Doctor Appointment Management System

Project Documentation

Created By: Gautam Kumawat

Abstract

The Doctor Appointment Management System (DAMS) is a comprehensive web-based application designed to streamline the process of scheduling and managing medical appointments. This system bridges the gap between patients and healthcare providers, enhancing accessibility and efficiency in the healthcare sector. Patients can easily register, book, and manage appointments with their preferred doctors through a user-friendly interface. The system also features automated notifications, ensuring users receive timely reminders about their upcoming appointments. On the provider side, doctors can manage their schedules, view patient histories, and improve their overall practice management. Built on a robust architecture using PHP and MySQL, the application ensures data security and reliability. Additionally, Azure services like Virtual Machines, DNS, and Scale Sets enhance the system's performance and scalability. The implementation of a chatbot further improves user interaction, allowing for instant query resolution. This project not only addresses common challenges in appointment management but also fosters a more efficient healthcare experience for both patients and providers. Ultimately, DAMS aims to contribute to better patient care and optimized healthcare workflows.

Table of Contents

1. Introduction

Overview

The Doctor Appointment Management System (DAMS) is a web-based application designed to streamline the process of booking, managing, and attending doctor appointments. It facilitates interactions between patients and healthcare providers, making it easier to schedule visits, manage patient records, and improve overall healthcare service efficiency.

Objectives

- To enable patients to book, cancel, and reschedule appointments online.
- To allow doctors to manage their schedules effectively.
- To maintain patient records securely.
- To provide notifications and reminders for appointments.

2. System Requirements

Functional Requirements

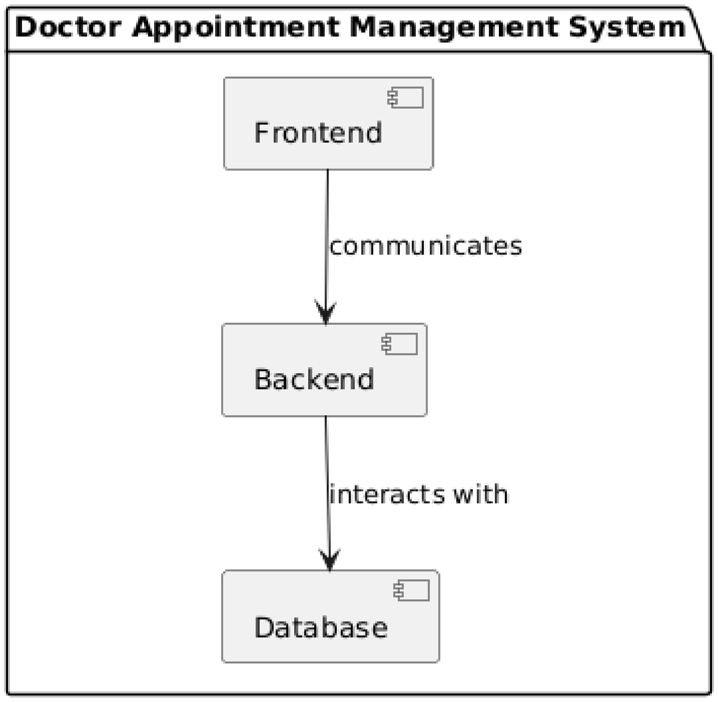
- User registration and authentication for patients and doctors.
- Appointment booking and cancellation.
- Viewing available slots and appointment history.
- Sending notifications via email/SMS.
- Admin panel for managing users and appointments.

Non-functional Requirements

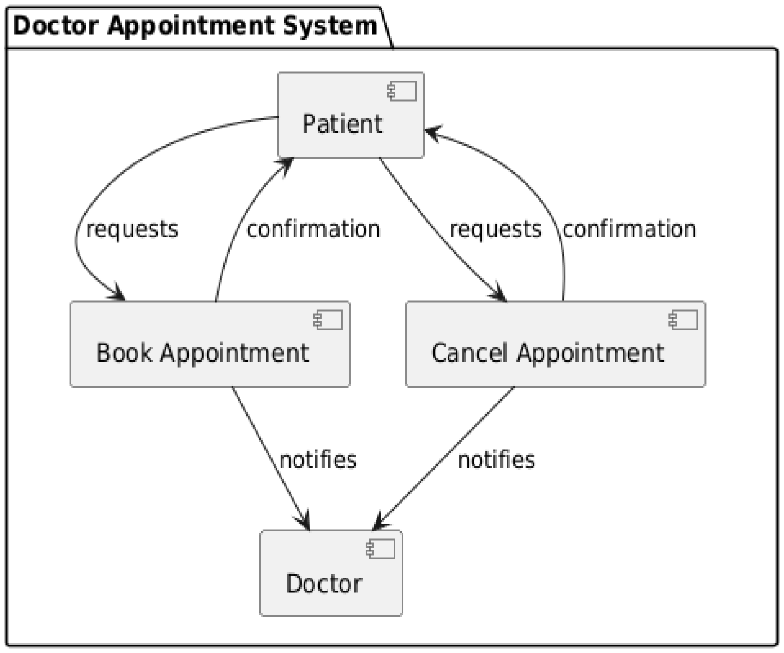
- Scalability to handle a large number of users.
- Security measures to protect patient data.
- Usability to ensure a user-friendly interface.
- Performance to ensure quick response times.

3. System Architecture

High-Level Architecture Diagram

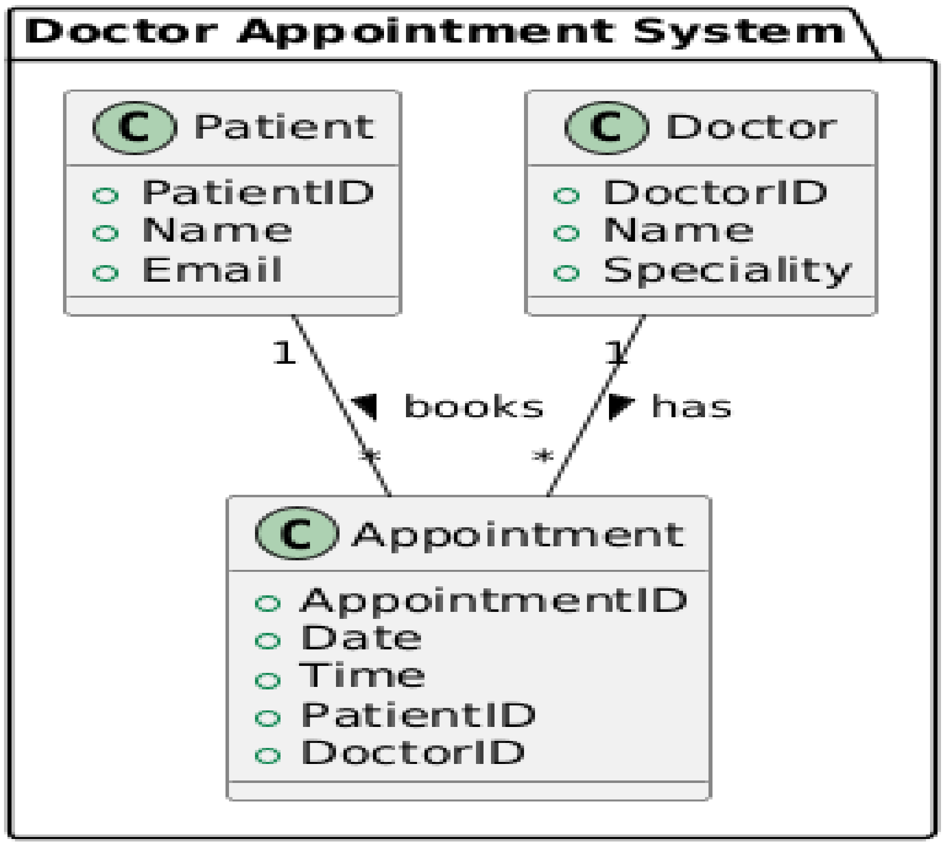


Component Diagram



4. Database Design

Entity-Relationship Diagram (ERD)



Database Tables and Relationships

- **Patients Table**: Contains patient information.
- **Doctors Table**: Contains doctor information and specializations.
- **Appointments Table**: Tracks scheduled appointments linking patients and doctors.

5. User Roles and Permissions

User Types

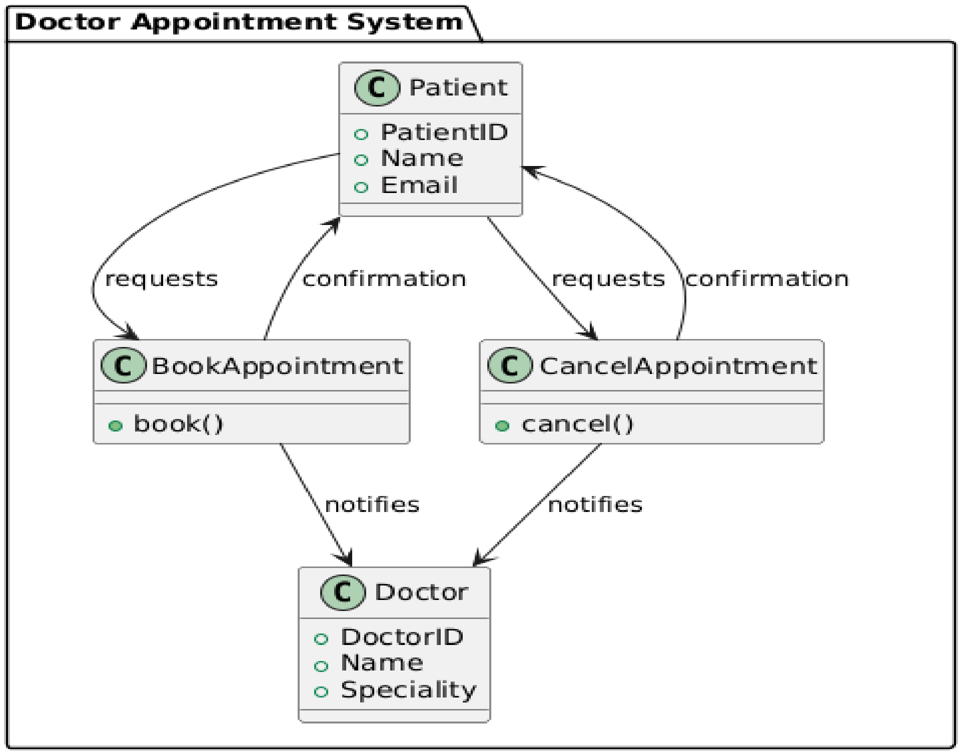
- **Patient**
- **Doctor**
- **Admin**

Permissions Matrix

Action	Patient	Doctor	Admin
Register	Yes	No	No
Book Appointment	Yes	No	No
Cancel Appointment	Yes	No	No
View Appointments	Yes	Yes	Yes
Manage Users	No	No	Yes
Manage Appointments	No	Yes	Yes

6. Use Cases

Use Case Diagrams



Detailed Use Cases

1. **Book Appointment**
   - **Actor**: Patient
   - **Precondition**: User is registered and logged in.
   - **Postcondition**: Appointment is saved in the system.
   - **Steps**:
     1. Select doctor.
     2. Choose date and time.
     3. Confirm appointment.
2. **Manage Appointments**
   - **Actor**: Doctor/Admin
   - **Precondition**: User is authenticated.
   - **Postcondition**: Appointments are updated.
   - **Steps**:
     1. View upcoming appointments.
     2. Modify or delete appointments as needed.

7. User Interface Design

UI Mockups

- **Home Page**: Displays login options and featured doctors.
- **Appointment Booking Page**: Calendar view to select dates and times.
- **User Dashboard**: Lists upcoming appointments and past visit history.

User Journey

1. **Patient Registration**: User fills out registration form.
2. **Login**: User logs in to the system.
3. **Booking Process**: User selects a doctor, date, and time to book an appointment.
4. **Appointment Confirmation**: User receives confirmation via email/SMS.

8. Implementation Plan

Technology Stack

- **Frontend**: HTML, AJAX, jQuery, JavaScript
- **Backend**: PHP, PDO, MySQL
- **Database**: MySQL
- **Hosting**: Azure

Development Phases

1. **Phase 1**: Requirement Gathering
2. **Phase 2**: Design
3. **Phase 3**: Development
4. **Phase 4**: Testing
5. **Phase 5**: Deployment

9. Testing Strategy

Testing Types

- **Unit Testing**: Testing individual components.
- **Integration Testing**: Testing the interaction between modules.
- **User Acceptance Testing**: Feedback from end-users.

Test Cases

1. **Test Case 1**: Verify user registration.
2. **Test Case 2**: Verify appointment booking process.
3. **Test Case 3**: Verify notifications are sent correctly.

10. Conclusion

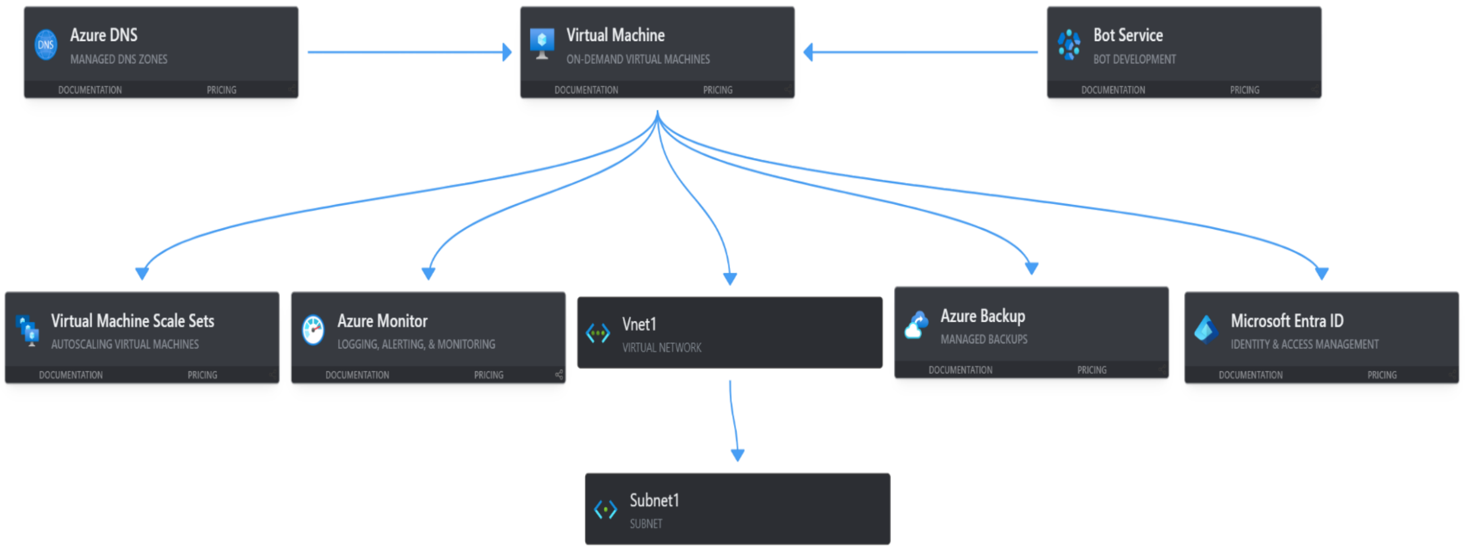
Summary

The Doctor Appointment Management System aims to simplify the process of managing doctor visits for patients and doctors alike, ensuring a smoother healthcare experience.

Future Enhancements

- Integration with telemedicine solutions.
- Mobile application development.
- Advanced analytics for appointment trends.

This document serves as a foundational guide for developing the Doctor Appointment Management System, outlining essential components, designs, and plans for implementation.


Azure DNS
MANAGED DNS ZONES
DOCUMENTATION
PRICING
Virtual Machine
ON-DEMAND VIRTUAL MACHINES
DOCUMENTATION
PRICING
Bot Service
BOT DEVELOPMENT
DOCUMENTATION
PRICING
Virtual Machine Scale Sets
AUTOSCALING VIRTUAL MACHINES
DOCUMENTATION
PRICING
Azure Monitor
LOGGING, ALERTING, & MONITORING
DOCUMENTATION
PRICING
Vnet1
VIRTUAL NETWORK
Azure Backup
MANAGED BACKUPS
DOCUMENTATION
PRICING
Microsoft Entra ID
IDENTITY & ACCESS MANAGEMENT
DOCUMENTATION
PRICING
Subnet1
SUBNET