Hospital Management System

Team Members

- **Ahmed Mahmoud**: Fullstack .NET Web Developer
- **Shahd Mahmoud**: .NET Backend Developer
- **Tasneem Eid**: .NET Backend Developer
- **Mina Magdy**: .NET Backend Developer
- **Norhan Mohamed**: .NET Backend Developer

Project Overview

The **Hospital Management System** is a web-based application developed using **.NET Core 8 MVC**. This project aims to streamline the management processes within hospitals, providing a centralized, fast, and secure platform that integrates all hospital data and minimizes vulnerabilities against hacking attempts.

Project Objective

Our primary goal is to enhance the efficiency of hospital management through a secure and robust system that facilitates data handling, patient management, and administrative operations. The system aims to ensure data accuracy and integrity while providing a user-friendly interface for healthcare professionals.

Technologies Used

- **Backend**: .NET Core 8 MVC, Entity Framework, Repository Pattern, MSSQL
- **Front-End**: HTML5, CSS3, JavaScript, Bootstrap
- **Design Patterns**: Repository Pattern
- **Database**: MSSQL with Entity Framework for data management
- **Operations**: CRUD (Create, Read, Update, Delete)
- **Responsive Design**: Ensuring a seamless experience across devices
- **User Management & Identity**: Currently in progress

Project Structure

The project is organized into the following structure:

- **Data**: Manages database interactions and models.

- **Migrations**: Database schema changes and version control.
- **Models**: Data models for application entities.
- **ViewModels**: Models used specifically for views.
- **Views**: Frontend user interface using MVC pattern.
- **Controllers**: Business logic and request handling.
- **Configurations**: Application configurations and settings.
- **Repositories**: Data access layer following the Repository pattern.
- **AutoMapper**: Object mapping for DTOs.
- **wwwroot**: Static files such as CSS, JavaScript, and images.

1. Models:

The **Models** folder contains definitions of the entities used within the application, where each entity represents a specific table in the database. Here are some of the entities in the project:

- **Admin.cs**: Represents the data of the system administrator. It includes fields related to admin information such as name, email, and credentials that can be used for authentication processes.
- **Booking.cs**: Manages patient bookings within the system, such as scheduling appointments between patients and doctors and recording visit details.
- **Department.cs**: Contains information about the medical departments in the hospital (e.g., Surgery Department, Pediatrics Department, etc.).
- **Doctor.cs**: Holds fields related to doctor information such as name, specialty, working hours, and relationships with other tables like **Booking**.
- **DoctorNurse.cs**: Likely represents the relationship between doctors and nurses to facilitate operations such as assigning specific nurses to groups of doctors.
- **MedicalRecord.cs**: Represents patients' medical records, including details about diagnoses, treatments, and medical history.
- **Nurse.cs**: Contains data about nurses working in the hospital, including name and specialties.
- **Patient.cs**: Represents patient data such as name, age, gender, and relationships with **MedicalRecord** and **Booking**.
- **SuperAdmin.cs**: Likely refers to a user with higher privileges in the system, such as overall control of hospital settings.
- **Identity Folder**: Contains entities related to user and identity management to facilitate authentication processes in the system.

2. Controllers:

The **Controllers** folder contains classes that handle the logic of HTTP requests and send data to the **Views**. Each class has a specific function, acting as a bridge between the user interface and the database.

- **AdminController.cs**: Manages all operations related to administrators, such as creating, updating, or deleting admin accounts and defining their permissions.
- **DoctorController.cs**: Handles requests related to doctors, such as displaying the list of doctors, adding a new doctor, or modifying existing doctor information.
- **HomeController.cs**: Typically responsible for requests related to the home page or the main interface of the application.
- **NurseController.cs**: Manages the administration of nurses' data, such as adding new nurses or displaying the list of nurses in the hospital.
- **PatientController.cs**: Handles operations related to patients, such as registering patients, managing their information, and booking appointments.
- **SuperAdminController.cs**: Likely responsible for operations that require advanced privileges in the system, such as managing primary users or overseeing all departments.

3. Views:

The **Views** folder contains the user interface (UI) elements for the application. Each **View** corresponds to the visual representation of data for a particular **Model** and is managed by its respective **Controller**. Here is a breakdown of the contents:

- **Admin**: Contains Razor View files (e.g., `.cshtml` files) responsible for displaying the UI related to administrator functionalities. This could include:
    - Viewing, creating, or updating administrator profiles.
    - Add, Delete and Edit Medical Records.
    - Viewing appointments.
- **Doctor**: Contains the views related to doctor functionalities, such as:
    - Displaying the list of doctors.
    - Managing doctor profiles and their schedules.
    - Allowing doctors to view and update their patient list or appointments.
- **Home**: Manages the views for the main home page of the application. It may include:
    - The landing page of the web app.
    - Common navigation links for users to access different sections of the hospital management system.
- **Nurse**: Contains views that handle nurse-specific tasks, including:
    - Listing all nurses.
    - Managing nurse profiles.
    - Assigning nurses to doctors or departments.
- **Patient**: Contains views that relate to patient management, such as:
    - Patient registration and profile management.
    - Viewing appointment details.
    - Accessing medical records and treatment history.
- **SuperAdmin**: Contains views for functionalities exclusive to users with **SuperAdmin** privileges. This may include:
    - High-level control over the entire system.
    - Viewing system analytics or usage reports.
    - Configuring system-wide settings.

- **Shared**: Contains reusable view components that can be used across different parts of the application. Key files include:
    - **_Layout.cshtml**: This is the master layout file that provides a common structure for all pages, such as headers, footers, and navigation bars.
    - **_ValidationScriptsPartial.cshtml**: Includes scripts for client-side validation.
    - **Error.cshtml**: Manages error pages, providing user-friendly error messages when something goes wrong.
    - **_ViewImports.cshtml**: Includes directives and common namespaces that can be shared across views.
    - **_ViewStart.cshtml**: Defines the default layout for views in the application, ensuring a consistent look and feel across pages.

Summary of the Relationship between Views and Controllers:

- **Views** handle the presentation logic and are responsible for displaying data to users.
- Each **Controller** typically returns a **View** after processing data, making the application dynamic and interactive.

For instance, when a user requests to see the list of patients, the **PatientController** will fetch the necessary data and return a **View** that presents this data in a user-friendly format.

Summary of the Relationship between Models and Controllers:

- **Models** provide the fundamental structure to represent data.
- **Controllers** handle CRUD (Create, Read, Update, Delete) operations for these entities.

Each controller can call a specific model to retrieve or modify data, then pass that data to a **View** to be displayed in the user interface. For example, when the system needs to display a list of doctors, the **DoctorController** will call the **Doctor Model** to fetch the list from the database and then present it on the user interface.

Future Enhancements

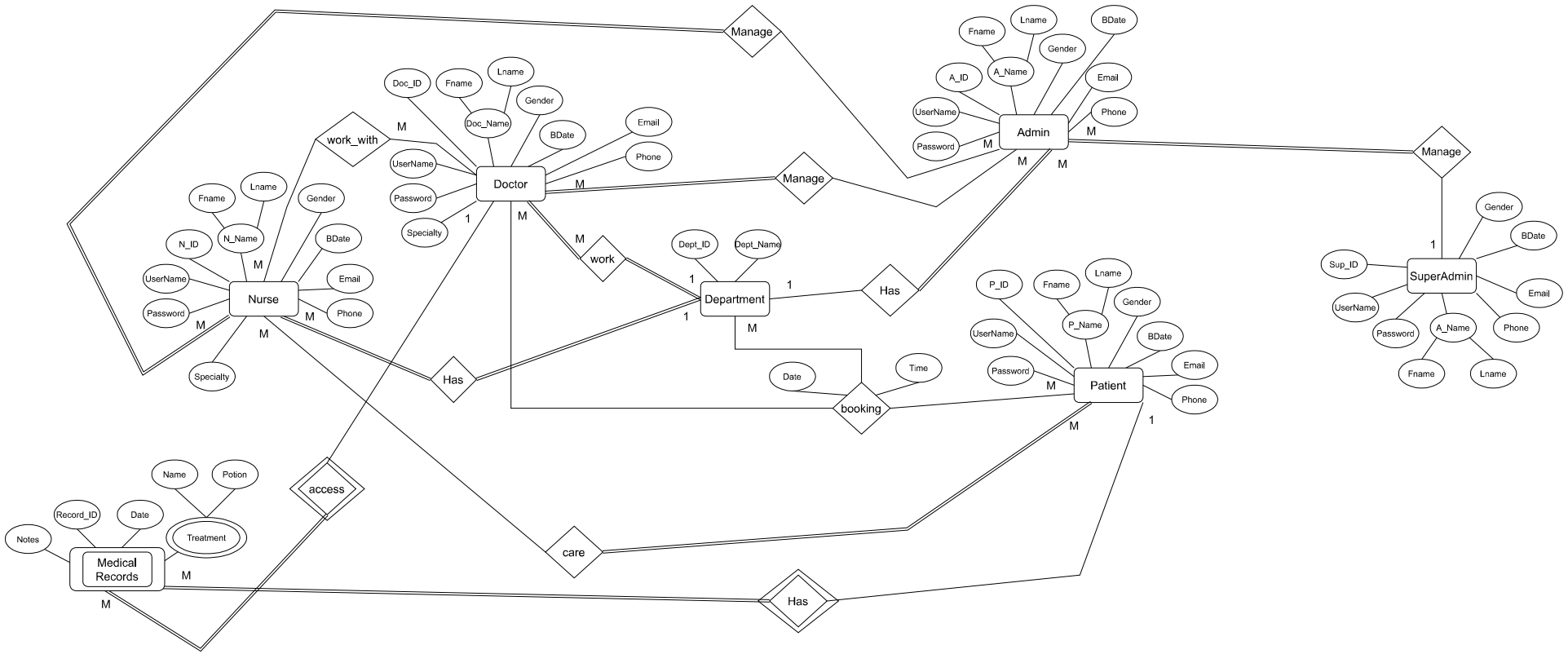
We have identified several areas for further improvements, including:

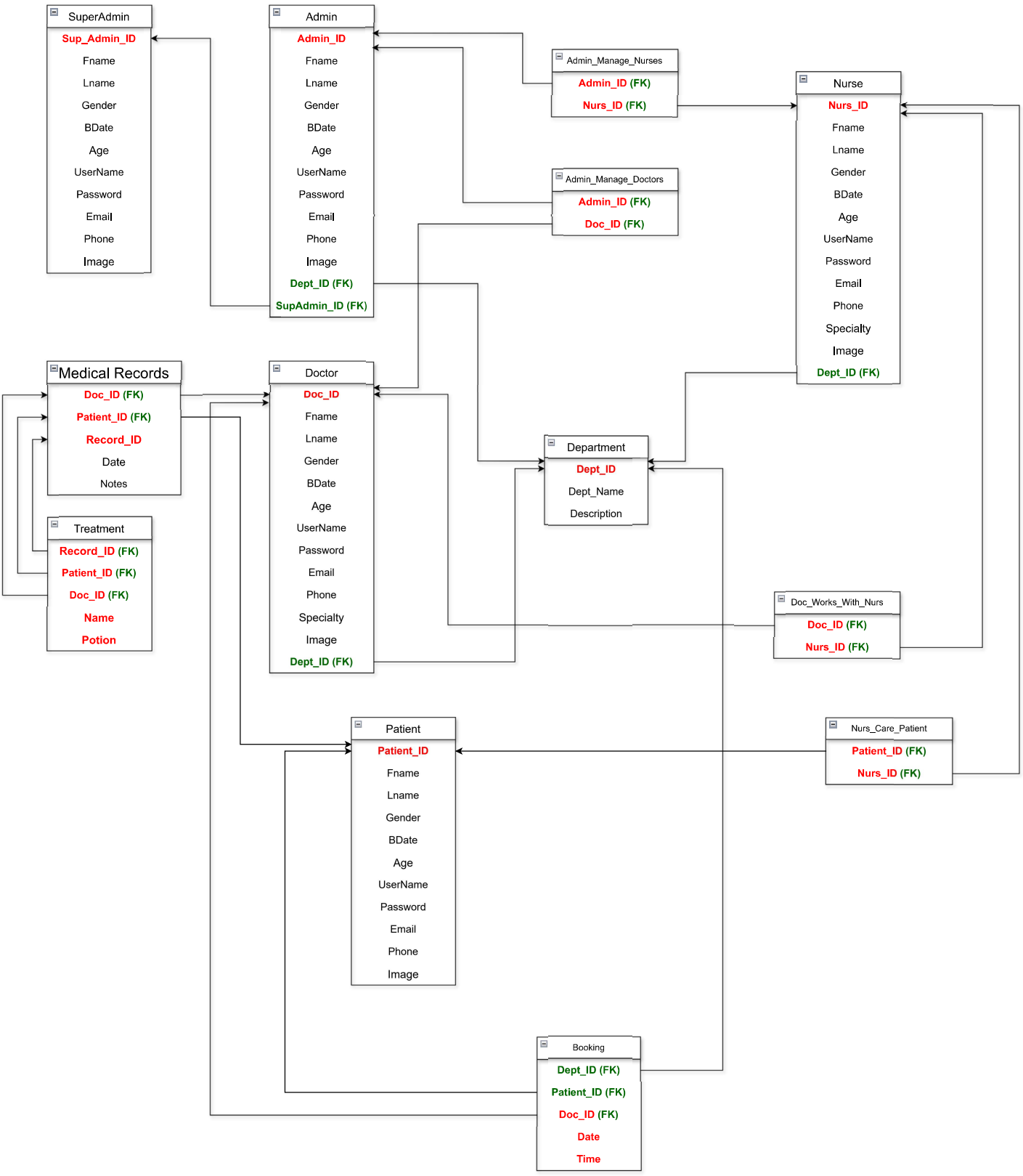
- **User Management & Identity**: Complete the implementation for secure user authentication and authorization.
- **N-Tier Architecture**: Refactor the application for better separation of concerns and scalability.
- **Localization**: Implement multi-language support for a broader user base.

- **Payment Integration**: Integrate popular payment gateways like **Stripe** or **PayPal** for processing payments.
- **Frontend Framework**: Use **Angular** for improved frontend functionality, and enhance features using **jQuery**, **DataTables**, and custom buttons.

How to Run the Project

1. Clone the repository.
2. Set up the database using **MSSQL**.
3. Update the **appsettings.json** with your database connection string.
4. Run **dotnet ef database update** to apply migrations.
5. Build and run the application using **Visual Studio** or **dotnet CLI**.
6. Access the web application through your local server (e.g., `https://localhost:5001`).


Manage
Admin
A_ID
A_Name
Fname
Lname
Gender
BDate
Email
Phone
UserName
Password
M
M
M
Manage
SuperAdmin
1
Sup_ID
Gender
BDate
Email
Phone
A_Name
Fname
Lname
UserName
Password
work_with
M
Doctor
Doc_ID
Doc_Name
Fname
Lname
Gender
BDate
Email
Phone
UserName
Password
Specialty
1
M
Manage
M
M
work
M
Department
Dept_ID
Dept_Name
1
1
1
M
Has
Nurse
N_ID
N_Name
Fname
Lname
Gender
BDate
Email
Phone
UserName
Password
Specialty
M
M
M
M
Has
Patient
P_ID
P_Name
Fname
Lname
Gender
BDate
Email
Phone
UserName
Password
M
M
1
booking
Date
Time
access
Medical Records
Record_ID
Date
Notes
Name
Potion
Treatment
M
M
care
Has


SuperAdmin
Sup_Admin_ID
Fname
Lname
Gender
BDate
Age
UserName
Password
Email
Phone
Image
Admin
Admin_ID
Fname
Lname
Gender
BDate
Age
UserName
Password
Email
Phone
Image
Dept_ID (FK)
SupAdmin_ID (FK)
Admin_Manage_Nurses
Admin_ID (FK)
Nurs_ID (FK)
Admin_Manage_Doctors
Admin_ID (FK)
Doc_ID (FK)
Nurse
Nurs_ID
Fname
Lname
Gender
BDate
Age
UserName
Password
Email
Phone
Specialty
Image
Dept_ID (FK)
Medical Records
Doc_ID (FK)
Patient_ID (FK)
Record_ID
Date
Notes
Treatment
Record_ID (FK)
Patient_ID (FK)
Doc_ID (FK)
Name
Potion
Doctor
Doc_ID
Fname
Lname
Gender
BDate
Age
UserName
Password
Email
Phone
Specialty
Image
Dept_ID (FK)
Department
Dept_ID
Dept_Name
Description
Doc_Works_With_Nurs
Doc_ID (FK)
Nurs_ID (FK)
Patient
Patient_ID
Fname
Lname
Gender
BDate
Age
UserName
Password
Email
Phone
Image
Nurs_Care_Patient
Patient_ID (FK)
Nurs_ID (FK)
Booking
Dept_ID (FK)
Patient_ID (FK)
Doc_ID (FK)
Date
Time