Smart Assistance Appointment System for NIMHANS

MAJOR PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY
(Computer Science and Engineering)

Submitted By:

Bharatdeep Singh (2104080)
Chandanbir Singh (2104083)
Divneet Kaur (2104093)

Submitted To.:

Dr. Parminder Singh
Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GURU NANAK DEV ENGINEERING COLLEGE
LUDHIANA, 141006
Novemeber, 2024

Abstract

The Doctor Appointment System project presents an intelligent, voice-enabled solution designed to transform and streamline the process of booking medical appointments and managing patient records. Developed to address the need for efficiency in medical settings, this system integrates advanced artificial intelligence, voice processing, and natural language processing (NLP) capabilities. By leveraging models such as ChatGroq, Whisper, and Deepgram, the system transcribes patient audio inputs, intelligently interprets responses, and dynamically updates medical records in a structured JSON format, all while providing a responsive, web-based user interface. The frontend, developed with Vite and React, includes a user-friendly interface for voice-driven interaction, appointment scheduling, and real-time status updates. Meanwhile, the backend is structured around Express.js for handling HTTP requests and MongoDB as a scalable, NoSQL database for storing patient information, appointment history, and interaction logs. A dedicated Django service supports machine learning tasks like voice command interpretation, allowing the system to intuitively process voice commands for booking, rescheduling, or cancelling appointments. Additionally, the system integrates seamlessly with existing hospital management systems (HMS) to enable real-time synchronization of doctor availability, department schedules, and appointment data. Notifications and reminders are sent to both patients and administrative staff, enhancing communication and reducing missed appointments. Overall, this Appointment System stands as a comprehensive solution to enhance patient-provider interactions, reduce administrative burden, and improve operational efficiency in healthcare facilities. Its use of state-of-the-art AI for voice recognition and NLP, combined with an intuitive user interface and secure backend, underscores its potential to redefine the patient experience in modern medical environments. By automating routine tasks and providing real-time data access, this system contributes to a more efficient, patient-centered approach to healthcare.

ACKNOWLEDGEMENT

We are highly grateful to the Dr. Sehijpal Singh Khangura, Principal, Guru Nanak Dev Engineering College (GNDEC), Ludhiana, for providing this opportunity to carry out the minorproject work.

The constant guidance and encouragement received from Dr. Kiran Jyoti, H.O.D. CSEDepartment, GNDEC Ludhiana has been of great help in carrying out the project work and is acknowledged with reverential thanks.

We would like to express a deep sense of gratitude and thanks profusely to Dr. Parminder Singh without his wise counsel and able guidance, it would have been impossible to complete the project in this manner.

We express gratitude to other faculty members of computer science and engineering department of GNDEC for their intellectual support throughout the course of this work.

Finally, we are indebted to all whosoever have contributed in this report work.

Bharatdeep Singh (2104080)

Chandanbir Singh (2104083)

Divneet Kaur (2104093)

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

Chapter 1 Introduction

1.1 Introduction to Project

The healthcare industry is constantly evolving, with a focus on enhancing patient experience and improving operational efficiency. One of the persistent challenges is the scheduling of outpatient department (OPD) appointments, which can often be cumbersome and time-consuming for both patients and administrative staff. To address this, we propose the development of a **Smart Assistance Appointment System** which is specifically designed for NIMHANS (National Institute of Mental Health & Neurosciences, Bangalore, Karnataka). This project aims to design and implement a voice assistant system that allows patients to easily interact with the hospital's appointment system through intuitive voice commands. Additionally, a smart, question-driven web portal will be developed for the categorization and booking of OPD appointments.

This system will improve the appointment scheduling process, reduce wait times, and enhance patient satisfaction by using Natural Language Processing (NLP) and speech recognition techniques. Traditional appointment scheduling methods can be time-consuming, prone to errors, and often inconvenient for patients, particularly for those with limited digital literacy or accessibility needs, hence by using this system these problems can be eliminated. The voice assistant will manage various patient requests, right from including booking, and cancellations, to ensuring real-time updates and maintaining data accuracy by integrating seamlessly with NIMHANS' existing management systems. The project will also cater to a diverse patient demographic, including those unfamiliar with traditional digital interfaces, by providing an intuitive, user-friendly voice-based interface. Through this initiative, we aim to demonstrate the transformative potential of AI-driven solutions in healthcare, thus showcasing how such technologies can optimize appointment management and improve overall service delivery. The goal is to create a scalable model that can enhance patient experience and operational efficiency at NIMHANS and potentially be adopted by other healthcare institutions. Implementing an automated voice assistant and smart web portal specifically designed for NIMHANS will address these challenges by streamlining the appointment process. The voice assistant will enable patients to book, and cancel appointments using simple voice commands, making the system more much more accessible to a broader patient demographic. The smart web portal will further enhance efficiency by categorizing the patients based on their specific needs, ensuring that appointments are scheduled appropriately and hence reducing wait times.

In addition to streamlining the appointment process, the proposed Smart Assistance Appointment System can significantly reduce the burden on hospital staff. By automating routine tasks like booking, and cancellations, administrative staff can focus on more critical duties, leading to increased productivity and better allocation of resources. This system hence will also provide real-time updates, ensuring that the appointment database is always current, reducing the risk of double bookings or missed appointments. Such automation will improve operational workflows, leading to better overall efficiency at NIMHANS.

The system's design tends to prioritizes accessibility and inclusivity, making it highly beneficial for patients with limited digital literacy or those who struggle with navigating traditional online systems. The voice assistant will enable patients to interact with the hospital's systems using simple, intuitive voice commands, reducing the frustration of navigating complex menus. By offering an easy-to-use voice interface and a question-driven web portal, the system will cater to a wide range of users, including elderly patients or those with disabilities, thereby increasing access to care for all demographics.

Moreover, the system’s potential scalability is a key advantage. Once implemented successfully at NIMHANS, this model can easily be adapted and deployed at other healthcare institutions facing similar challenges with OPD scheduling. The scalability of this solution lies in its ability to integrate with existing hospital management systems while offering customizable features to suit the specific needs of different healthcare providers. This makes it a valuable long-term investment not only for NIMHANS but also for the broader healthcare ecosystem, providing a template for future advancements in digital health services.

This system is not only a response to the operational needs of NIMHANS but also aligns with broader trends in healthcare towards digital transformation and patient-centered care. Thus, the project aims to improve the overall patient experience, reduce administrative workloads.

1.2 Project Category (Application Based and Industry Based)

This project falls under both **Application-Based** and **Industry-Based** categories, offering significant practical and industrial implications.

The main focus of the project is to develop a **Smart Assistance Appointment System for NIMHANS (National Institute of Mental Health and Neurosciences, Bangalore)** which directly addresses OPD appointment scheduling challenges. The system is designed to provide a voice assistant and **question-driven web portal** using advanced technologies such as NLP and speech recognition. The primary goal is to streamline booking and cancellation processes, improve the overall patient experience and **reduce administrative hurdles**. By automating **appointment scheduling** and offering real-time updates, the system improves the efficiency of hospital operations, making it an effective application-based solution that addresses the specific needs of both patients and medical staff.

In addition to practical applications, the project aligns with larger trends in the healthcare industry towards automation, AI-driven solutions and digital transformation. The project has the potential to revolutionize the functioning of **outpatient departments** by integrating with existing hospital management systems, making it adaptable to a range of healthcare facilities outside of NIMHANS. As healthcare providers increasingly look to the use of AI for better operational efficiency and patient care, this system is positioned as a scalable solution that can be deployed across the industry. The use of cutting-edge technologies such as **voice assistance** and AI data management reflects the industry's growing demand for smart, automated systems that increase productivity, **reduce manual errors** and improve patient engagement. This project thus serves as a key innovation for both hospital applications and the healthcare industry as a whole, demonstrating the impact AI can have in advancing healthcare solutions.

1.3 Problem Formulation

The problem formulation for the **Smart Assistance Appointment System for NIMHANS** project involved a comprehensive analysis of the challenges faced by both patients and administrative staff in scheduling outpatient department (OPD) appointments. NIMHANS, is a leading mental health and neuroscience institution, which experiences high patient volumes, making the traditional appointment process inefficient and prone to delays. This system aims to address the difficulties of manual scheduling, which often results in long wait times, human errors, and inconveniences for patients, particularly those with limited digital literacy or accessibility challenges.

The existing system at NIMHANS requires patients to book appointments via phone calls (manually) or by physical visits, both of which are time-consuming and lead to increased administrative workloads. Due to this inflexible and fragmented nature of the system, patients often face challenges booking rescheduling or cancelling appointments. In addition to this, these challenges are made worse by a varied patient population at NIMHANS many of who will be highly unlikely to deal with the complexities of digital systems or are simply unable to access them. They required a strategic automated solution that would provide the ability to handle all patient groups with minimal administrative burden yet seamless appointment management. The scope of the project involved addressing several key challenges:

a) Voice Assistant for Appointment Management: A voice-activated interface that enables patients to schedule, reschedule, or cancel appointments forms the basis of the system. This makes the process easy to understand, particularly for people who have physical constraints or are not accustomed with digital platforms. The solution guarantees that all interactions are efficient and accessible by creating a voice assistant, eliminating the need for difficult navigation or a great deal of manual intervention [1].

b) Smart Web Portal: In addition to the voice assistant, the project entails creating an intelligent web platform that effectively classifies and schedules OPD appointments through question-driven interaction. In order to help guarantee that appointments are arranged suitably based on the availability of specialists and medical requirements, patients will be led through a series of questions tailored to their individual needs. This system seeks to maximize resource allocation and decrease wait times.

c) Real-Time Integration with Hospital Systems: One of the most significant issues is to allow the voice assistant and the web-portal to smoothly interact with the existing systems in NIMHANS. Real-time updates of schedules will be made possible as well as avoidance of cases where duplicate appointments were made or patient records compromised. This integration will also be of benefit to administrative staff since RTI will handle certain flow processes automatically and reduce errors on part of the data input personnel.

d) User Accessibility and Inclusivity: One of the project objectives is to make the system universally accessible to the elderly, disabled or anyhow less technologically inclined patient. By using voice-based commands for patients, and making the web portal more user-friendly, those patients who find the existing information technology often off-putting will be catered for.

e) Automation and Efficiency Enhancement: It is aimed at decreasing the manual scheduling of tasks since this system should at least relieve the workload of more mundane activities. The freedom to manage their own requests will be provided to patient by voice assistant, Also, smart portal will group patients according to healthcare requirements to schedule appointments effectively and increase business productivity.

1.4 Identification/Recognition of Need

The rationale for the development of the **Smart Assistance Appointment System for NIMHANS** was based on the understanding of the difficulties in scheduling outpatient department (OPD) appointments and the concrete increase in the need for technological solutions in healthcare. The following key factors contributed to the recognition of this need:

a) Complexity of Appointment Scheduling: In the conventional approaches of scheduling new appointments, patients are required to contact the NIMHANS or visit the clinics in person resulting to time wastage whereby many patients are booked in advance hence long waiting time is experienced, patient/caregiver and clinical staff time is wisely used hence the need to develop an appointment scheduling system. High patient traffic and specialized setups make it learner for NIMHANS to get an effective mechanism to support appointment scheduling for their different departments including accuracy in booking, rescheduling and cancellation of scheduled appointments. The dynamic nature of customers, from simple checkups, which all need to be scheduled, to those in need of acute mental health care, presents the need for a smarter, self-sufficient system of catering to the populace's health needs [2].

b) Limitations of Current Methods: The current time tab in most clinical practices involves manual handling of appointment that is time consuming and full of errors. The long wait time, lack of real time updates, and challenges associated with the clinical pathways that are not amenable to traditional process mapping exhaust and frustrate the patients, primarily the less digitally competent ones. Like many other businesses, administrative staff often struggle with appointment scheduling leading to over-booking or cancellation of sessions. The above limitations of the traditional scheduling system show that there is need to develop a new system that is smart and capable of automating the healthcare services.

c) Advances in AI and Voice Technologies: As the technology in the field of AI, NLP and speech recognition progresses new possibilities appear to enhance appointment scheduling. With these technologies, it is possible to create a voice assistant and a web portal based on questions where people can enter to **avoid time-wasting** for booking, or canceling their appointments. These advancements offer the potential to enhance patient access to services and reduce administrative workloads, ensuring a more streamlined and error-free system. The growing adoption of AI-driven solutions in healthcare makes it an ideal time to implement a smart appointment system that integrates seamlessly with existing hospital management infrastructure.

d) Demand for Patient-Centered, Accessible Solutions: The need for healthcare systems for varied patient base is felt with specific attention towards patients who are not tech-savvy or patients with disability. Present appointment at NIMHANS disrupts are not effective to reach each patient, those who have problems in handling digital interfaces, or physical restriction in mobility. The use of voice assistant and an **intelligible web portal** means that any patient will be able to schedule an appointment with an easier manner, thus making it easier for anyone to get the help they need in terms of mental health issues. Such patient-centered approach corresponds with further tendencies in healthcare towards diversification and openness, which should be met with an intelligent, user-friendly solution.

e) Lack of Automated Real-Time Solutions: It is also missing on the everyday status of the appointment when it is available thus can lead to a problem of over booking. Further, automation in the current setting is completely missing with heavily manual intervention necessary from the side of administrative personnel. An automated system for generating appointment schedules that updates real-time information based on the patient’s schedule with no disruption to other hospital systems and addresses all facets of the appointment. An automated system that offers real-time information with no interference with the hospital's systems and handles all the phases of the appointment process will eliminate many of the operational problems. Since the Smart Assistance Appointment System can provide real-time visibility into patients' appointment schedules and help to avoid mistakes and enhance patients' satisfaction, its absence is not justified.

By addressing the limitations of the current scheduling system and leveraging cutting-edge AI technologies, the **Smart Assistance Appointment System** aims to enhance the user experience, improve operational efficiency, and make healthcare services more accessible for all patients at NIMHANS. This recognition of need stems from a clear demand for more **automated, inclusive**, and **efficient solutions**, making it a vital tool for modernizing healthcare services at the institution.

1.5 Existing Systems and their Limitations

Several AI-driven systems have been developed to enhance appointment booking and management in healthcare, addressing various inefficiencies related to patient scheduling. While these systems have made notable advancements, they still face several limitations in delivering an optimal patient experience and operational efficiency. Below are some examples of existing systems and their limitations:

a) SMART DOCTORS ASSISTANT (2023): This AI-powered system was developed to streamline appointment scheduling in hospitals, aiming to improve both patient accessibility and hospital workflows. It incorporates AI to automate booking processes, making scheduling more efficient. However, **SMART DOCTORS ASSISTANT** is primarily focused on operational workflows and lacks a personalized, voice-driven interface, limiting its accessibility for patients with low digital literacy or those with disabilities. Additionally, its integration capabilities with existing hospital management systems are relatively underdeveloped, which may hinder seamless data sharing and real-time updates [3].

b) AI-Based Medical Voice Assistant During COVID-19: This system was introduced to assist in managing appointments and disseminating medical information during the COVID-19 crisis. It uses voice-activated technologies to handle patient queries and appointment bookings. While this system proved effective during the pandemic by reducing physical contact and improving service accessibility, it is **limited in its adaptability** beyond crisis situations and lacks a more sophisticated **NLP** feature to handle complex patient requests or intricate appointment management tasks. Additionally, its integration into long-term hospital management solutions is minimal, focusing primarily on short-term needs [1].

c) Medicare: A Doctor Appointment Application System: **Medicare** provides a comprehensive mobile platform that allows patients to book, manage, and cancel appointments easily through a user-friendly interface. While it successfully integrates mobile technology to enhance patient engagement, it **lacks voice recognition features**, making it less accessible for elderly patients or those unfamiliar with digital platforms. Moreover, its primary focus on mobile-based interactions limits its adaptability to hospitals where voice-driven, hands-free appointment management would be more effective [5].

d) Appointment Maker Using Computerized Voice: This system utilizes **voice recognition technology** to allow patients to schedule appointments using simple voice commands. The system enhances user engagement by making the appointment process intuitive and efficient. However, it is **limited in scope**, with a focus on basic appointment scheduling tasks and lacks advanced features like real-time integration with existing hospital systems or proactive patient communication, which are crucial for maintaining accuracy and efficiency in busy hospital environments [7].

The limitations of existing appointment scheduling systems have significant implications for their scalability and effectiveness in addressing complex healthcare needs. Few limitations are discussed below:

a) Lack of Comprehensive Natural Language Understanding: While many systems incorporate voice recognition technology, their **NLP** capabilities are often limited, resulting in difficulties when interpreting more complex or context-specific patient requests. This makes it challenging to manage diverse patient needs or handle ambiguous commands effectively, reducing overall user satisfaction.

b) Limited Integration with Existing Hospital Systems: Existing solutions often face **integration challenges** with hospital management systems. Many current appointment scheduling platforms fail to provide real-time updates or synchronize appointment data across different departments, leading to inefficiencies and potential scheduling conflicts. A fully integrated system is necessary to ensure seamless data flow and error-free management.

c) Accessibility Gaps for Diverse Patient Demographics: Many systems rely heavily on mobile platforms or basic web interfaces, which can be inaccessible to certain patient groups, such as the elderly or those with disabilities. Additionally, systems without voice-driven interactions or multilingual support may fail to adequately serve patients with low digital literacy or non-native language speakers.

d) Limited Scalability for Larger Healthcare Institutions: While existing systems work well for smaller clinics or individual practices, they **lack the scalability** required for larger healthcare institutions, which manage higher patient volumes and more complex scheduling needs. These systems often struggle to handle concurrent bookings, rescheduling, and cancellations at scale without compromising performance.

e) Minimal Proactive Patient Communication: Many current systems focus on basic appointment scheduling without incorporating features that **proactively communicate** with patients, such as sending reminders, follow-up notifications, or handling rescheduling in response to patient preferences. This results in missed opportunities to enhance patient engagement and reduce no-show rates.

Our proposed **Smart Assistance Appointment System** for NIMHANS builds on the strengths of these existing systems while addressing their limitations. By integrating **advanced NLP**, real-time hospital system synchronization, and a voice-driven interface, our system is designed to provide a more **inclusive, scalable**, and **efficient** solution that meets the complex scheduling needs of a diverse patient population.

1.6 Objectives

The objectives outlined for the successful development of the **Smart Assistance Appointment System for NIMHANS** are critical for addressing the challenges in managing outpatient department (OPD) appointments. Each objective is designed to improve the system's efficiency, enhance patient accessibility, and reduce the administrative burden while ensuring seamless integration with the hospital's existing infrastructure. The objectives of our project are described below:

a) To select and implement voice recognition model for recognizing phone line in speech utterances.

b) To design and implement voice assistant system for interacting with patients seeking doctor appointment.

c) To develop a smart question driven web portal for categorization and appointment booking of OPD patients at NIMHANS.

1.7 Proposed System

Several components have been proposed to form the bases of the desired appointment scheduling system with increased accessibility, efficiency and higher patient satisfaction. They are discussed below:

a) End-to-End Appointment Management: The system is fully capable of opening up scheduling for the outpatient department (OPD and enables the patient to make, reschedule or cancel an appointment through a smooth and efficient touch-tone menu. Using voice commands or a smart web portal, patients can manage to communicate with the system, with the request analyzed, latest updates ensured, and data integrity checked continuously. This end-to-end approach makes it easy for the patients to manage their appointments from the time they book the appointment to when they are attending the appointment.

b) Natural Language Processing for Patient Interaction: Using recent advances in NLP the system will allow for precise analysis and interpretation of patients' requests. Comprehending the tone and context of the spoken language the voice assistant is easily capable to sort and react to the appointment types of calls that could be challenging for the patients to express. It improves the system's comprehensiveness of how it will interface with a broad patient population and those with reduced ease of use of technology devices.

c) Interactive User Interface: The system has a graphic interface to facilitate its usability and enable patients to manage appointment conveniently. The patient interface is either an auditory response from the voice activated system or a response through the smart web interface where the patient is able to search through available slots for an appointment, get feedback on their submitted request and other important information regarding their healthcare needs. The interface is also regulative in a way that persons can modify it according to their preference and disability needs.

d) User Registration and Personalization: The system allows patients to create personal accounts, providing a customized experience that includes tracking appointment history, saving preferences, and accessing personalized content. User registration fosters a tailored approach to appointment management, enabling patients to engage more effectively with the system and stay informed about their healthcare journeys.

e) Responsive Design for All Devices: Designed with responsiveness in mind, the system's interface adapts seamlessly to various screen sizes and device types. Whether accessed from a desktop computer, tablet, or smartphone, the system remains user-friendly and visually appealing, ensuring that patients can manage their appointments with ease across different platforms.

f) Data Security and Privacy Measures: Security is a top priority for the proposed system. User data, including personal information and appointment details, is protected through encryption and secure storage practices. By implementing robust security measures, the system ensures that patient information remains confidential and safeguarded against unauthorized access[8].

By incorporating these features, the Smart Assistance Appointment System aims to provide a secure, accessible, and personalized appointment scheduling experience. Whether through voice interaction, web portal access, or ensuring data protection, these enhancements improve convenience and satisfaction for patients engaging with NIMHANS' healthcare services.

1.8 Unique features of the Proposed System

The Smart Assistance Appointment System is designed with a range of innovative features that prioritize user experience, accuracy, and security. These unique elements collectively enhance the appointment scheduling process for patients, ensuring a seamless and efficient interaction with healthcare services.

a) Interactive User Interface: The Smart Assistance Appointment System features an interactive user interface designed to provide a user-friendly experience for patients scheduling and managing their appointments. Patients can easily navigate the system, using voice commands or the smart web portal to explore available appointment slots, receive real-time feedback on their requests, and interactively manage their appointments. The interface allows users to adjust settings, provide feedback, and customize the scheduling process according to their preferences, enhancing engagement and overall satisfaction.

b) Accurate Appointment Management: By integrating advanced NLP algorithms and sophisticated scheduling techniques, the system delivers precise appointment management solutions. It considers contextual constraints, patient needs, and real-time availability to generate accurate appointment confirmations. This ensures that patients receive timely and relevant information, leading to a more reliable scheduling experience that aligns with their healthcare requirements.

c) User Registration and Personalization: The system allows patients to create personal accounts, providing a tailored experience that includes tracking appointment history, managing preferences, and accessing personalized content. User registration empowers individuals to customize their interactions with the system, maintaining a record of their activities and ensuring a more efficient appointment management process over time.

d) Enhanced Security Features: The system incorporates robust security measures to protect user data and privacy. Proper user profiles are maintained to safeguard sensitive information and keep records securely. These security features ensure that patient information remains confidential and protected from unauthorized access.

Chapter 2. Requirement Analysis and System Specification

2.1 Feasibility study

The feasibility study for the proposed Smart Assistance Appointment System is described below:

a) Technical Feasibility

i. Technology Stack: The suitability of implementing the system utilizing the appropriate technologies which include NLP libraries, speech recognition frameworks and web development technologies for the voice assistant and smart web portal respectively was as well evaluated critically.

ii. Scalability: The efficacy of the system to deal with the applicative load of appointments requests and users interactions regarding to the aspects of database, servers and computational resources to guarantee a strong database management was analyzed.

iii. Integration: Assessed the possibility to include different aspects of the system: the NLP algorithms, the mechanisms of the user authentication, the option to update the appointments in real-time, and the integration with the hospital management systems.

iv. Data Management: Considered the admissibility of abstracting and indexing various type of data, both internal and external, from appointment records, user accounts to the feedback data while addressing data security and data integrity issues and regulatory health care privacy laws.

b) Economic Feasibility:

i. Cost Analysis: Identified various cost structures that were applied in a bid to determine the actual amount of costs that may be incurred in the development, deployment and maintenance of the system, such as costs of developing software, cost of the infrastructure, cost of licensing other tools as well as recurring costs.

ii. Revenue Model: Investigated possibilities of generating revenue, including charges to the hospitals for access to the system, contracts with healthcare providers for the delivery of their services using the system, and probably, the grants toward the development of the digital healthcare solutions as the possibilities for the system's revenue.

c) Operational Feasibility:

i. User Acceptance: Assessed the readiness of the target users, with regard to the use of the proposed system, through market research, patient and administrative staff surveys and feedback.

ii. Usability Testing: In particular, completed the number of usability tests and user experience surveys to determine possible usability problems or further enhancement opportunities regarding the system's comprehensibility, convenience to use, and availability to the target audience.

iii. Regulatory Compliance: Minimized legal risks by adhering to stringent laws and standards of data privacy, security in the healthcare sector and patients' rights to protect an ethical use of the system.

2.2 Software Requirement Specification Document

The Software Requirement Specification (SRS) document below outlines the essential requirements for the development of a Smart Assistance Appointment System:

a) Purpose: The need for the project is therefore to design an Appointment System for the OPD using NLP and voice recognition solution to make the process easier and more convenient to the patients and improve hospital productivity.

b) Scope: This is a very vast project because the appointment scheduling so not only covers NIMHANS but other healthcare centers as well will be helpful for patients of all ages.

c) Intended Audience: It is mainly designed for patients, caregivers and support staff working at NIMHANS but patients from different background with different levels of Computer Literacy can also use this system.

d) Overview: The system is an innovative technology that brings effective appointment scheduling through voice services and a web-based interface. The system interprets user demands, estimates probable appointment time, and offers the progress in real-time basis through such algorithms and NLP. This system has incorporated an interface through which the users are well taken through the time table for the appointments in a personalized manner and required feedback.

e) User Interface: The interface is made for it to be simple and easy to understand with little to no interference by complex graphic designs. It has easy to use controls, so that the user does not find it hard in the handling of the schedule aspect. It is fully mobile optimized and adaptable to tablets and phablets in addition to PCs and laptops; thus, users enjoy near-identical experiences across both desktop and mobile.

f) Constraints, Assumptions, Dependencies: The system assumes access to high-quality speech recognition and NLP tools, and it may require integration with existing hospital management systems. Reliable internet connectivity and adequate server resources are also essential for optimal performance.

g) Data Requirements:

i. Appointment Records: The system requires access to a comprehensive database of appointment slots and patient information from NIMHANS' existing management systems.

ii. User Profiles: The system saves user registration data that they contribute in the form of username, encrypted password, and other contact details to make it more customizable and safe to use.

iii. **Feedback Data:** The gathering and storage of information about the user experience with the appointment scheduling system in order to consider the quality of service delivery and refine the system continually.

iv. Real-Time Data: It has to cover some real-time information concerning available appointments, cancellations, modifications, etc.

h) Functional Requirements:

i. **User Registration:** The application allows users to create accounts with necessary details, ensuring secure access to personalized features.

ii. Appointment Scheduling: Core functionality includes enabling users to book, and cancel appointments through voice commands or the web portal.

iii. Interactive User Interface: The interface is intuitive and visually appealing, facilitating seamless navigation and interaction for users.

iv. **Real-Time Updates:** The application provides real-time feedback on appointment availability and status, enhancing user engagement.

v. User Feedback: Mechanisms for users to provide feedback on their experiences with the scheduling process to improve the application continuously.

vi. **Appointment History:** Users can access a record of their past appointments, enabling them to track their healthcare journey effectively.

i) Performance Requirements:

i. Response Time: The application responds promptly to user inputs to ensure a smooth and uninterrupted experience.

ii. Scalability: The system should efficiently handle concurrent user requests and dynamically scale resources during peak usage.

iii. **Resource Utilization:** Optimal management of computational resources, memory, and bandwidth is essential to deliver performance without unnecessary overhead.

j) Dependability Requirements:

i. **Reliability:** The application must operate reliably, providing accurate appointment scheduling without frequent crashes or failures.

ii. **Data Integrity:** Ensuring the integrity and consistency of user data and appointment records is critical for fostering user trust.

k) Security Requirements:

i. Authentication: Implementing robust user authentication mechanisms to ensure secure access to user accounts.

ii. **Encryption:** Utilizing industry-standard encryption algorithms to protect user credentials from unauthorized access.

iii. Data Privacy: Compliance with data privacy regulations is essential, employing measures to safeguard sensitive information.

l) Maintainability Requirement:

Modularity: The system should have a modular architecture, allowing for easy maintenance, updates, and the development of new features.

m) Look and Feel Requirements:

i. User Interface Design: The application should feature a modern, visually appealing interface with clear navigation and responsive layout.

ii. **Accessibility:** The application must include accessibility features to ensure inclusivity for users with diverse needs and abilities.

2.3 SDLC model

The Agile methodology was selected for the development of the Appointment System because of its flexibility and its ability to make improvements during and between the several stages, which are important for developing successful software that effectively meets the needs of users and is responsive to change.

It is also important to note that communication partnerships are an important Agile principle since Agile methodologies put a strong focus on complete collaboration. Due to the integrated nature of the development team, care givers, and other stakeholders, Agile ensures that everyone is in harmony with the project objectives. The mentioned environment contributes to the ability of the team to make corrections as they proceed with the development task, to ensure that users are met according to their needs and demands. A final major strength for Agile is that it is an iterative process. Instead of trying to bring the appointment system in one large product release, Agile distributes the whole development process into small and flexible units called iterations or sprints. It spans over a few weeks and culminates in a potentially shippable product increment giving organising teams to deliver value to the users at a faster pace without a long time to delivery. Indeed, agile emphasizes many customer-oriented principles, for example the idea that valuable working software should be created as often as possible. In terms of duration prioritization of tasks according to user needs Agile guarantees that system will incorporate functionalities most essential for patients and doctors first. This works in cycles also gives the team the opportunity to implement changes made by the users in later cycles, this makes the appointment system more sensitive to the environment and users of the system.

In summary, the Agile methodology empirical approach suits the development team to ensure that it works as a team, adapt to the changes in requirements and deliver a quality appointment system, suitable for users. Agile methodology facilitates addressing the challenges while delivering timely value to users as follows:

a) User Interface: Accessibility is an important part of the appointment system since clients have to interact with it directly through the user interface. Based on this process, the Agile methodology allows developers, designers, and healthcare staff to work cooperatively in the process of improving the user interface repeatedly through the opinions of users or a usability test. Hence, Agile prioritizes the user requirements as well as the user desire in order to guarantee a good chance for the improvement of the interface of the developed system.

b) Natural Language Processing: Therefore, natural language processing functionalities of the system play crucial roles in correctly understanding the demand made by the user and providing relevant outputs. Instead of approaching a development project as one long, singular process, Agile splits the whole process into a number of smaller cycles which each focus on a specific minor segment of the program, allowing the team to take what they have learned from the data and user activity, apply it to the specific algorithms for NLP and improve on it during the next iteration.

c) Voice Recognition: This informs the use of voice recognition and interaction as a critical enabler for patients to participate fully in the appointment process. This is because agile methodology call for the team to use available speech recognition framework which at the same time design solutions that cater for the needs of healthcare environment. Here it is suggested that an iterative approach to the further development of voice recognition would allow the team to fine-tune the system so that it can accept a more complex range of inputs from the user [8].

This informs the use of voice recognition and interaction as a critical enabler for patients to participate fully in the appointment process. This is because agile methodology call for the team to use available speech recognition framework which at the same time design solutions that cater for the needs of healthcare environment. Here it is suggested that an iterative approach to the further development of voice recognition would allow the team to fine-tune the system so that it can accept a more complex range of inputs from the user [8].

d) Iterative Improvement: Agile is very strong on the principle of customer collaboration and their provision of feedback. This approach allows the team to improve the Smart Assistance Appointment System overtime, as it aligns perfectly with the goal of being able to apprehend the needs of the users and deliver on valuable techniques for scheduling. This approach allows the team to improve the Smart Assistance Appointment System overtime, as it aligns perfectly with the goal of being able to apprehend the needs of the users and deliver on valuable techniques for

scheduling.

Fig 2.1 describes the model being used by our system.

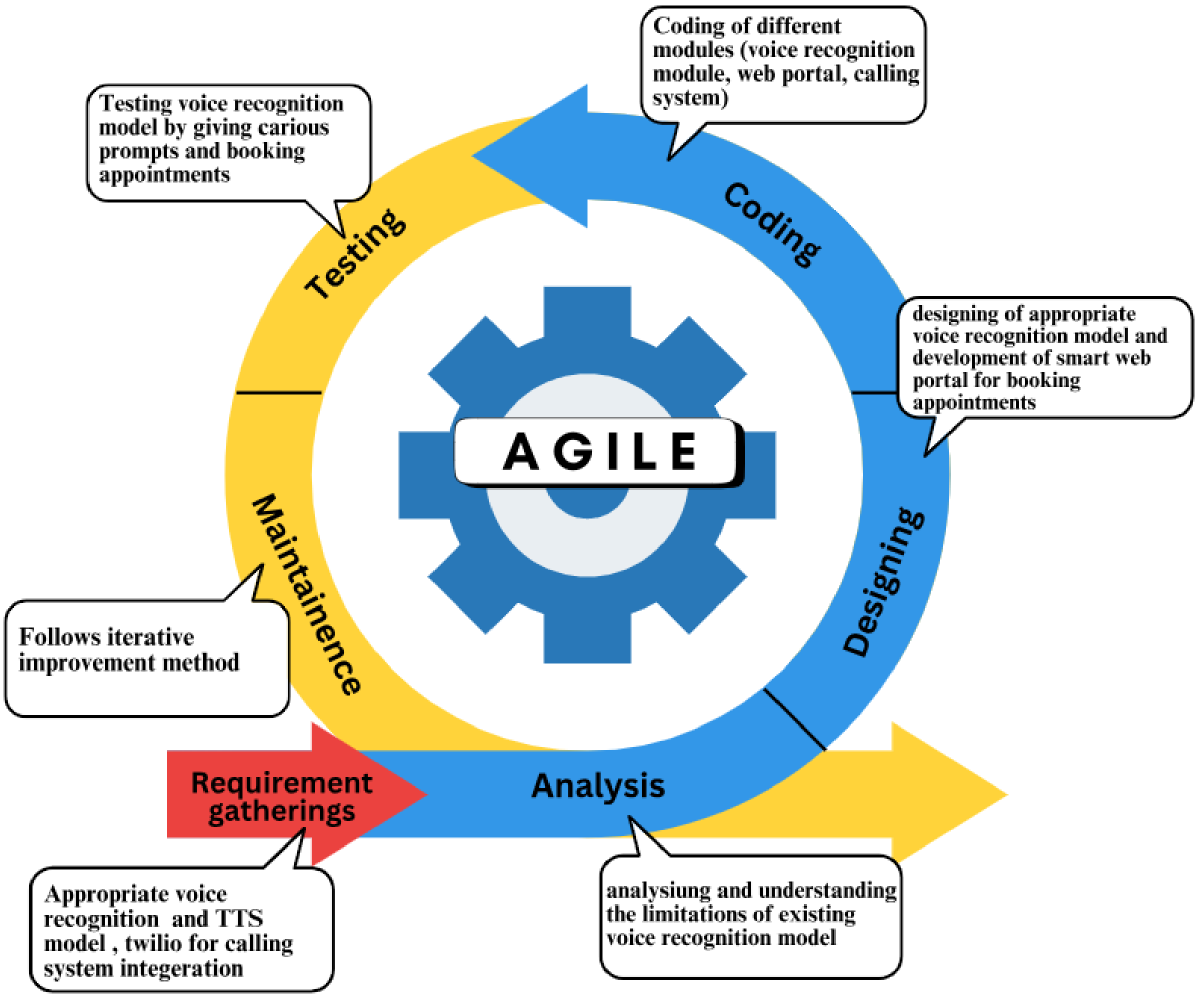

Coding of different modules (voice recognition module, web portal, calling system)
Testing voice recognition model by giving carious prompts and booking appointments
Coding
Testing
designing of appropriate voice recognition model and development of smart web portal for booking appointments
AGILE
Maintainence
Designing
Follows iterative improvement method
Requirement gatherings
Analysis
Appropriate voice recognition and TTS model , twilio for calling system integeration
analysiung and understanding the limitations of existing voice recognition model

Fig 2.1: Agile Model used for the appointment system

Chapter 3. System Design

3.1 Design Approach (Object oriented)

The system's Object-Oriented Design (OOD) framework enables a robust, modular, and flexible structure, aligning each component to manage specific functionalities related to audio processing, transcription, natural language understanding, JSON management, and user interaction. By using object-oriented principles such as encapsulation, inheritance, and polymorphism, this design ensures that each module can operate independently while still interacting cohesively, supporting ease of maintenance, scalability, and reusability across the system's complex functionalities.

Here is an expanded breakdown of the design approach across different modules:

a) Question Answering (QA) Module: QA module is the center of the conversation and it is in charge of handling the audio input stream to understand instructions or questions, controlling conversation progress and checking answers. It implements a connector for a variety of Machine Learning honed services that include ChatGroq, Deepgram, and Whisper to be involved in natural language understanding and control of the question sequencing relative to the conversational context. The last class defined in the file is the ChatGroq Model which offers methods for analyzing and natural language inputs and for checking the inputs against the format of the expected questions. It encapsulates the basic NLP functionality; one can always tweak or update the language model embedded within without changing its global flow. A QA Flow Manager class is tasked with managing the flow of the questions in the system, the flow being that the system only progresses to the next question in sequence once a proper response has been made. This class can also be inherited to support different conversation types like defaults, follow-ups, complex etc and so the flexibility of the module is well handled. **For Example:** This begins with a form of recognition where a user says 'Hello' to the system. ChatGroq takes each input, checks that the input matches the expected format described above and if valid moves on to the next question. The QA goes on unless a termination keyword (such as "goodbye") is sensed, and the conversation stops.

This modular design enables the system to maintain dynamic, question-driven conversations that can be customized for various healthcare applications without altering the underlying QA logic.

b) Data Collection Module: For this, all the audio data and the transcriptions will be sorted and stored in this module, which forms the base layer, supplying the information to the structured JSON medical form. The Audio Data Manager class deals with audio recordings, transcription processes as well as managing audio files. The Transcription Handler class encapsulates transcription methods, like the one that calls the Whisper, the model that inserts audio and stores this then it stores the text for further work. Since data collection has its own set of functionalities housed in their classes, it becomes easy to integrate an update or modification of some part of the transcription system, for example, replacing Whisper with a new model. This module makes sure that faithful audio and transcription data will prompt means to store and retrieve it and make it easily available to update the medical form JSON.

c) Audio Processing and Transcription Module: This module records Audio as input and determines language and transcribes it to text using pyaudio library along with the whisper model. The Audio Capture class of the software controls the recording process and cuts it into short bursts in.wav format. These .wav files are produced by The Whisper Transcription class and the class then produces accurate transcriptions for more processing. Techniques within these classes cater for each particular phase of processing (for example, language identification, transcription or error control), making a stream move from audio capture to text generation as seamless as possible. This way, encapsulation of each function in audio processing funnel makes the system modular and capable of handling different extents and formats of audio captures, and of shifting between different models of transcriptions.

d) Medical Form (JSON) Management Module: This module organizes and updates the structured JSON medical record based on new information extracted from audio transcriptions. A JSON Medical Form class represents the patient’s medical record in JSON format. It includes methods for loading existing records, updating fields, and saving changes. They inherit from the core JSON Medical Form class and include specific methods for updating individual fields within their section. This structure allows for modular and targeted updates, where only relevant fields are altered based on transcription input, preserving data integrity and reducing the risk of overwriting unrelated data [11].

The module’s structure ensures that each update is validated and properly integrated into the JSON form, making it reliable for medical data management.

e) User Interface and Interaction Module: This module also uses vite for easy interaction with the system and is implemented using react. It allows a healthcare worker check, modify, and engage with the data contained in a patient’s EHR. Each component like Patient Record, Appointment Field and Action Button in the maximum UI are based on a UI Component and each of them represents some specific part of the User Interface. It decomposes web page functionality into component-based parts; Subtype polymorphism and inheritance make these components have like form validation, update triggers or other; while it allows specific service type for different interaction kind. Encapsulation affords the state and behaviour of each UI element to be independent in order to decrease the likelihood of run-on side issues, and to enhance maintainability [10]. This means that each UI design is made as a module that does not blend with other UI designs to avoid situations where the UI changes affect the flow of use of other modules.

f) Audio Recording and WAV File Generation Module: This module records voice input and saves voice as.wav file to be transcribed later. The Audio Recorder class is for recording and playback of the sound and it stores short bursts of sound as .wav files. These files are managed by the WAV File Manager class in order to avail them when needed for transcription. Recording logic is thus encapsulated allowing the system to handle the audio effectively; the scalability issue is thus handled through scalability of the audio storage system with easy access to the data [2]. Here, modularity makes it very easy to extend or even transform audio processing without affecting post processing steps.

g) Inference and JSON Update Module: This module takes transcriptions, modifies the JSON form, and allows only particular fields to be changed depending on the content of the transcription. The Inference Engine class is also responsible for the retrieval of data and the check on fields that should not be overwriting. JSON Updater class is used to update the JSON structure with the extracted information so that update runs smoothly. JSON form protection from outside modification can be provided by encapsulation or making the separate method containing a list of methods for updating entries to keep the internal database consistent.

h) Error Handling and Data Validation: This module ensures the system's reliability and data integrity by validating JSON output and logging any errors encountered during processing. The Error Logger class records errors and warnings, such as transcription issues or JSON structure mismatches, for debugging and tracking purposes. The Validator class verifies the correctness of JSON structure and ensures data consistency, minimizing the risk of corrupted records. By employing inheritance, various error types (e.g., transcription, JSON parsing) are managed by specialized classes, enabling customized error handling strategies [12].
This structure ensures that any issues are efficiently detected and logged, supporting proactive error management and enhancing the system's reliability. Table 2.1 describes the description of all the models in the project.

Table 2.1: Basic description of all models

Module	Description	Technologies Used
Question Answering (QA)	Processes patient responses through ChatGroq and Deepgram, guiding them	ChatGroq, Deepgram,

	through structured questions.	NLP
Audio Processing	Records and transcribes voice inputs using Whisper model and pyaudio, with language detection capabilities.	Whisper, pyaudio
Medical Form Management	Updates a JSON file with patient data, integrating responses into a structured medical record format.	JSON, MongoDB
User Interface	Frontend for interacting with the system, allowing voice commands and appointment management.	Vite, React
Backend Integration	Manages data flow, session states, and APIs for appointments, interacting with both frontend and NLP tasks.	Express.js, Django, MongoDB
Voice Assistant	Interprets voice commands, executing appointment management tasks and providing real-time feedback.	NLP, Deepgram, Whisper
Authentication and Security	Implements secure, encrypted access and role-based permissions for different users.	token-based auth, encryption

The system enables recording of audio, transcribing it into text, and subsequently updating a structured medical form in JSON format with relevant details derived from the transcription. It utilizes a combination of audio processing libraries, natural language processing (NLP) capabilities, and JSON handling to create an efficient workflow for medical professionals. The integration of the ChatGroq model and Deepgram for TTS ensures accurate transcription and dynamic updating of medical records. The table 2.2 describes the functionality of each component:

Table 2.2: Functionality of each component

Component	Function
Deepgram	Converts recorded audio into text with high accuracy.
ChatGroq	Uses NLP to analyze transcribed text and extract key medical details.
JSON Library	Structures extracted data into a JSON format for recordkeeping.
JSON Medical Form	Stores patient data and medical details in a standardized format.

Analysis and Workflow of the Smart Assistance Appointment System is described below:

a) User Interaction

i. Account Management: Users can create an account through the web interface, accessing appointment management functions such as booking and cancelling appointments. Requests are processed by the Application Server, handling login and authentication processes securely.

ii. Voice Command Support: The system includes a voice interface where users can issue voice commands, enabling hands-free interaction for appointment management.

b) Voice Processing

i. Voice Assistant: Captures user voice commands and sends them to the Speech-to-Text Module for transcription.

ii. NLP Model: The transcribed text is processed by an NLP model to identify intent, enabling the system to recognize user instructions for booking, modifying, or cancelling appointments.

c) Appointment Management

i. Scheduler: Manages appointment requests from users by interacting with the Database. It records new appointments, updates modified appointments, and processes cancellations based on user inputs.

ii. Synchronization: Ensures the database is kept up-to-date with all user requests and that any changes are reflected across the system in real-time.

d) Database

i. Data Storage: Holds user login information, appointment data, and voice interaction logs. This enables comprehensive data management and tracking.

ii. Reporting: Generates system usage and appointment statistics reports for administrative oversight, providing valuable insights into user behavior and system performance.

e) Administrator:

i. Monitoring: Administrators access reports and manage system performance using the data stored in the database.

ii. Analytics: Detailed analytics enable administrators to assess system efficiency, identify areas for improvement, and ensure that the user experience remains optimal.

f) User Authentication and Security:

Encryption and Logging: Data is encrypted both at rest and in transit, and an audit log tracks all access attempts, ensuring data integrity and security compliance.

g) Integration with Hospital Management System (HMS):

i. Real-Time Data Sync: Uses APIs to synchronize appointment data with the existing HMS, ensuring up-to-date information on doctor availability and scheduling.

ii. Notifications: Provides real-time updates on appointment status changes, cancellations, or confirmations to both patients and administrative staff. Provides real-time updates on appointment status changes, cancellations, or confirmations to both patients and administrative staff.

The Fig 3.1 describes the **complete flowchart** of the system

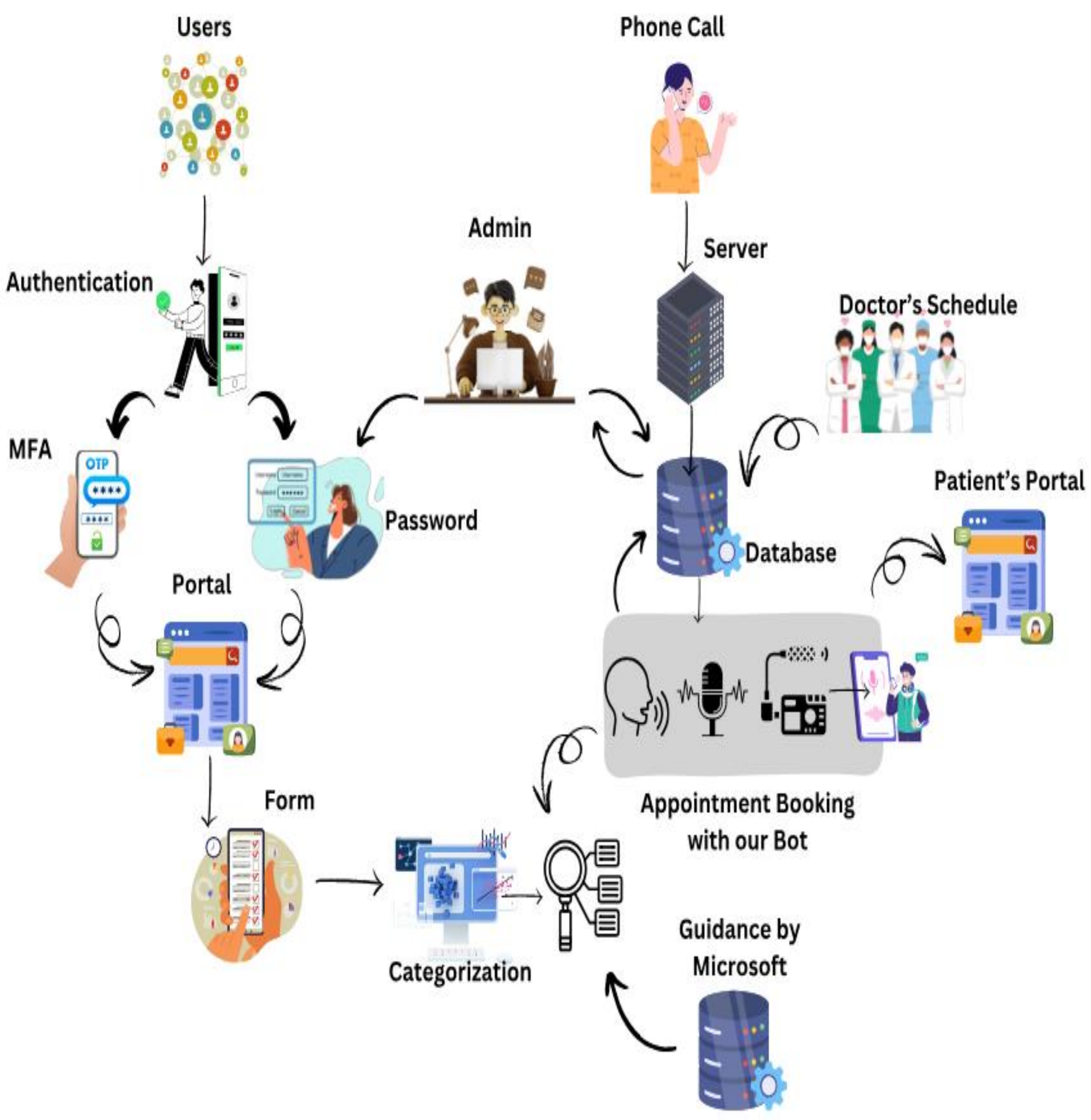


Fig 3.1: Flowchart of the Appointment System

The **Frontend Module (Vite + React)** compromises of the following features:

a) Voice Command Integration: Allows users to manage appointments through voice commands, processed in real-time for a seamless experience.

b) Appointment Dashboard: A comprehensive interface for viewing, scheduling, or cancelling appointments.

c) **Real-Time Status:** Appointment status is dynamically updated, reflecting changes immediately.

d) **Responsive Design:** Supports access on multiple device types, providing flexibility for users.

The Database Module (MongoDB) compromises of the following features:

a) **Patient Record Management:** Stores and manages patient data, including medical history and appointment preferences.

b) **Voice Logs:** Records patient interactions for performance evaluation and system improvement.

The Voice Assistant Module compromises of the following features:

a) **Speech-to-Text:** Converts spoken language to text, allowing the system to interpret commands.

b) **Command Processing:** Executes user requests for appointment management, ensuring accurate results.

c) **Immediate Feedback:** Provides real-time audio responses, enhancing the interactive experience.

The User Interface (UI) Module compromises of the following features:

a) **Web Portal:** It enables appointment management via a user-friendly, question-driven interface.

b) **Voice Command Interface:** It provides an alternative interaction mode where users can manage appointments verbally.

In the Smart Assistance Appointment System, VoIP technologies integrated with WebSocket-optimized to handle HTTP requires for real-time communication especially for capturing and handling phone line utterances with voice recognition system. This approach is one important to the system, given its goal of facilitating flow of voice data, as it offers a way of minimizing the latency and network load which can be critical where bandwidth may be constrained. In the case of a medical appointment system, the voice transcription requirement is crucial since it calibrates the rate of information capture and processing to match the rate of patients' requests, which are critical to the interaction between the system and the patient. It integrates normal HTTP requests and voip to make a persistent, symmetrical, full-duplex connection between the client and server, providing a Real-Socket environment for returning/forwarding real-time voice data without the need to create new connections repeatedly.

Latency is also brought down to an acceptable level and network traffic is minimized, so that every patient's speech input, be it to confirm, reschedule or cancel an appointment, reaches the server almost immediately, which allows for a smooth consumer interaction. This constant and low-delay transmission of data enhances the efficiency of the system particularly during such squeeze operations leading to an improved usership as the patients engage the voice interface of the appointment scheduling system. Also, the project's voice recognition part is based on the GET HTTP request for particular voice data processes, especially when interacting with other services for, for example, transcription or further natural language processing. These HTTP requests are actually fine tuned so that the data flow can be well contained by maximizing the compression of voice data and also by allowing only specific packets to be privileged in a way that will ensure that the network does not get congested. It is important that each level of hierarchy is selectively prioritized so that the voice recognition system may process the spoken commands within shortest time, improve on accuracy without compromising on the time to interpret commands or generate a response. This setup benefits the appointment system by ensuring that patient inputs are processed with minimal lag, resulting in an experience that feels seamless and conversational.

The combination of VoIP and WebSocket-optimized HTTP requests thus provides several key advantages for healthcare-oriented voice recognition applications. In a medical context, it is crucial that communication remains consistent and uninterrupted, particularly when handling sensitive patient information. By enabling efficient, real-time voice data transmission with minimized network impact, the system can handle patient queries or scheduling requests promptly, without interruption. This not only enhances the usability of the appointment system but also establishes trust, as patients and healthcare providers alike can rely on timely and accurate data processing in a secure manner.

This technology-driven approach serves to improve the overall user experience, making voice-based interaction with the Smart Assistance Appointment System a practical, reliable, and efficient process for both patients and healthcare administrators. The system's use of VoIP with WebSocket-optimized HTTP requests to manage voice recognition aligns closely with the goal of delivering high-quality, responsive services, and ultimately supports a more engaged, accessible, and streamlined experience for patients navigating their healthcare appointments.

3.2 Detail Design

This section provides an in-depth look at the system's architecture and data flow, detailing how various components interact to fulfill the Smart Assistance Appointment System's functionalities.

3.2.1 DFDs

a) Level 0

i. Initially, the patient voice commands or web requests are processed by the **Voice Command Processing** module. This translates user inputs into structured appointment requests and identifies the user's intent.

ii. Next, these requests are sent to the **Appointment Scheduler**, which interacts with the **Database** to book, modify, or cancel appointments as needed. It ensures that all actions are accurately recorded and synchronized.

iii. Simultaneously, user authentication occurs through the **User Authentication** module, verifying credentials and maintaining secure access to the system.

iv. Following this, data flow between the components is managed by the **Application Server**, ensuring smooth communication and coordination among all subsystems.

v. Finally, the **Administrator** can access analytical reports generated from user activities, allowing for insights into the system's performance and helping to ensure a high-quality user experience.

The Level 0 Data Flow Diagram (DFD) illustrating the described process is provided below in **Figure 3.2**. This diagram gives a high-level view of the primary data flows within the system, capturing the essential processes, data stores, and interactions with external entities. It provides a simplified yet comprehensive representation of the core operations, showcasing how the system captures, processes, and transmits information. This initial level DFD serves as the foundation for further detailed breakdowns in subsequent levels, where each process will be elaborated upon to highlight the specific functionalities and data exchanges involved.

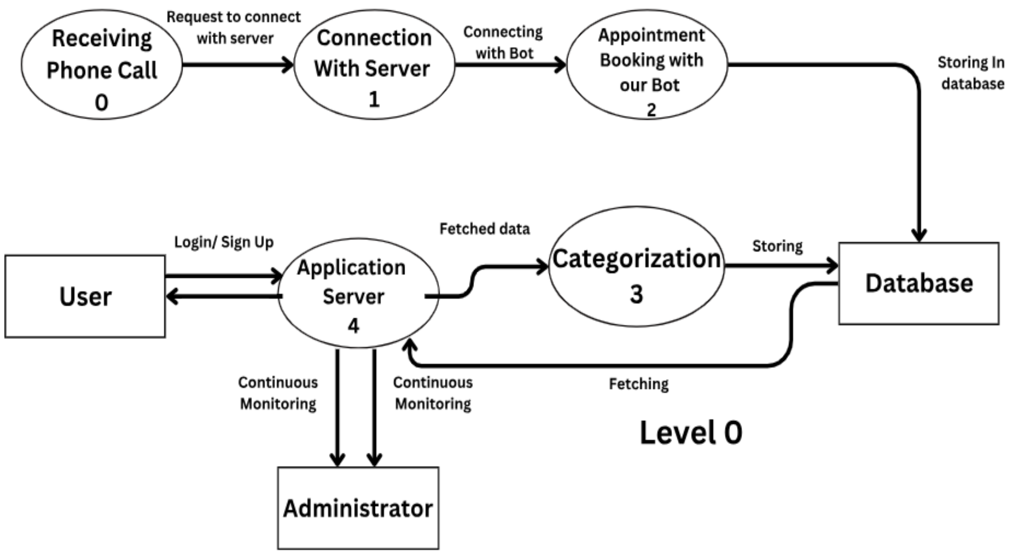


Fig 3.2: Level 0 DFD

b) Level 1

In the following Fig 3.3 the Level 1 DFD of Appointment Booking with the Bot, is described. It describes the complete workflow how the communication with bot starts and finally goes to the storing response module.

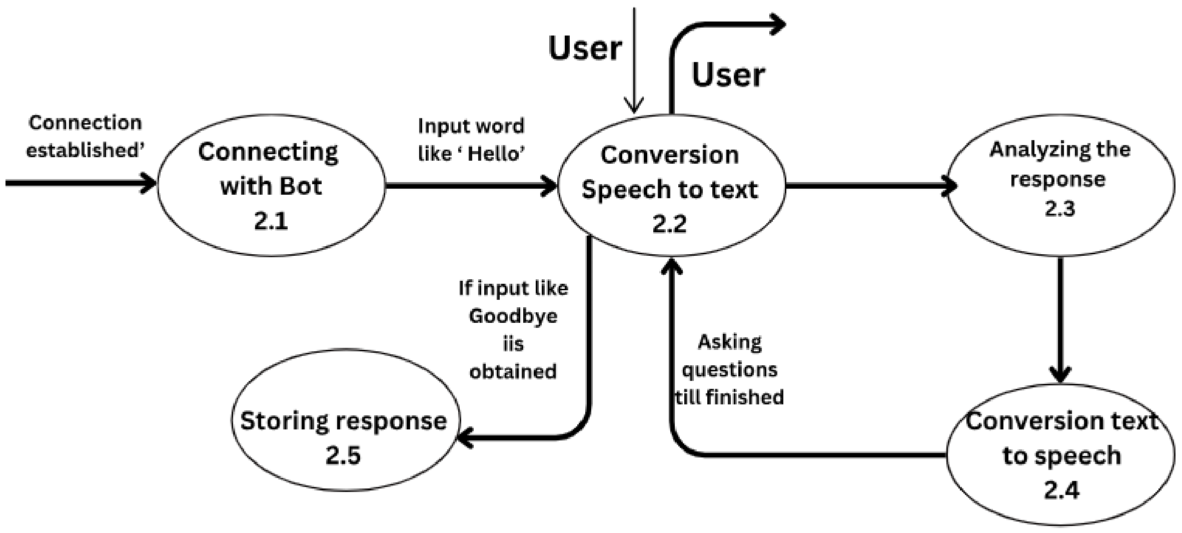

User
User
Connection established'
Connecting with Bot 2.1
Input word like ' Hello'
Conversion Speech to text 2.2
Analyzing the response 2.3
If input like Goodbye iis obtained
Asking questions till finished
Storing response 2.5
Conversion text to speech 2.4

Fig 3.3: Appointment Booking with the Bot

In the following Fig 3.4 the Level 1 DFD of Receiving Phone Call. It describes the complete workflow how the connection with call starts and finally goes to the storing response module.

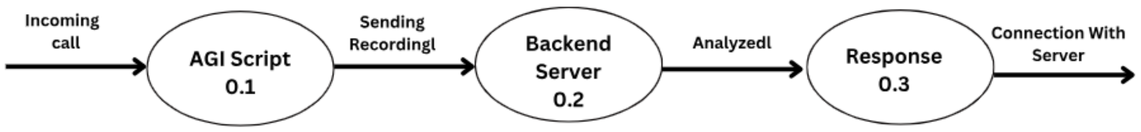


Fig 3.4: Receiving Phone Call

In the following Fig 3.5 and Fig 3.6 the Level 1 DFD of Receiving Phone Call and application server are described. It describes the complete workflow how the authentication and connection with the server are established.

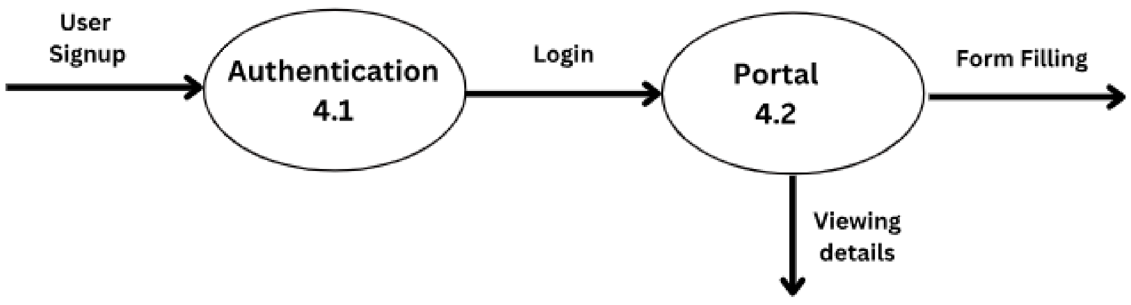


Fig 3.5: Application Server

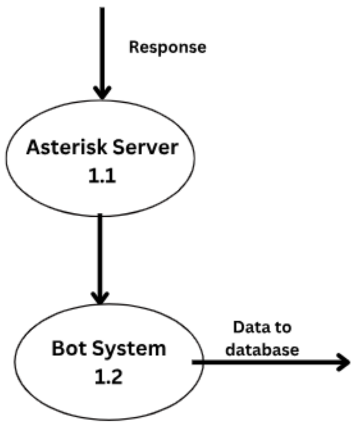


Fig 3.6: Connection with Server

In the following Fig 3.7 the Level 1 DFD of Categorization is described. It describes the complete workflow how the symptoms are classified as low, medium, high.

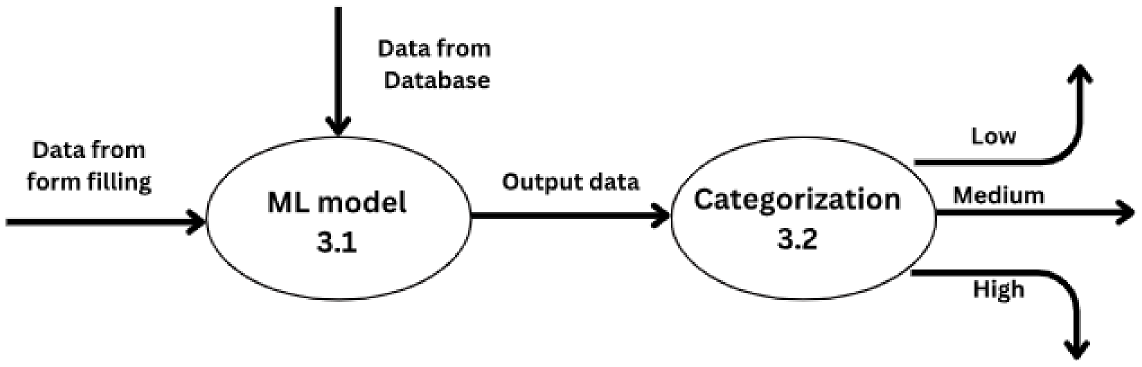


Fig 3.7: Categorization

c) Level 2

In the following Figures Fig 3.8 and Fig 3.9 the Text to Speech, Speech to Text and Response recording is described respectively:

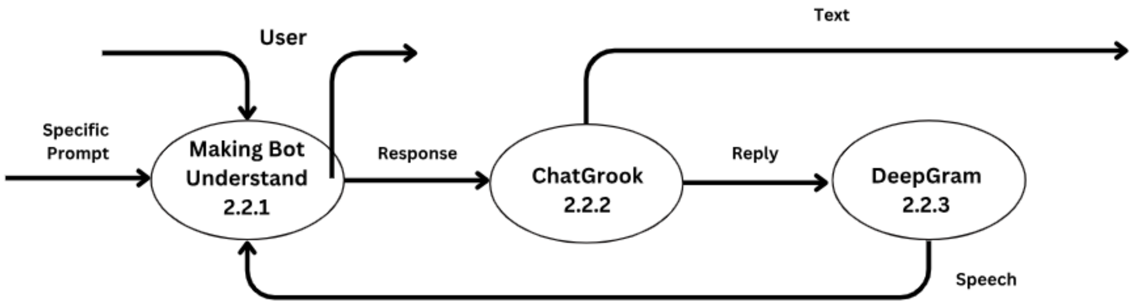


Fig 3.8: Text to Speech

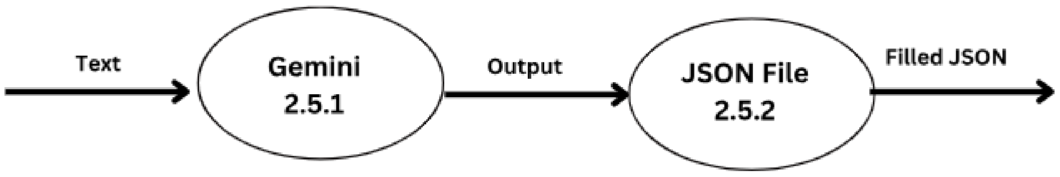


Fig 3.9: Response Recording

3.2.2 Database Design

Figure 3.10 presents a detailed **Entity-Relationship Diagram (ERD)** for the appointment booking system, meticulously outlining the core entities—**Users, Doctors, and Appointments**—along with their interconnections and significant attributes necessary for managing and executing the appointment booking workflow efficiently. This ERD offers a structured overview of how users and doctors interact through appointments, emphasizing essential data relationships that support functionality, secure access, and accurate record-keeping.

The **Users** entity encompasses critical details such as **User ID, Name, Contact Information (email, phone number), and Account Type** (e.g., patient, healthcare provider, or administrative staff), serving as the foundation for personalization, authentication, and access control. Each attribute within the Users entity plays a role in ensuring secure, role-specific access, allowing patients to book or manage their own appointments, while enabling administrative staff to oversee scheduling and maintain the database. The Account Type attribute, for instance, determines the level of access for each user, distinguishing between patients who can view their personal appointments and administrative users with broader access. This differentiation is fundamental in healthcare systems, where patient privacy and data security are paramount.

The **Doctors** entity represents healthcare providers within the system, including attributes such as **Doctor ID, Specialization, Contact Information, and Availability Schedule**. The specialization attribute is essential, as it helps patients and administrative staff match appointment requests to the relevant type of care, ensuring that users are directed to doctors best suited to address their specific health needs. Moreover, the availability schedule is vital in supporting dynamic appointment booking by showing which time slots are open for scheduling, thus avoiding double-booking and optimizing doctors' time. These details allow the system to balance patient demand with healthcare provider resources, enhancing both user experience and operational efficiency.

Central to the ERD is the **Appointments** entity, which functions as the core of the appointment booking system. Key attributes include **Appointment ID, Date and Time**. The **Status** field is critical for tracking an appointment’s progress and history, providing a clear record of past, current, and upcoming interactions. This status management, combined with timestamps, supports smooth appointment flow and helps administrative staff manage schedules effectively, ensuring that both users and doctors remain informed and prepared.

The **relationships** in the ERD clarify how data and users interact within the system. For example, a **One-to-Many** relationship exists between **Users and Appointments**, illustrating that a single user may book multiple appointments over time, while each appointment pertains to one unique user. Likewise, a **One-to-Many** relationship links **Doctors and Appointments**, showing that each doctor can serve numerous appointments, though each appointment is linked to a single doctor at any given time. These structured relationships prevent data redundancy, improve system scalability, and maintain a clear structure for query optimization.

In addition to these core relationships, Figure 3.10 incorporates **key attributes** that enable seamless integration and maintenance of records. With these relationships in place, the ERD provides an efficient system for tracking patient interactions, managing doctor schedules, and securing sensitive data. As the appointment system grows, these relationships and structured attributes help ensure that each component—users, doctors, and appointments—continues to function cohesively, supporting real-time scheduling updates and allowing for robust data analytics to improve healthcare operations. These relationships establish a well-structured ERD that supports efficient tracking of patient interactions, effective management of doctor schedules, and secure handling of sensitive data. As the appointment system scales, these relationships and organized attributes ensure that each component—whether users, doctors, or appointments—operates cohesively within the system. This structure not only allows for real-time scheduling updates but also enables advanced data analytics, ultimately enhancing healthcare operations and supporting continuous improvements in patient care.

These relationships establish a well-structured ERD that supports efficient tracking of patient interactions, effective management of doctor schedules, and secure handling of sensitive data. As the appointment system scales, these relationships and organized attributes ensure that each component—whether users, doctors, or appointments—operates cohesively within the system. This structure not only allows for real-time scheduling updates but also enables advanced data analytics, ultimately enhancing healthcare operations and supporting continuous improvements in patient care. This structure not only allows for real-time scheduling updates but also enables advanced data analytics, ultimately enhancing healthcare operations and supporting continuous improvements in patient care.

In essence, **Figure 3.10** is a comprehensive representation of the system’s data infrastructure. It highlights the structured pathways through which users, doctors, and appointments are managed, ensuring that essential information is readily accessible while maintaining secure and organized data flow. By capturing these elements, this ERD serves as a detailed blueprint that informs system design, supports future scalability, and establishes a robust foundation for high-quality, secure patient care and efficient healthcare resource management.

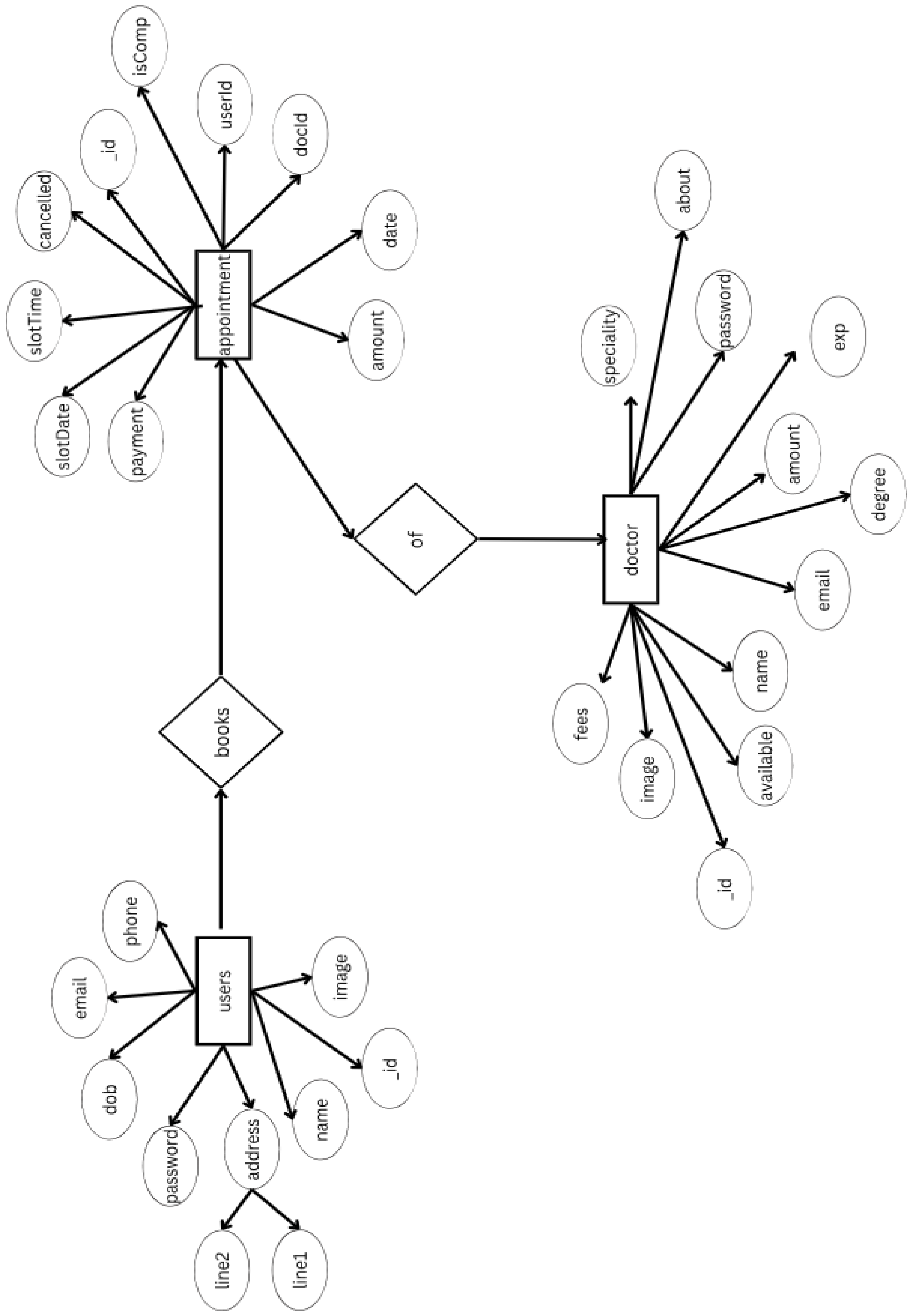


Fig 3.10: ER Diagram

The key attributes of the figure are described below:

a) Users

i. Attributes:

- _id: Unique identifier for the user.
- name: The user's full name.
- dob: Date of birth of the user.
- email: User's email address.
- phone: Contact phone number for the user.
- address: Address of the user, composed of:
- ▪ line1: First line of the address.
- ▪ line2: Second line of the address.
- password: Password for authentication.
- image: Profile image of the user.

b) Doctor

i. Attributes:

- _id: Unique identifier for the doctor.
- name: The doctor's name.
- email: Doctor's email address.
- password: Authentication password for the doctor.
- speciality: Doctor's area of specialization.
- about: Brief description or bio about the doctor.

- exp: Doctor’s years of experience.
- degree: Educational qualifications of the doctor.
- available: Doctor’s availability status.
- fees: Consultation fee for appointments.
- amount: Amount associated with appointments.
- image: Doctor's profile image.

c) Appointment

i. Attributes:

- _id: Unique identifier for the appointment.
- slotDate: Date of the appointment slot.
- slotTime: Time of the appointment slot.
- payment: Payment information for the appointment.
- cancelled: Indicates if the appointment has been canceled.
- isComp: Indicates if the appointment is completed.
- userId: Foreign key linking to the users entity.
- docId: Foreign key linking to the doctor entity.
- amount: Total amount charged for the appointment.

3.2.3 Flowchart

The flowchart in the Fig 3.11 illustrates the step-by-step process of the Smart Assistance Appointment System's interaction with a user, designed to gather and record essential information for appointment scheduling. The process begins with the system responding to the user’s initial greeting ("Hello"), establishing a conversational connection. The system uses **Whisper**, an audio transcription model, to transcribe the user's audio inputs into text, enabling the system to understand spoken language accurately. This transcription step is crucial as it allows the system to process voice inputs, making the interaction natural and accessible, especially for users who prefer voice over text.

Once the initial greeting is processed, the system checks if the user wants to end the conversation by saying "Goodbye." If the system detects a goodbye from the user, it promptly responds with a farewell, ending the interaction gracefully. However, if no goodbye is detected, the system transitions into the main data-gathering phase, where it begins asking the user a series of questions. Each question is designed to collect specific details required for scheduling an appointment, such as the user's name, age, or medical preferences. This structured questioning approach keeps the conversation focused and organized, ensuring that essential information is captured efficiently.

As the user responds to each question, their input is sent to a **Question Answering Model** to process and validate the response. The model analyzes the input for relevance and clarity, confirming if it provides the needed information. If the response is invalid or unclear, the system prompts the user to answer again, helping ensure data accuracy. This step highlights the system's resilience and user-friendliness, as it actively seeks to correct incomplete or vague responses without causing user frustration [11].

When a valid response is received, it is updated in a **JSON list**, which serves as the system's storage format for user data. This JSON file organizes responses in a structured manner, enabling easy retrieval and integration with other parts of the appointment system. The system continues this cycle of questioning and validation until all required information is collected. Once completed, the system checks if there are any pending questions; if not, it displays the compiled JSON file, effectively summarizing the user’s responses. This final display ensures that the user can review the recorded information before concluding the interaction, enhancing transparency and user satisfaction.

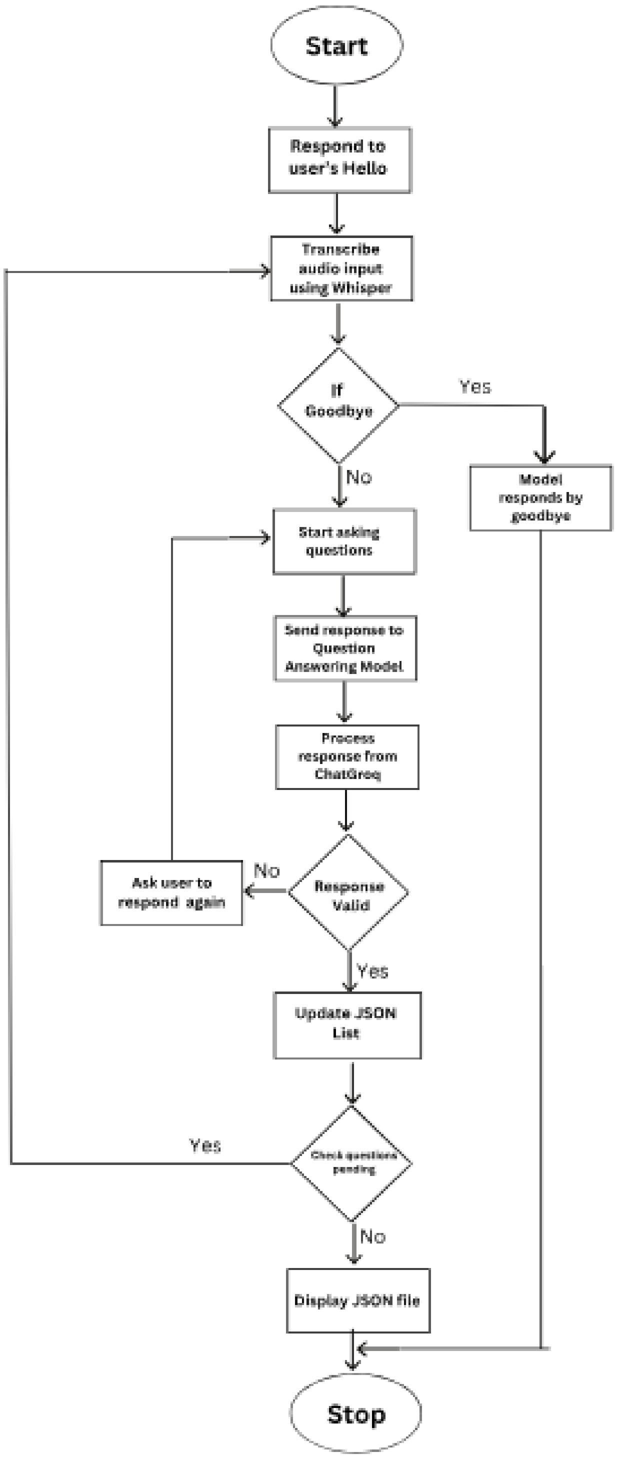

Start
Respond to user's Hello
Transcribe audio input using Whisper
If Goodbye
Yes
Model responds by goodbye
No
Start asking questions
Send response to Question Answering Model
Process response from ChatGroq
Response Valid
No
Ask user to respond again
Yes
Update JSON List
Check questions pending
Yes
No
Display JSON file
Stop

Fig 3.11: Flowchart of Text to Speech/ Speech to Text

3.3 User Interface Design

The UI of the appointment management system, built using the MERN stack (MongoDB, Express, React, and Node.js), is designed to provide users with an intuitive, efficient, and visually engaging platform for scheduling and managing appointments. The MERN stack enables a seamless integration of frontend and backend components, enhancing the overall responsiveness, scalability, and performance of the application. The description is described below:

a) MongoDB: MongoDB is the NoSQL database used to store or process all kinds of appointment related data including user details, doctors, appointments, and payments. MongoDB is a type of NoSQL database that utilizes documents as opposed to tables; this prove especially useful with an appointment system where the structure of the data varies and may be dynamic. This application requires scalable database systems that generate reasonably fast data retrieval; hence, using MongoDB as the database for this application will prove beneficial, given the large volumes of data involved.

b) Express: Express is employed as the backend framework with which the server-side functionality of the application is implemented. It gives a simplistic approach to running handling as well as handling of HTTP request, routing and middleware to ensure clear communication between the frontend and MongoDB. Express dispatches API endpoints for appointment creation, updating and reading, as well as for deletion, making handling and interacting with data seamless.
c) React: It used for making user interface of the appointment system more dynamic and responsive. For instance, with the help of React, application developers can create numerous familiar UI elements such as appointment scheduling forms, doctor's profile, users' dashboard, appointment calendars, etc. React's Virtual DOM allows a fast rendering process and a smooth user experience on the page, and it's easily managing the state of the appointments and interactions of the user.

d) Node.js: Node.js is the runtime software used in development on the server-side where JavaScript can be run. This non-synchronous, event-based structure allows for the management of multiple requests simultaneously simultaneously, which would otherwise cause the web application to become bogged down by the requests. Node .js connects successfully with MongoDB and creates a strong environment to implement server-side business logics from managing users' authentication to request handling of an appointment.

The appointment system basically includes several **core features** that enhance usability, security, and flexibility for both users and service providers. They are described below:

a) User Registration and Authentication: The application allows users to create accounts, providing them with personalized access to schedule and manage their appointments. With secure authentication mechanisms, such as password encryption, the system ensures that user credentials are protected, and access to sensitive data is restricted [4].

b) Responsive Design: The UI of the appointment system is fully responsive, enabling users to access the platform across a range of devices, including desktops, tablets, and smartphones. By utilizing responsive design principles in React and Bootstrap, the application provides a consistent and optimized experience across all screen sizes, catering to users who prefer different devices.

c) Interactive Appointment Scheduling: The appointment system features an interactive interface, allowing users to view available slots, select preferred times, and book appointments seamlessly. Real-time updates, such as slot availability, cancellation statuses, and confirmations, enhance user engagement. Interactive elements, such as calendar views, dropdowns, and auto-suggestions, streamline the scheduling process, making it user-friendly and efficient.

d) Role-Based Access Control: The system includes role-based access control, distinguishing between users and service providers (e.g., doctors). This feature allows different types of users to access different functionalities, ensuring that doctors can manage their schedules, view patient details, and update availability, while regular users can only book and manage their appointments [9].

e) Data Security and Encryption: Data security is a priority in the appointment system. User data, especially passwords, is encrypted to prevent unauthorized access and ensure confidentiality. By implementing secure hashing algorithms for passwords, the system adheres to industry standards for data protection and user privacy.

3.4 Methodology

a) Requirement Analysis

i. Gathering Requirements: Both functional and non functional requirements were obtained and recorded by questionnaires as well as studying characteristics of similar systems. At a later date, we have also classified public demands based on proportions such as importance, implement ability, and relevance to users, for features such as user registration, and appointment booking.

ii. Analysis and Documentation: For this activity, the requirements collected have been to some extent evaluated to define Dependencies, Risks and Conflicts between functionalities or Later on, the requirements documented in details have been maintained in the standardized format for the better understanding, complete, and traceability throughout the project life cycle.

b) Design Phase

i. Architectural Design: Whereas, in this project, we were responsible for the architecture of the overall appointment management system which comprises a frontend, a backend, and the database client; the frontend was built in React, the backend in Express and Nodejs, while the database was mongoDB. By Choosing the MERN stack as the technological base to implement further development, guarantee extensibility and real-time updates, and synchronizing the components [6].

ii. User Interface Design: We have created wireframes and mockups for core screens, such as the booking interface, user dashboard, and doctor profiles. Later on, designed user-friendly interfaces that facilitate smooth navigation and interaction, with responsiveness to support different devices.

iii. Design Reviews: Subsequently, we discussed with the stakeholders the specifications and got their feedback regarding the design and based on the feedback given we revised designs and made necessary changes so as to ensure that the designs would deliver what was intended for the project and what is expected out of it.

c) Development

i. Frontend Development: The frontend parts of the application have been built utilizing React, where UI for registration, appointment, and account has been created. We have also used aspects of responsive designs and trendy CSS frameworks such as Tailwind CSS in order to make the application flexible to work on various devices and screen resolutions.

ii. Backend Development: The backend functionality was created in Node.js and Express as significant features of the application, for example, user login and registration, scheduling of appointments, and doctor scheduling.

iii. Integration: To achieve smooth data flow between the client- and server-side, we have implemented combined frontend and backend parts of the application. We have also ensured proper and secure means of communication especially for sensitive operations like; user authentication and payments.

d) Testing

i. Testing Strategy: First, detailed testing plan was described including UNIT testing, INTEGRATION testing where frontend and back end would be united together, SYSTEM testing and lastly the USER TESTING. Later on, we created positive test scenarios for critical functions like scheduling, user registration and payment and negative test cases as well.

ii. Testing Automation: We automated testing processes using tools like Jest for unit testing React components and Mocha/Chai for backend testing.

e) Deployment

i. Deployment Planning: Defined the scheme of the deployment process, the setup of the servers, their configuration and the general deployment plan for a staging and production environment. Further on, some particular clouds, for example, AWS or Heroku, were chosen to deploy the application taking into consideration the need in scalability, reliability of the cloud platform, and the costs made.

ii. Deployment Automation: For the deployment we further precluded the use of CI/CD pipelines such as GitHub Actions or Jenkins to minimize on the long one-time manual intervention. It assisted in keeping track of backup and rollbacks to help provide risks cover during deployments.

iii. Monitoring and Maintenance: Set up monitoring and alerting systems (using tools like New Relic or AWS CloudWatch) to track application performance, availability, and security. Later, established a maintenance routine to promptly address issues, apply updates, and continuously improve the application based on user feedback.

The block diagram (see Fig 3.12) illustrates the flow of the appointment management system:

a) User Access and Authentication: Users login or create account and after verification gains access to a control panel through which he or she can see doctors, their calendar and open appointments.

b) Booking Process: Participants choose a doctor and preferred time. Again the system looks for availability and if the slot is open the appointment is made and the confirmation sent.

c) Notifications and Reminders: The system also issues alerts and notifications to the users about the scheduled appointment, about cancellations or rescheduling.

d) Monitoring and Feedback Loop: After the appointment, the users are able to give their feedback in a feedback section which is retained for progression purposes for the doctor as well as the patient.

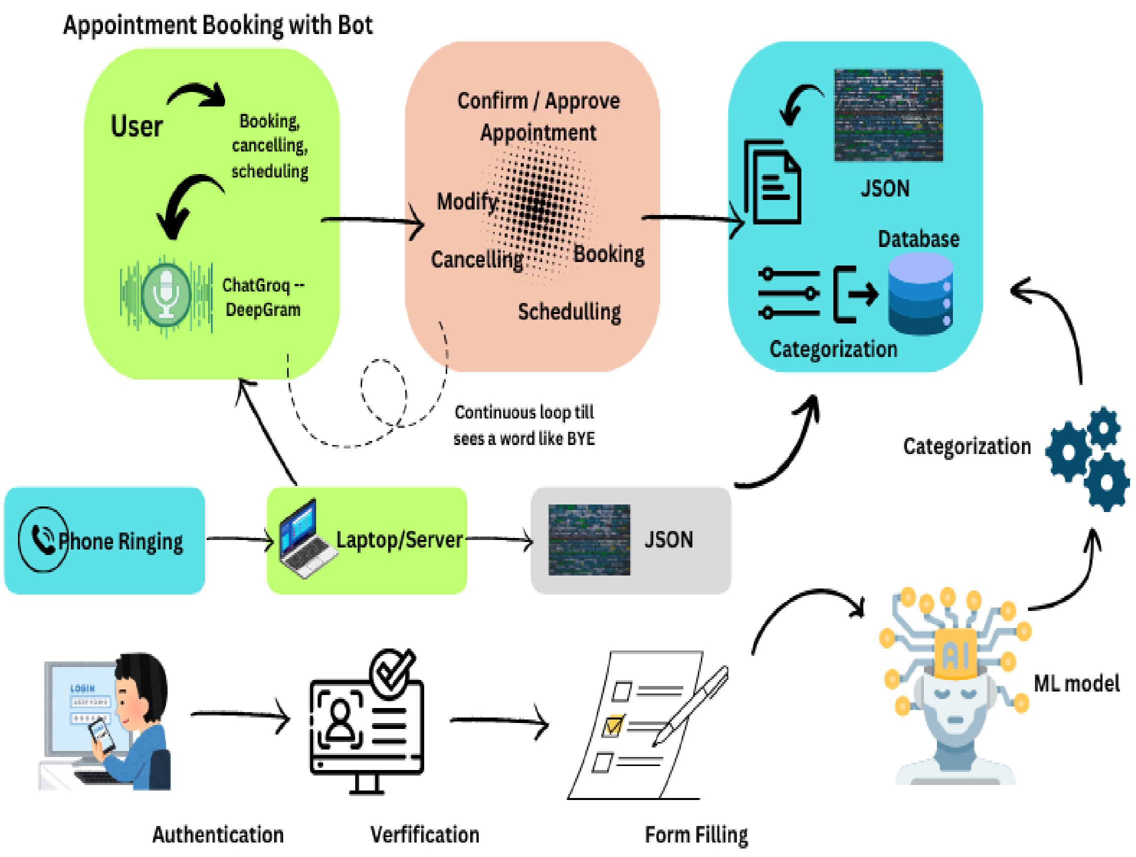


Fig 3.12: Block diagram for Appointment system

The project uses a variety of tools, including web frameworks, JSON handling, NLP, and audio processing libraries. The different modules are discussed below:

a) Question Answering (QA) Module: This module employs the ChatGroq, DeepGram and Whisper model, designed to process natural language transcriptions from voice notes. The transcribed user response is fed into the ChatGroq model. It interprets the input and keep a check that the respond belongs to a particular asked question and only move to next question if the present question is answered. The text response from ChatGroq is sent to **DeepGram** and the audio is sent back to the server and which is played using **ffplay** module. In the end the collected user responses are passed to **Gemini** to fill a particular JSON file. **The working goes as described below:**

i. User prompts with "Hello."

ii. The ChatGroq model processes the input and generates a response, prompting the response.

iii. This interaction continues until a termination word, such as "goodbye," is detected.

b) Data Collection Module: In this system, audio recordings and their transcriptions are systematically collected. The voice inputs are recorded and transformed into structured text format via the Whisper model. This text is then analyzed to update an existing medical form stored in a JSON file, thereby streamlining medical record-keeping by automating data entry tasks.

c) Audio Processing and Transcription Module: This module deals with audio listening, recording, and conversion to text through the pyaudio toolkit. Audio inputs are recorded in short duration (approximately 5000 ms) using the pyaudio and temporary .wav format chunks. These audio chunks are then fed into the Whisper model to generate accurate transcriptions on which further communication and data entry can be based on. Also, the module identifies the language of the input in order to transcribe properly and understand it.

d) Medical Form (JSON) Management Module: The system also keeps a medical record in JSON style and can load records from an existing file if available. This is a typical JSON form and contains data about patient – his/her identity data, contacts details, and history, and emergency contacts' data. The given module can change the dictionary which is nested and input the new data to the current one. Whenever there is a transcription to the call, specific fields in the JSON are modified with reference to the transcription results. Finally the collected user responses are handed over to Gemini in order to feed a specific JSON file.

e) User Interface and Interaction Module: The functional module of the structure is the **User Interface and Interaction Module** that offers a web-based interface for this smart assistant system. Clinicians using the web portal can view, amend, or modify patient records' information, and assessment of other structured data updates in an easily map like manner. The fields displayed on the portal include, personal information, medical history, appointments and others in such a way that the medical practitioners ease through the various records of the patients. It also made these processes easier to understand and the circuit helpful in improving user experience as the interface is very easy to use and accessible by all the health care staff.

f) Audio Recording and WAV File Generation Module: This module makes it easy capturing of voice notes using the pyaudio library to capture and store audio data in the form of WAV files for transcription. The obtained audio data is important for further corresponding to the user's input, which is then converted into a format that can be used to update the medical form [3].

g) Inference and JSON Update Module: The last of these/MERGE/combines new data parsing results taken from transcriptions with the JSON for the medical form. It has safeguard of preventing over-writing of wrong areas, hence ensuring only appropriate fields are updated. This module is for saving the modified JSON, which preserves the most recent information shared through transcribed audio for further purpose. Afterward, the processed JSON includes changes to accommodate them to guarantee all patient data to be recorded appropriately.

h) Error Handling and Data Validation: The system incorporates validation mechanisms to ensure that the JSON output maintains the correct format and structure. Any errors encountered during transcription or saving processes are logged, and appropriate messages are displayed to notify users of issues. This enhances reliability and ensures data integrity in medical records.

Chapter 4. Implementation and Testing

4.1 Introduction to Languages, IDE’s, Tools and Technologies used for Project work

The choice of languages, IDEs (Integrated Development Environments), tools, and technologies plays a crucial role in enabling efficient development, implementation, and deployment of the solution. An overview of the implementation is described as below:

a) Python: Python serves as the cornerstone of the Smart Assistance Appointment System's development, providing a versatile and powerful programming language ideal for addressing the various challenges in healthcare technology, particularly in automating appointment scheduling and integrating AI-based features. Several factors contribute to Python’s central role in the project:

i. **Versatility**: Python offers a wide range of libraries and frameworks tailored to various domains, including AI, NLP, and web development. Its versatility allows developers to seamlessly integrate components such as voice recognition, NLP processing, and appointment management into the system, making it well-suited for handling multiple functionalities in the healthcare domain.

ii. **Ease of Use**: Python’s clean and intuitive syntax enables quick development and easier collaboration among team members, regardless of their experience level. The simplicity of the language allows developers to focus on solving complex problems related to appointment booking, scheduling, and system optimization without getting bogged down by complex syntax or intricacies.

iii. **Rapid Prototyping**: Python’s dynamic nature allows for rapid prototyping and experimentation. This helps the development team quickly iterate and test different features, such as voice-based appointment scheduling, real-time updates, and interactive user interfaces. Tools like **Jupyter Notebook** and **Google Colab** provide an interactive environment for testing and refining algorithms, accelerating the development process and improving the flexibility of feature implementation.

The project also involves the utilization of command-line interfaces and text editors, particularly in a Linux environment.

a) ***Visual Studio Code (VS Code):*** VS Code is a lightweight and versatile code editor developed by Microsoft, featuring a wide range of extensions and integrations for various programming languages, including Python. VS Code offers an intuitive user interface, customizable layouts, built-in Git support, and an extensive marketplace for extensions, enabling developers to tailor their coding environment to their specific needs. Its integrated terminal allows for seamless interaction with the command line and execution of shell commands.

b) ***Bash Shell:*** The Bash shell serves as the command-line interface (CLI) for interacting with the Linux operating system. It provides a powerful scripting environment, enabling developers to automate tasks, execute shell commands, and manage system resources efficiently. Bash scripts are extensively used for tasks such as file manipulation, process management, environment configuration, and software installation, thereby streamlining development workflows and enhancing productivity.

c) ***Linux Environment:*** The project leverages Linux as the operating system for development and execution due to its robustness, flexibility, and compatibility with open-source software tools and libraries. Linux distributions offer comprehensive package managers, development libraries, and system utilities tailored for software development and scientific computing tasks. By harnessing the capabilities of Linux, the project benefits from enhanced performance, security, and scalability, while also facilitating seamless integration with cloud platforms and containerization technologies for deployment purposes.

4.2 Pseudocode

The pseudocode of the appointment system is described below:

a) Initialization and Setup

```
Initialize ChatGroq, DeepGram, Whisper, and Gemini models
Load existing JSON medical form file if available
Initialize pyaudio for audio recording
Set up UI elements on web portal
```

b) Main Flow (User Interaction Loop)

```
WHILE system is active
    Display "Hello, how can I help you today?"
    Capture user's voice input using Audio Processing and Transcription Module
    Transcribe audio input using Whisper model
    IF transcription contains "goodbye"
        Display "Thank you, goodbye!"
        Terminate interaction loop
    ELSE
        Send transcription to Question Answering Module
        Process response from ChatGroq model
        IF response is valid answer to current question
            Continue to next question in sequence
        ELSE
            Prompt user to respond to current question again
    Update JSON medical form based on valid responses using Medical Form Management
Module
    Display updated JSON on UI
```

```
END WHILE
```

c) Question Answering (QA) Module

```
FUNCTION questionAnsweringModule (user_input)
    Send user_input to ChatGroq model
    IF response is related to the asked question
        RETURN response and indicate that the question has been answered
    ELSE
        RETURN prompt to answer the current question
END FUNCTION
```

d) Data Collection Module

```
FUNCTION dataCollectionModule(transcribed_text)
    IF transcribed_text is valid
        Append transcribed_text to collected_data
        Update JSON medical form using Medical Form Management Module
    ELSE
        Log error: "Invalid transcription data"
END FUNCTION
```

e) Audio Processing and Transcription Module

```
FUNCTION audioProcessingModule()
    Record audio in 5-second bursts
    Convert recorded audio to WAV format
    Send WAV file to Whisper model for transcription
    Detect language of transcription
    RETURN transcribed_text
END FUNCTION
```

f) Medical Form (JSON) Management Module

```
FUNCTION updateMedicalForm(transcribed_text)
    Load JSON medical form (if not already loaded)
    Parse transcribed_text to identify relevant fields
    Update specific fields in JSON medical form (e.g., personal details, medical history)
    Save updated JSON medical form
END FUNCTION
```

g) User Interface and Interaction Module

```
FUNCTION userInterfaceModule()
    Display JSON form fields on web portal (personal details, medical history, appointments,
etc.)
    Provide options for manual editing and updating of patient records
    Display structured data updates in real-time
END FUNCTION
```

h) Audio Recording and WAV File Generation Module

```
FUNCTION audioRecordingModule()
    Use pyaudio to start recording
    Save recording in WAV file format for processing
    RETURN WAV file path
END FUNCTION
```

i) Inference and JSON Update Module

```
FUNCTION inferenceAndJsonUpdateModule(collected_data)
    FOR each item in collected_data
        Check for relevance to existing JSON fields
        Update only relevant fields in JSON medical form
        Avoid overwriting unrelated information
    Save updated JSON medical form for future reference
END FUNCTION
```

j) Error Handling and Data Validation

```
FUNCTION errorHandlingAndValidationModule(json_data)
    Validate JSON structure and format
    IF json_data is invalid
        Log error and notify user
    ELSE
        Save validated json_data
END FUNCTION
```

k) Integrating Modules

```
FUNCTION main()
    Initialize system
    Display "Hello, how can I assist you today?"
    WHILE system is active
        // Step 1: Capture and Transcribe User's Voice Input
        user_audio = audioRecordingModule()
        transcribed_text = audioProcessingModule(user_audio)
        // Step 2: Process User's Response with QA Module
        qa_response = questionAnsweringModule(transcribed_text
        IF qa_response is valid
            Display qa_response
            dataCollectionModule(transcribed_text)
            updateMedicalForm(transcribed_text)
        ELSE
            Prompt user to respond correctly
        // Step 3: Display Updated Information
        userInterfaceModule()
        // Step 4: Handle "goodbye" termination
        IF transcribed_text contains "goodbye"
            Display "Thank you, goodbye!"
            Terminate interaction
    END WHILE
    // Save and validate final JSON medical form
    errorHandlingAndValidationModule(updated_JSON_form)
END FUNCTION
```

4.3 Testing Techniques

Testing is an indispensable aspect of software development, ensuring that the system functions as intended and meets the specified requirements. For the *Smart Assistance Appointment System*, which allows users to schedule appointments through voice interactions or a web interface, thorough testing is essential to validate its stability, accuracy, and usability. The following testing proposal outlines key test cases and methodologies used in evaluating the system. The various testing done during project development is described below:

a) Unit Testing

i. **Objective**: The primary goal of unit testing is to validate the correctness of individual components within the system, including the front-end UI, voice processing, and backend services.

ii. **Techniques**: Stubbing and mocking were utilized to isolate each module for testing purposes, ensuring each component functions independently before integrating them.

b) Integration Testing

i. **Objective**: Integration testing focuses on assessing the interaction between different modules, such as voice recognition, NLP, and appointment scheduling.

ii. **Techniques**: Performed integration testing by simulating user scenarios, ensuring the various subsystems (voice input, NLP model, appointment scheduling, MongoDB database, etc.) interact smoothly.

c) System Testing:

i. **Objective**: System testing evaluates the overall functionality, performance, and reliability of the entire system as a cohesive unit, simulating real-world scenarios.

ii. **Techniques**: End-to-end testing was conducted to ensure all components work together seamlessly. Real-time interactions and data processing were also tested to verify system responsiveness and accuracy.

Various examples of Testing Scenarios for the Smart Assistance Appointment System:

a) Scenario 1: Tested voice commands with varying intents (e.g., scheduling, rescheduling, and cancelling appointments) to evaluate the system's ability to accurately interpret and process different user requests.

b) Scenario 2: Verified the system’s response to ambiguous voice commands or partially heard commands, ensuring it prompts the user for clarification instead of proceeding with incorrect actions.

c) Scenario 3: Assessed the system’s cross-browser compatibility by accessing the web portal on different browsers (e.g., Chrome, Firefox, Safari) and devices (desktop, tablet, mobile) to ensure consistent functionality.

d) Scenario 4: Measured system performance under high user load by simulating multiple simultaneous appointment requests, assessing response times and resource usage to ensure scalability.

e) Scenario 5: Conducted security tests by attempting unauthorized access to patient data, verifying that only authenticated users with valid permissions can view or modify sensitive information.

4.4 Test Cases designed for the project work

Testing is an indispensable aspect of software development, ensuring that the system functions as intended and meets the specified requirements. For the *Smart Assistance Appointment System*, which allows users to schedule appointments through voice interactions or a web interface, thorough testing is essential to validate its stability, accuracy, and usability. The following testing proposal outlines key test cases and methodologies used in evaluating the system. The various testing done during project development is described below:

a) Unit Testing

i. Test Case 1: Validated the voice command parsing functionality. **Steps**:

- Provided a specific voice command as input (e.g., "Book my appointment with Dr. Smith").
- Processed the voice command through the DeepGram speech recognition module.
- Verified if the recognized command initiates the intended action in the system.
- **Result**: The command was correctly identified and the cancellation process was successfully triggered.

ii. Test Case 2: Verified individual UI components for accurate functionality. **Steps**:

- Provided input to the appointment form and date picker components.
- Tested with invalid input scenarios, such as selecting past dates.
- Checked that appropriate error messages prompt users to enter valid information.
- **Result**: Components handled user input accurately and displayed error messages for invalid entries.

b) Integration Testing: Ensured seamless integration between the voice recognition and NLP modules to book, modify, and cancel appointments. **Test Case**:

- Provided a voice command to schedule an appointment.

- Used the ChatGroq model to interpret the command and send a request to the backend.
- Verified that the appointment was accurately created and stored in MongoDB.
- Checked if the confirmation message adhered to scheduling rules and prompt user interaction.

c) System Testing: Evaluated the complete functionality of the system as an integrated unit. **Test Case**:

- Provided a series of voice commands to schedule, and cancel appointments.
- Ran the full system, including the frontend UI and backend services.
- Measured the system’s response time and accuracy across different scenarios.
- Compared system performance with expected benchmarks to ensure reliability.

Chapter 5. Results and Discussions

5.1 User Interface Representation

The Smart Assistance Appointment System enables the recording of audio, transcription to text, and subsequent updating of a structured medical form in JSON format. It leverages a range of audio processing libraries, natural language processing (NLP) models, and JSON handling, providing a streamlined workflow for medical professionals. Through integration with models like ChatGroq, DeepGram, and Whisper, the system achieves accurate transcription and efficient medical record updates.

a) Frontend Module: Vite + React: The frontend is designed to facilitate user interaction with the appointment system, offering both text-based and voice-command-driven interfaces. Built with Vite and React, and styled using Tailwind CSS, the system ensures an accessible, user-friendly, and responsive experience.

i. **Voice Input Integration**: Allows users to book, reschedule, or cancel appointments using voice commands, which are processed by backend ML models.

ii. **Appointment Management Dashboard**: Provides a central interface for patients to manage appointments, including scheduling, rescheduling, and cancellations.

iii. **Real-Time Status Updates**: Reflects any updates or changes made by users or staff immediately on the frontend.

iv. **Responsive Design**: Ensures compatibility with desktops, tablets, and mobile devices for broad accessibility.

b) Backend Module: Express.js and Node.js: The backend of the system is divided into two core components: Express.js for managing web operations.

i. **Appointment APIs**: RESTful APIs support appointment management actions, including booking, rescheduling, and cancellation.

ii. **Authentication and Authorization**: Token-based security to protect patient data.

iii. **Error Handling**: Comprehensive error management ensures smooth communication between frontend and backend.

c) Database Module: MongoDB: MongoDB serves as the database, storing patient data, appointment details, and voice interaction logs. Its NoSQL structure is well-suited for the dynamic data relationships involved in patient interactions.

i. **Patient Records Management**: Stores patient details, medical history, and appointment preferences.

ii. **Appointment Scheduling Data**: Tracks appointment information such as dates, times, and statuses.

iii. **Voice Interaction Logs**: Records patient voice interactions, enhancing system analysis and improvement.

d) Audio Processing and Transcription Module

This module manages audio input, recording, and transcription. Using **pyaudio** and **Whisper** model, it processes voice input into accurate text transcriptions that facilitate data entry.

i. **Audio Capture**: Records short bursts of audio (5-second intervals) and saves as WAV files.

ii. **Language Detection**: Identifies input language, ensuring appropriate transcription handling.

e) Question Answering (QA) Module: This module leverages the ChatGroq, DeepGram, and Whisper models to handle the flow of questions and answers.

i. **Chat Loop**: Uses the ChatGroq model to guide users through questions, advancing only when valid responses are given.

ii. **TTS with DeepGram**: Converts text responses to audio and plays them via ffplay, enhancing interactivity.

iii. **Data Collection**: Sends collected responses to Gemini, which fills in a JSON file with structured patient information.

f) Medical Form (JSON) Management Module: This module manages a medical form in JSON format, which is updated dynamically as new information from the transcription is received.

i. **JSON Structure**: Contains fields for personal details, contact information, medical history, and more.

ii. **Field Updating**: Integrates new data into nested dictionaries without overwriting unrelated information, maintaining data integrity.

g) User Interface and Interaction Module: The system includes a web-based portal for easy navigation and management of patient records.

i. **Appointment Portal**: Allows users to book, view, or manage appointments via a question-driven interface.

ii. **Voice Command Interface**: Provides voice interaction options, enhancing accessibility and convenience for users.

h) Inference and JSON Update Module: In this module, extracted transcription data is merged with the existing JSON medical form, updating relevant fields only. **Data Merge and Save**: It Ensures all changes are correctly integrated into the JSON file, providing an up-to-date record for each patient.

i) Voice Assistant Module: This module enables voice-based interaction, employing advanced NLP for accurate recognition and task execution.

i. **Speech-to-Text Processing**: Converts voice inputs into structured data.

ii. **Command Recognition**: Identifies commands to schedule, reschedule, or cancel appointments.

iii. **Real-Time Feedback**: Provides voice-based feedback for confirmation or clarifications as needed.

j) Error Handling and Data Validation Module: This module ensures JSON output format and structure are correct, logging errors and notifying users of issues to maintain reliability.

i. **Data Validation**: Verifies JSON data structure before saving.

ii. **Error Logging**: Tracks and logs any errors encountered during transcription or data saving, ensuring system stability.

k) Integration with Hospital Management System: The appointment system integrates with the hospital's existing HMS, ensuring synchronized data and up-to-date scheduling.

i. **APIs for Data Sync**: Synchronizes appointment data with the hospital's main database in real time.

ii. **Doctor and Department Availability**: Retrieves updated scheduling information to prevent conflicts.

iii. **Notification System**: Sends real-time notifications to users and staff regarding appointment changes.

5.2 Snapshots of system with brief detail of each and discussion.

This section presents snapshots of the system interface along with brief explanations of each component, providing a visual overview and discussing how each part contributes to the system's functionality.

5.2.1 Frontend

The frontend interface enables users to interact with the Smart Assistance Appointment System, providing an intuitive layout for appointment booking, voice interactions, and displaying confirmations or notifications to enhance the user experience. **Fig 5.1 and 5.2** depict key functionalities of the Smart Assistance Appointment System's web portal. Figure 5.1 shows the main page, which serves as the entry point for users with a welcoming interface that simplifies access to essential functions like booking, viewing appointments. This page is designed to provide an intuitive navigation experience, making it easy for users to interact with the system. Fig 5.2 highlights the filtering feature of the portal, allowing users to efficiently sort and view appointments based on specific criteria. This functionality helps streamline the appointment management process, enhancing the overall user experience by allowing quick access to relevant information.

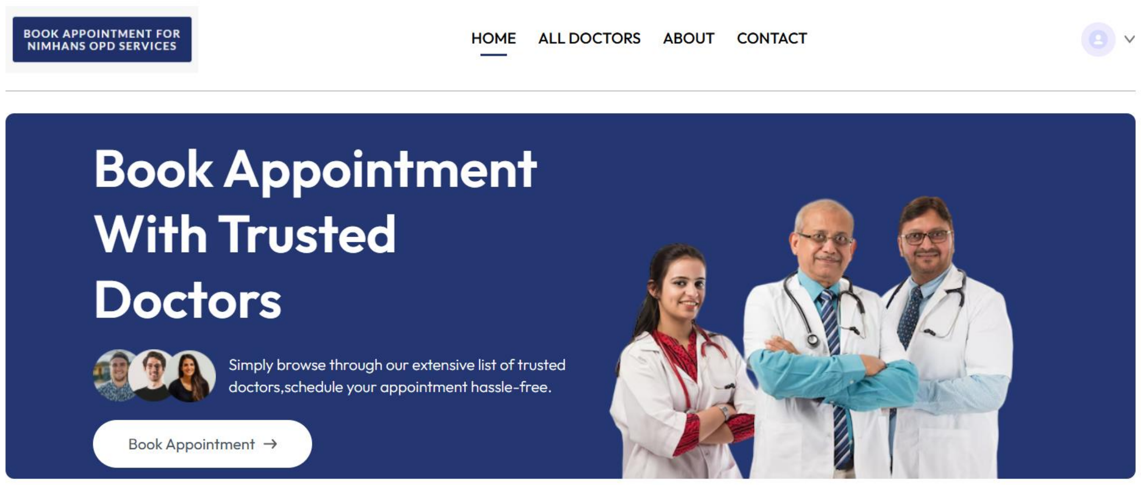

BOOK APPOINTMENT FOR NIMHANS OPD SERVICES
HOME
ALL DOCTORS
ABOUT
CONTACT
Book Appointment With Trusted Doctors
Simply browse through our extensive list of trusted doctors,schedule your appointment hassle-free.
Book Appointment →

Fig 5.1: Main Page of web Portal

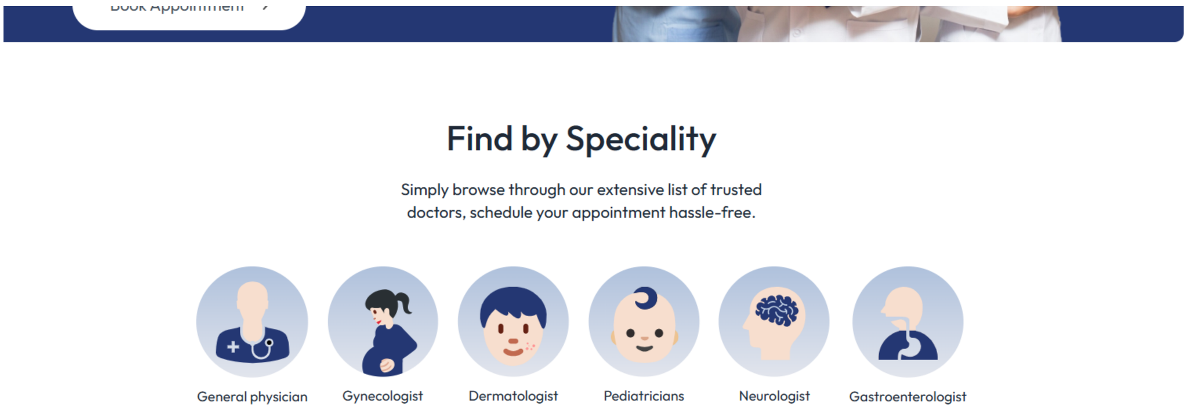


Fig 5.2: Filtering property of Web Portal

Fig 5.3 and 5.4 illustrate additional core components of the Smart Assistance Appointment System's web portal. Figure 5.3 displays a comprehensive list of all available doctors, enabling users to browse through doctors based on specialty, availability, and other relevant details. This organized list simplifies the process of selecting a suitable healthcare provider, making the appointment booking experience more efficient. Fig 5.4 shows the "Create Account" page, where new users can register by entering their personal details. This page is designed to be user-friendly, guiding new users through the account creation process to ensure smooth access to all portal features, including appointment scheduling and personalized notifications.

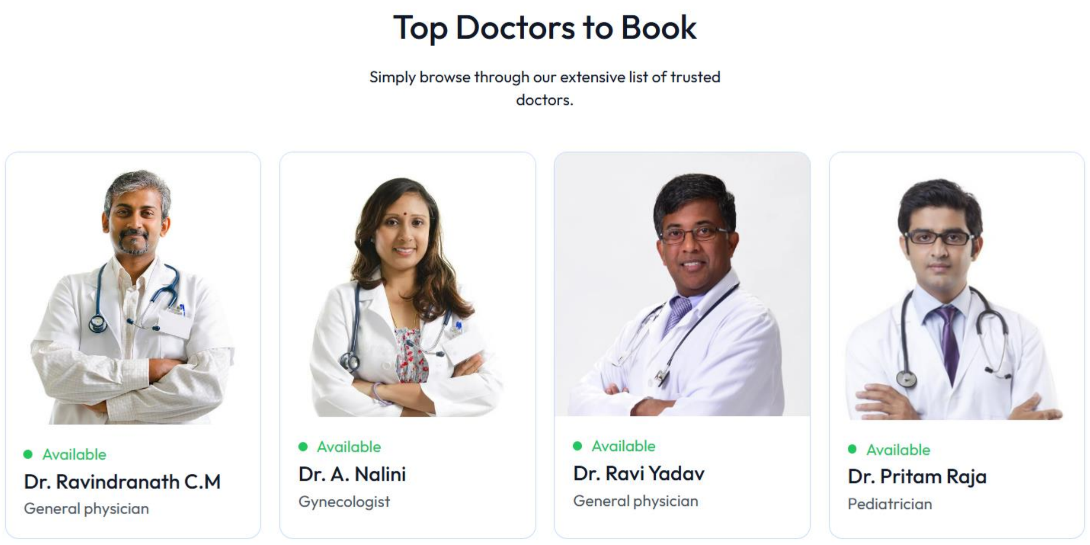


Fig 5.3: List of all the available doctors

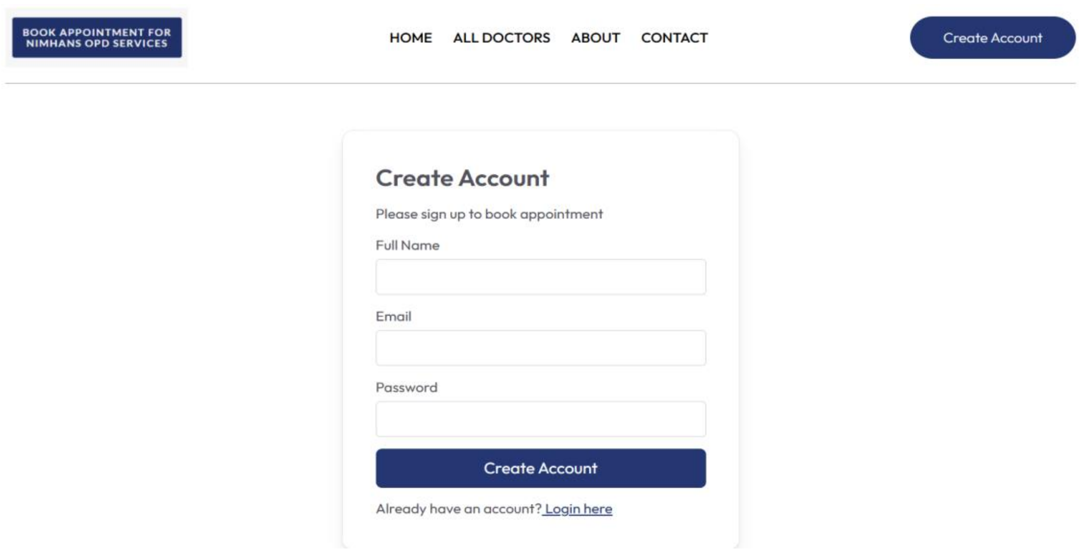
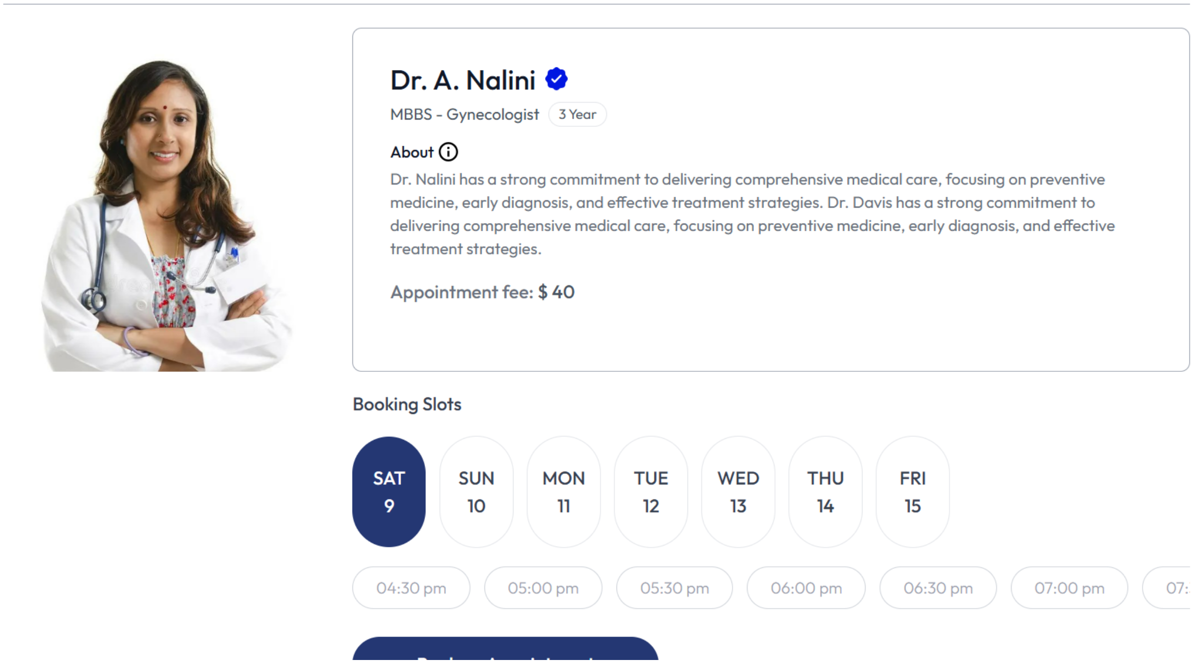

BOOK APPOINTMENT FOR NIMHANS OPD SERVICES
HOME
ALL DOCTORS
ABOUT
CONTACT
Create Account
Create Account
Please sign up to book appointment
Full Name
Email
Password
Create Account
Already have an account? Login here

Fig 5.4: Create Account page of web portal

Fig 5.5 and 5.6 showcase critical features in the Smart Assistance Appointment System's web portal that support informed and convenient booking. Fig 5.5 presents the Doctor Description Page, where users can view detailed profiles of individual doctors, including their specialties, qualifications, experience, and patient reviews. Fig 5.6 shows the Slot Booking Page, which enables users to select available appointment times based on their chosen doctor's schedule. This page provides a straightforward booking experience, displaying open slots clearly so users can quickly secure a convenient appointment time.



Dr. A. Nalini

MBBS - Gynecologist 3 Year

About

Dr. Nalini has a strong commitment to delivering comprehensive medical care, focusing on preventive medicine, early diagnosis, and effective treatment strategies. Dr. Davis has a strong commitment to delivering comprehensive medical care, focusing on preventive medicine, early diagnosis, and effective treatment strategies.

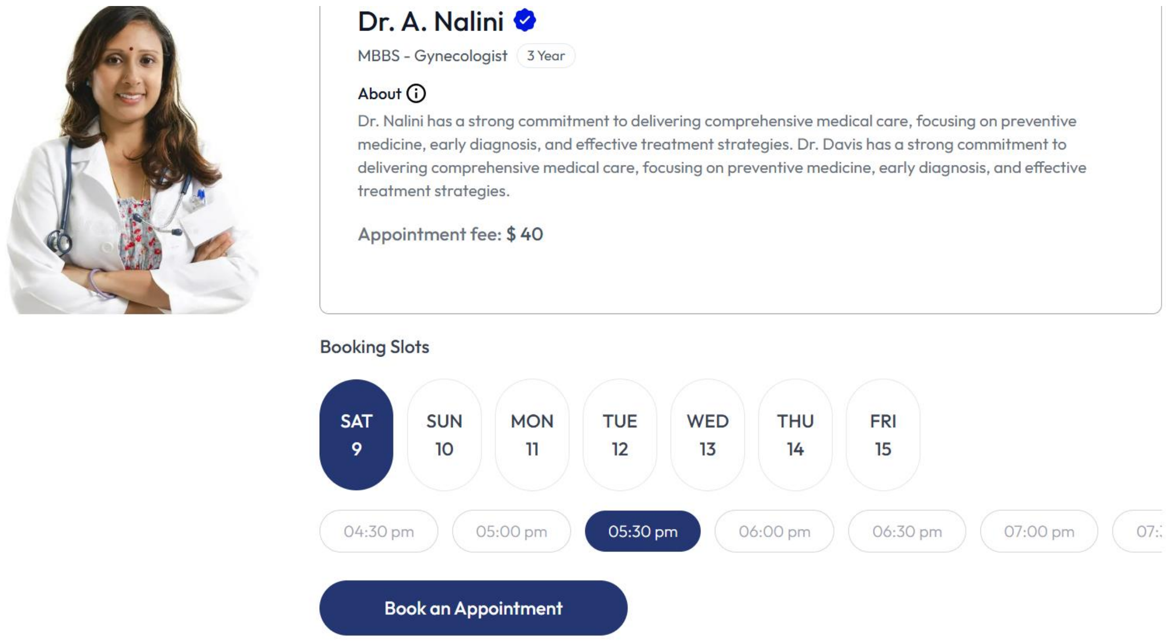
Appointment fee: $ 40

Booking Slots

SAT 9 | SUN 10 | MON 11 | TUE 12 | WED 13 | THU 14 | FRI 15

04:30 pm | 05:00 pm | 05:30 pm | 06:00 pm | 06:30 pm | 07:00 pm | 07:

Fig 5.5: Doctor description Page of web Portal



Dr. A. Nalini

MBBS - Gynecologist 3 Year

About

Dr. Nalini has a strong commitment to delivering comprehensive medical care, focusing on preventive medicine, early diagnosis, and effective treatment strategies. Dr. Davis has a strong commitment to delivering comprehensive medical care, focusing on preventive medicine, early diagnosis, and effective treatment strategies.

Appointment fee: $ 40

Booking Slots

SAT 9 | SUN 10 | MON 11 | TUE 12 | WED 13 | THU 14 | FRI 15

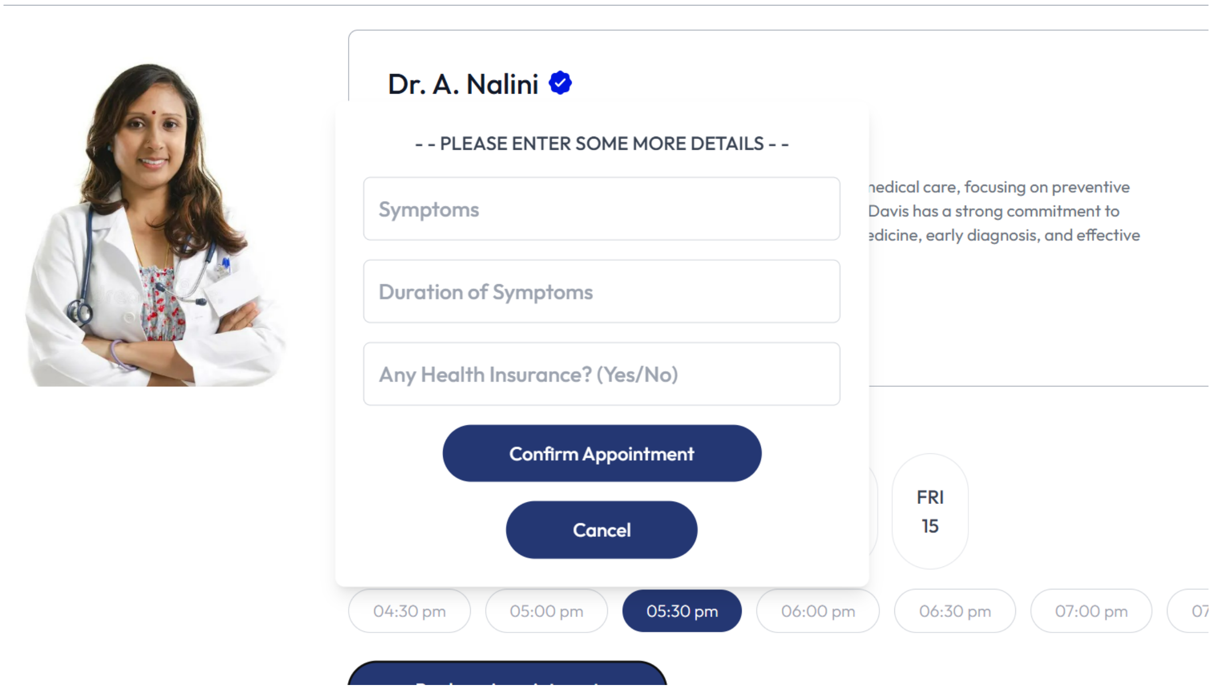
04:30 pm | 05:00 pm | 05:30 pm | 06:00 pm | 06:30 pm | 07:00 pm | 07:

Fig 5.6: Slot Booking Page of web Portal

Fig 5.7 and 5.8 depict essential steps in the appointment booking process within the Smart

Assistance Appointment System's web portal. Fig 5.7 shows the Detail Filling Page for appointments, where users enter specific information required for their appointment, such as reason for visit, preferred consultation method (e.g., in-person or online), and any other relevant details. This page ensures that all necessary information is collected to facilitate a smooth and personalized appointment experience.

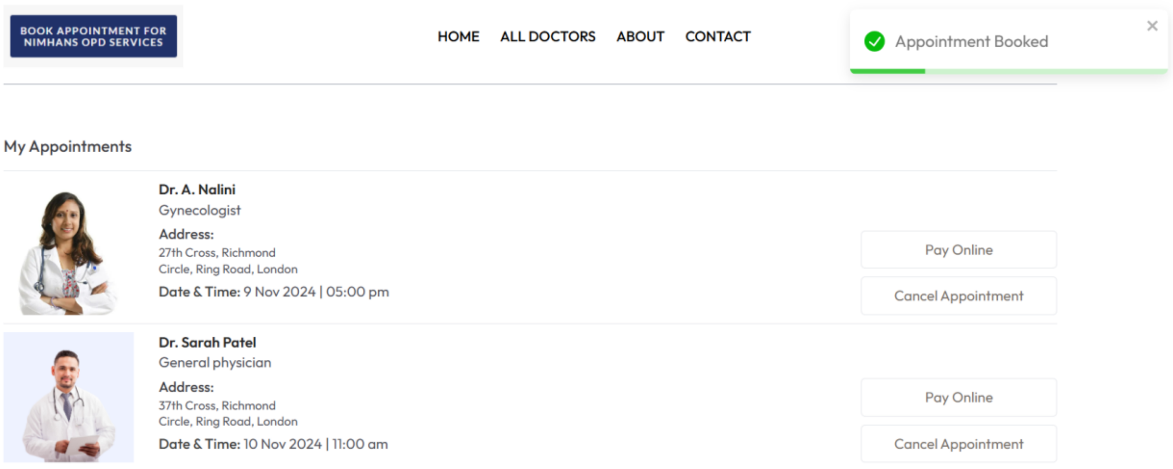
Fig 5.8 presents the Booked Appointments Page, which displays a consolidated list of all confirmed appointments for the user. This page allows users to view, manage, and, if needed, or cancel their upcoming appointments, providing a clear overview of their scheduled healthcare visits.



nedical care, focusing on preventive Davis has a strong commitment to edicine, early diagnosis, and effective

Fig 5.7: Detail filling for appointment

Fig 5.8 presents the Booked Appointments Page, which displays a consolidated list of all confirmed appointments for the user. This page allows users to view, manage, and, if needed, or cancel their upcoming appointments, providing a clear overview of their scheduled healthcare visits.


BOOK APPOINTMENT FOR NIMHANS OPD SERVICES

HOME ALL DOCTORS ABOUT CONTACT

Appointment Booked

My Appointments

Dr. A. Nalini
Gynecologist
Address:
27th Cross, Richmond
Circle, Ring Road, London
Date & Time: 9 Nov 2024 | 05:00 pm

Pay Online
Cancel Appointment

Dr. Sarah Patel
General physician
Address:
37th Cross, Richmond
Circle, Ring Road, London
Date & Time: 10 Nov 2024 | 11:00 am

Pay Online
Cancel Appointment

Fig 5.8: Page representing all the booked appointments

Fig 5.9 and 5.10 illustrate the administrative access features of the Smart Assistance Appointment System's web portal, essential for managing the backend and operational aspects of the system. Fig 5.9 shows the Admin Login Page, where authorized personnel can securely log into the system using their credentials. This page ensures that only verified administrators can access sensitive data and administrative functions. Fig 5.10 displays the Admin Dashboard Page, providing a comprehensive overview of system metrics, appointment statistics, and user management tools. Through this dashboard, administrators can oversee daily operations, manage doctor and patient profiles, monitor appointment scheduling, all of which support effective system oversight and maintenance.

Hence, Fig 5.9 and 5.10 present the admin functionality of the Smart Assistance Appointment System, including a secure login and a dashboard with tools for managing profiles, appointments, and system metrics. Real-time notifications and customizable settings allow administrators to promptly address updates and adapt the platform to different healthcare needs. These features help maintain smooth operations and enhance the user experience for patients and providers alike.

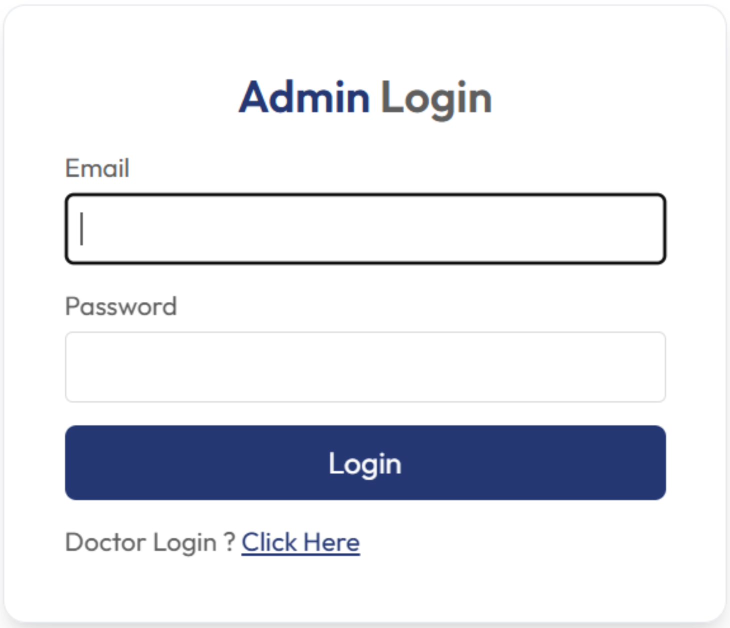
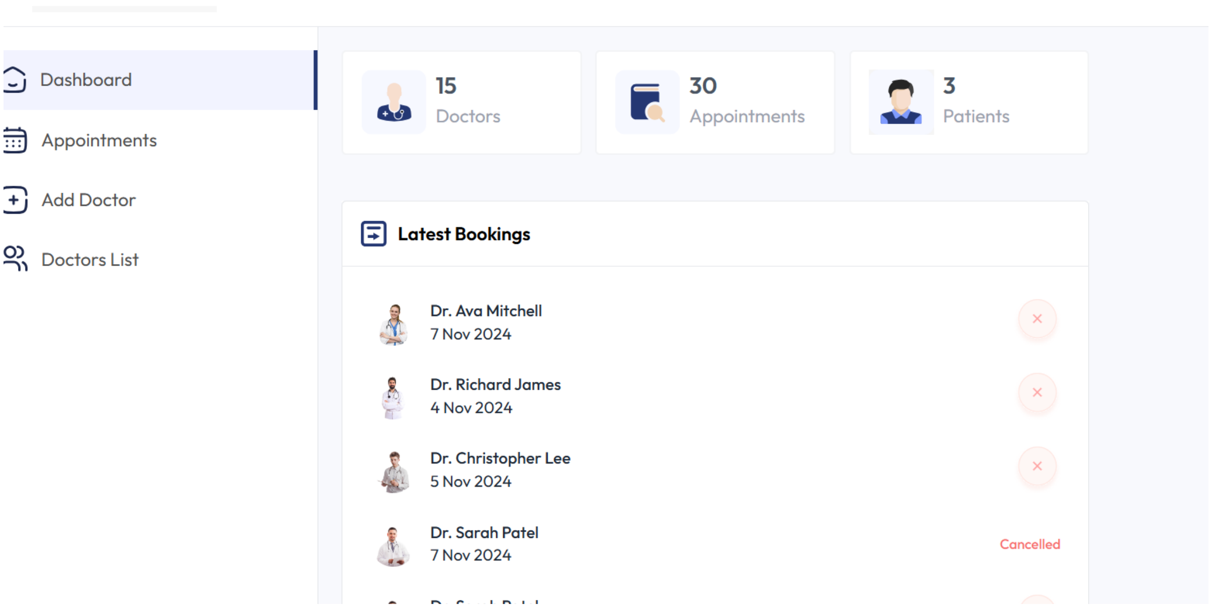


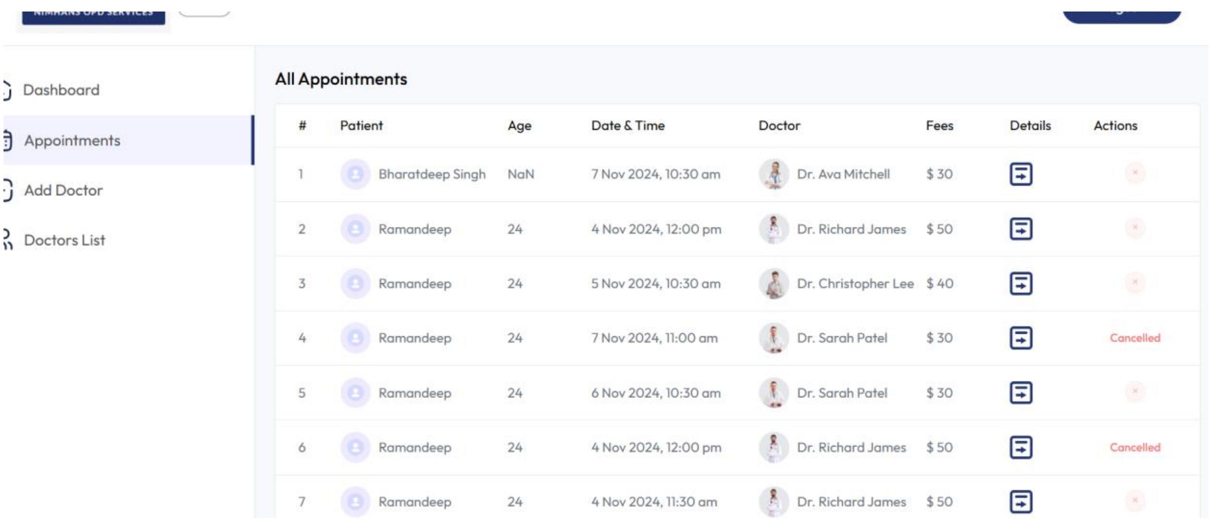
Fig 5.9: Admin Login Page



Dr. Ava Mitchell 7 Nov 2024	×
Dr. Richard James 4 Nov 2024	×
Dr. Christopher Lee 5 Nov 2024	×
Dr. Sarah Patel 7 Nov 2024	Cancelled

Fig 5.10: Admin dashboard page

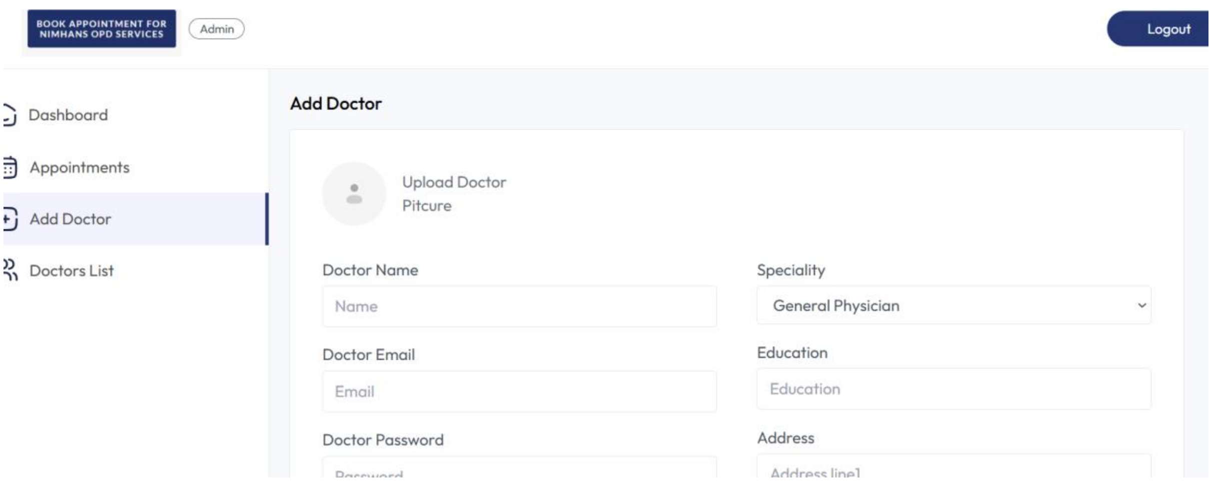
Fig 5.11 and 5.12 illustrate key administrative tools within the Smart Assistance Appointment System. Fig 5.11 shows the Admin Page displaying a comprehensive list of all scheduled appointments, allowing administrators to review, update, or manage appointment details as needed. This functionality helps ensure efficient tracking and handling of patient schedules. Fig 5.12 displays the Admin Page for adding a doctor, where administrators can input detailed information about new healthcare providers, such as specialty, qualifications, and availability. These tools streamline the onboarding process for doctors and facilitate efficient appointment management, ultimately enhancing overall system functionality.



All Appointments

#	Patient	Age	Date & Time	Doctor	Fees	Details	Actions
1	Bharatdeep Singh	NaN	7 Nov 2024, 10:30 am	Dr. Ava Mitchell	$ 30	▣	×
2	Ramandeep	24	4 Nov 2024, 12:00 pm	Dr. Richard James	$ 50	▣	×
3	Ramandeep	24	5 Nov 2024, 10:30 am	Dr. Christopher Lee	$ 40	▣	×
4	Ramandeep	24	7 Nov 2024, 11:00 am	Dr. Sarah Patel	$ 30	▣	Cancelled
5	Ramandeep	24	6 Nov 2024, 10:30 am	Dr. Sarah Patel	$ 30	▣	×
6	Ramandeep	24	4 Nov 2024, 12:00 pm	Dr. Richard James	$ 50	▣	Cancelled
7	Ramandeep	24	4 Nov 2024, 11:30 am	Dr. Richard James	$ 50	▣	×

Fig 5.11: Admin page with list of all appointments



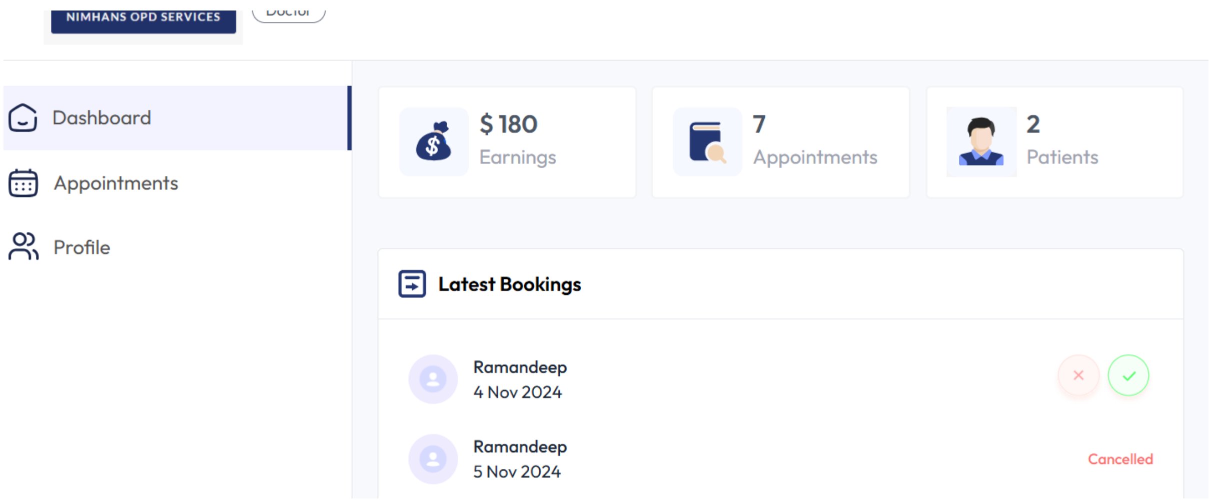
Add Doctor

Upload Doctor Pitcure

Doctor Name: Name	Speciality: General Physician
Doctor Email: Email	Education: Education
Doctor Password: Password	Address: Address line1

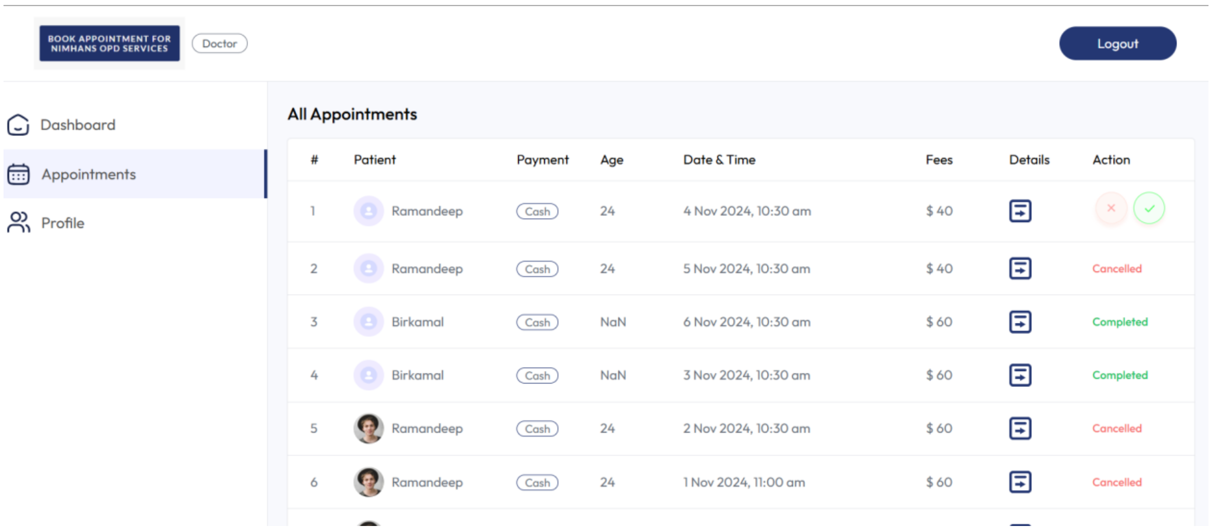
Fig 5.12: Admin page for adding a doctor

Fig 5.13 and 5.14 showcase the features accessible to doctors within the Smart Assistance Appointment System. Fig 5.13 presents the Doctor Dashboard, which provides an overview of key information such as upcoming appointments, recent patient interactions, and any important notifications. This dashboard enables doctors to stay organized and manage their schedules effectively. Fig 5.14 displays the All Appointments List, where doctors can view a consolidated list of appointments for all doctors within the system, if authorized. This list helps healthcare providers monitor schedules, coordinate with other team members, and ensure a smooth flow of patient appointments across the practice.



Ramandeep 4 Nov 2024	✕ ✓
Ramandeep 5 Nov 2024	Cancelled

Fig 5.13: Dashboard of doctor login



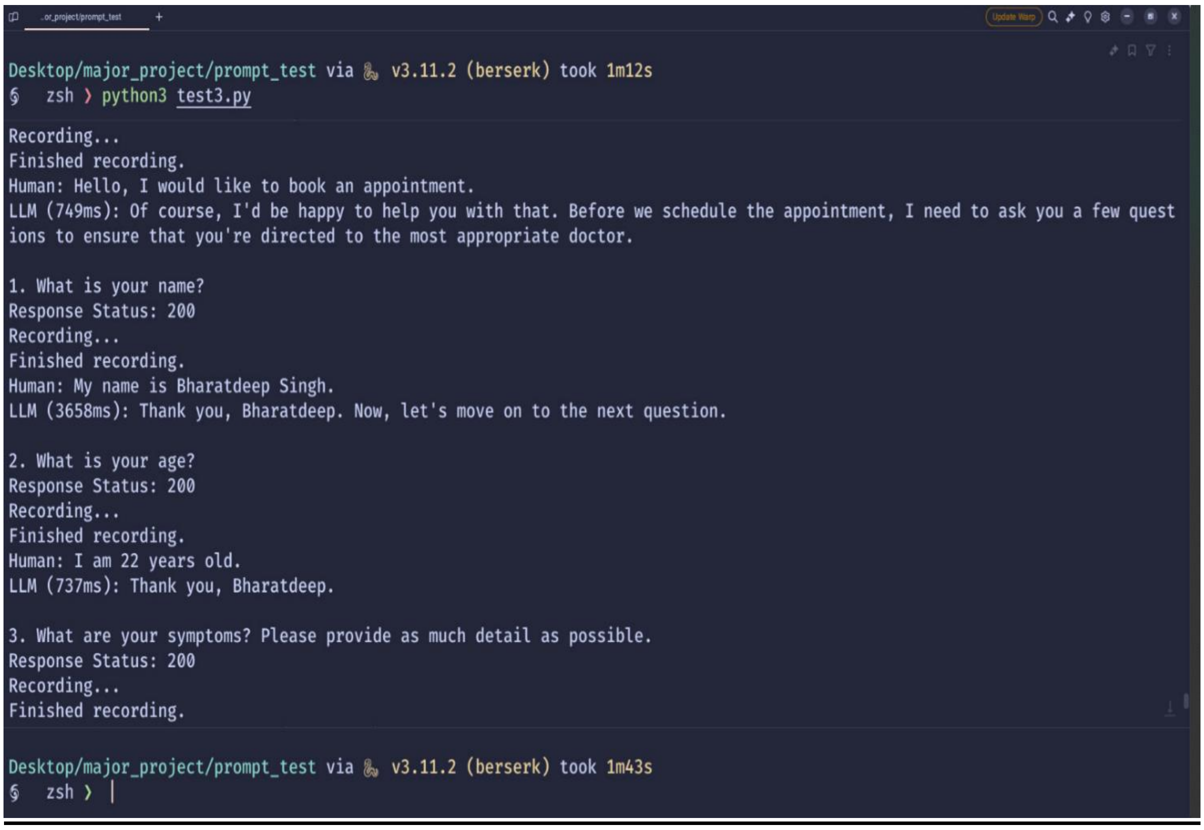
All Appointments

#	Patient	Payment	Age	Date & Time	Fees	Details	Action
1	Ramandeep	Cash	24	4 Nov 2024, 10:30 am	$ 40		✕ ✓
2	Ramandeep	Cash	24	5 Nov 2024, 10:30 am	$ 40		Cancelled
3	Birkamal	Cash	NaN	6 Nov 2024, 10:30 am	$ 60		Completed
4	Birkamal	Cash	NaN	3 Nov 2024, 10:30 am	$ 60		Completed
5	Ramandeep	Cash	24	2 Nov 2024, 10:30 am	$ 60		Cancelled
6	Ramandeep	Cash	24	1 Nov 2024, 11:00 am	$ 60		Cancelled

Fig 5.14: All appointment list of all doctors in Admin Panel

5.2.2 Backend

Fig 5.15 and Fig 5.16 basically represent the conversation of human being with our appointment system.

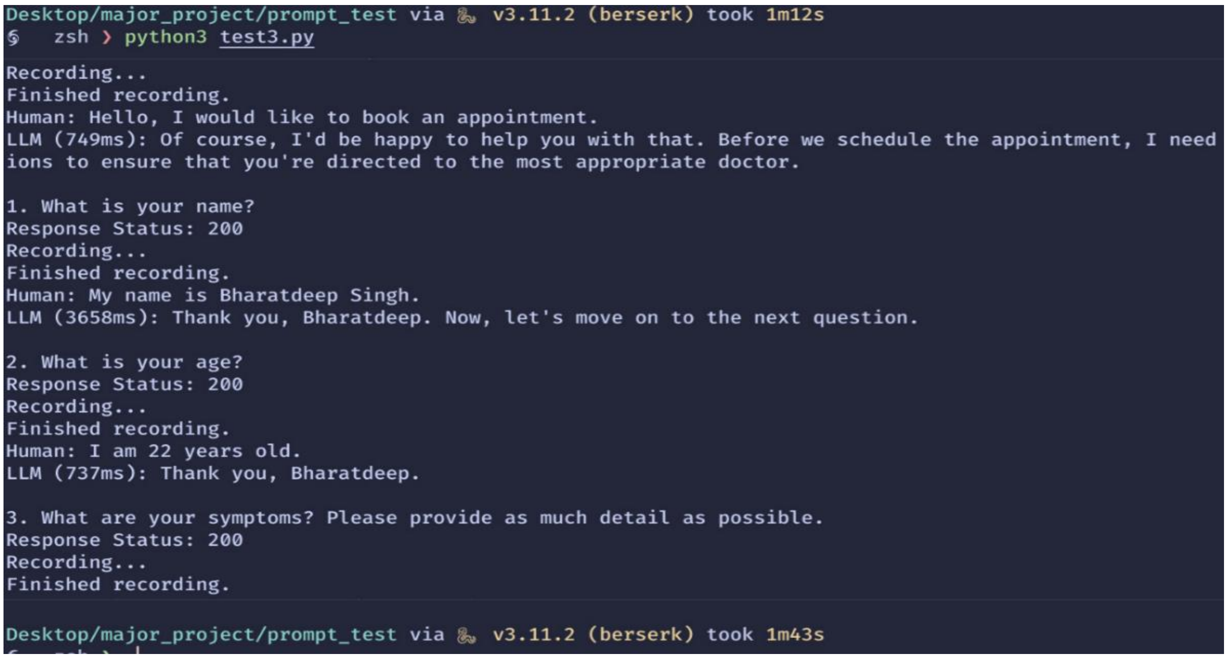


```
Desktop/major_project/prompt_test via 🐍 v3.11.2 (berserk) took 1m12s
§   zsh ❯ python3 test3.py

Recording...
Finished recording.
Human: Hello, I would like to book an appointment.
LLM (749ms): Of course, I'd be happy to help you with that. Before we schedule the appointment, I need to ask you a few quest
ions to ensure that you're directed to the most appropriate doctor.

1. What is your name?
Response Status: 200
Recording...
Finished recording.
Human: My name is Bharatdeep Singh.
LLM (3658ms): Thank you, Bharatdeep. Now, let's move on to the next question.

2. What is your age?
Response Status: 200
Recording...
Finished recording.
Human: I am 22 years old.
LLM (737ms): Thank you, Bharatdeep.

3. What are your symptoms? Please provide as much detail as possible.
Response Status: 200
Recording...
Finished recording.

Desktop/major_project/prompt_test via 🐍 v3.11.2 (berserk) took 1m43s
§   zsh ❯ |
```

Fig 5.15: Conversation with the System

The system captures voice inputs, detects language probabilities, and responds promptly with minimal latency, indicating efficient handling of natural language conversations and backend processing. The conversation showcases the system asking for basic patient details, such as name and age, while maintaining a structured dialogue flow. Each response from the language model includes timestamps, indicating the processing time and response efficiency, demonstrating smooth real-time communication and data handling.

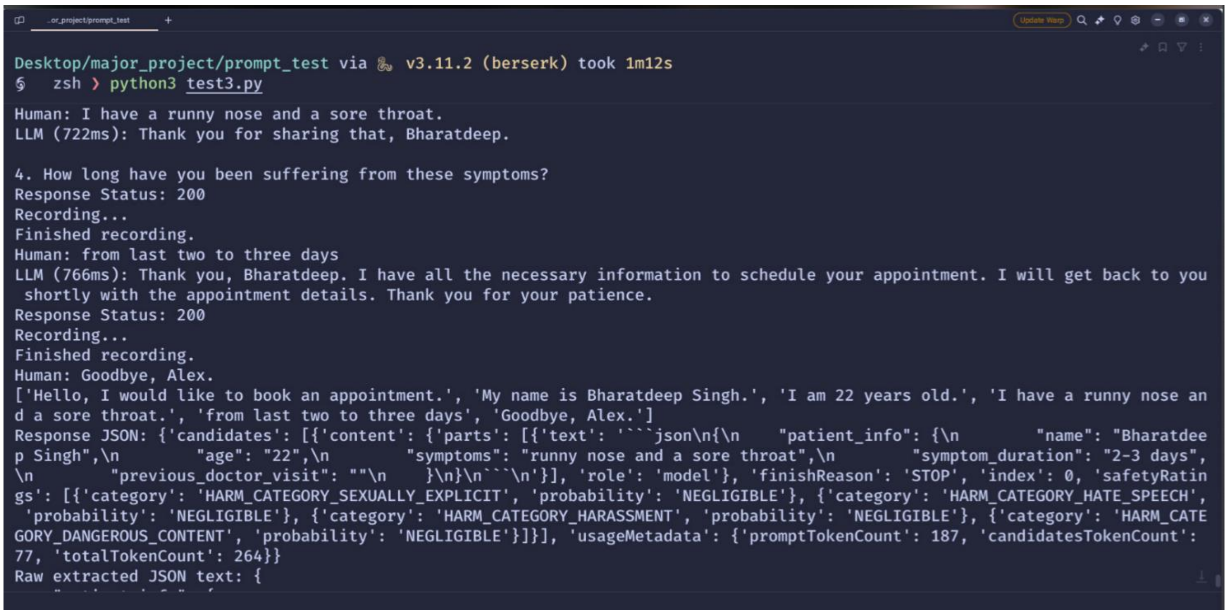
The system captures voice inputs, detects language probabilities, and responds promptly with minimal latency, indicating efficient handling of natural language conversations and backend processing. The conversation showcases the system asking for basic patient details, such as name and age, while maintaining a structured dialogue flow. Each response from the language model includes timestamps, indicating the processing time and response efficiency, demonstrating smooth real-time communication and data handling.



```
Desktop/major_project/prompt_test via 🐍 v3.11.2 (berserk) took 1m12s
$   zsh ❯ python3 test3.py
Recording...
Finished recording.
Human: Hello, I would like to book an appointment.
LLM (749ms): Of course, I'd be happy to help you with that. Before we schedule the appointment, I need
ions to ensure that you're directed to the most appropriate doctor.

1. What is your name?
Response Status: 200
Recording...
Finished recording.
Human: My name is Bharatdeep Singh.
LLM (3658ms): Thank you, Bharatdeep. Now, let's move on to the next question.

2. What is your age?
Response Status: 200
Recording...
Finished recording.
Human: I am 22 years old.
LLM (737ms): Thank you, Bharatdeep.

3. What are your symptoms? Please provide as much detail as possible.
Response Status: 200
Recording...
Finished recording.

Desktop/major_project/prompt_test via 🐍 v3.11.2 (berserk) took 1m43s
```

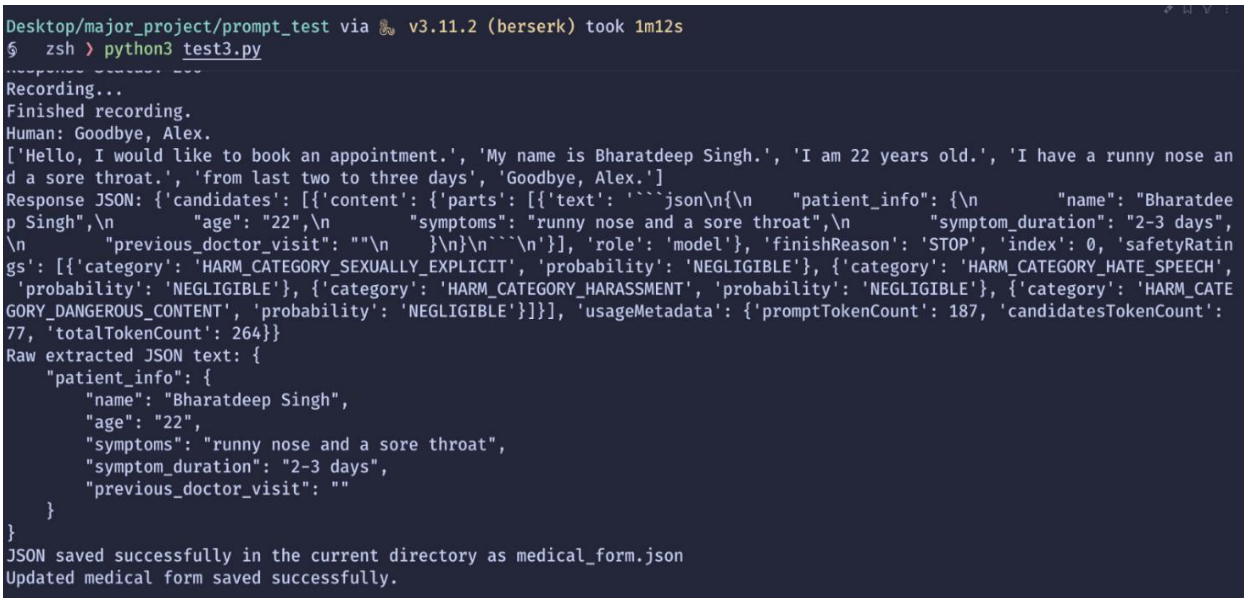
Fig 5.16: Conversation with the System2

The conversation highlights the system's structured dialogue flow as it gathers basic patient details, such as name and age, through a clear, guided interaction. Each response generated by the language model is timestamped, which not only tracks processing time but also illustrates response efficiency and data handling in near real-time. This ensures a smooth conversational experience for users, providing immediate feedback and reducing wait times. Fig 5.17 and 5.18 demonstrate how patient information is captured and formatted into JSON using Gemini, an essential step for securely storing structured data. This JSON format is used to store responses consistently, facilitating easy retrieval and integration with other healthcare data systems, thus enhancing the system's robustness and interoperability across platforms. This ensures a smooth conversational experience for users, providing immediate feedback and reducing wait times. Fig 5.17 and 5.18 demonstrate how patient information is captured and formatted into JSON using Gemini, an essential step for securely storing structured data. This JSON format is used to store responses consistently, facilitating easy retrieval and integration with other healthcare data systems, thus enhancing the system's robustness and interoperability across platforms.



```
Desktop/major_project/prompt_test via 🐍 v3.11.2 (berserk) took 1m12s
§  zsh ❯ python3 test3.py
Human: I have a runny nose and a sore throat.
LLM (722ms): Thank you for sharing that, Bharatdeep.

4. How long have you been suffering from these symptoms?
Response Status: 200
Recording...
Finished recording.
Human: from last two to three days
LLM (766ms): Thank you, Bharatdeep. I have all the necessary information to schedule your appointment. I will get back to you
 shortly with the appointment details. Thank you for your patience.
Response Status: 200
Recording...
Finished recording.
Human: Goodbye, Alex.
['Hello, I would like to book an appointment.', 'My name is Bharatdeep Singh.', 'I am 22 years old.', 'I have a runny nose an
d a sore throat.', 'from last two to three days', 'Goodbye, Alex.']
Response JSON: {'candidates': [{'content': {'parts': [{'text': '```json\n{\n    "patient_info": {\n        "name": "Bharatdee
p Singh",\n        "age": "22",\n        "symptoms": "runny nose and a sore throat",\n        "symptom_duration": "2-3 days",
\n        "previous_doctor_visit": ""\n    }\n}\n```\n'}], 'role': 'model'}, 'finishReason': 'STOP', 'index': 0, 'safetyRatin
gs': [{'category': 'HARM_CATEGORY_SEXUALLY_EXPLICIT', 'probability': 'NEGLIGIBLE'}, {'category': 'HARM_CATEGORY_HATE_SPEECH',
 'probability': 'NEGLIGIBLE'}, {'category': 'HARM_CATEGORY_HARASSMENT', 'probability': 'NEGLIGIBLE'}, {'category': 'HARM_CATE
GORY_DANGEROUS_CONTENT', 'probability': 'NEGLIGIBLE'}]}], 'usageMetadata': {'promptTokenCount': 187, 'candidatesTokenCount':
77, 'totalTokenCount': 264}}
Raw extracted JSON text: {
```

Fig 5.17: Displaying JSON



```
Desktop/major_project/prompt_test via 🐍 v3.11.2 (berserk) took 1m12s
§  zsh ❯ python3 test3.py
Recording...
Finished recording.
Human: Goodbye, Alex.
['Hello, I would like to book an appointment.', 'My name is Bharatdeep Singh.', 'I am 22 years old.', 'I have a runny nose an
d a sore throat.', 'from last two to three days', 'Goodbye, Alex.']
Response JSON: {'candidates': [{'content': {'parts': [{'text': '```json\n{\n    "patient_info": {\n        "name": "Bharatdee
p Singh",\n        "age": "22",\n        "symptoms": "runny nose and a sore throat",\n        "symptom_duration": "2-3 days",
\n        "previous_doctor_visit": ""\n    }\n}\n```\n'}], 'role': 'model'}, 'finishReason': 'STOP', 'index': 0, 'safetyRatin
gs': [{'category': 'HARM_CATEGORY_SEXUALLY_EXPLICIT', 'probability': 'NEGLIGIBLE'}, {'category': 'HARM_CATEGORY_HATE_SPEECH',
 'probability': 'NEGLIGIBLE'}, {'category': 'HARM_CATEGORY_HARASSMENT', 'probability': 'NEGLIGIBLE'}, {'category': 'HARM_CATE
GORY_DANGEROUS_CONTENT', 'probability': 'NEGLIGIBLE'}]}], 'usageMetadata': {'promptTokenCount': 187, 'candidatesTokenCount':
77, 'totalTokenCount': 264}}
Raw extracted JSON text: {
    "patient_info": {
        "name": "Bharatdeep Singh",
        "age": "22",
        "symptoms": "runny nose and a sore throat",
        "symptom_duration": "2-3 days",
        "previous_doctor_visit": ""
    }
}
JSON saved successfully in the current directory as medical_form.json
Updated medical form saved successfully.
```

Fig 5.18: Forming JSON

5.3 Back Ends Representation

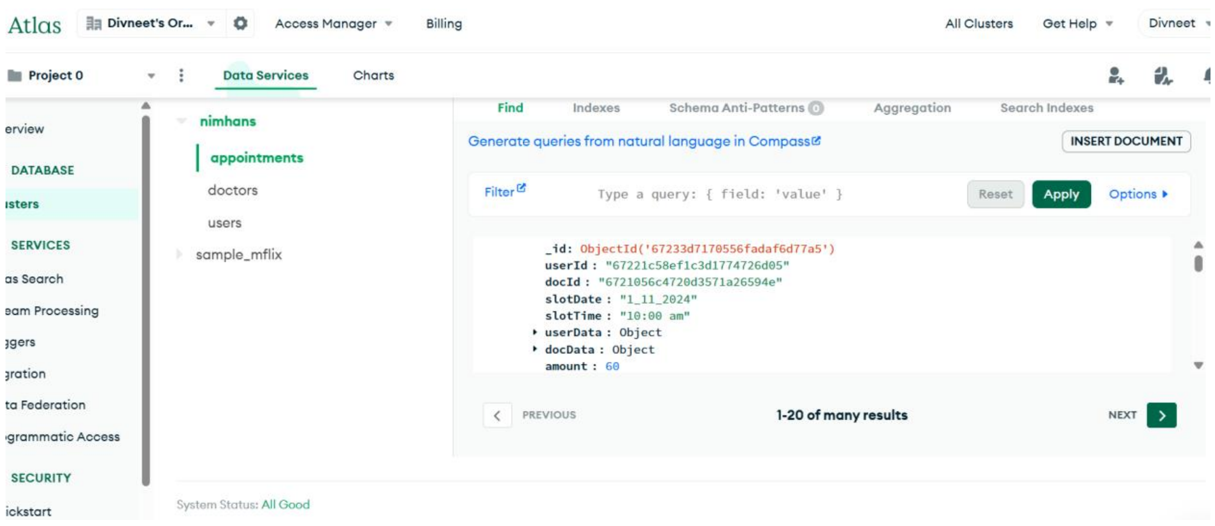
This section provides an overview of the back-end structure, focusing on how the system's data is organized, stored, and managed to support smooth operations and efficient data handling**.**

5.3.1 Snapshots of Database Tables with brief description

This subsection presents snapshots of the key database tables used in the system, along with brief descriptions of each table's purpose and structure. These tables store essential data, such as patient details, appointment information, and doctor profiles, enabling seamless data retrieval and integration within the Smart Assistance Appointment System.

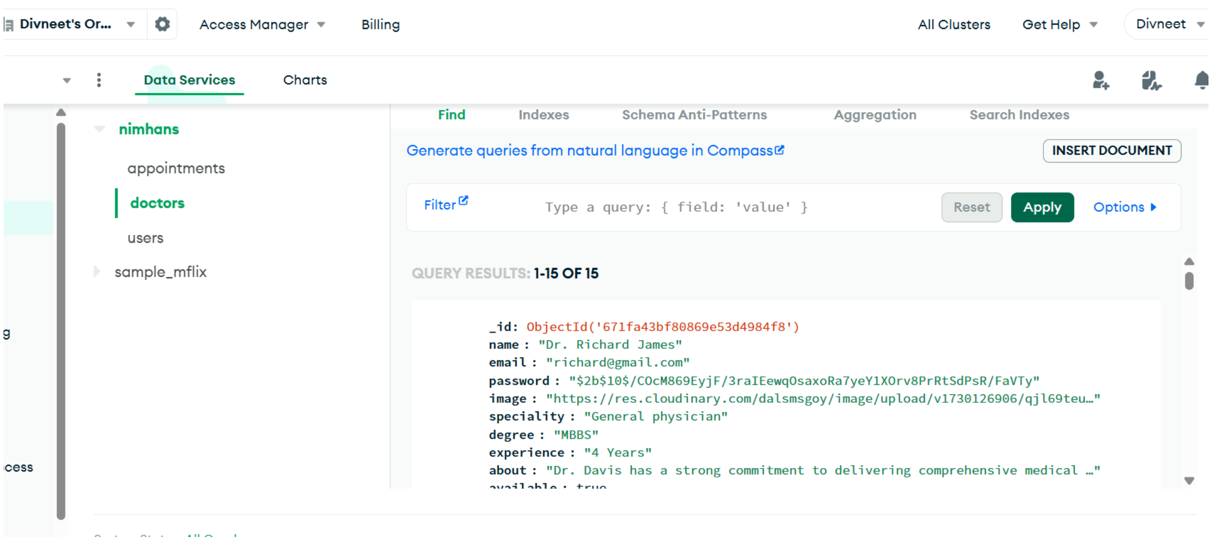
Fig 5.19 and 5.20 illustrate the core database tables that support the Smart Assistance Appointment System. Fig 5.19 shows the **Appointments Table**, which stores detailed information about each scheduled appointment, including patient ID, doctor ID, appointment date and time, and status. This table is crucial for tracking and managing all appointments within the system efficiently.

Fig 5.20 displays the **Doctors Table**, which contains essential data about each healthcare provider, such as doctor ID, name, specialty, contact information, and availability. This table enables the system to organize and retrieve doctor information seamlessly, ensuring that patients can view accurate provider details when booking appointments. Together, these tables form the backbone of the system's data management, enabling reliable and organized handling of patient and provider information.



```
_id: ObjectId('67233d7170556fadaf6d77a5')
userId : "67221c58ef1c3d1774726d05"
docId : "6721056c4720d3571a26594e"
slotDate : "1_11_2024"
slotTime : "10:00 am"
▸ userData : Object
▸ docData : Object
amount : 60
```

Fig 5.19: Table for appointments

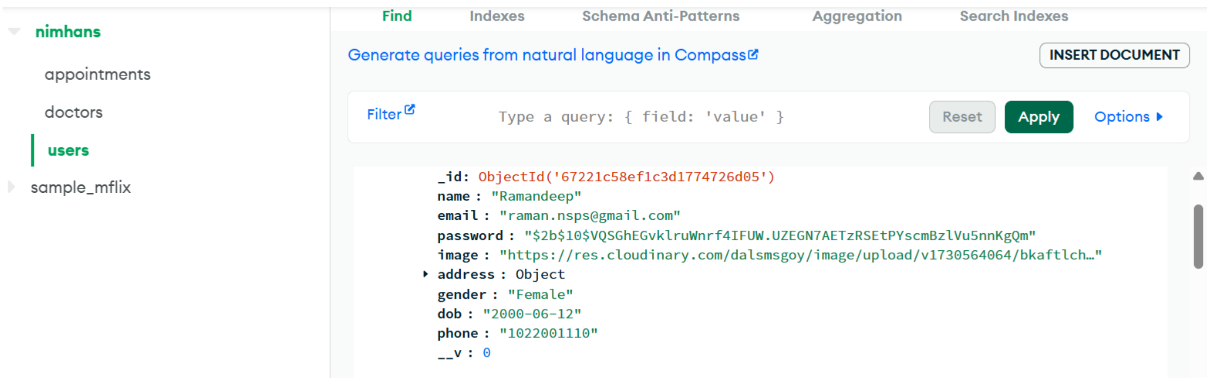


QUERY RESULTS: 1-15 OF 15

```
_id: ObjectId('671fa43bf80869e53d4984f8')
name : "Dr. Richard James"
email : "richard@gmail.com"
password : "$2b$10$/COcM869EyjF/3raIEewqOsaxoRa7yeY1XOrv8PrRtSdPsR/FaVTy"
image : "https://res.cloudinary.com/dalsmsgoy/image/upload/v1730126906/qjl69teu…"
speciality : "General physician"
degree : "MBBS"
experience : "4 Years"
about : "Dr. Davis has a strong commitment to delivering comprehensive medical …"
available : true
```

Fig 5.20: Table for doctors

Fig 5.21 shows the **Patients Table**, a critical component of the Smart Assistance Appointment System's database. This table stores essential patient information, including patient ID, name, age, contact details, and any relevant medical history or notes required for appointment preparation. The data in this table allows the system to quickly retrieve and display patient information, ensuring a personalized and efficient experience during appointment scheduling and management. By maintaining a structured record of patient details, the Patients Table supports secure, organized, and accessible data handling, which is fundamental for smooth operation and continuity of care within the system.



```
  _id: ObjectId('67221c58ef1c3d1774726d05')
  name : "Ramandeep"
  email : "raman.nsps@gmail.com"
  password : "$2b$10$VQSGhEGvklruWnrf4IFUW.UZEGN7AETzRSEtPYscmBzlVu5nnKgQm"
  image : "https://res.cloudinary.com/dalsmsgoy/image/upload/v1730564064/bkaftlch…"
▸ address : Object
  gender : "Female"
  dob : "2000-06-12"
  phone : "1022001110"
  __v : 0
```

Fig 5.21: Table for patients

Chapter 6. Conclusion and Future Scope

6.1 Conclusion

The development of the Smart Assistance Appointment System represents a significant achievement in enhancing the efficiency and accuracy of medical appointment scheduling and record-keeping through advanced natural language processing and machine learning technologies. The project aimed to streamline interactions between patients and medical staff by leveraging voice recognition, dynamic form updating, and real-time data synchronization. Throughout the project lifecycle, a range of modules—including audio transcription, question answering, JSON management, and frontend/backend integration—were developed and seamlessly integrated into a cohesive, robust system.

The use of sophisticated ML models such as ChatGroq, DeepGram, and Whisper enabled the system to handle complex voice commands, transforming them into structured data entries for appointment scheduling and medical record management. Additionally, the integration with MongoDB and a real-time hospital management system ensured accurate, synchronized data across all patient interactions. An extensive testing phase validated the system's stability, accuracy, and usability, ensuring its practicality across various use cases within a healthcare setting. In summary, the Smart Assistance Appointment System stands as a valuable tool for healthcare providers and patients, capable of automating essential processes and enhancing the accessibility and convenience of appointment scheduling. Its intuitive interface, coupled with reliable performance, demonstrates the transformative potential of AI in healthcare applications, where streamlined workflows and accurate record-keeping are essential. With a strong foundation established, the Smart Assistance Appointment System is well-positioned to evolve and adapt, opening doors for further innovations in natural language processing and healthcare management technology.

6.2 Future Scope

The current Smart Assistance Appointment System has met its primary objectives, yet several promising avenues for enhancement and expansion remain. First, by improving Natural Language Understanding (NLU), the system could better interpret complex and context-sensitive patient queries, refining the accuracy of voice-based interactions. Additionally, multi-lingual support would extend accessibility, enabling patients from diverse linguistic backgrounds to interact with the system in their preferred languages. Another key opportunity lies in increasing interoperability with other health information management and electronic medical record (EMR) systems, facilitating seamless access to patient data across healthcare networks. Advanced analytics and reporting capabilities could provide healthcare providers with valuable insights into scheduling patterns, interaction trends, and system performance, supporting more informed decision-making and resource allocation. Furthermore, real-time collaboration features for staff could enhance teamwork, allowing medical personnel to manage appointments and patient records collaboratively, improving communication and coordination. Pursuing these areas for future development could transform the Smart Assistance Appointment System into an essential tool in healthcare, continuously adding value and driving more efficient, intelligent healthcare management solutions.

7. References

[1] Balakesava Reddy, P., Ramasubbareddy, S., & Govinda, K. (2022). AI-based medical voice assistant during covid-19. In Lecture Notes in Networks and Systems (pp. 119–126). Springer Singapore.

[2] Laranjo, L., Dunn, A. G., Tong, H. L., Kocaballi, A. B., Chen, J., Bashir, R., Surian, D., Gallego, B., Magrabi, F., Lau, A. Y. S., & Coiera, E. (2018). Conversational agents in healthcare: a systematic review. Journal of the American Medical Informatics Association: JAMIA, *25*(9), 1248–1258. https://doi.org/10.1093/jamia/ocy072

[3]Appointment maker using computerized voice. (2024, June 3). IJSREM. https://ijsrem.com/download/appointment-maker-using-computerized-voice/

[4] Thomson Reuters Research ID. (2014, August 9). Article detail. International Journal of Advanced Research. https://www.journalijar.com/article/44940/

[5] Medicare: Doctor Appointment Website and hosting using Cloud Services. (2024, May 11). IJSREM. https://ijsrem.com/download/medicare-doctor-appointment-website-and-hosting-using-cloud-services/

[6] Brewer, R., Pierce, C., Upadhyay, P., & Park, L. (2022). An empirical study of older adult's voice assistant use for health information seeking. ACM Transactions on Interactive Intelligent Systems, 12(2), 1–32. https://doi.org/10.1145/3484507

[7] Jin, L., Liu, T., Haroon, A., Stoleru, R., Middleton, M., Zhu, Z., & Chaspari, T. (2023). EMSAssist: An end-to-end mobile voice assistant at the edge for emergency medical services. Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services.

[8] Security and privacy problems in voice assistant applications: A survey Author links open overlay panel Jingjin Li a , Chao Chen b , Mostafa Rahimi Azghadi a , Hossein Ghodosi a. (n.d.).

[9] Kyambille, G. G., & Kalegele, K. (2016). Enhancing patient appointments scheduling that uses mobile technology. In arXiv [cs.CY*]*. http://arxiv.org/abs/1602.03337

[10] Malik, S., Bibi, N., Khan, S., Sultana, R., & Rauf, S. A. (2017). Mr. Doc: A doctor appointment application system. In arXiv [cs.CY]. http://arxiv.org/abs/1701.08786

[11] Medical ChatBot Assistance for Primary Clinical Guidance using Machine Learning Techniques. (n.d.).

[12] Srivastava, K., Pandey, T. N., Roy, D., & Sahoo, S. (2023). A machine learning model on healthcare based chatbot and appointment system. 2023 3rd International Conference on Artificial Intelligence and Signal Processing (AISP).

[13] NIMHANS – National Institute of Mental Health and Neurosciences

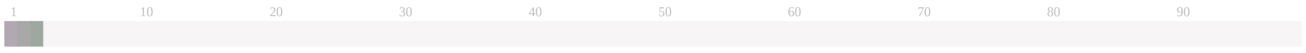
The Report is Generated by DrillBit Plagiarism Detection Software

Submission Information

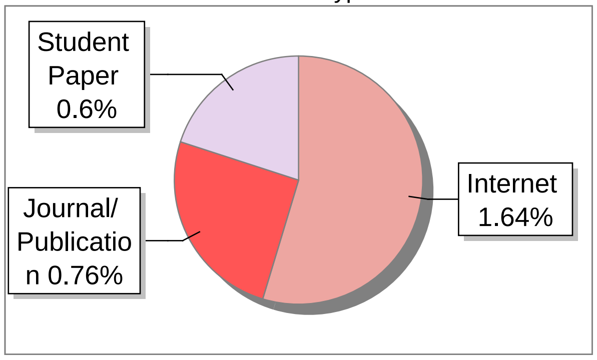
Author Name	Bharatdeep Singh Chandanbir Singh Divneet Kaur
Title	Smart assistance appointment system for NIMHANS
Paper/Submission ID	2549133
Submitted by	amanpreet_brar@gndec.ac.in
Submission Date	2024-11-20 09:26:22
Total Pages, Total Words	82, 17111
Document type	Project Work

Result Information

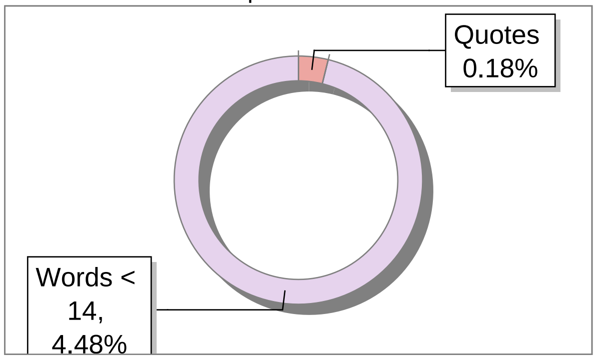
Similarity **3 %**



Sources Type



Report Content



Exclude Information

Quotes	Excluded
References/Bibliography	Excluded
Source: Excluded < 14 Words	Excluded
Excluded Source	**0 %**
Excluded Phrases	Not Excluded

Database Selection

Language	English
Student Papers	Yes
Journals & publishers	Yes
Internet or Web	Yes
Institution Repository	Yes

A Unique QR Code use to View/Download/Share Pdf File

QR CODE AND GITHUB LINK

https://github.com/singh-chandanbir/smart-assistance-appointment-system