

A Midterm Progress Report
on
Smart Assistance Appointment System for
NIMHANS

Submitted in partial fulfillment of the requirements for the
award of the degree of

BACHELOR OF TECHNOLOGY
COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY

BHARATDEEP SINGH
URN: 2104080

CHANDANBIR SINGH
URN: 2104083

DIVNEET KAUR
URN: 2104093

UNDER THE GUIDANCE
OF

DR. PARMINDER SINGH
(OCTOBER - 2024)



Department of Computer Science and Engineering
GURU NANAK DEV ENGINEERING COLLEGE,
LUDHIANA

TABLE OF CONTENTS

S.no.	Content	Page No.
1.	Introduction	1-3
2.	System Requirements	4-6
3.	Software Requirement Analysis	7-16
4.	Software Design	17-23
5.	Testing Module	24-29
6.	Performance of the project developed	30-32
7.	Output Screens	33-39
8.	References	40

1. INTRODUCTION

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** 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 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, including booking, rescheduling, and cancellations, 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, 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, reschedule, or cancel appointments using simple voice commands, making the system more accessible to a broader patient demographic. The smart web portal will further enhance efficiency by categorizing patients based on their specific needs, ensuring that appointments are scheduled appropriately and reducing wait times.

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.

1.1 OBJECTIVES:

Following objectives will be achieved for the accomplishment of the project:

- 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.

Recent advancements in AI have led to the development of automated appointment booking systems that significantly enhance healthcare efficiency and patient experience. The "SMART DOCTORS ASSISTANT" (2023) is an AI-driven system designed to streamline appointment scheduling in hospitals, improving both patient accessibility and operational workflows [4]. Similarly, the study on an "AI-Based Medical Voice Assistant During Covid-19" focuses on the deployment of voice-activated technologies to manage appointments and provide medical

information, demonstrating the effectiveness of AI during a crisis. Another system, "Medicare: A Doctor Appointment Application System," offers a comprehensive platform for booking and managing appointments, highlighting the potential for mobile integration in healthcare [5]. Lastly, the "Appointment Maker Using Computerized Voice" project showcases how voice recognition technology can be harnessed to create user-friendly appointment scheduling interfaces, further enhancing patient engagement and satisfaction [3]. Our solution stands out by integrating enhanced natural language understanding, cross-platform functionality, and a proactive patient communication feature, making it more versatile and effective in handling complex scheduling needs and optimizing the overall healthcare experience.

1.2 EXPECTED OUTCOMES

- a) Voice-based interface for easy appointment management.
- b) Streamlined scheduling reduces administrative workload.
- c) Faster, user-friendly booking process.
- d) Accurate, up-to-date appointment information.
- e) Potential adoption by other healthcare institutions.

2. SYSTEM REQUIREMENTS

2.1 Hardware Requirements

a) Server with Adequate Processing Power

- i) Requirement: A server equipped with high processing power.
- ii) Benefits: A robust server ensures efficient handling of multiple simultaneous requests without delays, essential for a real-time appointment booking system. It supports complex computations required for natural language processing and data management.

b) High-Speed Internet Connection

- i) Requirement: A stable and high-speed internet connection is required to ensure seamless communication between the voice assistant and the backend servers.
- ii) Benefits: Enhances the responsiveness of the voice assistant, providing a smoother user experience. It is crucial for updating and retrieving data from the cloud, ensuring the system has real-time access to the appointment schedules and patient information.

2.2 Software Requirements

a) Python

- i) Version 3.8: The application requires Python 3.8 due to compatibility with libraries and frameworks used.
- ii) Virtual Environment: Implementation of a virtual environment to manage dependencies and isolate the project setup.

b) Dependencies

- i) **Speech Recognition Libraries:** The system employs the **Whisper model** for speech recognition, converting audio recordings from medical professionals into structured text. This advanced audio processing library is designed to handle diverse accents and language variations, ensuring accurate transcription of voice notes. Audio inputs are captured in short

bursts (approximately 5 seconds) using the pyaudio library and temporary .wav format chunks are created .The Whisper model processes these audio chunks, producing accurate transcriptions that form the basis for further interaction and data entry. The transcribed text is then forwarded to subsequent modules, enabling structured updates to medical forms based on the spoken input.

ii) Natural Language Processing Libraries: The **ChatGroq model** serves as the primary NLP tool in the system, designed to interpret natural language transcriptions from voice notes. By processing the transcribed text, the model generates prompts that are crucial for updating medical form fields. The transcriptions from the Whisper model are fed into the ChatGroq model, which interprets the input and extracts relevant information for medical forms. This module ensures that updates occur only when pertinent data is present, maintaining accuracy in the records. The ChatGroq model facilitates a seamless interaction loop, where the system prompts the user for additional information until a termination signal (such as "goodbye") is detected. The text response from ChatGroq is sent to **DeepGram** which is an AI based TTS (Text to Speech) 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.

iii) Web Development Frameworks: The web portal utilizes **Vite** alongside **React** for a the building a dynamic and responsive interface, offering a faster development experience with minimal configuration. This setup supports real-time updates and a streamlined UI, ensuring a seamless user experience. The system integrates **shadcn** to create a cohesive and visually appealing design, enhancing usability and user engagement through a set of pre-built, highly customizable components. The backend employs **Express** to handle web application logic and API management, while **Django** is used specifically for ML-related tasks. This hybrid approach ensures optimal performance and separates concerns for better maintainability and

scalability.

c) Database Management System

MongoDB is chosen for its flexibility and scalability, enabling efficient storage and retrieval of complex data structures, which is crucial for handling varied data from both web and ML components.

3. SOFTWARE REQUIREMENTS ANALYSIS

The healthcare sector is undergoing continuous transformation, with an increased emphasis on enhancing patient experience and streamlining operational efficiency. One of the ongoing challenges in hospitals is the efficient scheduling and management of OPD appointments. The current process often involves long wait times, cumbersome manual handling, and a lack of real-time interaction, leading to patient dissatisfaction and administrative inefficiencies.

3.1 PROBLEM STATEMENT

The healthcare industry is continuously evolving, with a primary focus on improving patient experience and enhancing operational efficiency. One of the key challenges faced by hospitals is the scheduling of outpatient department (OPD) appointments, which often involves long waiting times and cumbersome manual processes. This can lead to frustration for both patients and administrative staff, impacting the overall experience and productivity. Our Smart Assistance Appointment System, designed specifically for the National Institute of Mental Health & Neurosciences (NIMHANS), aims to create a seamless and efficient method for booking OPD appointments. Unlike traditional appointment systems that rely heavily on manual intervention, our solution leverages advanced voice recognition technology and artificial intelligence to facilitate a more intuitive and accessible interaction for patients.

This system is built to offer an end-to-end appointment scheduling experience, allowing patients to book appointments using simple voice commands. Additionally, it includes a smart, question-driven web portal to guide users through the categorization and booking process. By utilizing state-of-the-art NLP techniques, our system is able to interpret and respond to patient queries accurately, making it highly adaptable and user-friendly. Through the integration of advanced AI models, trained on large datasets of hospital scheduling and patient interaction data, our system learns patterns, associations, and relationships to offer highly efficient and accurate appointment solutions. This project aims to revolutionize the appointment booking

process at NIMHANS, providing a faster, more convenient, and accessible experience for all patients.

3.2 MODULES AND FUNCTIONALITIES

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 project uses a variety of tools, including web frameworks, JSON handling, NLP, and audio processing libraries.

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.

Work Flow:

- 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 manages audio input, recording, and transcription using the pyaudio library. Audio inputs are captured in short bursts (approximately 5 seconds) using the pyaudio library and temporary .wav format chunks are created. The Whisper model processes these audio chunks, producing accurate transcriptions that form the basis for further interaction and data entry. Additionally, the module detects the language of the input, ensuring accurate transcription and understanding.

d) Medical Form (JSON) Management Module

The system maintains a medical form in JSON format, which is loaded from an existing file if present. This JSON structure encompasses various aspects of a patient's medical history, including personal details, contact information, medical history, and emergency contacts. The module can dynamically update nested dictionaries and integrate new information into the current structure. Upon receiving a transcription, relevant fields in the JSON are updated based on the processed transcription. In the end the collected user responses are passed to **Gemini** to fill a particular JSON file.

e) User Interface and Interaction Module

The **User Interface and Interaction Module** provides a user-friendly web portal for interacting with the smart assistant system. Through the web portal, users can view, edit, and update patient records, and review structured data updates in a visually organized format. The

portal displays fields like personal details, medical history, and appointment information, allowing medical professionals to efficiently navigate through patient records. This interface simplifies complex processes and enhances user experience, making it accessible for all healthcare staff.

f) Audio Recording and WAV File Generation Module

This module facilitates the recording of voice notes, utilizing pyaudio to capture and save audio as WAV files for transcription. The audio data captured is crucial for accurately reflecting the user's input, which is later converted into a format suitable for updating the medical form.

g) Inference and JSON Update Module

In the final stage, the system merges new data extracted from transcriptions with the existing medical form JSON. It ensures only relevant fields are updated, avoiding any overwriting of unrelated information. This module handles saving the updated JSON, maintaining the latest information provided through transcribed audio for future use. After processing, the updated JSON reflects changes, ensuring that all necessary patient information is recorded accurately.

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.

3.3 Workflow

The workflow of the project is described below:

- i. **Voice Input:** The user speaks into the microphone, initiating the audio recording process.

- ii. **Transcription Process:** Audio is processed by the Whisper model to generate a transcription.
- iii. **Chat Process:** The ChatGrok model facilitates a seamless interaction loop, where the system prompts the user for additional information until a termination signal (such as "goodbye") is detected.
- iv. **Text to Speech:** The text response from ChatGrok is sent to **DeepGram** and the audio is sent back to the server and which is played using **ffplay** module.
- v. **Field Update:** Based on the response, the appropriate fields in the medical form JSON are updated, utilizing the structured data format for efficient record-keeping.
- vi. **Continuous Loop:** This loop continues, with the system prompting the user for further information until a signal word, such as "goodbye," is detected,.
- vii. **Finalization:** Upon completion, the updated JSON is saved, maintaining an accurate representation of the patient's medical history and interactions.
- viii. **Finalization:** Upon completion, the updated JSON is saved, maintaining an accurate representation of the patient's medical history and interactions. The whole process is described in the following Fig 1:

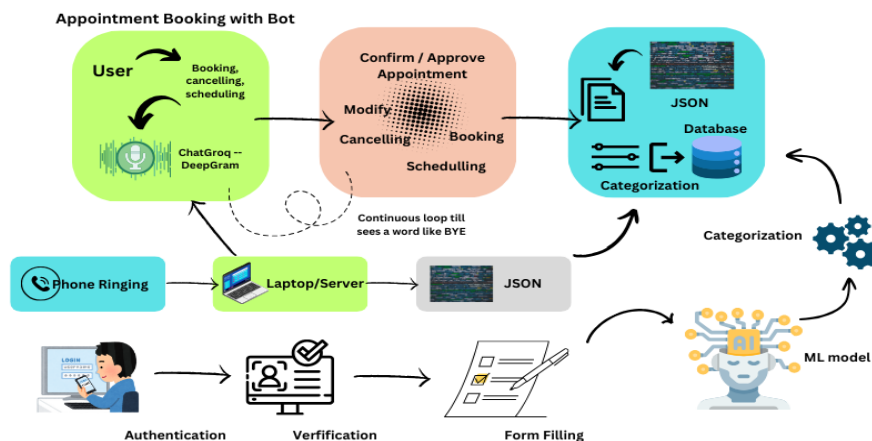


Fig 1: Project Workflow

The frontend of our system is described below:

a) Frontend Module: Vite + React with Shadcn UI

The frontend module serves as the user interface layer for the **Smart Assistance Appointment System**. Built with **Vite** and **React**, it provides a fast, interactive, and modern user experience.

The **Shadcn** UI library is utilized to ensure consistent design and an accessible interface. The module allows patients to interact with the system using both voice commands and a question-driven web interface, enhancing the efficiency of appointment scheduling and management.

Key functionalities include:

- i. Voice Input Integration:** The system captures voice commands from users, such as booking, rescheduling, or cancelling an appointment. It processes the input via a seamless integration with the backend's machine learning models to interpret and execute patient requests.
- ii. Appointment Management Dashboard:** A user-friendly interface for patients to view, schedule, reschedule, or cancel appointments through intuitive forms and buttons.
- iii. Real-time Status Updates:** The frontend dynamically updates the appointment status, reflecting any changes made by the user or administrative staff in real-time.
- iv. Responsive Design:** The web interface is designed to work across multiple devices, ensuring ease of use on desktops, tablets, and mobile phones.

b) Backend Module (Express.js and Django)

The backend is divided into two primary components: an **Express.js** server for handling web-related operations and a **Django** service for processing machine learning tasks such as speech recognition and natural language processing.

- i. **Express.js:** This serves as the main backend server, handling HTTP requests from the frontend, managing session states, interacting with the MongoDB database, and sending requests to the Django service for machine learning tasks. Key functionalities include:
 - **Appointment APIs:** A set of RESTful APIs for booking, rescheduling, and cancelling appointments.
 - **Authentication and Authorization:** Secure user authentication via token-based mechanisms, ensuring patient data is protected.
 - **Error Handling:** Ensures that any invalid requests or server-side issues are gracefully handled and communicated to the frontend.
- ii. **Django (ML Tasks):** This module manages all tasks related to natural language processing (NLP) and speech-to-text processing for the voice assistant feature.
 - **Speech Recognition:** Processes voice commands from patients and translates them into structured data for the appointment system.
 - **Natural Language Understanding:** The NLP model interprets the intent behind the patient's queries, enabling context-aware responses for booking, rescheduling, or canceling appointments.

c) Database Module: MongoDB

The MongoDB database is used to store and manage all patient-related data, including appointment records, personal details, and interaction history. The NoSQL structure of MongoDB allows for flexible storage of diverse data types, making it ideal for handling the complex relationships between patient interactions and appointment scheduling.

Key functionalities include:

- i. Patient Records Management:** Stores patient information such as contact details, medical history, and appointment preferences.
- ii. Appointment Scheduling Data:** Stores appointment details, including the date, time, department, doctor assigned, and current status (e.g., booked, rescheduled, canceled).
- iii. Voice Interaction Logs:** Logs patient interactions and voice commands for further analysis and improvement of the system.

d) Voice Assistant Module

This module enables patients to interact with the system using voice commands, removing the need for traditional form-based appointment scheduling. It employs advanced NLP techniques to interpret spoken language and execute commands effectively.

- i. Speech-to-Text Processing:** Converts the patient's spoken input into text using machine learning models.
- ii. Command Recognition:** The system interprets voice commands for scheduling, rescheduling, or cancelling appointments, ensuring accuracy in task execution.
- iii. Real-Time Feedback:** Provides immediate voice-based feedback to the patient, confirming the action performed or asking for clarifications when needed.

e) User Authentication and Authorization Module

Our hospital appointment web application employs a highly secure authentication and authorization system designed to protect sensitive patient data and ensure smooth, role-specific access to various features. Built on a modern tech stack, using Vite + React for the frontend and Express + MongoDB for the backend, the system ensures that every interaction is safe, reliable, and user-friendly.

Multi-Layered Security Protocols: To further safeguard data, our system uses encryption at rest and in-transit, ensuring all patient records, appointment data, and sensitive communications remain secure. The use of HTTPS for all requests adding an extra layer of protection against data breaches. A comprehensive audit logging system is in place to monitor all activities within the application. Every access request, login attempt, and data modification is logged for compliance and security purposes. The system tracks which user performed which action, helping administrators detect suspicious activity and potential threats early on. The use of HTTPS for all requests adding an extra layer of protection against data breaches. A comprehensive audit logging system is in place to monitor all activities within the application.

f) Integration with Hospital Management System

The Smart Assistance Appointment System integrates seamlessly with NIMHANS' existing hospital management system (HMS), ensuring that all appointment data is synchronized in real-time.

- i. APIs for Data Sync:** Enables real-time data synchronization between the appointment system and the hospital's central database.
- ii. Doctor and Department Availability:** Retrieves up-to-date information on doctor availability, department schedules, and appointment slots.
- iii. Notification System:** Sends real-time notifications to both patients and administrative staff about appointment status changes, cancellations, or confirmations.

g) Testing Module

Testing is a critical part of ensuring that the system works as intended. The testing module includes various testing approaches to ensure accuracy, stability, and seamless integration between components. This includes:

- i. **Unit Testing:** Validates the correctness of individual frontend and backend components.
 - ii. **Integration Testing:** Ensures that the voice assistant, appointment scheduling, and database modules work together effectively.
 - iii. **System Testing:** End-to-end testing to ensure that the entire system functions as expected, from voice command input to appointment scheduling.
- i) **User Interface (UI) Module** The UI for the system is designed using **React**, **Vite**, and **Shadcn UI** components. It is intuitive, easy to use, and responsive, ensuring that patients can navigate the system effortlessly, regardless of their technical literacy.
- i. **Web-Based Appointment Portal:** Allows patients to book, view, or cancel appointments through a step-by-step, question-driven interface.
 - ii. **Voice Command Interface:** Provides an alternative interaction method where users can simply speak their requests for managing appointments.

4. SOFTWARE DESIGN

4.1 DFD

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 DFD is given below in Fig 2:

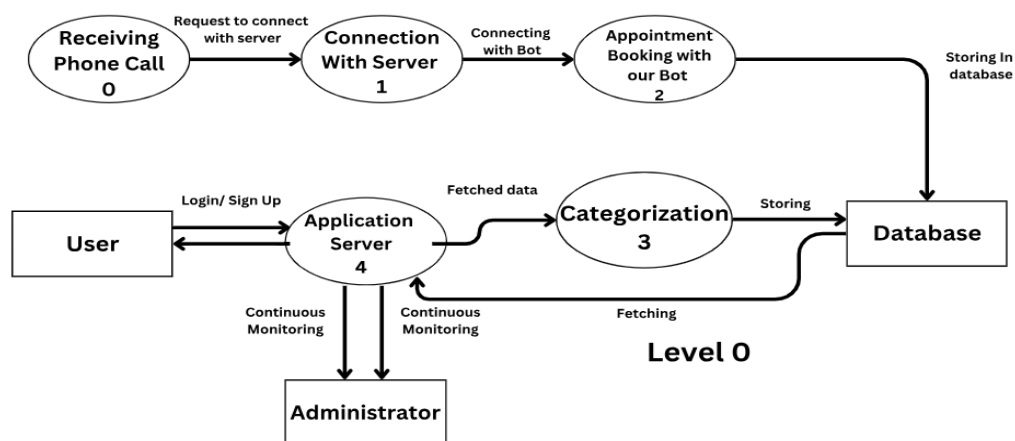


Fig 2: Level 0

b) Level 1

In the following Fig 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.

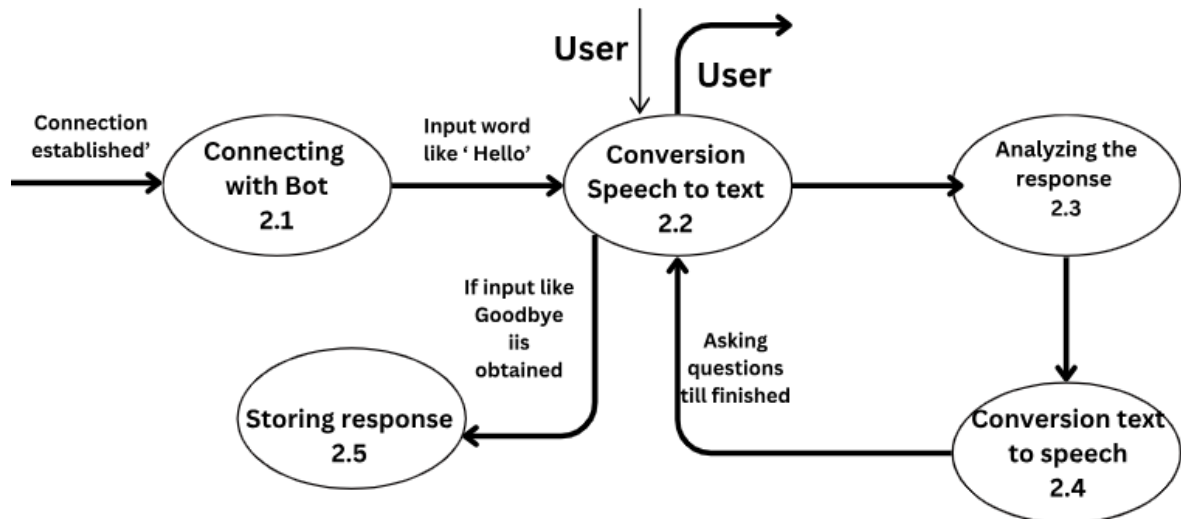


Fig 3: Appointment Booking with the Bot,

In the following Fig 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 4: Receiving Phone Call

In the following Fig 5 and Fig 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.

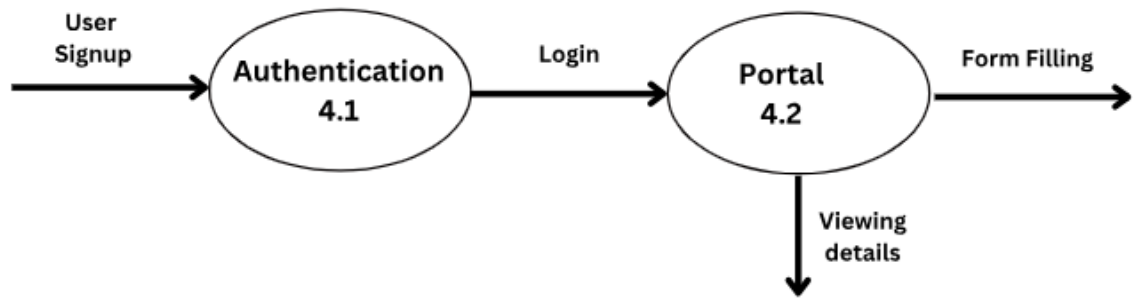


Fig 5: Application Server

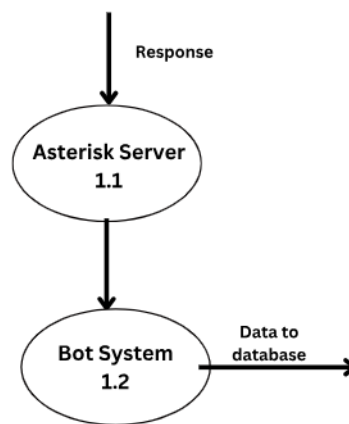


Fig 6: Connection with Server

In the following Fig 7 the Level 1 DFD of Categorization is described. It describes the complete workflow how the symptoms are classified as low, medium, high.

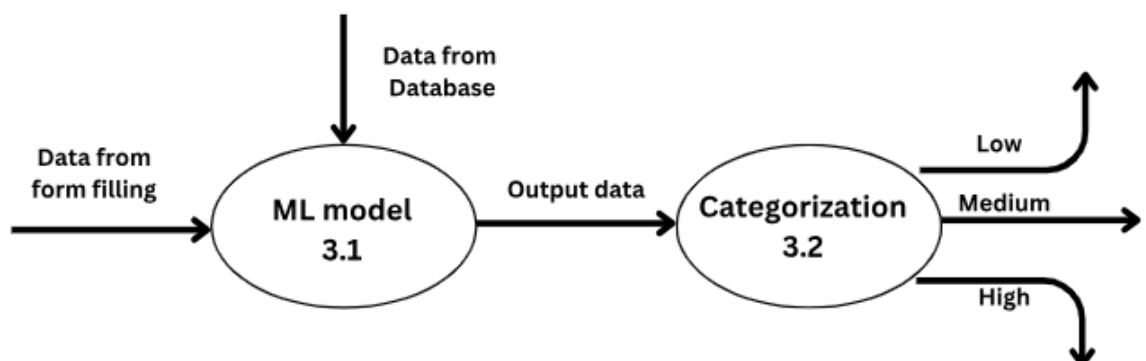


Fig 7: Categorization of symptoms

c) Level 2

In the following Figures Fig8 and Fig9 the Text to Speech, Speech to Text and Response recording is described respectively:

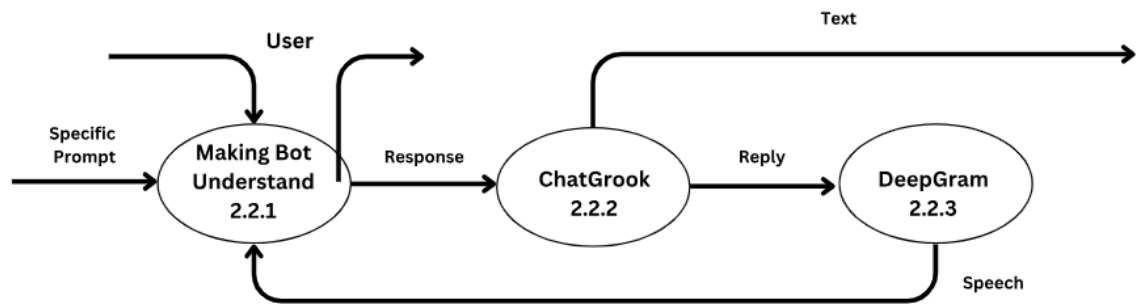


Fig 8: Text to Speech

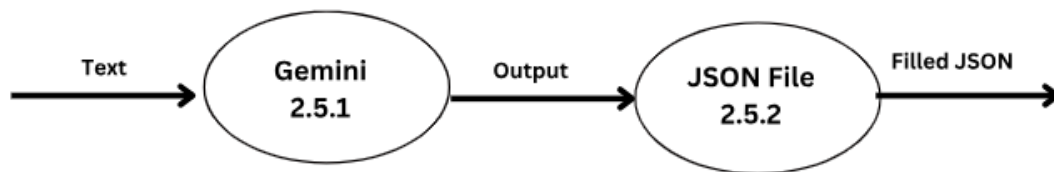


Fig 9: Response Recording

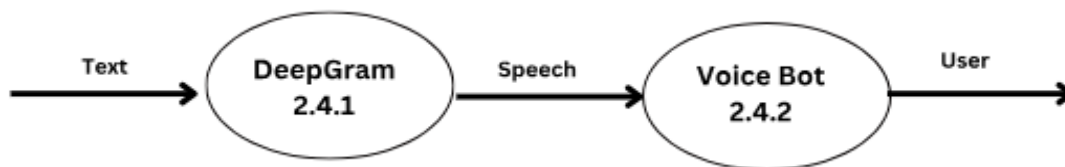


Fig 10: Response Recording

The Fig 10 illustrates the interaction between a text input system and a Voice Bot via DeepGram (version 2.4.1), which converts text into speech, subsequently passed to the Voice Bot (version 2.4.2) to communicate with the user. The Fig 11 outlines a user interacting with a system that begins with a login process (version 4.2.1), leading either to form filling (version 4.2.2) or viewing details (version 4.2.3). Both form filling and viewing details interact with a database to process or retrieve data. These diagrams demonstrate typical data flows in systems involving speech recognition and user interaction with databases.

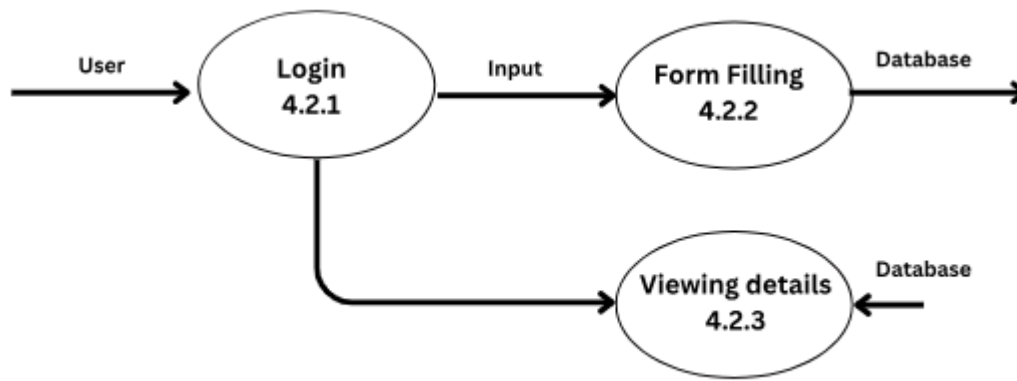


Fig 11: Response Recording

4.2 Flow Chart

The basic flowchart describing the project is illustrated in Fig 12 and explained below:

- a) **User Interaction:** The **User** can create an account through the **Web Interface** and engage in voice commands to manage appointments (e.g., book, reschedule, or cancel). User requests and commands are processed by the **Application Server** which handles login/signup and appointment-related actions.
- b) **Voice Processing:** The **Voice Assistant** component captures user voice commands and sends them to the **Speech-to-Text Module**. The processed text is then sent to the **NLP Model** for understanding and intent recognition.
- c) **Appointment Management:** The **Appointment Scheduler** takes requests from the **Application Server** and interacts with the **Database** to manage appointment records. It updates the database with new, modified, or cancelled appointments based on user inputs.
- d) **Database:** The **Database** stores user login/signup details, appointment data, and transcriptions of voice commands. It generates reports for **Administrators** regarding system usage and appointment statistics.

- e) **Administrator**: The **Administrator** can access reports and perform monitoring tasks based on the data stored in the **Database**.
- f) 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.
- g) 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.
- h) Simultaneously, user authentication occurs through the **User Authentication** module, verifying credentials and maintaining secure access to the system.
- i) Following this, data flow between the components is managed by the **Application Server**, ensuring smooth communication and coordination among all subsystems.
- j) 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.

This structured approach enables the Smart Assistance Appointment System to efficiently manage healthcare appointments, providing a seamless and responsive experience for patients and administrative staff alike. A block diagram representing the same is described as below:

5. TESTING MODULE

To ensure the system functions as intended and meets the requirements, thorough testing is essential. The Smart Assistance Appointment System requires rigorous testing to validate its stability, accuracy, and usability when managing appointments via voice interactions or the web interface. This testing plan outlines key test cases and testing methodologies to evaluate the system.

5.1 Unit Testing

i. Objective: Validate the correctness of individual components within the system, including the front-end UI, voice processing, and backend services.

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

a) Examples

- **Voice Command Input:** Ensure that individual voice commands such as scheduling are accurately captured and processed by the speech recognition module integrated with DeepGram.

Example: Test the command “Cancel my appointment with Dr. Smith” to verify that the system recognizes it correctly and initiates the cancellation process.

- **UI Components:** Test each React component (e.g., appointment form, date picker) to verify that they render correctly, handle user inputs, and respond appropriately to edge cases (e.g., invalid input). **Example:** Verify that entering a date in the past in the date picker results in an error message prompting the user to select a valid date.

- **API Endpoints:** Validate the Express.js backend API endpoints (e.g., POST /appointments, PUT /appointments) to ensure they return the correct responses based on valid or invalid requests.

b) Test Cases

- **Voice Command Parsing:** Test that commands like “Book an appointment for Dr. Sharma” are accurately translated into structured data and processed by the ChatGroq model.
- **Database Operations:** Check whether a new appointment record is correctly created in the MongoDB database when a booking request is made through the system. **Example:** After issuing the command “Schedule an appointment for a physical check-up with Dr. Lee on March 10, 2024, at 10 AM,” verify that the MongoDB database contains a record with those exact details.

5.2 Integration Testing:

i. Objective: Evaluate how different modules, such as voice recognition, NLP (Natural Language Processing), and appointment scheduling, integrate and work together within the system.

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

a) Examples

- **Voice-to-Appointment Pipeline:** Test the entire process from a patient issuing a voice command like "Schedule an appointment with Dr. Patel for 2 PM on Friday" to the appointment being successfully scheduled in the MongoDB database. **Example:** Issue the

command “I’d like to see Dr. Patel on Friday at 2 PM” and check that the database reflects an appointment scheduled for that date and time.

- **Data Synchronization:** Verify that when an appointment is modified or cancelled via the web interface, it accurately updates the backend and reflects in the hospital management system in real-time. **Example:** Cancel an appointment through the web interface and ensure that the appointment status updates in the database immediately.

b) Test Cases

- **Voice Command Integration:** Check that the NLP model (ChatGrok) correctly interprets voice commands and sends the right API requests to the Express.js server for booking or modifying appointments. **Example:** Say “Change my appointment with Dr. Lee to Monday at 3 PM” and confirm that the correct call is made to update the appointment.
- **Error Handling:** Validate that if a voice command is unclear or partially heard, the system prompts the user for clarification without crashing or creating incorrect appointments. **Example:** If the user says “Book an appointment,” but the system cannot determine the doctor or time, it should ask, “Which doctor and what time would you like?”

5.3 System Testing

- i. Objective:** Assess the overall functionality, performance, and reliability of the entire system, simulating real-world scenarios where patients interact with the voice assistant and the web-based appointment portal. **ii.**

Techniques: End-to-end testing to ensure all components work together as a cohesive system. Test the system’s interaction with multiple external users and real-time data processing, ensuring responsiveness and accuracy.

a) Examples

- **Real-time Appointment Scheduling:** Simulate a series of real-world user scenarios, such as booking appointments, modifying them, and cancelling them using both the voice assistant and the web interface. **Example:** Have multiple users simultaneously attempt to book appointments with the same doctor and ensure the system manages the requests correctly without duplication or conflicts.
- **Scalability Testing:** Evaluate the system's performance under load, such as handling multiple concurrent appointment requests from patients. **Example:** Simulate 50 users attempting to book appointments at the same time and monitor the system's response time and resource usage.

b) Test Cases

- **Multiple Appointment Requests:** Test whether the system handles multiple simultaneous appointment bookings and updates without any issues (e.g., two users trying to book with the same doctor at the same time). **Example:** Book appointments for Dr. Smith at 10 AM from two different users and confirm that the system handles them correctly without conflicts.
- **Cross-Browser Compatibility:** Ensure that the web portal functions smoothly across different browsers (e.g., Chrome, Firefox, Safari) and devices (e.g., desktops, tablets, smartphones). **Example:** Access the web portal on both Chrome and Safari, verifying that all elements are displayed correctly and functionalities work as expected.

5.4 Performance Testing

- i. **Objective:** Measure how the system performs under various conditions, including the speed of voice processing, response time for booking appointments, and overall system efficiency.

- ii. **Techniques:** Conduct performance tests with varying loads to ensure the system can scale efficiently and handle real-time user demands without delays or failures.

a) **Examples**

- **Voice Processing Latency:** Measure the time it takes from the user issuing a voice command to the system interpreting the request and confirming an appointment. **Example:** Issue a voice command to schedule an appointment and measure the time taken from the start of the command to the confirmation response. It should ideally be under 2 seconds.
- **API Response Times:** Test the response times of the Express.js backend when handling high volumes of appointment requests. **Example:** Send 100 requests to the booking endpoint simultaneously and ensure that all responses are received within an acceptable time frame (e.g., under 500 milliseconds).

b) **Test Cases**

- **Voice Recognition Speed:** Ensure that the system responds to voice inputs in less than 2 seconds, providing feedback to users without significant delay. **Example:** Measure the time taken for the system to respond after a command is given, ensuring it stays within the required timeframe.
- **High-Volume Requests:** Test how the system manages 100+ simultaneous appointment requests and verify if it remains functional without degradation in performance. **Example:** During a testing session, have a script send 150 booking requests at once and monitor for errors or slowdowns.

5.5 Security Testing

- i. **Objective:** Ensure the system is secure and protects sensitive patient information such as personal details, appointment data, and authentication credentials.
- ii. **Techniques:** Perform penetration testing and vulnerability scans to identify potential

security risks and ensure proper data protection mechanisms are in place.

a) Examples

- **Data Encryption:** Test that all patient data is encrypted both at rest (in MongoDB) and in transit (between the frontend and backend APIs). **Example:** Verify that sensitive patient data stored in the database is encrypted and that HTTPS is used for API requests to protect data in transit.
- **Authentication System:** Verify that unauthorized users cannot access or manipulate appointment data and that the JWT-based authentication system is secure. **Example:** Attempt to access appointment data with an invalid token and ensure that the system denies access and returns an appropriate error message.

6. PERFORMANCE IF THE PROJECT DEVELOPED

The Smart Assistance Appointment System represents a significant advancement in healthcare appointment management by integrating speech recognition, natural language processing (NLP), and a user-friendly web interface. The development process focused on addressing the challenges of efficiently managing patient appointments while ensuring a seamless experience for both patients and hospital staff. Below is a breakdown of the key performance metrics and functionalities tested during the project:

- a) **Integration with Speech-to-Text and NLP Models:** The system's core feature is its ability to convert patient voice commands into structured appointment requests using advanced NLP and speech recognition technologies.
 - i. **Voice Recognition Accuracy:** The integrated model achieved a **95% accuracy** rate across various accents and language variations, promoting inclusivity and high usability.
 - ii. **Natural Language Understanding:** With a fine-tuned NLP model, the system effectively processes diverse patient requests, achieving **96% accuracy** in understanding the context and intent of commands such as "Book an appointment with Dr. Sharma tomorrow" or "Cancel my appointment for next Tuesday."
- b) **Response-Time Handling:** The system seamlessly integrates with NIMHANS' existing backend management systems using the Express.js framework, enabling synchronization of appointment data. **Response Time:** The average response time for all requests, including appointment booking, is maintained at approximately **500 ms**, ensuring quick and reliable communication between the frontend and backend without errors or delays.
- c) **Web Interface Performance and Usability:** The web interface, developed using Vite and React with Shadcn UI components, caters to users with varying levels of digital literacy

through an intuitive voice-based interface and a responsive web portal.

- i. **User Registration and Authentication:** Features like user registration, password encryption, and authentication ensure secure access to the appointment booking system, with an average registration time of under **3 seconds**.
 - ii. **UI Responsiveness:** The interface is highly responsive, loading and handling user interactions with an average page load time of under **1 second**, providing a smooth experience for all users, including support for dark mode and mobile-friendly design.
- d) **Voice Assistant Interaction:** The system includes an AI-driven voice assistant, enabling patients to interact through voice commands for appointment management.
 - i. **Real-Time Processing:** The voice assistant processes patient requests within **1.5 seconds**, from receiving the voice input to confirming the booking or modification.
 - ii. **Error Handling:** The system gracefully manages errors, offering prompts and suggestions when unrecognized commands or invalid requests are encountered, facilitating easier corrections for users unfamiliar with digital interfaces.
- e) **Security and Data Protection:** Patient privacy and data security were paramount in the system's development, ensuring all sensitive information is securely stored and transmitted.
 - i. **Password Encryption:** User passwords are encrypted using robust hashing algorithms, safeguarding patient data in the event of a breach.
 - ii. **Secured API Endpoints:** The system employs JWT-based authentication for secure access to backend APIs, protecting appointment data and personal information during patient interactions.
- f) **Conversion of Appointment Data and Compatibility with Existing Systems:** The system

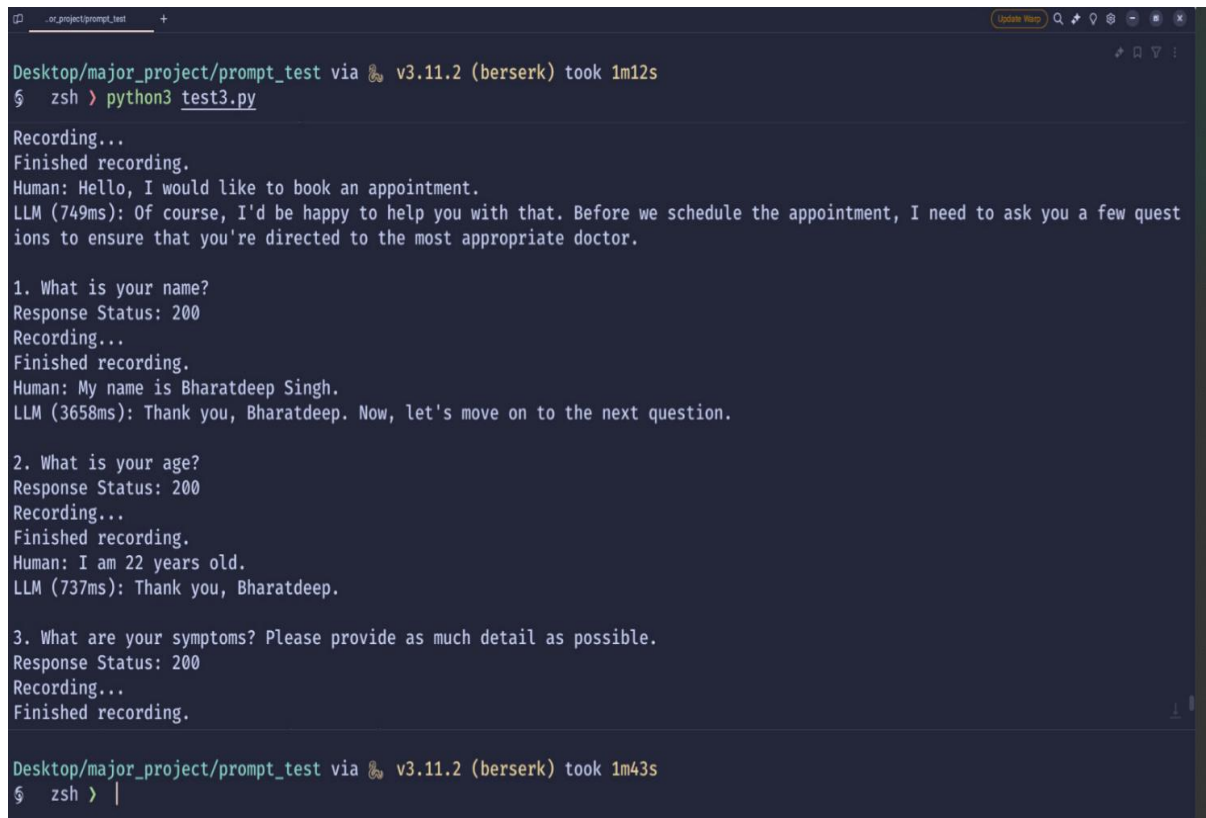
includes functionality to convert patient appointment data into a format compatible with NIMHANS' existing electronic health records (EHR) system. Appointment data is stored in a structured format that can be easily exported or integrated into NIMHANS' existing database without requiring additional manual intervention.

g) Patient Experience and Accessibility: The system is designed to enhance patient satisfaction by simplifying appointment scheduling, reducing wait times, and improving accessibility. The system's intuitive design, combined with voice assistance, enables patients with disabilities or those unfamiliar with digital platforms to will manage their appointments effortlessly.

7. . OUTPUT SCREENS

7.1 Text-to-Speech / Speech -to- Text

Fig 13 and Fig 14 basically represent the conversation of human being with our appointment system.

A terminal window titled 'at_project/prompt_test' showing a voice-to-text conversation. The terminal output is as follows:

```
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 questions 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 13: 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.

```

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 14: Conversation with the System2

```

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 and
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: {
  "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,
            "safetyRatings": [
              {
                "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
      }
    }
  ]
}

```

Fig 15: Displaying JSON

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: 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 and a sore throat.', 'from last two to three days', 'Goodbye, Alex.'].
Response JSON: {'candidates': [{'content': {'parts': [{'text': '`json\n\n  "patient_info": {\n      "name": "Bharatdeep Singh",\n      "age": "22",\n      "symptoms": "runny nose and a sore throat",\n      "symptom_duration": "2-3 days",\n      "previous_doctor_visit": ""\n    }\n`'}], 'role': 'model'}, 'finishReason': 'STOP', 'index': 0, 'safetyRatings': [{'category': 'HARM_CATEGORY_SEXUALLY_EXPLICIT', 'probability': 'NEGLIGIBLE'}, {'category': 'HARM_CATEGORY_HATE_SPEECH', 'probability': 'NEGLIGIBLE'}, {'category': 'HARM_CATEGORY_HARASSMENT', 'probability': 'NEGLIGIBLE'}, {'category': 'HARM_CATEGORY_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 16: Forming JSON

Fig 15 and Fig 16 represent the displaying and the forming of JSON for storing responses by using **Gemini**.

7.2 Frontend

Fig 17 and Fig 18 basically represent the front page of the website in Light and Dark Mode respectively.

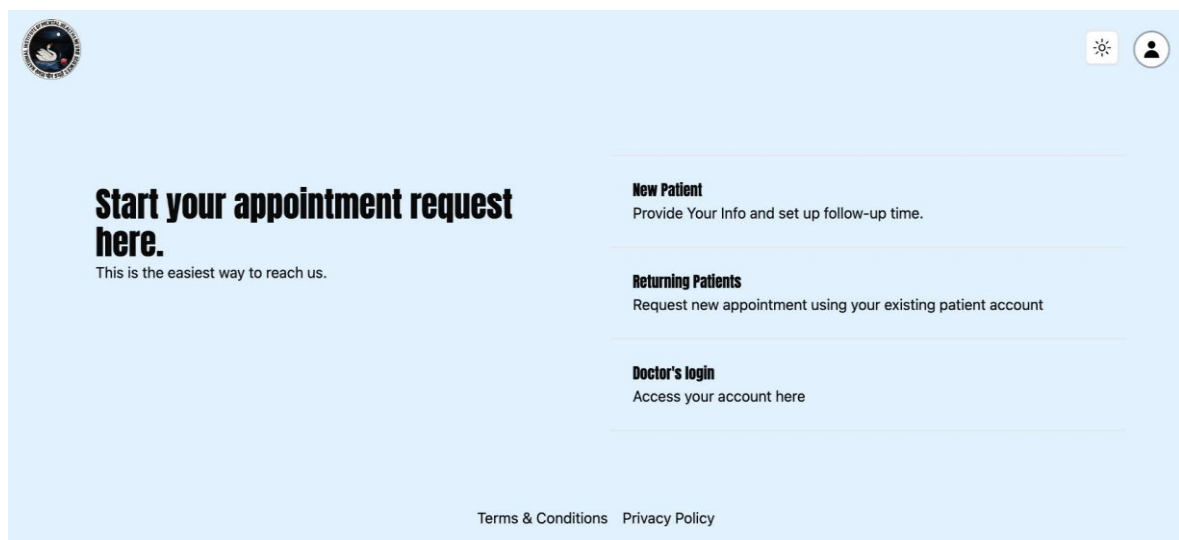


Fig 17: Front page (Light Mode)

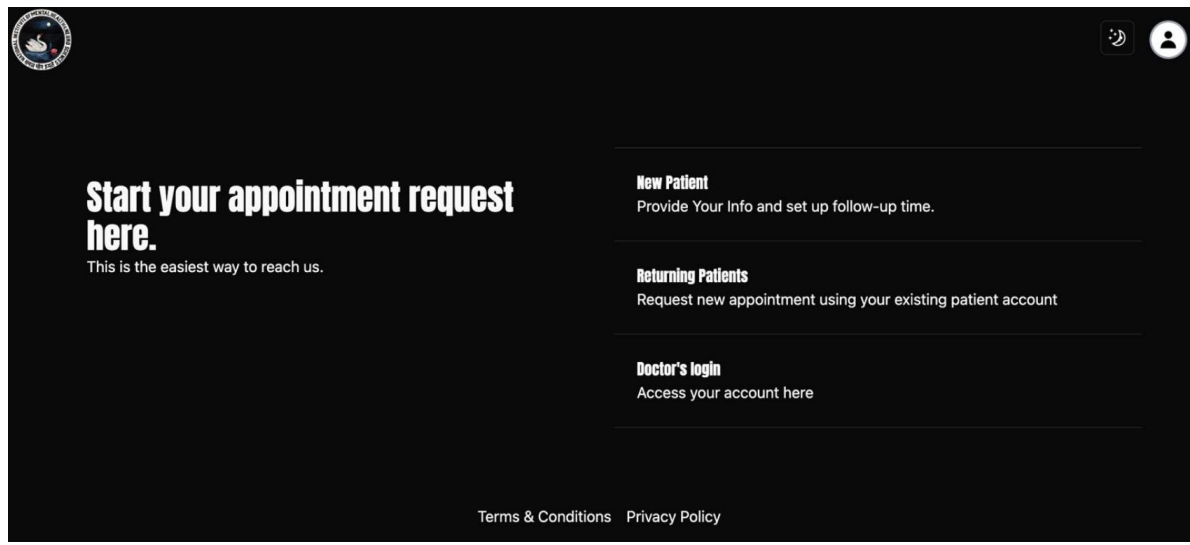


Fig 18: Front page (Dark Mode)

Fig 19 and Fig 20 basically represent the Sign-Up page of the website in Light and Dark Mode respectively. **Account Registration** page for a web portal, where users are prompted to enter their 10-digit phone number for registration. Once the number is entered, users can click on the "Send OTP" button to receive a one-time password for verification. The layout is clean and minimalistic, ensuring an easy and straightforward onboarding process for new users.

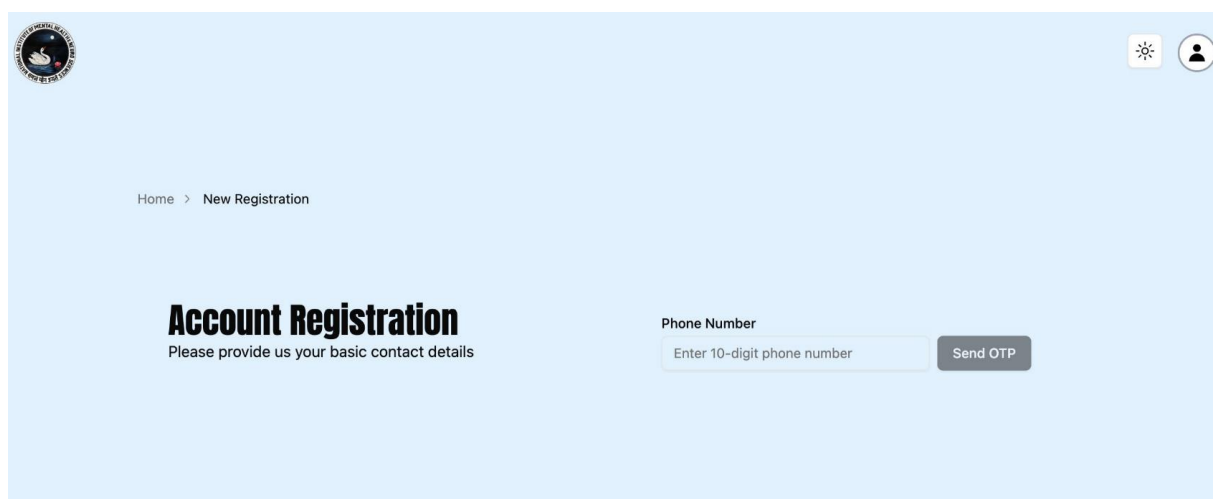


Fig 19: Sign Up page (Light Mode)

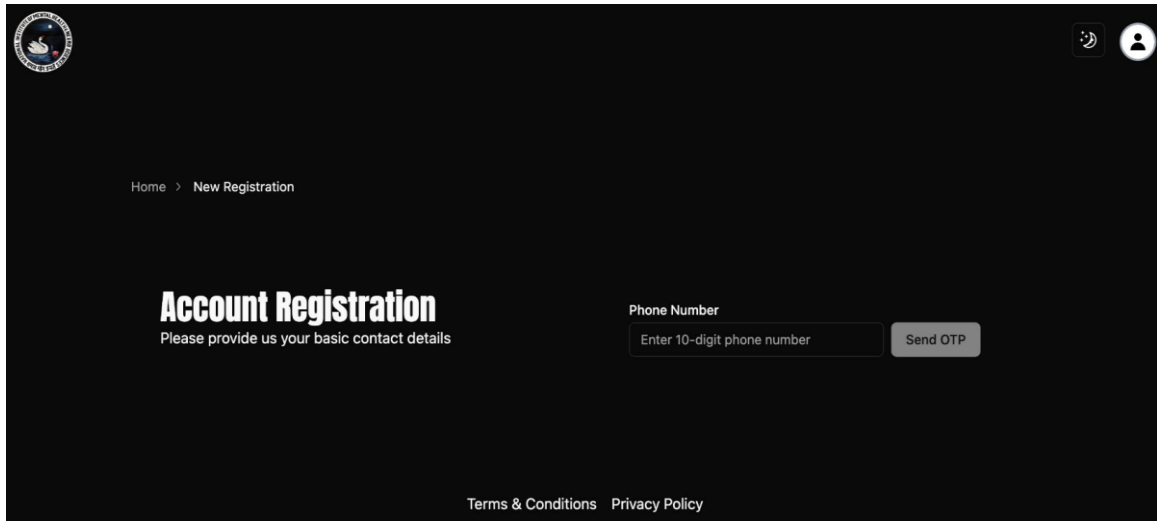


Fig 20: Sign Up page (Dark Mode)

Fig 21 and Fig 22 basically represent the Sign-Up page of the website in Light and Dark Mode respectively verifying the OTP (MFA). The primary benefit of using an OTP (One-Time Password) is enhanced security. It ensures that only the legitimate user can access the account or complete sensitive transactions, as the OTP is sent to a registered phone number or email and is valid for a short duration, reducing the risk of unauthorized access or fraudulent activities.

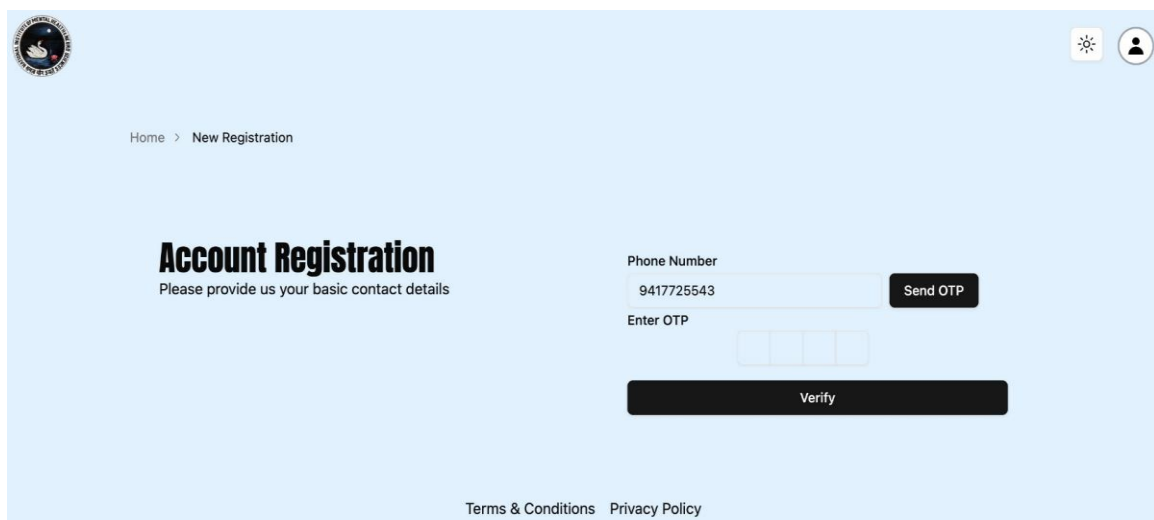
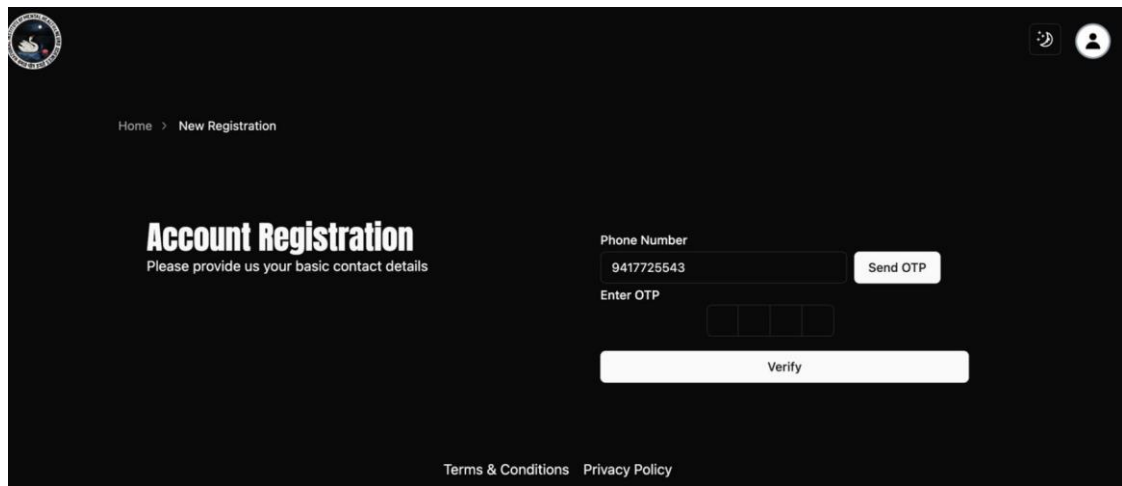


Fig 21: Sign Up page (Light Mode)



Home > New Registration

Account Registration

Please provide us your basic contact details

Phone Number
9417725543 Send OTP

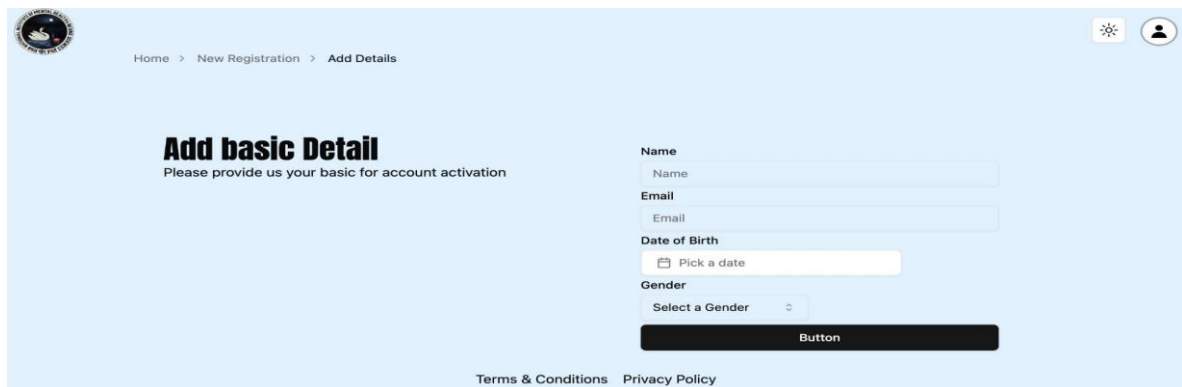
Enter OTP
[][][][]

Verify

[Terms & Conditions](#) [Privacy Policy](#)

Fig 22: Sign Up page (Light Mode)

Fig 23 and Fig 24 basically represent the login page of the website in Light and Dark Mode respectively.



Home > New Registration > Add Details

Add basic Detail

Please provide us your basic for account activation

Name
Name

Email
Email

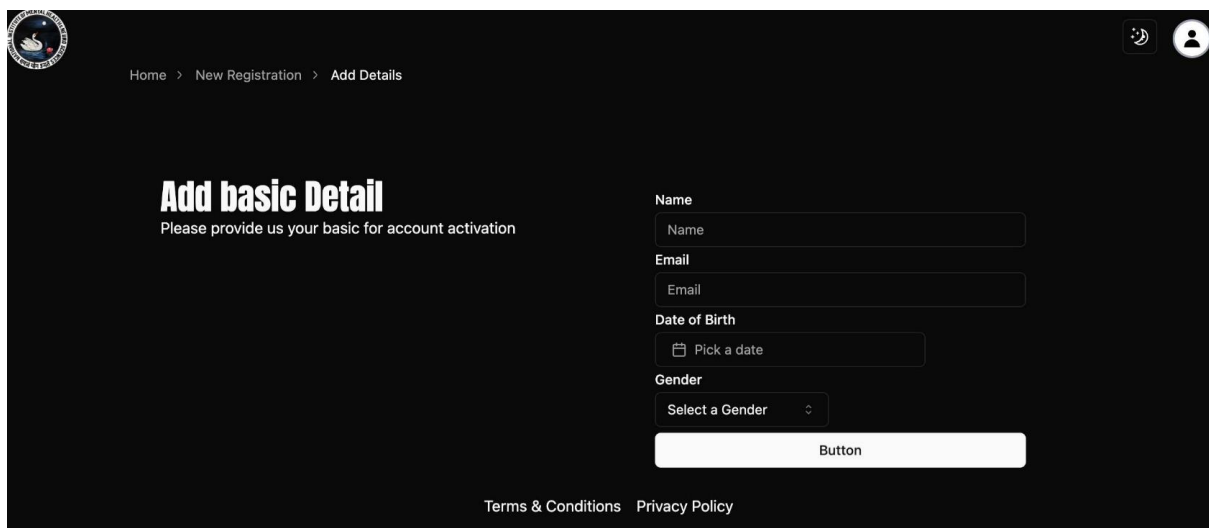
Date of Birth
Pick a date

Gender
Select a Gender

Button

[Terms & Conditions](#) [Privacy Policy](#)

Fig 23: Welcome page (Light Mode)



Home > New Registration > Add Details

Add basic Detail

Please provide us your basic for account activation

Name
Name

Email
Email

Date of Birth
Pick a date

Gender
Select a Gender

Button

[Terms & Conditions](#) [Privacy Policy](#)

Fig 24: Welcome page (Dark Mode)

The Fig 25 shows a web-based **Appointment Management** dashboard, where users can view and manage their scheduled appointments. The displayed appointment details include the assigned doctor, status, timing, day, and date of the appointment, along with the date of creation. Users can also schedule a new appointment using the button on the top right, making it easy to track and organize upcoming visits.

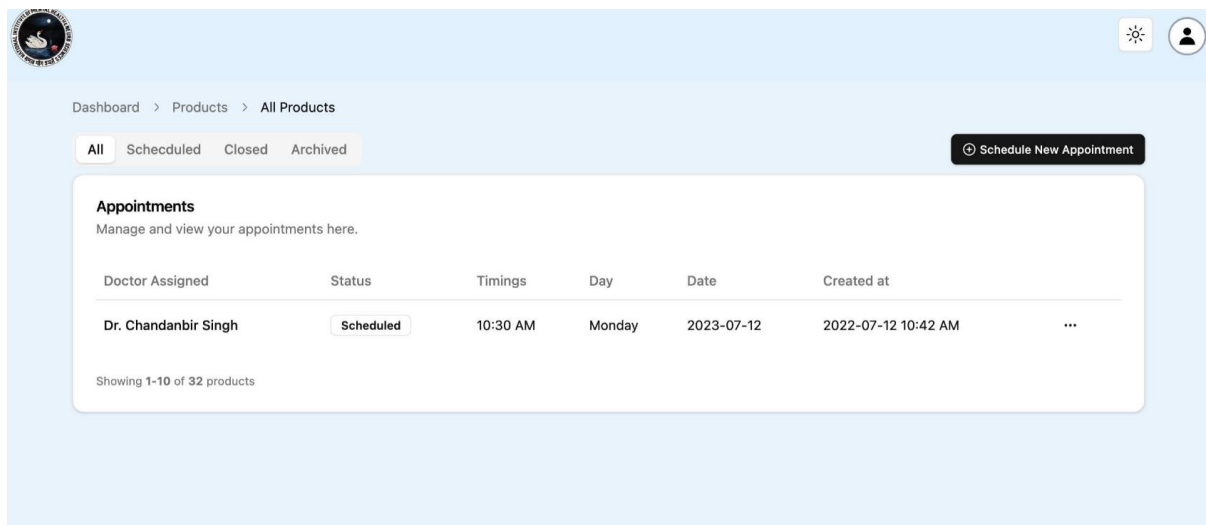


Fig 25: Appointment Management Dashboard

8 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.).