Application Overview:

This web application is followed the layered architecture. The layers consisting Database, Data Access Layer, Data Model Layer, Business Logic Layer, Controls Layer, Presentation Layer the total overview of the application is shown in image1. All Layers will be described in the next sections.

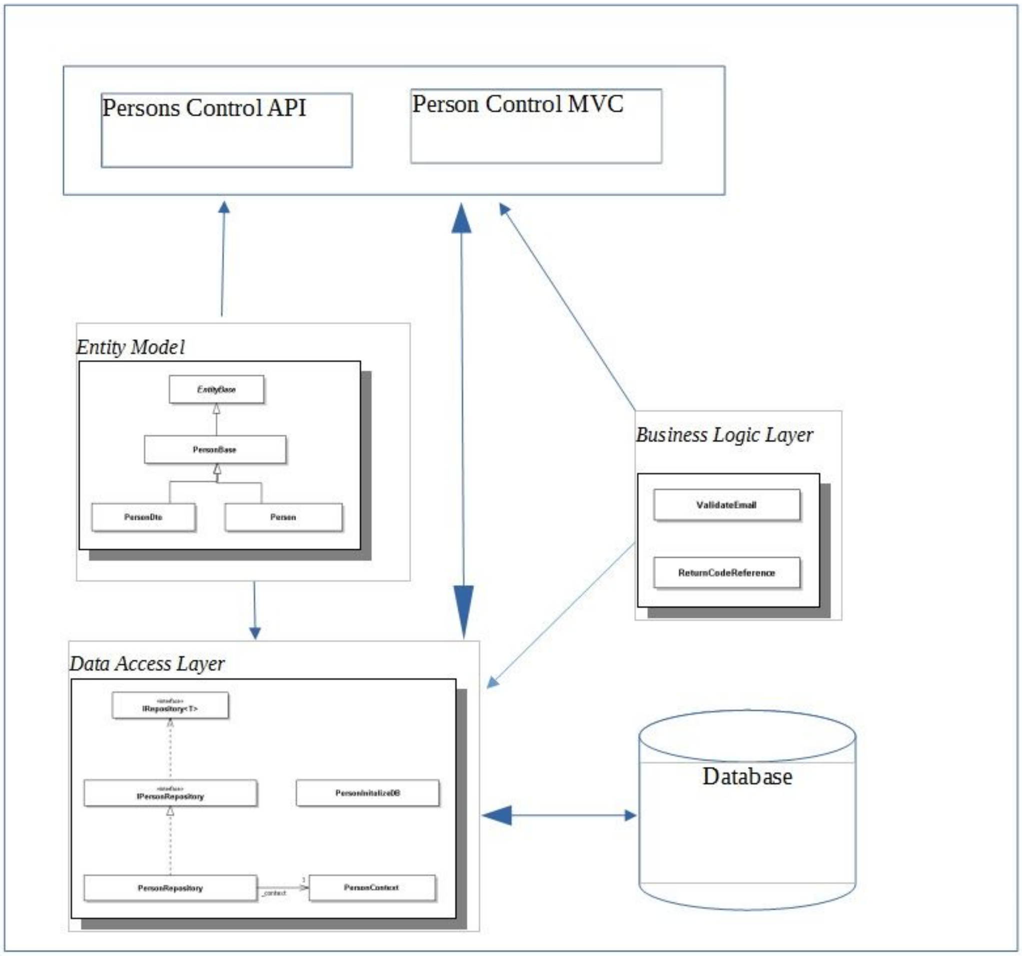


Image 1: Application Architecture

Application Flow:

As shown in image1, the application consists of layers and modules. The application flow is shown in image2.

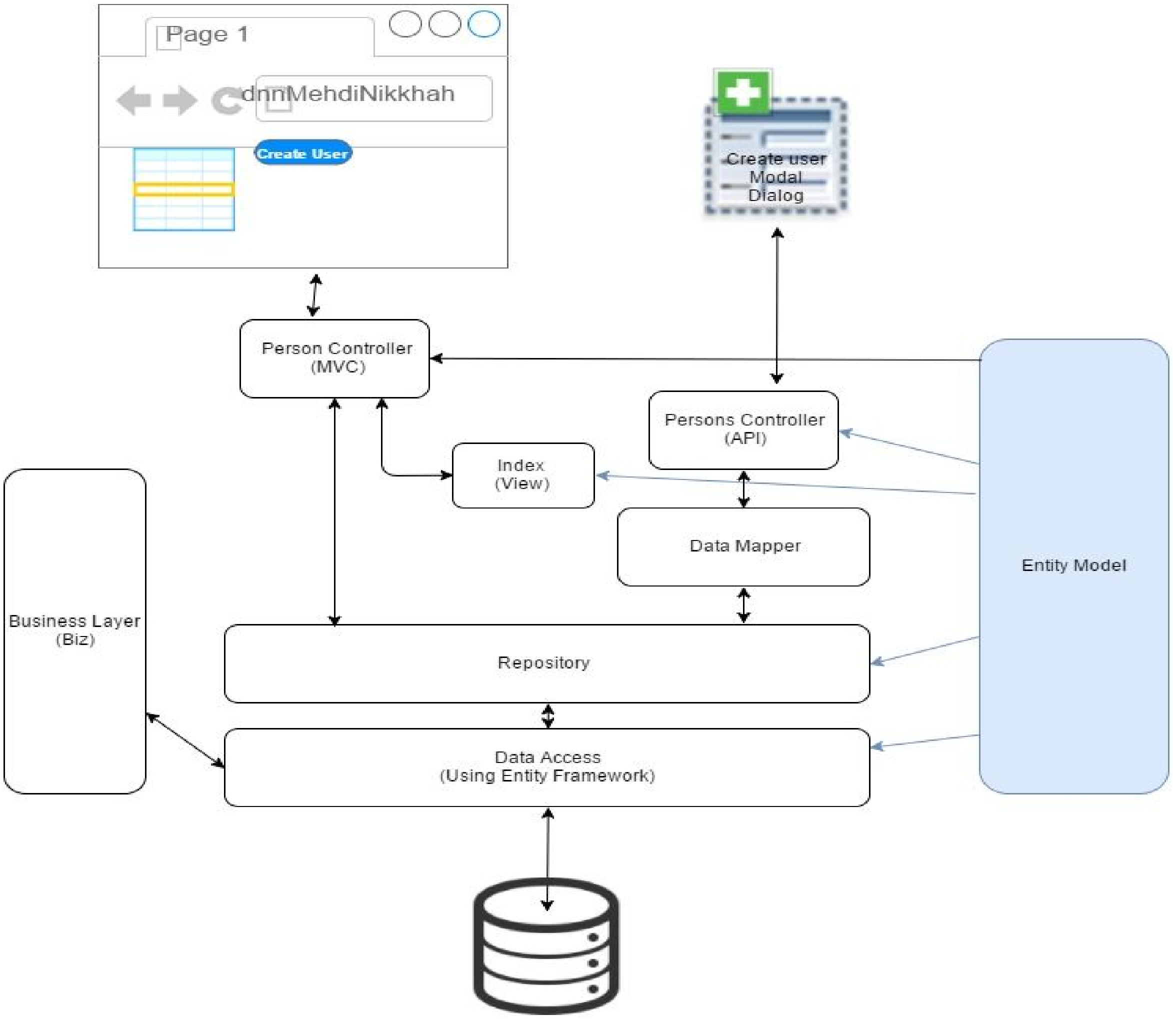
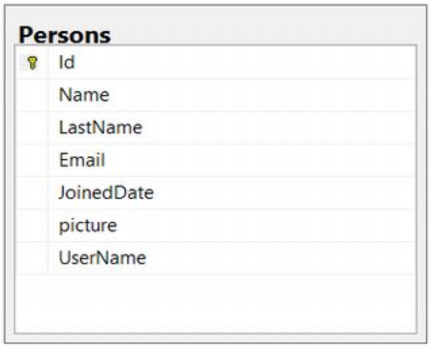


Image 2: Application Architecture

Database Layer:



Persons

Id
Name
LastName
Email
JoinedDate
picture
UserName

Image 3: Database Person Table

In the current project, Microsoft SQL Server 2014 is used. This database includes two tables of person and MigrationHistory. (image3)

Data Access Layer:

For the DAL layer, Microsoft Entity Framework is used to manage CRUD operations. All code is implemented in the DnnMehdiNikkhah01.DataAccess class library. The repository pattern is implemented in this layer.

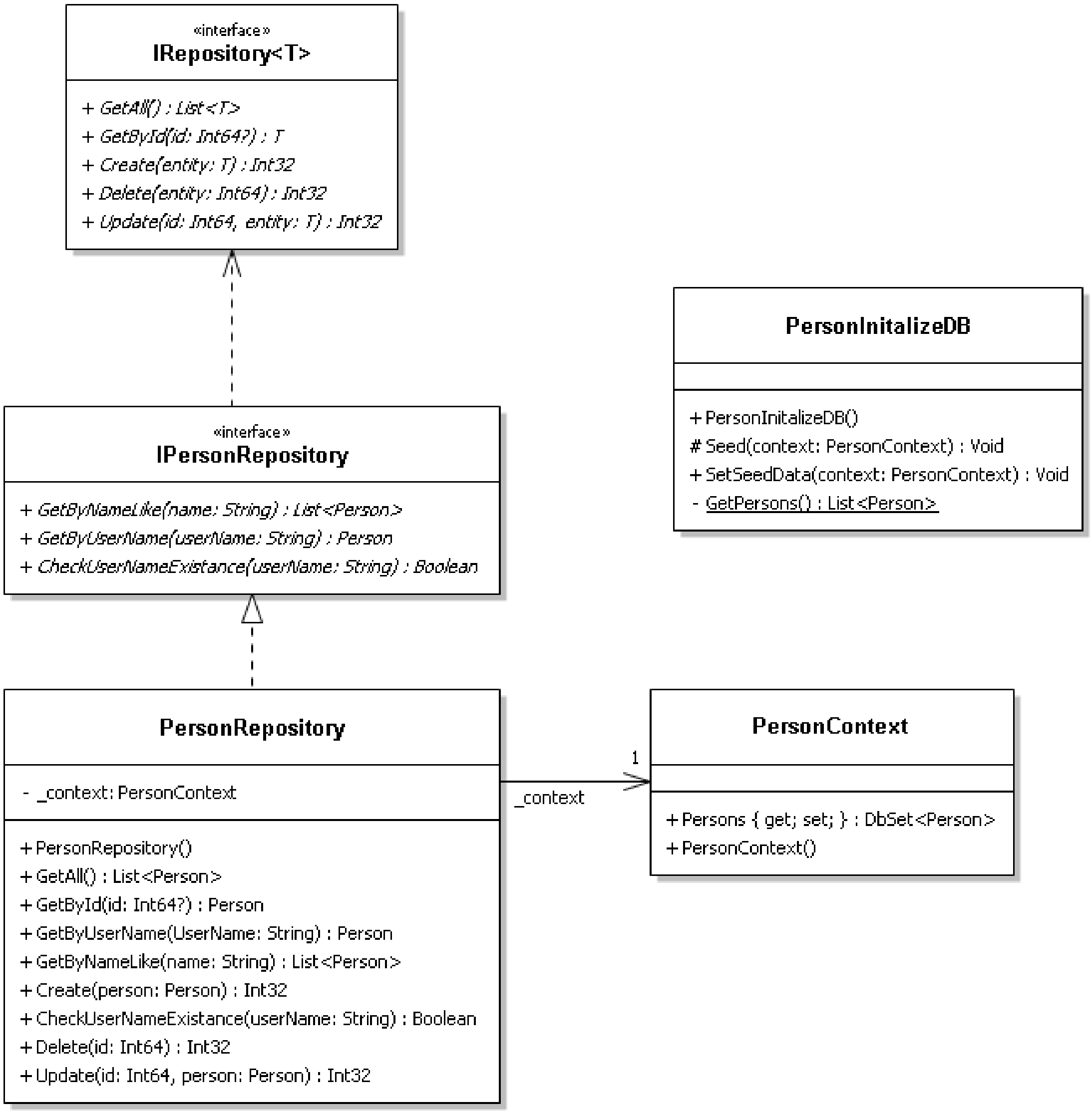

«interface»
IRepository<T>
+ GetAll() : List<T>
+ GetById(id: Int64?) : T
+ Create(entity: T) : Int32
+ Delete(entity: Int64) : Int32
+ Update(id: Int64, entity: T) : Int32
PersonInitalizeDB
+ PersonInitalizeDB()
# Seed(context: PersonContext) : Void
+ SetSeedData(context: PersonContext) : Void
- GetPersons() : List<Person>
«interface»
IPersonRepository
+ GetByNameLike(name: String) : List<Person>
+ GetByUserName(userName: String) : Person
+ CheckUserNameExistance(userName: String) : Boolean
PersonRepository
- _context: PersonContext
+ PersonRepository()
+ GetAll() : List<Person>
+ GetById(id: Int64?) : Person
+ GetByUserName(UserName: String) : Person
+ GetByNameLike(name: String) : List<Person>
+ Create(person: Person) : Int32
+ CheckUserNameExistance(userName: String) : Boolean
+ Delete(id: Int64) : Int32
+ Update(id: Int64, person: Person) : Int32
_context
1
PersonContext
+ Persons { get; set; } : DbSet<Person>
+ PersonContext()

Image 4: Data Access Layer

Entity Model Layer:

This layer includes entity model classes that represent database tables. ComponentModel.DataAnotation and ComponentModel.DataAnotation.Schema namespaces are used to describe the tables and data constrains by attributes. All code is implemented in the DnnMehdiNikkhah01.EntityModel .Net class library.

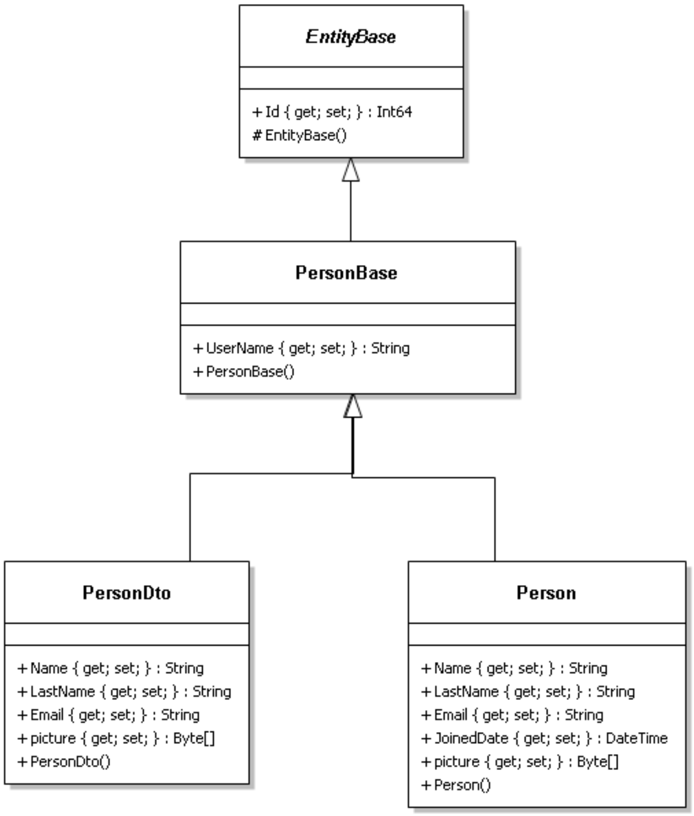

EntityBase
+ Id { get; set; } : Int64
# EntityBase()
PersonBase
+ UserName { get; set; } : String
+ PersonBase()
PersonDto
+ Name { get; set; } : String
+ LastName { get; set; } : String
+ Email { get; set; } : String
+ picture { get; set; } : Byte[]
+ PersonDto()
Person
+ Name { get; set; } : String
+ LastName { get; set; } : String
+ Email { get; set; } : String
+ JoinedDate { get; set; } : DateTime
+ picture { get; set; } : Byte[]
+ Person()

Image 5: Entiy Model

Business Logic Layer

In this layer all business logics and rolls for the application domain are defined, such as checking Email format or the return codes. The layer is implemented in the DnnMehdiNikkhah01.Biz .Net class library.

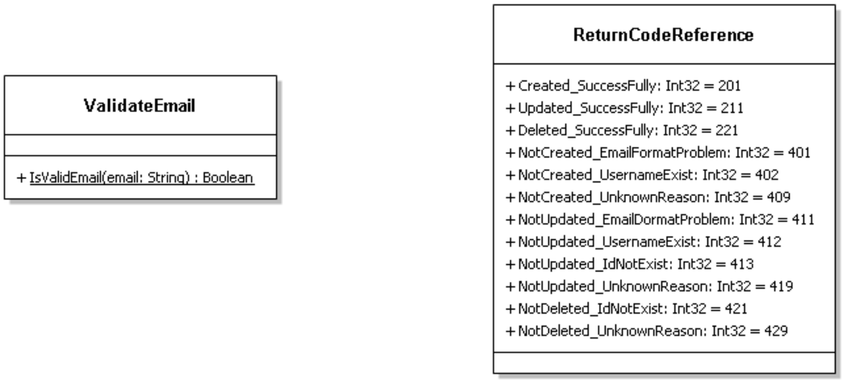

ValidateEmail
+ IsValidEmail(email: String) : Boolean
ReturnCodeReference
+ Created_SuccessFully: Int32 = 201
+ Updated_SuccessFully: Int32 = 211
+ Deleted_SuccessFully: Int32 = 221
+ NotCreated_EmailFormatProblem: Int32 = 401
+ NotCreated_UsernameExist: Int32 = 402
+ NotCreated_UnknownReason: Int32 = 409
+ NotUpdated_EmailDormatProblem: Int32 = 411
+ NotUpdated_UsernameExist: Int32 = 412
+ NotUpdated_IdNotExist: Int32 = 413
+ NotUpdated_UnknownReason: Int32 = 419
+ NotDeleted_IdNotExist: Int32 = 421
+ NotDeleted_UnknownReason: Int32 = 429

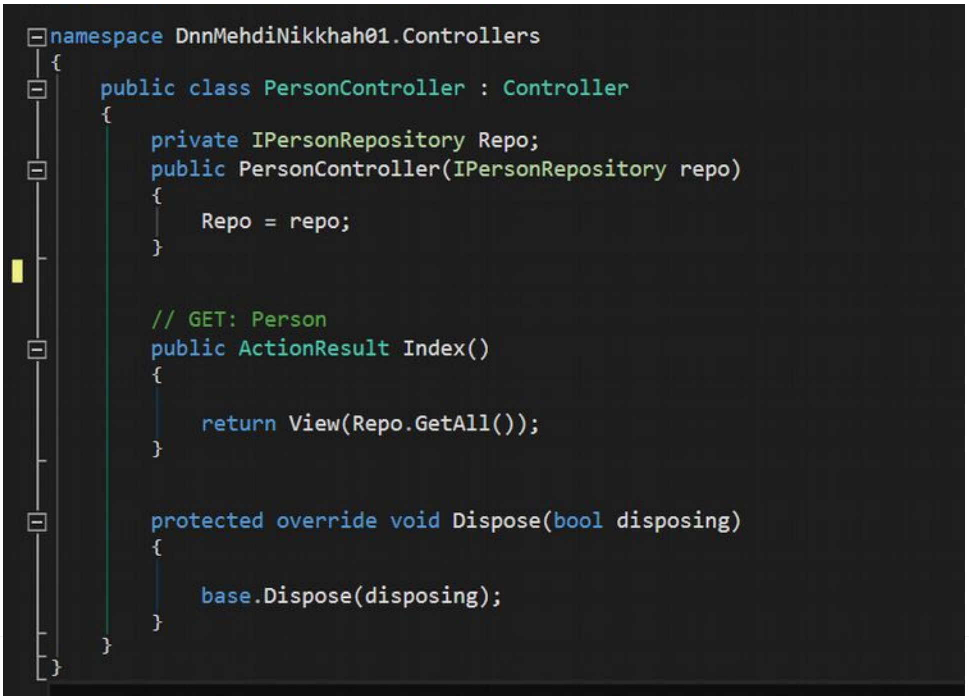
Image 6: Business Layer

Service Layer:

Web API is used and implemented in the DnnMehdiNikkhah01 which is the start up project of the solution. The PersonsController.cs contains the web API class, that is located in Controller/Api folder of the project. The only used action is POST HTTPAction.

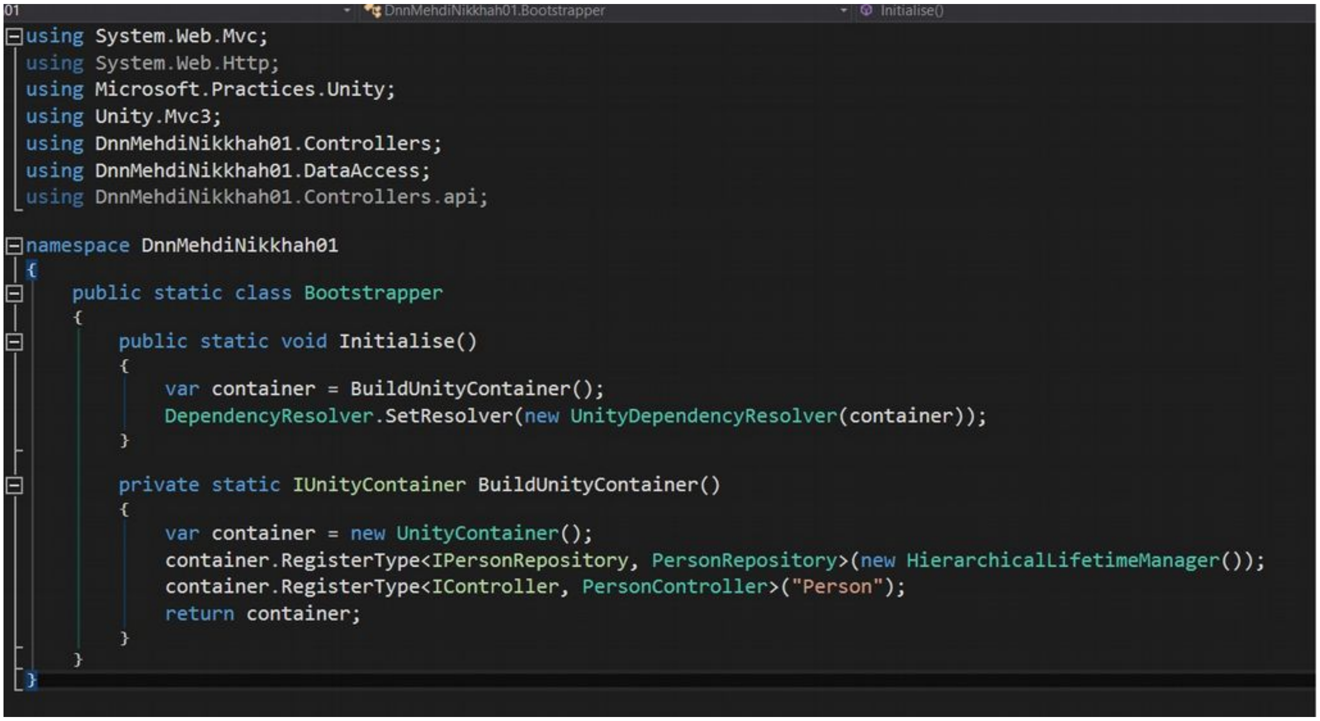
User Interface Layer:

MVC controller contains the index action to handle the end user requests and provide the corresponding view which return the user list. The PersonController.cs contains the index actions is located in the Controller folder of the project. The Inverse Injection Pattern is used in Person Controller.



```
namespace DnnMehdiNikkhah01.Controllers
{
    public class PersonController : Controller
    {
        private IPersonRepository Repo;
        public PersonController(IPersonRepository repo)
        {
            Repo = repo;
        }


        // GET: Person
        public ActionResult Index()
        {

            return View(Repo.GetAll());
        }


        protected override void Dispose(bool disposing)
        {

            base.Dispose(disposing);
        }
    }
}
```

Image 7: MVC PersonController

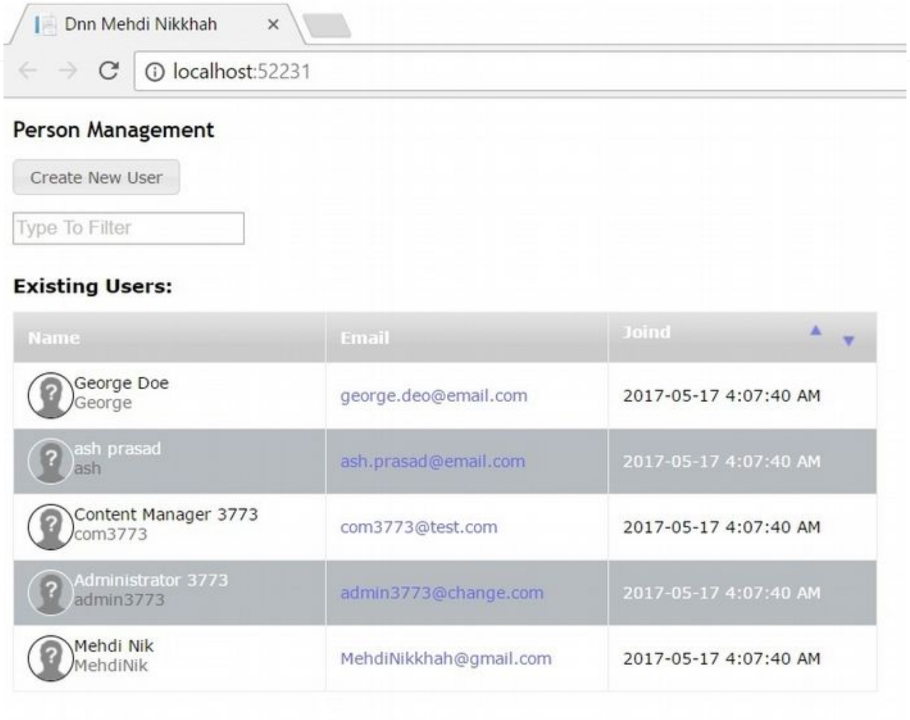


```
using System.Web.Mvc;
using System.Web.Http;
using Microsoft.Practices.Unity;
using Unity.Mvc3;
using DnnMehdiNikkhah01.Controllers;
using DnnMehdiNikkhah01.DataAccess;
using DnnMehdiNikkhah01.Controllers.api;

namespace DnnMehdiNikkhah01
{
    public static class Bootstrapper
    {
        public static void Initialise()
        {
            var container = BuildUnityContainer();
            DependencyResolver.SetResolver(new UnityDependencyResolver(container));
        }

        private static IUnityContainer BuildUnityContainer()
        {
            var container = new UnityContainer();
            container.RegisterType<IPersonRepository, PersonRepository>(new HierarchicalLifetimeManager());
            container.RegisterType<IController, PersonController>("Person");
            return container;
        }
    }
}
```

Image 8: Inverse Injection (dependency Injection using Unity)

JQuery

For the UI layer a combination of Razer formatting and Jquery code is used. The related views are located in View/Person folder of the project. By using Jquery browser is capable to insert new user async, update list, sort and search Names.

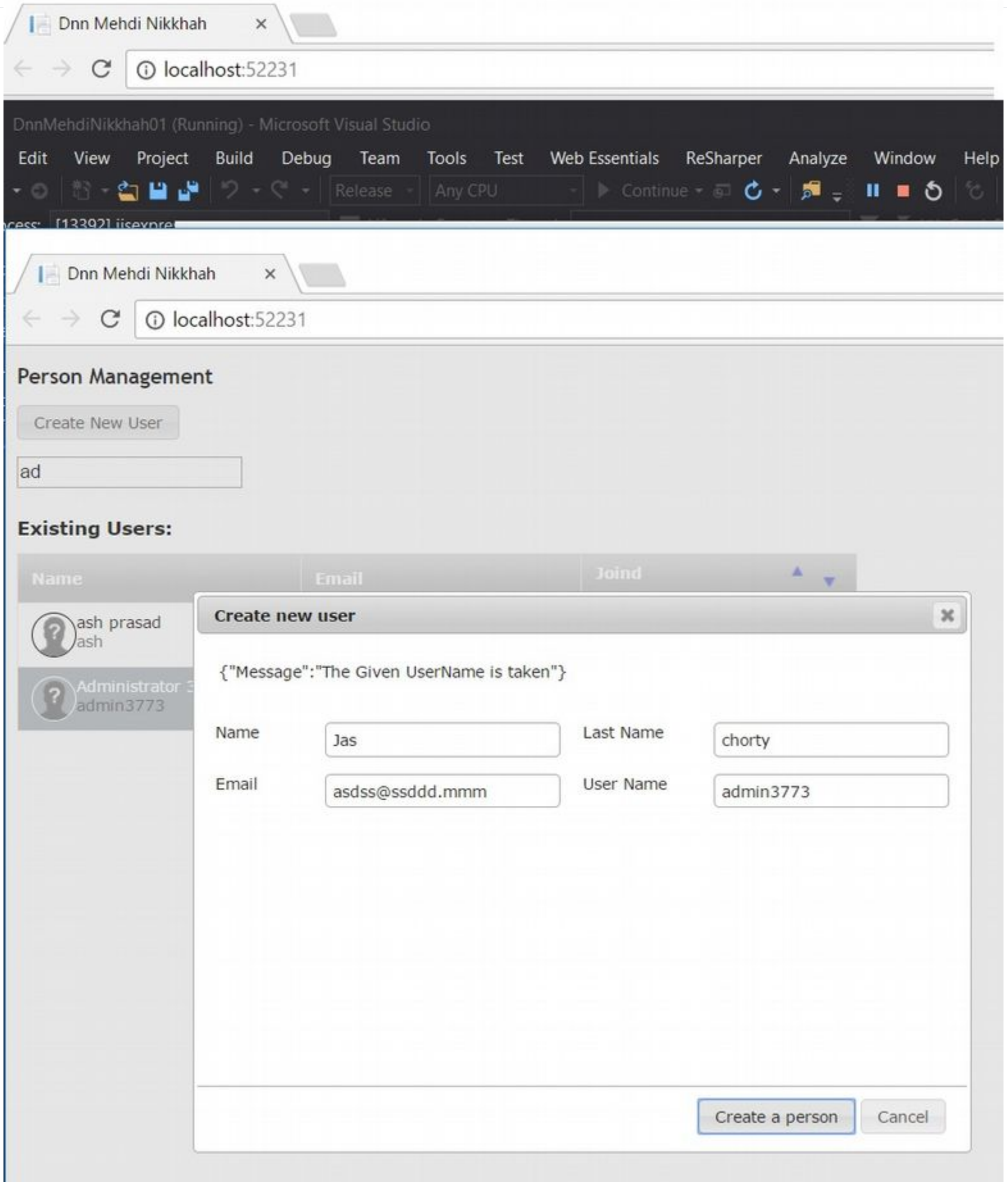


Person Management

Existing Users:

Name	Email	Joind ▲ ▼
George Doe George	george.deo@email.com	2017-05-17 4:07:40 AM
ash prasad ash	ash.prasad@email.com	2017-05-17 4:07:40 AM
Content Manager 3773 com3773	com3773@test.com	2017-05-17 4:07:40 AM
Administrator 3773 admin3773	admin3773@change.com	2017-05-17 4:07:40 AM
Mehdi Nik MehdiNik	MehdiNikkhah@gmail.com	2017-05-17 4:07:40 AM

Image 9: User List


Dnn Mehdi Nikkhah
localhost:52231
DnnMehdiNikkhah01 (Running) - Microsoft Visual Studio
Edit View Project Build Debug Team Tools Test Web Essentials ReSharper Analyze Window Help
Release
Any CPU
Continue
Dnn Mehdi Nikkhah
localhost:52231
Create New User
Name
Email
Joind
Create a person
Cancel

Person Management

ad

Existing Users:

Name
ash prasad ash
Administrator 3 admin3773

Create new user

{"Message":"The Given UserName is taken"}

Name	Jas	Last Name	chorty
Email	asdss@ssddd.mmm	User Name	admin3773

Image 10: Create User Modal Dialog