Create a Book model with properties like BookId, Title, Author, and AvailableCopies. Create a User model with properties like UserId, Name, and Email. Implement GET /books, POST /books, GET /users, and POST /users endpoints. Implement POST /borrow/{bookId} to allow a user to borrow a book and decrement the available copies

Step 2: Folder Structure

Your folder structure should look like this:

```
markdown
Copy code
/BookBorrowingSystem
    /Controllers
    /Models
    /Services
    /Program.cs
    /Startup.cs
```

Step 3: Create the Models

1. Book Model

1. **Right-click the `Models` folder** > **Add** > **Class**.
   - Name it `Book.cs`.
2. **Define the `Book` model**:

   Book.cs:

```
csharp
Copy code
namespace BookBorrowingSystem.Models
{
    public class Book
    {
        public int BookId { get; set; }
        public string Title { get; set; }
        public string Author { get; set; }
        public int AvailableCopies { get; set; }
    }
}
```

2. User Model

1. **Right-click the `Models` folder** > **Add** > **Class**.
   - Name it `User.cs`.
2. **Define the `User` model**:

   User.cs:

```
csharp
Copy code
namespace BookBorrowingSystem.Models
{
    public class User
    {
        public int UserId { get; set; }
        public string Name { get; set; }
        public string Email { get; set; }
    }
}
```

Step 4: Create the Service to Manage Books and Users

1. **Right-click the `Services` folder** > **Add** > **New Folder** and name it `Services`.
2. **Create a service class for book and user management**:
   - Right-click the `Services` folder > **Add** > **Class**.
   - Name it `LibraryService.cs`.
3. **Define the `LibraryService` class**:

   LibraryService.cs:

```
csharp
Copy code
using BookBorrowingSystem.Models;
using System.Collections.Generic;
using System.Linq;

namespace BookBorrowingSystem.Services
{
    public class LibraryService
    {
        private readonly List<Book> _books = new();
        private readonly List<User> _users = new();

        // Get all books
        public IEnumerable<Book> GetBooks()
        {
            return _books;
        }

        // Add a new book
        public void AddBook(Book book)
        {
            _books.Add(book);
        }

        // Get all users
        public IEnumerable<User> GetUsers()
        {
            return _users;
        }

        // Add a new user
        public void AddUser(User user)
        {
            _users.Add(user);
        }
        // Borrow a book
        public bool BorrowBook(int bookId, int userId)
        {
            var book = _books.FirstOrDefault(b => b.BookId ==
bookId);
            if (book == null || book.AvailableCopies <= 0)
            {
                return false;
            }

            book.AvailableCopies--;
            return true;
        }
    }
}
```

Step 5: Create the Controller for API Endpoints

1. **Right-click the `Controllers` folder** > **Add** > **Controller**.
   - Choose **API Controller - Empty**.
   - Name it `LibraryController`.
2. **Define the `LibraryController`** with endpoints for books, users, and borrowing:

   LibraryController.cs:

```
csharp
Copy code
using BookBorrowingSystem.Models;
using BookBorrowingSystem.Services;
using Microsoft.AspNetCore.Mvc;

namespace BookBorrowingSystem.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class LibraryController : ControllerBase
    {
        private readonly LibraryService _libraryService;

        public LibraryController(LibraryService libraryService)
        {
            _libraryService = libraryService;
        }

        // GET /books
        [HttpGet("books")]
        public IActionResult GetBooks()
        {
            var books = _libraryService.GetBooks();
            return Ok(books);
        }

        // POST /books
        [HttpPost("books")]
        public IActionResult AddBook([FromBody] Book book)
        {
            if (book == null)
            {
                return BadRequest("Book is null.");
            }

            _libraryService.AddBook(book);
            return CreatedAtAction(nameof(GetBooks), new { id =
book.BookId }, book);
        }

        // GET /users
        [HttpGet("users")]
        public IActionResult GetUsers()
        {
            var users = _libraryService.GetUsers();
            return Ok(users);
        }

        // POST /users
        [HttpPost("users")]
        public IActionResult AddUser([FromBody] User user)
        {
            if (user == null)
            {
                return BadRequest("User is null.");
            }

            _libraryService.AddUser(user);
            return CreatedAtAction(nameof(GetUsers), new { id =
user.UserId }, user);
        }

        // POST /borrow/{bookId}
        [HttpPost("borrow/{bookId}")]
        public IActionResult BorrowBook(int bookId, [FromBody] int
userId)
        {
            var result = _libraryService.BorrowBook(bookId, userId);
            if (result)
            {
                return Ok("Book borrowed successfully.");
            }
            else
            {
                return NotFound("Book not available or invalid
book.");
            }
        }
    }
}
```

Step 6: Register the Service in `Program.cs`

1. Open **`Program.cs`**.
2. Register `LibraryService` with the dependency injection container:

   Program.cs:

   ```
   csharp
   ```

```
Copy code
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllers();
builder.Services.AddSingleton<LibraryService>(); // Register
LibraryService

var app = builder.Build();

// Configure the HTTP request pipeline.
app.UseHttpsRedirection();
app.UseAuthorization();
app.MapControllers();

app.Run();
```

Step 7: Testing the Application

1. Run the Application

Press **F5** or **Ctrl + F5** to run the application. This will start the API and you can test the endpoints via Swagger (if enabled), Postman, or any other REST client.

2. Test the API Endpoints

- **GET /api/library/books**: Get all books in the system.
  - Example response:

    ```
    json
    Copy code
    [
        {
            "bookId": 1,
            "title": "The Great Gatsby",
            "author": "F. Scott Fitzgerald",
            "availableCopies": 5
        },
        {
            "bookId": 2,
            "title": "1984",
            "author": "George Orwell",
            "availableCopies": 3
        }
    ]
    ```

- **POST /api/library/books**: Add a new book.
  - Request body:

    ```
    json
    Copy code
    {
        "bookId": 3,
        "title": "Moby Dick",
        "author": "Herman Melville",
        "availableCopies": 2
    ```

```
}
```

- **GET /api/library/users**: Get all users in the system.
  - Example response:

```
json
Copy code
[
    {
        "userId": 1,
        "name": "John Doe",
        "email": "john.doe@example.com"
    }
]
```

- **POST /api/library/users**: Add a new user.
  - Request body:

```
json
Copy code
{
    "userId": 2,
    "name": "Jane Doe",
    "email": "jane.doe@example.com"
}
```

- **POST /api/library/borrow/{bookId}**: Borrow a book.
  - Request body:

```
json
Copy code
2  // UserId
```

    If successful, the available copies of the book will be decremented, and the response will be:

```
json
Copy code
"Book borrowed successfully."
```

Create ASP.Net MVC Web application for Job Portal with Master Page and minimum 4 Pages

Step 2: Define Folder Structure

Your folder structure will look like this:

```
bash
Copy code
/JobPortal
    /Controllers
    /Models
    /Views
```

```
        /Shared
        /Jobs
        /Users
    /wwwroot
    /Views/_Layout.cshtml (Master Page)
    /appsettings.json
    /Program.cs
    /Startup.cs
```

Step 3: Create the Models

1. **Right-click the `Models` folder** > **Add** > **Class**.
   - Name it `Job.cs`.

Job.cs:

```
csharp
Copy code
namespace JobPortal.Models
{
    public class Job
    {
        public int JobId { get; set; }
        public string JobTitle { get; set; }
        public string Company { get; set; }
        public string Location { get; set; }
        public string Description { get; set; }
        public decimal Salary { get; set; }
        public DateTime PostedDate { get; set; }
    }
}
```

2. **Add another model for users**:
   - Right-click the `Models` folder > **Add** > **Class**.
   - Name it `User.cs`.

User.cs:

```
csharp
Copy code
namespace JobPortal.Models
{
    public class User
    {
        public int UserId { get; set; }
        public string Name { get; set; }
        public string Email { get; set; }
        public string Role { get; set; } // Employer or Job Seeker
    }
}
```

Step 4: Create the Controllers

1. **Right-click the `Controllers` folder** > **Add** > **Controller**.
   - Choose **MVC Controller - Empty**.
   - Name it `JobController`.

JobController.cs:

```
csharp
Copy code
using JobPortal.Models;
using Microsoft.AspNetCore.Mvc;

namespace JobPortal.Controllers
{
    public class JobController : Controller
    {
        private static List<Job> jobs = new List<Job>
        {
            new Job { JobId = 1, JobTitle = "Software Developer", Company =
"TechCorp", Location = "New York", Description = "Develop software
solutions.", Salary = 60000, PostedDate = DateTime.Now.AddDays(-1) },
            new Job { JobId = 2, JobTitle = "Data Analyst", Company =
"DataInc", Location = "California", Description = "Analyze data and
generate reports.", Salary = 50000, PostedDate = DateTime.Now.AddDays(-2)
},
        };

        public IActionResult Index()
        {
            return View(jobs);
        }

        public IActionResult Details(int id)
        {
            var job = jobs.FirstOrDefault(j => j.JobId == id);
            if (job == null)
                return NotFound();
            return View(job);
        }

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

        [HttpPost]
        public IActionResult Create(Job job)
        {
            if (ModelState.IsValid)
            {
                job.JobId = jobs.Max(j => j.JobId) + 1;
                job.PostedDate = DateTime.Now;
                jobs.Add(job);
                return RedirectToAction("Index");
            }
            return View(job);
        }
    }
}
```

2. **Create another controller for Users**:
   - Right-click the `Controllers` folder > **Add** > **Controller**.
   - Choose **MVC Controller - Empty**.
   - Name it `UserController`.

UserController.cs:

```
csharp
Copy code
using JobPortal.Models;
using Microsoft.AspNetCore.Mvc;

namespace JobPortal.Controllers
{
    public class UserController : Controller
    {
        private static List<User> users = new List<User>
        {
            new User { UserId = 1, Name = "John Doe", Email =
"john.doe@example.com", Role = "Job Seeker" },
            new User { UserId = 2, Name = "Jane Smith", Email =
"jane.smith@example.com", Role = "Employer" },
        };

        public IActionResult Index()
        {
            return View(users);
        }

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

        [HttpPost]
        public IActionResult Create(User user)
        {
            if (ModelState.IsValid)
            {
                user.UserId = users.Max(u => u.UserId) + 1;
                users.Add(user);
                return RedirectToAction("Index");
            }
            return View(user);
        }
    }
}
```

Step 5: Create the Views

1. **Create Views for Jobs**:
    - Right-click the `Views` folder > **Add** > **New Folder** > name it `Jobs`.
    - Inside the `Jobs` folder, create the following views:
        - **Index.cshtml** (List of all jobs)
        - **Details.cshtml** (Job details page)
        - **Create.cshtml** (Form to create new job)

Index.cshtml (Jobs/Index.cshtml):

```
html
Copy code
@model IEnumerable<JobPortal.Models.Job>
```

```
<h2>Job Listings</h2>
<table class="table">
    <thead>
        <tr>
            <th>Job Title</th>
            <th>Company</th>
            <th>Location</th>
            <th>Salary</th>
            <th>Posted Date</th>
            <th>Action</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var job in Model)
        {
            <tr>
                <td>@job.JobTitle</td>
                <td>@job.Company</td>
                <td>@job.Location</td>
                <td>@job.Salary</td>
                <td>@job.PostedDate.ToShortDateString()</td>
                <td>
                    <a href="@Url.Action("Details", "Job", new { id =
job.JobId })">Details</a>
                </td>
            </tr>
        }
    </tbody>
</table>
<a href="@Url.Action("Create", "Job")">Create New Job</a>
```

Details.cshtml (Jobs/Details.cshtml):

```
html
Copy code
@model JobPortal.Models.Job

<h2>@Model.JobTitle</h2>
<p><strong>Company:</strong> @Model.Company</p>
<p><strong>Location:</strong> @Model.Location</p>
<p><strong>Salary:</strong> $@Model.Salary</p>
<p><strong>Description:</strong> @Model.Description</p>
<p><strong>Posted Date:</strong> @Model.PostedDate.ToShortDateString()</p>
<a href="@Url.Action("Index", "Job")">Back to Job Listings</a>
```

Create.cshtml (Jobs/Create.cshtml):

```
html
Copy code
@model JobPortal.Models.Job

<h2>Create New Job</h2>

<form method="post">
    <div>
        <label>Job Title</label>
        <input type="text" name="JobTitle" required />
    </div>
    <div>
        <label>Company</label>
        <input type="text" name="Company" required />
    </div>
    <div>
        <label>Location</label>
        <input type="text" name="Location" required />
    </div>
    <div>
        <label>Salary</label>
        <input type="number" name="Salary" required />
    </div>
    <div>
        <label>Description</label>
        <textarea name="Description"></textarea>
    </div>
    <button type="submit">Create Job</button>
</form>
```

2. **Create Views for Users**:
   - Right-click the `Views` folder > **Add** > **New Folder** > name it `Users`.
   - Inside the `Users` folder, create the following views:
     - **Index.cshtml** (List of all users)
     - **Create.cshtml** (Form to create new user)

Index.cshtml (Users/Index.cshtml):

```
html
Copy code
@model IEnumerable<JobPortal.Models.User>

<h2>User List</h2>
<table class="table">
    <thead>
        <tr>
            <th>Name</th>
            <th>Email</th>
            <th>Role</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var user in Model)
        {
            <tr>
                <td>@user.Name</td>
                <td>@user.Email</td>
                <td>@user.Role</td>
            </tr>
        }
    </tbody>
</table>
<a href="@Url.Action("Create", "User")">Create New User</a>
```

Create.cshtml (Users/Create.cshtml):

```
html
Copy code
@model JobPortal.Models.User

<h2>Create New User</h2>
```

```
<form method="post">
    <div>
        <label>Name</label>
        <input type="text" name="Name" required />
    </div>
    <div>
        <label>Email</label>
        <input type="email" name="Email" required />
    </div>
    <div>
        <label>Role</label>
        <select name="Role">
            <option value="Job Seeker">Job Seeker</option>
            <option value="Employer">Employer</option>
        </select>
    </div>
    <button type="submit">Create User</button>
</form>
```

Step 6: Layout (Master Page)

1. In **`Views/Shared/_Layout.cshtml`**, add the basic structure of the layout page.

```
html
Copy code
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
    <title>Job Portal</title>
</head>
<body>
    <header>
        <nav>
            <ul>
                <li><a href="@Url.Action("Index", "Job")">Jobs</a></li>
                <li><a href="@Url.Action("Index", "User")">Users</a></li>
            </ul>
        </nav>
    </header>

    <div>
        @RenderBody()
    </div>
</body>
</html>
```

Step 7: Configure `Program.cs`

Ensure that your `Program.cs` is set to use MVC.

```
csharp
Copy code
var builder = WebApplication.CreateBuilder(args);
```

```
builder.Services.AddControllersWithViews();

var app = builder.Build();

if (app.Environment.IsDevelopment())
{
    app.UseDeveloperExceptionPage();
}
else
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Job}/{action=Index}/{id?}");

app.Run();
```