

# Team Name: Slackers

## Team Strengths:

Our team, Slackers, brings together a diverse group of professionals with expertise in software development, cloud technologies, AI, fintech, and healthcare IT. We excel in problem-solving, teamwork, and technical proficiency, ensuring efficient collaboration and project execution. With a balanced mix of experienced professionals and students, we combine academic knowledge with industry expertise to achieve our goals effectively. Our team values open communication, adaptability, and continuous learning, ensuring we remain aligned and productive throughout our projects.

---

## Team Members

### Chetan Kapadia

Chetan holds a Bachelor's degree in Computer Applications and multiple IT certifications, including Microsoft's latest. Currently pursuing a Master's in Computer Science, he is passionate about healthcare IT, particularly databases and data-driven decision-making in medical applications. With extensive experience as a Senior Software Engineer, Chetan has worked on large-scale projects involving cloud computing and data engineering. He aims to establish organizations and pursue a Ph.D. in Healthcare IT, where he hopes to contribute to improving medical systems with AI-driven solutions. Outside of work, he enjoys sports, networking with professionals, and exploring new technologies. Originally from Ahmedabad, Gujarat, India, he has been residing in Charlotte for a few years and is set to graduate in December 2026.

### Avitesh Kesharwani

Avitesh is a Technical Architect and Transformation Delivery Leader with over 10 years of fintech experience. He specializes in Java, Python, cloud services, and application modernization. Currently a Senior Manager at Genpact, he leads technical initiatives for Ryan Turner Specialty, focusing on migration projects and team development. Avitesh holds a B.Tech in Information Technology and is pursuing an MS in Computer Science. Passionate about AI and emerging technologies, he also enjoys writing and publishing papers. In his free time, he engages in gardening, DIY projects, and spending time with his wife and son, Angad.

## **Rahul Joshi**

Rahul holds a B.Tech degree and has three years of experience as a developer at TCS. He is currently pursuing a Master's in Computer Science and has a strong background in software development, data structures, and algorithms. His expertise lies in Java, Python, cloud computing, and backend technologies. Rahul has contributed to multiple projects involving large-scale data processing and fintech solutions. He is passionate about optimizing code for efficiency and developing scalable software solutions. Outside of academics and work, Rahul enjoys exploring AI-driven innovations, mentoring aspiring developers, and playing chess.

## **Sudeepta Bal**

Sudeepta Bal is currently pursuing her Master of Science degree in Computer Science from UNCC, and this is her first semester. She holds a Bachelor's degree in Computer Science from India. Before starting her master's, she worked as a techno-functional procurement consultant and is certified in the Coupa Cloud Platform. With over five years of experience in SAP and Coupa implementation projects, she has been actively involved in full project life cycles, covering areas such as SAP ABAP, SAP ARIBA, and Coupa. She is keen on expanding her technical knowledge, particularly in Python. Beyond work, she enjoys spending time with her family, dancing, and playing badminton.

## **Nikhila Chitta**

Nikhila is currently pursuing a Master's degree in Data Science at the University of North Carolina at Charlotte (UNCC). She holds a Bachelor's degree in Electronics and Communication Engineering from JNTU Kakinada. After completing her undergraduate studies, she worked for four years as a Senior Software Engineer at Accenture, where she developed software solutions, collaborated with cross-functional teams, and honed her technical skills. Her interests lie in data science, machine learning, and artificial intelligence, with a strong enthusiasm for mobile application development and network technologies. Outside of academics, she enjoys nature, birdwatching, scenic walks, and traveling to new places. Originally from Rajahmundry, India and aims to build a career in data science or AI, applying her skills to drive innovation and create meaningful solutions.

### **Team Emails:**

- **Sudeepta Bal:** sbal1@charlotte.edu
  - **Avitesh Kesharwani:** Akesharw@charlotte.edu
  - **Chetan Kapadia:** ckapadi1@charlotte.edu
  - **Nikhila Chitta:** nchitta@charlotte.edu
  - **Rahul Joshi:** rjoshi19@charlotte.edu
-

# Team Agreement

## Methods of Communication

- **Primary:** Slack group (SSDipprojectWorkspace) is the main platform for discussions, sharing updates, and tracking progress.
- **Secondary:** A WhatsApp group will serve as a backup for urgent communications and quick updates.
- **Tertiary:** Email communication will be used for formal discussions, documentation sharing, and finalizing important decisions.

## Communication Response Times

- Team members should aim to respond to messages on Slack and WhatsApp within **one hour** during active work periods.
- Emails should be acknowledged within **12 hours** and responded to within **24 hours** unless urgent.

## Meeting Attendance

- Weekly team meetings will be held **every Sunday at 8:00 PM EST** via Slack (online).
- Attendance is **mandatory** unless a valid reason is provided in advance.
- If a member is unable to attend, they must review the meeting minutes and updates shared afterward.

## Meeting Preparation

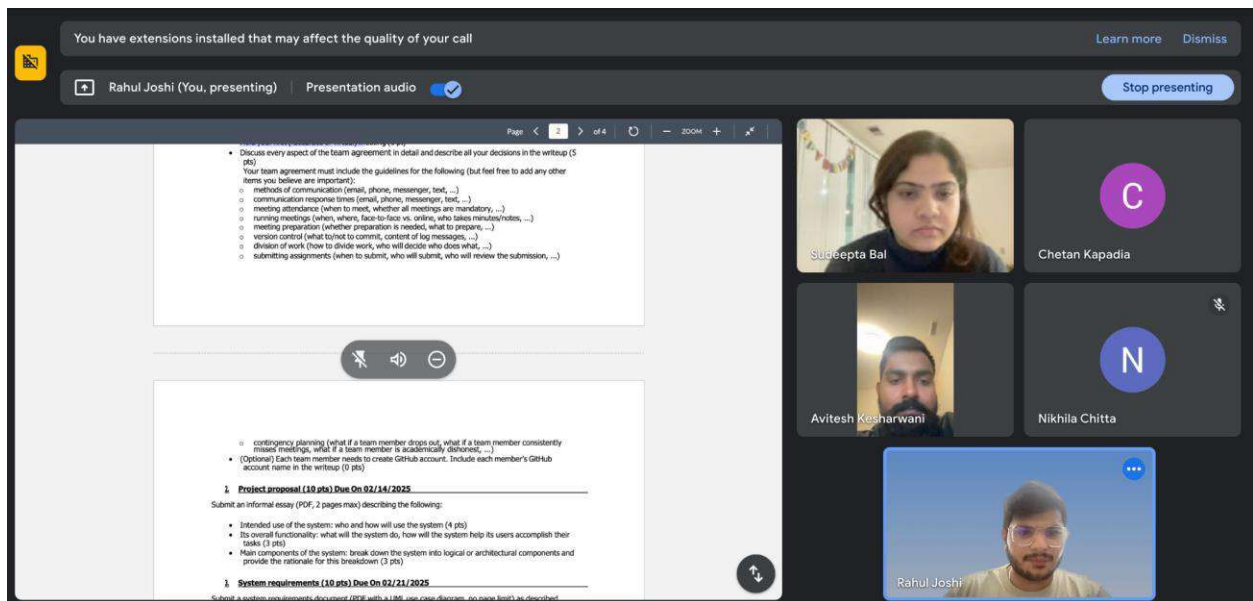
- **Chetan and Nikhila** will prepare the meeting agendas.
- Team members should review discussion topics before the meeting and be ready to provide updates.
- Any blockers or dependencies should be identified in advance to streamline discussions.

## Running Meetings

- Meetings will be conducted on Slack with a structured agenda.
- A team member will take minutes/notes on a rotational basis.
- Decisions and action items will be documented and shared post-meeting.

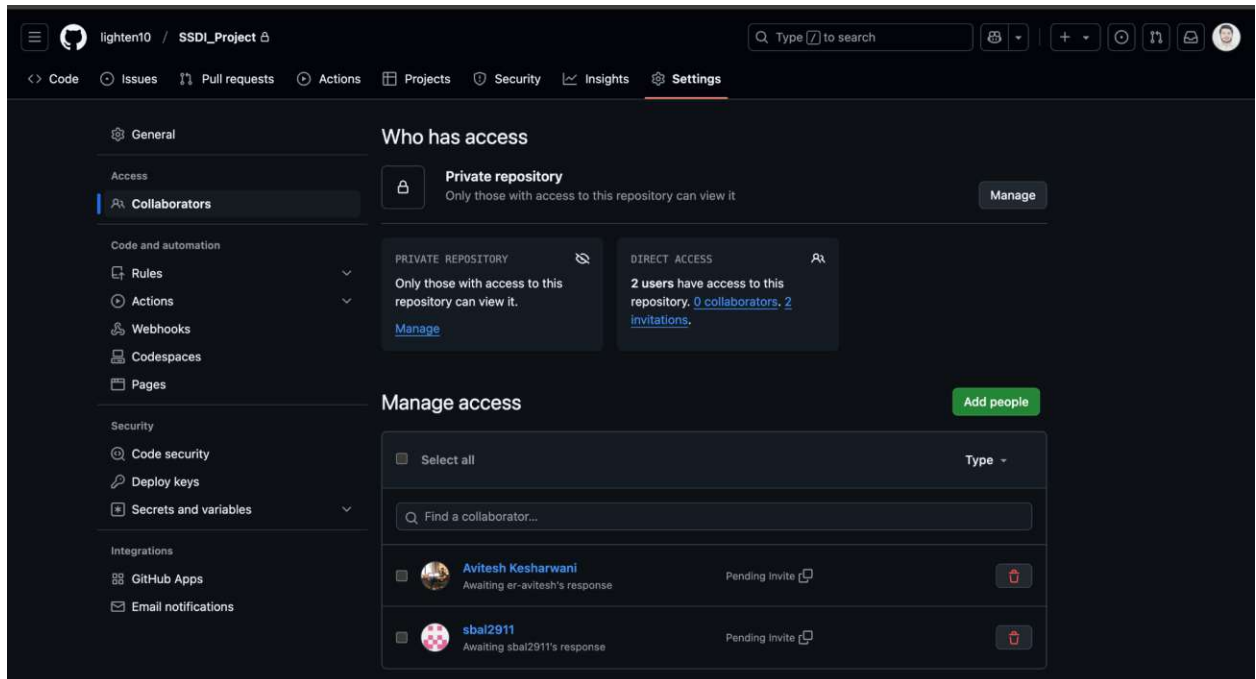
## Version Control

- **Rahul** created and manages the GitHub repository.
- Team members should commit frequently and provide meaningful commit messages.
- Branching and merging strategies will be followed to maintain a clean and collaborative workflow.
- Discussions regarding major changes or conflicts will be held on Slack before committing.



## Division of Work

- **Sudeepta** is the Scrum Master, responsible for planning and allocating tasks.
- Work will be divided based on skill sets and availability, ensuring a fair distribution of tasks.
- The team will hold sprint planning meetings to assign tasks and set priorities.
- Developers and testers will coordinate closely to ensure quality and efficiency.



## Submitting Assignments

- **Sudeepta** (Scrum Master) is responsible for reviewing and submitting assignments.
- Work should be completed at least **24 hours before submission deadlines** for review.
- Each team member is responsible for ensuring their contributions meet quality standards before submission.

## Contingency Planning

- In case a team member is unavailable, tasks will be reassigned to ensure project continuity.
- Backup plans will be in place for critical project components to prevent delays.
- Regular progress tracking will help identify and mitigate potential risks early.

## **Abstract**

In today's digital age, accessing timely and efficient healthcare continues to be a challenge, particularly for individuals living in remote or underserved areas. Traditional healthcare systems often suffer from issues like long wait times, limited access to specialists, poor interdepartmental coordination, and difficulty maintaining accurate patient records. To address these challenges, we developed a fully featured **Health Sphere System**.

This system follows a 3-Tier Architecture and includes dedicated modules for **patients**, **doctors**, and **administrators**. Patients can manage appointments, view medical and billing history, receive notifications, and provide feedback. Doctors can handle appointment requests, update medical records, generate bills, and review patient histories. The admin module provides tools to monitor clinic operations, manage users, and access departmental insights. A well-structured database schema ensures secure and efficient data handling across the application. The system aims to provide a reliable, easy-to-use platform that improves healthcare access and administrative efficiency in clinics.

## **Problem Statement**

Although technology has greatly evolved, many people, especially those in rural or less developed areas, still face difficulties when trying to access proper healthcare. Travelling long distances for basic medical services, experiencing delays due to long queues, and having limited options for specialist care are common problems. Moreover, many traditional healthcare systems suffer from poor coordination between departments, which can slow down treatment and cause communication gaps. Keeping patient information organized and up-to-date is also a major concern, as many systems still rely on outdated or disconnected record-keeping methods. These challenges point to the urgent need for a digital healthcare solution that is simple to use, highly efficient, and secure. A well-designed online system could make healthcare more accessible by offering features like taking appointments online without the need to stand in long queues, well-organized patient/doctor/other staff information, and getting to know the status of their appointment online without the need to call the clinic multiple times.

## **Intoduction**

**Health Sphere** is a comprehensive digital solution created to simplify and strengthen interactions between patients, medical professionals, and administrative personnel. It includes features such as secure access tailored to different user roles, online systems for booking appointments, access to patient records, electronic prescription management, and billing updates based on treatments received. The platform is built to boost the efficiency of healthcare services while offering a smooth and accessible experience for all users.

### **Intended use of the system**

**Health Sphere** is a software application designed to streamline and automate various administrative, medical and financial processes within a hospital or healthcare facility. It centralizes data, facilitates communication between departments and enhance operational efficiency.

**Patient:** The patient can access and review their profile. The patient can view a list of pending or approved appointments with a doctor. Patients can access a comprehensive history of their completed appointments, including detailed information about each visit. Patients have access to a comprehensive view of their completed treatment history. Patients have access to a comprehensive view of all available departments, enabling them to select their preferred department. Upon choosing a department, a list of doctors affiliated with that department is displayed. Patients can then choose a specific doctor and view their profile, which includes a “Take Appointment” that reveals the available appointment slots for that particular doctor. Patients can select a suitable slot and submit a request to the doctor for that appointment. The doctor will subsequently approve or reject the request. A notification is displayed whenever the doctor accepts or rejects the requested appointment. Following the completion of an appointment, patients are provided with the opportunity to provide feedback by rating it on a scale of 1 to 5. Patients are permitted to Schedule only one appointment at a time. They are not permitted to schedule additional appointments until the previously scheduled appointment has been completed.

**Provider (Doctor):** The provider can access their profile. The provider can view all pending appointments. The appointments for the current day will be displayed. The provider can then select or reject any appointment scheduled for that day. The provider can update the prescription, disease, and progress of the patient. The provider will be able to view the treatment history of all their treated patients.

**Management:** The management team has access to comprehensive clinic statistics, including weekly appointments, facility income (clinic, hospital, laboratory), registered patient and doctor counts, and a list of departments. The management team can also view the list of currently registered providers, including their departments and other relevant information. They can manage complete profiles for each provider. Additionally, the management team can access the list of currently registered patients, including their phone numbers and IDs. They can manage complete profiles for each patient. Furthermore, the management team can view other staff members, regardless of their designations. The management team can also search for a specific employee within the facility.

### **Overall Functionality of the Project Under the Health Sphere**

Our project under the Health Sphere aims to improve healthcare efficiency and accessibility by leveraging digital solutions for patients, healthcare providers, and administrators. It will streamline key healthcare functions such as appointment scheduling, medical record management, billing, and communication, reducing manual effort and enhancing service quality.

For patients, the system will provide an easy-to-use platform to book appointments, access medical history, receive notifications, and engage in remote consultations. Features like prescription requests and real-time health monitoring will help ensure proactive healthcare management. Additionally, a previous visit history feature may be included to allow patients to review their past appointments, diagnoses, and treatments. Healthcare providers will have access to automated appointment management, real-time patient records, and tools for prescribing medications and generating medical reports. A potential doctor visit summary feature could provide a concise overview of each patient’s visits, aiding in continuity of care and reducing redundant procedures. Telemedicine capabilities may also be integrated to support remote consultations. Administrators will manage hospital or clinic operations, oversee doctor and patient records, coordinate staff, and track financial transactions. Reporting and analytics tools will help improve healthcare services and decision-making. By integrating technology into healthcare, this project will create a well-organized and patient-centric system that enhances communication, reduces delays, and improves overall healthcare

outcomes. Whether for hospitals, clinics, or telehealth services, it will ensure better coordination and accessibility in medical services.

### **Main Components of the System**

The system will be broken down into the following logical and architectural components to ensure modularity, scalability, and maintainability:

**User Management Component:** Manage user authentication, roles, and permissions.

**Rationale:** Ensures secure access and defines user-specific functionalities based on roles, maintaining data confidentiality and integrity.

**Features:** User registration and login (Admins, Doctors, Nurses, Receptionists, Patients), Role-based access control, Password recovery and profile management

**Patient Management Component:** Healthcare professionals can handle patient-related information, including medical history, appointments, and reports.

**Rationale:** Centralizes patient data for quick retrieval by authorized personnel, improving diagnosis and treatment processes.

**Features:** Patient registration and profile management, viewing and updating medical records, uploading and retrieving test results and prescriptions

**Appointment Scheduling Component:** Enable seamless scheduling and management of appointments between patients and healthcare professionals.

**Rationale:** Reduces scheduling conflicts and improves time management for both staff and patients.

**Features:** Appointment booking, rescheduling, and cancellation, Automated notifications/reminders for upcoming appointments

**Staff Management Component:** Manage hospital staff information, including doctors, nurses, and administrative personnel.

**Rationale:** Optimizes staff allocation and ensures proper resource management for hospital operations.

**Features:** Staff profile creation and management, Duty and shift scheduling

**Billing and Invoicing Component:** Automate billing processes for patients, including payments and insurance claims.

**Rationale:** Simplifies financial operations and reduces errors in manual billing, ensuring smooth transactions.

**Features:** Generation of bills and invoices for treatments and services, Payment gateway integration for online payments, Insurance management, and claim processing

**Notifications and Alerts Component:** Facilitate timely communication between patients, staff, and the hospital system.

**Rationale:** Enhances communication, ensuring important updates are delivered promptly.

**Features:** Email and SMS alerts for appointments, medication schedules, and billing, Critical alerts for emergency cases and system errors, In-app notifications for updates and reminders

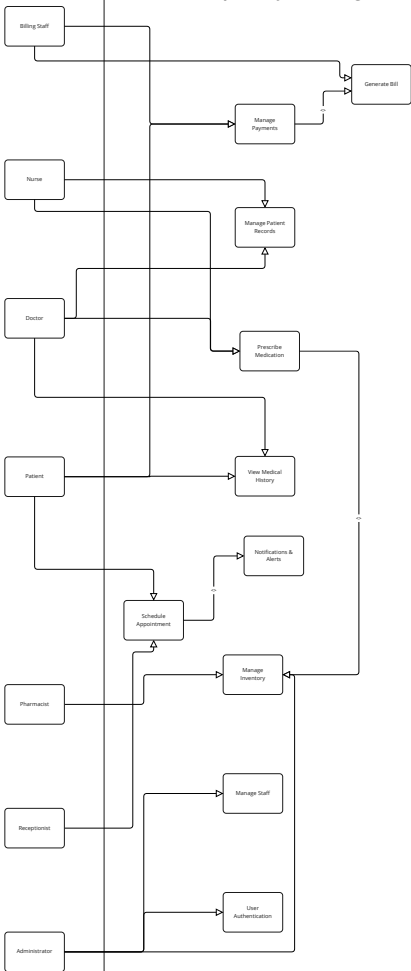
**Security and Compliance Component:** Ensure the system adheres to security best practices and healthcare regulations (e.g., HIPAA).

**Rationale:** Protects sensitive health data and maintains legal compliance with healthcare data regulations.

**Features:** Data encryption and secure API communication, Audit trails for sensitive actions, Regular security updates, and compliance checks.



# HealthSphere System Design



### **Brief narrative on overview of the system's functional requirements**

The Health Sphere system is a revolutionary tool in healthcare management that streamlines operations and improves patient care. Its user management feature allows healthcare facilities to create and manage user accounts tailored to various roles, ensuring secure access. The system also offers appointment scheduling, patient record management, efficient inventory management, streamlined billing and payment processes, and electronic prescription management. It also includes robust notifications and alerts for patients and staff, comprehensive reporting capabilities for better decision-making, and secure messaging for efficient communication between staff and patients. Overall, the Health Sphere system integrates functional requirements into a cohesive solution that significantly improves patient care and operational efficiency in healthcare facilities.

### **Summary of each use case from the "Health Sphere System Design"**

1. **User Authentication:** Confirms that only authorized individuals can access the system by using secure login details.
2. **Schedule Appointment:** Facilitates the process for patients to book appointments with healthcare providers effortlessly.
3. **Manage Inventory:** Empowers staff to monitor and control the availability of medical supplies and equipment.
4. **Manage Staff:** Administrators are responsible for overseeing and organizing staff roles and duties within the facility.
5. **Manage Payments:** Billing staff are responsible for processing and tracking payments from patients and insurers.
6. **Manage Patient Records:** Healthcare professionals are tasked with maintaining and updating patient medical files.
7. **Prescribe Medication:** Doctors can provide electronic prescriptions to ensure accurate medication distribution.
8. **Notifications & Alerts:** Sends alerts and reminders to patients as well as important notifications to staff about key tasks.
9. **View Medical History:** Healthcare providers can review a patient's complete medical background to make informed decisions.
10. **Generate Bill:** Billing personnel prepare detailed invoices for the healthcare services provided to patients.

## **Health Sphere user stories**

### **Patients:**

- + As a patient, I want the convenience of scheduling appointments online, allowing me to choose a time that suits my schedule.

#### Pre-condition:

- The patient must be registered in the system.
- Patient must be authenticated.

#### Post-condition:

- The patient will be able to schedule appointments.
- The patient will receive an appointment confirmation email.
- The patient will receive an appointment reminder email.

- + As a patient, I need access to my medical records and test results through a secure patient portal to make informed decisions about my health.

#### Pre-condition:

- The patient must be registered in the system.
- Patient must be authenticated.

#### Post-condition:

- The patient will be able to view their treatment history.

- + As a patient, I expect to receive appointment reminders via email to ensure timely attendance at scheduled healthcare appointments.

#### Pre-condition:

- The patient must be registered in the system.
- The patient must have scheduled appointments in the system.

#### Post-condition:

- The patient will receive an email notification for their scheduled appointment reminder.

### **Providers:**

- + As a provider, I need immediate access to a patient's comprehensive medical history within the system to support informed treatment decisions.

#### Pre-condition:

- The provider must be registered in the system.
- The provider must be authenticated.

#### Post-condition:

- Allow the provider to view the patient's treatment history.

- Allow the provider to make necessary changes to the patient's treatment plan.

- + As a provider, I want to minimize errors and streamline the medication prescription process by electronically prescribing medications.

Pre-condition:

- The provider must be registered in the system.
- The provider must be authenticated.

Post-condition:

- Allow the provider to prescribe new medications.

- + As a provider, I expect to receive immediate notifications of critical laboratory results to prompt timely interventions.

Pre-condition:

- The provider must be registered in the system.
- The provider must have an email address set up.

Post-condition:

- Send a critical email notification to the provider with sensitive test results.

### **Nurses:**

- + As a nurse, I need to update patients' vital signs and medication administration records.

Pre-condition:

- Nurses must be registered in the system and authenticated.

Post-condition:

- Nurses will be granted access to record patient vital readings and medication in the system.

- + As a nurse, I need to have seamless access to patient care plans and instructions to ensure accurate and effective treatment delivery.

Pre-condition:

- Nurses must be registered in the system and authenticated.

Post-condition:

- Nurses will be granted access to view the patient treatment plan.

## **Management:**

- + As a billing administrator, I need the automation of patient invoices based on treatment and insurance details to streamline the billing process.

Pre-condition:

- The billing executive must be registered in the system and authenticated.

Post-condition:

- The billing executive will be granted access to generate invoices.

- + As a receptionist, I anticipate the ease of patient registration and appointment details at the time of check-in.

Pre-condition:

- The receptionist must be registered in the system and authenticated.

Post-condition:

- The receptionist will be granted access to register patients and schedule appointments.

- + As a system administrator, my primary responsibility is to manage user access levels and security settings to ensure the protection of patient data.

Pre-condition:

- The admin must be registered in the system and authenticated.

Post-condition:

- The admin will be granted access to create users, disable users, and manage user roles.

## Smaller Epics derivation

Upon evaluating the user stories, we found that some are too broad and encompass multiple functionalities, making them complex to implement. These larger stories, often classified as epics, should be refined into smaller, more focused tasks to enhance clarity, estimation, and development efficiency.

### **Patients**

- **Online Appointment Scheduling** – Clearly defined and covers a single function. (No breakdown required)
- **Access to Medical Records and Test Results** – Since retrieving medical records and viewing test results are distinct actions, (this should be divided into two separate user stories.)
- **Appointment Reminder Emails** – Straightforward and specific. (No breakdown required.)

### **Providers**

- **Viewing a Patient's Medical History** – This involves accessing and modifying treatment plans, which are separate tasks. (Should be broken down into two stories: one for viewing history and another for updating treatment plans)
- **Electronic Prescription of Medications** – Well-scoped and focused on a single function. (No breakdown required.)
- **Receiving Critical Lab Result Notifications** – A clear and actionable story. (No breakdown required.)

### **Nurses**

- **Updating Vitals and Medication Records** – Since recording vital signs and logging medication administration are different tasks, (this should be split into two separate user stories.)
- **Accessing Patient Care Plans** – A well-defined story that focuses on a single function. (No breakdown required).

### **Management**

- **Automating Patient Invoicing** – A focused and clearly defined task. (No breakdown required.)
- **Patient Registration and Check-In** – Covers a specific process. (No breakdown required.)
- **Managing User Access and Security Settings** – Since user role management and security settings configuration are distinct administrative functions, (this should be divided into two separate stories.)

### **Conclusion**

Refining these complex user stories into smaller, well-defined tasks will improve project organization, simplify implementation, and enhance development efficiency.

## **Breakdown of the larger user stories into smaller, more manageable user stories:**

**Original:** As a patient, I need access to my medical records and test results through a secure patient portal to make informed decisions about my health.

**Revised Breakdown:**

1. As a patient, I want to securely access my past medical history online so that I can review my previous diagnoses and treatments.
  2. As a patient, I want to view my latest test results through the patient portal so that I can stay informed about my health status.
- 

### **Providers**

**Original:** As a provider, I need immediate access to a patient's comprehensive medical history within the system to support informed treatment decisions.

**Revised Breakdown:**

1. As a provider, I want to access a patient's complete medical history within the system so that I can make well-informed treatment decisions.
  2. As a provider, I need the ability to update and modify a patient's treatment plan to ensure accurate and personalized care.
- 

### **Nurses**

**Original:** As a nurse, I need to update patients' vital signs and medication administration records.

**Revised Breakdown:**

1. As a nurse, I want to record and update a patient's vital signs in the system to maintain accurate health records.
  2. As a nurse, I need to log administered medications to ensure proper documentation and tracking of patient treatment.
- 

### **Management (System Administrator)**

**Original:** As a system administrator, my primary responsibility is to manage user access levels and security settings to ensure the protection of patient data.

**Revised Breakdown:**

1. As a system administrator, I want to assign and manage user roles and permissions so that access to sensitive information is properly controlled.
2. As a system administrator, I need to configure security settings, such as password policies and multi-factor authentication, to enhance data protection.

## Non-functional Requirements for the Health Sphere Project:

The Health Sphere project will prioritize several key non-functional aspects to ensure the system is efficient, secure, and user-centric. The system will deliver high performance, capable of supporting up to 500 concurrent users without significant delays, with appointment scheduling and patient record retrieval processing within 2 seconds. The architecture will be scalable, allowing easy expansion to support additional healthcare facilities, increased patient records, and future telemedicine capabilities without system redesign.

Security and compliance are critical, with all patient data encrypted and secure API communication implemented to meet HIPAA standards. The system will perform regular security audits and maintain audit trails for all sensitive actions. To ensure reliability, the system will guarantee 99.9% uptime, supported by automatic failover mechanisms and regular data backups to prevent data loss.

The project will emphasize usability, providing an intuitive user interface accessible via both web and mobile platforms, ensuring ease of use for patients, doctors, and administrators. Real-time notifications and alerts for appointments and prescriptions will enhance the user experience. Additionally, the system will maintain data integrity, implementing secure storage and regular backups to prevent data corruption or loss.

From a maintainability perspective, the Health Sphere system will be built using a modular architecture, enabling easy updates and the addition of new features without interrupting existing services. The system will be designed for interoperability, ensuring compatibility with third-party healthcare platforms and external diagnostic tools. Additionally, it will adhere to compliance standards such as HL7 and HIPAA for healthcare data management.

Lastly, the system will ensure audibility by maintaining comprehensive logs of all user actions, facilitating security reviews, and ensuring accountability. Together, these non-functional requirements will contribute to building a robust, secure, and efficient healthcare management system that meets the needs of patients, healthcare providers, and administrators.

### Short Glossary

- **Patient:** An individual receiving healthcare services.
- **Provider:** A healthcare professional delivering medical care.
- **Nurse:** A licensed professional responsible for patient care and medication administration.
- **Billing Administrator:** Staff managing financial processes, including invoicing and payments.
- **Receptionist:** Frontline staff overseeing patient check-ins and appointment management.
- **System Administrator:** Individual managing system functionality and user security.
- **User Authentication:** Process of verifying a user's identity for secure access.
- **Appointment Scheduling:** Feature for booking, modifying, or canceling appointments.
- **Patient Portal:** A Secure online platform for accessing medical records and test results.



## HealthSphere backlog:

### Epic: **Management & Security**

**(User Story 1: low priority)** Create a HealthSphere database to manage users' information (patient/doctor/management team)

#### Tables

User  
Patient  
Provider  
Department  
Management  
Appointment

Story Points = 3

Sprint = 1

Pre-condition:

- SQL Express should be installed

Post-condition:

- able to connect to the SQL server and access the tables

Functionality: A backend database containing all the user information will be maintained. These users will be registered in the Health Sphere system (Patients/Doctors/Nurses/administrative staff, etc)

**(User Story 2: high priority)** Create a UI to register users (patient/doctor/management team)

Story Points = 8

Sprint = 1

Pre-condition:

- Visual Studio should be installed.

Post-condition:

- The Management team should be able to create/register the user.

Functionality: A user interface will be designed to allow new users to register themselves in the Health Sphere system.

**(User Story 3: Medium priority)** Create a UI to Login / Authenticate user

Story Points = 5

Sprint = 1

Pre-condition:

- Visual Studio should be installed.

Post-condition:

- Using the login screen allows the user to get authenticated.

Functionality: A user interface will be developed to validate/verify a user's login credentials and allow access to his/her profile.

**(User Story 4: Medium priority)** Create a UI to show users information.

Story Points = 5

Sprint = 1

Pre-condition:

- The user should be registered
- User should be authenticated

Post-condition:

- The list of users should be visible.

Functionality: A user interface will be designed to view the necessary user information via which the user registers himself/herself in the system.

**(User Story 5)** Create a UI to update user information.

Story Points = 5

Sprint = 1

Pre-condition:

- The user should be registered
- User should be authenticated

Post-condition:

- The list of users should be visible
- Allow logged-in users to select the user and edit the information

Functionality: This user interface will allow users to edit their information if necessary, and the changes will be updated accordingly.

---

Epic: **Patient & Provider**

**(User Story 6: Medium Priority)** Create a UI to show Patient profile information.

Story Points = 5

Sprint = 2

Pre-condition:

- Patients should be registered
- The patient should be authenticated

Post-condition:

- The patient should be able to view the profile information
- Patients should not be allowed to see other patients' profile information

**(User Story 7: High Priority)** As a patient, I want the convenience of scheduling appointments online, allowing me to choose a time that suits my schedule.

Story Points = 8

Sprint = 2

Pre-condition:

- The patient must be registered in the system.
- The patient must be authenticated.

Post-condition:

- The patient will be able to schedule appointments.
- The patient will receive an appointment confirmation email.
- The patient will receive an appointment reminder email.

**(User Story 8: High Priority)** As a patient, I need access to my medical records and test results through a secure patient portal to make informed decisions about my health.

Story Points = 8

Sprint = 2

Pre-condition:

- The patient must be registered in the system.
- The patient must be authenticated.

Post-condition:

- Patient will be able to view their treatment history.

**(User Story 9: Medium Priority)** As a patient, I expect to receive appointment reminders via email to ensure timely attendance at scheduled healthcare appointments.

Story Points = 5

Sprint = 2

Pre-condition:

- The patient must be registered in the system.
- Patients must have scheduled appointments in the system.

Post-condition:

- Patient will receive an email notification for their scheduled appointment reminder.

**(User Story 10: Medium Priority)** As a provider, I need immediate access to a patient's comprehensive medical history within the system to support informed treatment decisions.

Story Points = 5

Sprint = 2

Pre-condition:

- The provider must be registered in the system.
- The provider must be authenticated.

Post-condition:

- Allow the provider to view the patient's treatment history.
- Allow the provider to make necessary changes to the patient's treatment plan.

**(User Story 11: High Priority)** As a provider, I want to minimize errors and streamline the medication prescription process by electronically prescribing medications.

Story Points = 8

Sprint = 2

Pre-condition:

- The provider must be registered in the system.
- The provider must be authenticated.

Post-condition:

- Allow the provider to prescribe new medications.
- 

### Epic: **Account Receivable**

**(User Story 12: High Priority)** As a billing administrator, I need the automation of patient invoices based on treatment and insurance details to streamline the billing process.

Story Points = 8

Sprint = 3

Pre-condition:

- The billing executive must be registered in the system and authenticated.

Post-condition:

- Billing executives will be granted access to generate invoices.

**(User Story 13: High Priority)** As a receptionist, I anticipate the ease of patient registration and appointment details during check-in.

Story Points = 8

Sprint = 3

Pre-condition:

- Receptionists must be registered in the system and authenticated.

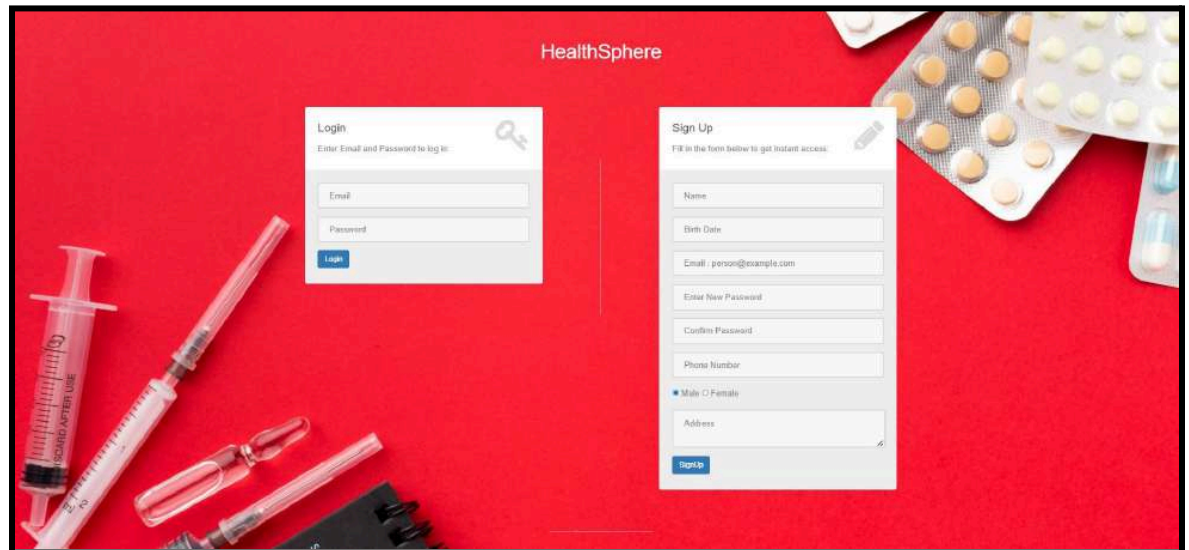
Post-condition:

- Receptionists will be granted access to register patients and schedule appointments.

## User Interface

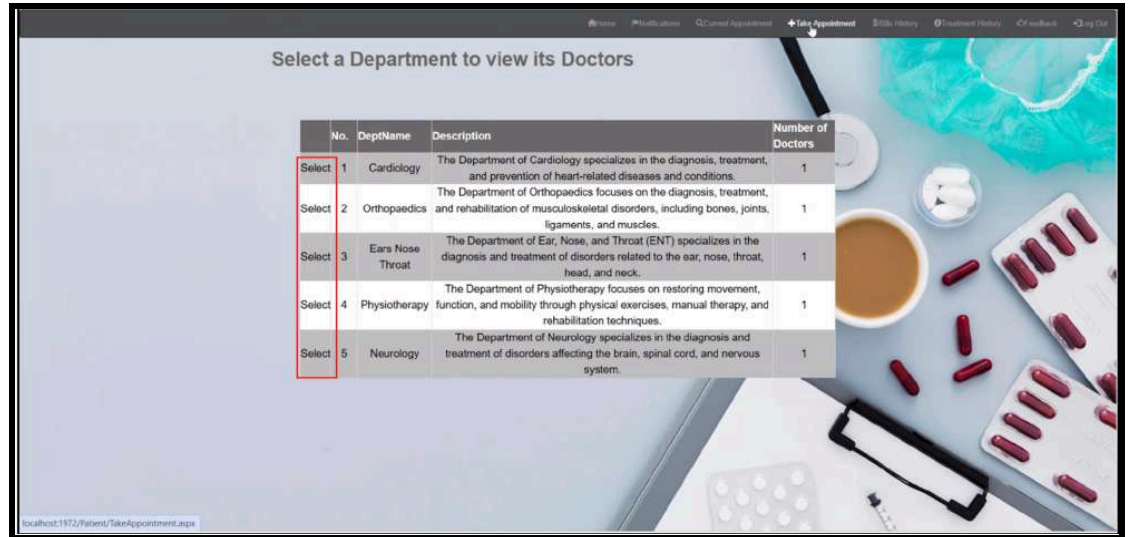
- **Sign Up Page/ Registration of New User**

The first Login page of the Health Sphere system, where the patient can log in/register in the Health Sphere System.



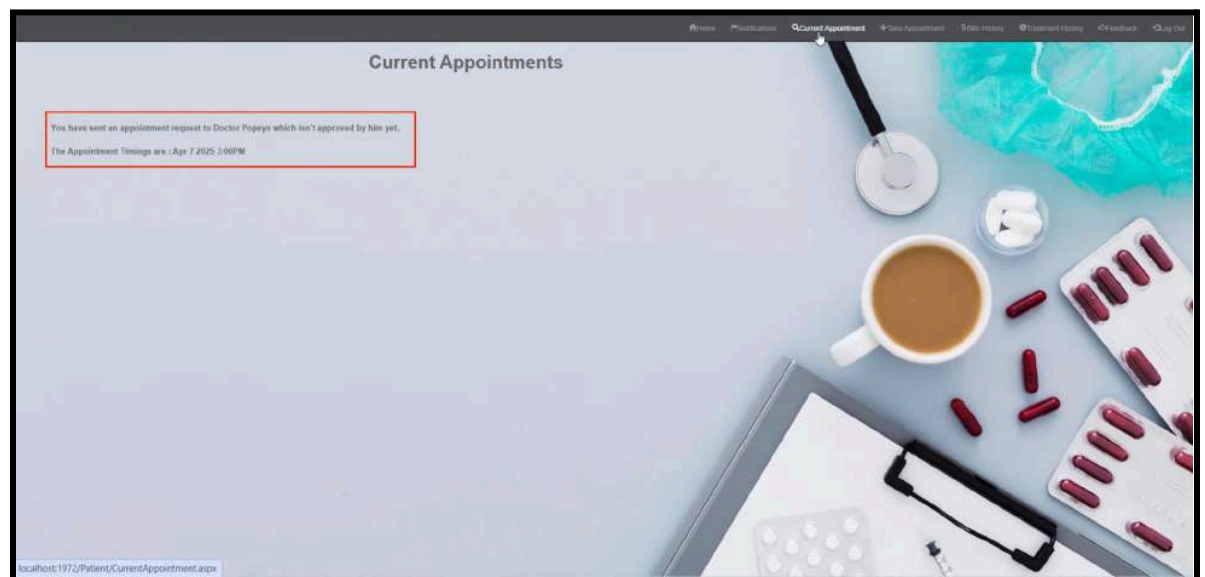
- **Take Appointment Page**

The description column in the user interface shown below specifies what the respective department deals with, when it comes to the treatment part. A patient will be able to select the department he/she need to get a treatment from by clicking on the Take Appointment tab in the right corner of the website.



- **Current Appointments**

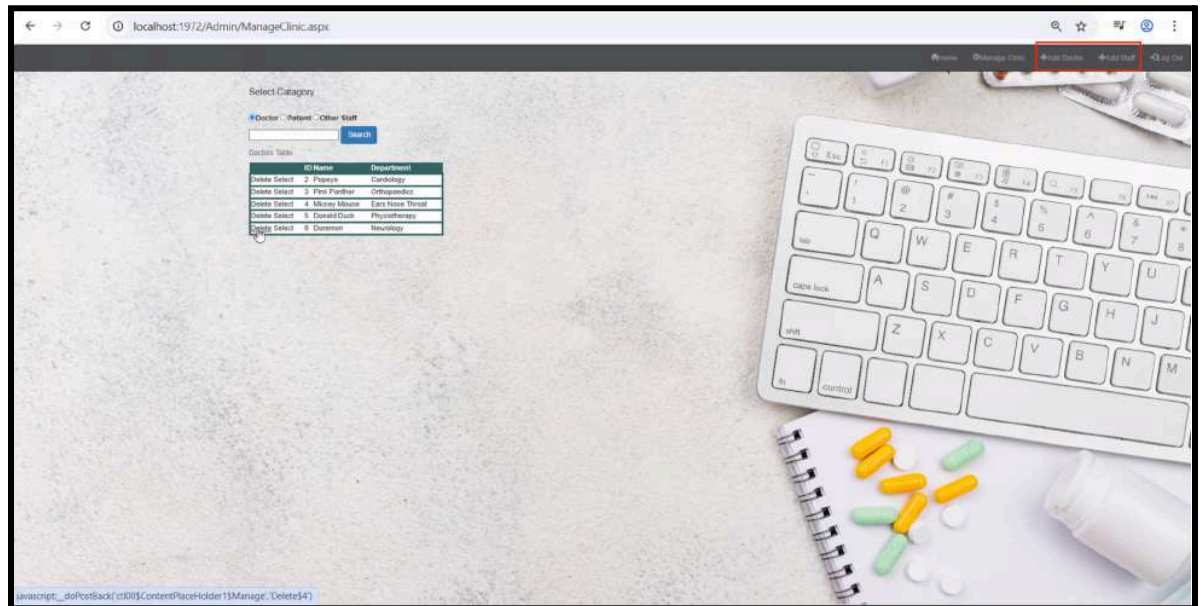
The Current Appointments Page redirects to the page that shows the status of the current appointment taken by the Patient, i.e., whether the requested appointment has been confirmed or is still pending with the Doctor, as well as the date and the timings too.



- **List of Doctors/Patients/Other Staff**



Only the Administrator has the role of viewing the total number of Patients/Doctors and Other Staff members registered with the Health Sphere system. The details of the users can also be viewed by the Administrator



# Sprint 1 Report – HealthSphere

- **Functionality Delivered at the End of Sprint 1**

By the end of Sprint 1, we completed the foundational setup of the HealthSphere system. The following functionalities are fully working:

- **Database setup:** A SQL Server Express-backed database with tables for users, patients, providers, departments, appointments, and management roles are successfully implemented and accessible.
- **User registration UI:** We created a web-based interface for registering patients, doctors, and admin staff into the system.
- **Authentication/Login:** Users can now log in using their credentials via a basic authentication form.
- **View and Update User Info:** Authenticated users can view their stored details and make updates via an editable UI table.

- **User Stories Successfully Implemented**

1. US#1: Create a HealthSphere database to manage users' information.
2. US#2: Create a UI to register users (patient/doctor/management team).
3. US#3: Create a UI to Login / authenticate the user.
4. US#4: Create a UI to show users' information.
5. US#5: Create a UI to update user information.

- **Changes, Refinements, and New Discoveries**

- **US#2 (Register Users):** Initially, it didn't specify validation logic, so we added basic input validation (e.g., email format, required fields).
- **US#4 and US#5 (View/Update):** Originally combined as a single view, we decided to split the logic for better separation of concerns. The "View" and "Edit" interfaces now use different modals/components. The "Edit/Update" option is currently within the Administrative roles.
- We identified a new story regarding the implementation of visual representation of patient's vitals in patient's portal (e.g., patient will be able to view their latest status on their vital right after logging into portal) to be implemented in future rollouts. We haven't added it in our Product Backlog yet, but a description of it will be mentioned in our 3rd sprint report and the User Manual.

- **Lessons Learned**

- **Plan for validation:** It's better to define UI validation rules upfront in the user stories to avoid rework.
- **Database-first helped:** Starting with the database schema helped clarify backend requirements early on.

- **Next time:** We'll aim to write more detailed user stories with pre/post-conditions and acceptance criteria defined more clearly.

- **Updated User Story Backlog**

User Story #	Description	Pre-condition	Post-condition
US#6	Show patient profile info	The patient must be registered and authenticated	The patient can view their profile only
US#7	Online appointment scheduling	The patient must be registered and authenticated	Patient can schedule appointments and receive confirmation & reminders
US#8	Access to medical records	The patient must be registered and authenticated	Patient can securely view their medical history
US#9	Appointment email reminders	The patient must have a scheduled appointment	Patient receives appointment reminder emails
US#10	Provider access to patient history	The provider must be registered and authenticated	The provider can view/update the patient's treatment plan
US#11	E-prescription	The provider must be registered and authenticated	Providers can prescribe medications electronically
US#12	Automated invoicing	The billing admin must be	Admin can generate

		registered and authenticated	invoices based on treatments
US#13	Receptionist registration and check-in	The receptionist must be registered and authenticated	The receptionist can register patients and view appointments

- **Plan for Sprint 2**

To align with the guideline of implementing roughly 1/3 of the backlog and to meet the deadline of the submission, we selected the following user stories for Sprint 2 based on their priority and dependencies.

- **Planned User Stories for Sprint 2:**

- US#6 (Show Patient Profile Information) – 5 points
- US#7 (Online Appointment Scheduling with Email Notifications) – 8 points
- US#8 (Secure Access to Medical Records) – 8 points
- US#9 (Appointment reminders to Patient) – 5 points
- US#10 (Patient's comprehensive Medical History) – 5 points
- US#11 (Streamline billing) – 8 points

- **Expected Functionality After Sprint 2:**

- Patients will be able to view their profiles, schedule appointments (with appointment confirmation and reminder alerts triggered to the Patient), and access medical history securely via the portal.
- These functionalities will help us move toward building a usable patient portal with core features that enhance user experience and autonomy.

## Sprint 2 Report – HealthSphere

- **Functionality at the End of Sprint 2**

At the end of Sprint 2, the HealthSphere system has evolved to support core patient-side features, contributing significantly to the usability and interactivity of the portal and also the billing of the treatment incurred upon the patient.

**Implemented functionalities include:**

- **Patient Profile View:** Patients can log in and view their personal information and health data.
- **Appointment Scheduling System:** Patients can schedule appointments with providers according to the provider's schedule, avoiding any time conflicts.
- **Medical Record Access:** Patients now have secure access to their past medical records, including treatments and prescriptions.
- **Notification alerts:** Patients receive notification alerts on confirmations after successfully booking appointments in the Patient portal.

- **User Stories Successfully Implemented**

- **US#6:** Show patient profile info
- **US#7:** Online appointment scheduling
- **US#8:** Access to medical records
- **US#9:** Appointment email alerts
- **US#10:** Provider access to patient medical history for informed treatment decisions
- **US#11:** Electronically prescribing medicines

- **Changes, Refinements, and New Discoveries**

- **US#7:** Online appointment scheduling(A patient is permitted to hold only one active appointment at any given time. Subsequent appointment requests are restricted until the existing appointment is completed).
- **US#9:** Appointment email reminders (A slight tweak has been done to this functionality, as once the provider confirms the appointment, the patient will come to know of it via the notification tab on their portal; this step has been taken for the ease of the patient in checking their confirmation)
- **US#10:** Both the patient and doctor can access the medical history of the patient via their respective portal.

- **US#11:** The Doctor will prescribe medicines directly from the portal to the patients, reducing the traditional paper usage.
- We identified a new story regarding the implementation of Payment Gateway from the Patient portal (e.g., patients will be able to pay online instead of paying offline via cash or any other methods) to be implemented in future rollouts. We haven't added it in our Product Backlog yet, but a description of it will be mentioned in our 3rd sprint report and the User Manual.

- **Lessons Learned**

- **Email notifications** should have been planned earlier.
- **Security:** Implementing secure medical record access emphasized the importance of encrypted endpoints and session validation.
- **Real-world alignment:** Many patients might reschedule, so rescheduling workflows should've been scoped earlier.

**Next time:** Include dependency mapping for features to reduce unexpected design effort (e.g., secure login, audit logs).

- **Updated User Story Backlog**

User Story #	Description	Pre-condition	Post-condition
US#12	Automate patient invoicing based on treatment and insurance details to streamline billing operations.	The billing executive must be registered in the system and authenticated.	Billing executives will be granted access to generate invoices
US#13	Enable easy patient registration and access to appointment details for a	Receptionists must be registered in the system and authenticated	Receptionists will be granted access to register patients and schedule appointments.

	smoother check-in process.		
--	----------------------------------	--	--

- **Plan for Sprint 3**

To align with the guideline of implementing roughly 2/3 of the backlog and to meet the deadline of the submission, we selected the following user stories for Sprint 3 based on their priority and dependencies.

- **Planned User Stories for Sprint 3:**

- US#12 (patient invoicing based on treatment) – 8 points
- US#13 (easy patient registration and access to appointment details for a smoother check-in process) – 8 points

- **Expected Functionality by End of Sprint 3**

- A bill concerning the treatment taken by the Patient should be generated along with the status of being paid/unpaid
- A person having the administrative role must be able to register a Patient/Doctor/Staff with ease without any difficulty.

## Sprint 3 Report – HealthSphere

- **Functionality at the End of Sprint 3**

At the end of Sprint 3, the HealthSphere system includes a fully operational Accounts Receivable module and administrative support via receptionist functionalities.

- Doctors can generate bills incurred for their patients based on the treatment provided to patients. They have the functionality to update the status of the payment as paid/Unpaid
- Admin users will be able to monitor the clinic's total income in their portal after the bill is paid by the patient.
- Receptionists (if acting as admin) can register new patients/providers/staff, schedule appointments on their behalf, and can remind them on time.

- **Implemented User Stories**

- User Story 12: Streamlining the Billing process.
- User Story 13: A streamlined process for accessing patient registration and appointments for receptionists.

- **Changes and Breakdown of User Stories**

During Sprint 3, some refinements have been performed on the original user stories:

- In User Story 12, the doctor has the provision of generating the treatment bill and also updating the payment status of the Patient as paid/Unpaid, which aligns better with the real-time workflow.
- Administrative role included monitoring the overall income generated by the hospital after successful payment of all the patients for the day.
- In User Story 13, the receptionist was granted elevated 'admin' role access, allowing broader control over the registration functionality of patient/doctor/staff and appointment scheduling of the patient.

These refinements were made as the sprint progressed, but were included within the original point allocation due to logical scope overlap.

- **Lessons Learned**

- Role-based access planning should be finalized earlier in the project to avoid mid-sprint role reassignments.
- Effective treatment bill generation logic helped ensure consistent data across multiple patients' profiles.



- We learned that real-world workflows (like doctors issuing invoices) often differ from initial assumptions and must be revised during sprint execution.
- In future projects, assigning concrete workflows to roles at the backlog stage would reduce ambiguity later.

- **Remaining User Stories in the Backlog**

Currently, there is no active product backlog since all scoped user stories have been implemented.

However, we have conceptualized two additional features for future development, part of which has also been discussed in Sprint reports 1 and 2, as described below.

- **Walkthrough of Two Additional Scenarios (Future Scope)**

- 1. Payment Gateway Integration:**

Currently, patients make payments manually (cash/debit/credit). A future enhancement is to integrate an online payment gateway.

- Patients will see a pending bill in their portal post-treatment and can pay online via card.
- Upon successful payment, the status automatically updates to 'Paid. '
- Admins will see this reflected in the income reports without manual entry by the doctor.
- This improves transparency, reduces errors, and enhances financial tracking.

- 2. Visual Representation of Vitals:**

Doctors currently input patient vitals manually, but there is no visual progression tracking.

- In the future, vitals like BP, sugar, weight, etc., can be plotted over time as bar/pie charts.
- These graphs would be accessible on both the patient and doctor dashboards.
- This would help track recovery trends or medical deterioration visually.

These features would significantly increase usability and value for both patients and medical staff.

- **Final Sprint Summary**

As this was the final sprint, the system met its initial core requirements.

The remaining user stories were noted as enhancements in future scope. Overall, Sprint 3 provided crucial administrative and financial control features, completing the minimal viable product for HealthSphere.

## **Health Sphere User Manual**

### **Installation/Setup Instructions:**

#### **1. Install the following prerequisites:**

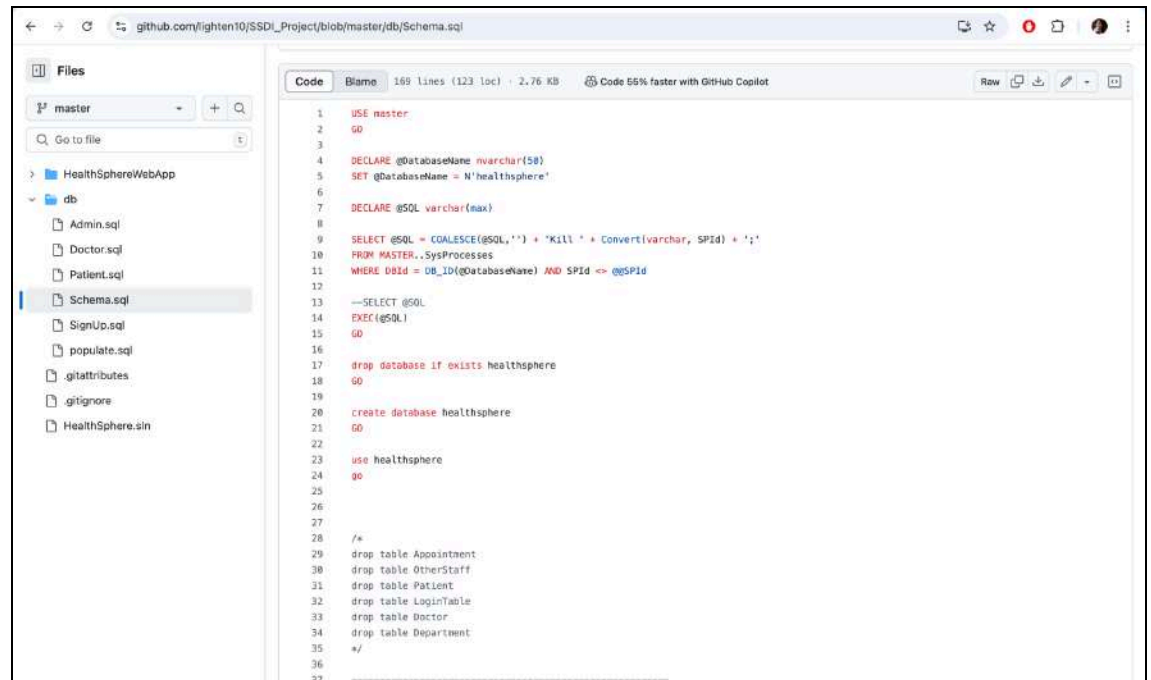
- Microsoft Visual Studio - Community edition
- Microsoft SQL Server Express
- Microsoft SQL Server Management Studio (SSMS)

#### **2. Connect to SQL Server:**

- Open **SQL Server Management Studio**
- In the "*Connect to Database Engine*" window, enter the following:
  - **Server Name:** .\SQL EXPRESS
  - **Authentication:** Windows Authentication

#### **3. Create the Database and Tables:**

- Open the Schema.sql file and execute it. This will create the necessary database and table structures.



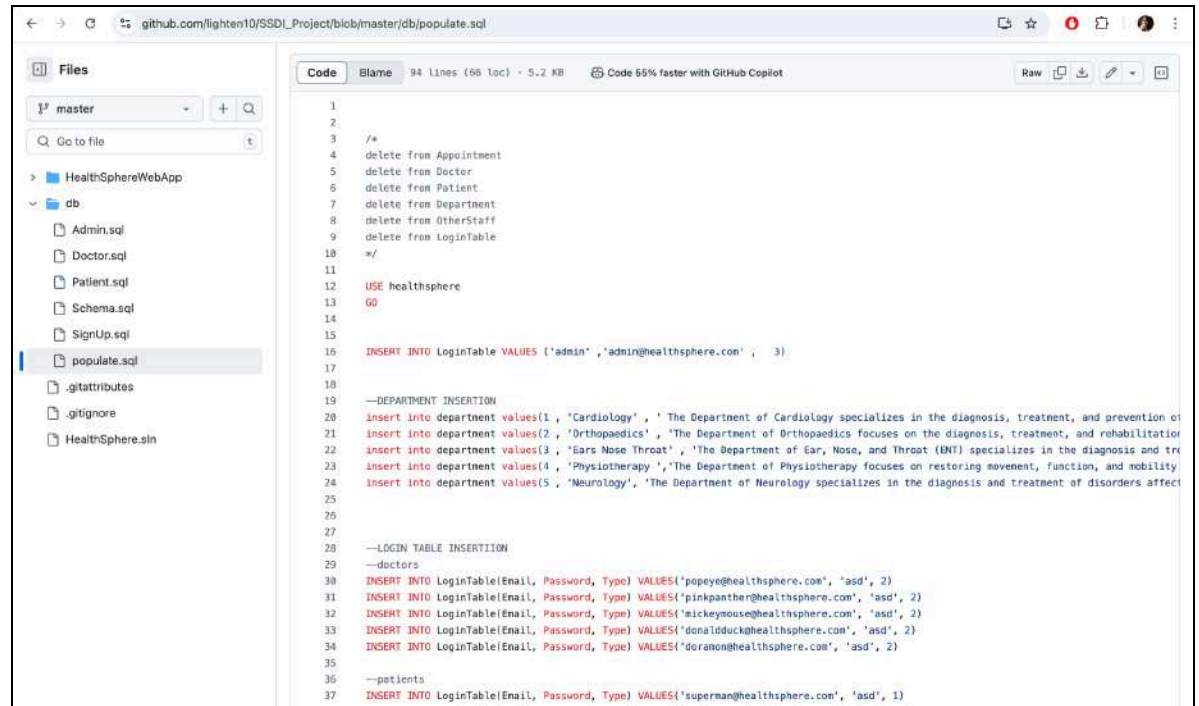
```
1  USE master
2  GO
3
4  DECLARE @DatabaseName nvarchar(50)
5  SET @DatabaseName = N'healthsphere'
6
7  DECLARE @SQL varchar(max)
8
9  SELECT @SQL = COALESCE(@SQL, '') + 'Kill ' + Convert(varchar, SPID) + ';'
10 FROM MASTER..SysProcesses
11 WHERE DBId = DB_ID(@DatabaseName) AND SPID <> @@SPID
12
13 --SELECT @SQL
14 EXEC(@SQL)
15 GO
16
17 drop database if exists healthsphere
18 GO
19
20 create database healthsphere
21 GO
22
23 use healthsphere
24 go
25
26
27
28 /*
29 drop table Appointment
30 drop table OtherStaff
31 drop table Patient
32 drop table LoginTable
33 drop table Doctor
34 drop table Department
35 */
36
37
```

- Then, execute the following SQL files in order:

- Admin.sql
- Doctor.sql
- Patient.sql
- Signup.sql

#### 4. Populate the Database:

- Run the populate.sql file to add sample data to the database.



```
1
2
3 /*
4 delete from Appointment
5 delete from Doctor
6 delete from Patient
7 delete from Department
8 delete from OtherStaff
9 delete from LoginTable
10 */
11
12 USE healthsphere
13 GO
14
15
16 INSERT INTO LoginTable VALUES ('admin', 'admin@healthsphere.com', 3)
17
18
19 --DEPARTMENT INSERTION
20 insert into department values(1, 'Cardiology', 'The Department of Cardiology specializes in the diagnosis, treatment, and prevention of
21 insert into department values(2, 'Orthopaedics', 'The Department of Orthopaedics focuses on the diagnosis, treatment, and rehabilitation
22 insert into department values(3, 'Ears Nose Throat', 'The Department of Ear, Nose, and Throat (ENT) specializes in the diagnosis and tre
23 insert into department values(4, 'Physiotherapy', 'The Department of Physiotherapy focuses on restoring movement, function, and mobility
24 insert into department values(5, 'Neurology', 'The Department of Neurology specializes in the diagnosis and treatment of disorders affect
25
26
27
28 --LOGIN TABLE INSERTION
29 --doctors
30 INSERT INTO LoginTable(Email, Password, Type) VALUES('popeye@healthsphere.com', 'asd', 2)
31 INSERT INTO LoginTable(Email, Password, Type) VALUES('pinkpanther@healthsphere.com', 'asd', 2)
32 INSERT INTO LoginTable(Email, Password, Type) VALUES('mickeymouse@healthsphere.com', 'asd', 2)
33 INSERT INTO LoginTable(Email, Password, Type) VALUES('donaldduck@healthsphere.com', 'asd', 2)
34 INSERT INTO LoginTable(Email, Password, Type) VALUES('doranon@healthsphere.com', 'asd', 2)
35
36 --patients
37 INSERT INTO LoginTable(Email, Password, Type) VALUES('superman@healthsphere.com', 'asd', 1)
```

- This file also contains sample login credentials for doctors, patients, and an admin, which you can use to test the system's functionality.

## 5. Run the Project:

- Open the Visual Studio solution file HealthSphere.sln.
- Set SignUp.aspx as the startup page.
- Click the **IIS Express** run button to launch the application.

---

## Technical Specification of the Health Sphere project



### System Requirements

- Operating System: Windows 10 or later (64-bit)
- Processor: Intel Core i5 or equivalent (minimum)

## Software Requirements

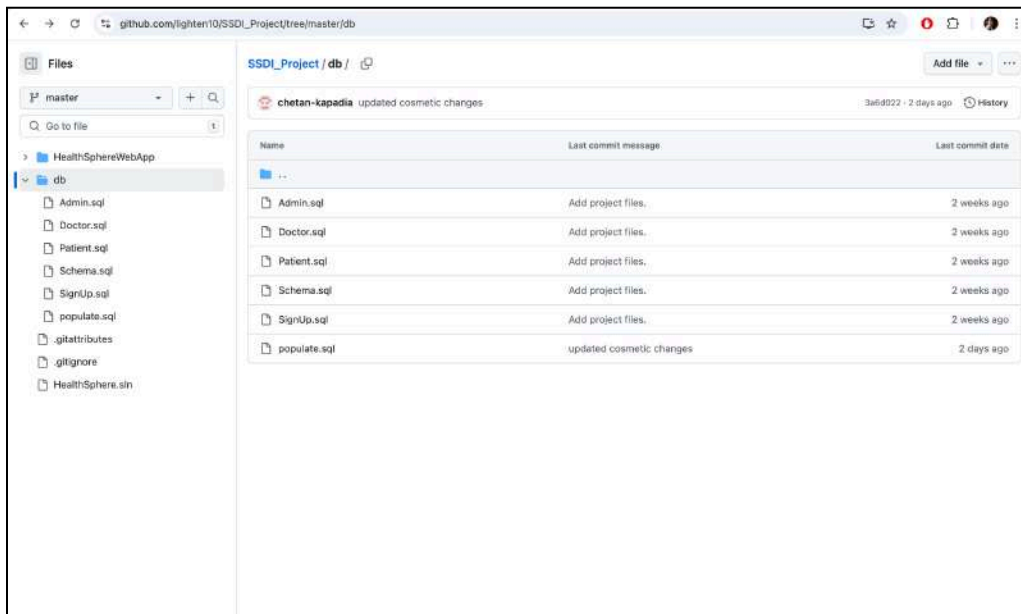
- Microsoft Visual Studio 2022 (or later)
  - Workload: ASP.NET and web development
- .NET Framework 4.7.2, Bootstrap(CSS, JS)
- Microsoft SQL Server Express 2019 (or later)
- SQL Server Management Studio (SSMS)

## Dependencies

- IIS Express (automatically included with Visual Studio)
- ASP.NET Web Forms
- SQL Server LocalDB or SQL Express instance
- Browsers: Google Chrome, Microsoft Edge (latest versions recommended)

## Project Structure

- Database Files - Admin.sql, Doctor.sql, Patient.sql, Schema.sql, SignUp.sql, Populate.sql



The screenshot shows a GitHub repository page for 'SSDI\_Project / db'. The left sidebar displays the file tree with the following files: Admin.sql, Doctor.sql, Patient.sql, Schema.sql, SignUp.sql, populate.sql, .gitattributes, .gitignore, and HealthSphere.sin. The main content area shows the commit history for the 'db' directory, with a table listing the files and their commit messages and dates.

Name	Last commit message	Last commit date
..		
Admin.sql	Add project files.	2 weeks ago
Doctor.sql	Add project files.	2 weeks ago
Patient.sql	Add project files.	2 weeks ago
Schema.sql	Add project files.	2 weeks ago
SignUp.sql	Add project files.	2 weeks ago
populate.sql	updated cosmetic changes	2 days ago

- Health Sphere Web App - 3 distinct roles (Admin's role, Doctor's role, and Patient's role)

github.com/lighten10/SSDL\_Project/tree/master/HealthSphereWebApp/Admin

lighten10 / SSDL\_Project

Type to search

Code Issues Pull requests Actions Projects Security Insights

Files

master

Go to file

HealthSphereWebApp

Admin

AddStaff.aspx

AddStaff.aspx.cs

AddStaff.aspx.designer.cs

Admin.Master

Admin.Master.cs

Admin.Master.designer.cs

AdminHome.aspx

AdminHome.aspx.cs

AdminHome.aspx.designer.cs

DoctorRegistrationForm.aspx

DoctorRegistrationForm.aspx.cs

DoctorRegistrationForm.aspx.d...

ManageClinic.aspx

ManageClinic.aspx.cs

ManageClinic.aspx.designer.cs

SSDL\_Project / HealthSphereWebApp / Admin /

chetan-kapadia updated cosmetic changes 3a6d022 · 2 days ago History

Name	Last commit message	Last commit date
..		
AddStaff.aspx	updated cosmetic changes	2 days ago
AddStaff.aspx.cs	Add project files.	2 weeks ago
AddStaff.aspx.designer.cs	Add project files.	2 weeks ago
Admin.Master	updated cosmetic changes	2 days ago
Admin.Master.cs	Add project files.	2 weeks ago
Admin.Master.designer.cs	Add project files.	2 weeks ago
AdminHome.aspx	updated cosmetic changes	2 days ago
AdminHome.aspx.cs	Add project files.	2 weeks ago
AdminHome.aspx.designer.cs	Add project files.	2 weeks ago
DoctorRegistrationForm.aspx	updated cosmetic changes	2 days ago
DoctorRegistrationForm.aspx.cs	Add project files.	2 weeks ago
DoctorRegistrationForm.aspx.designer.cs	Add project files.	2 weeks ago

github.com/lighten10/SSDL\_Project/tree/master/HealthSphereWebApp/Doctor

lighten10 / SSDL\_Project

Type to search

Code Issues Pull requests Actions Projects Security Insights

Files

master

Go to file

HealthSphereWebApp

DAL

Doctor

Bill.aspx

Bill.aspx.cs

Bill.aspx.designer.cs

DoctorHome.aspx

DoctorHome.aspx.cs

DoctorHome.aspx.designer.cs

DoctorMaster.Master

DoctorMaster.Master.cs

DoctorMaster.Master.designer.cs

HistoryUpdate.aspx

HistoryUpdate.aspx.cs

HistoryUpdate.aspx.designer.cs

PatientHistory.aspx

PatientHistory.aspx.cs

PatientHistory.aspx.designer.cs

SSDL\_Project / HealthSphereWebApp / Doctor /

chetan-kapadia updated cosmetic changes 3a6d022 · 2 days ago History

Name	Last commit message	Last commit date
..		
Bill.aspx	updated cosmetic changes	2 days ago
Bill.aspx.cs	updated cosmetic changes	2 days ago
Bill.aspx.designer.cs	updated cosmetic changes	2 days ago
DoctorHome.aspx	updated background images and default doctor's content	last week
DoctorHome.aspx.cs	Add project files.	2 weeks ago
DoctorHome.aspx.designer.cs	Add project files.	2 weeks ago
DoctorMaster.Master	updated cosmetic changes	2 days ago
DoctorMaster.Master.cs	Add project files.	2 weeks ago
DoctorMaster.Master.designer.cs	Add project files.	2 weeks ago
HistoryUpdate.aspx	updated cosmetic changes	2 days ago
HistoryUpdate.aspx.cs	Add project files.	2 weeks ago
HistoryUpdate.aspx.designer.cs	Add project files.	2 weeks ago
PatientHistory.aspx	updated cosmetic changes	2 days ago
PatientHistory.aspx.cs	Add project files.	2 weeks ago
PatientHistory.aspx.designer.cs	Add project files.	2 weeks ago

← → ↻ github.com/lighten10/SSDL\_Project/tree/master/HealthSphereWebApp/Patient

Files

master

Go to file

PatientHistory.aspx.designer.cs

PendingAppointment.aspx

PendingAppointment.aspx.cs

PendingAppointment.aspx.designer.cs

PreviousHistory.aspx

PreviousHistory.aspx.cs

PreviousHistory.aspx.designer.cs

Patient

AppointmentRequestSent.aspx

AppointmentRequestSent.aspx.cs

AppointmentRequestSent.aspx.designer.cs

AppointmentTaker.aspx

AppointmentTaker.aspx.cs

AppointmentTaker.aspx.designer.cs

BillsHistory.aspx

BillsHistory.aspx.cs

BillsHistory.aspx.designer.cs

CurrentAppointment.aspx

CurrentAppointment.aspx.cs

CurrentAppointment.aspx.designer.cs

DoctorProfile.aspx

DoctorProfile.aspx.cs

DoctorProfile.aspx.designer.cs

SSDL\_Project / HealthSphereWebApp / Patient /

chetan-kapadia updated cosmetic changes 3af6022 · 2 days ago

Name	Last commit message	Last commit date
..		
AppointmentRequestSent.aspx	Add project files.	2 weeks ago
AppointmentRequestSent.aspx.cs	Add project files.	2 weeks ago
AppointmentRequestSent.aspx.designer.cs	Add project files.	2 weeks ago
AppointmentTaker.aspx	updated cosmetic changes	2 days ago
AppointmentTaker.aspx.cs	Add project files.	2 weeks ago
AppointmentTaker.aspx.designer.cs	Add project files.	2 weeks ago
BillsHistory.aspx	updated cosmetic changes	2 days ago
BillsHistory.aspx.cs	Add project files.	2 weeks ago
BillsHistory.aspx.designer.cs	Add project files.	2 weeks ago
CurrentAppointment.aspx	Add project files.	2 weeks ago
CurrentAppointment.aspx.cs	Add project files.	2 weeks ago
CurrentAppointment.aspx.designer.cs	Add project files.	2 weeks ago
DoctorProfile.aspx	updated cosmetic changes	2 days ago
DoctorProfile.aspx.cs	Add project files.	2 weeks ago
DoctorProfile.aspx.designer.cs	Add project files.	2 weeks ago

- The application starts from SignUp.aspx.

## 🔑 Sample Login Credentials (Test Users)

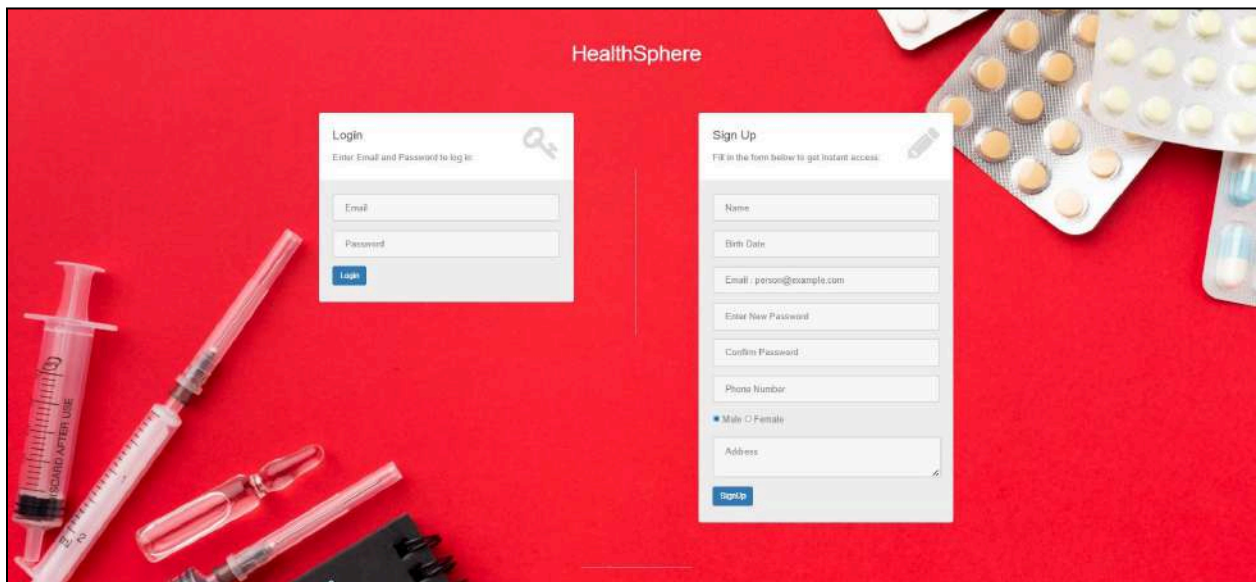
These are located in populate.sql, including credentials for:

- **Admin**
- **Doctors**
- **Patients**

These can be used to test role-based functionalities such as login, dashboard access, and other features.

## 🚀 Running the Application

- Open HealthSphereWebApp in Visual Studio.
- Set **SignUp.aspx** as the startup page.



- Click the **IIS Express** (Run) button.
  - The application will launch in your default browser.
-



## **Main Features of the System**

**Health Sphere** is an integrated digital healthcare management system developed to improve communication and coordination among patients, healthcare providers, and administrative personnel. The platform offers a secure and user-friendly interface that supports efficient user management, seamless appointment scheduling, and comprehensive access to medical history and billing information. By centralizing these essential healthcare functions, Health Sphere enhances the overall quality of care, simplifies administrative tasks, and contributes to more effective and streamlined healthcare delivery.

The **patient portal** offers a range of features designed to enhance user experience and streamline medical interactions. Upon logging in, patients can access the *Patient Home*, which displays their personal profile. In the *Current Appointment* section, patients can check for any pending or approved appointments with doctors. The *Bills History* tab provides a record of bills related to completed appointments, while the *Treatment History* section allows patients to view detailed information about past treatments. To schedule a new visit, patients can navigate to the *Take Appointment* section, where they can browse all available departments, select a specific department to view the associated doctors, and then choose a doctor to view their profile. From there, they can click the "Take Appointment" button to see the doctor's available time slots, select a preferred slot, and send a request to the doctor, who can then approve or reject the appointment. Notifications regarding the status of appointment requests are displayed under the *Notifications* tab. After a consultation is completed, patients are encouraged to provide feedback by rating their experience from 1 to 5 in the *Feedback* section. Importantly, the system allows patients to request only one appointment at a time; additional appointments cannot be scheduled until the current one has been completed.

The **doctor's portal** is designed to facilitate efficient management of appointments and patient records. Through the *Doctor Profile* section, doctors can view their personal and professional information. In the *Pending Appointments* tab, they can access a list of all appointment requests awaiting their approval, filtered by their doctor ID. The *Today's Appointments* section displays all appointments scheduled for the current day, allowing the doctor to either accept or reject each appointment as needed. After a consultation, the *History Update* feature enables the doctor to record key details such as prescriptions, diagnosed diseases, and the patient's progress. Following this, the doctor can use the *Generate Bill* section to create and issue the bill for the completed appointment. Additionally, the *Patient History* tab provides access to the full treatment history of all patients the doctor has previously treated, ensuring continuity of care and easy reference.

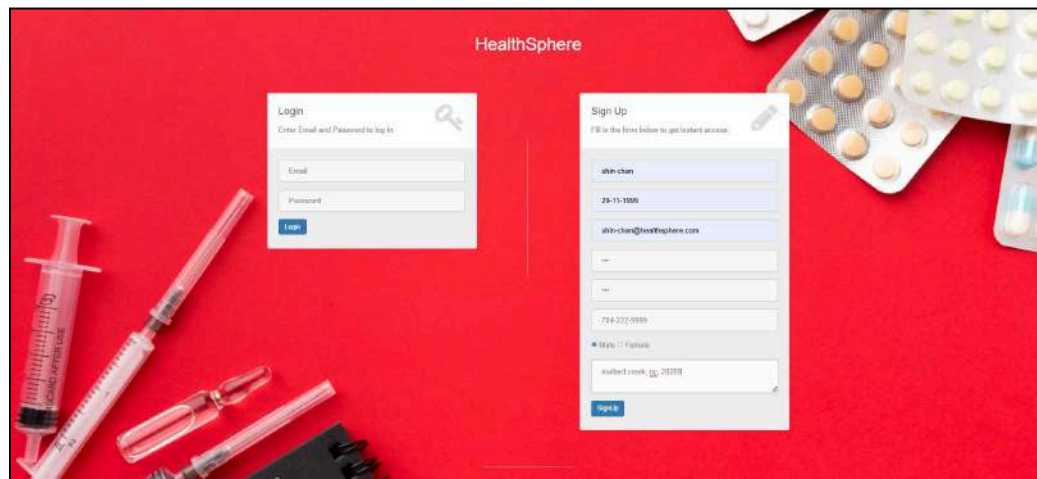
The **admin portal** provides a comprehensive interface for managing HealthSphere operations and overseeing all registered users. On the *Admin Home* page, the administrator can view key health center statistics, including the number of weekly appointments, overall income of the health center, the total number of registered patients and doctors, as well as a list of all medical departments. The *View Doctors* section displays a list of all currently registered doctors along with their associated departments and other relevant details; clicking on a doctor's name reveals their full profile. Similarly, the *View Patients* tab allows the admin to browse through the list of registered patients, including their phone numbers and IDs, with full profile access upon selection. The *View Other Staff* section lists all non-medical personnel along with their respective designations. To streamline navigation, a *Search Box* is available, enabling the admin

to quickly locate any employee by name. Additionally, the *Add/Remove* feature allows the admin to manage Health Center personnel by adding or removing doctors, patients, or staff members as needed.

---

## **Walkthrough for the main scenarios**

- To access Health Sphere, the first page is the **HealthSphere signUp** page as shown below. Doctors, Patients or Administrative personnel can use this page accordingly for either sign up or login to HealthSphere.



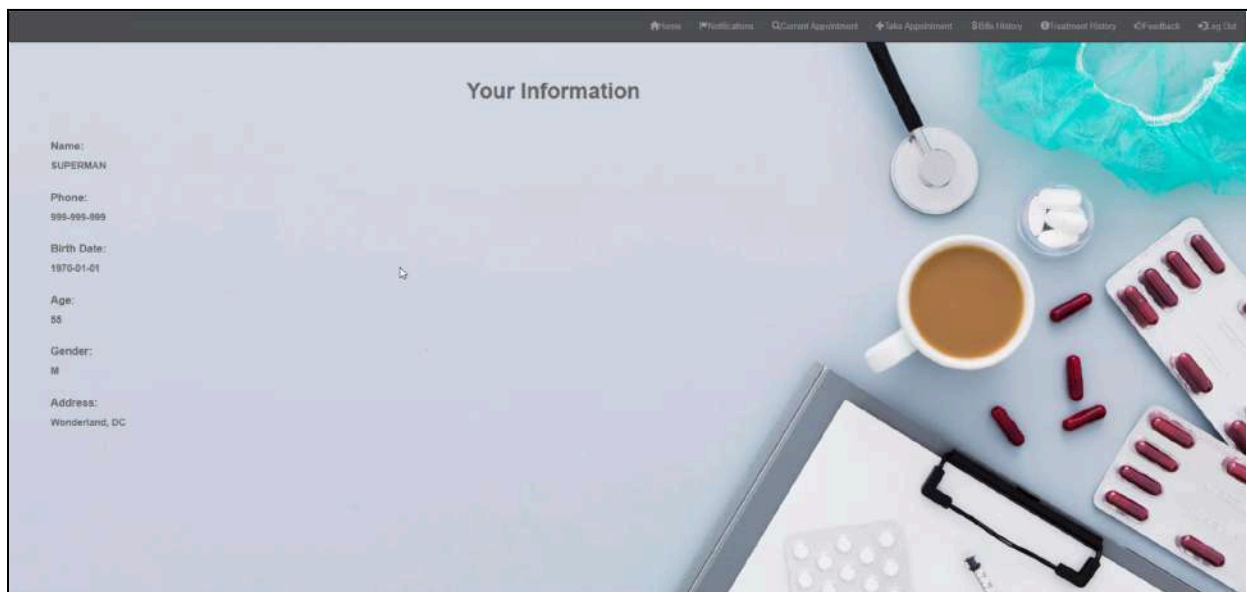
Below is the written code for it:

```

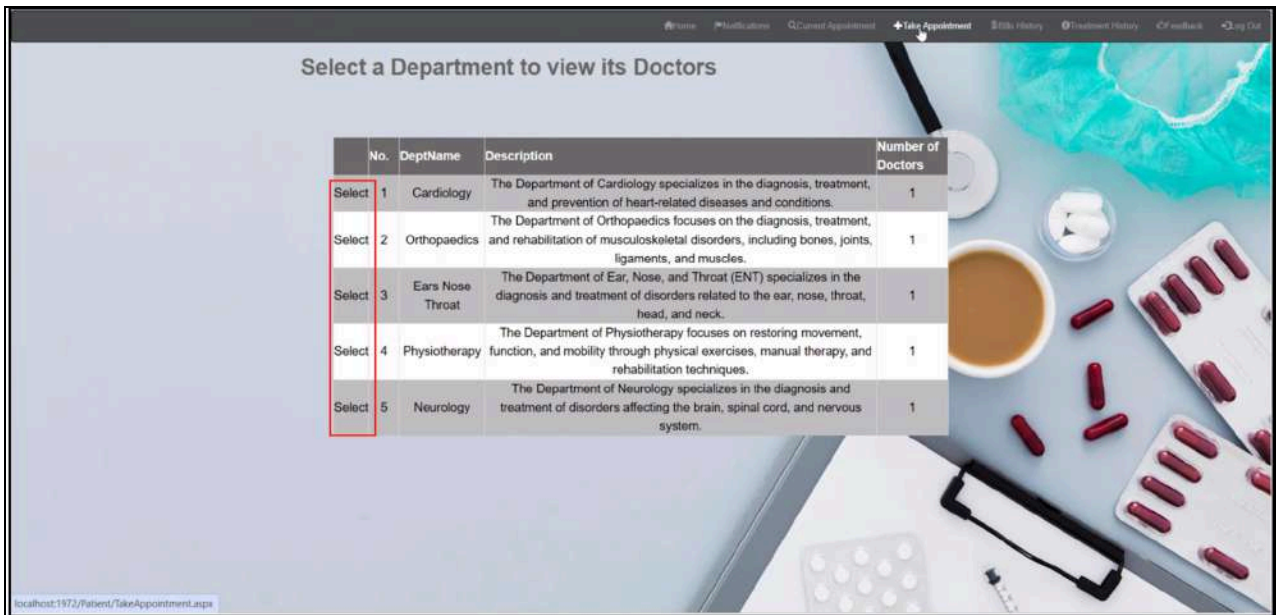
1 <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="SignUp.aspx.cs" Inherits="HealthSphereWebApp.SignUp" %>
2
3 <!DOCTYPE html>
4
5 <html xmlns="http://www.w3.org/1999/xhtml">
6 <head runat="server">
7
8     <title>HealthSphere Login & Register</title>
9
10
11     <script type="text/javascript">
12
13         //-----Function1-----//
14         function validateEmail(email) {
15             if (email == "") {
16                 alert("Email missing. Enter Email.");
17                 return false;
18             }
19
20             else if (email.indexOf("@") == -1 || email.indexOf(".") == -1) {
21                 alert("Your email address seems incorrect. Please enter a new one.");
22                 return false;
23             }
24
25             else {
26                 var location = email.indexOf("@");
27
28                 if (email[0] == '@' || email[location + 1] == '.') {
29                     alert("Your email address seems incorrect. Please enter a new one.");
30                     return false;
31                 }
32
33                 var emailPat = /^[^()\\[\]{};:'".,:~\s@!-]+(\.[^()\\[\]{};:'".,:~\s@!-]+)*@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.
34                 var EmailmatchArray = Email.match(emailPat);
35
36                 if (EmailmatchArray == null) {
37

```

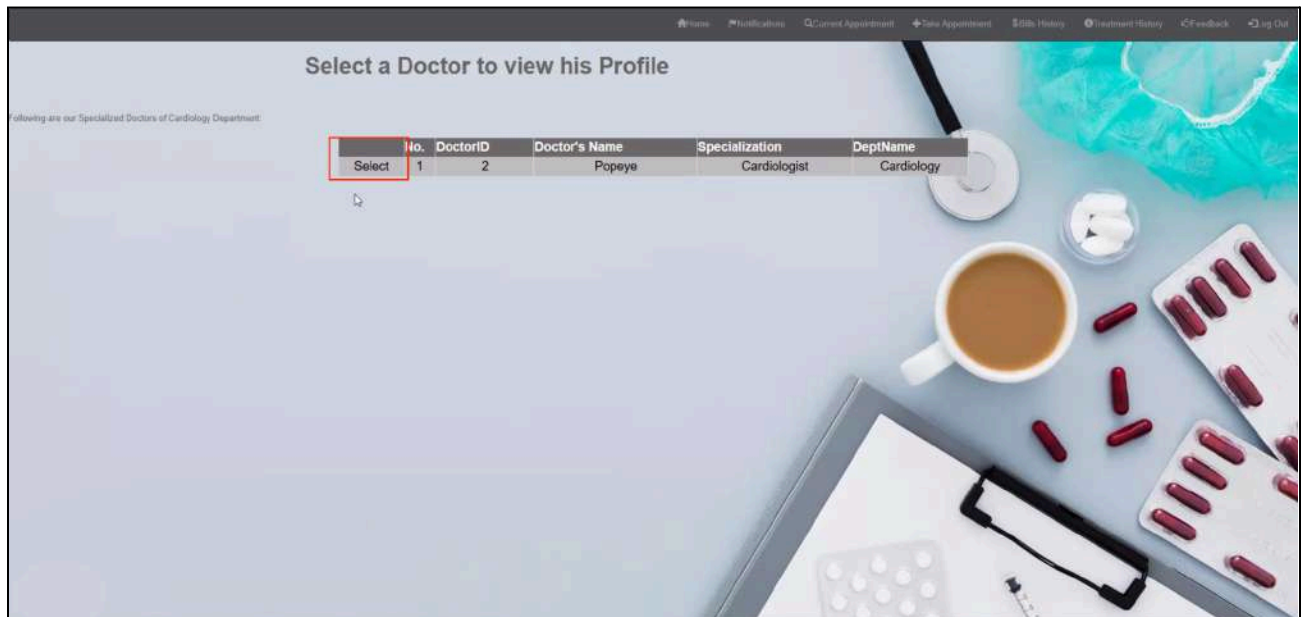
- **Patients** gain access to multiple facilities upon registering on the HealthSphere website. Below is the landing page of a patient named “SUPERMAN” after he logs into the HealthSphere website.



- **Patients** can take appointments as per their need by clicking on the “**Take Appointment**” button. A list of departments are then displayed after which the patient is allowed to select a particular department with the help of the “**Select**” option.




- **Patients** will be able to see the doctor for the department he/she has chosen and also his profile as shown below. Once the patient is sure about the doctor, the patient can then select a free time according to the doctor's schedule and send an appointment request to the doctor.



### Doctor's Profile

Name: Popeye  
Phone: 123456789  
Qualification: MD  
Specialization: Cardiologist  
Work Experience: 10  
Age: 35  
Gender: M  
Department: Cardiology  
Charges Per Appointment: 2500  
Repute Index: 4  
Number of Patients Treated: 1

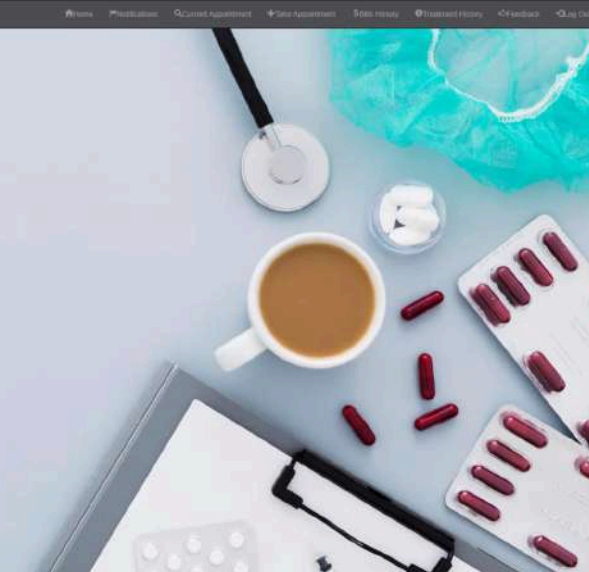
[Take Appointment](#)



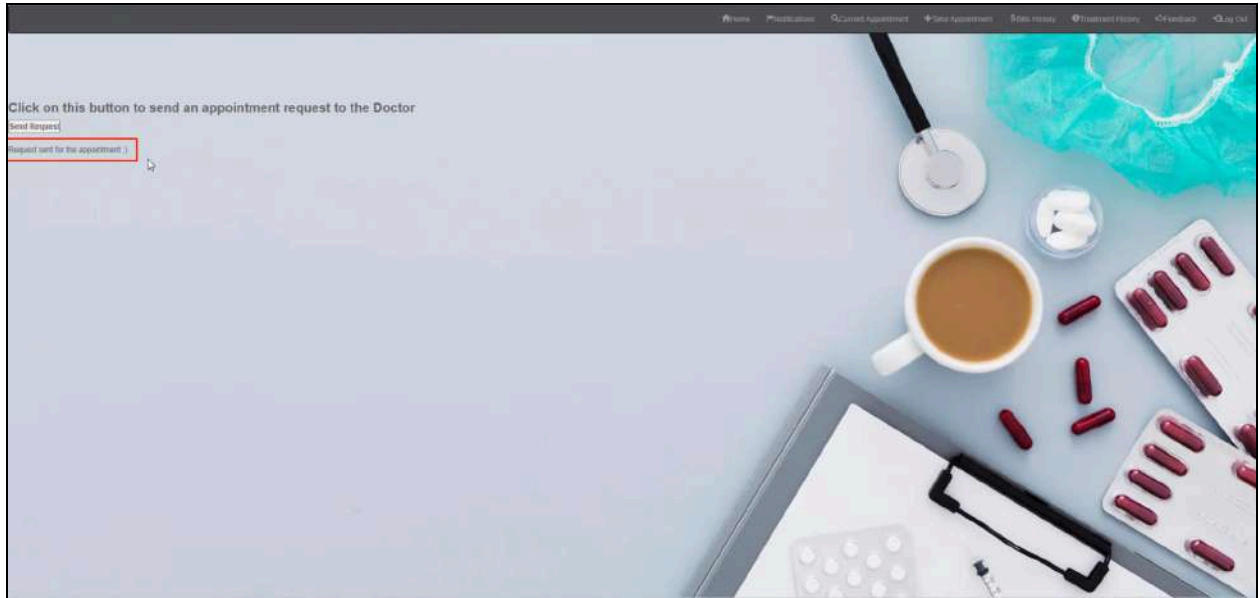
[Home](#) [Profile](#) [Quoted Appointment](#) [Take Appointment](#) [Send request](#) [Treatment History](#) [Feedback](#) [Logout](#)

Click on this button to send an appointment request to the Doctor

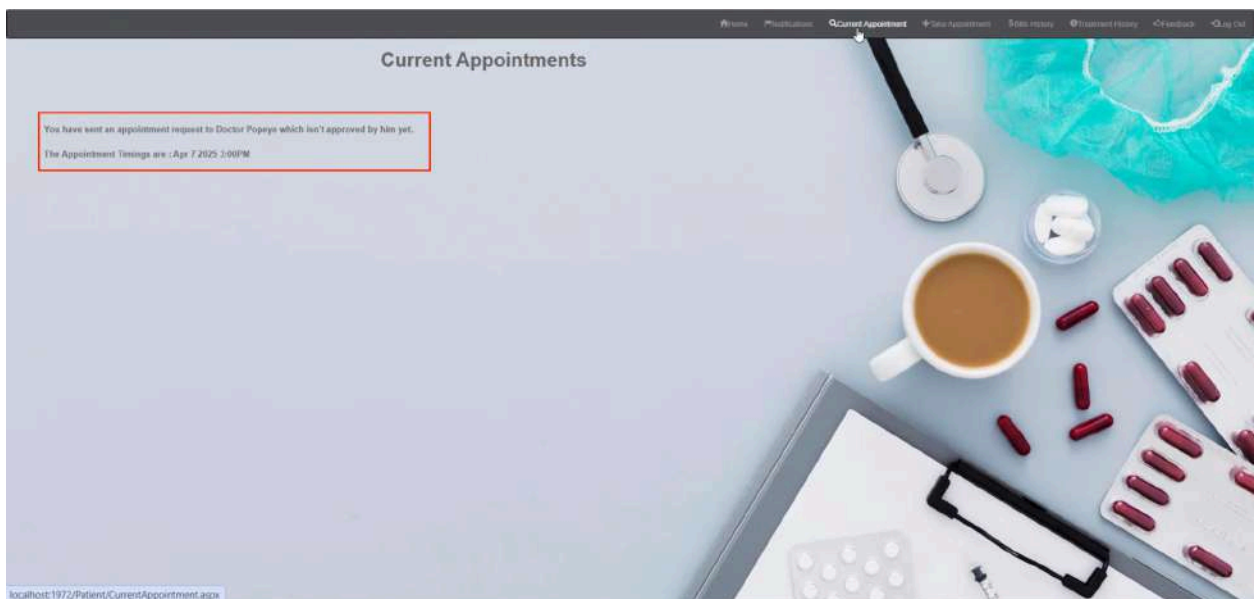
[Send Request](#)



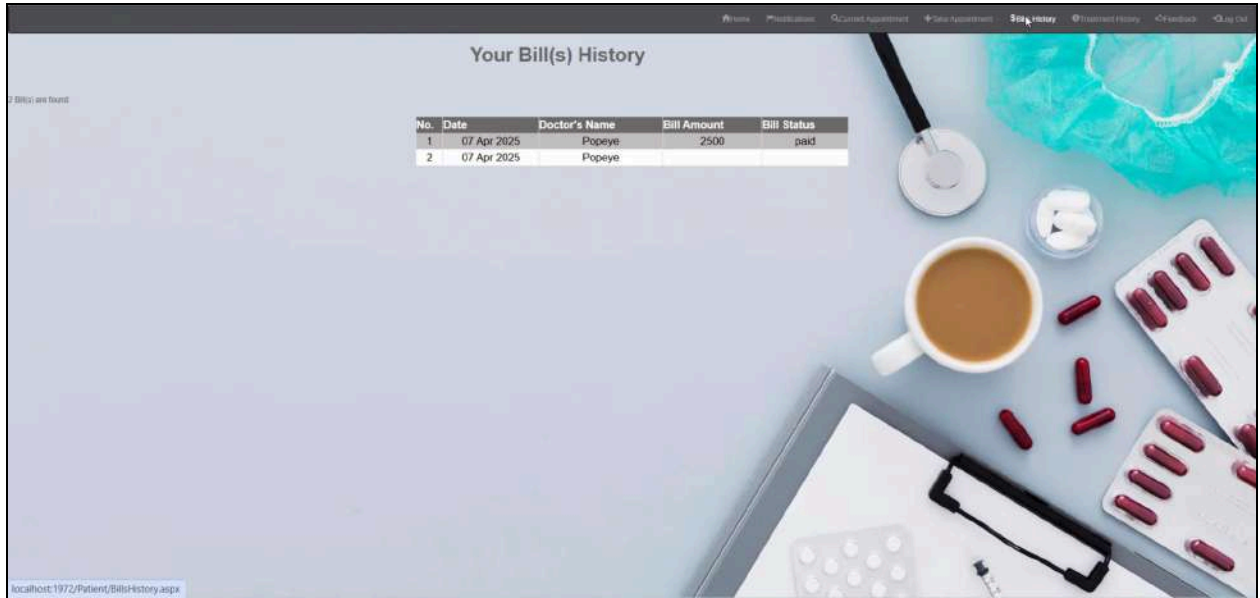




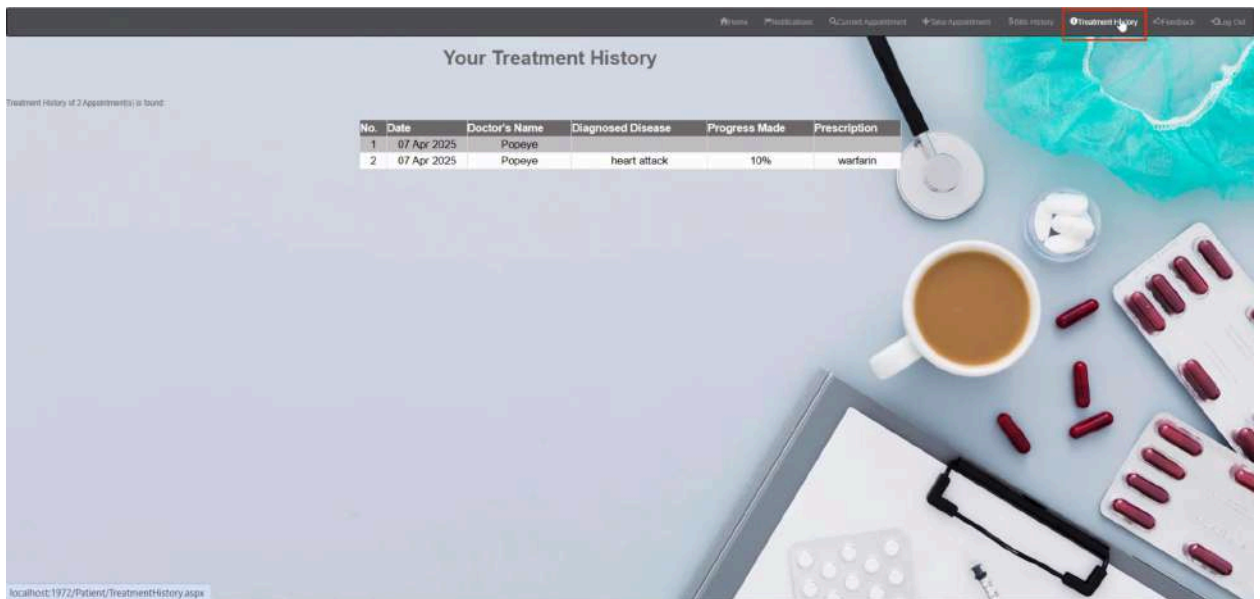
- **Patients** will be able to check their current appointments with any of the doctors.



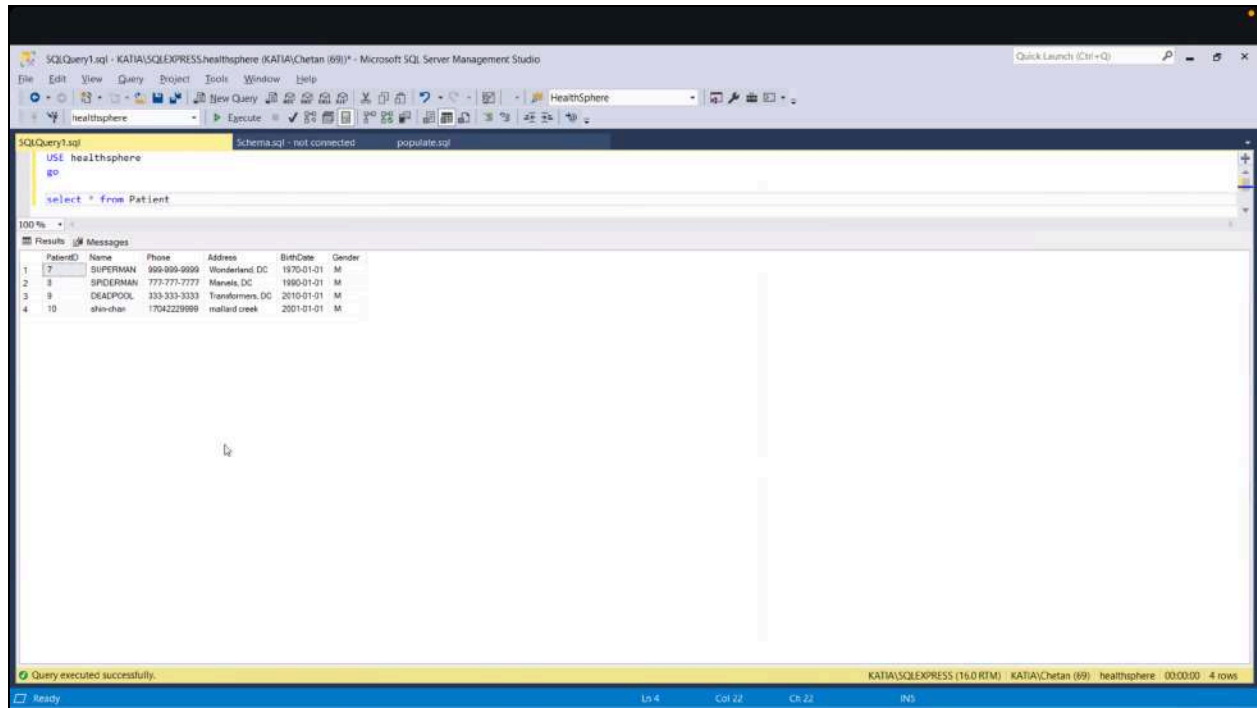
- **Patients** will be able to check the Bills generated on their name by clicking the “**Bills History**” button. The bills are generated based on the treatment they have requested for and also the prior treatments they have undergone.



- **Patients** will be able to check their treatment history from their respective doctors by clicking on the “**Treatment history**” button. Patients will be aware of the medicines prescribed by the doctor, the progress rate of the patient’s case, as well as the diagnosed disease.



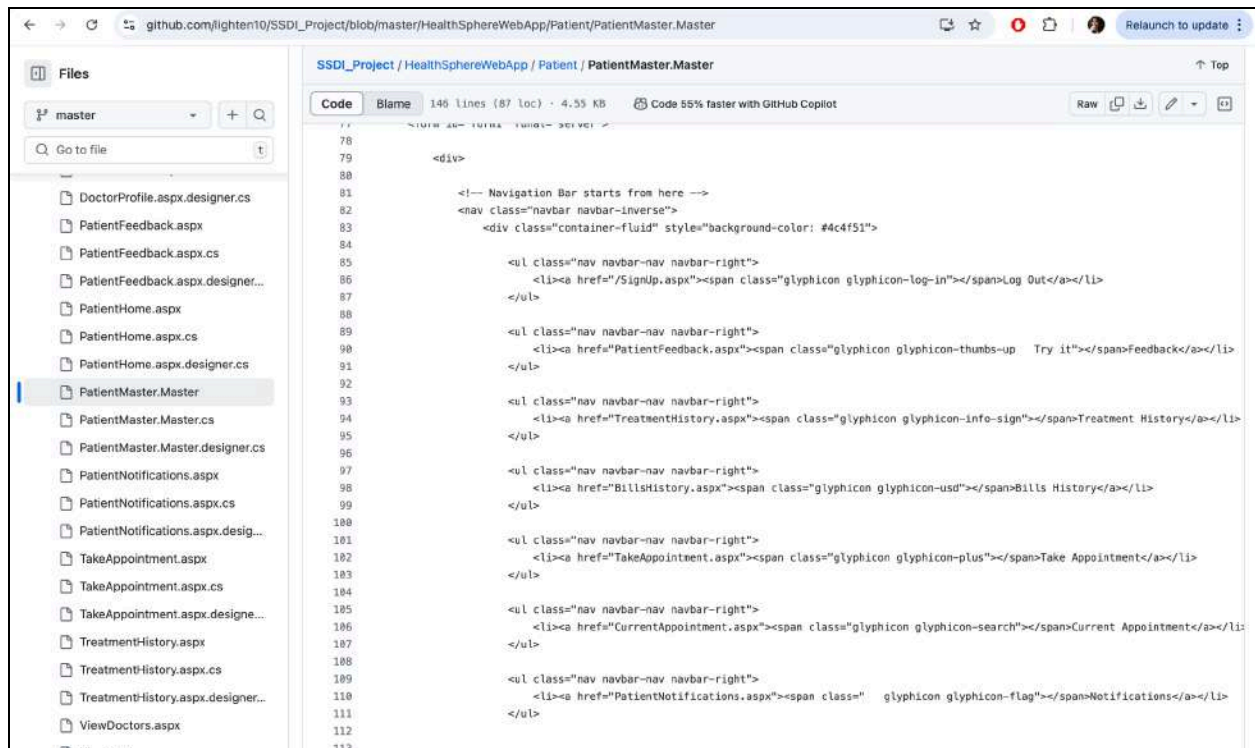
**Database Schema for the Patient record:**



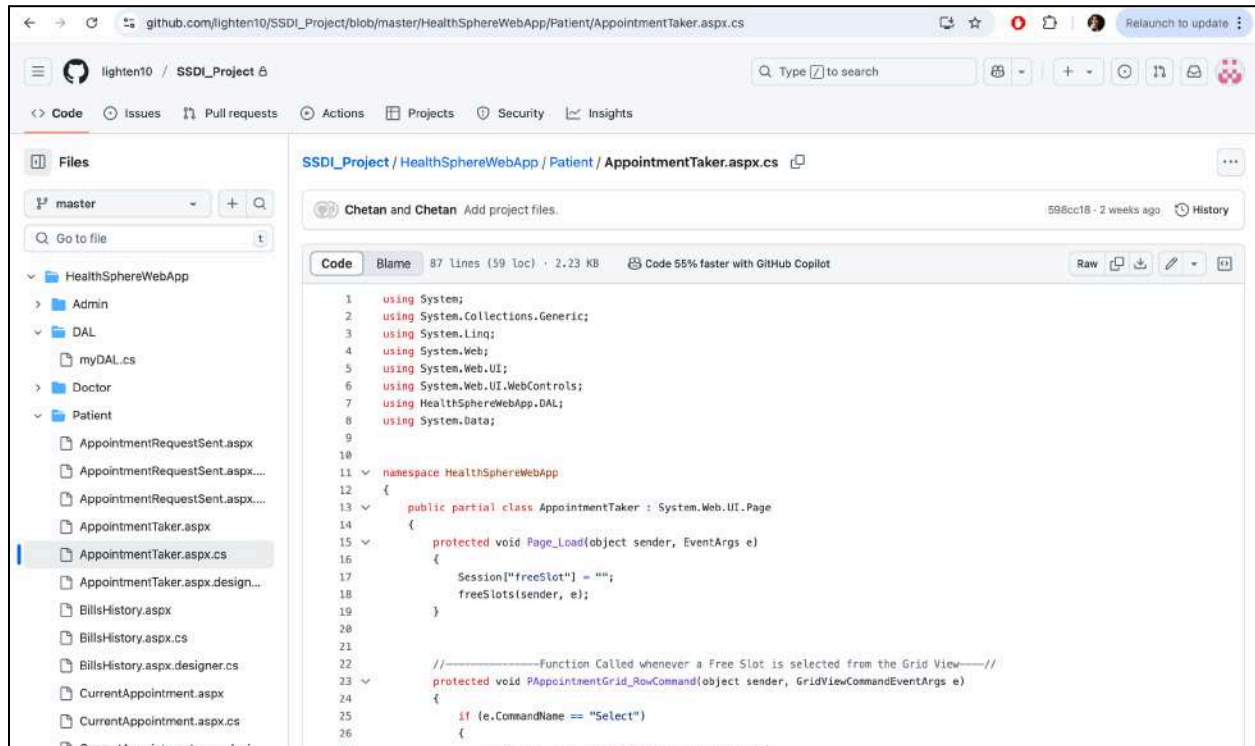
The shown screenshots depict a few of the code fragments of the main scenarios of the **Patient's** facilities.

## PatientMaster





## AppointmentTaker.aspx



## AppointmentRequestSent.aspx

github.com/IlighTen10/SSDI\_Project/blob/master/HealthSphereWebApp/Patient/AppointmentRequestSent.aspx.cs

SSDI\_Project / HealthSphereWebApp / Patient / AppointmentRequestSent.aspx.cs

Code Blame 62 lines (38 loc) · 1.23 KB Code 55% faster with GitHub Copilot

```
11 namespace HealthSphereWebApp
12 {
13     public partial class AppointmentNotificationSent : System.Web.UI.Page
14     {
15         protected void Page_Load(object sender, EventArgs e)
16         {
17         }
18     }
19
20 //-----Function1-----//
21
22 protected void sendAResult (object sender, EventArgs e)
23 {
24     myDAL objmyDAL = new myDAL();
25
26     string dID1 = (string)Session["dID"];
27
28     int dID = Convert.ToInt32(dID1);
29
30     int pID = (int)Session["idoriginal"];
31
32     string temp = (string)Session["freeSlot"];
33
34     int freeSlot = Convert.ToInt32(temp);
35
36     string mes = "";
37
38     int status = objmyDAL.insertAppointment(dID, pID, freeSlot, ref mes);
39
40     if (status == -1)
41     {
42     }
43 }
```

## BillsHistory.aspx

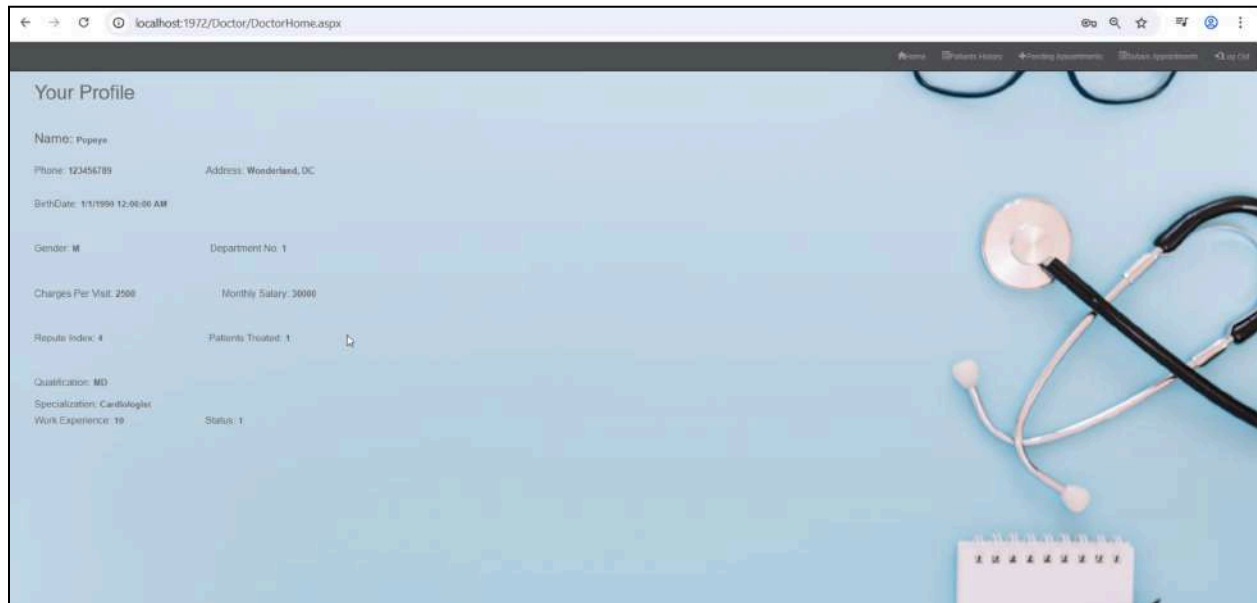
github.com/IlighTen10/SSDI\_Project/blob/master/HealthSphereWebApp/Patient/BillsHistory.aspx.cs

SSDI\_Project / HealthSphereWebApp / Patient / BillsHistory.aspx.cs

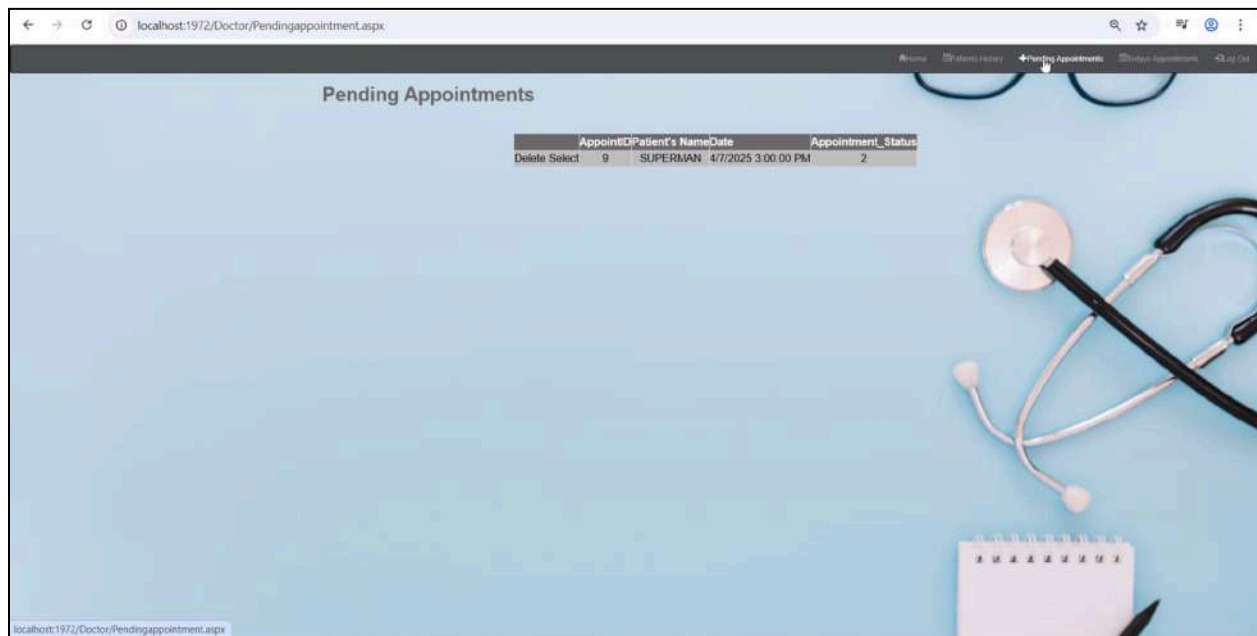
Code Blame 62 lines (42 loc) · 1.34 KB Code 55% faster with GitHub Copilot

```
11 namespace HealthSphereWebApp
12 {
13     public partial class BillsHistory : System.Web.UI.Page
14     {
15         billHistory(sender, e);
16     }
17
18 //-----Function1-----//
19
20 protected void billHistory(object sender, EventArgs e)
21 {
22     myDAL objmyDAL = new myDAL();
23
24     DataTable DT = new DataTable();
25
26     int id = (int)Session["idoriginal"];
27
28     int status = objmyDAL.getBillsHistory(id, ref DT);
29
30     if (status == -1)
31     {
32         BHistory.Text = "There was some error in retrieving the Patient's Bill History.";
33     }
34
35     else if (status == 0)
36     {
37         BHistory.Text = "There is currently no bill history of yours.";
38     }
39
40     else
41     {
42         BHistory.Text = status + " Bill(s) are found: ";
43         BHistoryGrid.DataSource = DT;
44         BHistoryGrid.DataBind();
45     }
46 }
```

- **After logging in, Doctor's** landing page will be his/her own profile page that can be viewed as below:



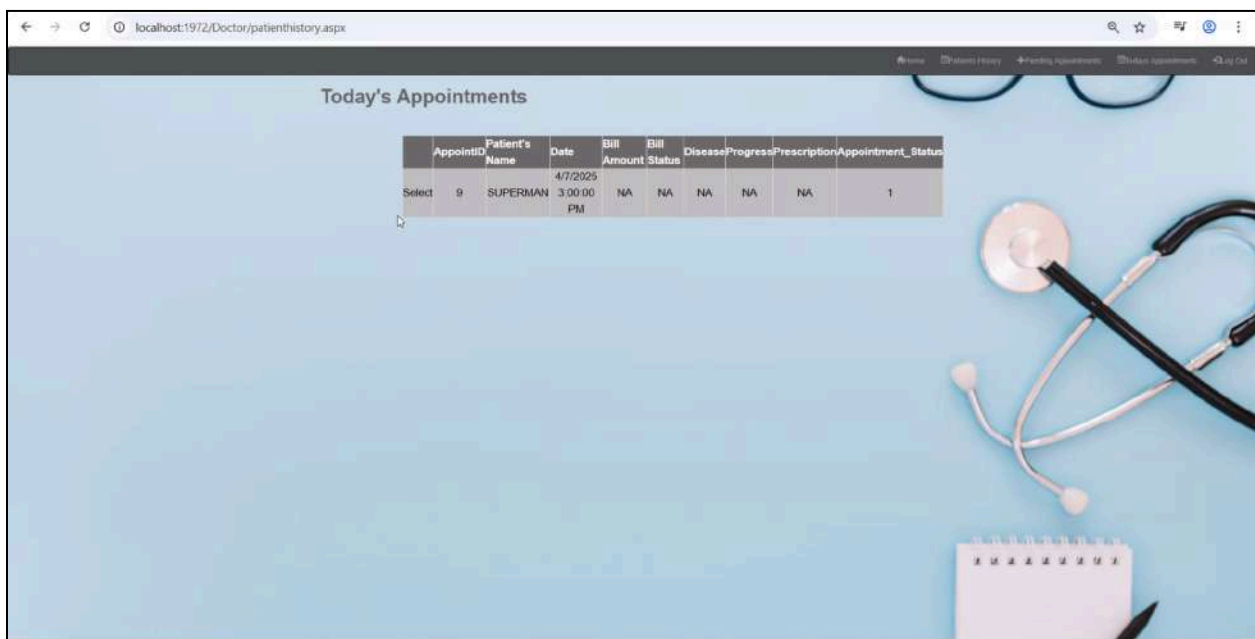
- **The doctor** will be able to see the pending appointments by clicking on the “Pending Appointments” button.

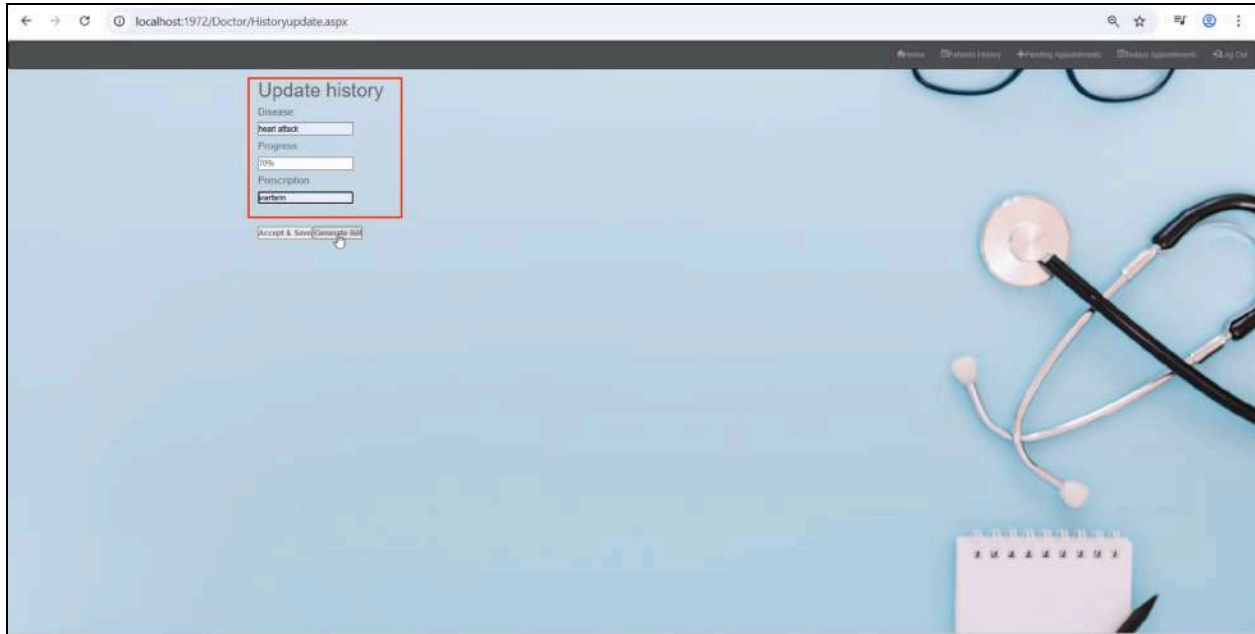


- **Doctors** will be able to see the patient's case history before beginning with the patient's treatment.



- **Doctors** will be able to update the case history of a patient that the doctor is treating by clicking on “**Today’s appointments**”. The doctor can update the status of the progress of the treatment for the patient.





## Database Schema for the Doctor's details:

SQLQuery2.sql - KATIA\SQLEXPRESS.healthsphere (KATIA\Chetan (72)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

healthsphere

SQLQuery2.sql - KATIA\SQLEXPRESS.healthsphere (KATIA\Chetan (72))

```
USE healthsphere
go
SELECT * FROM Doctor
```

100%

DoctorID	Name	Phone	Address	BirthDate	Gender	DeptNo	Charges_Fw_Visit	MonthlySalary	ReputationIndex	Patients_Treated	Qualification	Specialization	Work_Experience	Status
1	Peter	123456789	Wonderland, DC	1990-01-01	M	1	2500	30000	4	2	MD	Cardiologist	10	1
2	Pink Panther	123456789	Wonderland, DC	1990-01-01	M	2	2500	30000	4	0	MD	Orthopedic	10	1
3	Mickey Mouse	123456789	Wonderland, DC	1990-01-01	M	3	2500	30000	4	0	MD	General	10	1
4	Donald Duck	123456789	Wonderland, DC	1990-01-01	M	4	2500	30000	4	0	MD	Physio	10	1
5	Doramon	123456789	Wonderland, DC	1990-01-01	M	5	2500	30000	4	0	MD	Neurologist	10	0
6	Noddy	17942229999	inland creek	2003-01-01	M	3	700	75000	4	0	MBBS	ENT	3	1

Query executed successfully. KATIA\SQLEXPRESS (16.0 RTM) KATIA\Chetan (72) healthsphere 00:00:00 6 rows

Ready Lin 1 Col 1 RNS

The screenshots shown below depict a few of the code fragments of the main scenarios of the **Doctor's** facilities.

## DoctorMaster



github.com/flighten10/SSDI\_Project/blob/master/HealthSphereWebApp/Doctor/DoctorMaster.Master

SSDI\_Project / HealthSphereWebApp / Doctor / DoctorMaster.Master

Code Blame 141 lines (83 loc) · 4.19 KB Code 55% faster with GitHub Copilot

```
78
79
80 <body>
81     <form id="form1" runat="server">
82
83         <div>
84
85             <!-- Navigation Bar starts from here -->
86             <nav class="navbar navbar-inverse">
87                 <div class="container-fluid" style="background-color: #4c4f51">
88
89                     <ul class="nav navbar-nav navbar-right">
90                         <li><a href="/signup.aspx"><span class="glyphicon glyphicon-log-in"></span>Log Out</a></li>
91                     </ul>
92
93                     <ul class="nav navbar-nav navbar-right">
94                         <li><a href="patienthistory.aspx"><span class="glyphicon glyphicon-list-alt"></span>Today's Appointments</a></li>
95                     </ul>
96
97                     <ul class="nav navbar-nav navbar-right">
98                         <li><a href="Pendingappointment.aspx"><span class="glyphicon glyphicon-plus"></span>Pending Appointments</a></li>
99                     </ul>
100
101                     <ul class="nav navbar-nav navbar-right">
102                         <li><a href="PreviousHistory.aspx"><span class="glyphicon glyphicon-list-alt"></span>Patients History</a></li>
103                     </ul>
104
105                     <ul class="nav navbar-nav navbar-right">
106                         <li><a href="doctorhome.aspx"><span class="glyphicon glyphicon-home"></span>Home</a></li>
107                     </ul>
108
109                 </div>
110             </nav>
111             <!-- Navigation Bar ends here -->
112
113
```

## PatientHistory.aspx

github.com/flighten10/SSDI\_Project/blob/master/HealthSphereWebApp/Doctor/PatientHistory.aspx.cs

SSDI\_Project / HealthSphereWebApp / Doctor / PatientHistory.aspx.cs

Code Blame 52 lines (41 loc) · 1.43 KB Code 55% faster with GitHub Copilot

```
11 namespace HealthSphereWebApp
12
13 public partial class patienthistory : System.Web.UI.Page
14 {
15     protected void Page_Load(object sender, EventArgs e)
16     {
17         myDAL objmydal = new myDAL();
18         DataTable dt = new DataTable();
19         int found = 0;
20
21         int did = (int)Session["idoriginal"];
22
23         found = objmydal.search_patient_DAL(did, ref dt);
24         if (found != 1)
25         { Response.Write("<script>alert('There was some error');</script>"); }
26         else
27         {
28             patientsgrid.DataSource = dt;
29             patientsgrid.DataBind();
30         }
31     }
32
33     protected void patientsgrid_RowCommand(object sender, GridViewCommandEventArgs e)
34     {
35         if (e.CommandName == "Select")
36         {
37
38             Int16 num = Convert.ToInt16(e.CommandArgument);
39
40             string aId = patientsgrid.Rows[num].Cells[1].Text;
41
42             //retrieve appointmentid from that row (key-non editable)
43             int appointmentid = Convert.ToInt32(aId);
44
45             Session["appointid"] = appointmentid;
46             Response.Redirect("Historyupdate.aspx");
47         }
48     }
49 }
```

## PendingAppointment.aspx

github.com/lighten10/SSDI\_Project/blob/master/HealthSphereWebApp/Doctor/PendingAppointment.aspx.cs

SSDI\_Project / HealthSphereWebApp / Doctor / PendingAppointment.aspx.cs

Code Blame 86 lines (58 loc) · 2.42 KB Code 55% faster with GitHub Copilot

```
11 namespace HealthSphereWebApp
12 {
13     public partial class PendingAppointment : System.Web.UI.Page
14     {
15         public void loadgrid()
16         {
17         }
18
19         protected void update_appointment(Object sender, System.Web.UI.WebControls.GridViewCommandEventArgs e)
20         {
21             if (e.CommandName == "Select")
22             {
23                 // Retrieve the row that raised the event from the Rows
24
25                 Int16 num = Convert.ToInt16(e.CommandArgument);
26
27                 string aId = pendingappointments.Rows[num].Cells[1].Text;
28
29                 //retrieve appointmentid from that row (key-non editable)
30                 int appointmentid = Convert.ToInt32(aId);
31
32                 //=====updating the newly entered values in database=====
33                 myDAL objmyDAL = new myDAL();
34                 int result = objmyDAL.UpdateAppointment_DAL(appointmentid);
35                 //reload the page=====
36                 pendingappointments.EditIndex = -1;
37                 loadgrid();
38             }
39         }
40
41         protected void Delete_appointment(Object sender, GridViewDeleteEventArgs e)
42         {
43             // Retrieve the row that raised the event from the Rows
44             // collection of the GridView control.
45             GridViewRow row = pendingappointments.Rows[e.RowIndex];
46
47             //get appointmentid from that row
48             int appointmentid = Convert.ToInt32(row.Cells[1].Text.ToString());
```

## HistoryUpdate.aspx

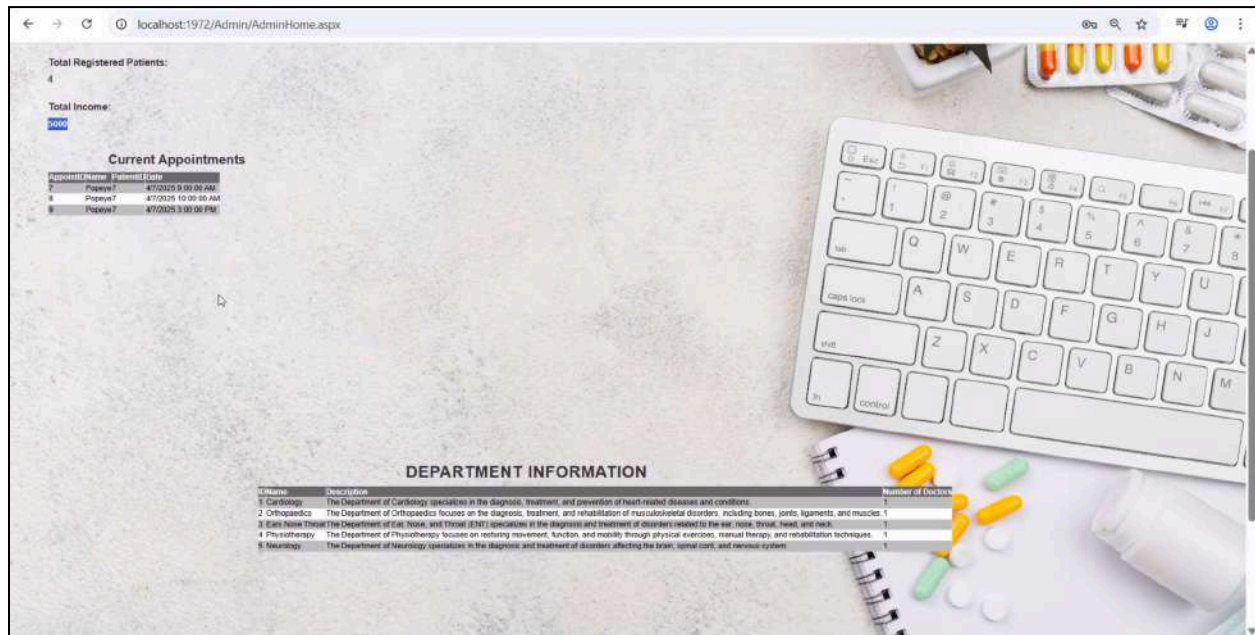
github.com/lighten10/SSDI\_Project/blob/master/HealthSphereWebApp/Doctor/HistoryUpdate.aspx.cs

SSDI\_Project / HealthSphereWebApp / Doctor / HistoryUpdate.aspx.cs

Code Blame 47 lines (38 loc) · 1.22 KB Code 55% faster with GitHub Copilot

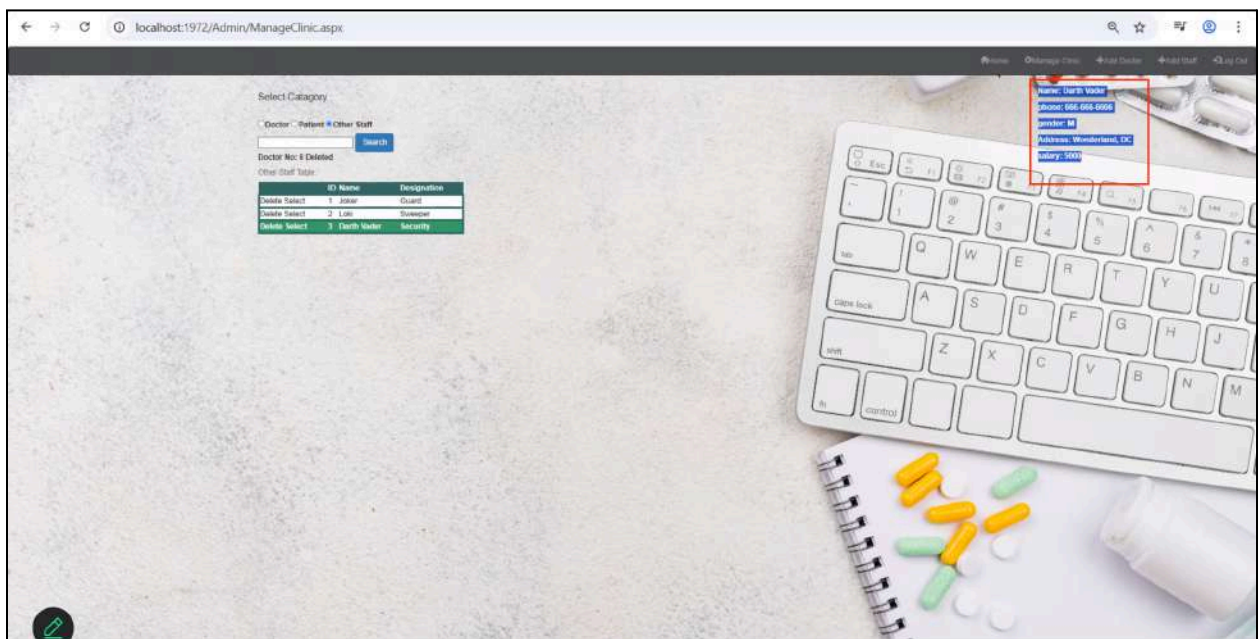
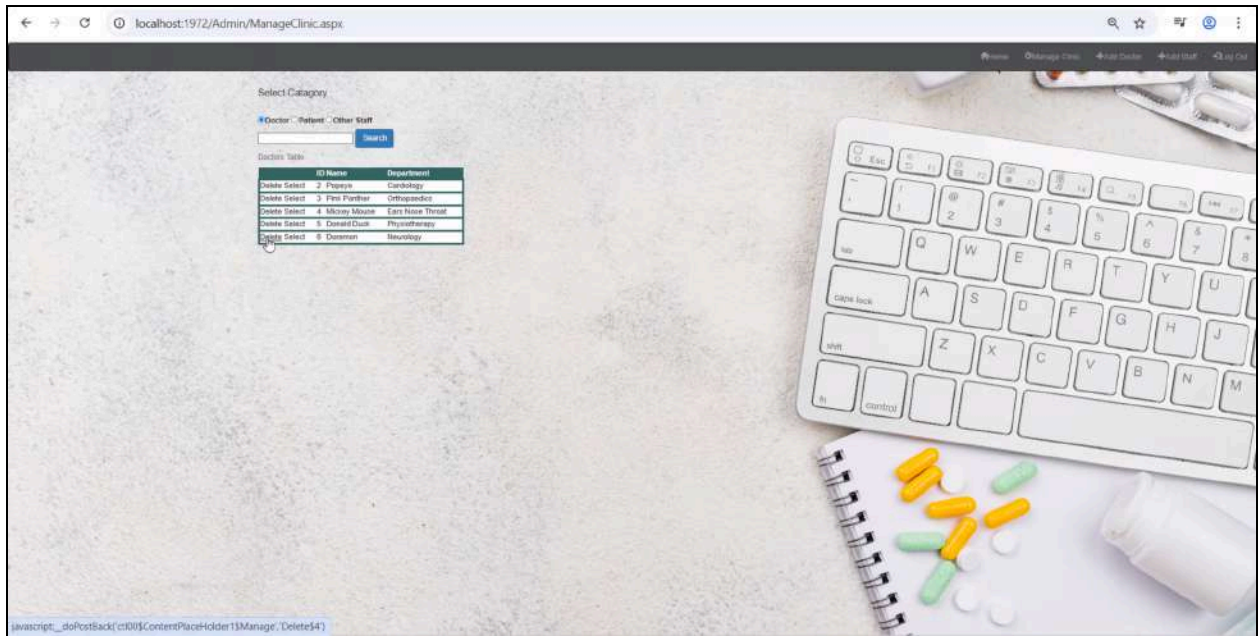
```
10 namespace HealthSphereWebApp
11 {
12     public partial class Historyupdate : System.Web.UI.Page
13     {
14         protected void Page_Load(object sender, EventArgs e)
15         {
16         }
17
18         public void saveindatabase(object sender, EventArgs e)
19         {
20             myDAL objmyDAL = new myDAL();
21             int found;
22             int did = (int)Session["idoriginal"];
23             string disease= Disease.Text;
24             string progres = progress.Text;
25             string prescrip = Prescription.Text;
26
27             int appid = (int)Session["appointmentid"];
28
29             found = objmyDAL.update_prescription_DAL(did,appid,disease,progres,prescrip);
30
31             if (found != 1)
32             { Response.Write("<script>alert('There was some error!');</script>"); }
33             else
34             { Response.Write("<script>alert('Information Successfully Updated');</script>"); }
35         }
36
37         public void generate_bill(object sender, EventArgs e)
38         {
39             Response.Redirect("bill.aspx");
40         }
41     }
42 }
```

- After logging in, the administrator personnel's landing page will be as per shown in the screenshot below. The administrator will be able to see the number of doctors in each department, total number of patients, total income generated and number of weekly appointments.

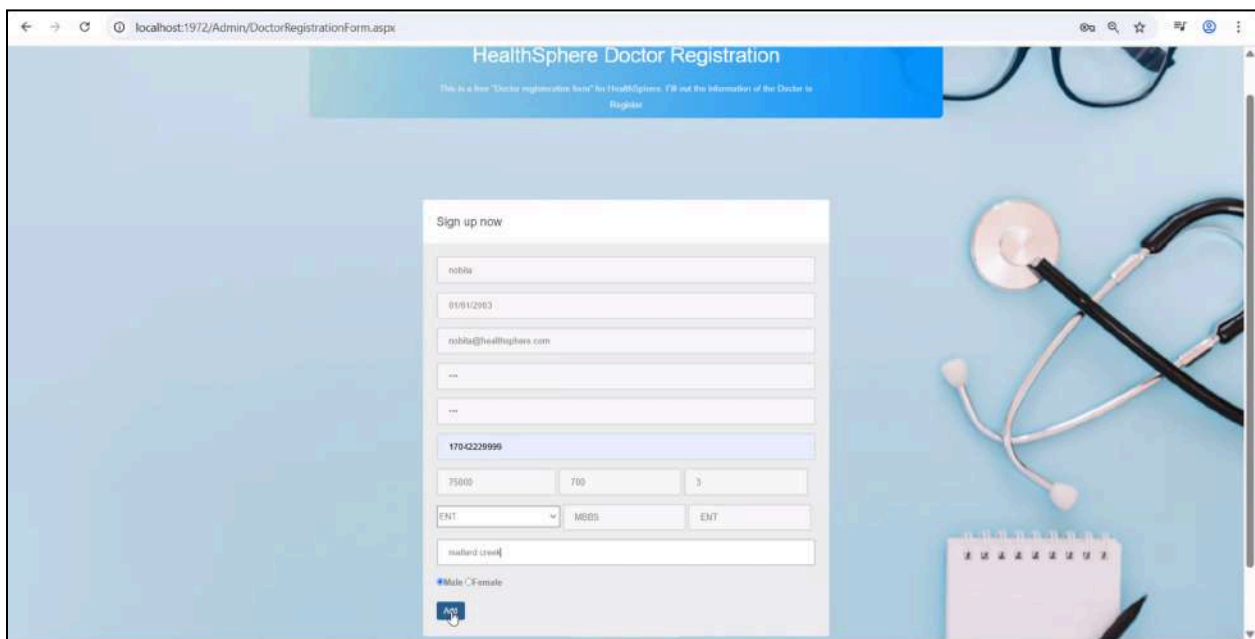
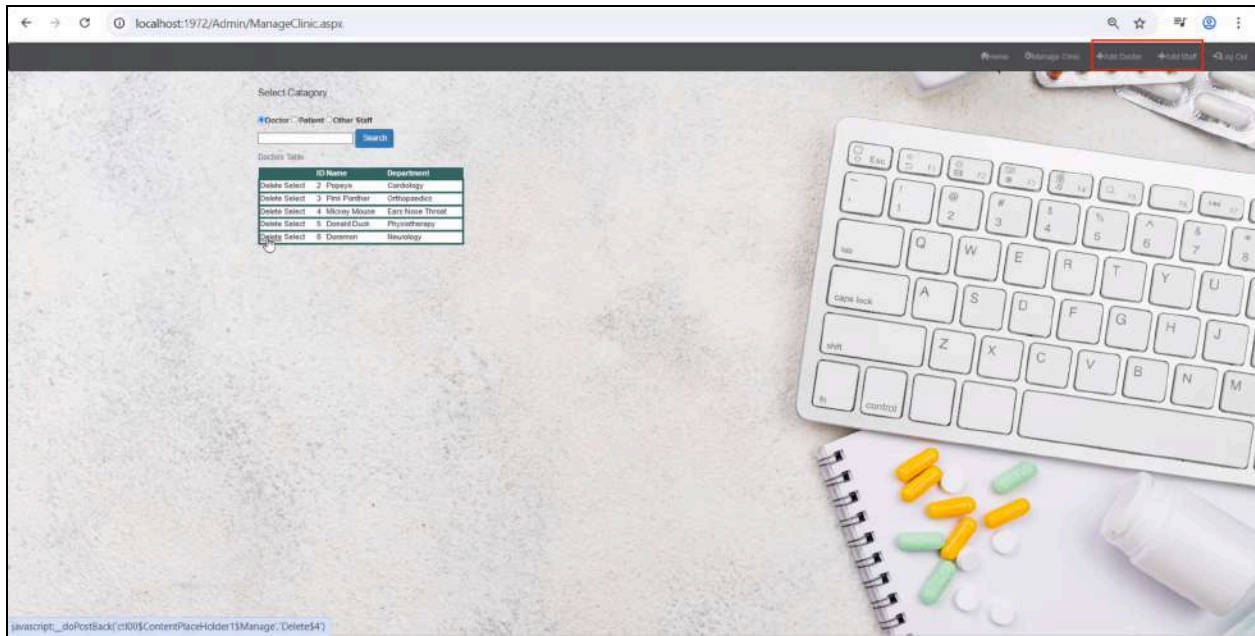


- The Administrator will be able to see the details of all doctors, patients and other staff by clicking on “Manage Clinic”. The administrator will be able to add/delete any doctor/patient/staff.





- The administrator will be able to add a new doctor/staff by clicking on the **Add doctor/staff** button.



The screenshots shown below depict a few of the code fragments of the main scenarios of the **Administrator's** facilities.

**AdminMaster:**

github.com/lylighten10/SSDI\_Project/blob/master/HealthSphereWebApp/Admin/Admin.Master

SSDI\_Project / HealthSphereWebApp / Admin / Admin.Master

Code Blame 145 lines (81 loc) · 3.97 KB Code 55% faster with GitHub Copilot

```
78
79
80 <title>Home</title>
81 <asp:ContentPlaceHolder ID="head" runat="server">
82 </asp:ContentPlaceHolder>
83 </head>
84
85
86 <body>
87
88
89 <div>
90
91
92 <!-- Navigation Bar starts from here -->
93 <nav class="navbar navbar-inverse">
94 <div class="container-fluid" style="background-color: #4c4f51;">
95
96 <ul class="nav navbar-nav navbar-right">
97 <li><a href="/SignUp.aspx"><span class="glyphicon glyphicon-log-in"></span>Log Out</a></li>
98 </ul>
99
100 <ul class="nav navbar-nav navbar-right">
101 <li><a href="AddStaff.aspx"><span class="glyphicon glyphicon-plus"></span>Add Staff</a></li>
102 </ul>
103
104 <ul class="nav navbar-nav navbar-right">
105 <li><a href="DoctorRegistrationForm.aspx"><span class="glyphicon glyphicon-plus"></span>Add Doctor</a></li>
106 </ul>
107
108
109 <ul class="nav navbar-nav navbar-right">
110 <li><a href="ManageClinic.aspx"><span class="glyphicon glyphicon-cog"></span>Manage Clinic</a></li>
111 </ul>
112
113 <ul class="nav navbar-nav navbar-right">
```

## AddStaff.aspx

github.com/lylighten10/SSDI\_Project/blob/master/HealthSphereWebApp/Admin/AddStaff.aspx.cs

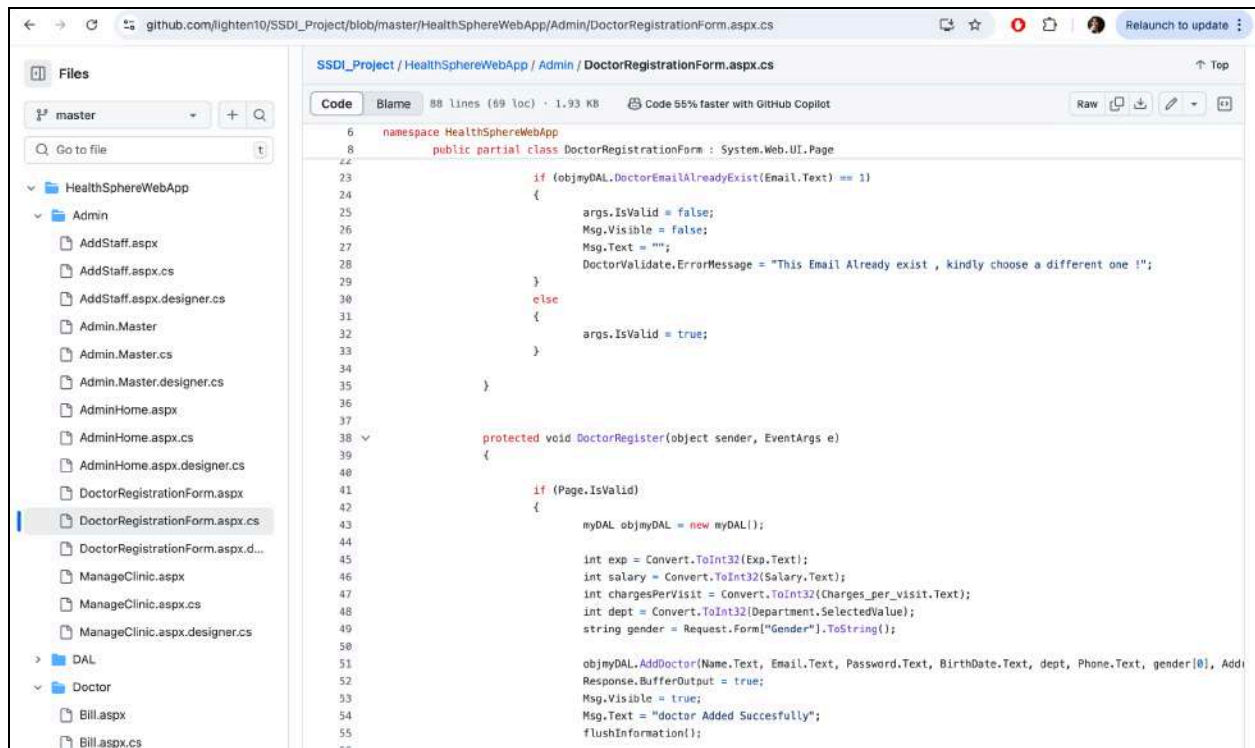
SSDI\_Project / HealthSphereWebApp / Admin / AddStaff.aspx.cs

Code Blame 50 lines (42 loc) · 1.03 KB Code 55% faster with GitHub Copilot

```
9 namespace HealthSphereWebApp
10 {
11     public partial class AddStaff : System.Web.UI.Page
12     {
13         protected void Page_Load(object sender, EventArgs e)
14         {
15         }
16     }
17
18     protected void StaffRegister(object sender, EventArgs e)
19     {
20         if (Page.IsValid)
21         {
22             myDAL objmyDAL = new myDAL();
23
24             int salary = Convert.ToInt32(Salary.Text);
25             string gender = Request.Form["Gender"].ToString();
26
27             if (objmyDAL.AddStaff(Name.Text, BirthDate.Text, Phone.Text, gender[0], Address.Text, salary, Qual.Text,
28             {
29                 Response.BufferOutput = true;
30                 Msg.Visible = true;
31                 Msg.Text = Designation.Text + " Added Successfully";
32                 flushInformation();
33             }
34         }
35     }
36
37     protected void flushInformation()
38     {
39         Name.Text = "";
40         BirthDate.Text = "";
41         Phone.Text = "";
42         Address.Text = "";
43         Salary.Text = "";
44         Qual.Text = "";
45         Designation.Text = "";
```

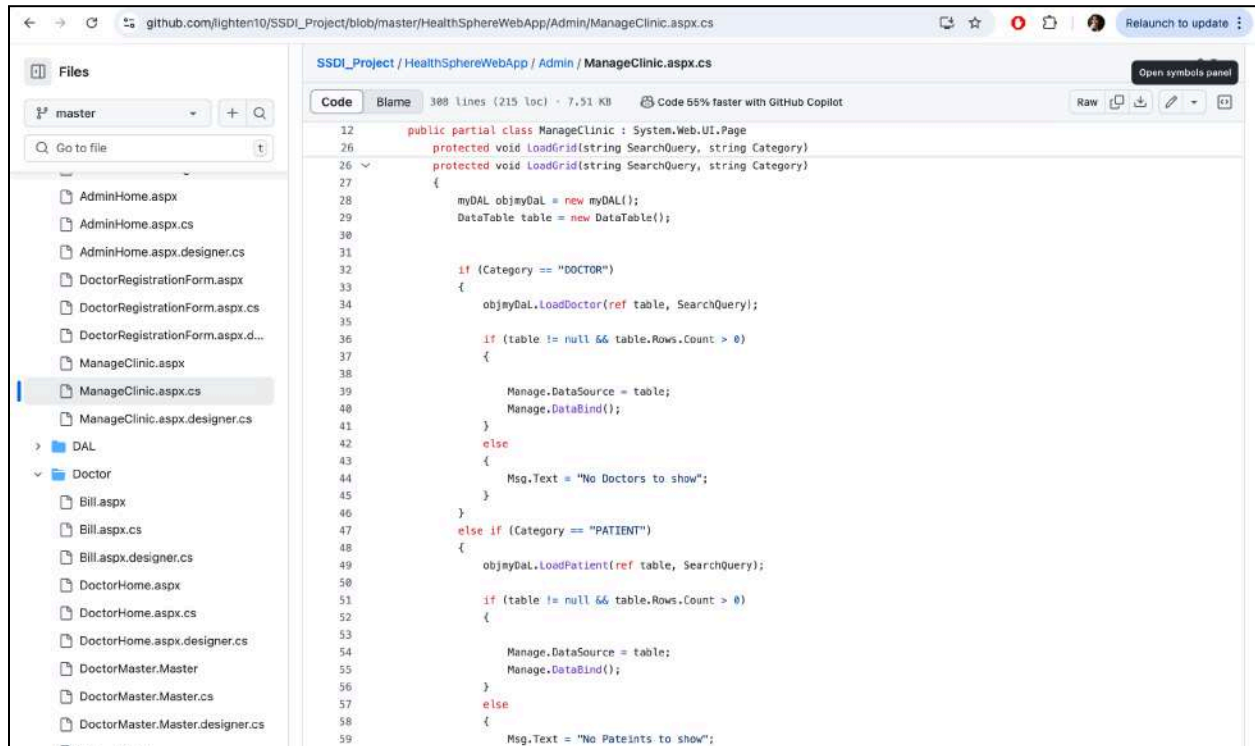
## DoctorRegistrationForm.aspx





```
6 namespace HealthSphereWebApp
7 {
8     public partial class DoctorRegistrationForm : System.Web.UI.Page
9     {
10         if (objmyDAL.DoctorEmailAlreadyExist(Email.Text) == 1)
11         {
12             args.IsValid = false;
13             Msg.Visible = false;
14             Msg.Text = "";
15             DoctorValidate.ErrorMessage = "This Email Already exist , kindly choose a different one !";
16         }
17         else
18         {
19             args.IsValid = true;
20         }
21     }
22
23     protected void DoctorRegister(object sender, EventArgs e)
24     {
25         if (Page.IsValid)
26         {
27             myDAL objmyDAL = new myDAL();
28
29             int exp = Convert.ToInt32(Exp.Text);
30             int salary = Convert.ToInt32(Salary.Text);
31             int chargesPerVisit = Convert.ToInt32(Charges_per_visit.Text);
32             int dept = Convert.ToInt32(Department.SelectedValue);
33             string gender = Request.Form["Gender"].ToString();
34
35             objmyDAL.AddDoctor(Name.Text, Email.Text, Password.Text, BirthDate.Text, dept, Phone.Text, gender[0], Add
36             Response.BufferOutput = true;
37             Msg.Visible = true;
38             Msg.Text = "doctor Added Successfully";
39             flushInformation();
40         }
41     }
42 }
```

## ManageClinic.aspx

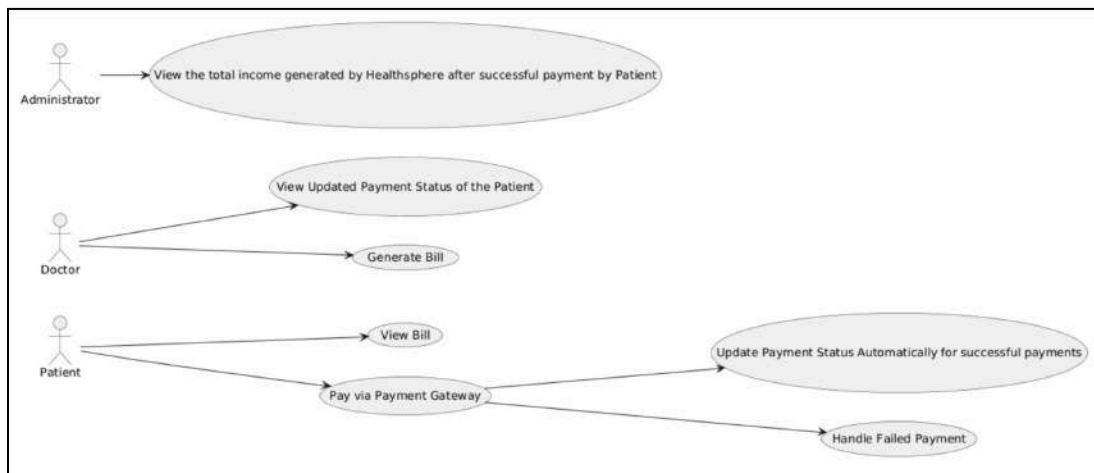


```
12 public partial class ManageClinic : System.Web.UI.Page
13 {
14     protected void LoadGrid(string SearchQuery, string Category)
15     {
16         protected void LoadGrid(string SearchQuery, string Category)
17         {
18             myDAL objmyDAL = new myDAL();
19             DataTable table = new DataTable();
20
21             if (Category == "DOCTOR")
22             {
23                 objmyDAL.LoadDoctor(ref table, SearchQuery);
24
25                 if (table != null && table.Rows.Count > 0)
26                 {
27                     Manage.DataSource = table;
28                     Manage.DataBind();
29                 }
30                 else
31                 {
32                     Msg.Text = "No Doctors to show";
33                 }
34             }
35             else if (Category == "PATIENT")
36             {
37                 objmyDAL.LoadPatient(ref table, SearchQuery);
38
39                 if (table != null && table.Rows.Count > 0)
40                 {
41                     Manage.DataSource = table;
42                     Manage.DataBind();
43                 }
44                 else
45                 {
46                     Msg.Text = "No Patients to show";
47                 }
48             }
49         }
50     }
51 }
```

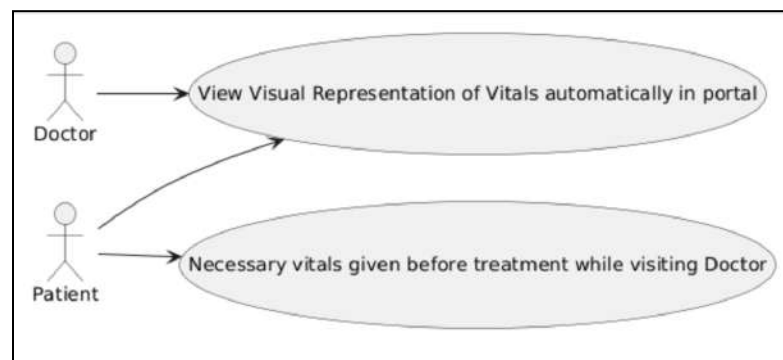
## Walkthrough of 2 additional scenarios (can be implemented in future scope)

- **Payment gateway** can be included as a part of the Health sphere in future scope, which would be accessible by the Administrator and Patients. Now, payment is done manually with cash/debit/credit cards, but there is no such provision on the website. With the help of a payment gateway, the process then starts with the treatment bill that pops up in the patient portal, the patient can pay the amount through online credit/debit card mode, and the payment status of the treatment updates to “paid” automatically in case of successful payments. Now, the doctor manually updates the status as paid or unpaid after the patient pays offline. Also, once the patient successfully pays the bill, the administrator can see the updated total income generated by Healthsphere.

Small use case diagram to demonstrate the scenario:



- Now, the Doctor has the provision to update the case history of the patient, which can include the necessary vitals of the patient as well. But, in future scope, **the latest vitals of the patient can be demonstrated in the form of a visual representation, say bar graph/pie chart** showing how well the patient has progressed from earlier or has deteriorated. The visual representation should be accessible in both the patient’s portal and the doctor’s portal treating the patient.



## **Conclusion**

In developing **Health Sphere**, our goal was to create a well-organized and efficient system that supports the smooth operation of healthcare services. Effective management tools are essential for improving workflow, enhancing communication, and ensuring that doctors, patients, and administrative staff can collaborate seamlessly. By incorporating features for handling medical records, scheduling appointments, and enabling easy access to data, the platform stays responsive and adaptable to changing requirements. This structured approach not only improves productivity and clarity but also boosts overall satisfaction for both users and the healthcare facility.

---