

# Assignment - 2

## 1) Create a C# application to demonstrate use of Generic Abstract class.

The screenshot shows the Visual Studio IDE interface. The title bar says "Assignment - 2". The Solution Explorer on the right shows a project named "Assignment-2\_1" with files like Program.cs, IntegerStore.cs, and StringStore.cs. The code editor window displays the following C# code:

```
Assignment-2_1
StringStore.cs IntegerStore.cs DataStore.cs Program.cs
Assignment-2_1
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Assignment_2_1
8  {
9      abstract class DataStore<T>
10     {
11         protected T data;
12
13         public abstract void Save(T value);
14
15         public abstract T Load();
16
17         public void DisplayType()
18         {
19             Console.WriteLine("Data type is : " + typeof(T));
20         }
21     }
22 }
```

The screenshot shows the Visual Studio IDE interface. The title bar says "Assignment - 2". The Solution Explorer on the right shows a project named "Assignment-2\_1" with files like Program.cs, IntegerStore.cs, and StringStore.cs. The code editor window displays the following C# code:

```
Assignment-2_1
StringStore.cs IntegerStore.cs DataStore.cs Program.cs
Assignment-2_1
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Assignment_2_1
8  {
9      class IntegerStore : DataStore<int>
10     {
11         public override void Save(int value)
12         {
13             data = value;
14         }
15
16         public override int Load()
17         {
18             return data;
19         }
20     }
21 }
```

Screenshot of Microsoft Visual Studio showing the code for Assignment\_2\_1.StringStore. The code defines a class StringStore that implements the DataStore<string> interface. It overrides the Save and Load methods to store and retrieve strings respectively.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Assignment_2_1
8  {
9      public class StringStore : DataStore<string>
10     {
11         public override void Save(string value)
12         {
13             data = value;
14         }
15
16         public override string Load()
17         {
18             return data;
19         }
20     }
21 }
```

The Solution Explorer shows the project structure with files Assignment\_2\_1.csproj, Assignment\_2\_1.cs, Program.cs, and StringStore.cs.

Screenshot of Microsoft Visual Studio showing the code for Assignment\_2\_1.Program. The main method creates an IntegerStore and a StringStore, displays their types, and then reads an integer from the console.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Assignment_2_1
8  {
9      internal class Program
10     {
11         static void Main(string[] args)
12         {
13             DataStore<int> intStore = new IntegerStore();
14             intStore.DisplayType();
15             Console.WriteLine("Stored Integer: " + intStore.Load());
16
17             Console.WriteLine();
18
19             DataStore<string> stringStore = new StringStore();
20             stringStore.Save("Hello Generics");
21             stringStore.DisplayType();
22             Console.WriteLine("Stored String: " + stringStore.Load());
23
24             Console.ReadLine();
25         }
26     }
27 }
```

The Solution Explorer shows the project structure with files Assignment\_2\_1.csproj, Assignment\_2\_1.cs, DataStore.cs, IntegerStore.cs, Program.cs, and StringStore.cs.

## DataStore.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Assignment_2_1
{
    abstract class DataStore<T>
    {
        protected T data;

        public abstract void Save(T value);
        public abstract T Load();

        public void DisplayType()
        {
            Console.WriteLine("Data type is : " + typeof(T));
        }
    }
}
```

## IntegerStore.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Assignment_2_1
{
    class IntegerStore : DataStore<int>
    {
        public override void Save(int value)
        {
            data = value;
        }
    }
}
```

```
        public override int Load()
        {
            return data;
        }
    }
}
```

### StringStore.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Assignment_2_1
{
    class StringStore : DataStore<string>
    {
        public override void Save(string value)
        {
            data = value;
        }

        public override string Load()
        {
            return data;
        }
    }
}
```

### Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Assignment_2_1
```

```

{
internal class Program
{
    static void Main(string[] args)
    {
        DataStore<int> intStore = new IntegerStore();
        intStore.Save(100);
        intStore.DisplayType();
        Console.WriteLine("Stored Integer: " + intStore.Load());

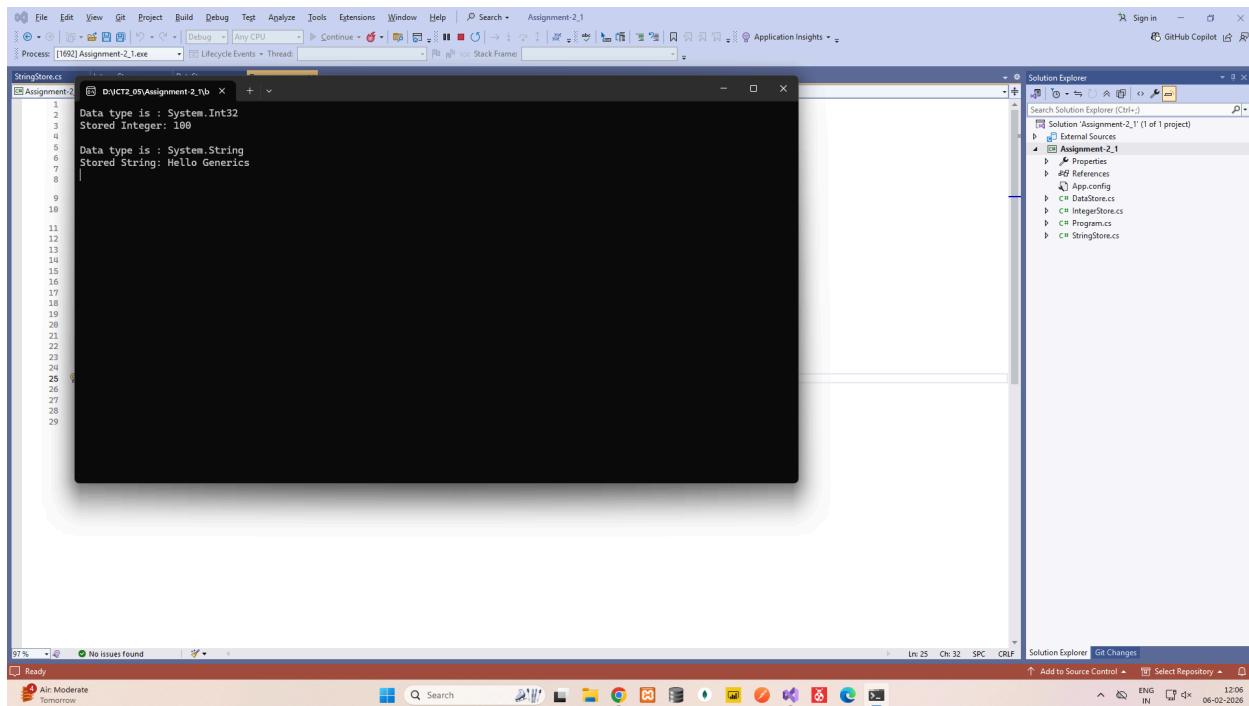
        Console.WriteLine();

        DataStore<string> stringStore = new StringStore();
        stringStore.Save("Hello Generics");
        stringStore.DisplayType();
        Console.WriteLine("Stored String: " + stringStore.Load());

        Console.ReadLine();
    }
}
}

```

## Output :



**2) Create an MVC Core web application to perform CRUD operations for Project allocation to employees using EntityFramework. Also create advance search functionality to search employee on project.**

### Models/Employee.cs

```
using System;
using System.Collections.Generic;

namespace Assignment_2_2.Models;

public partial class Employee
{
    public int EmployeeId { get; set; }

    public string? EmployeeName { get; set; }

    public string? Email { get; set; }

    public string? Department { get; set; }

    public DateOnly? JoiningDate { get; set; }

    public bool? IsActive { get; set; }

    public virtual ICollection<ProjectAllocation> ProjectAllocations { get; set; } = new
List<ProjectAllocation>();
}
```

### Models/Project.cs

```
using System;
using System.Collections.Generic;

namespace Assignment_2_2.Models;

public partial class Project
{
    public int ProjectId { get; set; }
```

```
public string? ProjectName { get; set; }

public string? Description { get; set; }

public DateOnly? StartDate { get; set; }

public DateOnly? EndDate { get; set; }

public string? Status { get; set; }

public virtual ICollection<ProjectAllocation> ProjectAllocations { get; set; } = new
List<ProjectAllocation>();
}
```

### Models/ProjectAllocation.cs

```
using Microsoft.AspNetCore.Mvc.ModelBinding.Validation;
using System;
using System.Collections.Generic;

namespace Assignment_2_2.Models;

public partial class ProjectAllocation
{
    public int AllocationId { get; set; }

    public int EmployeeId { get; set; }

    public int ProjectId { get; set; }

    public DateOnly? AllocationDate { get; set; }

    public string? Role { get; set; }

    public bool? IsActive { get; set; }

    [ValidateNever]
    public virtual Employee Employee { get; set; } = null!;

    [ValidateNever]
    public virtual Project Project { get; set; } = null!;

}
```

## Controllers/EmployeeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using Assignment_2_2.Models;

namespace Assignment_2_2.Controllers
{
    public class EmployeesController : Controller
    {
        private readonly Ict2projectAllocationDbContext _context;

        public EmployeesController(Ict2projectAllocationDbContext context)
        {
            _context = context;
        }

        // GET: Employees
        public async Task<IActionResult> Index()
        {
            return View(await _context.Employees.ToListAsync());
        }

        // GET: Employees/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var employee = await _context.Employees
                .FirstOrDefaultAsync(m => m.EmployeeId == id);
            if (employee == null)
            {
                return NotFound();
            }
        }
    }
}
```

```

        }

        return View(employee);
    }

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

    // POST: Employees/Create
    // To protect from overposting attacks, enable the specific properties you want to bind to.
    // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult>
Create([Bind("EmployeeId,EmployeeName,Email,Department,JoiningDate,IsActive")] Employee
employee)
    {
        if (ModelState.IsValid)
        {
            employee.IsActive = true;
            _context.Add(employee);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
        return View(employee);
    }

    // GET: Employees/Edit/5
    public async Task<IActionResult> Edit(int? id)
    {
        if (id == null)
        {
            return NotFound();
        }

        var employee = await _context.Employees.FindAsync(id);
        if (employee == null)
        {
            return NotFound();
        }
    }
}

```

```

        }

        return View(employee);
    }

// POST: Employees/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id,
[Bind("EmployeeId,EmployeeName,Email,Department,JoiningDate,IsActive")] Employee
employee)
{
    if (id != employee.EmployeeId)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(employee);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!EmployeeExists(employee.EmployeeId))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    return View(employee);
}

// GET: Employees/Delete/5

```

```

public async Task<IActionResult> Delete(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var employee = await _context.Employees
        .FirstOrDefaultAsync(m => m.EmployeeId == id);
    if (employee == null)
    {
        return NotFound();
    }

    return View(employee);
}

// POST: Employees/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var employee = await _context.Employees.FindAsync(id);
    if (employee != null)
    {
        _context.Employees.Remove(employee);
    }

    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}

private bool EmployeeExists(int id)
{
    return _context.Employees.Any(e => e.EmployeeId == id);
}

public IActionResult SearchByProjectName()
{
    return View();
}

```

```

[HttpPost]
public IActionResult SearchByProjectName(string projectName)
{
    var employees = _context.Employees
        .Include(e => e.ProjectAllocations)
        .ThenInclude(pa => pa.Project)
        .Where(e => e.ProjectAllocations.Any(pa =>
            pa.Project.ProjectName.Contains(projectName)))
        .ToList();

    return View("SearchResult", employees);
}

public IActionResult SearchResult(List<Employee> employeeList)
{
    return View(employeeList);
}
}
}

```

## Controllers/ProjectsController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using Assignment_2_2.Models;

namespace Assignment_2_2.Controllers
{
    public class ProjectsController : Controller
    {
        private readonly Ict2projectAllocationDbContext _context;

        public ProjectsController(Ict2projectAllocationDbContext context)
        {

```

```

        _context = context;
    }

    // GET: Projects
    public async Task<IActionResult> Index()
    {
        return View(await _context.Projects.ToListAsync());
    }

    // GET: Projects/Details/5
    public async Task<IActionResult> Details(int? id)
    {
        if (id == null)
        {
            return NotFound();
        }

        var project = await _context.Projects
            .FirstOrDefaultAsync(m => m.ProjectId == id);
        if (project == null)
        {
            return NotFound();
        }

        return View(project);
    }

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

    // POST: Projects/Create
    // To protect from overposting attacks, enable the specific properties you want to
    bind to.
    // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]

```

```

    public async Task<IActionResult>
Create([Bind("ProjectId,ProjectName,Description,StartDate,EndDate,Status")] Project
project)
{
    if (ModelState.IsValid)
    {
        _context.Add(project);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    return View(project);
}

// GET: Projects/Edit/5
public async Task<IActionResult> Edit(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var project = await _context.Projects.FindAsync(id);
    if (project == null)
    {
        return NotFound();
    }
    return View(project);
}

// POST: Projects/Edit/5
// To protect from overposting attacks, enable the specific properties you want to
bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id,
[Bind("ProjectId,ProjectName,Description,StartDate,EndDate,Status")] Project project)
{
    if (id != project.ProjectId)

```

```

    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(project);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!ProjectExists(project.ProjectId))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    return View(project);
}

// GET: Projects/Delete/5
public async Task<IActionResult> Delete(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var project = await _context.Projects
        .FirstOrDefaultAsync(m => m.ProjectId == id);
    if (project == null)
    {

```

```

        return NotFound();
    }

    return View(project);
}

// POST: Projects/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var project = await _context.Projects.FindAsync(id);
    if (project != null)
    {
        _context.Projects.Remove(project);
    }

    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}

private bool ProjectExists(int id)
{
    return _context.Projects.Any(e => e.ProjectId == id);
}
}

```

## Controllers/ProjectAllocations.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using Assignment_2_2.Models;

```

```

namespace Assignment_2_2.Controllers
{
    public class ProjectAllocationsController : Controller
    {
        private readonly Ict2projectAllocationDbContext _context;

        public ProjectAllocationsController(Ict2projectAllocationDbContext context)
        {
            _context = context;
        }

        // GET: ProjectAllocations
        public async Task<IActionResult> Index()
        {
            var ict2projectAllocationDbContext = _context.ProjectAllocations.Include(p =>
p.Employee).Include(p => p.Project);
            return View(await ict2projectAllocationDbContext.ToListAsync());
        }

        // GET: ProjectAllocations/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var projectAllocation = await _context.ProjectAllocations
                .Include(p => p.Employee)
                .Include(p => p.Project)
                .FirstOrDefaultAsync(m => m.AllocationId == id);
            if (projectAllocation == null)
            {
                return NotFound();
            }

            return View(projectAllocation);
        }
    }
}

```

```

// GET: ProjectAllocations/Create
public IActionResult Create()
{
    ViewData["EmployeeId"] = new SelectList(_context.Employees, "EmployeeId",
"EmployeeName");
    ViewData["ProjectId"] = new SelectList(_context.Projects, "ProjectId",
"ProjectName");
    return View();
}

// POST: ProjectAllocations/Create
// To protect from overposting attacks, enable the specific properties you want to
bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult>
Create([Bind("AllocationId,EmployeeId,ProjectId,AllocationDate,Role,IsActive")]
ProjectAllocation projectAllocation)
{
    if (ModelState.IsValid)
    {
        projectAllocation.IsActive = true;
        _context.Add(projectAllocation);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    ViewData["EmployeeId"] = new SelectList(_context.Employees, "EmployeeId",
"EmployeeName", projectAllocation.EmployeeId);
    ViewData["ProjectId"] = new SelectList(_context.Projects, "ProjectId",
"ProjectName", projectAllocation.ProjectId);
    return View(projectAllocation);
}

// GET: ProjectAllocations/Edit/5
public async Task<IActionResult> Edit(int? id)
{
    if (id == null)
    {

```

```

        return NotFound();
    }

    var projectAllocation = await _context.ProjectAllocations.FindAsync(id);
    if (projectAllocation == null)
    {
        return NotFound();
    }
    ViewData["EmployeeId"] = new SelectList(_context.Employees, "EmployeeId",
"EmployeeName", projectAllocation.EmployeeId);
    ViewData["ProjectId"] = new SelectList(_context.Projects, "ProjectId",
"ProjectName", projectAllocation.ProjectId);
    return View(projectAllocation);
}

// POST: ProjectAllocations/Edit/5
// To protect from overposting attacks, enable the specific properties you want to
bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id,
[Bind("AllocationId,EmployeeId,ProjectId,AllocationDate,Role,IsActive")]
ProjectAllocation projectAllocation)
{
    if (id != projectAllocation.AllocationId)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(projectAllocation);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {

```

```

        if (!ProjectAllocationExists(projectAllocation.AllocationId))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }
    return RedirectToAction(nameof(Index));
}

ViewData["EmployeeId"] = new SelectList(_context.Employees, "EmployeeId",
"EmployeeName", projectAllocation.EmployeeId);
ViewData["ProjectId"] = new SelectList(_context.Projects, "ProjectId",
"ProjectName", projectAllocation.ProjectId);
return View(projectAllocation);
}

// GET: ProjectAllocations/Delete/5
public async Task<IActionResult> Delete(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var projectAllocation = await _context.ProjectAllocations
        .Include(p => p.Employee)
        .Include(p => p.Project)
        .FirstOrDefaultAsync(m => m.AllocationId == id);
    if (projectAllocation == null)
    {
        return NotFound();
    }

    return View(projectAllocation);
}

// POST: ProjectAllocations/Delete/5

```

```

[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var projectAllocation = await _context.ProjectAllocations.FindAsync(id);
    if (projectAllocation != null)
    {
        _context.ProjectAllocations.Remove(projectAllocation);
    }

    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}

private bool ProjectAllocationExists(int id)
{
    return _context.ProjectAllocations.Any(e => e.AllocationId == id);
}
}

```

### SearchByProjectName.cshtml

```

<form asp-action="SearchByProjectName">
    <div class="form-group">
        <label for="productName">Project Name:</label>
        <input type="text" class="form-control" id="productName" name="productName"
required />
    </div>
    <button type="submit" class="btn btn-primary">Search</button>
</form>

```

### SearchResult.html

```
@model IEnumerable<Assignment_2_2.Models.Employee>
```

```
@{
```

```

ViewData["Title"] = "Search Result";
Layout = "~/Views/Shared/_Layout.cshtml";
}

<h1>Search Result</h1>

<p>
    <a asp-action="Create">Create New Employee</a>
</p>

<table class="table">
    <thead>
        <tr>
            <th>@Html.DisplayNameFor(model => model.First().EmployeeName)</th>
            <th>@Html.DisplayNameFor(model => model.First().Email)</th>
            <th>@Html.DisplayNameFor(model => model.First().Department)</th>
            <th>@Html.DisplayNameFor(model => model.First().JoiningDate)</th>
            <th>Projects</th>
            <th></th>
        </tr>
    </thead>
    <tbody>
        @foreach (var employee in Model)
        {
            <tr>
                <td>@Html.DisplayFor(m => employee.EmployeeName)</td>
                <td>@Html.DisplayFor(m => employee.Email)</td>
                <td>@Html.DisplayFor(m => employee.Department)</td>
                <td>@Html.DisplayFor(m => employee.JoiningDate)</td>
                <td>
                    @foreach(var alloc in employee.ProjectAllocations)
                    {
                        @alloc.Project.ProjectName <br />
                    }
                </td>
                <td>
                    <a asp-action="Edit" asp-route-id="@employee.EmployeeId">Edit</a> |
                    <a asp-action="Details" asp-route-id="@employee.EmployeeId">Details</a> |
                    <a asp-action="Delete" asp-route-id="@employee.EmployeeId">Delete</a>
                </td>
            </tr>
        }
    </tbody>
</table>

```

```

        </td>
    </tr>
}
</tbody>
</table>
```

## Shared/\_Layout.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Assignment_2_2</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="~/Assignment_2_2.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">Assignment_2_2</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="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-controller="Employees"
asp-action="Index">Employees</a>
    </li>
    <li class="nav-item">
        <a class="nav-link text-dark" asp-controller="Projects"
asp-action="Index">Projects</a>
    </li>
    <li class="nav-item">
        <a class="nav-link text-dark" asp-controller="ProjectAllocations"
asp-action="Index">Project Allocations</a>
    </li>
</ul>
<form class="d-flex" asp-controller="Employees"
asp-action="SearchByProjectName" method="post">
    <input class="form-control me-2" type="text" name=" projectName"
placeholder="Search by Project" />
    <button class="btn btn-outline-success" type="submit">Search</button>
</form>

        </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; 2026 - Assignment_2_2 - <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>
```

## Appsettings.json

```
{  
  "Logging": {  
    "LogLevel": {  
      "Default": "Information",  
      "Microsoft.AspNetCore": "Warning"  
    }  
  },  
  "AllowedHosts": "*",  
  "ConnectionStrings": {  
    "ProjectAllocateString": "Data Source=DESKTOP-E6KULOA;Initial  
Catalog=ICT2ProjectAllocationDB;Integrated Security=True;Trust Server  
Certificate=True"  
  }  
}
```

## Output

The screenshot shows a web browser window titled 'Create - Assignment\_2\_2'. The URL in the address bar is 'localhost:7006/Employees/Create'. The page content is a 'Create' form for an 'Employee'. It includes fields for 'EmployeeName' (Dhrushiya Bhalia), 'Email' (dhrushi@gmail.com), 'Department' (Engineer), and 'JoiningDate' (2025-02-06). The 'JoiningDate' field is highlighted with a blue border. Below the form is a 'Create' button and a 'Back to List' link. At the bottom of the page, there is a copyright notice: '© 2026 - Assignment\_2\_2 - [Privacy](#)'.



Index - Assignment\_2\_2

localhost:7005/Employees

Assignment\_2\_2 Home Employees Projects Project Allocations

Search by Project Search

## Index

Create New

EmployeeName	Email	Department	JoiningDate	IsActive	
Rahul Sharma	rahul@gmail.com	IT	17-08-2024	False	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Anita Verma	anita@gmail.com	HR	23-06-2025	True	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Karan Patel	karan@gmail.com	Finance	13-09-2025	True	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Dhruvisha Bhaliya	dhrushi@gmail.com	Engineer	06-02-2025	True	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

© 2026 - Assignment\_2\_2 - [Privacy](#)



Details - Assignment\_2\_2

localhost:7005/Projects/Details/3

Assignment\_2\_2 Home Employees Projects Project Allocations

Search by Project Search

## Details

### Project

ProjectName	Description
CRM Upgrade	Upgrade customer relationship management system to v2.0

[Edit](#) | [Back to List](#)

© 2026 - Assignment\_2\_2 - [Privacy](#)



Edit - Assignment\_2\_2

localhost:7005/ProjectAllocations/Edit/1002

Assignment\_2\_2 Home Employees Projects Project Allocations

Search by Project Search

## Edit

### ProjectAllocation

EmployeeId	Anita Verma
ProjectId	CRM Upgrade
AllocationDate	12-09-2026
Role	Tester
IsActive	<input type="checkbox"/> True <input checked="" type="checkbox"/> True <input type="checkbox"/> False

© 2026 - Assignment\_2\_2 - [Privacy](#)



Index - Assignment\_2\_2

localhost:7005/ProjectAllocations

Assignment\_2\_2 Home Employees Projects Project Allocations

Search by Project Search

## Index

[Create New](#)

AllocationDate	Role	IsActive	Employee	Project	
12-09-2026	Developer	<input type="checkbox"/> False	1	1	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
12-09-2026	Tester	<input checked="" type="checkbox"/> True	2	3	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

© 2026 - Assignment\_2\_2 - [Privacy](#)



The screenshot shows a web browser window titled "Search Result - Assignment\_2\_2". The URL is "localhost:7005/Employees/SearchByProjectName". The page header includes "Assignment\_2\_2 Home Employees Projects Project Allocations" and "Payroll System | Search". Below the header is a table titled "Search Result" with columns: EmployeeName, Email, Department, JoiningDate, and Projects. Two rows of data are listed:

EmployeeName	Email	Department	JoiningDate	Projects
Rahul Sharma	rahul@gmail.com	IT	17-08-2024	Payroll System <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Dhruvisha Bhaliya	dhrushi@gmail.com	Engineer	06-02-2025	Payroll System <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

At the bottom left is a copyright notice: "© 2026 - Assignment\_2\_2 - Privacy".



### 3) Demonstrate use of String Indexer and multivalued Indexer to search data from a Generic collection.

The screenshot shows the Visual Studio IDE interface. The title bar says "Assignment\_2\_3". The main area displays the following C# code for a generic collection:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Assignment_2_3
{
    class Book
    {
        public string ISBN { get; set; }
        public string Title { get; set; }
        public string Author { get; set; }
        public int Year { get; set; }

        public Book(string ISBN, string title, string author, int year)
        {
            ISBN = ISBN;
            Title = title;
            Author = author;
            Year = year;
        }
    }
}

```

The Solution Explorer on the right shows the project structure with files: Assignment\_2\_3.csproj, Properties, App.config, Book.cs, Library.cs, and Program.cs.

This screenshot shows the Microsoft Visual Studio IDE interface. The title bar reads "Assignment\_2\_3". The left pane displays the "Server Explorer" and "Solution Explorer" windows, which show a single project named "Assignment\_2\_3" containing files like App.config, Book.cs, Library.cs, and Program.cs. The right pane shows the "Toolbox" and "Properties" windows. The main code editor window contains the following C# code for the Book.cs class:

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Assignment_2_3
8  {
9      class Library<T> where T : Book
10     {
11         private List<T> books = new List<T>();
12
13         public void AddBook(T book)
14         {
15             books.Add(book);
16         }
17
18         public T this[string title]
19         {
20             get
21             {
22                 foreach (T book in books)
23                 {
24                     if (book.Title == title)
25                         return book;
26                 }
27                 return null;
28             }
29         }
30
31         public T this[string author, int year]
32         {
33             get
34             {
35                 foreach (T book in books)
36                 {
37                     if (book.Author == author && book.Year == year)
38                         return book;
39                 }
40                 return null;
41             }
42         }
43     }
44 }
45 
```

This screenshot shows the Microsoft Visual Studio IDE interface. The title bar reads "Assignment\_2\_3". The left pane displays the "Server Explorer" and "Solution Explorer" windows, which show a single project named "Assignment\_2\_3" containing files like App.config, Book.cs, Library.cs, and Program.cs. The right pane shows the "Toolbox" and "Properties" windows. The main code editor window contains the following C# code for the Program.cs file:

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Assignment_2_3
8  {
9      internal class Program
10     {
11         static void Main(string[] args)
12         {
13             Library<Book> library = new Library<Book>();
14
15             library.AddBook(new Book("101", "C# Basics", "John", 2000));
16             library.AddBook(new Book("102", "ASP.NET Core", "Smith", 2021));
17             library.AddBook(new Book("103", "Data Structures", "John", 2022));
18
19             Book b1 = library["C# Basics"];
20             Console.WriteLine("Search by Title:");
21             Console.WriteLine($"{b1.ISBN} {b1.Title} {b1.Author} {b1.Year}");
22
23             Console.WriteLine();
24
25             Book b2 = library["John", 2022];
26             Console.WriteLine("Search by Author and Year:");
27             Console.WriteLine($"{b2.ISBN} {b2.Title} {b2.Author} {b2.Year}");
28
29             Console.ReadLine();
30         }
31     }
32 }
```

## Book.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Assignment_2_3
{
    class Book
    {
        public string ISBN { get; set; }

        public string Title { get; set; }

        public string Author { get; set; }

        public int Year { get; set; }

        public Book(string iISBN, string title, string author, int year)
        {
            ISBN = iISBN;
            Title = title;
            Author = author;
            Year = year;
        }
    }
}
```

## Library.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Assignment_2_3
{
    class Library<T> where T : Book
    {

        private List<T> books = new List<T>();

        public void AddBook(T book)
        {
            books.Add(book);
        }

        public T this[string title]
        {
            get
            {
                foreach (T book in books)
                {
                    if (book.Title == title)
                        return book;
                }
                return null;
            }
        }

        public T this[string author, int year]
        {
            get
            {
                foreach (T book in books)
                {
                    if (book.Author == author && book.Year == year)
                        return book;
                }
                return null;
            }
        }
    }
}
```

## Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Assignment_2_3
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Library<Book> library = new Library<Book>();

            library.AddBook(new Book("101", "C# Basics", "John", 2020));
            library.AddBook(new Book("102", "ASP.NET Core", "Smith", 2021));
            library.AddBook(new Book("103", "Data Structures", "John", 2022));

            Book b1 = library["C# Basics"];
            Console.WriteLine("Search by Title:");
            Console.WriteLine($"{b1.ISBN} {b1.Title} {b1.Author} {b1.Year}");

            Console.WriteLine();

            Book b2 = library["John", 2022];
            Console.WriteLine("Search by Author and Year:");
            Console.WriteLine($"{b2.ISBN} {b2.Title} {b2.Author} {b2.Year}");

            Console.ReadLine();
        }
    }
}
```

## Output

The screenshot shows a Microsoft Visual Studio interface. On the left, a terminal window displays the following text:

```
Search by Title:  
101 C# Basics John 2020  
Search by Author and Year:  
103 Data Structures John 2022
```

To the right of the terminal is the Solution Explorer, which lists the project structure:

- External Sources
- Assignment-2\_3 (1 of 1 project)
  - Properties
  - References
    - App.config
    - C# Book.cs
    - C# Library.cs
    - C# Program.cs