```
// Program.cs - Entry Point and Minimal API Setup
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Components;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using System.Collections.Generic;
using System.Linq;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddRazorPages();
builder.Services.AddServerSideBlazor();
builder.Services.AddSingleton<TaskService>();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");
}

app.UseStaticFiles();
app.UseRouting();
app.MapBlazorHub();
app.MapFallbackToPage("/_Host");
app.Run();

// TaskService.cs - Task Data Service
public class TaskService
{
    private List<TaskItem> tasks = new List<TaskItem>();

    public List<TaskItem> GetTasks() => tasks;
    public void AddTask(TaskItem task) => tasks.Add(task);
}

public class TaskItem
{
    public string Title { get; set; }
    public bool IsComplete { get; set; }
}

// Index.razor - Blazor UI Component
@page "/"
@inject TaskService TaskService

<h3>Task List</h3>
<ul>
    @foreach (var task in tasks)
    {
        <li>@task.Title - @(task.IsComplete ? "Done" : "Pending")</li>
```

```
    }
</ul>

<input @bind="newTaskTitle" placeholder="Add new task" />
<button @onclick="AddTask">Add Task</button>

@code {
    private List<TaskItem> tasks;
    private string newTaskTitle;

    protected override void OnInitialized()
    {
        tasks = TaskService.GetTasks();
    }

    private void AddTask()
    {
        if (!string.IsNullOrWhiteSpace(newTaskTitle))
        {
             TaskService.AddTask(new TaskItem { Title = newTaskTitle, IsComplete = false
});
            newTaskTitle = string.Empty;
        }
    }
}
```