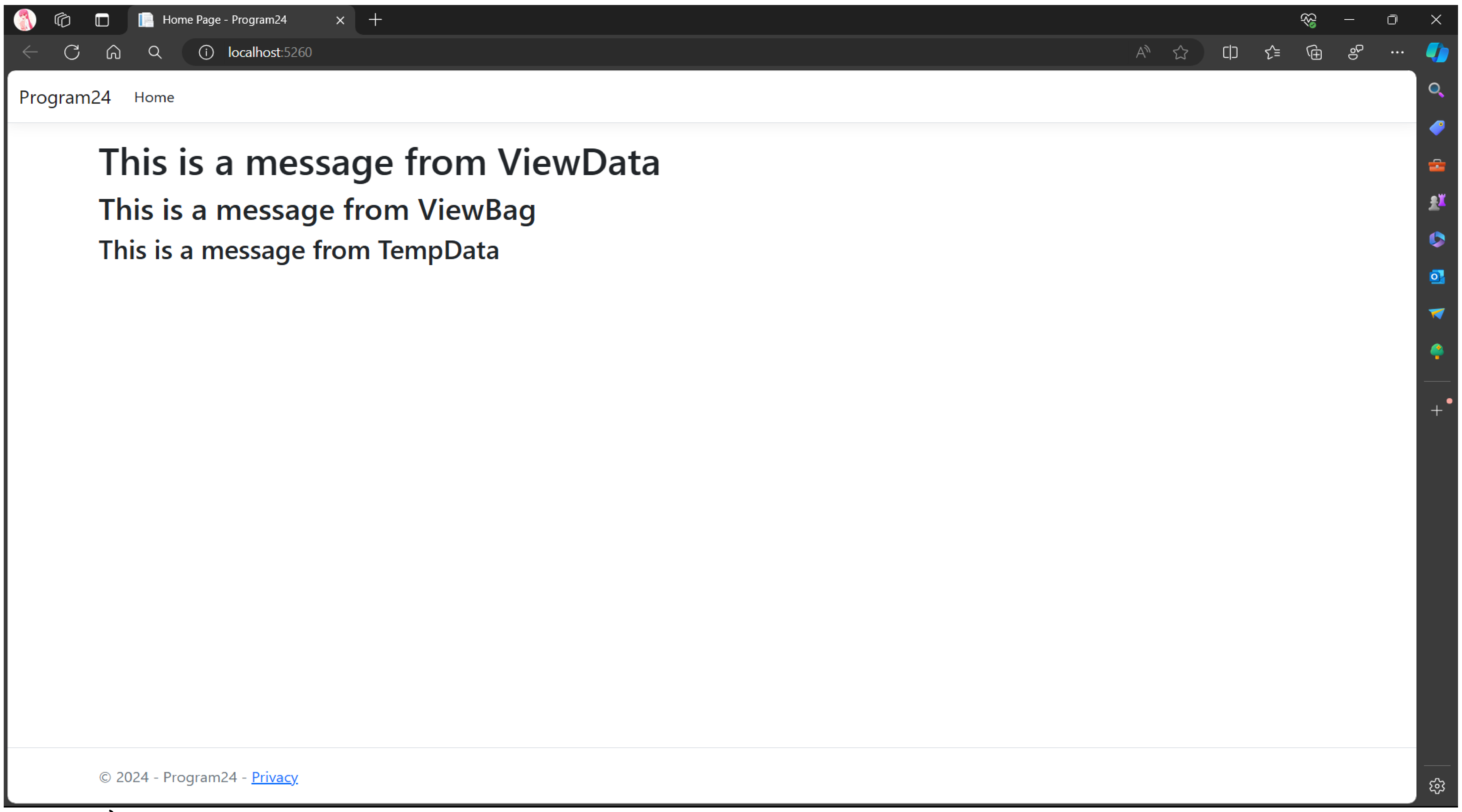
16. Develop ASP.NET Core MVC web application to demonstrate the use of ViewData, Viewbag and TempData.

Ans - index.cshtml

```
@{
        ViewData["Title"] = "Home Page";
}
        <h1>@ViewData["Sandesh"]</h1>
        <h2>@ViewBag.Message</h2>
        <h3>@TempData["Message"]</h3>
```

HomeController.cs

```
using Microsoft.AspNetCore.Mvc;
using Program24.Models;
using System.Diagnostics;

namespace Program24.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;

        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }

        public IActionResult Index()
        {
            ViewData["Sandesh"] = "This is a message from ViewData";
            ViewBag.Message = "This is a message from ViewBag";
            TempData["Message"] = "This is a message from TempData";

            return View();
        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
    HttpContext.TraceIdentifier });
        }
    }
```



This is a message from ViewData

This is a message from ViewBag

This is a message from TempData

© 2024 - Program24 - Privacy

```
}
```

17. Develop ASP.NET Core MVC web application to Create a Layout View in ASP.NET Core MVC.

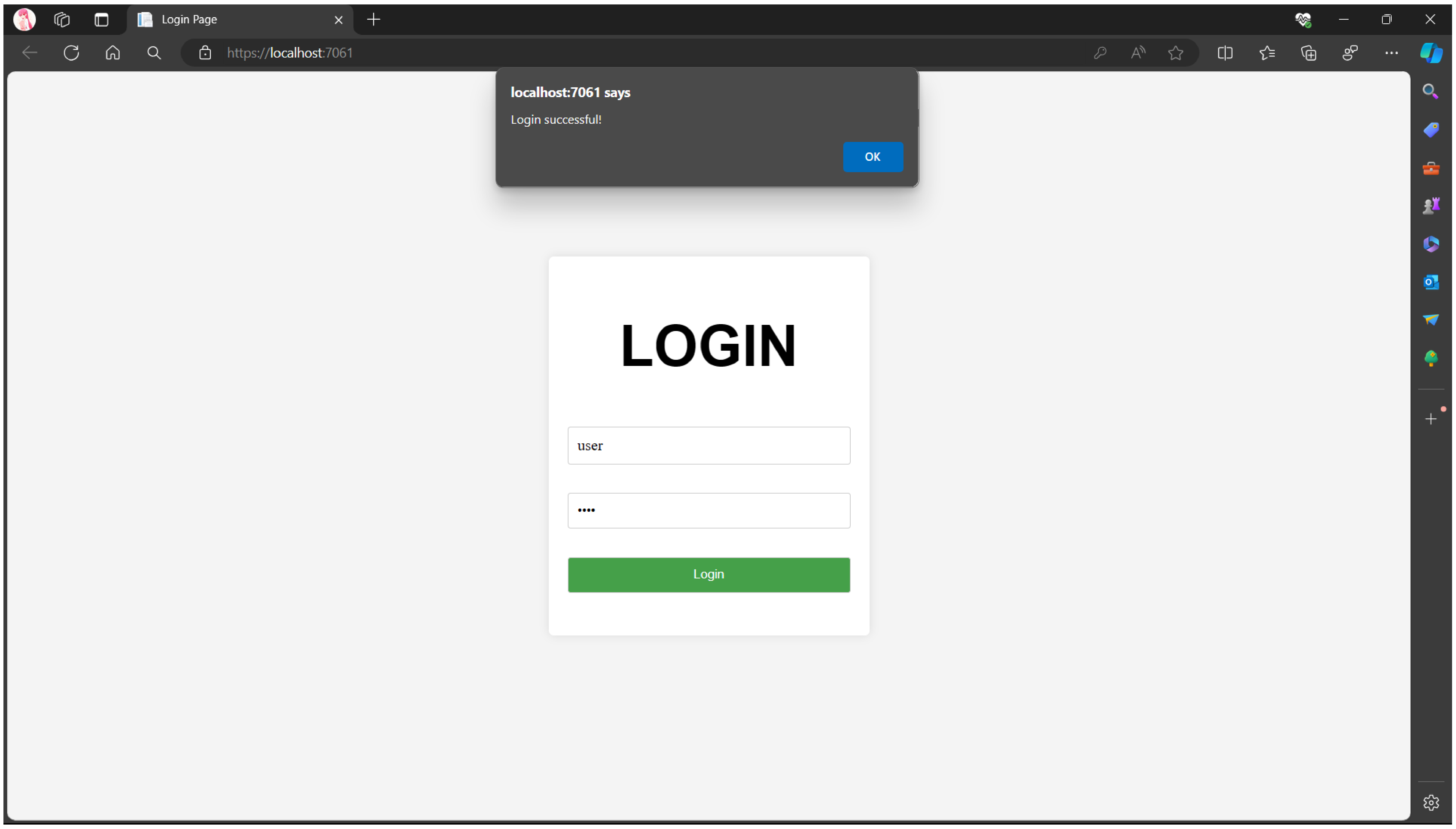
Ans – index.cshtml

```
    @{
    ViewBag.Title = "Home Page";
    Layout = "~/Views/Shared/_Layout1.cshtml";
}
<center><h1 style="font-size: 60px;">LOGIN</h1></center>
```

Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login Page</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }
        .container {
            width: 300px;
            background-color: #fff;
    border-radius: 5px;
    padding: 20px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
h2 {
    text-align: center;
}
input[type="text"],
input[type="password"],
input[type="submit"] {
    width: 100%;
    padding: 10px;
    margin-top: 10px;
    margin-bottom: 20px;
    border: 1px solid #ccc;
    border-radius: 3px;
    box-sizing: border-box;
}
input[type="submit"] {
    background-color: #4caf50;
    color: white;
    cursor: pointer;
}
    input[type="submit"]:hover {
        background-color: #45a049;
    }
.error-message {
    color: red;
    font-size: 14px;
```

```
            margin-top: 5px;
        }
    </style>
</head>
<body>
    <div class="container">
        @RenderBody()
        <form id="loginForm">
            <input type="text" id="username" placeholder="Username">
            <input type="password" id="password" placeholder="Password">
            <input type="submit" value="Login">
            <div class="error-message" id="errorMessage"></div>
        </form>
    </div>
    <script>
        document.getElementById("loginForm").addEventListener("submit", function(event) {
            event.preventDefault();
            var username = document.getElementById("username").value;
            var password = document.getElementById("password").value;
            if (username === "" || password === "") {
                document.getElementById("errorMessage").innerText = "Please enter both username and
password.";
                return;
            }
            if (username === "user" && password === "password") {
                alert("Login successful!");
            } else {
                document.getElementById("errorMessage").innerText = "Invalid username or password.";
            }
        });
    </script>
</body>
</html>
```


localhost:7061 says

Login successful!

18. Develop ASP.NET Core MVC web application to demonstrate the use of Razor Syntax (If ...else, Switch, For Loop, For each Loop, While Loop)

Ans – index.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"]="Program 26"</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         margin: 0;
         padding: 0;
         color: #333;
         background-color: #f9f9f9;
      }
      section {
         padding: 25px 0;
         text-align: center;
         background-color: #fff;
         border-bottom: 1px solid #ddd;
      }
      h1 {
         font-size: 25px;
         margin: 0;
      }
      ul {
            list-style: none;
            padding: 0;
        }
        li {
            margin-bottom: 5px;
        }
        p {
            margin: 0;
        }
        . special-day {
            color: #e74c3c;
            font-weight: bold;
        }
    </style>
</head>
<body>
    <section style="margin-top: 10px;">
        <div class="container">
            <center><h1>IF...ELSE</h1></center>
            <h1>——————————</h1>
            @{
                int num1 = 10, num2 = 3;
                String num3;
                if (num1 % num2 == 0)
                {
                    num3 = "The number is Even!";
                }
                else
                {
```

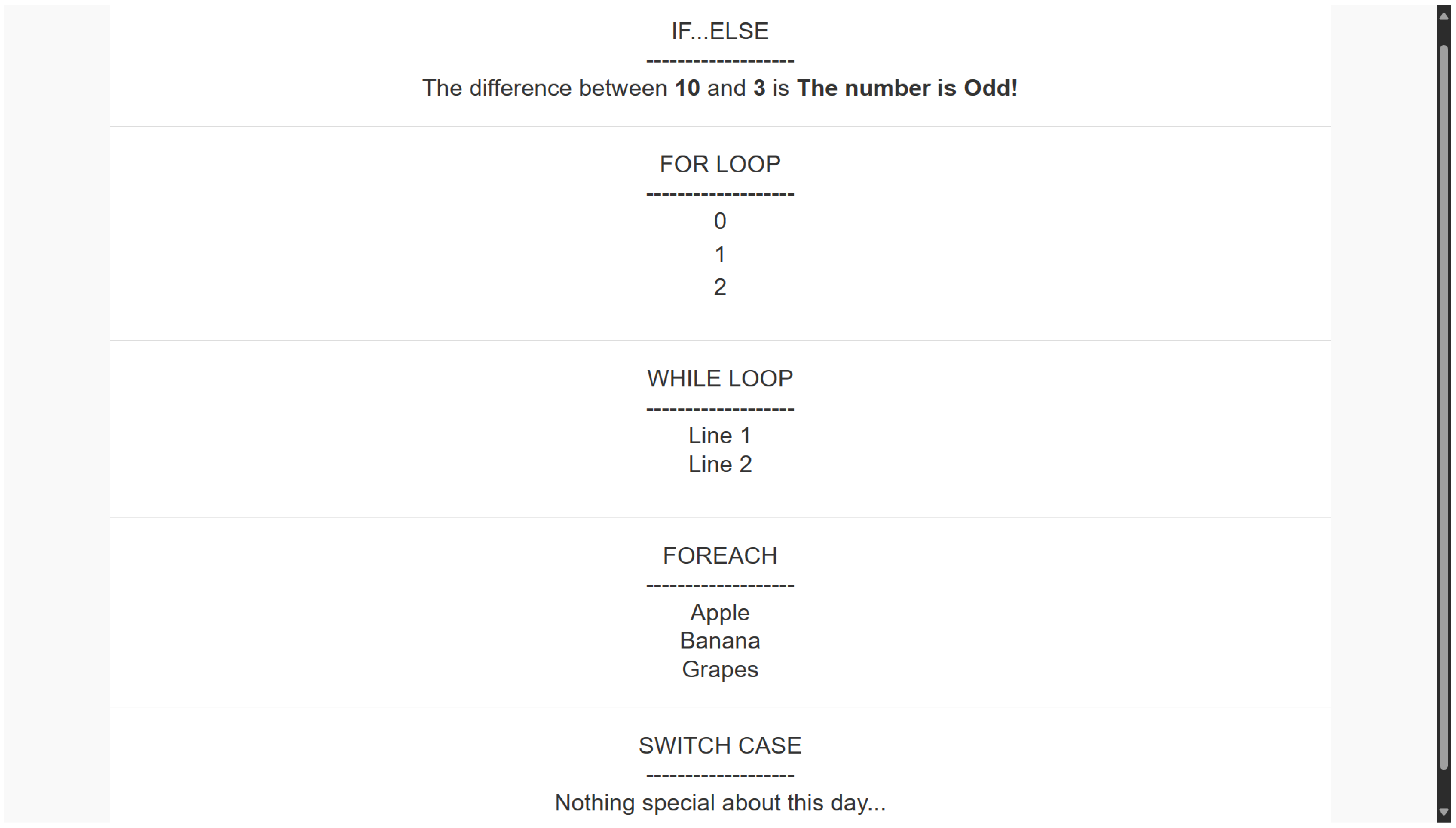
```
                num3 = "The number is Odd!";
            }
        }
        <h1>The difference between <b>@num1</b> and <b>@num2</b> is <b>@num3</b></h1>
    </div>
</section>
<section>
    <div class="container">
        <center><h1>FOR LOOP</h1></center>
        <h1>——————————</h1>
        <ul>
            @for (int i = 0; i <= 2; i++)
            {
                <li><h1>@i</h1></li>
            }
        </ul>
    </div>
</section>
<section>
    <div class="container">
        <center><h1>WHILE LOOP</h1></center>
        <h1>——————————</h1>
        <ul>
            @{
                var n = 0;
                while (n < 2)
                {
                    n += 1;
                    <p><h1>Line @n</h1></p>
                }
            }
        </ul>
    </div>
</section>
<section>
    <div class="container">
        <center><h1>FOREACH</h1></center>
        <h1>—————————</h1>
        @{
            string[] members = { "Apple", "Banana","Grapes"};
            foreach (var person in members)
            {
                <p><h1>@person</h1></p>
            }
        }
    </div>
</section>
<section>
    <div class="container">
        <center><h1>SWITCH CASE</h1></center>
        <h1>—————————</h1>
        @switch (DateTime.Now.DayOfWeek)
        {
            case DayOfWeek.Monday:
                <span class="special-day">Uh-oh...</span>
                break;
            case DayOfWeek.Friday:
                <span class="special-day">Weekend coming up!</span>
                break;
            case DayOfWeek.Saturday:
                <span class="special-day">Saturday AA Gaya!</span>
                break;
            case DayOfWeek.Sunday:
                <span class="special-day">Finally weekend!</span>
                break;
            default:
                <h1><span>Nothing special about this day...</span></h1>
                break;
        }
    </div>
  </section>
</body>
```

</html>

19. Create ASP.NET Core MVC registration page using Razor Syntax and Html Helper methods to register new student.
    First name (Textbox)
    Last name or initial (Textbox)
    Gender (Radio Button)
    Age (Take numeric input only)
    Existing Qualification (Dropdown List)
    Email address (Regular Expression)
    Choose a password (min. 8 characters)

Ans – HomeController.cs

```
using Microsoft.AspNetCore.Mvc;

using Register. Models;

using System. Diagnostics;

namespace Register. Controllers

{
```



```
IF...ELSE
-------------------
The difference between 10 and 3 is The number is Odd!

FOR LOOP
-------------------
0
1
2

WHILE LOOP
-------------------
Line 1
Line 2

FOREACH
-------------------
Apple
Banana
Grapes

SWITCH CASE
-------------------
Nothing special about this day...
```

```
   public class HomeController : Controller
   {
       private readonly ILogger<HomeController> _logger;
       public HomeController(ILogger<HomeController> logger)
       {
           _logger = logger;
       }
       [HttpGet]
       public IActionResult Index()
       {
           return View();
       }
       [HttpPost]
       public IActionResult Index(Registration registration)
       {
           return View("Congratulation", registration);
        }
       public IActionResult Privacy()
       {
           return View();
       }
       [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
       public IActionResult Error()
       {
           return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
       } }}
```

Index.cshtml

```
@model Register.Models.Registration
@{
    Layout = null;
}
<!DOCTYPE html>
<html>
<head>
    <title>RsvpForm</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f5f5f5;
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }
        .container {
            width: 90%; /* Relative width */
            max-width: 600px;
            margin: 4% auto;
            padding: 20px;
            background-color: #fff;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        input[type="text"],
        input[type="email"],
        input[type="password"],
        select,
        input[type="number"] {
            width: calc(100% - 22px); /* Adjusted width */
            padding: 10px;
            margin-bottom: 10px;
            border: 1px solid #ccc;
            border-radius: 5px;
            box-sizing: border-box;
            font-size: 16px;
        }
        input[type="submit"] {
            width: 100%; /* Relative width */
            padding: 10px;
            border: none;
            border-radius: 5px;
            background-color: #4CAF50;
            color: #fff;
            font-size: 16px;
            cursor: pointer;
        }
            input[type="submit"]:hover {
                background-color: #45a049;
            }
        p {
            margin-bottom: 10px;
        }
    </style>
    <script>
        function validateForm() {
            var firstName = document.getElementById("FirstName").value;
            var lastName = document.getElementById("LastName").value;
            var gender = document.querySelector('input[name="Gender"]:checked');
            var age = document.getElementById("Age").value;
            var qualification = document.getElementById("Qualification").value;
            var email = document.getElementById("Email").value;
            var password = document.getElementById("Password").value;
            if (firstName.trim() == "") {
                alert("Enter your first name");
                return false;
            }
            if (lastName.trim() == "") {
                alert("Enter your last name");
                return false;
            }
            if (!gender) {
                alert("Select your gender");
                return false;
            }
            if (age.trim() == "") {
                alert("Enter your age");
                return false;
            }
            if (qualification.trim() == "") {
                alert("Enter your qualification");
                return false;
            }
            if (email.trim() == "") {
                alert("Enter your email");
                return false;
```

```
            }
            if (password.trim() == "") {
                alert("Enter your password");
                return false;
            }
            return true;
        }
    </script>
</head>
<body>
    <div class="container">
        <center><h1>Registration Form</h1></center>
        @using (Html.BeginForm("Index", "Home", FormMethod.Post, new { onsubmit = "return
validateForm();" }))
        {
            <p>
                Your First Name : @Html.TextBoxFor(x => x.FirstName, new { id = "FirstName" })
            </p>
            <p>
                Your Last Name: @Html.TextBoxFor(x => x.LastName, new { id = "LastName" })
            </p>
            <p>
                Your Gender:
                @Html.RadioButtonFor(x => x.Gender, "Male", new { id = "GenderMale" }) Male
                @Html.RadioButtonFor(x => x.Gender, "Female", new { id = "GenderFemale" }) Female
            </p>
            <p>
                Your Age: @Html.TextBoxFor(x => x.Age, new { id = "Age", type = "number", min = "11", max =
"24" })
```

```
            </p>
            <p>
                Your Qualification: @Html.DropDownListFor(x => x.Qualification, new SelectList(new
List<string> { "High School", "College", "Bachelor's Degree", "Master's Degree", "PhD" }), "Select", new { id
= "Qualification" })
            </p>
            <p>
                Your email: @Html.TextBoxFor(x => x.Email, new { id = "Email", type = "email" })
            </p>
            <p>
                Your password: @Html.PasswordFor(x => x.Password, new { id = "Password", minlength = "3",
maxlength = "8" })
            </p>
            <input type="submit" value="Register Me" />
        }
    </div>
</body>
</html>
```

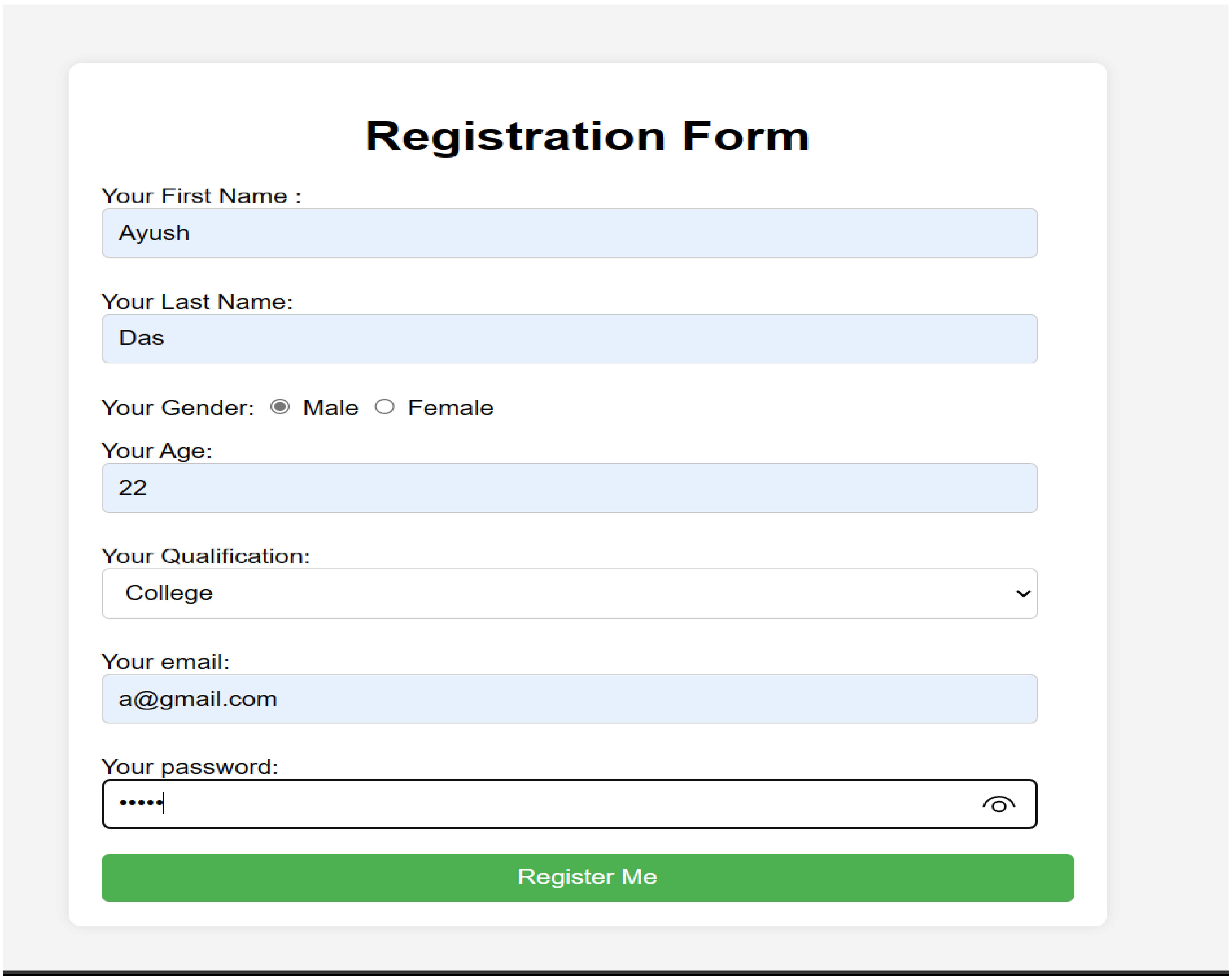
Registration.cs

```
using System.ComponentModel.DataAnnotations;
namespace Register.Models
{
    public class Registration
    {
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Gender { get; set; }
        public int Age { get; set; }
        public string Qualification { get; set; }
        public string Email { get; set; }
        public string Password { get; set; }
    }}
```

Congradulation.cshtml

```
@model Register.Models.Registration
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Congratulations</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f4f4f4;
        }
        .container {
            max-width: 600px;
            margin: 20px auto;
            padding: 20px;
            background-color: #fff;
            border-radius: 8px;
            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
        }
        h1, h2 {
            text-align: center;
        }
        h1 {
          color: #333;
        }
        h2 {
          color: #666;
        }
        .details {
          margin-top: 20px;
        }
          .details p {
            margin: 10px 0;
          }
    </style>
</head>
<body>
    <div class="container">
        <h1>Congratulations, @Model.FirstName!</h1>
        <h2>Your Registration has been Successfully completed.</h2>
        <div class="details">
            <p><strong>Your full name:</strong> @Model.FirstName @Model.LastName</p>
            <p><strong>Your age:</strong> @Model.Age</p>
            <p><strong>Your qualification:</strong> @Model.Qualification</p>
            <p><strong>Your email:</strong> @Model.Email</p>
            <p><strong>Your password:</strong> @Model.Password</p>
        </div>
    </div>
</body>
```

```
</html>
```



Registration Form

Your First Name : Ayush

Your Last Name: Das

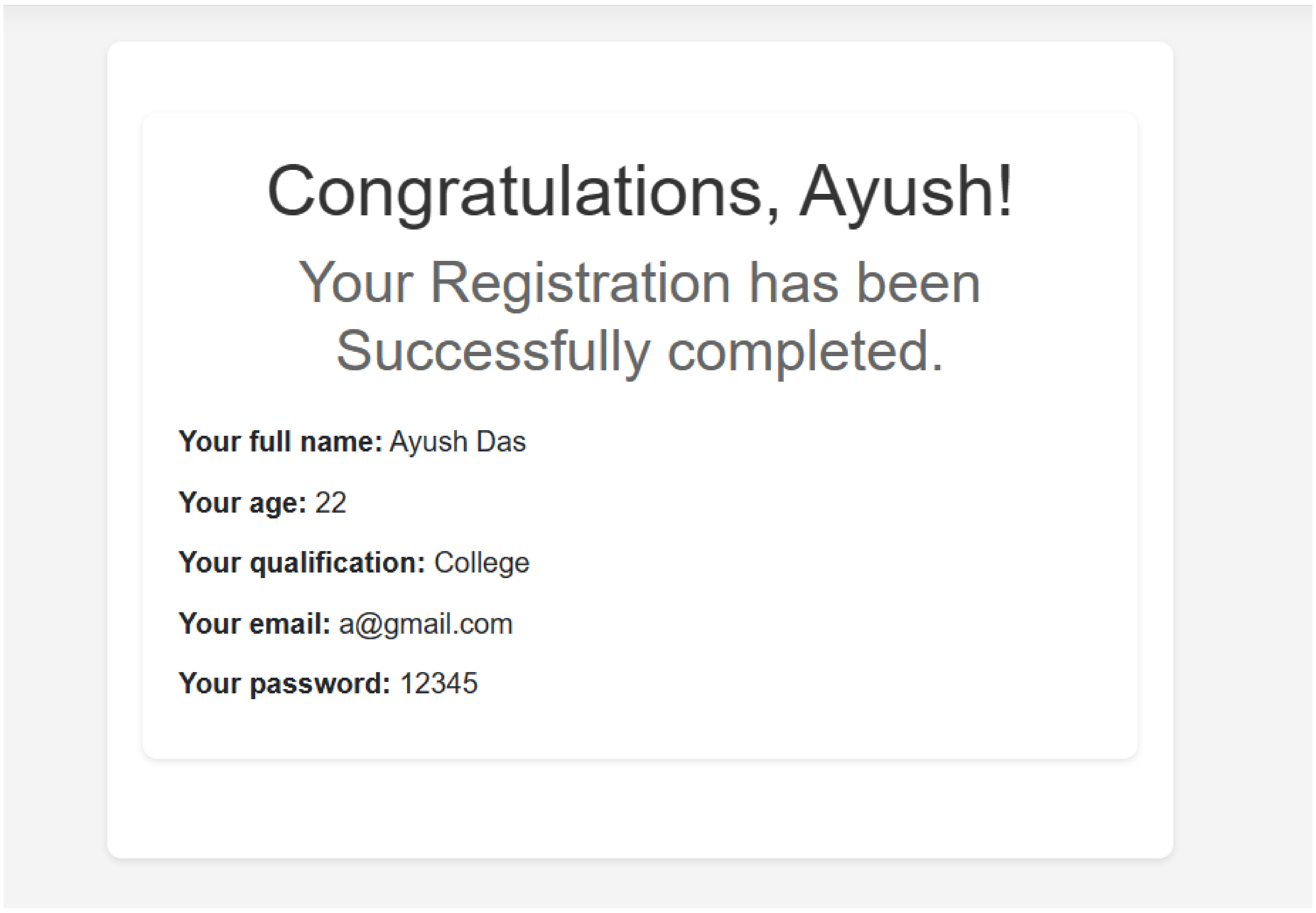
Your Gender: [✓ Male] Female

Your Age: 22

Your Qualification: College

Your email: a@gmail.com

Your password: •••••



Congratulations, Ayush!

Your Registration has been Successfully completed.

Your full name: Ayush Das

Your age: 22

Your qualification: College

Your email: a@gmail.com

Your password: 12345

20. Develop ASP.NET core MVC web application to perform CRUD operation by using entity framework with validation on Book model.
    Book (BookId, BookTitle, AuthorName, Publication and Price)

Ans - Book.cs

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

```
namespace Program28.Models
{
   public class Book
   {
      [Key]
      public int BookId { get; set; }
      [Required(ErrorMessage = "Title is required")]
      public string BookTitle { get; set; }
      [Required(ErrorMessage = "Author name is required")]
      public string AuthorName { get; set; }
      [Required(ErrorMessage = "Publication is required")]
      public string Publication { get; set; }
      [Required(ErrorMessage = "Price is required")]
      [Range(0, double.MaxValue, ErrorMessage = "Price must be a positive number")]
      public decimal Price { get; set; }
   }}
```

Program28Context.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using Program28.Models;
namespace Program28.Data
{
   public class Program28Context : DbContext
   {
      public Program28Context (DbContextOptions<Program28Context> options)
         : base(options)
```

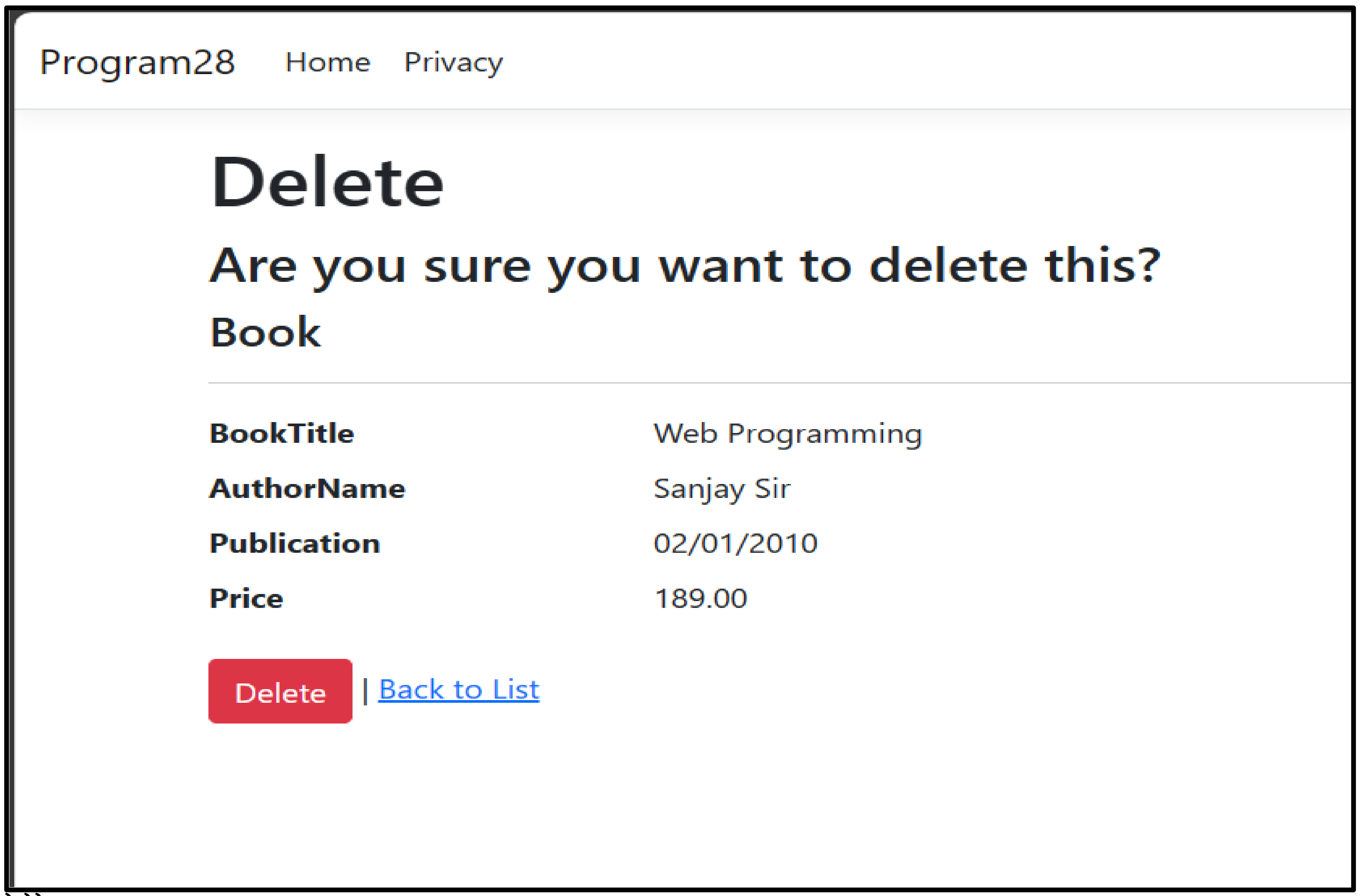
```
        {
        }
        public DbSet<Program28.Models.Book> Book { get; set; } = default!;
    }}
```

InitialMigration.cs

```
using Microsoft.EntityFrameworkCore.Migrations;
namespace Program28.Migrations
{
    public partial class InitialMigration : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Book",
                columns: table => new
                {
                    BookId = table.Column<int>(type: "int", nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
                    BookTitle = table.Column<string>(type: "nvarchar(max)", nullable: false),
                    AuthorName = table.Column<string>(type: "nvarchar(max)", nullable: false),
                    Publication = table.Column<string>(type: "nvarchar(max)", nullable: false),
                    Price = table.Column<decimal>(type: "decimal(18,2)", nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Book", x => x.BookId);
                });
        }
        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "Book");
        } }}
```

BookController.cs

```
        public IActionResult Create()
        {
            return View();
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("BookId,BookTitle,AuthorName,Publication,Price")]
Book book)
        {
            if (ModelState.IsValid)
            {
                _context.Add(book);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(book);
        }
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }
            var book = await _context.Book.FindAsync(id);
            if (book == null)
            {
                return NotFound();
            }
            return View(book);
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id,
[Bind("BookId,BookTitle,AuthorName,Publication,Price")] Book book)
        {
            if (id != book.BookId)
            {
                return NotFound();
            }
            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(book);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!BookExists(book.BookId))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
                return RedirectToAction(nameof(Index));
            }
            return View(book);
        }
        public async Task<IActionResult> Delete(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }
            var book = await _context.Book
                .FirstOrDefaultAsync(m => m.BookId == id);
            if (book == null)
            {
                return NotFound();
            }
            return View(book);
        }
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            var book = await _context.Book.FindAsync(id);
            if (book != null)
            {
                _context.Book.Remove(book);
            }

            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
        private bool BookExists(int id)
        {
            return _context.Book.Any(e => e.BookId == id);
```



Program28 Home Privacy

Delete

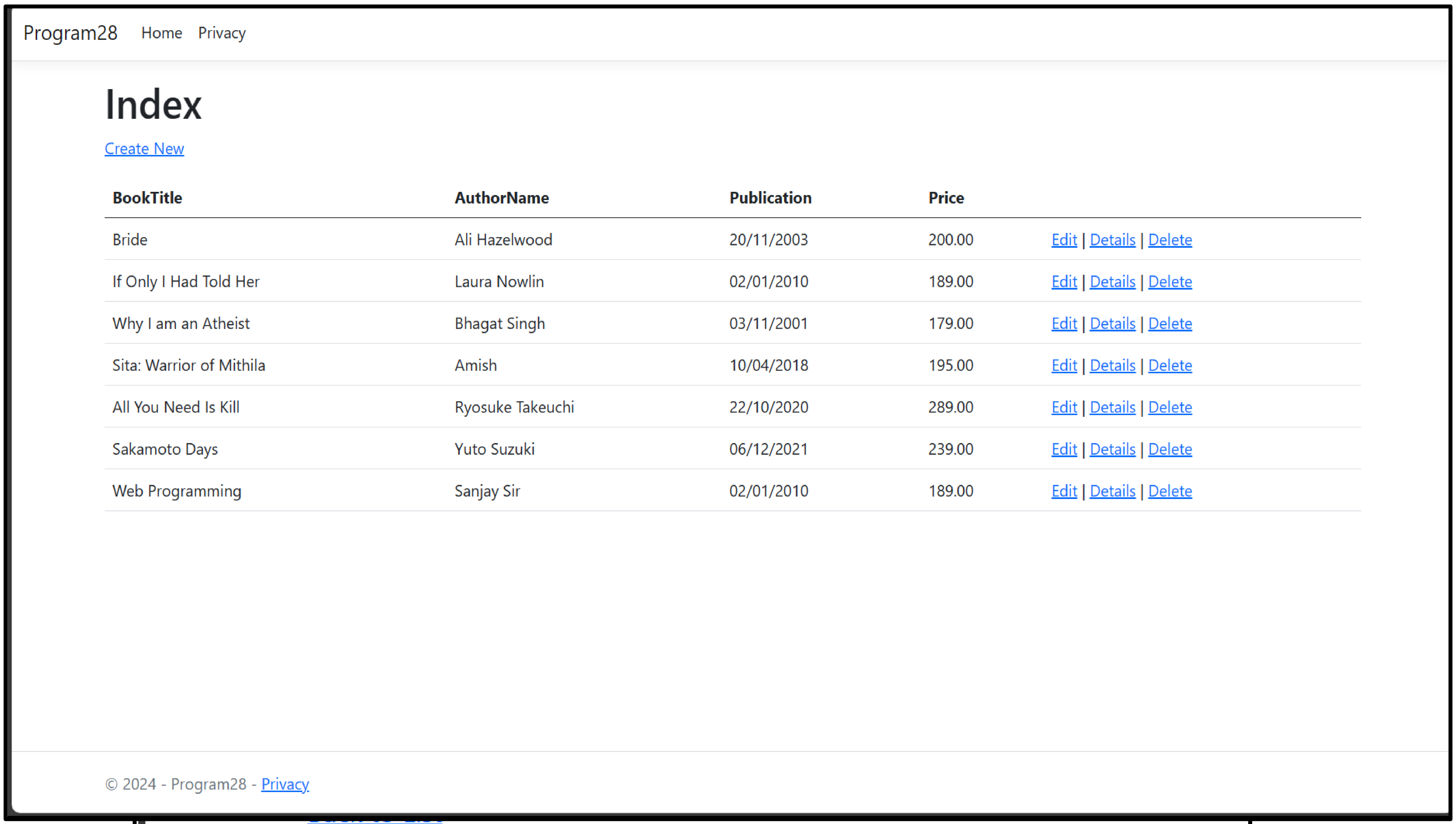
Are you sure you want to delete this?

Book

BookTitle	Web Programming
AuthorName	Sanjay Sir
Publication	02/01/2010
Price	189.00

Delete | Back to List

}}}



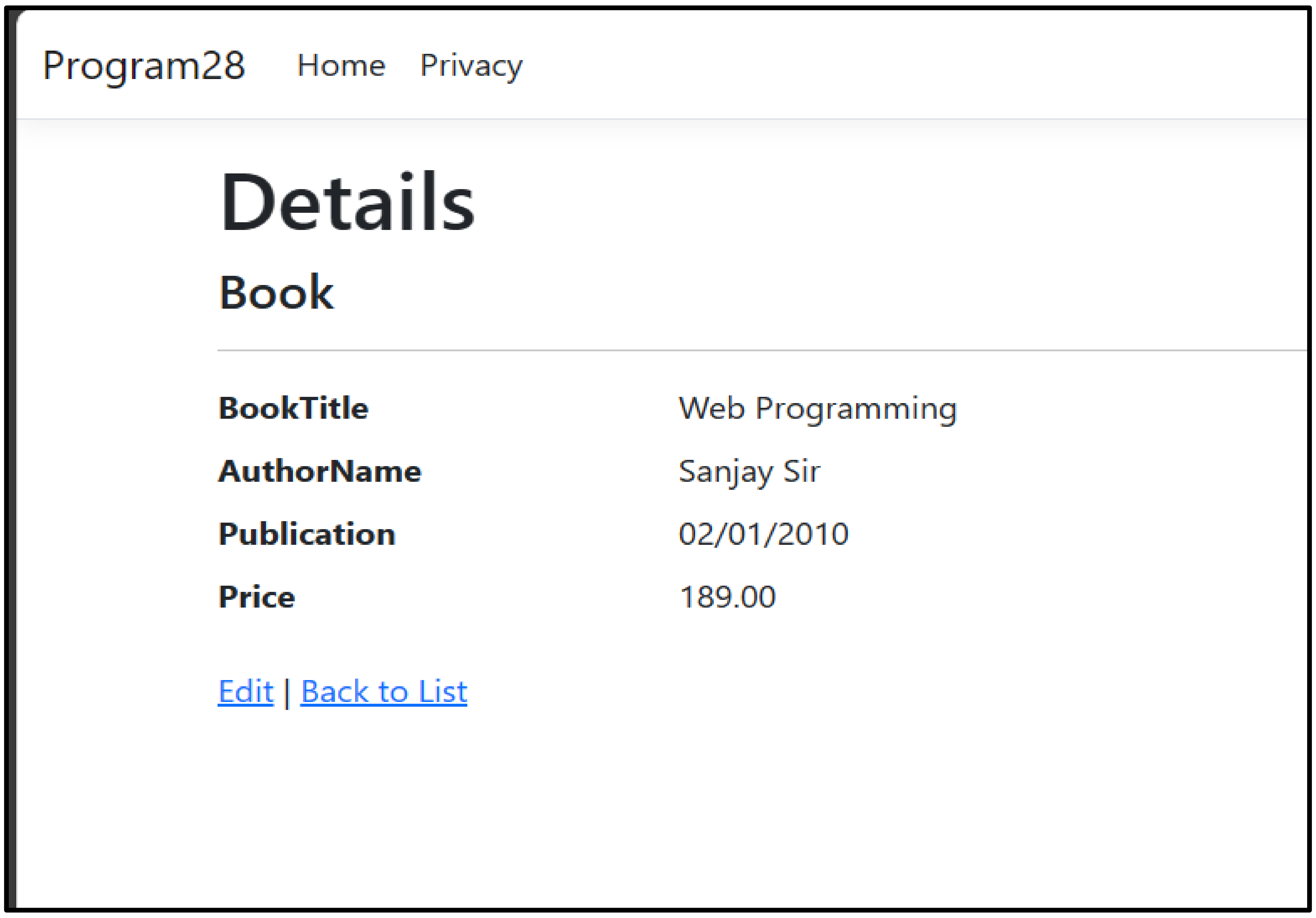
Program28 Home Privacy

Index

Create New

BookTitle	AuthorName	Publication	Price	
Bride	Ali Hazelwood	20/11/2003	200.00	Edit | Details | Delete
If Only I Had Told Her	Laura Nowlin	02/01/2010	189.00	Edit | Details | Delete
Why I am an Atheist	Bhagat Singh	03/11/2001	179.00	Edit | Details | Delete
Sita: Warrior of Mithila	Amish	10/04/2018	195.00	Edit | Details | Delete
All You Need Is Kill	Ryosuke Takeuchi	22/10/2020	289.00	Edit | Details | Delete
Sakamoto Days	Yuto Suzuki	06/12/2021	239.00	Edit | Details | Delete
Web Programming	Sanjay Sir	02/01/2010	189.00	Edit | Details | Delete

© 2024 - Program28 - Privacy



Program28 Home Privacy

Details

Book

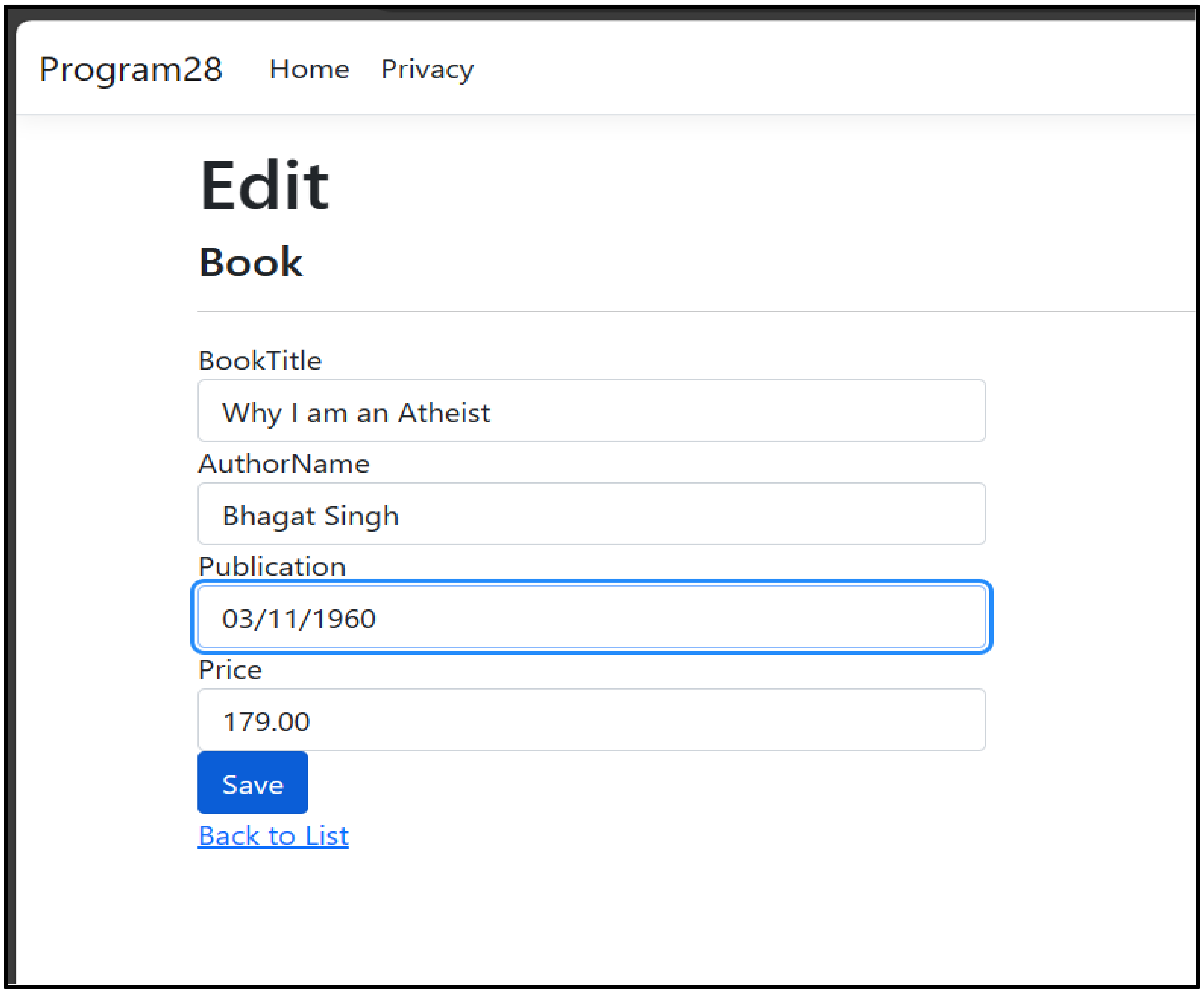
BookTitle	Web Programming
AuthorName	Sanjay Sir
Publication	02/01/2010
Price	189.00

Edit | Back to List

21. Develop ASP.NET Core MVC application to perform CRUD operation by using entity framework with validation on Student, Course and Enrollment models.

Ans - Student.cs

```
using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace CRUD.Models

{

  public class Student

  {

    [Key]

    public int StudentID { get; set; }

    [Required(ErrorMessage = "First Name is required")]
```



Edit

Book

BookTitle

Why I am an Atheist

AuthorName

Bhagat Singh

Publication

03/11/1960

Price

179.00

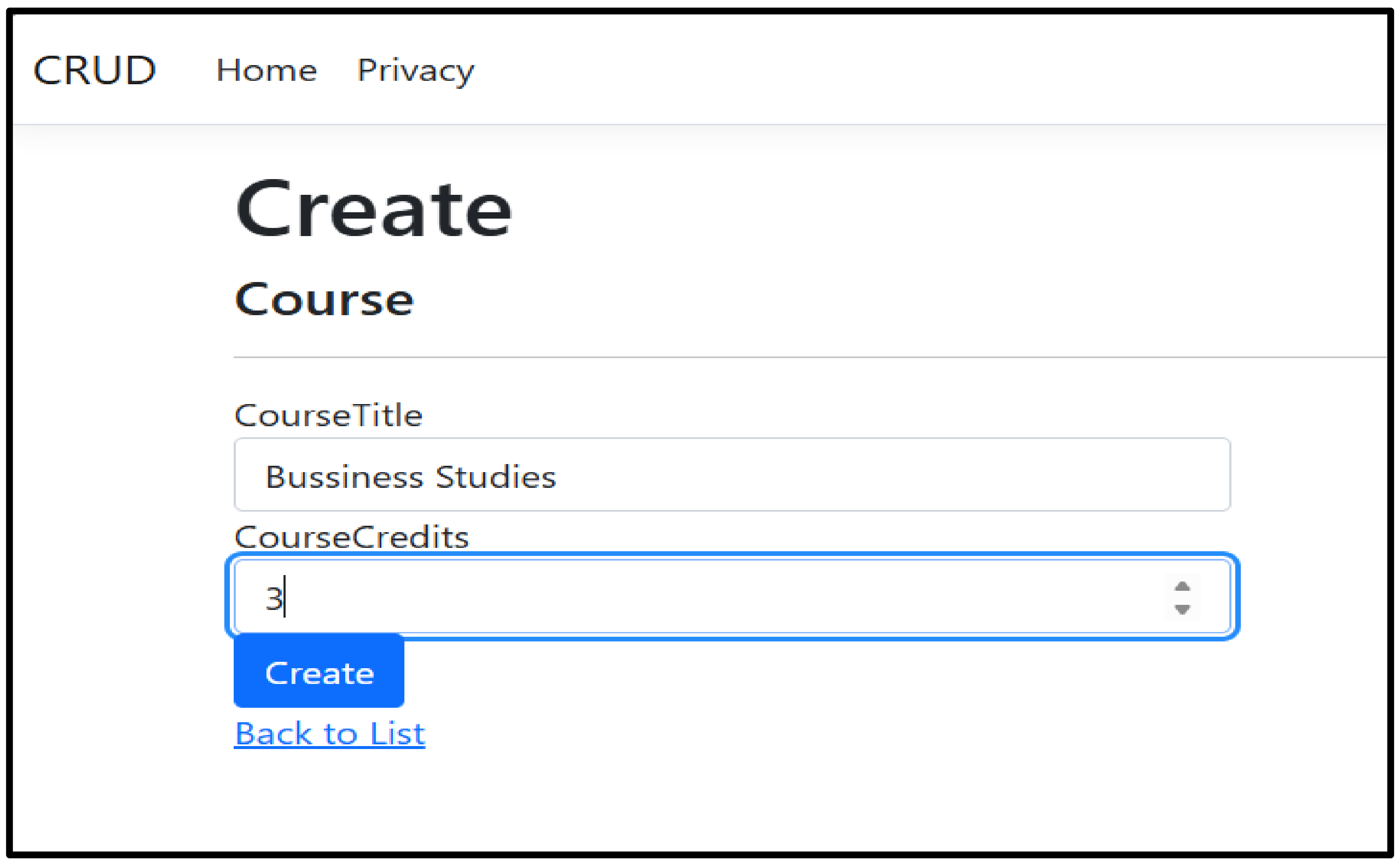
```
        [StringLength(50)]
        public string FirstName { get; set; }
        [Required(ErrorMessage = "Last Name is required")]
        [StringLength(50)]
        public string LastMidName { get; set; }
        [DataType(DataType.Date)]
        [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
        public DateTime EnrollmentDate { get; set; }
        public ICollection<Enrollment> Enrollments { get; set; }
        public Student()
        {
            Enrollments = new List<Enrollment>();
        }
    }
}
```

Course.cs

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
namespace CRUD.Models
{
    public class Course
    {
        [Key]
        public int CourseID { get; set; }
        [Required(ErrorMessage = "Course Title is required")]
        [StringLength(100, ErrorMessage = "Title cannot exceed 100 characters")]
        public string CourseTitle { get; set; }
        [Required(ErrorMessage = "Credits are required")]
        [Range(1, int.MaxValue, ErrorMessage = "Credits must be a positive number")]
        public int CourseCredits { get; set; }
        public ICollection<Enrollment> Enrollments { get; set; }
        public Course()
        {
            Enrollments = new List<Enrollment>();
        } }}
```

Enrollment.cs

```
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
namespace CRUD.Models
{
    public class Enrollment
    {
        [Key]
        public int EnrollmentID { get; set; }
        [Required(ErrorMessage = "CourseID is required")]
        public int CourseID { get; set; }
        [Required(ErrorMessage = "StudentID is required")]
        public int StudentID { get; set; }
        [DataType(DataType.Date)]
        [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
        public DateTime EnrollmentDate { get; set; }
        [ForeignKey("CourseID")]
        public Course? Course { get; set; }
        [ForeignKey("StudentID")]
        public Student? Student { get; set; }
    }}
```



Create

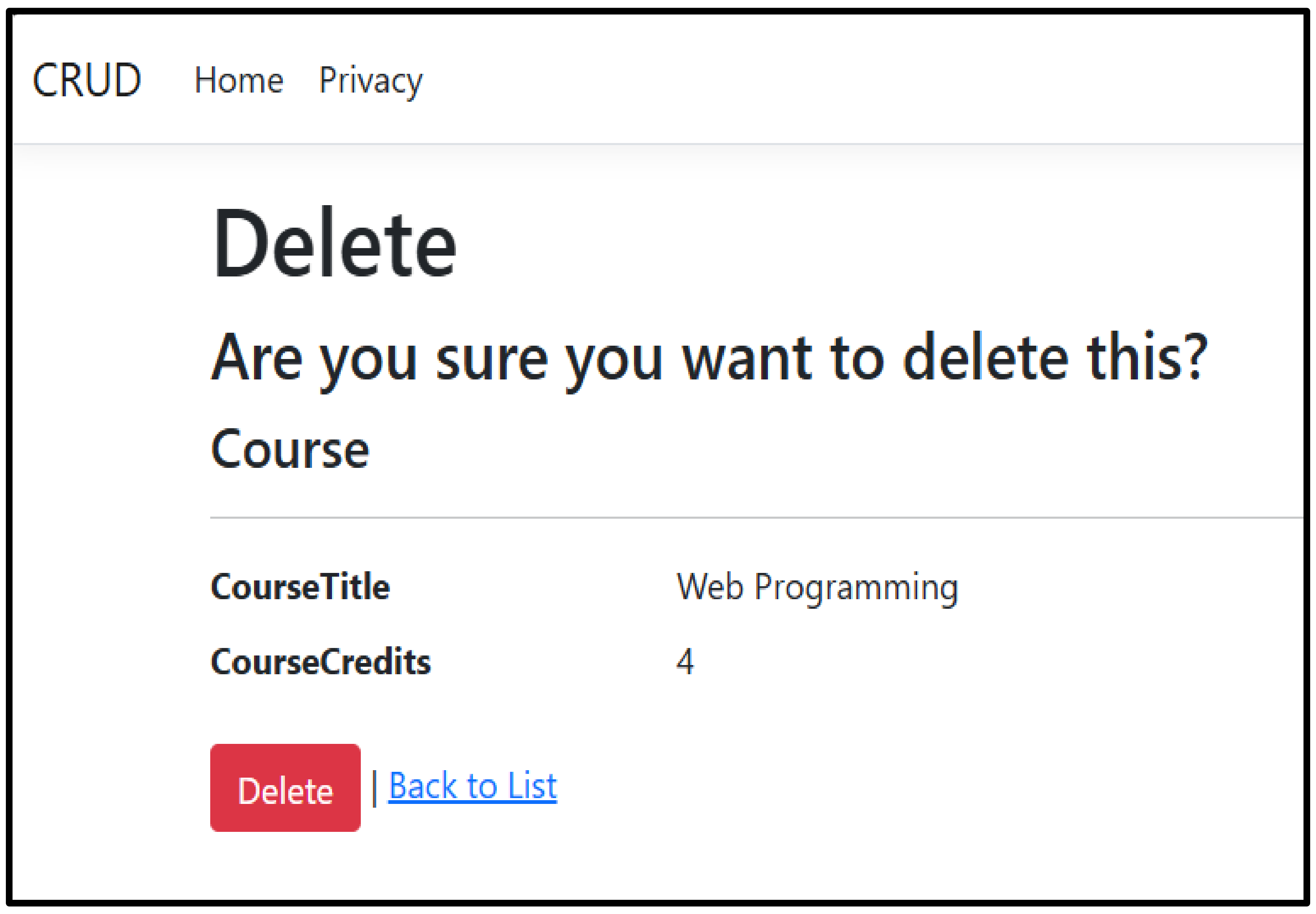
Course

CourseTitle

Bussiness Studies

CourseCredits

3

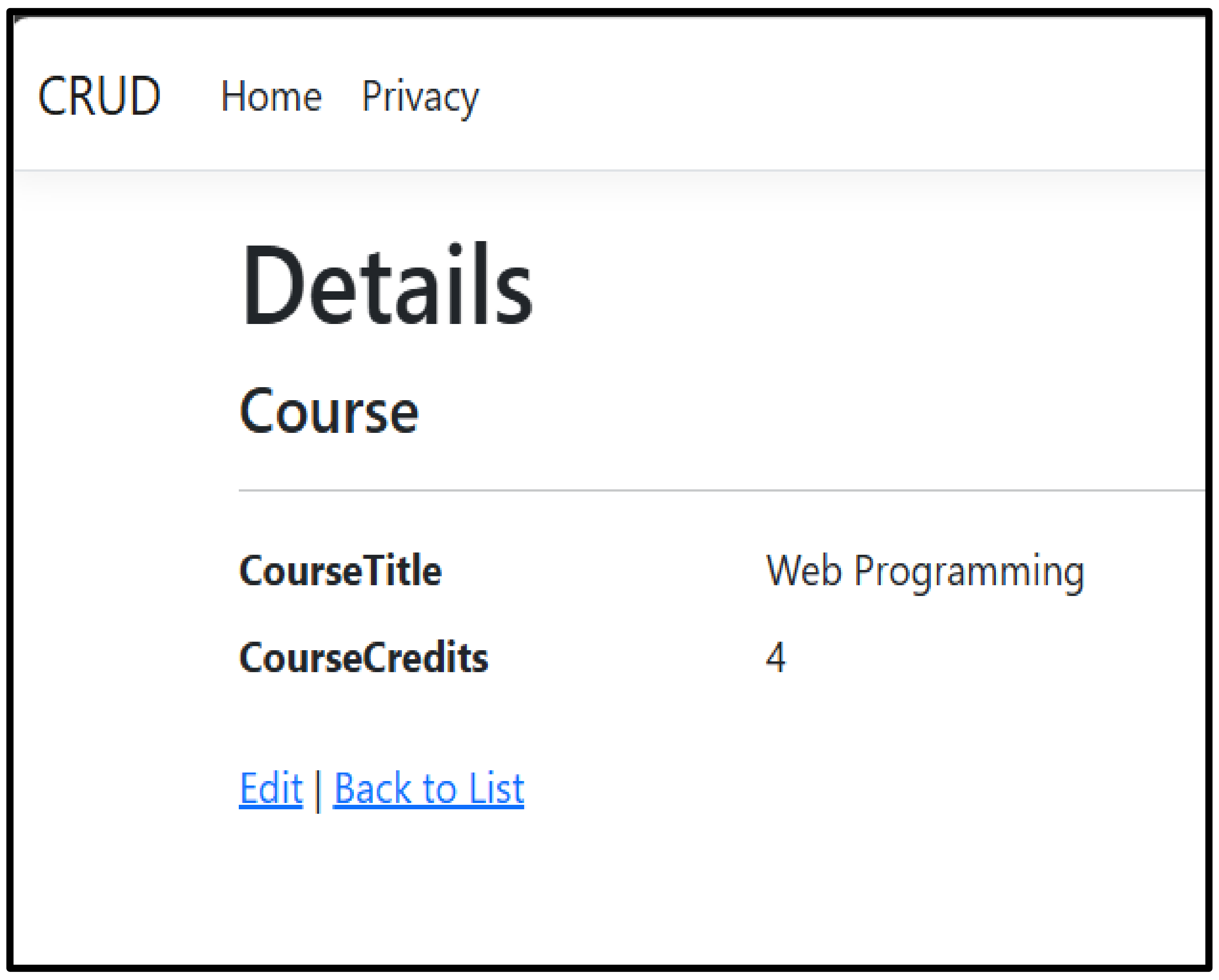


Delete

Are you sure you want to delete this?

Course

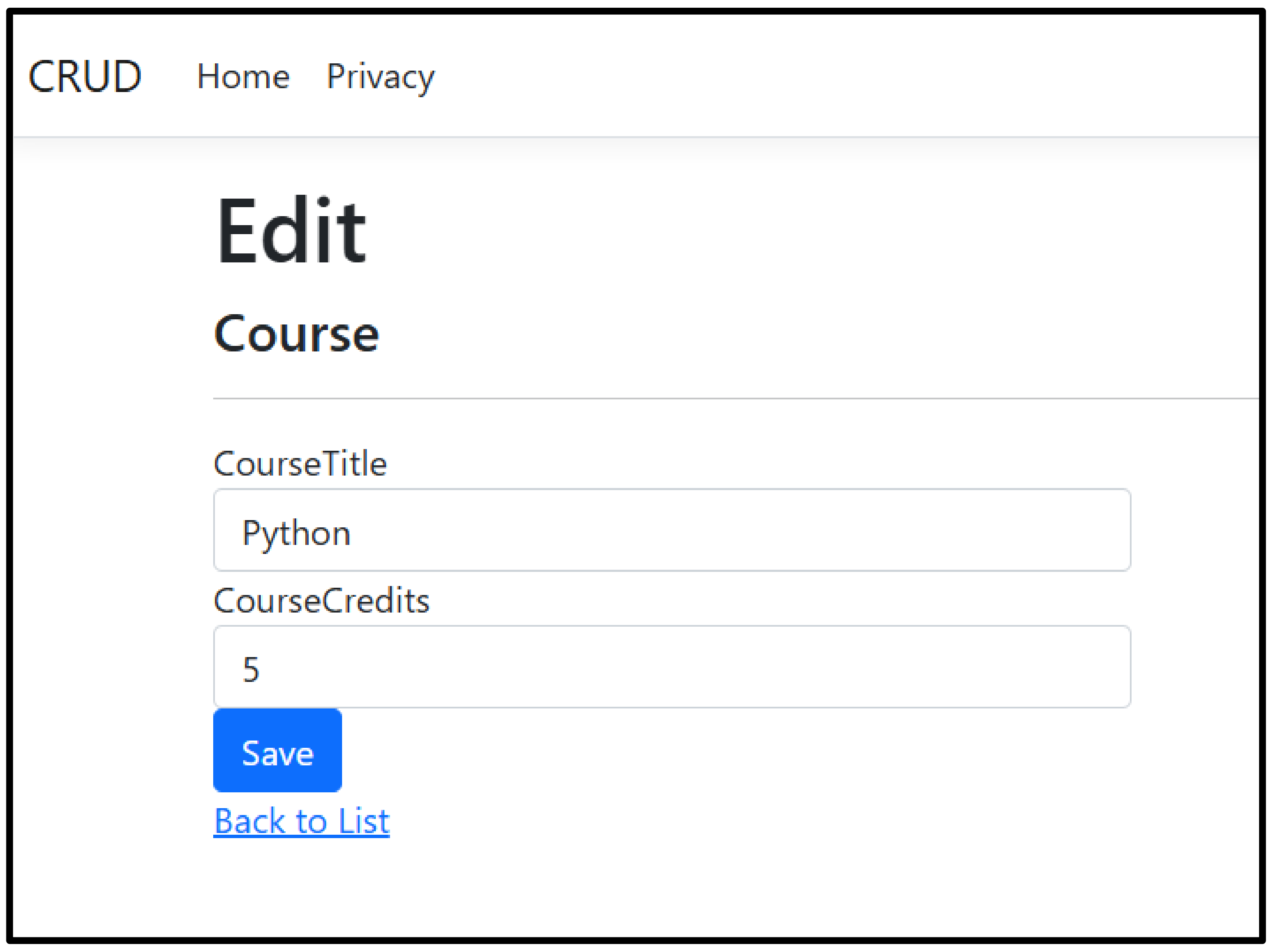
CourseTitle	Web Programming
CourseCredits	4



Details

Course

CourseTitle	Web Programming
CourseCredits	4

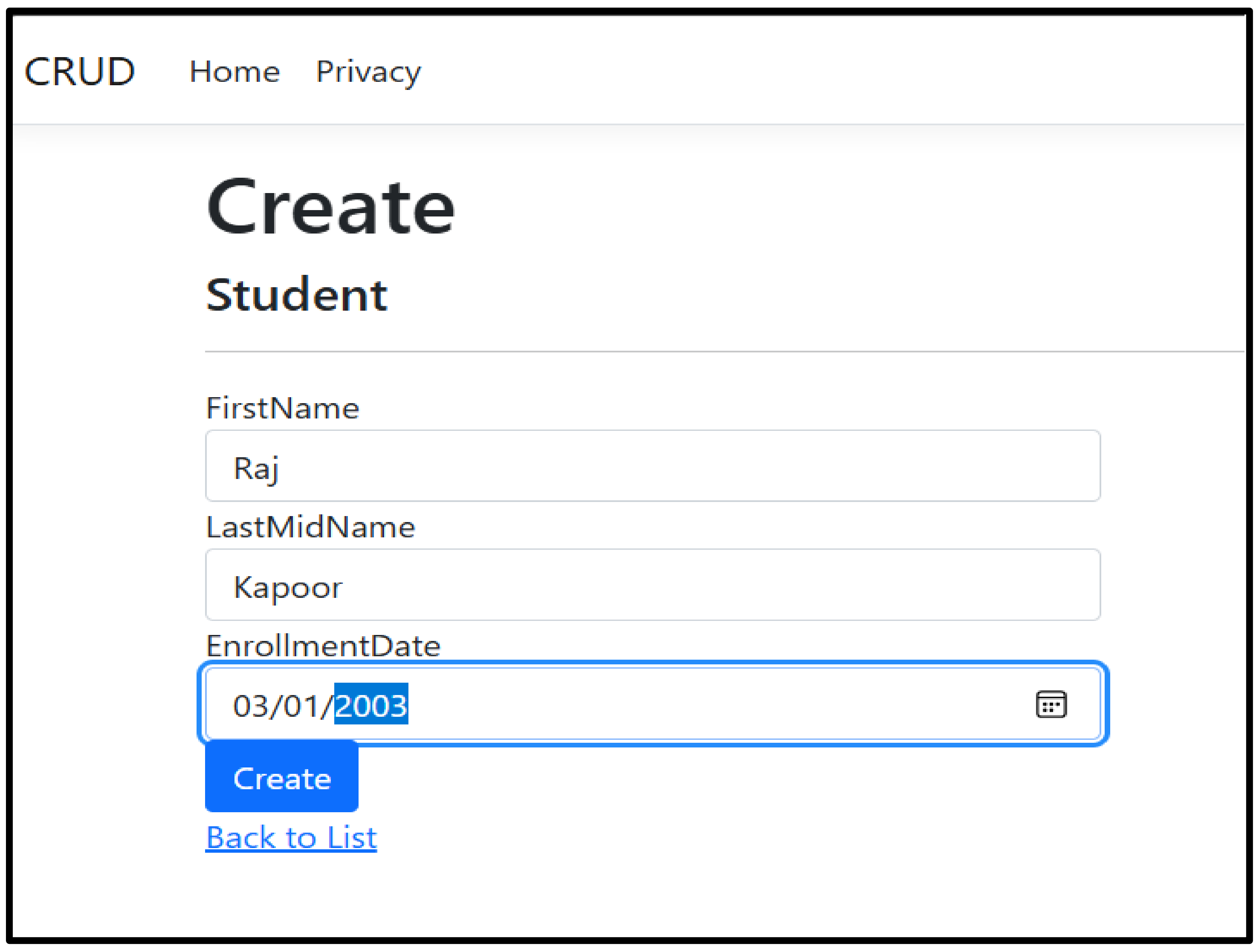


Edit

Course

CourseTitle: Python

CourseCredits: 5



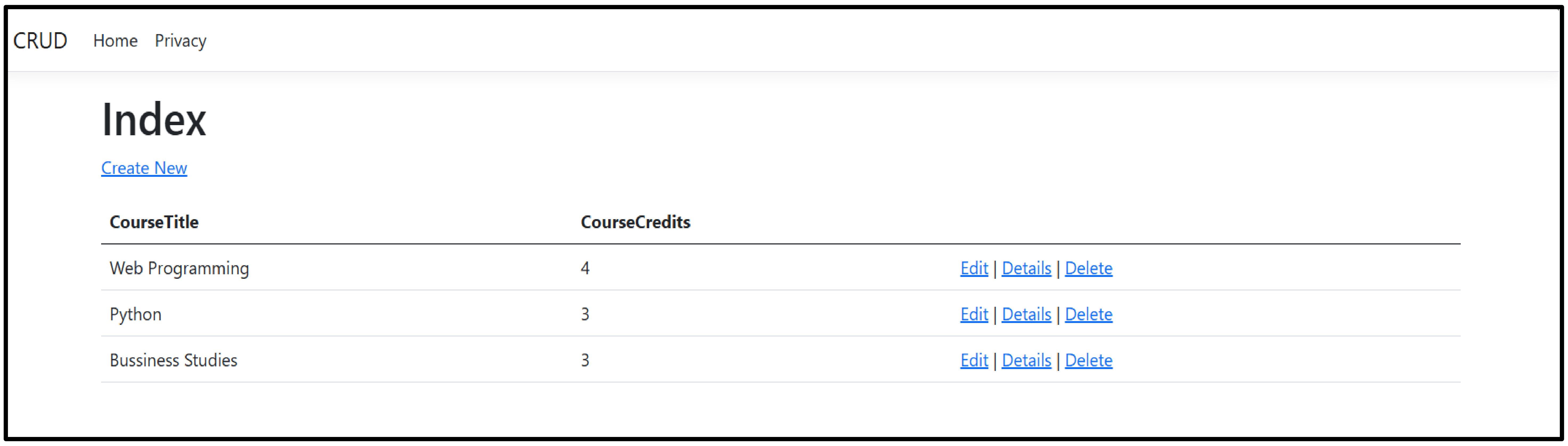
Create

Student

FirstName: Raj

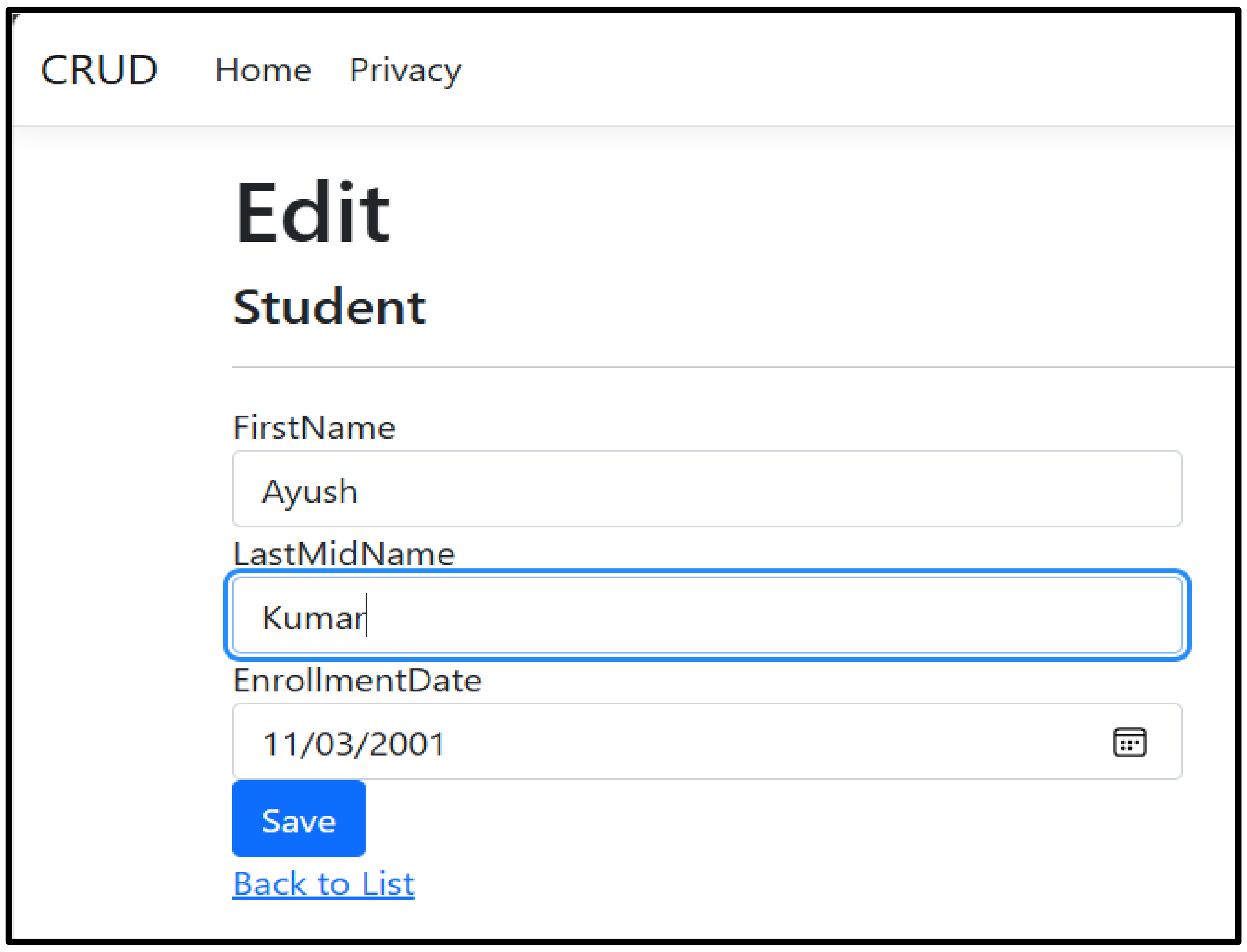
LastMidName: Kapoor

EnrollmentDate: 03/01/2003



Index

CourseTitle	CourseCredits	
Web Programming	4	Edit | Details | Delete
Python	3	Edit | Details | Delete
Bussiness Studies	3	Edit | Details | Delete


CRUD Home Privacy

Edit

Student

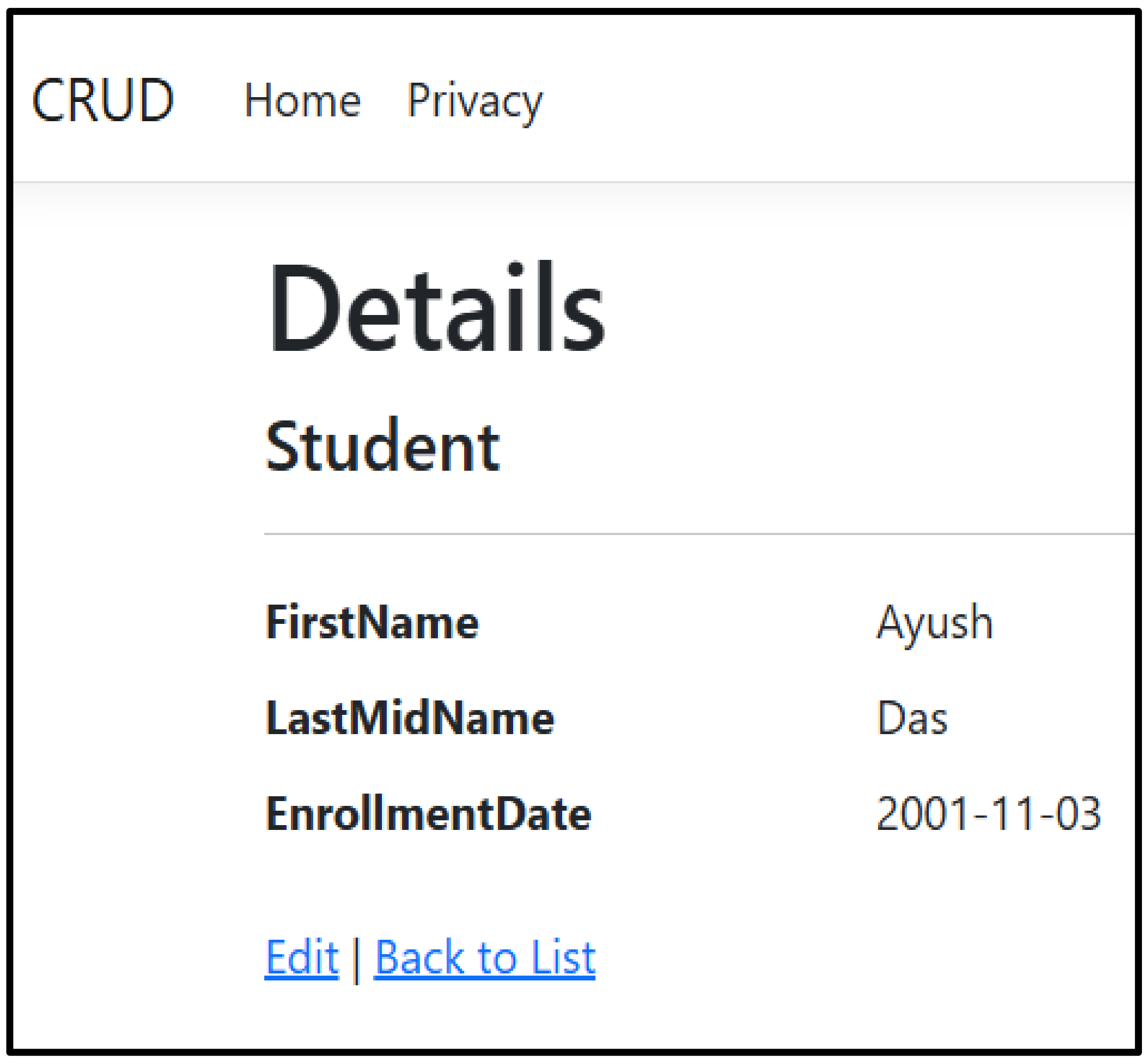
FirstName: Ayush

LastMidName: Kumar

EnrollmentDate: 11/03/2001

Save

Back to List

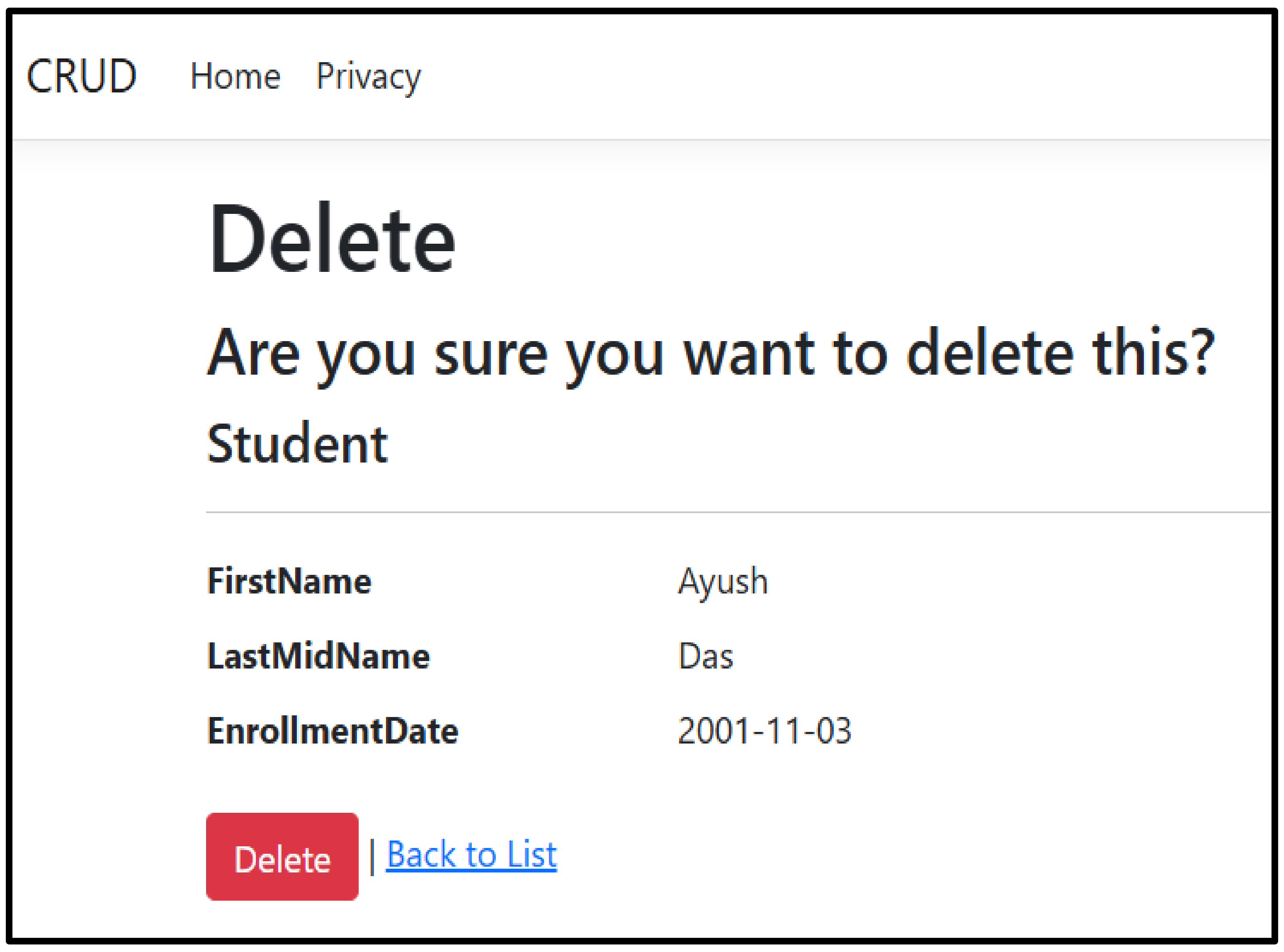

CRUD Home Privacy

Details

Student

FirstName	Ayush
LastMidName	Das
EnrollmentDate	2001-11-03

Edit | Back to List


CRUD Home Privacy

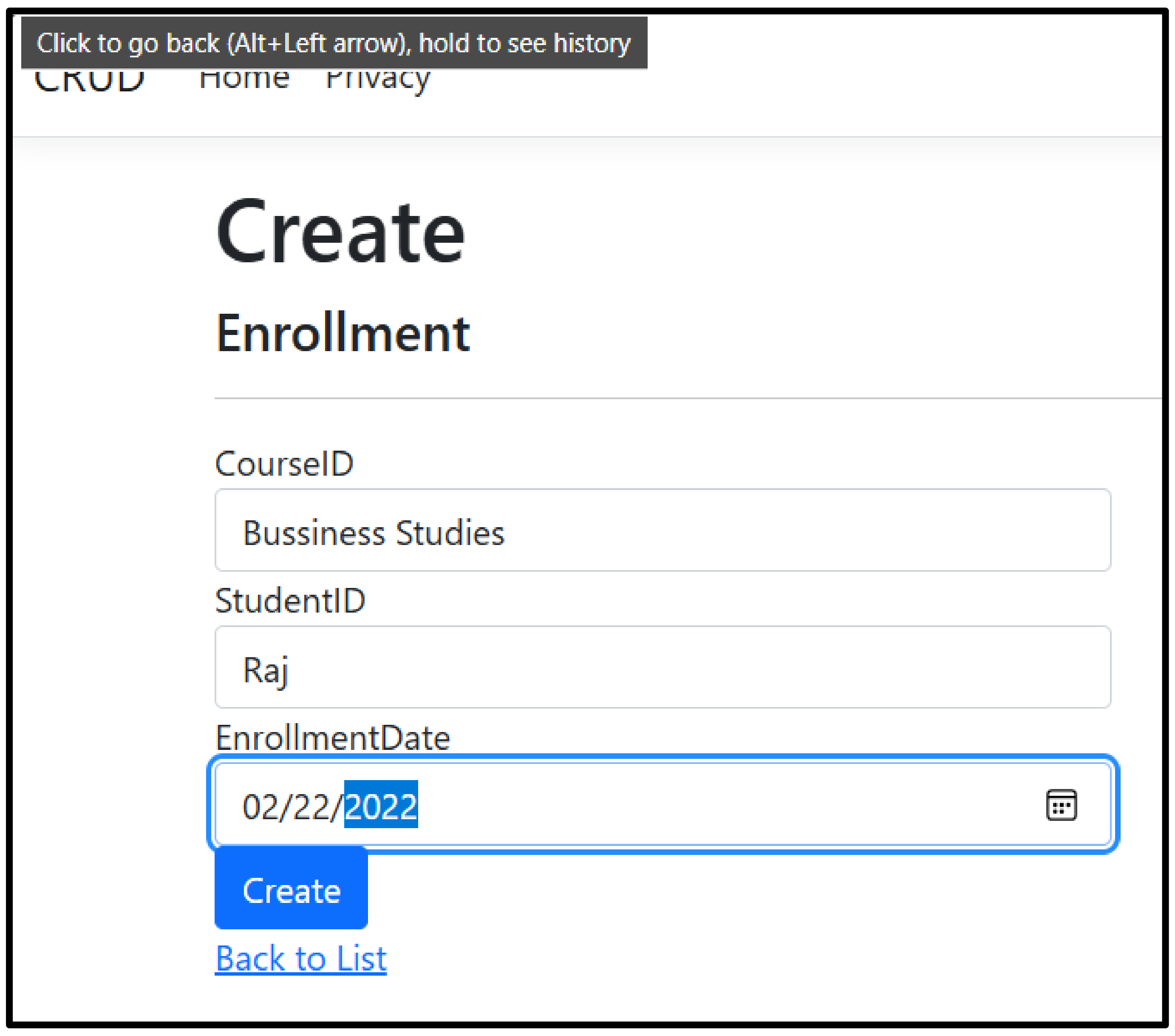
Delete

Are you sure you want to delete this?

Student

FirstName	Ayush
LastMidName	Das
EnrollmentDate	2001-11-03

Delete | Back to List


Click to go back (Alt+Left arrow), hold to see history

CRUD Home Privacy

Create

Enrollment

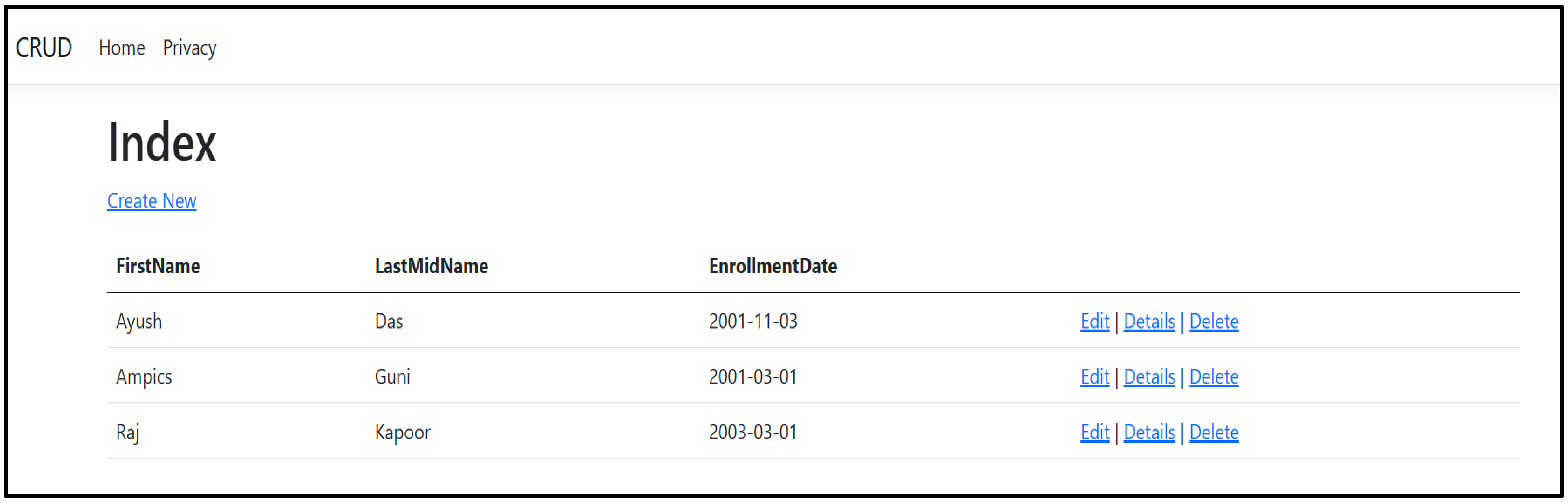
CourseID: Bussiness Studies

StudentID: Raj

EnrollmentDate: 02/22/2022

Create

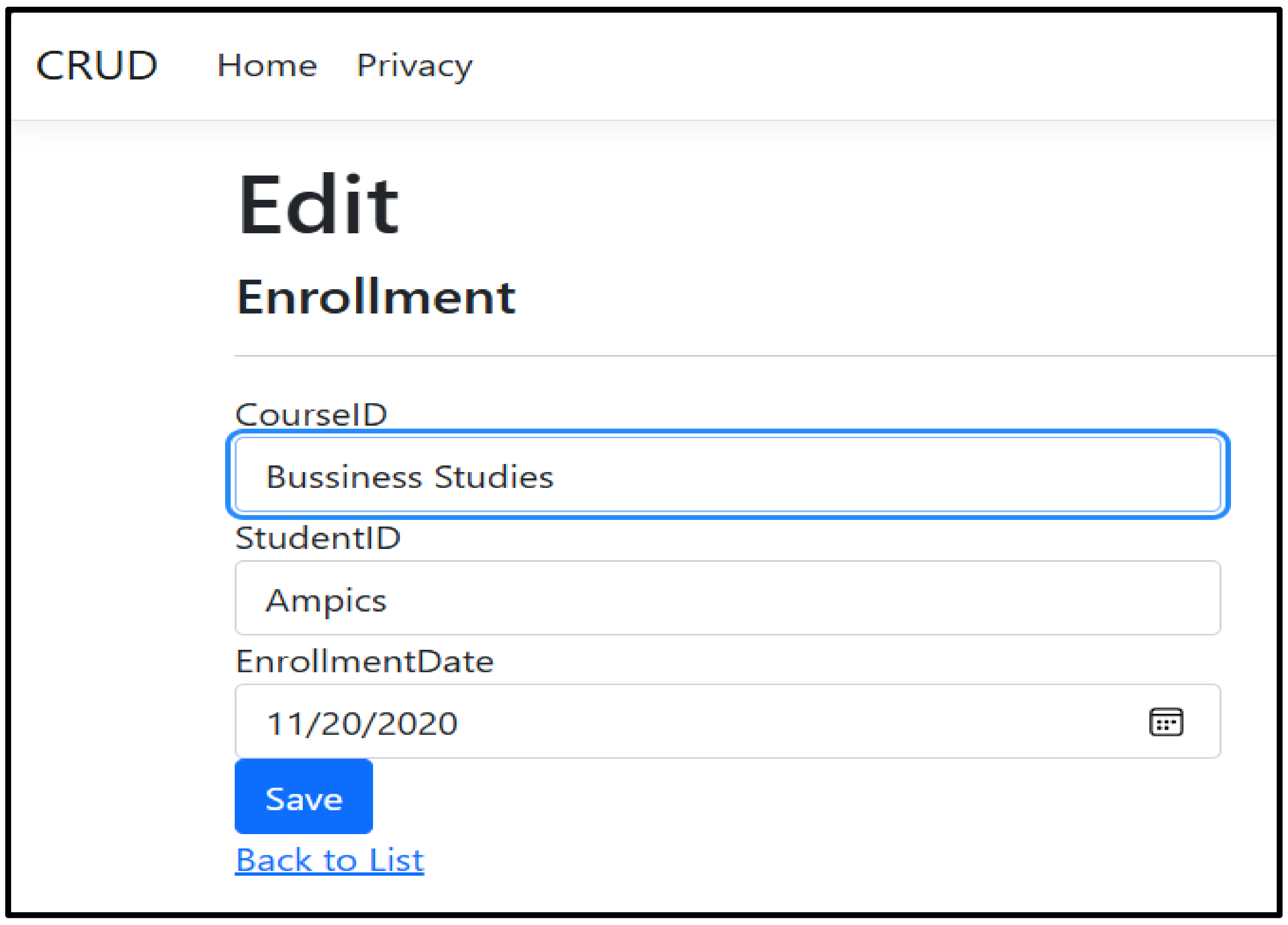
Back to List


CRUD Home Privacy

Index

Create New

FirstName	LastMidName	EnrollmentDate	
Ayush	Das	2001-11-03	Edit | Details | Delete
Ampics	Guni	2001-03-01	Edit | Details | Delete
Raj	Kapoor	2003-03-01	Edit | Details | Delete



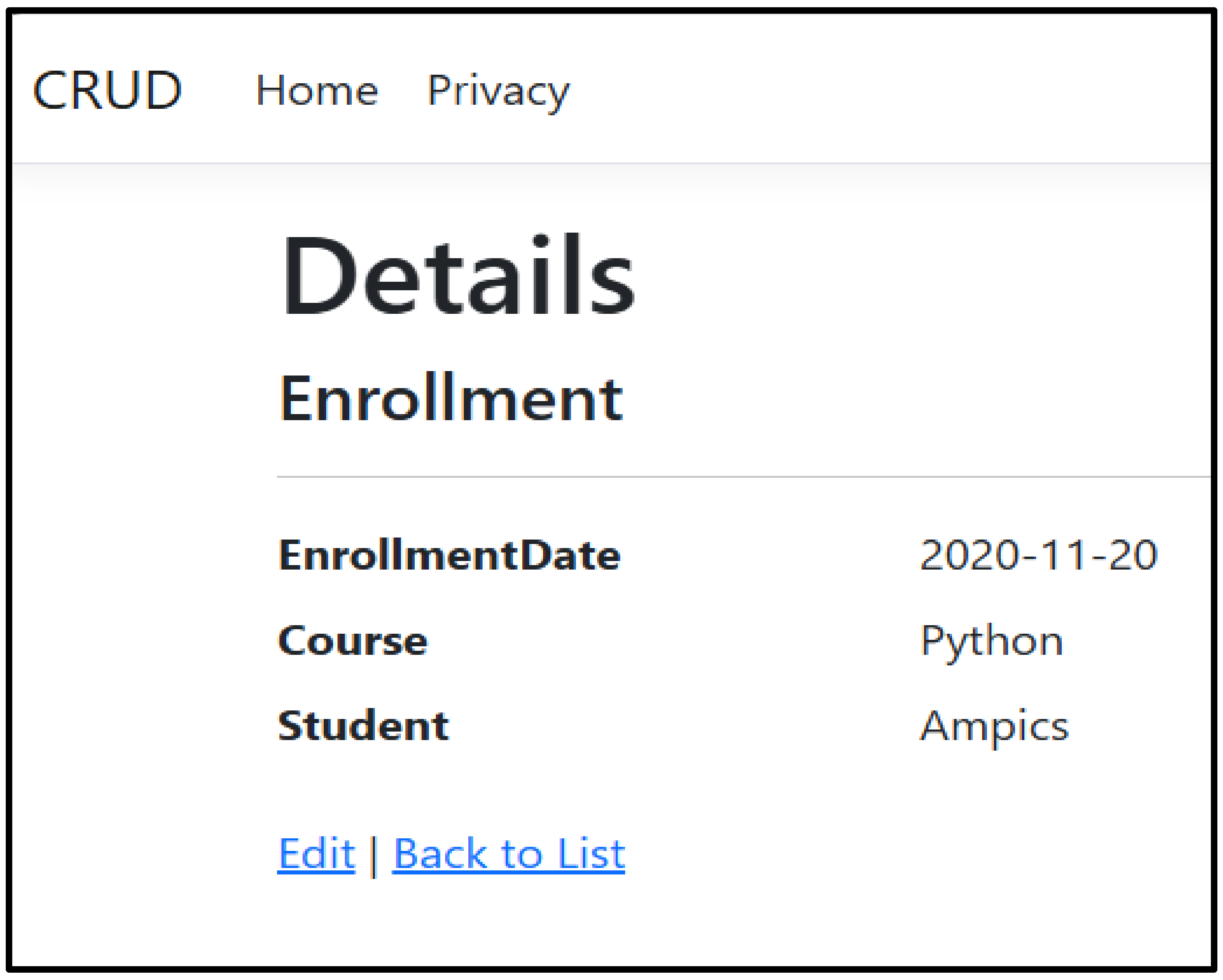
Edit

Enrollment

CourseID: Bussiness Studies

StudentID: Ampics

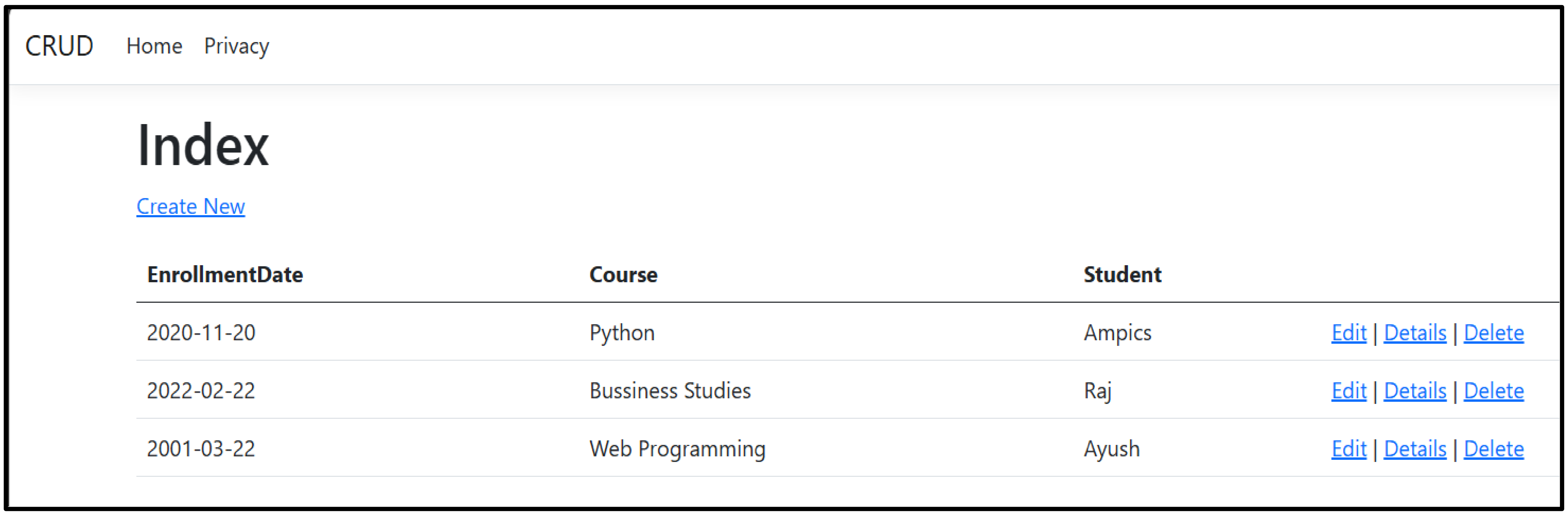
EnrollmentDate: 11/20/2020



Details

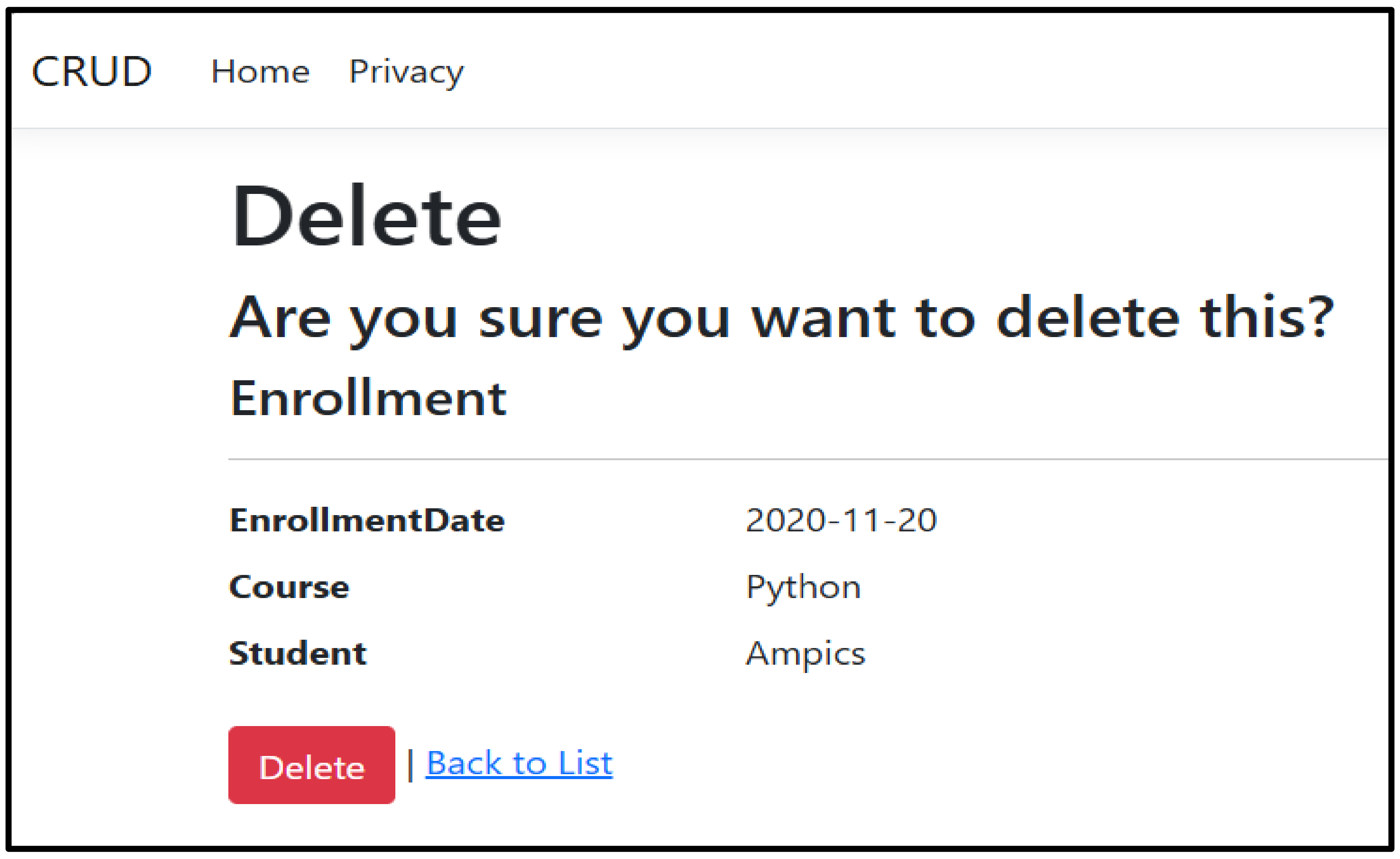
Enrollment

EnrollmentDate	2020-11-20
Course	Python
Student	Ampics



Index

EnrollmentDate	Course	Student	
2020-11-20	Python	Ampics	Edit | Details | Delete
2022-02-22	Bussiness Studies	Raj	Edit | Details | Delete
2001-03-22	Web Programming	Ayush	Edit | Details | Delete



Delete

Are you sure you want to delete this?

Enrollment

EnrollmentDate	2020-11-20
Course	Python
Student	Ampics