Karachi Campus

Discovering Knowledge

Department of Software Engineering

SOFTWARE DESIGN ARCHITECTURE (LAB)

Student Management System

Group Members

Uzair Mehmood	51739
Noman Ali Siddique	51777
Hammad Shaikh	51717

BSE-6A

Spring 2020

Submitted to

Engr. Muhammad Rehan Baig

Table of Contents

1. ABSTRACT

Students form a main part of any institution that concerns with. But the institutions find it difficult to keep details of so many students of the organization just in one stretch. It will involve a lot of pen paper work. Sometimes there will be some huge heap of files bundled up and kept together in some corner of the office. If you want any information regarding the particular student then it can be obtained by just entering the roll number or the name of the student to be searched. This student management system will make the work of storing the data in an organized way.

The student management system application will help in managing the student's reports, results and exams will become easier with one such system. It will also help in saving time and effort. The user interface must be user friendly and easy to understand. The information of the particular student will be obtained in just one mouse click. Some of the features that it can include are as follows:

Student database management: The details of the students of the organizations can be stored in the database with the use this application.

Results: The results of the students can also be accessed and stored through this application.

Security: The data that will be disclosed will be more secure since there will be no access to the unknown users.

Performance: The performance of the students might be in curriculum as well as co-curriculum can also be stored through the use of this application.

One-click access: You will obtain the details of the students by entering his/her name or the roll number just in one click.

User interface: The user interface must be simple and easy to understand.

Personal details: All the personal details of the students can be obtained in just one mouse click.

2. INTRODUCTION

Student management system is an environment where all the process of the student in the institution is managed. It is done through the automated computerized method. Conventionally this system is done using papers, file and binders.

This system saves the time of the student and of the administrator. It includes process like registration of the student's details, assigning the department based on their course and maintenance of the record.

This system reduces the cost and workforce required for this job. As the system is online the information is globally present to everyone. This makes the system easy to handle and feasible for finding the omission with updating at the same time.

As for the existing system, they use to maintain their record manually which makes it vulnerable to security. If filed a query to search or update in a manual system, it will take a lot of time to process the query and make a report which is a tedious job.

As the system used in the institute is outdated as it requires paper, files and the binders, which will require the human workforce to maintain them. To get registered in the institute, a student in this system one should come to the university.

Get the forms from the counter while standing in the queue which consumes a lot of the student's time as well as of management team. As the number of the student increases in the institute manually managing the strength becomes a hectic job for the administrator.

This computerized system store all the data in the database which makes it easy to fetch and update whenever needed.

3. SCOPE

The scope of this project is limited to the design and implementation of a Student Information Management System. The purpose of the Student Information Management system is to allow for storing information of large number of students. Different privileges are given to different types of users. The Student Information Management System will make use of Visual Basic 6.0 as its front end and Microsoft Access as its back end.

4. CODE

4.1. Properties

4.1.1. AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("Student_Management_System")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("Student_Management_System")]
[assembly: AssemblyCopyright("Copyright ©  2020")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components.  If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is exposed to COM
[assembly: Guid("987e33ff-db5f-42a4-b70c-0b36d39f9174")]

// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
```

```
//
// You can specify all the values or you can default the Revision and Build Numbers
// by using the '*' as shown below:
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

4.2. App_Start

4.2.1. BuildConfig.cs

```
using System.Web;
using System.Web.Optimization;

namespace Student_Management_System
{
    public class BundleConfig
    {
        // For more information on bundling, visit
http://go.microsoft.com/fwlink/?LinkId=301862
        public static void RegisterBundles(BundleCollection bundles)
        {
            bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
                        "~/Scripts/jquery-{version}.js"));

            bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
                        "~/Scripts/jquery.validate*"));

            // Use the development version of Modernizr to develop with and learn
from. Then, when you're
            // ready for production, use the build tool at http://modernizr.com to
pick only the tests you need.
            bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
                        "~/Scripts/modernizr-*"));

            bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(
                      "~/Scripts/bootstrap.js",
                      "~/Scripts/respond.js"));

            bundles.Add(new StyleBundle("~/Content/css").Include(
                      "~/Content/bootstrap.css",
                      "~/Content/site.css"));
        }
    }
}
```

4.2.2. FilterConfig.cs

```
using System.Web;
using System.Web.Mvc;

namespace Student_Management_System
{
    public class FilterConfig
    {
        public static void RegisterGlobalFilters(GlobalFilterCollection filters)
        {
            filters.Add(new HandleErrorAttribute());
        }
    }
}
```

4.2.3. IdentityCongig.cs

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;
using System.Web;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin;
using Microsoft.Owin.Security;
using Student_Management_System.Models;

namespace Student_Management_System
{
    public class EmailService : IIdentityMessageService
    {
        public Task SendAsync(IdentityMessage message)
        {
            // Plug in your email service here to send an email.
            return Task.FromResult(0);
        }
    }

    public class SmsService : IIdentityMessageService
    {
        public Task SendAsync(IdentityMessage message)
        {
            // Plug in your SMS service here to send a text message.
            return Task.FromResult(0);
        }
    }

    // Configure the application user manager used in this application. UserManager
is defined in ASP.NET Identity and is used by the application.
    public class ApplicationUserManager : UserManager<ApplicationUser>
    {
        public ApplicationUserManager(IUserStore<ApplicationUser> store)
            : base(store)
        {
        }

        public static ApplicationUserManager
Create(IdentityFactoryOptions<ApplicationUserManager> options, IOwinContext context)
        {
            var manager = new ApplicationUserManager(new
UserStore<ApplicationUser>(context.Get<ApplicationDbContext>()));
            // Configure validation logic for usernames
            manager.UserValidator = new UserValidator<ApplicationUser>(manager)
            {
                AllowOnlyAlphanumericUserNames = false,
                RequireUniqueEmail = true
            };
            // Configure validation logic for passwords
            manager.PasswordValidator = new PasswordValidator
            {
                RequiredLength = 6,
                RequireNonLetterOrDigit = true,
                RequireDigit = true,
                RequireLowercase = true,
                RequireUppercase = true,
            };

            // Configure user lockout defaults
            manager.UserLockoutEnabledByDefault = true;
            manager.DefaultAccountLockoutTimeSpan = TimeSpan.FromMinutes(5);
            manager.MaxFailedAccessAttemptsBeforeLockout = 5;

            // Register two factor authentication providers. This application uses
Phone and Emails as a step of receiving a code for verifying the user
            // You can write your own provider and plug it in here.
            manager.RegisterTwoFactorProvider("Phone Code", new
PhoneNumberTokenProvider<ApplicationUser>
            {
                MessageFormat = "Your security code is {0}"
            });
            manager.RegisterTwoFactorProvider("Email Code", new
EmailTokenProvider<ApplicationUser>
            {
                Subject = "Security Code",
                BodyFormat = "Your security code is {0}"
            });
            manager.EmailService = new EmailService();
            manager.SmsService = new SmsService();
            var dataProtectionProvider = options.DataProtectionProvider;
            if (dataProtectionProvider != null)
            {
                manager.UserTokenProvider =
                    new
DataProtectorTokenProvider<ApplicationUser>(dataProtectionProvider.Create("ASP.NET
Identity"));
            }
            return manager;
        }
    }

    // Configure the application sign-in manager which is used in this application.
    public class ApplicationSignInManager : SignInManager<ApplicationUser, string>
    {
        public ApplicationSignInManager(ApplicationUserManager userManager,
IAuthenticationManager authenticationManager)
            : base(userManager, authenticationManager)
        {
        }

        public override Task<ClaimsIdentity> CreateUserIdentityAsync(ApplicationUser
user)
        {
            return
user.GenerateUserIdentityAsync((ApplicationUserManager)UserManager);
        }
        public static ApplicationSignInManager
Create(IdentityFactoryOptions<ApplicationSignInManager> options, IOwinContext
context)
        {
            return new
ApplicationSignInManager(context.GetUserManager<ApplicationUserManager>(),
context.Authentication);
        }
    }
}
```

4.2.4. RouteConfig.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace Student_Management_System
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Course", action = "Index", id =
UrlParameter.Optional }
            );
        }
    }
}
```

4.2.5. Startup.Auth.cs

```
using System;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin;
using Microsoft.Owin.Security.Cookies;
using Microsoft.Owin.Security.Google;
using Owin;
using Student_Management_System.Models;

namespace Student_Management_System
{
    public partial class Startup
    {
        // For more information on configuring authentication, please visit
http://go.microsoft.com/fwlink/?LinkId=301864
        public void ConfigureAuth(IAppBuilder app)
        {
            // Configure the db context, user manager and signin manager to use a
single instance per request
```

```
            app.CreatePerOwinContext(ApplicationDbContext.Create);

app.CreatePerOwinContext<ApplicationUserManager>(ApplicationUserManager.Create);

app.CreatePerOwinContext<ApplicationSignInManager>(ApplicationSignInManager.Create);

            // Enable the application to use a cookie to store information for the
signed in user
            // and to use a cookie to temporarily store information about a user
logging in with a third party login provider
            // Configure the sign in cookie
            app.UseCookieAuthentication(new CookieAuthenticationOptions
            {
                AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
                LoginPath = new PathString("/Account/Login"),
                Provider = new CookieAuthenticationProvider
                {
                    // Enables the application to validate the security stamp when
the user logs in.
                    // This is a security feature which is used when you change a
password or add an external login to your account.
                    OnValidateIdentity =
SecurityStampValidator.OnValidateIdentity<ApplicationUserManager, ApplicationUser>(
                        validateInterval: TimeSpan.FromMinutes(30),
                        regenerateIdentity: (manager, user) =>
user.GenerateUserIdentityAsync(manager))
                }
            });
            app.UseExternalSignInCookie(DefaultAuthenticationTypes.ExternalCookie);

            // Enables the application to temporarily store user information when
they are verifying the second factor in the two-factor authentication process.
            app.UseTwoFactorSignInCookie(DefaultAuthenticationTypes.TwoFactorCookie,
TimeSpan.FromMinutes(5));

            // Enables the application to remember the second login verification
factor such as phone or email.
            // Once you check this option, your second step of verification during
the login process will be remembered on the device where you logged in from.
            // This is similar to the RememberMe option when you log in.

app.UseTwoFactorRememberBrowserCookie(DefaultAuthenticationTypes.TwoFactorRememberBr
owserCookie);

            // Uncomment the following lines to enable logging in with third party
login providers
            //app.UseMicrosoftAccountAuthentication(
            //    clientId: "",
            //    clientSecret: "");

            //app.UseTwitterAuthentication(
            //   consumerKey: "",
            //   consumerSecret: "");

            //app.UseFacebookAuthentication(
            //   appId: "",
            //   appSecret: "");

            //app.UseGoogleAuthentication(new GoogleOAuth2AuthenticationOptions()
            //{
            //    ClientId = "",
            //    ClientSecret = ""
            //});
        }
    }
}
```

4.3. Controller

4.3.1. AccountController.cs

```
using System;

using System.Globalization;

using System.Linq;

using System.Security.Claims;

using System.Threading.Tasks;

using System.Web;

using System.Web.Mvc;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.Owin;

using Microsoft.Owin.Security;

using Student_Management_System.Models;


namespace Student_Management_System.Controllers

{

    [Authorize]

    public class AccountController : Controller

    {

        private ApplicationSignInManager _signInManager;

        private ApplicationUserManager _userManager;


        public AccountController()

        {

        }
        public         AccountController(ApplicationUserManager         userManager,
ApplicationSignInManager signInManager )

        {

            UserManager = userManager;

            SignInManager = signInManager;

        }


        public ApplicationSignInManager SignInManager

        {

            get

            {

                return                       _signInManager                       ??
HttpContext.GetOwinContext().Get<ApplicationSignInManager>();

            }

            private set

            {

                _signInManager = value;

            }

        }


        public ApplicationUserManager UserManager

        {

            get

            {

                return                        _userManager                        ??
HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();

            }

            private set

            {

                _userManager = value;

            }
        }

        //
        // GET: /Account/Login
        [AllowAnonymous]
        public ActionResult Login(string returnUrl)
        {
            ViewBag.ReturnUrl = returnUrl;
            return View();
        }

        //
        // POST: /Account/Login
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
        {
            if (!ModelState.IsValid)
            {
                return View(model);
            }

            // This doesn't count login failures towards account lockout
            // To enable password failures to trigger account lockout, change to
shouldLockout: true
            var result = await SignInManager.PasswordSignInAsync(model.Email,
model.Password, model.RememberMe, shouldLockout: false);
            switch (result)
            {
                case SignInStatus.Success:
                    return RedirectToLocal(returnUrl);
                case SignInStatus.LockedOut:
                    return View("Lockout");
                case SignInStatus.RequiresVerification:
                    return RedirectToAction("SendCode", new { ReturnUrl = returnUrl,
RememberMe = model.RememberMe });
                case SignInStatus.Failure:
                default:
                    ModelState.AddModelError("", "Invalid login attempt.");
                    return View(model);
            }
        }

        //
        // GET: /Account/VerifyCode
        [AllowAnonymous]
        public async Task<ActionResult> VerifyCode(string provider, string returnUrl,
bool rememberMe)
        {
            // Require that the user has already logged in via username/password or
external login
            if (!await SignInManager.HasBeenVerifiedAsync())
            {
                return View("Error");
            }
            return View(new VerifyCodeViewModel { Provider = provider, ReturnUrl =
returnUrl, RememberMe = rememberMe });
        }

        //
        // POST: /Account/VerifyCode
        }


        [HttpPost]

        [AllowAnonymous]

        [ValidateAntiForgeryToken]

        public async Task<ActionResult> VerifyCode(VerifyCodeViewModel model)

        {

            if (!ModelState.IsValid)

            {

                return View(model);

            }


            // The following code protects for brute force attacks against the two
factor codes.

            // If a user enters incorrect codes for a specified amount of time then
the user account

            // will be locked out for a specified amount of time.

            // You can configure the account lockout settings in IdentityConfig

            var  result  =  await  SignInManager.TwoFactorSignInAsync(model.Provider,
model.Code, isPersistent:  model.RememberMe, rememberBrowser: model.RememberBrowser);

            switch (result)

            {

                case SignInStatus.Success:

                    return RedirectToLocal(model.ReturnUrl);

                case SignInStatus.LockedOut:

                    return View("Lockout");

                case SignInStatus.Failure:

                default:

                    ModelState.AddModelError("", "Invalid code.");

                    return View(model);

            }

        }
        //
        // GET: /Account/Register
        [AllowAnonymous]
        public ActionResult Register()
        {
            return View();
        }

        //
        // POST: /Account/Register
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> Register(RegisterViewModel model)
        {
            if (ModelState.IsValid)
            {
                var user = new ApplicationUser { UserName = model.Email, Email =
model.Email };
                var result = await UserManager.CreateAsync(user, model.Password);
                if (result.Succeeded)
                {
                    await SignInManager.SignInAsync(user, isPersistent:false,
rememberBrowser:false);

                    // For more information on how to enable account confirmation and
password reset please visit http://go.microsoft.com/fwlink/?LinkID=320771
                    // Send an email with this link
                    // string code = await
UserManager.GenerateEmailConfirmationTokenAsync(user.Id);
                    // var callbackUrl = Url.Action("ConfirmEmail", "Account", new {
userId = user.Id, code = code }, protocol: Request.Url.Scheme);
```

```
                    // await UserManager.SendEmailAsync(user.Id, "Confirm your account", "Please confirm your account by clicking <a href=\"" + callbackUrl + "\">here</a>");

                    return RedirectToAction("Index", "Home");
                }
                AddErrors(result);
            }

            // If we got this far, something failed, redisplay form
            return View(model);
        }

        //
        // GET: /Account/ConfirmEmail
        [AllowAnonymous]
        public async Task<ActionResult> ConfirmEmail(string userId, string code)
        {
            if (userId == null || code == null)
            {
                return View("Error");
            }
            var result = await UserManager.ConfirmEmailAsync(userId, code);
            return View(result.Succeeded ? "ConfirmEmail" : "Error");
        }

        //
        // GET: /Account/ForgotPassword
        [AllowAnonymous]
        public ActionResult ForgotPassword()
        {
            return View();
        }

        //
        // POST: /Account/ForgotPassword
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> ForgotPassword(ForgotPasswordViewModel model)
        {
            if (ModelState.IsValid)
            {
                var user = await UserManager.FindByNameAsync(model.Email);
                if (user == null || !(await
UserManager.IsEmailConfirmedAsync(user.Id)))
                {
                    // Don't reveal that the user does not exist or is not confirmed
                    return View("ForgotPasswordConfirmation");
                }

                // For more information on how to enable account confirmation and
password reset please visit http://go.microsoft.com/fwlink/?LinkID=320771
                // Send an email with this link
                // string code = await
UserManager.GeneratePasswordResetTokenAsync(user.Id);
                // var callbackUrl = Url.Action("ResetPassword", "Account", new {
userId = user.Id, code = code }, protocol: Request.Url.Scheme);
                // await UserManager.SendEmailAsync(user.Id, "Reset Password",
"Please reset your password by clicking <a href=\"" + callbackUrl + "\">here</a>");
                // return RedirectToAction("ForgotPasswordConfirmation", "Account");
            }
```

```
    // If we got this far, something failed, redisplay form
    return View(model);
}

//
// GET: /Account/ForgotPasswordConfirmation
[AllowAnonymous]
public ActionResult ForgotPasswordConfirmation()
{
    return View();
}

//
// GET: /Account/ResetPassword
[AllowAnonymous]
public ActionResult ResetPassword(string code)
{
    return code == null ? View("Error") : View();
}

//
// POST: /Account/ResetPassword
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> ResetPassword(ResetPasswordViewModel model)
{
    if (!ModelState.IsValid)
    {
        return View(model);
```

```
            }
            var user = await UserManager.FindByNameAsync(model.Email);
            if (user == null)
            {
                // Don't reveal that the user does not exist
                return RedirectToAction("ResetPasswordConfirmation", "Account");
            }
            var result = await UserManager.ResetPasswordAsync(user.Id, model.Code,
model.Password);
            if (result.Succeeded)
            {
                return RedirectToAction("ResetPasswordConfirmation", "Account");
            }
            AddErrors(result);
            return View();
        }

        //
        // GET: /Account/ResetPasswordConfirmation
        [AllowAnonymous]
        public ActionResult ResetPasswordConfirmation()
        {
            return View();
        }

        //
        // POST: /Account/ExternalLogin
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public ActionResult ExternalLogin(string provider, string returnUrl)
        {
            // Request a redirect to the external login provider
            return new ChallengeResult(provider, Url.Action("ExternalLoginCallback",
"Account", new { ReturnUrl = returnUrl }));
        }


        //
        // GET: /Account/SendCode
        [AllowAnonymous]
        public async Task<ActionResult> SendCode(string returnUrl, bool rememberMe)
        {
            var userId = await SignInManager.GetVerifiedUserIdAsync();
            if (userId == null)
            {
                return View("Error");
            }
            var                 userFactors                 =                 await
UserManager.GetValidTwoFactorProvidersAsync(userId);
            var factorOptions = userFactors.Select(purpose => new SelectListItem {
Text = purpose, Value = purpose }).ToList();
            return View(new SendCodeViewModel { Providers = factorOptions, ReturnUrl
= returnUrl, RememberMe = rememberMe });
        }


        //
        // POST: /Account/SendCode
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> SendCode(SendCodeViewModel model)
        {
            if (!ModelState.IsValid)
            {
                return View();
            }

            // Generate the token and send it
            if (!await SignInManager.SendTwoFactorCodeAsync(model.SelectedProvider))
            {
                return View("Error");
            }
            return RedirectToAction("VerifyCode", new { Provider =
model.SelectedProvider, ReturnUrl = model.ReturnUrl, RememberMe = model.RememberMe
});
        }

        //
        // GET: /Account/ExternalLoginCallback
        [AllowAnonymous]
        public async Task<ActionResult> ExternalLoginCallback(string returnUrl)
        {
            var loginInfo = await AuthenticationManager.GetExternalLoginInfoAsync();
            if (loginInfo == null)
            {
                return RedirectToAction("Login");
            }

            // Sign in the user with this external login provider if the user already
has a login
            var result = await SignInManager.ExternalSignInAsync(loginInfo,
isPersistent: false);
            switch (result)
            {
                case SignInStatus.Success:
                    return RedirectToLocal(returnUrl);
                case SignInStatus.LockedOut:
                    return View("Lockout");
                case SignInStatus.RequiresVerification:
                    return RedirectToAction("SendCode", new { ReturnUrl = returnUrl,
RememberMe = false });
                case SignInStatus.Failure:
                default:
                    // If the user does not have an account, then prompt the user to
create an account
                    ViewBag.ReturnUrl = returnUrl;
                    ViewBag.LoginProvider = loginInfo.Login.LoginProvider;
                    return View("ExternalLoginConfirmation", new
ExternalLoginConfirmationViewModel { Email = loginInfo.Email });
            }
        }

        //
        // POST: /Account/ExternalLoginConfirmation
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult>
ExternalLoginConfirmation(ExternalLoginConfirmationViewModel model, string
returnUrl)
        {
            if (User.Identity.IsAuthenticated)
            {
                return RedirectToAction("Index", "Manage");
            }
            if (ModelState.IsValid)
            {
                // Get the information about the user from the external login provider
                var info = await AuthenticationManager.GetExternalLoginInfoAsync();
                if (info == null)
                {
                    return View("ExternalLoginFailure");
                }
                var user = new ApplicationUser { UserName = model.Email, Email =
model.Email };
                var result = await UserManager.CreateAsync(user);
                if (result.Succeeded)
                {
                    result = await UserManager.AddLoginAsync(user.Id, info.Login);
                    if (result.Succeeded)
                    {
                        await SignInManager.SignInAsync(user, isPersistent: false,
rememberBrowser: false);
                        return RedirectToLocal(returnUrl);
                    }
                }
                AddErrors(result);
            }

            ViewBag.ReturnUrl = returnUrl;
            return View(model);
        }

        //
        // POST: /Account/LogOff
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult LogOff()
        {

AuthenticationManager.SignOut(DefaultAuthenticationTypes.ApplicationCookie);
            return RedirectToAction("Index", "Home");
        }

        //
        // GET: /Account/ExternalLoginFailure
        [AllowAnonymous]
        public ActionResult ExternalLoginFailure()
        {
            return View();
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                if (_userManager != null)
                {
                    _userManager.Dispose();
                    _userManager = null;
                }

                if (_signInManager != null)
                {
                    _signInManager.Dispose();
                    _signInManager = null;
                }
            }

            base.Dispose(disposing);
        }

        #region Helpers
        // Used for XSRF protection when adding external logins
        private const string XsrfKey = "XsrfId";

        private IAuthenticationManager AuthenticationManager
        {
            get
            {
                return HttpContext.GetOwinContext().Authentication;
            }
        }

        private void AddErrors(IdentityResult result)
        {
            foreach (var error in result.Errors)
            {
                ModelState.AddModelError("", error);
            }
        }

        private ActionResult RedirectToLocal(string returnUrl)
        {
            if (Url.IsLocalUrl(returnUrl))
            {
```

```
                return Redirect(returnUrl);
            }
            return RedirectToAction("Index", "Home");
        }

        internal class ChallengeResult : HttpUnauthorizedResult
        {
            public ChallengeResult(string provider, string redirectUri)
                : this(provider, redirectUri, null)
            {
            }

            public  ChallengeResult(string  provider,  string  redirectUri,  string
userId)
            {
                LoginProvider = provider;
                RedirectUri = redirectUri;
                UserId = userId;
            }

            public string LoginProvider { get; set; }
            public string RedirectUri { get; set; }
            public string UserId { get; set; }

            public override void ExecuteResult(ControllerContext context)
            {
                var  properties  =  new  AuthenticationProperties  {  RedirectUri  =
RedirectUri };
                if (UserId != null)
                {
                    properties.Dictionary[XsrfKey] = UserId;
                }

context.HttpContext.GetOwinContext().Authentication.Challenge(properties,
LoginProvider);
            }
        }
        #endregion
    }
}
```

4.3.2. BatchController.cs

```
/using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using Student_Management_System.Models;

namespace Student_Management_System.Controllers
{
    public class BatchController : Controller
    {
        private StudentEntities db = new StudentEntities();

        // GET: Batch
        public ActionResult Index()
        {
            return View(db.batches.ToList());
        }

        // GET: Batch/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            batch batch = db.batches.Find(id);
            if (batch == null)
            {
                return HttpNotFound();
            }
            return View(batch);
        }

        // GET: Batch/Create
        public ActionResult Create()
        {
            return View();
        }

        // POST: Batch/Create
        // To protect from overposting attacks, please enable the specific
properties you want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = "id,batch1,year")] batch batch)
        {
            if (ModelState.IsValid)
            {
                db.batches.Add(batch);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            return View(batch);
        }

        // GET: Batch/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            batch batch = db.batches.Find(id);
            if (batch == null)
            {
                return HttpNotFound();
            }
            return View(batch);
        }

        // POST: Batch/Edit/5
        // To protect from overposting attacks, please enable the specific
properties you want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include = "id,batch1,year")] batch batch)
        {
            if (ModelState.IsValid)
            {
                db.Entry(batch).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(batch);
        }

        // GET: Batch/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            batch batch = db.batches.Find(id);
            if (batch == null)
            {
                return HttpNotFound();
            }
            return View(batch);
        }

        // POST: Batch/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            batch batch = db.batches.Find(id);
            db.batches.Remove(batch);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

4.3.3. CourseController.cs

```
/using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using Student_Management_System.Models;

namespace Student_Management_System.Controllers
{
    public class CourseController : Controller
    {
        private StudentEntities db = new StudentEntities();

        // GET: Course
        public ActionResult Index()
        {
            return View(db.courses.ToList());
        }

        // GET: Course/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            course course = db.courses.Find(id);
            if (course == null)
            {
                return HttpNotFound();
            }
            return View(course);
        }

        // GET: Course/Create
        public ActionResult Create()
        {
            return View();
        }

        // POST: Course/Create
        // To protect from overposting attacks, please enable the specific 
properties you want to bind to, for 
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = "id,course1,duration")] course 
course)
        {
            if (ModelState.IsValid)
            {
                db.courses.Add(course);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            return View(course);
        }

        // GET: Course/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            course course = db.courses.Find(id);
            if (course == null)
            {
                return HttpNotFound();
            }
            return View(course);
        }

        // POST: Course/Edit/5
        // To protect from overposting attacks, please enable the specific 
properties you want to bind to, for 
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include = "id,course1,duration")] course 
course)
        {
            if (ModelState.IsValid)
            {
                db.Entry(course).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(course);
        }

        // GET: Course/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            course course = db.courses.Find(id);
            if (course == null)
            {
                return HttpNotFound();
            }
            return View(course);
        }

        // POST: Course/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            course course = db.courses.Find(id);
            db.courses.Remove(course);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

4.3.4. HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Student_Management_System.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult About()
        {
            ViewBag.Message = "Your application description page.";

            return View();
        }

        public ActionResult Contact()
        {
            ViewBag.Message = "Your contact page.";

            return View();
        }
    }
}
```

4.3.5. LoginController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Data.Entity;
using System.Data.Entity.Infrastructure;
using Student_Management_System.Models;
namespace Student_Management_System.Controllers
{
    public class LoginController : Controller
    {
        StudentEntities db = new StudentEntities();
        // GET: Login
        public ActionResult Index()
        {
            return View();
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Index(user objchk)
        {
            if (ModelState.IsValid)
            {
                using (StudentEntities db = new StudentEntities())
                {
            var obj = db.users.Where(a => a.username.Equals(objchk.username) &&
a.password.Equals(objchk.password)).FirstOrDefault();
            if (obj != null)
            {
                Session["userID"] = obj.id.ToString();
                Session["UserName"] = obj.username.ToString();
                return RedirectToAction("Index", "Home");
            }
            else
            {
                        ModelState.AddModelError("", "The username or Password is
Incorrect");
            }
             }
           }
            return View();
        }
        public ActionResult Logout()
        {
            Session.Clear();
            return RedirectToAction("Index", "Login");
        }
    }
}
```

4.3.6. ManageController.cs

```
/

using System;

using System.Linq;

using System.Threading.Tasks;

using System.Web;

using System.Web.Mvc;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.Owin;

using Microsoft.Owin.Security;

using Student_Management_System.Models;


namespace Student_Management_System.Controllers

{

    [Authorize]

    public class ManageController : Controller

    {

        private ApplicationSignInManager _signInManager;

        private ApplicationUserManager _userManager;


        public ManageController()

        {

        }


        public         ManageController(ApplicationUserManager          userManager,
ApplicationSignInManager signInManager)
```

```
        {
            UserManager = userManager;
            SignInManager = signInManager;
        }

        public ApplicationSignInManager SignInManager
        {
            get
            {
                return                       _signInManager                        ??
HttpContext.GetOwinContext().Get<ApplicationSignInManager>();
            }
            private set
            {
                _signInManager = value;
            }
        }

        public ApplicationUserManager UserManager
        {
            get
            {
                return                        _userManager                         ??
HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();
            }
            private set
            {
                _userManager = value;
            }
        }
        //
        // GET: /Manage/Index
        public async Task<ActionResult> Index(ManageMessageId? message)
        {
            ViewBag.StatusMessage =
                message == ManageMessageId.ChangePasswordSuccess ? "Your password has
been changed."
                : message == ManageMessageId.SetPasswordSuccess ? "Your password has
been set."
                : message == ManageMessageId.SetTwoFactorSuccess ? "Your two-factor
authentication provider has been set."
                : message == ManageMessageId.Error ? "An error has occurred."
                : message == ManageMessageId.AddPhoneSuccess ? "Your phone number was
added."
                : message == ManageMessageId.RemovePhoneSuccess ? "Your phone number
was removed."
                : "";

            var userId = User.Identity.GetUserId();
            var model = new IndexViewModel
            {
                HasPassword = HasPassword(),
                PhoneNumber = await UserManager.GetPhoneNumberAsync(userId),
                TwoFactor = await UserManager.GetTwoFactorEnabledAsync(userId),
                Logins = await UserManager.GetLoginsAsync(userId),
                BrowserRemembered = await
AuthenticationManager.TwoFactorBrowserRememberedAsync(userId)
            };
            return View(model);
        }

        //
        // POST: /Manage/RemoveLogin
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> RemoveLogin(string loginProvider, string
providerKey)
        {
            ManageMessageId? message;
            var result = await
UserManager.RemoveLoginAsync(User.Identity.GetUserId(), new
UserLoginInfo(loginProvider, providerKey));
            if (result.Succeeded)
            {
                var user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());
                if (user != null)
                {
                    await SignInManager.SignInAsync(user, isPersistent: false,
rememberBrowser: false);
                }
                message = ManageMessageId.RemoveLoginSuccess;
            }
            else
            {
                message = ManageMessageId.Error;
            }
            return RedirectToAction("ManageLogins", new { Message = message });
        }

        //
        // GET: /Manage/AddPhoneNumber
        public ActionResult AddPhoneNumber()
        {
            return View();
        }

        //
        // POST: /Manage/AddPhoneNumber
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> AddPhoneNumber(AddPhoneNumberViewModel model)
        {
            if (!ModelState.IsValid)
            {
                return View(model);
            }
            // Generate the token and send it
            var code = await
UserManager.GenerateChangePhoneNumberTokenAsync(User.Identity.GetUserId(),
model.Number);
            if (UserManager.SmsService != null)
            {
                var message = new IdentityMessage
                {
                    Destination = model.Number,
                    Body = "Your security code is: " + code
                };
                await UserManager.SmsService.SendAsync(message);
            }
            return RedirectToAction("VerifyPhoneNumber", new { PhoneNumber =
model.Number });
        }

        //
        // POST: /Manage/EnableTwoFactorAuthentication
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> EnableTwoFactorAuthentication()
        {
            await    UserManager.SetTwoFactorEnabledAsync(User.Identity.GetUserId(),
true);
            var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user != null)
            {
                await    SignInManager.SignInAsync(user,    isPersistent:    false,
rememberBrowser: false);
            }
            return RedirectToAction("Index", "Manage");
        }

        //
        // POST: /Manage/DisableTwoFactorAuthentication
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> DisableTwoFactorAuthentication()
        {
            await    UserManager.SetTwoFactorEnabledAsync(User.Identity.GetUserId(),
false);
            var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user != null)
            {
                await    SignInManager.SignInAsync(user,    isPersistent:    false,
rememberBrowser: false);
            }
            return RedirectToAction("Index", "Manage");
        }

        //
        // GET: /Manage/VerifyPhoneNumber
        public async Task<ActionResult> VerifyPhoneNumber(string phoneNumber)
        {
            var code = await
UserManager.GenerateChangePhoneNumberTokenAsync(User.Identity.GetUserId(),
phoneNumber);
            // Send an SMS through the SMS provider to verify the phone number
            return phoneNumber == null ? View("Error") : View(new
VerifyPhoneNumberViewModel { PhoneNumber = phoneNumber });
        }

        //
        // POST: /Manage/VerifyPhoneNumber
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> VerifyPhoneNumber(VerifyPhoneNumberViewModel
model)
        {
            if (!ModelState.IsValid)
            {
                return View(model);
            }
            var result = await
UserManager.ChangePhoneNumberAsync(User.Identity.GetUserId(), model.PhoneNumber,
model.Code);
            if (result.Succeeded)
            {
                var user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());
```

```
                if (user != null)
                {
                    await SignInManager.SignInAsync(user, isPersistent: false,
rememberBrowser: false);
                }
                return RedirectToAction("Index", new { Message =
ManageMessageId.AddPhoneSuccess });
            }
            // If we got this far, something failed, redisplay form
            ModelState.AddModelError("", "Failed to verify phone");
            return View(model);
        }

        //
        // POST: /Manage/RemovePhoneNumber
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> RemovePhoneNumber()
        {
            var result = await
UserManager.SetPhoneNumberAsync(User.Identity.GetUserId(), null);
            if (!result.Succeeded)
            {
                return RedirectToAction("Index", new { Message =
ManageMessageId.Error });
            }
            var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user != null)
            {
                await SignInManager.SignInAsync(user, isPersistent: false,
rememberBrowser: false);
            }
            return RedirectToAction("Index", new { Message =
ManageMessageId.RemovePhoneSuccess });
        }

        //
        // GET: /Manage/ChangePassword
        public ActionResult ChangePassword()
        {
            return View();
        }

        //
        // POST: /Manage/ChangePassword
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> ChangePassword(ChangePasswordViewModel model)
        {
            if (!ModelState.IsValid)
            {
                return View(model);
            }
            var result = await
UserManager.ChangePasswordAsync(User.Identity.GetUserId(), model.OldPassword,
model.NewPassword);
            if (result.Succeeded)
            {
                var user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());
                if (user != null)
                {
                    await SignInManager.SignInAsync(user, isPersistent: false,
rememberBrowser: false);
```

```
                }
                return     RedirectToAction("Index",     new     {     Message     =
ManageMessageId.ChangePasswordSuccess });
            }
            AddErrors(result);
            return View(model);
        }

        //
        // GET: /Manage/SetPassword
        public ActionResult SetPassword()
        {
            return View();
        }

        //
        // POST: /Manage/SetPassword
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> SetPassword(SetPasswordViewModel model)
        {
            if (ModelState.IsValid)
            {
                var                    result                    =                    await
UserManager.AddPasswordAsync(User.Identity.GetUserId(), model.NewPassword);
                if (result.Succeeded)
                {
                    var                  user                   =                   await
UserManager.FindByIdAsync(User.Identity.GetUserId());
                    if (user != null)
                    {
                        await  SignInManager.SignInAsync(user,  isPersistent:  false,
rememberBrowser: false);
                    }
                    return    RedirectToAction("Index",    new    {    Message    =
ManageMessageId.SetPasswordSuccess });
                }
                AddErrors(result);
            }

            // If we got this far, something failed, redisplay form
            return View(model);
        }

        //
        // GET: /Manage/ManageLogins
        public async Task<ActionResult> ManageLogins(ManageMessageId? message)
        {
            ViewBag.StatusMessage =
                message == ManageMessageId.RemoveLoginSuccess ? "The external login
was removed."
                : message == ManageMessageId.Error ? "An error has occurred."
                : "";
            var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user == null)
            {
                return View("Error");
            }
            var                 userLogins                 =                 await
UserManager.GetLoginsAsync(User.Identity.GetUserId());
            var                        otherLogins                               =
AuthenticationManager.GetExternalAuthenticationTypes().Where(auth              =>
userLogins.All(ul => auth.AuthenticationType != ul.LoginProvider)).ToList();
```

```
            ViewBag.ShowRemoveButton = user.PasswordHash != null || userLogins.Count
> 1;
            return View(new ManageLoginsViewModel
            {
                CurrentLogins = userLogins,
                OtherLogins = otherLogins
            });
        }

        //
        // POST: /Manage/LinkLogin
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult LinkLogin(string provider)
        {
            // Request a redirect to the external login provider to link a login for
the current user
            return          new          AccountController.ChallengeResult(provider,
Url.Action("LinkLoginCallback", "Manage"), User.Identity.GetUserId());
        }

        //
        // GET: /Manage/LinkLoginCallback
        public async Task<ActionResult> LinkLoginCallback()
        {
            var                  loginInfo                  =                 await
AuthenticationManager.GetExternalLoginInfoAsync(XsrfKey, User.Identity.GetUserId());
            if (loginInfo == null)
            {
                return    RedirectToAction("ManageLogins",    new    {    Message    =
ManageMessageId.Error });
            }
            var result = await UserManager.AddLoginAsync(User.Identity.GetUserId(),
loginInfo.Login);
            return    result.Succeeded    ?    RedirectToAction("ManageLogins")    :
RedirectToAction("ManageLogins", new { Message = ManageMessageId.Error });
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing && _userManager != null)
            {
                _userManager.Dispose();
                _userManager = null;
            }

            base.Dispose(disposing);
        }

#region Helpers
        // Used for XSRF protection when adding external logins
        private const string XsrfKey = "XsrfId";

        private IAuthenticationManager AuthenticationManager
        {
            get
            {
                return HttpContext.GetOwinContext().Authentication;
            }
        }

        private void AddErrors(IdentityResult result)
        {
```

```
    foreach (var error in result.Errors)
    {
        ModelState.AddModelError("", error);
    }
}

private bool HasPassword()
{
    var user = UserManager.FindById(User.Identity.GetUserId());
    if (user != null)
    {
        return user.PasswordHash != null;
    }
    return false;
}

private bool HasPhoneNumber()
{
    var user = UserManager.FindById(User.Identity.GetUserId());
    if (user != null)
    {
        return user.PhoneNumber != null;
    }
    return false;
}

public enum ManageMessageId
{
    AddPhoneSuccess,
    ChangePasswordSuccess,
            SetTwoFactorSuccess,

            SetPasswordSuccess,

            RemoveLoginSuccess,

            RemovePhoneSuccess,

            Error

        }


#endregion

    }

}
```

4.3.7. RegistrationController.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using Student_Management_System.Models;

namespace Student_Management_System.Controllers
{
    public class RegistrationController : Controller
    {
        private StudentEntities db = new StudentEntities();

        // GET: Registration
        public ActionResult Index()
        {
            var registrations = db.registrations.Include(r => r.batch).Include(r =>
r.course);
            return View(registrations.ToList());
        }

        // GET: Registration/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            registration registration = db.registrations.Find(id);
            if (registration == null)
            {
                return HttpNotFound();
            }
            return View(registration);
        }
        // GET: Registration/Create
        public ActionResult Create()
        {
            ViewBag.barch_id = new SelectList(db.batches, "id", "batch1");
            ViewBag.course_id = new SelectList(db.courses, "id", "course1");
            return View();
        }

        // POST: Registration/Create
        // To protect from overposting attacks, please enable the specific
properties you want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include =
"id,firstname,lastname,nic,course_id,barch_id,telno")] registration registration)
        {
            if (ModelState.IsValid)
            {
                db.registrations.Add(registration);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.barch_id = new SelectList(db.batches, "id", "batch1",
registration.barch_id);
            ViewBag.course_id = new SelectList(db.courses, "id", "course1",
registration.course_id);
            return View(registration);
        }

        // GET: Registration/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            registration registration = db.registrations.Find(id);
            if (registration == null)
            {
                return HttpNotFound();
            }
            ViewBag.barch_id = new SelectList(db.batches, "id", "batch1",
registration.barch_id);
            ViewBag.course_id = new SelectList(db.courses, "id", "course1",
registration.course_id);
            return View(registration);
        }

        // POST: Registration/Edit/5
        // To protect from overposting attacks, please enable the specific
properties you want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include =
"id,firstname,lastname,nic,course_id,barch_id,telno")] registration registration)
        {
            if (ModelState.IsValid)
            {
                db.Entry(registration).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            ViewBag.barch_id = new SelectList(db.batches, "id", "batch1",
registration.barch_id);
            ViewBag.course_id = new SelectList(db.courses, "id", "course1",
registration.course_id);
            return View(registration);
        }

        // GET: Registration/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            registration registration = db.registrations.Find(id);
            if (registration == null)
            {
                return HttpNotFound();
            }
            return View(registration);
        }

        // POST: Registration/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            registration registration = db.registrations.Find(id);
            db.registrations.Remove(registration);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

4.3.8. UserController.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using Student_Management_System.Models;

namespace Student_Management_System.Controllers
{
    public class UserController : Controller
    {
        private StudentEntities db = new StudentEntities();

        // GET: User
        public ActionResult Index()
        {
            return View(db.users.ToList());
        }

        // GET: User/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            user user = db.users.Find(id);
            if (user == null)
            {
                return HttpNotFound();
            }
            return View(user);
        }

        // GET: User/Create
        public ActionResult Create()
        {
            return View();
        }

        // POST: User/Create
        // To protect from overposting attacks, please enable the specific
properties you want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include =
"id,firstname,lastname,username,password")] user user)
        {
            if (ModelState.IsValid)
            {
                db.users.Add(user);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            return View(user);
        }

        // GET: User/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            user user = db.users.Find(id);
            if (user == null)
            {
                return HttpNotFound();
            }
            return View(user);
        }

        // POST: User/Edit/5
        // To protect from overposting attacks, please enable the specific
properties you want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include =
"id,firstname,lastname,username,password")] user user)
        {
            if (ModelState.IsValid)
            {
                db.Entry(user).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(user);
        }

        // GET: User/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            user user = db.users.Find(id);
            if (user == null)
            {
                return HttpNotFound();
            }
            return View(user);
        }

        // POST: User/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            user user = db.users.Find(id);
            db.users.Remove(user);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
```

```
}
```

4.4. Model

4.5. AccountViewModel.cs

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace Student_Management_System.Models
{
    public class ExternalLoginConfirmationViewModel
    {
        [Required]
        [Display(Name = "Email")]
        public string Email { get; set; }
    }

    public class ExternalLoginListViewModel
    {
        public string ReturnUrl { get; set; }
    }

    public class SendCodeViewModel
    {
        public string SelectedProvider { get; set; }
        public ICollection<System.Web.Mvc.SelectListItem> Providers { get; set; }
        public string ReturnUrl { get; set; }
        public bool RememberMe { get; set; }
    }

    public class VerifyCodeViewModel
    {
        [Required]
        public string Provider { get; set; }

        [Required]
        [Display(Name = "Code")]
        public string Code { get; set; }
        public string ReturnUrl { get; set; }

        [Display(Name = "Remember this browser?")]
        public bool RememberBrowser { get; set; }

        public bool RememberMe { get; set; }
    }

    public class ForgotViewModel
    {
        [Required]
        [Display(Name = "Email")]
        public string Email { get; set; }
    }

    public class LoginViewModel
    {
        [Required]
        [Display(Name = "Email")]
        [EmailAddress]
        public string Email { get; set; }
        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [Display(Name = "Remember me?")]
        public bool RememberMe { get; set; }
    }

    public class RegisterViewModel
    {
        [Required]
        [EmailAddress]
        [Display(Name = "Email")]
        public string Email { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters 
long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm password")]
        [Compare("Password", ErrorMessage = "The password and confirmation password 
do not match.")]
        public string ConfirmPassword { get; set; }
    }

    public class ResetPasswordViewModel
    {
        [Required]
        [EmailAddress]
        [Display(Name = "Email")]
        public string Email { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters 
long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm password")]
        [Compare("Password", ErrorMessage = "The password and confirmation password 
do not match.")]
        public string ConfirmPassword { get; set; }

        public string Code { get; set; }
    }

    public class ForgotPasswordViewModel
    {
        [Required]
        [EmailAddress]
        [Display(Name = "Email")]
        public string Email { get; set; }
    }}
```

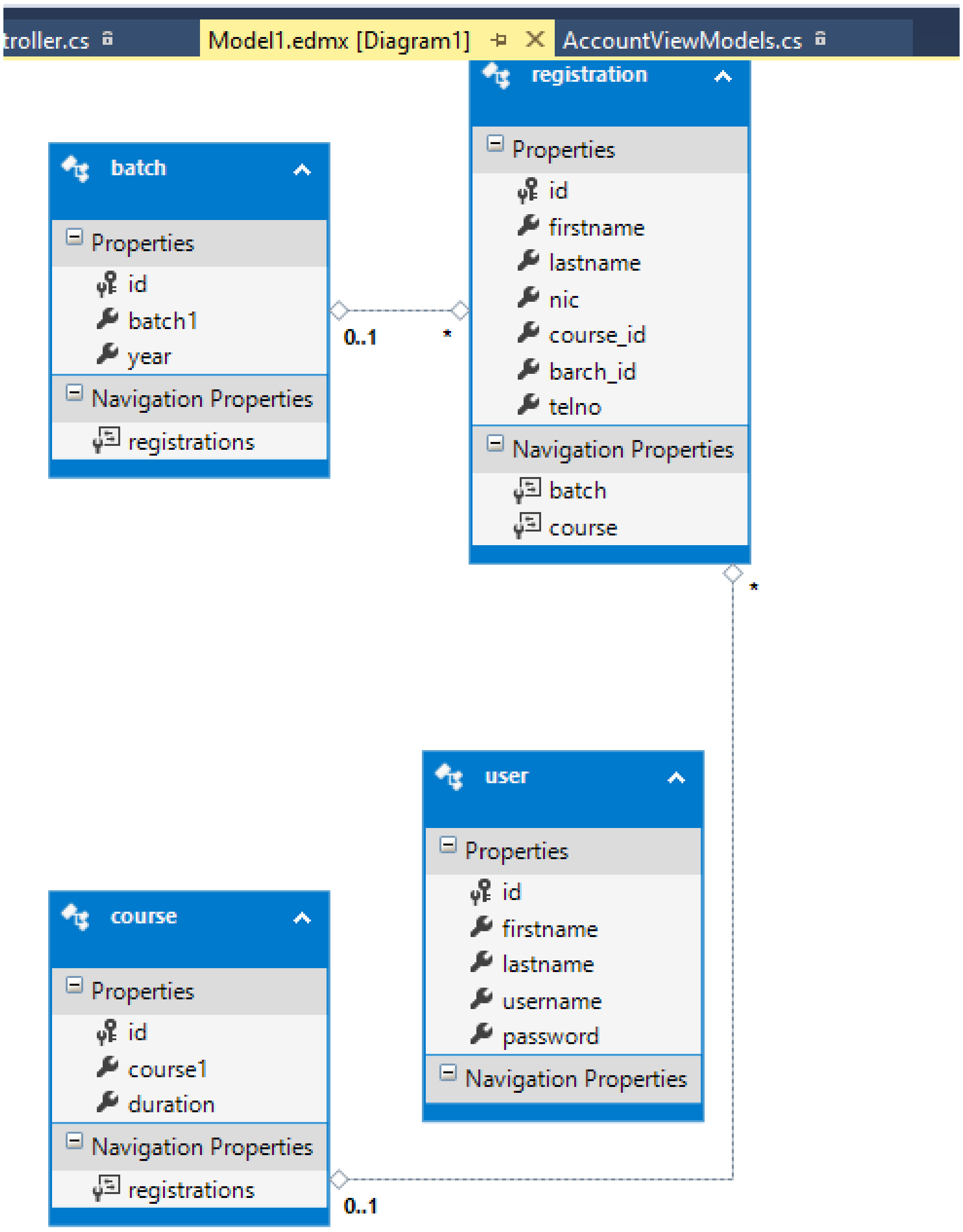
4.6. IdentityModel.cs

```
using System.Data.Entity;
using System.Security.Claims;
using System.Threading.Tasks;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;

namespace Student_Management_System.Models
{
    // You can add profile data for the user by adding more properties to your 
ApplicationUser class, please visit http://go.microsoft.com/fwlink/?LinkID=317594 to 
learn more.
    public class ApplicationUser : IdentityUser
    {
        public async Task<ClaimsIdentity> 
GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)
        {
            // Note the authenticationType must match the one defined in 
CookieAuthenticationOptions.AuthenticationType
            var userIdentity = await manager.CreateIdentityAsync(this, 
DefaultAuthenticationTypes.ApplicationCookie);
            // Add custom user claims here
            return userIdentity;
        }
    }

    public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
    {
        public ApplicationDbContext()
            : base("DefaultConnection", throwIfV1Schema: false)
        {
        }

        public static ApplicationDbContext Create()
        {
            return new ApplicationDbContext();
        }
    }
}
```

4.7. ManageViewModel.cs

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNet.Identity;
using Microsoft.Owin.Security;

namespace Student_Management_System.Models
{
    public class IndexViewModel
    {
        public bool HasPassword { get; set; }
        public IList<UserLoginInfo> Logins { get; set; }
        public string PhoneNumber { get; set; }
        public bool TwoFactor { get; set; }
        public bool BrowserRemembered { get; set; }
    }

    public class ManageLoginsViewModel
    {
        public IList<UserLoginInfo> CurrentLogins { get; set; }
        public IList<AuthenticationDescription> OtherLogins { get; set; }
    }

    public class FactorViewModel
    {
        public string Purpose { get; set; }
    }

    public class SetPasswordViewModel
    {
        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters
long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "New password")]
        public string NewPassword { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm new password")]
        [Compare("NewPassword", ErrorMessage = "The new password and confirmation
password do not match.")]
        public string ConfirmPassword { get; set; }
    }

    public class ChangePasswordViewModel
    {
        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Current password")]
        public string OldPassword { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters
long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "New password")]
        public string NewPassword { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm new password")]
        [Compare("NewPassword", ErrorMessage = "The new password and confirmation
password do not match.")]
        public string ConfirmPassword { get; set; }
    }

    public class AddPhoneNumberViewModel
    {
        [Required]
        [Phone]
        [Display(Name = "Phone Number")]
        public string Number { get; set; }
    }

    public class VerifyPhoneNumberViewModel
    {
        [Required]
        [Display(Name = "Code")]
        public string Code { get; set; }

        [Required]
        [Phone]
        [Display(Name = "Phone Number")]
        public string PhoneNumber { get; set; }
    }

    public class ConfigureTwoFactorViewModel
    {
        public string SelectedProvider { get; set; }
        public ICollection<System.Web.Mvc.SelectListItem> Providers { get; set; }
    }
}
```

4.8. Model.edmx



4.9. Views:

4.10. Account:

4.10.1. ExternalLogins

```
@model Student_Management_System.Models.ExternalLoginListViewModel
@using Microsoft.Owin.Security

<h4>Use another service to log in.</h4>
```

```
<hr />
@{
    var loginProviders =
Context.GetOwinContext().Authentication.GetExternalAuthenticationTypes();
    if (loginProviders.Count() == 0) {
        <div>
            <p>
                There are no external authentication services configured. See <a
href="http://go.microsoft.com/fwlink/?LinkId=403804">this article</a>
                for details on setting up this ASP.NET application to support
logging in via external services.
            </p>
        </div>
    }
    else {
        using (Html.BeginForm("ExternalLogin", "Account", new { ReturnUrl =
Model.ReturnUrl })) {
            @Html.AntiForgeryToken()
            <div id="socialLoginList">
                <p>
                    @foreach (AuthenticationDescription p in loginProviders) {
                        <button type="submit" class="btn btn-default"
id="@p.AuthenticationType" name="provider" value="@p.AuthenticationType" title="Log
in using your @p.Caption account">@p.AuthenticationType</button>
                    }
                </p>
            </div>
        }
    }
}
```

4.10.2. ConfirmEmail

```
@{
    ViewBag.Title = "Confirm Email";
}

<h2>@ViewBag.Title.</h2>
<div>
    <p>
        Thank you for confirming your email. Please @Html.ActionLink("Click here to
Log in", "Login", "Account", routeValues: null, htmlAttributes: new { id =
"loginLink" })
    </p>
</div>
```

4.10.3. ExternalLoginConfirmation

```
@model Student_Management_System.Models.ExternalLoginConfirmationViewModel
@{
    ViewBag.Title = "Register";
}
<h2>@ViewBag.Title.</h2>
<h3>Associate your @ViewBag.LoginProvider account.</h3>
```

```
@using (Html.BeginForm("ExternalLoginConfirmation", "Account", new { ReturnUrl =
ViewBag.ReturnUrl }, FormMethod.Post, new { @class = "form-horizontal", role =
"form" }))
{
    @Html.AntiForgeryToken()

    <h4>Association Form</h4>
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })
    <p class="text-info">
        You've successfully authenticated with
<strong>@ViewBag.LoginProvider</strong>.
        Please enter a user name for this site below and click the Register button
to finish
        logging in.
    </p>
    <div class="form-group">
        @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
            @Html.ValidationMessageFor(m => m.Email, "", new { @class = "text-
danger" })
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-default" value="Register" />
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

4.10.4. ExternalLoginFailure

```
@{
    ViewBag.Title = "Login Failure";
}

<hgroup>
    <h2>@ViewBag.Title.</h2>
    <h3 class="text-danger">Unsuccessful login with service.</h3>
</hgroup>
```

4.10.5. ForgotPassword

```
@model Student_Management_System.Models.ForgotPasswordViewModel
@{
    ViewBag.Title = "Forgot your password?";
}

<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("ForgotPassword", "Account", FormMethod.Post, new { @class =
"form-horizontal", role = "form" }))
```

```
{
    @Html.AntiForgeryToken()
    <h4>Enter your email.</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-default" value="Email Link" />
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

4.10.6. ForgotPasswordConfirmation

```
@{
    ViewBag.Title = "Forgot Password Confirmation";
}

<hgroup class="title">
    <h1>@ViewBag.Title.</h1>
</hgroup>
<div>
    <p>
        Please check your email to reset your password.
    </p>
</div>
```

4.10.7. Login

```
@using Student_Management_System.Models
@model LoginViewModel
@{
    ViewBag.Title = "Log in";
}

<h2>@ViewBag.Title.</h2>
<div class="row">
    <div class="col-md-8">
        <section id="loginForm">
            @using (Html.BeginForm("Login", "Account", new { ReturnUrl =
ViewBag.ReturnUrl }, FormMethod.Post, new { @class = "form-horizontal", role =
"form" }))
            {
                @Html.AntiForgeryToken()
                <h4>Use a local account to log in.</h4>
                <hr />
                @Html.ValidationSummary(true, "", new { @class = "text-danger" })
                <div class="form-group">
                    @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-
label" })
                    <div class="col-md-10">
                        @Html.TextBoxFor(m => m.Email, new { @class = "form-control"
})
                        @Html.ValidationMessageFor(m => m.Email, "", new { @class =
"text-danger" })
                    </div>
                </div>
                <div class="form-group">
                    @Html.LabelFor(m => m.Password, new { @class = "col-md-2
control-label" })
                    <div class="col-md-10">
                        @Html.PasswordFor(m => m.Password, new { @class = "form-
control" })
                        @Html.ValidationMessageFor(m => m.Password, "", new { @class
= "text-danger" })
                    </div>
                </div>
                <div class="form-group">
                    <div class="col-md-offset-2 col-md-10">
                        <div class="checkbox">
                            @Html.CheckBoxFor(m => m.RememberMe)
                            @Html.LabelFor(m => m.RememberMe)
                        </div>
                    </div>
                </div>
                <div class="form-group">
                    <div class="col-md-offset-2 col-md-10">
                        <input type="submit" value="Log in" class="btn btn-default"
/>
                    </div>
                </div>
                <p>
                    @Html.ActionLink("Register as a new user", "Register")
                </p>
                @* Enable this once you have account confirmation enabled for
password reset functionality
                    <p>
                        @Html.ActionLink("Forgot your password?", "ForgotPassword")
                    </p>*@
            }
        </section>
    </div>
    <div class="col-md-4">
        <section id="socialLoginForm">
            @Html.Partial("_ExternalLoginsListPartial", new
ExternalLoginListViewModel { ReturnUrl = ViewBag.ReturnUrl })
        </section>
    </div>
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

4.10.8. Register

```
@model Student_Management_System.Models.RegisterViewModel
@{
    ViewBag.Title = "Register";
}

<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("Register", "Account", FormMethod.Post, new { @class = "form-
horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()
    <h4>Create a new account.</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.Password, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.Password, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.ConfirmPassword, new { @class = "col-md-2 control-
label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.ConfirmPassword, new { @class = "form-control"
})
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-default" value="Register" />
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

4.10.9. ResetPassword

```
@model Student_Management_System.Models.RegisterViewModel
@{
    ViewBag.Title = "Register";
}

<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("Register", "Account", FormMethod.Post, new { @class = "form-
horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()
    <h4>Create a new account.</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.Password, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.Password, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.ConfirmPassword, new { @class = "col-md-2 control-
label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.ConfirmPassword, new { @class = "form-control"
})
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-default" value="Register" />
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

4.10.10. ResetPasswordConfirmation

```
@{
    ViewBag.Title = "Reset password confirmation";
}

<hgroup class="title">
    <h1>@ViewBag.Title.</h1>
</hgroup>
<div>
    <p>
        Your password has been reset. Please @Html.ActionLink("click here to log
in", "Login", "Account", routeValues: null, htmlAttributes: new { id = "loginLink"
})
    </p>
</div>
```

4.10.11. SendCode

```
@model Student_Management_System.Models.SendCodeViewModel
@{
    ViewBag.Title = "Send";
}

<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("SendCode", "Account", new { ReturnUrl = Model.ReturnUrl },
FormMethod.Post, new { @class = "form-horizontal", role = "form" })) {
    @Html.AntiForgeryToken()
    @Html.Hidden("rememberMe", @Model.RememberMe)
    <h4>Send verification code</h4>
    <hr />
    <div class="row">
        <div class="col-md-8">
            Select Two-Factor Authentication Provider:
            @Html.DropDownListFor(model => model.SelectedProvider, Model.Providers)
            <input type="submit" value="Submit" class="btn btn-default" />
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

4.10.12. VerifyCode

```
@model Student_Management_System.Models.VerifyCodeViewModel
@{
    ViewBag.Title = "Verify";
}

<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("VerifyCode", "Account", new { ReturnUrl = Model.ReturnUrl },
FormMethod.Post, new { @class = "form-horizontal", role = "form" })) {
    @Html.AntiForgeryToken()
    @Html.Hidden("provider", @Model.Provider)
    @Html.Hidden("rememberMe", @Model.RememberMe)
    <h4>Enter verification code</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.Code, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Code, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <div class="checkbox">
                @Html.CheckBoxFor(m => m.RememberBrowser)
                @Html.LabelFor(m => m.RememberBrowser)
            </div>
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-default" value="Submit" />
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

4.11. Batch

4.11.1. Create

```
@model Student_Management_System.Models.batch

@{
    ViewBag.Title = "Create";
}

<h2>Create</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>batch</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.batch1, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.batch1, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.batch1, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.year, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.year, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.year, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

```
@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

4.11.2. Delete

```
@model Student_Management_System.Models.batch

@{
    ViewBag.Title = "Delete";
}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>batch</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.batch1)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.batch1)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.year)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.year)
        </dd>

    </dl>

    @using (Html.BeginForm()) {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-default" /> |
            @Html.ActionLink("Back to List", "Index")
        </div>
    }
</div>
```

4.11.3. Details

```
@model Student_Management_System.Models.batch

@{
    ViewBag.Title = "Details";
}

<h2>Details</h2>
```

```
<div>
    <h4>batch</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.batch1)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.batch1)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.year)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.year)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.id }) |
    @Html.ActionLink("Back to List", "Index")
</p>
```

4.11.4. Edit

```
@model Student_Management_System.Models.batch

@{
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>batch</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.id)

        <div class="form-group">
            @Html.LabelFor(model => model.batch1, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.batch1, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.batch1, "", new { @class =
"text-danger" })
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.year, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.year, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.year, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

4.11.5. Index

```
@model IEnumerable<Student_Management_System.Models.batch>

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.batch1)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.year)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.batch1)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.year)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.id }) |
            @Html.ActionLink("Details", "Details", new { id=item.id }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.id })
        </td>
    </tr>
}

</table>
```

4.12. Course

4.12.1. Create

```
@model Student_Management_System.Models.course

@{
    ViewBag.Title = "Create";
}

<h2>Create</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>course</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.course1, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.course1, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.course1, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.duration, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.duration, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.duration, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

4.12.2. Delete

```
@model Student_Management_System.Models.course

@{
    ViewBag.Title = "Delete";
}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>course</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.course1)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.course1)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.duration)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.duration)
        </dd>

    </dl>

    @using (Html.BeginForm()) {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-default" /> |
            @Html.ActionLink("Back to List", "Index")
        </div>
    }
</div>
```

4.12.3. Details

```
@model Student_Management_System.Models.course
```

```
@{
    ViewBag.Title = "Details";
}

<h2>Details</h2>

<div>
    <h4>course</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.course1)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.course1)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.duration)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.duration)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.id }) |
    @Html.ActionLink("Back to List", "Index")
</p>
```

4.12.4. Edit

```
@model Student_Management_System.Models.course

@{
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>course</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.id)

        <div class="form-group">
            @Html.LabelFor(model => model.course1, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
```

```
                @Html.EditorFor(model => model.course1, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.course1, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.duration, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.duration, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.duration, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

4.12.5. Index

```
@model IEnumerable<Student_Management_System.Models.course>

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.course1)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.duration)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.course1)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.duration)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.id }) |
            @Html.ActionLink("Details", "Details", new { id=item.id }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.id })
        </td>
    </tr>
}

</table>
```

5. HOME

5.1.1. About

```
@{
    ViewBag.Title = "About";
}
<h2>@ViewBag.Title.</h2>
<h3>@ViewBag.Message</h3>

<p>Use this area to provide additional information.</p>
```

5.1.2. Contact

```
@{
    ViewBag.Title = "Contact";
}
<h2>@ViewBag.Title.</h2>
<h3>@ViewBag.Message</h3>

<address>
    One Microsoft Way<br />
    Redmond, WA 98052-6399<br />
    <abbr title="Phone">P:</abbr>
    425.555.0100
</address>

<address>
    <strong>Support:</strong>   <a
href="mailto:Support@example.com">Support@example.com</a><br />
    <strong>Marketing:</strong> <a
href="mailto:Marketing@example.com">Marketing@example.com</a>
</address>
```

5.1.3. Index

```
@{
    ViewBag.Title = "Home Page";
}

<div class="jumbotron">
    <h1>ASP.NET</h1>
    <p class="lead">ASP.NET is a free web framework for building great Web sites and 
Web applications using HTML, CSS and JavaScript.</p>
    <p><a href="http://asp.net" class="btn btn-primary btn-lg">Learn more 
&raquo;</a></p>
</div>

<div class="row">
    <div class="col-md-4">
        <h2>Getting started</h2>
        <p>
            ASP.NET MVC gives you a powerful, patterns-based way to build dynamic 
websites that
            enables a clean separation of concerns and gives you full control over 
markup
            for enjoyable, agile development.
        </p>
        <p><a class="btn btn-default" 
href="http://go.microsoft.com/fwlink/?LinkId=301865">Learn more &raquo;</a></p>
    </div>
    <div class="col-md-4">
        <h2>Get more libraries</h2>
        <p>NuGet is a free Visual Studio extension that makes it easy to add, 
remove, and update libraries and tools in Visual Studio projects.</p>
        <p><a class="btn btn-default" 
href="http://go.microsoft.com/fwlink/?LinkId=301866">Learn more &raquo;</a></p>
    </div>
    <div class="col-md-4">
        <h2>Web Hosting</h2>
        <p>You can easily find a web hosting company that offers the right mix of 
features and price for your applications.</p>
        <p><a class="btn btn-default" 
href="http://go.microsoft.com/fwlink/?LinkId=301867">Learn more &raquo;</a></p>
    </div>
</div>
```

5.2. Login

5.2.1. Index.cs.html

```
@model Student_Management_System.Models.user

@{
    ViewBag.Title = "Create";
}

<h2>Create</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>user</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.username, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.username, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.username, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.password, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.PasswordFor(model => model.password, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.password, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Login" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

6. MANAGE

6.1.1. AddPhoneNumber

```
@model Student_Management_System.Models.AddPhoneNumberViewModel
@{
    ViewBag.Title = "Phone Number";
}

<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("AddPhoneNumber", "Manage", FormMethod.Post, new { @class =
"form-horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()
    <h4>Add a phone number</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.Number, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Number, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-default" value="Submit" />
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

6.1.2. ChangePassword

```
@model Student_Management_System.Models.ChangePasswordViewModel
@{
    ViewBag.Title = "Change Password";
}

<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("ChangePassword", "Manage", FormMethod.Post, new { @class =
"form-horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()
    <h4>Change Password Form</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.OldPassword, new { @class = "col-md-2 control-label"
})
        <div class="col-md-10">
            @Html.PasswordFor(m => m.OldPassword, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.NewPassword, new { @class = "col-md-2 control-label"
})
        <div class="col-md-10">
            @Html.PasswordFor(m => m.NewPassword, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.ConfirmPassword, new { @class = "col-md-2 control-
label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.ConfirmPassword, new { @class = "form-control"
})
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Change password" class="btn btn-default" />
        </div>
    </div>
}
@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

6.1.3. Index.cs.html

```
@model Student_Management_System.Models.IndexViewModel
@{
    ViewBag.Title = "Manage";
}

<h2>@ViewBag.Title.</h2>

<p class="text-success">@ViewBag.StatusMessage</p>
<div>
    <h4>Change your account settings</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>Password:</dt>
        <dd>
            [
            @if (Model.HasPassword)
            {
                @Html.ActionLink("Change your password", "ChangePassword")
            }
            else
            {
                @Html.ActionLink("Create", "SetPassword")
            }
            ]
        </dd>
        <dt>External Logins:</dt>
        <dd>
            @Model.Logins.Count [
            @Html.ActionLink("Manage", "ManageLogins") ]
        </dd>
        @*
            Phone Numbers can used as a second factor of verification in a two-
factor authentication system.

             See <a href="http://go.microsoft.com/fwlink/?LinkId=403804">this
article</a>
                for details on setting up this ASP.NET application to support two-
factor authentication using SMS.

             Uncomment the following block after you have set up two-factor
authentication
        *@
        @*
            <dt>Phone Number:</dt>
            <dd>
                @(Model.PhoneNumber ?? "None")
                @if (Model.PhoneNumber != null)
                {
                    <br />
                    <text>[&nbsp;&nbsp;@Html.ActionLink("Change",
"AddPhoneNumber")&nbsp;&nbsp;]</text>
```

```
                    using (Html.BeginForm("RemovePhoneNumber", "Manage",
FormMethod.Post, new { @class = "form-horizontal", role = "form" }))
                    {
                        @Html.AntiForgeryToken()
                        <text>[<input type="submit" value="Remove" class="btn-link"
/>]</text>
                    }
                }
                else
                {
                    <text>[&nbsp;&nbsp;@Html.ActionLink("Add", "AddPhoneNumber")
                }
            </dd>
        *@
        <dt>Two-Factor Authentication:</dt>
        <dd>
            <p>
                There are no two-factor authentication providers configured. See <a
href="http://go.microsoft.com/fwlink/?LinkId=403804">this article</a>
                for details on setting up this ASP.NET application to support two-
factor authentication.
            </p>
            @*@if (Model.TwoFactor)
                {
                    using (Html.BeginForm("DisableTwoFactorAuthentication",
"Manage", FormMethod.Post, new { @class = "form-horizontal", role = "form" }))
                    {
                        @Html.AntiForgeryToken()
                        <text>Enabled
                        <input type="submit" value="Disable" class="btn btn-link" />
                        </text>
                    }
                }
                else
                {
                    using (Html.BeginForm("EnableTwoFactorAuthentication", "Manage",
FormMethod.Post, new { @class = "form-horizontal", role = "form" }))
                    {
                        @Html.AntiForgeryToken()
                        <text>Disabled
                        <input type="submit" value="Enable" class="btn btn-link" />
                        </text>
                    }
                }*@
        </dd>
    </dl>
</div>
```

6.1.4. ManageLogins.cshtml

```
@model Student_Management_System.Models.ManageLoginsViewModel
@using Microsoft.Owin.Security
@{
    ViewBag.Title = "Manage your external logins";
}

<h2>@ViewBag.Title.</h2>

<p class="text-success">@ViewBag.StatusMessage</p>
```

```
@{
    var loginProviders =
Context.GetOwinContext().Authentication.GetExternalAuthenticationTypes();
    if (loginProviders.Count() == 0) {
        <div>
            <p>
                There are no external authentication services configured. See <a
href="http://go.microsoft.com/fwlink/?LinkId=313242">this article</a>
                for details on setting up this ASP.NET application to support
logging in via external services.
            </p>
        </div>
    }
    else
    {
        if (Model.CurrentLogins.Count > 0)
        {
            <h4>Registered Logins</h4>
            <table class="table">
                <tbody>
                    @foreach (var account in Model.CurrentLogins)
                    {
                        <tr>
                            <td>@account.LoginProvider</td>
                            <td>
                                @if (ViewBag.ShowRemoveButton)
                                {
                                    using (Html.BeginForm("RemoveLogin", "Manage"))
                                    {
                                        @Html.AntiForgeryToken()
                                        <div>
                                            @Html.Hidden("loginProvider",
account.LoginProvider)
                                            @Html.Hidden("providerKey",
account.ProviderKey)
                                            <input type="submit" class="btn btn-
default" value="Remove" title="Remove this @account.LoginProvider login from your
account" />
                                        </div>
                                    }
                                }
                                else
                                {
                                    @: &nbsp;
                                }
                            </td>
                        </tr>
                    }
                </tbody>
            </table>
        }
        if (Model.OtherLogins.Count > 0)
        {
            using (Html.BeginForm("LinkLogin", "Manage"))
            {
                @Html.AntiForgeryToken()
                <div id="socialLoginList">
                <p>
                    @foreach (AuthenticationDescription p in Model.OtherLogins)
                    {
                        <button type="submit" class="btn btn-default"
id="@p.AuthenticationType" name="provider" value="@p.AuthenticationType" title="Log
in using your @p.Caption account">@p.AuthenticationType</button>
                    }
                </p>
                </div>
            }
        }
    }
}
```

6.1.5. SetPassword.cshtml

```
@model Student_Management_System.Models.SetPasswordViewModel
@{
    ViewBag.Title = "Create Password";
}

<h2>@ViewBag.Title.</h2>
<p class="text-info">
    You do not have a local username/password for this site. Add a local
    account so you can log in without an external login.
</p>

@using (Html.BeginForm("SetPassword", "Manage", FormMethod.Post, new { @class =
"form-horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()

    <h4>Create Local Login</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.NewPassword, new { @class = "col-md-2 control-label"
})
        <div class="col-md-10">
            @Html.PasswordFor(m => m.NewPassword, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.ConfirmPassword, new { @class = "col-md-2 control-
label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.ConfirmPassword, new { @class = "form-control"
})
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Set password" class="btn btn-default" />
        </div>
    </div>
}
@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

6.1.6. VerifyPhoneNumber.cshtml

```
@model Student_Management_System.Models.VerifyPhoneNumberViewModel
@{
    ViewBag.Title = "Verify Phone Number";
}

<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("VerifyPhoneNumber", "Manage", FormMethod.Post, new { @class
= "form-horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()
    @Html.Hidden("phoneNumber", @Model.PhoneNumber)
    <h4>Enter verification code</h4>
    <h5>@ViewBag.Status</h5>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.Code, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Code, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-default" value="Submit" />
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

6.2. Registration:

6.2.1. Create

```
@model Student_Management_System.Models.registration

@{
    ViewBag.Title = "Create";
}

<h2>Create</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>registration</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.firstname, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
```

```
                @Html.EditorFor(model => model.firstname, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.firstname, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.lastname, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.lastname, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.lastname, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.nic, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.nic, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.nic, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.course_id, "course_id", htmlAttributes:
new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("course_id", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.course_id, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.barch_id, "barch_id", htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("barch_id", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.barch_id, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.telno, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.telno, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.telno, "", new { @class =
"text-danger" })
            </div>
        </div>
        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

6.2.2. Delete

```
@model Student_Management_System.Models.registration

@{
    ViewBag.Title = "Delete";
}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>registration</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.firstname)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.firstname)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.lastname)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.lastname)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.nic)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.nic)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.telno)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.telno)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.batch.batch1)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.batch.batch1)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.course.course1)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.course.course1)
        </dd>

    </dl>

    @using (Html.BeginForm()) {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-default" /> |
            @Html.ActionLink("Back to List", "Index")
        </div>
    }
</div>
```

6.2.3. Details

```
@model Student_Management_System.Models.registration

@{
    ViewBag.Title = "Details";
}

<h2>Details</h2>

<div>
    <h4>registration</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.firstname)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.firstname)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.lastname)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.lastname)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.nic)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.nic)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.telno)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.telno)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.batch.batch1)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.batch.batch1)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.course.course1)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.course.course1)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.id }) |
    @Html.ActionLink("Back to List", "Index")
</p>
```

6.2.4. Edit

```
@model Student_Management_System.Models.registration

@{
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
```

```
        <h4>registration</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.id)

        <div class="form-group">
            @Html.LabelFor(model => model.firstname, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.firstname, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.firstname, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.lastname, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.lastname, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.lastname, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.nic, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.nic, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.nic, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.course_id, "course_id", htmlAttributes:
new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("course_id", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.course_id, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.barch_id, "barch_id", htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("barch_id", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.barch_id, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
```

```
            @Html.LabelFor(model => model.telno, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.telno, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.telno, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

6.2.5. Index

```
@model IEnumerable<Student_Management_System.Models.registration>

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.firstname)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.lastname)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.nic)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.telno)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.batch.batch1)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.course.course1)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.firstname)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.lastname)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.nic)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.telno)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.batch.batch1)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.course.course1)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.id }) |
            @Html.ActionLink("Details", "Details", new { id=item.id }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.id })
        </td>
    </tr>
}

</table>
```

6.3. Shared

6.3.1. Layout.cshtml

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")

</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse"
data-target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("Application name", "Index", "Home", new { area =
"" }, new { @class = "navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("Course", "Index", "Course")</li>
                    <li>@Html.ActionLink("Batch", "Index", "Batch")</li>
                    <li>@Html.ActionLink("Registration", "Index",
"Registration")</li>
                    <li>@Html.ActionLink("User", "Index", "User")</li>
                </ul>
                <ul class="nav navbar-right" styyle="color=white">
                    @if (Session["userid"] != null)
                    {
                        <h4 style="color: white">Hello @Session["UserName"]</h4>
                        @Html.ActionLink("Logout","Logout","Login")
                    }
                </ul>
                @Html.Partial("_LoginPartial")
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>
        </footer>
    </div>

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>
```

6.3.2. LoginPartial.cshtml

```
@using Microsoft.AspNet.Identity
@if (Request.IsAuthenticated)
{
    using (Html.BeginForm("LogOff", "Account", FormMethod.Post, new { id =
"logoutForm", @class = "navbar-right" }))
    {
    @Html.AntiForgeryToken()

    <ul class="nav navbar-nav navbar-right">
        <li>
            @Html.ActionLink("Hello " + User.Identity.GetUserName() + "!", "Index",
"Manage", routeValues: null, htmlAttributes: new { title = "Manage" })
        </li>
        <li><a href="javascript:document.getElementById('logoutForm').submit()">Log
off</a></li>
    </ul>
    }
}
else
{
    <ul class="nav navbar-nav navbar-right">
        <li>@Html.ActionLink("Register", "Register", "Account", routeValues: null,
htmlAttributes: new { id = "registerLink" })</li>
```

```
        <li>@Html.ActionLink("Login", "Index", "Login", routeValues: null,
htmlAttributes: new { id = "loginLink" })</li>
    </ul>
}
```

6.3.3. Error.cshtml

```
@model System.Web.Mvc.HandleErrorInfo

@{
    ViewBag.Title = "Error";
}

<h1 class="text-danger">Error.</h1>
<h2 class="text-danger">An error occurred while processing your request.</h2>
```

6.3.4. Lockout.cs.html

```
@model System.Web.Mvc.HandleErrorInfo

@{
    ViewBag.Title = "Locked Out";
}

<hgroup>
    <h1 class="text-danger">Locked out.</h1>
    <h2 class="text-danger">This account has been locked out, please try again
later.</h2>
</hgroup>
```

6.4. User

6.4.1. Create

```
@model Student_Management_System.Models.user

@{
    ViewBag.Title = "Create";
}

<h2>Create</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>user</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.firstname, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
```

```
                @Html.EditorFor(model => model.firstname, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.firstname, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.lastname, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.lastname, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.lastname, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.username, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.username, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.username, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.password, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.PasswordFor(model => model.password, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.password, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

6.4.2. Delete

```
@model Student_Management_System.Models.user
```

```
@{
    ViewBag.Title = "Delete";
}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>user</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.firstname)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.firstname)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.lastname)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.lastname)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.username)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.username)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.password)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.password)
        </dd>

    </dl>

    @using (Html.BeginForm()) {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-default" /> |
            @Html.ActionLink("Back to List", "Index")
        </div>
    }
</div>
```

6.4.3. Details

```
@model Student_Management_System.Models.user
```

```
@{
    ViewBag.Title = "Details";
}

<h2>Details</h2>

<div>
    <h4>user</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.firstname)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.firstname)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.lastname)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.lastname)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.username)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.username)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.password)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.password)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.id }) |
    @Html.ActionLink("Back to List", "Index")
</p>
```

6.4.4. Edit

```
@model Student_Management_System.Models.user

@{
    ViewBag.Title = "Details";
}

<h2>Details</h2>
```

```
<div>
    <h4>user</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.firstname)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.firstname)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.lastname)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.lastname)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.username)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.username)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.password)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.password)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.id }) |
    @Html.ActionLink("Back to List", "Index")
</p>
```

6.4.5. Index

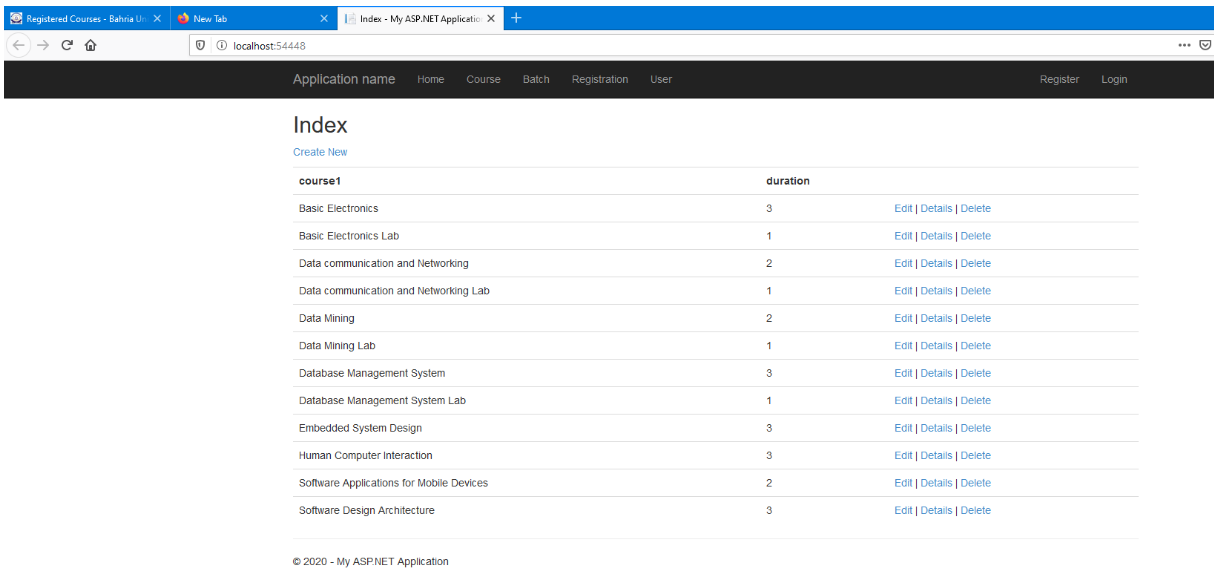
```
@model IEnumerable<Student_Management_System.Models.user>

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
```

```
        <th>
            @Html.DisplayNameFor(model => model.firstname)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.lastname)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.username)
        </th>
   @*    <th>
           @Html.DisplayNameFor(model => model.password)
        </th>*@
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.firstname)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.lastname)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.username)
        </td>
        @*<td>
            @Html.DisplayFor(modelItem => item.password)
        </td>*@
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.id }) |
            @Html.ActionLink("Details", "Details", new { id=item.id }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.id })
        </td>
    </tr>
}

</table>
```

7. OUTPUT

7.1. Course

7.1.1. Course Details



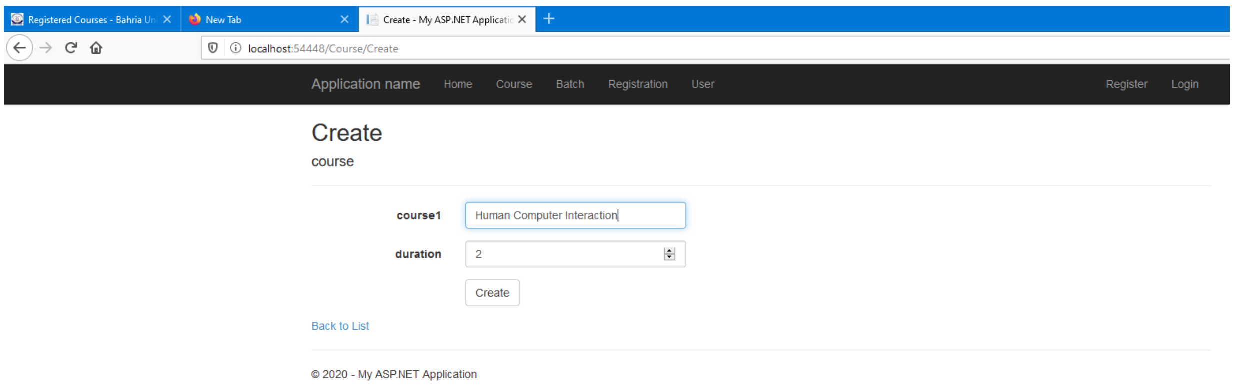
Index

Create New

course1	duration	
Basic Electronics	3	Edit | Details | Delete
Basic Electronics Lab	1	Edit | Details | Delete
Data communication and Networking	2	Edit | Details | Delete
Data communication and Networking Lab	1	Edit | Details | Delete
Data Mining	2	Edit | Details | Delete
Data Mining Lab	1	Edit | Details | Delete
Database Management System	3	Edit | Details | Delete
Database Management System Lab	1	Edit | Details | Delete
Embedded System Design	3	Edit | Details | Delete
Human Computer Interaction	3	Edit | Details | Delete
Software Applications for Mobile Devices	2	Edit | Details | Delete
Software Design Architecture	3	Edit | Details | Delete

© 2020 - My ASP.NET Application

7.1.2. Add Course



Registered Courses - Bahria Un
New Tab
Create - My ASP.NET Applicati
localhost:54448/Course/Create
Application name
Home
Course
Batch
Registration
User
Register
Login

Create

course

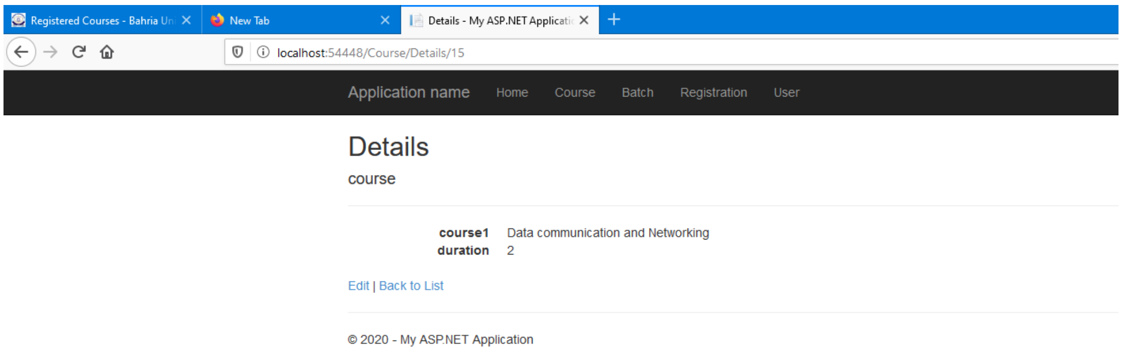
course1	Human Computer Interaction
duration	2

Create

Back to List

© 2020 - My ASP.NET Application

7.1.3. View Course


Registered Courses - Bahria Un
New Tab
Details - My ASP.NET Applicati
localhost:54448/Course/Details/15
Application name
Home
Course
Batch
Registration
User

Details

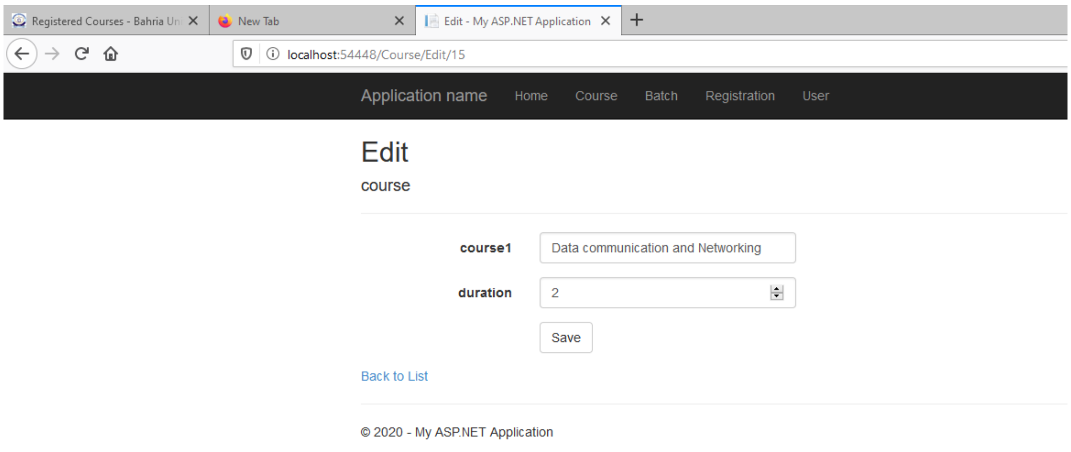
course

course1	Data communication and Networking
duration	2

Edit | Back to List

© 2020 - My ASP.NET Application

7.1.4. Edit Course


Registered Courses - Bahria Un
New Tab
Edit - My ASP.NET Application
localhost:54448/Course/Edit/15
Application name
Home
Course
Batch
Registration
User

Edit

course

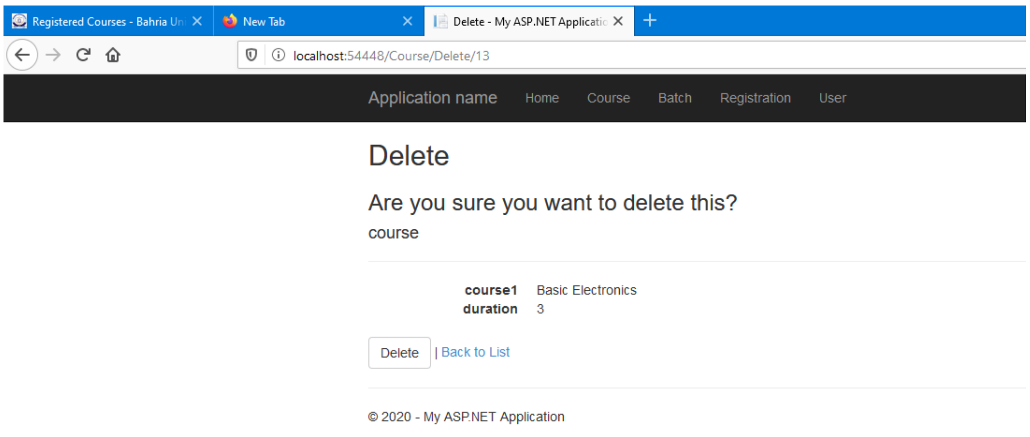
course1	Data communication and Networking
duration	2

Save

Back to List

© 2020 - My ASP.NET Application

7.1.5. Delete Course


Registered Courses - Bahria Un
New Tab
Delete - My ASP.NET Applicatio
localhost:54448/Course/Delete/13
Application name
Home
Course
Batch
Registration
User

Delete

Are you sure you want to delete this?

course

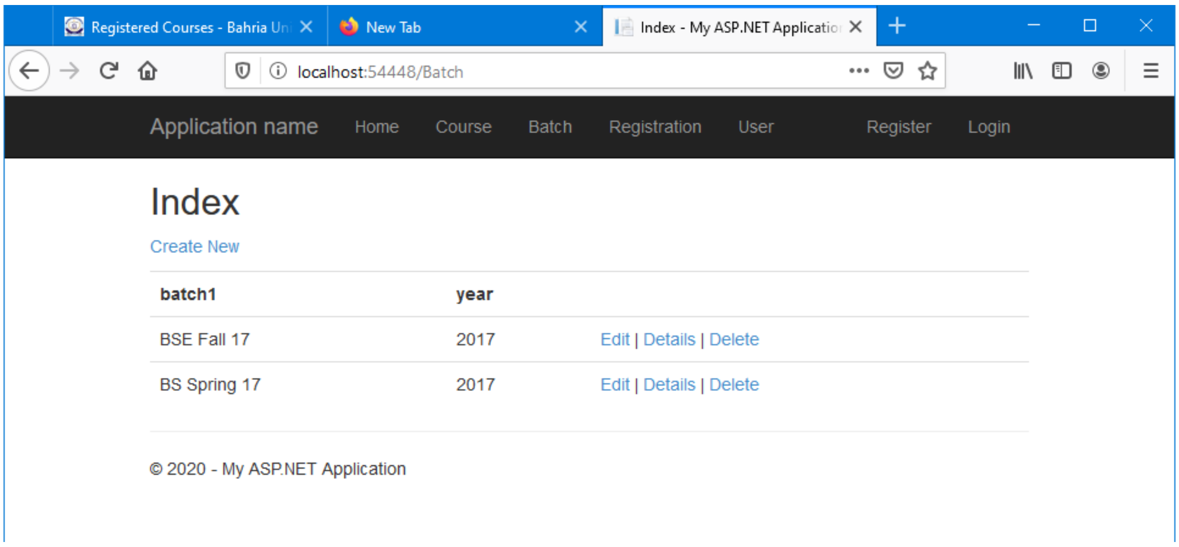
course1	Basic Electronics
duration	3

Delete | Back to List

© 2020 - My ASP.NET Application

7.2. Batch:

7.2.1. Batch Details



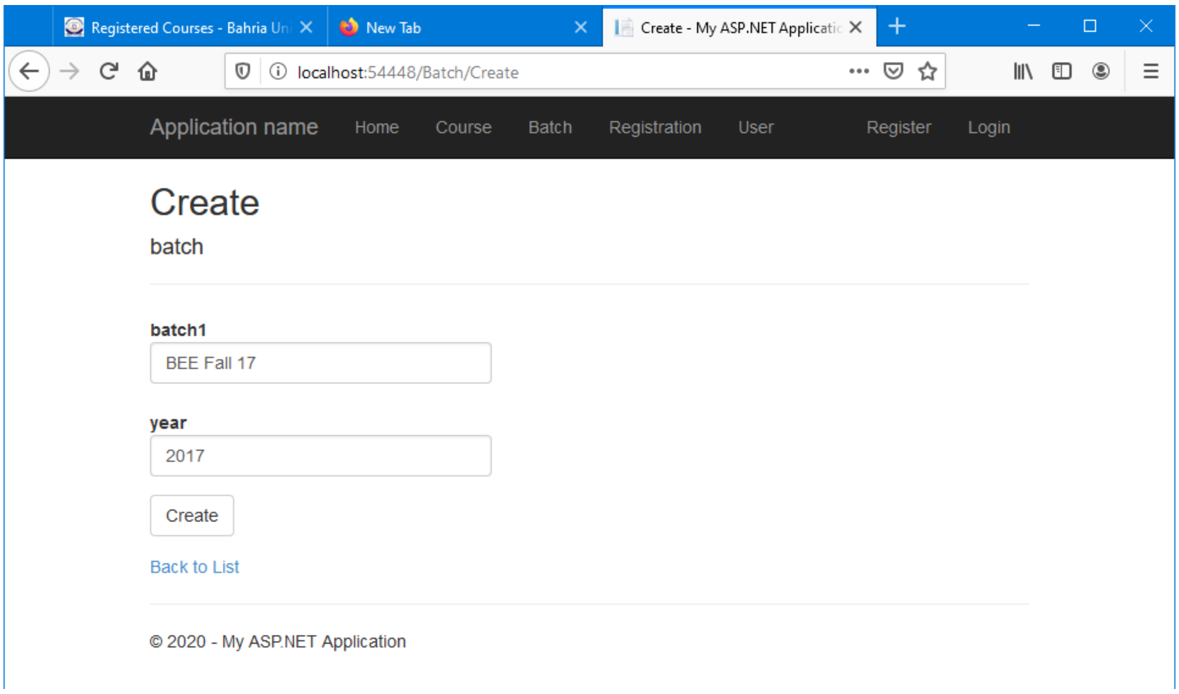
Index

Create New

batch1	year	
BSE Fall 17	2017	Edit | Details | Delete
BS Spring 17	2017	Edit | Details | Delete

© 2020 - My ASP.NET Application

7.2.2. Add Batch


Registered Courses - Bahria Un
New Tab
Create - My ASP.NET Applicatic
localhost:54448/Batch/Create
Application name
Home
Course
Batch
Registration
User
Register
Login

Create

batch

batch1: BEE Fall 17

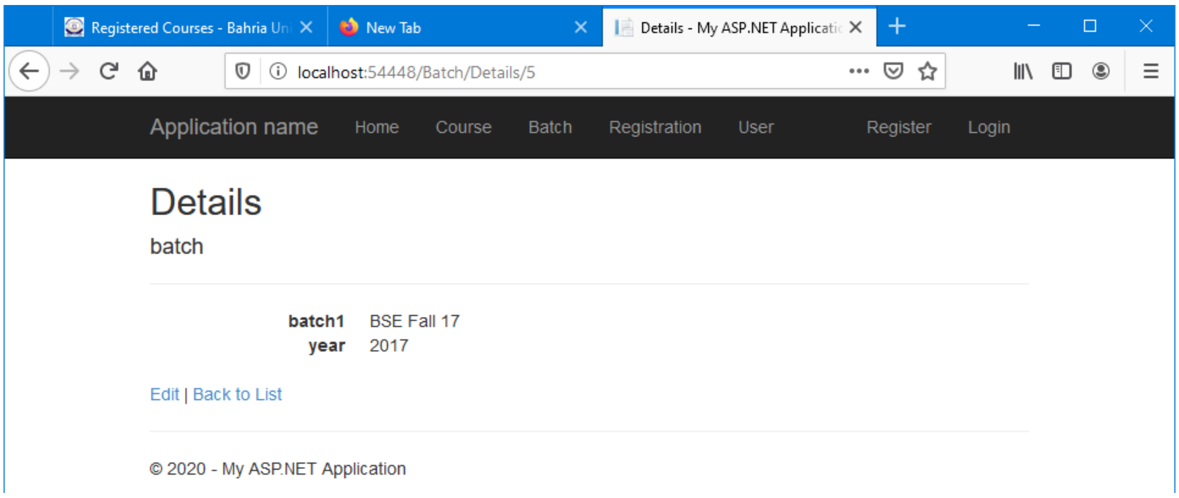
year: 2017

Create

Back to List

© 2020 - My ASP.NET Application

7.2.3. View Batch


Registered Courses - Bahria Uni
New Tab
Details - My ASP.NET Applicati
localhost:54448/Batch/Details/5
Application name
Home
Course
Batch
Registration
User
Register
Login

Details

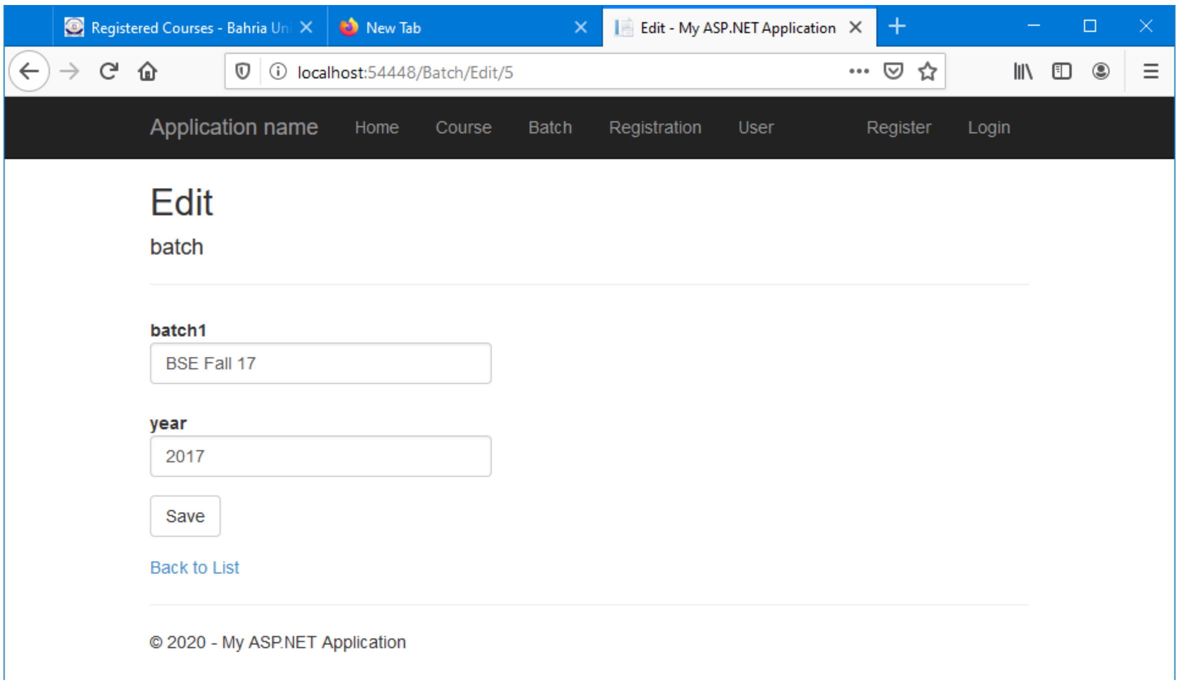
batch

batch1	BSE Fall 17
year	2017

Edit | Back to List

© 2020 - My ASP.NET Application

7.2.4. Edit Batch


Registered Courses - Bahria Uni
New Tab
Edit - My ASP.NET Application
localhost:54448/Batch/Edit/5
Application name
Home
Course
Batch
Registration
User
Register
Login

Edit

batch

batch1
BSE Fall 17

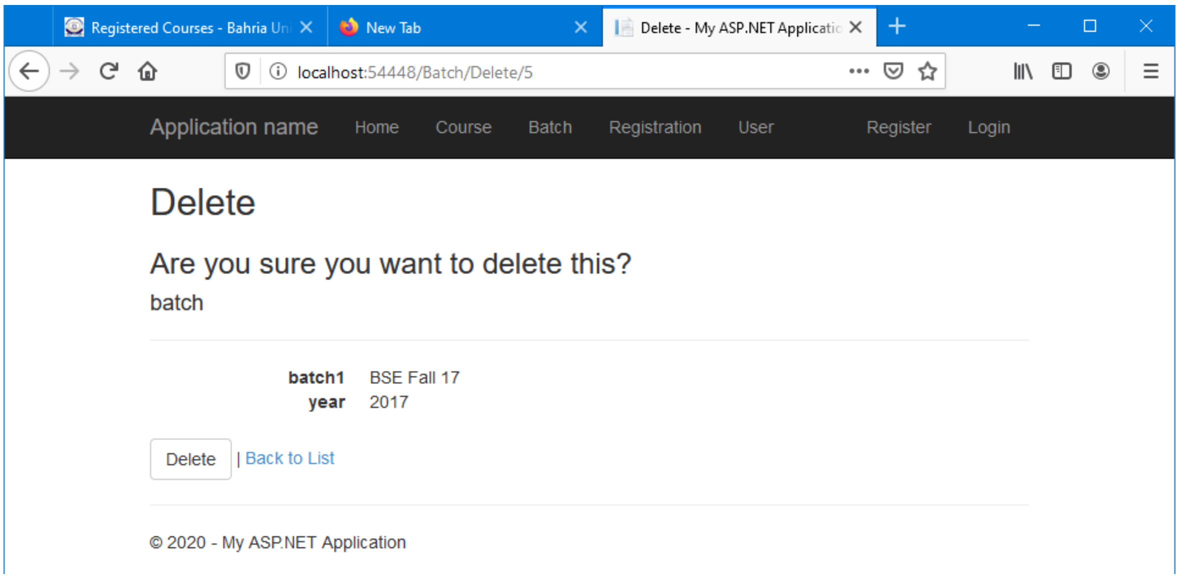
year
2017

Save

Back to List

© 2020 - My ASP.NET Application

7.2.5. Delete Batch



Registered Courses - Bahria Un
New Tab
Delete - My ASP.NET Applicatio
localhost:54448/Batch/Delete/5
Application name
Home
Course
Batch
Registration
User
Register
Login

Delete

Are you sure you want to delete this?

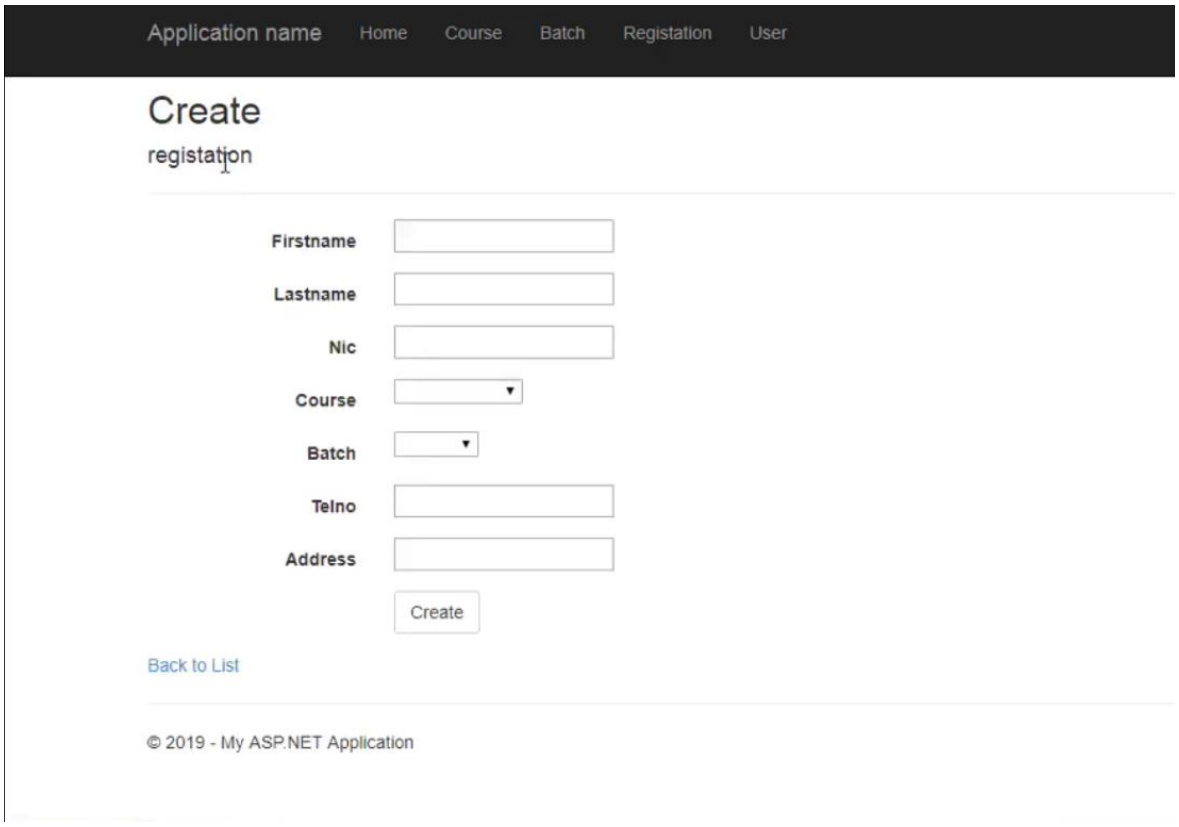
batch

batch1	BSE Fall 17
year	2017

Delete | Back to List

© 2020 - My ASP.NET Application

7.3. Registration:



Create

registation

Firstname	
Lastname	
Nic	
Course	
Batch	
Telno	
Address	

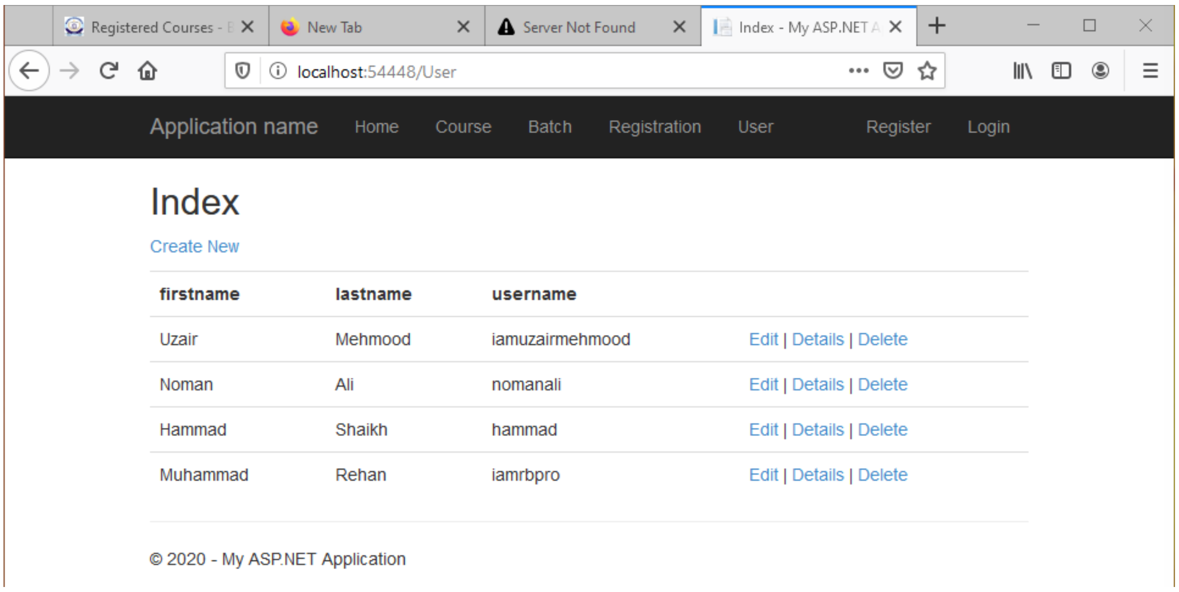
Create

Back to List

© 2019 - My ASP.NET Application

7.4. User:

7.4.1. User Details



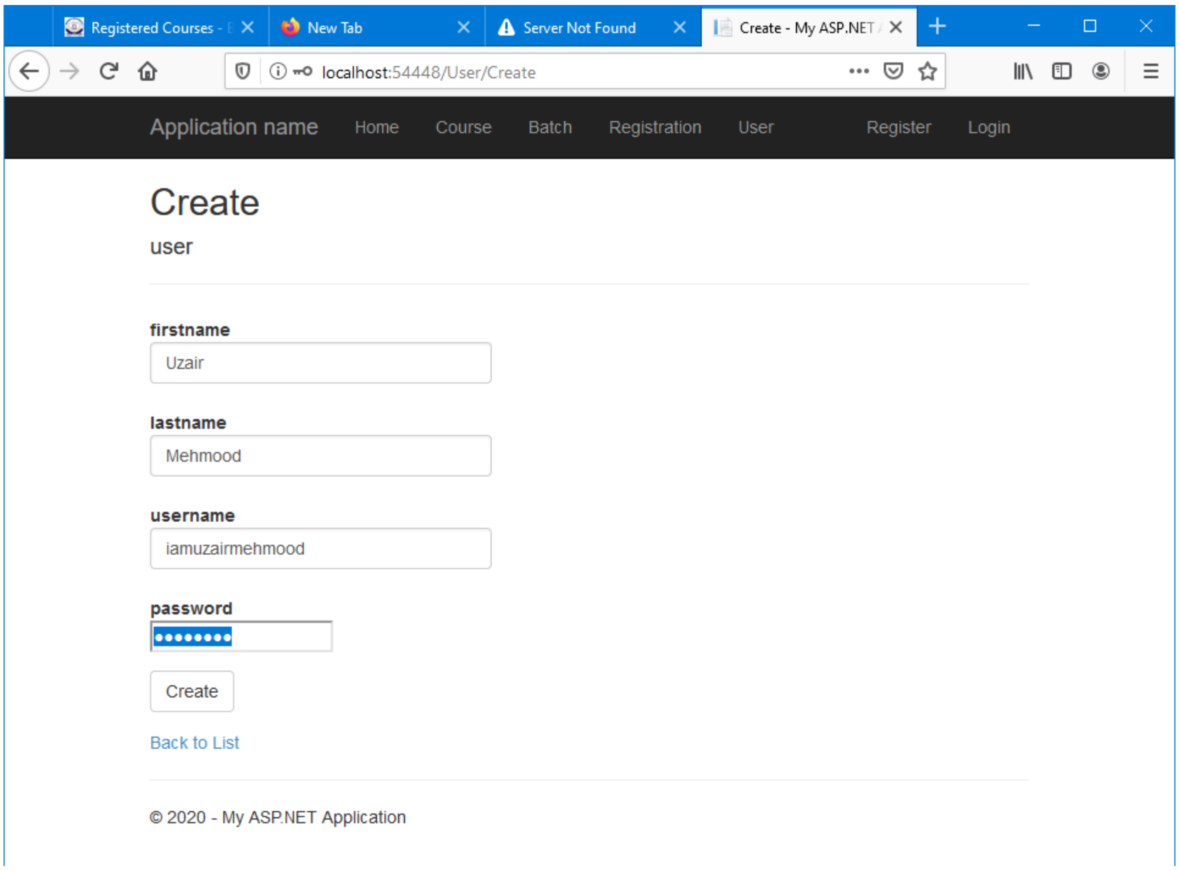
Index

Create New

firstname	lastname	username	
Uzair	Mehmood	iamuzairmehmood	Edit | Details | Delete
Noman	Ali	nomanali	Edit | Details | Delete
Hammad	Shaikh	hammad	Edit | Details | Delete
Muhammad	Rehan	iamrbpro	Edit | Details | Delete

© 2020 - My ASP.NET Application

7.4.2. Add User



Create

user

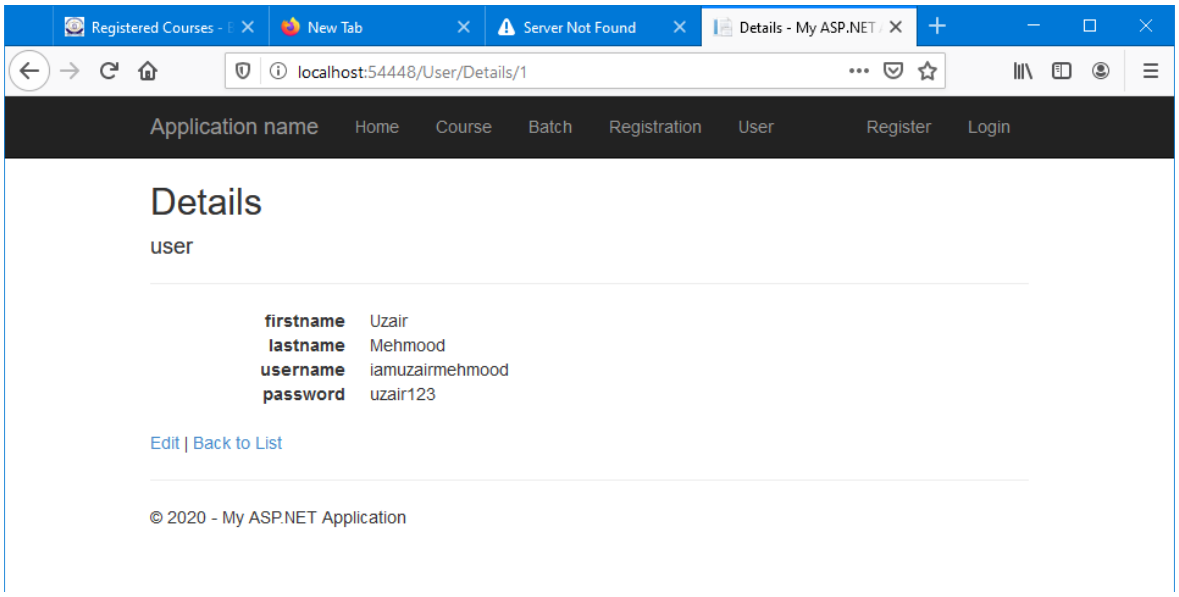
firstname	Uzair
lastname	Mehmood
username	iamuzairmehmood
password	••••••••

Create

Back to List

© 2020 - My ASP.NET Application

7.4.3. View User



Details

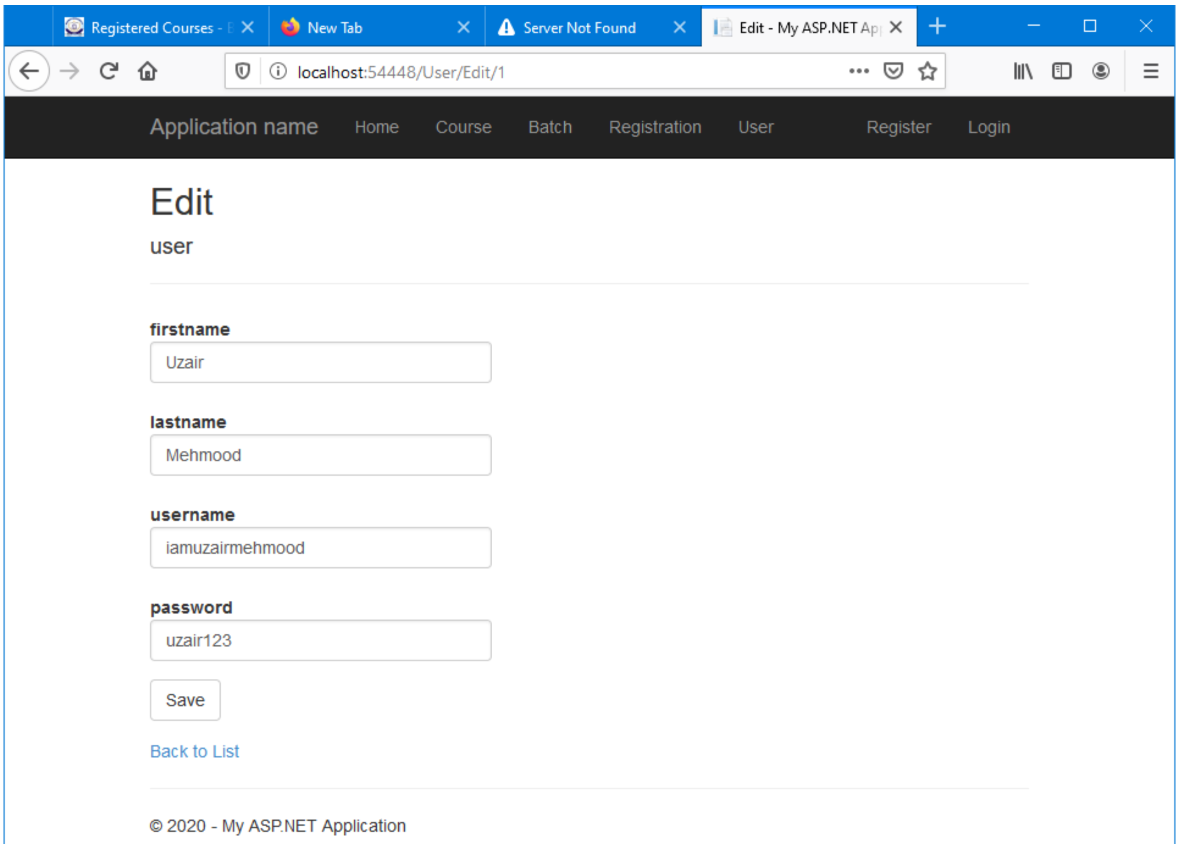
user

firstname	Uzair
lastname	Mehmood
username	iamuzairmehmood
password	uzair123

Edit | Back to List

© 2020 - My ASP.NET Application

7.4.4. Edit User



Edit

user

firstname: Uzair

lastname: Mehmood

username: iamuzairmehmood

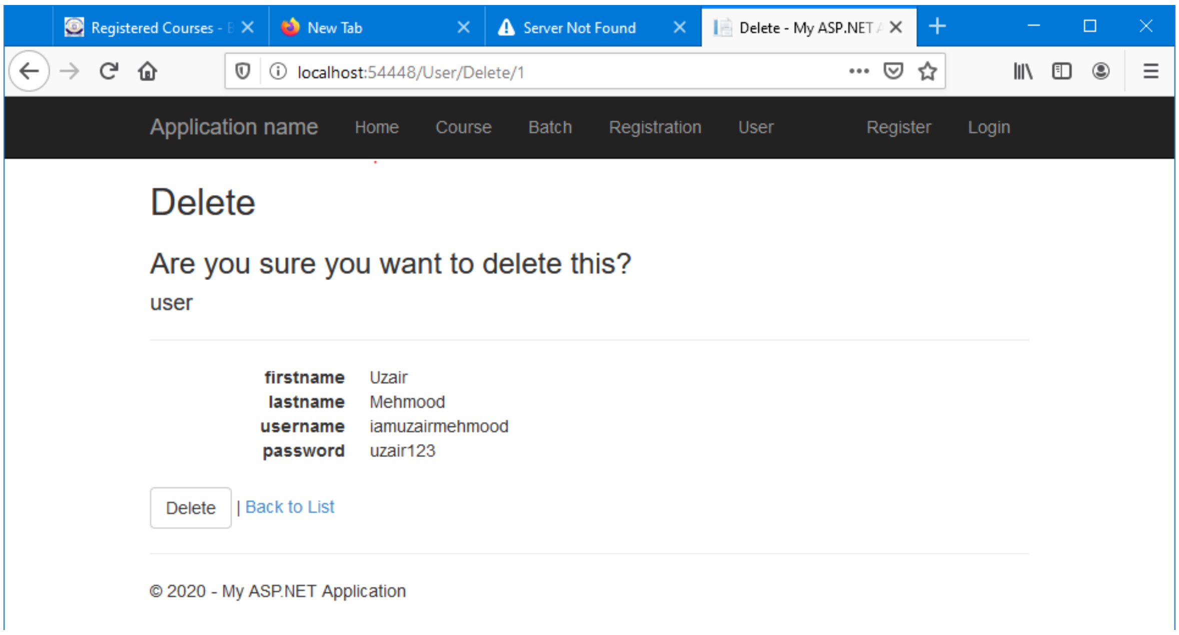
password: uzair123

Save

Back to List

© 2020 - My ASP.NET Application

7.4.5. Delete User



Delete

Are you sure you want to delete this?

user

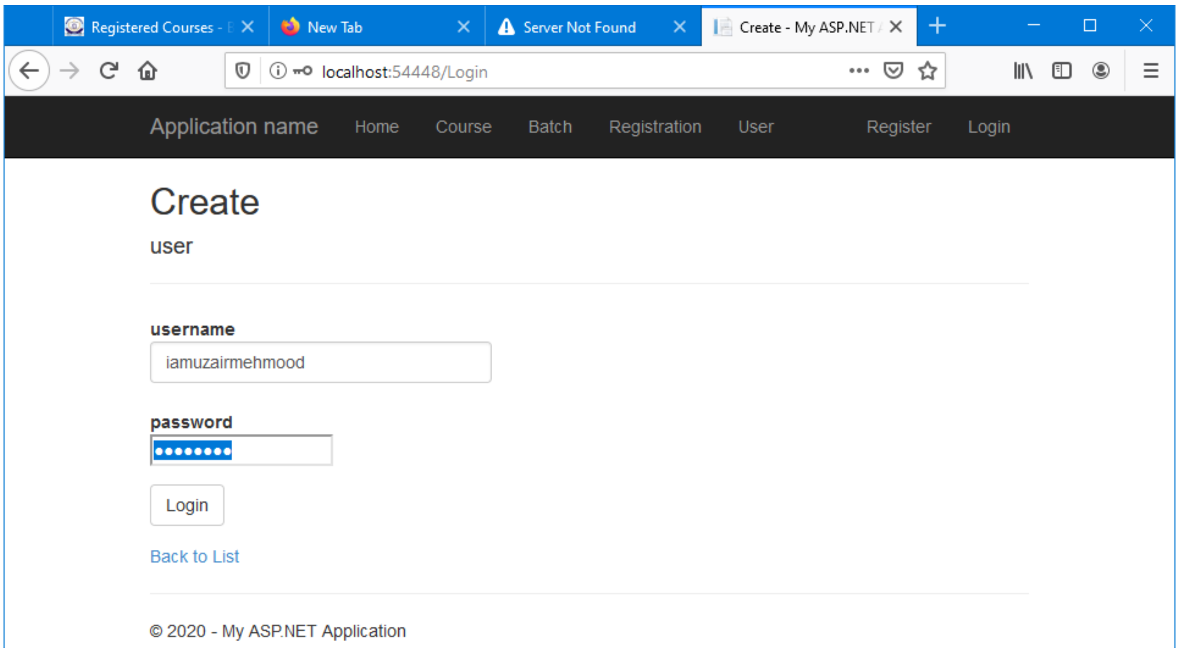
firstname	Uzair
lastname	Mehmood
username	iamuzairmehmood
password	uzair123

Delete | Back to List

© 2020 - My ASP.NET Application

7.5. Login

7.5.1. Login View



Create

user

username

iamuzairmehmood

password

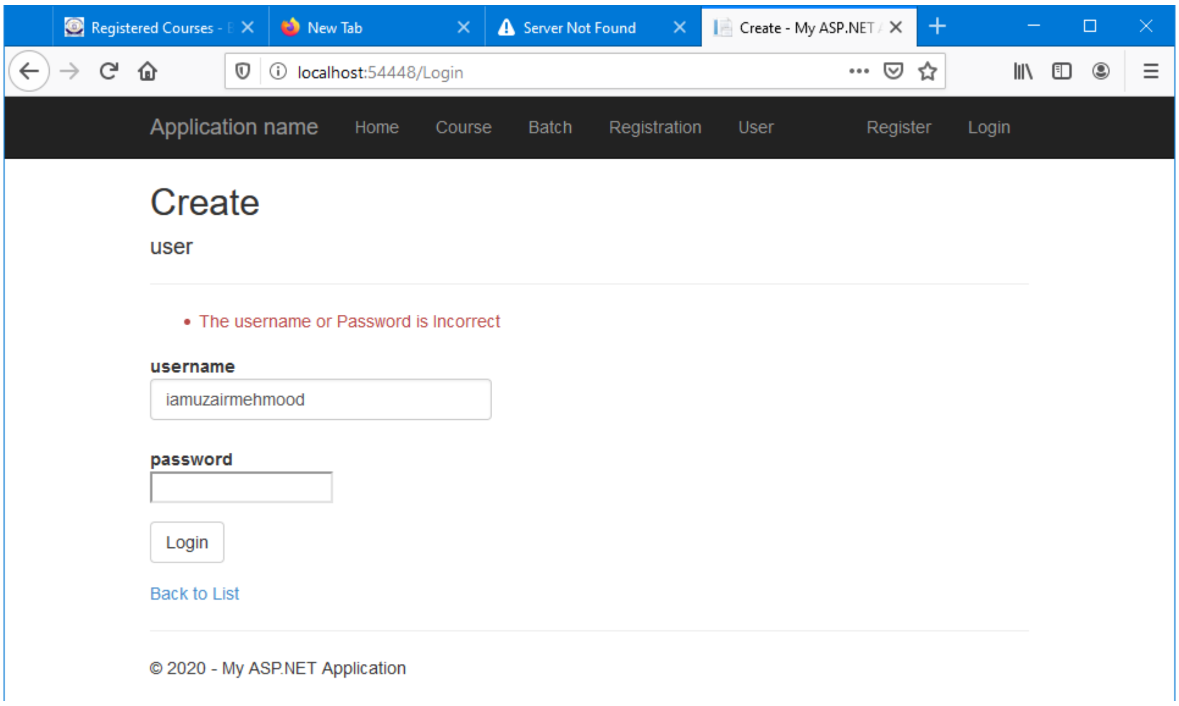
••••••••

Login

Back to List

© 2020 - My ASP.NET Application

7.5.2. Error



Create

user

- The username or Password is Incorrect

username

iamuzairmehmood

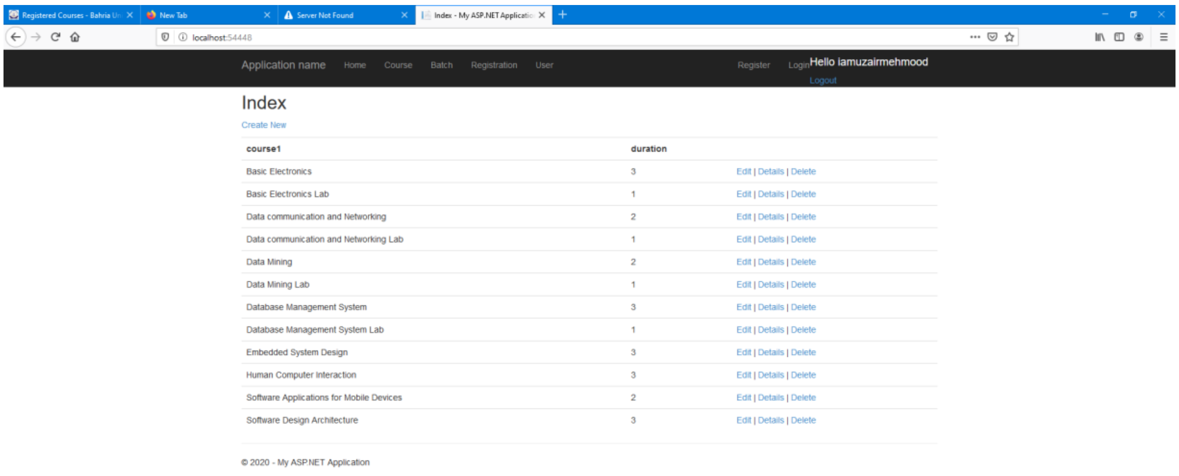
password

Login

Back to List

© 2020 - My ASP.NET Application

7.6. Display User Name



Hello iamuzairmehmood

Index

Create New

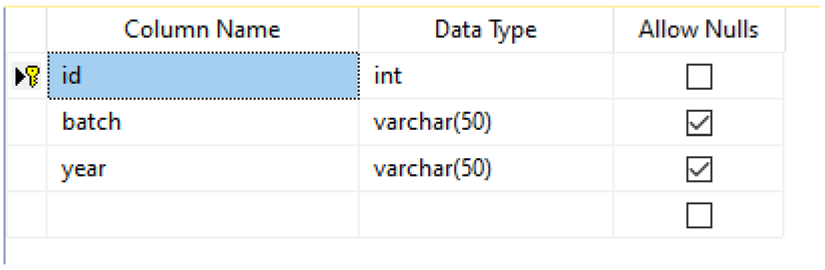
course1	duration	
Basic Electronics	3	Edit | Details | Delete
Basic Electronics Lab	1	Edit | Details | Delete
Data communication and Networking	2	Edit | Details | Delete
Data communication and Networking Lab	1	Edit | Details | Delete
Data Mining	2	Edit | Details | Delete
Data Mining Lab	1	Edit | Details | Delete
Database Management System	3	Edit | Details | Delete
Database Management System Lab	1	Edit | Details | Delete
Embedded System Design	3	Edit | Details | Delete
Human Computer Interaction	3	Edit | Details | Delete
Software Applications for Mobile Devices	2	Edit | Details | Delete
Software Design Architecture	3	Edit | Details | Delete

© 2020 - My ASP.NET Application

8. DATABASE

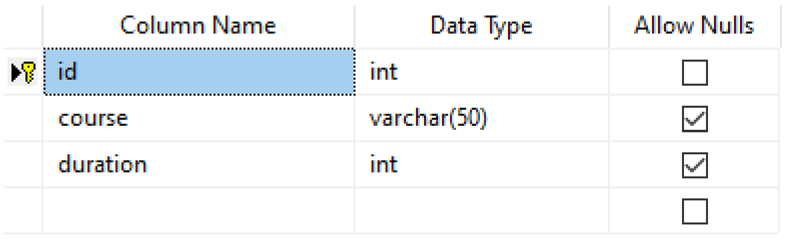
8.1. Table-Design

8.1.1. Batch



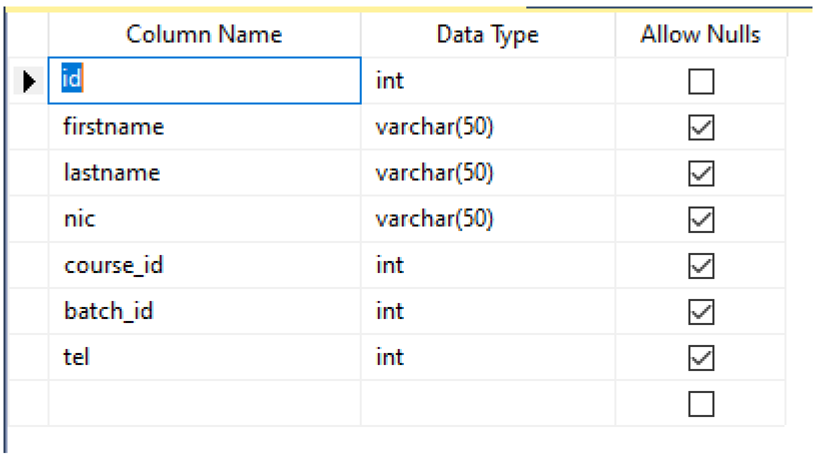
Column Name	Data Type	Allow Nulls
id	int	☐
batch	varchar(50)	☑
year	varchar(50)	☑
		☐

8.1.2. Course



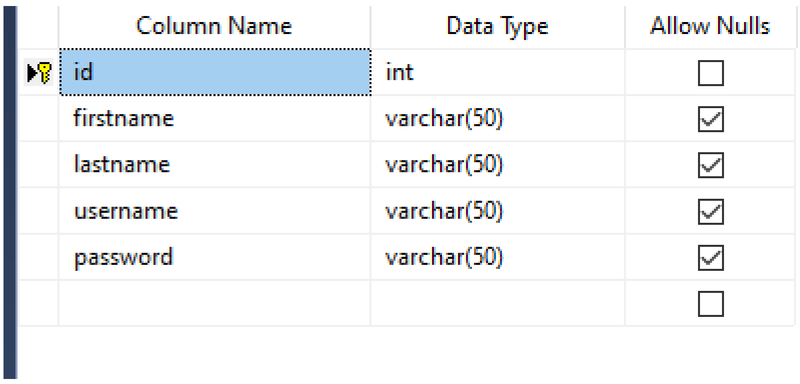
Column Name	Data Type	Allow Nulls
id	int	☐
course	varchar(50)	☑
duration	int	☑
		☐

8.1.3. Registration



Column Name	Data Type	Allow Nulls
id	int	☐
firstname	varchar(50)	☑
lastname	varchar(50)	☑
nic	varchar(50)	☑
course_id	int	☑
batch_id	int	☑
tel	int	☑
		☐

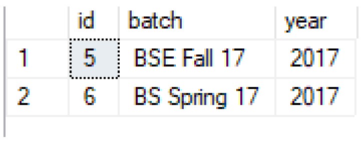
8.1.4. User



Column Name	Data Type	Allow Nulls
id	int	☐
firstname	varchar(50)	✓
lastname	varchar(50)	✓
username	varchar(50)	✓
password	varchar(50)	✓
		☐

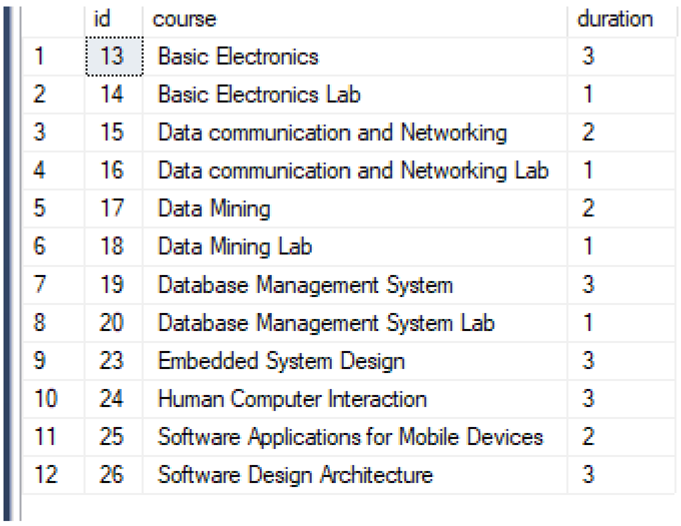
8.2. Data In Tables

8.2.1. Batch



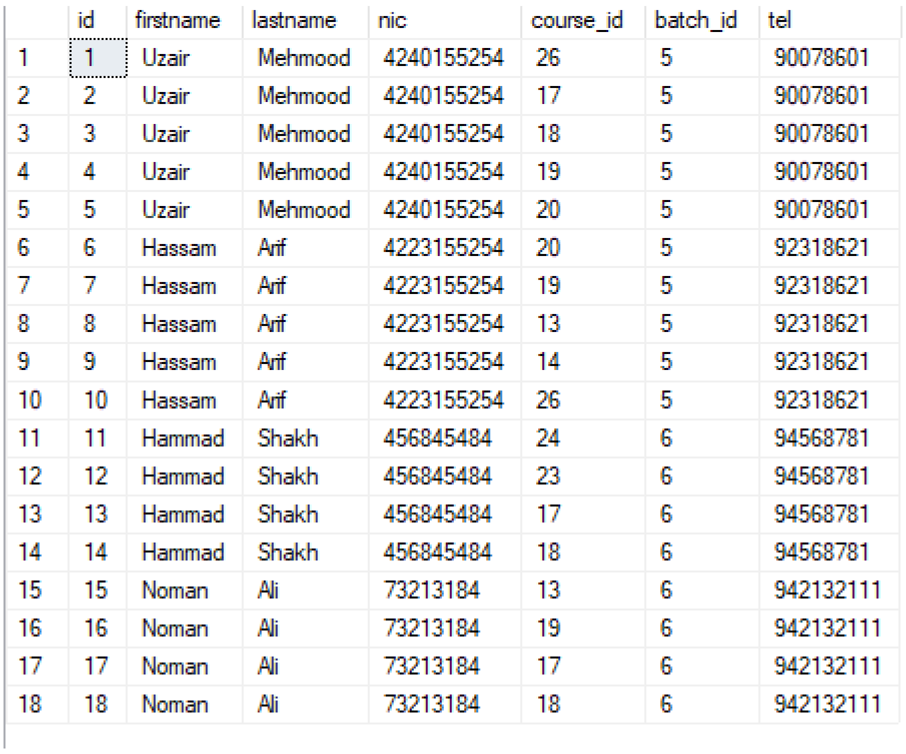
	id	batch	year
1	5	BSE Fall 17	2017
2	6	BS Spring 17	2017

8.2.2. Course



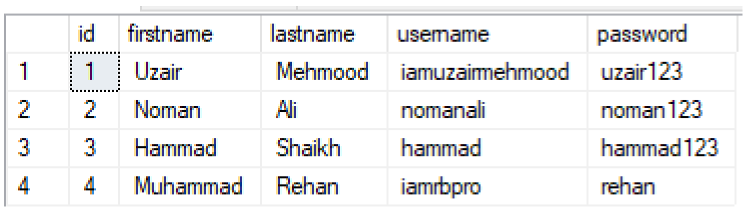
	id	course	duration
1	13	Basic Electronics	3
2	14	Basic Electronics Lab	1
3	15	Data communication and Networking	2
4	16	Data communication and Networking Lab	1
5	17	Data Mining	2
6	18	Data Mining Lab	1
7	19	Database Management System	3
8	20	Database Management System Lab	1
9	23	Embedded System Design	3
10	24	Human Computer Interaction	3
11	25	Software Applications for Mobile Devices	2
12	26	Software Design Architecture	3

8.2.3. Registration



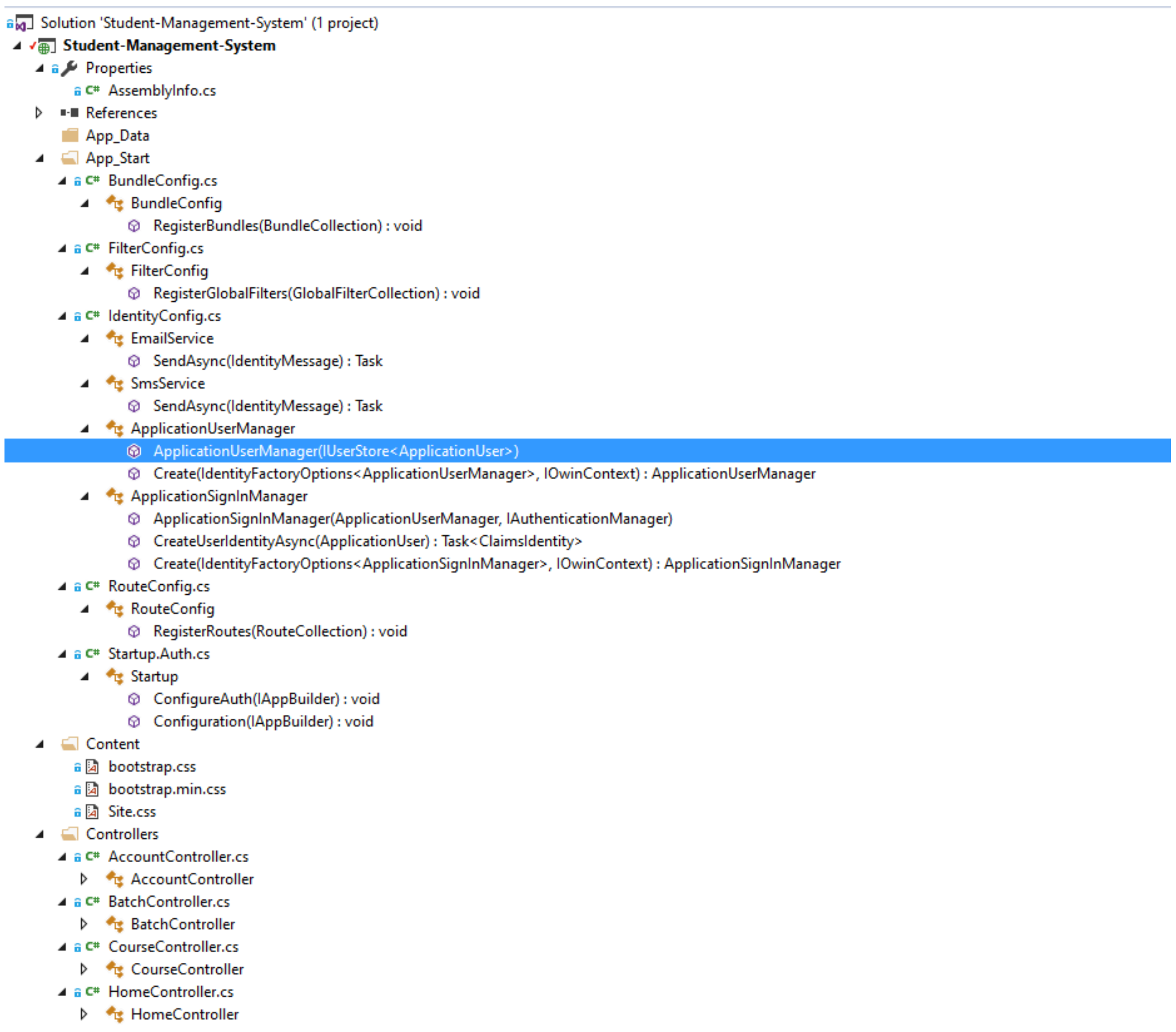
	id	firstname	lastname	nic	course_id	batch_id	tel
1	1	Uzair	Mehmood	4240155254	26	5	90078601
2	2	Uzair	Mehmood	4240155254	17	5	90078601
3	3	Uzair	Mehmood	4240155254	18	5	90078601
4	4	Uzair	Mehmood	4240155254	19	5	90078601
5	5	Uzair	Mehmood	4240155254	20	5	90078601
6	6	Hassam	Arif	4223155254	20	5	92318621
7	7	Hassam	Arif	4223155254	19	5	92318621
8	8	Hassam	Arif	4223155254	13	5	92318621
9	9	Hassam	Arif	4223155254	14	5	92318621
10	10	Hassam	Arif	4223155254	26	5	92318621
11	11	Hammad	Shakh	456845484	24	6	94568781
12	12	Hammad	Shakh	456845484	23	6	94568781
13	13	Hammad	Shakh	456845484	17	6	94568781
14	14	Hammad	Shakh	456845484	18	6	94568781
15	15	Noman	Ali	73213184	13	6	942132111
16	16	Noman	Ali	73213184	19	6	942132111
17	17	Noman	Ali	73213184	17	6	942132111
18	18	Noman	Ali	73213184	18	6	942132111

8.2.4. User

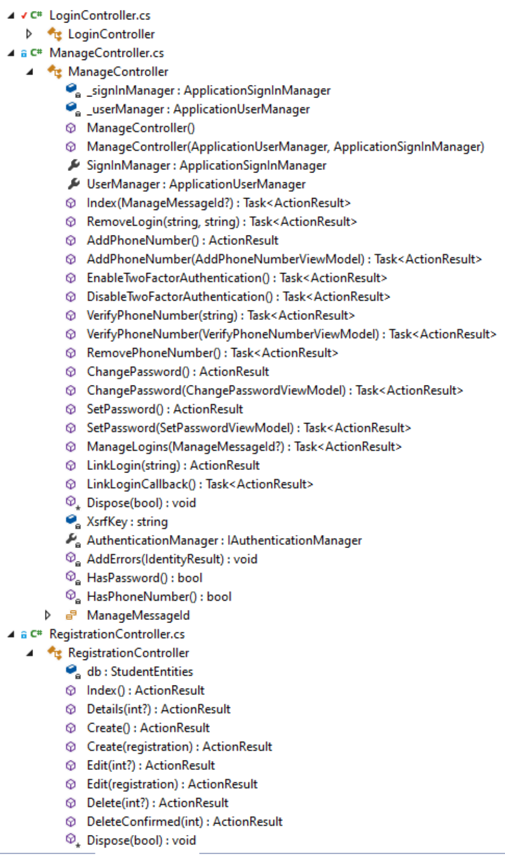


	id	firstname	lastname	username	password
1	1	Uzair	Mehmood	iamuzairmehmood	uzair123
2	2	Noman	Ali	nomanali	noman123
3	3	Hammad	Shaikh	hammad	hammad123
4	4	Muhammad	Rehan	iamrbpro	rehan

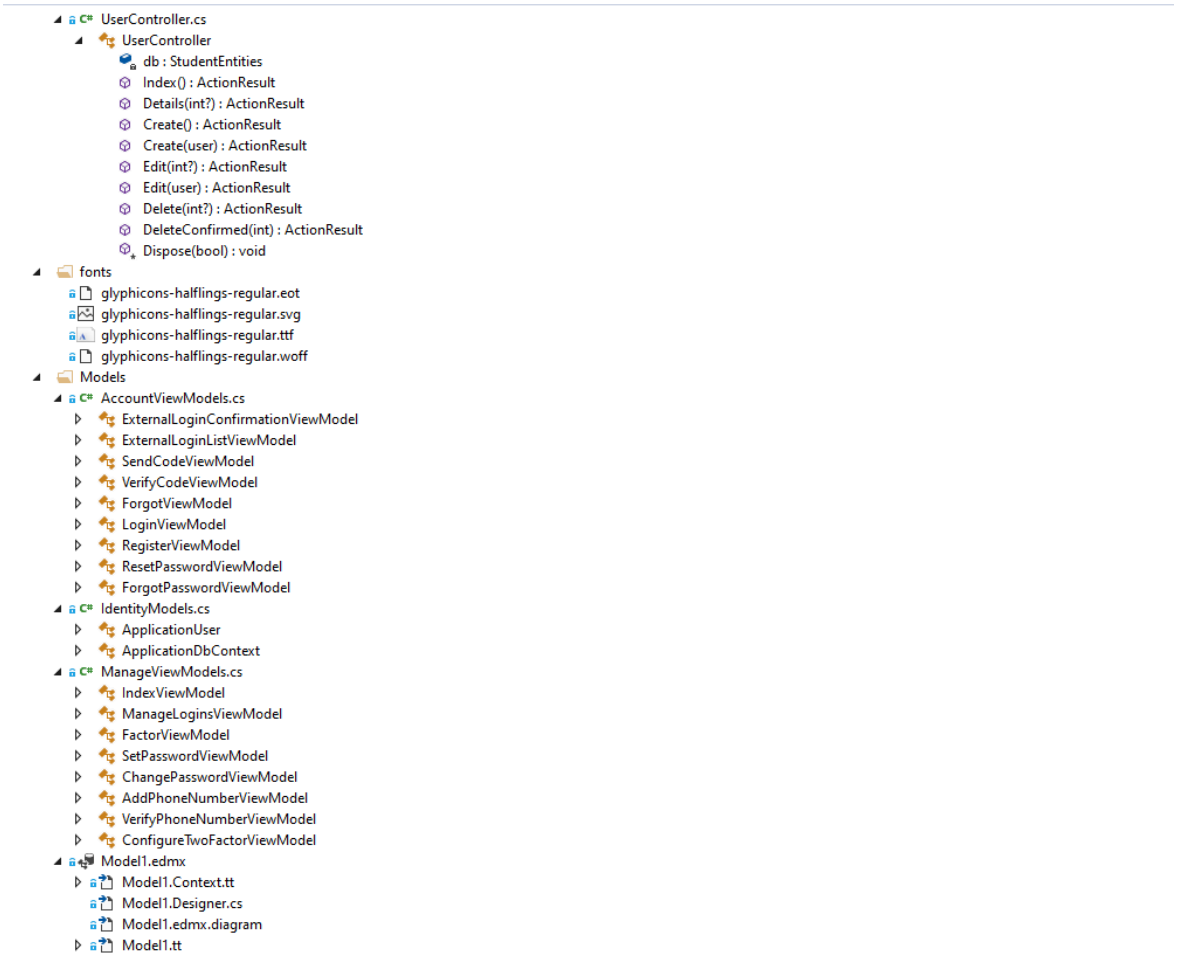
9. SOLUTION EXPOLRER



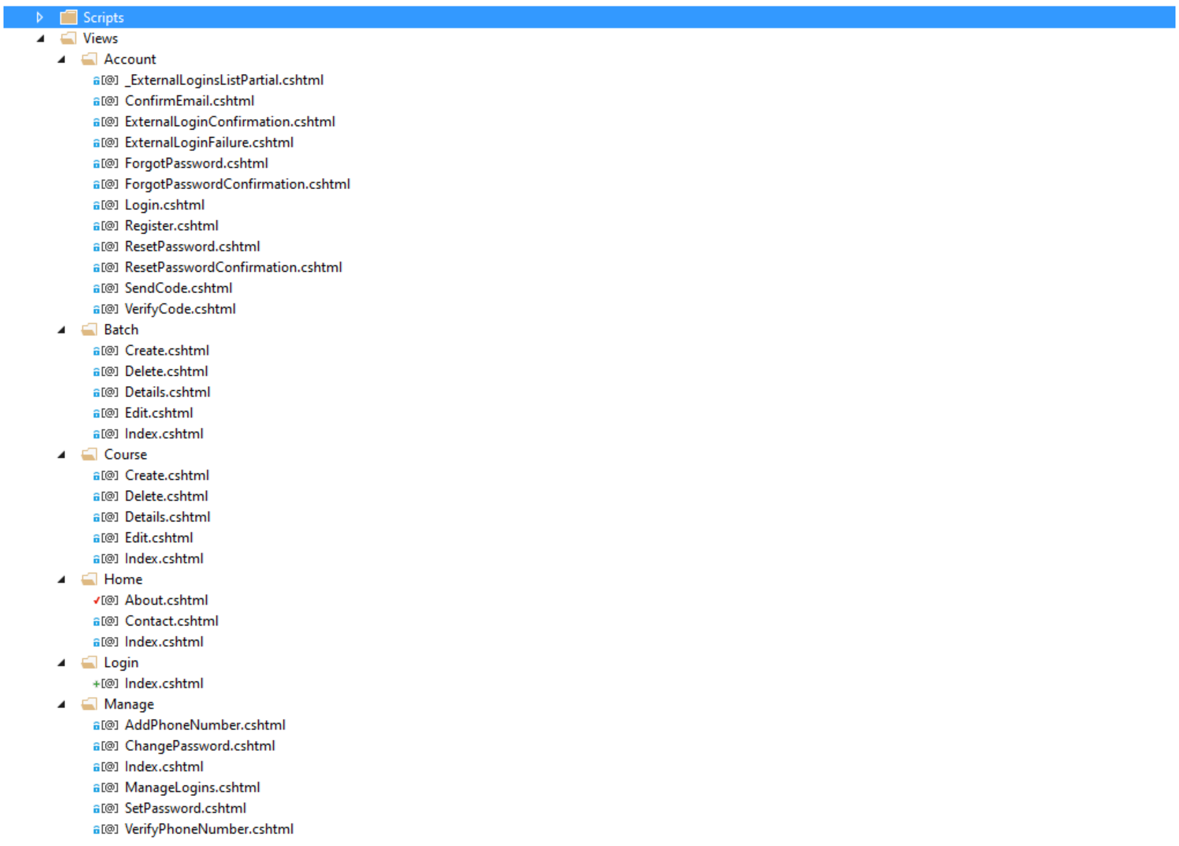
```
Solution 'Student-Management-System' (1 project)
  Student-Management-System
    Properties
      AssemblyInfo.cs
    References
    App_Data
    App_Start
      BundleConfig.cs
        BundleConfig
          RegisterBundles(BundleCollection) : void
      FilterConfig.cs
        FilterConfig
          RegisterGlobalFilters(GlobalFilterCollection) : void
      IdentityConfig.cs
        EmailService
          SendAsync(IdentityMessage) : Task
        SmsService
          SendAsync(IdentityMessage) : Task
        ApplicationUserManager
          ApplicationUserManager(IUserStore<ApplicationUser>)
          Create(IdentityFactoryOptions<ApplicationUserManager>, IOwinContext) : ApplicationUserManager
        ApplicationSignInManager
          ApplicationSignInManager(ApplicationUserManager, IAuthenticationManager)
          CreateUserIdentityAsync(ApplicationUser) : Task<ClaimsIdentity>
          Create(IdentityFactoryOptions<ApplicationSignInManager>, IOwinContext) : ApplicationSignInManager
      RouteConfig.cs
        RouteConfig
          RegisterRoutes(RouteCollection) : void
      Startup.Auth.cs
        Startup
          ConfigureAuth(IAppBuilder) : void
          Configuration(IAppBuilder) : void
    Content
      bootstrap.css
      bootstrap.min.css
      Site.css
    Controllers
      AccountController.cs
        AccountController
      BatchController.cs
        BatchController
      CourseController.cs
        CourseController
      HomeController.cs
        HomeController
```



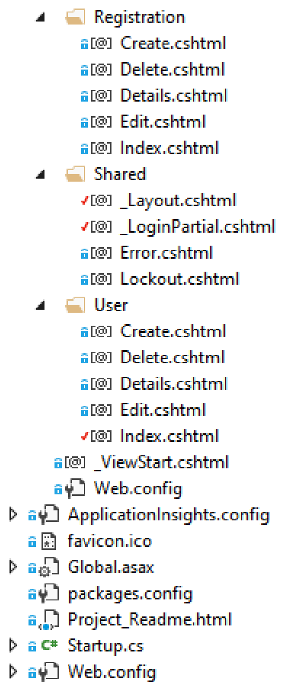
```
◢ ✓ C# LoginController.cs
    ▷ LoginController
◢ C# ManageController.cs
    ◢ ManageController
        _signInManager : ApplicationSignInManager
        _userManager : ApplicationUserManager
        ManageController()
        ManageController(ApplicationUserManager, ApplicationSignInManager)
        SignInManager : ApplicationSignInManager
        UserManager : ApplicationUserManager
        Index(ManageMessageId?) : Task<ActionResult>
        RemoveLogin(string, string) : Task<ActionResult>
        AddPhoneNumber() : ActionResult
        AddPhoneNumber(AddPhoneNumberViewModel) : Task<ActionResult>
        EnableTwoFactorAuthentication() : Task<ActionResult>
        DisableTwoFactorAuthentication() : Task<ActionResult>
        VerifyPhoneNumber(string) : Task<ActionResult>
        VerifyPhoneNumber(VerifyPhoneNumberViewModel) : Task<ActionResult>
        RemovePhoneNumber() : Task<ActionResult>
        ChangePassword() : ActionResult
        ChangePassword(ChangePasswordViewModel) : Task<ActionResult>
        SetPassword() : ActionResult
        SetPassword(SetPasswordViewModel) : Task<ActionResult>
        ManageLogins(ManageMessageId?) : Task<ActionResult>
        LinkLogin(string) : ActionResult
        LinkLoginCallback() : Task<ActionResult>
        Dispose(bool) : void
        XsrfKey : string
        AuthenticationManager : IAuthenticationManager
        AddErrors(IdentityResult) : void
        HasPassword() : bool
        HasPhoneNumber() : bool
      ▷ ManageMessageId
◢ C# RegistrationController.cs
    ◢ RegistrationController
        db : StudentEntities
        Index() : ActionResult
        Details(int?) : ActionResult
        Create() : ActionResult
        Create(registration) : ActionResult
        Edit(int?) : ActionResult
        Edit(registration) : ActionResult
        Delete(int?) : ActionResult
        DeleteConfirmed(int) : ActionResult
        Dispose(bool) : void
```



```
◢ C# UserController.cs
    ◢ UserController
        db : StudentEntities
        Index() : ActionResult
        Details(int?) : ActionResult
        Create() : ActionResult
        Create(user) : ActionResult
        Edit(int?) : ActionResult
        Edit(user) : ActionResult
        Delete(int?) : ActionResult
        DeleteConfirmed(int) : ActionResult
        Dispose(bool) : void
◢ fonts
    glyphicons-halflings-regular.eot
    glyphicons-halflings-regular.svg
    glyphicons-halflings-regular.ttf
    glyphicons-halflings-regular.woff
◢ Models
  ◢ C# AccountViewModels.cs
    ▷ ExternalLoginConfirmationViewModel
    ▷ ExternalLoginListViewModel
    ▷ SendCodeViewModel
    ▷ VerifyCodeViewModel
    ▷ ForgotViewModel
    ▷ LoginViewModel
    ▷ RegisterViewModel
    ▷ ResetPasswordViewModel
    ▷ ForgotPasswordViewModel
  ◢ C# IdentityModels.cs
    ▷ ApplicationUser
    ▷ ApplicationDbContext
  ◢ C# ManageViewModels.cs
    ▷ IndexViewModel
    ▷ ManageLoginsViewModel
    ▷ FactorViewModel
    ▷ SetPasswordViewModel
    ▷ ChangePasswordViewModel
    ▷ AddPhoneNumberViewModel
    ▷ VerifyPhoneNumberViewModel
    ▷ ConfigureTwoFactorViewModel
  ◢ Model1.edmx
    ▷ Model1.Context.tt
      Model1.Designer.cs
      Model1.edmx.diagram
    ▷ Model1.tt
```



```
▷ Scripts
◢ Views
  ◢ Account
      _ExternalLoginsListPartial.cshtml
      ConfirmEmail.cshtml
      ExternalLoginConfirmation.cshtml
      ExternalLoginFailure.cshtml
      ForgotPassword.cshtml
      ForgotPasswordConfirmation.cshtml
      Login.cshtml
      Register.cshtml
      ResetPassword.cshtml
      ResetPasswordConfirmation.cshtml
      SendCode.cshtml
      VerifyCode.cshtml
  ◢ Batch
      Create.cshtml
      Delete.cshtml
      Details.cshtml
      Edit.cshtml
      Index.cshtml
  ◢ Course
      Create.cshtml
      Delete.cshtml
      Details.cshtml
      Edit.cshtml
      Index.cshtml
  ◢ Home
      About.cshtml
      Contact.cshtml
      Index.cshtml
  ◢ Login
      Index.cshtml
  ◢ Manage
      AddPhoneNumber.cshtml
      ChangePassword.cshtml
      Index.cshtml
      ManageLogins.cshtml
      SetPassword.cshtml
      VerifyPhoneNumber.cshtml
```



```
    Registration
      Create.cshtml
      Delete.cshtml
      Details.cshtml
      Edit.cshtml
      Index.cshtml
    Shared
      _Layout.cshtml
      _LoginPartial.cshtml
      Error.cshtml
      Lockout.cshtml
    User
      Create.cshtml
      Delete.cshtml
      Details.cshtml
      Edit.cshtml
      Index.cshtml
    _ViewStart.cshtml
    Web.config
  ApplicationInsights.config
  favicon.ico
  Global.asax
  packages.config
  Project_Readme.html
  Startup.cs
  Web.config
```

10. CONCLUSION

Student Management System can be used by educational institutions to maintain their student records easily. Achieving this objective is difficult using the manual system as the information is scattered, can be redundant and collecting relevant information may be very time-consuming.

All these problems are solved by this project. This system helps in maintaining the information of pupil of the organization. It can be easily accessed by the manager and kept safe for a long period of time without any changes.