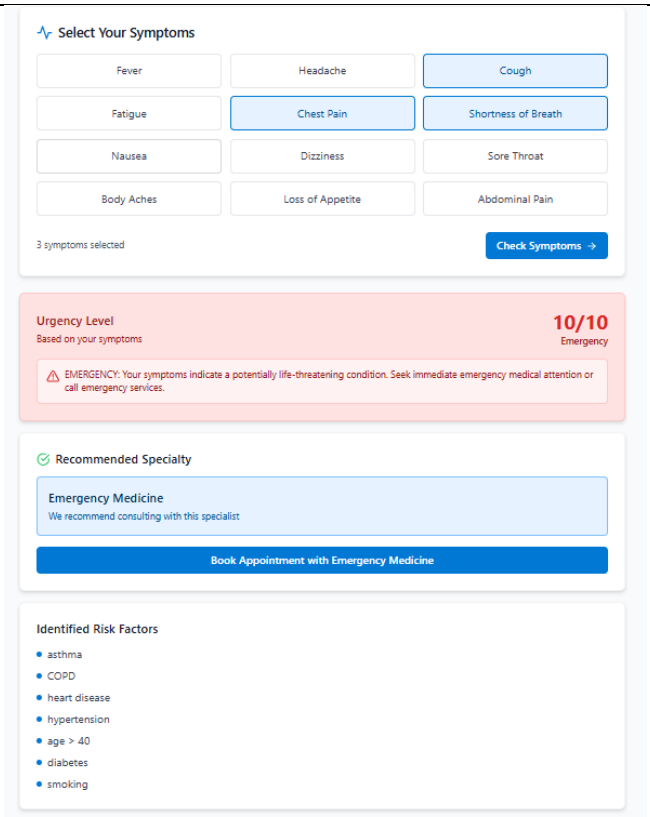
DATE	02-12-2025
TEST ID	TC-10
SYSTEM	NEURAHEALTH
TEST TYPE	Functional Testing
Objective	Verify AI-based symptom analysis and triage decision making
Status	Pass
Version	1.1
Input Enter symptoms: fever, chest pain, shortness of breath. Submit symptoms to Smart Symptom Checker.	Login as patient.
Expected Result	System analyzes symptoms, calculates urgency score (1–10), detects emergency indicators, determines urgency level, suggests medical specialty, and provides next-step guidance.
Actual Result	AI correctly detected emergency indicators, assigned high urgency score, suggested Cardiology, and provided immediate medical guidance.
Screenshot	

Table 6.8: Doctor Managing Availability Slots

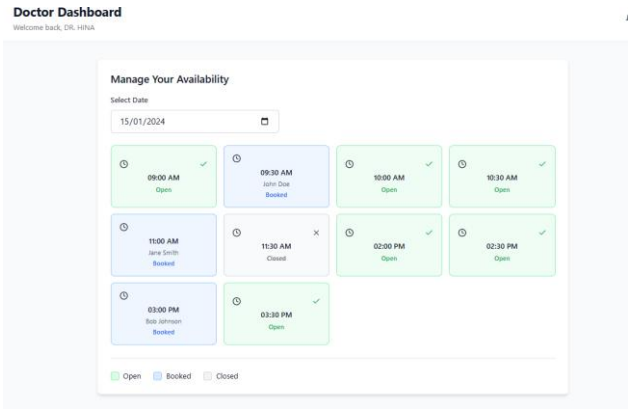
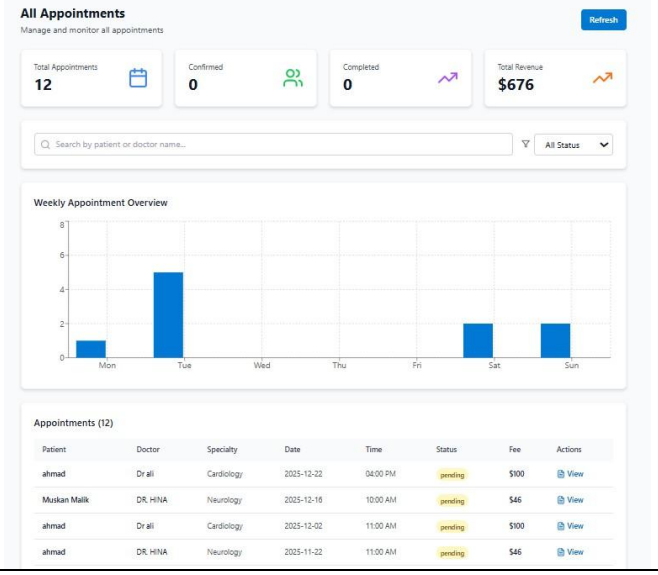
DATE	02-12-2025
TEST ID	TC-11
SYSTEM	NEURAHEALTH
TEST TYPE	Functional Testing
Objective	Verify doctor can manage availability slots
Status	Pass
Version	1.1
Input Clicks "Slot Manager" from sidebar. Adds, updates, or removes desired time slots.	Doctor logs in.
Expected Result	System should save updated slots and reflect changes in appointment booking availability.
Actual Result	Slots updated successfully and availability reflected correctly for patients.
Screenshot	

Table 6.9: Table 5.12: Admin Viewing Patient Appointment Insights

DATE	02-12-2025
TEST ID	TC-12
SYSTEM	NEURAHEALTH
TEST TYPE	Functional Testing
Objective	Verify admin can view appointment insights and details
Status	Pass
Version	1.1
Input Clicks "Appointments" from sidebar. Opens detailed appointment insights view.	Admin logs in.
Expected Result	System should display detailed appointment data including patient info, doctor, date, status, and analytics insights.
Actual Result	Appointment insights and detailed information displayed correctly.
Screenshot	

Chapter 7

Conclusions

The development and implementation of the *NeuraHealth – AI Enhanced Healthcare Platform* has demonstrated the feasibility and effectiveness of integrating artificial intelligence into hospital management systems.[4, 32, 33] The key conclusions of this project are:

- **Effective AI Integration:** The system successfully utilizes AI-driven symptom analysis, predictive analytics, and chatbot-based triage to assist patients and reduce administrative workload.[1, 3]
- **Improved Workflow Efficiency:** Automated appointment scheduling, patient history summarization, and intelligent triage improve hospital operational efficiency.[33]
- **User-Centric Design:** The platform provides tailored dashboards for patients, doctors, and administrators, ensuring usability, accessibility, and relevant insights for each role.[33]
- **Robust Security and Compliance:** Implementation of encryption, JWT-based authentication, and role-based access control ensures secure handling of sensitive patient data.
- **Modular and Scalable Architecture:** The service-oriented design supports future integration with Electronic Health Records (EHR) systems, additional AI models, and potential cloud deployment.
- **Data-Driven Decision Making:** Real-time analytics and visual dashboards allow hospital administrators and doctors to make informed decisions, monitor performance, and identify at-risk patients proactively.

Future Work

The following extensions could enhance the system and broaden its impact:

- **Enhanced AI Capabilities:** Integration of more advanced NLP models and continuous learning mechanisms for improved symptom interpretation and predictive analytics.[4]
- **Mobile Application Development:** Developing native mobile apps to complement the web platform for wider accessibility.
- **Integration with Wearables:** Collecting patient health data from wearable devices to provide real-time monitoring and personalized recommendations.[32]
- **Expanded Security Measures:** Implementation of multi-factor authentication, anomaly detection, and automated security audits.
- **Internationalization:** Support for multiple languages and regional healthcare regulations to broaden the system's adoption.
- **Telemedicine Integration:** Enabling video consultations and remote patient monitoring to further enhance healthcare delivery.[3]

In conclusion, the NeuraHealth platform provides a strong foundation for AI-assisted healthcare management and demonstrates significant potential for further expansion and refinement in both technical and operational aspects

Appendix A

User Manual

Overview

Appendices provide supplementary information that supports the main text without distracting from the central theme. They help readers access additional details like datasets, model logs, and references relevant to the work.

- Appendices should be numbered using letters, e.g., Appendix A, Appendix B, etc.
- Tables and references appearing in appendices should be numbered and cited appropriately, as done in the main chapters.
- Each appendix should carry the title of the work it reports, and this title should also appear in the Table of Contents.

System Access

- **Patients** can access the system via the hospital's website using a chatbot widget.
- **Doctors** log in through a secure dashboard URL using their hospital credentials.
- **Admins** manage appointments, users, and data through the administrative panel.

Chatbot Interaction Flow (Patient Side)

1. The chatbot greets the patient: "How can I help you today?"
2. The patient describes symptoms in natural language.
3. The chatbot:
    - Asks clarifying questions about symptoms and history.

- Suggests precautionary tips.
- Recommends tests (e.g., BP, cholesterol) if necessary.
- Suggests the appropriate specialist doctor.

4. The chatbot shows available time slots and books the appointment.
5. A summary of the symptoms and responses is auto-generated and sent to the doctor's dashboard.

Doctor Dashboard Functionality

- View upcoming appointments.
- Access AI-generated patient summaries.
- Add diagnosis notes and prescriptions.
- Update patient history for future visits.

System Requirements

Minimum Requirements (Client Side):

- Web Browser: Chrome, Firefox, or Edge
- Internet Connection

Minimum Requirements (Server Side):

- OS: Ubuntu 20.04+ or Windows 10+
- Python 3.10+
- GPU (Optional, for LLM training)
- MongoDB or PostgreSQL

Security and Privacy

- All patient data is stored in encrypted form.
- Chatbot conversations are anonymized and processed within the hospital's secure network.
- The system complies with local hospital privacy guidelines.

Troubleshooting

- If the chatbot is unresponsive, refresh the page or check the internet connection.
- For login or data issues, contact the hospital IT administrator.
- GPU-related issues should be checked by verifying CUDA installation and driver compatibility.

References

[1] F. Jiang, Y. Jiang, H. Zhi, Y. Dong, H. Li, S. Ma, and Y. Wang. Artificial intelligence in healthcare: Past, present and future. *Stroke and Vascular Neurology*, 2:230–243, 2017. Cited on pp. 1, 2, and 46.

[2] World Health Organization. Digital health and artificial intelligence in healthcare. https://www.who.int/health-topics/digital-health, 2023. Accessed: 2025-11-24. Cited on pp. 1 and 2.

[3] Y. Shen et al. Ai-assisted chatbots for healthcare: Opportunities and challenges. *Journal of Healthcare Informatics Research*, 6:1–25, 2022. Cited on pp. 1, 2, 3, 46, and 47.

[4] Eric Topol. Deep medicine: How artificial intelligence can make healthcare human again. *Nature Medicine*, 25:44–56, 2019. Cited on pp. 1, 2, 46, and 47.

[5] Marham – find a doctor. https://apps.apple.com/pk/app/marham-find-a-doctor/id1095243102. Accessed: 2025-01-01. Cited on p. 5.

[6] Dawaai – online pharmacy. https://dawaai.pk/. Accessed: 2025-01-01. Cited on p. 5.

[7] Sehat kahani – telemedicine platform. https://sehatkahani.com/. Accessed: 2025-01-01. Cited on p. 5.

[8] Oladoc – doctor appointment booking platform. https://oladoc.com/. Accessed: 2025-01-01. Cited on p. 5.

[9] Ada health – research and studies. https://about.ada.com/studies/. Accessed: 2025-01-01. Cited on p. 6.

[10] Babylon health – case study. https://static.ie.edu/CGC/Case-Study-1-Babylon-Health.-IE-CGC.pdf. Accessed: 2025-01-01. Cited on p. 6.

[11] Infermedica – symptom checker product. https://infermedica.com/product/symptom-checker. Accessed: 2025-01-01. Cited on p. 6.

[12] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2020. Cited on p. 6.

[13] Hannah Semigran et al. Accuracy of online symptom checkers and triage tools: A systematic review. *Journal of Medical Internet Research*, 2021. Cited on p. 6.

[14] Aitriage (2020) – internal or unpublished source. No public link available; cite according to your institution's guidelines. Cited on p. 7.

[15] Daniel James and Rohan Patel. Hospital workflow optimization and challenges in modern healthcare systems. *Journal of Healthcare Management*, 66:112–125, 2021. Cited on p. 8.

[16] S. Alam, L. Wang, and G. Torres. Ai-driven clinical decision support and triage systems: A review. *Healthcare Informatics Research*, 29:45–59, 2023. Cited on pp. 8 and 9.

[17] Hassan Sharif and Min-Jae Lee. Natural language processing applications in healthcare: Opportunities and challenges. *Journal of Biomedical Informatics*, 130:104–120, 2022. Cited on pp. 8 and 9.

[18] Y. Xu, P. Sharma, and K. Lee. Predictive analytics for healthcare operations: Patient flow, resource allocation, and risk prediction. *Health Information Science and Systems*, 8:1–14, 2020. Cited on pp. 8 and 9.

[19] World Health Organization. Ethics and governance of artificial intelligence in health. Technical report, WHO, 2021. Cited on p. 9.

[20] Ian Sommerville. Software engineering: Architectural design and development models. *Pearson*, 2016. Cited on pp. 23 and 32.

[21] Sam Newman. Building microservices: Designing fine-grained systems. *O'Reilly Media*, 2015. Cited on pp. 23, 24, and 32.

[22] Martin Fowler and James Lewis. Microservices: A definition of this new architectural term. *martinfowler.com*, 2014. Cited on p. 23.

[23] Len Bass, Paul Clements, and Rick Kazman. Software architecture in practice. *Addison-Wesley*, 2015. Cited on pp. 23, 24, and 32.

[24] World Health Organization. Ethics and cybersecurity for digital health systems. Technical report, WHO, 2021. Cited on pp. 24 and 34.

[25] Martina Seidl, Christian Huemer, and Gerti Kappel. Uml @ classroom: An introduction to object-oriented modeling. *Springer*, 2015. Cited on p. 24.

[26] Daniel Jurafsky and James Martin. Speech and language processing: Natural language processing overview. *Pearson*, 2020. Cited on p. 25.

[27] Dirk Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014:2–11, 2014. Cited on pp. 25 and 35.

[28] Michael Stonebraker. The seven database sins. *Communications of the ACM*, 61:62–68, 2018. Cited on pp. 25, 26, and 35.

[29] Jakob Nielsen. Usability 101: Introduction to usability. *Nielsen Norman Group*, 2012. Cited on p. 34.

[30] Roger Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 2014. Cited on p. 36.

[31] Ilene Burnstein. *Practical Software Testing: A Process-Oriented Approach*. Springer, 2006. Cited on p. 36.

[32] A. Ramesh, C. Kambhampati, J. Monson, and P. Drew. Artificial intelligence in medicine. *Annual Review of Medicine*, 70:33–49, 2019. Cited on pp. 46 and 47.

[33] Eric J. Topol. High-performance medicine: The convergence of human and artificial intelligence. *Nature Medicine*, 24:45–56, 2018. Cited on p. 46.