University of Zakho

Department computer sciences

Year (2021-2022)

Project About:

"Library Application"

Prepared By Students :

Hakar Tayar Abdi **Hozheen Sadiq Abdullah**

Sada Kadhim Abdulrahim **Aeshi Shahin Hassan**

Parwar shaban Ali

Supervised By :

Yousif Arshak

Introduction :

Xamarin.forms

Xamarin.Forms is an open-source UI framework. Xamarin.Forms allows developers to build Xamarin.Android, Xamarin.iOS, and Windows applications from a single shared codebase.

Xamarin.Forms allows developers to create user interfaces in XAML with code-behind in C#. These interfaces are rendered as performant native controls on each platform

Who Xamarin.Forms is for

Xamarin.Forms is for developers with the following goals:

- Share UI layout and design across platforms.
- Share code, test and business logic across platforms.
- Write cross-platform apps in C# with Visual Studio.

Features of Xamarin:

1. **Complete Binding for the underlying SDKs** – Xamarin contains bindings for nearly the entire underlying platform SDKs in both iOS and Android. Additionally, these bindings are strongly-typed, which means that they're easy to navigate and use, and provide robust compile-time type checking and during development. This leads to fewer runtime errors and higher quality applications.
2. **Objective-C, Java, C, and C++ Interop** – Xamarin provides facilities for directly invoking Objective-C, Java, C, and C++ libraries, giving you the power to use a wide array of third party code that has already been created. This lets you take advantage of existing iOS and Android libraries written in Objective-C, Java, or C/C++. Additionally, Xamarin offers binding projects that allow you to easily bind native Objective-C and Java libraries using a declarative syntax.
3. **Modern Language Constructs** – Xamarin applications are written in C#, a modern language that includes significant improvements over Objective-C and Java such as Dynamic Language Features, Functional Constructs such as Lambdas, LINQ, Parallel Programming features, sophisticated Generics, and more.

4. **Amazing Base Class Library (BCL)**– Xamarin applications use the .NET BCL, a massive collection of classes that have comprehensive and streamlined features such as powerful XML, Database, Serialization, IO, String, and Networking support, just to name a few. Additionally, existing C# code can be compiled for use in applications, which provides access to thousands upon thousands of libraries that will let you do things that aren't already covered in the BCL.
5. **Modern Integrated Development Environment (IDE)** – Xamarin uses Xamarin Studio on Mac OS X and Visual Studio on Windows. These are both modern IDE's that include features such as code auto-completion, a sophisticated Project and Solution management system, a comprehensive project template library, integrated source control, and many others.

6. **Mobile Cross-Platform Support** – Xamarin offers sophisticated cross-platform support for the three major mobile platforms of iOS, Android, and Windows Phone. Applications can be written to share up to 90% of their code, and our Xamarin. The mobile library offers a unified API to access common resources across all three platforms. This can significantly reduce both development costs and time to market for mobile developers that target the three most popular mobile platforms.

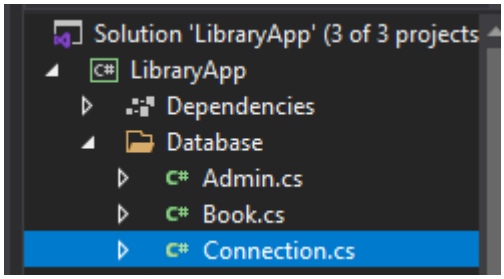
What is a Sqlite

SQLite is a database engine. It is software that allows users to interact with a relational database. In SQLite, a database is stored in a single file — a trait that distinguishes it from other database engines. This fact allows for a great deal of accessibility: copying a database is no more complicated than copying the file that stores the data, sharing a database can mean sending an email attachment.

Application and design and implementation

Database Design :

To create our app we first needed to create a local database to store and retrieve admin and book information, for that purpose we created 3 class files the first is called connection.cs which will be the database and for the second and third files we created are the tables that are going to be created upon creating the database, our database files:



Connection class

```
public class Connection
{
    private readonly SQLiteAsyncConnection _connection;
    1 reference
    public Connection(string dbPath)...
    3 references
    public Task<List<Book>> GetBooksAsync()...
    0 references
    public Task<List<Admin>> GetTeachrsAsync()...
    0 references
    public Task<List<Book>> SearchForBook(string bookTitle)...
    0 references
    public Task<Book> GetBookInfo(int stdId)...
    1 reference
    public Task<int> UpdateBookAsync(Book Book)...
    1 reference
    public Task<int> DeleteBookAsync(Book Book)...
    1 reference
    public Task<int> InsertBookAsync(Book Book)...
    1 reference
    public Task<int> InsertAdminAsync(Admin Admin)...
    1 reference
    public bool LoginValidateAdmin(string eml, string psword)...
}
```

Admin.cs:

This file contains all admin information as well as data type , we also can create constraints such as the id Primary key, and AutoIncrement Constraints.

```
namespace LibraryApp
{
    6 references
    public class Admin
    {
        [PrimaryKey, AutoIncrement]
        0 references
        public int Id { get; set; }
        2 references
        public string Email { get; set; }
        2 references
        public string Password { get; set; }
    }
}
```

Book.cs :

This file contains all book information as well as data type , we also can create constraints such as the id Primary key, and AutoIncrement Constraints.

```
public class Book
{
    [PrimaryKey, AutoIncrement]
    4 references
    public int Id { get; set; }
    4 references
    public string BookTitle { get; set; }
    4 references
    public string BookType { get; set; }
    4 references
    public string BookAuthor { get; set; }
    4 references
    public string PublishDate { get; set; }
    4 references
    public string Quantity { get; set; }

}
```

Constructor: this a construct for connection with database and crate table for book , admin

And insert one row for admin to can used later to login into application

```
private readonly SQLiteAsyncConnection _connection;
1 reference
public Connection(string dbPath)
{
    _connection = new SQLiteAsyncConnection(dbPath);

    _connection.CreateTableAsync<Book>();

    _connection.CreateTableAsync<Admin>();

    var data = _connection.Table<Book>();

    var ad = data.FirstOrDefaultAsync();

    if (ad.Result == null)
    {
        InsertAdminAsync(new Admin
        {
            Email = "root",
            Password = "admin",
        });
    }
}
```

GetBooksAsync Function:

This function return all row in table book and convert into list of book

```
public Task<List<Book>> GetBooksAsync()
{
    return _connection.Table<Book>().ToListAsync();
}
```

GetAdminsAsync Function:

This function return all row in table book and convert into list of admin

```
public Task<List<Admin>> GetAdminsAsync()
{
    return _connection.Table<Admin>().ToListAsync();
}
```

Select book where first name = `bookTitle`

```
public Task<List<Book>> SearchForBook(string bookTitle)
{
    return _connection.QueryAsync<Book>($"SELECT * FROM Book WHERE BookTitle LIKE '{bookTitle}%'");
}
```

Only get information book is you want and select

```
public Task<Book> GetBookInfo(int stdId)
{
    return _connection.Table<Book>().Where(x => x.Id == stdId).FirstAsync();
}
1 reference
```

This for update and edit book information

```
1 reference
public Task<int> UpdateBookAsync(Book Book)
{
    return _connection.UpdateAsync(Book);
}
```

This for delete book

```
public Task<int> DeleteBookAsync(Book Book)
{
    return _connection.DeleteAsync(Book);
}
```

This function for insert a new book

```
public Task<int> InsertBookAsync(Book Book)
{
    return _connection.InsertAsync(Book);
}
```

This function for insert Admin

```
public Task<int> InsertAdminAsync(Admin Admin)
{
    return _connection.InsertAsync(Admin);
}
1 reference
```

This function for login when email and password is true he can go to a new page

```
public bool LoginValidateAdmin(string eml, string psword)
{
    var data = _connection.Table<Admin>();

    var d1 = data.Where(x => x.Email == eml && x.Password == psword).FirstOrDefaultAsync();

    if (d1.Result != null)
        return true;
    else
        return false;
}
```

This function to insert book in database

```
private void btnInsert_Clicked(object sender, EventArgs e)
{
    if (entBookTitle.Text == "")
    {
        DisplayAlert("Enter","Please enter the book's title","Ok");
        return;
    }
    if (entBookType.Text == "")
    {
        DisplayAlert("Enter", "Please enter the book's type", "Ok");
        return;
    }
    if (entAuthorName.Text == "")
    {
        DisplayAlert("Enter", "Please enter the book's author's name", "Ok");
        return;
    }
    if (entPublishDate.Text == "")
    {
        DisplayAlert("Enter", "Please enter the book publishdate", "Ok");
        return;
    }
    if (entQntity.Text == "")
    {
        DisplayAlert("Enter", "Please enter the book Quntity", "Ok");
        return;
    }
    App.DBConnection.InsertBookAsync(new Book
    {
        BookTitle = entBookTitle.Text,
        BookType = entBookType.Text,
        BookAuthor = entAuthorName.Text,
        PublishDate = entPublishDate.Text,
        Quantity = entQntity.Text
    });
    Navigation.PopModalAsync();
}
```

This to show information for each book and you selected

```
protected override void OnAppearing()
{
    base.OnAppearing();

    bookTitle.Text += home.bookTitle;
    bookType.Text += home.bookType;
    bookAuthor.Text += home.bookAuthor;
    bookDate.Text += home.bookDate;
    bookQuantity.Text += home.bookQntitiy;
}
```

This function used for update information in book

```
private async void btnUpdate_Clicked(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(entBookTitle.Text) || string.IsNullOrEmpty(entBookType.Text) || string.IsNullOrEmpty(entAuthorName.Text)
        || string.IsNullOrEmpty(entDate.Text) || string.IsNullOrEmpty(entQntity.Text))
    {
        await DisplayAlert("Name", "Please enter all the infromation about the book!", "Ok");
        return;
    }

    Book book = new Book()
    {
        Id = Convert.ToInt32(entBookID.Text),
        BookTitle = entBookTitle.Text,
        BookType = entBookType.Text,
        BookAuthor = entAuthorName.Text,
        PublishDate = entDate.Text,
        Quantity = entQntity.Text
    };

    await App.DBConnection.UpdateBookAsync(book);


    await DisplayAlert("Updated", $"A book has been updated!", "Ok");

    bookView.ItemsSource = await App.DBConnection.GetBooksAsync();

    await Navigation.PopModalAsync();
}
```

```
public update(CollectionView BV)
{
    InitializeComponent();
    bookView = BV;
}

0 references
protected override void OnAppearing()
{
    base.OnAppearing();

    entBookID.Text = home.id;
    entBookTitle.Text = home.bookTitle;
    entBookType.Text = home.bookType;
    entAuthorName.Text = home.bookAuthor;
    entDate.Text = home.bookDate;
    entQntity.Text = home.bookQntitiy;
}
0 references
```

Insert function code

```
private void insert_data_Clicked(object sender, EventArgs e)
{
    Navigation.PushModalAsync(new insert());
}
```

Update function code

```
private void btnEdit_Clicked(object sender, EventArgs e)
{
    SelectedBook = BookView.SelectedItem as Book;
    if (SelectedBook != null)
    {
        id = SelectedBook.Id.ToString();
        bookTitle = SelectedBook.BookTitle;
        bookType = SelectedBook.BookType;
        bookAuthor = SelectedBook.BookAuthor;
        bookDate = SelectedBook.PublishDate;
        bookQntitiy = SelectedBook.Quantity;

        BookView.SelectedItem = null;

        _ = Navigation.PushModalAsync(new update(BookView));
    }
    else
    {
        DisplayAlert("Select Book", "No book is selected!", "Ok");
    }
}
```

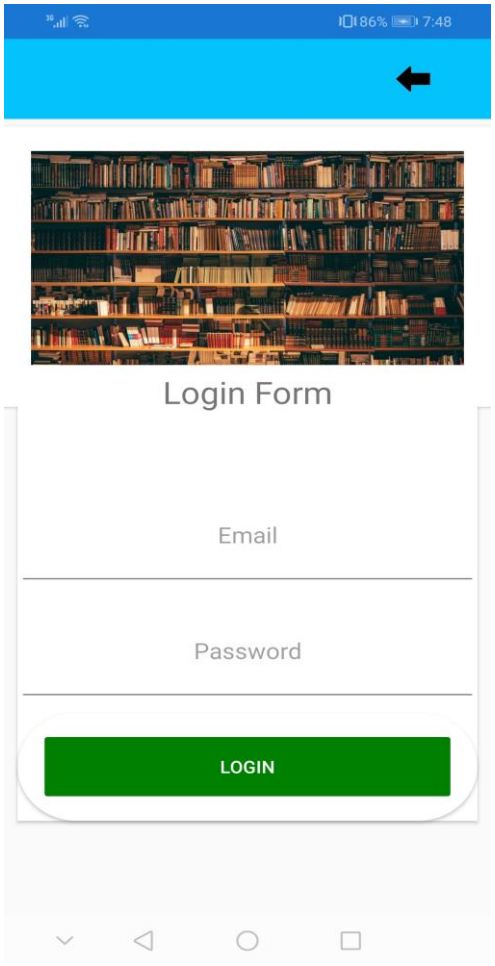
Delete function code

```
private async void btnRemove_Clicked(object sender, EventArgs e)
{
    SelectedBook = BookView.SelectedItem as Book;
    if (SelectedBook != null)
    {
        await App.DBConnection.DeleteBookAsync(SelectedBook);

        await DisplayAlert("Deletion", "A book has been deleted", "Ok");

        BookView.SelectedItem = null;

        BookView.ItemsSource = await App.DBConnection.GetBooksAsync();
    }
    else
    {
        await DisplayAlert("Select Book", "No book is selected!", "Ok");
    }
}
```

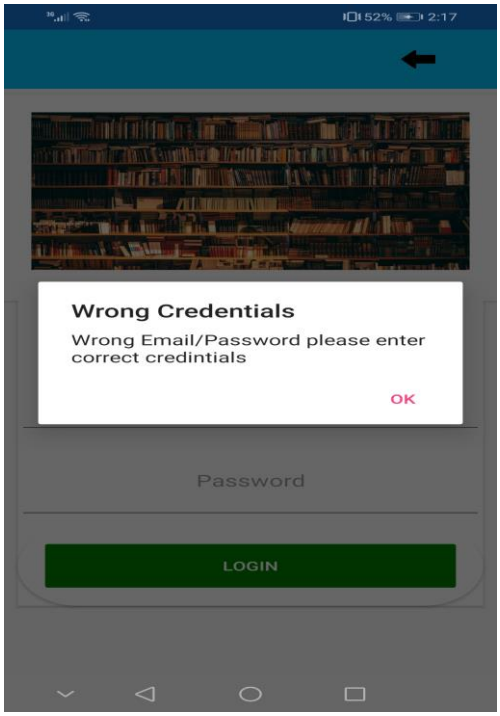
Show function code

```
private void btnShow_Clicked(object sender, EventArgs e)
{
    SelectedBook = BookView.SelectedItem as Book;
    if (SelectedBook != null)
    {
        id = SelectedBook.Id.ToString();
        bookTitle = SelectedBook.BookTitle;
        bookType = SelectedBook.BookType;
        bookAuthor = SelectedBook.BookAuthor;
        bookDate = SelectedBook.PublishDate;
        bookQntitiy = SelectedBook.Quantity;
        BookView.SelectedItem = null;

        Navigation.PushModalAsync(new show());
    }
    else
    {
        DisplayAlert("Select Book", "No book is selected!", "Ok");
    }
}
```

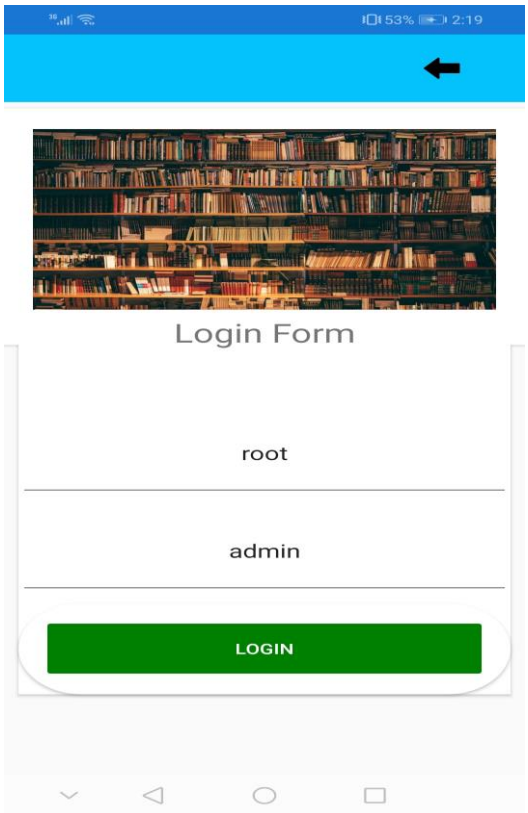
This is my interface ui application



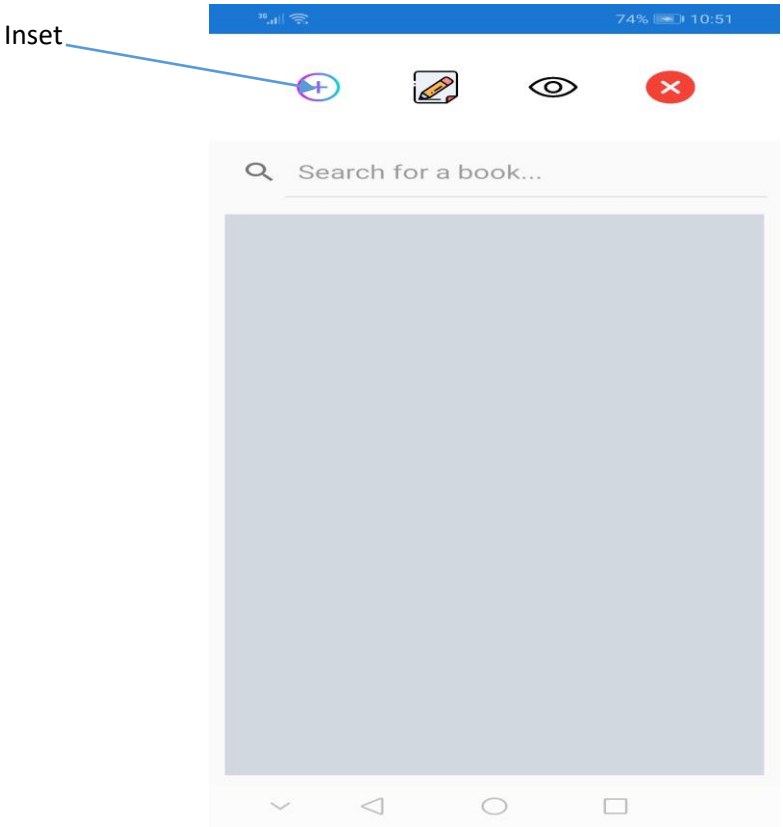
If email or password is empty show this alert



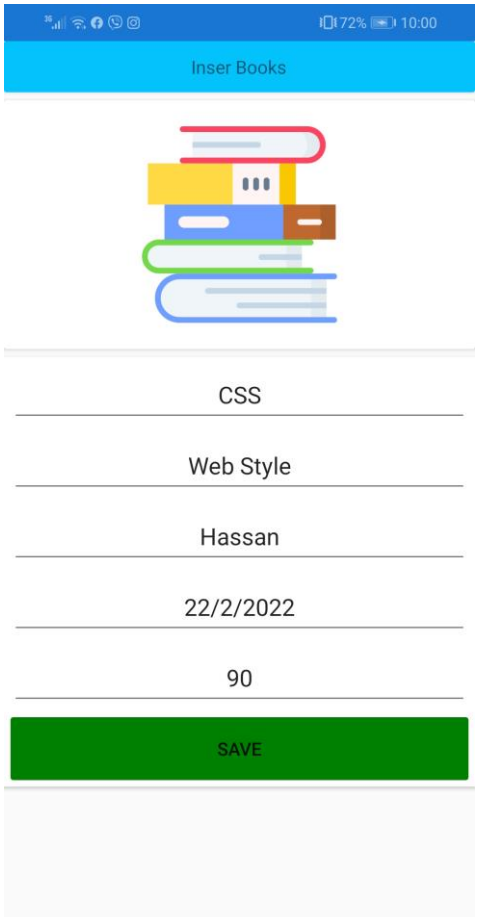
If email and password is tur



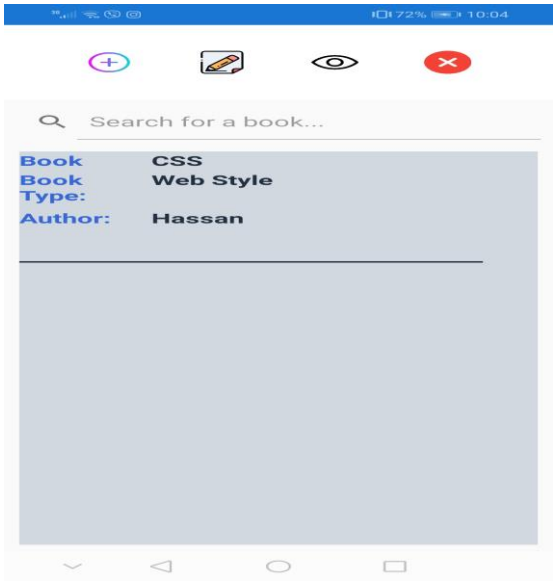
After you login open this page for you



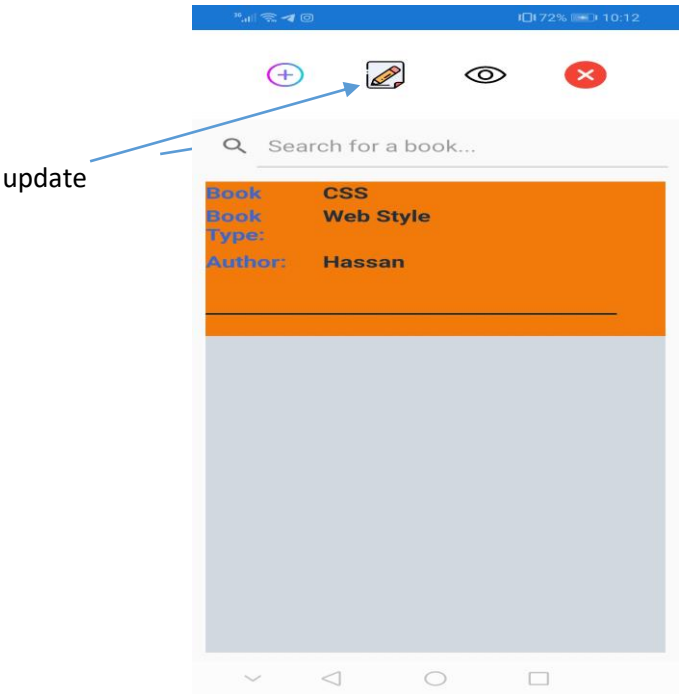
Insert a new book into table



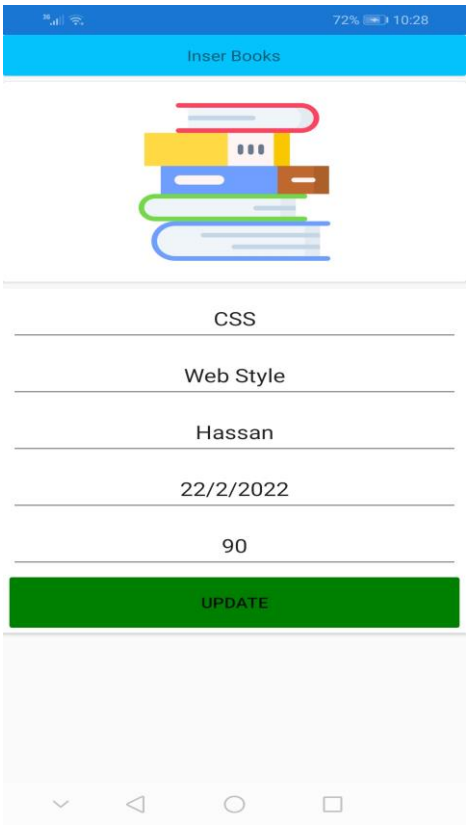
After you clicked button save it save to table



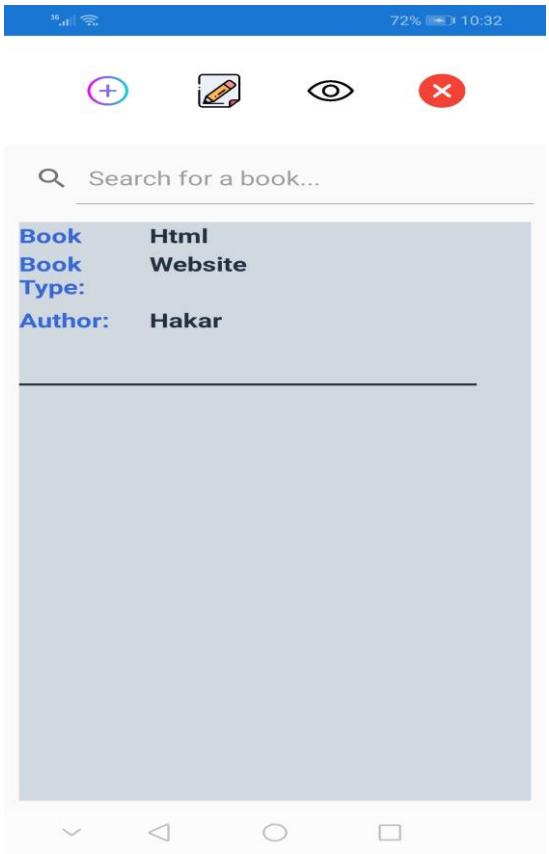
Update book you needed select book you want update



Now you can update information in this book

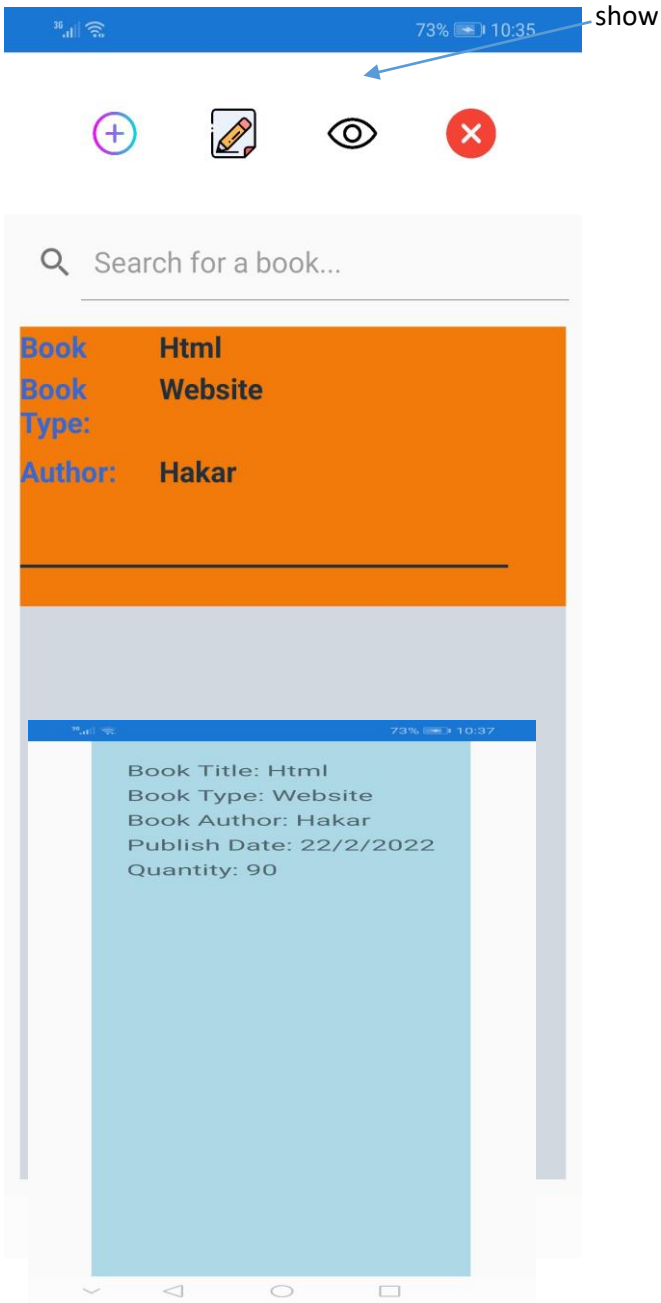


Now updated book name and author and book type

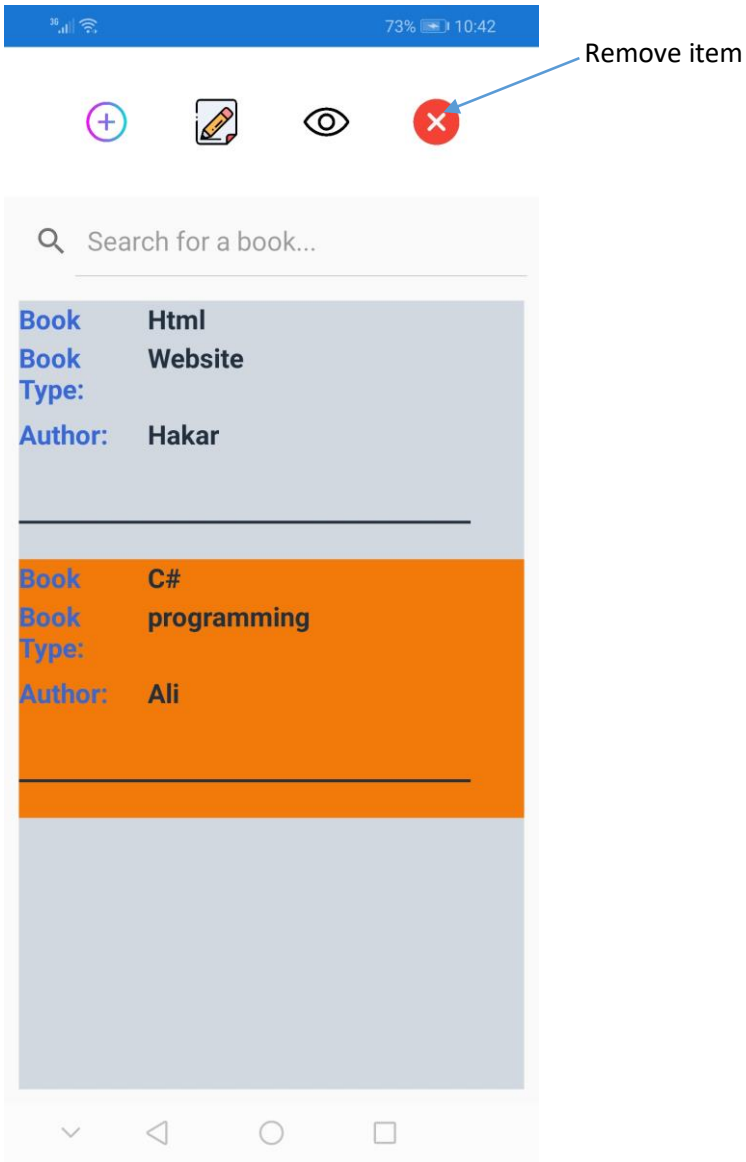


Show data

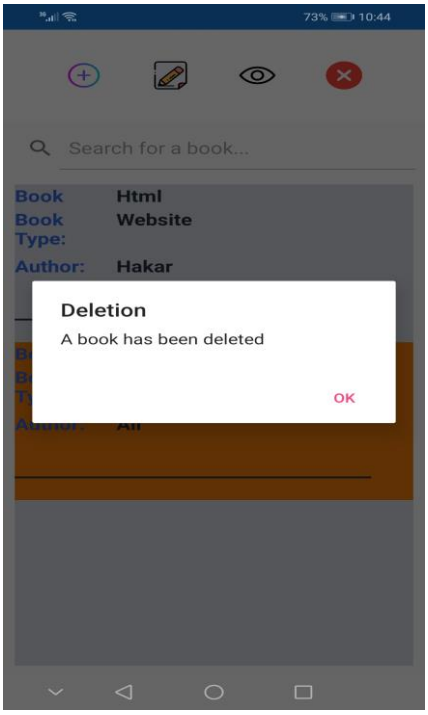
select item and click show icon



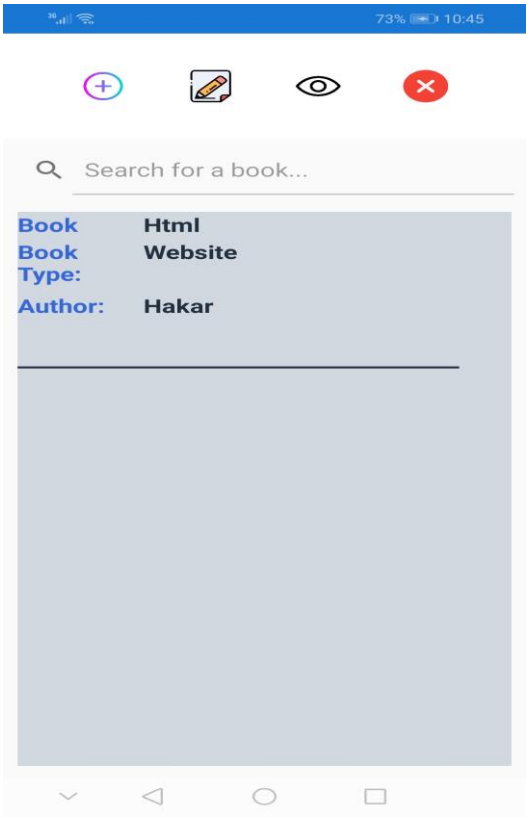
Delete item first select item you want to delete and after click button remove



if you want deleted click ok



Now deleted item



Search book by book name