

Cognizant Academy **Novel Hub**

**ASP.Net Core MVC, Entity
Framework Core,**

**SQLServer – Integrated
Capability Test**

Table of Contents

1.0 Introduction	3
1.1 Purpose of this document.....	3
1.2 Definitions & Acronyms.....	3
1.3 Project Overview	3
1.4 Use case Diagram	4
1.5 Scope	4
1.6 Target Audience	4
1.7 Hardware and Software Requirement.....	4
1.7.1 Hardware Requirements.....	4
1.7.2 Software Requirements.....	4
2.0 System Diagram	5
3.0 Architecture	5
4.0 Solution Creation	6
5.0 Create Branch Details.....	8
5.1 Requirement Flow	8
5.2 Technical Guidelines	10
5.2.1 Implementing POCO/Entity class.....	10
Implementing BranchDetails.cs.....	10
6.0 Create Book Details.....	11
6.1 Requirement Flow	11
6.2 Technical Guidelines	14
6.2.1 Implementing POCO/Entity class.....	14
Implementing BookDetails.cs.....	14
7.0 Create Librarian Details.....	16
7.1 Requirement Flow	16
7.2 Technical Guidelines	19
7.2.1 Implementing POCO/Entity class.....	19
Implementing Librarian.cs	19
8.0 Create Members Details	21
8.1 Requirement Flow	21
8.2 Technical guideline	23
8.2.1 Implementing POCO/Entity class.....	23
Implementing Members.cs	23
9.0 Data Context Implementation.....	24
10.0 Controllers and Views Implementation.....	25
11.0 Evaluation Areas	28

1.0 Introduction

1.1 Purpose of this document

The code is intended to facilitate managing a library system through a web application interface. It allows users to perform various tasks such as creating and viewing details related to branches, books, librarians, and memberships.

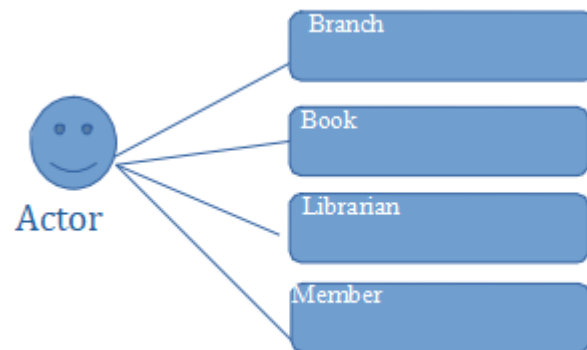
1.2 Definitions & Acronyms

Definition / Acronym	Description
C#	C# (pronounced C sharp) is an object-oriented server-side programming language for developing .NET application
SQL Server	SQL Server is a powerful relational database for storing data
Asp.NET Core MVC	Light weight framework for developing server-side application which provides separation of concerns for developing applications using asp.net core
Entity framework Core	Provides an ORM to map a relational model with the object oriented model.

1.3 Project Overview

Novel Hub is One of the Library Management System. This Website is used to maintaining the records of Books, librarian, Members details

1.4 Use case Diagram



1.5 Scope

The system encompasses modules for managing branches, books, librarians, memberships, and data persistence, facilitating comprehensive library administration and user engagement.

1.6 Target Audience

Advance Level

1.7 Hardware and Software Requirement

1.7.1 Hardware Requirements

#	Item	Specification/Version
1.	PC	8GB RAM

1.7.2 Software Requirements

#	Item	Specification/Version
1.	.NET Framework	6.0
2.	Visual Studio Professional	2022

3	SQLSERVER Enterprise	2014
4	Internet Explorer/Google Chrome/Firefox	-

2.0 System Diagram

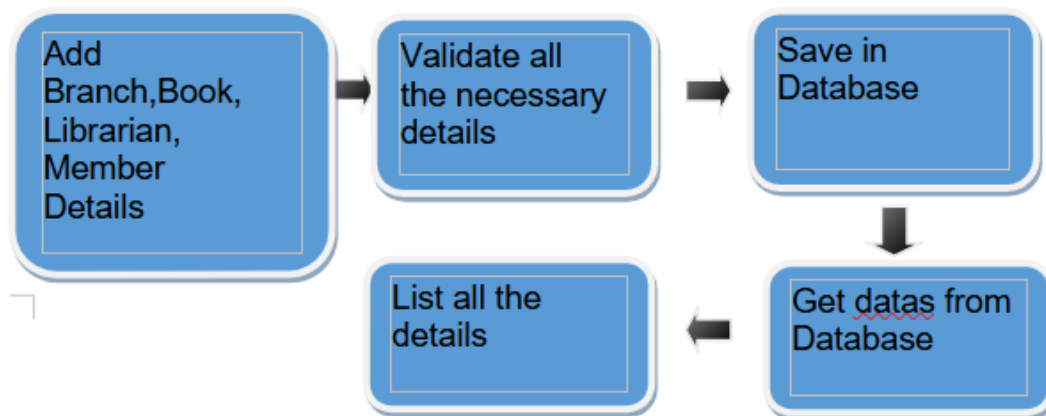


Figure: System Diagram

3.0 Architecture

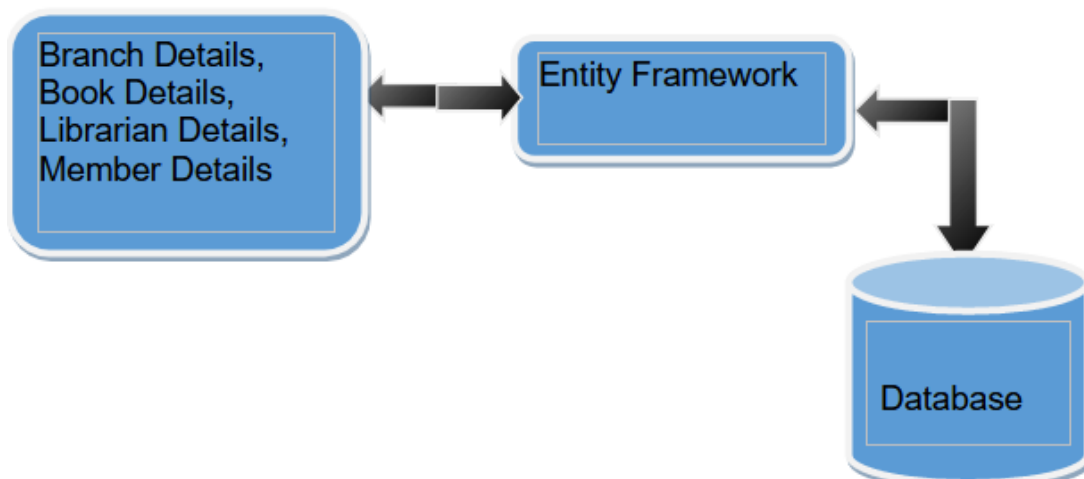


Figure: Architecture Diagram

4.0 Solution Creation

1. Create a new project and choose **ASP.NET Core Web App (Model-View-Controller) (C#)**.
2. Give the project name as "**Novel Hub**" and click the Next button.
3. Choose the version **.NET 6.0**.
4. Change the authentication mode to "No Authentication."
5. Ensure to uncheck the following checkboxes
 1. Configure for HTTPS.
 2. Docker support.
 3. Create unit.
6. Go to NuGet Package Manager.
7. Download the below-mentioned packages:
 1. **Microsoft.EntityFrameworkCore.**
 2. **Microsoft.EntityFrameworkCore.SqlServer.**
 3. **Microsoft.EntityFrameworkCore.Tools.**
8. Create the below-mentioned models:
 1. BranchDetails.
 2. BookDetails.
 3. Librarian.
 4. Members.
9. Models should relate to constraints. Branch' primary key, "branchId," should be used as a foreign key in the other three models.
10. Create Dependency Injection and register all table names on the Dependency Injection page.
11. Configure the connection in "**Appsettings.json.**"
 1. Give a database name as LYMS.
12. Register the connection string in "**Program.cs.**"
13. Migrate the model to create tables in the database.
14. Create a controller named "**LibraryController.**"
15. Create views for the UI.

```

<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Microsoft.EntityFrameworkCore" Version="6.0.29" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="6.0.29" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="6.0.29">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
    </PackageReference>
    <PackageReference Include="Microsoft.VisualStudio.Web.CodeGeneration.Design" Version="6.0.16" />
  </ItemGroup>
</Project>

```

Figure: LibraryManagementSystem.csproj

1. Go to solution explorer, Program.cs - in that set the pattern as per the below snapshot

```

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

```

Figure: Program.cs Pattern

NOTE:

- A. Solution is already created for you on the platform. Do not make any changes to connection string in **appsettings.json** file on the platform
- B. After creating all 4 models, Follow the below-given instructions to create migrations
- C. For Creating Migration, Use this below code in the Package Manager Console

1. add-migration SampleName- once this migration is successful, then move to next step

2. update-database

You can give any name instead of **SampleName**

5.0 Create Branch Details

5.1 Requirement Flow

Steps Explanation:

- 1 User launches the application and index.cshtml page is displayed to the user as follows

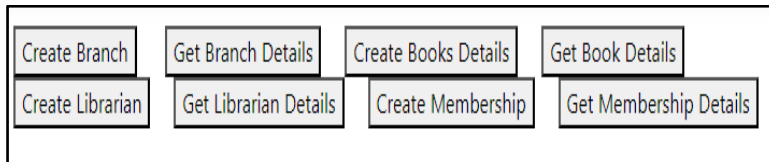


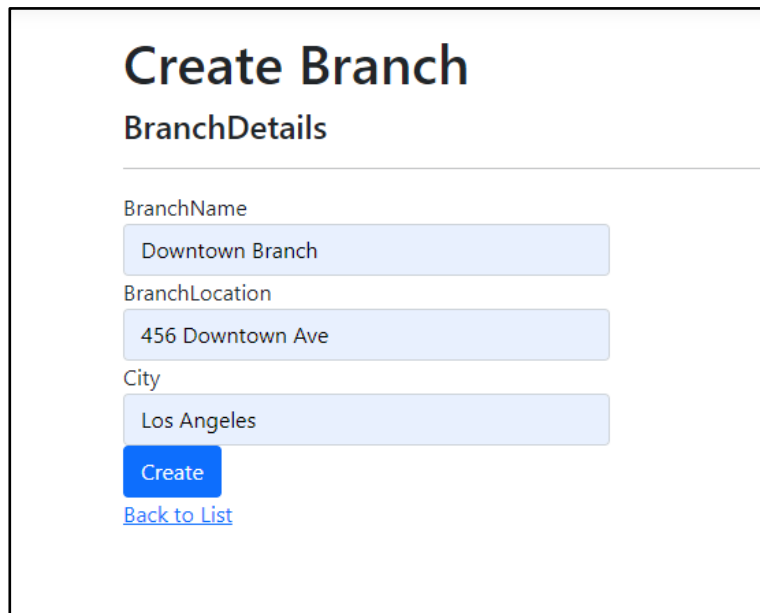
Figure : Index page

- 2 When the user clicks the "Create Branch" button, the Create.cshtml page is displayed to the user as follows.

A screenshot of the 'Create Branch' form. The form has a title 'Create Branch' and a subtitle 'BranchDetails'. Below the subtitle, there are three input fields labeled 'BranchName', 'BranchLocation', and 'City'. At the bottom of the form, there is a blue 'Create' button and a blue link labeled 'Back to List'.

Figure: Create Branch form

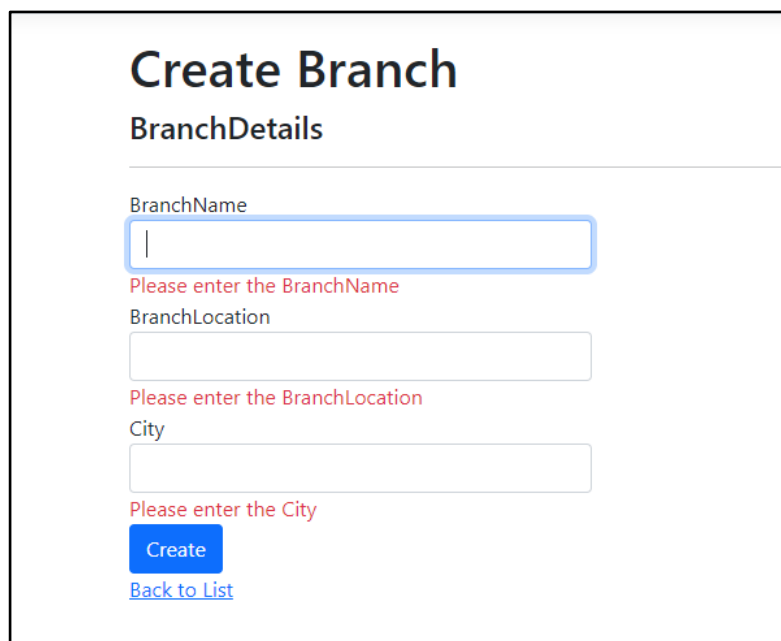
- 3 Users can create a new branch as follows:



The image shows a web form titled "Create Branch" with a sub-header "BranchDetails". It contains three text input fields: "BranchName" with the value "Downtown Branch", "BranchLocation" with the value "456 Downtown Ave", and "City" with the value "Los Angeles". Below the fields are a blue "Create" button and a blue link "Back to List".

Figure: Create Branch form

- 4 If there is any validation failures application will display appropriate validation message as follows



The image shows the same "Create Branch" form, but with validation errors. The "BranchName" field has a red border and a red message "Please enter the BranchName" below it. The "BranchLocation" field has a red message "Please enter the BranchLocation" below it. The "City" field has a red message "Please enter the City" below it. The "Create" button and "Back to List" link are still present.

Figure: Create Branch form validation

- 5 User will fill up all the required details and click the create button which will save the Division details into the database and displays a message to user as follows

LibraryManagementSystem [Home](#) [Privacy](#)

Create Branch

BranchDetails

Branch Details Added Successfully

BranchName

BranchLocation

City

[Create](#)

[Back to List](#)

Figure: Branch Details Added

GetBranchDetails

[Create New](#)

BranchName	BranchLocation	City
Main Branch	123 Main St	New York
Downtown Branch	456 Downtown Ave	Los Angeles

Figure: Branch Details Retrieved

5.2 Technical Guidelines

5.2.1 Implementing POCO/Entity class

Implementing BranchDetails.cs

1. Create a new class in the “Models” folder with the name as “**BranchDetails**” With the following specification.

Table: BranchDetails

Property Name	Type	Modifier
BranchId	int	public

BranchName	string	public
BranchLocation	string	public
City	string	public

2. BranchDetails entity class will be used as POCO class for generating the database table and also for creating strongly typed views for the actions method.
3. Modify the “**BranchDetails**” class with appropriate DataAnnotations to match the following validation rules

Property	Validation	Error Message
BranchId	Key	
BranchName	Must not be blank	Please enter the BranchName
BranchLocation	Must not be blank	Please enter the BranchLocation
City	Must not be blank	Please enter the City

Table : BranchDetails class validations specifications

6.0 Create Book Details

6.1 Requirement Flow

Steps Explanation :

1. User launches the application and index.cshtml page is displayed to the user as follows

Create Branch	Get Branch Details	Create Books Details	Get Book Details
Create Librarian	Get Librarian Details	Create Membership	Get Membership Details

Figure : Index page

2. When the user clicks the "Create Books Details" button, the **"CreateBooksDetails.cshtml"** page is displayed to the user as follows:

CreateBooksDetails

BookDetails

Title

Author

Genre

TotalCopies

AvailableCopies

BranchId

[Back to List](#)

Figure : CreateBooksDetails page

3. The user can add new books as follow

CreateBooksDetails

BookDetails

Title
To Kill a Mockingbird

Author
Harper Lee

Genre
Fiction

TotalCopies
0

AvailableCopies
8

BranchId
-- Select Any Option --
-- Select Any Option --
Main Branch
Downtown Branch

Figure: CreateBooksDetails form

4. If there is any validation failures application will display appropriate validation message as follows

CreateBooksDetails

BookDetails

Title
|
Please enter the Title

Author
|
Please enter the Author

Genre
|
Please enter the Genre

TotalCopies
|
Please enter the TotalCopies

AvailableCopies
|
Please enter the AvailableCopies

BranchId
-- Select Any Option --
The BranchId field is required.

Create

[Back to List](#)

Figure : CreateBookDetails form validation

5. User will fill up all the required details and click the create button which will save the Book details into the database and displays a message to user as follows

CreateBooksDetails

BookDetails

Book Details Added Successfully

Title

Author

Genre

TotalCopies

AvailableCopies

BranchId

Figure : BookDetails Added

Title	Author	Genre	TotalCopies	AvailableCopies	Branch
The Great Gatsby	F. Scott Fitzgerald	Classic	10	5	Main Branch
To Kill a Mockingbird	Harper Lee	Fiction	8	0	Downtown Branch

Figure : BookDetails Retrieved

6.2 Technical Guidelines

6.2.1 Implementing POCO/Entity class

Implementing BookDetails.cs

1. Create a new class in the “Models” folder with the name as “**BookDetails**” with the following specification.

Table : BookDetails class

Property Name	Type	Modifier
Id	int	public
Title	string	public

Author	string	public
Genre	string	public
TotalCopies	int	public
AvailableCopies	int	public
BranchId	int	public
Branch	BranchDetails?	public
Branches	SelectList?	public

2. **BookDetails** entity class will be used as POCO class for generating the database table and also for creating strongly-typed views for the actions method.
3. Modify the “**BookDetails**” class with appropriate DataAnnotations to match the following validation rules

Table : BookDetails class validations specifications

Property	Validation	Error Message
Id	Key	
Title	Must not be blank	Please enter the Title
Author	Must not be blank	Please enter the Author
Genre	Must not be left blank	Please enter the Genre
TotalCopies	Must not be left blank	Please enter the TotalCopies
AvailableCopies	Must not be blank	Please enter the AvailableCopies

BranchId	Foreign key From Branches Table	
Branch	Add the NotMapped Attribute	
Branches	Add the NotMapped Attribute	

7.0 Create Librarian Details

7.1 Requirement Flow

Steps Explanation :

1. User launches the application and index.cshtml page is displayed to the user as follows

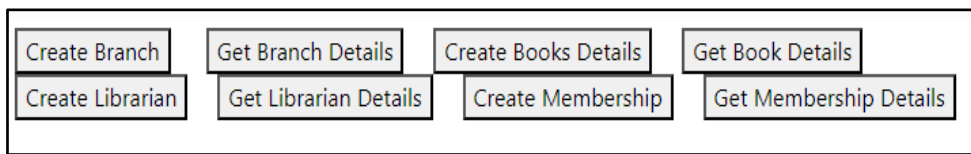
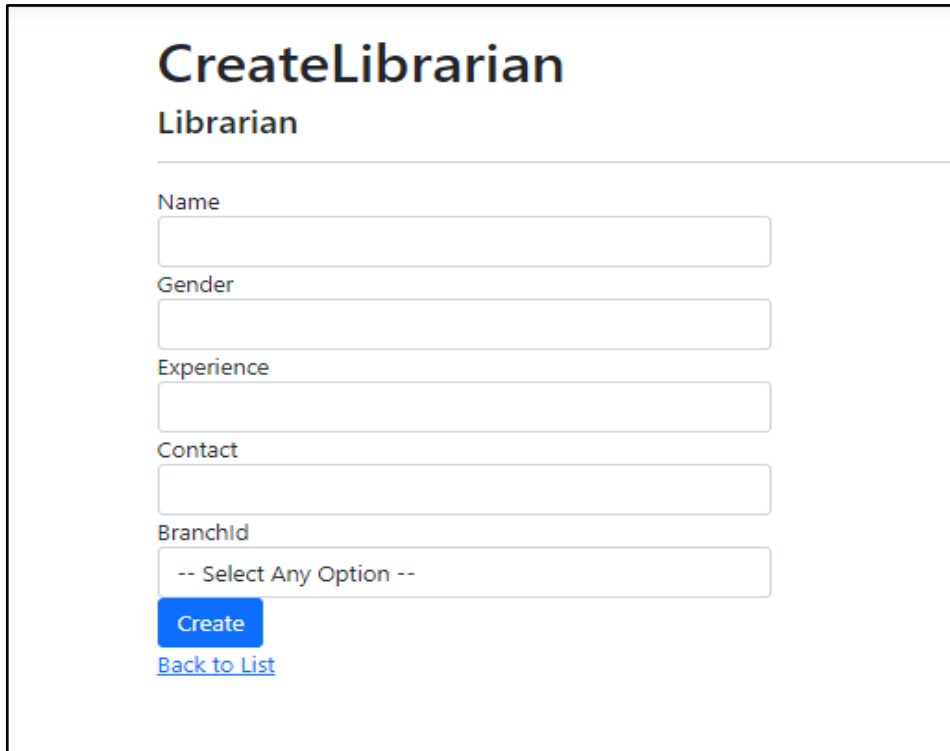


Figure : Index page

2. When the user clicks the "Create Librarian" button, the CreateLibrarian.cshtml page is displayed to the user as follows:

A screenshot of a web form titled "Create Librarian". The form has a header section with the title and a sub-header "Librarian". Below this, there are five input fields: "Name", "Gender", "Experience", "Contact", and "BranchId". The "BranchId" field is a dropdown menu with the text "-- Select Any Option --". Below the input fields, there is a blue "Create" button and a blue link labeled "Back to List".

Create Librarian

Librarian

Name

Gender

Experience

Contact

BranchId

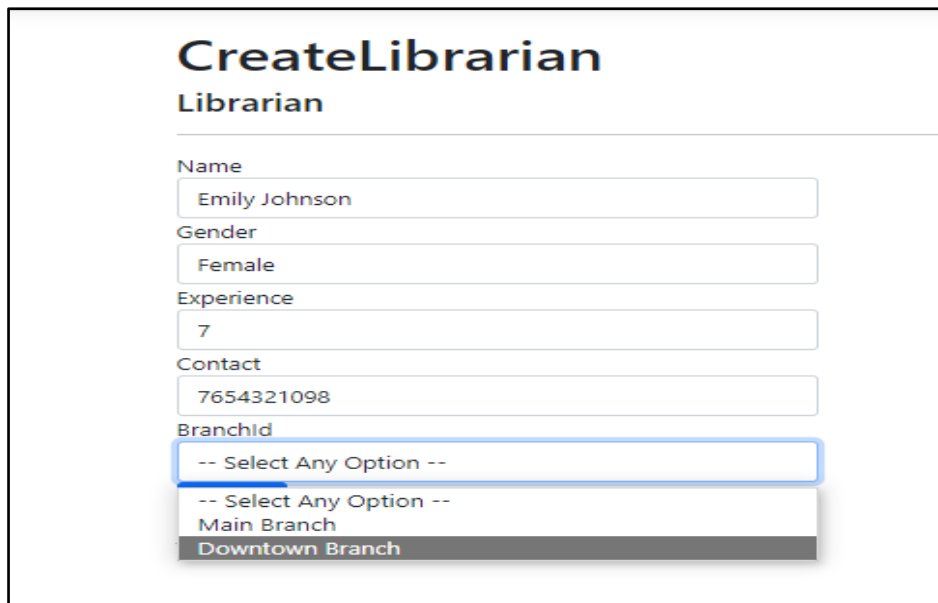
-- Select Any Option --

Create

[Back to List](#)

Figure : Create Librarian form

3. The user can add new librarian details as follows:

A screenshot of the "Create Librarian" form with sample data entered. The "Name" field contains "Emily Johnson", "Gender" contains "Female", "Experience" contains "7", and "Contact" contains "7654321098". The "BranchId" dropdown menu is open, showing three options: "-- Select Any Option --", "Main Branch", and "Downtown Branch".

Create Librarian

Librarian

Name

Emily Johnson

Gender

Female

Experience

7

Contact

7654321098

BranchId

-- Select Any Option --

-- Select Any Option --

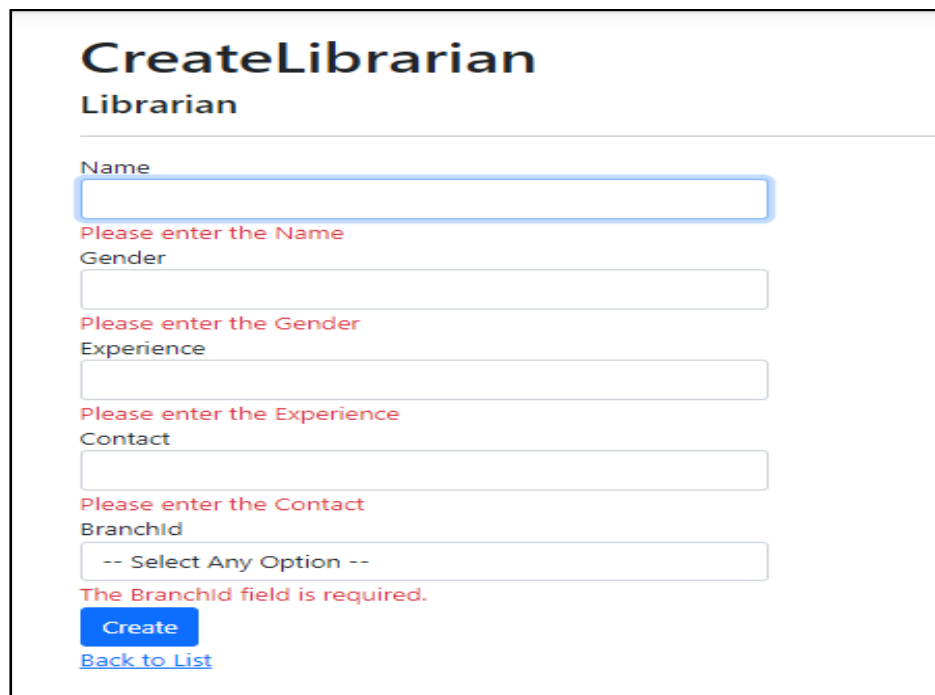
Main Branch

Downtown Branch

Figure: Create Librarian form

4. If there is any validation failures application will display appropriate

validation message as follows



Create Librarian
Librarian

Name

Please enter the Name

Gender

Please enter the Gender

Experience

Please enter the Experience

Contact

Please enter the Contact

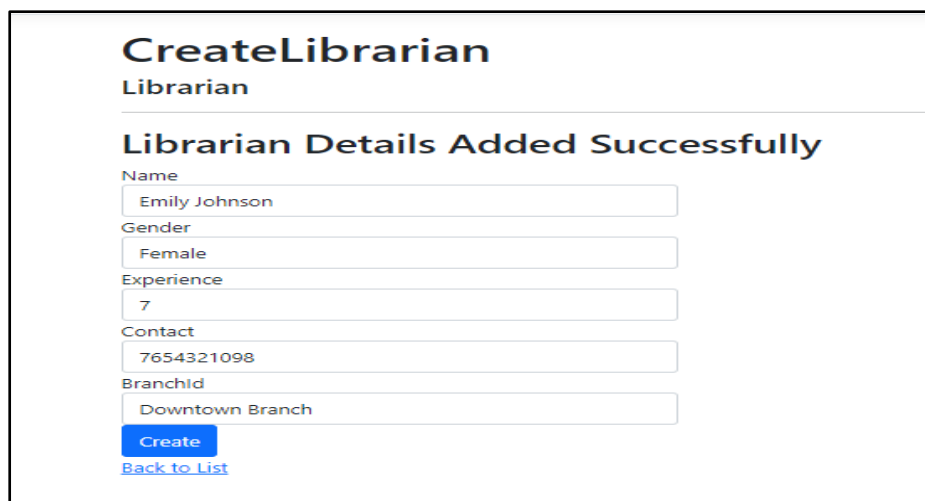
BranchId

The BranchId field is required.

[Create](#)
[Back to List](#)

Figure : Create Librarian form validation

5. User will fill up all the required details and click the create button which will save the Librarian details into the database and displays a message to user as follows



Create Librarian
Librarian

Librarian Details Added Successfully

Name

Gender

Experience

Contact

BranchId

[Create](#)
[Back to List](#)

Figure : Librarian Details Added

GetLibrarianDetails				
Create New				
Name	Gender	Experience	Contact	Branch
John Doe	Male	5	9876543210	Main Branch
Michael Smith	Male	3	8765432109	Main Branch
Emily Johnson	Female	7	7654321098	Downtown Branch

Figure : Librarian Details Retrieved

7.2 Technical Guidelines

7.2.1 Implementing POCO/Entity class

Implementing Librarian.cs

1. Create a new class in the “Models” folder with the name as “**Librarian**” with the following specification.

Table : Librarian class

Property Name	Type	Modifier
Id	int	public
Name	string	public
Gender	string	public
Experience	int	public
Contact	string	public
BranchId	int	public
Branch	BranchDetails?	public

Branches	SelectList?	public

2. Stipend entity class will be used as POCO class for generating the database table and also for creating strongly-typed views for the actions method.

3. Modify the “**Librarian**” class with appropriate DataAnnotations to match the following validation rules

Table : Librarian class validations specifications

Property	Validation	Error Message
Id	Key	
Name	Must not be blank	Please enter the Name
Gender	Must not be blank	Please enter the Gender
Experience	Must not be blank	Please enter the Experience
Contact	Must not be blank	Please enter the Contact
BranchId	Must not be blank	
Branch	Add the NotMapped Attribute	
Branches	Add the NotMapped Attribute	

8.0 Create Members Details

8.1 Requirement Flow

Steps Explanation :

1. User launches the application and index.cshtml page is displayed to the user as follows

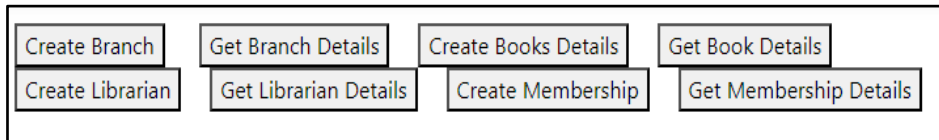


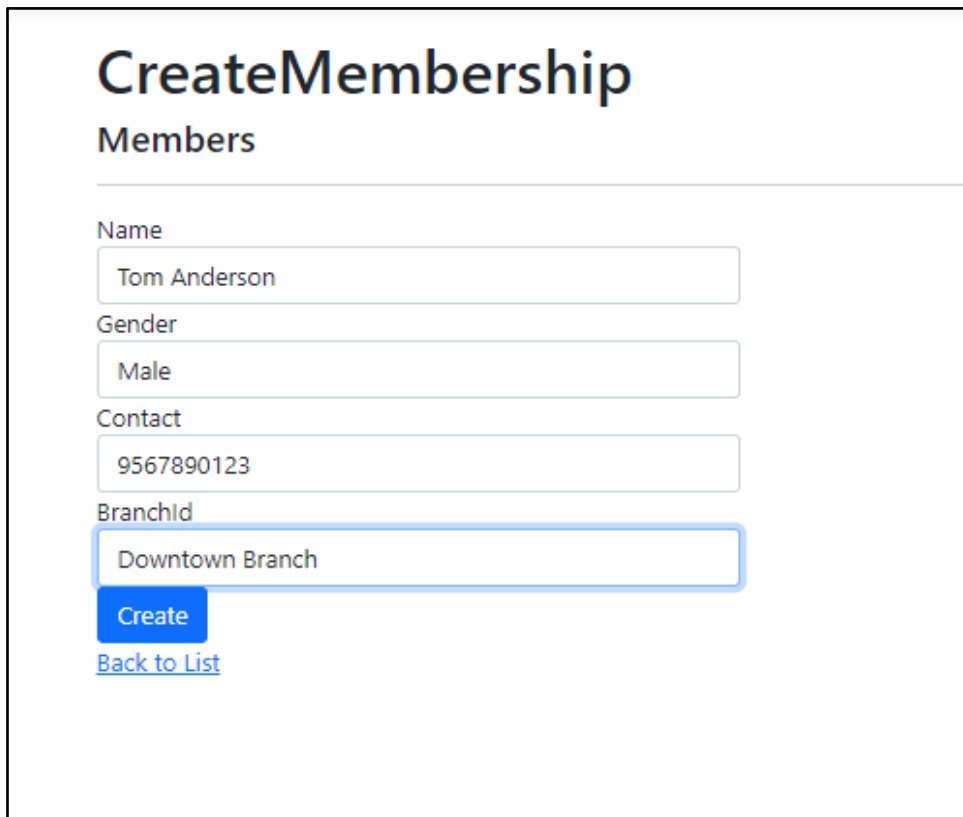
Figure : Index.cshtml

2. When the user clicks the "Create Membership" button, the CreateMembership.cshtml page is displayed to the user as follows:

The screenshot shows a web form titled 'CreateMembership' with a subtitle 'Members'. Below the title is a horizontal line. The form contains the following elements: a 'Name' label followed by a text input field; a 'Gender' label followed by a text input field; a 'Contact' label followed by a text input field; a 'BranchId' label followed by a dropdown menu showing '-- Select Any Option --'; a blue 'Create' button; and a blue link labeled 'Back to List'.

Figure : Membership Details Added form

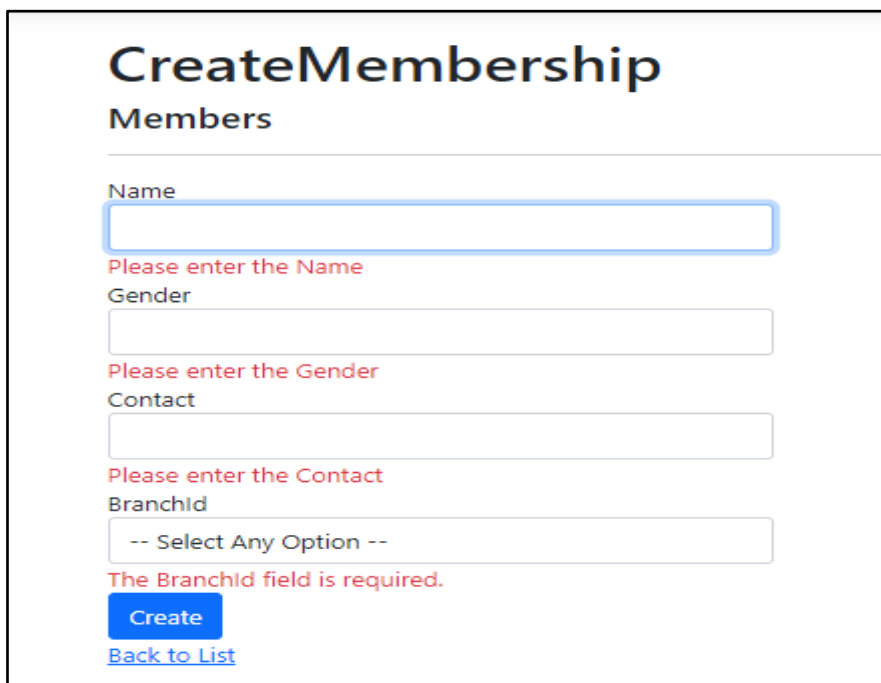
3. The user can add new member details as follows:



The screenshot shows a web form titled "CreateMembership" with a subtitle "Members". The form contains five input fields: "Name" with the value "Tom Anderson", "Gender" with the value "Male", "Contact" with the value "9567890123", and "BranchId" with the value "Downtown Branch". Below the "BranchId" field is a blue "Create" button and a blue link labeled "Back to List".

Figure : Membership Details Added form

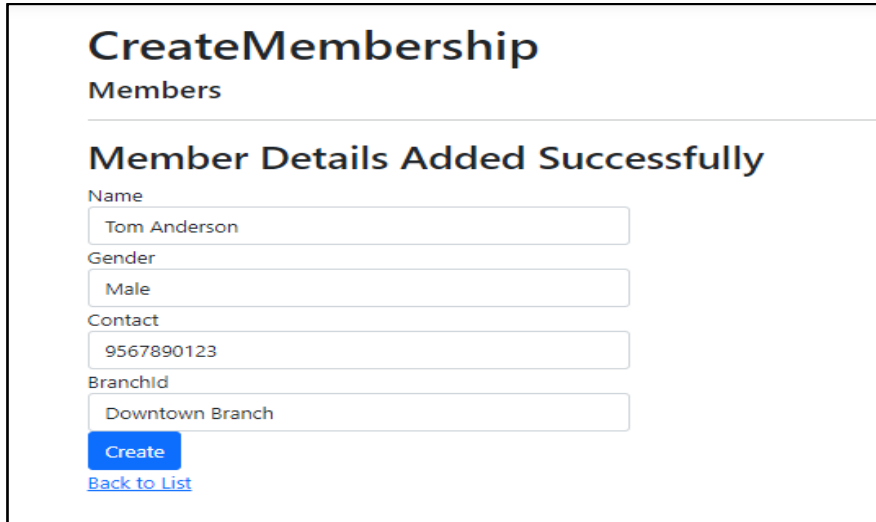
4. If there is any validation failures application will display appropriate validation message as follows



The screenshot shows the same "CreateMembership" form, but with validation errors. The "Name" field is empty and has a red error message "Please enter the Name" below it. The "Gender" field is empty and has a red error message "Please enter the Gender" below it. The "Contact" field is empty and has a red error message "Please enter the Contact" below it. The "BranchId" field is empty and has a red error message "The BranchId field is required." below it. The "Create" button and "Back to List" link are still present.

Figure : Membership Details Added form with Validation

5. **"Member Details Added Successfully"** message visible when user submits valid input.



The screenshot shows a web form titled "CreateMembership" with a sub-header "Members". A success message "Member Details Added Successfully" is displayed. Below the message, the form fields are populated with the following values: Name: Tom Anderson, Gender: Male, Contact: 9567890123, and BranchId: Downtown Branch. At the bottom of the form, there is a blue "Create" button and a blue link labeled "Back to List".

Figure: Membership Details Added

8.1 Technical guideline

8.2.1 Implementing POCO/Entity class

Implementing Members.cs

1. Create a new class in the "Models" folder with the name as **"Members"** with the following specification.

Table: Members class

Property Name	Type	Modifier
Id	int	public
Name	string	public
Gender	string	public
Contact	string	public
BranchId	int	public

Branch	BranchDetails?	public
Branches	SelectList?	public

- Members entity class will be used as POCO class for generating the database table and for creating strongly typed views for the actions method.
- Modify the “Members” class with appropriate DataAnnotations to match the following validation rules

Table: Members class validations specifications

Property	Validation	Error Message
Id	Key	
Name	Must not be blank	Please enter the Name
Gender	Must not be blank	Please enter the Gender
Contact	Must not be blank	Please enter the Contact
BranchId	Must not be blank	
Branch	Foreign key From Branches Table	
Branches	Add the NotMapped Attribute	

9.0 Data Context Implementation

- Create a new folder and name it as Data and create a new file named “**DatabaseContext**” which inherits from the “**DbContext**” class.

2. Modify the **DatabaseContext** to add following details

Table : DatabaseContext class

Property Name	Type	Modifier
BranchesDetails	DbSet<BranchDetails>	public
BooksDetails	DbSet<BookDetails>	public
LibrarianDetails	DbSet<Librarian>	public
MembersDetails	DbSet<Members>	public

10.0 Controllers and Views Implementation

1. Add a new controller named **“HomeController”** in the controllers folder with the following specification.

Table : HomeController Actions

Action Name	Input Parameters	Http Request Type	Modifier	Return Type
Index()	-	Get	public	ActionResult

2. Go to Home/Index.cshtml set title as **“Index”**.
 - A. In Index.cshtml view, add a hyperlink or button to LibraryController “Create” action with id=“ **InkCrBranch**” attribute
 - B. In Index.cshtml view, add a hyperlink or button to LibraryController “GetBranchDetails” action with id=“ **InkGetBranch**” attribute
 - C. In Index.cshtml view, add a hyperlink or button to LibraryController “CreateBooksDetails” action with id=“ **InkCrBook**” attribute
 - D. In Index.cshtml view, add a hyperlink or button to LibraryController “GetBookDetails” action with id=“ **InkGetBook**” attribute

- E.** In Index.cshtml view, add a hyperlink or button to LibraryController “CreateLibrarian” action with id=“**InkCrLibrarian**” attribute
- F.** In Index.cshtml view, add a hyperlink or button to LibraryController “GetLibrarianDetails” action with id=“**InkGetLibrarian**” attribute
- G.** In Index.cshtml view, add a hyperlink or button to LibraryController “CreateMembership” action with id=“**InkCrMember**” attribute
- H.** In Index.cshtml view, add a hyperlink or button to LibraryController “GetMembershipDetails” action with id=“**InkGetMember**” attribute

3. Add a new controller named “LibraryController” in the controllers folder with the following specification.

Table : LibraryController Constructors

Constructor Type	Input Parameters	Modifier
Parameterized	_dbContext	public

Table : LibraryController Actions

Action Name	Input Parameters	Http Request Type	Modifier	Return Type
Create()	BranchDetails	Post	public	ActionResult
GetBranchDetails()		Get	public	ActionResult
CreateBooksDetails()	BookDetails	Post	public	ActionResult
GetBookDetails()		Get	public	ActionResult
CreateLibrarian()	Librarian	Post	public	ActionResult
GetLibrarianDetails()		Get	public	ActionResult
CreateMembership()	Members	Post	public	ActionResult
GetMembershipDetails()		Get	public	ActionResult

4. Implement the constructor
5. Implement the CreateBranch(), CreateBooksDetails() and CreateLibrarian()
,CreateMembership () action methods with [HttpGet].
 - a. Use Scaffold the view using Create template.
 - b. For CreateBranch-Set an ID attribute on the submit button with the value "**BranchCreationBtn**"
 - c. For CreateBooksDetails - Set an ID attribute on the submit button with the value "**BookCreationBtn**"
 - d. For CreateLibrarian - Set an ID attribute on the submit button with the value "**LibrarianCreationBtn**"
 - e. For Create CreateMembers Stipend - Set an ID attribute on the submit button with the value "**MemberCreationBtn**"
6. Implement the CreateBranch(),CreateBooksDetails()and CreateLibrarian() ,
CreateMembership()
7. action for http postrequest to carryout the following operations
 - f. For all 4 post actions Validate the model and return the view if model is invalid
 - g. If model is valid save the model in the database
 - h. When the **Branch** details are saved successfully in the database, the put an acknowledgement in ViewBag by creating a "**Message**" property with value as "**Branch Details Added Successfully**"
 - i. When the **Book** details are saved successfully in the database, the put an acknowledgement in ViewBag by creating a "**Message**" property with value as "**Book Details Added Successfully**"
 - j. When the **Librarian** details are saved successfully in the database, the put an acknowledgement in ViewBag by creating a "**Message**" property with value as "**Librarian Details Added Successfully**"
 - k. When the **Member** details are saved successfully in the database, the put an acknowledgement in ViewBag by creating a "**Message**" property with value as "**Member Details Added Successfully**"
8. Modify the Create views to display the value of ViewBag's Message property inside an <h2> element. Assign the ID="**Message**" attribute to <h2>
9. For all the Lists page, the table tag should have the same id as per the below-given id

BranchList Table Id - **getBranches**

BookList Table Id - **getBooks**

LibrarianList Table Id – **getLibrarians**

MemberList Table Id - **getMembers**

11.0 Evaluation Areas

01	Launch of the application from Branch, Book, Librarian and Member pages
02	Logic in create functionality and Success message
03	Validation of input on controls on all pages
04	Creating properties and get post method checking
05	Checking ability in razor engine