**Atea Console Application**

**Introduction:**

The purpose of this application is to develop a console application that takes two arguments, creates a generic extension method that adds these two arguments and display the result to the users. The extension method is generic method that can take different types of arguments like integer, decimal or string. The application also a database to store and retrieve the data. The application also performs the testing by using the XUnit tests.

**Description:**

Visual Studio 2022 and .Net 6 is used to develop this application. When the application starts, it ask user to enter the two arguments. Then the application displays a menu to the user where the user can select a choice to add the arguments, save these arguments to the database or display a list of all arguments saved in the database.

If the user enters the arguments empty, it does not allow users to do so. The system will ask again and again to enter the argument. If both the arguments are not empty then the system will display the menu. When saving these arguments, the system will display a message to the user showing that the record is added successfully. In case arguments are not stored, it will display an error message.

When displaying a list of arguments, the system will display the arguments to the users. In case of error or there is no record in the database, the system will show an error message to the user.

**Extension Methods:**

These are the additional methods that allow to inject the additional methods with out modifying or deriving the original class, struct or interface. These are the static method defined in a static class and the method must have this modifier before the first parameter. A library project has been created containing the class calculator. This is a static class containing an extension metho Add.

public static T Add<T>(this T num1, T num2)

{

dynamic x = num1;

dynamic y = num2;

return x + y;

}

**Generic Methods:**

Generic methods are the methods that are defined using the type parameters without the specific data type. The above method Add is a generic method that can take paraments of any data type. It is defined by <t>. In this application, the Add method is taking 3 types of parameters, int, decimal and float. This method add these parameters in case of int and decimal and concatenate if parameters are of type string and then return the arguments back.

**Database:**

For database connectivity, Entity framework core is used and at backend SQL Server is used to store the data. The application also retrieves the data and display this data to the user.



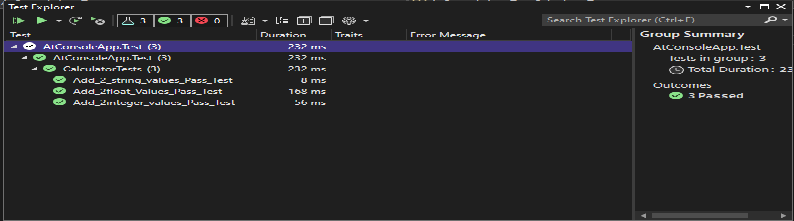
The user is asked to enter the arguments then to select a number between 1-3 for choice selection. He has chosen to select 2 to add this data to the database. The system takes these arguments, save this data and then shows a message “Record added successfully”.



When user selects option3, the system showing the list of saved arguments.

**XUnit Tests:**

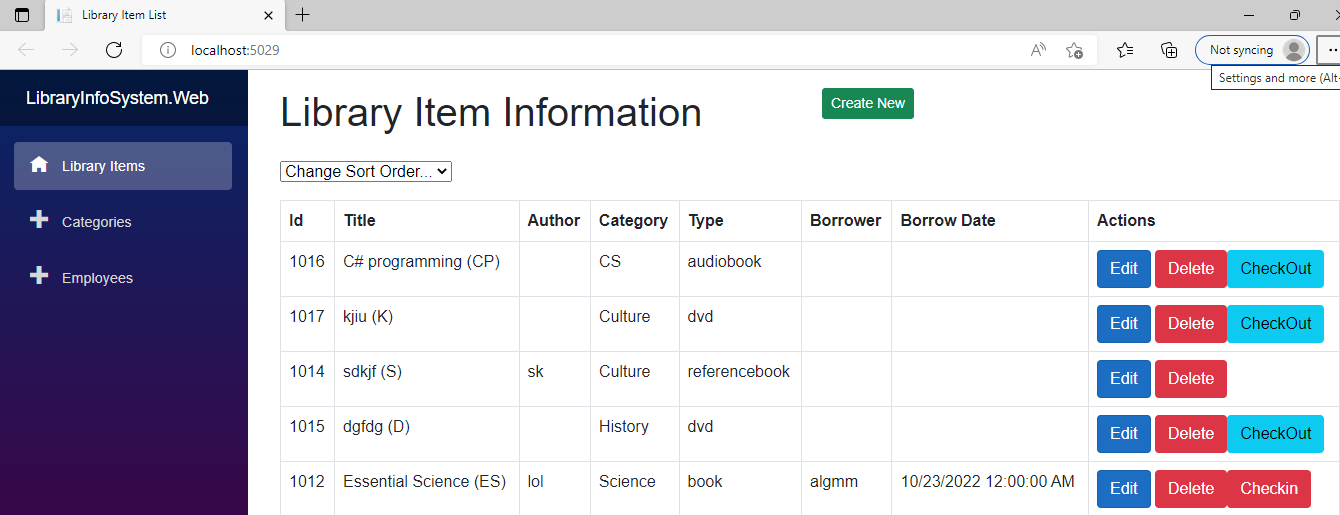
XUnit tests are used to do the auto testing for the extension method. XUnits are more extensible and flexible than other tests. A new project of type XUnit is created for the testing. There are 3 tests to verify the extension method. First test takes the integers values, second test takes the decimal values and third takes the string values. All these are passed tests.



**Users’ Guide:**

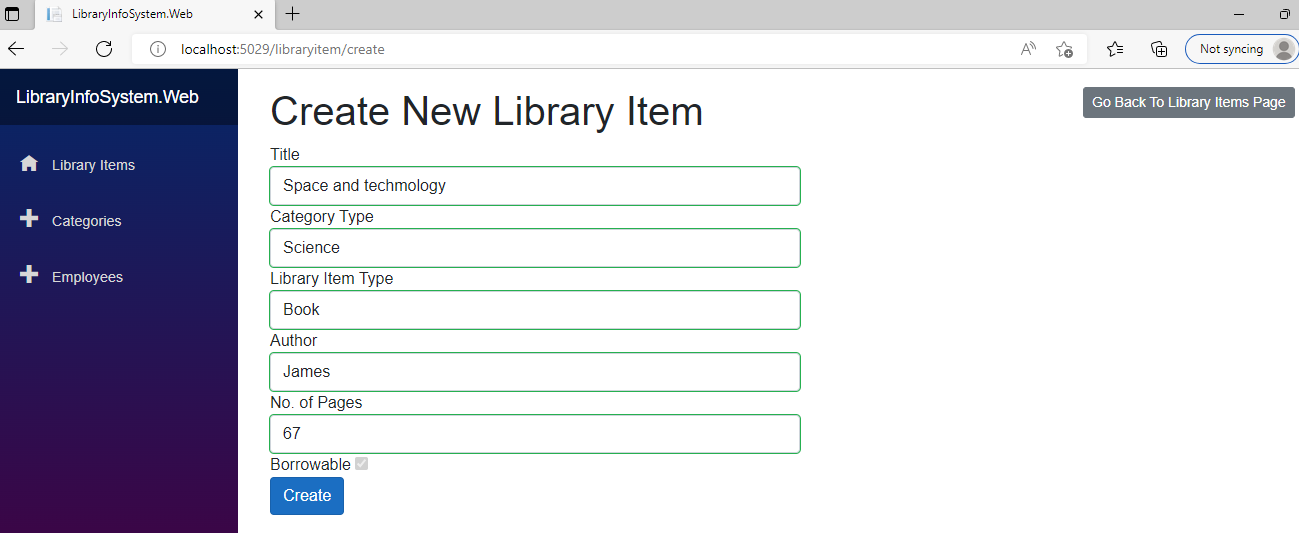
**List of Library Item:**

When the application starts, it displays the list of all the library items that are stored in the database. One the left-hand side of the page is the menu bar, using this menu user can open the different options. The list is sorted by category. The user can also sort by type of the library item. The tile of each library item is decorated with acronym. The user can create, edit, delete, check in and check out the library item. There are 4 different types of library items, book, reference book, DVD and audio book. These items can by checked out by the user except the reference book. If an item is checked out, it will display the borrower’s name and date of borrow. When an item is checkout, it can not be checked in. The user can only check out this item. Similarly, if an item is in the library, the user can check out this item.

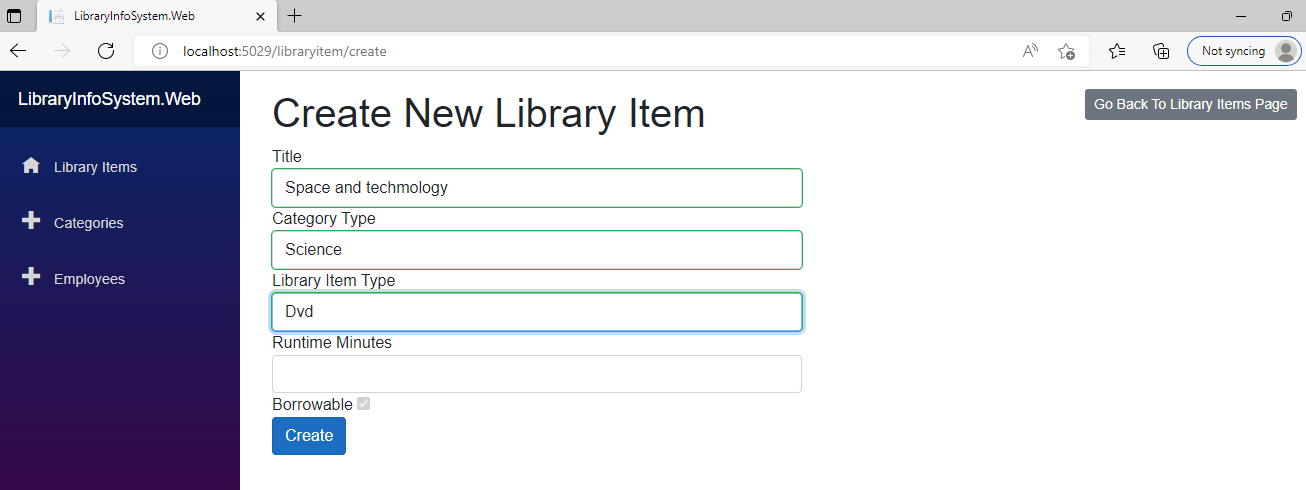


**List Item Create page:**

When user will click on the create button, the create page will open when user can add a new library item. Depending on the library item type, the user has to fill in the different information. If the type of library item is book and reference book, the user will fill in the title, category type, author and number of pages.



If the type of library item is dvd or audio, the user will fill in the title, category type and run type minutes. This page also contains a button to move to the main page.



If compulosry fields are left blank the application will give errors.

Graphical user interface, application

Description automatically generated

**Edit Page:**

The edit page is used to edit a library item. It will display the record of the library item to be edited.

Graphical user interface, application, Word

Description automatically generated

**Categories Page:**

The categories page information about different type of categories. Each library item will belong to a category. The page has button to create a new category, edit and delete a category. The user can not delete a category if it is being referenced by the library item.

Graphical user interface, application, table

Description automatically generated

**Create New Category:**

This page is used to create a new category. The only field to enter is the name of the category. Name of each category must be unique otherwise the system will not store it in the database.

Graphical user interface, text, application

Description automatically generated

**Employees Information Page:**

This page contains information about the employee’s status like their first name, last name, salary etc. The user can edit, delete and create the new employee using this page. The list is sorted by the category of employees. It is first displaying the CEO, then employees and at the last the managers.

Graphical user interface, application

Description automatically generated

**Create New Employee Page:**

This page is used to create a new employee. The application will ask the user to enter the first name, last name, salary scale and employee type. The employee can be CEO, Manager or employee. The application will first ask the user what type of employee he is going to create. If the type of employee is manager, the application will display name of CEO and list of other managers who can manage this manager.

Graphical user interface, application

Description automatically generated

As the employee is managed by any manager so it will display a list of managers to be selected for this employee.

Graphical user interface, text, application, Word

Description automatically generated

If the type of the employee is CEO, it will do nothing as there can be no manager of the CEO. The application doesn’t allow to create a CEO if an CEO already exists in the database.

Graphical user interface, text, application

Description automatically generated

**Delete Employee:**

The delete button that is present on the employee information page is used to delete the record of an employee. If type of the employee is manager or CEO and he is managing other managers or employees, it will not allow to delete it.

**Test Run:**

To run the application, following steps must be done:

"AllowedHosts": "\*",

"ConnectionStrings": {

"LibraryContext": "Server=localhost;Database=Library;Trusted\_Connection=True;MultipleActiveResultSets=true"

}

* Run these entity framework commands for migration
* Add-Migration migration-name
* Update-database
* Seed the data manually in the tables
* In the client application, fix the port in program.cs file on whice server is running
* Make both server and client projects as startup projects(if not).