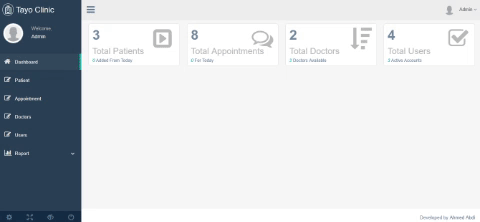
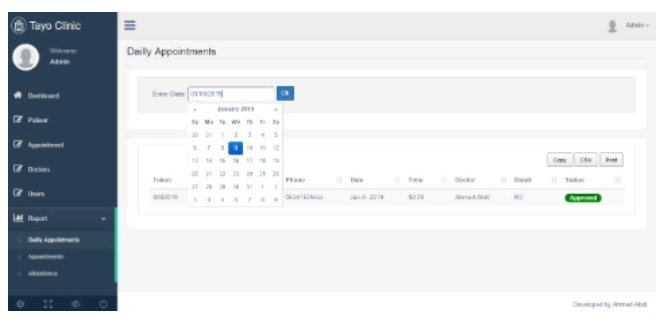
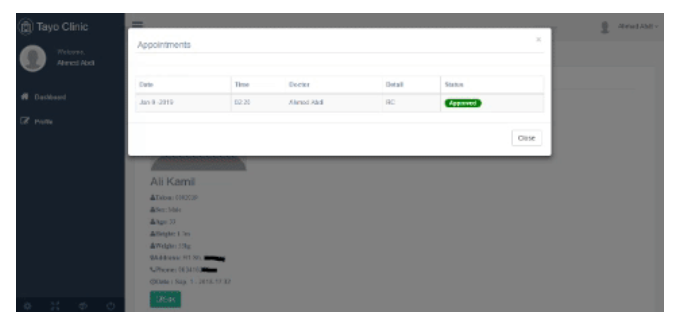
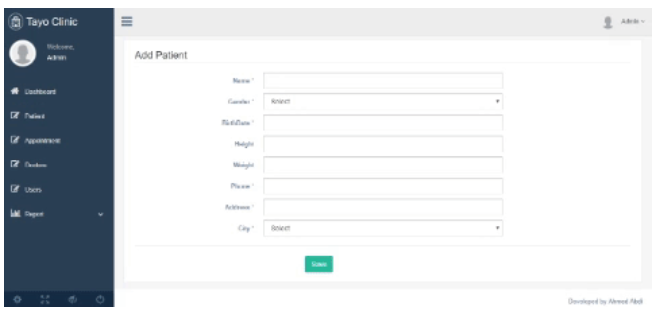
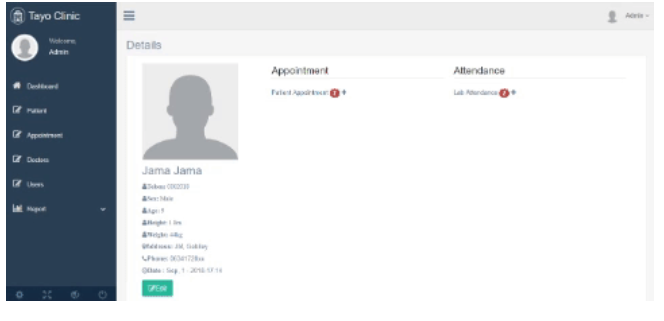
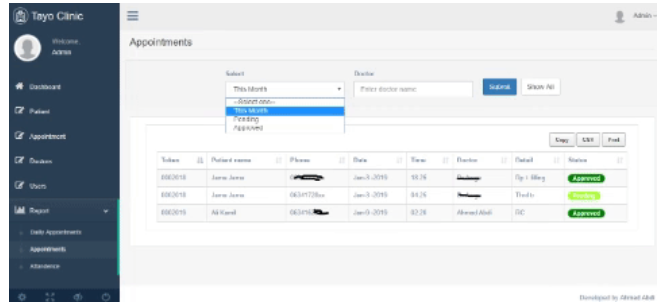
Develop the clinic web portal that allows patients to visit the clinic and get registered and make an appointment by selecting the populated available doctors, which in turn lists the upcoming appointments for the selected doctor. By default, the appointment gets a pending status because it needs to be reviewed. After that, the doctor is going to work out the patient attendance. Under the report, we should have daily and monthly appointments.









**Domain Entities**

Now let’s start looking at the domain entities that we had generated in the database using code first migration. The Patient entity has information such as tokens (auto-generated serial numbers), name, phone number, and age (derived from birthdate). A patient will have one or more appointments and/or visits, so we need to add a collection of appointments and visits.

public class Patient

{

public int Id { get; set; }

public string Token { get; set; }

public string Name { get; set; }

public Gender Sex { get; set; }

public DateTime BirthDate { get; set; }

public string Phone { get; set; }

public string Address { get; set; }

public byte CityId { get; set; }

public City Cities { get; set; }

public DateTime DateTime { get; set; }

public string Height { get; set; }

public string Weight { get; set; }

public int Age

{

get

{

var now = DateTime.Today;

var age = now.Year - BirthDate.Year;

if (BirthDate > now.AddYears(-age)) age--;

return age;

}

}

public ICollection<Appointment> Appointments { get; set; }

public ICollection<Attendance> Attendances { get; set; }

public Patient()

{

Appointments = new Collection<Appointment>();

Attendances = new Collection<Attendance>();

}

}

The appointment has date-time, details, and status (approved or pending).

public class Appointment

{

public int Id { get; set; }

public DateTime StartDateTime { get; set; }

public string Detail { get; set; }

public bool Status { get; set; }

public int PatientId { get; set; }

public Patient Patient { get; set; }

public int DoctorId { get; set; }

public Doctor Doctor { get; set; }

}

Each visit has information such as diagnosis, therapy, and clinic remarks. It has also a reference to a specific patient.

public class Attendance

{

public int Id { get; set; }

public string ClinicRemarks { get; set; }

public string Diagnosis { get; set; }

public string SecondDiagnosis { get; set; }

public string ThirdDiagnosis { get; set; }

public string Therapy { get; set; }

public DateTime Date { get; set; }

public int PatientId { get; set; }

public Patient Patient { get; set; }

}

Each doctor has one or more appointments and also has a reference to their specialization.

public class Doctor

{

public int Id { get; set; }

public string Name { get; set; }

public string Phone { get; set; }

public bool IsAvailable { get; set; }

public string Address { get; set; }

public int SpecializationId { get; set; }

public Specialization Specialization { get; set; }

public string PhysicianId { get; set; }

public ApplicationUser Physician { get; set; }

public ICollection<Appointment> Appointments { get; set; }

public Doctor()

{

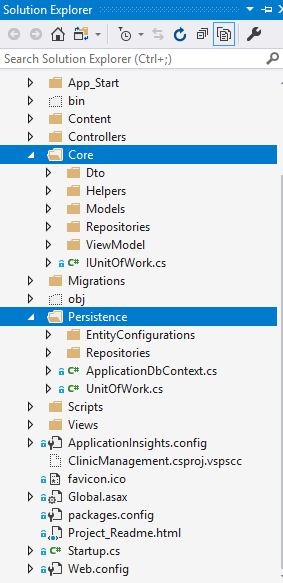
Appointments = new Collection<Appointment>();

}

}

### ****Application Structure****

The persistence ignorant parts are in the **Core folder** are domain entities, Dtos, view models, and interfaces. The **Persistence folder** contains entity configurations and the implementation of repository interfaces.



### ****Accounts****

There are two users; administrator and doctors.

Admin has full application access, including adding new patients and assign them to available doctors.

The doctor is being registered first by an administrator and has a view that displays the patients assigned to them.

Template URL:

<https://github.com/ColorlibHQ/gentelella>