Hexaware C# Code Challenge (3. Hospital Management System)
Mageshkannan U:
Check Drive link in ReadMe for Video Demo

Project Directory:

db.properties
.gitignore
entity/
- **Patient.cs**
- **Doctor.cs**
- **Appointment.cs**

dao/
- **IHospitalService.cs**
- **HospitalServiceImpl.cs**

util/
- **DBPropertyUtil.cs**
- **DBConnUtil.cs**

exception/
- **PatientNumberNotFoundException.cs**

main/
- **MainModule.cs**

Code:

entity/

Patient.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace HospitalManagementSystem.entity
{
   public class Patient
   {
      public int PatientId { get; set; }
      public string FirstName { get; set; }
      public string LastName { get; set; }
      public DateTime DateOfBirth { get; set; }
      public string Gender { get; set; }
      public string ContactNumber { get; set; }
      public string Address { get; set; }

      public Patient() { }

      public Patient(int patientId, string firstName, string lastName, DateTime dob, string gender,
string contactNumber, string address)
      {
         PatientId = patientId;
```

```
            FirstName = firstName;
            LastName = lastName;
            DateOfBirth = dob;
            Gender = gender;
            ContactNumber = contactNumber;
            Address = address;
        }

        public override string ToString()
        {
            return $"PatientId: {PatientId}, Name: {FirstName} {LastName}, DOB:
{DateOfBirth.ToShortDateString()}, Gender: {Gender}, Contact: {ContactNumber}, Address:
{Address}";
        }
    }
}
```

Doctor.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace HospitalManagementSystem.entity
{
    public class Doctor
    {
        public int DoctorId { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Specialization { get; set; }
        public string ContactNumber { get; set; }

        public Doctor() { }

        public Doctor(int doctorId, string firstName, string lastName, string specialization, string
contactNumber)
        {
            DoctorId = doctorId;
            FirstName = firstName;
            LastName = lastName;
            Specialization = specialization;
            ContactNumber = contactNumber;
        }

        public override string ToString()
        {
```

```
        return $"DoctorId: {DoctorId}, Name: Dr. {FirstName} {LastName}, Specialization:
{Specialization}, Contact: {ContactNumber}";
      }
   }
}
```

Appointment.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace HospitalManagementSystem.entity
{
   public class Appointment
   {
      public int AppointmentId { get; set; }
      public int PatientId { get; set; }
      public int DoctorId { get; set; }
      public DateTime AppointmentDate { get; set; }
      public string Description { get; set; }

      public Appointment() { }

      public Appointment(int appointmentId, int patientId, int doctorId, DateTime appointmentDate,
string description)
      {
         AppointmentId = appointmentId;
         PatientId = patientId;
         DoctorId = doctorId;
         AppointmentDate = appointmentDate;
         Description = description;
      }

      public override string ToString()
      {
         return $"AppointmentId: {AppointmentId}, PatientId: {PatientId}, DoctorId: {DoctorId}, Date:
{AppointmentDate}, Description: {Description}";
      }
   }
}
```

dao/

IHospitalService.cs

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using HospitalManagementSystem.entity;

namespace HospitalManagementSystem.dao
{
   public interface IHospitalService
   {
      Appointment GetAppointmentById(int appointmentId);
      List<Appointment> GetAppointmentsForPatient(int patientId);
      List<Appointment> GetAppointmentsForDoctor(int doctorId);
      bool ScheduleAppointment(Appointment appointment);
      bool UpdateAppointment(Appointment appointment);
      bool CancelAppointment(int appointmentId);
   }
}
```

HospitalServiceImpl.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient;
using HospitalManagementSystem.entity;
using HospitalManagementSystem.util;

namespace HospitalManagementSystem.dao
{
   public class HospitalServiceImpl : IHospitalService
   {
      private readonly string propFile = "db.properties";

      private SqlConnection GetDbConnection()
      {
         return DBConnUtil.GetConnection(propFile);
      }

      public Appointment GetAppointmentById(int appointmentId)
      {
         Appointment appointment = null;
         using (SqlConnection conn = GetDbConnection())
         {
            if (conn == null) return null;

            string query = "SELECT * FROM Appointment WHERE AppointmentId = @AppointmentId";
            SqlCommand cmd = new SqlCommand(query, conn);
            cmd.Parameters.AddWithValue("@AppointmentId", appointmentId);
```

```
        using (SqlDataReader reader = cmd.ExecuteReader())
        {
            if (reader.Read())
            {
                appointment = new Appointment
                {
                    AppointmentId = reader.GetInt32(0),
                    PatientId = reader.GetInt32(1),
                    DoctorId = reader.GetInt32(2),
                    AppointmentDate = reader.GetDateTime(3),
                    Description = reader.GetString(4)
                };
            }
        }
    }
    return appointment;
}

public List<Appointment> GetAppointmentsForPatient(int patientId)
{
    List<Appointment> appointments = new List<Appointment>();
    using (SqlConnection conn = GetDbConnection())
    {
        if (conn == null) return appointments;

        string query = "SELECT * FROM Appointment WHERE PatientId = @PatientId";
        SqlCommand cmd = new SqlCommand(query, conn);
        cmd.Parameters.AddWithValue("@PatientId", patientId);

        using (SqlDataReader reader = cmd.ExecuteReader())
        {
            while (reader.Read())
            {
                appointments.Add(new Appointment
                {
                    AppointmentId = reader.GetInt32(0),
                    PatientId = reader.GetInt32(1),
                    DoctorId = reader.GetInt32(2),
                    AppointmentDate = reader.GetDateTime(3),
                    Description = reader.GetString(4)
                });
            }
        }
    }
    return appointments;
}

public List<Appointment> GetAppointmentsForDoctor(int doctorId)
{
    List<Appointment> appointments = new List<Appointment>();
```

```
        using (SqlConnection conn = GetDbConnection())
        {
            if (conn == null) return appointments;

            string query = "SELECT * FROM Appointment WHERE DoctorId = @DoctorId";
            SqlCommand cmd = new SqlCommand(query, conn);
            cmd.Parameters.AddWithValue("@DoctorId", doctorId);

            using (SqlDataReader reader = cmd.ExecuteReader())
            {
                while (reader.Read())
                {
                    appointments.Add(new Appointment
                    {
                        AppointmentId = reader.GetInt32(0),
                        PatientId = reader.GetInt32(1),
                        DoctorId = reader.GetInt32(2),
                        AppointmentDate = reader.GetDateTime(3),
                        Description = reader.GetString(4)
                    });
                }
            }
        }
        return appointments;
    }

    public bool ScheduleAppointment(Appointment appointment)
    {
        using (SqlConnection conn = GetDbConnection())
        {
            if (conn == null) return false;

            string query = @"INSERT INTO Appointment (PatientId, DoctorId, AppointmentDate,
Description)
                            VALUES (@PatientId, @DoctorId, @AppointmentDate, @Description)";
            SqlCommand cmd = new SqlCommand(query, conn);
            cmd.Parameters.AddWithValue("@PatientId", appointment.PatientId);
            cmd.Parameters.AddWithValue("@DoctorId", appointment.DoctorId);
            cmd.Parameters.AddWithValue("@AppointmentDate", appointment.AppointmentDate);
            cmd.Parameters.AddWithValue("@Description", appointment.Description);

            return cmd.ExecuteNonQuery() > 0;
        }
    }

    public bool UpdateAppointment(Appointment appointment)
    {
        using (SqlConnection conn = GetDbConnection())
        {
            if (conn == null) return false;
```

```
            string query = @"UPDATE Appointment
                        SET PatientId = @PatientId, DoctorId = @DoctorId,
                            AppointmentDate = @AppointmentDate, Description = @Description
                        WHERE AppointmentId = @AppointmentId";
            SqlCommand cmd = new SqlCommand(query, conn);
            cmd.Parameters.AddWithValue("@PatientId", appointment.PatientId);
            cmd.Parameters.AddWithValue("@DoctorId", appointment.DoctorId);
            cmd.Parameters.AddWithValue("@AppointmentDate", appointment.AppointmentDate);
            cmd.Parameters.AddWithValue("@Description", appointment.Description);
            cmd.Parameters.AddWithValue("@AppointmentId", appointment.AppointmentId);

            return cmd.ExecuteNonQuery() > 0;
        }
    }

    public bool CancelAppointment(int appointmentId)
    {
        using (SqlConnection conn = GetDbConnection())
        {
            if (conn == null) return false;

            string query = "DELETE FROM Appointment WHERE AppointmentId = @AppointmentId";
            SqlCommand cmd = new SqlCommand(query, conn);
            cmd.Parameters.AddWithValue("@AppointmentId", appointmentId);

            return cmd.ExecuteNonQuery() > 0;
        }
    }
  }
}
```

util/

DBPropertyUtil.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace HospitalManagementSystem.util
{
  public static class DBPropertyUtil
  {
    public static string GetConnectionString(string filePath)
    {
      Dictionary<string, string> config = new Dictionary<string, string>();
      try
      {
```

```
            foreach (var line in File.ReadAllLines(filePath))
            {
                if (line.Contains("="))
                {
                    var parts = line.Split('=');
                    config[parts[0].Trim()] = parts[1].Trim();
                }
            }

            return
$"Server={config["Server"]};Database={config["Database"]};Trusted_Connection={config["Trusted_Co
nnection"]};TrustServerCertificate={config["TrustServerCertificate"]};";
        }
        catch (Exception ex)
        {
            Console.WriteLine("Error reading properties file: " + ex.Message);
            return null;
        }
    }
  }
}
```

DBConnUtil.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient;

namespace HospitalManagementSystem.util
{
  public static class DBConnUtil
  {
    public static SqlConnection GetConnection(string propertyFilePath)
    {
      string connectionString = DBPropertyUtil.GetConnectionString(propertyFilePath);
      SqlConnection connection = new SqlConnection(connectionString);
      try
      {
        connection.Open();
        return connection;
      }
      catch (SqlException ex)
      {
        Console.WriteLine("Connection Error: " + ex.Message);
        return null;
      }
    }
```

```
    }
}
```

exception/

PatientNumberNotFoundException.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace HospitalManagementSystem.exception
{
    public class PatientNumberNotFoundException : Exception
    {
        public PatientNumberNotFoundException()
            : base("Patient number not found in the database.") { }

        public PatientNumberNotFoundException(string message)
            : base(message) { }

        public PatientNumberNotFoundException(string message, Exception inner)
            : base(message, inner) { }
    }
}
```

main/

MainModule.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using HospitalManagementSystem.dao;
using HospitalManagementSystem.entity;
using HospitalManagementSystem.exception;


namespace HospitalManagementSystem.main
{
    class MainModule
    {
        static void Main(string[] args)
        {
            IHospitalService service = new HospitalServiceImpl();

            while (true)
            {
                Console.WriteLine("\n===== Hospital Management System =====");
```

```
            Console.WriteLine("1. Schedule Appointment");
            Console.WriteLine("2. Update Appointment");
            Console.WriteLine("3. Cancel Appointment");
            Console.WriteLine("4. Get Appointment by ID");
            Console.WriteLine("5. Get Appointments for Patient");
            Console.WriteLine("6. Get Appointments for Doctor");
            Console.WriteLine("7. Exit");
            Console.Write("Enter your choice: ");

            string input = Console.ReadLine();
            int choice;
            if (!int.TryParse(input, out choice))
            {
               Console.WriteLine("Invalid input. Enter a number.");
               continue;
            }

            try
            {
               switch (choice)
               {
                  case 1:
                     Appointment newAppt = new Appointment();
                     Console.Write("Enter Patient ID: ");
                     newAppt.PatientId = int.Parse(Console.ReadLine());
                     Console.Write("Enter Doctor ID: ");
                     newAppt.DoctorId = int.Parse(Console.ReadLine());
                     Console.Write("Enter Appointment Date (yyyy-MM-dd HH:mm): ");
                     newAppt.AppointmentDate = DateTime.Parse(Console.ReadLine());
                     Console.Write("Enter Description: ");
                     newAppt.Description = Console.ReadLine();

                     bool added = service.ScheduleAppointment(newAppt);
                     Console.WriteLine(added ? "Appointment scheduled successfully." : "Failed to
schedule appointment.");
                     break;

                  case 2:
                     Appointment updateAppt = new Appointment();
                     Console.Write("Enter Appointment ID to update: ");
                     updateAppt.AppointmentId = int.Parse(Console.ReadLine());
                     Console.Write("Enter new Patient ID: ");
                     updateAppt.PatientId = int.Parse(Console.ReadLine());
                     Console.Write("Enter new Doctor ID: ");
                     updateAppt.DoctorId = int.Parse(Console.ReadLine());
                     Console.Write("Enter new Appointment Date (yyyy-MM-dd HH:mm): ");
                     updateAppt.AppointmentDate = DateTime.Parse(Console.ReadLine());
                     Console.Write("Enter new Description: ");
                     updateAppt.Description = Console.ReadLine();

                     bool updated = service.UpdateAppointment(updateAppt);
```

```
                    Console.WriteLine(updated ? "Appointment updated successfully." : "Update
failed.");
                    break;

                case 3:
                    Console.Write("Enter Appointment ID to cancel: ");
                    int cancelId = int.Parse(Console.ReadLine());
                    bool deleted = service.CancelAppointment(cancelId);
                    Console.WriteLine(deleted ? "Appointment cancelled." : "Cancellation failed.");
                    break;

                case 4:
                    Console.Write("Enter Appointment ID: ");
                    int apptId = int.Parse(Console.ReadLine());
                    Appointment foundAppt = service.GetAppointmentById(apptId);
                    if (foundAppt == null)
                        Console.WriteLine("Appointment not found.");
                    else
                        Console.WriteLine(foundAppt);
                    break;

                case 5:
                    Console.Write("Enter Patient ID: ");
                    int patId = int.Parse(Console.ReadLine());
                    List<Appointment> patientAppts = service.GetAppointmentsForPatient(patId);
                    if (patientAppts.Count == 0)
                        throw new PatientNumberNotFoundException($"No appointments found for
patient ID {patId}");
                    foreach (var appt in patientAppts)
                        Console.WriteLine(appt);
                    break;

                case 6:
                    Console.Write("Enter Doctor ID: ");
                    int docId = int.Parse(Console.ReadLine());
                    List<Appointment> doctorAppts = service.GetAppointmentsForDoctor(docId);
                    if (doctorAppts.Count == 0)
                        Console.WriteLine("No appointments found for this doctor.");
                    else
                        foreach (var appt in doctorAppts)
                            Console.WriteLine(appt);
                    break;

                case 7:
                    Console.WriteLine("Exiting...");
                    return;

                default:
                    Console.WriteLine("Invalid choice. Try again.");
                    break;
            }
```

```
            }
            catch (PatientNumberNotFoundException ex)
            {
                Console.WriteLine("Exception: " + ex.Message);
            }
            catch (FormatException)
            {
                Console.WriteLine("Invalid input format.");
            }
            catch (Exception ex)
            {
                Console.WriteLine("Unexpected error: " + ex.Message);
            }
        }
    }
  }
}
```

db.properties

```
Server=MAGS-LAPTOP\SQLEXPRESS
Database=HospitalDB
Trusted_Connection=True
TrustServerCertificate=True
```

Database Creation SQL:

```
CREATE DATABASE HospitalDB;
GO

USE HospitalDB;
GO

CREATE TABLE Patient (
    PatientId INT PRIMARY KEY IDENTITY(1,1),
    FirstName NVARCHAR(50) NOT NULL,
    LastName NVARCHAR(50) NOT NULL,
    DateOfBirth DATE NOT NULL,
    Gender NVARCHAR(10) NOT NULL,
    ContactNumber NVARCHAR(15) NOT NULL,
    Address NVARCHAR(200)
);

CREATE TABLE Doctor (
    DoctorId INT PRIMARY KEY IDENTITY(1,1),
    FirstName NVARCHAR(50) NOT NULL,
    LastName NVARCHAR(50) NOT NULL,
    Specialization NVARCHAR(100) NOT NULL,
    ContactNumber NVARCHAR(15) NOT NULL
```

```
);

CREATE TABLE Appointment (
   AppointmentId INT PRIMARY KEY IDENTITY(1,1),
   PatientId INT NOT NULL,
   DoctorId INT NOT NULL,
   AppointmentDate DATETIME NOT NULL,
   Description NVARCHAR(500),

   FOREIGN KEY (PatientId) REFERENCES Patient(PatientId),
   FOREIGN KEY (DoctorId) REFERENCES Doctor(DoctorId)
);

SELECT * FROM Patient;
SELECT * FROM Doctor;
SELECT * FROM Appointment;
```

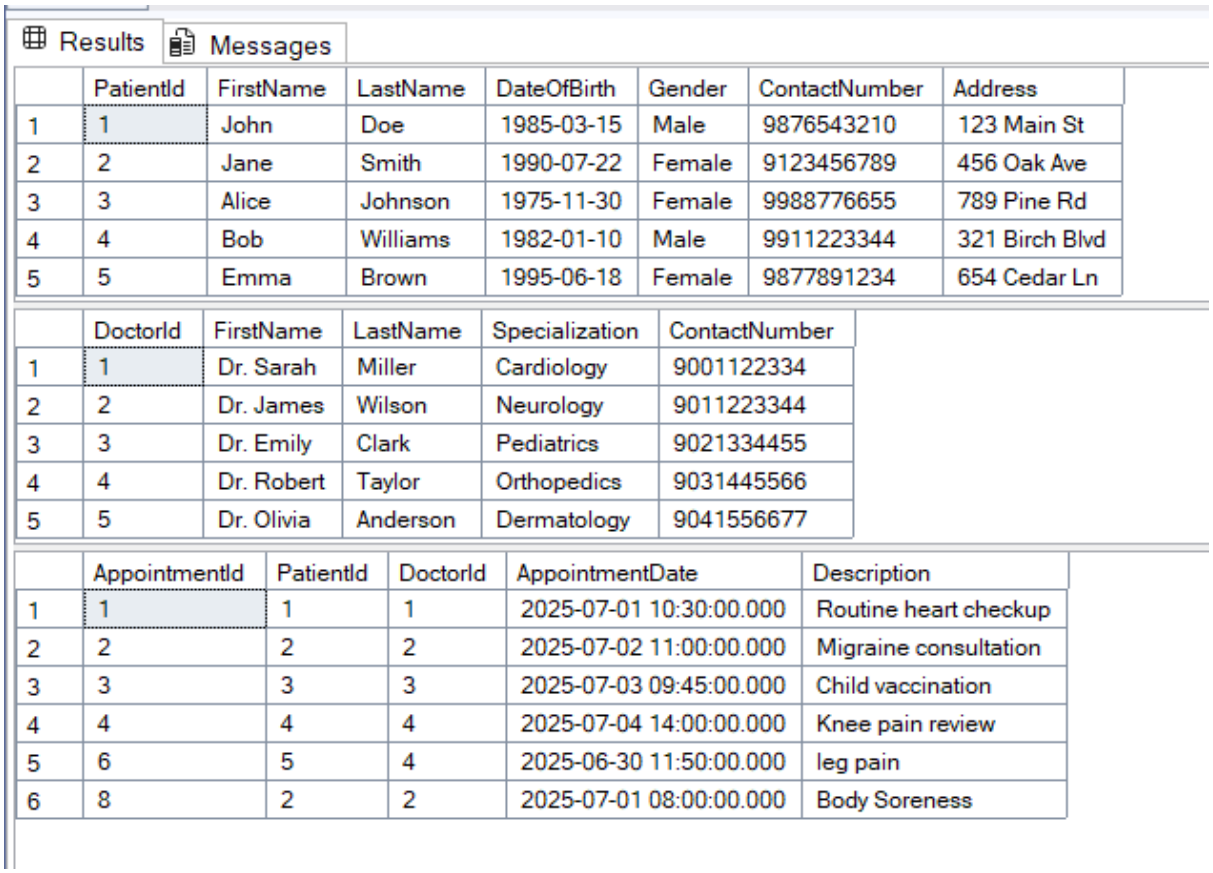
Output of C# (Console UI):

```
===== Hospital Management System =====
1. Schedule Appointment
2. Update Appointment
3. Cancel Appointment
4. Get Appointment by ID
5. Get Appointments for Patient
6. Get Appointments for Doctor
7. Exit
Enter your choice: 6
Enter Doctor ID: 2
AppointmentId: 2, PatientId: 2, DoctorId: 2, Date: 02-07-2025 11:00:00 AM, Description: Migraine consultation

===== Hospital Management System =====
1. Schedule Appointment
2. Update Appointment
3. Cancel Appointment
4. Get Appointment by ID
5. Get Appointments for Patient
6. Get Appointments for Doctor
7. Exit
Enter your choice: 1
Enter Patient ID: 2
Enter Doctor ID: 2
Enter Appointment Date (yyyy-MM-dd HH:mm): 2025-07-01 08:00
Enter Description: Body Soreness
Appointment scheduled successfully.

===== Hospital Management System =====
1. Schedule Appointment
2. Update Appointment
3. Cancel Appointment
4. Get Appointment by ID
5. Get Appointments for Patient
6. Get Appointments for Doctor
7. Exit
Enter your choice: 6
Enter Doctor ID: 2
AppointmentId: 2, PatientId: 2, DoctorId: 2, Date: 02-07-2025 11:00:00 AM, Description: Migraine consultation
AppointmentId: 8, PatientId: 2, DoctorId: 2, Date: 01-07-2025 08:00:00 AM, Description: Body Soreness
```

ALL THE FUNCTIONS WORK

TO CHECK THE REFLECTION ON DATABASE, SEE THE IMAGE BELOW

Output SQL Database:



Results | Messages

	PatientId	FirstName	LastName	DateOfBirth	Gender	ContactNumber	Address
1	1	John	Doe	1985-03-15	Male	9876543210	123 Main St
2	2	Jane	Smith	1990-07-22	Female	9123456789	456 Oak Ave
3	3	Alice	Johnson	1975-11-30	Female	9988776655	789 Pine Rd
4	4	Bob	Williams	1982-01-10	Male	9911223344	321 Birch Blvd
5	5	Emma	Brown	1995-06-18	Female	9877891234	654 Cedar Ln

	DoctorId	FirstName	LastName	Specialization	ContactNumber
1	1	Dr. Sarah	Miller	Cardiology	9001122334
2	2	Dr. James	Wilson	Neurology	9011223344
3	3	Dr. Emily	Clark	Pediatrics	9021334455
4	4	Dr. Robert	Taylor	Orthopedics	9031445566
5	5	Dr. Olivia	Anderson	Dermatology	9041556677

	AppointmentId	PatientId	DoctorId	AppointmentDate	Description
1	1	1	1	2025-07-01 10:30:00.000	Routine heart checkup
2	2	2	2	2025-07-02 11:00:00.000	Migraine consultation
3	3	3	3	2025-07-03 09:45:00.000	Child vaccination
4	4	4	4	2025-07-04 14:00:00.000	Knee pain review
5	6	5	4	2025-06-30 11:50:00.000	leg pain
6	8	2	2	2025-07-01 08:00:00.000	Body Soreness