Lab 6: Define MVC. Write a CRUD operation to display, insert, update and delete S information using ASP.NET Core MVC.

MVC stands for Model-View-Controller, which is a software design pattern used for developing web and desktop applications. It separates an application into three interconnected components to separate internal representations of information from the ways information is presented and accepted by the user. This helps in organizing code, making it more maintainable and scalable.

explanation of each component:

1. Model
   - Represents the data and the business logic of the application.
   - Handles data retrieval, storage, and manipulation.
   - Example: Database records, calculations, validation logic.
2. View
   - Represents the user interface (UI) of the application.
   - Displays data from the Model to the user.
   - Example: HTML pages, UI forms, charts.
3. Controller
   - Handles user input and interactions.
   - Acts as a bridge between Model and View.
   - Updates the Model based on user input and selects which View to display.

Flow:
User → Controller → Model → View → User

Source Code:

Program.cs

```
using Microsoft.EntityFrameworkCore;
using Microsoft.AspNetCore;
using MVC_CRUD.Data;
using Microsoft.EntityFrameworkCore.SqlServer;
namespace MVC_CRUD
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);
            // Add MVC services
            builder.Services.AddControllersWithViews();
            // Add DbContext
            builder.Services.AddDbContext<ApplicationDbContext>(options =>
                options.UseSqlServer(
                    builder.Configuration.GetConnectionString("DefaultConnection"))
            );
            var app = builder.Build();
            // Middleware
            if (!app.Environment.IsDevelopment())
            {
                app.UseExceptionHandler("/Home/Error");
            }
            app.UseStaticFiles();
            app.UseRouting();
            app.UseAuthorization();
            // Default route
            app.MapControllerRoute(
                name: "default",
                pattern: "{controller=Students}/{action=Index}/{id?}");
            app.Run();
        }
    }
}
```

Data/ApplicationDbContext.cs

```
using Microsoft.EntityFrameworkCore;
using MVC_CRUD.Models;
namespace MVC_CRUD.Data
{
    public class ApplicationDbContext : DbContext
    {
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
        : base(options)
        {
        }
        public DbSet<Student> Students { get; set; }
    }
}
```

appsetting.json

```
{
  "ConnectionStrings": {
    "DefaultConnection":
"Server=.\\SQLEXPRESS;Database=StudentDB;Trusted_Connection=True;TrustServerCertific
ate=True"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

Controller/StudentsController.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using MVC_CRUD.Data;
using MVC_CRUD.Models;
using System.Threading.Tasks;
using static MVC_CRUD.Models.Student;
namespace MVC_CRUD.Controllers
{
    public class StudentsController : Controller
    {
        private readonly ApplicationDbContext _context;

        public StudentsController(ApplicationDbContext context)
        {
            _context = context;
        }
        // READ
        public async Task<IActionResult> Index()
        {
            return View(await _context.Students.ToListAsync());
        }

        // CREATE - GET
        public IActionResult Create()
        {
            return View();
        }
        // CREATE - POST
        [HttpPost]
        public async Task<IActionResult> Create(Student student)
        {
            if (ModelState.IsValid)
            {
                _context.Students.Add(student);
                await _context.SaveChangesAsync();
                return RedirectToAction("Index");
            }
            return View(student);
        }
        // UPDATE - GET
        public async Task<IActionResult> Edit(int id)
        {
            var student = await _context.Students.FindAsync(id);
            return View(student);
        }
        // UPDATE - POST
        [HttpPost]
        public async Task<IActionResult> Edit(Student student)
        {
            if (ModelState.IsValid)
            {
                _context.Students.Update(student);
                await _context.SaveChangesAsync();
                return RedirectToAction("Index");
            }
            return View(student);
        }
        // DELETE - GET
        public async Task<IActionResult> Delete(int id)
        {
            var student = await _context.Students.FindAsync(id);
            return View(student);
        }
        // DELETE - POST
        [HttpPost, ActionName("Delete")]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            var student = await _context.Students.FindAsync(id);
            _context.Students.Remove(student);
            await _context.SaveChangesAsync();
            return RedirectToAction("Index");
        }
    }
}
```

Models/Student.cs

```
using System.ComponentModel.DataAnnotations;
namespace MVC_CRUD.Models
{
    public class Student
    {
        public int Id { get; set; }
        [Required]
        public string Name { get; set; }
        public int Age { get; set; }
        [EmailAddress]
        public string Email { get; set; }
    }
}
```

Views/Students/Index.cshtml

```
@model IEnumerable<Student>
<a asp-action="Create">Add Student</a>
<table border="1">
    <tr>
        <th>Name</th>
        <th>Age</th>
        <th>Email</th>
        <th>Actions</th>
    </tr>
    @foreach (var s in Model)
    {
        <tr>
            <td>@s.Name</td>
            <td>@s.Age</td>
            <td>@s.Email</td>
            <td>
                <a asp-action="Edit" asp-route-id="@s.Id">Edit</a> |
                <a asp-action="Delete" asp-route-id="@s.Id">Delete</a>
            </td>
        </tr>
    }
</table>
```

Views/Students/Create.cshtml

```
@model Student
<form asp-action="Create" method="post">
    Name: <input asp-for="Name" /><br />
    Age: <input asp-for="Age" /><br />
    Email: <input asp-for="Email" /><br />
    <button type="submit">Save</button>
</form>
```

Views/Students/Edit.cshtml

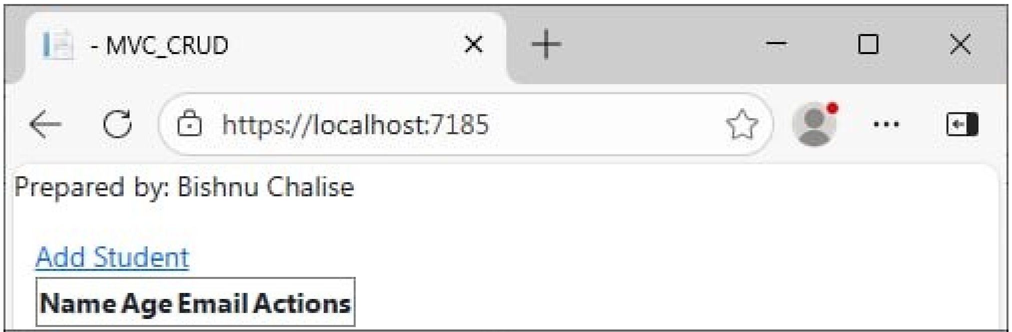
```
@model Student
<form asp-action="Edit" method="post">
    <input type="hidden" asp-for="Id" />
    Name: <input asp-for="Name" /><br />
    Age: <input asp-for="Age" /><br />
    Email: <input asp-for="Email" /><br />
    <button type="submit">Update</button>
</form>
```

Views/Students/Delete.cshtml

```
@model Student
<h3>Are you sure?</h3>
<form asp-action="Delete" method="post">
    <input type="hidden" asp-for="Id" />
    <button type="submit">Delete</button>
</form>
```

Views/Shared/_Layout.cshtml

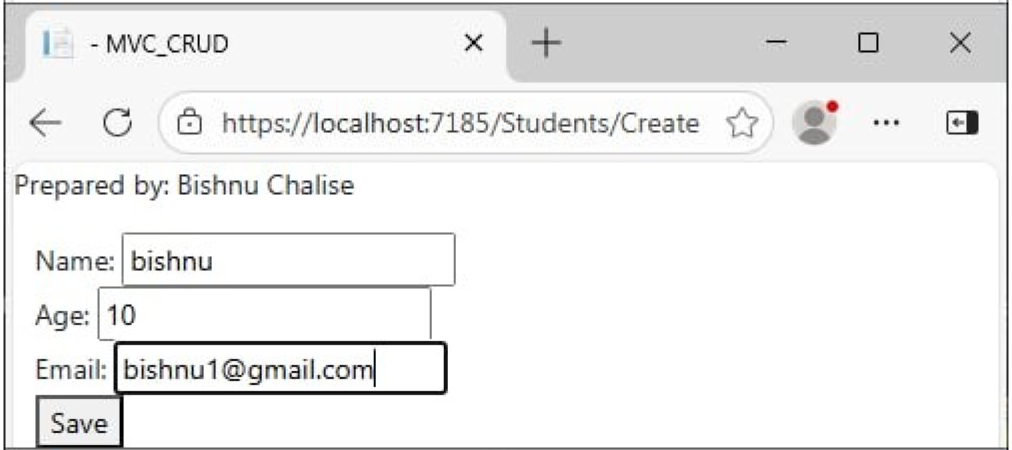
```
 <!DOCTYPE html>
 <html lang="en">
 <head>
     <meta charset="utf-8" />
     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
     <title>@ViewData["Title"] - MVC_CRUD</title>
     <script type="importmap"></script>
 </head>
 <body>
     <p>Prepared by: Bishnu Chalise</p>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>
<hr>

</body>
</html>
```



Prepared by: Bishnu Chalise

Add Student

Name	Age	Email	Actions

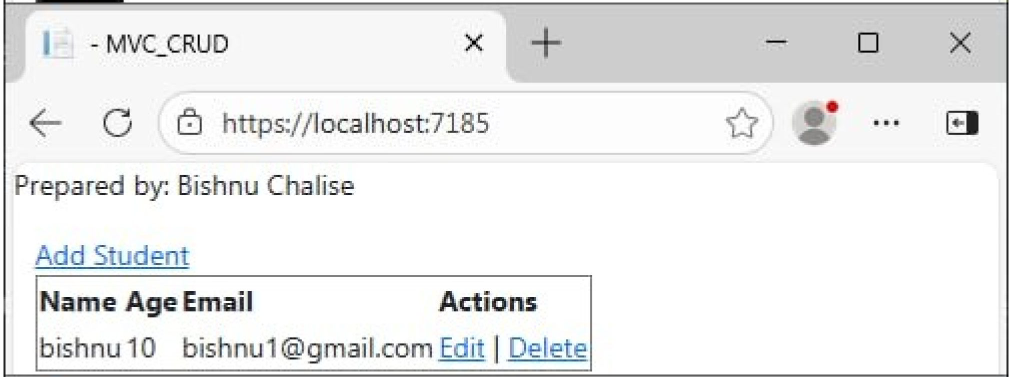


Prepared by: Bishnu Chalise

Name: bishnu

Age: 10

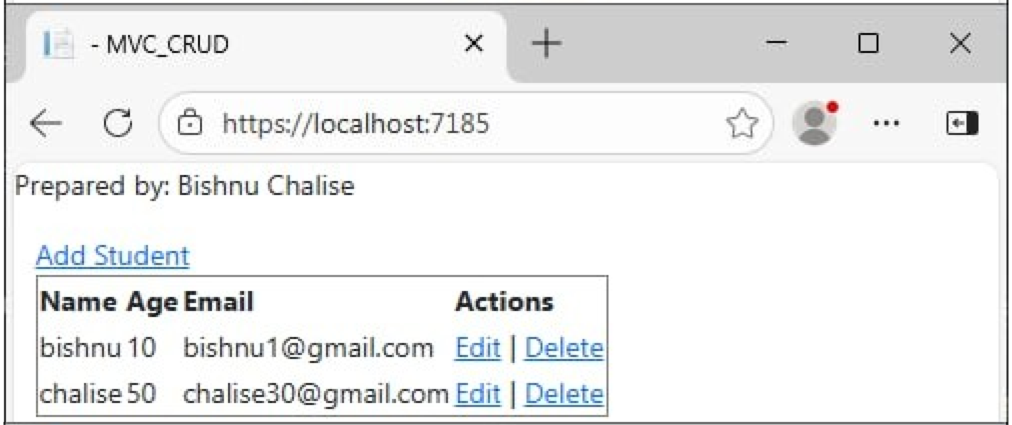
Email: bishnu1@gmail.com



Prepared by: Bishnu Chalise

Add Student

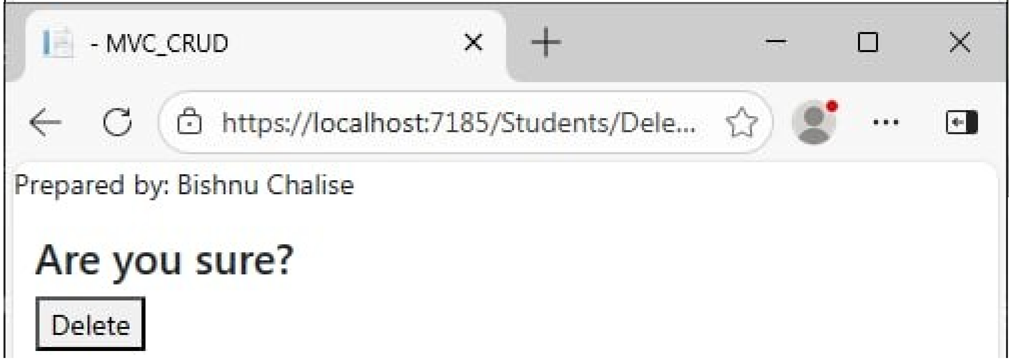
Name	Age	Email	Actions
bishnu	10	bishnu1@gmail.com	Edit | Delete



Prepared by: Bishnu Chalise

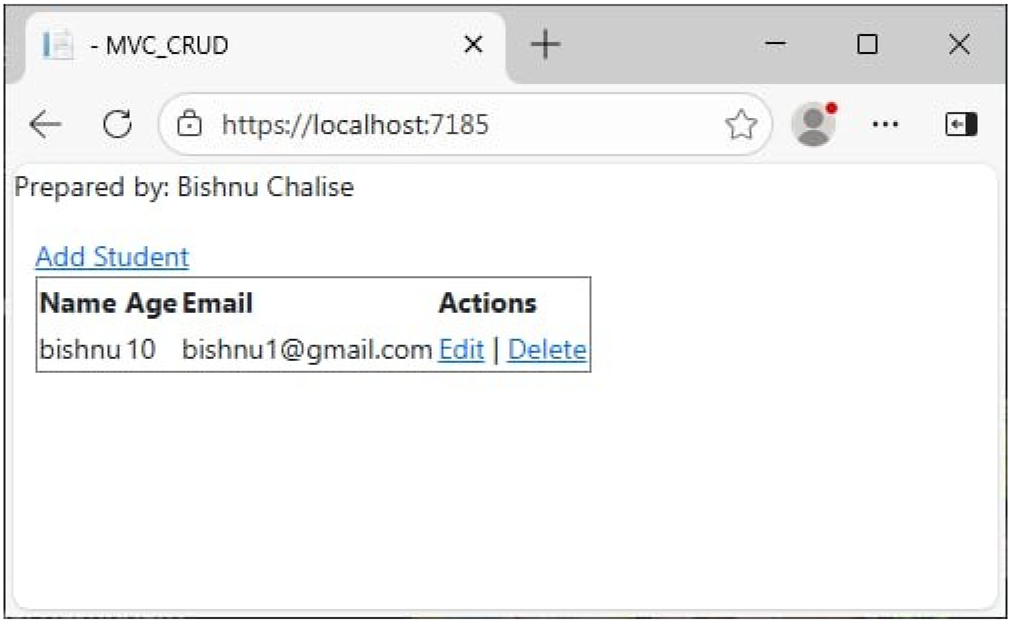
Add Student

Name	Age	Email	Actions
bishnu	10	bishnu1@gmail.com	Edit | Delete
chalise	50	chalise30@gmail.com	Edit | Delete



Prepared by: Bishnu Chalise

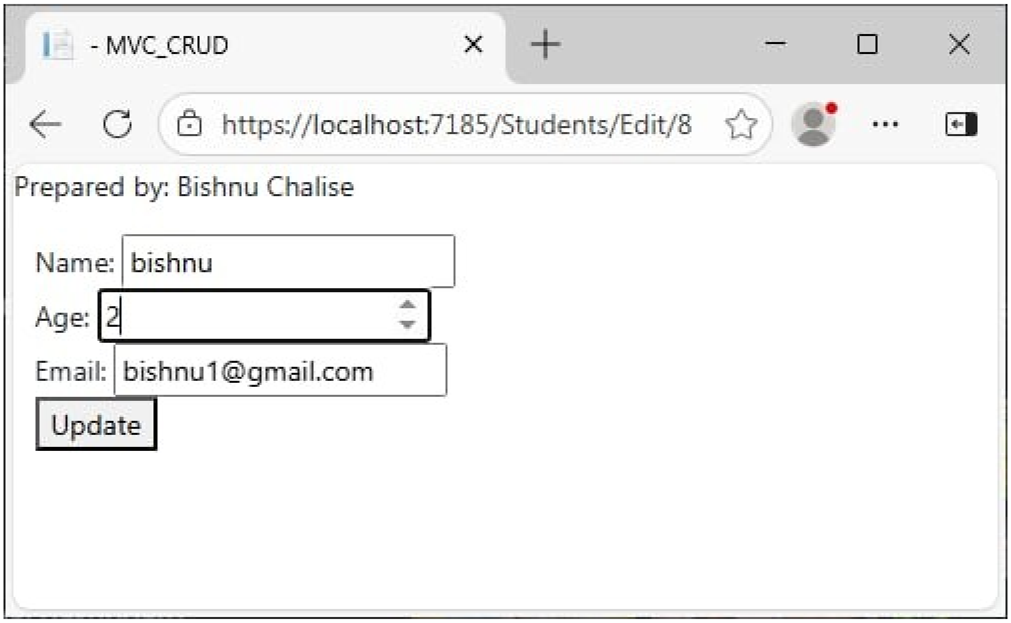
Are you sure?



Prepared by: Bishnu Chalise

Add Student

Name	Age	Email	Actions
bishnu	10	bishnu1@gmail.com	Edit | Delete

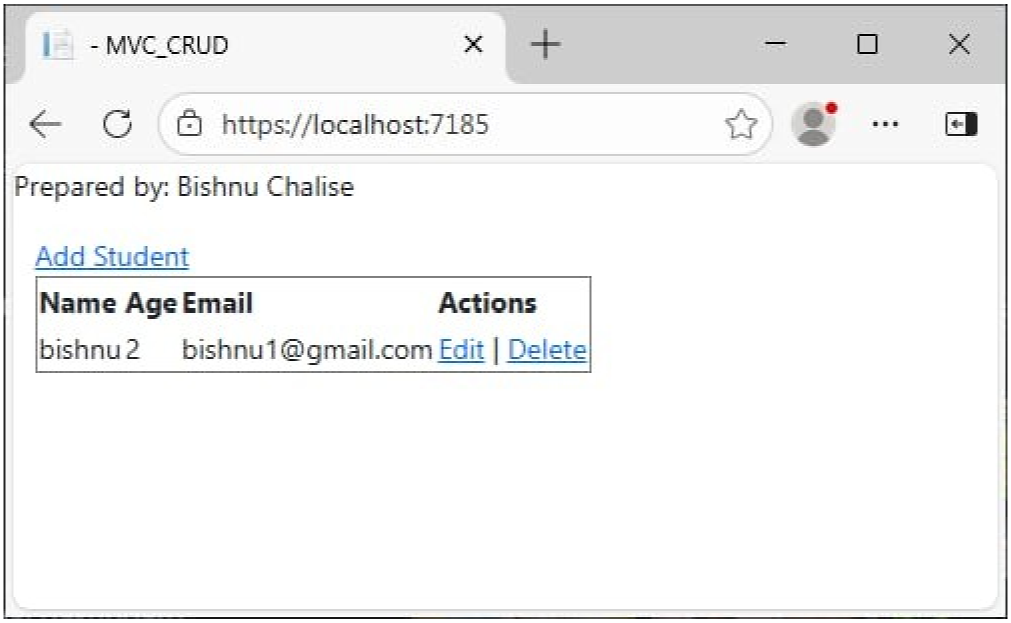


Prepared by: Bishnu Chalise

Name: bishnu

Age: 2

Email: bishnu1@gmail.com



Prepared by: Bishnu Chalise

Add Student

Name	Age	Email	Actions
bishnu	2	bishnu1@gmail.com	Edit | Delete

Lab 7: What is Data Annotation? Write a program to validate Player information when click on save button using MVC pattern.

Data Annotation is a technique in C# / ASP.NET used to add metadata to class properties. It helps in validating data, formatting how data is displayed, and sometimes in defining relationships between data fields. Essentially, it provides rules and instructions to the framework about how to handle the data in models.

Source Code:
Models/Player.cs

```
using System;
using System.ComponentModel.DataAnnotations;

namespace MyApp.Models
{
    public class Player
    {
        public int Id { get; set; }

        [Required(ErrorMessage = "Player name is required.")]
        [StringLength(50, ErrorMessage = "Name can't be longer than 50 characters.")]
        public string Name { get; set; }

        [Required(ErrorMessage = "Email is required.")]
        [EmailAddress(ErrorMessage = "Enter a valid email address.")]
        public string Email { get; set; }

        [Required(ErrorMessage = "Jersey number is required.")]
        [Range(1, 99, ErrorMessage = "Jersey number must be between 1 and 99.")]
        public int JerseyNumber { get; set; }

        [Required(ErrorMessage = "Position is required.")]
        [StringLength(30)]
        public string Position { get; set; }

        [Required(ErrorMessage = "Date of birth is required.")]
        [DataType(DataType.Date)]
        [MinimumAge(18, ErrorMessage = "Player must be at least 18 years old.")]
        public DateTime DateOfBirth { get; set; }

        [Range(0, double.MaxValue, ErrorMessage = "Salary must be non-negative.")]
        public decimal Salary { get; set; }
    }

    // Custom validation attribute (server-side)
    public class MinimumAgeAttribute : ValidationAttribute
    {
        private readonly int _minAge;

        public MinimumAgeAttribute(int minAge)
        {
            _minAge = minAge;
        }
        protected override ValidationResult IsValid(object value, ValidationContext validationContext)
        {
            if (value == null)
                return new ValidationResult(ErrorMessage ?? $"Minimum age is {_minAge}.");

            if (value is DateTime dob)
            {
                var today = DateTime.Today;
                int age = today.Year - dob.Year;
                if (dob > today.AddYears(-age)) age--;

                return (age >= _minAge)
                    ? ValidationResult.Success
                    : new ValidationResult(ErrorMessage ?? $"Minimum age is {_minAge}.");
            }

            return new ValidationResult("Invalid date");
        }
    }
}
```

Controllers/PlayersController.cs

```
using Microsoft.AspNetCore.Mvc;
using MyApp.Models;
using System.Collections.Generic;

namespace MyApp.Controllers
{
    public class PlayersController : Controller
    {
        // Temp in-memory store for example; in production use DB
        private static readonly List<Player> _players = new List<Player>();

        // GET: /Players/Create
        public IActionResult Create()
        {
            return View();
        }

        // POST: /Players/Create
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Create(Player player)
        {
            // Model binding happens before this and DataAnnotations are evaluated.
            if (!ModelState.IsValid)
            {
                // If validation failed, return the same view with the model so validation messages are displayed.
                return View(player);
            }

            // Simulate saving to DB
            player.Id = _players.Count + 1;
            _players.Add(player);

            // Redirect after POST to avoid resubmission; could go to Details/Index.
            return RedirectToAction(nameof(Index));
        }

        // GET: /Players
        public IActionResult Index()
        {
            return View(_players);
        }
    }
}
```

Views/Player/Create.cshtml

```
@model MyApp.Models.Player
@{
    ViewData["Title"] = "Create Player";
}

<h2>Add Player</h2>

<form asp-action="Create" method="post" class="needs-validation" novalidate>
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>

    <div class="form-group">
        <label asp-for="Name"></label>
        <input asp-for="Name" class="form-control" />
        <span asp-validation-for="Name" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label asp-for="Email"></label>
        <input asp-for="Email" class="form-control" />
        <span asp-validation-for="Email" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label asp-for="JerseyNumber"></label>
        <input asp-for="JerseyNumber" class="form-control" type="number" />
        <span asp-validation-for="JerseyNumber" class="text-danger"></span>
    </div>

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

```
        <label asp-for="Position"></label>
        <input asp-for="Position" class="form-control" />
        <span asp-validation-for="Position" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label asp-for="DateOfBirth"></label>
        <input asp-for="DateOfBirth" class="form-control" type="date" />
        <span asp-validation-for="DateOfBirth" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label asp-for="Salary"></label>
        <input asp-for="Salary" class="form-control" type="number" step="0.01" />
        <span asp-validation-for="Salary" class="text-danger"></span>
    </div>

    <button type="submit" class="btn btn-primary">Save</button>
</form>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

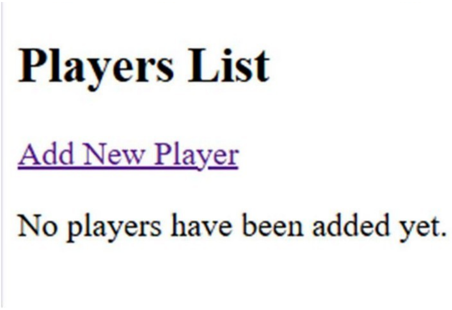
Views/Players/Index.cshtml

```
@model IEnumerable<MyApp.Models.Player>
@{
    ViewData["Title"] = "Players List";
}

<h2>Players List</h2>

<p>
    <a asp-action="Create" class="btn btn-primary">Add New Player</a>
</p>

@if (!Model.Any())
{
    <p>No players have been added yet.</p>
}
else
{
    <table class="table table-striped">
        <thead>
            <tr>
                <th>Name</th>
                <th>Email</th>
                <th>Jersey Number</th>
                <th>Position</th>
                <th>Date of Birth</th>
                <th>Salary</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var player in Model)
            {
                <tr>
                    <td>@player.Name</td>
                    <td>@player.Email</td>
                    <td>@player.JerseyNumber</td>
                    <td>@player.Position</td>
                    <td>@player.DateOfBirth.ToString("yyyy-MM-dd")</td>
                    <td>@player.Salary.ToString("C")</td>
                </tr>
            }
        </tbody>
    </table>
}
```

Source Code.cs

```
namespace Data_Annotation
{
    public class Source Code
    {
        public static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);

            // Add services to the container.
            builder.Services.AddControllersWithViews();

            var app = builder.Build();

            // Configure the HTTP request pipeline.
            if (!app.Environment.IsDevelopment())
            {
                app.UseExceptionHandler("/Home/Error");
                // The default HSTS value is 30 days. You may want to change this for production scenarios, see
https://aka.ms/aspnetcore-hsts.
                app.UseHsts();
            }

            app.UseHttpsRedirection();
            app.UseStaticFiles();

            app.UseRouting();

            app.UseAuthorization();
            app.MapControllerRoute(
                name: "default",
                pattern: "{controller=Players}/{action=Index}/{id?}");

            app.Run();
        }
    }
}
```

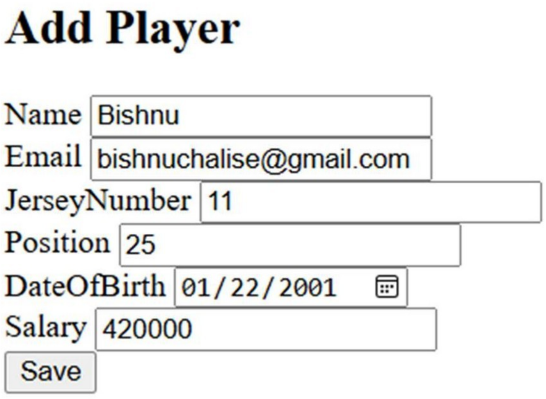
Output



Players List

Add New Player

No players have been added yet.



Add Player

Name: Bishnu

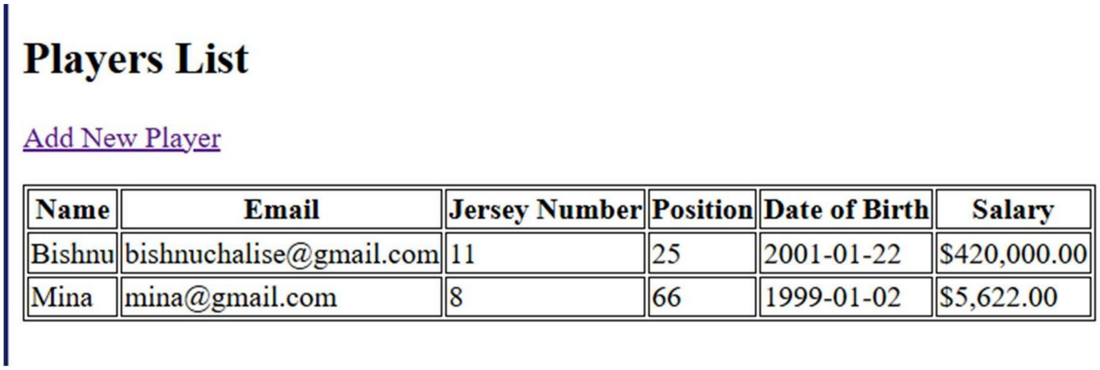
Email: bishnuchalise@gmail.com

JerseyNumber: 11

Position: 25

DateOfBirth: 01/22/2001

Salary: 420000



Players List

Add New Player

Name	Email	Jersey Number	Position	Date of Birth	Salary
Bishnu	bishnuchalise@gmail.com	11	25	2001-01-22	$420,000.00
Mina	mina@gmail.com	8	66	1999-01-02	$5,622.00

Lab 8: Define Authentication. Write a program to implement Authentication and Authorization using User Roles.

Authentication is the process of verifying the identity of a user or system before granting access to resources. It ensures that the person or entity is who they claim to be. It helps to confirm identity which differ from authorization which deals with what user can access.

Source Code:

Controller/AccountController.cs

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Authentication_Authorization.Models;
using Microsoft.EntityFrameworkCore.Metadata.Internal;
using Authentication_Authorization.ViewModels;
namespace Authentication_Authorization.Controllers
{
    public class AccountController : Controller
    {
        private readonly UserManager<ApplicationUser> _userManager;
        private readonly SignInManager<ApplicationUser> _signInManager;
        public AccountController(UserManager<ApplicationUser> userManager,
SignInManager<ApplicationUser> signInManager)
        {
            _userManager = userManager;
            _signInManager = signInManager;
        }
        [HttpGet]
        public IActionResult Register() => View();
        [HttpPost]
        public async Task<IActionResult> 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 _userManager.AddToRoleAsync(user, "User");
                    await _signInManager.SignInAsync(user, false);
                    return RedirectToAction("Index", "Home");
                }
                foreach (var error in result.Errors)
                    ModelState.AddModelError("", error.Description);
            }
            return View(model);
        }
        [HttpGet]
        public IActionResult Login() => View();
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Login(LoginViewModel model)
        {
            if (ModelState.IsValid)
            {
                var user = await _userManager.FindByEmailAsync(model.Email);
                if(user !=null & !await _userManager.IsEmailConfirmedAsync(user))
                {
                    ModelState.AddModelError("", "You need to confirm your email before logging in.");
                    return View(model);
                }

                var result = await _signInManager.PasswordSignInAsync(model.Email, model.Password, false,
false);
                if (result.Succeeded)
                {
                    if (await _userManager.IsInRoleAsync(user, "Admin"))
                    {
                        return RedirectToAction("Admin", "Dashboard");
                    }
                    else if (await _userManager.IsInRoleAsync(user, "User"))
                    {
                        return RedirectToAction("User", "Dashboard");
                    }
                    else
                    {
                        return RedirectToAction("Index", "Home");
                    }
                }
                ModelState.AddModelError("", "Invalid Login Attempt");
            }
            return View(model);
        }
        public async Task<IActionResult> Logout()
        {
            await _signInManager.SignOutAsync();
            return RedirectToAction("Index", "Home");
        }
    }
}
```

Controller/DashboardController.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace Authentication_Authorization.Controllers
{
    public class DashboardController : Controller
    {
        // Accessible only by Admin role
        [Authorize(Roles = "Admin")]
        public IActionResult Admin()
        {
            return View();
        }
        // Accessible only by Users role
        [Authorize(Roles = "User")]
        public IActionResult User()
        {
            return View();
        }
        // Accessible by Admin and User role
        [Authorize(Roles = "Admin, User")]
        public IActionResult Common()
        {
            return View();
        }
    }
}
```

Data/ApplicationDbContext.cs

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using Authentication_Authorization.Models;

namespace Authentication_Authorization.Data
{
    public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
    {
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
            : base(options)
        {
        }
    }
}
```

Models/ApplicationUser.cs

```
using Microsoft.AspNetCore.Identity;
namespace Authentication_Authorization.Models
{
    public class ApplicationUser : IdentityUser
    {
    }
}
```

ViewModels/LoginViewModel.cs

```
using System.ComponentModel.DataAnnotations;
namespace Authentication_Authorization.ViewModels
{
    public class LoginViewModel
    {
        [Required]
        [EmailAddress]
        public required string Email { get; set; }
        [Required]
        [DataType(DataType.Password)]
        public required string Password { get; set; }
    }
}
```

ViewModels/RegisterViewModel.cs

```
using System.ComponentModel.DataAnnotations;
namespace Authentication_Authorization.ViewModels
{
    public class RegisterViewModel
    {
        [Required]
        [EmailAddress]
        public required string Email { get; set; }
        [Required]
        [DataType(DataType.Password)]
        public required string Password { get; set; }
        [DataType(DataType.Password)]
        [Compare("Password", ErrorMessage = "Passwords do not match")]
        public required string ConfirmPassword { get; set; }
    }
}
```

Views/Account/Login.cshtml

```
@model Authentication_Authorization.ViewModels.LoginViewModel
@{
    ViewData["Title"] = "Login";
}
<h2>Login</h2>
<form asp-action="Login" method="post">
<div class="form-group">
<label asp-for="Email"></label>
<input asp-for="Email" class="form-control" />
<span asp-validation-for="Email" class="text-danger"></span>
</div>
<div class="form-group">
<label asp-for="Password"></label>
<input asp-for="Password" type="password" class="form-control" />
<span asp-validation-for="Password" class="text-danger"></span>
</div>
<button type="submit" class="btn btn-primary">Login</button>
</form>
@section Scripts {
    <partial name="_ValidationScriptsPartial"></partial>
}
```

Views/Account/Register.cshtml

```
@model Authentication_Authorization.ViewModels.RegisterViewModel
@{
    ViewData["Title"] = "Register";
}

<h2 class="text-center mb-4">Register</h2>

<div class="row justify-content-center">
    <div class="col-md-6">
        <div class="card shadow-lg p-4 rounded-3">
            <form asp-action="Register" method="post">
                <div asp-validation-summary="ModelOnly" class="text-danger"></div>

                <!-- Email -->
                <div class="form-group mb-3">
                    <label asp-for="Email" class="form-label"></label>
                    <input asp-for="Email" class="form-control" placeholder="Enter your email" />
                    <span asp-validation-for="Email" class="text-danger"></span>
                </div>

                <!-- Password -->
                <div class="form-group mb-3">
                    <label asp-for="Password" class="form-label"></label>
                    <input asp-for="Password" type="password" class="form-control" placeholder="Enter
password" />
                    <span asp-validation-for="Password" class="text-danger"></span>
                </div>

                <!-- Confirm Password -->
                <div class="form-group mb-4">
                    <label asp-for="ConfirmPassword" class="form-label"></label>
                    <input asp-for="ConfirmPassword" type="password" class="form-control"
placeholder="Confirm password" />
                    <span asp-validation-for="ConfirmPassword" class="text-danger"></span>
                </div>

                <div class="text-center">
                    <button type="submit" class="btn btn-primary px-4">Register</button>
                </div>
            </form>

            <div class="text-center mt-3">
                <p>Already have an account? <a asp-controller="Account" asp-action="Login">Login
here</a></p>
            </div>
        </div>
    </div>
</div>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}
```

Views/Dashboard/Admin.cshtml

```
@{
    ViewData["Title"] = "Admin Dashboard";
}
<h2>Admin Dashboard</h2>
<p>Welcome, Admin!, You can manage the system from here.</p>
```

Views/Dashboard/User.cshtml

```
@{
    ViewData["Title"] = "User Dashboard";
}
<h2>User Dashboard</h2>
<p>Welcome, User!, You can view your personal data here.</p>
```

Views/Dashboard/Common.cshtml

```
@{
    ViewData["Title"] = "Common Dashboard";
}
<h2>Common Dashboard</h2>
<p>Both Admin and User roles can see this page.</p>
```

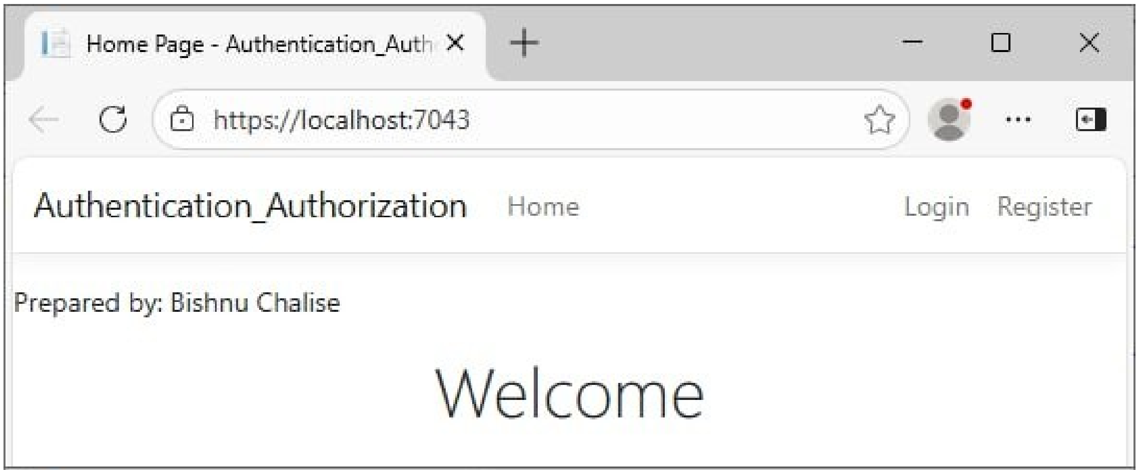
Views/Shared/_Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Authentication_Authorization</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="~/Authentication_Authorization.styles.css" asp-append-version="true" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">
            <div class="container-fluid">
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">Authentication_Authorization</a>
                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent"
                        aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav me-auto">
                    <li class="nav-item">
                        <a class="nav-link" asp-controller="Home" asp-action="Index">Home</a>
                    </li>

                    @* Role-based links *@
                    @if (User.IsInRole("Admin"))
                    {
                        <li class="nav-item">
                            <a class="nav-link" asp-controller="Dashboard" asp-action="Admin">Admin Dashboard</a>
                        </li>
                    }

                    @if (User.IsInRole("User"))
                    {
                        <li class="nav-item">
                            <a class="nav-link" asp-controller="Dashboard" asp-action="User">User Dashboard</a>
                        </li>
                    }

                    @if (User.IsInRole("Admin") || User.IsInRole("User"))
                    {
                        <li class="nav-item">
                            <a class="nav-link" asp-controller="Dashboard" asp-action="Common">Common Dashboard</a>
                        </li>
                    }
                </ul>

                <ul class="navbar-nav ms-auto">
                    @if (User.Identity != null && User.Identity.IsAuthenticated)
                    {
                        <li class="nav-item">
                            <span class="nav-link">Hello, @User.Identity.Name!</span>
                        </li>
                        <li class="nav-item">
                            <form class="d-inline" asp-controller="Account" asp-action="Logout" method="post">
                                <button type="submit" class="btn btn-link nav-link" style="display:inline; padding:0;">Logout</button>
                            </form>
                        </li>
                    }
                    else
                    {
                        <li class="nav-item">
                            <a class="nav-link" asp-controller="Account" asp-action="Login">Login</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" asp-controller="Account" asp-action="Register">Register</a>
                        </li>
                    }
                </ul>
            </div>
        </div>
    </nav>
</header>
```

```
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>
    <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>
```

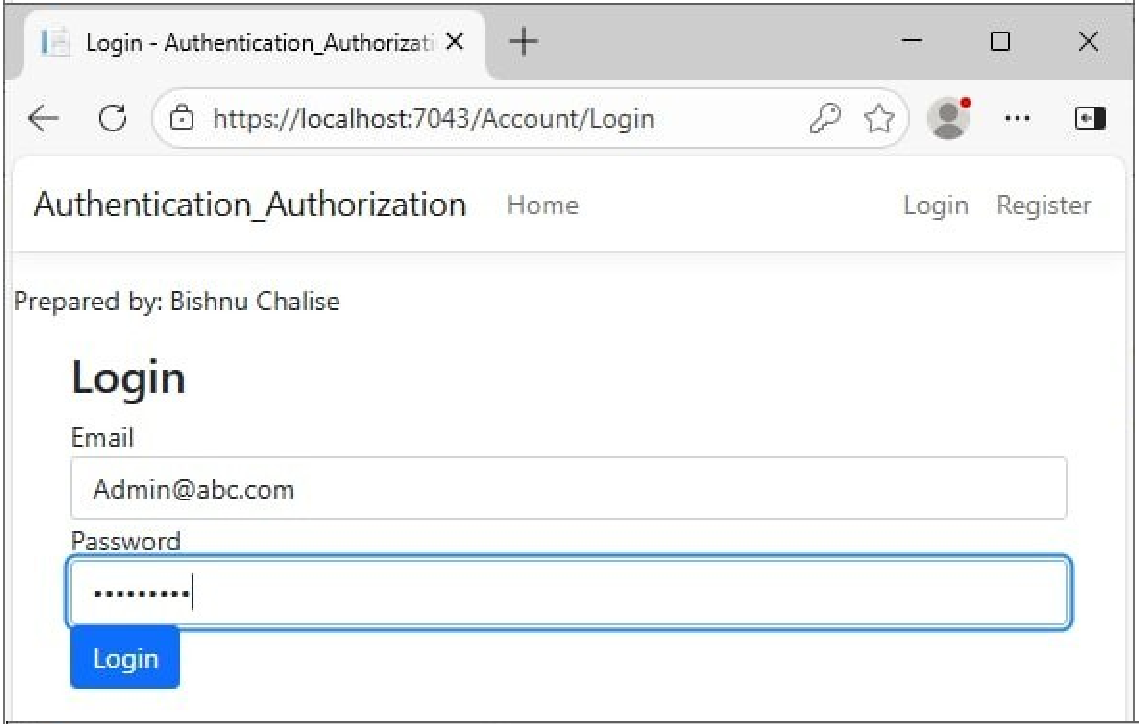
Views/Home/Index.cshtml

```
@{
    ViewData["Title"] = "Home Page";
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Prepared by:Bishnu Chalise</p>
</div>
```



Prepared by: Bishnu Chalise

Welcome


Login - Authentication_Authorizati
https://localhost:7043/Account/Login
Authentication_Authorization
Home
Login
Register

Prepared by: Bishnu Chalise

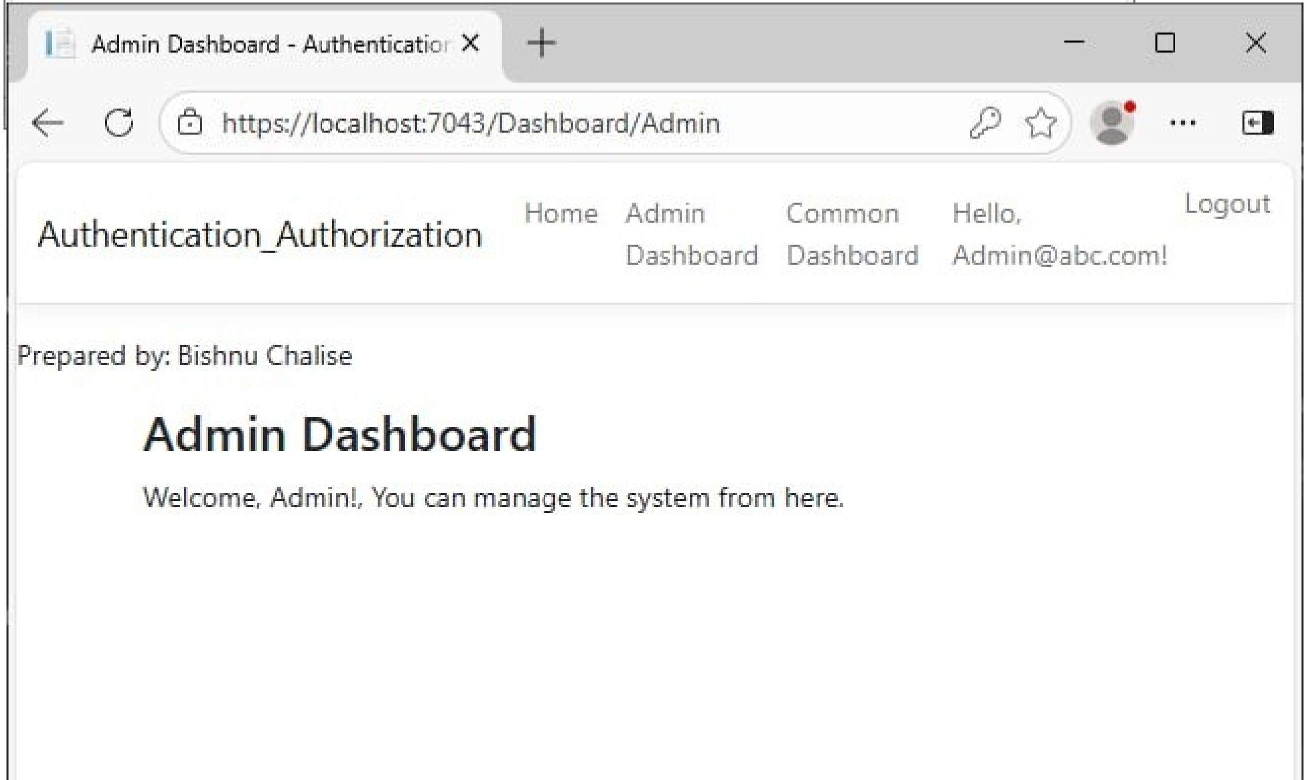
Login

Email

Admin@abc.com

Password

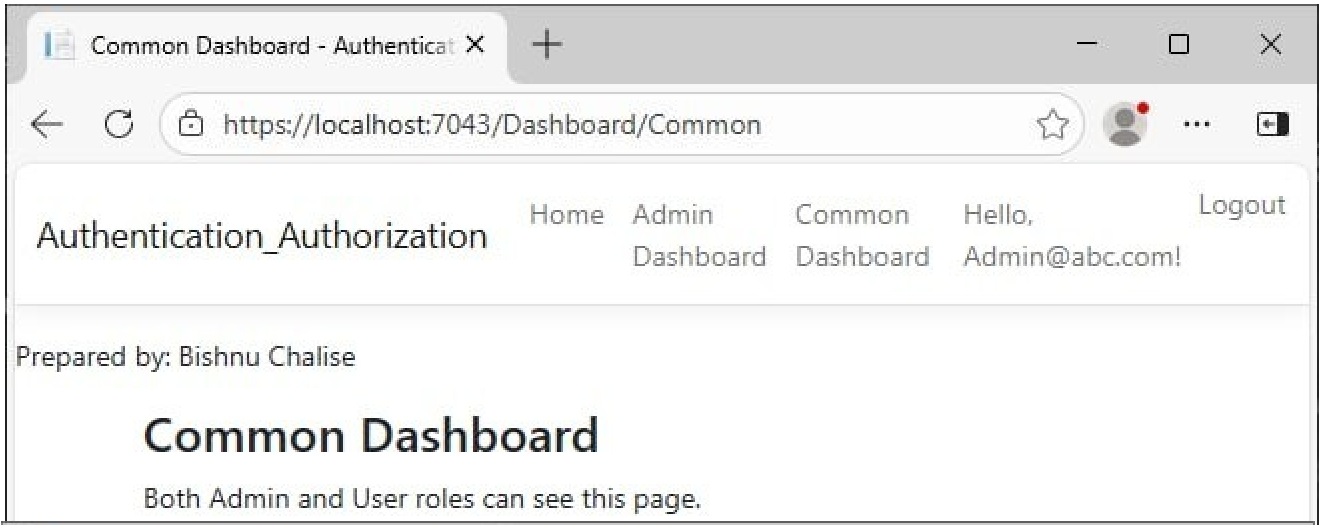
•••••••••


Admin Dashboard - Authenticatior
https://localhost:7043/Dashboard/Admin
Authentication_Authorization
Home
Admin Dashboard
Common Dashboard
Hello, Admin@abc.com!
Logout

Prepared by: Bishnu Chalise

Admin Dashboard

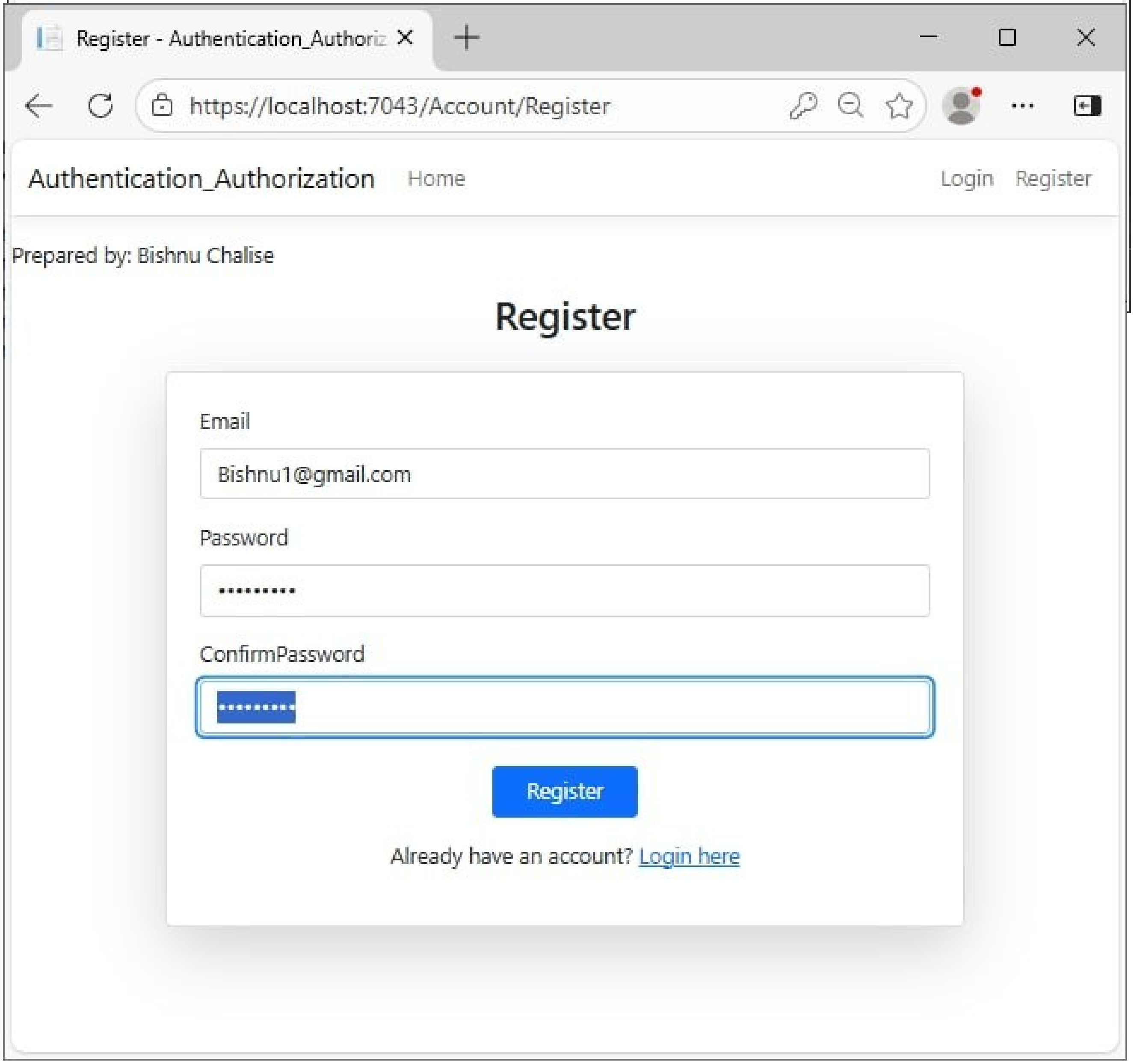
Welcome, Admin!, You can manage the system from here.


Common Dashboard - Authenticat
https://localhost:7043/Dashboard/Common
Authentication_Authorization
Home
Admin Dashboard
Common Dashboard
Hello, Admin@abc.com!
Logout

Prepared by: Bishnu Chalise

Common Dashboard

Both Admin and User roles can see this page.


Register - Authentication_Authoriz
https://localhost:7043/Account/Register
Authentication_Authorization
Home
Login
Register

Prepared by: Bishnu Chalise

Register

Email

Bishnu1@gmail.com

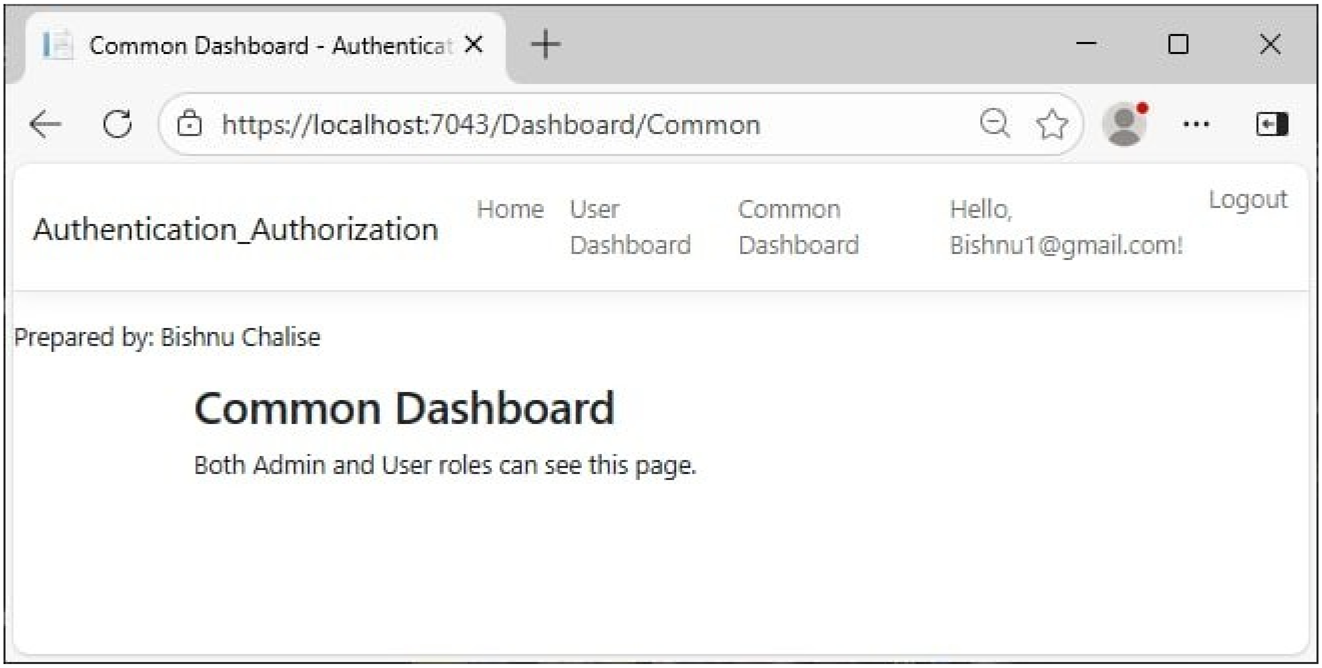
Password

•••••••••

ConfirmPassword

•••••••••

Already have an account? Login here

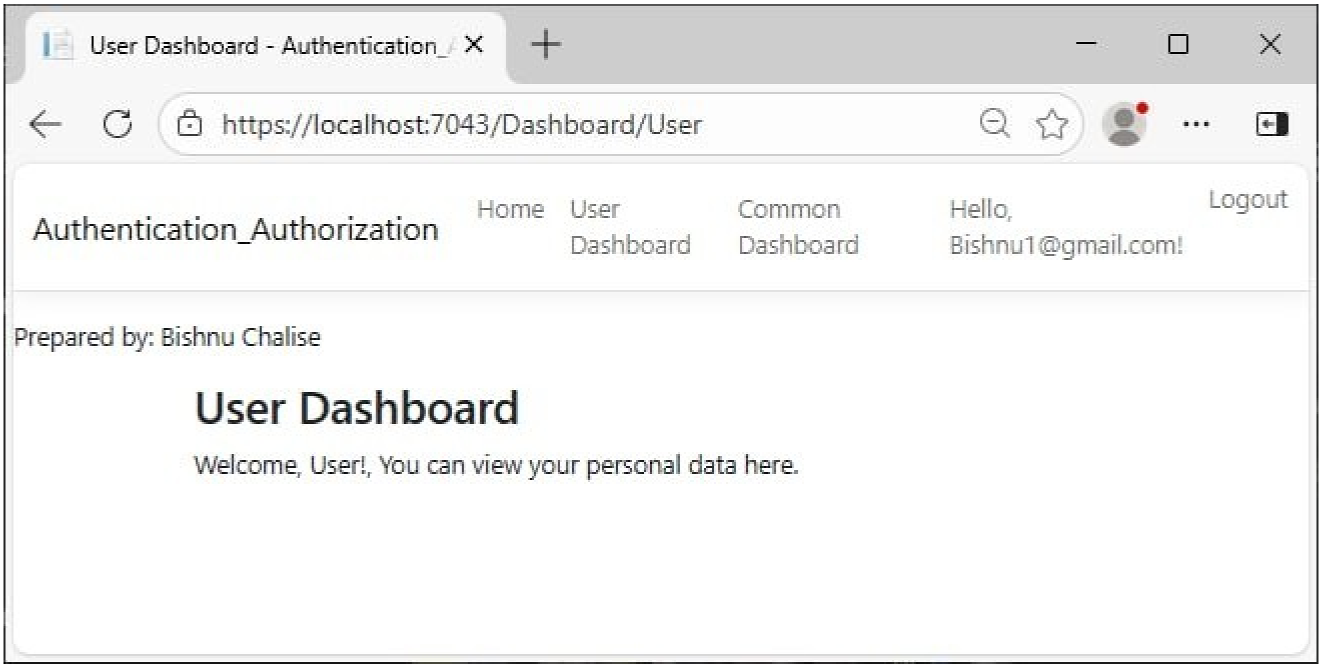

Common Dashboard - Authenticat
https://localhost:7043/Dashboard/Common

Authentication_Authorization Home User Dashboard Common Dashboard Hello, Bishnu1@gmail.com! Logout

Prepared by: Bishnu Chalise

Common Dashboard

Both Admin and User roles can see this page.


User Dashboard - Authentication_/
https://localhost:7043/Dashboard/User

Authentication_Authorization Home User Dashboard Common Dashboard Hello, Bishnu1@gmail.com! Logout

Prepared by: Bishnu Chalise

User Dashboard

Welcome, User!, You can view your personal data here.

Lab 9: Define cookie. Write a program to store User login information for 5 days using Cookie.

A cookie is a small piece of data stored on the client's browser by a web server.
It is used to remember information between HTTP requests – such as user login details, preferences, or session identifiers.

In ASP.NET Core, cookies are often used for:
→ Authentication and Authorization
→ User preferences or settings
→ Remembering logged-in users

Source:

Controllers/AccountController.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Http;
using System;

namespace MVC_Cookies.Controllers
{
   public class AccountController : Controller
   {
      // GET: Account/Login
      [HttpGet]
      public IActionResult Login()
      {
         return View();
      }

      // POST: Account/Login
      [HttpPost]
      public IActionResult Login(string username, string password)
      {
         // Dummy validation (for example only)
         if (username == "admin" && password == "1234")
         {
            // Create cookie options
            CookieOptions options = new CookieOptions
            {
               Expires = DateTime.Now.AddDays(5), // Cookie valid for 5 days
               HttpOnly = true, // Prevents client-side script access
               Secure = true    // Use HTTPS
            };

            // Store user login info in cookie
            Response.Cookies.Append("UserName", username, options);

            ViewBag.Message = "Login successful! Cookie stored for 5 days.";
            return View("Welcome");
         }
         else
         {
            ViewBag.Message = "Invalid username or password.";
            return View();
            }
        }

        // GET: Account/Welcome
        public IActionResult Welcome()
        {
            string username = Request.Cookies["UserName"];
            if (username != null)
            {
                ViewBag.User = username;
                return View();
            }

            return RedirectToAction("Login");
        }

        // Logout Action
        public IActionResult Logout()
        {
            Response.Cookies.Delete("UserName");
            return RedirectToAction("Login");
        }
    }
}
```

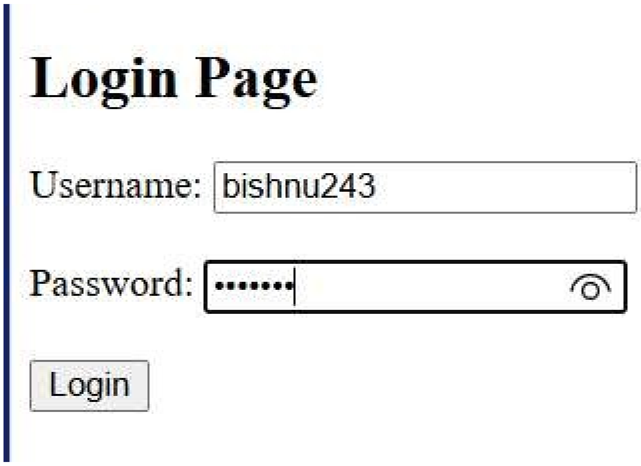
Views/Account/Login.cshtml

```
@{
    ViewData["Title"] = "Login";
}
<h2>Login Page</h2>

<form method="post" asp-controller="Account" asp-action="Login">
    <label>Username:</label>
    <input type="text" name="username" required /><br /><br />
    <label>Password:</label>
    <input type="password" name="password" required /><br /><br />
    <button type="submit">Login</button>
</form>

<p style="color:red">@ViewBag.Message</p>
```

Views/Account/Welcome.cshtml

```
@{
    ViewData["Title"] = "Welcome";
}
<h2>Welcome, @ViewBag.User!</h2>

<p>Your login information is stored in a cookie for 5 days.</p>

<a asp-controller="Account" asp-action="Logout">Logout</a>
```

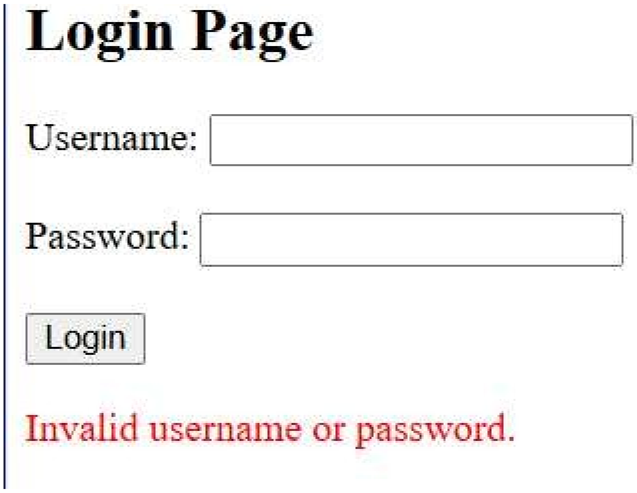
Output:



Login Page

Username: bishnu243

Password: •••••••

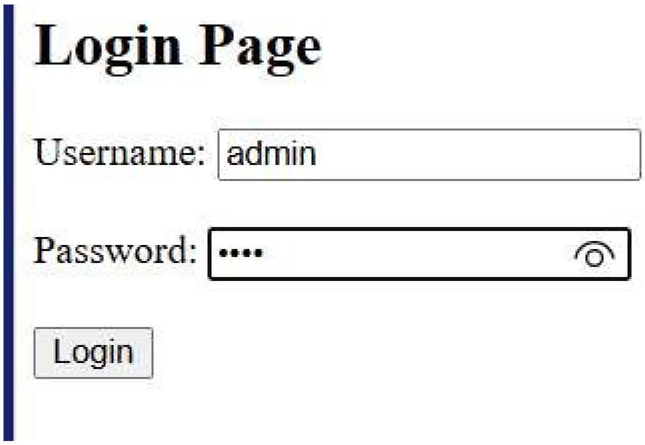


Login Page

Username:

Password:

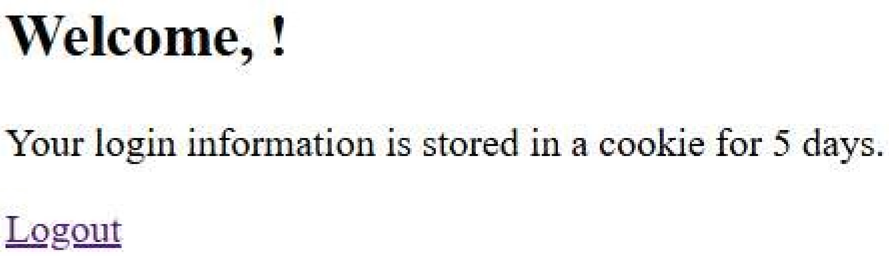
Invalid username or password.



Login Page

Username: admin

Password: ••••



Welcome, !

Your login information is stored in a cookie for 5 days.

Lab 10: Define single page application. Write a program to validation the login form when user submit empty value using jQuery.

A Single Page Application (SPA) is a type of web application that loads a single HTML page and dynamically updates content as the user interacts with the app – without reloading the entire page from the server.
It uses AJAX and JavaScript frameworks (like Angular, React, or Vue.js) to fetch and render data dynamically.
Only required data is exchanged with the server, improving speed and user experience.
The browser updates content via client-side routing.
Example: Gmail, Google Maps, Facebook.

Source Code:
Controller/AccountController.cs

```
using Microsoft.AspNetCore.Mvc;
using MVC_jQuery.Models;

namespace MVC_jQuery.Controllers
{
   public class AccountController : Controller
   {
      [HttpGet]
      public IActionResult Login()
      {
         return View();
      }

      [HttpPost]
      public IActionResult Login(LoginViewModel model)
      {
         if (!ModelState.IsValid)
         {
            return View(model);
         }

         // Add authentication logic here...
         return RedirectToAction("Index", "Home");
      }
   }
}
```

Models/LoginViewModel.cs

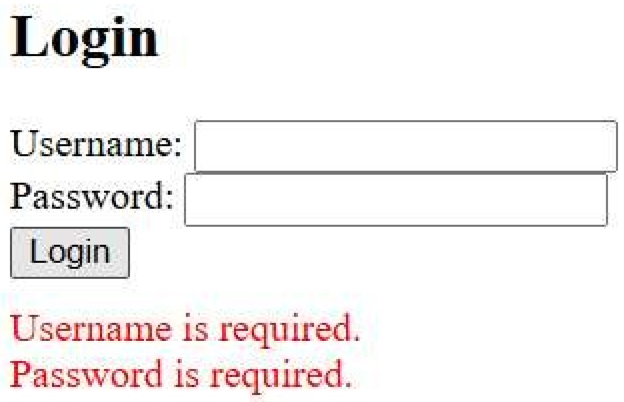
```
using System.ComponentModel.DataAnnotations;

namespace MVC_jQuery.Models
{
   public class LoginViewModel
   {
      [Required(ErrorMessage = "Username is required")]
      public string Username { get; set; }
```

```
        [Required(ErrorMessage = "Password is required")]
        public string Password { get; set; }
    }
}
```

Views/Account/Login.cshtml

```
@model MVC_jQuery.Models.LoginViewModel

@{
    ViewData["Title"] = "Login";
}

<h2>Login</h2>

<form id="loginForm" asp-action="Login" asp-controller="Account" method="post">
    <div>
        <label>Username:</label>
        <input type="text" id="Username" name="Username" />
    </div>
    <div>
        <label>Password:</label>
        <input type="password" id="Password" name="Password" />
    </div>
    <div>
        <input type="submit" value="Login" />
    </div>
</form>

<!-- Validation Message Display -->
<div id="errorMessages" style="color:red; margin-top:10px;"></div>

@section Scripts {
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

    <script>
        $(document).ready(function () {
            $("#loginForm").submit(function (e) {
                var username = $("#Username").val().trim();
                var password = $("#Password").val().trim();
                var errorMsg = "";

                if (username === "" || password === "") {
                    e.preventDefault(); // Stop form submission

                    if (username === "") {
                        errorMsg += "Username is required.<br/>";
                    }
                    if (password === "") {
                        errorMsg += "Password is required.<br/>";
                    }
                    $("#errorMessages").html(errorMsg);
                }
            });
        });
    </script>
}
```

Output:



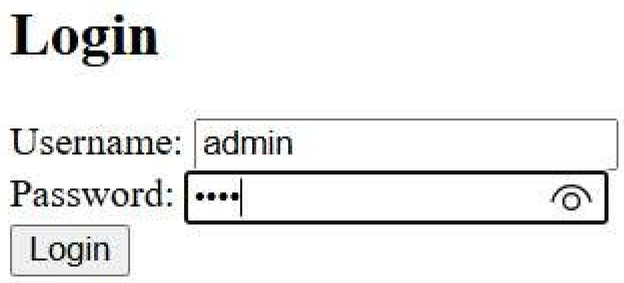
Login

Username:

Password:

Username is required.

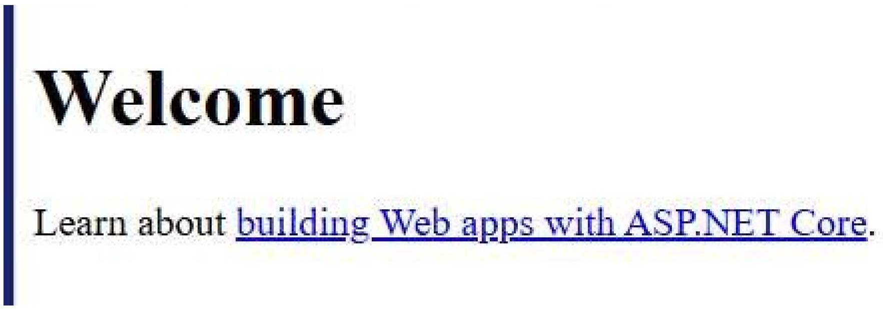
Password is required.



Login

Username: admin

Password: ••••



Welcome

Learn about building Web apps with ASP.NET Core.

Lab 11: Define Web API. Write a program to get the list of products in json format using ASP.NET Web API.

A Web API (Application Source Codeming Interface) is a framework that allows applications to communicate with each other over the web using HTTP.
In ASP.NET, a web API is used to build RESTful services that can be consumed by wide range of clients – such as browsers, mobile apps, or other web application.

Source Code:

Models/Product.cs

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

namespace WebAPI_Example.Models
{
    public class Product
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public double Price { get; set; }
    }
}
```

Controller/ProductsController.cs

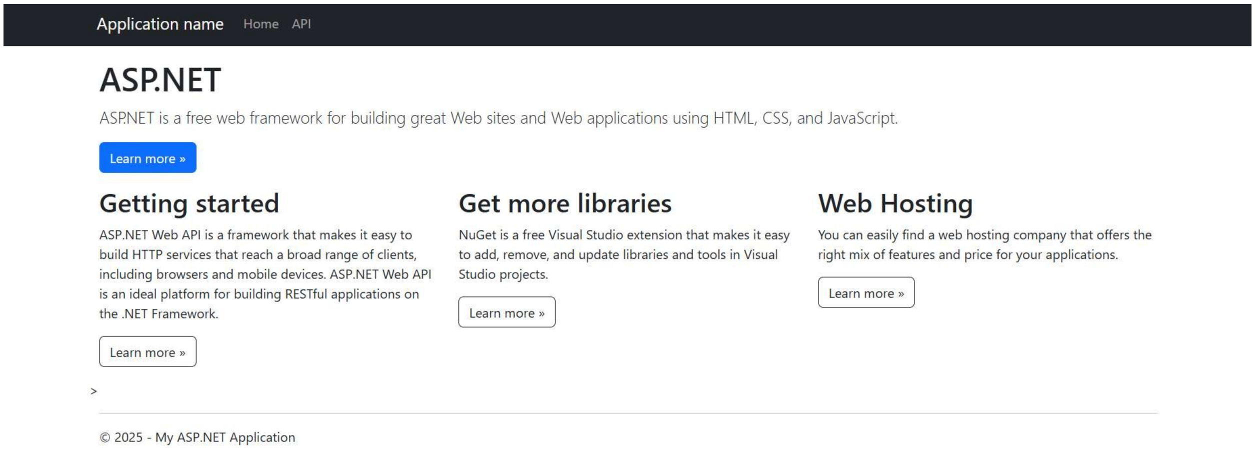
```
using System;
using System.Collections.Generic;
using System.Web.Http;
using WebAPI_Example.Models;

namespace WebAPI_Example.Controllers
{
    public class ProductsController : ApiController
    {
        public IEnumerable<Product> Get()
        {
            var products = new List<Product>
            {
                new Product { Id = 1, Name = "Laptop", Price = 55000 },
                new Product { Id = 2, Name = "Smartphone", Price = 25000 },
                new Product { Id = 3, Name = "Headphones", Price = 3000 }
            };
            return products;
        }
    }
}
```

App_Start/WebApiConfig.cs

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

namespace WebAPI_Example
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            // Web API configuration and services

            // Web API routes
            config.MapHttpAttributeRoutes();

            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
            // Removing XML Formatter to force JSON
            config.Formatters.Remove(config.Formatters.XmlFormatter);
            // Returning JSON Formatter
            config.Formatters.JsonFormatter.SerializerSettings.Formatting =
Newtonsoft.Json.Formatting.Indented;
        }
    }
}
```

Output:

Output:



ASP.NET

ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS, and JavaScript.

Getting started

ASP.NET Web API is a framework that makes it easy to build HTTP services that reach a broad range of clients, including browsers and mobile devices. ASP.NET Web API is an ideal platform for building RESTful applications on the .NET Framework.

Get more libraries

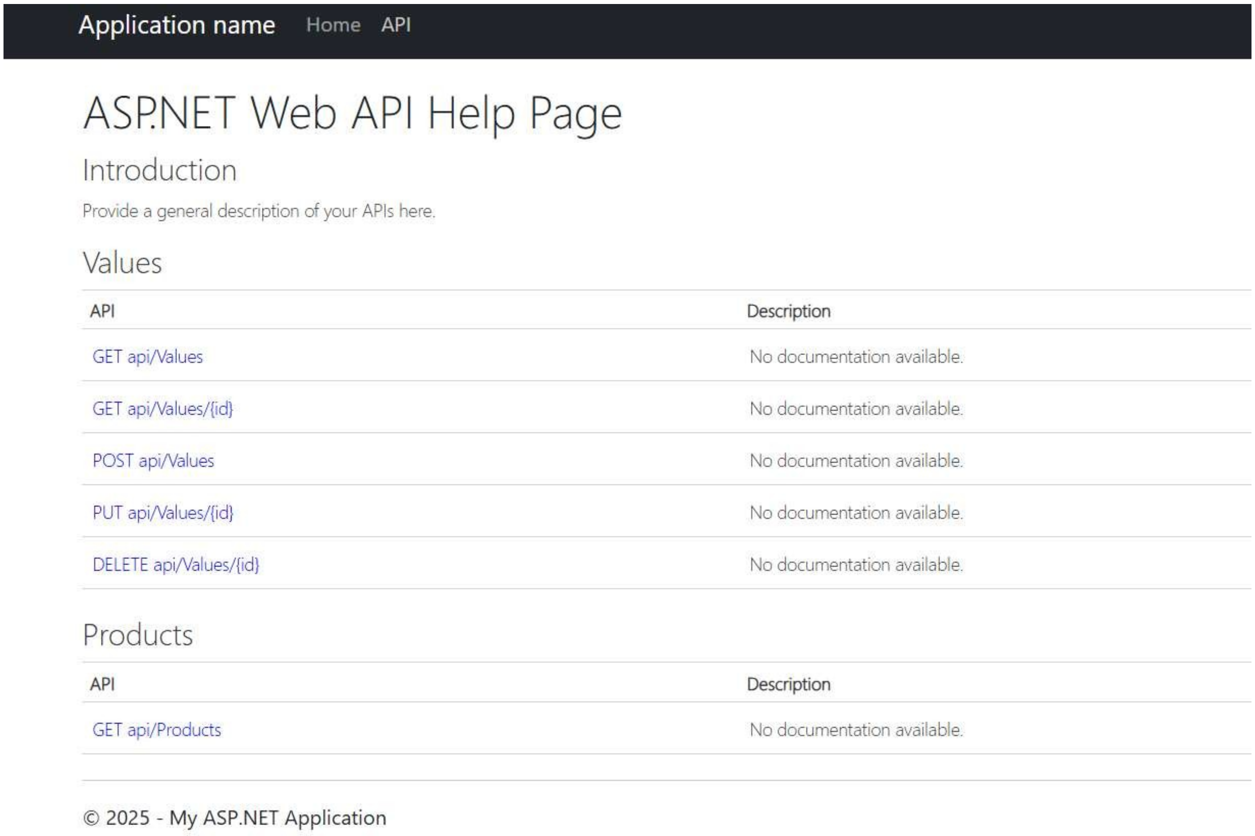
NuGet is a free Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects.

Web Hosting

You can easily find a web hosting company that offers the right mix of features and price for your applications.

>

© 2025 - My ASP.NET Application



ASP.NET Web API Help Page

Introduction

Provide a general description of your APIs here.

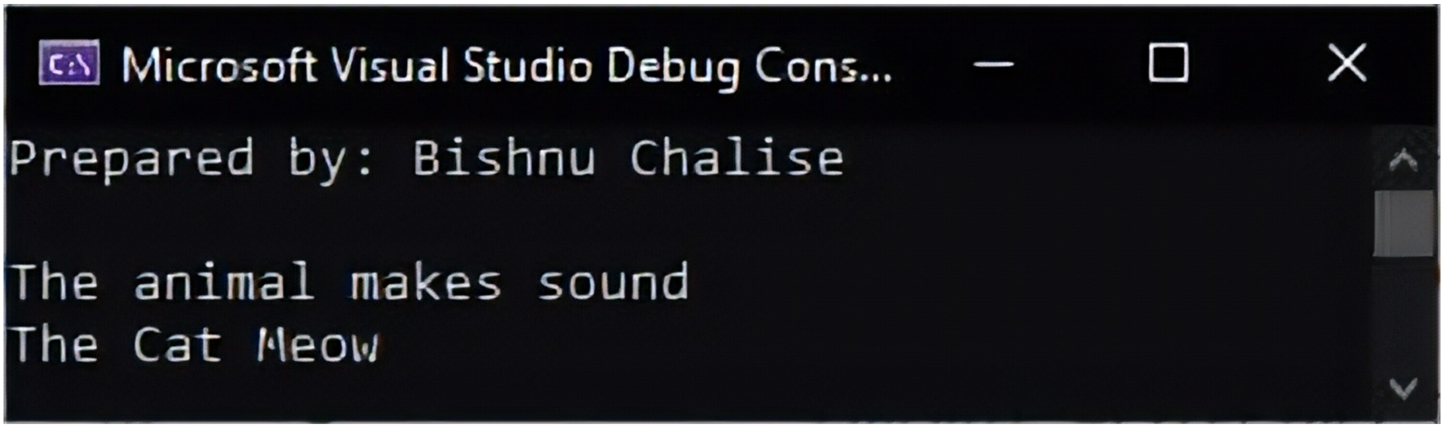
Values

API	Description
GET api/Values	No documentation available.
GET api/Values/{id}	No documentation available.
POST api/Values	No documentation available.
PUT api/Values/{id}	No documentation available.
DELETE api/Values/{id}	No documentation available.

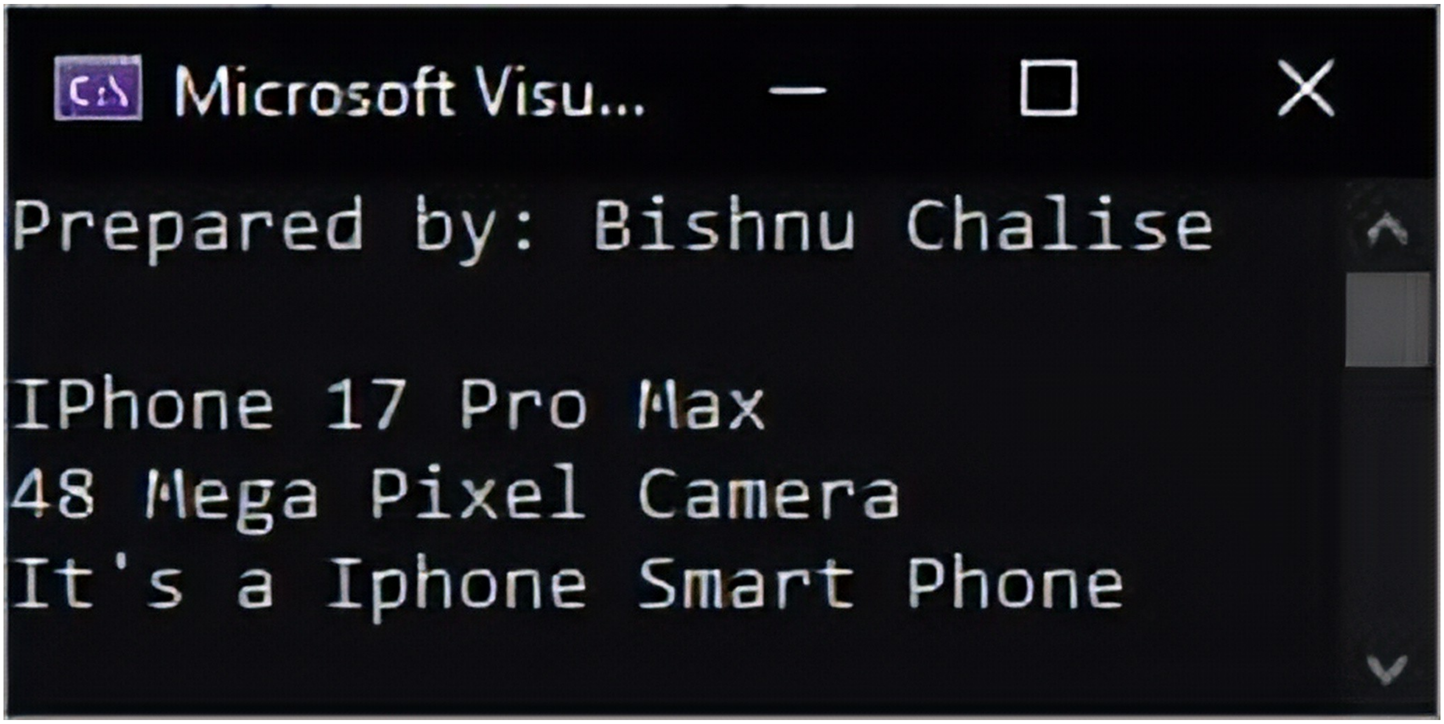
Products

API	Description
GET api/Products	No documentation available.

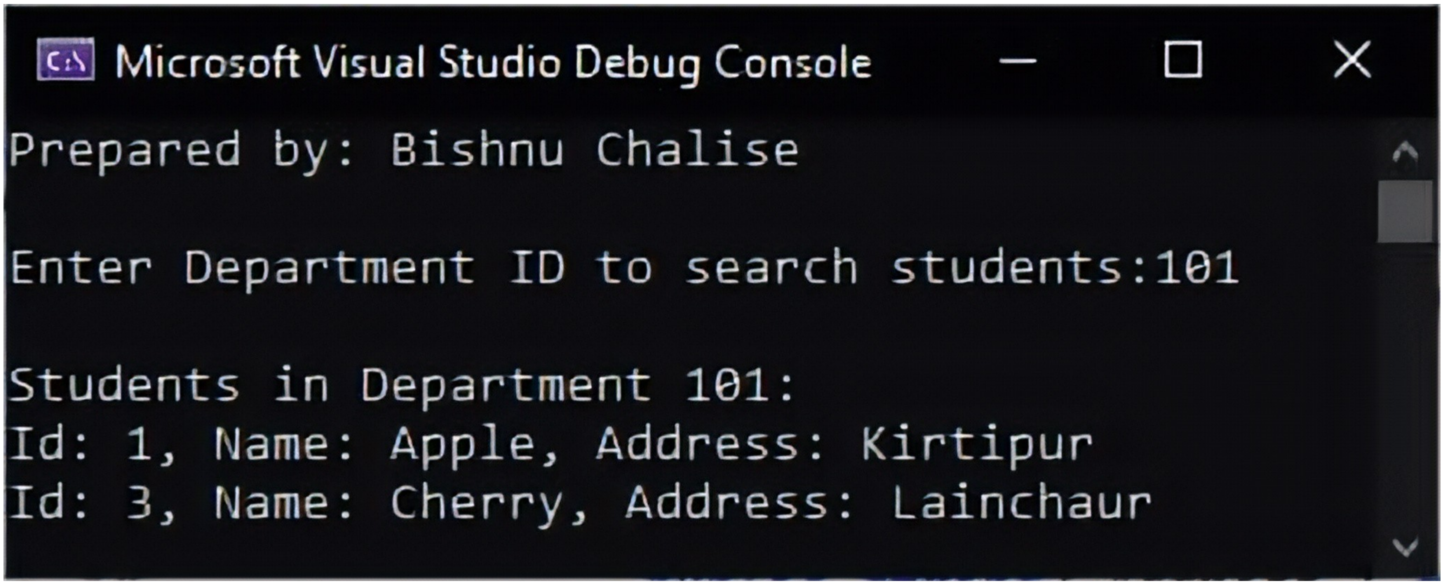
© 2025 - My ASP.NET Application



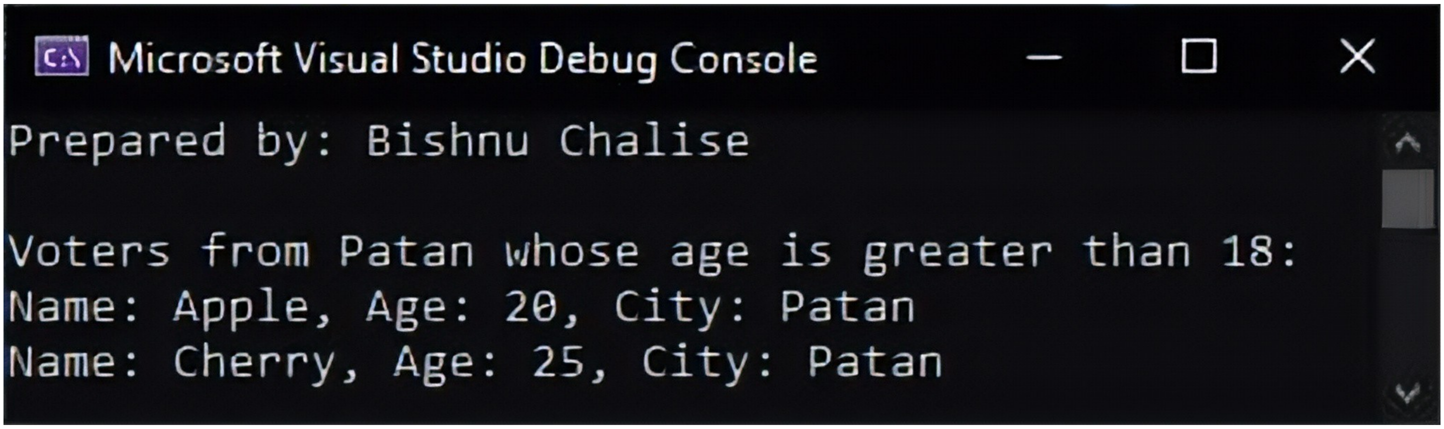
```
Prepared by: Bishnu Chalise

The animal makes sound
The Cat Meow
```



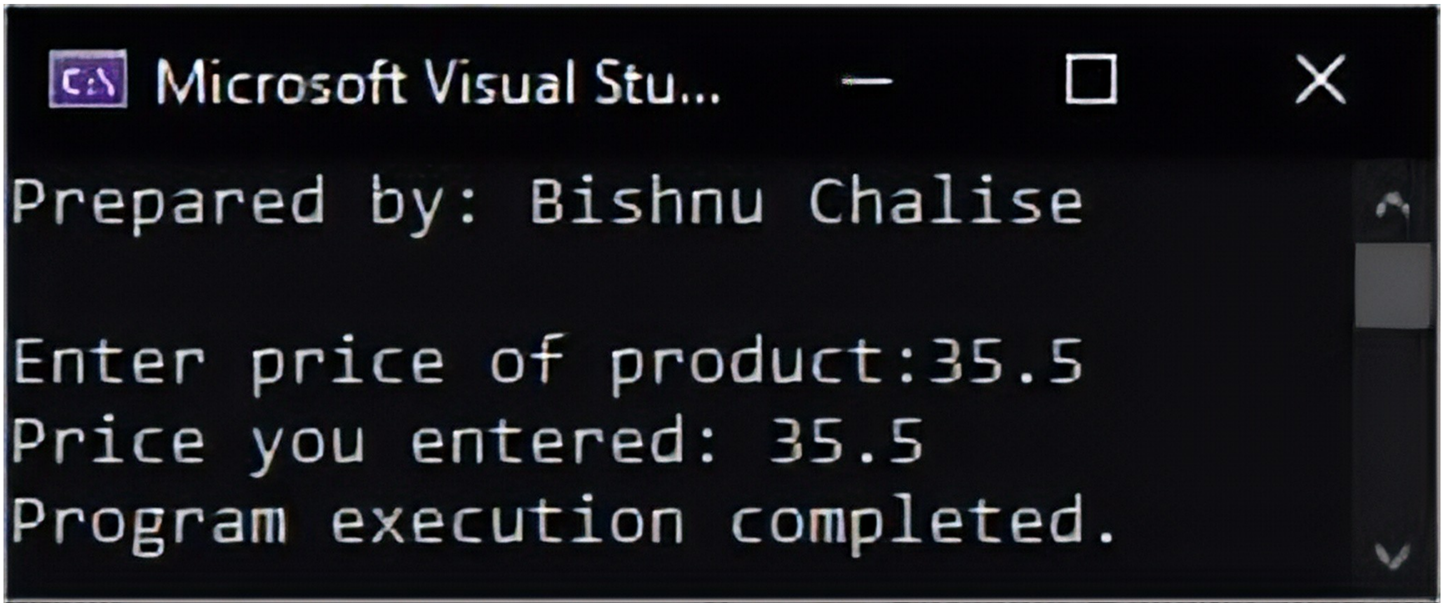
```
Prepared by: Bishnu Chalise

IPhone 17 Pro Max
48 Mega Pixel Camera
It's a Iphone Smart Phone
```



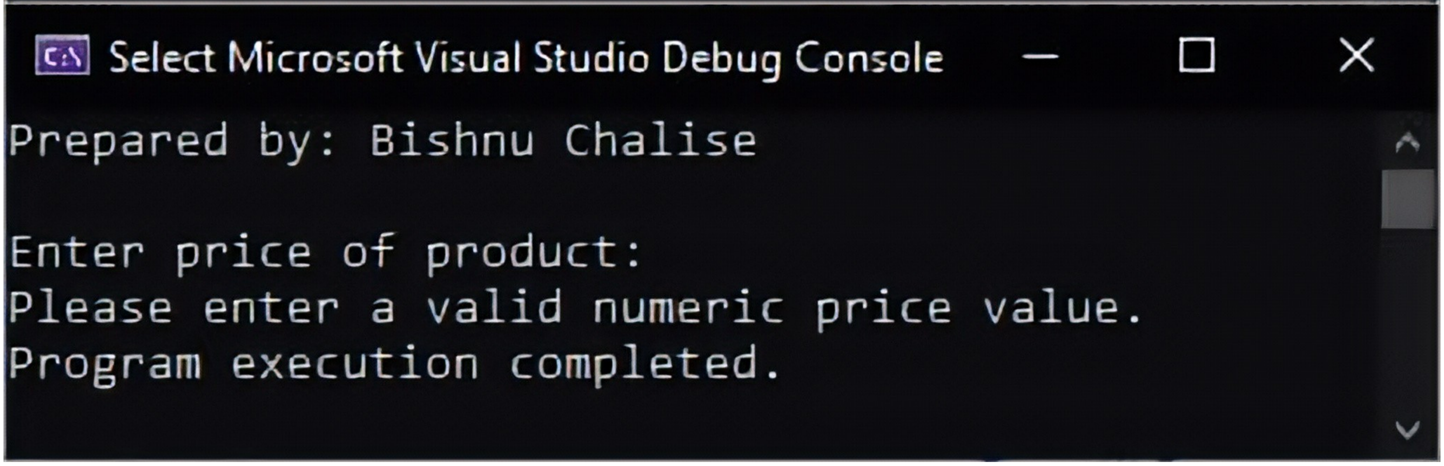
```
Prepared by: Bishnu Chalise

Enter Department ID to search students:101

Students in Department 101:
Id: 1, Name: Apple, Address: Kirtipur
Id: 3, Name: Cherry, Address: Lainchaur
```



```
Prepared by: Bishnu Chalise

Voters from Patan whose age is greater than 18:
Name: Apple, Age: 20, City: Patan
Name: Cherry, Age: 25, City: Patan
```



```
Prepared by: Bishnu Chalise

Enter price of product:35.5
Price you entered: 35.5
Program execution completed.
```



```
Prepared by: Bishnu Chalise

Enter price of product:
Please enter a valid numeric price value.
Program execution completed.
```

NET CENTRIC COMPUTING

TRIBHUVAN UNIVERSITY

AMRIT SCIENCE CAMPUS

Thamel, Kathmandu

Submitted By:
Bishnu Chalise

Faculty: CSIT
Section: A
Combination: CSIT 6th Sem

Submitted To: Binod Thapa

Internal Examiner
Signature:

External Examiner
Signature:

INDEX