**Step 1:**Creating database using code first approach.

Create new MVC empty project and also add another project with class library under Visual C#.  
  
In this process I have created a class user with the following fields.

1. **public** **class** User
2. {
3. [Key]
4. **public** **int** RegistrationId
5. {
6. **get**;
7. **set**;
8. } //This will be primary key column with auto increment
9. **public** **string** FirstName
10. {
11. **get**;
12. **set**;
13. }
14. **public** **string** LastName
15. {
16. **get**;
17. **set**;
18. }
19. **public** **string** UserName
20. {
21. **get**;
22. **set**;
23. }
24. **public** **string** EmailId
25. {
26. **get**;
27. **set**;
28. }
29. **public** **string** Password
30. {
31. **get**;
32. **set**;
33. }
34. **public** **string** Gender
35. {
36. **get**;
37. **set**;
38. }
39. **public** **string** VCode
40. {
41. **get**;
42. **set**;
43. }
44. **public** DateTime CreateDate
45. {
46. **get**;
47. **set**;
48. }
49. **public** DateTime ModifyDate
50. {
51. **get**;
52. **set**;
53. }
54. **public** **bool** Status
55. {
56. **get**;
57. **set**;
58. }
59. }

Create a class context as follows. Before creating this class install Entity Framework from NuGet Packages.

1. **public** **class** CmsDbContext : DbContext
2. {
3. **public** DbSet<User> ObjRegisterUser { **get**; **set**; } // Here User is the class
4. }

**Step 2:**Creating Helper class.

I have used Helper class instead of adding methods in the controllers.

1. **public** **static** **class** Helper
2. {
3. **public** **static** **string** ToAbsoluteUrl(**this** **string** relativeUrl) //Use absolute URL instead of adding phycal path for CSS, JS and Images
4. {
5. **if** (**string**.IsNullOrEmpty(relativeUrl)) **return** relativeUrl;
6. **if** (HttpContext.Current == **null**) **return** relativeUrl;
7. **if** (relativeUrl.StartsWith("/")) relativeUrl = relativeUrl.Insert(0, "~");
8. **if** (!relativeUrl.StartsWith("~/")) relativeUrl = relativeUrl.Insert(0, "~/");
9. var url = HttpContext.Current.Request.Url;
10. var port = url.Port != 80 ? (":" + url.Port) : String.Empty;
11. **return** String.Format("{0}://{1}{2}{3}", url.Scheme, url.Host, port, VirtualPathUtility.ToAbsolute(relativeUrl));
12. }
13. **public** **static** **string** GeneratePassword(**int** length) //length of salt
14. {
15. **const** **string** allowedChars = "abcdefghijkmnopqrstuvwxyzABCDEFGHJKLMNOPQRSTUVWXYZ0123456789";
16. var randNum = **new** Random();
17. var chars = **new** **char**[length];
18. var allowedCharCount = allowedChars.Length;
19. **for** (var i = 0; i <= length - 1; i++)
20. {
21. chars[i] = allowedChars[Convert.ToInt32((allowedChars.Length) \* randNum.NextDouble())];
22. }
23. **return** **new** **string**(chars);
24. }
25. **public** **static** **string** EncodePassword(**string** pass, **string** salt) //encrypt password
26. {
27. **byte**[] bytes = Encoding.Unicode.GetBytes(pass);
28. **byte**[] src = Encoding.Unicode.GetBytes(salt);
29. **byte**[] dst = **new** **byte**[src.Length + bytes.Length];
30. System.Buffer.BlockCopy(src, 0, dst, 0, src.Length);
31. System.Buffer.BlockCopy(bytes, 0, dst, src.Length, bytes.Length);
32. HashAlgorithm algorithm = HashAlgorithm.Create("SHA1");
33. **byte**[] inArray = algorithm.ComputeHash(dst);
34. //return Convert.ToBase64String(inArray);
35. **return** EncodePasswordMd5(Convert.ToBase64String(inArray));
36. }
37. **public** **static** **string** EncodePasswordMd5(**string** pass) //Encrypt using MD5
38. {
39. Byte[] originalBytes;
40. Byte[] encodedBytes;
41. MD5 md5;
42. //Instantiate MD5CryptoServiceProvider, get bytes for original password and compute hash (encoded password)
43. md5 = **new** MD5CryptoServiceProvider();
44. originalBytes = ASCIIEncoding.Default.GetBytes(pass);
45. encodedBytes = md5.ComputeHash(originalBytes);
46. //Convert encoded bytes back to a 'readable' string
47. **return** BitConverter.ToString(encodedBytes);
48. }
49. **public** **static** **string** base64Encode(**string** sData) // Encode
50. {
51. **try**
52. {
53. **byte**[] encData\_byte = **new** **byte**[sData.Length];
54. encData\_byte = System.Text.Encoding.UTF8.GetBytes(sData);
55. **string** encodedData = Convert.ToBase64String(encData\_byte);
56. **return** encodedData;
57. }
58. **catch** (Exception ex)
59. {
60. **throw** **new** Exception("Error in base64Encode" + ex.Message);
61. }
62. }
63. **public** **static** **string** base64Decode(**string** sData) //Decode
64. {
65. **try**
66. {
67. var encoder = **new** System.Text.UTF8Encoding();
68. System.Text.Decoder utf8Decode = encoder.GetDecoder();
69. **byte**[] todecodeByte = Convert.FromBase64String(sData);
70. **int** charCount = utf8Decode.GetCharCount(todecodeByte, 0, todecodeByte.Length);
71. **char**[] decodedChar = **new** **char**[charCount];
72. utf8Decode.GetChars(todecodeByte, 0, todecodeByte.Length, decodedChar, 0);
73. **string** result = **new** String(decodedChar);
74. **return** result;
75. }
76. **catch** (Exception ex)
77. {
78. **throw** **new** Exception("Error in base64Decode" + ex.Message);
79. }
80. }
81. }

**Step 3:**Changing Web.Config File.

1. <add name="CmsDbContext" connectionString="Data Source=(local);Initial Catalog=WebCMS;User ID=sa;Password=Admin@321;" providerName="System.Data.SqlClient" />

See the name is the name we have given in the context class i.e **CmsDbContext.**After this the database with name WebCMS and table as user with columns as per class parameters will be created after doing an Insert / Update / Delete operation.

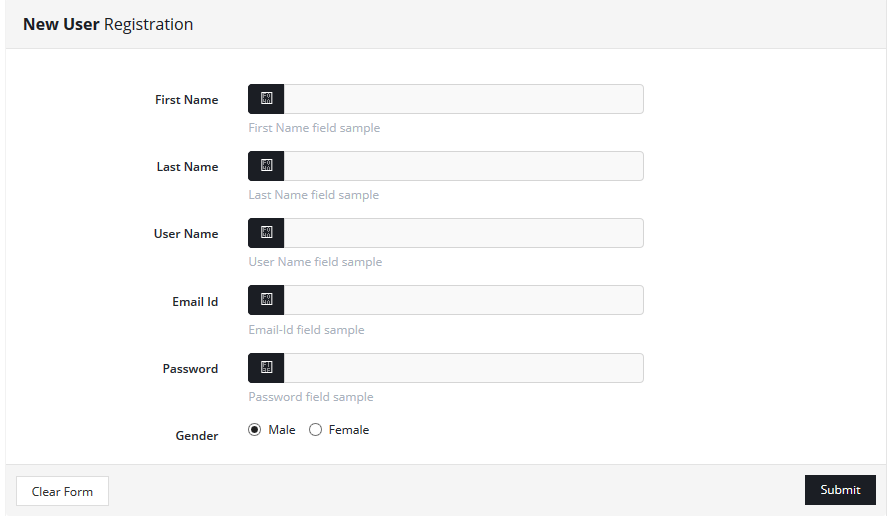
**Step 4:** Registration Page Design

1. <div **class**="panel panel-default mb0">
2. <div **class**="panel-heading ui-draggable-handle">
3. <h3 **class**="panel-title"><strong>New User </strong>  Registration</h3> </div> @**using** (Html.BeginForm("Registration", "Admin", **null**, FormMethod.Post)) { @Html.AntiForgeryToken()
4. <div **class**="panel-body">
5. <div **class**="form-group pt20">
6. <label **class**="col-md-3 col-xs-12 control-label align-right pt7">First Name</label>
7. <div **class**="col-md-6 col-xs-12">
8. <div **class**="input-group"> <span **class**="input-group-addon"><span **class**="fa fa-pencil"></span></span>
9. <input type="text" **class**="form-control form-group" required="" name="FirstName"> @\*Getting value by name and the name should be the name given **in** database column\*@ </div> <span **class**="help-block">First Name field sample</span> </div>
10. <div **class**="clearfix"></div>
11. </div>
12. <div **class**="form-group">
13. <label **class**="col-md-3 col-xs-12 control-label align-right pt7">Last Name</label>
14. <div **class**="col-md-6 col-xs-12">
15. <div **class**="input-group"> <span **class**="input-group-addon"><span **class**="fa fa-pencil"></span></span>
16. <input type="text" **class**="form-control form-group" required="" name="LastName"> @\*Getting value by name and the name should be the name given **in** database column\*@ </div> <span **class**="help-block">Last Name field sample</span> </div>
17. <div **class**="clearfix"></div>
18. </div>
19. <div **class**="form-group">
20. <label **class**="col-md-3 col-xs-12 control-label align-right pt7">User Name</label>
21. <div **class**="col-md-6 col-xs-12">
22. <div **class**="input-group"> <span **class**="input-group-addon"><span **class**="fa fa-pencil"></span></span>
23. <input type="text" **class**="form-control form-group" required="" name="UserName"> @\*Getting value by name and the name should be the name given **in** database column\*@ </div> <span **class**="help-block">User Name field sample</span> </div>
24. <div **class**="clearfix"></div>
25. </div>
26. <div **class**="form-group">
27. <label **class**="col-md-3 col-xs-12 control-label align-right pt7">Email Id</label>
28. <div **class**="col-md-6 col-xs-12">
29. <div **class**="input-group"> <span **class**="input-group-addon"><span **class**="fa fa-pencil"></span></span>
30. <input type="text" **class**="form-control form-group" required="" name="EmailId"> @\*Getting value by name and the name should be the name given **in** database column\*@ </div> <span **class**="help-block">Email-Id field sample</span> </div>
31. <div **class**="clearfix"></div>
32. </div>
33. <div **class**="form-group">
34. <label **class**="col-md-3 col-xs-12 control-label align-right pt7">Password</label>
35. <div **class**="col-md-6 col-xs-12">
36. <div **class**="input-group"> <span **class**="input-group-addon"><span **class**="fa fa-unlock-alt"></span></span>
37. <input type="password" **class**="form-control" required="" name="Password"> </div> <span **class**="help-block">Password field sample</span> </div>
38. <div **class**="clearfix"></div>
39. </div>
40. <div **class**="form-group">
41. <label **class**="col-md-3 col-xs-12 control-label align-right pt7">Gender</label>
42. <div **class**="col-md-6 col-xs-12">
43. <label **class**="radio-inline">
44. <input type="radio" **checked**="checked" value="Male" name="Gender">Male</label>
45. <label **class**="radio-inline">
46. <input type="radio" value="Female" name="Gender">Female</label>
47. </div>
48. <div **class**="clearfix"></div>
49. </div>
50. </div>
51. <div **class**="panel-footer">
52. <input type="reset" value="Clear Form" name="btnReset" **class**="btn btn-default" />
53. <input type="submit" id="btnSubmit" name="btnSubmit" value="Submit" **class**="btn btn-primary pull-right" /> </div> }
54. </div>

I have created a Controller for Registration with Name as Admin and added the following codes,

1. **public** ActionResult Registration()
2. {
3. **return** View();
4. }
5. [ValidateAntiForgeryToken]
6. [HttpPost]
7. **public** ActionResult Registration(User objNewUser)
8. {
9. **try**
10. {
11. **using**(var context = **new** CmsDbContext())
12. {
13. var chkUser = (from s **in** context.ObjRegisterUser where s.UserName == objNewUser.UserName || s.EmailId == objNewUser.EmailId select s).FirstOrDefault();
14. **if** (chkUser == **null**)
15. {
16. var keyNew = Helper.GeneratePassword(10);
17. var password = Helper.EncodePassword(objNewUser.Password, keyNew);
18. objNewUser.Password = password;
19. objNewUser.CreateDate = DateTime.Now;
20. objNewUser.ModifyDate = DateTime.Now;
21. objNewUser.VCode = keyNew;
22. context.ObjRegisterUser.Add(objNewUser);
23. context.SaveChanges();
24. ModelState.Clear();
25. **return** RedirectToAction("LogIn", "Login");
26. }
27. ViewBag.ErrorMessage = "User Allredy Exixts!!!!!!!!!!";
28. **return** View();
29. }
30. }
31. **catch** (Exception e)
32. {
33. ViewBag.ErrorMessage = "Some exception occured" + e;
34. **return** View();
35. }
36. }

After this step the registration page will look like the following,



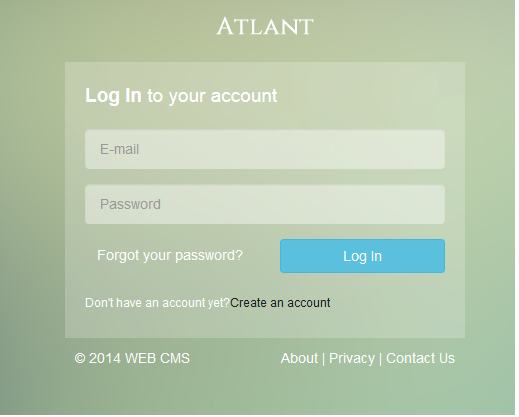
After this you can check your database and see the password column.  Also give the same password and see the password column you will find different password in your database table,

**Step 5:**Login Page Design

1. <div **class**="form-horizontal"> @**using** (Html.BeginForm("LogIn", "Login", **null**, FormMethod.Post)) { @Html.AntiForgeryToken()
2. <div **class**="form-group">
3. <div **class**="col-md-12">
4. <input type="text" **class**="form-control" required="" placeholder="E-mail" name="UserName" /> </div>
5. </div>
6. <div **class**="form-group">
7. <div **class**="col-md-12">
8. <input type="password" **class**="form-control" required="" placeholder="Password" name="Password" /> </div>
9. </div>
10. <div **class**="form-group">
11. <div **class**="col-md-6"> <a href="#" **class**="btn btn-link btn-block">Forgot your password?</a> </div>
12. <div **class**="col-md-6">
13. <button **class**="btn btn-info btn-block">Log In</button>
14. </div>
15. </div>
16. <div **class**="login-subtitle"> Don't have an account yet?@Html.ActionLink("Create an account", "Registration", "Admin") </div> }
17. </div>

I have created a controller for Login with Name Login and added the following code,

1. **public** ActionResult Login()
2. {
3. **return** View();
4. }
5. [ValidateAntiForgeryToken]
6. [HttpPost]
7. **public** ActionResult LogIn(**string** userName, **string** password)
8. {
9. **try**
10. {
11. **using**(var context = **new** CmsDbContext())
12. {
13. var getUser = (from s **in** context.ObjRegisterUser where s.UserName == userName || s.EmailId == userName select s).FirstOrDefault();
14. **if** (getUser != **null**)
15. {
16. var hashCode = getUser.VCode;
17. //Password Hasing Process Call Helper Class Method
18. var encodingPasswordString = Helper.EncodePassword(password, hashCode);
19. //Check Login Detail User Name Or Password
20. var query = (from s **in** context.ObjRegisterUser where(s.UserName == userName || s.EmailId == userName) && s.Password.Equals(encodingPasswordString) select s).FirstOrDefault();
21. **if** (query != **null**)
22. {
23. //RedirectToAction("Details/" + id.ToString(), "FullTimeEmployees");
24. //return View("../Admin/Registration"); url not change in browser
25. **return** RedirectToAction("Index", "Admin");
26. }
27. ViewBag.ErrorMessage = "Invallid User Name or Password";
28. **return** View();
29. }
30. ViewBag.ErrorMessage = "Invallid User Name or Password";
31. **return** View();
32. }
33. }
34. **catch** (Exception e)
35. {
36. ViewBag.ErrorMessage = " Error!!! contact cms@info.in";
37. **return** View();
38. }
39. }

After this step you will see the Login Page as in the following,  
  
   
  
Give proper user name, password and then you can login. Also you can download attached code from the attachment.