BookBiz Desktop Application Report

Xiao Feng Huang

2195414

Desktop Development

Department: Faculty of Science and Technology, Computer Science Department

Teacher: Alex Vilvet

Author Contact: hawkhxf2000@gmail.com

Abstract

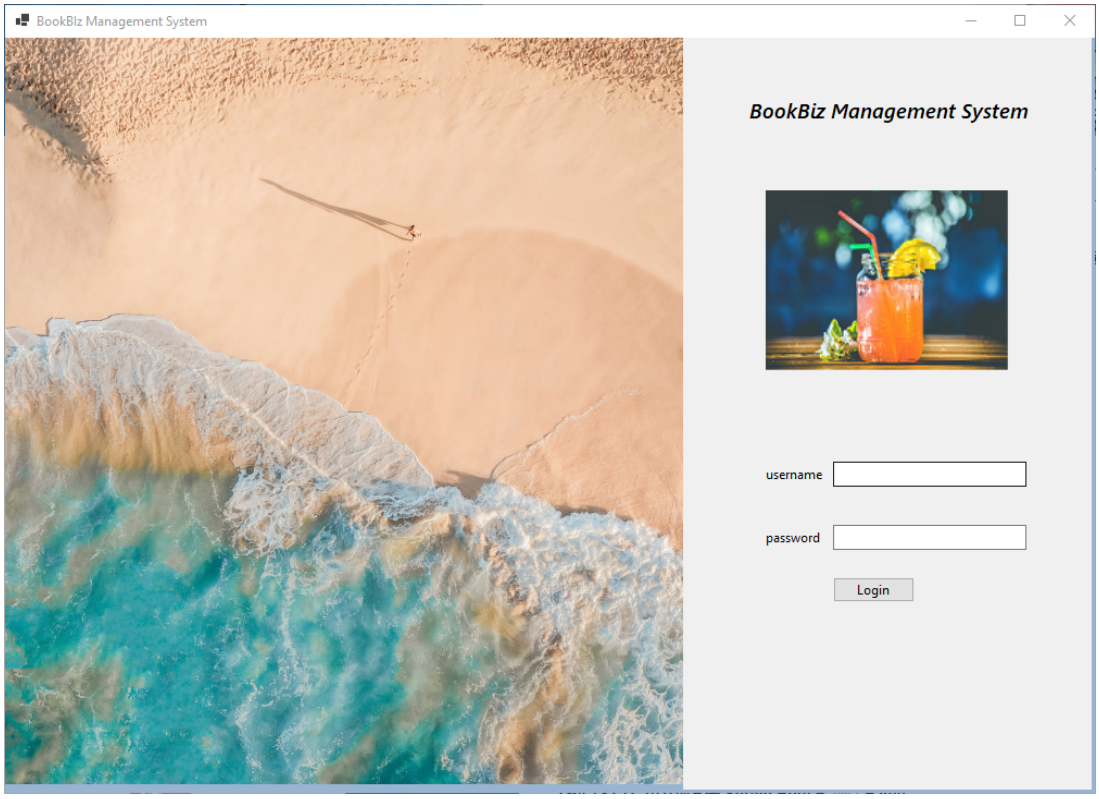
This project is the final project for the Desktop Development course. It is a simple book store management system and includes four user groups and different workflows and functions for them. Its user interface is designed by winForm and programming by C#. It basically implements add/search/list functions of user group/employee/client/book/order.

Keywords: book store, Management system, desktop application, c#

Project Description

1. UI design

Login page: summer theme with beach and nice cocktail wine to bring a refresh and relax feeling to user

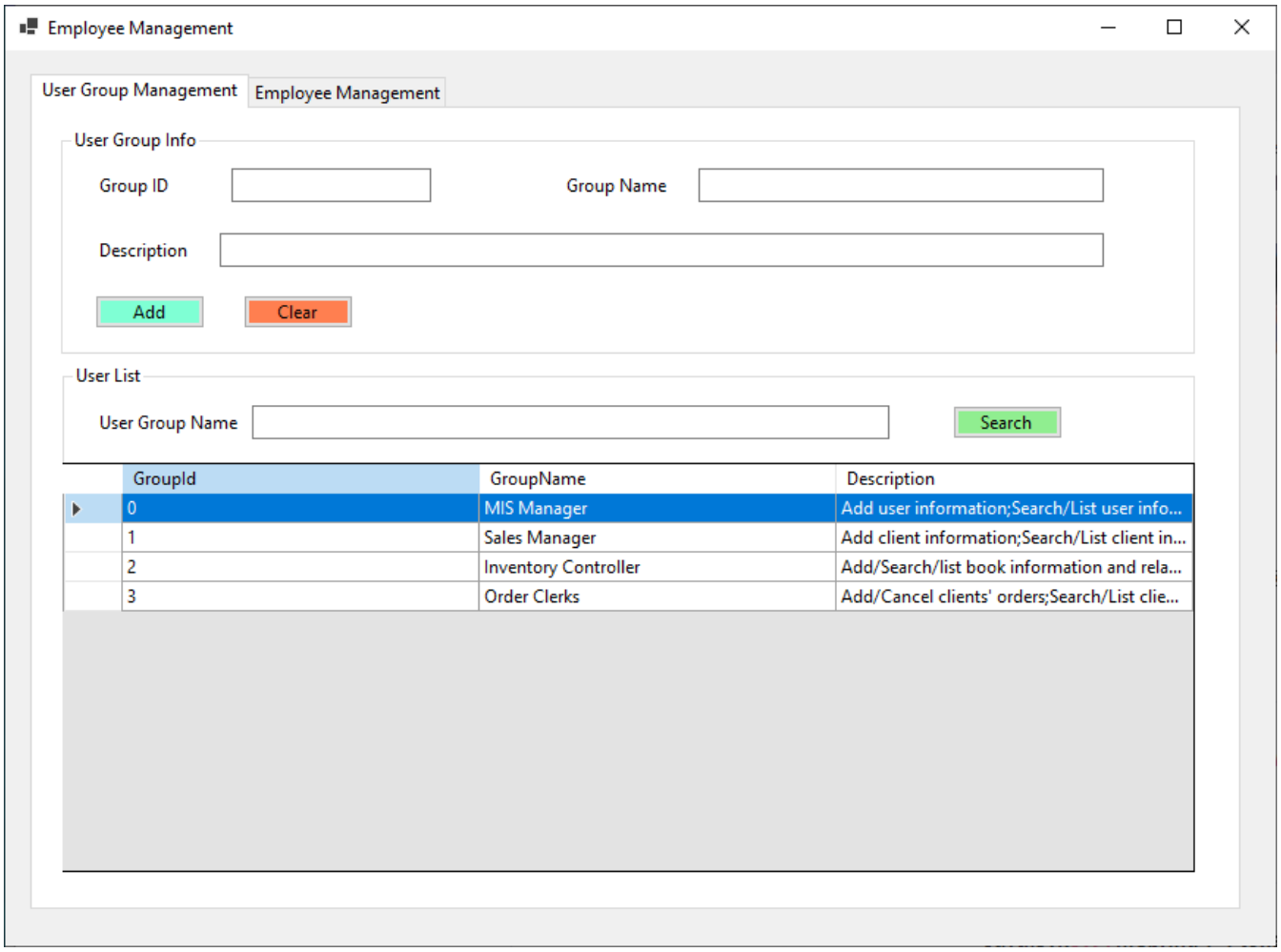


After Login with different username and password, UI will redirect to different page according the the user group which the user belongs to

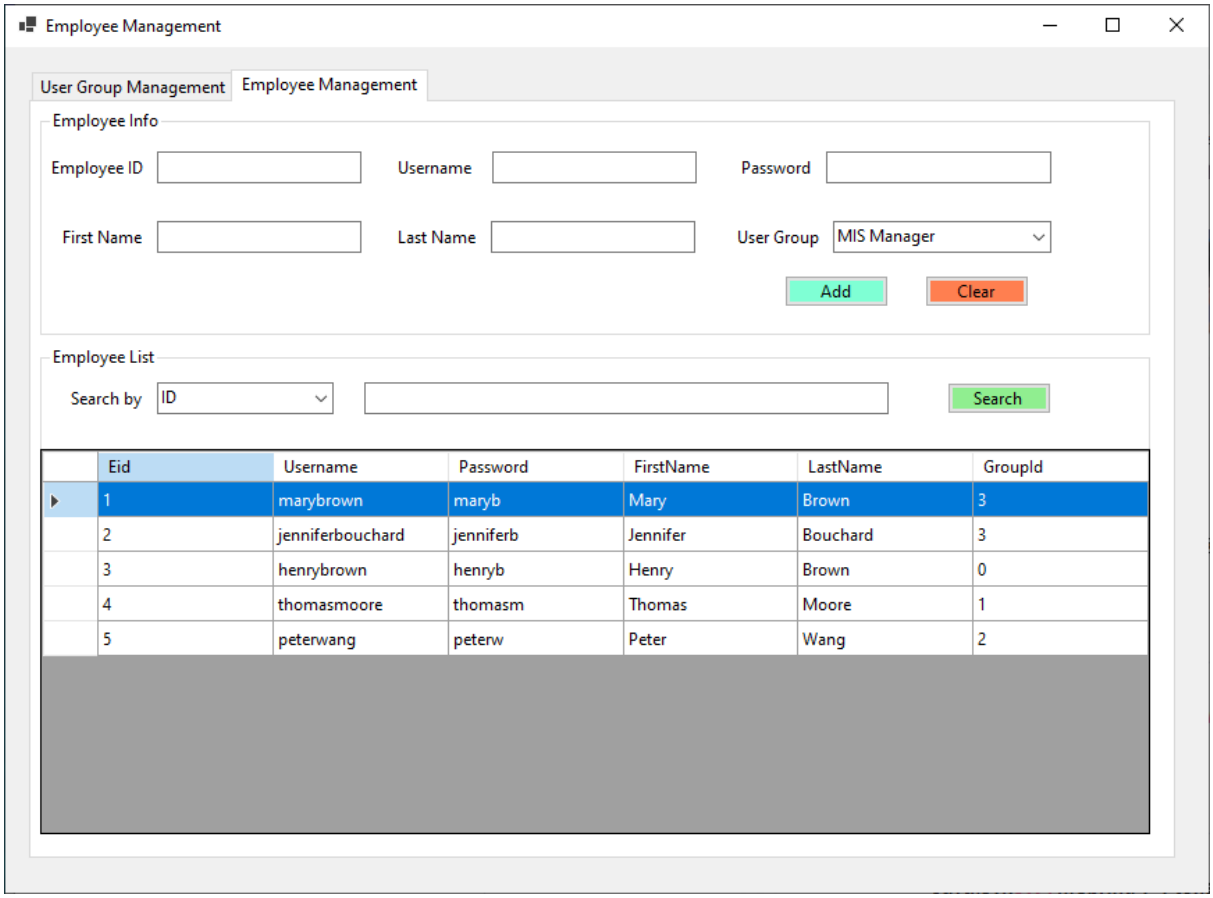
1.1 MIS manager

This page has two tabs for different management functions: user group and employee. Different buttons with different background colors to emphasize the functions.

User Group Management

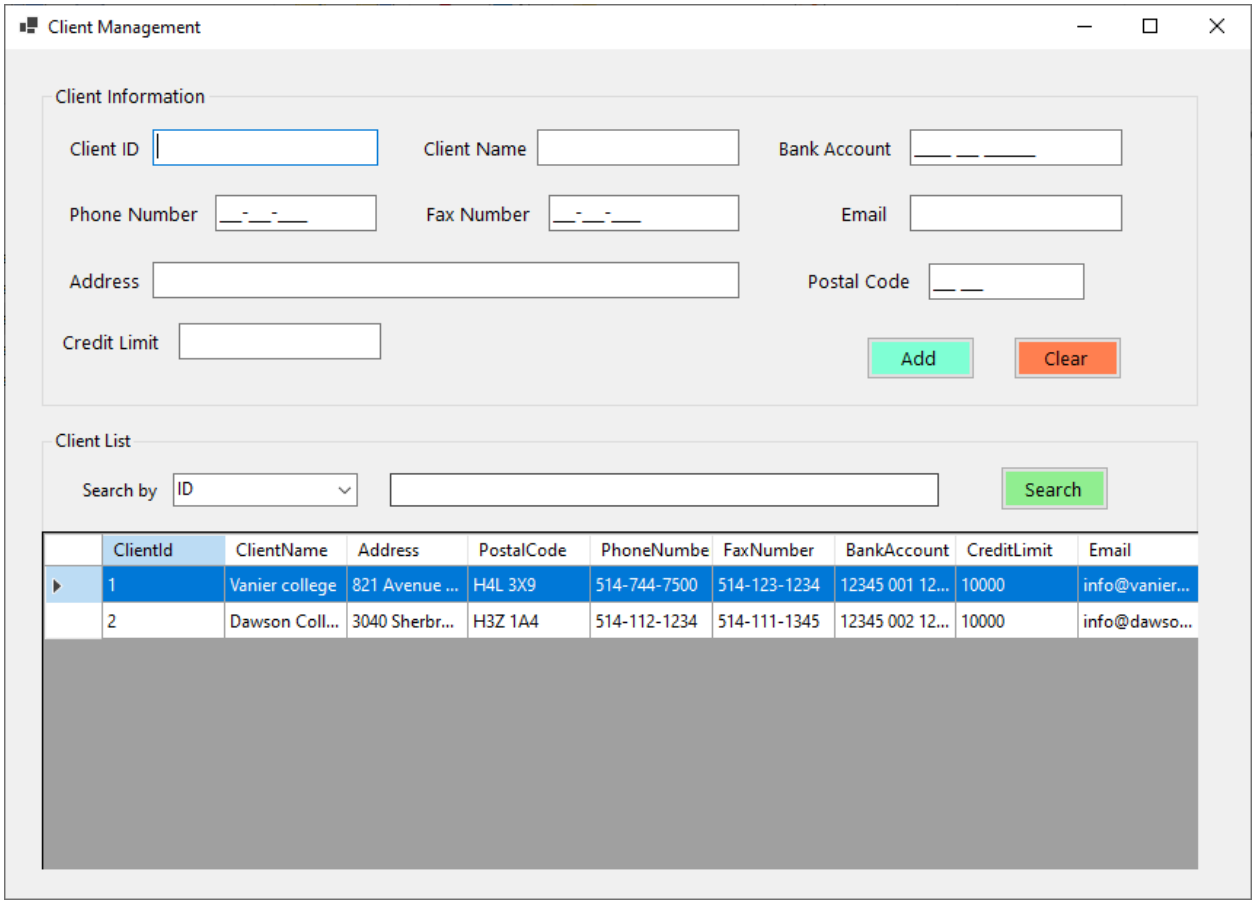


Employee Management



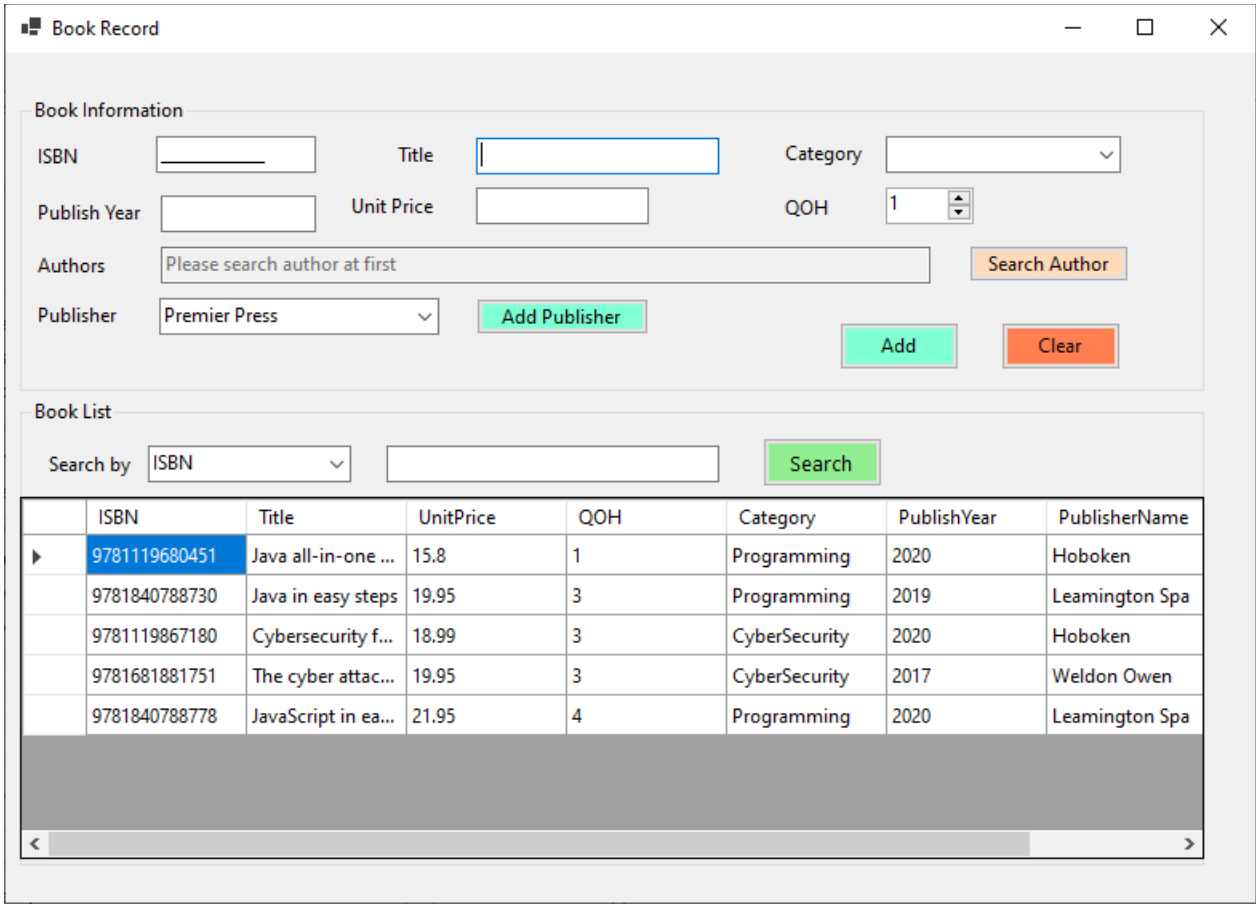
1.2 Sales Manager

The main function of Sales manager is to manage client. So the main function is add/search/list function of client. Search criteria can be client ID, client name, phone number and fax number



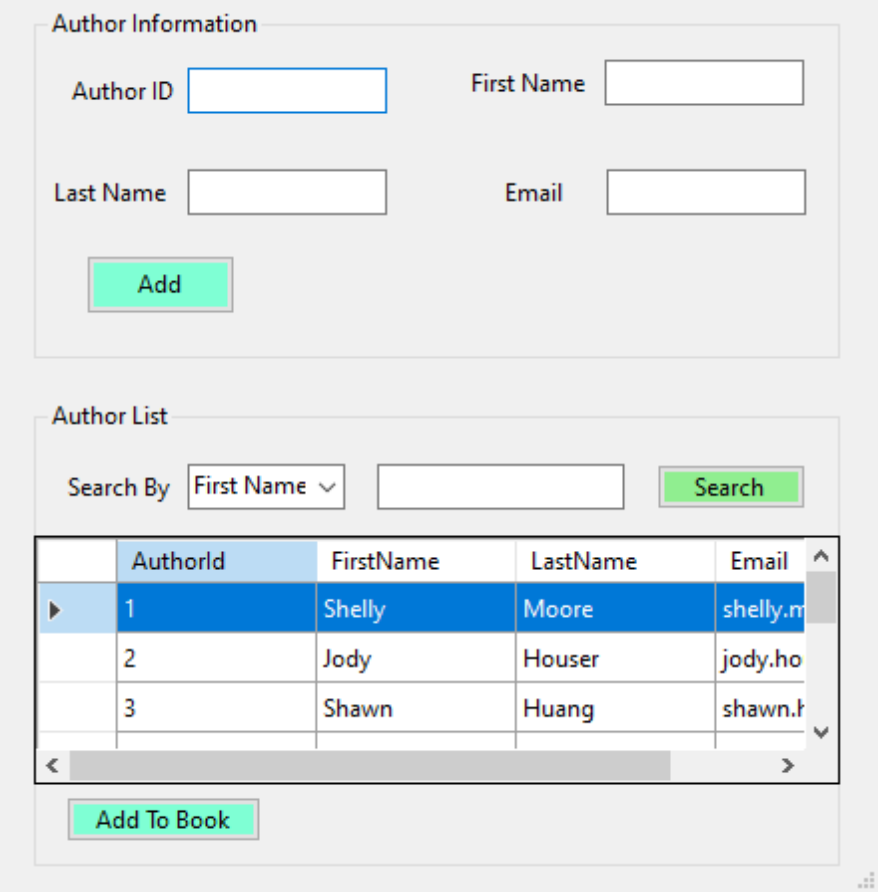
1.3 Inventory controller

Main function of this user group is to add/search/list the book and related information. So beside the basic add/clear/search function, two new function: “Search Author” and “Add publisher” is added and will show different child forms once button is click



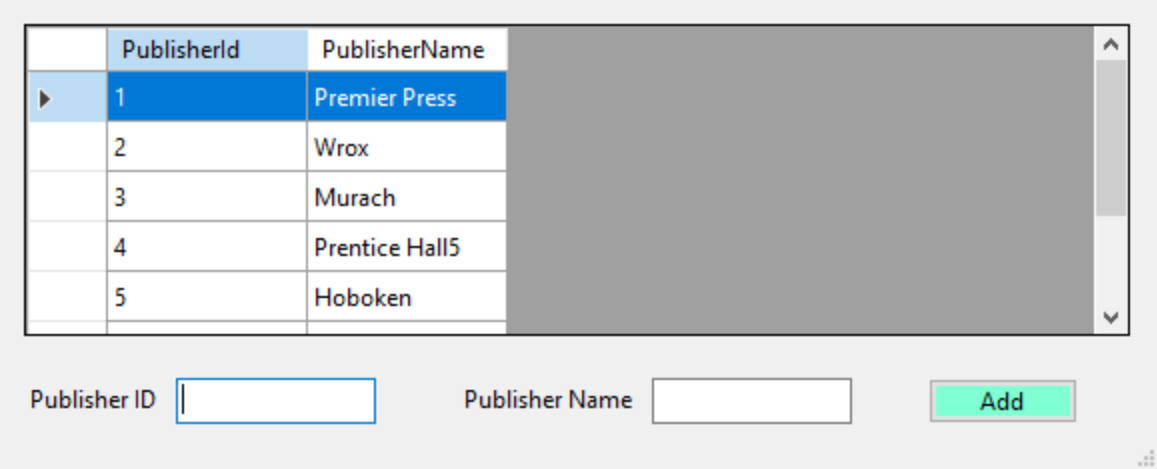
1.3.1 Search Author

In this form, you can search the author from the list, once the author is found, click the "Add To Book" to add the author to BookAuthor.dat file and also the author list on the book page. If the author is not found, a new author can be added to the Author.dat file and also the author list on the book page.



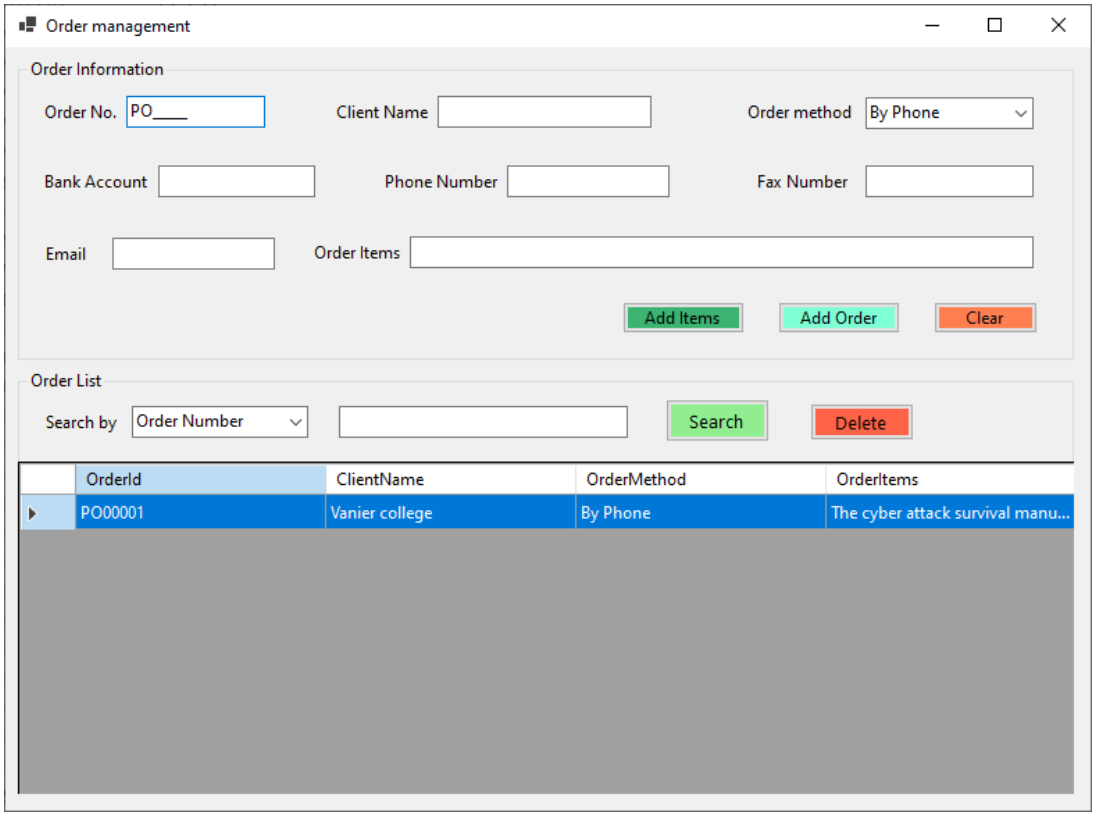
1.3.2 Add Publisher page

If a publisher can't be found in the combobox of the publisher, a new publisher can be added to the publisher.dat file and also shown on the book page.



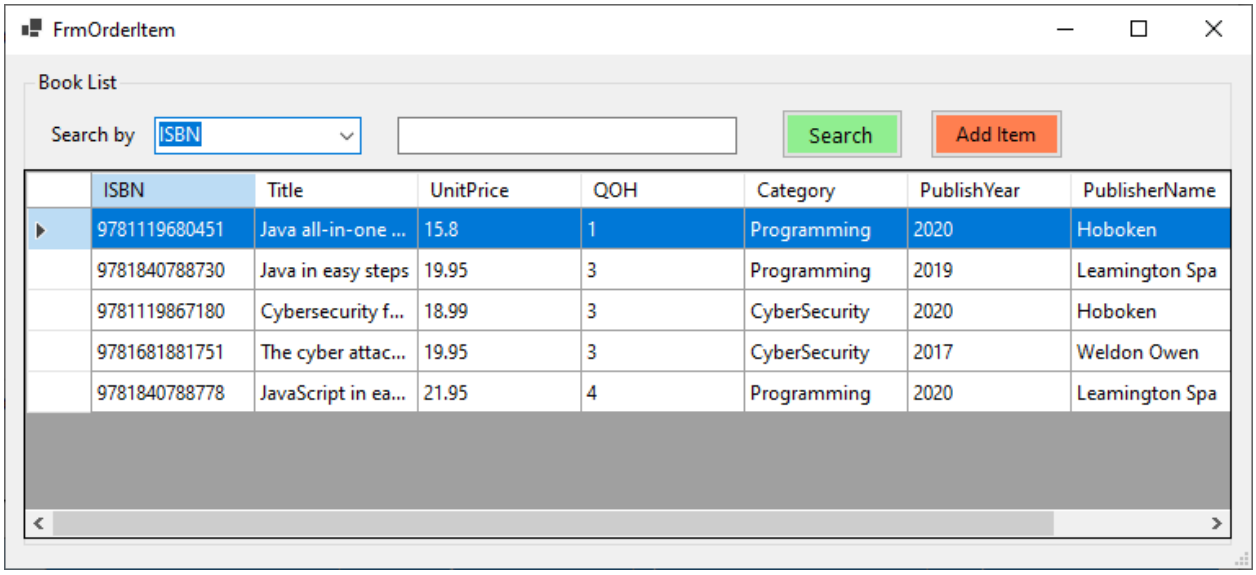
1.4 Order Management

The operation of order clerk is add/cancel and search/list order. So here a new “Add Items” function is added.



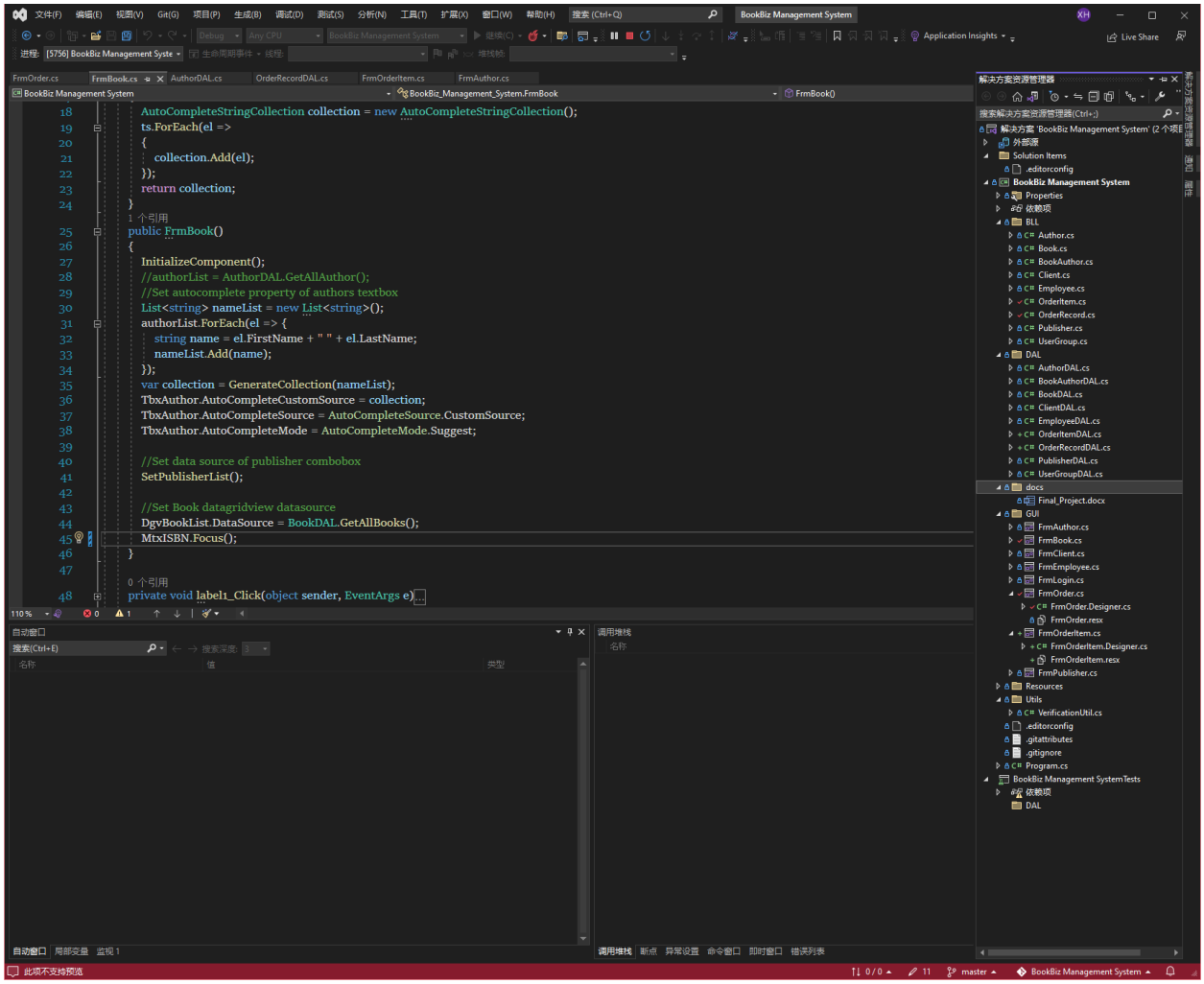
1.4.1 Add Items page

Books can be searched by ISBN, Title, author, and click “Add Item” to add items to the list. The multi-line of items can be chosen. Here I should add quantity for each item but I don't have time to finish it and it will be done later.



2. Code design

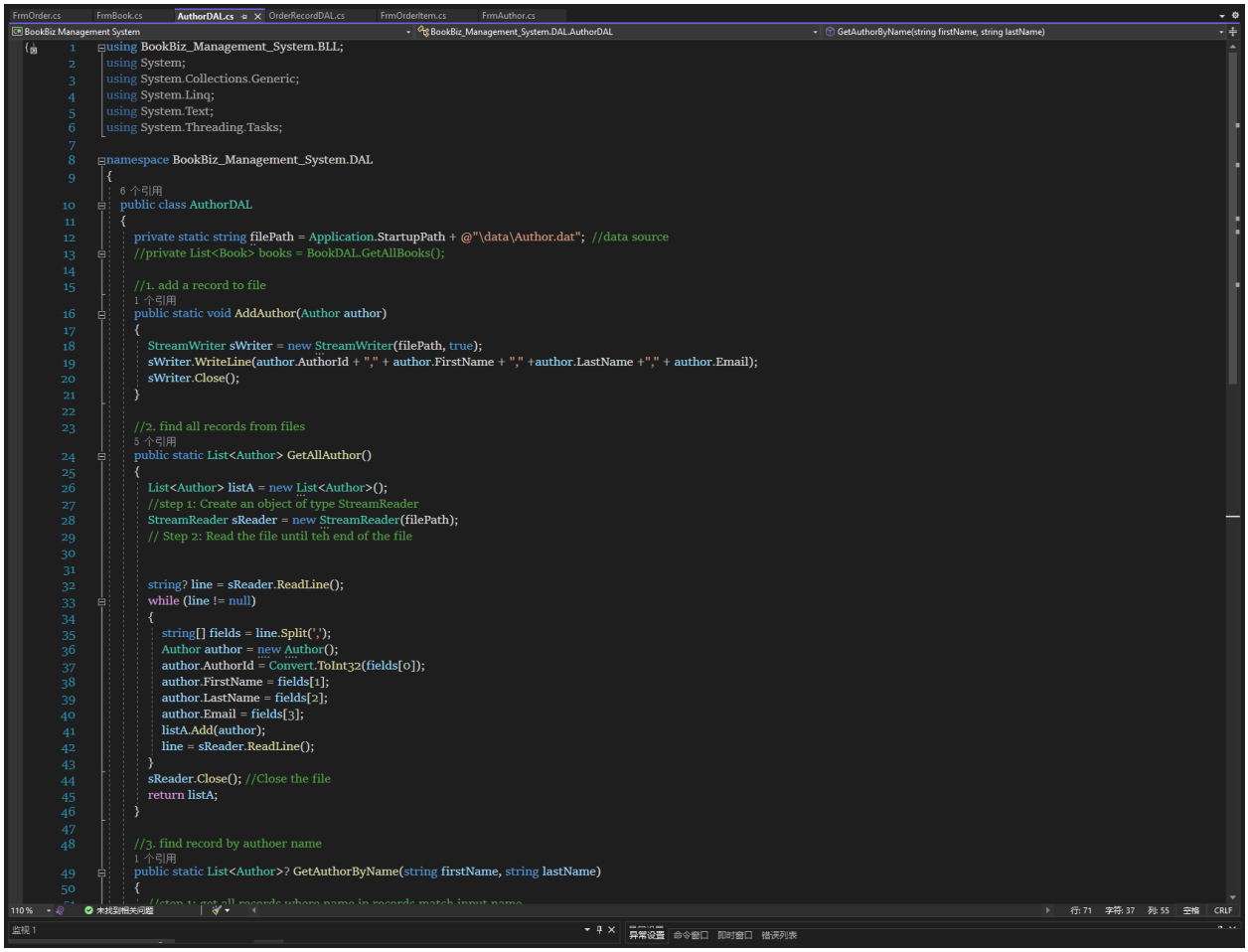
The programming language is C# and software architecture is MVC. There are three layers created: BLL for entity class, DAL for the manipulation of data, and GUI for the business logic and user interface. There are 9 entity classes designed, including 7 data entities and 2 middle entities corresponding to many-to-many relationships. Each data entity have their corresponding DAL implementation class.



3. Implementation

3.1 DAL class

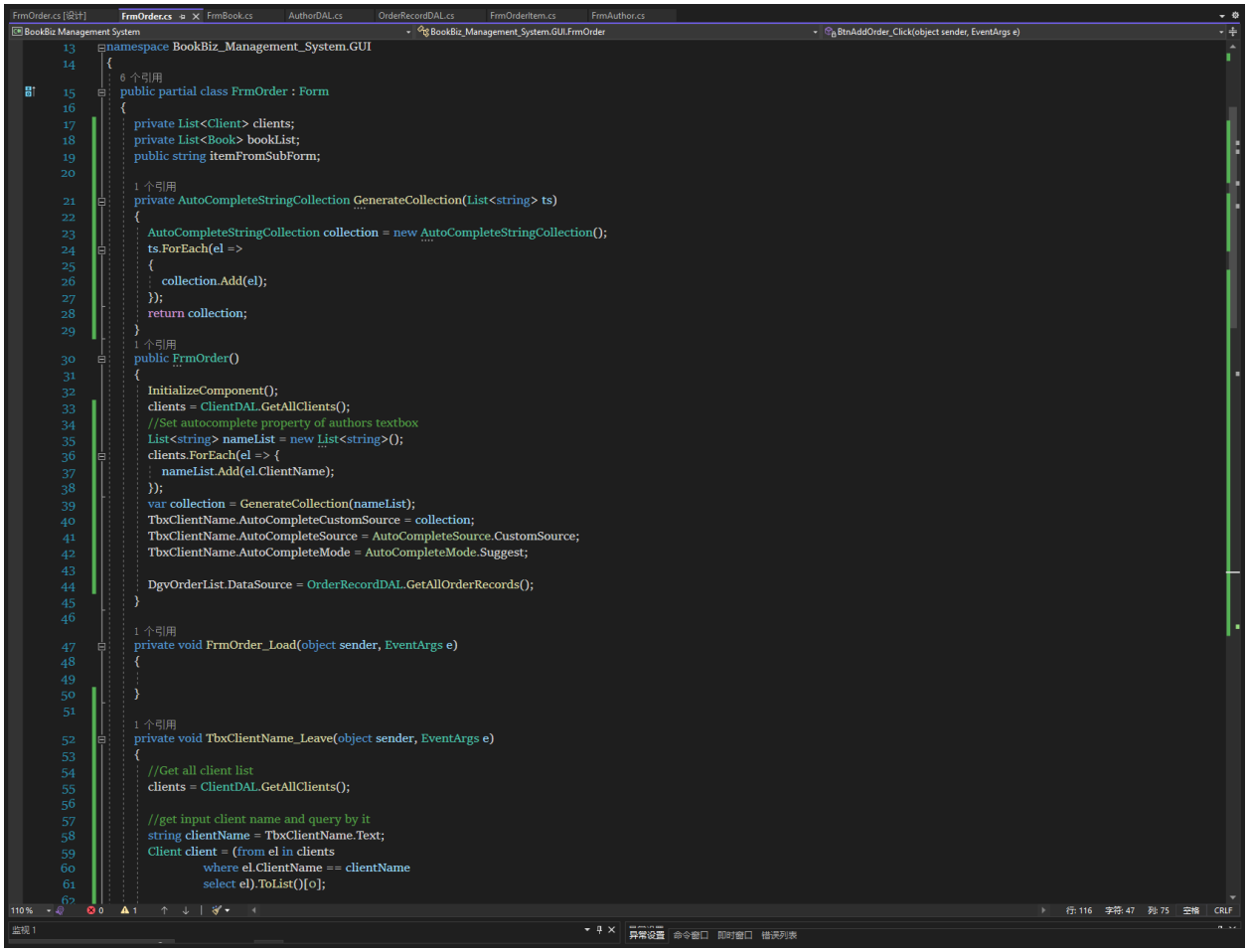
In each DAL class, the basic Add, Get all, and get entity by different search criteria dynamically. It's all about the manipulation of data, any processing of data will be done in the GUI layer.



```
using BookBiz_Management_System.BLL;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BookBiz_Management_System.DAL
{
    6 个引用
    public class AuthorDAL
    {
        private static string filePath = Application.StartupPath + @"\data\Author.dat";  //data source
        //private List<Book> books = BookDAL.GetAllBooks();

        //1. add a record to file
        1 个引用
        public static void AddAuthor(Author author)
        {
            StreamWriter sWriter = new StreamWriter(filePath, true);
            sWriter.WriteLine(author.AuthorId + "," + author.FirstName + "," +author.LastName +"," + author.Email);
            sWriter.Close();
        }

        //2. find all records from files
        5 个引用
        public static List<Author> GetAllAuthor()
        {
            List<Author> listA = new List<Author>();
            //step 1: Create an object of type StreamReader
            StreamReader sReader = new StreamReader(filePath);
            // Step 2: Read the file until teh end of the file


            string? line = sReader.ReadLine();
            while (line != null)
            {
                string[] fields = line.Split(',');
                Author author = new Author();
                author.AuthorId = Convert.ToInt32(fields[0]);
                author.FirstName = fields[1];
                author.LastName = fields[2];
                author.Email = fields[3];
                listA.Add(author);
                line = sReader.ReadLine();
            }
            sReader.Close(); //Close the file
            return listA;
        }

        //3. find record by authoer name
        1 个引用
        public static List<Author>? GetAuthorByName(string firstName, string lastName)
        {
```

3.2 GUI layer

All business logic, including add/search/list function and user interaction are included in this layer. Some forms involve data passing between parent/child forms. I use the instance of parent form as a parameter of the child form's constructor and the public properties of parent form to pass the data, it works fine. Due to the limit of time and skill, I didn't implement the dynamic append of components, such as new textbox, listview items. I will try to do that later.



```
namespace BookBiz_Management_System.GUI
{
    6 个引用
    public partial class FrmOrder : Form
    {
        private List<Client> clients;
        private List<Book> bookList;
        public string itemFromSubForm;

        1 个引用
        private AutoCompleteStringCollection GenerateCollection(List<string> ts)
        {
            AutoCompleteStringCollection collection = new AutoCompleteStringCollection();
            ts.ForEach(el =>
            {
                collection.Add(el);
            });
            return collection;
        }
        1 个引用
        public FrmOrder()
        {
            InitializeComponent();
            clients = ClientDAL.GetAllClients();
            //Set autocomplete property of authors textbox
            List<string> nameList = new List<string>();
            clients.ForEach(el => {
                nameList.Add(el.ClientName);
            });
            var collection = GenerateCollection(nameList);
            TbxClientName.AutoCompleteCustomSource = collection;
            TbxClientName.AutoCompleteSource = AutoCompleteSource.CustomSource;
            TbxClientName.AutoCompleteMode = AutoCompleteMode.Suggest;

            DgvOrderList.DataSource = OrderRecordDAL.GetAllOrderRecords();
        }

        1 个引用
        private void FrmOrder_Load(object sender, EventArgs e)
        {


        }

        1 个引用
        private void TbxClientName_Leave(object sender, EventArgs e)
        {
            //Get all client list
            clients = ClientDAL.GetAllClients();

            //get input client name and query by it
            string clientName = TbxClientName.Text;
            Client client = (from el in clients
                             where el.ClientName == clientName
                             select el).ToList()[0];
```

Conclusion

This project is very good for practice of desktop application development. The MVC architecture and file as data source are applied in this project. Verification, regex, collection class and passing data between parent/child forms is also used. All basic requirements are met except some modifications need to be done later due to the limit of time. Through this project, I am more familiar with C# syntax, function, and the operation of VS. And It's very lucky to have Alex as our teacher for this course. He explain C# and the project clearly teaches us the

application development hand-by-hand. And he is very knowledgeable and warm-heart, answering almost every question, even including the job-hunting question with his experience.