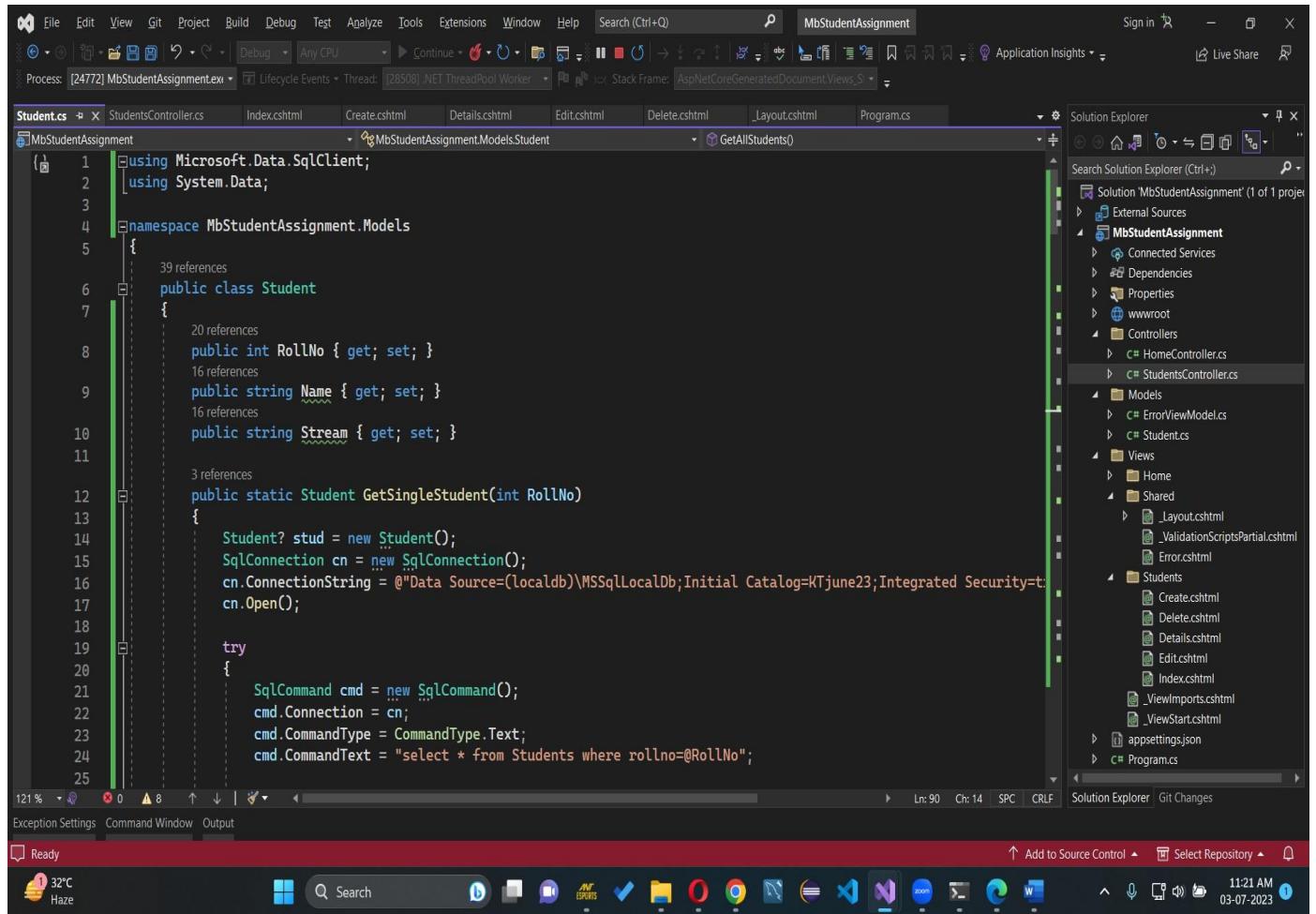
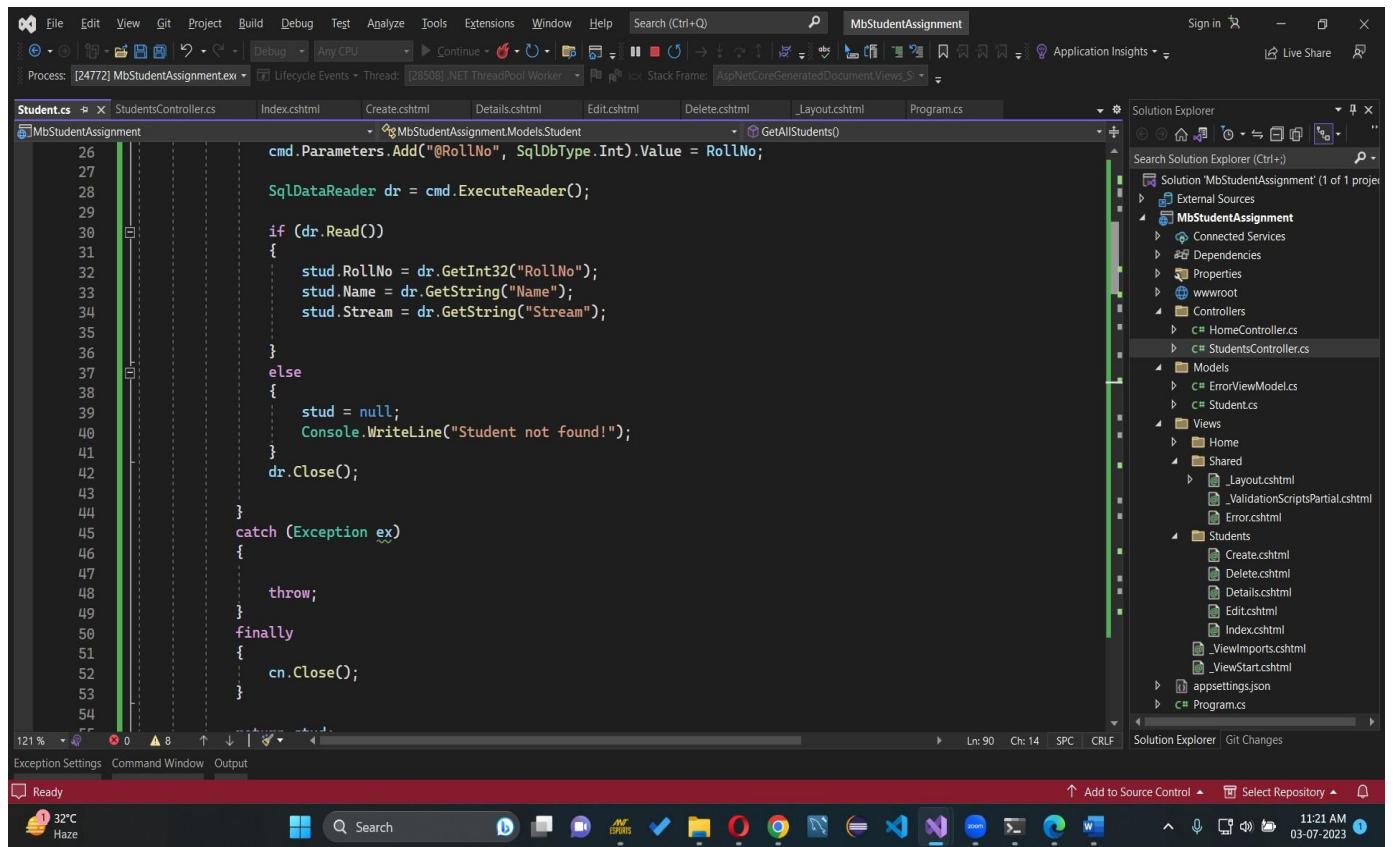


1. Create an MVC application for storing the details of student as well as display them, such as RollNo, Name, Stream, Marks, etc. These should be stored in a database and various operations such as Create Read Update and Delete must take place. The inputs must be entered by the user in a form

Student.cs



```
1  using Microsoft.Data.SqlClient;
2  using System.Data;
3
4  namespace MbStudentAssignment.Models
5  {
6      public class Student
7      {
8          public int RollNo { get; set; }
9          public string Name { get; set; }
10         public string Stream { get; set; }
11
12         public static Student GetSingleStudent(int RollNo)
13         {
14             Student? stud = new Student();
15             SqlConnection cn = new SqlConnection();
16             cn.ConnectionString = @"Data Source=(localdb)\MSSqlLocalDb;Initial Catalog=KTJune23;Integrated Security=True";
17             cn.Open();
18
19             try
20             {
21                 SqlCommand cmd = new SqlCommand();
22                 cmd.Connection = cn;
23                 cmd.CommandType = CommandType.Text;
24                 cmd.CommandText = "select * from Students where rollno=@RollNo";
25             }
26             catch (Exception ex)
27             {
28                 throw;
29             }
30             finally
31             {
32                 cn.Close();
33             }
34         }
35     }
36 }
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
```



```
cmd.Parameters.Add("@RollNo", SqlDbType.Int).Value = RollNo;
SqlDataReader dr = cmd.ExecuteReader();
if (dr.Read())
{
    stud.RollNo = dr.GetInt32("RollNo");
    stud.Name = dr.GetString("Name");
    stud.Stream = dr.GetString("Stream");
}
else
{
    stud = null;
    Console.WriteLine("Student not found!");
}
dr.Close();
}
catch (Exception ex)
{
    throw;
}
finally
{
    cn.Close();
}
```

This screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the code for the `GetAllStudents()` method in the `Student.cs` file. The code uses a `SqlDataReader` to fetch student data from a local database. The Solution Explorer on the right shows the project structure for 'MbStudentAssignment'.

```
55     return stud;
56 }
57
58     1 reference
59     public static List<Student> GetAllStudents()
60     {
61         List<Student> lstStuds = new List<Student>();
62         SqlConnection cn = new SqlConnection();
63         cn.ConnectionString = @"Data Source=(localdb)\MSSqlLocalDb;Initial Catalog=KTJune23;Integrated Security=True";
64         cn.Open();
65
66         try
67         {
68             //SqlCommand cmd = cn.CreateCommand();
69             SqlCommand cmd = new SqlCommand();
70             cmd.Connection = cn;
71             cmd.CommandType = CommandType.Text;
72             cmd.CommandText = "select * from Students";
73
74             SqlDataReader dr = cmd.ExecuteReader();
75             Student stud;
76
77             while (dr.Read())
78             {
79                 stud = new Student();
80                 stud.RollNo = dr.GetInt32("RollNo");
81                 stud.Name = dr.GetString("Name");
82                 stud.Stream = dr.GetString("Stream");
83                 lstStuds.Add(stud);
84             }
85
86             dr.Close();
87
88         }
89         catch (Exception ex)
90         {
91             throw;
92         }
93         finally
94         {
95             cn.Close();
96         }
97
98         return lstStuds;
99     }
100
101
102     1 reference
103     public static void InsertStudent(Student obj)
104     {
105         SqlConnection cn = new SqlConnection();
106         cn.ConnectionString = @"Data Source=(localdb)\MSSqlLocalDb;Initial Catalog=KTJune23;Integrated Security=True";
107         cn.Open();
108
109         try
110         {
111             //SqlCommand cmd = cn.CreateCommand();
112             SqlCommand cmd = new SqlCommand();
113             cmd.Connection = cn;
114             cmd.CommandType = CommandType.Text;
115             cmd.CommandText = "insert into Students values(@Name, @RollNo, @Stream)";
116             cmd.Parameters.AddWithValue("@Name", obj.Name);
117             cmd.Parameters.AddWithValue("@RollNo", obj.RollNo);
118             cmd.Parameters.AddWithValue("@Stream", obj.Stream);
119             cmd.ExecuteNonQuery();
120         }
121         catch (Exception ex)
122         {
123             throw;
124         }
125         finally
126         {
127             cn.Close();
128         }
129
130         return;
131     }
132 }
```

This screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the code for the `InsertStudent()` method in the `Student.cs` file. The code uses a `SqlCommand` to insert a new student record into the database. The Solution Explorer on the right shows the project structure for 'MbStudentAssignment'.

```
82             stud.Stream = dr.GetString("Stream");
83             lstStuds.Add(stud);
84         }
85
86         dr.Close();
87
88     }
89     catch (Exception ex)
90     {
91
92         throw;
93     }
94     finally
95     {
96         cn.Close();
97     }
98
99     return lstStuds;
100
101
102     1 reference
103     public static void InsertStudent(Student obj)
104     {
105         SqlConnection cn = new SqlConnection();
106         cn.ConnectionString = @"Data Source=(localdb)\MSSqlLocalDb;Initial Catalog=KTJune23;Integrated Security=True";
107         cn.Open();
108
109         try
110         {
111             //SqlCommand cmd = cn.CreateCommand();
112             SqlCommand cmd = new SqlCommand();
113             cmd.Connection = cn;
114             cmd.CommandType = CommandType.Text;
115             cmd.CommandText = "insert into Students values(@Name, @RollNo, @Stream)";
116             cmd.Parameters.AddWithValue("@Name", obj.Name);
117             cmd.Parameters.AddWithValue("@RollNo", obj.RollNo);
118             cmd.Parameters.AddWithValue("@Stream", obj.Stream);
119             cmd.ExecuteNonQuery();
120         }
121         catch (Exception ex)
122         {
123             throw;
124         }
125         finally
126         {
127             cn.Close();
128         }
129
130         return;
131     }
132 }
```

This screenshot shows the Microsoft Visual Studio interface. The main area is the code editor displaying a C# file named 'Student.cs'. The code implements a static method 'UpdateStudent' that inserts or updates a student record in a database. The 'Solution Explorer' pane on the right shows the project structure for 'MbStudentAssignment'.

```
110 //SqlCommand cmd = cn.CreateCommand();
111 SqlCommand cmd = new SqlCommand();
112 cmd.Connection = cn;
113 cmd.CommandType = CommandType.Text;
114 cmd.CommandText = "Insert into Students values(@RollNo,@Name,@Stream)";
115
116 cmd.Parameters.AddWithValue("@RollNo", obj.RollNo);
117 cmd.Parameters.AddWithValue("@Name", obj.Name);
118 cmd.Parameters.AddWithValue("@Stream", obj.Stream);
119
120 cmd.ExecuteNonQuery();
121
122 }
123 catch (Exception ex)
124 {
125
126     throw;
127 }
128 finally
129 {
130     cn.Close();
131 }
132
133
134 public static void UpdateStudent(Student obj)
135 {
136
137     SqlConnection cn = new SqlConnection();
138     cn.ConnectionString = @"Data Source=(localdb)\MSSqlLocalDb;Initial Catalog=KTjune23;Integrated Security=True";
139
140     cn.Open();
141     try
142     {
143         SqlCommand cmd = new SqlCommand();
144         cmd.Connection = cn;
145         cmd.CommandType = CommandType.Text;
146         cmd.CommandText = "Update Students set Name=@Name,Stream=@Stream where RollNo=@RollNo";
147
148
149         cmd.Parameters.AddWithValue("@RollNo", obj.RollNo);
150         cmd.Parameters.AddWithValue("@Name", obj.Name);
151         cmd.Parameters.AddWithValue("@Stream", obj.Stream);
152
153
154         int rowsAffected = cmd.ExecuteNonQuery();
155
156     }
157     catch (Exception ex)
158     {
159
160         throw;
161
162     }
163
164 }
```

This screenshot shows the Microsoft Visual Studio interface. The main area is the code editor displaying a C# file named 'Student.cs'. The code implements a static method 'UpdateStudent' that inserts or updates a student record in a database. The 'Solution Explorer' pane on the right shows the project structure for 'MbStudentAssignment'.

```
137 SqlConnection cn = new SqlConnection();
138 cn.ConnectionString = @"Data Source=(localdb)\MSSqlLocalDb;Initial Catalog=KTjune23;Integrated Security=True";
139
140 cn.Open();
141 try
142 {
143     SqlCommand cmd = new SqlCommand();
144     cmd.Connection = cn;
145     cmd.CommandType = CommandType.Text;
146     cmd.CommandText = "Update Students set Name=@Name,Stream=@Stream where RollNo=@RollNo";
147
148
149     cmd.Parameters.AddWithValue("@RollNo", obj.RollNo);
150     cmd.Parameters.AddWithValue("@Name", obj.Name);
151     cmd.Parameters.AddWithValue("@Stream", obj.Stream);
152
153
154     int rowsAffected = cmd.ExecuteNonQuery();
155
156 }
157 catch (Exception ex)
158 {
159
160     throw;
161
162 }
163
164 }
```

The screenshot shows the Microsoft Visual Studio interface. The code editor displays the `StudentsController.cs` file, which contains C# code for updating student records in a database. The Solution Explorer on the right shows the project structure for 'MbStudentAssignment'.

```
137     SqlConnection cn = new SqlConnection();
138     cn.ConnectionString = @"Data Source=(localdb)\MSSqlLocalDb;Initial Catalog=KTjune23;Integrated Security=True";
139
140     cn.Open();
141     try
142     {
143         SqlCommand cmd = new SqlCommand();
144         cmd.Connection = cn;
145         cmd.CommandType = CommandType.Text;
146         cmd.CommandText = "Update Students set Name=@Name,Stream=@Stream where RollNo=@RollNo";
147
148
149         cmd.Parameters.AddWithValue("@RollNo", obj.RollNo);
150         cmd.Parameters.AddWithValue("@Name", obj.Name);
151         cmd.Parameters.AddWithValue("@Stream", obj.Stream);
152
153
154         int rowsAffected = cmd.ExecuteNonQuery();
155
156     }
157     catch (Exception ex)
158     {
159         throw;
160     }
161
162
163     finally { cn.Close(); }
164
165 }
```

Exception Settings Command Window Output

Ready 32°C Haze 11:25 AM 03-07-2023

StudentsController.cs

The screenshot shows the Microsoft Visual Studio interface. The code editor displays the `StudentsController.cs` file, which contains C# code for managing student data. The Solution Explorer on the right shows the project structure for 'MbStudentAssignment'.

```
1 using MbStudentAssignment.Models;
2 using Microsoft.AspNetCore.Http;
3 using Microsoft.AspNetCore.Mvc;
4
5 namespace MbStudentAssignment.Controllers
6 {
7     public class StudentsController : Controller
8     {
9         // GET: StudentsController
10        public ActionResult Index()
11        {
12            List<Student> students = Student.GetAllStudents();
13            return View(students);
14        }
15
16        // GET: StudentsController/Details/5
17        public ActionResult Details(int id)
18        {
19            Student obj = Student.GetSingleStudent(id);
20            return View(obj);
21        }
22
23        // GET: StudentsController/Create
24        public ActionResult Create()
25        {
26            return View();
27        }
28
29    }
30 }
```

121% No issues found

Exception Settings Command Window Output

Ready 32°C Haze 11:28 AM 03-07-2023

This screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the `StudentsController.cs` file under the `MbStudentAssignment` project. The code implements the `Create` and `Edit` actions for managing students. The `Create` action handles a POST request, inserts a new student into the database, and returns a success message. The `Edit` action handles a GET request for a specific student by ID and returns the student object for editing. The Solution Explorer on the right shows the project structure, including controllers like `HomeController.cs` and `StudentsController.cs`, and views such as `Index.cshtml` and `Create.cshtml`. The status bar at the bottom indicates the code is 121% complete with no issues found.

```
27     }
28
29     // POST: StudentsController/Create
30     [HttpPost]
31     [ValidateAntiForgeryToken]
32     public ActionResult Create(Student obj, IFormCollection collection)
33     {
34         try
35         {
36             Student.InsertStudent(obj);
37             ViewBag.message = "Success!";
38             return View();
39             //return RedirectToAction(nameof(Index));
40         }
41         catch (Exception ex)
42         {
43             ViewBag.message = ex.Message;
44             return View();
45         }
46     }
47
48     // GET: StudentsController/Edit/5
49     public ActionResult Edit(int? id)
50     {
51         if (id == null)
52             return NotFound();
53         Student obj = Student.GetSingleStudent(id.Value);
54         return View(obj);
55     }
56
57     // POST: StudentsController/Edit/5
58     [HttpPost]
59     [ValidateAntiForgeryToken]
60     public ActionResult Edit(int id, Student obj)
61     {
62         try
63         {
64             Student.UpdateStudent(obj);
65             return RedirectToAction(nameof(Index));
66         }
67         catch (Exception ex)
68         {
69             ViewBag.message = ex.Message;
70             return View();
71         }
72     }
73
74     // GET: StudentsController/Delete/5
75     public ActionResult Delete(int? id)
76     {
77         if (id == null)
78             return NotFound();
79         Student obj = Student.GetSingleStudent(id.Value);
80         return View(obj);
81     }

```

This screenshot shows the Microsoft Visual Studio IDE interface, identical to the one above but with a different set of code. The main window displays the `StudentsController.cs` file under the `MbStudentAssignment` project. The code implements the `Edit` and `Delete` actions for managing students. The `Edit` action handles a POST request, updates the student in the database, and returns a success message. The `Delete` action handles a GET request for a specific student by ID and returns the student object for deletion. The Solution Explorer on the right shows the project structure, including controllers like `HomeController.cs` and `StudentsController.cs`, and views such as `Index.cshtml` and `Delete.cshtml`. The status bar at the bottom indicates the code is 121% complete with no issues found.

```
54     return View(obj);
55 }
56
57     // POST: StudentsController/Edit/5
58     [HttpPost]
59     [ValidateAntiForgeryToken]
60     public ActionResult Edit(int id, Student obj)
61     {
62         try
63         {
64             Student.UpdateStudent(obj);
65             return RedirectToAction(nameof(Index));
66         }
67         catch (Exception ex)
68         {
69             ViewBag.message = ex.Message;
70             return View();
71         }
72     }
73
74     // GET: StudentsController/Delete/5
75     public ActionResult Delete(int? id)
76     {
77         if (id == null)
78             return NotFound();
79         Student obj = Student.GetSingleStudent(id.Value);
80         return View(obj);
81     }

```

```
76     if (id == null)
77         return NotFound();
78     Student obj = Student.GetSingleStudent(id.Value);
79     return View(obj);
80 }
81
82 // POST: StudentsController/Delete/5
83 [HttpPost]
84 [ValidateAntiForgeryToken]
85 0 references
86 public ActionResult Delete(int id, Student obj)
87 {
88     try
89     {
90         Student.DeleteStudent(id);
91         return RedirectToAction(nameof(Index));
92     }
93     catch (Exception ex)
94     {
95         ViewBag.message = ex.Message;
96         return View();
97     }
98 }
99 }
100 }
```

133 % No issues found | Ln: 2 Ch: 33 SPC CRLF

Exception Settings Command Window Output

Ready 32°C Haze Add to Source Control Select Repository 11:30 AM 03-07-2023

Program.cs

```
1 namespace MbStudentAssignment
2 {
3     0 references
4     public class Program
5     {
6         0 references
7         public static void Main(string[] args)
8         {
9             var builder = WebApplication.CreateBuilder(args);
10
11             // Add services to the container.
12             builder.Services.AddControllersWithViews();
13
14             var app = builder.Build();
15
16             // Configure the HTTP request pipeline.
17             if (app.Environment.IsDevelopment())
18             {
19                 app.UseExceptionHandler("/Home/Error");
20             }
21             app.UseStaticFiles();
22
23             app.UseRouting();
24
25             app.UseAuthorization();
26
27             app.MapControllerRoute(
28                 name: "default",
29                 pattern: "{controller=Students}/{action=Index}/{id?}");
30
31             app.Run();
32         }
33     }
34 }
```

100 % No issues found | Ln: 27 Ch: 62 SPC CRLF

Solution Explorer Git Changes

_Layout.cshtml

This screenshot shows the Visual Studio IDE interface with the _Layout.cshtml file open in the main editor window. The code displays the structure of a Bootstrap-based navigation bar. The Solution Explorer on the right shows the project structure for 'MbStudentAssignment'.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>@ ViewData["Title"] - MbStudentAssignment</title>
        <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
        <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
        <link rel="stylesheet" href="~/MbStudentAssignment.styles.css" asp-append-version="true" />
    </head>
    <body>
        <header>
            <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-info border-bottom box-shadow mb-3">
                <div class="container-fluid">
                    <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">StudentsDetails</a>
                    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" aria-expanded="false" aria-label="Toggle navigation">
                        <span class="navbar-toggler-icon"></span>
                    </button>
                    <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
                        <ul class="navbar-nav flex-grow-1">
                            <li class="nav-item">
                                <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
                            </li>
                            <li class="nav-item">
                                <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
                            </li>
                        </ul>
                    </div>
                </div>
            </nav>
        </header>
        <div class="container">
            <main role="main" class="pb-3">
                @RenderBody()
            </main>
        </div>
        <footer class="border-top footer text-muted">
            <div class="container">
                &copy; 2023 - MbStudentAssignment - <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
            </div>
        </footer>
        <script src="~/lib/jquery/dist/jquery.min.js"></script>
        <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
        <script src="~/js/site.js" asp-append-version="true"></script>
        @await RenderSectionAsync("Scripts", required: false)
    </body>
</html>
```

This screenshot shows the Visual Studio IDE interface with the _Layout.cshtml file open in the main editor window. The code is identical to the one in the previous screenshot, displaying the navigation bar structure and footer information. The Solution Explorer on the right shows the project structure for 'MbStudentAssignment'.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>@ ViewData["Title"] - MbStudentAssignment</title>
        <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
        <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
        <link rel="stylesheet" href="~/MbStudentAssignment.styles.css" asp-append-version="true" />
    </head>
    <body>
        <header>
            <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-info border-bottom box-shadow mb-3">
                <div class="container-fluid">
                    <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
                </div>
            </nav>
        </header>
        <div class="container">
            <main role="main" class="pb-3">
                @RenderBody()
            </main>
        </div>
        <footer class="border-top footer text-muted">
            <div class="container">
                &copy; 2023 - MbStudentAssignment - <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
            </div>
        </footer>
        <script src="~/lib/jquery/dist/jquery.min.js"></script>
        <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
        <script src="~/js/site.js" asp-append-version="true"></script>
        @await RenderSectionAsync("Scripts", required: false)
    </body>
</html>
```

Index.cshtml

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the `Index.cshtml` file under the `Views/Students` folder. The code implements a table-based student list with columns for RollNo, Name, Stream, and actions (Edit, Details, Delete). The `Layout.cshtml` file is referenced at the top. The Solution Explorer on the right shows the project structure, including the `MbStudentAssignment` solution, `Controllers`, `Models`, and `Views` folders. The status bar at the bottom indicates the date as 03-07-2023 and the time as 11:32 AM.

```
1 @model IEnumerable<MbStudentAssignment.Models.Student>
2
3 @{
4     ViewData["Title"] = "Index";
5 }
6
7 <h1>Student List</h1>
8
9 <p>
10    <a href="#" asp-action="Create">Add New Student</a>
11 </p>
12 <table class="table">
13     <thead>
14         <tr>
15             <th>
16                 @Html.DisplayNameFor(model => model.RollNo)
17             </th>
18             <th>
19                 @Html.DisplayNameFor(model => model.Name)
20             </th>
21             <th>
22                 @Html.DisplayNameFor(model => model.Stream)
23             </th>
24             <th></th>
25         </tr>
26     </thead>
27     <tbody>
28         @foreach (var item in Model) {
29             <tr>
30                 <td>
31                     @Html.DisplayFor(modelItem => item.RollNo)
32                 </td>
33                 <td>
34                     @Html.DisplayFor(modelItem => item.Name)
35                 </td>
36                 <td>
37                     @Html.DisplayFor(modelItem => item.Stream)
38                 </td>
39                 <td>
40                     @Html.ActionLink("Edit", "Edit", new { id=item.RollNo }) |
41                     @Html.ActionLink("Details", "Details", new { id=item.RollNo }) |
42                     @Html.ActionLink("Delete", "Delete", new { id=item.RollNo })
43                 </td>
44             </tr>
45         }
46     </tbody>
47 </table>
```

This screenshot shows the same Visual Studio environment after modifications. The `Index.cshtml` file now uses the `Student` model instead of the `Student` interface. The code structure remains largely the same, displaying student data in a table with edit, details, and delete links. The Solution Explorer and status bar are identical to the first screenshot.

```
22             <th>
23                 @Html.DisplayNameFor(model => model.Stream)
24             </th>
25         </thead>
26         <tbody>
27             @foreach (var item in Model) {
28                 <tr>
29                     <td>
30                         @Html.DisplayFor(modelItem => item.RollNo)
31                     </td>
32                     <td>
33                         @Html.DisplayFor(modelItem => item.Name)
34                     </td>
35                     <td>
36                         @Html.DisplayFor(modelItem => item.Stream)
37                     </td>
38                     <td>
39                         @Html.ActionLink("Edit", "Edit", new { id=item.RollNo }) |
40                         @Html.ActionLink("Details", "Details", new { id=item.RollNo }) |
41                         @Html.ActionLink("Delete", "Delete", new { id=item.RollNo })
42                     </td>
43                 </tr>
44             }
45         </tbody>
46     </table>
```

Create.cshtml

This screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the `Create.cshtml` file under the `Views/Students` folder. The code implements a form for creating a student, using Bootstrap classes like `form-group`, `col-md-4`, and `text-danger` for validation. The `asp-validation-summary` tag is used to display validation errors at the top of the form. The `asp-for` attribute is used to bind input fields to properties in the `Student` model. The `asp-action="Create"` attribute on the `form` tag specifies the action URL. The `Program.cs` file is also visible in the Solution Explorer.

```
1 @model MbStudentAssignment.Models.Student
2
3 @{
4     ViewData["Title"] = "Create";
5 }
6
7 <h1>Create</h1>
8
9 <h4>Student</h4>
10 <hr />
11 <div class="row">
12     <div class="col-md-4">
13         <form asp-action="Create">
14             <div asp-validation-summary="ModelOnly" class="text-danger"></div>
15             <div class="form-group">
16                 <label asp-for="RollNo" class="control-label"></label>
17                 <input asp-for="RollNo" class="form-control" />
18                 <span asp-validation-for="RollNo" class="text-danger"></span>
19             </div>
20             <div class="form-group">
21                 <label asp-for="Name" class="control-label"></label>
22                 <input asp-for="Name" class="form-control" />
23                 <span asp-validation-for="Name" class="text-danger"></span>
24             </div>
25             <div class="form-group">
```

This screenshot shows the Microsoft Visual Studio IDE interface, similar to the previous one but with more code in the `Create.cshtml` file. The code now includes a message from the `@ViewBag.message` variable, a submit button, and a link to return to the list. It also contains a `@section Scripts { ... }` block for rendering partial scripts. The rest of the code structure remains the same, including the `asp-validation-summary` and `asp-for` attributes for validation.

```
25     <div class="form-group">
26         <label asp-for="Stream" class="control-label"></label>
27         <input asp-for="Stream" class="form-control" />
28         <span asp-validation-for="Stream" class="text-danger"></span>
29     </div>
30     <h4>
31         @ViewBag.message
32     </h4>
33     <div class="form-group">
34         <input type="submit" value="Create" class="btn btn-primary" />
35     </div>
36     </form>
37 </div>
38 </div>
39
40 <div>
41     <a asp-action="Index">Back to List</a>
42 </div>
43
44 @section Scripts {
45     @await Html.RenderPartialAsync("_ValidationScriptsPartial");
46 }
```

Details.cshtml

The screenshot shows the Visual Studio IDE with the Details.cshtml file open in the code editor. The code is a standard ASP.NET MVC view for displaying student details. It includes a title, a row-based table layout, and action links for edit and back to list.

```
1 @model MbStudentAssignment.Models.Student
2
3 @{
4     ViewData["Title"] = "Details";
5 }
6
7 <h1>Details</h1>
8
9 <div>
10    <h4>Student</h4>
11    <hr />
12    <dl class="row">
13        <dt class = "col-sm-2">
14            @Html.DisplayNameFor(model => model.RollNo)
15        </dt>
16        <dd class = "col-sm-10">
17            @Html.DisplayFor(model => model.RollNo)
18        </dd>
19        <dt class = "col-sm-2">
20            @Html.DisplayNameFor(model => model.Name)
21        </dt>
22        <dd class = "col-sm-10">
23            @Html.DisplayFor(model => model.Name)
24        </dd>
25        <dt class = "col-sm-2">
26            @Html.DisplayNameFor(model => model.Stream)
27        </dt>
28        <dd class = "col-sm-10">
29            @Html.DisplayFor(model => model.Stream)
30        </dd>
31    </dl>
32 </div>
33 <div>
34     @Html.ActionLink("Edit", "Edit", new { id=Model.RollNo }) |
35     <a href="#" asp-action="Index">Back to List</a>
36 </div>
37
```

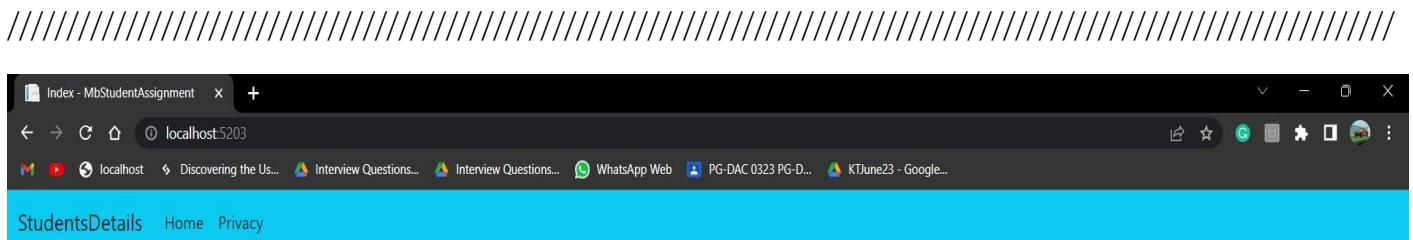
Edit.cshtml

The screenshot shows the Visual Studio IDE with the Edit.cshtml file open in the code editor. This is a form-based view for editing student information. It uses Bootstrap's grid system and includes validation messages for the RollNo, Name, and Stream fields.

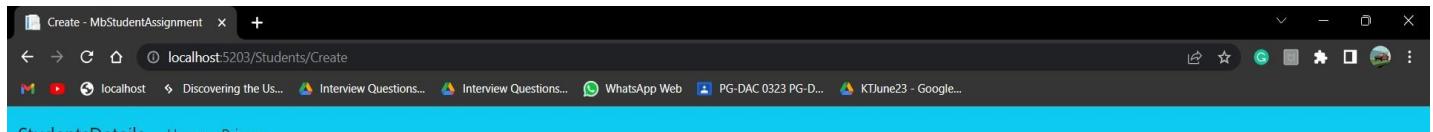
```
1 @model MbStudentAssignment.Models.Student
2
3 @{
4     ViewData["Title"] = "Edit";
5 }
6
7 <h1>Edit</h1>
8
9 <h4>Student</h4>
10 <hr />
11 <div class="row">
12     <div class="col-md-4">
13         <form asp-action="Edit">
14             <div asp-validation-summary="ModelOnly" class="text-danger"></div>
15             <div class="form-group">
16                 <label asp-for="RollNo" class="control-label"></label>
17                 <input asp-for="RollNo" class="form-control" />
18                 <span asp-validation-for="RollNo" class="text-danger"></span>
19             </div>
20             <div class="form-group">
21                 <label asp-for="Name" class="control-label"></label>
22                 <input asp-for="Name" class="form-control" />
23                 <span asp-validation-for="Name" class="text-danger"></span>
24             </div>
25             <div class="form-group">
26                 <label asp-for="Stream" class="control-label"></label>
27                 <input asp-for="Stream" class="form-control" />
28                 <span asp-validation-for="Stream" class="text-danger"></span>
29             </div>
30             <h4>
31                 @ViewBag.message
32             </h4>
33             <div class="form-group">
34                 <input type="submit" value="Save" class="btn btn-primary" />
35             </div>
36         </form>
37     </div>
38 </div>
39
40 <div>
41     <a href="#" asp-action="Index">Back to List</a>
42 </div>
43
44 @section Scripts {
45     @await Html.RenderPartialAsync("_ValidationScriptsPartial");
46 }
```

Delete.cshtml

```
1 @model MbStudentAssignment.Models.Student
2
3 @{
4     ViewData["Title"] = "Delete";
5 }
6 <h1>Delete</h1>
7
8 <h3>Are you sure you want to delete this?</h3>
9 <div>
10    <h4>Student</h4>
11    <hr />
12    <dl class="row">
13        <dt class = "col-sm-2">
14            @Html.DisplayNameFor(model => model.RollNo)
15        </dt>
16        <dd class = "col-sm-10">
17            @Html.DisplayFor(model => model.RollNo)
18        </dd>
19        <dt class = "col-sm-2">
20            @Html.DisplayNameFor(model => model.Name)
21        </dt>
22        <dd class = "col-sm-10">
23            @Html.DisplayFor(model => model.Name)
24        </dd>
25        <dt class = "col-sm-2">
26            @Html.DisplayNameFor(model => model.Stream)
27        </dt>
28        <dd class = "col-sm-10">
29            @Html.DisplayFor(model => model.Stream)
30        </dd>
31    </dl>
32    <h4>
33        @ViewBag.message
34    </h4>
35    <form asp-action="Delete">
36        <input type="submit" value="Delete" class="btn btn-danger" /> |
37        <a asp-action="Index">Back to List</a>
38        <input type="hidden" asp-for="RollNo" class="form-control" />
39    </form>
40 </div>
```



Student List			
Add New Student			
RollNo	Name	Stream	
1	Tejas C	PGDAC	Edit Details Delete
2	Tejas S	PGDAC	Edit Details Delete
3	Tushar C	PGDAC	Edit Details Delete
4	Shweta J	PGDAC	Edit Details Delete
5	Yogesh R	PGDAC	Edit Details Delete



Create

Student

RollNo

6

Name

Adam

Stream

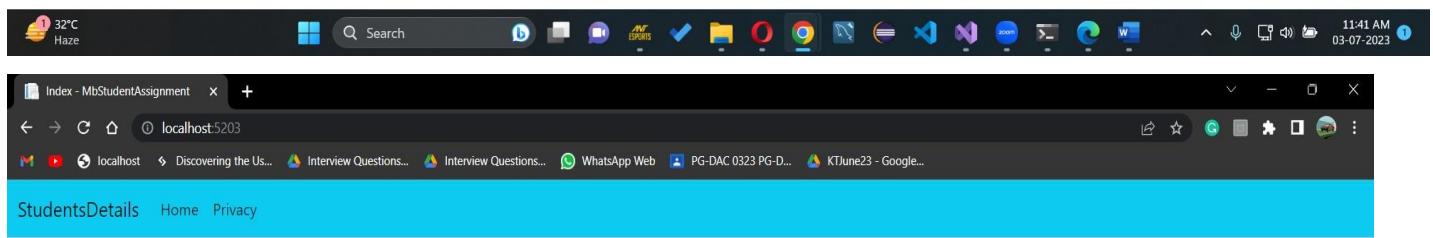
PGDCSF

Success!

[Create](#)

[Back to List](#)

© 2023 - MbStudentAssignment - [Privacy](#)



Student List

[Add New Student](#)

RollNo	Name	Stream	
1	Tejas C	PGDAC	Edit Details Delete
2	Tejas S	PGDAC	Edit Details Delete
3	Tushar C	PGDAC	Edit Details Delete
4	Shweta J	PGDAC	Edit Details Delete
5	Yogesh R	PGDAC	Edit Details Delete
6	Adam	PGDCSF	Edit Details Delete

© 2023 - MbStudentAssignment - [Privacy](#)



The screenshot shows a web browser window titled "Index - MbStudentAssignment". The address bar indicates the URL is "localhost:5203". The page content is titled "Student List" and includes a link "Add New Student". Below this is a table listing seven students with columns for RollNo, Name, and Stream. Each row contains links for Edit, Details, and Delete.

RollNo	Name	Stream	
1	Tejas C	PGDAC	Edit Details Delete
2	Tejas S	PGDAC	Edit Details Delete
3	Tushar C	PGDAC	Edit Details Delete
4	Shweta J	PGDAC	Edit Details Delete
5	Yogesh R	PGDAC	Edit Details Delete
6	Adam	PGDCSF	Edit Details Delete
7	Alan	PGDBDA	Edit Details Delete

© 2023 - MbStudentAssignment - [Privacy](#)

The screenshot shows a web browser window titled "Details - MbStudentAssignment". The address bar indicates the URL is "localhost:5203/Students/Details/6". The page content is titled "Details" and "Student". It displays the student information for RollNo 6, Name Adam, and Stream PGDCSF. At the bottom are links for "Edit" and "Back to List".

© 2023 - MbStudentAssignment - [Privacy](#)

This screenshot is identical to the one above, showing the "Details - MbStudentAssignment" page for student ID 6. The information displayed is the same: RollNo 6, Name Adam, and Stream PGDCSF. The "Edit" and "Back to List" links are also present at the bottom.

Index - MbStudentAssignment x PG Diploma Courses x | +

localhost 5203 Discovering the Us... Interview Questions... Interview Questions... WhatsApp Web PG-DAC 0323 PG-D... KTJune23 - Google...

StudentsDetails Home Privacy

Student List

[Add New Student](#)

RollNo	Name	Stream	
1	Tejas C	PGDAC	Edit Details Delete
2	Tejas S	PGDAC	Edit Details Delete
3	Tushar C	PGDAC	Edit Details Delete
4	Shweta J	PGDAC	Edit Details Delete
5	Yogesh R	PGDAC	Edit Details Delete
6	Adam	PGDCSF	Edit Details Delete
7	Alan	PGDBDA	Edit Details Delete
8	Rowan	PGDAI	Edit Details Delete

Edit - MbStudentAssignment x PG Diploma Courses x | +

localhost:5203/Students/Edit/8 Discovering the Us... Interview Questions... Interview Questions... WhatsApp Web PG-DAC 0323 PG-D... KTJune23 - Google...

StudentsDetails Home Privacy

Edit

Student

RollNo

Name

Stream

[Save](#)

[Back to List](#)

Index - MbStudentAssignment x PG Diploma Courses x | +

localhost 5203 Discovering the Us... Interview Questions... Interview Questions... WhatsApp Web PG-DAC 0323 PG-D... KTJune23 - Google...

StudentsDetails Home Privacy

Student List

[Add New Student](#)

RollNo	Name	Stream	
1	Tejas C	PGDAC	Edit Details Delete
2	Tejas S	PGDAC	Edit Details Delete
3	Tushar C	PGDAC	Edit Details Delete
4	Shweta J	PGDAC	Edit Details Delete
5	Yogesh R	PGDAC	Edit Details Delete
6	Adam	PGDCSF	Edit Details Delete
7	Alan	PGDBDA	Edit Details Delete
8	Rowan	PGDMC	Edit Details Delete

The screenshot shows a web browser window with the URL localhost:5203/Students/Delete/8. The page title is "Delete". A confirmation message asks, "Are you sure you want to delete this? Student". Below the message is a table with three rows: RollNo (8), Name (Rowan), and Stream (PGDMC). At the bottom are two buttons: a red "Delete" button and a blue "Back to List" button.

© 2023 - MbStudentAssignment - [Privacy](#)

2. Write a C# program that reads a number from the user and calculates its square root. Handle the exception if the number is negative.

```
namespace ConsoleApp1
{
    internal class Program
    {
        static void Main(string[] args)
        {
            try
            {
                Console.Write("Enter a number: ");
                double number = double.Parse(Console.ReadLine());

                if (number < 0)
                {
                    throw new ArgumentOutOfRangeException("Number cannot be negative.");
                }

                double squareRoot = Math.Sqrt(number);
                Console.WriteLine($"Square root of {number} is: {squareRoot}");
            }
            catch (FormatException)
            {
                Console.WriteLine("Invalid input. Please enter a valid number.");
            }
            catch (ArgumentOutOfRangeException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
}
```

Output:-

```
Enter a number: -6
Specified argument was out of the range of valid values. (Parameter 'Number cannot be negative.')
```

```
Enter a number: 4
Square root of 4 is: 2
```