**Hospital Management System Documentation**

**Project Overview**

**Project Name**: CareMaster  
**Technology Stack**: ASP.NET Core, MVC, Entity Framework, SQL Server, Repository Design Pattern, User Authentication  
**Objective**: The goal of this project is to build a web application that allows hospital administrators to manage appointments, patient records, staff scheduling, and a pharmacy. The system includes role-based access control, allowing patients, doctors, pharmacists, and administrators to interact with the system according to their roles.

**1. System Analysis**

**1.1. Problem Statement**

Hospitals often face challenges in efficiently managing patient records, appointments, staff schedules, and pharmacy operations. Manual processes may lead to errors, inefficiencies, and delays, affecting the overall quality of care. Specific problems identified include:

* **Appointment Conflicts**: Manual appointment scheduling may result in conflicts or overbookings, causing delays and frustration for patients.
* **Medical Record Inaccessibility**: In some cases, doctors may not have timely access to patients' medical records, hindering diagnosis and treatment.
* **Staff Scheduling Issues**: Inefficient scheduling of staff (especially doctors and nurses) can lead to shortages or overstaffing, impacting operational efficiency.
* **Pharmacy Management**: Managing pharmacy stock, dispensing medications, and tracking prescriptions manually is error-prone and time-consuming, potentially leading to medication errors.
* **Communication with Patients**: Patients may not receive timely reminders for their appointments or prescription refills.

**1.2. Solution Overview**

The Hospital Management System (HMS) will provide solutions for these issues by:

* **Automated Appointment Scheduling**: The system will prevent appointment conflicts and allow patients and administrators to view, book, and manage appointments in real-time.
* **Centralized Medical Records**: Doctors will have secure, role-based access to patient medical records, allowing them to update and retrieve medical data efficiently.
* **Efficient Staff Scheduling**: The system will allow administrators to manage and optimize staff schedules based on availability, reducing shortages or overstaffing.
* **Pharmacy Management System**: A dedicated module for the pharmacy will be implemented, enabling pharmacists to track stock levels, manage prescriptions, and dispense medications. Doctors can also send prescriptions electronically to the pharmacy.
* **Notification System**: Patients will receive automated notifications for upcoming appointments and prescription refills via email or SMS.

**1.3. System Features**

* **User Authentication & Authorization**: Role-based access control for patients, doctors, pharmacists, and administrators.
* **Patient Management**: Create, read, update, and delete (CRUD) operations for patient data.
* **Appointment Scheduling**: Patients can book appointments, and administrators can manage schedules.
* **Staff Management**: Administrators manage staff (doctors, nurses) and their availability.
* **Pharmacy Management**: Pharmacists can manage drug inventory, process prescriptions, and track dispensing activities.
* **Medical Record Management**: Doctors can view and update patient medical records.
* **Notifications**: Email/SMS notifications for appointment reminders and prescription refills.

**2. System Design**

**2.1. Architecture**

The system follows the MVC (Model-View-Controller) architecture. ASP.NET Core will be used for creating the web application, while Entity Framework will handle database interactions with SQL Server.

In addition, the **Repository Design Pattern** will be used to abstract the data access layer from the business logic. This will promote loose coupling and make the system easier to maintain and extend.

* **Model**: Represents data structure (patients, staff, appointments, medical records, pharmacy).
* **View**: The UI layer for displaying data and collecting user input.
* **Controller**: Handles user requests and interacts with the Model and View to render responses.
* **Repository**: Encapsulates data access logic and provides an abstraction layer to interact with the database.

**2.2. Database Design**

The system will use SQL Server for data storage, managed through Entity Framework (code-first approach).

**Database Schema:**

* **Patients**: Stores patient information (ID, name, contact, medical history).
* **Appointments**: Tracks scheduled appointments, including patient ID, doctor ID, date, and time.
* **Staff**: Stores staff details (ID, name, role, schedule, and department).
* **Medical Records**: Tracks patient diagnoses, treatments, and visit history.
* **Pharmacy**: Manages drug inventory, prescriptions, and dispensing records.
* **Users**: Stores authentication data, including roles (patients, doctors, pharmacists, administrators).

**3. Features Implementation**

**3.1. Setup and Patient Management**

**Tasks:**

* **Project Setup**: Install and configure ASP.NET Core, MVC, and Entity Framework.
* **Repository Pattern**: Set up repositories for Patient, Appointment, Staff, and Pharmacy entities to handle database operations.
* **Patient Management**: Administrators can add, edit, and delete patient details using a web interface.
* **User Authentication**: Implement role-based authentication with separate access for patients, doctors, pharmacists, and administrators.

**Deliverables:**

* Basic project setup with functioning patient management.
* Repository pattern implemented for data access.
* User authentication enabled.

**3.2. Appointment Scheduling, Staff Management, and Pharmacy Module**

**Tasks:**

* **Appointment Scheduling**: Allow administrators and doctors to schedule and manage appointments.
* **Staff Management**: Implement CRUD operations for managing staff details and schedules.
* **Pharmacy Module**: Implement a pharmacy module where pharmacists can manage inventory, process prescriptions, and track dispensing.
* **Role-Based Access**: Ensure only authorized users (e.g., administrators, pharmacists) can manage relevant data.
* **Testing**: Test the functionalities developed for patient, staff, and pharmacy management.

**Deliverables:**

* Functional appointment scheduling and staff management system.
* Functional pharmacy management system.
* Role-based access control implemented.

**3.3. Medical Records and Notifications**

**Tasks:**

* **Medical Records**: Implement functionality for doctors to view and update patient medical records.
* **Notifications**: Set up email or SMS notifications for appointment reminders and prescription refills.
* **Testing**: Ensure medical records and notifications work as expected.

**Deliverables:**

* Fully functional medical record system.
* Notification system for patient reminders and prescription refills.
* Completed testing of the new features.

**3.4. Final Testing, UI Enhancements, and Deployment**

**Tasks:**

* **UI Enhancements**: Improve the user interface for better usability and responsiveness.
* **Final Testing**: Comprehensive testing of all features (appointments, medical records, pharmacy, staff, patients).
* **Deployment**: Deploy the system to a cloud hosting provider.
* **Documentation**: Prepare and finalize technical documentation, user guides, and deployment instructions.

**Deliverables:**

* Fully functional, tested, and deployed hospital management system.
* Complete project documentation and user guides.

**4. Technologies Used**

* **ASP.NET Core**: Web application framework.
* **MVC**: Architectural pattern used to organize the application.
* **Repository Design Pattern**: Provides an abstraction layer for data access, ensuring separation of concerns and making the system more maintainable.
* **Entity Framework**: ORM for managing database operations.
* **SQL Server**: Database for storing hospital data.
* **Email/SMS Service**: For sending notifications to patients.
* **User Authentication**: Role-based authentication using ASP.NET Identity.

**5. Testing and Quality Assurance**

**5.1. Unit Testing**

* Use unit tests to validate the logic for appointment scheduling, patient management, medical records updates, and pharmacy functionality. Ensure repositories are correctly handling database operations.

**5.2. Integration Testing**

* Verify that database interactions through the Repository layer and Entity Framework are working as expected.

**5.3. User Acceptance Testing**

* Involve real users (patients, doctors, pharmacists, administrators) to ensure the system meets their needs.

**6. Deployment**

**6.1. Current Testing and Code Management**

* Local Machine Testing: At this stage, the system is being tested on a local development environment to ensure that all features are working as expected.
* Version Control with GitHub: The code is published to GitHub to manage versions, collaborate, and keep a history of changes. It allows easy access to the latest stable versions of the project and facilitates teamwork.

**6.2. Future Deployment Plan**

* Full Deployment: Once the system is 100% complete, the plan is to deploy it to a cloud hosting provider such as Azure or AWS. This will include setting up a production environment with a full CI/CD pipeline to automate the deployment process.
* Production Environment: The deployment will involve configuring a live environment where users (patients, doctors, pharmacists, administrators) can access the system in real-time.

**7. User Guide**

**7.1. For Administrators**

* **Login**: Use administrator credentials to log in.
* **Manage Patients**: Navigate to the patient management page to add, edit, or delete patient details.
* **Manage Appointments**: Go to the appointment page to schedule or modify appointments.
* **Manage Staff**: Navigate to the staff management page to update doctor and nurse schedules.

**7.2. For Doctors**

* **Login**: Use doctor credentials to log in.
* **View Medical Records**: Access the medical records section to view or update patient information.

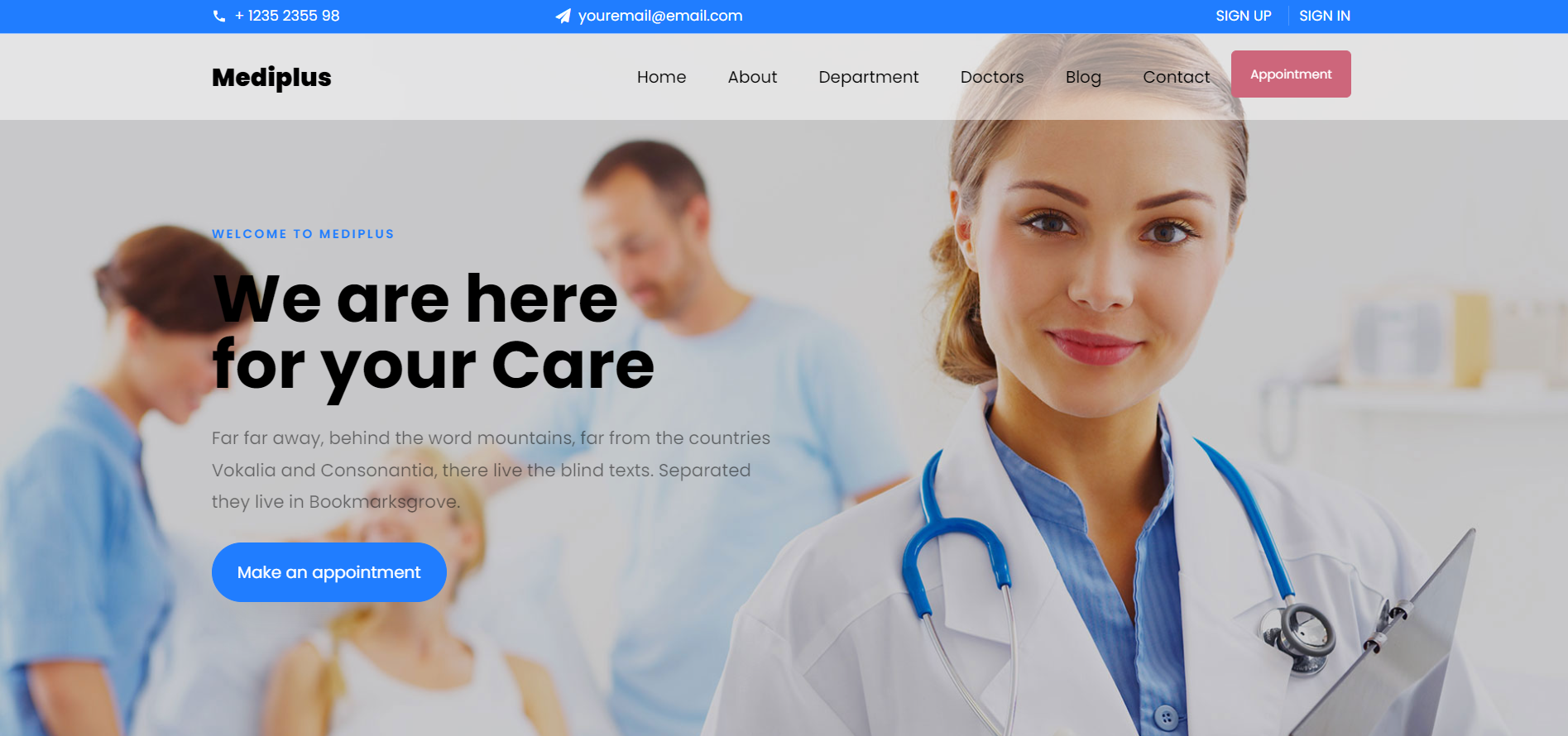
**7.3. For Pharmacists**

* **Login**: Use pharmacist credentials to access the pharmacy module.
* **Manage Prescriptions**: Process prescriptions sent by doctors and manage drug stock levels.

**7.4. For Patients**

* **Login**: Use patient credentials to log in.
* **View Appointments**: Check your scheduled appointments under the appointments section.
* **Receive Notifications**: Stay updated on upcoming appointments and prescription refills through email/SMS notifications

**8. Prototype**



A person and person holding hands

Description automatically generated

A screenshot of a medical website

Description automatically generated

A screenshot of a medical website

Description automatically generated

A group of people in white coats

Description automatically generated

A screenshot of a contact us page

Description automatically generated

A screenshot of a computer

Description automatically generated