A PROJECT ON

"ONLINE HEALTH SERVICES"

SUBMITTED IN

PARTIAL FULFILLMENT OF THE REQUIREMENT

FOR THE COURSE OF DIPLOMA IN ADVANCED COMPUTING FROM CDAC

SUNBEAM INSTITUTE OF INFORMATION TECHNOLOGY

Hinjewadi

SUBMITTED BY:

SHRINIVAS DHOLE,

PARTH JIVARAJ KHEDEKAR,

SUDHIR SANJAY ZUGE,

GAJENDRA MAHENDRA BAGI

UNDER THE GUIDENCE OF:

Mr. Snehal Jadhav Faculty Member

Sunbeam Institute of Information Technology, Pune

ACKNOWLEDGEMENT

A project usually falls short of its expectation unless aided and guided by the right persons at the right time. We avail this opportunity to express our deep sense of gratitude towards Mr. Nitin Kudale (Center Coordinator, SIIT, Pune) and Mr. Yogesh Kolhe (Course Coordinator, SIIT ,Pune) .

We are deeply indebted and grateful to them for their guidance, encouragement and deep concern for our project. Without their critical evaluation and suggestions at every stage of the project, this project could never have reached its present form.

Last but not the least we thank the entire faculty and the staff members of Sunbeam Institute of Information Technology, Pune for their support.

Shrinivas D

Parth K,

Sudhir Z,

Gajendra B

0824 PG-DAC

SIIT Pune

CERTIFICATE

This is to certify that the project work under the title 'Online Health Services' is done by Shrinivas Dhole, Parth Khedekar, Sudhir Zuge, Gajendra Bagi in partial fulfillment of the requirement for award of Diploma in Advanced Computing Course.

Project Guide **Mr. Yogesh Kolhe**

Date: 12-02-2025 **Course Co-Coordinator**

INDEX

INTRODUCTION TO PROJECT

In today's fast-paced world, accessing healthcare should be seamless and convenient. However, many patients struggle with long waiting times, difficulty in booking appointments, and inefficient hospital management systems. Our platform addresses these challenges by offering a digital solution that simplifies the entire process—from booking appointments from home to managing hospitals and doctors efficiently.

Our online health services platform provides an easy-to-use interface where patients can book appointments with doctors without visiting the hospital in person. It also empowers administrators to manage healthcare facilities by adding or removing doctors and hospitals based on availability and demand. This ensures a smooth, organized, and effective healthcare system.

This platform serves as a one-stop solution for healthcare services, offering:

- Patients: A hassle-free way to book appointments from home.
- Doctors: An organized system to manage their schedules.
- Admins: A tool to efficiently add, remove, or manage doctors, departments and hospitals.

By integrating technology into healthcare, we aim to enhance accessibility, reduce waiting times, and improve the overall patient experience.

PROJECT OVERVIEW AND SUMMARY

2.1) **Purpose:**
The primary purpose of this online health services platform is to streamline and modernize the healthcare appointment booking and management system. It aims to achieve the following objectives:

For Patients:

- **Easy Appointment Booking** – Enables patients to book doctor appointments remotely from their homes, eliminating the need for physical visits just for scheduling.
- **Reduced Waiting Time** – Helps avoid long queues and waiting times at hospitals and clinics.
- **Seamless Access to Healthcare** – Provides a user-friendly interface where patients can search for doctors and hospitals based on availability and specialization.

For Hospitals & Administrators:

- **Efficient Doctor & Hospital Management** – Allows administrators to add, update, or remove hospitals, departments and doctors as per demand and availability.
- **Optimized Resource Utilization** – Ensures better scheduling and management of healthcare professionals to reduce overbooking or underutilization.
- **Centralized System** – Offers a structured and organized database for better tracking of appointments, doctor availability, and hospital operations.

For Overall Healthcare System:

- **Enhanced Operational Efficiency** – Reduces manual administrative work, ensuring a smooth and error-free process.
- **Scalability & Future Growth** – Lays the foundation for future integration of advanced features like telemedicine, electronic health records.
- **Improved Patient Satisfaction** – Ensures a more convenient and hassle-free healthcare experience for both patients and providers.

2.2) Scope:

The Online Health Services Platform aims to streamline the process of booking doctor appointments and managing healthcare facilities efficiently. It provides benefits to patients, doctors, and administrators through an intuitive and automated system.

Features & Functionalities:

- **Online Appointment Booking** – Patients can schedule appointments with doctors remotely.
- **Doctor & Hospital Management** – Administrators can **add, update, or remove** hospitals and doctors as needed.
- **Search & Filter Options** – Users can find doctors and hospitals based on specialization, location, and availability.
- **User-Friendly Interface** – Ensures accessibility for both tech-savvy and non-tech-savvy users.

User Roles & Capabilities

- **Patients** – Can register, log in, search for doctors, book appointments, and receive reminders.
- **Doctors** – Can manage their schedules and update availability.
- **Administrators** – Can oversee the platform, manage doctors and hospitals, and handle appointment-related operations.

System Constraints & Scalability

- **Secure Data Handling** – Protects sensitive patient and hospital data with encryption and secure authentication.
- **Scalability for Future Expansion** – Can be extended to include additional features such as telemedicine, electronic health records (EHR), and AI-driven recommendations.

Expected Benefits

- **Reduces Hospital Overcrowding** – By allowing remote appointment booking, minimizing in-person visits. Enhances Healthcare Accessibility –
- Patients from remote areas can easily book consultations.
- **Optimizes Resource Utilization** – Ensures hospitals and doctors operate efficiently without overbooking or underutilization.
- **Improves Patient Satisfaction** – Provides a seamless and convenient way to access medical services.

2.3) User Classes and Characteristics:

The platform is designed to cater to different types of users, each with specific needs:

1. **Patients**
   - Can register and log in to book appointments.
   - Search for hospitals and doctors based on availability and specialization.

2. **Doctors**
   - Can view and manage their appointment schedules.
   - Update their availability for patient consultations.

3. **Administrators (Hospitals/Clinic Managers)**
   - Have the authority to add, update, or remove doctors and hospitals.
   - Oversee appointment scheduling and availability management.
   - Ensure system functionality and accuracy.

2.4) Design and Implementation Constraints:

1. **Data Privacy and Security**
   - Patient information must be kept secure and confidential.
   - Strong passwords and encryption will protect user data.

2. **Internet Requirement**
   - The platform needs an internet connection to work.

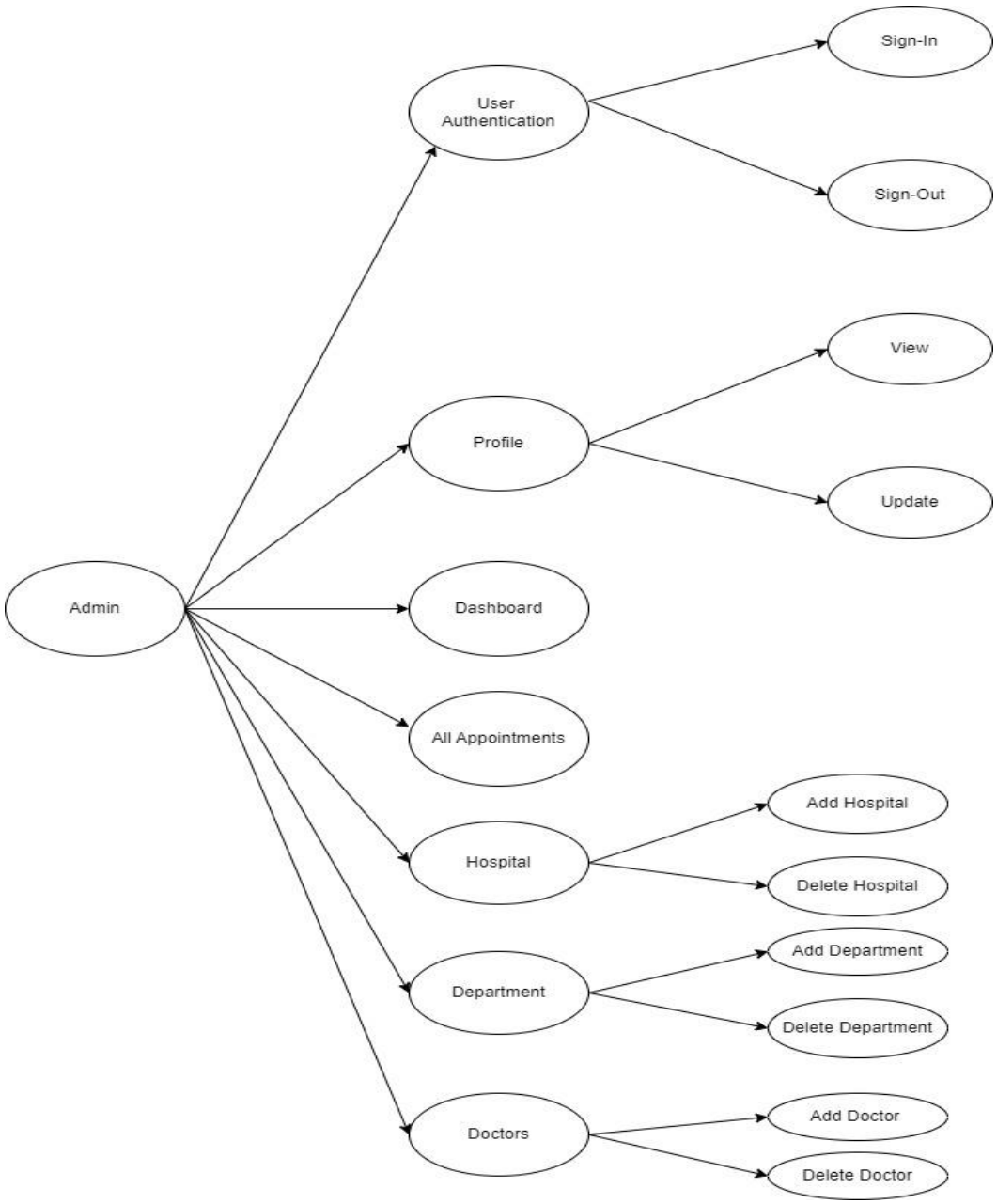
3. **Device Compatibility**
    - The system should work on computers, tablets, and mobile phones.
4. **Scalability**
    - It must handle a growing number of users and appointments without slowing down.
5. **Regulatory Compliance**
    - The platform must follow healthcare rules and laws for data protection and patient management.
6. **User Accessibility**
    - It should be easy to use for all types of users, including those with limited technical knowledge.

7. **System Maintenance**
    - Regular updates and maintenance will be required to keep the platform running smoothly.

REQUIREMENTS

3.1) Functional Requirements

3.1.1) Use Case for Admin-



1. **User Authentication**
   - **Sign-In** – Allows the admin to log in to the system securely.
   - **Sign-Out** – Ends the admin session to maintain security.

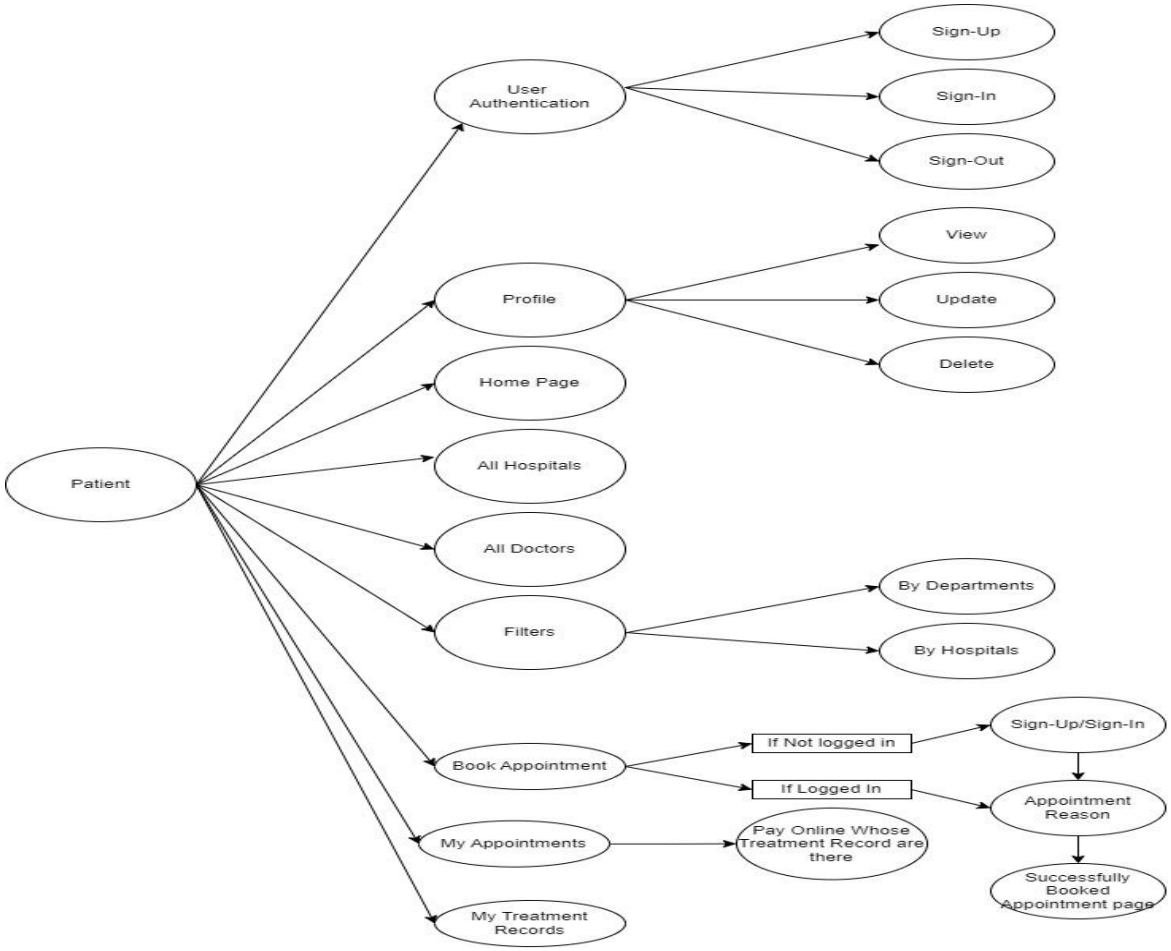
2. **Profile Management**
   - **View Profile** – Displays the admin’s personal information and

system role.

- **Update Profile** – Enables updating personal details, such as name or email.

3. **Dashboard Access**
   - **View Dashboard** – Provides a high-level overview of system statistics and activity.
4. **Appointment Management**
   - **View All Appointments** – Allows the admin to monitor and manage all booked appointments.
5. **Hospital Management**
   - **Add Hospital** – Enables adding new hospital details to the system.
   - **Delete Hospital** – Removes a hospital entry when it’s no longer needed.
6. **Department Management**
   - **Add Department** – Adds a new department to an existing hospital.
   - **Delete Department** – Deletes a department that is no longer active.
7. **Doctor Management**
   - **Add Doctor** – Registers a new doctor under a specific hospital and department.
   - **Delete Doctor** – Removes a doctor’s record when they are no longer part of the system.

3.1.2) Use Case for Patient-



1. **User Authentication**

- **Sign-Up** – Allows a new patient to create an account.
- **Sign-In** – Logs the patient into the system to access services.
- **Sign-Out** – Ends the current session to secure the account.

2. **Profile Management**
   - **View Profile** – Displays patient details and information.
   - **Update Profile** – Allows the patient to update their personal information.
   - **Delete Profile** – Enables the patient to delete their account permanently.
3. **Home Page Access**
   - Serves as the starting point for navigating different features.
4. **View All Hospitals**
   - Lists all hospitals registered in the system for selection.
5. **View All Doctors**
   - Displays a list of doctors that patients can filter by department or hospital.
6. **Filters**
   - **By Departments** – Narrows the search for doctors based on medical specialties.
   - **By Hospitals** – Helps patients find doctors associated with a specific hospital.
7. **Book Appointment**
   - **If Not Logged In** – Redirects the patient to the sign-up or sign-in page.
   - **If Logged In** – The patient provides an appointment reason and completes booking.
8. **Payment for Appointment**
   - Allows online payment for appointments linked to the patient’s treatment records.
9. **View My Appointments**
   - Displays a list of all booked appointments.
10. My **Treatment Records**
    - Provides access to the patient’s past treatment history.

3.1.2) Use Case for Doctors-

Use Cases for Doctor

1. **User Authentication**
   - Doctor can **sign up**, **sign in**, and **sign out** of the system.
   - Requires valid credentials for authentication.
2. **Sign-Up**
   - Doctor registers by providing necessary details.
   - Account is created successfully if all details are valid.
3. **Sign-In**
   - Doctor logs into the system using their credentials.
   - Upon success, access is granted to system functionalities.
4. **Sign-Out**
   - Doctor logs out of the system.
   - The session is terminated, requiring re-login for further access.
5. **Profile Management**
   - Doctor can manage their personal and professional details.
   - Includes options to **view** and **update** profile.
6. **View Profile**
   - Displays the doctor’s stored details.
7. **Update Profile**
   - Allows updating of information like contact details, specialization, etc.

8. **Home Page**
    - Serves as a dashboard for the doctor.
    - Provides access to appointments, patient records, and other functionalities.
9. **View All Appointments**
    - Doctor can see a list of scheduled appointments.
    - Helps in managing patient consultations effectively.
10. **Add Treatment Record**

- Doctor can add diagnosis and treatment details for a patient.
- Ensures medical records are maintained for future reference.

3.2 Non - Functional Requirements

3.2.1) Usability Requirement

1. **User Interface Simplicity**
    - The system has an intuitive and user-friendly interface for all users (admin, doctors, and patients).
    - Navigation is consistent across all modules, with a clean and minimalistic design.
2. **Accessibility**
    - It is compatible with common browsers (Chrome, Firefox, Edge, Safari).
    - Large, readable fonts and high-contrast colours are used for visually impaired users.
3. **Responsiveness**
    - The system responds promptly to user actions, ensuring a smooth user experience.
    - Pages load quickly, even during high-traffic periods.
4. **Error Handling and Feedback**
    - Clear and descriptive error messages are provided with suggestions for corrective actions.
    - Success and confirmation messages are displayed for critical actions (e.g., booking an appointment, adding a hospital).
5. **Learnability**
    - New users (patients and doctors) can understand and use the system effectively with minimal guidance.
    - **Contextual tooltips** and an **FAQ section** are available for self-help.
6. **Consistency**
    - Terminology, icons, and action buttons are consistent across all pages.
    - Standard UI conventions are followed to reduce the learning curve.
7. **Security Feedback**
    - Visual cues (like padlock icons) are displayed during secure actions (e.g., login, appointment booking).
    - Users are notified of successful login, logout, or session timeout.

3.2.2) Performance Requirements

1. **System Availability**
    - The system is expected to have a high availability rate, ensuring it is accessible and operational **99.9%** of the time.
    - Downtime for maintenance is minimized and scheduled during off-peak hours to reduce impact on users.
2. **Scalability**
    - The platform is scalable to accommodate an increasing number of users (patients, doctors, and administrators) without compromising performance.
    - The system can handle growing data loads (hospital details, doctor records, and patient appointments) without significant degradation in performance.
3. **Response Time**
    - The system responds promptly to user actions, with typical operations such as booking appointments, viewing doctor profiles, or loading hospital details being processed quickly.
    - Critical actions (e.g., submitting a booking request or adding a new doctor) are executed without noticeable delays.
4. **Load Handling**
    - The system can handle a high volume of concurrent users, particularly during peak times, without crashing or experiencing significant slowdowns.
    - It supports simultaneous access by multiple admins, doctors, and patients without affecting system performance.
5. **Data Processing Speed**
    - Data retrieval (e.g., doctor schedules, patient details, or hospital listings) is efficient, ensuring smooth and uninterrupted operations.
    - Operations like searching for doctors, hospitals, or scheduling appointments are completed quickly.
6. **System Backup and Recovery**
    - Regular backups are performed to prevent data loss, and recovery procedures are efficient, ensuring minimal service disruption in case of system failure.
    - The system can recover from failures within an acceptable time frame, ensuring users can resume operations quickly.
7. **Optimization for Mobile Devices**
    - The system is optimized for mobile use, ensuring that performance remains smooth on smartphones and tablets without lag, even when accessing data-rich pages.

3.2.3) Reliability Requirements

1. **System Availability**
    - The system is designed for high reliability, ensuring continuous availability with minimal downtime.
    - Scheduled maintenance is planned during off-peak hours to avoid user disruption, and emergency fixes are handled promptly.
2. **Error Rate**
    - The system maintains a low error rate for all core functions, including appointment booking, doctor management, and hospital information updates.
    - Any system errors or bugs are logged, categorized, and addressed within an acceptable timeframe.
3. **Data Integrity**
    - The system ensures the integrity of data by implementing error-checking mechanisms and validating inputs for all critical data (e.g., patient information, appointment records, doctor details).
    - Data inconsistencies are detected and corrected to avoid

potential system failures or incorrect processing.

4. **Backup and Recovery**
    - Regular system backups are performed, ensuring that data is recoverable in case of system failure.
    - The system includes a reliable recovery process to restore services and data within a reasonable period.
5. **Failover Mechanisms**
    - The system incorporates failover mechanisms to ensure reliability in case of component failure, with automatic switching to backup systems to prevent disruption.
    - Critical services (e.g., patient appointment management) continue to function seamlessly, even in the event of partial system failures.
6. **Load Balancing**
    - The platform uses load balancing techniques to distribute user requests efficiently across servers, ensuring consistent performance under heavy usage.
    - Load balancing ensures that the system remains stable, even during peak traffic periods.
7. **Monitoring and Reporting**
    - Continuous system monitoring is in place to detect performance issues, failures, or abnormal behaviour early.
    - Alerts and notifications are generated to inform administrators of potential issues, allowing for proactive resolution.

3.2.4) Portability Requirements

1. **Cross-Platform Compatibility**
    - The system is designed to be compatible across various platforms, including Windows, macOS, Linux, and mobile operating systems (iOS and Android).
    - Users can access the platform via any major web browser (Chrome, Firefox, Safari, Edge) without requiring additional installations.
2. **Ease of Deployment**
    - The system is easy to deploy across different environments (development, testing, production), with clear installation and configuration instructions.
    - The deployment process is automated and can be replicated on new servers with minimal manual intervention.
3. **Cloud and On-Premises Compatibility**
    - The platform can be deployed on cloud environments (e.g., AWS, Azure) or on-premises servers, depending on user needs.
    - It supports integration with various cloud-based services for scalability and backup.
4. **Internationalization and Localization Support**
    - The system supports easy localization for different regions and languages, enabling it to be adapted for use in multiple countries.
    - It allows easy customization of language, time formats, and currency settings based on geographical location.
5. **Modular Design for Portability**
    - The system’s architecture is modular, allowing for easy portability of individual components (e.g., user management, appointment scheduling) to different platforms or environments.
    - Updates and patches can be rolled out selectively to specific modules without affecting the entire system.

DESIGN

4.1 Database Design

The following table structures depict the database design.

Table 1: Patients

Column Name	Data Type	Allow Nulls
PatientID	int	☐
FirstName	nvarchar(100)	☐
LastName	nvarchar(100)	☐
DateOfBirth	datetime2(7)	☐
Phone	nvarchar(15)	☑
Email	nvarchar(100)	☑
Gender	nvarchar(10)	☐
Address	nvarchar(255)	☑
CreatedAt	datetime2(7)	☐
UpdatedAt	datetime2(7)	☐
Image	varbinary(MAX)	☑
UserId	nvarchar(MAX)	☐

Table 2: Department Table

Column Name	Data Type	Allow Nulls
DepartmentID	int	☐
HospitalID	int	☐
Name	nvarchar(100)	☐
Description	nvarchar(255)	☑
CreatedAt	datetime2(7)	☐
UpdatedAt	datetime2(7)	☐
Image	varbinary(MAX)	☑

Table 3: Hospitals

Column Name	Data Type	Allow Nulls
HospitalID	int	☐
Name	nvarchar(255)	☐
Address	nvarchar(255)	☐
Phone	nvarchar(15)	☑
Email	nvarchar(100)	☑
CreatedAt	datetime2(7)	☐
UpdatedAt	datetime2(7)	☐
Image	varbinary(MAX)	☑

Table 4:Doctors

Column Name	Data Type	Allow Nulls
DoctorID	int	☐
HospitalID	int	☐
DepartmentID	int	☐
Name	nvarchar(100)	☐
Specialization	nvarchar(100)	☑
Phone	nvarchar(15)	☑
Email	nvarchar(100)	☑
CreatedAt	datetime2(7)	☐
UpdatedAt	datetime2(7)	☐
Image	varbinary(MAX)	☑
UserId	nvarchar(MAX)	☐
Degree	nvarchar(200)	☑
Experience	nvarchar(50)	☑
Fees	decimal(18, 2)	☑
IsAvailable	bit	☐

Table 5: Treatment Record

Column Name	Data Type	Allow Nulls
TreatmentRecordID	int	☐
PatientID	int	☐
DoctorID	int	☐
TreatmentDate	datetime2(7)	☐
Description	nvarchar(500)	☐
CreatedAt	datetime2(7)	☐
UpdatedAt	datetime2(7)	☐
IsVisibleToPatient	bit	☐
AppointmentID	int	☐

Table 6: AspNetUserToken

Column Name	Data Type	Allow Nulls
UserId	nvarchar(450)	☐
LoginProvider	nvarchar(450)	☐
Name	nvarchar(450)	☐
Value	nvarchar(MAX)	☑

Table 7: AspNetRoleClaims

Column Name	Data Type	Allow Nulls
Id	int	☐
RoleId	nvarchar(450)	☐
ClaimType	nvarchar(MAX)	☑
ClaimValue	nvarchar(MAX)	☑

Table 8: AspNetRoles

Column Name	Data Type	Allow Nulls
Id	nvarchar(450)	☐
Name	nvarchar(256)	☑
NormalizedName	nvarchar(256)	☑
ConcurrencyStamp	nvarchar(MAX)	☑

Table 9: AspNetUserClaims

Column Name	Data Type	Allow Nulls
Id	int	☐
UserId	nvarchar(450)	☐
ClaimType	nvarchar(MAX)	☑
ClaimValue	nvarchar(MAX)	☑

Table 10: AspNetUserLogins

Column Name	Data Type	Allow Nulls
LoginProvider	nvarchar(450)	☐
ProviderKey	nvarchar(450)	☐
ProviderDisplayName	nvarchar(MAX)	☑
UserId	nvarchar(450)	☐

Table 11: AspNetRoles

Column Name	Data Type	Allow Nulls
UserId	nvarchar(450)	☐
RoleId	nvarchar(450)	☐

Table 12: AspNetUsers

Column Name	Data Type	Allow Nulls
Id	nvarchar(450)	☐
UserName	nvarchar(256)	☑
NormalizedUserName	nvarchar(256)	☑
Email	nvarchar(256)	☑
NormalizedEmail	nvarchar(256)	☑
EmailConfirmed	bit	☐
PasswordHash	nvarchar(MAX)	☑
SecurityStamp	nvarchar(MAX)	☑
ConcurrencyStamp	nvarchar(MAX)	☑
PhoneNumber	nvarchar(MAX)	☑
PhoneNumberConfirmed	bit	☐
TwoFactorEnabled	bit	☐
LockoutEnd	datetimeoffset(7)	☑
LockoutEnabled	bit	☐
AccessFailedCount	int	☐

Table 13: Treatment Record

Column Name	Data Type	Allow Nulls
TreatmentRecordID	int	☐
PatientID	int	☐
DoctorID	int	☐
TreatmentDate	datetime2(7)	☐
Description	nvarchar(500)	☐
CreatedAt	datetime2(7)	☐
UpdatedAt	datetime2(7)	☐
IsVisibleToPatient	bit	☐
AppointmentID	int	☐

Table 14: Admin Table

Column Name	Data Type	Allow Nulls
AdminID	int	☐
Name	nvarchar(100)	☐
Email	nvarchar(256)	☐
CreatedAt	datetime2(7)	☐
UpdatedAt	datetime2(7)	☐
UserId	nvarchar(450)	☑

Coding Standards Implemented

Below summarizes the naming recommendations for identifiers in your .NET project. Pascal casing is used mainly (i.e., capitalize the first letter of each word), with camel casing (capitalize each word except for the first one) being used in certain circumstances.

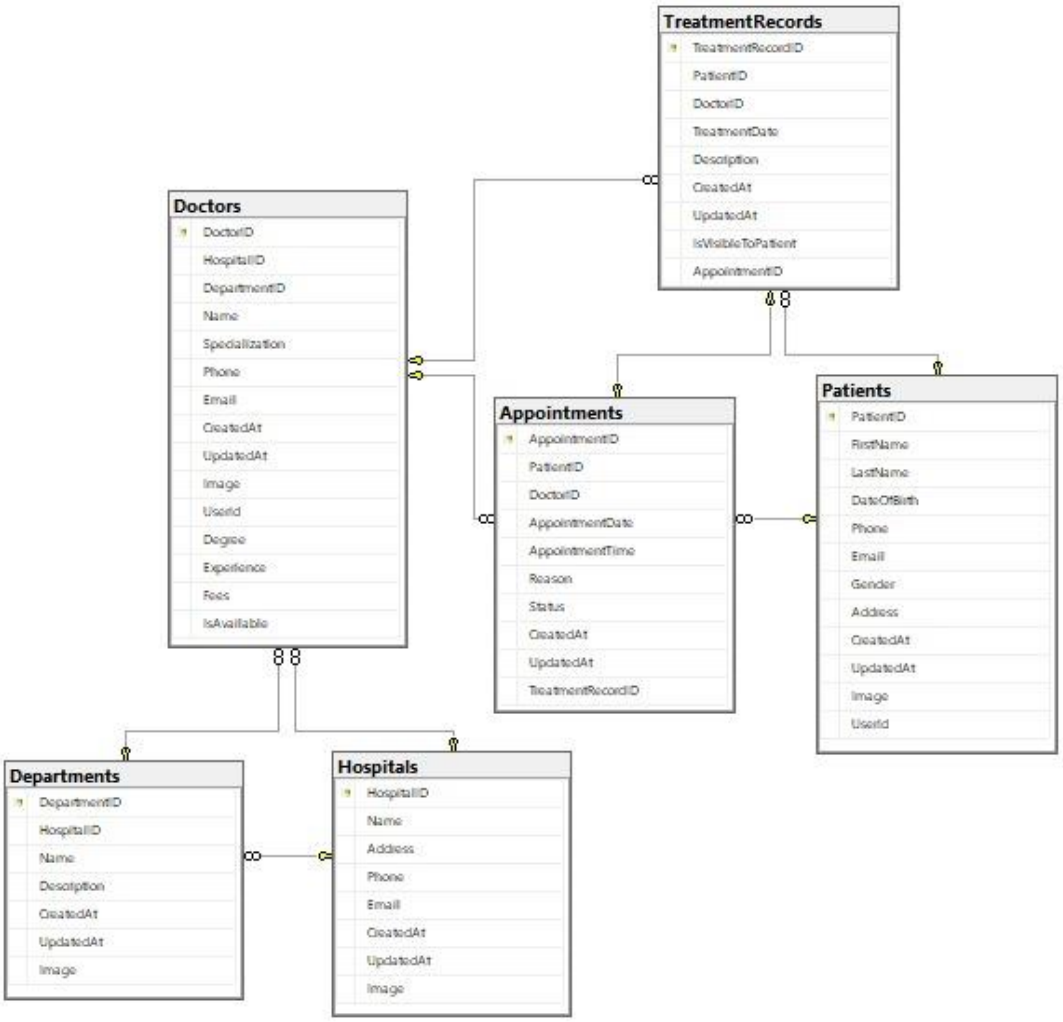
Identifier	Case	Examples	Additional Notes
Class	Pascal	`Admin`, `IdentityUser`, `MultiHospitalContext`	Class names should be based on "objects" or "real things" and should generally be **nouns**. No `_` signs allowed.
Method	Camel	`GetAdminById()`, `UpdateAdminDetails()`	Method names should use **verbs** or **verb phrases**.
Parameter	Camel	`adminId`, `userId`, `email`, `name`	Use **descriptive** parameter names. Names should be clear enough to determine meaning without additional context.
Interface	Pascal with "I" prefix	`IAdminService`, `IUserRepository`	Prefix interfaces with **"I"**. Do not use `_` signs.
Annotation	Pascal	`[Required]`, `[EmailAddress]`, `[Key]`	Use **@** at the start of an annotation when needed.
DTOs	Pascal with DTO suffix	`AdminDTO`, `UserDTO`	Used to transfer data between processes.
Exception Class	Pascal with "Exception" suffix	`AdminNotFoundException`, `InvalidEmailException`	Exception class names should **end with "Exception"** to indicate an error type.

Comments

- Comment each type, each non-public type member, and each region declaration.
- Use end-line comments only on variable declaration lines.
- Separate comments from comment delimiters (// or /* */) with one space.
- Begin the comment text with an uppercase letter.
- End the comment with a period.
- Explain the code; do not repeat it.

SR-NO	TEST CASE	EXPECTED RESULT	ACTUAL RESULT	ERROR MESSAGE
1	User Registration	User registered successfully	OK	Nothing
2	User Login	Pop-up message for incorrect credentials	OK	Please enter valid username and password.
3	View Doctor List	List of available doctors displayed	OK	Failed to load doctor list
4	Book Appointment	Appointment booked successfully	OK	Nothing
5	Cancel Appointment	Appointment cancelled successfully	OK	Appointment not found
6	View Appointment History	Displays past and upcoming appointments	OK	Failed to fetch appointment history
7	Search Doctor by Specialization	List of specialized doctors displayed	OK	No doctors found
8	Search Doctor by Department	List of doctors by department displayed	OK	No doctors found
9	Admin Add Doctor	Doctor added successfully	OK	Nothing
10	Admin Remove Doctor	Doctor removed successfully	OK	No doctor found
11	Update Doctor Schedule	Schedule updated successfully	OK	Failed to update schedule
12	View Hospital List	List of hospitals displayed	OK	Failed to fetch hospital list
13	Payment Processing	Payment completed successfully	OK	Payment failed
14	Generate Appointment Report	Report generated successfully	OK	Failed to generate report
15	Logout	User logged out successfully	OK	Nothing

ENTITY RELATIONSHIP DIAGRAM


TreatmentRecords
TreatmentRecordID
PatientID
DoctorID
TreatmentDate
Description
CreatedAt
UpdatedAt
IsVisibleToPatient
AppointmentID
Doctors
DoctorID
HospitalID
DepartmentID
Name
Specialization
Phone
Email
CreatedAt
UpdatedAt
Image
UserId
Degree
Experience
Fees
IsAvailable
Appointments
AppointmentID
PatientID
DoctorID
AppointmentDate
AppointmentTime
Reason
Status
CreatedAt
UpdatedAt
TreatmentRecordID
Patients
PatientID
FirstName
LastName
DateOfBirth
Phone
Email
Gender
Address
CreatedAt
UpdatedAt
Image
UserId
Departments
DepartmentID
HospitalID
Name
Description
CreatedAt
UpdatedAt
Image
Hospitals
HospitalID
Name
Address
Phone
Email
CreatedAt
UpdatedAt
Image

```
C# AddUserRolesModel.cs
    AddUserRolesModel
        Email : string
        Roles : string[]
```

```
C# LoginModel.cs
    LoginModel
        Email : string
        Password : string
```

```
C# Department.cs
    Department
        DepartmentID : int
        HospitalID : int
        Name : string
        Description : string
        CreatedAt : DateTime
        UpdatedAt : DateTime
        Hospital : Hospital
        Image : byte[]
        Doctors : ICollection<Doctor>
```

```
C# Doctor.cs
    Doctor
        DoctorID : int
        UserId : string
        HospitalID : int
        DepartmentID : int
        Name : string
        Specialization : string
        Phone : string
        Email : string
        Degree : string
        Experience : string
        CreatedAt : DateTime
        UpdatedAt : DateTime
        Image : byte[]
        IsAvailable : bool
        Fees : decimal?
        Hospital : Hospital
        Department : Department
        Appointments : ICollection<Appointment>
```

```
Hospital.cs
  Hospital
    HospitalID : int
    Name : string
    Address : string
    Phone : string
    Email : string
    Image : byte[]
    CreatedAt : DateTime
    UpdatedAt : DateTime
    Doctors : ICollection<Doctor>
    Departments : ICollection<Department>
```

```
TreatmentRecord.cs
  TreatmentRecord
    TreatmentRecordID : int
    PatientID : int
    Patient : Patient
    DoctorID : int
    Doctor : Doctor
    AppointmentID : int
    Appointment : Appointment
    TreatmentDate : DateTime
    Description : string
    CreatedAt : DateTime
    UpdatedAt : DateTime
    IsVisibleToPatient : bool
```

```
Appointment.cs
  Appointment
    AppointmentID : int
    PatientID : int
    Patient : Patient
    DoctorID : int
    Doctor : Doctor
    AppointmentDate : DateTime
    AppointmentTime : TimeSpan
    Reason : string
    Status : AppointmentStatus
    CreatedAt : DateTime
    UpdatedAt : DateTime
    TreatmentRecordID : int
    TreatmentRecord : TreatmentRecord
```

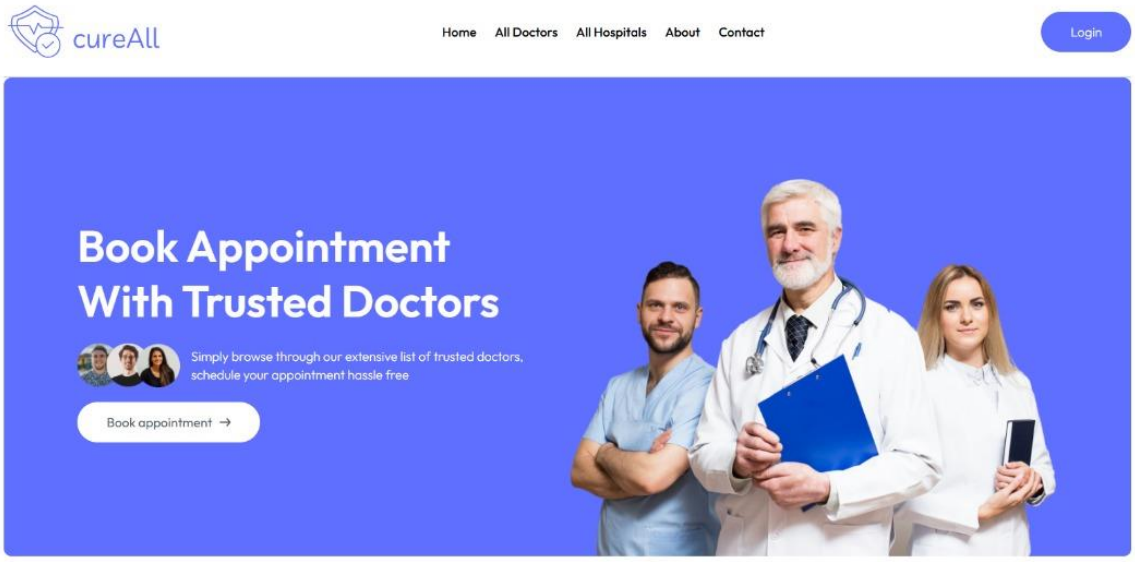
```
RoleModel.cs
  RoleModel
    Id : Guid
    Name : string
```

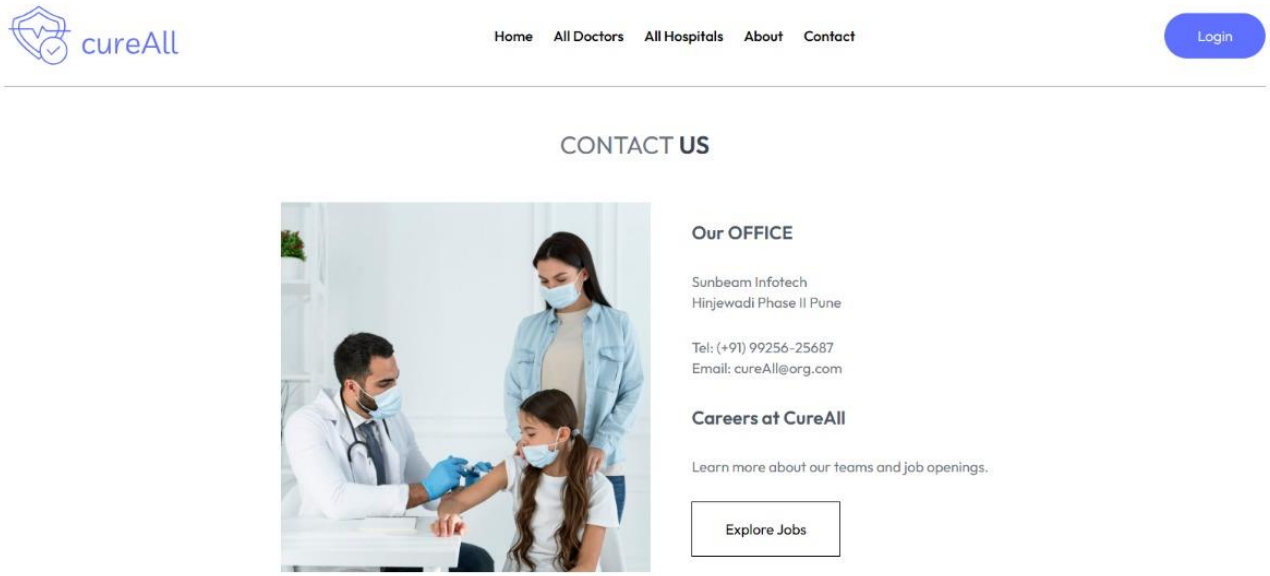
```
Admin.cs
  Admin
    AdminID : int
    UserId : string
    User : IdentityUser
    Name : string
    Email : string
    CreatedAt : DateTime
    UpdatedAt : DateTime
```

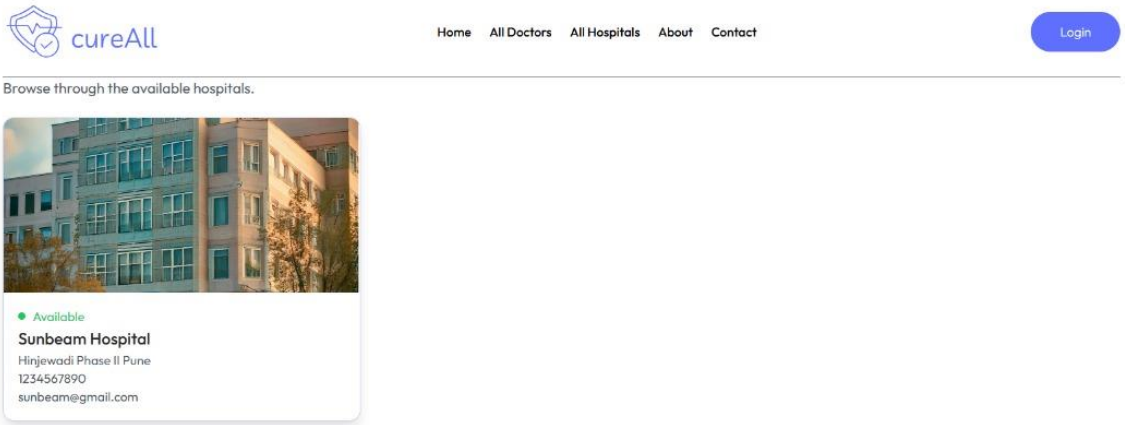
```
Patient.cs
  Patient
    PatientID : int
    UserId : string
    FirstName : string
    LastName : string
    DateOfBirth : DateTime?
    Phone : string
    Email : string
    Gender : string
    Address : string
    CreatedAt : DateTime
    UpdatedAt : DateTime
    Image : byte[]
    Appointments : ICollection<Appointment>
```

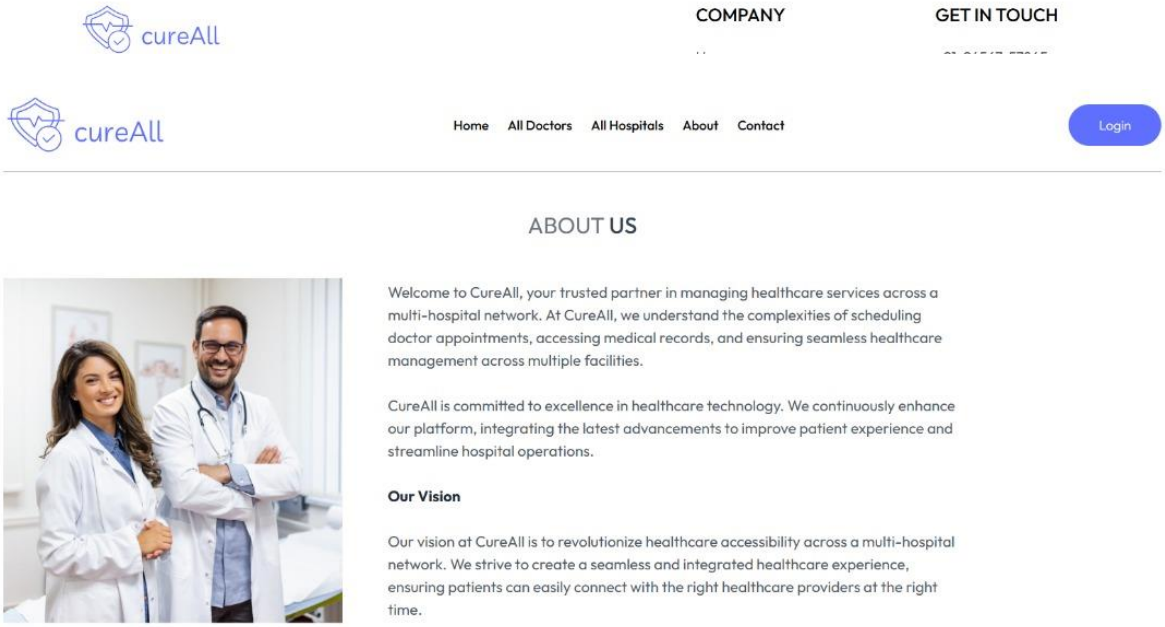
USER INTERFACE

1) Home Page

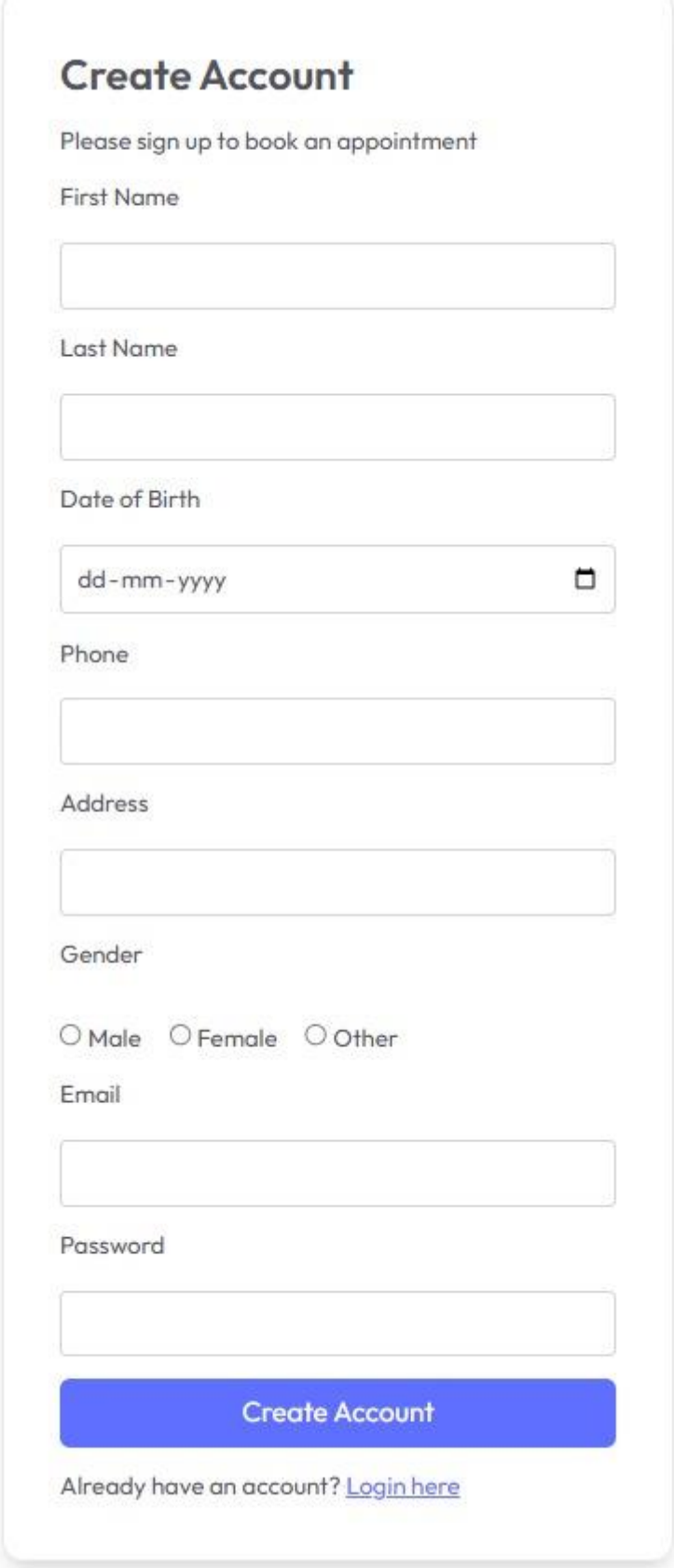




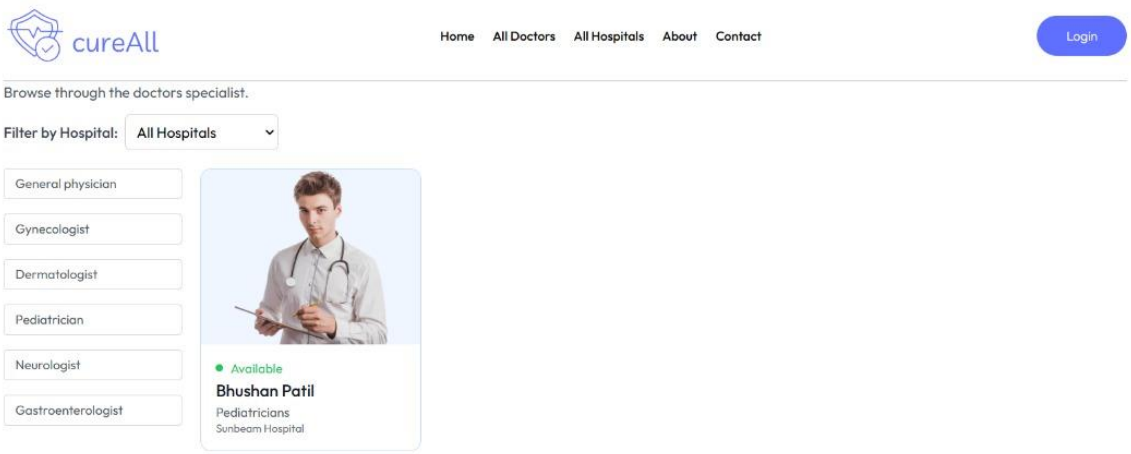




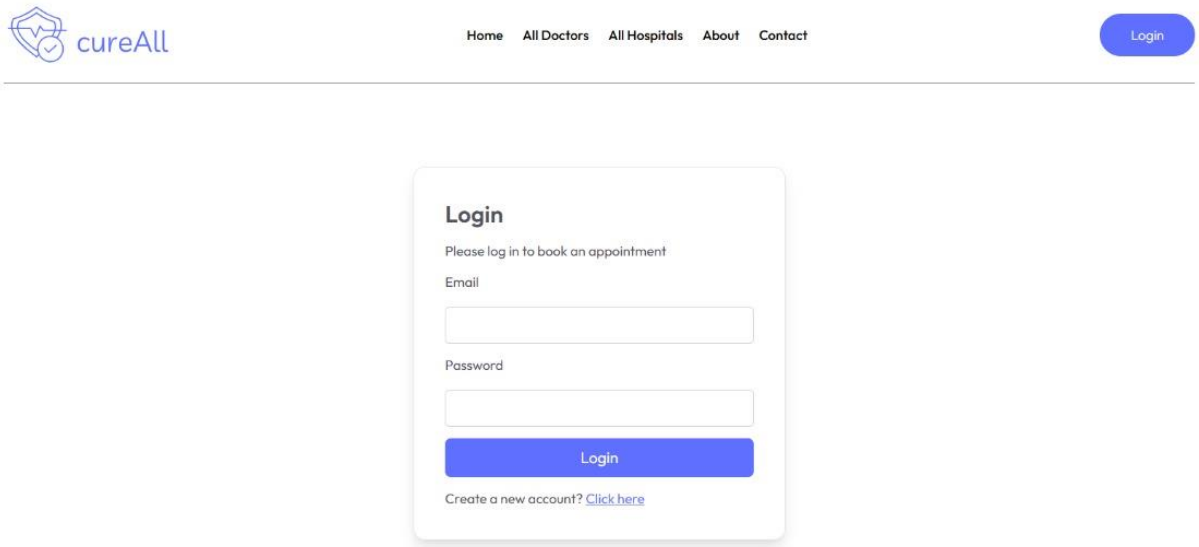
2)Create Account Page for Patient



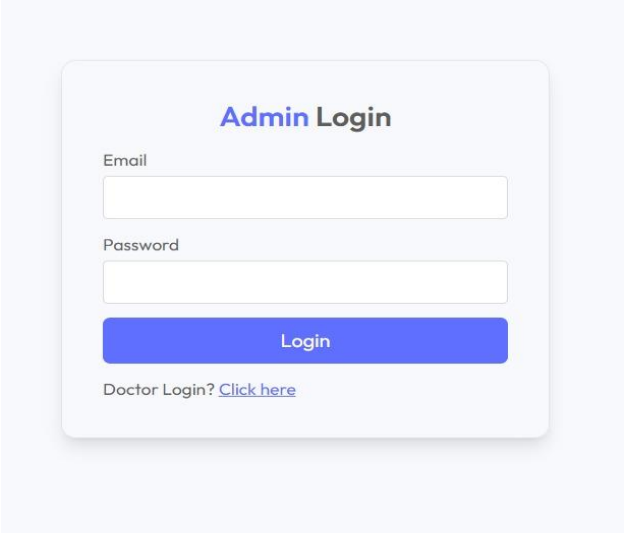
3) Search by department and hospitals



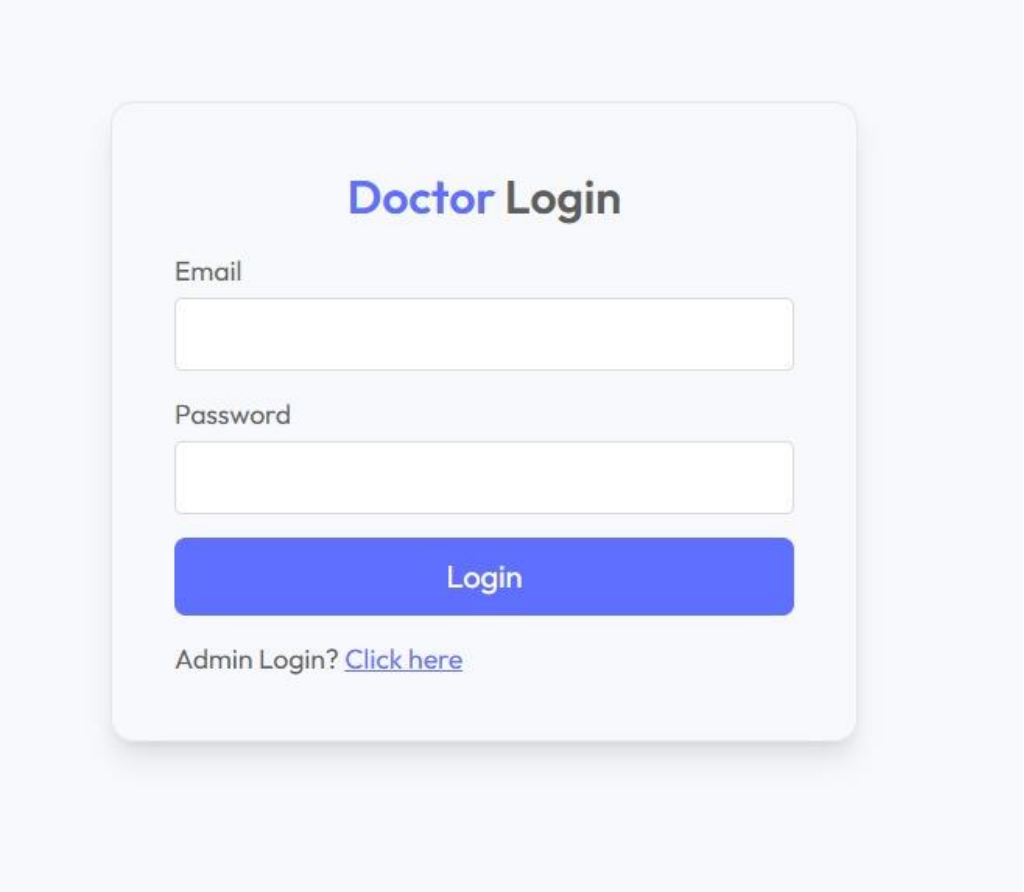
4) Login Page



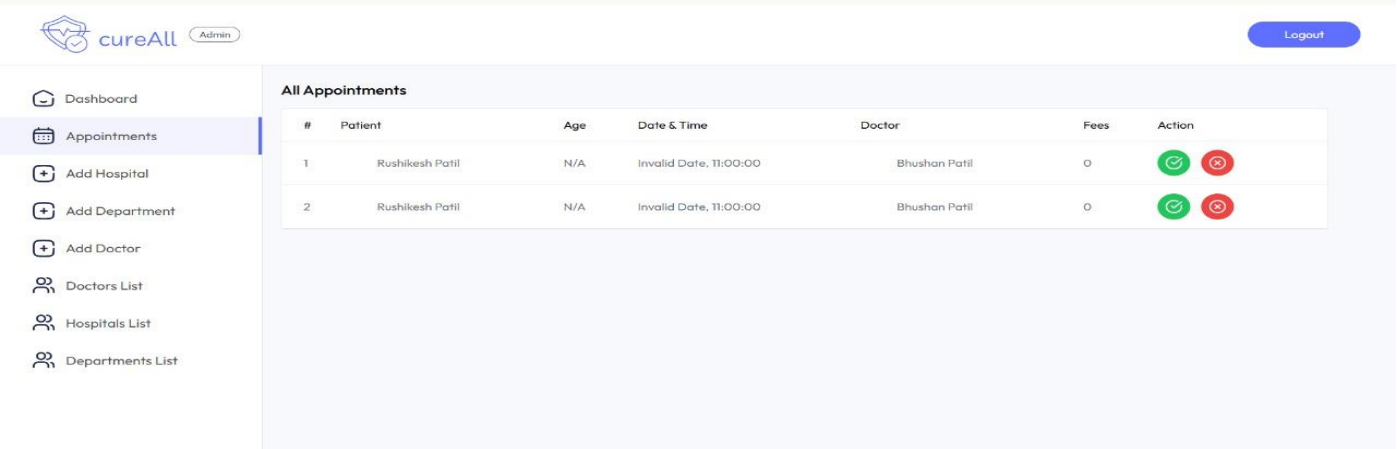
5) Admin Login



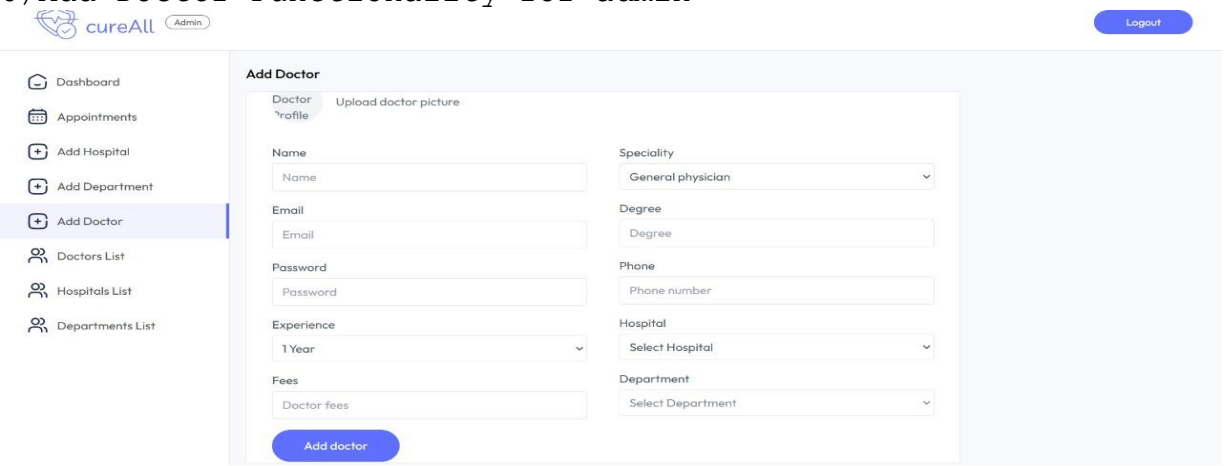
6) Doctor Login:

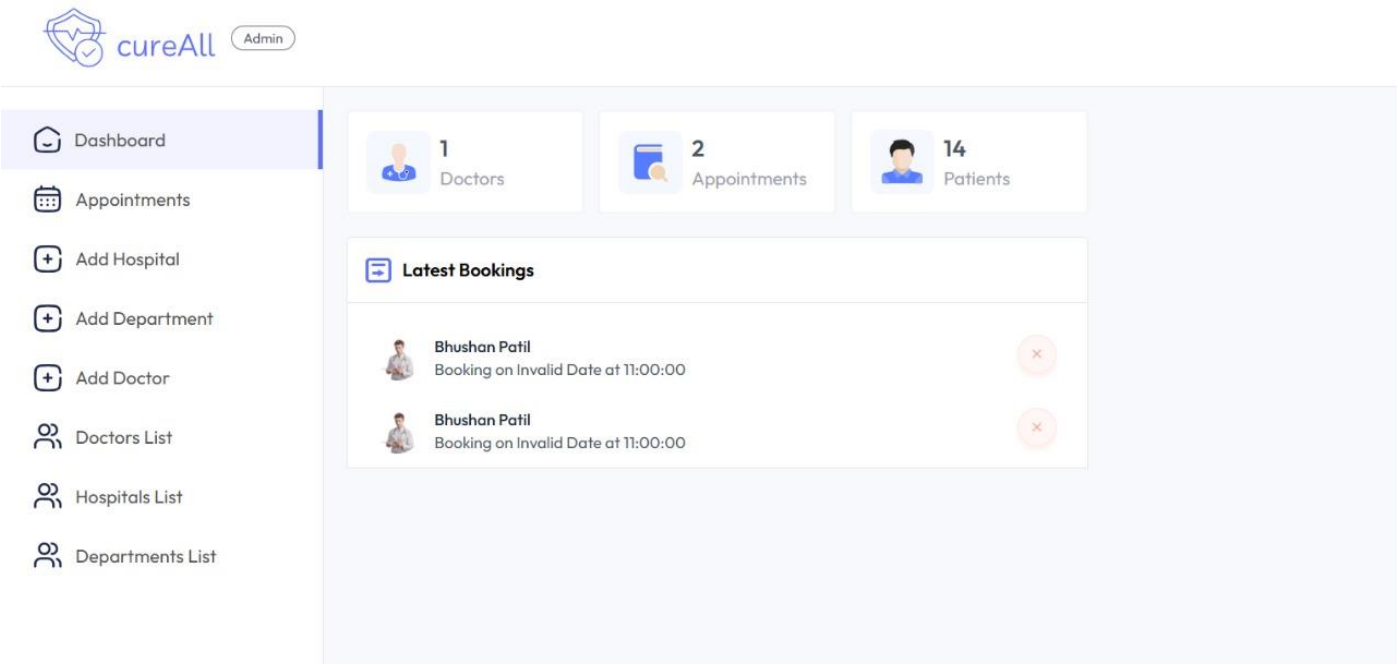


7) Home Page for Admin

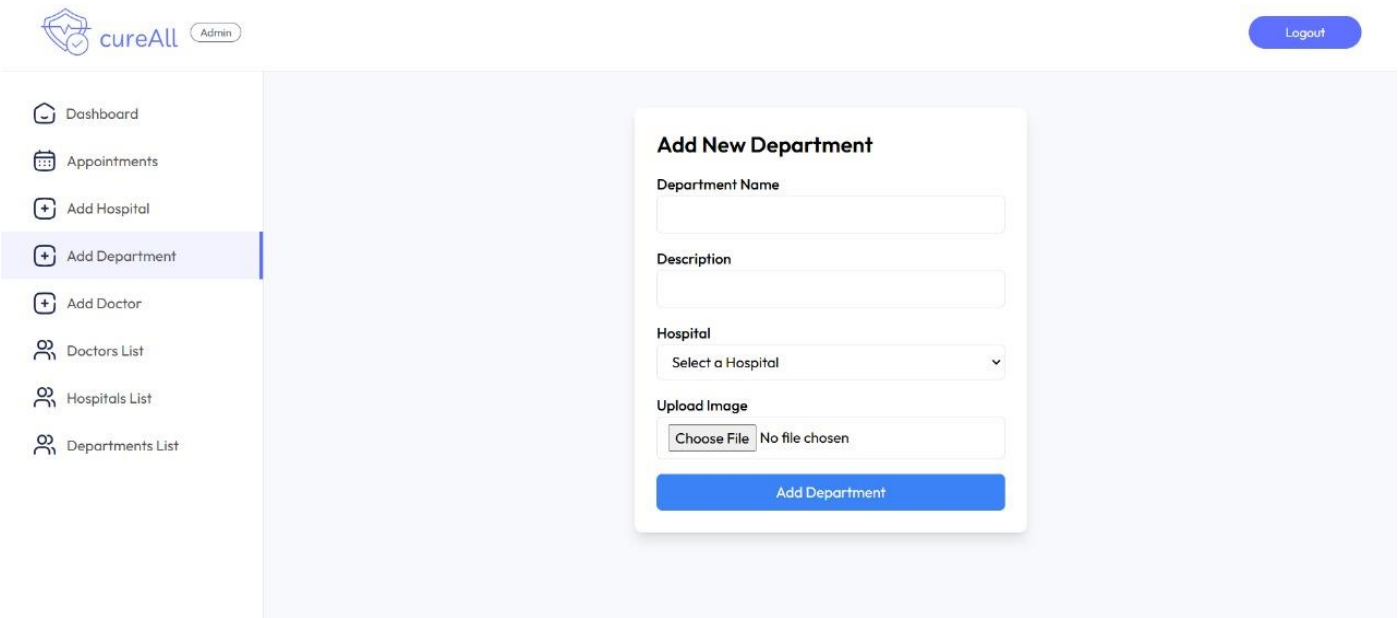


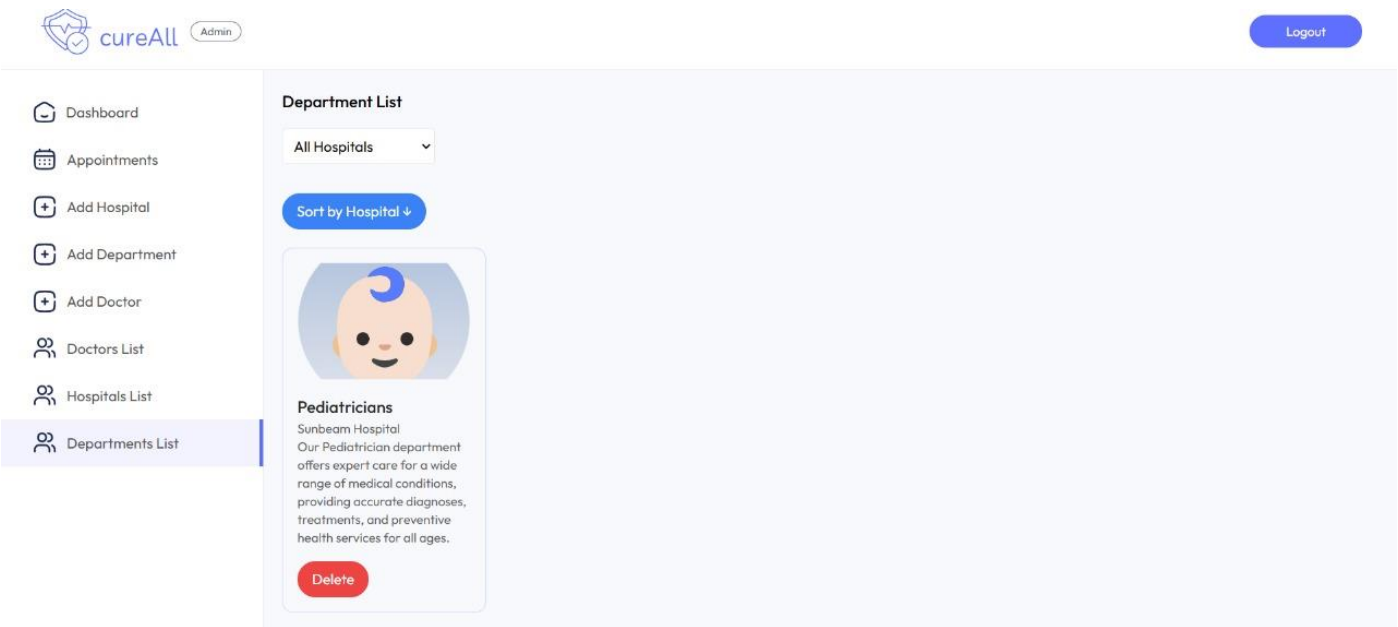
8) Add Doctor Functionality for admin





9)Add new Dept for Admin





10)Add new Hospital feature for admin