MULTI-TIER APPLICATIONS – FINAL REPORT

Project Name: Hi-Tech

Present to MR. Quang Hoang Cao

LaSalle College

2020-12-15

By: Xiao Chen

Table of Contents

[Project Description 4](#_Toc58903767)

[Project Design 5](#_Toc58903768)

[Database Design 5](#_Toc58903769)

[Authors Table 5](#_Toc58903770)

[Books Table 5](#_Toc58903771)

[Categories Table 5](#_Toc58903772)

[Customers Table 5](#_Toc58903773)

[Employees Table 6](#_Toc58903774)

[Jobs Table 6](#_Toc58903775)

[OrderLines Table 6](#_Toc58903776)

[Orders Table 6](#_Toc58903777)

[Publishers Table 6](#_Toc58903778)

[Users Table 6](#_Toc58903779)

[Design of Application Domain Classes 7](#_Toc58903780)

[Class Diagram 8](#_Toc58903781)

[Design of GUI Classes 9](#_Toc58903782)

[FormLogIn 9](#_Toc58903783)

[FormMISManager 10](#_Toc58903784)

[FormOrderClerks-Order 10](#_Toc58903785)

[FormOrderClerks-OrderLine 11](#_Toc58903786)

[FormInventoryController-Books 12](#_Toc58903787)

[FormInventoryController-Authors 13](#_Toc58903788)

[FormInventoryController-Categories 14](#_Toc58903789)

[FormInventoryController-Publishers 15](#_Toc58903790)

[FormSalesManager 16](#_Toc58903791)

[Design of Data Access Classes 17](#_Toc58903792)

[Project Implementation 18](#_Toc58903793)

[BLL 18](#_Toc58903794)

[EmployeeAndUser.cs 18](#_Toc58903795)

[Customer.cs 18](#_Toc58903796)

[Author.cs 18](#_Toc58903797)

[Books.cs 18](#_Toc58903798)

[Category.cs 18](#_Toc58903799)

[Publisher.cs 18](#_Toc58903800)

[DAL 18](#_Toc58903801)

[AuthorDB.cs 18](#_Toc58903802)

[BookDB.cs 18](#_Toc58903803)

[CategoryDB.cs 18](#_Toc58903804)

[CustomerDB.cs 18](#_Toc58903805)

[EmployeeAndUserDB.cs 18](#_Toc58903806)

[PublisherDB.cs 18](#_Toc58903807)

[Utility.cs 18](#_Toc58903808)

[Project Testing 19](#_Toc58903809)

[Conclusion 22](#_Toc58903810)

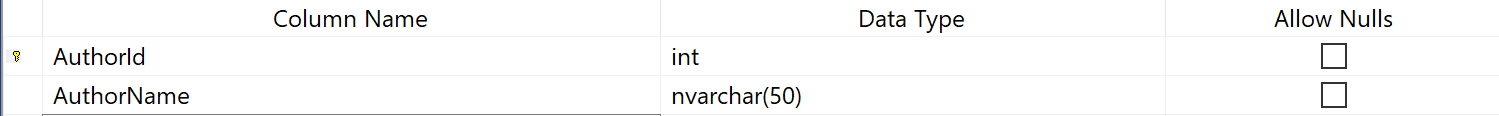
# Project Description

Hi-Tech project is supplying computer science books to nearly all the colleges and universities in Quebec. The computer language of Hi-Tech is most of C#. Hi-Tech uses Microsoft Visual Studio 2019 SQL Server 2017 to complete. In Hi-Tech, you will have different form, each form has the different managers, they can add/update/delete/search/list the information that store in the database already.

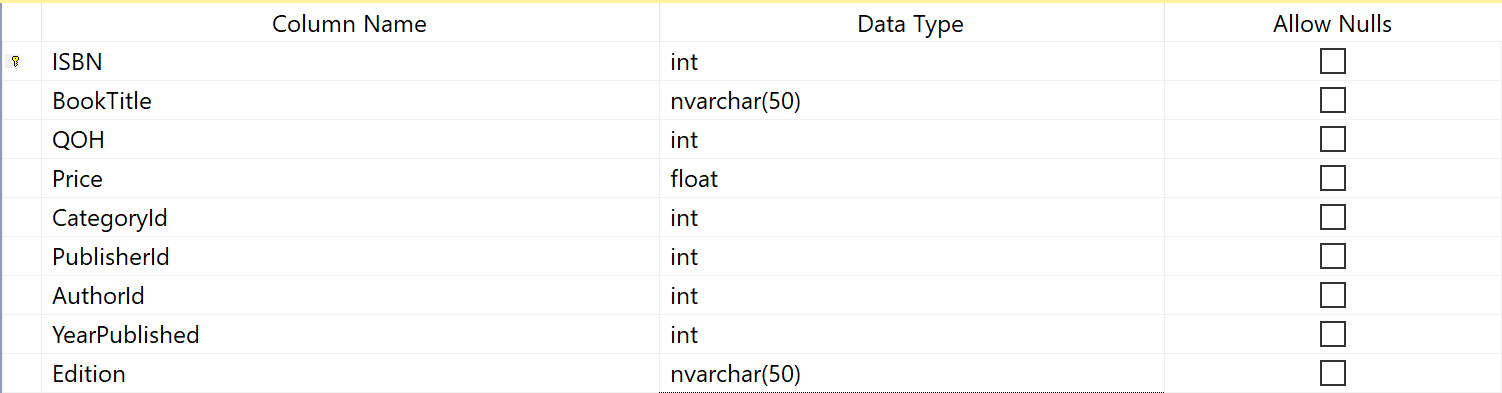
# Project Design

## Database Design

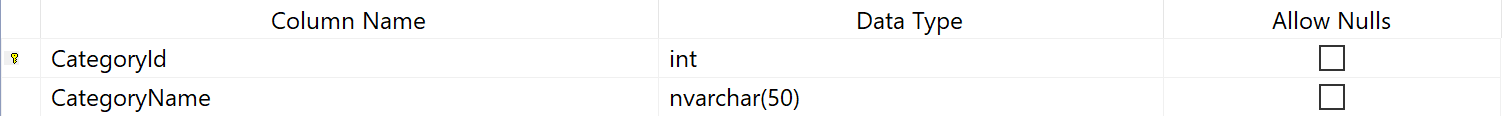
### Authors Table



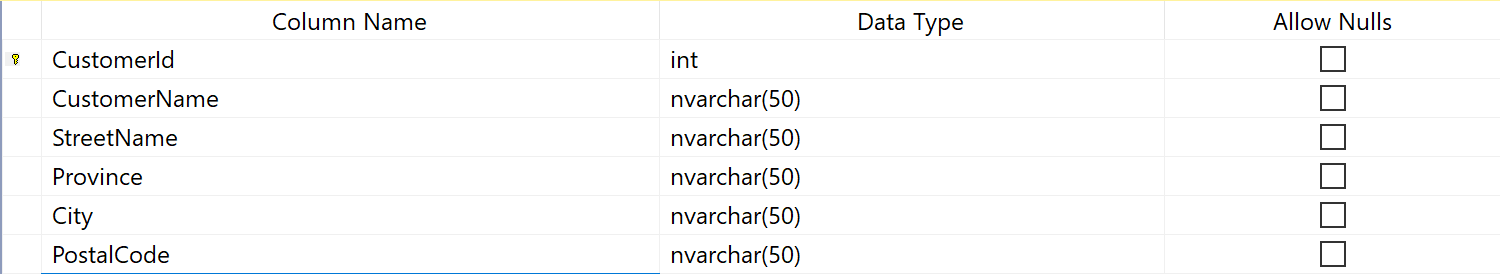
### Books Table



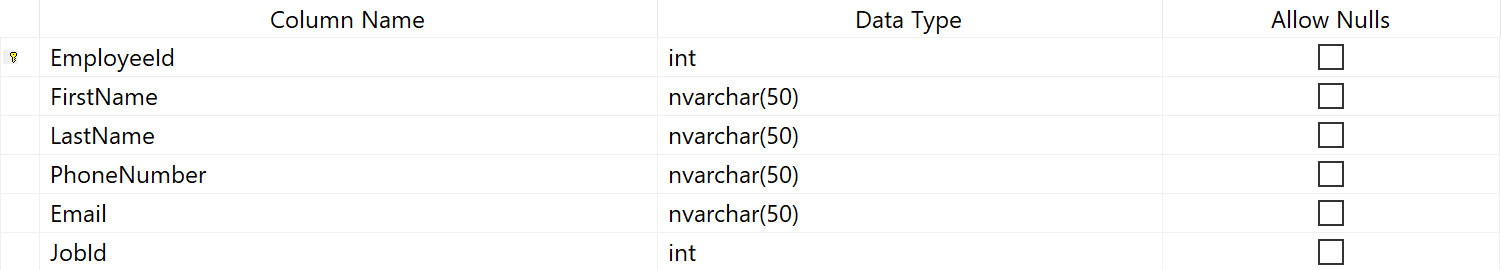
### Categories Table



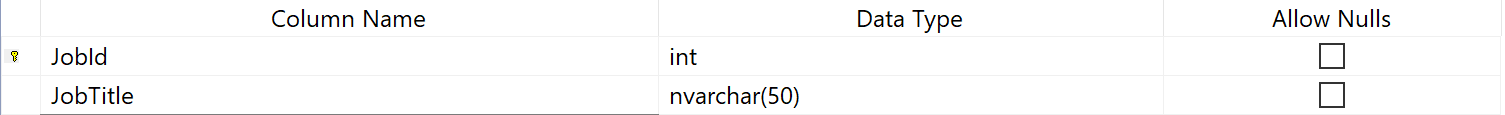
### Customers Table



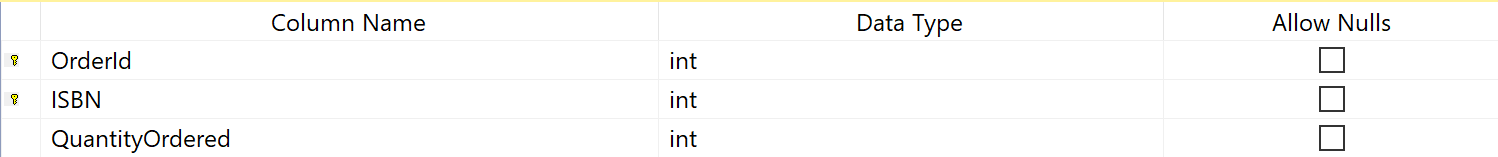
### Employees Table



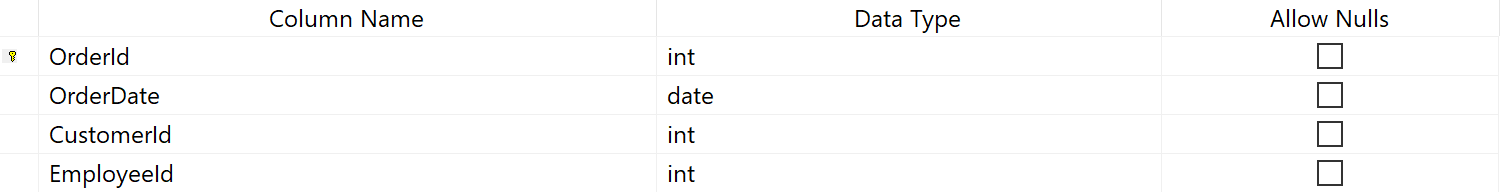
### Jobs Table



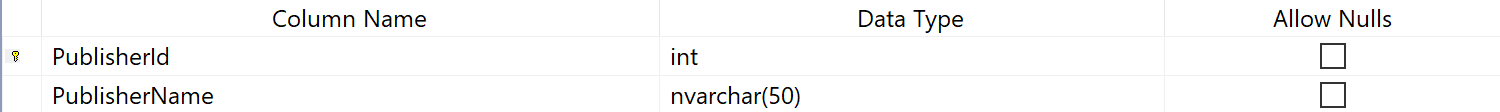
### OrderLines Table



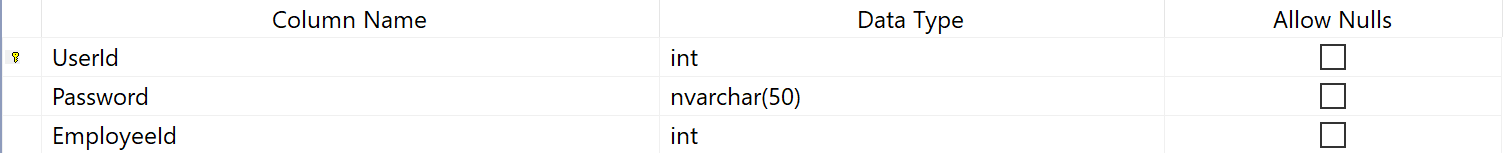
### Orders Table



### Publishers Table

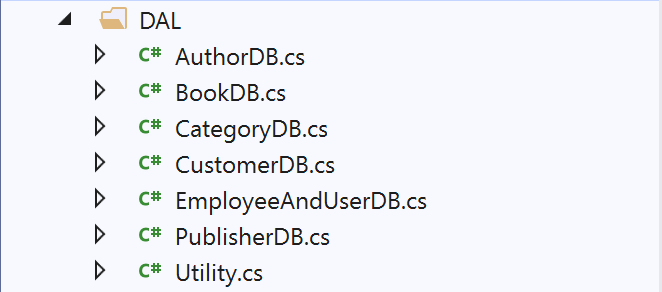


### Users Table

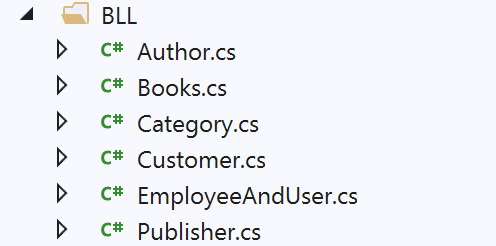


## Design of Application Domain Classes

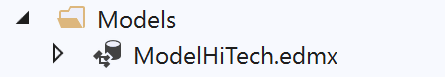
DAL folder:



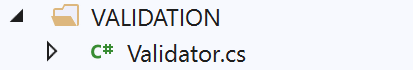
BLL folder:



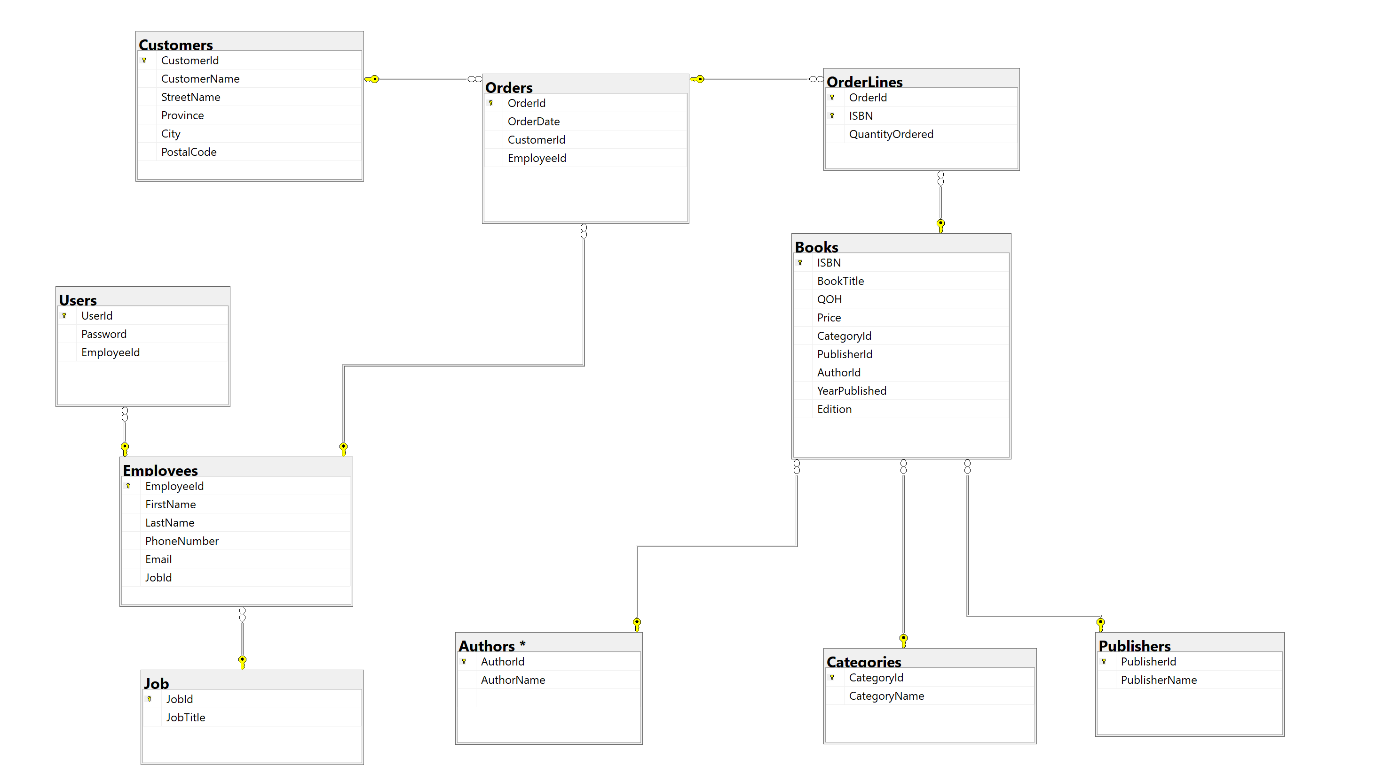
Models folder:



Validation folder:

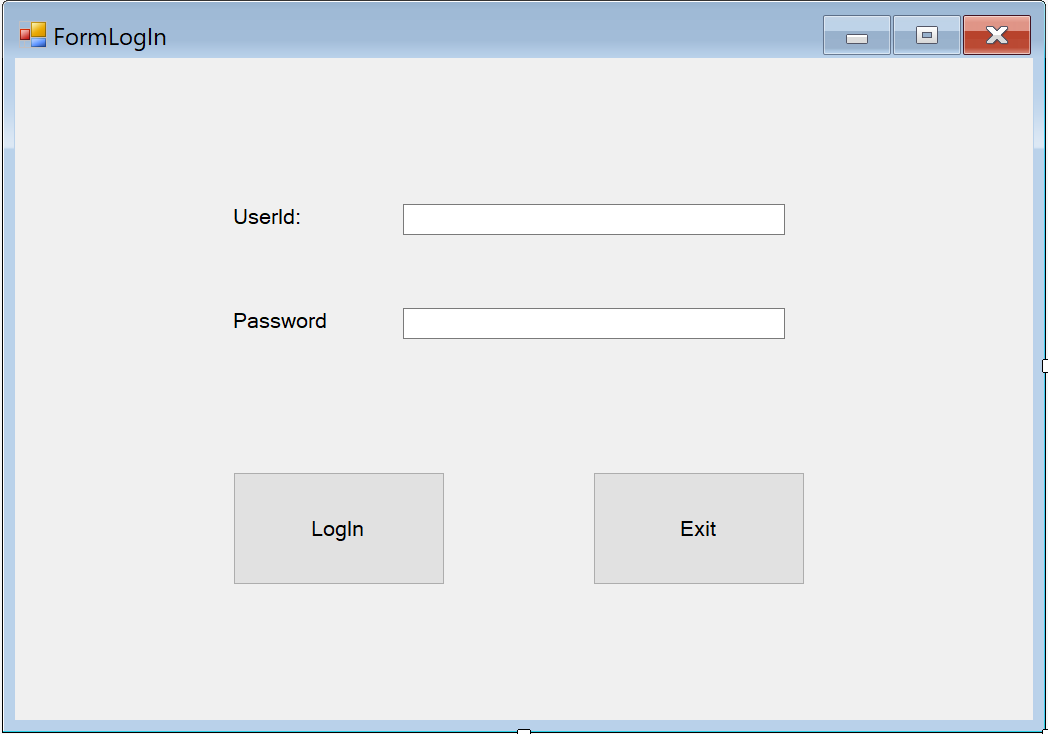


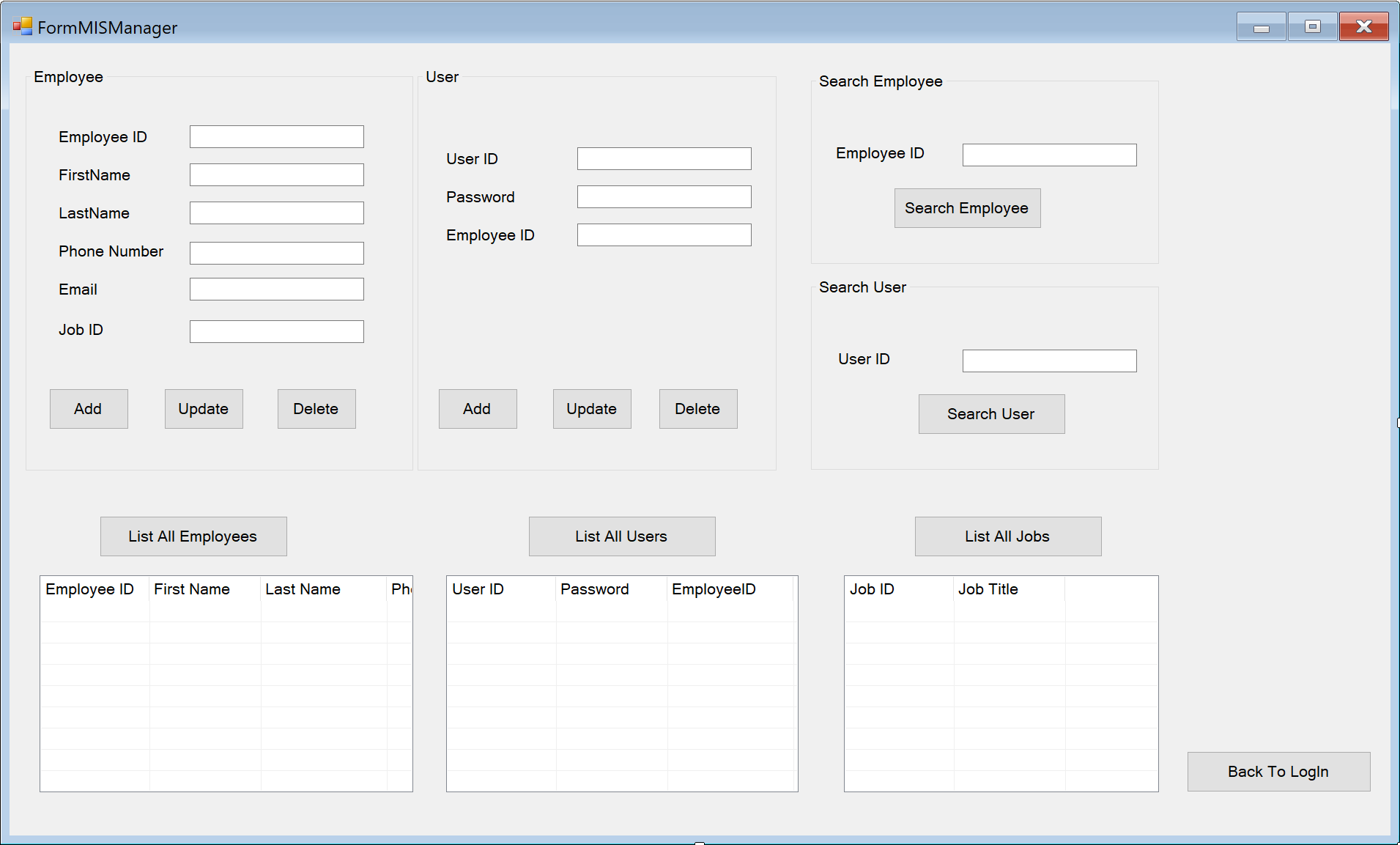
## Class Diagram

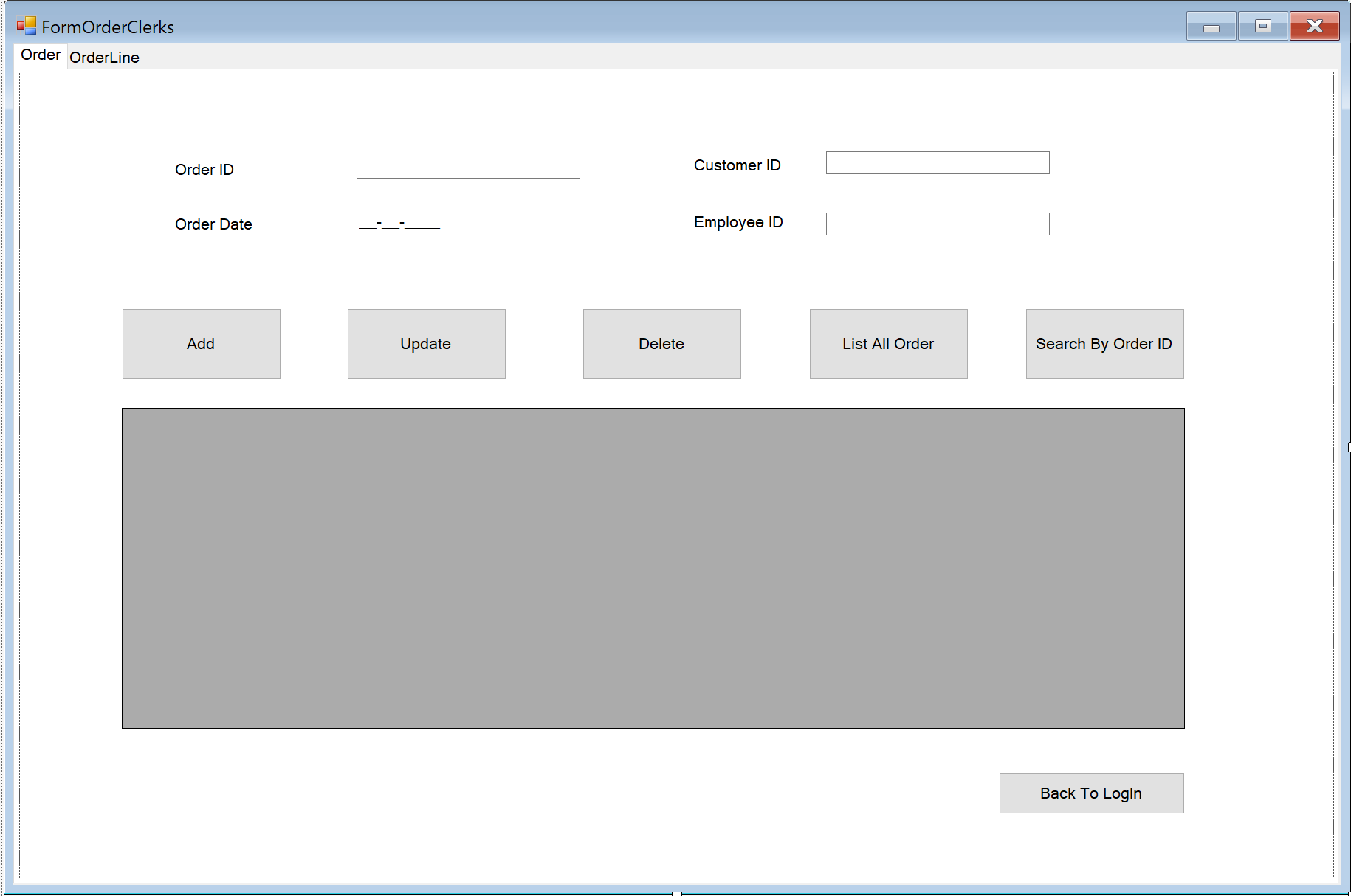


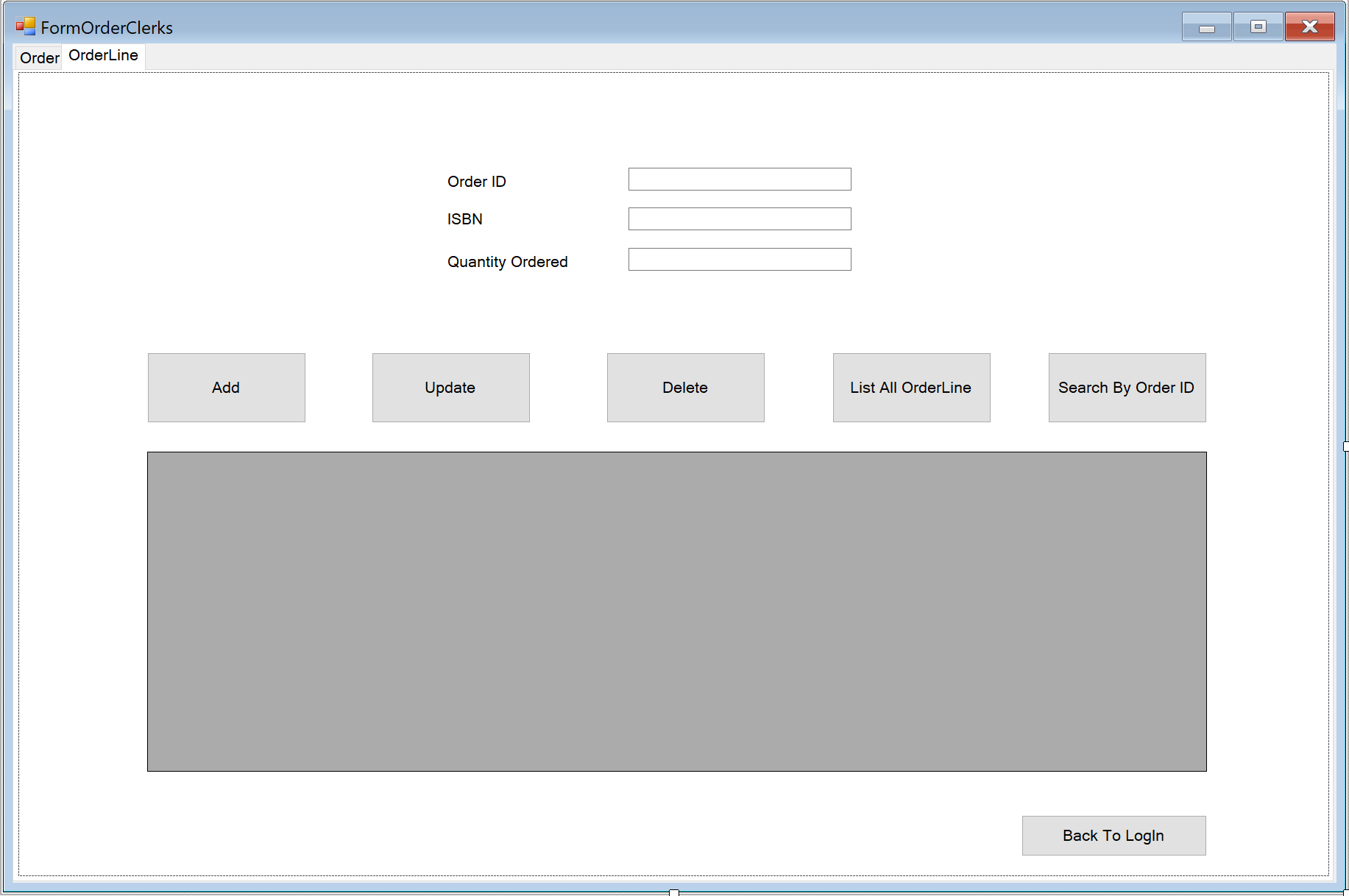
## ***Design of GUI Classes***

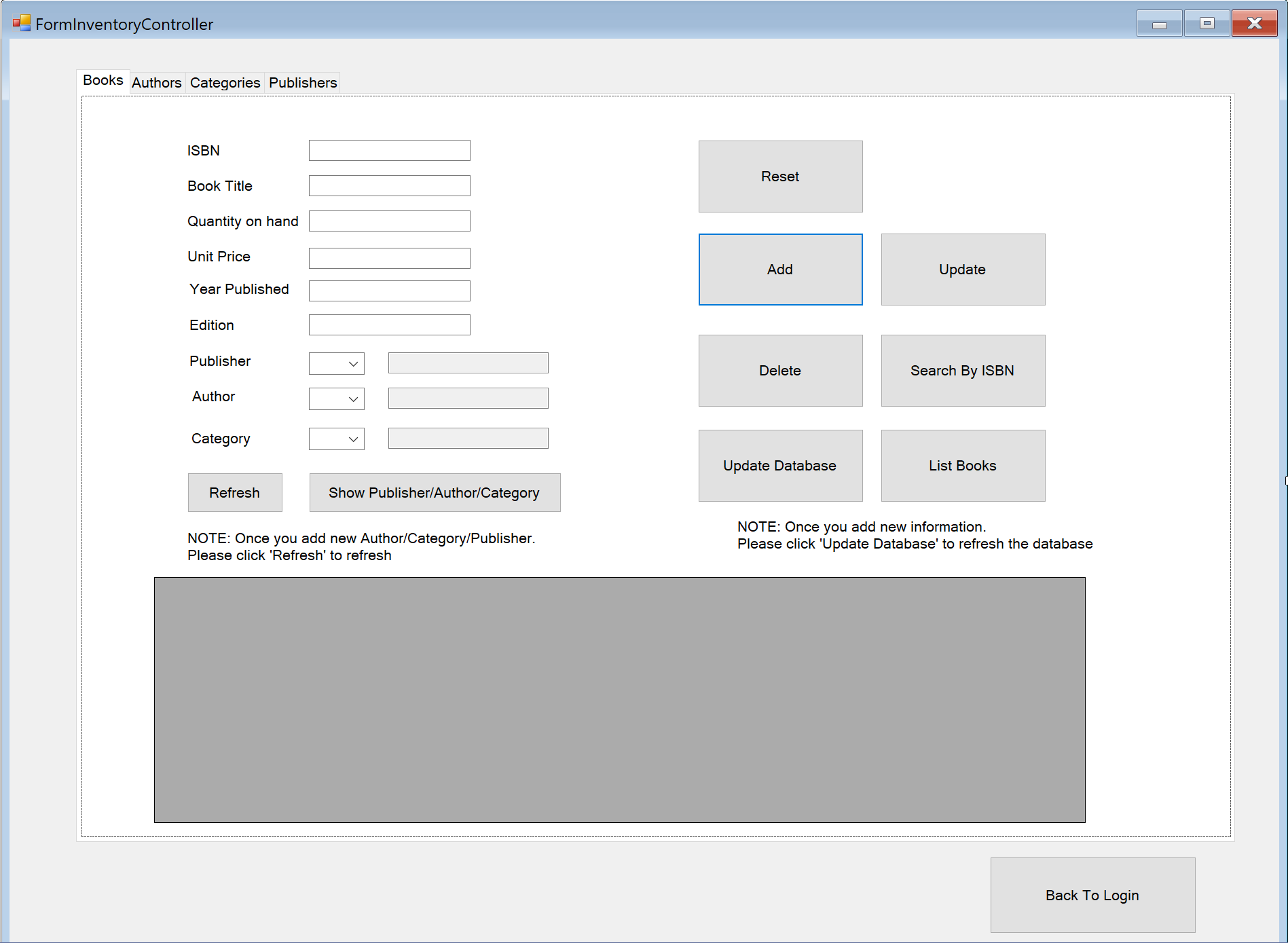
### **FormLogIn**

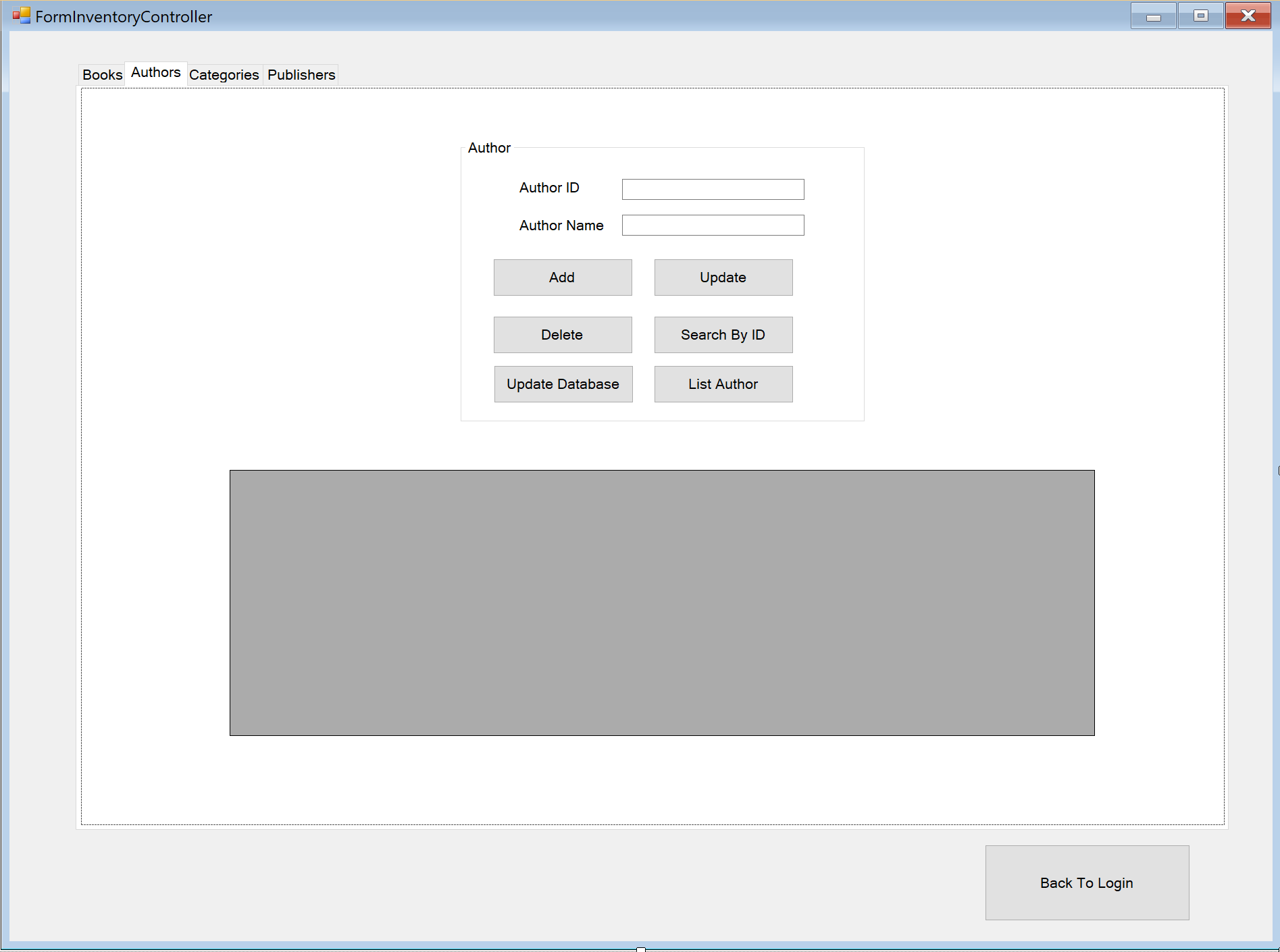


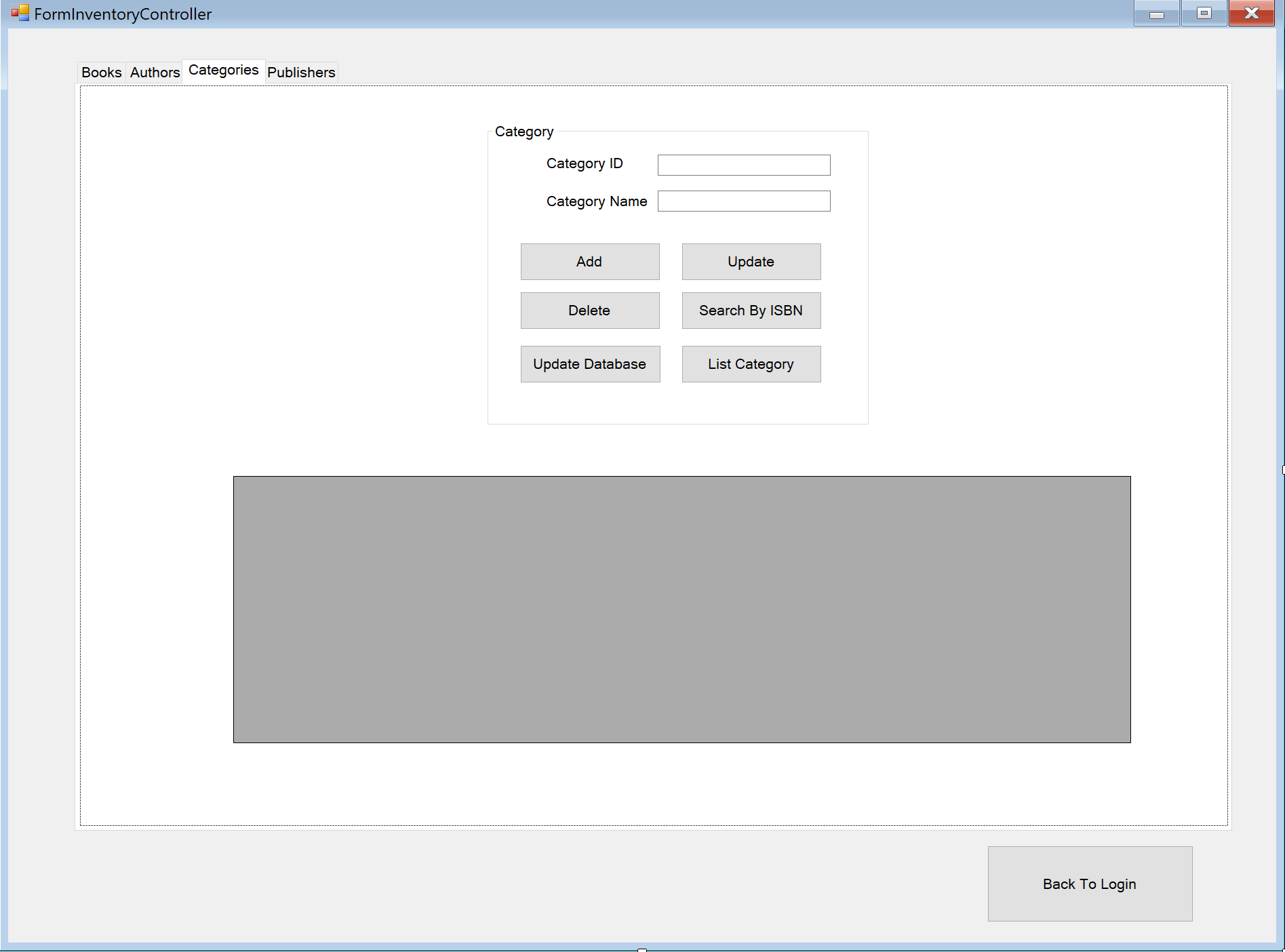
FormMISManager 

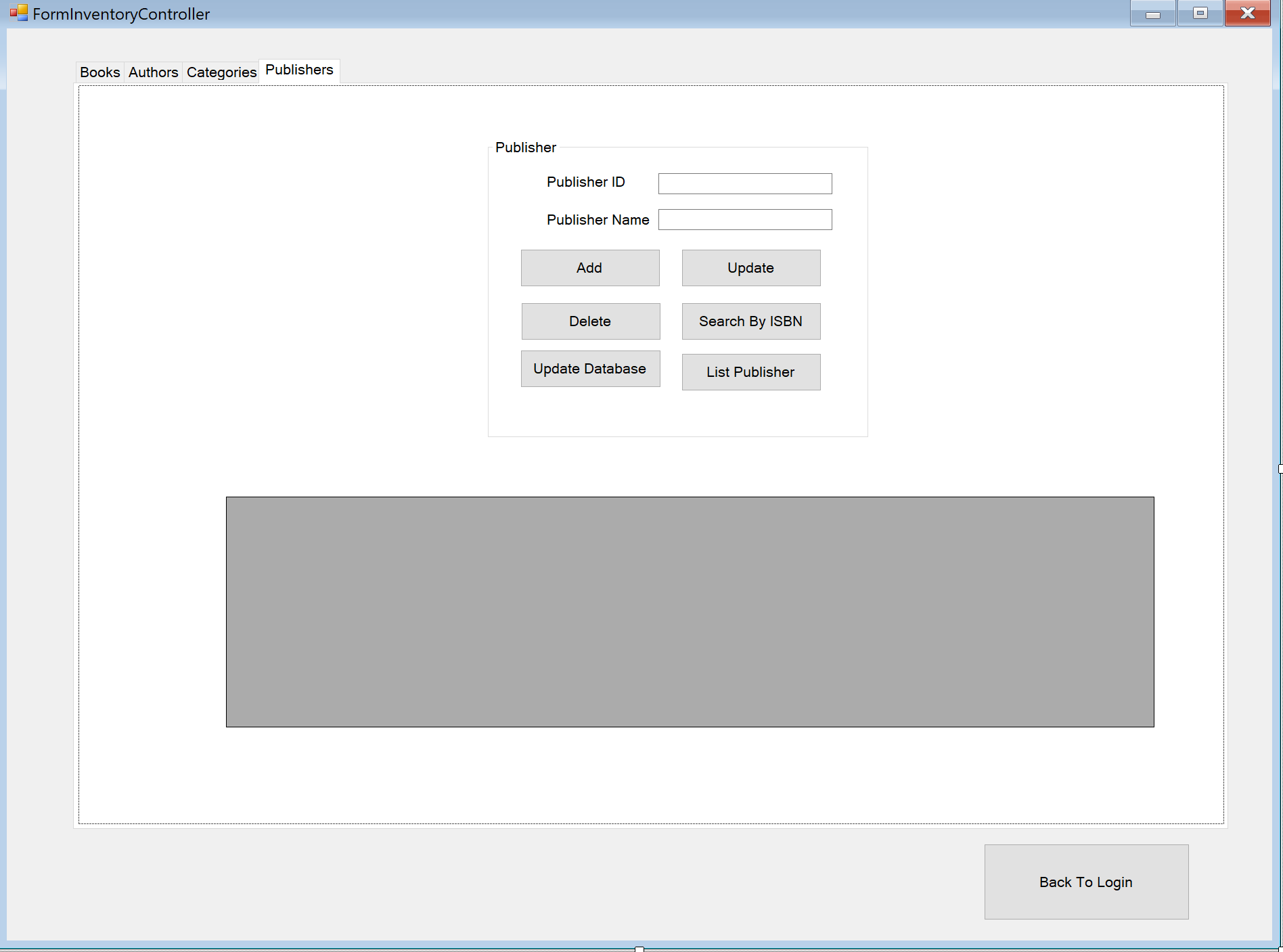
FormOrderClerks-Order 

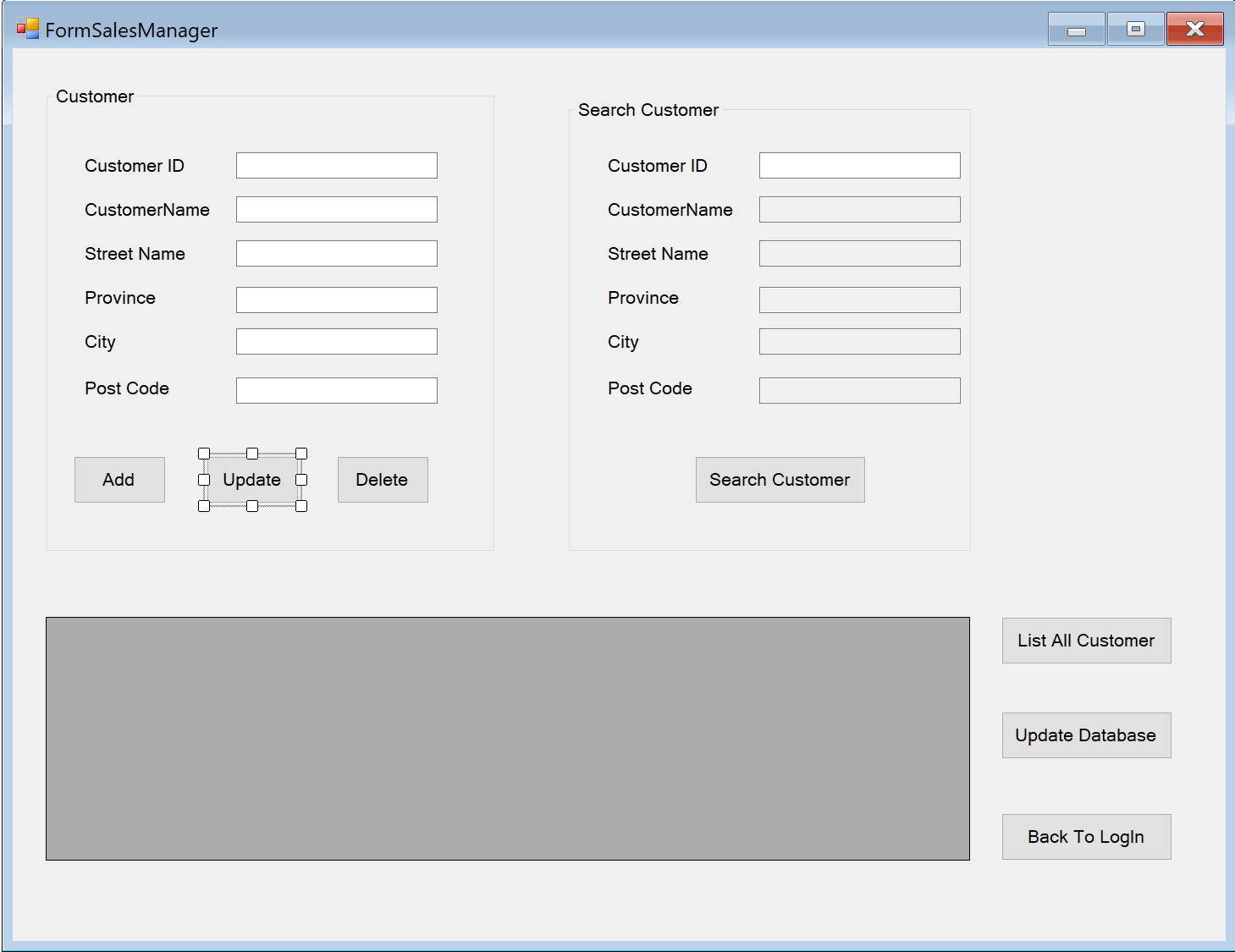
FormOrderClerks-OrderLine

FormInventoryController-Books

FormInventoryController-Authors 

FormInventoryController-Categories 

FormInventoryController-Publishers

FormSalesManager

Design of Data Access Classes

- AuthorDB.cs

- BookDB.cs

- CategoryDB.cs

- CustomerDB.cs

- EmployeeAndUserDB.cs

- PublisherDB.cs

- UtilityDB.cs

Project Implementation

## BLL

### EmployeeAndUser.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using HiTech.DAL;

namespace HiTech.BLL

{

public class Employee\_and\_User

{

private int employeeId;

private string firstName;

private string lastName;

private string phoneNumber;

private string email;

private int jobId;

private string jobTitle;

private int userId;

private string password;

public int EmployeeId { get => employeeId; set => employeeId = value; }

public int JobId { get => jobId; set => jobId = value; }

public string JobTitle { get => jobTitle; set => jobTitle = value; }

public int UserId { get => userId; set => userId = value; }

public string Password { get => password; set => password = value; }

public string FirstName { get => firstName; set => firstName = value; }

public string LastName { get => lastName; set => lastName = value; }

public string PhoneNumber { get => phoneNumber; set => phoneNumber = value; }

public string Email { get => email; set => email = value; }

public void SaveEmployee(Employee\_and\_User emp)

{

Employee\_and\_UserDB.SaveEmployee(emp);

}

public Employee\_and\_User SearchEmployee(int empId)

{

return Employee\_and\_UserDB.SearchEmployee(empId);

}

public void UpdateEmployee(Employee\_and\_User emp)

{

Employee\_and\_UserDB.UpdateEmployee(emp);

}

public List<Employee\_and\_User> ListEmployees()

{

return Employee\_and\_UserDB.ListEmployees();

}

public void DeleteUser(int userId)

{

Employee\_and\_UserDB.DeleteUser(userId);

}

public void DeleteEmployee(int empId)

{

Employee\_and\_UserDB.DeleteEmployee(empId);

}

public List<Employee\_and\_User> listJobs()

{

return Employee\_and\_UserDB.listJobs();

}

public List<Employee\_and\_User> listUsers()

{

return Employee\_and\_UserDB.listUsers();

}

public void SaveUsers(Employee\_and\_User user)

{

Employee\_and\_UserDB.SaveUsers(user);

}

public Employee\_and\_User SearchUser(int empId)

{

return Employee\_and\_UserDB.SearchUser(empId);

}

public void UpdateUser(Employee\_and\_User user)

{

Employee\_and\_UserDB.UpdateUser(user);

}

}

}

### Customer.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using HiTech.DAL;

namespace HiTech.BLL

{

public class Customer

{

private int customerId;

private string customerName;

private string streetName;

private string province;

private string city;

private string postalCode;

public int CustomerId { get => customerId; set => customerId = value; }

public string CustomerName { get => customerName; set => customerName = value; }

public string StreetName { get => streetName; set => streetName = value; }

public string Province { get => province; set => province = value; }

public string City { get => city; set => city = value; }

public string PostalCode { get => postalCode; set => postalCode = value; }

public List<Customer> ListCustomer()

{

return CustomerDB.GetListCustomer();

}

}

}

### Author.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using HiTech.DAL;

namespace HiTech.BLL

{

public class Author

{

private int authorId;

private string authorName;

public int AuthorId { get => authorId; set => authorId = value; }

public string AuthorName { get => authorName; set => authorName = value; }

public List<Author> ListAuthor()

{

return AuthorDB.GetListAuthor();

}

}

}

### Books.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using HiTech.DAL;

namespace HiTech.BLL

{

public class Books

{

private long iSBN;

private string bookTitle;

private int qOH;

private double price;

private int categoryId;

private int authorId;

private int publisherId;

private string yearPublished;

private string edition;

public long ISBN { get => iSBN; set => iSBN = value; }

public string BookTitle { get => bookTitle; set => bookTitle = value; }

public int QOH { get => qOH; set => qOH = value; }

public double Price { get => price; set => price = value; }

public int CategoryId { get => categoryId; set => categoryId = value; }

public int AuthorId { get => authorId; set => authorId = value; }

public int PublisherId { get => publisherId; set => publisherId = value; }

public string YearPublished { get => yearPublished; set => yearPublished = value; }

public string Edition { get => edition; set => edition = value; }

public List<Books> ListBook()

{

return BookDB.GetListBook();

}

}

}

### Category.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using HiTech.DAL;

namespace HiTech.BLL

{

public class Category

{

private int categoryId;

private string categoryName;

public int CategoryId { get => categoryId; set => categoryId = value; }

public string CategoryName { get => categoryName; set => categoryName = value; }

public List<Category> ListCategory()

{

return CategoryDB.GetListCategory();

}

}

}

### Publisher.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using HiTech.DAL;

namespace HiTech.BLL

{

public class Publisher

{

private int publisherId;

private string publisherName;

public int PublisherId { get => publisherId; set => publisherId = value; }

public string PublisherName { get => publisherName; set => publisherName = value; }

public List<Publisher> ListPublisher()

{

return PublisherDB.GetListPublisher();

}

}

}

## DAL

### AuthorDB.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using HiTech.BLL;

using System.Data.SqlClient;

namespace HiTech.DAL

{

public static class AuthorDB

{

public static List<Author> GetListAuthor()

{

List<Author> listAuthor = new List<Author>();

Author author;

using (SqlConnection conn = UtilityDB.ConnectDB())

{

SqlCommand cmdSelect = new SqlCommand("SELECT \* FROM Authors", conn);

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

if (sqlReader.HasRows)

{

while (sqlReader.Read())

{

author = new Author();

author.AuthorId = Convert.ToInt32(sqlReader["AuthorId"]);

author.AuthorName = sqlReader["AuthorName"].ToString();

listAuthor.Add(author);

}

}

else

{

listAuthor = null;

}

}

return listAuthor;

}

}

}

### BookDB.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using HiTech.BLL;

using System.Data.SqlClient;

namespace HiTech.DAL

{

public static class BookDB

{

public static List<Books> GetListBook()

{

List<Books> listBook = new List<Books>();

Books books;

using (SqlConnection conn = UtilityDB.ConnectDB())

{

SqlCommand cmdSelect = new SqlCommand("SELECT \* FROM Books", conn);

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

if (sqlReader.HasRows)

{

while (sqlReader.Read())

{

books = new Books();

books.ISBN = Convert.ToInt64(sqlReader["ISBN"]);

books.BookTitle = sqlReader["BookTitle"].ToString();

books.QOH = Convert.ToInt32(sqlReader["QOH"]);

books.Price = Convert.ToDouble(sqlReader["Price"]);

books.AuthorId = Convert.ToInt32(sqlReader["AuthorId"]);

books.CategoryId = Convert.ToInt32(sqlReader["CategoryId"]);

books.PublisherId = Convert.ToInt32(sqlReader["PublisherId"]);

books.YearPublished = sqlReader["YearPublished"].ToString();

books.Edition = sqlReader["Edition"].ToString();

listBook.Add(books);

}

}

else

{

listBook = null;

}

}

return listBook;

}

}

}

### CategoryDB.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using HiTech.BLL;

using System.Data.SqlClient;

namespace HiTech.DAL

{

public static class CategoryDB

{

public static List<Category> GetListCategory()

{

List<Category> listCategory = new List<Category>();

Category acategory;

using (SqlConnection conn = UtilityDB.ConnectDB())

{

SqlCommand cmdSelect = new SqlCommand("SELECT \* FROM Categories", conn);

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

if (sqlReader.HasRows)

{

while (sqlReader.Read())

{

acategory = new Category();

acategory.CategoryId = Convert.ToInt32(sqlReader["CategoryId"]);

acategory.CategoryName = sqlReader["CategoryName"].ToString();

listCategory.Add(acategory);

}

}

else

{

listCategory = null;

}

}

return listCategory;

}

}

}

### CustomerDB.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using HiTech.BLL;

using System.Data.SqlClient;

namespace HiTech.DAL

{

public static class CustomerDB

{

public static List<Customer> GetListCustomer()

{

List<Customer> listCustomer = new List<Customer>();

Customer customer;

using (SqlConnection conn = UtilityDB.ConnectDB())

{

SqlCommand cmdSelect = new SqlCommand("SELECT \* FROM Customers", conn);

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

if (sqlReader.HasRows)

{

while (sqlReader.Read())

{

customer = new Customer();

customer.CustomerId = Convert.ToInt32(sqlReader["CustomerId"]);

customer.CustomerName = sqlReader["CustomerName"].ToString();

customer.Province = sqlReader["Province"].ToString();

customer.StreetName = sqlReader["StreetName"].ToString();

customer.City = sqlReader["City"].ToString();

customer.PostalCode = sqlReader["PostalCode"].ToString();

listCustomer.Add(customer);

}

}

else

{

listCustomer = null;

}

}

return listCustomer;

}

}

}

### EmployeeAndUserDB.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using HiTech.BLL;

using System.Data.SqlClient;

namespace HiTech.DAL

{

public static class Employee\_and\_UserDB

{

public static void SaveEmployee(Employee\_and\_User emp)

{

SqlConnection conn = UtilityDB.ConnectDB();

SqlCommand cmdInsert = new SqlCommand();

cmdInsert.CommandText = "INSERT INTO Employees(EmployeeId,FirstName,LastName,PhoneNumber,Email,JobId) " +

" VALUES (@EmployeeId,@FirstName,@LastName,@PhoneNumber,@Email,@JobId)";

cmdInsert.Parameters.AddWithValue("@EmployeeId", emp.EmployeeId);

cmdInsert.Parameters.AddWithValue("@FirstName", emp.FirstName);

cmdInsert.Parameters.AddWithValue("@LastName", emp.LastName);

cmdInsert.Parameters.AddWithValue("@PhoneNumber", emp.PhoneNumber);

cmdInsert.Parameters.AddWithValue("@Email", emp.Email);

cmdInsert.Parameters.AddWithValue("@JobId", emp.JobId);

cmdInsert.Connection = conn;

cmdInsert.ExecuteNonQuery();

conn.Close();

}

public static Employee\_and\_User SearchEmployee(int Id)

{

Employee\_and\_User emp = new Employee\_and\_User();

SqlConnection conn = UtilityDB.ConnectDB();

conn = UtilityDB.ConnectDB();

SqlCommand cmdSelect = new SqlCommand();

cmdSelect.CommandText = "SELECT \* FROM Employees " +

"WHERE EmployeeId = @EmployeeId";

cmdSelect.Parameters.AddWithValue("@EmployeeId", Id);

cmdSelect.Connection = conn;

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

if (sqlReader.Read())

{

emp.EmployeeId = Convert.ToInt32(sqlReader["EmployeeId"]);

emp.FirstName = sqlReader["FirstName"].ToString();

emp.LastName = sqlReader["LastName"].ToString();

emp.PhoneNumber = sqlReader["PhoneNumber"].ToString();

emp.Email = sqlReader["Email"].ToString();

emp.JobId = Convert.ToInt32(sqlReader["JobId"]);

}

else

{

emp = null;

}

return emp;

}

public static List<Employee\_and\_User> ListEmployees()

{

List<Employee\_and\_User> listEmp = new List<Employee\_and\_User>();

SqlConnection conn = UtilityDB.ConnectDB();

conn = UtilityDB.ConnectDB();

SqlCommand cmdSelect = new SqlCommand();

cmdSelect.CommandText = "SELECT \* FROM Employees ";

cmdSelect.Connection = conn;

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

Employee\_and\_User emp;

while (sqlReader.Read())

{

emp = new Employee\_and\_User();

emp.EmployeeId = Convert.ToInt32(sqlReader["EmployeeId"]);

emp.FirstName = sqlReader["FirstName"].ToString();

emp.LastName = sqlReader["LastName"].ToString();

emp.PhoneNumber = sqlReader["PhoneNumber"].ToString();

emp.Email = sqlReader["Email"].ToString();

emp.JobId = Convert.ToInt32(sqlReader["JobId"]);

listEmp.Add(emp);

}

return listEmp;

}

public static List<Employee\_and\_User> listJobs()

{

List<Employee\_and\_User> listjob = new List<Employee\_and\_User>();

SqlConnection conn = UtilityDB.ConnectDB();

conn = UtilityDB.ConnectDB();

SqlCommand cmdSelect = new SqlCommand();

cmdSelect.CommandText = "SELECT \* FROM Job ";

cmdSelect.Connection = conn;

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

Employee\_and\_User emp;

while (sqlReader.Read())

{

emp = new Employee\_and\_User();

emp.JobId = Convert.ToInt32(sqlReader["JobId"]);

emp.JobTitle = sqlReader["JobTitle"].ToString();

listjob.Add(emp);

}

return listjob;

}

public static List<Employee\_and\_User> listUsers()

{

List<Employee\_and\_User> listUser = new List<Employee\_and\_User>();

SqlConnection conn = UtilityDB.ConnectDB();

conn = UtilityDB.ConnectDB();

SqlCommand cmdSelect = new SqlCommand();

cmdSelect.CommandText = "SELECT \* FROM Users ";

cmdSelect.Connection = conn;

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

Employee\_and\_User user;

while (sqlReader.Read())

{

user = new Employee\_and\_User();

user.UserId = Convert.ToInt32(sqlReader["UserId"]);

user.Password = sqlReader["Password"].ToString();

user.EmployeeId = Convert.ToInt32(sqlReader["EmployeeId"]);

listUser.Add(user);

}

return listUser;

}

public static void SaveUsers(Employee\_and\_User user)

{

SqlConnection conn = UtilityDB.ConnectDB();

SqlCommand cmdInsert = new SqlCommand();

cmdInsert.CommandText = "INSERT INTO Users(UserId,Password,EmployeeId) " +

" VALUES (@UserId,@Password,@EmployeeId)";

cmdInsert.Parameters.AddWithValue("@UserId", user.UserId);

cmdInsert.Parameters.AddWithValue("@Password", user.Password);

cmdInsert.Parameters.AddWithValue("@EmployeeId", user.EmployeeId);

cmdInsert.Connection = conn;

cmdInsert.ExecuteNonQuery();

conn.Close();

}

public static Employee\_and\_User SearchUser(int Id)

{

Employee\_and\_User user = new Employee\_and\_User();

SqlConnection conn = UtilityDB.ConnectDB();

conn = UtilityDB.ConnectDB();

SqlCommand cmdSelect = new SqlCommand();

cmdSelect.CommandText = "SELECT \* FROM Users " +

"WHERE UserId = @UserId";

cmdSelect.Parameters.AddWithValue("@UserId", Id);

cmdSelect.Connection = conn;

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

if (sqlReader.Read())

{

user.UserId = Convert.ToInt32(sqlReader["UserId"]);

user.Password = sqlReader["Password"].ToString();

user.EmployeeId = Convert.ToInt32(sqlReader["EmployeeId"]);

}

else

{

user = null;

}

return user;

}

public static void UpdateEmployee(Employee\_and\_User emp)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdUpate = new SqlCommand();

cmdUpate.CommandText = "UPDATE Employees " +

"SET EmployeeId = @EmployeeId," +

" FirstName = @FirstName," +

" LastName = @LastName," +

" PhoneNumber = @PhoneNumber," +

" Email = @Email," +

" JobId = @JobId " +

"WHERE EmployeeId = @EmployeeId";

cmdUpate.Parameters.AddWithValue("@EmployeeId", emp.EmployeeId);

cmdUpate.Parameters.AddWithValue("@FirstName", emp.FirstName);

cmdUpate.Parameters.AddWithValue("@LastName", emp.LastName);

cmdUpate.Parameters.AddWithValue("@PhoneNumber", emp.PhoneNumber);

cmdUpate.Parameters.AddWithValue("@Email", emp.Email);

cmdUpate.Parameters.AddWithValue("@JobId", emp.JobId);

cmdUpate.Connection = connDB;

cmdUpate.ExecuteNonQuery();

connDB.Close();

}

public static void UpdateUser(Employee\_and\_User user)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdUpate1 = new SqlCommand();

cmdUpate1.CommandText = "UPDATE Users " +

"SET UserId = @UserId," +

" Password = @Password," +

" EmployeeId = @EmployeeId" +

"WHERE UserId = @UserId";

cmdUpate1.Parameters.AddWithValue("@UserId", user.UserId);

cmdUpate1.Parameters.AddWithValue("@Password", user.Password);

cmdUpate1.Parameters.AddWithValue("@EmployeeId", user.EmployeeId);

cmdUpate1.Connection = connDB;

cmdUpate1.ExecuteNonQuery();

connDB.Close();

}

public static void DeleteUser(int userId)

{

SqlConnection conn = UtilityDB.ConnectDB();

SqlCommand cmdDelete = new SqlCommand();

cmdDelete.CommandText = "DELETE FROM Users " +

"WHERE UserId= @UserId";

cmdDelete.Parameters.AddWithValue("@UserId", userId);

cmdDelete.Connection = conn;

cmdDelete.ExecuteNonQuery();

conn.Close();

}

public static void DeleteEmployee(int empId)

{

SqlConnection conn = UtilityDB.ConnectDB();

SqlCommand cmdDelete = new SqlCommand();

cmdDelete.CommandText = "DELETE FROM Employees " +

"WHERE EmployeeId= @EmployeeId";

cmdDelete.Parameters.AddWithValue("@EmployeeId", empId);

cmdDelete.Connection = conn;

cmdDelete.ExecuteNonQuery();

conn.Close();

}

}

}

### PublisherDB.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using HiTech.BLL;

using System.Data.SqlClient;

namespace HiTech.DAL

{

public static class PublisherDB

{

public static List<Publisher> GetListPublisher()

{

List<Publisher> listPublisher = new List<Publisher>();

Publisher apublisher;

using (SqlConnection conn = UtilityDB.ConnectDB())

{

SqlCommand cmdSelect = new SqlCommand("SELECT \* FROM Publishers", conn);

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

if (sqlReader.HasRows)

{

while (sqlReader.Read())

{

apublisher = new Publisher();

apublisher.PublisherId = Convert.ToInt32(sqlReader["PublisherId"]);

apublisher.PublisherName = sqlReader["PublisherName"].ToString();

listPublisher.Add(apublisher);

}

}

else

{

listPublisher = null;

}

}

return listPublisher;

}

}

}

### Utility.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SqlClient;

using System.Configuration;

namespace HiTech.DAL

{

public static class UtilityDB

{

public static SqlConnection ConnectDB()

{

SqlConnection conn = new SqlConnection();

conn.ConnectionString = ConfigurationManager.ConnectionStrings["HiTechDBConnection"].ConnectionString;

conn.Open();

return conn;

}

}

}

# Project Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Users |  | | Operations | Result | Note |
| Login | Login with different user ID | Success |  |
| **MIS Manager**  User Id: 1111  Password: henrybrown | Add employee | Success |  |
|  | Update employee | Success |  |
|  | Delete employee | Success |  |
|  | List employee | Success |  |
|  | Search employee | Success |  |
|  | Add user | Success |  |
|  | Update user | Fail | 'The parameterized query '(@EmployeeId int,@FirstName nvarchar(4000),@LastName nvarchar(40' expects the parameter '@FirstName', which was not supplied.' |
|  | Delete user | Success |  |
|  | List user | Success |  |
|  | Search user | Success |  |
|  | List job user | Success |  |
| **Sales Manager**  User Id: 2222  Password: thomasmoore | Add customer | Success |  |
|  | Update customer | Success |  |
|  | Delete customer | Success |  |
|  | Search customer | Success |  |
|  | List customer | Success |  |
|  | Update database | Success |  |
| **Inventory Controller**  User Id: 3333  Password: peterwang | Add books | Success |  |
|  | Update books | Success |  |
|  | Delete books | Success |  |
|  | Search books | Success |  |
|  | List books | Success |  |
|  | Add authors | Success |  |
|  | Update authors | Success |  |
|  | Delete authors | Success |  |
|  | Search authors | Success |  |
|  | List authors | Success |  |
|  | Add categories | Success |  |
|  | Update categories | Success |  |
|  | Delete categories | Success |  |
|  | Search categories | Success |  |
|  | List categories | Success |  |
|  | Add publishers | Success |  |
|  | Update publishers | Success |  |
|  | Delete publishers | Success |  |
|  | Search publishers | Success |  |
|  | List publishers | Success |  |
|  | Update database | Success |  |
| **Order Clerks**  User Id: 4444  Password: marybrown  -------------------  User Id: 5555  Password: jenniferbouchard | Add orders | Success | Even all can work, but there is a problem. Table Orders only have OrderId, Orderdate, customerId and EmployeeId, but when I list the order, there are three other columns in the table that is not include in table Orders (customer, employee and orderline) |
|  | Update orders | Success |  |
|  | Delete orders | Success |  |
|  | List orders | Success |  |
|  | Search orders | Success |  |
|  | Add orderlines | Fail | For this part, I finish the most of code, but I get some problems, I search a lot of information about entity, but I still can not fix it. I think problem is about the PK. orderliness has two PK, but when I do this part, it always says I am missing one PK. |
|  | Update orderlines | Fail |  |
|  | Delete orderlines | Fail |  |
|  | List orderlines | Fail |  |
|  | Search orderlines | Fail |  |

# Conclusion

In this course, I learned a lot in this semester. At the beginning of the semester, I did not understand the relationship between BLL and DAL, why I need to create these two folders. But as the course progressed, I understood. I used the connection mode for the first part of the project. Although I get some bugs at the beginning, I think it because I do not understand the relationship between the table in databases. But you helped us to design the database, which helped me a lot. In the second and third part, I used the disconnected mode. In dataset, I understand better than connect mode. I understand the relationship between the tables, and I know it needs to update the database after adding the new data. In the fourth part, I used the entity framework mode. I did not understand this part very well. I searched a lot of information to help me complete this part, but there are still some bugs that I cannot deal with. Nevertheless, I still complete the project. In this semester, I learn a lot, the most important is I need to understand the requirements of users. As a programmer, I cannot just stand on my side and think about problems. I should learn how to make users have a better experience. In the final presentation, you explained this problem to me, it gives me a lot of help. I believe this will be helpful in my future study and work.