Course Number: 420-DA3-AS

Course Title: Multi-tier Applications Development

Session: Winter 2022

**FINAL PROJECT REPORT**

**TITLE: HI-TECH ORDER MANAGEMENT SYSTEM**

**DUE DATE: 28 APRIL, 2022**

**SUBMITTED**

**TO**

**TEACHER: QUANG HOANG CAO**

**BY**

**AAFRIN (2030150) & ANUBHA(2032178)**

1. **PROJECT DESCRIPTION**

A design and implementation of a windows application called **Hi-Tech Ordering Management System** using Visual Studio 2022 and SQL Server 2017, C# which will allow the user to manage the data and perform the operations like **Adding**, **Updating, Deleting**, and **Searching and Viewing** the objects.

For the security purpose, the access is only given to the user upon the insertion of username and password to **authenticate the user** from the database.

1. **PROJECT DEVELOPMENT**

**PHASE I ANALYSIS**

1. Understood the requirements and structure of the project.
2. Users: 5

* MIS Manager

Login the system

Add user Information

Update User/Employee Information

Delete User/Employee

Search Users/Employees

View all the Users/Employees

* Sales Manager

Login the system

Add Customer

Update Customer Information

Delete Customer

Search Customer

View all the Customer

* Inventory Controller

Login the system

Add Book

Update Book Information

Delete Book

Search Book

View all the Books

* Order Clerks

Login the system

Add Order

Update Order Information

Delete Order

Search Order

View all the Orders

1. Technical requirements (Connected Mode, Disconnected Mode, ADO.NET Entity Framework)

**PHASE II DESIGN**

1. Design the Database

Create tables

Create Relationships between the tables

Create a Database Diagram

1. Design the Windows forms (GUI)

User Form.

Employee Form.

Customer Form.

Book Form.

Order Form.

1. Structured Folder.

GUI

DAL

BLL

VALIDATOR

Reference Library

1. Classes Design

User class

Employee class

Customer class

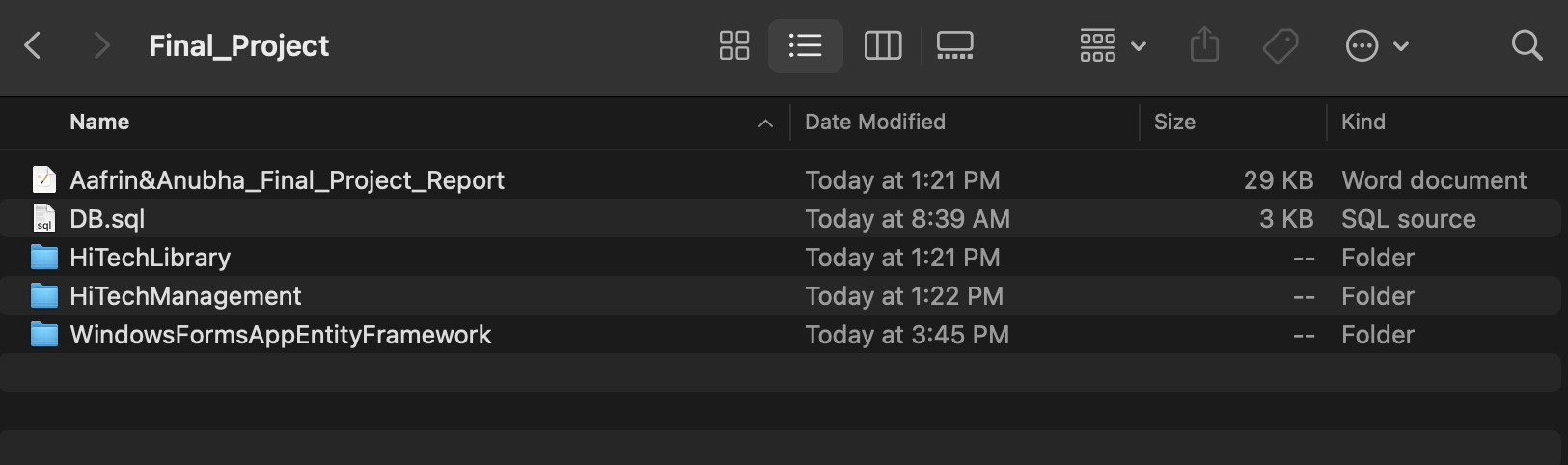
Book class

Order class

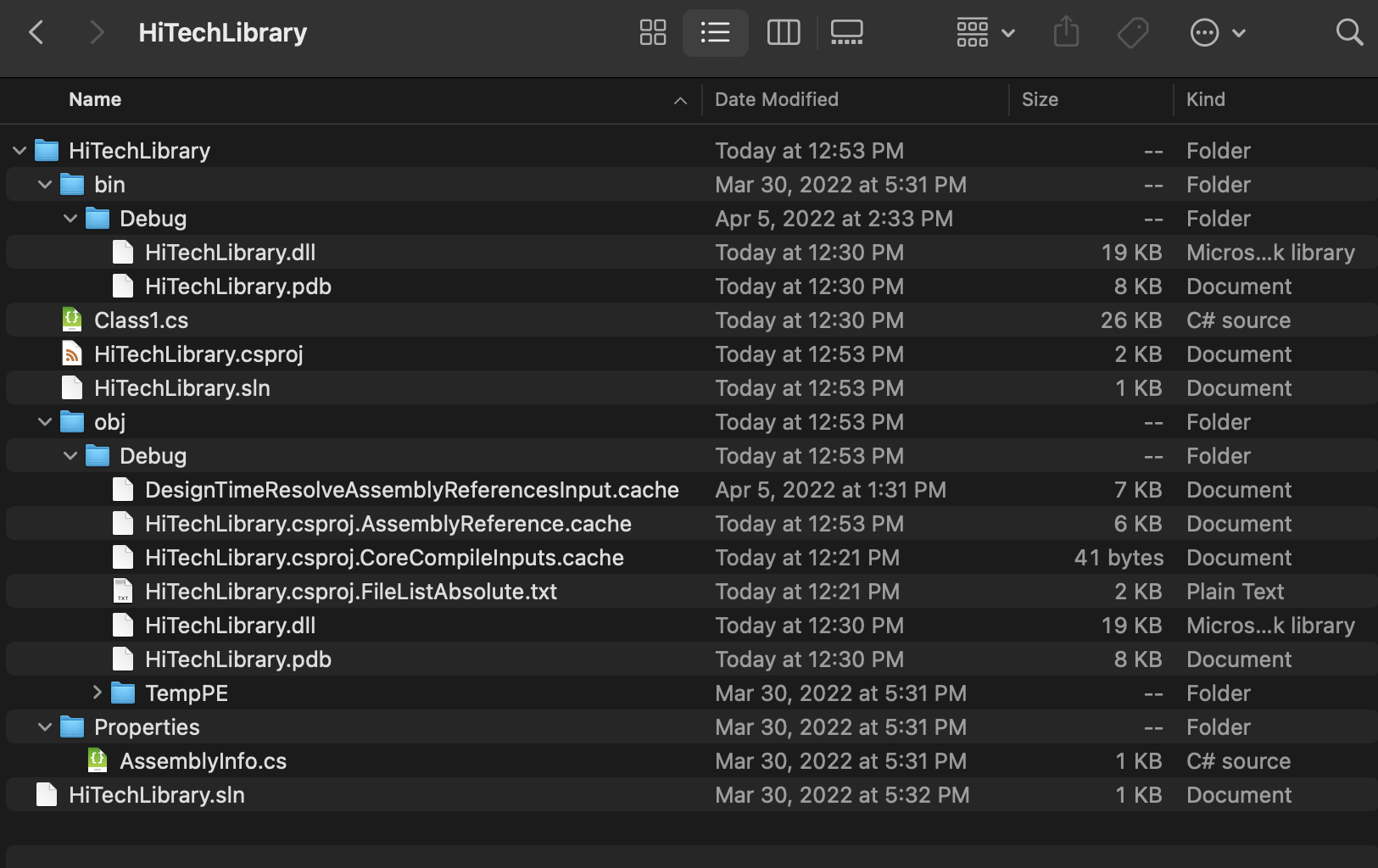
**PHASE III IMPLEMENTATION**

Entire source code for implementing HI Tech Library submitted on Omnivox and sent via Mio

Main Project Folder >



HiTechLibrary Folder >



**CLASS1.cs**

**using system;**

**using system.collections.generic;**

**using system.linq;**

**using system.text;**

**using system.threading.tasks;**

**using system.data.sqlclient;**

**using system.configuration;**

**using system.text.regularexpressions; // use the class regex**

**using microsoft.visualbasic; // information.isnumeric(input)**

**using hitechmanagement.bll;**

**using hitechmanagement.dal;**

**using hitechmanagement.validator;**

**namespace hitechmanagement.bll**

**{**

**internal class employee**

**{**

**private int employeeid;**

**private string firstname;**

**private string lastname;**

**private string phone;**

**private string email;**

**private int jobid;**

**private int statusid;**

**// properties**

**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 phone { get => phone; set => phone = value; }**

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

**public int jobid { get => jobid; set => jobid = value; }**

**public int statusid { get => statusid; set => statusid = value; }**

**//custom methods**

**public void saveemployee(employee emp)**

**{**

**employeedb.saverecord(emp);**

**}**

**public employee searchemployee(int empid)**

**{**

**return employeedb.searchrecord(empid);**

**}**

**public bool idexists(int id)**

**{**

**return employeedb.isduplicateid(id);**

**}**

**public void updateemployee(employee emp)**

**{**

**employeedb.updaterecord(emp);**

**}**

**public void deleteemployee(int eid)**

**{**

**employeedb.deleterecord(eid);**

**}**

**public list<employee> getemployeelist()**

**{**

**return employeedb.getallrecords();**

**}**

**}**

**internal class user**

**{**

**// private variables**

**private int userid;**

**private string password;**

**private int employeeid;**

**private string username;**

**// properties**

**public int userid { get => userid; set => userid = value; }**

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

**public int employeeid { get => employeeid; set => employeeid = value; }**

**public string username { get => username; set => username = value; }**

**// methods**

**public void saveuser(user usr)**

**{**

**userdb.saverecord(usr);**

**}**

**public bool idexists(int id)**

**{**

**return userdb.isduplicateid(id);**

**}**

**public void updateuser(user emp)**

**{**

**userdb.updaterecord(emp);**

**}**

**public void deleteuser(int eid)**

**{**

**userdb.deleterecord(eid);**

**}**

**public user searchuser(int usrid)**

**{**

**return userdb.searchrecord(usrid);**

**}**

**public user validateuserbyname(string username)**

**{**

**return userdb.searchrecordbyname(username);**

**}**

**public list<user> getuserlist()**

**{**

**return userdb.getallrecords();**

**}**

**}**

**internal class customer**

**{**

**public int customerid { get; set; }**

**public string customername { get; set; }**

**public string address { get; set; }**

**public string city { get; set; }**

**public string postalcode { get; set; }**

**public string phone { get; set; }**

**public string fax { get; set; }**

**public int creditlimit { get; set; }**

**public string email { get; set; }**

**public list<customer> getstudentlist()**

**{**

**return customerdb.getallrecords();**

**}**

**}**

**public class book**

**{**

**// private variables**

**private int isbn;**

**private string title;**

**private decimal unitprice;**

**private int yearpublished;**

**private int qoh;**

**private int categoryid;**

**private int publisherid;**

**// properties**

**public int isbn { get => isbn; set => isbn = value; }**

**public string title { get => title; set => title = value; }**

**public decimal unitprice { get => unitprice; set => unitprice = value; }**

**public int yearpublished { get => yearpublished; set => yearpublished = value; }**

**public int qoh { get => qoh; set => qoh = value; }**

**public int categoryid { get => categoryid; set => categoryid = value; }**

**public int publisherid { get => publisherid; set => publisherid = value; }**

**//custom methods**

**public void savebook(book bk)**

**{**

**bookdb.saverecord(bk);**

**}**

**public book searchbook(int ebk)**

**{**

**return bookdb.searchrecord(ebk);**

**}**

**public bool idexists(int id)**

**{**

**return bookdb.isduplicateid(id);**

**}**

**public void updatebook(book bk)**

**{**

**bookdb.updaterecord(bk);**

**}**

**public void deletebook(int eid)**

**{**

**bookdb.deleterecord(eid);**

**}**

**public list<book> getbooklist()**

**{**

**return bookdb.getallrecords();**

**}**

**}**

**}**

**namespace hitechmanagement.dal**

**{**

**internal class bookdb**

**{**

**public static void saverecord(book bk)**

**{**

**// insert into employees values(6555, 'mary', 'green', 'programmer analyst')**

**//1. connect db**

**sqlconnection conn = utilitydb.connectdb();**

**//2. create and customize an sqlcommand object**

**sqlcommand cmdinsert = new sqlcommand();**

**cmdinsert.commandtext = "insert into books (isbn,title,unitprice,year,qoh,categoryid,publisherid)" +**

**" values (@isbn,@title,@unitprice,@year,@qoh,@categoryid,@publisherid)";**

**cmdinsert.parameters.addwithvalue("@isbn", bk.isbn);**

**cmdinsert.parameters.addwithvalue("@title", bk.title);**

**cmdinsert.parameters.addwithvalue("@unitprice", bk.unitprice);**

**cmdinsert.parameters.addwithvalue("@year", bk.yearpublished);**

**cmdinsert.parameters.addwithvalue("@qoh", bk.qoh);**

**cmdinsert.parameters.addwithvalue("@categoryid", bk.qoh);**

**cmdinsert.parameters.addwithvalue("@publisherid", bk.publisherid);**

**cmdinsert.connection = conn;**

**// 3. perform the insert command**

**cmdinsert.executenonquery();**

**//4. close the db**

**conn.close();**

**conn.dispose();**

**}**

**public static bool isduplicateid(int bid)**

**{**

**book bk = searchrecord(bid);**

**if (bk != null)**

**{**

**return false;**

**}**

**return true;**

**}**

**public static book searchrecord(int bid)**

**{**

**book bk = new book();**

**sqlconnection conn = utilitydb.connectdb();**

**sqlcommand cmdsearchbyid = new sqlcommand();**

**cmdsearchbyid.commandtext = "select \* from books where isbn =@isbn";**

**cmdsearchbyid.connection = conn;**

**cmdsearchbyid.parameters.addwithvalue("@isbn", bid);**

**sqldatareader reader = cmdsearchbyid.executereader();**

**if (reader.read())**

**{**

**bk.isbn = convert.toint32(reader["isbn"]);**

**bk.title = reader["title"].tostring();**

**bk.unitprice = convert.todecimal(reader["unitprice"]);**

**bk.yearpublished = convert.toint32(reader["year"]);**

**bk.qoh = convert.toint32(reader["qoh"]);**

**bk.categoryid = convert.toint32(reader["categoryid"]);**

**bk.publisherid = convert.toint32(reader["publisherid"]);**

**}**

**conn.close();**

**conn.dispose();**

**return bk;**

**}**

**public static void updaterecord(book bk)**

**{**

**// insert into employees values(6555, 'mary', 'green', 'programmer analyst')**

**//1. connect db**

**sqlconnection conn = utilitydb.connectdb();**

**//2. create and customize an sqlcommand object**

**sqlcommand cmdupdate = new sqlcommand();**

**cmdupdate.commandtext = "update books " + "set isbn = @isbn, title = @title," +**

**" unitprice = @unitprice,year = @year,qoh = @qoh, categoryid = @categoryid, publisherid = @publisherid where isbn =@isbn";**

**cmdupdate.parameters.addwithvalue("@isbn", bk.isbn);**

**cmdupdate.parameters.addwithvalue("@title", bk.title);**

**cmdupdate.parameters.addwithvalue("@unitprice", bk.unitprice);**

**cmdupdate.parameters.addwithvalue("@year", bk.yearpublished);**

**cmdupdate.parameters.addwithvalue("@qoh", bk.qoh);**

**cmdupdate.parameters.addwithvalue("@categoryid", bk.categoryid);**

**cmdupdate.parameters.addwithvalue("@publisherid", bk.publisherid);**

**cmdupdate.connection = conn;**

**// 3. perform the insert command**

**cmdupdate.executenonquery();**

**//4. close the db**

**conn.close();**

**conn.dispose();**

**}**

**public static void deleterecord(int ebk)**

**{**

**sqlconnection conn = utilitydb.connectdb();**

**sqlcommand cmddelete = new sqlcommand();**

**book bk = new book();**

**cmddelete.commandtext = "delete from books where isbn=@isbn";**

**cmddelete.connection = conn;**

**cmddelete.parameters.addwithvalue("@isbn", ebk);**

**cmddelete.executenonquery();**

**conn.close();**

**conn.dispose();**

**}**

**public static list<book> getallrecords()**

**{**

**list<book> listb = new list<book>();**

**sqlconnection conn = utilitydb.connectdb();**

**sqlcommand cmdselect = new sqlcommand("select \* from books ", conn);**

**// perform theselect quer**

**sqldatareader reader = cmdselect.executereader();**

**book bk;**

**///employee emp = new employee();**

**while (reader.read())**

**{**

**bk = new book();**

**bk.isbn = convert.toint32(reader["isbn"]);**

**bk.title = reader["title"].tostring();**

**bk.unitprice = convert.toint32(reader["unitprice"]);**

**bk.yearpublished = convert.toint32(reader["year"]);**

**bk.qoh = convert.toint32(reader["qoh"]);**

**bk.categoryid = convert.toint32(reader["categoryid"]);**

**bk.publisherid = convert.toint32(reader["publisherid"]);**

**listb.add(bk);**

**}**

**conn.close();**

**conn.dispose();**

**return listb;**

**}**

**}**

**internal class customerdb**

**{**

**public static list<customer> getallrecords()**

**{**

**list<customer> listc = new list<customer>();**

**customer acustomer;**

**using (sqlconnection conn = utilitydb.connectdb())**

**{**

**sqlcommand cmdselectall = new sqlcommand("select \* from customers", conn);**

**sqldatareader reader = cmdselectall.executereader();**

**while (reader.read())**

**{**

**acustomer = new customer();**

**acustomer.customerid = convert.toint32(reader["customerid"]);**

**acustomer.customername = reader["customername"].tostring();**

**acustomer.address = reader["address"].tostring();**

**acustomer.city = reader["city"].tostring();**

**acustomer.postalcode = reader["postalcode"].tostring();**

**acustomer.phone = reader["phone"].tostring();**

**acustomer.fax = reader["fax"].tostring();**

**acustomer.creditlimit = convert.toint32(reader["creditlimit"].tostring());**

**acustomer.email = reader["email"].tostring();**

**listc.add(acustomer);**

**}**

**}**

**return listc;**

**}**

**}**

**internal class employeedb**

**{**

**public static void saverecord(employee emp)**

**{**

**// insert into employees values(6555, 'mary', 'green', 'programmer analyst')**

**//1. connect db**

**sqlconnection conn = utilitydb.connectdb();**

**//2. create and customize an sqlcommand object**

**sqlcommand cmdinsert = new sqlcommand();**

**cmdinsert.commandtext = "insert into employees (employeeid,firstname,lastname,phone,email,jobid,statusid)" +**

**" values (@employeeid,@firstname,@lastname,@phone,@email,@jobid,@statusid)";**

**cmdinsert.parameters.addwithvalue("@employeeid", emp.employeeid);**

**cmdinsert.parameters.addwithvalue("@firstname", emp.firstname);**

**cmdinsert.parameters.addwithvalue("@lastname", emp.lastname);**

**cmdinsert.parameters.addwithvalue("@phone", emp.phone);**

**cmdinsert.parameters.addwithvalue("@email", emp.email);**

**cmdinsert.parameters.addwithvalue("@jobid", emp.jobid);**

**cmdinsert.parameters.addwithvalue("@statusid", emp.statusid);**

**cmdinsert.connection = conn;**

**// 3. perform the insert command**

**cmdinsert.executenonquery();**

**//4. close the db**

**conn.close();**

**conn.dispose();**

**}**

**public static bool isduplicateid(int eid)**

**{**

**employee emp = searchrecord(eid);**

**if (emp != null)**

**{**

**return false;**

**}**

**return true;**

**}**

**public static employee searchrecord(int eid)**

**{**

**employee emp = new employee();**

**sqlconnection conn = utilitydb.connectdb();**

**sqlcommand cmdsearchbyid = new sqlcommand();**

**cmdsearchbyid.commandtext = "select \* from employees where employeeid =@employeeid";**

**cmdsearchbyid.connection = conn;**

**cmdsearchbyid.parameters.addwithvalue("@employeeid", eid);**

**sqldatareader reader = cmdsearchbyid.executereader();**

**if (reader.read())**

**{**

**emp.employeeid = convert.toint32(reader["employeeid"]);**

**emp.firstname = reader["firstname"].tostring();**

**emp.lastname = reader["lastname"].tostring();**

**emp.phone = reader["phone"].tostring();**

**emp.email = reader["email"].tostring();**

**emp.jobid = convert.toint32(reader["jobid"]);**

**emp.statusid = convert.toint32(reader["statusid"]);**

**}**

**conn.close();**

**conn.dispose();**

**return emp;**

**}**

**public static void updaterecord(employee emp)**

**{**

**// insert into employees values(6555, 'mary', 'green', 'programmer analyst')**

**//1. connect db**

**sqlconnection conn = utilitydb.connectdb();**

**//2. create and customize an sqlcommand object**

**sqlcommand cmdupdate = new sqlcommand();**

**cmdupdate.commandtext = "update employees " + "set employeeid = @employeeid, firstname = @firstname," +**

**" lastname = @lastname,phone = @phone,jobid = @jobid,statusid = @statusid where employeeid =@employeeid";**

**cmdupdate.parameters.addwithvalue("@employeeid", emp.employeeid);**

**cmdupdate.parameters.addwithvalue("@firstname", emp.firstname);**

**cmdupdate.parameters.addwithvalue("@lastname", emp.lastname);**

**cmdupdate.parameters.addwithvalue("@phone", emp.phone);**

**cmdupdate.parameters.addwithvalue("@jobid", emp.jobid);**

**cmdupdate.parameters.addwithvalue("@statusid", emp.statusid);**

**cmdupdate.connection = conn;**

**// 3. perform the insert command**

**cmdupdate.executenonquery();**

**//4. close the db**

**conn.close();**

**conn.dispose();**

**}**

**public static list<employee> getallrecords()**

**{**

**list<employee> liste = new list<employee>();**

**sqlconnection conn = utilitydb.connectdb();**

**sqlcommand cmdselect = new sqlcommand("select \* from employees ", conn);**

**// perform theselect quer**

**sqldatareader reader = cmdselect.executereader();**

**employee emp;**

**///employee emp = new employee();**

**while (reader.read())**

**{**

**emp = new employee();**

**emp.employeeid = convert.toint32(reader["employeeid"]);**

**emp.firstname = reader["firstname"].tostring();**

**emp.lastname = reader["lastname"].tostring();**

**emp.phone = reader["phone"].tostring();**

**emp.email = reader["email"].tostring();**

**emp.jobid = convert.toint32(reader["jobid"]);**

**emp.statusid = convert.toint32(reader["statusid"]);**

**liste.add(emp);**

**}**

**conn.close();**

**conn.dispose();**

**return liste;**

**}**

**public static void deleterecord(int empid)**

**{**

**sqlconnection conn = utilitydb.connectdb();**

**sqlcommand cmddelete = new sqlcommand();**

**employee emp = new employee();**

**cmddelete.commandtext = "delete from employees where employeeid=@employeeid";**

**cmddelete.connection = conn;**

**cmddelete.parameters.addwithvalue("@employeeid", empid);**

**cmddelete.executenonquery();**

**conn.close();**

**conn.dispose();**

**}**

**}**

**internal class userdb**

**{**

**public static void saverecord(user usr)**

**{**

**// insert data into user table**

**//1. connect db**

**sqlconnection conn = utilitydb.connectdb();**

**//2. create and customize an sqlcommand object**

**sqlcommand cmdinsert = new sqlcommand();**

**cmdinsert.commandtext = "insert into users (userid,password,employeeid)" +**

**" values (@userid,@password,@employeeid)";**

**cmdinsert.parameters.addwithvalue("@userid", usr.userid);**

**cmdinsert.parameters.addwithvalue("@password", usr.password);**

**cmdinsert.parameters.addwithvalue("@employeeid", usr.employeeid);**

**cmdinsert.connection = conn;**

**// 3. perform the insert command**

**cmdinsert.executenonquery();**

**//4. close the db**

**conn.close();**

**conn.dispose();**

**}**

**public static bool isduplicateid(int uid)**

**{**

**user usr = searchrecord(uid);**

**if (usr != null)**

**{**

**return false;**

**}**

**return true;**

**}**

**public static bool validusername(string username)**

**{**

**user usr = searchrecordbyname(username);**

**if (usr != null)**

**{**

**return false;**

**}**

**return true;**

**}**

**public static user searchrecord(int uid)**

**{**

**user usr = new user();**

**sqlconnection conn = utilitydb.connectdb();**

**sqlcommand cmdsearchbyid = new sqlcommand();**

**cmdsearchbyid.commandtext = "select \* from users where userid =@userid";**

**cmdsearchbyid.connection = conn;**

**cmdsearchbyid.parameters.addwithvalue("@userid", uid);**

**sqldatareader reader = cmdsearchbyid.executereader();**

**if (reader.read())**

**{**

**usr.userid = convert.toint32(reader["userid"]);**

**usr.employeeid = convert.toint32(reader["employeeid"]);**

**usr.password = reader["password"].tostring();**

**}**

**conn.close();**

**conn.dispose();**

**return usr;**

**}**

**public static user searchrecordbyname(string username)**

**{**

**user usr = new user();**

**sqlconnection conn = utilitydb.connectdb();**

**sqlcommand cmdsearchbyid = new sqlcommand();**

**cmdsearchbyid.commandtext = "select \* from users join employees on users.employeeid = employees.employeeid where firstname = @firstname;";**

**cmdsearchbyid.connection = conn;**

**cmdsearchbyid.parameters.addwithvalue("@firstname", username);**

**sqldatareader reader = cmdsearchbyid.executereader();**

**if (reader.read())**

**{**

**usr.username = reader["firstname"].tostring();**

**usr.password = reader["password"].tostring();**

**}**

**conn.close();**

**conn.dispose();**

**return usr;**

**}**

**public static void updaterecord(user usr)**

**{**

**// insert into users values(6555, 'mary', 'green', 'programmer analyst')**

**//1. connect db**

**sqlconnection conn = utilitydb.connectdb();**

**//2. create and customize an sqlcommand object**

**sqlcommand cmdupdate = new sqlcommand();**

**cmdupdate.commandtext = "update users " + "set userid = @userid, password = @password," +**

**" employeeid = @employeeid where userid =@userid";**

**cmdupdate.parameters.addwithvalue("@userid", usr.userid);**

**cmdupdate.parameters.addwithvalue("@password", usr.password);**

**cmdupdate.parameters.addwithvalue("@employeeid", usr.employeeid);**

**cmdupdate.connection = conn;**

**// 3. perform the insert command**

**cmdupdate.executenonquery();**

**//4. close the db**

**conn.close();**

**conn.dispose();**

**}**

**public static void deleterecord(int usrid)**

**{**

**sqlconnection conn = utilitydb.connectdb();**

**sqlcommand cmddelete = new sqlcommand();**

**user usr = new user();**

**cmddelete.commandtext = "delete from users where userid=@userid";**

**cmddelete.connection = conn;**

**cmddelete.parameters.addwithvalue("@userid", usrid);**

**cmddelete.executenonquery();**

**conn.close();**

**conn.dispose();**

**}**

**public static list<user> getallrecords()**

**{**

**list<user> liste = new list<user>();**

**sqlconnection conn = utilitydb.connectdb();**

**sqlcommand cmdselect = new sqlcommand("select \* from users ", conn);**

**// perform theselect quer**

**sqldatareader reader = cmdselect.executereader();**

**user usr;**

**///employee emp = new employee();**

**while (reader.read())**

**{**

**usr = new user();**

**usr.userid = convert.toint32(reader["userid"]);**

**usr.password = reader["password"].tostring();**

**usr.employeeid = convert.toint32(reader["employeeid"]);**

**liste.add(usr);**

**}**

**conn.close();**

**conn.dispose();**

**return liste;**

**}**

**}**

**internal class utilitydb**

**{**

**public static sqlconnection connectdb()**

**{**

**/\* sqlconnection conn = new sqlconnection();**

**conn.connectionstring = "server=desktop-mdnt271\\sqlserverexpress;database=employeedb;user=sa;password=123456";**

**conn.open();**

**return conn;**

**\*/**

**sqlconnection conn = new sqlconnection();**

**conn.connectionstring = configurationmanager.connectionstrings["connectdb"].connectionstring;**

**conn.open();**

**return conn;**

**}**

**}**

**}**

**namespace hitechmanagement.validator**

**{**

**internal class validator**

**{**

**// user id: 4-digit number**

**public static bool isvalidid(string input)**

**{**

**if (!(regex.ismatch(input, @"^\d{4}$")))**

**{**

**return false;**

**}**

**return true;**

**}**

**// valid user name**

**public static bool isvalidname(string input)**

**{**

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

**{**

**if ((!char.isletter(input[i])) && (!char.iswhitespace(input[i])))**

**{**

**return false;**

**}**

**}**

**return true;**

**}**

**// student id : 7-digit number**

**public static bool isvalid(string input)**

**{**

**if (input.length != 4)**

**{**

**return false;**

**}**

**else // 7 : 1234abc/ abc1234/**

**{**

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

**{**

**if (!(char.isdigit(input[i])))**

**{**

**return false;**

**}**

**}**

**}**

**return true;**

**}**

**public static bool isvalidid(string input)**

**{**

**if (input.length != 7)**

**{**

**return false;**

**}**

**else // 7 : 1234abc/ abc1234/**

**{**

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

**{**

**if (!(char.isdigit(input[i])))**

**{**

**return false;**

**}**

**}**

**}**

**return true;**

**}**

**}**

**}**

**PHASE IV TESTING THE PROGRAM**

| **User** | **Functional Requirement** | **Test Result/Problem(s)** |
| --- | --- | --- |
| Henry Brown | Login, Add/Update/Delete/Search/List the User/Employee information. | Successful |
| Thomas Moore | Login, Add/Update/Delete/Search the Customer information. | Successful |
| Peter Wang | Login, Add/Update/Delete/Search/List the Book information. | Successful |
| Mary Brown | Login, Add/Update/Delete/Search/List the Order information. | Successful |
| Jennifer Bouchard | Login, Add/Update/Delete/Search/List the Order information. | Successful |

1. **CONCLUSION**

Well Understood the Object-Oriented Concepts like Classes and Objects and how to use them.

Learned the connection of the Database and manipulation of the data with Database.

Learned the good practice of creating Folders.

Well Understood the 3 Layered Development i.e., Business Layer, Data Access Layer, and GUI.

Validations of the Input fields before inserting into the database.

1. **BIBLIOGRAPHY**

Class Work posted on Lea.

Stack Overflow.