|  |
| --- |
|  |
| Hi-Tech Distribution Inc. Management System  Presented to Mr. Quang Hoang Cao |
| |  |  |  | | --- | --- | --- | | Mahan Moulaei | 12/15/20 | 420-DA3-AS | |

Table of Contents

[Project Description 1](#_Toc58963564)

[Project Design 2](#_Toc58963565)

[Database Design: 2](#_Toc58963566)

[Design of Application Domain Classes, and Class Diagram – Design of The Data Access Classes 3](#_Toc58963567)

[Design of The GUI Classes 4](#_Toc58963568)

[Project Implementation 5](#_Toc58963569)

[UtilityDB 5](#_Toc58963570)

[Employee 6](#_Toc58963571)

[User 11](#_Toc58963572)

[Category 15](#_Toc58963573)

[Publisher 19](#_Toc58963574)

[Book 23](#_Toc58963575)

[Job 29](#_Toc58963576)

[Customer 31](#_Toc58963577)

[Order 34](#_Toc58963578)

[OrderLine 35](#_Toc58963579)

[Project Testing 36](#_Toc58963580)

[Project Deployment 39](#_Toc58963581)

[MIS Manager 40](#_Toc58963582)

[Sales Manager 44](#_Toc58963583)

[Inventory Controller 50](#_Toc58963584)

[Order Clerk 54](#_Toc58963585)

[Conclusion 59](#_Toc58963586)

# Project Description

Hi-Tech Distribution Inc. is supplying books to nearly all the colleges and universities in Quebec. Hi-Tech Distribution Inc. Management System is a Windows-Form based application written in C# language with Microsoft Visual Studio 2019, and Microsoft SQL Server Management Studio 2018.

The application provides control of 4 areas of the Hi-Tech Company. The application is divided in four modules:

1. MIS Manager Module: Allows MIS Manager(s) to Save, Update, Delete, List, and Search Employees Info. This Module also allows MIS Manager(s) to create an User Account for each Employee.
2. Sales Manager Module: Allows Sale Manager(s) to Save, Update, Delete, List, and Search Customers Info.
3. Inventory Controller Module: Allows Inventory Controller(s) to Save, Update, Delete, List, and Search Books Info. This Module also allows Inventory Controller(s) to Save, Update, Delete, List, and Search Book Categories and Book Publishers Info.
4. Order Clerk Module: Allows Order Clerk(s) to Save, Update, Delete, List, and Search Customers’ Orders.

# Project Design

## Database Design:

In SQL Server, 11 tables were created to provide all of the required functionalities for this application. The database design diagram is at follows:

Diagram, schematic

Description automatically generated

## Design of Application Domain Classes, and Class Diagram – Design of The Data Access Classes

In the Business Level Layer for the Connected-Mode and Disconnected-Mode of the application, these classes that contains properties and methods related to each database tables were created and used:

A picture containing text

Description automatically generated

In the Data Access Layer for the Connected-Mode and Disconnected-Mode of the application, these classes that contains constructors related to each database tables were created and used:

Text

Description automatically generated

For the Entity Framework of the application, these class diagram and classes that contains properties were auto-generated by the Visual Studio 2019 based on the tables and models that exists in the database:

Text

Description automatically generatedA picture containing text, screenshot, electronics, display

Description automatically generated

## Design of The GUI Classes

For the whole parts of the project, only 1 windows form was used, and different parts of the project were divided into different tab pages. Also, another form was also added to show the company’s “about us” section:

Text

Description automatically generated

# Project Implementation

In Visual Studio, within the 2 BLL and DAL folders, two libraries for connected-mode and disconnected-were created.

The library for connected mode has 6 classes in BLL folder and 7 classes in DAL folder:

### UtilityDB

The UtilityDB contains the method ConnectDB, to connect the application to the database.

namespace FinalProject.DAL

{

public static class UtilityDB

{

public static SqlConnection ConnectDB()

{

SqlConnection connDB = new SqlConnection();

connDB.ConnectionString = ConfigurationManager.ConnectionStrings["Hi-TechDB\_Connection"].ConnectionString;

connDB.Open();

return connDB;

}

}

}

### Employee

The Employee class contains the properties and methods related to Employees table of database.

namespace FinalProject.BLL

{

public class Employee

{

private int empID;

private string empFName;

private string empLName;

private string empPhone;

private string empEmail;

private int empJobID;

public Employee(int empID, string empFName, string empLName, string empPhone, string empEmail, int empJobID)

{

this.empID = empID;

this.empFName = empFName;

this.empLName = empLName;

this.empPhone = empPhone;

this.empEmail = empEmail;

this.empJobID = empJobID;

}

public Employee()

{

}

public int EmployeeID { get => empID; set => empID = value; }

public string EmployeeFName { get => empFName; set => empFName = value; }

public string EmployeeLName { get => empLName; set => empLName = value; }

public string EmployeePhone { get => empPhone; set => empPhone = value; }

public string EmployeeEmail { get => empEmail; set => empEmail = value; }

public int EmployeeJobID { get => empJobID; set => empJobID = value; }

public void EmpSaveEmployee(Employee emp)

{

EmployeeDB.EmpSaveRecord(emp);

}

public List<Employee> GetEmployeeList()

{

return EmployeeDB.EmpGetRecordList();

}

public Employee GetEmployee(int eId)

{

return EmployeeDB.EmpGetRecord(eId);

}

public List<Employee> GetEmployeeList(string name)

{

return EmployeeDB.EmpGetRecordList();

}

public void UpdateEmployee(Employee emp)

{

EmployeeDB.EmpUpdateRecord(emp);

}

public void DeleteEmployee(Employee emp)

{

EmployeeDB.EmpDeleteRecord(emp);

}

public List<Employee> GetEmployeeListF(string firstName)

{

return EmployeeDB.EmpGetRecordListFirstName(firstName);

}

public List<Employee> GetEmployeeListL(string lastName)

{

return EmployeeDB.EmpGetRecordListLastName(lastName);

}

}

}

The EmployeeDB class contains the constructors related to Employees table of database to make the Employee section of the application works.

namespace FinalProject.DAL

{

public static class EmployeeDB

{

public static void EmpSaveRecord(Employee emp)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdNewEmp = new SqlCommand();

cmdNewEmp.CommandText = "INSERT INTO Employees (EmployeeID,FirstName,LastName,PhoneNumber,Email,JobID)" +

"VALUES (@EmployeeID,@EmployeeFName,@EmployeeLName,@EmployeePhone,@EmployeeEmail,@EmployeeJobID)";

cmdNewEmp.Parameters.AddWithValue("@EmployeeID", emp.EmployeeID);

cmdNewEmp.Parameters.AddWithValue("@EmployeeFName", emp.EmployeeFName);

cmdNewEmp.Parameters.AddWithValue("@EmployeeLName", emp.EmployeeLName);

cmdNewEmp.Parameters.AddWithValue("@EmployeePhone", emp.EmployeePhone);

cmdNewEmp.Parameters.AddWithValue("@EmployeeEmail", emp.EmployeeEmail);

cmdNewEmp.Parameters.AddWithValue("@EmployeeJobID", emp.EmployeeJobID);

cmdNewEmp.Connection = connDB;

cmdNewEmp.ExecuteNonQuery();

connDB.Close();

}

public static Employee EmpGetRecord(int empID)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectEmp = new SqlCommand();

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

"WHERE EmployeeID = " + empID;

cmdSelectEmp.Connection = connDB;

SqlDataReader sqlReader = cmdSelectEmp.ExecuteReader();

Employee emp = new Employee();

if (sqlReader.Read())

{

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

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

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

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

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

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

}

else

{

emp = null;

}

return emp;

}

public static List<Employee> EmpGetRecordList()

{

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

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectEmp = new SqlCommand("SELECT \* FROM Employees", connDB);

SqlDataReader sqlReader = cmdSelectEmp.ExecuteReader();

Employee emp;

while (sqlReader.Read())

{

emp = new Employee();

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

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

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

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

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

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

listEmployee.Add(emp);

}

return listEmployee;

}

public static List<Employee> EmpGetRecordListFirstName(string firstName)

{

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

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectEmp = new SqlCommand("SELECT \* FROM Employees WHERE FirstName = '" + firstName + "'", connDB);

SqlDataReader sqlReader = cmdSelectEmp.ExecuteReader();

Employee emp;

while (sqlReader.Read())

{

emp = new Employee();

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

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

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

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

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

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

listEmployee.Add(emp);

}

return listEmployee;

}

public static List<Employee> EmpGetRecordListLastName(string lastName)

{

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

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectEmp = new SqlCommand("SELECT \* FROM Employees WHERE LastName = '" + lastName + "'", connDB);

SqlDataReader sqlReader = cmdSelectEmp.ExecuteReader();

Employee emp;

while (sqlReader.Read())

{

emp = new Employee();

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

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

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

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

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

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

listEmployee.Add(emp);

}

return listEmployee;

}

public static void EmpUpdateRecord(Employee emp)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdUpdateEmp = new SqlCommand();

//Parameterized Query

cmdUpdateEmp.CommandText = "UPDATE Employees " +

"SET EmployeeID='" + emp.EmployeeID + "', FirstName='" + emp.EmployeeFName + "' ,LastName='" + emp.EmployeeLName + "', PhoneNumber='" + emp.EmployeePhone + "' ,Email='" + emp.EmployeeEmail + "' ,JobID ='" + emp.EmployeeJobID + "' " +

"WHERE EmployeeID =" + emp.EmployeeID;

cmdUpdateEmp.Connection = connDB;

cmdUpdateEmp.ExecuteNonQuery();

connDB.Close();

}

public static void EmpDeleteRecord(Employee emp)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdDeleteEmp = new SqlCommand();

//Parameterized Query

cmdDeleteEmp.CommandText = "DELETE FROM Employees " +

"WHERE EmployeeID = " + emp.EmployeeID;

cmdDeleteEmp.Connection = connDB;

cmdDeleteEmp.ExecuteNonQuery();

connDB.Close();

}

}

}

### User

The User class contains the properties and methods related to UserAccounts table of database.

namespace FinalProject.BLL

{

public class User

{

private int userID;

private string userPass;

private int userEmpID;

private string userComment;

public User(int userID, string userPass, int userEmpID, string userComment)

{

this.userID = userID;

this.userPass = userPass;

this.userEmpID = userEmpID;

this.userComment = userComment;

}

public User()

{

}

public int UserID { get => userID; set => userID = value; }

public string UserPassword { get => userPass; set => userPass = value; }

public int UserEmployeeID { get => userEmpID; set => userEmpID = value; }

public string UserComment { get => userComment; set => userComment = value; }

public void SaveUser(User usr)

{

UserDB.SaveUserRecord(usr);

}

public List<User> GetUserList()

{

return UserDB.GetUserRecordList();

}

public List<User> GetUserList(string name)

{

return UserDB.GetUserRecordList();

}

public User GetUser(int uID)

{

return UserDB.UserGetRecord(uID);

}

public User GetUserPass(string pass)

{

return UserDB.UserGetRecordPass(pass);

}

public List<User> GetUserListPass(string password)

{

return UserDB.UserGetRecordListFirstPassword(password);

}

public void UpdateUser(User usr)

{

UserDB.UpdateUserRecord(usr);

}

public void DeleteUser(User usr)

{

UserDB.DeleteUserRecord(usr);

}

}

}

The UserDB class contains the constructors related to UserAccounts table of database to make the User section of the application works.

namespace FinalProject.DAL

{

public static class UserDB

{

public static void SaveUserRecord(User usr)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdNewUser = new SqlCommand();

cmdNewUser.CommandText = "INSERT INTO UserAccounts (UserID,Password,EmployeeID,Comment)" +

"VALUES (@UserID,@Password,@EmployeeID,@Comment)";

cmdNewUser.Parameters.AddWithValue("@UserID", usr.UserID);

cmdNewUser.Parameters.AddWithValue("@Password", usr.UserPassword);

cmdNewUser.Parameters.AddWithValue("@EmployeeID", usr.UserEmployeeID);

cmdNewUser.Parameters.AddWithValue("@Comment", usr.UserComment);

cmdNewUser.Connection = connDB;

cmdNewUser.ExecuteNonQuery();

connDB.Close();

}

public static List<User> GetUserRecordList()

{

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

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectUser = new SqlCommand("SELECT \* FROM UserAccounts", connDB);

SqlDataReader sqlReader = cmdSelectUser.ExecuteReader();

User usr;

while (sqlReader.Read())

{

usr = new User();

usr.UserID = Convert.ToInt32(sqlReader["UserID"]);

usr.UserPassword = sqlReader["Password"].ToString();

usr.UserEmployeeID = Convert.ToInt32(sqlReader["EmployeeID"]);

usr.UserComment = sqlReader["Comment"].ToString();

listUser.Add(usr);

}

return listUser;

}

public static User UserGetRecord(int userID)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectUser = new SqlCommand();

cmdSelectUser.CommandText = "SELECT \* FROM UserAccounts " +

"WHERE UserID = " + userID;

cmdSelectUser.Connection = connDB;

SqlDataReader sqlReader = cmdSelectUser.ExecuteReader();

User usr = new User();

if (sqlReader.Read())

{

usr.UserID = Convert.ToInt32(sqlReader["UserID"]);

usr.UserPassword = sqlReader["Password"].ToString();

usr.UserEmployeeID = Convert.ToInt32(sqlReader["EmployeeID"]);

usr.UserComment = sqlReader["Comment"].ToString();

}

else

{

usr = null;

}

return usr;

}

public static User UserGetRecordPass(string pass)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectUserPass = new SqlCommand();

cmdSelectUserPass.CommandText = "SELECT \* FROM UserAccounts " +

"WHERE Password= " + pass;

cmdSelectUserPass.Connection = connDB;

SqlDataReader sqlReader = cmdSelectUserPass.ExecuteReader();

User usr = new User();

if (sqlReader.Read())

{

usr.UserID = Convert.ToInt32(sqlReader["UserID"]);

usr.UserPassword = sqlReader["Password"].ToString();

usr.UserEmployeeID = Convert.ToInt32(sqlReader["EmployeeID"]);

usr.UserComment = sqlReader["Comment"].ToString();

}

else

{

usr = null;

}

return usr;

}

public static List<User> UserGetRecordListFirstPassword(string password)

{

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

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectUser = new SqlCommand("SELECT \* FROM UserAccounts WHERE Password = '" + password + "'", connDB);

SqlDataReader sqlReader = cmdSelectUser.ExecuteReader();

User usr;

while (sqlReader.Read())

{

usr = new User();

usr.UserID = Convert.ToInt32(sqlReader["UserID"]);

usr.UserPassword = sqlReader["Password"].ToString();

usr.UserEmployeeID = Convert.ToInt32(sqlReader["EmployeeID"]);

usr.UserComment = sqlReader["Comment"].ToString();

listUser.Add(usr);

}

return listUser;

}

public static void UpdateUserRecord(User usr)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdUpdateUser = new SqlCommand();

//Parameterized Query

cmdUpdateUser.CommandText = "UPDATE UserAccounts " +

"SET UserID='" + usr.UserID + "', Password='" + usr.UserPassword + "' ,EmployeeID='" + usr.UserEmployeeID + "', Comment='" + usr.UserComment + "' " +

"WHERE UserID =" + usr.UserID;

cmdUpdateUser.Connection = connDB;

cmdUpdateUser.ExecuteNonQuery();

connDB.Close();

}

public static void DeleteUserRecord(User usr)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdDeleteUser = new SqlCommand();

//Parameterized Query

cmdDeleteUser.CommandText = "DELETE FROM UserAccounts " +

"WHERE UserID = " + usr.UserID;

cmdDeleteUser.Connection = connDB;

cmdDeleteUser.ExecuteNonQuery();

connDB.Close();

}

}

}

### Category

The Category class contains the properties and methods related to Categories table of database.

namespace FinalProject.BLL

{

public class Category

{

private int categID;

private string categName;

public Category(int categID, string categName)

{

this.categID = categID;

this.categName = categName;

}

public Category()

{

}

public int CategoryID { get => categID; set => categID = value; }

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

//SAVE

public void SaveCategory(Category cat)

{

CategoryDB.SaveCategoryRecord(cat);

}

//DELETE

public void DeleteCategory(Category cat)

{

CategoryDB.DeleteCategoryRecord(cat);

}

//LIST

public List<Category> GetCategoryList()

{

return CategoryDB.GetCategoryList();

}

//UPDATE

public void UpdateCategory(Category cat)

{

CategoryDB.UpdateCategoryRecord(cat);

}

//SEARCH

public Category GetCategoryRecordID(int catId)

{

return CategoryDB.GetCategoryRecordID(catId);

}

public Category GetCategoryRecordName(string catName)

{

return CategoryDB.GetCategoryRecordName(catName);

}

}

}

The CategoryDB class contains the constructors related to Categories table of database to make the Category section of the application works.

namespace FinalProject.DAL

{

public static class CategoryDB

{

//SAVE

public static void SaveCategoryRecord(Category cat)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdNewCat = new SqlCommand();

cmdNewCat.CommandText = "INSERT INTO Categories (CategoryID,CategoryName)" +

"VALUES (@CategoryID,@CategoryName)";

cmdNewCat.Parameters.AddWithValue("@CategoryID", cat.CategoryID);

cmdNewCat.Parameters.AddWithValue("@CategoryName", cat.CategoryName);

cmdNewCat.Connection = connDB;

cmdNewCat.ExecuteNonQuery();

connDB.Close();

}

//DELETE

public static void DeleteCategoryRecord(Category cat)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdDeleteCat = new SqlCommand();

//Parameterized Query

cmdDeleteCat.CommandText = "DELETE FROM Categories " +

"WHERE CategoryID = " + cat.CategoryID;

cmdDeleteCat.Connection = connDB;

cmdDeleteCat.ExecuteNonQuery();

connDB.Close();

}

//LIST

public static List<Category> GetCategoryList()

{

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

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectCategory = new SqlCommand("SELECT \* FROM Categories", connDB);

SqlDataReader sqlReader = cmdSelectCategory.ExecuteReader();

Category cat;

while (sqlReader.Read())

{

cat = new Category();

cat.CategoryID = Convert.ToInt32(sqlReader["CategoryID"]);

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

listCategory.Add(cat);

}

return listCategory;

}

//UPDATE

public static void UpdateCategoryRecord(Category cat)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdUpdateCat = new SqlCommand();

//Parameterized Query

cmdUpdateCat.CommandText = "UPDATE Categories " +

"SET CategoryID='" + cat.CategoryID + "', CategoryName='" + cat.CategoryName + "' " +

"WHERE CategoryID =" + cat.CategoryID;

cmdUpdateCat.Connection = connDB;

cmdUpdateCat.ExecuteNonQuery();

connDB.Close();

}

//SEARCH

public static Category GetCategoryRecordID(int caId)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectCatID = new SqlCommand();

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

"WHERE CategoryID = " + caId;

cmdSelectCatID.Connection = connDB;

SqlDataReader sqlReader = cmdSelectCatID.ExecuteReader();

Category cat = new Category();

if (sqlReader.Read())

{

cat.CategoryID = Convert.ToInt32(sqlReader["CategoryID"]);

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

}

else

{

cat = null;

}

return cat;

}

public static Category GetCategoryRecordName(string caName)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectCatName = new SqlCommand();

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

"WHERE CategoryName = '" + caName + "' ";

cmdSelectCatName.Connection = connDB;

SqlDataReader sqlReader = cmdSelectCatName.ExecuteReader();

Category cat = new Category();

if (sqlReader.Read())

{

cat.CategoryID = Convert.ToInt32(sqlReader["CategoryID"]);

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

}

else

{

cat = null;

}

return cat;

}

}

}

### Publisher

The Publisher class contains the properties and methods related to Publishers table of database.

namespace FinalProject.BLL

{

public class Publisher

{

private int pubID;

private string pubName;

private string pubWeb;

public Publisher(int pubID, string pubName, string pubWeb)

{

this.pubID = pubID;

this.pubName = pubName;

this.pubWeb = pubWeb;

}

public Publisher()

{

}

public int PublisherID { get => pubID; set => pubID = value; }

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

public string PublisherWeb { get => pubWeb; set => pubWeb = value; }

//SAVE

public void SavePublisher(Publisher pub)

{

PublisherDB.SavePublisherRecord(pub);

}

//DELETE

public void DeletePublisher(Publisher pub)

{

PublisherDB.DeletePublisherRecord(pub);

}

//LIST

public List<Publisher> GetPublisherList()

{

return PublisherDB.GetPublisherList();

}

//UPDATE

public void UpdatePublisher(Publisher pub)

{

PublisherDB.UpdatePublisherRecord(pub);

}

//SEARCH

public Publisher GetPublisher(int pId)

{

return PublisherDB.GetPublisher(pId);

}

public Publisher GetPublisherRecordName(string pubName)

{

return PublisherDB.GetPublisherRecordName(pubName);

}

public Publisher GetPublisherRecordWeb(string pubWeb)

{

return PublisherDB.GetPublisherRecordWeb(pubWeb);

}

}

}

The PublisherDB class contains the constructors related to Publishers table of database to make the Publisher section of the application works.

namespace FinalProject.DAL

{

public static class PublisherDB

{

//SAVE

public static void SavePublisherRecord(Publisher pub)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdNewPub = new SqlCommand();

cmdNewPub.CommandText = "INSERT INTO Publishers (PublisherID,PublisherName,WebAddress)" +

"VALUES (@PublisherID,@PublisherName,@WebAddress)";

cmdNewPub.Parameters.AddWithValue("@PublisherID", pub.PublisherID);

cmdNewPub.Parameters.AddWithValue("@PublisherName", pub.PublisherName);

cmdNewPub.Parameters.AddWithValue("@WebAddress", pub.PublisherWeb);

cmdNewPub.Connection = connDB;

cmdNewPub.ExecuteNonQuery();

connDB.Close();

}

//DELETE

public static void DeletePublisherRecord(Publisher pub)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdDeletePub = new SqlCommand();

//Parameterized Query

cmdDeletePub.CommandText = "DELETE FROM Publishers " +

"WHERE PublisherID = " + pub.PublisherID;

cmdDeletePub.Connection = connDB;

cmdDeletePub.ExecuteNonQuery();

connDB.Close();

}

//LIST

public static List<Publisher> GetPublisherList()

{

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

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectPublisher = new SqlCommand("SELECT \* FROM Publishers", connDB);

SqlDataReader sqlReader = cmdSelectPublisher.ExecuteReader();

Publisher pub;

while (sqlReader.Read())

{

pub = new Publisher();

pub.PublisherID = Convert.ToInt32(sqlReader["PublisherID"]);

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

pub.PublisherWeb = sqlReader["WebAddress"].ToString();

listPublisher.Add(pub);

}

return listPublisher;

}

//UPDATE

public static void UpdatePublisherRecord(Publisher pub)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdUpdatePub = new SqlCommand();

//Parameterized Query

cmdUpdatePub.CommandText = "UPDATE Publishers " +

"SET PublisherID='" + pub.PublisherID + "', PublisherName='" + pub.PublisherName + "', WebAddress='" + pub.PublisherWeb + "' " +

"WHERE PublisherID =" + pub.PublisherID;

cmdUpdatePub.Connection = connDB;

cmdUpdatePub.ExecuteNonQuery();

connDB.Close();

}

//SEARCH

public static Publisher GetPublisher(int pId)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectPubID = new SqlCommand();

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

"WHERE PublisherID = " + pId;

cmdSelectPubID.Connection = connDB;

SqlDataReader sqlReader = cmdSelectPubID.ExecuteReader();

Publisher pub = new Publisher();

if (sqlReader.Read())

{

pub.PublisherID = Convert.ToInt32(sqlReader["PublisherID"]);

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

pub.PublisherWeb = sqlReader["WebAddress"].ToString();

}

else

{

pub = null;

}

return pub;

}

public static Publisher GetPublisherRecordName(string pubName)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectPubName = new SqlCommand();

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

"WHERE PublisherName = '" + pubName + "' ";

cmdSelectPubName.Connection = connDB;

SqlDataReader sqlReader = cmdSelectPubName.ExecuteReader();

Publisher pub = new Publisher();

if (sqlReader.Read())

{

pub.PublisherID = Convert.ToInt32(sqlReader["PublisherID"]);

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

pub.PublisherWeb = sqlReader["WebAddress"].ToString();

}

else

{

pub = null;

}

return pub;

}

public static Publisher GetPublisherRecordWeb(string pubWeb)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectPubWeb = new SqlCommand();

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

"WHERE WebAddress = '" + pubWeb + "' ";

cmdSelectPubWeb.Connection = connDB;

SqlDataReader sqlReader = cmdSelectPubWeb.ExecuteReader();

Publisher pub = new Publisher();

if (sqlReader.Read())

{

pub.PublisherID = Convert.ToInt32(sqlReader["PublisherID"]);

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

pub.PublisherWeb = sqlReader["WebAddress"].ToString();

}

else

{

pub = null;

}

return pub;

}

}

}

### Book

The Book class contains the properties and methods related to Books table of database.

namespace FinalProject.BLL

{

public class Book

{

private Int64 bukISBN;

private string bukTitle;

private int bukQOH;

private float bukUnitPrice;

private int bukCategoryID;

private int bukPublisherID;

public Book(Int64 bukISBN, string bukTitle, int bukQOH, float bukUnitPrice, int bukCategoryID, int bukPublisherID)

{

this.bukISBN = bukISBN;

this.bukTitle = bukTitle;

this.bukQOH = bukQOH;

this.bukUnitPrice = bukUnitPrice;

this.bukCategoryID = bukCategoryID;

this.bukPublisherID = bukPublisherID;

}

public Book()

{

}

public Int64 BookISBN { get => bukISBN; set => bukISBN = value; }

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

public int BookQOH { get => bukQOH; set => bukQOH = value; }

public float BookPrice { get => bukUnitPrice; set => bukUnitPrice = value; }

public int BookCategoryID { get => bukCategoryID; set => bukCategoryID = value; }

public int BookPublisherID { get => bukPublisherID; set => bukPublisherID = value; }

//SAVE

public void SaveBook(Book buk)

{

BookDB.SaveBookRecord(buk);

}

//DELETE

public void DeleteBook(Book buk)

{

BookDB.DeleteBookRecord(buk);

}

//LIST

public List<Book> GetBookList()

{

return BookDB.GetBookList();

}

public List<Book> GetBookList(string name)

{

return BookDB.GetBookList();

}

//UPDATE

public void UpdateBook(Book buk)

{

BookDB.UpdateBookRecord(buk);

}

//SEARCH

public Book GetBookRecord(long bukISBN)

{

return BookDB.GetBookRecord(bukISBN);

}

public List<Book> GetBookTitleRecordList(string bookTitle)

{

return BookDB.GetBookTitleRecordList(bookTitle);

}

public List<Book> GetBookCategoryIDRecordList(int categoryID)

{

return BookDB.GetBookCategoryIDRecordList(categoryID);

}

public List<Book> GetBookPublisherIDRecordList(int publisherID)

{

return BookDB.GetBookPublisherIDRecordList(publisherID);

}

//SEARCH BY QUANTITY

public Book GetBookQuantity(long bukISBN)

{

return BookDB.GetBookQuantity(bukISBN);

}

}

}

The BookDB class contains the constructors related to Books table of database to make the Book section of the application works.

namespace FinalProject.DAL

{

public static class BookDB

{

//SAVE

public static void SaveBookRecord(Book buk)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdNewBuk = new SqlCommand();

cmdNewBuk.CommandText = "INSERT INTO Books (ISBN,BookTitle,QOH,UnitPrice,CategoryID,PublisherID)" +

"VALUES (@BookISBN,@BookTitle,@BookQOH,@BookPrice,@BookCategoryID,@BookPublisherID)";

cmdNewBuk.Parameters.AddWithValue("@BookISBN", buk.BookISBN);

cmdNewBuk.Parameters.AddWithValue("@BookTitle", buk.BookTitle);

cmdNewBuk.Parameters.AddWithValue("@BookQOH", buk.BookQOH);

cmdNewBuk.Parameters.AddWithValue("@BookPrice", buk.BookPrice);

cmdNewBuk.Parameters.AddWithValue("@BookCategoryID", buk.BookCategoryID);

cmdNewBuk.Parameters.AddWithValue("@BookPublisherID", buk.BookPublisherID);

cmdNewBuk.Connection = connDB;

cmdNewBuk.ExecuteNonQuery();

connDB.Close();

}

//DELETE

public static void DeleteBookRecord(Book buk)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdDeleteBuk = new SqlCommand();

//Parameterized Query

cmdDeleteBuk.CommandText = "DELETE FROM Books " +

"WHERE ISBN = " + buk.BookISBN;

cmdDeleteBuk.Connection = connDB;

cmdDeleteBuk.ExecuteNonQuery();

connDB.Close();

}

//LIST

public static List<Book> GetBookList()

{

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

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectBook = new SqlCommand("SELECT \* FROM Books", connDB);

SqlDataReader sqlReader = cmdSelectBook.ExecuteReader();

Book buk;

while (sqlReader.Read())

{

buk = new Book();

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

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

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

buk.BookPrice = (float)Convert.ToDecimal(sqlReader["UnitPrice"]);

buk.BookCategoryID = Convert.ToInt32(sqlReader["CategoryID"]);

buk.BookPublisherID = Convert.ToInt32(sqlReader["PublisherID"]);

listBook.Add(buk);

}

return listBook;

}

//UPDATE

public static void UpdateBookRecord(Book Buk)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdUpdateBuk = new SqlCommand();

//Parameterized Query

cmdUpdateBuk.CommandText = "UPDATE Books " +

"SET ISBN='" + Buk.BookISBN + "', BookTitle='" + Buk.BookTitle + "', QOH='" + Buk.BookQOH + "', UnitPrice='" + Buk.BookPrice + "', CategoryID='" + Buk.BookCategoryID + "', PublisherID ='" + Buk.BookPublisherID + "' " +

"WHERE ISBN =" + Buk.BookISBN;

cmdUpdateBuk.Connection = connDB;

cmdUpdateBuk.ExecuteNonQuery();

connDB.Close();

}

//SEARCH

public static Book GetBookRecord(long bukISBN)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectBuk = new SqlCommand();

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

"WHERE ISBN = " + bukISBN;

cmdSelectBuk.Connection = connDB;

SqlDataReader sqlReader = cmdSelectBuk.ExecuteReader();

Book buk = new Book();

if (sqlReader.Read())

{

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

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

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

buk.BookPrice = (float)Convert.ToDecimal(sqlReader["UnitPrice"]);

buk.BookCategoryID = Convert.ToInt32(sqlReader["CategoryID"]);

buk.BookPublisherID = Convert.ToInt32(sqlReader["PublisherID"]);

}

else

{

buk = null;

}

return buk;

}

//SEARCH BY QUANTITY

public static Book GetBookQuantity(long bukISBN)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectBuk = new SqlCommand();

cmdSelectBuk.CommandText = "SELECT QOH FROM Books " +

"WHERE ISBN = " + bukISBN;

cmdSelectBuk.Connection = connDB;

SqlDataReader sqlReader = cmdSelectBuk.ExecuteReader();

Book buk = new Book();

if (sqlReader.Read())

{

//buk.BookISBN = Convert.ToInt64(sqlReader["ISBN"]);

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

}

else

{

buk = null;

}

return buk;

}

public static List<Book> GetBookTitleRecordList(string bookTitle)

{

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

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectBook = new SqlCommand("SELECT \* FROM Books WHERE BookTitle = '" + bookTitle + "'", connDB);

SqlDataReader sqlReader = cmdSelectBook.ExecuteReader();

Book buk;

while (sqlReader.Read())

{

buk = new Book();

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

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

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

buk.BookPrice = (float)Convert.ToDecimal(sqlReader["UnitPrice"]);

buk.BookCategoryID = Convert.ToInt32(sqlReader["CategoryID"]);

buk.BookPublisherID = Convert.ToInt32(sqlReader["PublisherID"]);

listBook.Add(buk);

}

return listBook;

}

public static List<Book> GetBookCategoryIDRecordList(int categoryID)

{

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

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectBook = new SqlCommand("SELECT \* FROM Books WHERE CategoryID = '" + categoryID + "'", connDB);

SqlDataReader sqlReader = cmdSelectBook.ExecuteReader();

Book buk;

while (sqlReader.Read())

{

buk = new Book();

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

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

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

buk.BookPrice = (float)Convert.ToDecimal(sqlReader["UnitPrice"]);

buk.BookCategoryID = Convert.ToInt32(sqlReader["CategoryID"]);

buk.BookPublisherID = Convert.ToInt32(sqlReader["PublisherID"]);

listBook.Add(buk);

}

return listBook;

}

public static List<Book> GetBookPublisherIDRecordList(int publisherID)

{

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

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectBook = new SqlCommand("SELECT \* FROM Books WHERE PublisherID = '" + publisherID + "'", connDB);

SqlDataReader sqlReader = cmdSelectBook.ExecuteReader();

Book buk;

while (sqlReader.Read())

{

buk = new Book();

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

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

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

buk.BookPrice = (float)Convert.ToDecimal(sqlReader["UnitPrice"]);

buk.BookCategoryID = Convert.ToInt32(sqlReader["CategoryID"]);

buk.BookPublisherID = Convert.ToInt32(sqlReader["PublisherID"]);

listBook.Add(buk);

}

return listBook;

}

}

}

### Job

The Job class contains the properties and methods related to Jobs table of database.

namespace FinalProject.BLL

{

public class Job

{

private int jobID;

private string jobTitle;

public Job(int jobID, string jobTitle)

{

this.jobID = jobID;

this.jobTitle = jobTitle;

}

public Job()

{

}

public int JobID { get => jobID; set => jobID = value; }

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

public Job GetJob(int eId)

{

return JobDB.GetJob(eId);

}

}

}

The JobDB class contains the constructors related to Jobs table of database to make the Employee section of the application works.

namespace FinalProject.DAL

{

public static class JobDB

{

public static Job GetJob(int eId)

{

SqlConnection connDB = UtilityDB.ConnectDB();

SqlCommand cmdSelectJobID = new SqlCommand();

cmdSelectJobID.CommandText = "SELECT \* FROM Jobs " +

"WHERE JobID = " + eId;

cmdSelectJobID.Connection = connDB;

SqlDataReader sqlReader = cmdSelectJobID.ExecuteReader();

Job job = new Job();

if (sqlReader.Read())

{

job.JobID = Convert.ToInt32(sqlReader["JobID"]);

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

}

else

{

job = null;

}

return job;

}

}

}

The library for disconnected mode has 1 class in BLL folder and 2 classes in DAL folder which one of the classes in DAL is the UtilityDB used for connected mode:

### Customer

The Customer class contains the properties and methods related to Customers table of database.

namespace FinalProject.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 Customer(int customerID, string customerName, string streetName, string province, string city, string postalCode, string contactName, string contactEmail, int creditLimit)

{

this.customerID = customerID;

this.customerName = customerName;

this.streetName = streetName;

this.province = province;

this.city = city;

this.postalCode = postalCode;

this.contactName = contactName;

this.contactEmail = contactEmail;

this.creditLimit = creditLimit;

}

public Customer()

{

}

public int CustomerID { get => customerID; set => customerID = value; }

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

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

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

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

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

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

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

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

public List<Customer> GetCustomerList()

{

return CustomerDB.CustomerGetRecordList();

}

}

}

The CustomerDB class contains the constructors related to Customers table of database to make the Customer section of the application works.

namespace FinalProject.DAL

{

public static class CustomerDB

{

public static SqlDataAdapter SDA = new SqlDataAdapter();

public static SqlConnection connDB = UtilityDB.ConnectDB();

public static List<Customer> CustomerGetRecordList()

{

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

using (SqlConnection connDB = UtilityDB.ConnectDB())

//SqlConnection connDB = UtilityDB.ConnectDB();

{

Customer cus;

SqlCommand cmdSelectCust = new SqlCommand("SELECT \* FROM Customers", connDB);

SqlDataReader sqlReader = cmdSelectCust.ExecuteReader();

if (sqlReader.HasRows)

{

while (sqlReader.Read())

{

cus = new Customer();

cus.CustomerID = Convert.ToInt32(sqlReader["CustomerID"]);

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

cus.CustomerStreet = sqlReader["StreetName"].ToString();

cus.CustomerProvince = sqlReader["Province"].ToString();

cus.CustomerCity = sqlReader["City"].ToString();

cus.CustomerPostal = sqlReader["PostalCode"].ToString();

cus.CustomerContactName = sqlReader["ContactName"].ToString();

cus.CustomerContactEmail = sqlReader["ContactEmail"].ToString();

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

listCustomer.Add(cus);

}

}

else

{

listCustomer = null;

}

}

return listCustomer;

}

}

}

In Visual Studio, 1 Entity Framework was created. The Entity Framework auto-generated 2 classes based on the database models and tables:

### Order

namespace FinalProject

{

using System;

using System.Collections.Generic;

public partial class Order

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Order()

{

this.OrderLines = new HashSet<OrderLine>();

}

public int OrderID { get; set; }

public string OrderDate { get; set; }

public string OrderType { get; set; }

public string RequiredDate { get; set; }

public string ShippingDate { get; set; }

public string OrderStatus { get; set; }

public int CustomerID { get; set; }

public int EmployeeID { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<OrderLine> OrderLines { get; set; }

}

}

### OrderLine

namespace FinalProject

{

using System;

using System.Collections.Generic;

public partial class OrderLine

{

public int OrderID { get; set; }

public long ISBN { get; set; }

public int QuantityOrdered { get; set; }

public virtual Order Order { get; set; }

}

}

# Project Testing

|  |  |
| --- | --- |
| INPUT | RESULT |
| Log in – User ID: 111  Password: hebr | Successful |
| Employee Section  Add Employee | Successful |
| Employee Section  List Employee | Successful |
| Employee Section  Search Employee | Successful |
| Employee Section  Update Employee | Successful |
| Employee Section  Delete Employee | Successful |
| User Section  Add User | Successful |
| User Section  List User | Successful |
| User Section  Search User | Successful |
| User Section  Update User | Successful |
| User Section  Delete User | Successful |

|  |  |
| --- | --- |
| INPUT | RESULT |
| Log in – User ID: 222  Password: thmo | Successful |
| Customer Section  Add Customer | Successful |
| Customer Section  List Customer From DataSet | Successful |
| Customer Section  Search Customer | Successful |
| Customer Section  Update Customer | Successful |
| Customer Section  Delete Customer | Successful |
| Customer Section  Update Customer Database | Successful |
| Customer Section  List Customer From DataBase | Successful |

|  |  |
| --- | --- |
| INPUT | RESULT |
| Log in – User ID: 333  Password: pewa | Successful |
| Category Section  Add Category | Successful |
| Category Section  List Category | Successful |
| Category Section  Search Category | Successful |
| Category Section  Update Category | Successful |
| Category Section  Delete Category | Successful |
| Publisher Section  Add Publisher | Successful |
| Publisher Section  List Publisher | Successful |
| Publisher Section  Search Publisher | Successful |
| Publisher Section  Update Publisher | Successful |
| Publisher Section  Delete Publisher | Successful |
| Book Section  Add Book | Successful |
| Book Section  List Book | Successful |
| Book Section  Search Book | Successful |
| Book Section  Update Book | Successful |
| Book Section  Delete Book | Successful |

|  |  |
| --- | --- |
| INPUT | RESULT |
| Log in – User ID: 444  Password: mabr | Successful |
| Order Section  Add Order | Successful |
| Order Section  List Order | Successful |
| Order Section  Search Order | Successful |
| Order Section  Update Order | Successful |
| Order Section  Delete Order | Successful |

# Project Deployment

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated

### MIS Manager

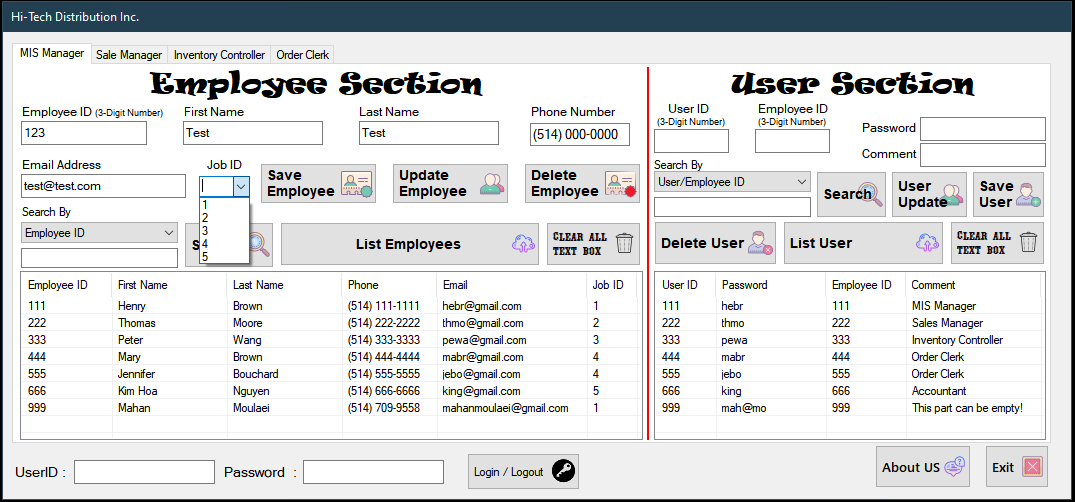
Graphical user interface, text, application

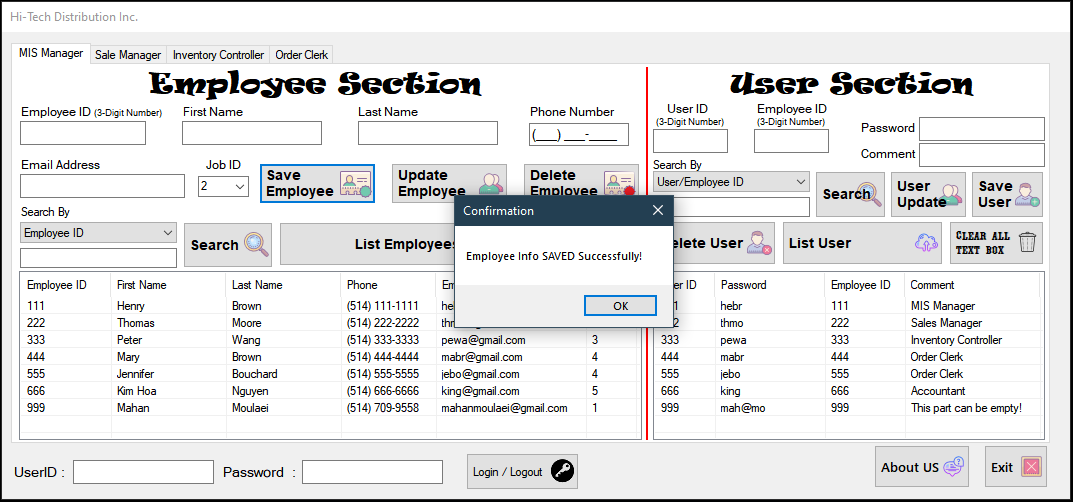
Description automatically generated

A screenshot of a computer

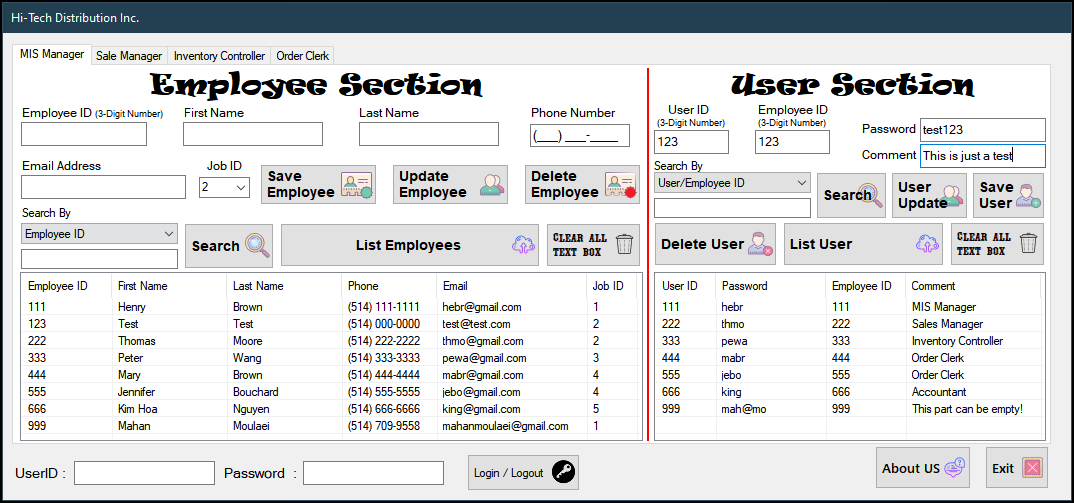
Description automatically generated

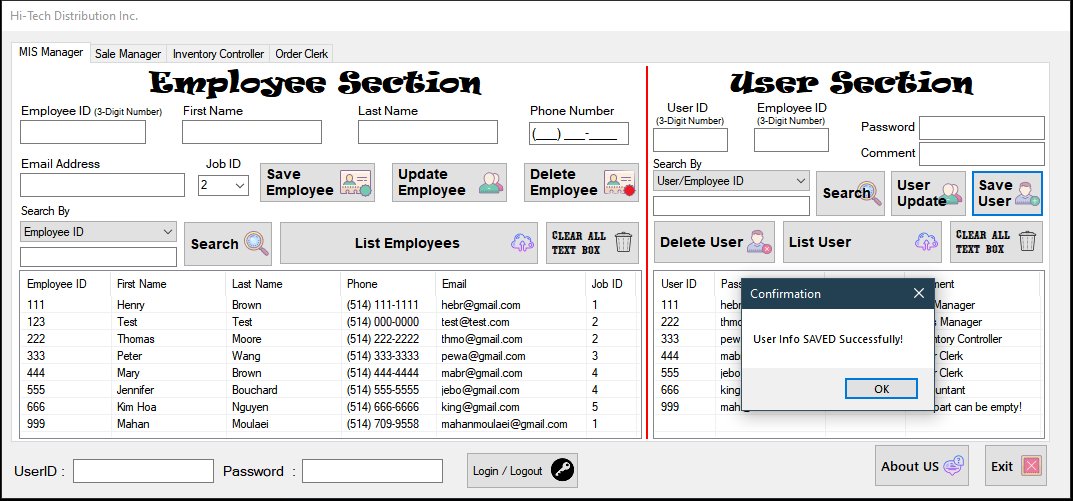
\*Adding Employee\*





\*Adding User\*

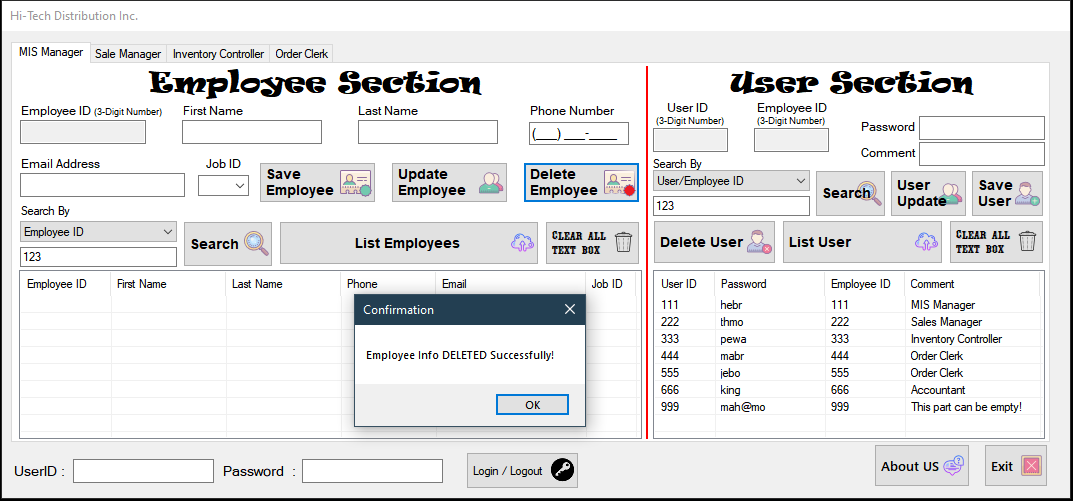




\*Searching and Deleting Employee\*

A screenshot of a computer

Description automatically generated

\*Searching and Deleting Employee After Deleting its User\*

\*All text-boxes of each section will be reset to their default by clicking on “Clear All Text Box” Button\*

### Sales Manager

Graphical user interface, text, application

Description automatically generated

A screenshot of a computer

Description automatically generated

\*Adding Customer\*

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

\*Searching Customer\*Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated\*Updating Credit Limit\*

Graphical user interface, application

Description automatically generated

\*Updating Customer Database \*A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

\*Listing Customer Database\*Graphical user interface, application

Description automatically generated

\*All text-boxes of each section will be reset to their default by clicking on “Clear All Text Box” Button\*

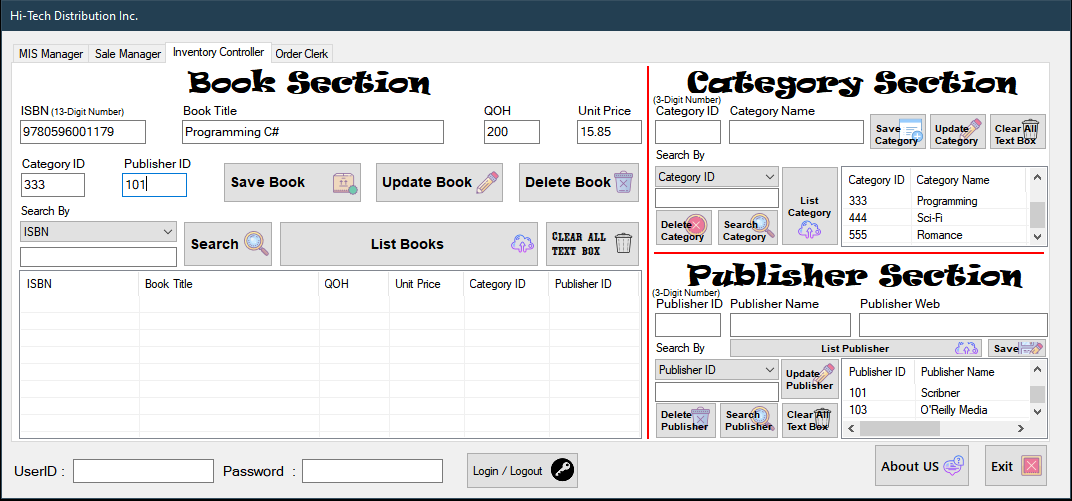
### Inventory Controller

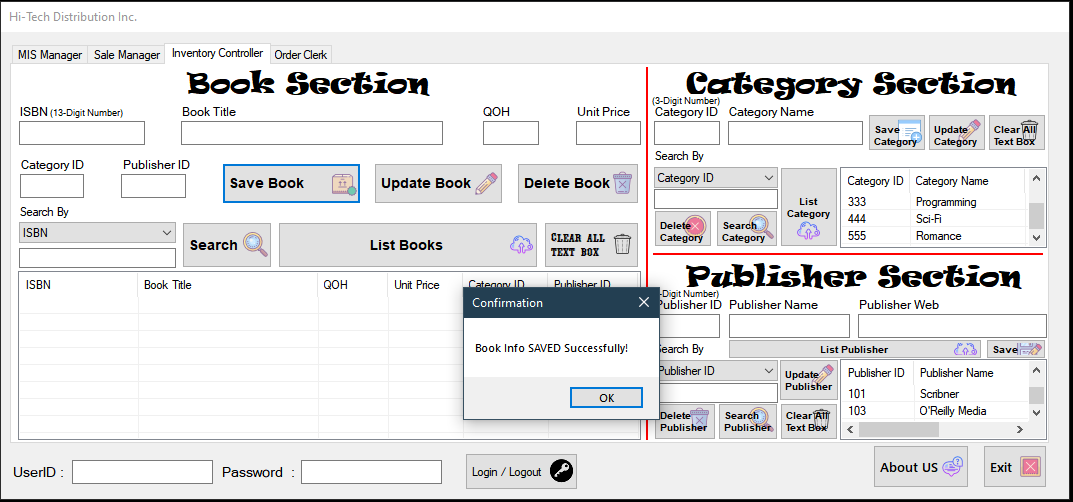
Graphical user interface, text, application

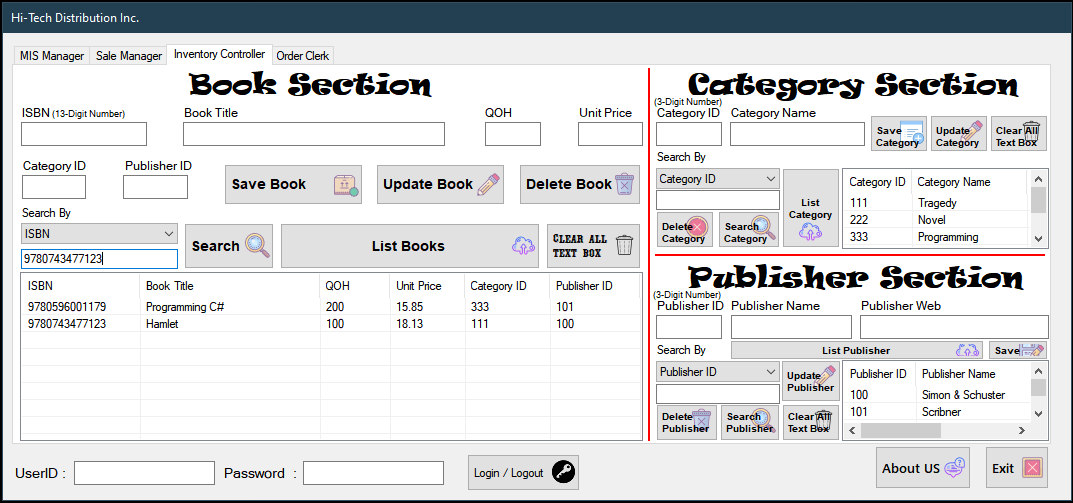
Description automatically generated

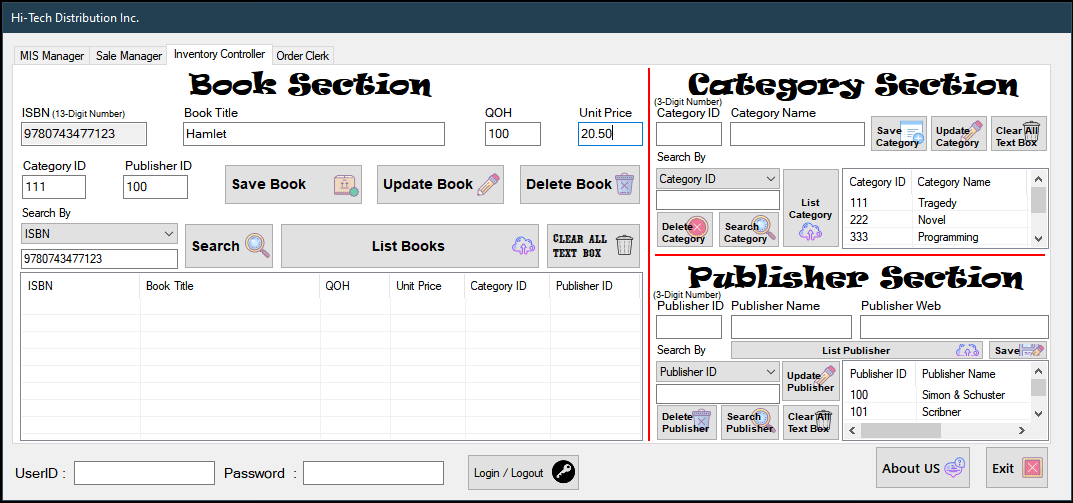
A screenshot of a computer

Description automatically generated

\*Adding Book based on the available Categories and Publishers\*

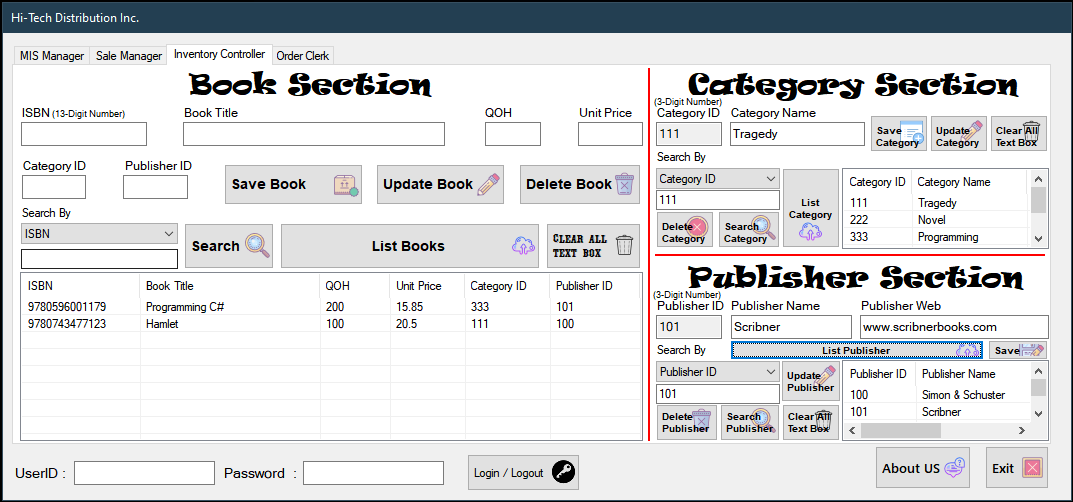


\*Searching Book by ISBN\*

\*Updating Book Price\*

A screenshot of a computer

Description automatically generated

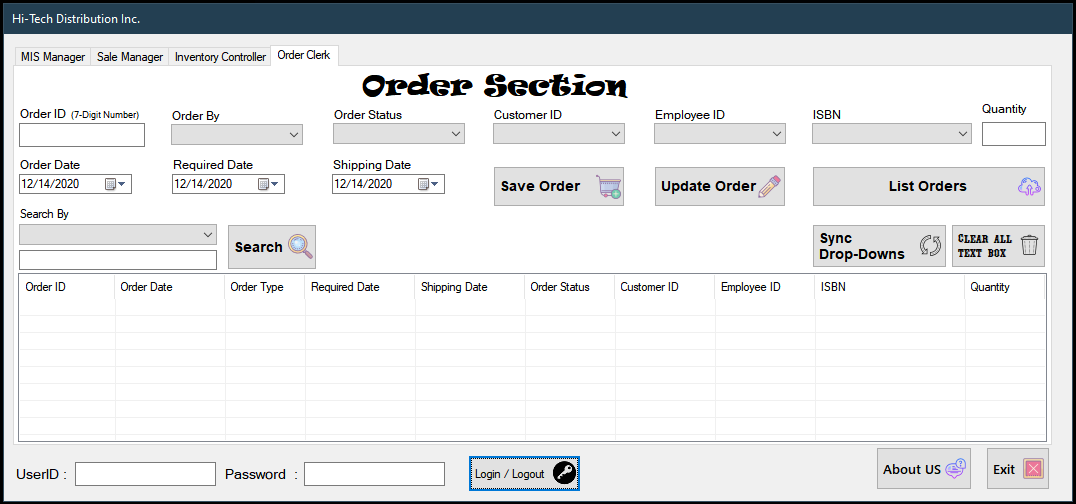
\*Listing Book\*

\*All text-boxes of each section will be reset to their default by clicking on “Clear All Text Box” Button\*

### Order Clerk

Graphical user interface, text, application

Description automatically generated



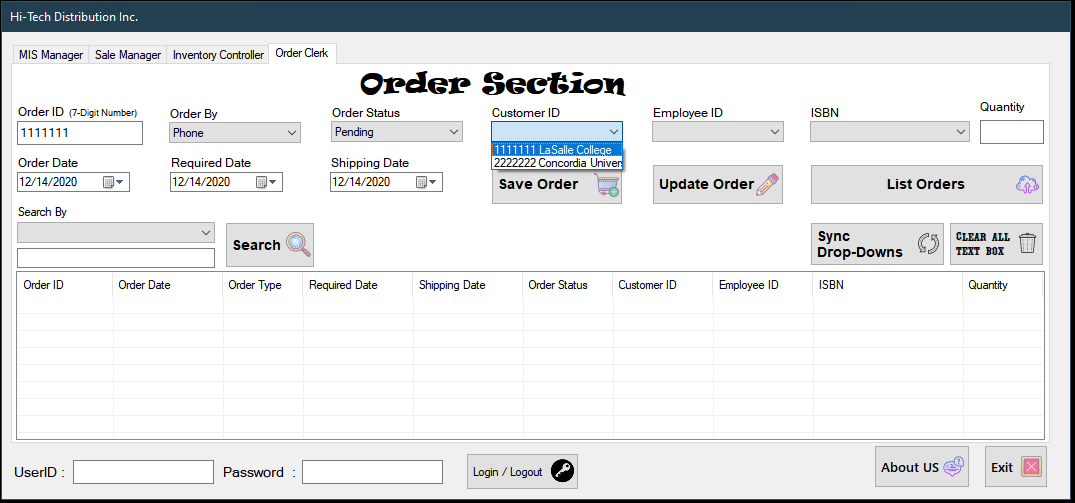
\*Before Adding an Order, the “Sync Drop-Downs” must be clicked to sync the new added Customers, Employees, and Books with the database to the drop downs\*

Graphical user interface, application, table

Description automatically generated\*Adding Order\*

A screenshot of a computer

Description automatically generated



Graphical user interface, application

Description automatically generated

A screenshot of a computer

Description automatically generated

\*Error while entering more quantity than the available quantity of the book in the inventory \*Graphical user interface, application, table

Description automatically generated

A screenshot of a computer

Description automatically generated

\*Searching Order\*A screenshot of a computer

Description automatically generated

\*All text-boxes and drop downs will be reset to their default by clicking on “Clear All Text Box” Button\*

# Conclusion

From this project, I have learnt to create and implement a Database and connect it to the Windows Form Application. By practicing and creating this project, I also learned how to integrate and work with Connected-Mode, Disconnected-Mode and Entity Framework all in one single project. Besides that, I learned set of tasks such as data validation, and how to Save, List, Search, Update, and Delete data from Database in ADO.Net Object Model and .Net Entity Framework.

Mahan Moulaei