1. Project Description

1.A short description of the project

1.Each book record should contain the fields: ISBN, Title, UnitPrice, YearPub1ished and QOH (quantity on hand). Each book must be categorized.

2. More than one author can write each book and each author may participate in writing more than one book Each book can be published by only one publisher.

3. Each author record should contain the following pieces of information such as authorlD (for identification in case authors have the same name), first name, last name and email )

4. Hi-Tech 's customers are Colleges and Universities in Quebec A customer's information should include name, street, city, postal code, phone number, fax number and credit limit.

5. Hi-Tech has received computer science books from different publishers (suppliers): Premier Press, Wrox, Murach, Prentice Hall and more

6. Order clerks can take customers 'orders (by Phone, Fax, or Email) and order payments Will be made by direct withdrawal from the college /university' s bank account as specified in the contract between Hi-Tech and the customer.

7.At present, Hi-Tech has two order clerks who are responsible for taking the customers' orders.

1. Users and Operations



1. Technologies used to develop the application

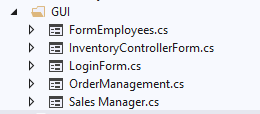
It was developed in C# in Visual Studio 2019 according to the object oriented programming principles.

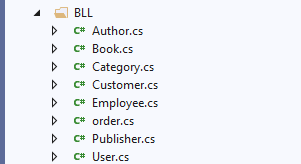
1. There is no problem with the login page, different people can login to their pages and if there is a userid and password error, people will not be able to login.
2. formEmployee works perfectly with list forms and can add employees, update, delete and search employees.
3. The user page works perfectly. You can list out the contents of the form. Add, update, delete, and find user ids.
4. The form works perfectly in the sales manager tool. form includes add, update database, delete, update data, find id and name
5. The inventoryControllerForm works perfectly, and the tables of book, author, category, and publisher can be listed out. And add, delete, update, search all work very well.
6. orderManagement This form can list data, but update, delete and search do not work, I have tried many ways and write a little code, but still can not, I hope the teacher can help me solve this problem.

2.PROJECT DEVELOPMENT PROCESS

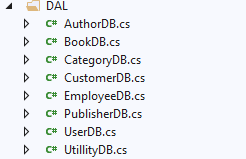
1. Analysis: To access the system, each user is required to enter his/her valid username and password.

2. Design:

GUI:



BLL:



DAL:

Models:



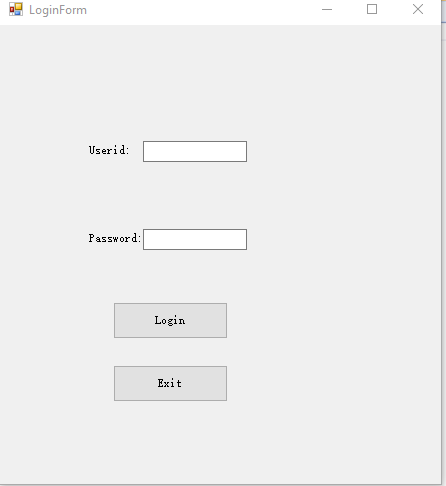
**3.Implementation**: Source Code Listings

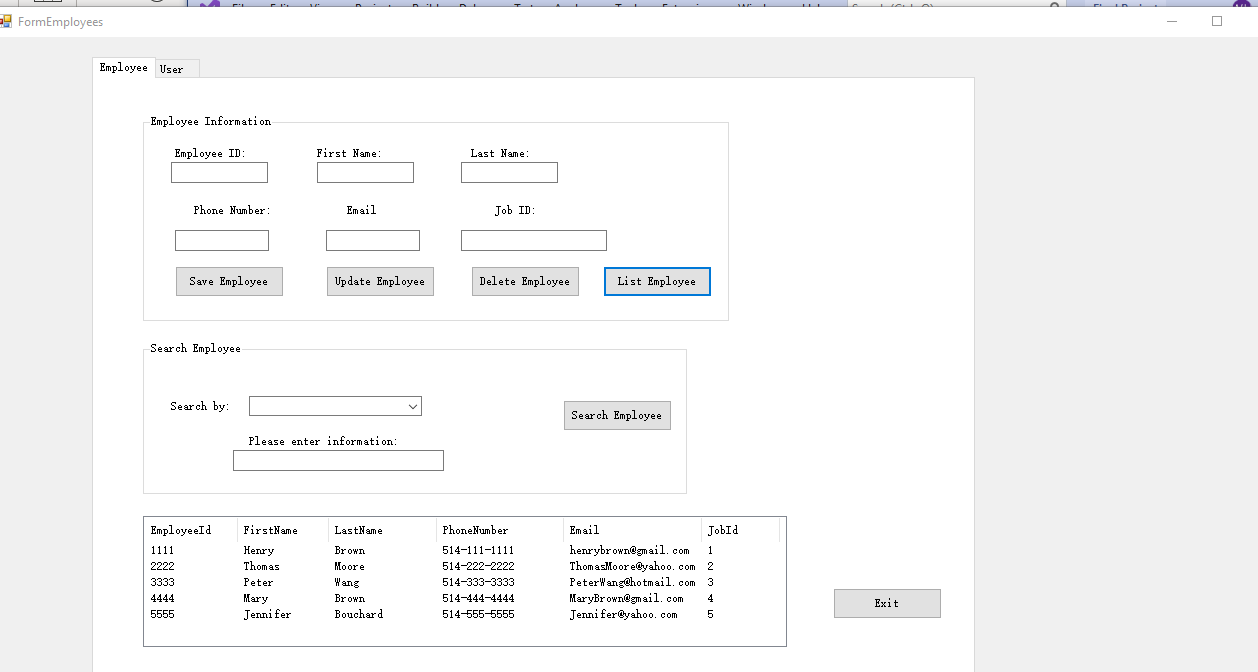
4. **Testing**: Test results of the application in well-defined table format

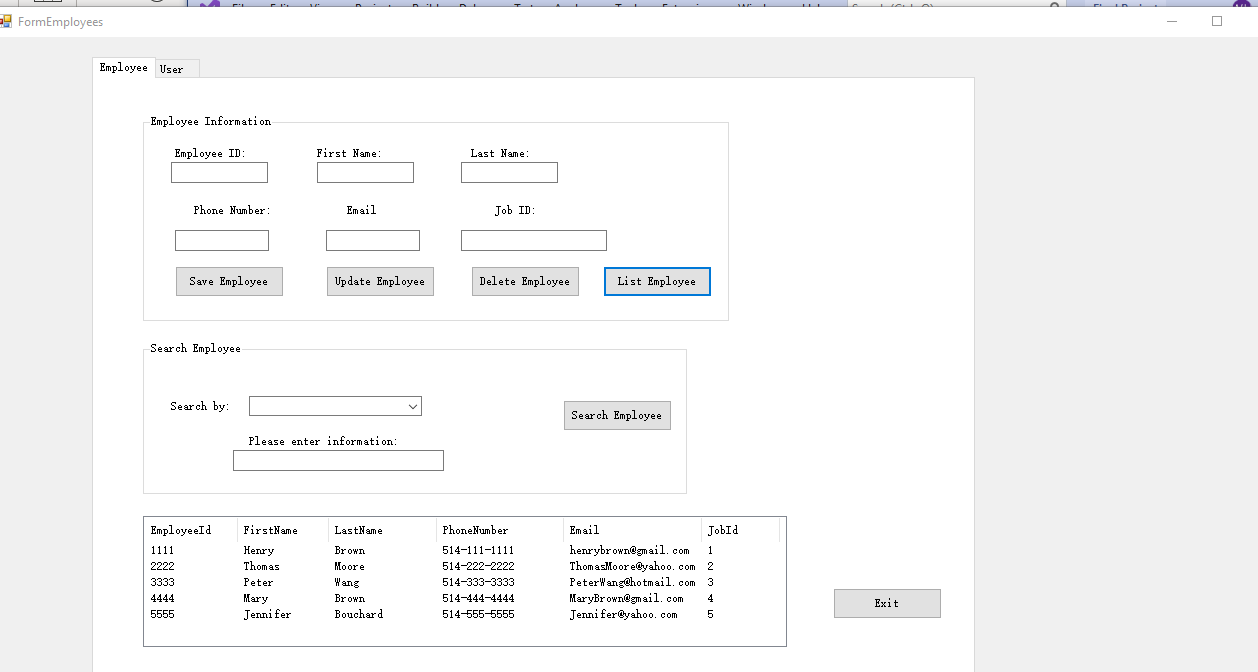
5.In conclusion:

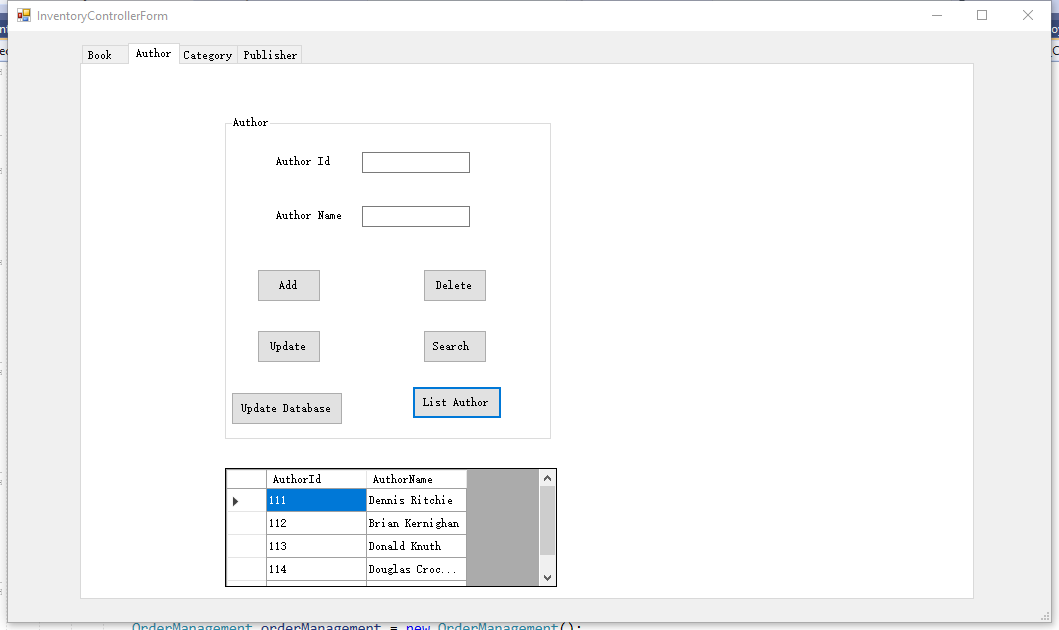
This semester in c# I learned what is bll, dll, gui and mastered these knowledge under the teacher's teaching. This project, I think the first few problems I have mastered the knowledge points, but Entity Framework still some can not write, I googled a lot of information, and try to write the code themselves, still have problems, I hope to get a solution.

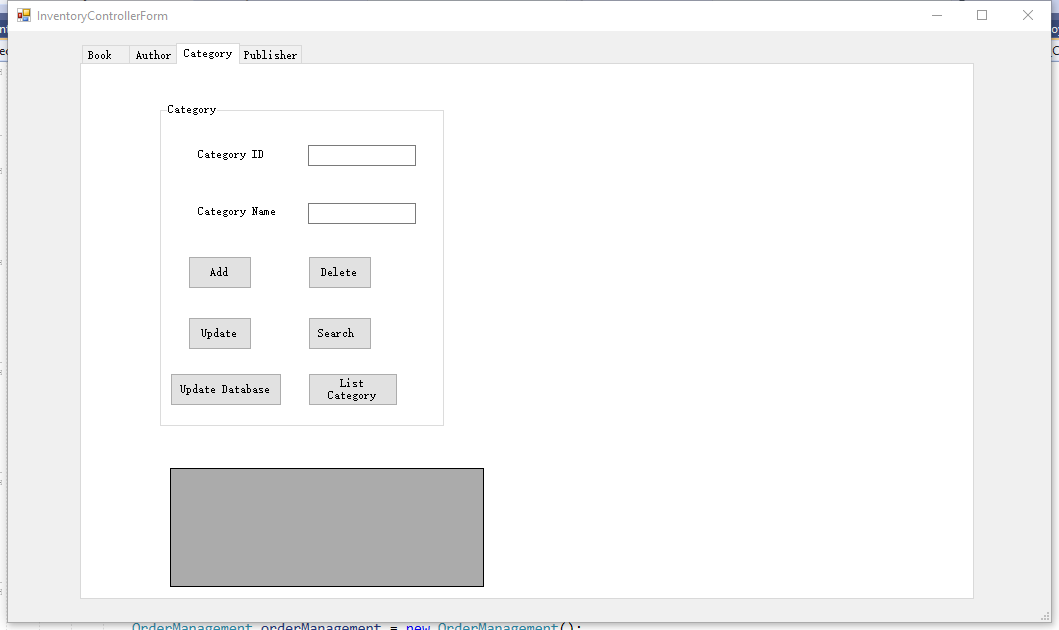
Login Form:

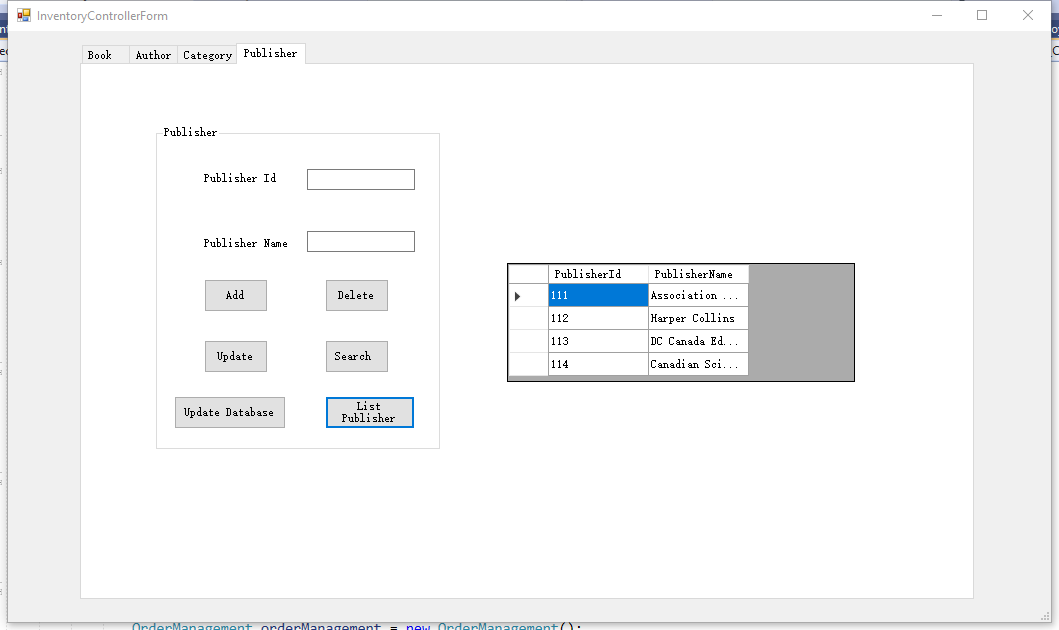


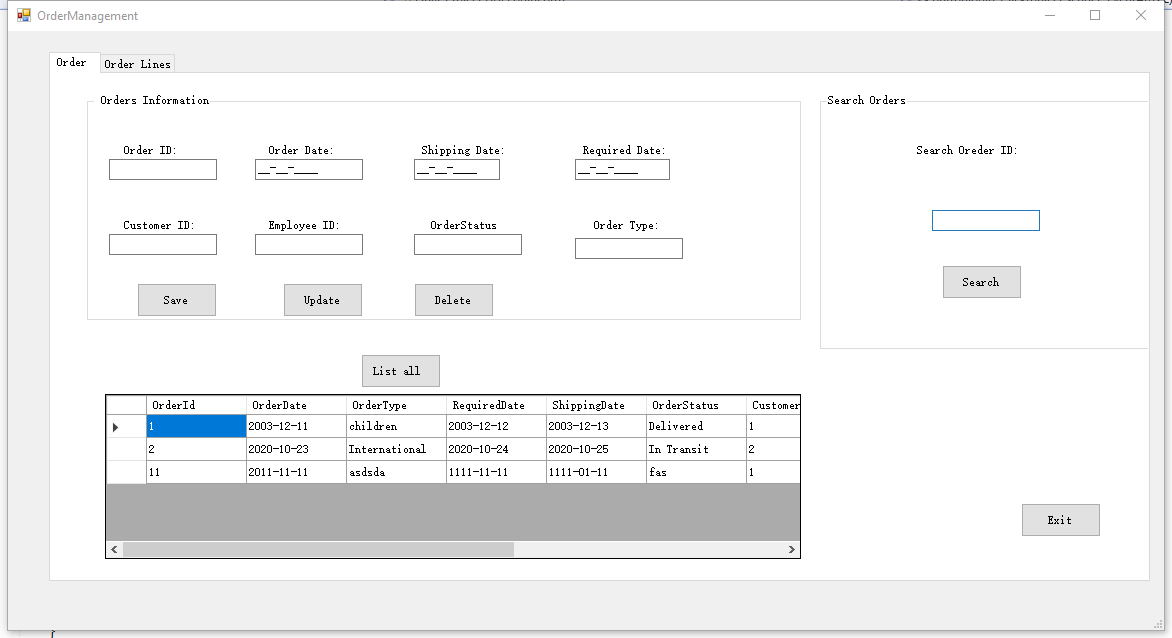


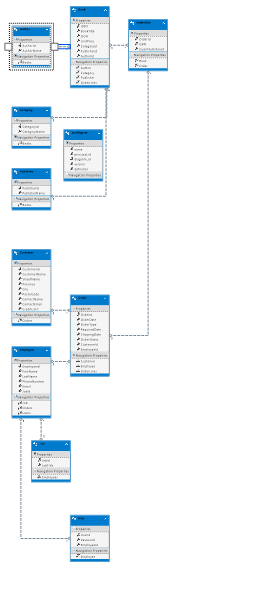












using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Final\_Project.DAL;

namespace Final\_Project.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> AuthorList()

{

return (AuthorDB.GetListRecord());

}

public void SearchAuthor(int authorId)

{

AuthorDB.SearchRecord(authorId);

}

public List<Author> SearchAuthor(string name)

{

return AuthorDB.SearchRecord(name);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Final\_Project.DAL;

namespace Final\_Project.BLL

{

public class Book

{

private string isbn;

private string bookTitel;

private string qoh;

private int unitPrice;

private int categoryId;

private int publisherId;

private int authorId;

public string Isbn { get => isbn; set => isbn = value; }

public string Qoh { get => qoh; set => qoh = value; }

public int UnitPrice { get => unitPrice; set => unitPrice = value; }

public string BookTitel { get => bookTitel; set => bookTitel = value; }

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

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

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

public List<Book> BookList()

{

return (BookDB.GetListRecord());

}

public void SearchBook(int bookId)

{

BookDB.SearchRecord(bookId);

}

public void UpdateStudent(Book book)

{

BookDB.UpdateRecord(book);

}

public void DeleteStudent(int bookId)

{

BookDB.DeleteRecord(bookId);

}

public List<Book> Searchbook(string name)

{

return BookDB.SearchRecord(name);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Final\_Project.DAL;

namespace Final\_Project.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> CategoryList()

{

return (CategoryDB.GetListRecord());

}

public void SearchCategory(int categoryId)

{

CategoryDB.SearchRecord(categoryId);

}

public List<Category> SearchCategory(string name)

{

return CategoryDB.SearchRecord(name);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Final\_Project.DAL;

namespace Final\_Project.BLL

{

public class Customer

{

private int customerId;

private string customerName;

private string streetName;

private string province;

private string city;

private string postalCode;

private string contactName;

private string contactEmail;

private int creditLimit;

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

public int CreditLimit { get => creditLimit; set => creditLimit = value; }

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

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

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

public string ContactName { get => contactName; set => contactName = value; }

public string ContactEmail { get => contactEmail; set => contactEmail = value; }

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

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

public List<Customer> CustomersList()

{

return (CustomerDB.GetListRecord());

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Final\_Project.DAL;

namespace Final\_Project.BLL

{

public class Employee

{

private int employeeId;

private string firstName;

private string lastName;

private string phoneNumber;

private string email;

private int jobId;

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

public List<Employee> ListEmployees()

{

return EmployeeDB.ListAllRecords();

}

public void UpdateEmployee(Employee employee)

{

EmployeeDB.UpdateRecord(employee);

}

public void DeleteEmployee(int EmployeeId)

{

EmployeeDB.DeleteRecord(EmployeeId);

}

public void SaveEmployee(Employee employee)

{

EmployeeDB.SaveRecord(employee);

}

public Employee SearchEmployee(int empId)

{

return EmployeeDB.SearchRecord(empId);

}

public List<Employee> SearchEmployee(string name)

{

return EmployeeDB.SearchRecord(name);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Final\_Project.DAL;

namespace Final\_Project.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> SuppliersList()

{

return (PublisherDB.GetListRecord());

}

public void SearchStudent(int suppliersId)

{

PublisherDB.SearchRecord(suppliersId);

}

public void UpdateStudent(Publisher suppliers)

{

PublisherDB.UpdateRecord(suppliers);

}

public void DeleteStudent(int suppliersId)

{

PublisherDB.DeleteRecord(suppliersId);

}

public List<Publisher> SearchStudent(string name)

{

return PublisherDB.SearchRecord(name);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Final\_Project.DAL;

namespace Final\_Project.BLL

{

public class User

{

private int userId;

private string password;

private int employeeId;

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

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

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

public bool VerfyLogIn(int userid, string password)

{

return UserDB.LogIn(userid, password);

}

public List<User> ListUsers()

{

return UserDB.ListAllRecords();

}

public void UpdateUser(User user)

{

UserDB.UpdateRecord(user);

}

public void DeleteUser(int UserId)

{

UserDB.DeleteRecord(UserId);

}

public void SaveUser(User user)

{

UserDB.SaveRecord(user);

}

public User SearchUser(int userId)

{

return UserDB.SearchRecord(userId);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Final\_Project.BLL;

using System.Data.SqlClient;

namespace Final\_Project.DAL

{

public static class AuthorDB

{

public static List<Author> GetListRecord()

{

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

Author author;

using (SqlConnection conn = UtillityDB.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;

}

public static void SearchRecord(int Id)

{

Author author = new Author();

SqlConnection conn = UtillityDB.ConnectDB();

conn = UtillityDB.ConnectDB();

SqlCommand cmdSelect = new SqlCommand();

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

"WHERE AuthorId = @AuthorId";

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

cmdSelect.Connection = conn;

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

if (sqlReader.Read())

{

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

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

}

else

{

author = null;

}

}

public static List<Author> SearchRecord(string name)

{

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

SqlConnection conn = UtillityDB.ConnectDB();

conn = UtillityDB.ConnectDB();

SqlCommand cmdSelect = new SqlCommand();

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

"WHERE AuthorName = @AuthorName ";

cmdSelect.Parameters.AddWithValue("@AuthorName", name);

cmdSelect.Connection = conn;

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

Author author;

while (sqlReader.Read())

{

author = new Author();

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

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

listAuthor.Add(author);

}

return listAuthor;

}

public static void UpdateRecord(Author author)

{

SqlConnection conn = UtillityDB.ConnectDB();

SqlCommand cmdUpdate = new SqlCommand();

cmdUpdate.CommandText = "UPDATE Authors " +

"SET AuthorId = @AuthorId," +

" AuthorName = @AuthorName," +

"WHERE AuthorId = @AuthorId";

cmdUpdate.Parameters.AddWithValue("@AuthorId", author.AuthorId);

cmdUpdate.Parameters.AddWithValue("@AuthorName", author.AuthorName);

cmdUpdate.Connection = conn;

cmdUpdate.ExecuteNonQuery();

conn.Close();

}

public static void DeleteRecord(int authorId)

{

SqlConnection conn = UtillityDB.ConnectDB();

SqlCommand cmdDelete = new SqlCommand();

cmdDelete.CommandText = "DELETE FROM Authors " +

"WHERE AuthorId= @AuthorId";

cmdDelete.Parameters.AddWithValue("@AuthorId", authorId);

cmdDelete.Connection = conn;

cmdDelete.ExecuteNonQuery();

conn.Close();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Final\_Project.BLL;

using System.Data.SqlClient;

namespace Final\_Project.DAL

{

public static class BookDB

{

public static List<Book> GetListRecord()

{

List<Book> ListBook = new List<Book>();

Book book;

using (SqlConnection conn=UtillityDB.ConnectDB()){

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

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

if (sqlReader.HasRows)

{

while (sqlReader.Read())

{

book = new Book();

book.BookTitel = sqlReader["BookTitle"].ToString();

book.Isbn = sqlReader["Isbn"].ToString();

book.Qoh = sqlReader["QOH"].ToString();

book.UnitPrice = Convert.ToInt32(sqlReader["UnitPrice"]);

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

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

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

ListBook.Add(book);

}

}

else

{

ListBook = null;

}

}

return ListBook;

}

public static void SearchRecord(int Id)

{

Book book = new Book();

SqlConnection conn = UtillityDB.ConnectDB();

conn = UtillityDB.ConnectDB();

SqlCommand cmdSelect = new SqlCommand();

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

"WHERE BookId = @BookId";

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

cmdSelect.Connection = conn;

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

if (sqlReader.Read())

{

book = new Book();

book.BookTitel = sqlReader["BookTitle"].ToString();

book.Isbn = sqlReader["Isbn"].ToString();

book.Qoh = sqlReader["QOH"].ToString();

book.UnitPrice = Convert.ToInt32(sqlReader["UnitPrice"]);

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

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

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

}

else

{

book = null;

}

}

public static List<Book> SearchRecord(string name)

{

List<Book> listStudent = new List<Book>();

SqlConnection conn = UtillityDB.ConnectDB();

conn = UtillityDB.ConnectDB();

SqlCommand cmdSelect = new SqlCommand();

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

"WHERE Author = @Author " +

" Or Isbn = @LastName ";

cmdSelect.Parameters.AddWithValue("@Author", name);

cmdSelect.Parameters.AddWithValue("@Isbn", name);

cmdSelect.Connection = conn;

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

Book book;

while (sqlReader.Read())

{

book = new Book();

book.BookTitel = sqlReader["BookTitle"].ToString();

book.Isbn = sqlReader["Isbn"].ToString();

book.Qoh = sqlReader["QOH"].ToString();

book.UnitPrice = Convert.ToInt32(sqlReader["UnitPrice"]);

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

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

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

listStudent.Add(book);

}

return listStudent;

}

public static void UpdateRecord(Book book)

{

SqlConnection conn = UtillityDB.ConnectDB();

SqlCommand cmdUpdate = new SqlCommand();

cmdUpdate.CommandText = "UPDATE Books " +

"SET Isbn = @Isbn," +

" BookTitel = @BookTitel," +

" QOH = @QOH," +

" UnitPrice = @UnitPrice " +

" CategoryId = @CategoryId " +

" PublisherId = @PublisherId " +

" AuthorId = @AuthorId " +

"WHERE Isbn = @Isbn";

cmdUpdate.Parameters.AddWithValue("@Isbn", book.Isbn);

cmdUpdate.Parameters.AddWithValue("@BookTitel", book.BookTitel);

cmdUpdate.Parameters.AddWithValue("@QOH", book.Qoh);

cmdUpdate.Parameters.AddWithValue("@CategoryId", book.CategoryId);

cmdUpdate.Parameters.AddWithValue("@UnitPrice", book.UnitPrice);

cmdUpdate.Parameters.AddWithValue("@PublisherId", book.PublisherId);

cmdUpdate.Parameters.AddWithValue("@AuthorId", book.AuthorId);

cmdUpdate.Connection = conn;

cmdUpdate.ExecuteNonQuery();

conn.Close();

}

public static void DeleteRecord(int BookId)

{

SqlConnection conn = UtillityDB.ConnectDB();

SqlCommand cmdDelete = new SqlCommand();

cmdDelete.CommandText = "DELETE FROM Books " +

"WHERE BookId= @BookId";

cmdDelete.Parameters.AddWithValue("@BookId", BookId);

cmdDelete.Connection = conn;

cmdDelete.ExecuteNonQuery();

conn.Close();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Final\_Project.BLL;

using System.Data.SqlClient;

namespace Final\_Project.DAL

{

public static class CategoryDB

{

public static List<Category> GetListRecord()

{

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

Category category;

using (SqlConnection conn = UtillityDB.ConnectDB())

{

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

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

if (sqlReader.HasRows)

{

while (sqlReader.Read())

{

category = new Category();

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

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

listCategory.Add(category);

}

}

else

{

listCategory = null;

}

}

return listCategory;

}

public static void SearchRecord(int Id)

{

Category category = new Category();

SqlConnection conn = UtillityDB.ConnectDB();

conn = UtillityDB.ConnectDB();

SqlCommand cmdSelect = new SqlCommand();

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

"WHERE CategoryId = @CategoryId";

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

cmdSelect.Connection = conn;

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

if (sqlReader.Read())

{

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

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

}

else

{

category = null;

}

}

public static List<Category> SearchRecord(string name)

{

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

SqlConnection conn = UtillityDB.ConnectDB();

conn = UtillityDB.ConnectDB();

SqlCommand cmdSelect = new SqlCommand();

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

"WHERE CategoryName = @CategoryName ";

cmdSelect.Parameters.AddWithValue("@CategoryName", name);

cmdSelect.Connection = conn;

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

Category category;

while (sqlReader.Read())

{

category = new Category();

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

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

listCategory.Add(category);

}

return listCategory;

}

public static void UpdateRecord(Category category)

{

SqlConnection conn = UtillityDB.ConnectDB();

SqlCommand cmdUpdate = new SqlCommand();

cmdUpdate.CommandText = "UPDATE Categories " +

"SET CategoryId = @CategoryId," +

" CategoryName = @CategoryName," +

"WHERE CategoryId = @CategoryId";

cmdUpdate.Parameters.AddWithValue("@CategoryId", category.CategoryId);

cmdUpdate.Parameters.AddWithValue("@CategoryName", category.CategoryName);

cmdUpdate.Connection = conn;

cmdUpdate.ExecuteNonQuery();

conn.Close();

}

public static void DeleteRecord(int categoryId)

{

SqlConnection conn = UtillityDB.ConnectDB();

SqlCommand cmdDelete = new SqlCommand();

cmdDelete.CommandText = "DELETE FROM Categories " +

"WHERE CategoryId= @CategoryId";

cmdDelete.Parameters.AddWithValue("@CategoryId", categoryId);

cmdDelete.Connection = conn;

cmdDelete.ExecuteNonQuery();

conn.Close();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Final\_Project.BLL;

using System.Data.SqlClient;

namespace Final\_Project.DAL

{

public static class CustomerDB

{

public static List<Customer> GetListRecord()

{

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

Customer aCustomer;

using (SqlConnection conn = UtillityDB.ConnectDB())

{

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

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

if (sqlReader.HasRows)

{

while (sqlReader.Read())

{

aCustomer = new Customer();

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

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

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

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

aCustomer.CreditLimit = Convert.ToInt32(sqlReader["CreditLimit"]);

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

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

aCustomer.ContactName = sqlReader["ContactName"].ToString();

aCustomer.ContactEmail = sqlReader["ContactEmail"].ToString();

listCustomer.Add(aCustomer);

}

}

else

{

listCustomer = null;

}

}

return listCustomer;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Final\_Project.BLL;

using System.Data.SqlClient;

namespace Final\_Project.DAL

{

public static class EmployeeDB

{

public static List<Employee> ListAllRecords()

{

List<Employee> listEmployee = new List<Employee>();

SqlConnection conn = UtillityDB.ConnectDB();

conn = UtillityDB.ConnectDB();

SqlCommand cmdSelect = new SqlCommand();

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

cmdSelect.Connection = conn;

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

Employee emp;

while (sqlReader.Read())

{

emp = new Employee();

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

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

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

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

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

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

listEmployee.Add(emp);

}

return listEmployee;

}

public static void UpdateRecord(Employee employee)

{

SqlConnection conn = UtillityDB.ConnectDB();

SqlCommand cmdUpdate = new SqlCommand();

cmdUpdate.CommandText = "UPDATE Employees " +

"SET EmployeeId = @EmployeeId," +

"FirstName = @FirstName," +

"LastName = @LastName," +

"Email = @Email," +

"PhoneNumber = @PhoneNumber," +

"JobId = @JobId " +

"WHERE EmployeeId = @EmployeeId";

cmdUpdate.Parameters.AddWithValue("@EmployeeId",employee.EmployeeId);

cmdUpdate.Parameters.AddWithValue("@FirstName", employee.FirstName);

cmdUpdate.Parameters.AddWithValue("@LastName", employee.LastName);

cmdUpdate.Parameters.AddWithValue("@JobId", employee.JobId);

cmdUpdate.Parameters.AddWithValue("@Email", employee.Email);

cmdUpdate.Parameters.AddWithValue("@PhoneNumber", employee.PhoneNumber);

cmdUpdate.Connection = conn;

cmdUpdate.ExecuteNonQuery();

conn.Close();

}

public static void DeleteRecord(int Id)

{

SqlConnection conn = UtillityDB.ConnectDB();

SqlCommand cmdDelete = new SqlCommand();

cmdDelete.CommandText = "DELETE FROM Employees " +

"WHERE EmployeeId = @EmployeeId";

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

cmdDelete.Connection = conn;

cmdDelete.ExecuteNonQuery();

conn.Close();

}

public static void SaveRecord(Employee employee)

{

SqlConnection conn = UtillityDB.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", employee.EmployeeId);

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

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

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

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

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

cmdInsert.Connection = conn;

cmdInsert.ExecuteNonQuery();

conn.Close();

}

public static Employee SearchRecord(int Id)

{

Employee emp = new Employee();

SqlConnection conn = UtillityDB.ConnectDB();

conn = UtillityDB.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> SearchRecord(string name)

{

List<Employee> listEmp = new List<Employee>();

SqlConnection conn = UtillityDB.ConnectDB();

conn = UtillityDB.ConnectDB();

SqlCommand cmdSelect = new SqlCommand();

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

"WHERE FirstName = @FirstName " +

" Or LastName = @LastName ";

cmdSelect.Parameters.AddWithValue("@FirstName", name);

cmdSelect.Parameters.AddWithValue("@LastName", name);

cmdSelect.Connection = conn;

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

Employee emp;

while (sqlReader.Read())

{

emp = new Employee();

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;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Final\_Project.BLL;

using System.Data.SqlClient;

namespace Final\_Project.DAL

{

public static class PublisherDB

{

public static List<Publisher> GetListRecord()

{

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

Publisher publisher;

using (SqlConnection conn = UtillityDB.ConnectDB())

{

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

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

if (sqlReader.HasRows)

{

while (sqlReader.Read())

{

publisher = new Publisher();

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

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

listPublisher.Add(publisher);

}

}

else

{

listPublisher = null;

}

}

return listPublisher;

}

public static void SearchRecord(int Id)

{

Publisher publisher = new Publisher();

SqlConnection conn = UtillityDB.ConnectDB();

conn = UtillityDB.ConnectDB();

SqlCommand cmdSelect = new SqlCommand();

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

"WHERE PublisherId = @PublisherId";

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

cmdSelect.Connection = conn;

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

if (sqlReader.Read())

{

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

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

}

else

{

publisher = null;

}

}

public static List<Publisher> SearchRecord(string name)

{

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

SqlConnection conn = UtillityDB.ConnectDB();

conn = UtillityDB.ConnectDB();

SqlCommand cmdSelect = new SqlCommand();

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

"WHERE PublisherName = @PublisherName ";

cmdSelect.Parameters.AddWithValue("@PublisherName", name);

cmdSelect.Connection = conn;

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

Publisher publisher;

while (sqlReader.Read())

{

publisher = new Publisher();

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

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

listPublisher.Add(publisher);

}

return listPublisher;

}

public static void UpdateRecord(Publisher publisher)

{

SqlConnection conn = UtillityDB.ConnectDB();

SqlCommand cmdUpdate = new SqlCommand();

cmdUpdate.CommandText = "UPDATE Publishers " +

"SET PublisherId = @PublisherId," +

" PublisherName = @PublisherName," +

"WHERE PublisherId = @PublisherId";

cmdUpdate.Parameters.AddWithValue("@PublisherId", publisher.PublisherId);

cmdUpdate.Parameters.AddWithValue("@SupplierName", publisher.PublisherName);

cmdUpdate.Connection = conn;

cmdUpdate.ExecuteNonQuery();

conn.Close();

}

public static void DeleteRecord(int PublisherId)

{

SqlConnection conn = UtillityDB.ConnectDB();

SqlCommand cmdDelete = new SqlCommand();

cmdDelete.CommandText = "DELETE FROM Publishers " +

"WHERE PublisherId= @PublisherId";

cmdDelete.Parameters.AddWithValue("@PublisherId", PublisherId);

cmdDelete.Connection = conn;

cmdDelete.ExecuteNonQuery();

conn.Close();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SqlClient;

using Final\_Project.BLL;

namespace Final\_Project.DAL

{

public class UserDB

{

public static bool LogIn(int userid, string password)

{

SqlConnection conn = UtillityDB.ConnectDB();

SqlCommand cmdSelect = new SqlCommand();

cmdSelect.Connection = conn;

cmdSelect.CommandText = "SELECT \* FROM Users " + "WHERE UserId=@UserId and Password=@Password";

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

cmdSelect.Parameters.AddWithValue("@Password", password);

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

if (sqlReader.Read())

{

return true;

}

return false;

}

public static List<User> ListAllRecords()

{

List<User> listUser = new List<User>();

SqlConnection conn = UtillityDB.ConnectDB();

conn = UtillityDB.ConnectDB();

SqlCommand cmdSelect = new SqlCommand();

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

cmdSelect.Connection = conn;

SqlDataReader sqlReader = cmdSelect.ExecuteReader();

User user;

while (sqlReader.Read())

{

user = new 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 UpdateRecord(User user)

{

SqlConnection conn = UtillityDB.ConnectDB();

SqlCommand cmdUpdate = new SqlCommand();

cmdUpdate.CommandText = "UPDATE Users " +

"SET UserId = @UserId," +

"Password = @Password," +

"EmployeeId = @EmployeeId," +

"WHERE UserId = @UserId";

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

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

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

cmdUpdate.Connection = conn;

cmdUpdate.ExecuteNonQuery();

conn.Close();

}

public static void DeleteRecord(int Id)

{

SqlConnection conn = UtillityDB.ConnectDB();

SqlCommand cmdDelete = new SqlCommand();

cmdDelete.CommandText = "DELETE FROM User " +

"WHERE UserId = @UserId";

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

cmdDelete.Connection = conn;

cmdDelete.ExecuteNonQuery();

conn.Close();

}

public static void SaveRecord(User user)

{

SqlConnection conn = UtillityDB.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 User SearchRecord(int Id)

{

User user = new User();

SqlConnection conn = UtillityDB.ConnectDB();

conn = UtillityDB.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;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SqlClient;

using System.Configuration;

namespace Final\_Project.DAL

{

public static class UtillityDB

{

public static SqlConnection ConnectDB()

{

SqlConnection connDB = new SqlConnection();

connDB.ConnectionString = ConfigurationManager.ConnectionStrings["HiTechDB\_Connection"].ConnectionString;

connDB.Open();

return connDB;

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Final\_Project.BLL;

using Final\_Project.VALIDATION;

namespace Final\_Project.GUI

{

public partial class FormEmployees : Form

{

public FormEmployees()

{

InitializeComponent();

}

private void buttonlistemp\_Click(object sender, EventArgs e)

{

Employee emp = new Employee();

List<Employee> listEmp = emp.ListEmployees();

listViewemp.Items.Clear();

if (listEmp.Count != 0)

{

foreach (Employee anEmp in listEmp)

{

ListViewItem item = new ListViewItem(anEmp.EmployeeId.ToString());

item.SubItems.Add(anEmp.FirstName);

item.SubItems.Add(anEmp.LastName);

item.SubItems.Add(anEmp.PhoneNumber);

item.SubItems.Add(anEmp.Email);

item.SubItems.Add(anEmp.JobId.ToString());

listViewemp.Items.Add(item);

}

}

else

{

MessageBox.Show("Empty List!", "No Employee");

}

}

private void buttonsave\_Click(object sender, EventArgs e)

{

string tempId = textBoxEmpid.Text.Trim();

if (!(Validator.IsValidId(tempId)))

{

MessageBox.Show("Employee ID must be 4-digit number", "Invalid ID", MessageBoxButtons.OK, MessageBoxIcon.Error);

textBoxEmpid.Clear();

textBoxEmpid.Focus();

return;

}

Employee tempEmp = new Employee();

tempEmp = tempEmp.SearchEmployee(Convert.ToInt32(textBoxEmpid.Text.Trim()));

if (tempEmp != null)

{

MessageBox.Show("This Employee ID already exists!", "Duplicate ID", MessageBoxButtons.OK, MessageBoxIcon.Error);

textBoxEmpid.Clear();

textBoxEmpid.Focus();

return;

}

string tempFirstName = textBoxfirstname.Text.Trim();

if (!(Validator.IsValidName(tempFirstName)))

{

MessageBox.Show("Invalid First Name", "Invalid", MessageBoxButtons.OK, MessageBoxIcon.Error);

textBoxfirstname.Clear();

textBoxfirstname.Focus();

return;

}

string tempLastName = textBoxlastname.Text.Trim();

if (!(Validator.IsValidName(tempLastName)))

{

MessageBox.Show("Invalid Last Name", "Invalid", MessageBoxButtons.OK, MessageBoxIcon.Error);

textBoxlastname.Clear();

textBoxlastname.Focus();

return;

}

string tempJob = textBoxjobid.Text.Trim();

if ((Validator.IsEmpty(tempJob)))

{

MessageBox.Show("Job Title is required", "Invalid", MessageBoxButtons.OK, MessageBoxIcon.Error);

textBoxjobid.Clear();

textBoxjobid.Focus();

return;

}

string tempPhoneNumber = textBoxphonenumber.Text.Trim();

if ((Validator.IsEmpty(tempPhoneNumber)))

{

MessageBox.Show("PhoneNumber is required", "Invalid", MessageBoxButtons.OK, MessageBoxIcon.Error);

textBoxphonenumber.Clear();

textBoxphonenumber.Focus();

}

string tempEmail = textBoxemail.Text.Trim();

if ((Validator.IsEmpty(tempEmail)))

{

MessageBox.Show("Invalid Email", "Invalid", MessageBoxButtons.OK, MessageBoxIcon.Error);

textBoxemail.Clear();

textBoxemail.Focus();

}

Employee emp = new Employee();

emp.EmployeeId = Convert.ToInt32(textBoxEmpid.Text.Trim());

emp.FirstName = textBoxfirstname.Text.Trim();

emp.LastName = textBoxlastname.Text.Trim();

emp.PhoneNumber = textBoxphonenumber.Text.Trim();

emp.Email = textBoxemail.Text.Trim();

emp.JobId = Convert.ToInt32(textBoxjobid.Text.Trim());

emp.SaveEmployee(emp);

MessageBox.Show("Employee data saved successfully.", "Confirmation", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

private void buttonupdate\_Click(object sender, EventArgs e)

{

Employee emp = new Employee();

emp.EmployeeId = Convert.ToInt32(textBoxEmpid.Text.Trim());

emp.FirstName = textBoxfirstname.Text.Trim();

emp.LastName = textBoxlastname.Text.Trim();

emp.PhoneNumber = textBoxphonenumber.Text.Trim();

emp.Email = textBoxemail.Text.Trim();

emp.JobId = Convert.ToInt32(textBoxjobid.Text.Trim());

DialogResult answer = MessageBox.Show("Do you really want to update this employee info? ", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Information);

if (answer == DialogResult.Yes)

{

emp.UpdateEmployee(emp);

}

}

private void buttondeleteemp\_Click(object sender, EventArgs e)

{

Employee delemp = new Employee();

delemp = delemp.SearchEmployee(Convert.ToInt32(textBoxEmpid.Text));

if (delemp != null)

{

delemp.DeleteEmployee(Convert.ToInt32(textBoxEmpid.Text));

DialogResult delete=MessageBox.Show("Delete Successful!", "Successful", MessageBoxButtons.OK);

if (delete == DialogResult.Yes)

{

textBoxEmpid.Clear();

textBoxfirstname.Clear();

textBoxlastname.Clear();

textBoxphonenumber.Clear();

textBoxemail.Clear();

textBoxjobid.Clear();

}

}

else

{

MessageBox.Show("This Employee does not exist", "Error", MessageBoxButtons.OK);

textBoxEmpid.Text = "";

textBoxEmpid.Focus();

return;

}

}

private void buttonexit\_Click(object sender, EventArgs e)

{

DialogResult exit = MessageBox.Show("Do you really want to exit the application?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Information);

this.Hide();

LoginForm loginForm = new LoginForm();

loginForm.ShowDialog();

}

private void buttonsearch\_Click(object sender, EventArgs e)

{

int indexSelected = comboBoxsearchby.SelectedIndex;

switch (indexSelected)

{

case 1:

//Search by FirstName

Employee emp = new Employee();

List<Employee> listEmp = emp.SearchEmployee(textBoxinfo.Text.ToString());

listViewemp.Items.Clear();

if (listEmp.Count != 0)

{

foreach (Employee anEmp in listEmp)

{

ListViewItem item = new ListViewItem(anEmp.EmployeeId.ToString());

item.SubItems.Add(anEmp.FirstName);

item.SubItems.Add(anEmp.LastName);

item.SubItems.Add(anEmp.PhoneNumber);

item.SubItems.Add(anEmp.Email);

item.SubItems.Add(anEmp.JobId.ToString());

listViewemp.Items.Add(item);

}

}

else

{

MessageBox.Show("Employee not found!", "Error");

}

break;

case 2:

//Search by last name

Employee emp2 = new Employee();

List<Employee> listEmp2 = emp2.SearchEmployee(textBoxinfo.Text.Trim());

listViewemp.Items.Clear();

if (listEmp2.Count != 0)

{

foreach (Employee anEmp in listEmp2)

{

ListViewItem item = new ListViewItem(anEmp.EmployeeId.ToString());

item.SubItems.Add(anEmp.FirstName);

item.SubItems.Add(anEmp.LastName);

item.SubItems.Add(anEmp.Email);

item.SubItems.Add(anEmp.JobId.ToString());

item.SubItems.Add(anEmp.PhoneNumber);

listViewemp.Items.Add(item);

}

}

else

{

MessageBox.Show("Employee not found!", "Error");

}

break;

case 3:

//Search by id

Employee emp3 = new Employee();

List<Employee> listEmp3 = emp3.SearchEmployee(textBoxinfo.Text.Trim());

listViewemp.Items.Clear();

if (listEmp3.Count != 0)

{

foreach (Employee anEmp in listEmp3)

{

ListViewItem item = new ListViewItem(anEmp.EmployeeId.ToString());

item.SubItems.Add(anEmp.FirstName);

item.SubItems.Add(anEmp.LastName);

item.SubItems.Add(anEmp.Email);

item.SubItems.Add(anEmp.JobId.ToString());

item.SubItems.Add(anEmp.PhoneNumber);

listViewemp.Items.Add(item);

}

}

else

{

MessageBox.Show("Employee not found!", "Error");

}

break;

default:

break;

}

}

private void comboBoxsearchby\_SelectedIndexChanged(object sender, EventArgs e)

{

int indexSelected = comboBoxsearchby.SelectedIndex;

switch (indexSelected)

{

case 1:

labelinfo.Text = "Please enter the Employee ID";

textBoxinfo.Clear();

textBoxinfo.Focus();

break;

case 2:

labelinfo.Text = "Please enter the First Name";

textBoxinfo.Clear();

textBoxinfo.Focus();

break;

case 3:

labelinfo.Text = "Please enter the last Name";

textBoxinfo.Clear();

textBoxinfo.Focus();

break;

default:

break;

}

}

private void listViewemp\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void buttonuserlistall\_Click(object sender, EventArgs e)

{

User user = new User();

List<User> listUser = user.ListUsers();

listViewUser.Items.Clear();

if (listUser.Count != 0)

{

foreach (User anUser in listUser)

{

ListViewItem item = new ListViewItem(anUser.UserId.ToString());

item.SubItems.Add(anUser.Password);

item.SubItems.Add(anUser.EmployeeId.ToString());

listViewUser.Items.Add(item);

}

}

else

{

MessageBox.Show("Empty List!", "No User");

}

}

private void buttonuserSave\_Click(object sender, EventArgs e)

{

string userId = textBoxuserid.Text.Trim();

if (!(Validator.IsValidId(userId)))

{

MessageBox.Show("Employee ID must be 4-digit number", "Invalid ID", MessageBoxButtons.OK, MessageBoxIcon.Error);

textBoxuserid.Clear();

textBoxuserid.Focus();

return;

}

User user = new User();

user = user.SearchUser(Convert.ToInt32(textBoxuserid.Text.Trim()));

if (user != null)

{

MessageBox.Show("This User ID already exists!", "Duplicate ID", MessageBoxButtons.OK, MessageBoxIcon.Error);

textBoxuserid.Clear();

textBoxuserid.Focus();

return;

}

string userPassword = textBoxpassword.Text.Trim();

if (!(Validator.IsValidName(userPassword)))

{

MessageBox.Show("Invalid Password", "Invalid", MessageBoxButtons.OK, MessageBoxIcon.Error);

textBoxpassword.Clear();

textBoxpassword.Focus();

return;

}

string UserempId = textBoxuseremployeeid.Text.Trim();

if ((Validator.IsEmpty(UserempId)))

{

MessageBox.Show("EmployeeId is required", "Invalid", MessageBoxButtons.OK, MessageBoxIcon.Error);

textBoxuseremployeeid.Clear();

textBoxuseremployeeid.Focus();

return;

}

User user1 = new User();

user1.UserId = Convert.ToInt32(textBoxuseremployeeid.Text.Trim());

user1.Password = textBoxpassword.Text.Trim();

user1.EmployeeId = Convert.ToInt32(textBoxuseremployeeid.Text.Trim());

user1.SaveUser(user1);

MessageBox.Show("User data saved successfully.", "Confirmation", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

private void buttonuserupdate\_Click(object sender, EventArgs e)

{

User user = new User();

user.UserId = Convert.ToInt32(textBoxuserid.Text.Trim());

user.Password = textBoxpassword.Text.Trim();

user.EmployeeId = Convert.ToInt32(textBoxuseremployeeid.Text.Trim());

DialogResult answer = MessageBox.Show("Do you really want to update this employee info? ", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Information);

if (answer == DialogResult.Yes)

{

user.UpdateUser(user);

}

}

private void buttonuserdelete\_Click(object sender, EventArgs e)

{

User deluser = new User();

deluser = deluser.SearchUser(Convert.ToInt32(textBoxuserid.Text));

if (deluser != null)

{

deluser.DeleteUser(Convert.ToInt32(textBoxuserid.Text));

DialogResult delete = MessageBox.Show("Delete Successful!", "Successful", MessageBoxButtons.OK);

if (delete == DialogResult.Yes)

{

textBoxuserid.Clear();

textBoxpassword.Clear();

textBoxuseremployeeid.Clear();

}

}

else

{

MessageBox.Show("This User does not exist", "Error", MessageBoxButtons.OK);

textBoxuserid.Text = "";

textBoxuserid.Focus();

return;

}

}

private void buttonusersearch\_Click(object sender, EventArgs e)

{

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Final\_Project.BLL;

using Final\_Project.DAL;

using System.Data.SqlClient;

namespace Final\_Project.GUI

{

public partial class InventoryControllerForm : Form

{

Book book = new Book();

Author author = new Author();

Category category = new Category();

Publisher Publisher = new Publisher();

SqlDataAdapter daBook,daAuthor, daCategory, daPublisher;

DataSet dsBook,dsAuthor, dsCategory, dsPublisher;

DataTable dtBook, dtAuthor, dtCategory, dtPublisher;

SqlCommandBuilder sqlBuilder;

public InventoryControllerForm()

{

InitializeComponent();

}

private void textBox9\_TextChanged(object sender, EventArgs e)

{

}

private void groupBox1\_Enter(object sender, EventArgs e)

{

}

private void buttonlistbook\_Click(object sender, EventArgs e)

{

Book book = new Book();

dataGridViewbook.DataSource = book.BookList();

}

private void buttonsearchbook\_Click(object sender, EventArgs e)

{

int searchisbn = Convert.ToInt32(textBoxisbn.Text.Trim());

DataRow drBook = dtBook.Rows.Find(searchisbn);

if (drBook != null)

{

textBoxtitle.Text = drBook["BookTitle"].ToString();

textBoxisbn.Text = drBook["Isbn"].ToString();

textBoxPublisher.Text = drBook["PublisherId"].ToString();

textBoxauthor.Text = drBook["AuthorId"].ToString();

textBoxcategory.Text = drBook["CategoryId"].ToString();

textBoxqoh.Text = drBook["Qoh"].ToString();

textBoxprice.Text = drBook["UnitPrice"].ToString();

}

else

{

MessageBox.Show("Books not found!", "Invalid Book ID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void buttonlistSuppliers\_Click(object sender, EventArgs e)

{

Publisher publisher = new Publisher();

dataGridViewsuppliser.DataSource = publisher.SuppliersList();

}

private void buttonaddauthor\_Click(object sender, EventArgs e)

{

DataRow dr = dtAuthor.NewRow();

dr["AuthorId"] = Convert.ToInt32(textBoxauthorid.Text.Trim());

dr["AuthorName"] = textBoxauthorname.Text.Trim();

dtAuthor.Rows.Add(dr);

MessageBox.Show(dr.RowState.ToString());

}

private void buttonAddbook\_Click(object sender, EventArgs e)

{

DataRow dr = dtBook.NewRow();

dr["BookTitle"] = textBoxtitle.Text.Trim();

dr["Isbn"] = textBoxisbn.Text.Trim();

dr["PublisherId"] = Convert.ToInt32(textBoxPublisher.Text.Trim());

dr["AuthorId"] = Convert.ToInt32(textBoxauthor.Text.Trim());

dr["CategoryId"] = Convert.ToInt32(textBoxcategory.Text.Trim());

dr["Qoh"] =textBoxqoh.Text.Trim();

dr["UnitPrice"] = Convert.ToInt32(textBoxprice.Text.Trim());

dtBook.Rows.Add(dr);

MessageBox.Show(dr.RowState.ToString());

}

private void buttondeleteauthor\_Click(object sender, EventArgs e)

{

int searchId = Convert.ToInt32(textBoxauthorid.Text.Trim());

DataRow drAuthor = dtAuthor.Rows.Find(searchId);

drAuthor.Delete();

MessageBox.Show(drAuthor.RowState.ToString());

}

private void buttonupdateauthor\_Click(object sender, EventArgs e)

{

int searchId = Convert.ToInt32(textBoxauthorid.Text.Trim());

DataRow drAuthor = dtAuthor.Rows.Find(searchId);

drAuthor["AuthorId"] = Convert.ToInt32(textBoxauthorid.Text.Trim());

drAuthor["AuthorName"] = textBoxauthorname.Text.Trim();

MessageBox.Show(drAuthor.RowState.ToString());

}

private void buttonsearchauthor\_Click(object sender, EventArgs e)

{

int searchId = Convert.ToInt32(textBoxauthorid.Text.Trim());

DataRow drAuthor = dtAuthor.Rows.Find(searchId);

if (drAuthor != null)

{

textBoxauthorid.Text = drAuthor["AuthorId"].ToString();

textBoxauthorname.Text = drAuthor["AuthorName"].ToString();

}

else

{

MessageBox.Show("Author not found!", "Invalid Author ID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void buttonupdatedatabseauthor\_Click(object sender, EventArgs e)

{

daAuthor.Update(dsAuthor.Tables["Authors"]);

MessageBox.Show("Database has been updated sucessfully.", "Confirmation");

}

private void buttonaddCategory\_Click(object sender, EventArgs e)

{

DataRow dr = dtCategory.NewRow();

dr["CategoryId"] = Convert.ToInt32(textBoxCategoryid.Text.Trim());

dr["CategoryName"] = textBoxCategoryname.Text.Trim();

dtCategory.Rows.Add(dr);

MessageBox.Show(dr.RowState.ToString());

}

private void buttondeleteCategory\_Click(object sender, EventArgs e)

{

int searchId = Convert.ToInt32(textBoxCategoryid.Text.Trim());

DataRow drCategory = dtCategory.Rows.Find(searchId);

drCategory.Delete();

MessageBox.Show(drCategory.RowState.ToString());

}

private void buttonupdateCategory\_Click(object sender, EventArgs e)

{

int searchId = Convert.ToInt32(textBoxCategoryid.Text.Trim());

DataRow drCategory = dtCategory.Rows.Find(searchId);

drCategory["CategoryId"] = Convert.ToInt32(textBoxCategoryid.Text.Trim());

drCategory["CategoryName"] = textBoxCategoryname.Text.Trim();

MessageBox.Show(drCategory.RowState.ToString());

}

private void buttondeleteSuppliers\_Click(object sender, EventArgs e)

{

int searchId = Convert.ToInt32(textBoxPublisherid.Text.Trim());

DataRow drPublisher = dtPublisher.Rows.Find(searchId);

drPublisher.Delete();

MessageBox.Show(drPublisher.RowState.ToString());

}

private void buttonupdateSuppliers\_Click(object sender, EventArgs e)

{

int searchId = Convert.ToInt32(textBoxPublisherid.Text.Trim());

DataRow drPublisher = dtPublisher.Rows.Find(searchId);

drPublisher["PublisherId"] = Convert.ToInt32(textBoxPublisherid.Text.Trim());

drPublisher["PublisherName"] = textBoxPublishername.Text.Trim();

MessageBox.Show(drPublisher.RowState.ToString());

}

private void buttonsearchSuppliers\_Click(object sender, EventArgs e)

{

int searchId = Convert.ToInt32(textBoxPublisherid.Text.Trim());

DataRow drPublisher = dtPublisher.Rows.Find(searchId);

if (drPublisher != null)

{

textBoxPublisherid.Text = drPublisher["PublisherId"].ToString();

textBoxPublishername.Text = drPublisher["PublisherName"].ToString();

}

else

{

MessageBox.Show("Suppliers not found!", "Invalid Suppliers ID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void buttonupdaSuppliers\_Click(object sender, EventArgs e)

{

daPublisher.Update(dsPublisher.Tables["Publishers"]);

MessageBox.Show("Database has been updated sucessfully.", "Confirmation");

}

private void buttonexit\_Click(object sender, EventArgs e)

{

DialogResult exit = MessageBox.Show("Do you really want to exit the application?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Information);

this.Hide();

LoginForm loginForm = new LoginForm();

loginForm.ShowDialog();

}

private void buttonsearchCategory\_Click(object sender, EventArgs e)

{

int searchId = Convert.ToInt32(textBoxCategoryid.Text.Trim());

DataRow drCategory = dtCategory.Rows.Find(searchId);

if (drCategory != null)

{

textBoxCategoryid.Text = drCategory["CategoryId"].ToString();

textBoxCategoryname.Text = drCategory["CategoryName"].ToString();

}

else

{

MessageBox.Show("Category not found!", "Invalid Category ID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void buttonupdaCategory\_Click(object sender, EventArgs e)

{

daCategory.Update(dsCategory.Tables["Categories"]);

MessageBox.Show("Database has been updated sucessfully.", "Confirmation");

}

private void buttonaddSuppliers\_Click(object sender, EventArgs e)

{

DataRow dr = dtPublisher.NewRow();

dr["PublisherId"] = Convert.ToInt32(textBoxPublisherid.Text.Trim());

dr["PublisherName"] = textBoxPublishername.Text.Trim();

dtPublisher.Rows.Add(dr);

MessageBox.Show(dr.RowState.ToString());

}

private void InventoryControllerForm\_Load(object sender, EventArgs e)

{

dsBook = new DataSet("BookDB");

dtBook = new DataTable("Books");

dsBook.Tables.Add(dtBook);

dtBook.Columns.Add("BookTitle", typeof(string));

dtBook.Columns.Add("Isbn", typeof(string));

dtBook.Columns.Add("SupplierId", typeof(Int32));

dtBook.Columns.Add("AuthorId", typeof(Int32));

dtBook.Columns.Add("CategoryId", typeof(Int32));

dtBook.Columns.Add("Qoh", typeof(string));

dtBook.Columns.Add("UnitPrice", typeof(Int32));

dtBook.PrimaryKey = new DataColumn[] { dtBook.Columns["ISBN"] };

daBook = new SqlDataAdapter("SELECT \* FROM Books", UtillityDB.ConnectDB());

sqlBuilder = new SqlCommandBuilder(daBook);

daBook.Fill(dsBook.Tables["Books"]);

dsAuthor = new DataSet("AuthorDB");

dtAuthor = new DataTable("Authors");

dsAuthor.Tables.Add(dtAuthor);

dtAuthor.Columns.Add("AuthorId", typeof(Int32));

dtAuthor.Columns.Add("AuthorName", typeof(string));

dtAuthor.PrimaryKey = new DataColumn[] { dtAuthor.Columns["AuthorId"] };

daAuthor = new SqlDataAdapter("SELECT \* FROM Authors", UtillityDB.ConnectDB());

sqlBuilder = new SqlCommandBuilder(daAuthor);

daAuthor.Fill(dsAuthor.Tables["Authors"]);

dsCategory = new DataSet("CategoryDB");

dtCategory = new DataTable("Categories");

dsCategory.Tables.Add(dtCategory);

dtCategory.Columns.Add("CategoryId", typeof(Int32));

dtCategory.Columns.Add("CategoryName", typeof(string));

dtCategory.PrimaryKey = new DataColumn[] { dtCategory.Columns["CategoryId"] };

daCategory = new SqlDataAdapter("SELECT \* FROM Categories", UtillityDB.ConnectDB());

sqlBuilder = new SqlCommandBuilder(daCategory);

daCategory.Fill(dsCategory.Tables["Categories"]);

dsPublisher = new DataSet("PublisherDB");

dtPublisher = new DataTable("Publishers");

dsPublisher.Tables.Add(dtPublisher);

dtPublisher.Columns.Add("PublisherId", typeof(Int32));

dtPublisher.Columns.Add("PublisherName", typeof(string));

dtPublisher.PrimaryKey = new DataColumn[] { dtPublisher.Columns["PublisherId"] };

daPublisher = new SqlDataAdapter("SELECT \* FROM Publishers", UtillityDB.ConnectDB());

sqlBuilder = new SqlCommandBuilder(daPublisher);

daPublisher.Fill(dsPublisher.Tables["Publishers"]);

}

private void buttonlistauthor\_Click(object sender, EventArgs e)

{

Author author = new Author();

dataGridViewauthor.DataSource = author.AuthorList();

}

private void buttonlistCategory\_Click(object sender, EventArgs e)

{

Category category = new Category();

dataGridViewcategory.DataSource = category.CategoryList();

}

private void buttonUpdatedatabasebook\_Click(object sender, EventArgs e)

{

daBook.Update(dsBook.Tables["Books"]);

MessageBox.Show("Database has been updated sucessfully.", "Confirmation");

}

private void buttonDeletebook\_Click(object sender, EventArgs e)

{

string searchisbn = textBoxisbn.Text.Trim();

DataRow drBook = dtBook.Rows.Find(searchisbn);

drBook.Delete();

MessageBox.Show(drBook.RowState.ToString());

}

private void buttonUpdatebook\_Click(object sender, EventArgs e)

{

string searchisbn = textBoxisbn.Text.Trim();

DataRow drBook = dtBook.Rows.Find(searchisbn);

drBook["BookTitle"] =textBoxtitle.Text.Trim();

drBook["Isbn"] =textBoxisbn.Text.Trim();

drBook["PublisherId"] = Convert.ToInt32(textBoxPublisher.Text.Trim());

drBook["AuthorId"] = Convert.ToInt32(textBoxauthor.Text.Trim());

drBook["CategoryId"] = Convert.ToInt32(textBoxcategory.Text.Trim());

drBook["Qoh"] =textBoxqoh.Text.Trim();

drBook["UnitPrice"] =Convert.ToInt32(textBoxprice.Text.Trim());

MessageBox.Show(drBook.RowState.ToString());

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Final\_Project.BLL;

using Final\_Project.DAL;

using System.Data.SqlClient;

namespace Final\_Project.GUI

{

public partial class LoginForm : Form

{

public static string userid= null;

public LoginForm()

{

InitializeComponent();

}

private void buttonexit\_Click(object sender, EventArgs e)

{

this.Close();

}

private void buttonlogin\_Click(object sender, EventArgs e)

{

int Userid = Convert.ToInt32(textBoxuserid.Text.Trim());

string Password = textBoxpassword.Text.Trim();

if (Userid == 1111 && Password == "henrybrown")

{

FormEmployees formEmployees = new FormEmployees();

this.Hide();

formEmployees.ShowDialog();

}

else if (Userid == 2222 && Password == "thomasmoore")

{

Sales\_Manager sales\_Manager = new Sales\_Manager();

this.Hide();

sales\_Manager.ShowDialog();

}

else if (Userid == 3333 && Password == "peterwang")

{

InventoryControllerForm inventoryControllerForm = new InventoryControllerForm();

this.Hide();

inventoryControllerForm.ShowDialog();

}

else if (Userid == 4444 && Password == "jennifer")

{

OrderManagement orderManagement = new OrderManagement();

this.Hide();

orderManagement.ShowDialog();

}

else

{

MessageBox.Show("Wrong User or Password");

}

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Final\_Project.BLL;

using Final\_Project.DAL;

using System.Data.SqlClient;

namespace Final\_Project.GUI

{

public partial class Sales\_Manager : Form

{

SqlDataAdapter daCustomer;

DataSet dsCustomer;

DataTable dtCustomer;

SqlCommandBuilder sqlBuilder;

Customer aCustomer = new Customer();

public Sales\_Manager()

{

InitializeComponent();

}

private void buttonlistallcustomers\_Click(object sender, EventArgs e)

{

Customer customer = new Customer();

dataGridViewlistcustomers.DataSource = customer.CustomersList();

}

private void buttonSave\_Click(object sender, EventArgs e)

{

DataRow dr = dtCustomer.NewRow();

dr["CustomerId"] = textBoxcustomerid.Text.Trim();

dr["CustomerName"] = textBoxcustomername.Text.Trim();

dr["StreetName"] = textBoxstreename.Text.Trim();

dr["Province"] = textBoxprovince.Text.Trim();

dr["City"] = textBoxcity.Text.Trim();

dr["PostalCode"] = textBoxpostalcode.Text.Trim();

dr["CreditLimit"] = Convert.ToInt32(textBoxcreditlimit.Text.Trim());

dr["ContactEmail"] = textBoxemail.Text.Trim();

dr["ContactName"] = textBoxcontactname.Text.Trim();

dtCustomer.Rows.Add(dr);

MessageBox.Show(dr.RowState.ToString());

}

private void buttonUpdate\_Click(object sender, EventArgs e)

{

string customerID = textBoxcustomerid.Text.Trim();

DataRow drCustomer = dtCustomer.Rows.Find(customerID);

drCustomer["CustomerId"] = textBoxcustomerid.Text.Trim();

drCustomer["CustomerName"] = textBoxcustomername.Text.Trim();

drCustomer["StreetName"] = textBoxstreename.Text.Trim();

drCustomer["Province"] = textBoxprovince.Text.Trim();

drCustomer["City"] = textBoxcity.Text.Trim();

drCustomer["PostalCode"] = textBoxpostalcode.Text.Trim();

drCustomer["CreditLimit"] = textBoxcreditlimit.Text.Trim();

drCustomer["ContactEmail"] = textBoxemail.Text.Trim();

drCustomer["ContactName"] = textBoxcontactname.Text.Trim();

MessageBox.Show(drCustomer.RowState.ToString());

}

private void buttonDelete\_Click(object sender, EventArgs e)

{

string searchId = textBoxcustomerid.Text.ToString();

DataRow drCustomer = dtCustomer.Rows.Find(searchId);

drCustomer.Delete();

MessageBox.Show(drCustomer.RowState.ToString());

}

private void buttonSearch\_Click(object sender, EventArgs e)

{

string searchID = textBoxsearch.Text.ToString();

DataRow drCustomer = dtCustomer.Rows.Find(searchID);

if (drCustomer != null)

{

textBoxcustomerid.Text = drCustomer["CustomerId"].ToString();

textBoxcustomername.Text= drCustomer["CustomerName"].ToString();

textBoxstreename.Text= drCustomer["StreetName"].ToString();

textBoxprovince.Text= drCustomer["Province"].ToString();

textBoxcity.Text= drCustomer["City"].ToString();

textBoxpostalcode.Text= drCustomer["PostalCode"].ToString();

textBoxcreditlimit.Text= drCustomer["CreditLimit"].ToString();

textBoxemail.Text= drCustomer["ContactEmail"].ToString();

textBoxcontactname.Text = drCustomer["ContactName"].ToString();

}

else

{

MessageBox.Show("Customer not found!", "Invalid Customer ID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void Sales\_Manager\_Load(object sender, EventArgs e)

{

dsCustomer = new DataSet("CustomerDS");

dtCustomer = new DataTable("Customers");

dsCustomer.Tables.Add(dtCustomer);

dtCustomer.Columns.Add("CustomerId", typeof(string));

dtCustomer.Columns.Add("CustomerName", typeof(string));

dtCustomer.Columns.Add("StreetName", typeof(string));

dtCustomer.Columns.Add("Province", typeof(string));

dtCustomer.Columns.Add("City", typeof(string));

dtCustomer.Columns.Add("PostalCode", typeof(string));

dtCustomer.Columns.Add("CreditLimit", typeof(Int32));

dtCustomer.Columns.Add("ContactEmail", typeof(string));

dtCustomer.Columns.Add("ContactName", typeof(string));

dtCustomer.PrimaryKey = new DataColumn[] { dtCustomer.Columns["CustomerId"] };

daCustomer = new SqlDataAdapter("SELECT \* FROM Customers", UtillityDB.ConnectDB());

sqlBuilder = new SqlCommandBuilder(daCustomer);

daCustomer.Fill(dsCustomer.Tables["Customers"]);

}

private void button1\_Click(object sender, EventArgs e)

{

daCustomer.Update(dsCustomer.Tables["Customers"]);

MessageBox.Show("Database has been updated sucessfully.", "Confirmation");

}

private void buttonexit\_Click(object sender, EventArgs e)

{

DialogResult exit = MessageBox.Show("Do you really want to exit the application?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Information);

this.Hide();

LoginForm loginForm = new LoginForm();

loginForm.ShowDialog();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Text.RegularExpressions;

namespace Final\_Project.VALIDATION

{

public static class Validator

{

public static bool IsValidId(string input)

{

if (!(Regex.IsMatch(input, @"^\d{4}$")))

{

return false;

}

return true;

}

public static bool IsValidName(string input)

{

if (input.Length == 0)

{

return false;

}

for (int i = 0; i < input.Length; i++)

{

if (!(Char.IsLetter(input[i])) && !(Char.IsWhiteSpace(input[i])))

{

return false;

}

}

return true;

}

public static bool IsEmpty(string input)

{

if (input.Length == 0)

{

return true;

}

return false;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

using Final\_Project.GUI;

namespace Final\_Project

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new LoginForm());

}

}

}