```
***CourseRegistration

//This class is for reference only. Make change in
required place only
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Registration
{
    class Course
    {
        public int CourseId { get; set; }
        public String CourseName { get; set; }
        public int Duration { get; set; }
        public double Fees { get; set; }
        //Include a collection navigation property of
type ICollection<Student>

        public virtual ICollection<Student> Student
        {
            get;
            set;
        }
    }
}

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

namespace Registration   //DO NOT change the namespace
name
{
   public class Program      //DO NOT change the class
name
    {
       public static void Main(string[] args)   //DO
NOT change the 'Main' method signature
        {
            //Implement the code here

            Course course=new Course();
            Student student=new Student();
            SchoolContext context=new SchoolContext();
            SchoolUtil sutil=new SchoolUtil();

            Console.WriteLine("Enter Course ID");

course.CourseId=Convert.ToInt32(Console.ReadLine());
```

```
            Console.WriteLine("Enter Course Name");
            course.CourseName=Console.ReadLine();
            Console.WriteLine("Enter Duration");

course.Duration=Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter Fees");

course.Fees=Convert.ToDouble(Console.ReadLine());

            sutil.AddCourse(course);
            Console.WriteLine("Course Inserted
Successfully");

            Console.WriteLine("Enter Course Id");

course.CourseId=Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter Course Name");
            course.CourseName=Console.ReadLine();
            Console.WriteLine("Enter Duration");

course.Duration=Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter Fees");

course.Fees=Convert.ToDouble(Console.ReadLine());

            sutil.AddCourse(course);
            Console.WriteLine("Course Inserted
Successfully");

           // Console.WriteLine("Enter Student Id");
           //
student.StudentId=Convert.ToInt32(Console.ReadLine());

            Console.WriteLine("Enter Student Name");

student.StudentId=Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter Course Name");
            student.CourseName=Console.ReadLine();
            Console.WriteLine("Enter Date of Join");

student.DateOfJoin=Convert.ToDateTime(Console.ReadLine
());
            Console.WriteLine("Enter City");
            student.City=Console.ReadLine();
            sutil.AddStudent(student);
            Console.WriteLine("Student Inserted
Successfully");

          //  Console.WriteLine("Enter Student Id");
           //
student.StudentId=Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter Student Name");

student.StudentId=Convert.ToInt32(Console.ReadLine());
```

```
            Console.WriteLine("Enter Course Name");
            student.CourseName=Console.ReadLine();
            Console.WriteLine("Enter Date Of Join");

student.DateOfJoin=Convert.ToDateTime(Console.ReadLine
());
            Console.WriteLine("Enter City");
            student.City=Console.ReadLine();

            sutil.AddStudent(student);
            Console.WriteLine("Student Inserted
Successfully");
            Console.WriteLine("Retrive all Students
based on Course Name");
            Console.WriteLine("Enter Course Name");
            string cn=Console.ReadLine();
            var res=sutil.GetStudentByCourseName(cn);
            foreach(Student scn in res)
            {

Console.WriteLine("{0}",scn.StudentName.ToString());
            }




        }
    }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.Entity;

namespace Registration  //DO NOT change the namespace
name
{
    class SchoolContext: DbContext   //DO NOT change
the class name
    {
        //Implement property for 'Students' and
'Courses' with required declaration
        public virtual DbSet<Student>
Students{get;set;}
        public virtual DbSet<Course> Courses{get;set;}

        public SchoolContext() :
base("RegistrationContext")   //DO NOT change the
Context name
        {
```

```
        }
        protected override void
OnModelCreating(DbModelBuilder modelBuilder)
        {
           //Map Student entity to StudentDetail table

modelBuilder.Entity<Student>().ToTable("StudentDetail"
);
            //Map Course entity to CourseDetail table

modelBuilder.Entity<Course>().ToTable("CourseDetail");

            //Make 'CourseName' as Foreign key in
Student Entity


modelBuilder.Entity<Course>().HasKey(s=>s.CourseName).
Property(s=>s.CourseName).IsRequired();
            //configure one-to-many relationship as
mentioned in the problem statement

modelBuilder.Entity<Student>().HasRequired<Course>(c=>
c.Course).WithMany(s=>s.Student).HasForeignKey(c=>c.Co
urseName).WillCascadeOnDelete();

        }
}
}


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

namespace Registration      //DO NOT change the
namespace name
{
   class SchoolUtil    //DO NOT change the class name
    {
        SchoolContext context;
        public Student AddStudent(Student student)
//DO NOT change method signature
        {
            //Implement the code here

            using(context=new SchoolContext())
            {
                context.Students.Add(student);
                context.SaveChanges();
                return student;
```

```
            }
        }

        public Course AddCourse(Course course)   //DO
NOT change method signature
        {
            //Implement the code here

            using(context=new SchoolContext())
            {
                context.Courses.Add(course);
                context.SaveChanges();
                return course;
            }

        }

        public List<Student>
GetStudentByCourseName(string courseName)       //DO
NOT change method signature
        {
            //Implement the code here
            using(context=new SchoolContext())
            {
            var studentlist
=context.Students.Where(s=>s.CourseName.Equals(courseN
ame,StringComparison.InvariantCultureIgnoreCase)).ToLi
st();
            return studentlist;
            }

        }
    }
}


//This class is for reference only. Make change in
required place only
using System.ComponentModel.DataAnnotations.Schema;
using System;
using System.Collections.Generic;

namespace Registration
{
    class Student
    {
        public int StudentId { get; set; }
        public String StudentName { get; set; }
        public DateTime DateOfJoin { get; set; }
        public String City { get; set; }
        // Add 2 properties
        //1. Include a reference navigation property
of Course type
        //2. foreign key property of CourseName
```

```
     public virtual Course Course{get;set;}
     public virtual String CourseName{get;set;}

    }
}
```

***EZYGO

```
//THIS IS FOR REFERENCE ONLY. YOU ARE NOT REQUIRED TO
MAKE ANY CHANGES HERE
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace EZYGORepository
{
    interface IProductRepository
    {
        Product GetProductById(int id);
        Product UpdateStatus(int productId, int
status);
        Product ProductCheckIn(Product p);
        int TotalUnitsOfProduct(string productName);
    }
}


//THIS IS FOR REFERENCE ONLY. YOU ARE NOT REQUIRED TO
MAKE ANY CHANGES HERE
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;

namespace EZYGORepository
{
    public class Product
    {
        public int ProductId { get; set; }
        public String ProductName { get; set; }
        public String ProductType { get; set; }
        public int Status { get; set; }
        public int NoOfUnits { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
```

```
namespace EZYGORepository   //DO NOT change the
namespace name
{
    public class ProductContext : DbContext      //DO
NOT change the class name
    {
        public ProductContext() : base("ProductDB"){
}     //DO NOT change the Context name

        //Implement property for 'Products' if type
DbSet with required declaration
        public virtual DbSet<Product> Products{ get;
set; }
    }
}

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

namespace EZYGORepository        //DO NOT change the
namespace name
{
    class ProductRepository : IProductRepository
//DO NOT change the class name. Implement the
interface
    {
       //Add the required methods
       ProductContext context=new ProductContext();
       public Product GetProductById(int id){
           return context.Products.Find(id);
       }

       public Product UpdateStatus(int productId,int
status){
           Product
product=context.Products.Find(productId);
           product.Status=status;
           context.SaveChanges();
           return product;
       }

       public Product ProductCheckIn(Product p){

           int units=0;
           var productList=context.Products.Where(x =>
x.ProductName == p.ProductName);
           foreach(var pr in productList){
               units += pr.NoOfUnits;
           }
```

```
            if(units + p.NoOfUnits < 100000){

                p.Status=1;
                context.Products.Add(p);
                context.SaveChanges();
                Console.WriteLine("Product added
successfully.");
                return p;
            }
            else{
                Console.WriteLine("Product NOT added as
it exceeds maximum units 100000.");
                return null;
            }
        }
        public int TotalUnitsOfProduct(string
productname){
            int units=0;
            var product=context.Products.Where(x =>
x.ProductName == productname);
            foreach(var pr in product){
                units += pr.NoOfUnits;
            }
            return units;
        }

    }
}


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

namespace EZYGORepository         //DO NO Change the
namespace name
{
    public class Program      //DO NO Change the class
name
    {
        static void Main(string[] args)           //DO
NO change the method signature
        {
            //Implement the code here
            Product pObj=new Product();
            ProductRepository prObj=new
ProductRepository();

            Console.WriteLine("Enter Product Name:");
            pObj.ProductName=Console.ReadLine();
            Console.WriteLine("Enter Product Type:");
            pObj.ProductType=Console.ReadLine();
```

```
            Console.WriteLine("Enter Number of
units:");

pObj.NoOfUnits=Convert.ToInt32(Console.ReadLine());
            pObj=prObj.ProductCheckIn(pObj);

            Console.WriteLine("Enter product id:");

pObj.ProductId=Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter product
status:");

pObj.Status=Convert.ToInt32(Console.ReadLine());

pObj=prObj.UpdateStatus(pObj.ProductId,pObj.Status);

            Console.WriteLine("Enter product id:");

pObj.ProductId=Convert.ToInt32(Console.ReadLine());
            pObj=prObj.GetProductById(pObj.ProductId);

Console.WriteLine("ProductId\tProductName\tStatus\tNoO
fUnits");

Console.WriteLine("{0}\t{1}\t{2}\t{3},pObj.ProductId,p
Obj.ProductName,pObj.Status,pObj.NoOfUnits");

            Console.WriteLine("Enter product name");
            pObj.ProductName=Console.ReadLine();

pObj.NoOfUnits=prObj.TotalUnitsOfProduct(pObj.ProductN
ame);
            Console.WriteLine("Total units:
{0}",pObj.NoOfUnits);

        }
    }
}
```

***FoodDelivery

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace FoodDeliveryApp                    //Do not
change the namespace name
{
/* Use Data Annotations for the below work */
```

```
    //Add Agent_Tb as table name
    [Table("Agent_Tb")]
    public class AgentDomain              //Do not
change the class name
    {
        //Make Id as Primary Key
        [Key]
        public int Id { get; set; }

        //Agent name is required
        [Required(AllowEmptyStrings=false)]
        public string Agent_Name { get; set; }
        public Int64 Mobile_Number { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FoodDeliveryApp          //Do not change the
namespace name
{
/* Use Data Annotations for the below work */

    //Add Company_Tb as table name
    [Table("Company_Tb")]
    public class DeliveryCompanyDomain         //Do
not change the class name
    {
        //Make Company_Registration_Code as primary
key
        [Key]
        public int Company_Registration_Code { get;
set; }

      //Company name is required [manditory]
          [Required(AllowEmptyStrings = false)]
        public string Company_Name { get; set; }
        public DateTime Registration_Date { get; set;
}
    }
}

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

```
namespace FoodDeliveryApp                //Do not
change the namespace name
{
    public class DeliveryContext:DbContext
//Do not change the class name
    {
      //Do NOT change the context name
'DeliveryContext'
        public
DeliveryContext():base("name=DeliveryDbString")
        {

        }

        public virtual DbSet<DeliveryCompanyDomain>
Companies {get; set;}
        public virtual DbSet<AgentDomain> Agents {get;
set;}
         //Implement property for 'Companies' and
'Agents' with required 'DbSet' declaration
    }
}

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

namespace FoodDeliveryApp          //Do not change the
namespace name
{
    public class DeliveryRepository              //Do
not change the class name
    {
        /*private DeliveryContext context;

        public DeliveryRepository(DeliveryContext
context)
        {
            //fill code here
            this.context=context;
        }*/

        public int AddCompany(DeliveryCompanyDomain
company)
        {
            //fill code here
            var _context  = new DeliveryContext();
            _context.Companies.Add(company);
            if(_context.SaveChanges()>0){
                return 1;
```

```
            }
            else
            {
                return 0;
            }
        }

        public int AddAgent(AgentDomain agent)
        {
           //fill code here
           /*var _context = new DeliveryContext();
           _context.Agents.Add(agent);
           if(_context.SaveChanges()>0){
               return 1;
           }
           else{
               return 0;
           }*/
           using(var context = new DeliveryContext())
           {
               context.Agents.Add(agent);
               context.SaveChanges();
               return 1;
           }

        }

        public IList<DeliveryCompanyDomain>
DisplayAllCompanies()
        {
            //fill code here
            var _context = new DeliveryContext();
            IList<DeliveryCompanyDomain> all =
_context.Companies.ToList();
            return all;
        }

        public IList<AgentDomain> DisplayAllAgents()
            {
            //fill code here
            var _context = new DeliveryContext();
            IList<AgentDomain> all =
_context.Agents.ToList();
            return all;
            }
    }
}

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

```
namespace FoodDeliveryApp        //Do not change the
namespace name
{
    class Program               //Do not change the
class name
    {
        static void Main(string[] args)
        {
            //fill code here
            DeliveryRepository dRepo = new
DeliveryRepository();
            Console.WriteLine("Enter Company Name and
Registration Date information separated by comma
(,):");
            string resp = Console.ReadLine();

            String[] respArr;
            respArr = resp.Split(",");
            DeliveryCompanyDomain company = new
DeliveryCompanyDomain();
            company.Company_Name = respArr[0];
            company.Registration_Date =
DateTime.Parse(respArr[1]);
            dRepo.AddCompany(company);
        }
    }
}
```

***Hospital mgmt

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace HospitalManagementSystem
{
    //Fill your code here
    [Table("tblDoctor")]

    public class Doctor {
        [Key,
DatabaseGenerated(DatabaseGeneratedOption.None)]
        public int DoctorId {get; set;}
        public string DoctorName {get; set;}
        public string Specialization {get; set;}
        public virtual ICollection<DoctorPatient>
DoctorPatient {get; set;}
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace HospitalManagementSystem
{
    //Fill your code here
    [Table("tblDoctorPatient")]
    public class DoctorPatient {
        [Key,Column(Order = 1),
DatabaseGenerated(DatabaseGeneratedOption.None)]
        public int PatientId {get; set;}
        [Key,Column(Order = 2),
DatabaseGenerated(DatabaseGeneratedOption.None)]
        public int DoctorId {get; set;}
        public virtual Patient PatientInformation
{get;set;}
        public virtual Doctor DoctorInformation
{get;set;}
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.Entity;

namespace HospitalManagementSystem
{
    public class HospitalManagementContext:DbContext
    {
//Do NOT change the context name
HospitalManagementContext or DataConnection name
        public
HospitalManagementContext():base("Name=DataConnection"
)
        {

        }
        public virtual DbSet<Patient> Patients {get;
set;}
        public virtual DbSet<Doctor> Doctors {get;
set;}
        public virtual DbSet<DoctorPatient>
DoctorsPatients {get; set;}
```

```
//Implement property for 'Patients' with required
'DbSet' declaration
//Implement property for 'Doctors' with required
'DbSet' declaration
//Implement property for 'DoctorsPatients' with
required 'DbSet' declaration

    }
}

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

namespace HospitalManagementSystem
{
    public class HospitalManagementRepository
    {
        HospitalManagementContext
hospitalManagementContext;


//Fill your code here to implement RegisterDoctor
Method
        public int RegisterDoctor(IList<Doctor>
doctorList)
        {
            int res = 0;
            using(hospitalManagementContext = new
HospitalManagementContext()) {
                foreach(var doctor in doctorList) {

hospitalManagementContext.Doctors.Add(doctor);
                }
                res =
hospitalManagementContext.SaveChanges();
            }
            return res;
        }


        //Fill your code here to implement
RegisterPatient Method
        public int RegisterPatient(IList<Patient>
patientList)
        {
            int res = 0;
            using(hospitalManagementContext = new
HospitalManagementContext()) {
                foreach(var patient in patientList) {

hospitalManagementContext.Patients.Add(patient);
```

```
                }
                res =
hospitalManagementContext.SaveChanges();
            }
            return res;
        }

//Fill your code here to implement
RegisterPatientDoctorVisit Method

        public int
RegisterPatientDoctorVisit(DoctorPatient
doctorPatient)
        {
            int res = 0;
            using(hospitalManagementContext = new
HospitalManagementContext()) {

hospitalManagementContext.DoctorsPatients.Add(doctorPa
tient);
                res =
hospitalManagementContext.SaveChanges();
            }
            return res;
        }
//Fill your code here to implement DisplayPatients
Method
        public IList<Patient> DisplayPatients()
        {
            IList<Patient> res = new List<Patient>();
            using(hospitalManagementContext = new
HospitalManagementContext()) {
                res =
hospitalManagementContext.Patients.ToList<Patient>();
            }
            return res;

        }

//Fill your code here to implement DisplayDoctors
Method
        public IList<Doctor> DisplayDoctors()
        {
            IList<Doctor> res = new List<Doctor>();
            using(hospitalManagementContext = new
HospitalManagementContext()) {
                res =
hospitalManagementContext.Doctors.ToList<Doctor>();
            }
            return res;
        }

    public IList<DoctorPatient>
DisplayPatientDoctorVisit()
```

```
    {
        IList<DoctorPatient> res = new
List<DoctorPatient>();
            using(hospitalManagementContext = new
HospitalManagementContext()) {
                res =
hospitalManagementContext.DoctorsPatients.ToList<Docto
rPatient>();
            }
            return res;

    }
//Fill your code here to implement
DisplayPatientDoctorVisit Method

    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace HospitalManagementSystem
{
    //Fill your code here
    [Table("tblPatient")]
    public class Patient {
        [Key,
DatabaseGenerated(DatabaseGeneratedOption.None)]
        public int PatientId {get;set;}
        public string PatientName {get; set;}
        public long PatientMobileNumber {get; set;}
        public virtual ICollection<DoctorPatient>
DoctorPatient {get;set;}
    }
}

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

namespace HospitalManagementSystem
{
    class Program
    {
        static void Main(string[] args)
        {
```

```
            HospitalManagementRepository
hospitalManagementRepository = new
HospitalManagementRepository();
            int menu = 0;
            string menuRepeatLoopInput = string.Empty;

            do
            {
                Console.WriteLine("Menu:\nEnter 1 to
Insert new Doctors and Patients Information to
database\n" +
               "Enter 2 to Display all Doctors and
Patients information present in the database\n" +
               "Enter 3 to Insert Mapped Doctor-
Patient details to database\n" +
               "Enter 4 to Display all mapped Doctor-
Patient Information from the datbase");

                Console.WriteLine("Enter your menu
choice:");
                menu = int.Parse(Console.ReadLine());
                switch (menu)
                {
                    case 1:
                        try
                        {
                            IList<Doctor> doctorInfo =
new List<Doctor>
                 {
                new
Doctor{DoctorId=1001,DoctorName="Sam",Specialization="
Orthopedic"},
                new
Doctor{DoctorId=1002,DoctorName="Joe",Specialization="
Medicine"},
                new
Doctor{DoctorId=1003,DoctorName="Merry",Specialization
="Surgery"}
                 };
                            IList<Doctor>
CheckExistingDoctorInfo = new List<Doctor>();

                            if
(CheckExistingDoctorInfo.Count == 0)
                            {
                                if
(hospitalManagementRepository.RegisterDoctor(doctorInf
o) > 0)

Console.WriteLine("Doctor Information Added
Successfully");
                            }
```

```
                            IList<Patient> patientInfo
= new List<Patient>
            {
                new
Patient{PatientId=1,PatientName="Venkat",PatientMobile
Number=9600197755},
                new
Patient{PatientId=2,PatientName="Nivetha",PatientMobil
eNumber=9600197756},
                new
Patient{PatientId=3,PatientName="Vishnupriya",PatientM
obileNumber=9600197757}
            };
                            IList<Patient>
CheckExisitingPatientInfo = new List<Patient>();
                            if
(CheckExisitingPatientInfo.Count == 0)
                            {
                                if
(hospitalManagementRepository.RegisterPatient(patientI
nfo) > 0)

Console.WriteLine("Patient Information Added
Successfully");
                            }
                        }
                        catch
                        {

Console.WriteLine("Insertion Failed.Records already
exist");
                        }
                        break;

                    case 2:
                        Console.WriteLine("Display
Doctor List:");
                        Console.WriteLine("{0,-20}{1,-
20}{2}",
                            "Doctor Id", "Doctor
Name", "Specialization");
                        IList<Doctor> doctorList =
hospitalManagementRepository.DisplayDoctors();
                        foreach (var doctor in
doctorList)
                        {
                            Console.WriteLine("{0,-
20}{1,-20}{2}", doctor.DoctorId,
                                doctor.DoctorName,
doctor.Specialization);
                        }

                        Console.WriteLine("\nDisplay
Patient List:");
```

```
                        Console.WriteLine("{0,-20}{1,-
20}{2}", "Patient Id", "Patient Name", "Mobile
Number");
                        IList<Patient> patientList =
hospitalManagementRepository.DisplayPatients();

                        foreach (var patient in
patientList)
                        {
                            Console.WriteLine("{0,-
20}{1,-20}{2}", patient.PatientId,
patient.PatientName,

patient.PatientMobileNumber);
                        }
                        break;

                    case 3:
                        DoctorPatient doctorPatient =
new DoctorPatient();
                        string loopInput =
string.Empty;
                        do
                        {
                            try
                            {

Console.WriteLine("\nPatient Doctor Visit Registry:");

Console.WriteLine("Enter Patient Id:");

doctorPatient.PatientId =
int.Parse(Console.ReadLine());

Console.WriteLine("Enter Doctor Id:");
                                doctorPatient.DoctorId
= int.Parse(Console.ReadLine());

                                if
(hospitalManagementRepository.RegisterPatientDoctorVis
it(doctorPatient) > 0)

Console.WriteLine("Patient Doctor Visit Successfully
Registered");
                            }
                            catch
                            {

Console.WriteLine("Invalid Patient Id or Doctor
Id.Outdoor Visit Registration Failed");
                            }
                            Console.WriteLine("Enter
YES to continue..Any other key to terminate:");
```

```
                            loopInput =
Console.ReadLine();

                        }
                        while (loopInput.Equals("yes",
StringComparison.InvariantCultureIgnoreCase));
                        break;

                    case 4:
                        Console.WriteLine("\nDisplay
Patient Doctor Visit List:");
                        IList<DoctorPatient>
doctorPatientList =
hospitalManagementRepository.DisplayPatientDoctorVisit
();
                        Console.WriteLine("{0,-20}{1,-
20}{2,-20}{3}", "Patient Name", "Mobile Number",
                            "Doctor Name",
"Specialization");
                        foreach (var doctorPatientInfo
in doctorPatientList)
                        {
                            Console.WriteLine("{0,-
20}{1,-20}{2,-20}{3}",

doctorPatientInfo.PatientInformation.PatientName,

doctorPatientInfo.PatientInformation.PatientMobileNumb
er,

doctorPatientInfo.DoctorInformation.DoctorName,

doctorPatientInfo.DoctorInformation.Specialization);
                        }
                        break;
                }
                Console.WriteLine("Press Yes to repeat
menu..Any other key to terminate");
                menuRepeatLoopInput =
Console.ReadLine();
            }
            while (menuRepeatLoopInput.Equals("yes",
StringComparison.InvariantCultureIgnoreCase));

            Console.WriteLine("\nThank you. Have a
nice day");
        }
    }
}


***MakeMobile
//THIS IS FOR REFERENCE ONLY. YOU ARE NOT REQUIRED TO
MAKE ANY CHANGES HERE
```

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

namespace MobileProject //Do Not change the namespace
name
{
    public interface IMobileRepository //Do Not change
the interface Name
    {
        List<Mobile> GetAllMobiles();
        Mobile UpdateMobileDetails(Mobile mobile);
        Mobile AddMobileDetails(Mobile mobile);
        Boolean CheckAvailability(string mobileModel,
int ram, int memory);
    }
}

//THIS IS FOR REFERENCE ONLY. YOU ARE NOT REQUIRED TO
MAKE ANY CHANGES HERE
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MobileProject
{
    public class Mobile //DO NOT Change the class name
    {
        public int MobileId { get; set; }
        public String MobileName { get; set; }
        public String MobileModel { get; set; }
        public int Ram { get; set; }
        public int Memory { get; set; }
        public Double Cost { get; set; }

    }
}


using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MobileProject //DO NOT change the namespace
name
{
```

```
    public class MobileContext : DbContext  //DO NOT
change the class name
    {

        //DO NOT change the context name
         public MobileContext() : base("MobileDB")
         {}
          //Implement property for Mobiles of type
DbSet
          public virtual DbSet<Mobile> Mobiles {get;
set;}

    }
}

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

namespace MobileProject // Do NOT Change the namespace
{
    public class MobileRepository:IMobileRepository
//Do Not change the class name. Implement the
interface
    {
        //Implemen all the required methods
        public List<Mobile>GetAllMobiles()
        {
            using(var context=new MobileContext())
            {
                return context.Mobiles.ToList();
            }
        }
        public Boolean CheckAvailability(string
mobileModel,int ram,int memory)
        {
            bool res=false;
            Mobile mo=new Mobile();
            using(var context=new MobileContext())
            {
                mo =
context.Mobiles.Where(x=>x.MobileModel.Equals(mobileMo
del,StringComparison.InvariantCultureIgnoreCase)&&x.Ra
m==ram&&x.Memory==memory).FirstOrDefault();
                if(mo!=null)
                {
                    res=true;
                    return res;
                }
                else{
                    return res;
                }
```

```
            }
        }
        public Mobile AddMobileDetails(Mobile mobile)
        {
            using(var context=new MobileContext())
            {

if(CheckAvailability(mobile.MobileModel,mobile.Ram,mob
ile.Memory)==false)
                {
                    context.Mobiles.Add(mobile);
                    context.SaveChanges();
                }
                else{
                    Mobile mo=new Mobile();
                    mo=UpdateMobileDetails(mobile);
                    return mo;
                }
                return mobile;
            }
        }
        public Mobile UpdateMobileDetails(Mobile
mobile)
        {
            Mobile mo=new Mobile();
            using(var context=new MobileContext())
            {

mo=context.Mobiles.Where(x=>x.MobileModel.Equals(mobil
e.MobileModel,StringComparison.InvariantCultureIgnoreC
ase)&&x.Ram==mobile.Ram&&x.Memory==mobile.Memory).Firs
tOrDefault();
                mo.MobileName=mobile.MobileName;
                mo.Cost=mobile.Cost;
                context.SaveChanges();
                Console.WriteLine("Mobile Details
Updated successfully.");
                return mo;
            }
        }
    }
}

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

namespace MobileProject   //DO NOT change the
namespace name
{
    public class Program   //DO NOT change the class
name
```

```
    {
        static void Main(string[] args)      //DO NOT
change the 'Main' method signature
        {
            //Implement code here
        }
    }
}

***ManageStaff
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace StudentProject  //DO NOT change the
namespace name
{
    class Program        //DO NOT change the class
name
    {
        static void Main(string[] args)
        {
            //Fill code here


        }
    }
}

using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;

namespace StudentProject     //DO NOT change the
namespace name
{
    public class StaffContext : DbContext      //DO
NOT change the class name
    {

        public StaffContext():
base("name=StaffContext"){}


        //Implement virtual property for 'Staffs' and
'Departments' with required 'DbSet' declaration

        public DbSet<Staff> Staffs{get; set;}
        public DbSet<Department>Departments{get; set;}
```

```
        protected override void
OnModelCreating(DbModelBuilder modelBuilder)
        {
            //Map Staff to StaffDetail table


modelBuilder.Entity<Staff>().ToTable("StaffDetail");

            //Map Department to DepartmentDetail table


modelBuilder.Entity<Department>().ToTable("DepartmentD
etail");

            //Make 'DeptName' as primary key in
Department Entity


modelBuilder.Entity<Department>().HasKey(t=>t.DeptName
);

            // configures one-to-many relationship as
mentioned in the problem statement

modelBuilder.Entity<Department>().HasMany<Staff>(g=>g.
StaffList).WithRequired(s=>s.Department).HasForeignKey
(s=>s.DeptName);
        }
    }

    public class Staff  //DO NOT change the class name
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public int Experience { get; set; }
        public double Salary { get; set; }
        // Add 2 properties
        //1. Include a reference navigation property
of Department type

        public Department Department{get; set;}

        //2. foreign key property of DeptName

        public string DeptName{get; set;}

    }

    public class Department     //DO NOT change the
class name
    {
        public int Code { get; set; }
        public string DeptName { get; set; }
```

```
        //Include a collection navigation property of
type ICollection<Staff>
        public ICollection<Staff>StaffList{get; set;}
    }
}

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

namespace StudentProject   //DO NOT change the
namespace name
{
    class StaffUtility      //DO NOT change the class
name
    {
        StaffContext sc = new StaffContext();
        public Staff AddStaff(Staff staff)      //DO
NOT change the method signature
        {
            //fill code here
            var s = sc.Staffs.Add(staff);
            sc.SaveChanges();
            return s;
        }

        public Department AddDepartment(Department
department) //DO NOT change the method signature
        {
            //fill code here
            var d = sc.Departments.Add(department);
            sc.SaveChanges();
            return d;
        }

        public Staff GetStaffById(int Id)  //DO NOT
change the method signature
        {
           //fill code here
           var s1 =
sc.Staffs.FirstOrDefault(s=>s.Id==Id);
           return s1;
        }

        public List<Staff> GetStaffsList(string
deptName)   //DO NOT change the method signature
        {
            //fill code here
            var s2 =
sc.Staffs.Where(a=>a.DeptName==deptName).Select(b=>b);
            sc.SaveChanges();
            return s2.ToList();
```

```
        }
    }
}

***Shipment

//This class is for your Reference dont make any
changes to the class below
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Shipment
{
    public class Container
    {

        public int ContainerId { get; set; }
        public String ContainerName { get; set; }
        public int ContainerWeight { get; set; }
        public double CostOfShipment { get; set; }
        public double HoursToReach { get; set; }

        // Add 2 properties
        //1. Include a reference navigation property
of Ship type
        //2. foreign key property of ShipId

        public int ShipId { get; set; }
        public Ship Ship { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.Entity;

namespace Shipment //Do not change the namespace name
{
    public class ContainerUtil //Do not change the
class name
    {
        public Container AddContainer(Container con)
//Don't change the method signature
        {
            //Implement code here
            ShipmentContext context = new
ShipmentContext();
```

```
            DbSet<Container> cont =
context.Containers;
            cont.Add(con);
          if (context.SaveChanges()>0)
          {
              return con;
          }
          else
          {
              return null;
          }
        }
        public List<Container>
GetContainerByShipName(string shipname) //Don't change
the method signature
        {
           //Implement code here
           ShipmentContext context = new
ShipmentContext();
            DbSet<Container> cont = context.Containers;
            return cont.Where(c =>
c.Ship.ShipName.Equals(shipname)).AsQueryable().ToList
();




        }


    }
}


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

namespace Shipment //Do not change the namespace name
{
    public class Program //Don't change the class name
    {
        static void Main(string[] args) //Don't change
the method signature
        {
           //You Implementation goes here
           Ship ship_obj = new Ship();
           Console.WriteLine("Enter Ship Details:");
           Console.WriteLine("Enter Ship Name:");
           ship_obj.ShipName = Console.ReadLine();
           Console.WriteLine("Enter Destination
Longitude:");
```

```
ship_obj.DestLongitude=Convert.ToDouble(Console.ReadLi
ne());
           Console.WriteLine("Enter Destination
Latitude:");

ship_obj.DestLatitude=Convert.ToDouble(Console.ReadLin
e());
           Console.WriteLine("Enter current
Longitude:");

ship_obj.currentLongitude=Convert.ToDouble(Console.Rea
dLine());
           Console.WriteLine("Enter current
Latitude:");

ship_obj.currentLatitude=Convert.ToDouble(Console.Read
Line());
           Console.WriteLine("Enter speed:");
           ship_obj.ShipSpeed =
Convert.ToInt32(Console.ReadLine());

           ShipmentUtil su = new ShipmentUtil();
           su.AddShip(s);
           Console.WriteLine("Ship Details Added.");

           Container container_obj =new Container();
           Console.WriteLine("Enter Container Name:");
           container_obj.ContainerName =
Console.ReadLine();
            Console.WriteLine("Enter Ship id:");
           container_obj.ShipId =
Convert.ToInt32(Console.ReadLine());
           Console.WriteLine("Enter Container
Weight:");
           container_obj.ContainerWeight =
Convert.ToInt32(Console.ReadLine());
           Console.WriteLine("Enter Shipment Cost:");
           container_obj.CostOfShipment =
Convert.ToInt32(Console.ReadLine());
           Console.WriteLine("Enter Hours to Reach:");
           container_obj.HoursToReach =
Convert.ToInt32(Console.ReadLine());

           CountainerUtil cu = new CountainerUtil();
           cu.AddContainer(c);
            Console.WriteLine("Container Details
Added.");

            Console.WriteLine("Enter Ship Name to
retrieve Container details");
            string s1 = Console.ReadLine();
            List<Container> l1 =
cu.GetContainerByShipName(s1);
```

```
            Console.WriteLine("Container Name:");
            foreach (Container c1 in l1)
            {
                Console.WriteLine(c1.ContainerName);
            }
            Console.ReadLine();









        }
    }
}

//This class is for your Reference dont make any
changes to the class below
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Shipment
{
    public class Ship
    {

        public int ShipId { get; set; }
        public String ShipName { get; set; }
        public double DestLongitude { get; set; }
        public double DestLatitude { get; set; }
        public double CurrentLongitude { get; set; }
        public double CurrentLatitude { get; set; }
        public int ShipSpeed { get; set; }
        //Include a collection navigation property of
type ICollection<Container>
        public IList<Container> ContainerList { get;
set;  }
    }
}


using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Shipment //Do not change the namespace name
{
```

```
    public class ShipmentContext : DbContext    //Do
not change the class name
    {
        //Implement property for 'Ships' and
'Containers' with required 'DbSet' declaration
        public DbSet<Ship> Ships { get; set; }
        public DbSet<Container> Containers { get; set;
}

        public ShipmentContext() :
base("ShippingContext")//Dont't change the base name
        {

        }
        protected override void
OnModelCreating(DbModelBuilder modelBuilder)//Don't
chnge the method signature
        {
            //Map Ship entity to ShipDetail table
            //Map Container entity to ContainerDetail
table

modelBuilder.Entity<Ship>().ToTable("ShipDetail");

modelBuilder.Entity<Container>().ToTable("ContainerDet
ail");

             modelBuilder.Entity<Ship>().HasKey(t =>
t.ShipId);

modelBuilder.Entity<Container>().HasRequired(t =>
t.Ship).WithMany().HasForeignKey(t => t.ShipId);

             modelBuilder.Entity<Ship>().HasMany(c =>
c.ContainerList);

             modelBuilder.Entity<Ship>().Property(s =>
s.ShipId).HasColumnName("Ship_Id");
             modelBuilder.Entity<Ship>().Property(s =>
s.ShipName).HasColumnName("Ship_Name");
             modelBuilder.Entity<Ship>().Property(s =>
s.DestLongitude).HasColumnName("DLO");
             modelBuilder.Entity<Ship>().Property(s =>
s.DestLatitude).HasColumnName("DLA");
             modelBuilder.Entity<Ship>().Property(s =>
s.CurrentLongitude).HasColumnName("CLO");
             modelBuilder.Entity<Ship>().Property(s =>
s.CurrentLatitude).HasColumnName("CLA");



            //Make 'ShipId' as Foreign key in
Container Entity
```

```
            //configures one-to-many relationship as
mentioned in the problem statement
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.Entity;

namespace Shipment //Do not change the namespace
{
    public class ShipmentUtil //Dont'Change the class
name
    {
        public Ship AddShip(Ship Ship)//Don't change
the method signature
        {
          //Implement code here

          ShipmentContext context = new
ShipmentContext();
          DbSet<Ship> cont = context.Ships;
          cont.Add(Ship);
          if (context.SaveChanges()>0)
          {
              return Ship;
          }
          else
          {
              return null;
          }
        }

    }
}


***VehicleRepo

//THIS IS FOR REFERENCE ONLY. YOU ARE NOT REQUIRED TO
MAKE ANY CHANGES HERE
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace VehicleProject //DO NOT change the namespace
name
```

```
{
    public interface IVehicleRepository
    {
        List<Vehicle> GetVehicles();
        List<Vehicle> GetVehicleByStatus(String
Status);
        Vehicle UpdateRepairStatus(int VehicleId,
String Status);
        double CalulateRepairCost(String defectType);
        Vehicle VehicleCheckIn(Vehicle v);
        Boolean CheckDefectType(String defectType);
    }
}

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

namespace VehicleProject
{
    public class Program
    {
        static void Main(string[] args) //DO NOT
change the 'Main' method signature
        {

           //Implement the code here

        }
    }
}


//THIS IS FOR REFERENCE ONLY. YOU ARE NOT REQUIRED TO
MAKE ANY CHANGES HERE
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace VehicleProject  //DO NOT change the
namespace name
{
    public class Vehicle  //DO NOT change the class
name
    {
        public int VehicleId { get; set; }
        public String VehicleName { get; set; }
        public String VehicleType { get; set; }
        public String Color { get; set; }
        public String DefectType { get; set; }
```

```
        public String Status { get; set; }
        public Double RepairCost { get; set; }


        public override string ToString()
        {
            return VehicleId + "=" + VehicleName + "="
+ VehicleType + "=" + Color + "=" + DefectType + "=" +
Status + "=" + RepairCost;
        }
    }
}


using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace VehicleProject //DO NOT change the namespace
name
{
    public class VehicleContext : DbContext //Do Not
change the definition
    {

        public VehicleContext() : base("VehicleDB")
// Do Not change the definition and base db name
        {


        }
       //Implement property for 'Vehicles' with
required declaration
       public DbSet<Vehicle> Vehicles{ get;set;}
    }
}


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

namespace VehicleProject //DO NOT change the namespace
name
{
    public class VehicleRepository :
IVehicleRepository //DO NOT change the class
definition
```

```
    {
        VehicleContext context=new VehicleContext();
        public double CalulateRepairCost(string
defectType) //DO NOT change the  method signature and
return type
        {
          double cost=0;
          if(defectType=="Tyre")
          {
              cost=17000;
          }
          else if(defectType=="Fuel System")
          {
              cost=15000;
          }
          else if(defectType=="Engine")
          {
              cost=20000;
          }
          else if(defectType=="Break System")
          {
              cost=8000;
          }
          return cost;
        }



        public List<Vehicle> GetVehicles() //DO NOT
change the  method signature and return type
        {
           return context.Vehicles.ToList();

        }



        public Vehicle UpdateRepairStatus(int
VehicleId, string status) //DO NOT change the  method
signature and return type
        {
         Vehicle update=new Vehicle();
         update=context.Vehicles.Find(VehicleId);
         update.Status=status;
         context.SaveChanges();
         Console.WriteLine(update.ToString());
         return update;
        }

        public Vehicle VehicleCheckIn(Vehicle v) //DO
NOT change the  method signature and return type
        {
           Vehicle v1=new Vehicle();
           v1=context.Vehicles.Add(v);
```

```
            context.SaveChanges();
            Console.WriteLine("Vehicles Inserted
Successfully");
            Console.WriteLine(v1.ToString());
            return v1;
        }


        public List<Vehicle> GetVehicleByStatus(string
status) //DO NOT change the  method signature and
return type
        {
            List<Vehicle> query=new List<Vehicle>();

query=context.Vehicles.Where(c=>c.Status==status).ToLi
st();
            return query;
        }

        public bool CheckDefectType(string defectType)
//DO NOT change the  method signature and return type
        {
          if(defectType=="Tyre"||defectType=="Fuel
System"||defectType=="Engine"||defectType=="Break
System")
          {
              return true;
          }
          else
          {
              return false;
          }

        }
    }
}
```