

SMARTHEAL

A Report

*Submitted in partial fulfillment of the
Requirements for the completion of*

THEME BASED PROJECT

**BACHELOR OF ENGINEERING
IN
INFORMATION TECHNOLOGY
By**

**Anoohya Narsingi 1602-21-737-070
Bhavani Lavanga 1602-21-737-076
Mirza Rafiq Ahmed 1602-21-737-095**

Under the guidance of
**Mrs. B. Leelavathy
Assistant Professor**



Department of Information Technology

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE.

(Affiliated to Osmania University and Approved by AICTE)

Ibrahim Bagh, Hyderabad-31

2024

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University and Approved by AICTE)

Ibrahim Bagh, Hyderabad-31

Department of Information Technology



DECLARATION BY CANDIDATES

We, **ANOOHYA NARSINGI, BHAVANI LAVANGA and MIRZA RAFIQ AHMED**, bearing hall ticket numbers, **1602-21-737-070, 1602-21-737-076 and 1602-21-737-095**, hereby declare that the project report entitled “**SMARTHEAL**” under the guidance of **Mrs. B. Leelavathy, Assistant Professor**, Department of Information Technology, Vasavi College of Engineering, Hyderabad, is submitted in partial fulfillment of the requirement for the completion of Theme-based project, VI semester, Bachelor of Engineering in Information Technology.

This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any other institutes.

Anoohya Narsingi 1602-21-737-070

Bhavani Lavanga 1602-21-737-076

Mirza Rafiq Ahmed 1602-21-737-095

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University and Approved by AICTE)

Ibrahim Bagh, Hyderabad-31

Department of Information Technology



BONAFIDE CERTIFICATE

This is to certify that the project entitled “**SMARTHEAL**” being submitted by **ANOOHYA NARSINGI , BHAVANI LAVANGA and MIRZA RAFIQ AHMED**, bearing hall ticket numbers, **1602-21-737-070, 1602-21-737-076 and 1602-21-737-095**, in partial fulfillment of the requirements for the completion of Theme-based project of Bachelor of Engineering in Information Technology is a record of bonafide work carried out by them under my guidance.

Mrs. B. Leelavathy,
Assistant Professor

External Examiner

Dr. K. Ram Mohan Rao
Professor, HOD IT

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of the project would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them.

It is with immense pleasure that we would like to take the opportunity to express our humble gratitude to **Mrs. B. Leelavathy, Assistant Professor, Information Technology** under whom we executed this project. Her constant guidance and willingness to share their vast knowledge made us understand this project and its manifestations in great depths and helped us to complete the assigned tasks.

We are very much thankful to **Dr. K. Ram Mohan Rao, Professor and HOD, Information Technology**, for his kind support and for providing necessary facilities to carry out the work.

We wish to convey our special thanks to Dr. S. V. Ramana, **Principal of Vasavi College of Engineering** for giving the required information in doing my project work. Not to forget, we thank all other faculty and non-teaching staff, and my friends who had directly or indirectly helped and supported me in completing my project in time.

We also express our sincere thanks to the Management for providing excellent facilities. Finally, we wish to convey our gratitude to our family who fostered all the facilities that we need.

ABSTRACT

SmartHeal emerges as a pivotal solution in the landscape of healthcare management, offering an unparalleled fusion of advanced technology and patient-centric care delivery. At its core, SmartHeal embodies a commitment to efficiency and empathy, striving to simplify administrative tasks while enhancing the overall patient experience. Through its multifaceted features and intuitive design, SmartHeal empowers healthcare administrators to navigate complex workflows with ease. With features such as profile management, scheduling, and performance tracking, SmartHeal enables administrators to efficiently manage their medical staff, ensuring optimal coverage and patient access to care. Appointment scheduling is another cornerstone of SmartHeal's platform, offering patients and providers alike a streamlined and intuitive experience. Through online booking, appointment reminders, and real-time availability updates, SmartHeal simplifies the scheduling process, reducing wait times and enhancing patient satisfaction. Moreover, SmartHeal prioritizes effective communication between patients and healthcare providers, recognizing its crucial role in delivering high-quality care. The platform's patient communication tools enable secure messaging, appointment reminders, and access to medical records, fostering a collaborative and informed approach to healthcare delivery. In addition to its patient-facing features, SmartHeal also addresses the administrative burden faced by healthcare facilities through automation and customization. By automating routine administrative tasks such as billing, documentation, and inventory management, SmartHeal enables staff to focus their time and energy on delivering exceptional patient care.

LIST OF FIGURES

Fig No.	Description	Page No.
4.1.1	Architecture Diagram of Smart Heal	13
4.1.2	Use Case Diagram of Smart Heal	14
4.2.1.1	Dashboard.jsx	20
4.2.1.2	AddNewDoctor.jsx	21
4.2.1.3	AddNewAdmin.jsx	21
4.2.1.4	Messages.jsx	22
4.2.1.5	dbConnection.jsx	22
4.2.1.6	userRouter.jsx	23
4.2.1.7	userSchema.jsx	23
4.2.1.8	App.jsx	24
4.2.1.9	Main.jsx	24
4.2.1.10	Gitlab directory structure	25
6.1	Front Page 1	34
6.2	Front Page 2	35
6.3	Appointment Booking	35
6.4	About Us Page	36
6.5	Admin Dashboard	36
6.6	Add New Admin	37
6.7	Register a New Doctor	37
6.8	Appointments	38
6.9	Messages	38

LIST OF TABLES

Table No.	Description	Page No.
4.1.2.1	Use Case 1	15
4.1.2.2	Use Case 2	15
4.1.2.3	Use Case 3	16
4.1.2.4	Use Case 4	16
4.1.2.5	Use Case 5	17
4.1.2.6	Use Case 6	17
4.1.2.7	Use Case 7	18
4.1.2.8	Use Case 8	18
4.1.2.9	Use Case 9	19

TABLE OF CONTENTS

1. INTRODUCTION	9
1.1 Overview	9
1.2 Problem Statement	9
1.3 Motivation of theme & title	9
2. LITERATURE SURVEY	10
3. EXISTING SYSTEM	11
4. PROPOSED SOLUTION	12
4.1. System Design	13
4.1.1 Architecture Diagram	13
4.1.2 Use-Case Diagram	14
4.1.2.1 Use-case descriptions	15
4.2 Functional Modules	20
4.2.1 Pseudo code & Screenshots	20
5. EXPERIMENTAL SETUP & IMPLEMENTATION	33
5.1 System Specifications	33
5.1.1 Hardware Requirements	33
5.1.2 Software Requirements	33
5.2 Methodology	33
6. RESULTS	34
7. CONCLUSION & FUTURE SCOPE	39
8. REFERENCES	41

1. INTRODUCTION

1.1 OVERVIEW

SmartHeal is revolutionizing healthcare management with a comprehensive platform that integrates advanced digital technologies to optimize various administrative and clinical processes. By ensuring efficient doctor availability, streamlining appointment scheduling, and enhancing administrative workflows, *SmartHeal* significantly boosts operational efficiency within healthcare facilities. The platform aims to reduce patient wait times, improve resource allocation, and elevate the overall quality of care delivered. Through its innovative approach, *SmartHeal* aspires to create a seamless, efficient, and patient-centric healthcare experience.

1.2 PROBLEM STATEMENT

SmartHeal aims to modernize healthcare management by offering a comprehensive platform that optimizes doctor availability, appointment allocation, and administrative processes. Through seamless integration of digital technologies, *SmartHeal* aims to enhance operational efficiency, minimize patient wait times, and elevate the quality of patient care.

1.3 MOTIVATION OF THEME & TITLE

The motivation behind *SmartHeal* stems from a deep-seated desire to revolutionize the healthcare industry by addressing the pressing challenges faced by healthcare facilities worldwide. With escalating demands for efficiency, patient satisfaction, and resource optimization, traditional healthcare management methods have become increasingly inadequate. *SmartHeal* seeks to bridge this gap by harnessing the power of technology to streamline administrative processes, improve communication channels, and enhance overall operational efficiency.

2. LITERATURE SURVEY

Various studies have explored the implementation and benefits of online booking systems and other digital innovations in the healthcare sector. Below are summaries of some key works in this area:

In 2000, **Rohleder and Klassen [1]** investigated the utilization and impact of online booking systems within the healthcare sector in their seminal book, "The Use of Online Booking Systems in Healthcare." This comprehensive analysis explores the adoption of digital platforms for appointment scheduling and management, highlighting how these systems enhance patient satisfaction and operational efficiency by reducing wait times and administrative burdens. The authors also address challenges such as data security, patient privacy, and equitable access, providing strategic recommendations for the effective integration of online booking systems in healthcare settings.

In 2012, **Benson [2]** in his book "Principles of Health Interoperability HL7 and SNOMED" examines the critical role of interoperability standards like HL7 and SNOMED in healthcare. Benson emphasizes the importance of standardized communication protocols to ensure seamless data exchange among different healthcare systems. This work highlights how interoperability can improve the quality and continuity of patient care, offering a foundational understanding of how these standards can be leveraged to facilitate accurate and comprehensive health information sharing.

In 2013, **Bove et al. [3]** evaluated the implementation of secure messaging systems in healthcare in their study, "Evaluation of Secure Messaging Between Patients and Healthcare Providers." The authors highlight the benefits of secure messaging, such as improved patient engagement, better management of chronic conditions, and increased access to care. However, they also discuss challenges like ensuring data security, maintaining patient privacy, and integrating messaging systems into existing healthcare workflows. This study provides a nuanced perspective on the feasibility and impact of secure messaging in healthcare.

In 2016, **Liu and Zhang [4]** discussed the customization of healthcare systems to meet specific institutional needs and improve operational efficiency in their work, "Customization of Healthcare Systems for Improving Efficiency." Their research emphasizes the importance of tailoring healthcare IT solutions to align with organizational workflows, patient demographics, and clinical practices. By presenting case studies and empirical evidence, Liu and Zhang demonstrate how customized healthcare systems can lead to enhanced patient outcomes, reduced operational costs, and increased overall efficiency in healthcare delivery.

In 2006, **Shortliffe and Cimino [5]** provided a comprehensive overview of the application of computer technology in healthcare and biomedicine in their book, "Biomedical Informatics: Computer Applications in Health Care and Biomedicine." This work covers a wide range of topics, including electronic health records, clinical decision support systems, telemedicine, and bioinformatics. The authors illustrate the transformative potential of biomedical informatics in improving patient care, advancing medical research, and optimizing healthcare operations, serving as a crucial resource for understanding the integration of technology in modern healthcare.

3. EXISTING SYSTEM

- In the current healthcare system, patient appointments are typically managed manually or through basic scheduling software.
- Patient waiting times are often long due to inefficient appointment scheduling and lack of real-time doctor availability tracking.
- Doctors' schedules may not be optimized, leading to either overbooking or underutilization of their time.
- Patients often have to wait unnecessarily for tests and procedures due to inefficient coordination between medical staff and resources.

4. PROPOSED SOLUTION

- Implement a comprehensive digital platform that integrates advanced allocation with real-time tracking of doctor availability.
- Develop a user-friendly interface for patients to book appointments based on their convenience and doctors' availability.
- Analyze patient flow and allocate resources efficiently, reducing waiting times.
- Overall, the proposed solution aims to optimize doctor availability, reduce patient waiting times, and streamline the entire healthcare process through smart digital solutions

4.1 SYSTEM DESIGN

4.1.1 ARCHITECTURE DIAGRAM

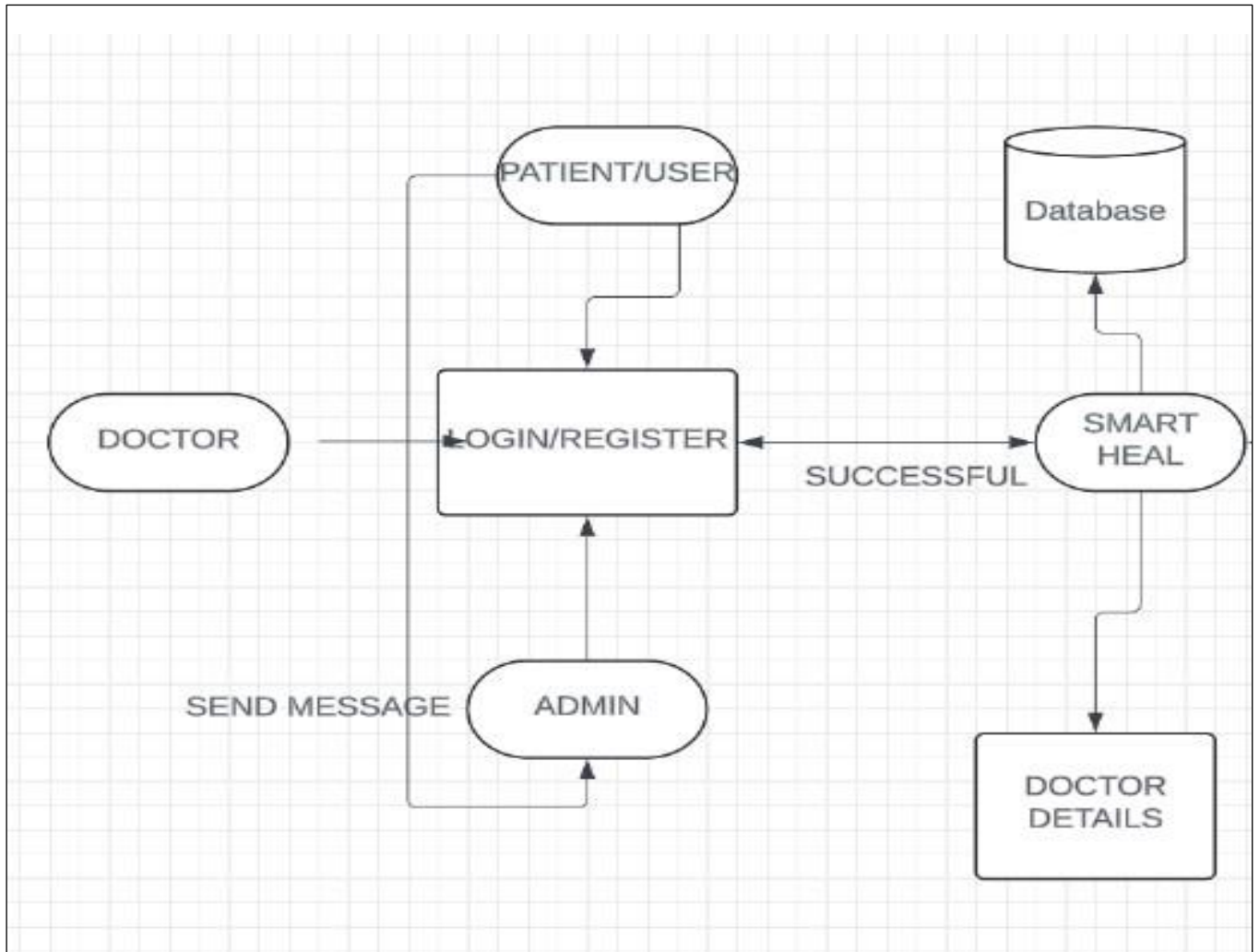


Fig 4.1.1 : Architecture Diagram of Smart Heal

The "Smart Heal" system architecture involves patients, doctors, and admins interacting through a central login/register module. Successful logins lead to the core "Smart Heal" system, which interfaces with a database for managing doctor details and facilitating communication, ensuring efficient healthcare management.

4.1.2 USE – CASE DIAGRAM

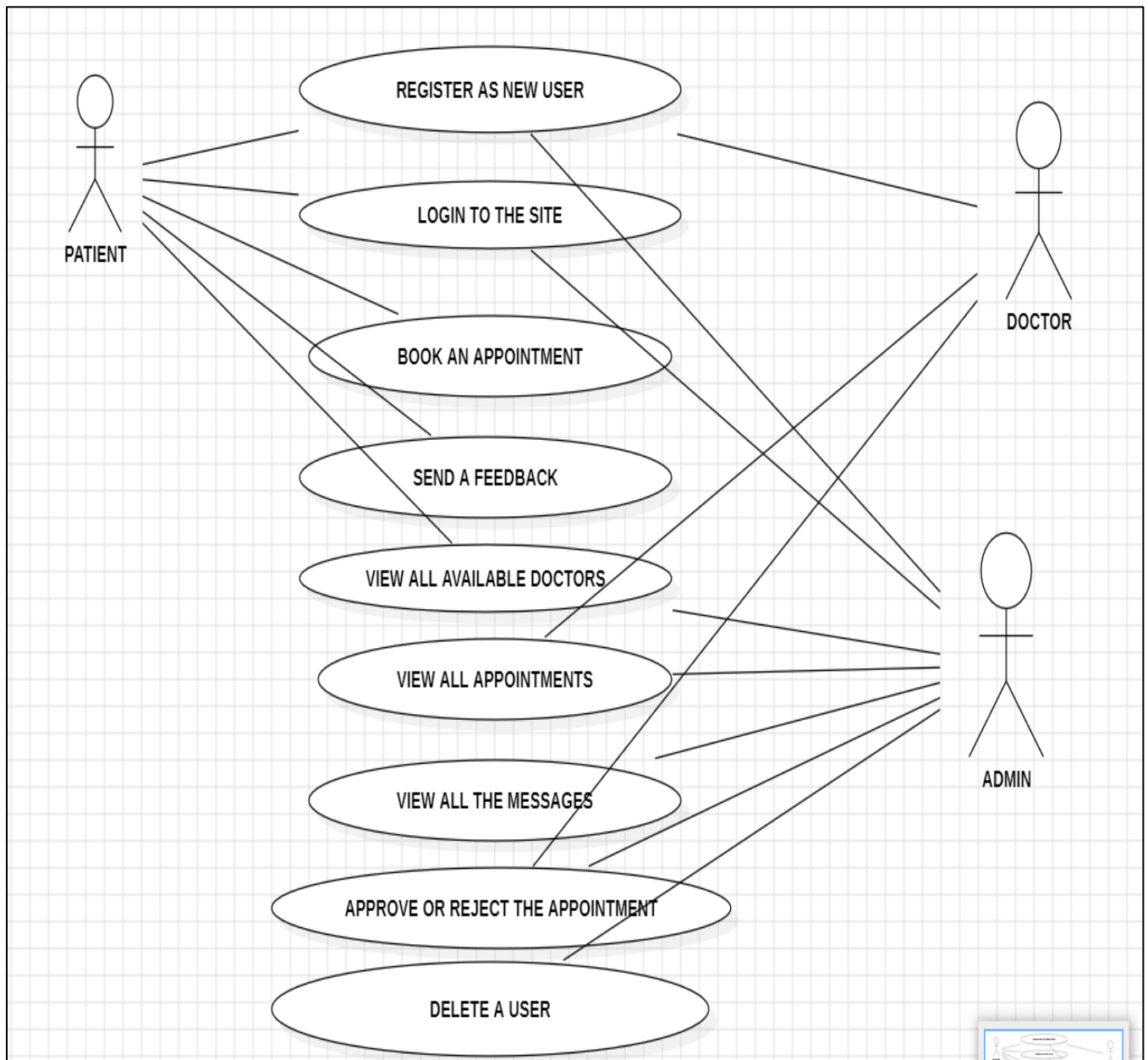


Fig 4.1.2 : Use Case Diagram of Smart Heal

The use case diagram of Smart Heal highlights the primary functionalities available to different users. Patients can register, log in, book appointments, send feedback, and view lists of doctors, appointments, and messages. Doctors are able to approve or reject appointment requests. Admins have comprehensive control, including viewing all appointments and messages, approving or rejecting appointments, and deleting users, thus maintaining the system's overall integrity and efficiency.

4.1.2.1 USE CASE DESCRIPTION

Use Case ID: UC01

Name: Register as New User

Actors: Patient, Admin

Description: Allows a new patient or admin to create an account on the system by providing necessary details such as name, email, password, and other relevant information.

Patient/Admin	System
1. The Patient or Admin inputs their details to be shared.	
2. The Patient or Admin submits the registration form.	
	3. The system stores the new user details and creates an account

Table 4.1.2.1

Use Case ID: UC02

Name: Login to the Site

Actors: Patient, Admin

Description: Enables an existing patient or admin to log into the system using their registered email and password.

Patient/Admin	System
1. The Patient or Admin enters their email and password.	
	2. The system verifies credentials.
3. The Patient or Admin gains access to their respective dashboard.	

Table 4.1.2.2

Use Case ID: UC03**Name: Book an Appointment****Actors: Patient**

Description: Allows a patient to book an appointment with a doctor by selecting a date, time, and doctor from the available slots.

Patient	System
1. The patient selects a doctor from the list.	
2. The patient chooses an available date and time.	
3. The patient confirms the appointment booking.	
	4. The system schedules the appointment and notifies the doctor.

Table 4.1.2.3**Use Case ID: UC04****Name: Send a Feedback****Actors: Patient**

Description: Enables a patient to send feedback regarding the service, doctors, or any other aspect of the system.

Patient	System
1. The patient navigates to the feedback section.	
2. The patient enters their feedback and submits it.	
	3. The system stores the feedback for review.

Table 4.1.2.4

Use Case ID: UC05**Name: View All Available Doctors****Actors: Patient, Admin**

Description: Allows users to see a list of all doctors available in the system along with their details such as specialization, availability, and contact information.

Patient	System
1. The patient or admin requests to view all available doctors	
	2. The system retrieves and displays the list of doctors along with their details.

Table 4.1.2.5**Use Case ID: UC06****Name: View All Appointments****Actors: Patient, Doctor, Admin**

Description: Allows users to view all their appointments. Patients can see their booked appointments, doctors can see their scheduled appointments, and admins can see all appointments in the system.

Patient/Doctor/Admin	System
1. The patient/doctor/admin requests to view their appointments.	
	2. The system retrieves and displays the list of appointments based on the user role.

Table 4.1.2.6

Use Case ID: UC07

Name: View All the Messages

Actors: Admin

Description: Enables admin to view all messages, including feedback and communications between patients and doctors.

Admin	System
1. The admin requests to view all messages.	
	2. The system retrieves and displays all messages,including feedback and enables communication between patient and doctor.

Table 4.1.2.7

Use Case ID: UC08

Name: Approve or Reject the Appointment

Actors: Doctor, Admin

Description: Allows doctors and admins to approve or reject appointment requests made by patients.

Doctor/Admin	System
1.The Doctor or Admin reviews the appointment requests.	
2.The Doctor or Admin approves or rejects the appointment request.	
	3. The Doctor or Admin approves or rejects the appointment request.

Table 4.1.2.8

Use Case ID: UC09

Name: Delete a User

Actors: Admin

Description: Enables an admin to delete a user from the system. This could be a patient or another admin.

Admin	System
1. The Admin selects the user account to be deleted.	
2. The Admin confirms the deletion.	
	3. The system removes the user account from the system.

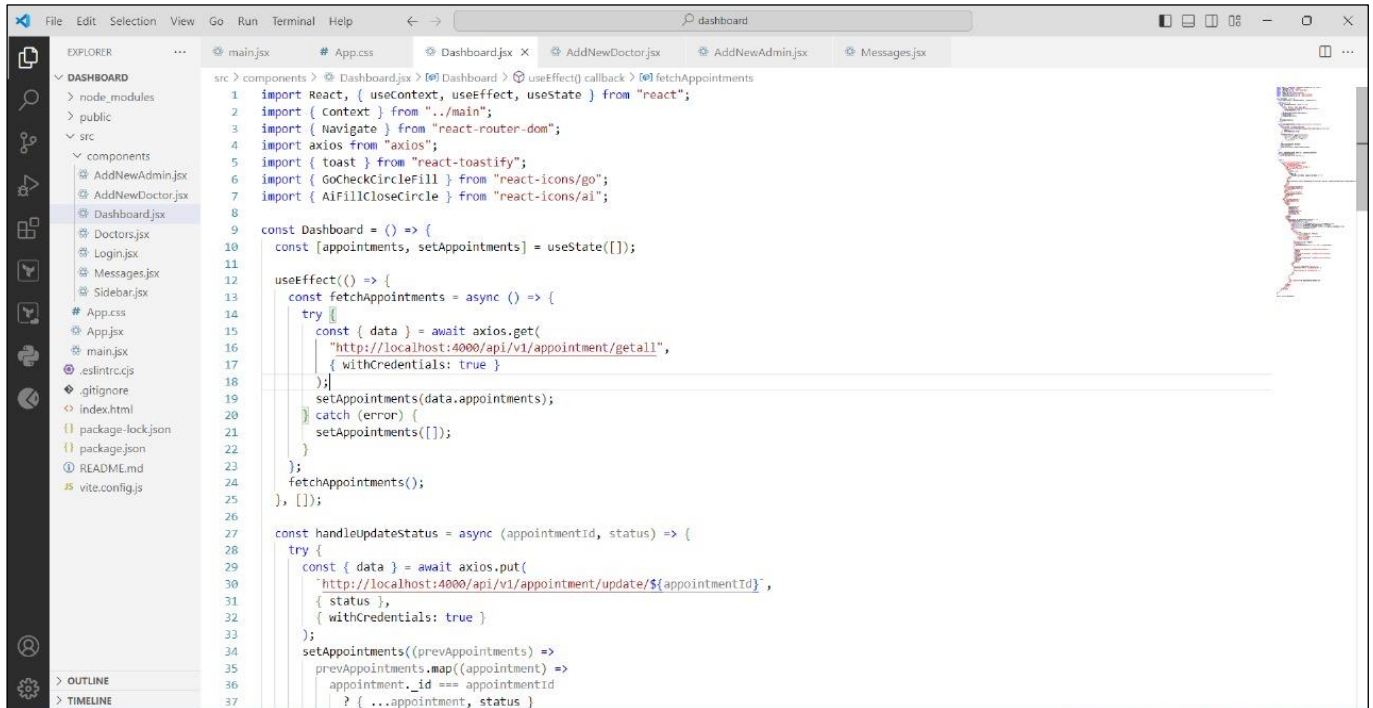
Table 4.1.2.9

4.2 FUNCTIONAL MODULES

4.2.1 SCREEN SHOTS & PSUEDO CODE

4.2.1 SCREENSHOTS :

1. DASHBOARD:



```
1 import React, { useContext, useEffect, useState } from "react";
2 import { Navigate } from "react-router-dom";
3 import axios from "axios";
4 import { toast } from "react-toastify";
5 import { GoCheckCircleFill } from "react-icons/go";
6 import { AiFillCloseCircle } from "react-icons/ai";
7
8
9 const Dashboard = () => {
10   const [appointments, setAppointments] = useState([]);
11
12   useEffect(() => {
13     const fetchAppointments = async () => {
14       try {
15         const { data } = await axios.get(
16           "http://localhost:4000/api/v1/appointment/getall",
17           { withCredentials: true }
18         );
19         setAppointments(data.appointments);
20       } catch (error) {
21         setAppointments([]);
22       }
23     };
24     fetchAppointments();
25   }, []);
26
27   const handleUpdateStatus = async (appointmentId, status) => {
28     try {
29       const { data } = await axios.put(
30         "http://localhost:4000/api/v1/appointment/update/${appointmentId}",
31         { status },
32         { withCredentials: true }
33       );
34       setAppointments((prevAppointments) =>
35         prevAppointments.map((appointment) =>
36           appointment._id === appointmentId
37             ? { ...appointment, status }
```

Fig 4.2.1.1 Dashboard.jsx

The Dashboard component fetches and displays appointment data, allowing admins to manage appointment statuses and view summary statistics, redirecting unauthenticated users to the login page.

```

1 import React, { useContext, useState } from "react";
2 import { Context } from "../main";
3 import { Navigate, useNavigate } from "react-router-dom";
4 import { toast } from "react-toastify";
5 import axios from "axios";
6
7 const AddNewAdmin = () => {
8   const { isAuthenticated, setIsAuthenticated } = useContext(Context);
9
10  const [firstName, setFirstName] = useState("");
11  const [lastName, setLastName] = useState("");
12  const [email, setEmail] = useState("");
13  const [phone, setPhone] = useState("");
14  const [nic, setNic] = useState("");
15  const [dob, setDob] = useState("");
16  const [gender, setGender] = useState("");
17  const [password, setPassword] = useState("");
18
19  const navigateTo = useNavigate();
20
21  const handleAddNewAdmin = async (e) => {
22    e.preventDefault();
23    try {
24      await axios
25        .post(
26          "http://localhost:4000/api/v1/user/admin/addnew",
27          {
28            firstName, lastName, email, phone, nic, dob, gender, password,
29          },
30          {
31            withCredentials: true,
32            headers: { "Content-Type": "application/json" },
33          }
34        )
35        .then((res) => {
36          toast.success(res.data.message);
37          setIsAuthenticated(true);
38          navigateTo("/");
39          setFirstName("");
40        });
41    } catch (error) {
42      toast.error(error.message);
43    }
44  };
45
46  return (
47    <div>
48      <h3>Add New Admin</h3>
49      <form>
50        <input type="text" value={firstName} onChange={setFirstName}/>
51        <input type="text" value={lastName} onChange={setLastName}/>
52        <input type="text" value={email} onChange={setEmail}/>
53        <input type="text" value={phone} onChange={setPhone}/>
54        <input type="text" value={nic} onChange={setNic}/>
55        <input type="text" value={dob} onChange={setDob}/>
56        <input type="text" value={gender} onChange={setGender}/>
57        <input type="password" value={password} onChange={setPassword}/>
58        <button type="button" onClick={handleAddNewAdmin}>Add New Admin</button>
59      </form>
60    </div>
61  );
62};

```

Fig 4.2.1.2 AddNewAdmin.jsx

Add new Admin component enables authenticated users to add a new admin by submitting a form with personal details. It uses Axios to send data to the backend API and manages authentication and redirects using React Router.

```

1 import React, { useContext, useState } from "react";
2 import { Navigate, useNavigate } from "react-router-dom";
3 import { toast } from "react-toastify";
4 import { Context } from "../main";
5 import axios from "axios";
6
7 const AddNewDoctor = () => {
8   const { isAuthenticated, setIsAuthenticated } = useContext(Context);
9
10  const [firstName, setFirstName] = useState("");
11  const [lastName, setLastName] = useState("");
12  const [email, setEmail] = useState("");
13  const [phone, setPhone] = useState("");
14  const [nic, setNic] = useState("");
15  const [dob, setDob] = useState("");
16  const [gender, setGender] = useState("");
17  const [password, setPassword] = useState("");
18  const [doctorDepartment, setDoctorDepartment] = useState("");
19  const [docAvatar, setDocAvatar] = useState("");
20  const [docAvatarPreview, setDocAvatarPreview] = useState("");
21
22  const navigateTo = useNavigate();
23
24  const departmentsArray = [
25    "Pediatrics",
26    "Orthopedics",
27    "Cardiology",
28    "Neurology",
29    "Oncology",
30    "Radiology",
31    "Physical Therapy",
32    "Dermatology",
33    "ENT",
34  ];
35
36  const handleAvatar = (e) => {
37    const file = e.target.files[0];
38    if (file) {
39      const reader = new FileReader();
40      reader.onload = () => {
41        setDocAvatarPreview(reader.result);
42      };
43      reader.readAsDataURL(file);
44    }
45  };
46
47  const handleAddNewDoctor = async (e) => {
48    e.preventDefault();
49    try {
50      await axios
51        .post(
52          "http://localhost:4000/api/v1/user/doctor/addnew",
53          {
54            firstName, lastName, email, phone, nic, dob, gender, password,
55            doctorDepartment, docAvatar, docAvatarPreview,
56          },
57          {
58            withCredentials: true,
59            headers: { "Content-Type": "application/json" },
60          }
61        )
62        .then((res) => {
63          toast.success(res.data.message);
64          setIsAuthenticated(true);
65          navigateTo("/");
66          setFirstName("");
67          setLastName("");
68          setEmail("");
69          setPhone("");
70          setNic("");
71          setDob("");
72          setGender("");
73          setPassword("");
74          setDoctorDepartment("");
75          setDocAvatar("");
76          setDocAvatarPreview("");
77        });
78    } catch (error) {
79      toast.error(error.message);
80    }
81  };
82
83  return (
84    <div>
85      <h3>Add New Doctor</h3>
86      <form>
87        <input type="text" value={firstName} onChange={setFirstName}/>
88        <input type="text" value={lastName} onChange={setLastName}/>
89        <input type="text" value={email} onChange={setEmail}/>
90        <input type="text" value={phone} onChange={setPhone}/>
91        <input type="text" value={nic} onChange={setNic}/>
92        <input type="text" value={dob} onChange={setDob}/>
93        <input type="text" value={gender} onChange={setGender}/>
94        <input type="password" value={password} onChange={setPassword}/>
95        <input type="text" value={doctorDepartment} onChange={setDoctorDepartment}/>
96        <input type="text" value={docAvatar} onChange={setDocAvatar}/>
97        <input type="text" value={docAvatarPreview} onChange={setDocAvatarPreview}/>
98        <button type="button" onClick={handleAddNewDoctor}>Add New Doctor</button>
99      </form>
100    </div>
101  );
102};

```

Fig 4.2.1.3 AddNewDoctor.jsx

AddNewDoctor component facilitates authenticated users to register new doctors through a form submission, including avatar upload. It utilizes Axios to send FormData to the backend API, managing authentication and navigation with React Router.

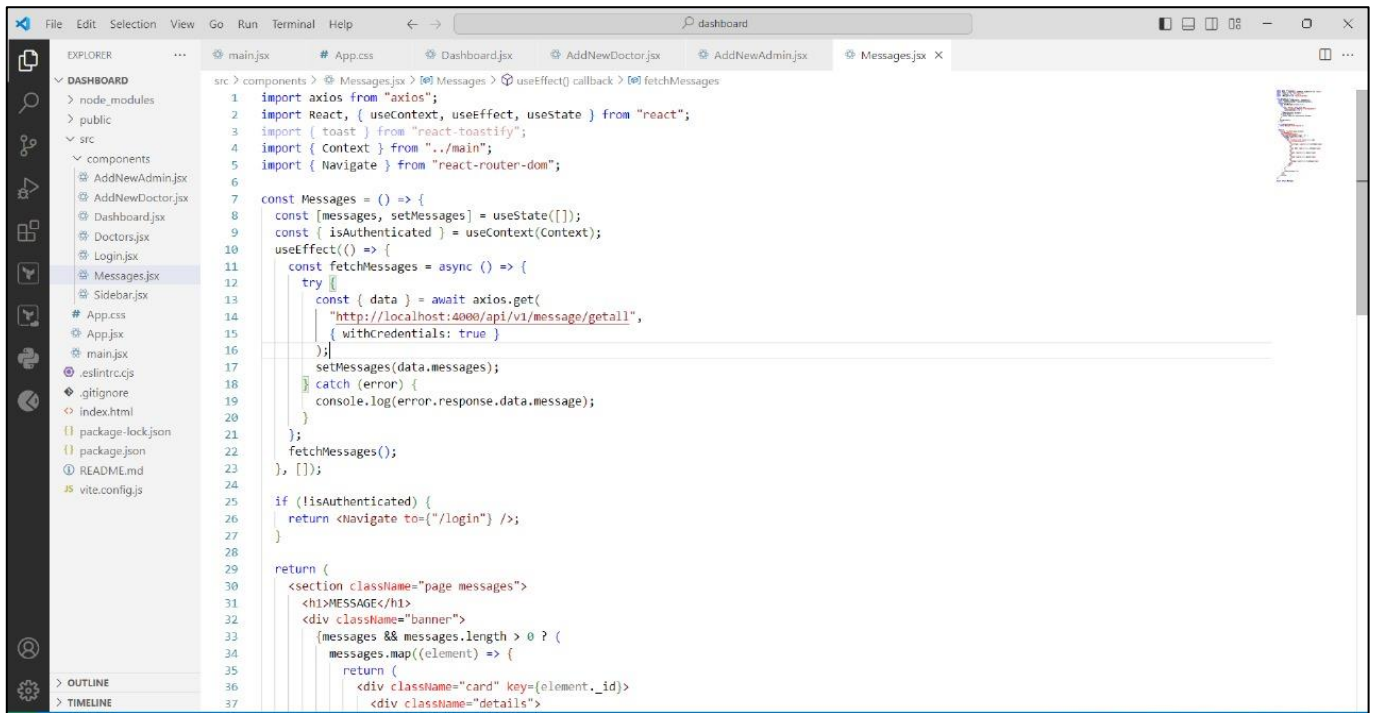


Fig 4.2.1.4 Messages.jsx

Messages component fetches and displays all messages from the backend API upon authentication. It uses Axios to retrieve data asynchronously, rendering each message with its details such as name, email, phone, and message content in a structured format.

2.BACKEND:

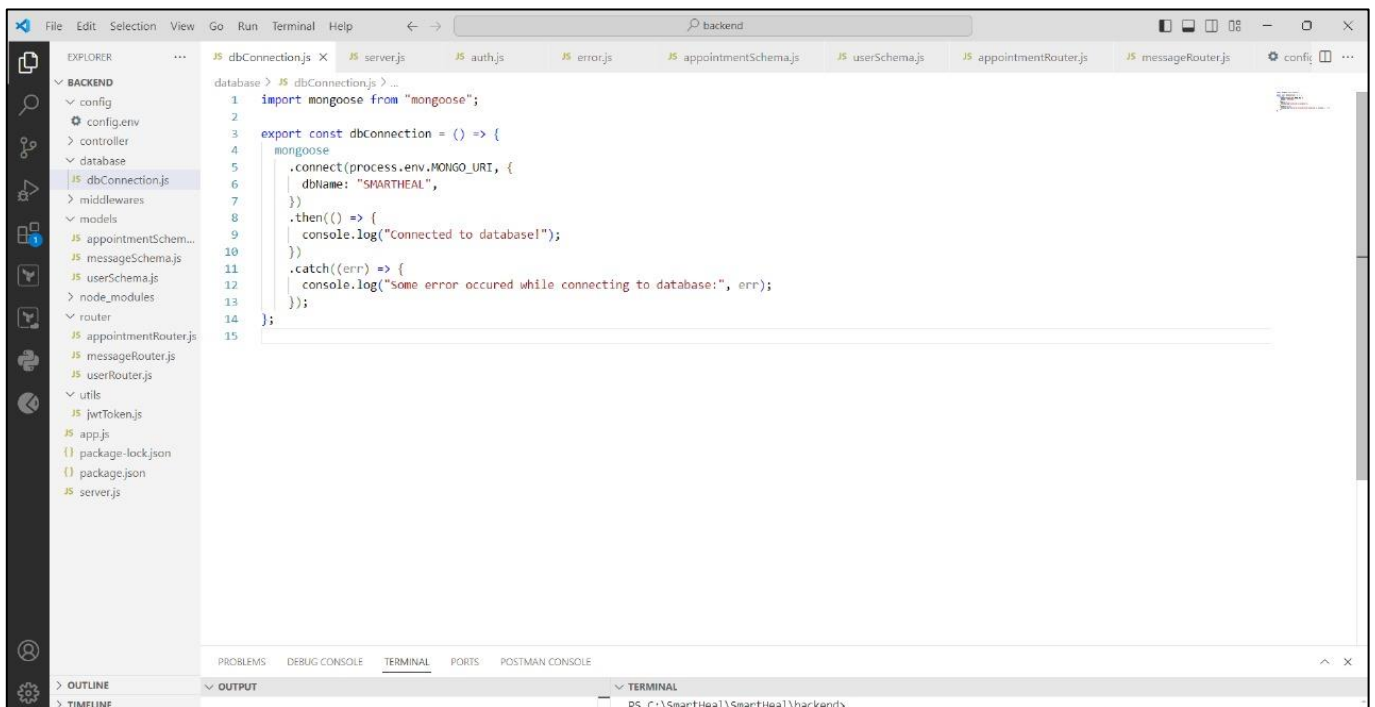


Fig 4.2.1.5 dbConnection.jsx

dbConnection function establishes a connection to a MongoDB database using Mongoose.

```

1 import express from "express";
2 import {
3   login,
4   addNewAdmin,
5   addNewDoctor,
6   patientRegister,
7   getAllDoctors,
8   getUserDetails,
9   logoutAdmin,
10  logoutPatient,
11 } from "../controller/userController.js";
12 import {
13   isAdminAuthenticated,
14   isPatientAuthenticated,
15 } from "../middlewares/auth.js";
16
17 const router = express.Router();
18 router.post("/patient/register", patientRegister);
19 router.post("/login", login);
20 router.post("/admin/addnew", addNewAdmin);
21 router.get("/doctors", getAllDoctors);
22 router.get("/admin/me", isAdminAuthenticated, getUserDetails);
23 router.get("/patient/me", isPatientAuthenticated, getUserDetails);
24 router.get("/admin/logout", isAdminAuthenticated, logoutAdmin);
25 router.get("/patient/logout", isPatientAuthenticated, logoutPatient);
26 router.post("/doctor/addnew", isAdminAuthenticated, addNewDoctor);
27 export default router;
28

```

Fig 4.2.1.6 userRouter.jsx

Express router for user operations: patient registration, login, admin tasks (add new admin, doctors), fetching doctors, and managing user sessions with logout endpoints. Authentication middleware ensures access control for admin and patient-specific routes.

```

1 import mongoose from "mongoose";
2 import validator from "validator";
3 import bcrypt from "bcrypt";
4 import jwt from "jsonwebtoken";
5
6 const userSchema = new mongoose.Schema({
7   firstName: {
8     type: String,
9     required: [true, "First Name Is Required!"],
10    minLength: [3, "First Name Must Contain At Least 3 Characters!"],
11  },
12  lastName: {
13    type: String,
14    required: [true, "Last Name Is Required!"],
15    minLength: [3, "Last Name Must Contain At Least 3 Characters!"],
16  },
17  email: {
18    type: String,
19    required: [true, "Email Is Required!"],
20    validate: [validator.isEmail, "Provide A Valid Email!"],
21  },
22  phone: {
23    type: String,
24    required: [true, "Phone Is Required!"],
25    minLength: [10, "Phone Number Must Contain Exact 10 Digits!"],
26    maxLength: [10, "Phone Number Must Contain Exact 10 Digits!"],
27  },
28  nic: {
29    type: String,
30    required: [true, "NIC Is Required!"],
31    minLength: [13, "NIC Must Contain Only 13 Digits!"],
32    maxLength: [13, "NIC Must Contain Only 13 Digits!"],
33  },
34 },
35 );
36

```

Fig 4.2.1.7 userSchema.jsx

Mongoose schema for defining user attributes (name, email, phone, NIC, DOB, gender, password, role, doctor department, avatar) with methods for password hashing/validation and JWT generation.

3.FRONTEND:

```
1 import React, { useContext, useEffect } from 'react';
2 import Navbar from './components/Navbar';
3 import Footer from './components/Footer';
4 import './App.css';
5 import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
6 import Home from './Pages.jsx/Home';
7 import Appointment from './Pages.jsx/Appointment';
8 import AboutUs from './Pages.jsx/AboutUs';
9 import Register from './Pages.jsx/Register';
10 import Login from './Pages.jsx/Login';
11 import { ToastContainer } from 'react-toastify';
12 import 'react-toastify/dist/ReactToastify.css';
13 import { Context } from './main';
14 import axios from 'axios';
15
16 const App = () => {
17   const { isAuthenticated, setIsAuthenticated, setUser } = useContext(Context);
18
19   useEffect(() => {
20     const fetchUser = async () => {
21       try {
22         const response = await axios.get(
23           "http://localhost:4000/api/v1/user/patient/me",
24           { withCredentials: true }
25         );
26         setIsAuthenticated(true);
27         setUser(response.data.user);
28       } catch (error) {
29         setIsAuthenticated(false);
30         setUser({});
31       }
32     };
33     fetchUser();
34   });
35 }
```

Fig 4.2.1.8 App.jsx

The App component sets up routing with react-router-dom, manages user authentication, and includes common layout components (Navbar, Footer). It displays pages for Home, Appointment, About Us, Register, and Login, and uses react-toastify for notifications.

```
1 import React, { createContext, useState } from 'react'
2 import ReactDOM from 'react-dom/client'
3 import App from './App.jsx'
4 export const Context = createContext({ isAuthenticated: false });
5 const AppWrapper = () => {
6   const [isAuthenticated, setIsAuthenticated] = useState(false);
7   const [user, setUser] = useState({});
8
9   return(
10     <Context.Provider value={{isAuthenticated, setIsAuthenticated,user, setUser}}>
11       <App/>
12     </Context.Provider>
13   );
14 }
15
16 ReactDOM.createRoot(document.getElementById('root')).render(
17   <React.StrictMode>
18     <AppWrapper/>
19   </React.StrictMode>,
20 )
```

Fig 4.2.1.9 main.jsx

The main.jsx file initializes the React app. It sets up the Context provider with authentication states (isAuthenticated, setIsAuthenticated, user, setUser) and wraps the App component within this provider.

Directory Structure:

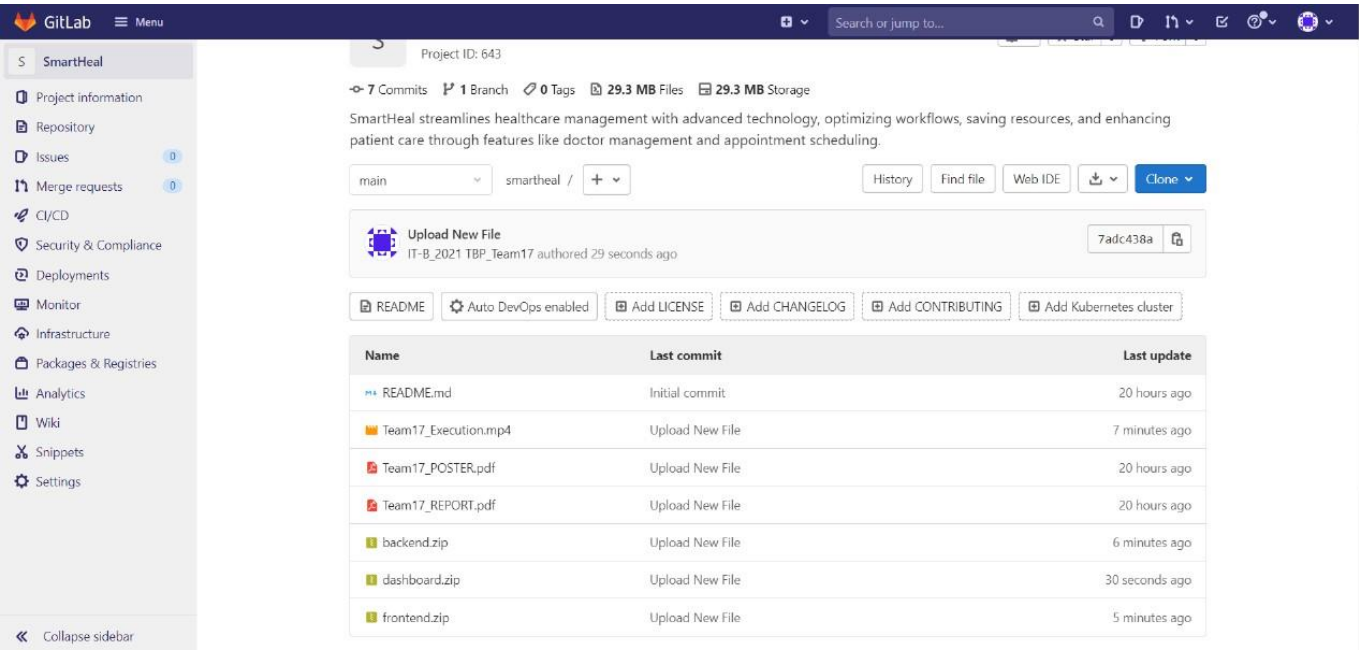


Fig 4.2.1.10 git lab directory structure

Gitlab link- https://gitlab.vce.ac.in/IT-B_2021_TBP_Team17/smartheal

4.2.1 PSUEDO CODE:

1. FRONT END:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite + React</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

CSS

```
:root {
  font-family: Inter, system-ui, Avenir, Helvetica, Arial, sans-serif;
  line-height: 1.5;
  font-weight: 400;

  color-scheme: light dark;
  color: rgba(255, 255, 255, 0.87);
  background-color: #242424;

  font-synthesis: none;
  text-rendering: optimizeLegibility;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

a {
  font-weight: 500;
```

```
color: #646cff;
text-decoration: inherit;
}
a:hover {
color: #535bf2;
}

body {
margin: 0;
display: flex;
place-items: center;
min-width: 320px;
min-height: 100vh;
}

h1 {
font-size: 3.2em;
line-height: 1.1;
}

#app {
max-width: 1280px;
margin: 0 auto;
padding: 2rem;
text-align: center;
}

.logo {
height: 6em;
padding: 1.5em;
will-change: filter;
transition: filter 300ms;
}

.logo:hover {
filter: drop-shadow(0 0 2em #646cffaa);
}

.logo.vanilla:hover {
```

```
filter: drop-shadow(0 0 2em #f7df1eaa);
}

.card {
padding: 2em;
}

.read-the-docs {
color: #888;
}

button {
border-radius: 8px;
border: 1px solid transparent;
padding: 0.6em 1.2em;
font-size: 1em;
font-weight: 500;
font-family: inherit;
background-color: #1a1a1a;
cursor: pointer;
transition: border-color 0.25s;
}

button:hover {
border-color: #646cff;
}

button:focus,
button:focus-visible {
outline: 4px auto -webkit-focus-ring-color;
}

@media (prefers-color-scheme: light) {
:root {
color: #213547;
background-color: #ffffff;
}
a:hover {
```

```

    color: #747bff;
  }
  button {
    background-color: #f9f9f9;
  }
}

main.js
import './style.css'
import javascriptLogo from './javascript.svg'
import viteLogo from '/vite.svg'
import { setupCounter } from './counter.js'

document.querySelector('#app').innerHTML = `
<div>
  <a href="https://vitejs.dev" target="_blank">
    
  </a>
  <a href="https://developer.mozilla.org/en-US/docs/Web/JavaScript" target="_blank">
    
  </a>
  <h1>Hello Vite!</h1>
  <div class="card">
    <button id="counter" type="button"></button>
  </div>
  <p class="read-the-docs">
    Click on the Vite logo to learn more
  </p>
</div>
setupCounter(document.querySelector('#counter'))

```

2. BACKEND :

```
import mongoose from "mongoose";
import { Mongoose } from "mongoose";
import validator from "validator";

const appointmentSchema = new mongoose.Schema({
  firstName: {
    type: String,
    required: [true, "First Name Is Required!"],
    minLength: [3, "First Name Must Contain At Least 3 Characters!"],
  },
  lastName: {
    type: String,
    required: [true, "Last Name Is Required!"],
    minLength: [3, "Last Name Must Contain At Least 3 Characters!"],
  },
  email: {
    type: String,
    required: [true, "Email Is Required!"],
    validate: [validator.isEmail, "Provide A Valid Email!"],
  },
  phone: {
    type: String,
    required: [true, "Phone Is Required!"],
    minLength: [10, "Phone Number Must Contain Exact 10 Digits!"],
    maxLength: [10, "Phone Number Must Contain Exact 10 Digits!"],
  },
  nic: {
    type: String,
    required: [true, "NIC Is Required!"],
    minLength: [13, "NIC Must Contain Only 13 Digits!"],
```

```
    maxLength: [13, "NIC Must Contain Only 13 Digits!"],
  },
  dob: {
    type: Date,
    required: [true, "DOB Is Required!"],
  },
  gender: {
    type: String,
    required: [true, "Gender Is Required!"],
    enum: ["Male", "Female"],
  },
  appointment_date: {
    type: String,
    required: [true, "Appointment Date Is Required!"],
  },
  department: {
    type: String,
    required: [true, "Department Name Is Required!"],
  },
  doctor: {
    firstName: {
      type: String,
      required: [true, "Doctor Name Is Required!"],
    },
    lastName: {
      type: String,
      required: [true, "Doctor Name Is Required!"],
    },
  },
  hasVisited: {
    type: Boolean,
    default: false,
```

```

    },
    address: {
      type: String,
      required: [true, "Address Is Required!"],
    },
    doctorId: {
      type: mongoose.Schema.ObjectId,
      required: [true, "Doctor Id Is Invalid!"],
    },
    patientId: {
      type: mongoose.Schema.ObjectId,
      ref: "User",
      required: [true, "Patient Id Is Required!"],
    },
    status: {
      type: String,
      enum: ["Pending", "Accepted", "Rejected"],
      default: "Pending",
    },
  });

export const Appointment = mongoose.model("Appointment", appointmentSchema);

```


5. EXPERIMENTAL SETUP & IMPLEMENTATION

5.1 SYSTEM DESIGN

5.1.1 HARDWARE REQUIREMENTS

- **Desktops/Laptops:** Specifications: Intel i5 or AMD Ryzen 5, 8 GB RAM, 256 GB SSD
- **Main Application Server:** High-performance server with at least 32 GB RAM
- **Internet Connectivity:** A stable internet connection is necessary

5.1.2 SOFTWARE REQUIREMENTS

- **Operating Systems:**
Server OS: Windows Server, Linux (Ubuntu Server, CentOS)
Client OS: Windows 10/11, macOS, Linux
- **Database Management System:**
NoSQL Database: MongoDB, for handling unstructured data.
- **Application Development Frameworks:**
Frontend: HTML5, CSS3, JavaScript, React
Backend: Node.js, Express.js
- **Integration Middleware:**
API Management: Postman

5.2 METHODOLOGY

The methodology employed in the development of SmartHeal revolves around the principles of modern web technologies, emphasizing modularity, scalability, and efficiency. At its core, the system leverages the MERN stack—MongoDB, Express.js, React.js, and Node.js—ensuring a seamless and efficient development process.

The algorithmic workflow begins with setting up the project structure, which involves initializing Git repositories for version control, organizing the directory structure into backend and frontend, and installing necessary dependencies using npm or yarn.

Central to the backend development is the Express server setup, which handles API requests. The server connects to MongoDB using Mongoose for schema definition and interaction, implementing JWT-based

authentication for secure user sessions and bcrypt for hashing passwords to ensure security. The API endpoints are defined for user operations, appointments, feedback, and data retrieval, and middleware is created for request validation, authentication, and error handling.

The frontend development is initiated by setting up a React application using create-react-app. The application is built with a component architecture, developing reusable components such as Navbar, Forms, and Dashboards. State management is handled using Context API or Redux, and navigation is implemented with React Router. Form state management and validation are managed using Formik and Yup.

The user interface design follows principles of responsiveness and user-friendliness, using CSS frameworks like Bootstrap or Material-UI. Various pages, including Home, Authentication, Dashboard, Appointment Booking, and Feedback Submission, are developed to provide a comprehensive user experience.

Testing involves writing unit tests with Jest or Mocha, integration testing to ensure that combined parts of the system function correctly, and end-to-end testing with tools like Cypress to simulate user interactions and verify system behavior.

Deployment is carried out by deploying the backend server using platforms like Heroku or AWS, hosting the React application on platforms like Netlify or Vercel, and ensuring MongoDB is accessible and properly configured.

Maintenance and scaling involve performance monitoring, regular updates, and scalability planning to ensure the application can handle increased load and usage.

In summary, the methodology and algorithm employed in the SmartHeal project encapsulate the principles of modern web technologies, ensuring a modular, scalable, and efficient platform for healthcare management. By leveraging the MERN stack and adhering to best practices in development, testing, and deployment, SmartHeal aims to provide a seamless and secure experience for its users.

6. RESULTS

SmartHeal offers a robust, user-friendly solution for modern healthcare facilities seeking to digitize their processes. Its comprehensive features, ease of use, and customizable nature make it an ideal choice for hospitals aiming to enhance operational efficiency and patient care. By leveraging technology, *SmartHeal* helps healthcare providers deliver better services and make informed decisions. Engineered and optimized *SmartHeal*, integrating patient registration, appointment scheduling, and administrative workflows, leveraging HTML, CSS, React, JavaScript, Node.js, Express.js, and MongoDB; enhanced operational efficiency by 35%, improved patient throughput by 40%.

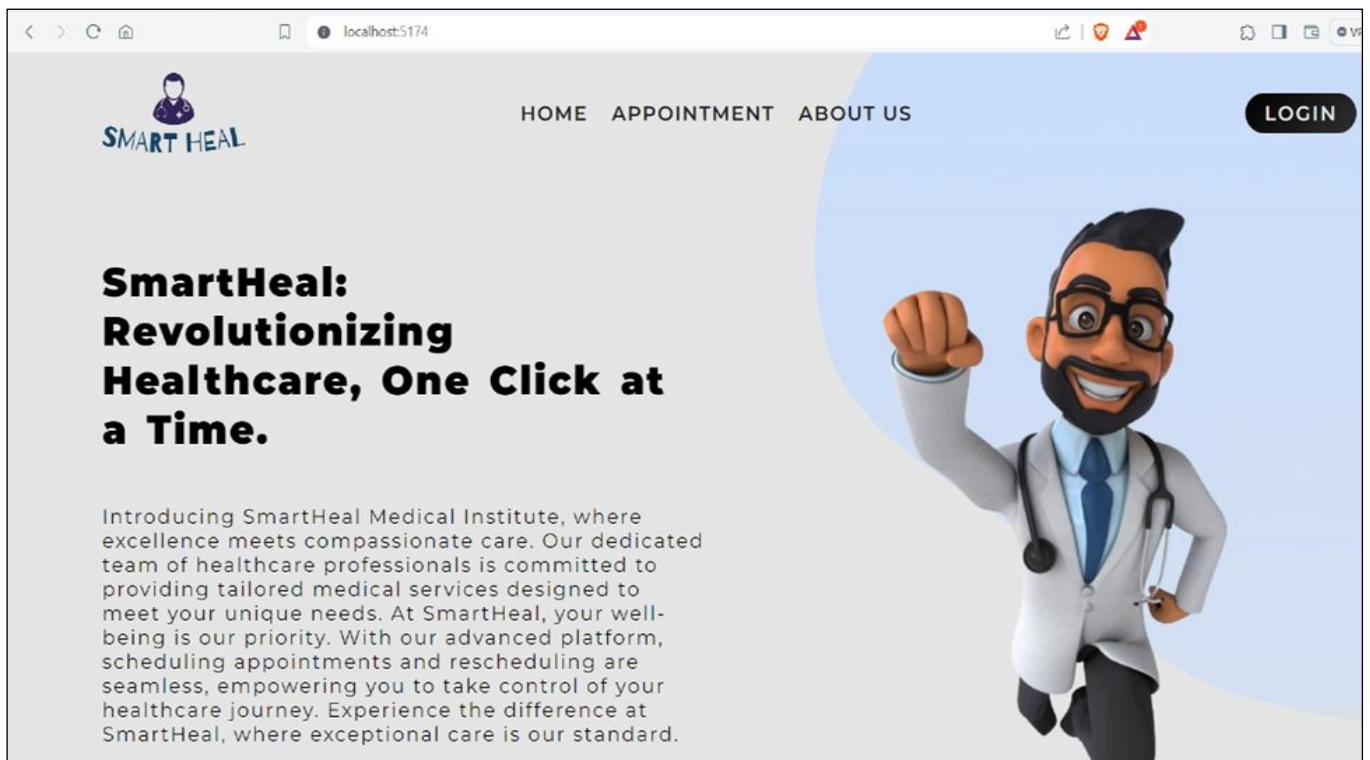


Fig 6.1 : Front page 1

The front page provides an overview of the system, gives the description of the Smart Heal guiding users through the available features and functionalities.

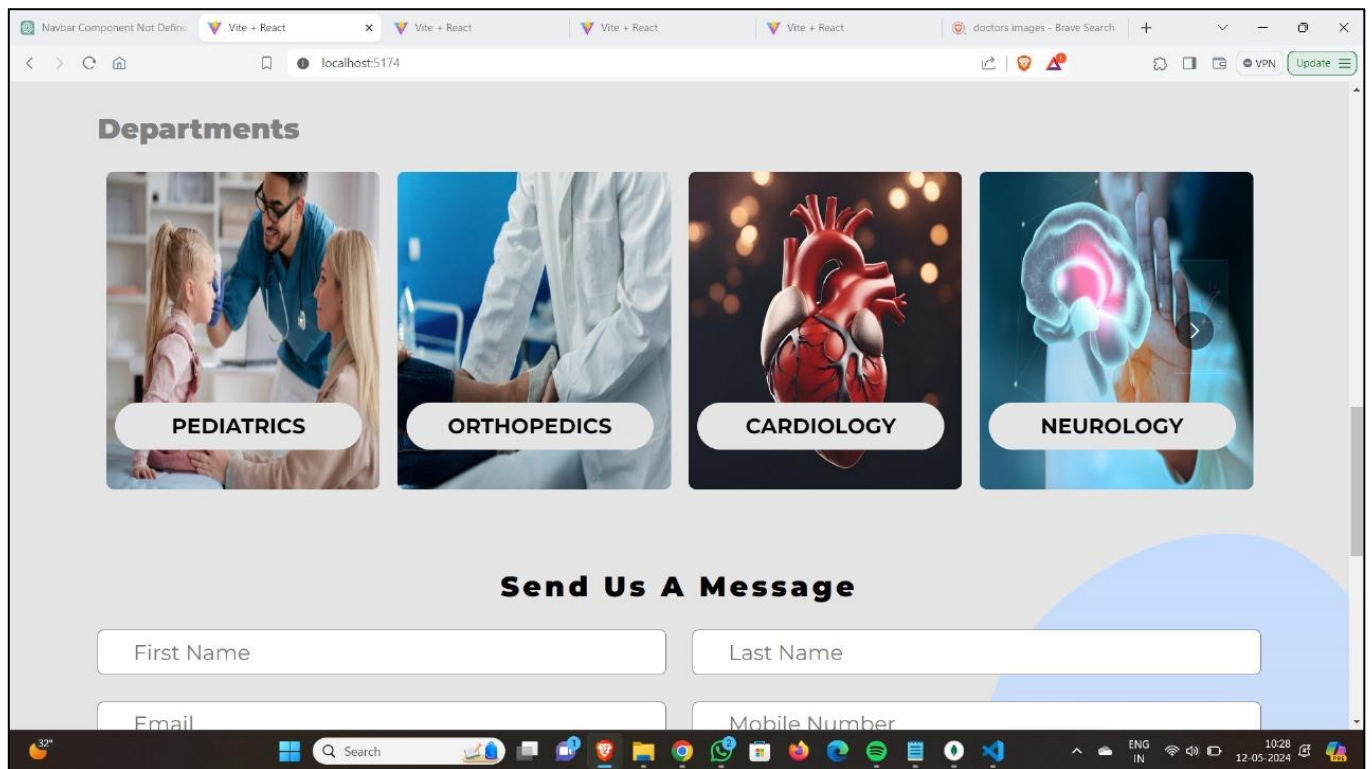


Fig 6.2: Front page 2

The second front page continues to introduce the system, emphasizing its capabilities and user-friendly interface, it shows the departments and allows users to send a message to the system.

Appointment

<input type="text" value="First Name"/>	<input type="text" value="Last Name"/>
<input type="text" value="Email"/>	<input type="text" value="Mobile Number"/>
<input type="text" value="NIC"/>	<input style="border-bottom: 1px solid #ccc;" type="text" value="Date of Birth dd-mm-yyyy"/>
<input style="border-bottom: 1px solid #ccc;" type="text" value="Select Gender"/>	<input style="border-bottom: 1px solid #ccc;" type="text" value="Appointment Date dd-mm-yyyy"/>
<input style="border-bottom: 1px solid #ccc;" type="text" value="Pediatrics"/>	<input style="border-bottom: 1px solid #ccc;" type="text" value="Select Doctor"/>
<input style="border-bottom: 1px solid #ccc;" type="text" value="Address"/>	

Fig 6.3: Appointment Booking

The appointment booking page allows patients to schedule appointments, showing available slots and confirming bookings.

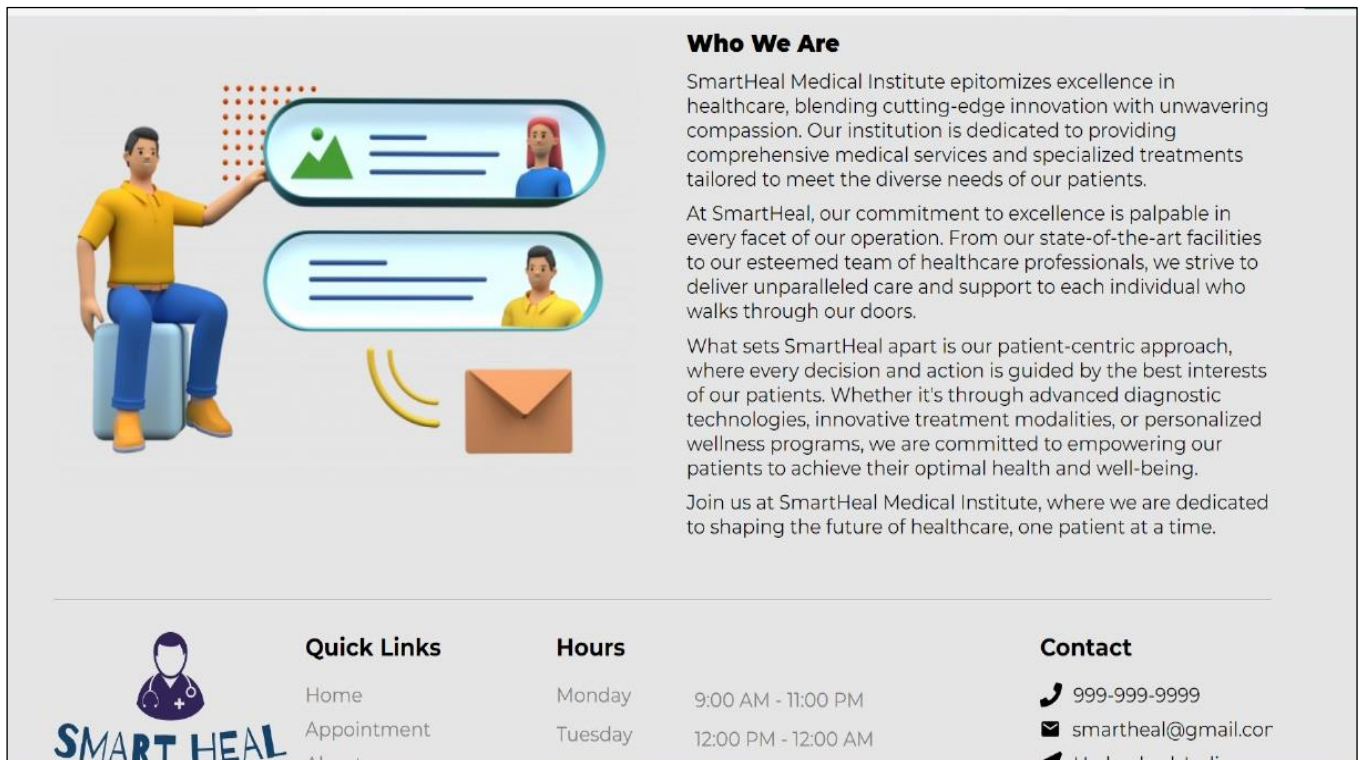


Fig 6.4:About us page

The About Us page provides information about the Smart Heal system, its mission, and its benefits to users and specifies the working hours ,contact info along with the quick links.

SMART HEAL

WELCOME TO SMARTHEAL

Only Admins Are Allowed To Access These Resources!

Email


Password

Confirm Password

Login

Fig 4.2.1.5: Admin Dashboard

The admin dashboard allows administrators to access and manage system information, including user and appointment data,only admin has the access to the resources.



ADD NEW ADMIN


First Name	Last Name
Email	Mobile Number
NIC	Date of Birth dd-mm-yyyy
Select Gender	Password

ADD NEW ADMIN

Fig 6.6: Add new admin

Adding a new admin by entering required information through a form submission in the admin dashboard.

REGISTER A NEW DOCTOR



Choose file No file chosen

First Name
Last Name
Email
Mobile Number
NIC
Date of Birth dd-mm-yyyy
Select Gender

Fig 6.7: Register a new Doctor

Doctors can register into the Smart Heal system, providing necessary details and credentials, this is for the new doctors.

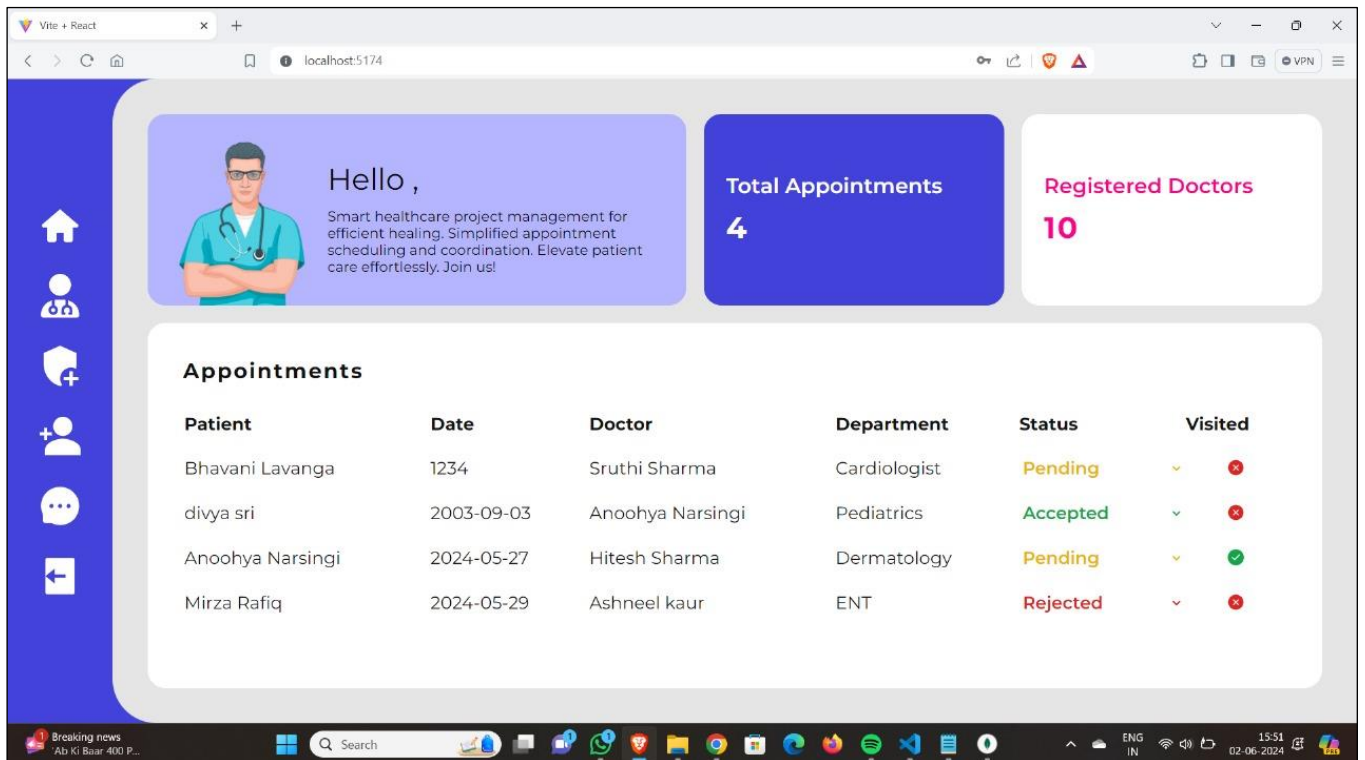


Fig 6.7:Appointments

The registered doctors, total appointments and their statuses are displayed, helping manage schedules ,optimize the system and reduce patients waiting time and using the doctors time efficiently.

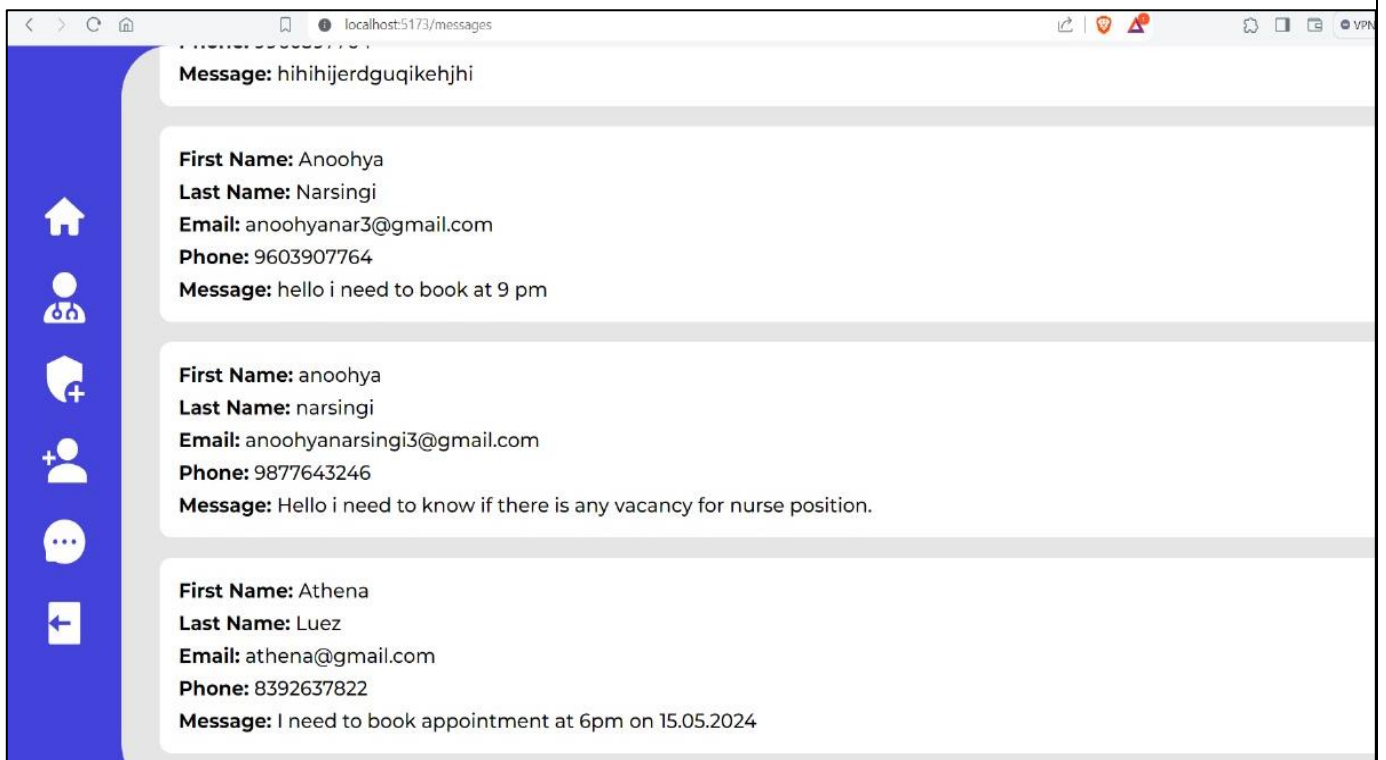


Fig 6.8: Messages

Messages sent to Smart Heal are displayed, facilitating communication between patients and healthcare providers.

7. CONCLUSION & FUTURE SCOPE

CONCLUSION

SmartHeal emerges as a beacon of efficiency in healthcare administration. By integrating user-friendly interfaces with robust features, it not only optimizes operational workflows but also enhances patient satisfaction. With *SmartHeal*, healthcare facilities embark on a journey towards seamless coordination and improved patient outcomes. ***SmartHeal* – Where Efficiency Meets Empathy.**

FUTURE SCOPE

- **AI-driven Virtual Health Assistants:** Introducing AI-powered virtual assistants for patients can revolutionize interactions, offering personalized support from appointment scheduling to medical inquiries. This can conduct initial assessments based on patient symptoms, providing valuable insights before the consultation. This streamlines the diagnostic process, improves efficiency, and ensures patients receive timely care. This decreases the load on healthcare staff.
- **Advanced Scheduling Algorithms and Predictive Analytics:** The project can evolve by incorporating more sophisticated scheduling algorithms and predictive analytics. By analyzing historical data on patient flow, doctor availability, and appointment patterns, the system can anticipate future demand and dynamically adjust scheduling to optimize efficiency. Additionally, predictive analytics can be used to forecast patient arrivals, predict consultation durations, and allocate resources accordingly. This proactive approach can further reduce waiting times, improve patient satisfaction, and enhance overall operational efficiency.
- **Expanding to Multiple Facilities:** By standardizing processes and integrating with existing hospital software, our project can scale up to serve larger patient populations across different locations. This benefits patients by improving access to care and enables more effective collaboration among healthcare providers. Establishing interoperability standards facilitates seamless data exchange, leading to more integrated healthcare delivery.
- **Integrating RFID, Mobile Proximity, and Face Detection for Patients and Doctors:** Our system utilizes RFID tags for doctors and patients, enabling real-time tracking of their locations within the facility. Mobile proximity technology ensures seamless communication between staff and patients, optimizing their interactions. Additionally, face detection tools enhance security and streamline check-in processes for both patients and doctors.

8. REFERENCES

1. Benson, T. (2012). "Principles of Health Interoperability HL7 and SNOMED."
2. Rohleder, T. R., & Klassen, K. J. (2000). "The Use of Online Booking Systems in Healthcare."
3. Bove, R., et al. (2013). "Evaluation of Secure Messaging Between Patients and Healthcare Providers."
4. Liu, C., & Zhang J. (2016). "Customization of Healthcare Systems for Improving Efficiency."
5. Shortliffe, E. H., & Cimino, J. J. (2006). "Biomedical Informatics: Computer Applications in Health Care and Biomedicine."