- CODING -

Coding Contents

- Coding for Common Classes and Methods 02
- Main Form (frmMain) 16
- Login Form (frmLogin) 20
- Notifications (frmNotif) 22
- Add Books Form (frmAddBook) 26
- Book Details Form (frmBookDetails) 32
- Remove Books Form (frmRemBook) 38
- Add Member Form (frmMemNew) 42
- Member Details / Update / Remove / Add-Remove from Blacklist Form (frmMemDetails) 46
- Lend Books Form (frmIssues) 57
- Receive / Extend / Report Loss Form (frmReturnExt) 61
- Check In Check Out Form (frmCheckIO) 68
- BookFinder Form (frmBSearch) 72
- Transactions Form (frmCash) 77
- Member Logbook Form (frmViewCheckIO) 80
- Statistics Form (frmExtra) 83
- Preferences Form (frmSettings) 90
- End 96

Coding for Common Classes and Methods

```
namespace Alexandria
{
    public class db
    {
        public static OleDbConnection con = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=|DataDirectory|\LibraryDB.mdb; Jet OLEDB:Database Password=masterpwd");
    }

    public class cvar // Common Variables
    {
        public static bool InsertID = false;
        public static string MemID;
    }

    public partial class set // Setting variables and settings recall method
    {

        public static string[] SType = new string[] { "A", "C" };
        public static string[] SNameI = new string[] { "Btime", "NoBooks", "Expire",
"Alert", "LostBookTimes"};
        public static string[] SNameD = new string[] { "Fine", "RenewC", "NewC" };

        //A 2 dimensional array is created to store values of SettingType and
SettingName
        public static int[,] SValueI = new int[,] { { 0, 0 }, { 0, 0 }, { 0, 0 }, { 0,
0 }, {0, 0} };
        public static double[,] SValueD = new double[,] { { 0, 0 }, { 0, 0 }, { 0, 0 }
};

        public static int AgeMin = 0;

        public static double[] Fine = new double[2] { 0, 0 };
        public static double[] RenewC = new double[2] { 0, 0 };
        public static double[] NewC = new double[2] { 0, 0 };

        public static int[] Btime = new int[2] { 0, 0 };
        public static int[] NoBooks = new int[2] { 0, 0 };
        public static int[] Expire = new int[2] { 0, 0 };
        public static int[] Alert = new int[2] { 0, 0 };
        public static int[] LostBookTimes = new int[2] { 0, 0 };

        public void recall() // Recall Setting Values from DB and load them to global
variables.
        {
            if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();

            // Main For loop of Int to repeatedly retrive interger values from DB
(using SNameI) and add values to SValue[*,_] 15 times.
                for (int iSN = 0; iSN < SNameI.Length; iSN++)
            {
```

```
                // Sub for loop to repeatedly add values from DB (using SType) and add
to SValue[_,*] twice during each main loop.
                for (int iST = 0; iST < SType.Length; iST++)
                {
                    string sqlSet = string.Format("SELECT SValue FROM Setting WHERE
SName = '{0}' AND SType = '{1}'", SNameI[iSN], SType[iST]);
                    OleDbCommand cmdSet = new OleDbCommand(sqlSet, db.con);
OleDbDataReader drSet = cmdSet.ExecuteReader();

                    if (drSet.Read()) { SValueI[iSN, iST] =
int.Parse(drSet["SValue"].ToString()); }
                }

            }

            // Main For loop of double to repeatedly retrive double values from DB
(using SNameD) and add values to SValue[*,_] 15 times.
            for (int iSN = 0; iSN < SNameD.Length; iSN++)
            {
                // Sub for loop to repeatedly add values from DB (using SType) and add
to SValue[_,*] twice during each main loop.
                for (int iST = 0; iST < SType.Length; iST++)
                {
                    string sqlSet = string.Format("SELECT SValue FROM Setting WHERE
SName = '{0}' AND SType = '{1}'", SNameD[iSN], SType[iST]);
                    OleDbCommand cmdSet = new OleDbCommand(sqlSet, db.con);
OleDbDataReader drSet = cmdSet.ExecuteReader();

                    if (drSet.Read()) { SValueD[iSN, iST] =
double.Parse(drSet["SValue"].ToString()); }
                }

            }

            string sqlSetAge = "SELECT SValue FROM Setting WHERE SName = 'AgeMin'";
            OleDbCommand cmdSetAge = new OleDbCommand(sqlSetAge, db.con);
OleDbDataReader drSetAge = cmdSetAge.ExecuteReader();


            //Add Minimun Age information.
            if (drSetAge.Read()) { AgeMin = int.Parse(drSetAge["SValue"].ToString()); }

            //Add Values to User-friendly variables.

            Fine[0] = SValueD[0, 0]; Fine[1] = SValueD[0, 1];
            RenewC[0] = SValueD[1, 0]; RenewC[1] = SValueD[1, 1];
            NewC[0] = SValueD[2, 0]; NewC[1] = SValueD[2, 1];

            Btime[0] = SValueI[0, 0]; Btime[1] = SValueI[0, 1];
            NoBooks[0] = SValueI[1, 0]; NoBooks[1] = SValueI[1, 1];
            Expire[0] = SValueI[2, 0]; Expire[1] = SValueI[2, 1];
            Alert[0] = SValueI[3, 0]; Alert[1] = SValueI[3, 1];
            LostBookTimes[0] = SValueI[4, 0]; LostBookTimes[1] = SValueI[4, 1];

        }
```

```
    }

    public partial class Popul // Update Popularity Points.
    {

        //Calculate STar Points for each member and input each of them to DB
        public void MemCalcAll()
        {
            // Creat command to get MemberID one by one)

            string sqlMem = string.Format("SELECT MemberID FROM Member");
            OleDbCommand cmdMem = new OleDbCommand(sqlMem, db.con);
            if(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drMem = cmdMem.ExecuteReader();

            //For Each member, Call MemCalc method and calculate and add StarPoints
            while (drMem.Read())
            {
                MemCalc(drMem["MemberID"].ToString(), true);

            } // To next member


        if (db.con.State.Equals(ConnectionState.Closed)) db.con.Close();


        }

        //Calculate StarPoints of A Member with Given ID.
        public double MemCalc(string MemID1, bool updateDB)
        {

            //Declarations
            double StarPoints, Stars =0;

            string sqlMem = string.Format("SELECT TimesBorrow, Renewed, DateJoined FROM
Member WHERE MemberID = '{0}'", MemID1);
            OleDbCommand cmdMem = new OleDbCommand(sqlMem, db.con);
            if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drMem = cmdMem.ExecuteReader();


            if (drMem.Read())
            {

                int TimesBorrow = int.Parse(drMem["TimesBorrow"].ToString()), Renewed =
int.Parse(drMem["Renewed"].ToString()), CheckIn = 0, Months;
                DateTime DateJ = DateTime.Parse(drMem["DateJoined"].ToString());

                //Count Times Checked in from DB
                string sqlCheckIn = string.Format("SELECT COUNT(CDate) AS CntDate FROM
CheckInOut WHERE MemberID = '{0}' AND Event = 'In' GROUP BY MemberID, Event", MemID1);
                OleDbCommand cmdCheckIn = new OleDbCommand(sqlCheckIn, db.con);
                OleDbDataReader drCheckIn = cmdCheckIn.ExecuteReader();
```

```
                if (drCheckIn.Read()) CheckIn =
int.Parse(drCheckIn["CntDate"].ToString());

                // Find no.of months passed since the member first joined.

                int TotDays =
int.Parse(DateTime.Today.Subtract(DateJ).TotalDays.ToString()) + 1;
                Months = (Renewed * 365 + TotDays ) / 30;

                // Star points is sum of Times borrowd and Checkin/4 (4 checkins have
equal points for one borrowal)
                StarPoints = (TimesBorrow + CheckIn / 4);

                //Stars is Popularity Points per Months till today.
                Stars = StarPoints / (Months + 1);


                // Add Stars to each Member
                if (updateDB)
                {
                    string sqlPop = string.Format("UPDATE Member SET Stars = {0} WHERE
MemberID = '{1}'", Stars, MemID1);
                    OleDbCommand cmdPop = new OleDbCommand(sqlPop, db.con);
cmdPop.ExecuteNonQuery();
                }
            }
            else MessageBox.Show("Member ID not read");

            return Stars;
        }

        //Calculate Popularity for a given title
        public double BookCalc(string BookID, string TitleID, bool updateDB)
        {
            // Declarations
            int Times, NoBooks; double Pop;


            if (TitleID == null) // If titleID not given, get it
            {
                //Get TitleID from Book table
                string sqlBook = string.Format("SELECT TitleID FROM Book WHERE BookID =
'{0}'", BookID);
                OleDbCommand cmdBook = new OleDbCommand(sqlBook, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drBook = cmdBook.ExecuteReader(); drBook.Read();
                TitleID = drBook["TitleID"].ToString();
            }

            //Get TimesTaken from Title
            string sqlTimes = string.Format("SELECT TimesBorrowed FROM Title WHERE
TitleID = '{0}'", TitleID);
            OleDbCommand cmdTimes = new OleDbCommand(sqlTimes, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drTimes = cmdTimes.ExecuteReader(); drTimes.Read();
```

```
            Times = int.Parse(drTimes["TimesBorrowed"].ToString());

            //Get no. of books from Book
            string sqlNoBook= string.Format("SELECT COUNT(BookID) as NoBook FROM Book
WHERE TitleID = '{0}'", TitleID);
            OleDbCommand cmdNoBook= new OleDbCommand(sqlNoBook, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drNoBook= cmdNoBook.ExecuteReader(); drNoBook.Read();
            NoBooks = int.Parse(drNoBook["NoBook"].ToString());

            //Calculate Popularity = Time taken per book
            Pop = double.Parse(Math.Round(decimal.Parse((Times / NoBooks).ToString()),
2).ToString());

            if (updateDB)
            {
                string sqlPop = string.Format("UPDATE Title SET Popularity = {0} WHERE
TitleID = '{1}'", Pop, TitleID);
                OleDbCommand cmdPop = new OleDbCommand(sqlPop, db.con);
cmdPop.ExecuteNonQuery();
            }

            return Pop;
        }

        public void BookCalcAll()
        {
            // Creat command to get TitleID one by one)

            string sqlBookD = string.Format("SELECT TitleID FROM Title");
            OleDbCommand cmdBookD = new OleDbCommand(sqlBookD, db.con);
            if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drBookD = cmdBookD.ExecuteReader();

            //For Each BookDber, Call BookDCalc method and calculate and add StarPoints
            while (drBookD.Read())
            {
                BookCalc(null, drBookD["TitleID"].ToString(), true);

            } // To next BookDber
        }

    }

    public partial class Methods // Common Methods
    {
        // Create New ID from MaxID method
        public string NewID(string MaxID)
        {

            //Getting Initial Characted from MAx ID
            string a = MaxID.Substring(0, 1);

            //Incrementing MaxID into newID
            int iID = int.Parse(MaxID.Substring(1, 4));
```

```
            string sID = (iID + 1).ToString();

            // Using a for loop to create newID with legnth 5 (1 + 4)
            for (int i = sID.Length; i != 4; i++) { sID = "0" + sID; }
            string NewID = a + sID;

            return NewID;
        }

        // Check for Alert Notifications [It works PERFECTLY, in few tries.. WOW]
        public void Notif(ref int N, ref DateTime[] nLend, ref string[] nMemID, ref
string[] nMemType, ref string[] nBookID, ref string[] nTitleID, ref string[] nTitle,
ref double[] nPrice, ref bool NError)
        {

            //Disclaimer for Settings not loaded.
            if (set.Btime[0] == 0 && set.Btime[1] == 0) { MessageBox.Show("Settings not
loaded. Load settings and try again."); NError = true; goto End; }

            // Lists are declared as the values cannot be added to arrays since we dont
know the array legnth aka no. of notif.

            List<DateTime> lLend = new List<DateTime>();
            List<string> lMemID = new List<string>();
            List<string> lMemType = new List<string>();
            List<string> lBookID = new List<string>();
            List<string> lTitleID = new List<string>();
            List<string> lTitle = new List<string>();
            List<double> lPrice = new List<double>();


            // Read the LendStatus Relation to get all data
            string sqlLend = "SELECT BookID, MemberID, LendDate FROM LendStatus";
            OleDbCommand cmdLend = new OleDbCommand(sqlLend, db.con);
            if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drLend = cmdLend.ExecuteReader();

            N = 0; // No notification at this point

            while (drLend.Read()) // For each record in LendStatus
            {
                //Declare temporary VARIABLES for each record
                string tBookID, tMemID, tMemType, tTitleID, tTitle; double tPrice;
DateTime tLend; int i;

                //Lend Date and MemberID
                tLend = DateTime.Parse(drLend["LendDate"].ToString()); tMemID =
drLend["MemberID"].ToString();

                //Check MemType of Member since the Alert time differs acc. to MemType
                string sqlMType = string.Format("SELECT MType FROM Member WHERE
MemberID = '{0}'", tMemID);
                OleDbCommand cmdMType = new OleDbCommand(sqlMType, db.con);
OleDbDataReader drMType = cmdMType.ExecuteReader();
```

```
                if (drMType.Read()) tMemType = drMType["MType"].ToString(); else
tMemType = "ERROR";

                //Get respective Alert time and calculate ideal time aka time when the
alert should be raised.
                if (tMemType == "Adult") i = 0; else i = 1;
                DateTime idealDT = tLend.AddDays(set.Alert[i]);

                if (DateTime.Today.Date > idealDT) // If Alert should have been raised
in the past (before today), Get details and count it.
                {
                    N = N + 1; // Take this record into count.
                    tBookID = drLend["BookID"].ToString(); // Get bookID

                    //From BookID in Book relation, get TitleID & Load it to temp var.
                    string sqlBookID = string.Format("SELECT TitleID FROM Book WHERE
BookID = '{0}'", tBookID);
                    OleDbCommand cmdBookID = new OleDbCommand(sqlBookID, db.con);
OleDbDataReader drBookID = cmdBookID.ExecuteReader();
                    if (drBookID.Read()) tTitleID = drBookID["TiTleID"].ToString();
else tTitleID = "ERROR";

                    //From TitleID in Title relation, get Title's information
                    string sqlTitle = string.Format("SELECT BTitle, Price, BTitle FROM
Title WHERE TitleID = '{0}'", tTitleID);
                    OleDbCommand cmdTitle = new OleDbCommand(sqlTitle, db.con);
OleDbDataReader drTitle = cmdTitle.ExecuteReader();

                    //Load them to temp. variables
                    if (drTitle.Read()) { tTitle = drTitle["BTitle"].ToString(); tPrice
= double.Parse(drTitle["Price"].ToString()); }
                    else { tTitle = "ERROR"; tPrice = 404; }

                    //Add all temp. variables to the lists.
                    lLend.Add(tLend); lMemID.Add(tMemID); lMemType.Add(tMemType);
lBookID.Add(tBookID); lTitleID.Add(tTitleID); lTitle.Add(tTitle); lPrice.Add(tPrice);

                }
            } // To next record in LendStatus

            //All records and their values have been inserted into 7 lists and no. of
Notif loaded to variable 'N'

            if (N == 0) goto End; // If No alerts, end the method with empty arrays

            //If alerts, change all lists to respective arrays and out them via ref
            nLend = lLend.ToArray(); nMemID = lMemID.ToArray(); nMemType =
lMemType.ToArray(); nBookID = lBookID.ToArray(); nTitleID = lTitleID.ToArray(); nTitle
= lTitle.ToArray();  nPrice = lPrice.ToArray();


        End: ;
        }

        //Check for Member's old fines
```

```
        public void FindAccFines(string MemID, ref bool isFine, ref double accFine)
        {
            //Get old fines in account
            string sqlMFine = string.Format("SELECT Fine FROM Member WHERE MemberID =
'{0}'", MemID);
            OleDbCommand cmdMFine = new OleDbCommand(sqlMFine, db.con);
            OleDbDataReader drMFine = cmdMFine.ExecuteReader();

            if (drMFine.Read() && drMFine.HasRows)
            {
                accFine = double.Parse(drMFine["Fine"].ToString());

                if (accFine != 0) // If unpaid fines are there.
                { isFine = true; }

            }
            else MessageBox.Show("Invalid Member ID", "", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }

        // Pay Member's old fines
        public void PayAccFines(string MemID, double accFine)
        {

            //Reset the account fines.
            string sqlaccFineNow = string.Format("UPDATE Member SET Fine = Fine - {0}
WHERE MemberID = '{1}'", accFine, MemID);
            OleDbCommand cmdaccFineNow = new OleDbCommand(sqlaccFineNow, db.con);
            if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            cmdaccFineNow.ExecuteNonQuery();

            //Record transaction details to Cash
            string sqlaccFineNowCash = string.Format("INSERT INTO Cash(Event, TDetail,
TDate, Amount) VALUES ('AccFinePaid', '{0}', '{1}', '{2}')", string.Format("MemberID =
{0}", MemID), DateTime.Now.ToString(), accFine);
            OleDbCommand cmdaccFineNowCash = new OleDbCommand(sqlaccFineNowCash,
db.con);
            cmdaccFineNowCash.ExecuteNonQuery();

            MessageBox.Show(string.Format("Transaction of account fine balance of
Rs.{0} has beeen successfully recorded and the Member {1}'s account has been cleared of
all previous fines.", accFine, MemID), "Transaction Successful", MessageBoxButtons.OK,
MessageBoxIcon.Information);

        }

        //Check and return fine details for one book
        public void FindBookFine(string BookID, out bool isBFine, out double Fine, out
int Days, out DateTime iDate, out string MemID, out bool IsBlocked, out int MTi)
        {
            //Declarations
            string MType; DateTime LDate; MTi = 2;

            //Temporary Declarations
```

```
            isBFine = false; Fine = 0; Days = 0; iDate = DateTime.Today; MemID =
"ERROR";

            //Get the Lending detains from LendStatus relation
            string sqlLDetail = string.Format("SELECT LendDate, MemberID, Extend FROM
LendStatus WHERE BookID = '{0}'", BookID);
            OleDbCommand cmdLDetail = new OleDbCommand(sqlLDetail, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drLDetail = cmdLDetail.ExecuteReader(); drLDetail.Read();


            MemID = drLDetail["MemberID"].ToString(); // Get MemID

            //Get MemberType from MemID to get number of days and get old fines in
account
            string sqlMDetail = string.Format("SELECT MType, MStatus FROM Member WHERE
MemberID = '{0}'", MemID);
            OleDbCommand cmdMDetail = new OleDbCommand(sqlMDetail, db.con);
            OleDbDataReader drMDetail = cmdMDetail.ExecuteReader(); drMDetail.Read();

            //Load Status to variable
            if (drMDetail["MStatus"].ToString() == "Blocked") { IsBlocked = true;} else
IsBlocked = false;

            //Load Type to variable
            MType = drMDetail["MType"].ToString();

            //From MType, get number of days can be kept
            if (MType == "Adult") MTi = 0; else MTi = 1;

            LDate = DateTime.Parse(drLDetail["LendDate"].ToString());
            iDate = LDate.AddDays(set.Btime[MTi]); // Find the date when the book
should be returned
            Days = DateTime.Today.Subtract(iDate).Days; // Find the days between today
and that date

            if (Days <= 0) isBFine = false;

            else
            {
                isBFine = true;

                Fine = set.Fine[MTi] * Days; // Calculate Fines from Days and fine per
day
                Fine = double.Parse(Math.Round(decimal.Parse(Fine.ToString()),
1).ToString()); //Round to 1 decimal
            }


        }

        //Remove a book of given BookID
        public void RemBook(string BookID, string Reason, bool isBlack)
        {
            //Get TitleID from Book table
```

```
            string sqlBook = string.Format("SELECT TitleID FROM Book WHERE BookID =
'{0}'", BookID);
            OleDbCommand cmdBook = new OleDbCommand(sqlBook, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drBook = cmdBook.ExecuteReader(); drBook.Read();
            string TitleID = drBook["TitleID"].ToString();

            //Select Price from Title Table with BookID
            string sqlTPrice = string.Format("SELECT Price FROM Title WHERE TitleID =
'{0}'", TitleID);
            OleDbCommand cmdTPrice = new OleDbCommand(sqlTPrice, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drTPrice = cmdTPrice.ExecuteReader(); drTPrice.Read();
            double Price = double.Parse(drTPrice["Price"].ToString());

            //Count No. of books under Title
            string sqlBookCnt = string.Format("SELECT COUNT(BookID) as BookCnt FROM
Book WHERE TitleID = '{0}'", TitleID);
            OleDbCommand cmdBookCnt = new OleDbCommand(sqlBookCnt, db.con);
            OleDbDataReader drBookCnt = cmdBookCnt.ExecuteReader(); drBookCnt.Read();
            double BookCnt = double.Parse(drBookCnt["BookCnt"].ToString());

            //Check book in Book Table
            string sqlLendCheck = string.Format("SELECT MemberID FROM LenDstatus WHERE
BookID = '{0}'", BookID);
            OleDbCommand cmdLendCheck = new OleDbCommand(sqlLendCheck, db.con);
            OleDbDataReader drLendCheck = cmdLendCheck.ExecuteReader();
drLendCheck.Read();

            //If book is lent, remove book from there.
            if (drLendCheck.HasRows)
            {
                if (isBlack)
                {
                    string MemID1 = drLendCheck["MemberID"].ToString();
                    DialogResult black = MessageBox.Show(string.Format("This book has
been borrowed by Member '{0}'. \r\n\r\nIf you think the member has stolen the book, you
may add him/her to blacklist and if the member checks in or tries to create another
membership with same NIC number or Email, an alert will be raised. Do you want to add
this member to blacklist?", MemID1), "Add to Blacklist?", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);
                    if (black == DialogResult.Yes)
                    {
                        string sqlBlack = string.Format("UPDATE Member SET MStatus =
'Blocked' WHERE MemberID = '{0}'", MemID1);
                        OleDbCommand cmdBlack = new OleDbCommand(sqlBlack, db.con);
                        cmdBlack.ExecuteNonQuery();
                    }
                }
                {
                    string sqlRemLend = string.Format("DELETE FROM LendStatus WHERE
BookID = '{0}'", BookID);
                    OleDbCommand cmdRemLend = new OleDbCommand(sqlRemLend, db.con);
                    cmdRemLend.ExecuteNonQuery();
                }
```

```
            }
            //Remove book from Book Table
            string sqlRemBook = string.Format("DELETE FROM Book WHERE BookID = '{0}'",
BookID);
            OleDbCommand cmdRemBook = new OleDbCommand(sqlRemBook, db.con);
            cmdRemBook.ExecuteNonQuery();

            // If only one book was under the title, remove the title also, since there
are no books now.
            if (BookCnt == 1)
            {
                string sqlRemTitle = string.Format("DELETE FROM Title WHERE TitleID =
'{0}'", TitleID);
                OleDbCommand cmdRemTitle = new OleDbCommand(sqlRemTitle, db.con);
                cmdRemTitle.ExecuteNonQuery();
            }

        }

        //Get Title Details of TitleID
        public void TitleInfo(string TitleID, out bool IsValidID, out string Title,out
string Author,out string Genre,out string  Publish,out string  Pg,out string  ISBN,out
string  Ty,out string  Price,out string  Popula,out string Times, out int NoAvail, out
string NoBooks)
        {
            //Temporary Declarations
            Title = Author = Genre = Publish = Pg = ISBN = Ty = Price = Popula = Times
= NoBooks = ""; NoAvail = 0;

            //Get Details from Title Table
            string sqlTitle = string.Format("SELECT * FROM Title WHERE TitleID =
'{0}'", TitleID);
            OleDbCommand cmdTitle = new OleDbCommand(sqlTitle, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drTitle = cmdTitle.ExecuteReader(); drTitle.Read();

            //Disclaimer: If Invalid TitleID
            if (drTitle.HasRows) IsValidID = true; else { IsValidID = false; goto End;}

            //Declarations of Title Information

            Title = drTitle["BTitle"].ToString();
            Author = drTitle["Author"].ToString();
            Genre = drTitle["Genre"].ToString();
            Publish = drTitle["Publisher"].ToString();
            Pg = drTitle["Pg"].ToString();
            ISBN = drTitle["ISBN"].ToString();
            Ty = drTitle["BType"].ToString();
            if (Ty == "ALend") Ty = "Adult Lending"; else if (Ty == "CLend") Ty =
"Children Lending"; else Ty = "Reference";
            Price = drTitle["Price"].ToString();
            Popul p = new Popul();
            Popula = p.BookCalc(null, TitleID, false).ToString();
            Times = drTitle["TimesBorrowed"].ToString();
```

```
            //Get Associated Book Details
            string sqlBookD = string.Format("SELECT COUNT(BookID) as BCount FROM Book
WHERE TitleID = '{0}'", TitleID);
            OleDbCommand cmdBookD = new OleDbCommand(sqlBookD, db.con);
            OleDbDataReader drBookD = cmdBookD.ExecuteReader(); drBookD.Read();

            NoBooks = drBookD["BCount"].ToString();
            NoAvail = int.Parse(NoBooks);

            //Get BooksIDs associated with Title
            string sqlEachBookID = string.Format("SELECT BookID FROM Book WHERE TitleID
= '{0}'", TitleID);
            OleDbCommand cmdEachBookID = new OleDbCommand(sqlEachBookID, db.con);
            OleDbDataReader drEachBookID = cmdEachBookID.ExecuteReader();

            //For Each BookID
            while (drEachBookID.Read())
            {

                string BookID = drEachBookID["BookID"].ToString();

                //Check wheather each book has been lent.
                string sqlLent = string.Format("SELECT BookID FROM LendStatus WHERE
BookID = '{0}'", BookID);
                OleDbCommand cmdLent = new OleDbCommand(sqlLent, db.con);
                OleDbDataReader drLent = cmdLent.ExecuteReader(); drLent.Read();

                //IF lent, decrement the no. of available books
                if (drLent.HasRows) NoAvail = NoAvail - 1;
            }
        End: ;
        }

        //Expire the members
        public void MemExpire()
        {
            //Declarations
            DateTime JDate, iDate, Now; string MemID, Mtype; int MTi, x = 0;

            Now = DateTime.Today;

            //Read Each members
            string sqlMem = string.Format("SELECT MemberID, MType, DateJoined FROM
Member WHERE MStatus = 'Valid'");
            OleDbCommand cmdMem = new OleDbCommand(sqlMem, db.con);
            if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drMem = cmdMem.ExecuteReader();

            //For each member
            while (drMem.Read())
            {
                //Declare
                MemID = drMem["MemberID"].ToString(); Mtype =
drMem["MType"].ToString(); JDate = DateTime.Parse(drMem["DateJoined"].ToString());
```

```
                //Set Member Type index
                //Get the date, a member of such type should have joined, if his
membership to be expired today.
                if (Mtype == "Adult") MTi = 0; else MTi = 1;
                iDate = Now.AddMonths(-set.Expire[MTi]);

                //Compare dates
                int i = DateTime.Compare(JDate, iDate); // if in past,
                if (i < 0)
                {
                    string sqlUMem = string.Format("UPDATE Member SET MStatus =
'Expired' WHERE MemberID = '{0}'", MemID) ;
                    OleDbCommand cmdUMem = new OleDbCommand(sqlUMem, db.con);
                    if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                    cmdUMem.ExecuteNonQuery(); x++;
                }
            }
            if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();

        }
    }
}
```

Main Form (frmMain)

```
namespace Alexandria
{
    public partial class frmMain : Form
    {
        public frmMain()
        {
            InitializeComponent(); menu.Renderer = new MyRenderer();
        }

       private void frmMain_Load(object sender, EventArgs e)
        {
            set s = new set(); s.recall();
            Methods m = new Methods(); m.MemExpire();


            if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();

            //Open login form
            frmLogin f = new frmLogin(); f.ShowDialog();

            if (cvar.UType == "Mem") Mem.Visible = true;
            else if (cvar.UType == "Lib") { Issues.Visible = MgBook.Visible =
MgMem.Visible = Notif.Visible = true; }
            else if (cvar.UType == "Admin") { Mem.Visible = Issues.Visible =
MgBook.Visible = MgMem.Visible = Admin.Visible = Notif.Visible = true; }

            if (cvar.UType == "Admin" || cvar.UType == "Lib")
            {
                //Check for Notifications
                if(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();

                int N = 0; DateTime[] nLend = new DateTime[] { }; ; string[] nMemID =
new string[] { }; string[] nMemType = new string[] { }; string[] nBookID = new string[]
{ }; string[] nTitleID = new string[] { }; string[] nTitle = new string[] { }; double[]
nPrice = new double[] { }; bool NError = false;
                m.Notif(ref N, ref nLend, ref nMemID, ref nMemType, ref nBookID, ref
nTitleID, ref nTitle, ref nPrice, ref NError);

                if (N > 0) // If notifications are there.
                {
                    Notif.BackColor = Color.Red;
                    DialogResult notif = MessageBox.Show(string.Format("There are {0}
alert notifications about the books which have not been returned for a long time. Do
you want to check the notification window?", N), "Alert Notifications Pending",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);
                    if (notif == DialogResult.Yes) { frmNotif n = new frmNotif();
n.ShowDialog(); }
                }
                if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            }
        }
```

```
private void newBooksStockToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmAddBook f = new frmAddBook(); f.MdiParent = this; f.Show();
}

private void testFormToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmTest f = new frmTest(); f.MdiParent = this; f.Show();
}

private void newMembershipToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmMemNew f = new frmMemNew(); f.MdiParent = this; f.Show();
}

private void memberDetailsToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmMemDetails f = new frmMemDetails(); f.MdiParent = this; f.Show();

}

private void bookIssuesToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmIssues f = new frmIssues(); f.MdiParent = this; f.Show();

}

private void loadSettingsToolStripMenuItem_Click(object sender, EventArgs e)
{
    set s = new set(); s.recall();
}

private void bookReturnsToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmReturnExt f = new frmReturnExt(); f.MdiParent = this; f.Show();

}

private void bookDetailsToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmBookDetail f = new frmBookDetail(); f.MdiParent = this;
    f.Show(); f.Height = 145; // Later 395
}

private void removeBooksToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmRemBook f = new frmRemBook(); f.MdiParent = this; f.Show();
}

private void preferancesToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmSetting f = new frmSetting(); f.ShowDialog();
}
```

```
private void transactionsToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmCash f = new frmCash(); f.MdiParent = this; f.Show();
}

private void notificationsToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmNotif f = new frmNotif(); f.MdiParent = this; f.Show();
}

private void memberLogToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmViewCheckIO f = new frmViewCheckIO(); f.MdiParent = this; f.Show();
}

private void extrasToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmExtra f = new frmExtra(); f.MdiParent = this; f.Show();
}

private void bookDetailsToolStripMenuItem_Click_1(object sender, EventArgs e)
{
    frmBookDetail f = new frmBookDetail(); f.Show();
}

private void checkInOutToolStripMenuItem_Click_1(object sender, EventArgs e)
{
    frmCheckIO f = new frmCheckIO(); f.MdiParent = this; f.Show();
}

private void bookSearchToolStripMenuItem_Click_1(object sender, EventArgs e)
{
    frmBSearch f = new frmBSearch(); f.MdiParent = this; f.Show();
}

//Menu selected color

private class MyRenderer : ToolStripProfessionalRenderer
{
    public MyRenderer() : base(new MyColors()) { }
}

private class MyColors : ProfessionalColorTable
{
    public override Color MenuItemSelected
    {
        get { return Color.LightGray; }
    }
    public override Color MenuItemSelectedGradientBegin
    {
        get { return Color.DarkGray; }
    }
    public override Color MenuItemSelectedGradientEnd
    {
        get { return Color.DarkSlateGray; }
```

```
            }
            public override Color MenuItemPressedGradientBegin
            {
                get
                { return Color.LightGray; }
            }

            public override Color MenuItemPressedGradientEnd
            {
                get
                { return Color.DimGray; }
            }

        }

        private void updateMembershipToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            frmMemDetails f = new frmMemDetails(); f.MdiParent = this;

            f.btnUpdate.Size = f.lblButtons.Size;
            f.btnUpdate.Location = f.lblButtons.Location;
            f.btnRemove.Visible = f.btnRenew.Visible = false;
            f.Text = "Update Membership";

            f.Show();
        }

        private void renewMembershipToolStripMenuItem_Click(object sender, EventArgs e)
        {
            frmMemDetails f = new frmMemDetails(); f.MdiParent = this;

            f.btnRenew.Size = f.lblButtons.Size;
            f.btnRenew.Location = f.lblButtons.Location;
            f.btnRemove.Visible = f.btnUpdate.Visible = false;
            f.Text = "Renew Membership";
            f.TlblMiD.Text = " Old Member ID";

            f.Show();
        }

        private void deleteMemberToolStripMenuItem_Click(object sender, EventArgs e)
        {
            frmMemDetails f = new frmMemDetails(); f.MdiParent = this;

            f.btnRemove.Size = f.lblButtons.Size;
            f.btnRemove.Location = f.lblButtons.Location;
            f.btnRenew.Visible = f.btnUpdate.Visible = false;
            f.Text = "Delete Membership";
            f.TlblMiD.Text = "Enter Member ID";

            f.Show();
        }

        private void loginToolStripMenuItem_Click(object sender, EventArgs e)
```

```
        {
            frmLogin f = new frmLogin(); f.Show();
        }

    }

}
```

Login Form (frmLogin)

```
namespace Alexandria
{
    public partial class frmLogin : Form
    {
        string Type;

        public frmLogin()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Hide(); frmLogin f = new frmLogin(); f.ShowDialog();
        }

        public void btnLogIn_Click(object sender, EventArgs e)
        {
            frmMain f = new frmMain();

            //Remove single & double quoteswhich cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }

            //If member

            if (Type == "Member") { cvar.UType = "Mem"; this.Close(); goto End; }

            //Check Null
            if ((txtPW.Text == "" || txtUser.Text == ""))
            { MessageBox.Show("Username and password fields cannot be left blank",
"Blank Fields", MessageBoxButtons.OK, MessageBoxIcon.Warning); goto End; }

            //Declaratioons
            string UN = txtUser.Text, PW = txtPW.Text;

            //Check in db.
            string sqlUser = string.Format("SELECT UName FROM UserAccount WHERE UName =
'{0}' AND UType = '{1}' AND Pwd = '{2}'", UN, Type, PW);
            OleDbCommand cmdUser = new OleDbCommand(sqlUser, db.con);
            if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drUser = cmdUser.ExecuteReader();

            //If UN, UT and PW mismatch, Error
            if (!(drUser.Read() && drUser.HasRows)) { MessageBox.Show("Username,
Password and User Type mismatch. Please try again.", "Credentials Mismatch",
MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }
            //Else continue


            if (Type == "Admin") cvar.UType = "Admin"; else if (Type == "Librarian")
cvar.UType = "Lib"; else { goto End; };
```

```
    this.Close();

        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void cboxTy_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (cboxTy.SelectedIndex == 0) { Type = "Member"; btnLogIn.Enabled = true;
cboxTy.Enabled = false; }
            else if (cboxTy.SelectedIndex == 1) { Type = "Librarian"; gboxCred.Enabled
= true; btnLogIn.Enabled = true; cboxTy.Enabled = false; }
            else if (cboxTy.SelectedIndex == 2) { Type = "Admin"; gboxCred.Enabled =
true; btnLogIn.Enabled = true; cboxTy.Enabled = false; }
        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
    }
}
```

Notifications (frmNotif)

```
namespace Alexandria
{
    public partial class frmNotif : Form
    {
        //Create Datatable

        public DataTable dtAllNotif = new DataTable();

        public frmNotif()
        {
            InitializeComponent();
        }

        private void frmNotif_Load(object sender, EventArgs e)
        {
            fillAllNotif();
        }

        private void fillAllNotif()
        {
            //Call Notif Method

            Methods m = new Methods();
            int N = 0; DateTime[] nLend = new DateTime[] { }; ; string[] nMemID = new
string[] { }; string[] nMemType = new string[] { }; string[] nBookID = new string[] {
}; string[] nTitleID = new string[] { }; string[] nTitle = new string[] {}; double[]
nPrice = new double[] { }; bool NError = false;
            m.Notif(ref N, ref nLend, ref nMemID, ref nMemType, ref nBookID, ref
nTitleID, ref nTitle, ref nPrice, ref NError);

            if (NError) { MessageBox.Show("There was an error reading notifications");
goto End; }
            if (N == 0) { MessageBox.Show("There are no alerts to be displayed", "No
Alerts", MessageBoxButtons.OK, MessageBoxIcon.Information); goto End; }
            this.Text = this.Text + string.Format(" [{0}]", N);

            // Design DataTable
            dtAllNotif.Columns.Add("DTDate");
            dtAllNotif.Columns[0].DataType = System.Type.GetType("System.DateTime");


            dtAllNotif.Columns.Add("Date_Lent");
            dtAllNotif.Columns.Add("Title");
            dtAllNotif.Columns.Add("Price");
            dtAllNotif.Columns.Add("MemberID");
            dtAllNotif.Columns.Add("Member_Type");
            dtAllNotif.Columns.Add("Title ID");
            dtAllNotif.Columns.Add("Book ID");

            //Add contents in loop
```

```
            for (int i = 0; i < nLend.Length; i++)
            {
                dtAllNotif.Rows.Add(nLend[i],
DateTime.Parse(nLend[i].ToString()).ToString("dd-MM-yyyy"), nTitle[i], nPrice[i],
nMemID[i], nMemType[i], nTitleID[i], nBookID[i]);
            }
            dtResults.DataSource = dtAllNotif;
            lblNo.Text = dtAllNotif.Rows.Count.ToString();

            //Design Data View
            dtResults.Columns[1].HeaderText = "Date Lent";
            dtResults.Columns[4].HeaderText = "Member ID";
            dtResults.Columns[5].HeaderText = "Member Type";
            dtResults.Columns[0].Visible = false;

        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void FilterNoif(string MemID, string MType, string DateSQL)
        {
            if (dtAllNotif.Rows.Count == 0) { MessageBox.Show("There are no alterts to
filter.", "No Alerts", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }

            string Filter = string.Format("MemberID LIKE '%{0}%' AND Member_Type LIKE
'%{1}%'{2}", MemID, MType, DateSQL);

            DataView dvresults = new DataView(dtAllNotif, Filter , "",
DataViewRowState.CurrentRows);

            //Show Results
            dtResults.DataSource = dvresults;

            //Design Data View
            dtResults.Columns[1].HeaderText = "Date Lent";
            dtResults.Columns[4].HeaderText = "Member ID";
            dtResults.Columns[5].HeaderText = "Member Type";

        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void btnFilter_Click(object sender, EventArgs e)
        {
            //Remove single & double quoteswhich cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }

            //Declarations
            string MemID = txtMemID.Text, MType = cboxMType.Text, Time = cboxDate.Text;
; DateTime iDate;

            if (MemID == "") { }
            else
            {
```

```
                string sqlIsMem = string.Format("SELECT MemberID FROM Member WHERE
MemberID = '{0}'", MemID);
                OleDbCommand cmdIsMem = new OleDbCommand(sqlIsMem, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drIsMem = cmdIsMem.ExecuteReader(); drIsMem.Read();

                //Disclaimer
                if (!drIsMem.HasRows) { MessageBox.Show("No member is found with such
Member ID", "Invalid Member ID", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End;
}
            }

            //Declare MemType
            if (MType == "Child Member") MType = "Child"; else if (MType == "Adult
Member") MType = "Adult"; else MType = "";

            //Declare Date
            if (Time == "Today") iDate =
DateTime.Parse(DateTime.Today.ToShortDateString());
            else if (Time == "Past Week") iDate = DateTime.Today.AddDays(-7);
            else if (Time == "Last Two Weeks") iDate = DateTime.Today.AddDays(-14);
            else if (Time == "Past Month") iDate = DateTime.Today.AddMonths(-1);
            else if (Time == "Last Two Months") iDate = DateTime.Today.AddMonths(-2);
            else if (Time == "Last Six Months") iDate = DateTime.Today.AddMonths(-6);
            else if (Time == "Past Year") iDate = DateTime.Today.AddYears(-1);
            else iDate = DateTime.Today;

            //Write Date sql
            string DateSQL;
            if (iDate == DateTime.Today) DateSQL = "";
            else DateSQL = string.Format(" AND DTDate >= #{0}#", iDate);


            //Call Filter Method
            FilterNoif(MemID, MType, DateSQL);

            if (dtResults.Rows.Count == 0) MessageBox.Show("There are no Notifications
within the given filters. Try removing some filters.", "No Notifications",
MessageBoxButtons.OK, MessageBoxIcon.Warning);


        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void dtResults_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            if (dtResults.SelectedCells.Count != 0)
            {
                string NowValue = dtResults.CurrentCell.Value.ToString();
                int NowCol = dtResults.SelectedCells[0].ColumnIndex;
                string NowBook =
dtResults.Rows[dtResults.CurrentCell.RowIndex].Cells[7].Value.ToString();

                frmBookDetail b = new frmBookDetail();
                frmMemDetails m = new frmMemDetails();
```

```
                if (NowCol == 4)
                { m.txtMemID.Text = NowValue; m.ShowDialog(); }
                else if (NowCol == 6)
                { b.txtTitleID.Text = NowValue; b.btnGetTitle_Click(sender, e);
b.ShowDialog(); }
                else if (NowCol == 7 || NowCol == 2)
                { b.txtBookID.Text = NowBook; b.btnGetBook_Click(sender, e);
b.ShowDialog();}
            }
        }

    }
}
```

Add Books Form (frmAddBook)

```
namespace Alexandria
{
    public partial class frmAddBook : Form
    {

        public bool TitleNew = false; string TitleID = "";

        public frmAddBook()
        {
            InitializeComponent();
        }

        private void frmAddBook_Load(object sender, EventArgs e)
        {

            string sqlTitle = "SELECT BTitle FROM Title";
            string sqlAuthor = "SELECT DISTINCT Author FROM Title";
            string sqlGenre = "SELECT DISTINCT Genre FROM Title";

            OleDbCommand cmdTitle = new OleDbCommand(sqlTitle, db.con);
            OleDbCommand cmdAuthor = new OleDbCommand(sqlAuthor, db.con);
            OleDbCommand cmdGenre = new OleDbCommand(sqlGenre, db.con);

            if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();

            OleDbDataReader drTitle = cmdTitle.ExecuteReader();
            OleDbDataReader drAuthor = cmdAuthor.ExecuteReader();
            OleDbDataReader drGenre = cmdGenre.ExecuteReader();

            while (drTitle.Read()) { cboxTitle.Items.Add(drTitle["BTitle"].ToString());
}
            while (drAuthor.Read()) {
cboxAuthor.Items.Add(drAuthor["Author"].ToString()); }
            while (drGenre.Read()) { cboxGenre.Items.Add(drGenre["Genre"].ToString());
}


            cboxType.SelectedIndex = 1;
            db.con.Close();



        }


        private void btnAdd_Click(object sender, EventArgs e)
        {


            //Disclaimers Start
```

```
            if (cboxTitle.Text == "" || cboxType.Text == "" || cboxAuthor.Text == "" ||
cboxGenre.Text == "" || txtPages.Text == "" || txtPrice.Text == "" || txtQty.Text == ""
|| txtPublisher.Text == "")
            { MessageBox.Show("Stock cannot be added because, one or many of the
nessasary fields are left blank. Please fill all the fields and try again", "Field Left
Blank", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }

            //Remove single & double quotes which cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }

            int Qty; bool q = int.TryParse(txtQty.Text, out Qty);
            if(q==false) {MessageBox.Show("Insert a valid value for No. of books",
"Invalid no. of books", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }
            //Disclaimers End

            //Declarations
            string Title = cboxTitle.Text; string Maxbook = "B0000"; string BookID;
double Price, Trans; Methods methods = new Methods();

            //When Creating new title.
            if (TitleNew == true)
            {

                //Disclaimer + integer Declarations

                int Pages; bool pg = int.TryParse(txtPages.Text, out Pages); bool p =
double.TryParse(txtPrice.Text, out Price);
                if (q == false || p == false || pg == false) { MessageBox.Show("Insert
a valid value for No. of books, No. Pages or Price"); goto End; }


                //Declarations

                string Type, Author, Genre, ISBN, Publisher; TitleID = "";
                Author = cboxAuthor.Text; Genre = cboxGenre.Text; ISBN = txtISBN.Text;
Publisher = txtPublisher.Text;

                if (cboxType.SelectedIndex == 0) Type = "CLend"; else if
(cboxType.SelectedIndex == 1) Type = "ALend"; else Type = "Ref";

                //Creating New TitleID
                {

                    //Getting MaxID from Databse
                    string sqlMaxTitle = "SELECT MAX(TitleID) as MaxTitle,
COUNT(TitleID) as CountID FROM Title";
                    OleDbCommand cmdMaxTitle = new OleDbCommand(sqlMaxTitle, db.con);
                    if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                    OleDbDataReader drMaxTitle = cmdMaxTitle.ExecuteReader();
drMaxTitle.Read();


                    //If there are titles in DB, formulate a new TitleID
```

```
                    if (drMaxTitle["CountID"].ToString() != "0")
                    {
                        // Calling NewID method
                        TitleID = methods.NewID(drMaxTitle["MaxTitle"].ToString());
                    }
                    else TitleID = "T0001";
                }

                //Declartions End


                //Add Values to Relation: Title

                string sqlTitleNew = string.Format("INSERT INTO Title(TitleID, BTitle,
BType, Price, Genre, ISBN, Pg, Author, Publisher) VALUES('{0}', '{1}', '{2}', '{3}',
'{4}', '{5}', '{6}', '{7}', '{8}')", TitleID, Title, Type, Price, Genre, ISBN, Pages,
Author, Publisher);
                OleDbCommand cmdTitleNew = new OleDbCommand(sqlTitleNew, db.con);
                if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                cmdTitleNew.ExecuteNonQuery();

            }
            else Price = double.Parse(txtPrice.Text);

            // Get the highest BookID from Database
            string sqlMaxBook = "SELECT MAX(BookID) as MaxBook, COUNT(BookID) as
CountID FROM Book";
                OleDbCommand cmdMaxBook = new OleDbCommand(sqlMaxBook, db.con);
                if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drMaxbook =
cmdMaxBook.ExecuteReader();drMaxbook.Read();

                if (drMaxbook["CountID"].ToString() != "0") Maxbook =
drMaxbook["MaxBook"].ToString(); else Maxbook = "B0000";

            // Adding books in a loop.

            for (int book = 1; book <= Qty; book++)
            {
                //Calling new ID method to create newID
                BookID = methods.NewID(Maxbook);

                //Adding a book with such book ID & TitleID to DB

                string sqlbook = string.Format("INSERT INTO Book (BookID, TitleID)
VALUES ('{0}','{1}')", BookID, TitleID);
                OleDbCommand cmdbook = new OleDbCommand(sqlbook, db.con);
                if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                cmdbook.ExecuteNonQuery();

                //Make the BookID as the new Book_MaxID to help the next loop
                Maxbook = BookID;
            }

            Maxbook = ""; BookID = "";
```

```
            // Adding Transaction details

            Trans = Price * Qty; string TDetail = string.Format("{0} books under the
Title: {1}", Qty, Title);

            string sqlTrans = string.Format("INSERT INTO Cash (TDate, Amount, TDetail,
Event) VALUES('{0}', {1}, '{2}', 'NewStock')", DateTime.Now.ToString(), -Trans,
TDetail);
            OleDbCommand cmdTrans = new OleDbCommand(sqlTrans, db.con);
            cmdTrans.ExecuteNonQuery();
            // Transactions added.

            //Successful Message
            MessageBox.Show(string.Format("{0} Book(s) and/or Book Title have been
added to collection and transaction has been recorded successfully", Qty.ToString()),
"Books Successfully Added", MessageBoxButtons.OK, MessageBoxIcon.Information);

            //Reset form
            this.Close(); frmAddBook f = new frmAddBook(); f.Show();

            End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void btnCancel_Click(object sender, EventArgs e)
        {
            DialogResult sure = MessageBox.Show("Do you want to discard these
details?", "Are you sure?", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
            if (sure == DialogResult.Yes) this.Close(); frmAddBook f = new
frmAddBook(); f.Show();
        }

        private void btnSet_Click(object sender, EventArgs e)
        {


            if (cboxTitle.Text == "") goto End;

            //Remove single & double quoteswhich cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }


                cboxTitle.Enabled = false;
                string sqlTitle = string.Format("SELECT TitleID FROM Title WHERE BTitle
= '{0}'", cboxTitle.Text);
                OleDbCommand cmdTitle = new OleDbCommand(sqlTitle, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drTitle = cmdTitle.ExecuteReader();

                string sqlTitleC = string.Format("SELECT Count(TitleID) AS TitCount
FROM Title WHERE BTitle = '{0}'", cboxTitle.Text);
                OleDbCommand cmdTitleC = new OleDbCommand(sqlTitleC, db.con);
                OleDbDataReader drTitCount = cmdTitleC.ExecuteReader();
```

```
                if (drTitle.Read() && drTitCount.Read() &&
int.Parse(drTitCount["TitCount"].ToString()) > 0)
                {
                    TitleNew = false; TitleID = drTitle["TitleID"].ToString();
                    DialogResult a = MessageBox.Show("The Title and corresponding
details cannot be changed later. Do you want to continue?", "Are you sure?",
MessageBoxButtons.YesNo, MessageBoxIcon.Information);

                    if (a == DialogResult.Yes)
                    {
                        TitleNew = false;

                        string sqlDetail = string.Format("SELECT * FROM Title WHERE
BTitle = '{0}'", cboxTitle.Text);
                        OleDbCommand cmdDetail = new OleDbCommand(sqlDetail, db.con);
OleDbDataReader drDetail = cmdDetail.ExecuteReader();

                        if (drDetail.Read())
                        {
                            string sType = drDetail["BType"].ToString();
                            if (sType == "CLend") cboxType.SelectedIndex = 0; else
cboxType.SelectedIndex = 1;

                            txtPrice.Text = drDetail["Price"].ToString();
                            txtISBN.Text = drDetail["ISBN"].ToString();
                            txtPages.Text = drDetail["Pg"].ToString();
                            cboxGenre.Text = drDetail["Genre"].ToString();
                            txtPublisher.Text = drDetail["Publisher"].ToString();
                            cboxAuthor.Text = drDetail["Author"].ToString();

                        }
                        else { MessageBox.Show("ERROR Reading Details"); goto End; }

                        btnCancel.Enabled = cboxGenre.Enabled = cboxAuthor.Enabled =
cboxType.Enabled = txtISBN.Enabled = txtPages.Enabled = txtPrice.Enabled =
btnSet.Enabled = txtPublisher.Enabled = false; txtQty.Enabled = btnAdd.Enabled = true;

                    }
                    else if (a == DialogResult.No) { cboxTitle.Focus();
cboxTitle.Enabled = true; }

                }
                else
                {
                    DialogResult b = MessageBox.Show("The Title cannot be changed
later. Do you want to continue?", "Are you sure?", MessageBoxButtons.YesNo,
MessageBoxIcon.Information);
                    if (b == DialogResult.Yes) { btnCancel.Enabled = TitleNew = true;
cboxGenre.Enabled = cboxAuthor.Enabled = cboxType.Enabled = txtISBN.Enabled =
txtPages.Enabled = txtPrice.Enabled = txtQty.Enabled = btnAdd.Enabled =
txtPublisher.Enabled = true; cboxTitle.Enabled = btnSet.Enabled = false; }
                    else { cboxTitle.Focus(); cboxTitle.Enabled = true; }

                }
```

```
            End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();



        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            if (cboxTitle.Text != "")
            {
                DialogResult sure = MessageBox.Show("Do you want to discard these
details?", "Are you sure?", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
                if (sure == DialogResult.Yes) this.Close();
            }
            else this.Close();
        }


    }
}
```

Book Details Form (frmBookDetails)

```
namespace Alexandria
{
    public partial class frmBookDetail : Form
    {
        //Common Variable Decalartions
        string TitleID, BookID, Title, Author, Genre, Publish, Pg, ISBN, Ty, Price,
Popul, Times;
        //Declare specific variables
        int NoAvail = 0; string NoBooks; bool IsValidID;


        public frmBookDetail()
        {
            InitializeComponent();
        }

        public void btnGetTitle_Click(object sender, EventArgs e)
        {
            //Disclaimers
            if (txtTitleID.Text == "") goto End; TitleID = txtTitleID.Text;
            //Remove single & double quoteswhich cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }


            //Call method
            FillTitleInfo(TitleID);

            btnRemBook.Enabled = false;
        End: ;
        }

        public void FillTitleInfo(string TitleID)
        {


            //Call Method for details.
            Methods m = new Methods();
            m.TitleInfo(TitleID, out IsValidID, out Title, out Author, out Genre, out
Publish, out Pg, out ISBN, out Ty, out Price, out Popul, out Times, out NoAvail, out
NoBooks);

            //Disclaimer: If Invalid TitleID
            if (IsValidID == false) { MessageBox.Show("No Title exist with the given
TitleID", "Invalid Title ID", MessageBoxButtons.OK, MessageBoxIcon.Error);
txtTitleID.Clear(); goto End; }
            //If valid, continue

            //Modify the form
```

```
            lblLLAvail.Visible = lblLLLentTo.Visible = lblLLLentOn.Visible =
lblLLFine.Visible = lblAvail.Visible = lblLentTo.Visible = lblLentOn.Visible =
lblFine.Visible = false;
            txtBookID.Clear(); txtBookID.Enabled = txtTitleID.Enabled =
btnGetTitle.Visible = btnGetBook.Visible = false;
            MakeTaller();

            //Display all details
            lblTitle.Text = Title;
            lblAuthor.Text = Author;
            lblGenre.Text = Genre;
            lblPublish.Text = Publish;
            lblPg.Text = Pg;
            lblISBN.Text = ISBN;
            lblTy.Text = Ty;
            lblPrice.Text = Price;
            lblPop.Text = Popul;
            lblTimes.Text = Times;
            lblNoBooks.Text = NoBooks;
            lblAvNo.Text = NoAvail.ToString();


        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void frmBookDetail_Load(object sender, EventArgs e)
        {
           if (txtBookID.Text == "" && txtTitleID.Text == "") this.Height = 186;
        }

        public void btnGetBook_Click(object sender, EventArgs e)
        {
            // Disclaimers
            if (txtBookID.Text == "") goto End;
            //Remove single & double quoteswhich cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }


            //Declarations
            string Avail, BookID, MemID; DateTime LendD; double Fine;
            BookID = txtBookID.Text;

            //Get Details from Book Table
            string sqlBook = string.Format("SELECT TitleID FROM Book WHERE BookID =
'{0}'", BookID);
            OleDbCommand cmdBook = new OleDbCommand(sqlBook, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drBook = cmdBook.ExecuteReader(); drBook.Read();

            //Disclaimer: If Invalid BookID
            if (drBook.HasRows) { IsValidID = true; TitleID =
drBook["TitleID"].ToString(); } else { MessageBox.Show("No Book exist with the given
BookID", "Invalid BookID", MessageBoxButtons.OK, MessageBoxIcon.Error); IsValidID =
false; txtBookID.Clear(); goto End; }
```

```
            //Call Method for details.
            Methods m = new Methods();
            m.TitleInfo(TitleID, out IsValidID, out Title, out Author, out Genre, out
Publish, out Pg, out ISBN, out Ty, out Price, out Popul, out Times, out NoAvail, out
NoBooks);

            //Error Handling
            if (IsValidID == false) { MessageBox.Show("Some Error Occured"); goto End;
}

            //Get Lend Details from lendStatus
            string sqlLendD = string.Format("SELECT * FROM LendStatus WHERE BookID =
'{0}'", BookID);
            OleDbCommand cmdLendD = new OleDbCommand(sqlLendD, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drLendD = cmdLendD.ExecuteReader(); drLendD.Read();

            if (drLendD.HasRows) // If lent
            {
                //Call method to calculate fines
                bool isBFine, IsBlocked; int Days; DateTime iDate; int MTi;
                m.FindBookFine(BookID, out isBFine, out  Fine, out  Days, out  iDate,
out  MemID, out IsBlocked, out MTi);

                //Load into variables
                MemID = drLendD["MemberID"].ToString();
                Avail = "Lent";
                LendD = DateTime.Parse(drLendD["LendDate"].ToString());

                //Show details
                lblLentTo.Text = MemID;
                lblLentOn.Text = LendD.ToString("dd MMMM yyyy");
                lblFine.Text = Fine.ToString();

            }
            else
            {
                Avail = "Available";

                //Modify Form
                lblLLFine.Visible = lblLLLentOn.Visible = lblLLLentTo.Visible =
lblFine.Visible = lblLentOn.Visible = lblLentTo.Visible = false;
            }

            //Display all details

            lblTitle.Text = Title;
            lblAuthor.Text = Author;
            lblGenre.Text = Genre;
            lblPublish.Text = Publish;
            lblPg.Text = Pg;
            lblISBN.Text = ISBN;
            lblTy.Text = Ty;
            lblPrice.Text = Price;
```

```
            lblPop.Text = Popul;
            lblAvail.Text = Avail;

            btnRemtitle.Enabled = false;
            //Modify Form
            lblTimes.Visible = lblNoBooks.Visible = lblAvNo.Visible = false;
            txtTitleID.Clear(); txtBookID.Enabled = txtTitleID.Enabled =
btnGetTitle.Visible = btnGetBook.Visible = lblLLNoAvail.Visible = lblLLNoBooks.Visible
= lblLLTimes.Visible = false;
            MakeTaller();


        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void btnRemBook_Click(object sender, EventArgs e)
        {
            // Disclaimer
            DialogResult sure = MessageBox.Show("Are you sure you want to remove this
book? This cannot be undone.", "Are you sure?", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);
            if (sure == DialogResult.No) goto End;
            //If sure, continue

            BookID = txtBookID.Text;
            //Calling VisualBasic inputBox for Reason
            string Reason = Microsoft.VisualBasic.Interaction.InputBox("Please enter
the reasons for this removal:", "Give Reasons");

            //Call method to delete book
            Methods m = new Methods();
            m.RemBook(BookID, Reason, true);

            //Success
            MessageBox.Show("Book has been removed and details of reason have been
logged successfully.", "Removal Successful", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            Reset();


        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void btnRemtitle_Click(object sender, EventArgs e)
        {
            // Disclaimer
            DialogResult sure = MessageBox.Show("Are you sure you want to remove all
books under this title? This cannot be undone.", "Are you sure?",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (sure == DialogResult.No) goto End;
            //If sure, continue

            TitleID = txtTitleID.Text;
            //Calling VisualBasic inputBox for Reason
```

```
            string Reason = Microsoft.VisualBasic.Interaction.InputBox("Please enter
the reasons for this removal:", "Give Reasons");

            //Get BookIDs
            string sqlBook = string.Format("SELECT BookID FROM Book WHERE TitleID =
'{0}'", TitleID);
            OleDbCommand cmdBook = new OleDbCommand(sqlBook, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drBook = cmdBook.ExecuteReader();

            int n = 0;
            while (drBook.Read())
            {
                BookID = drBook["BookID"].ToString();

                //Call method to delete book
                Methods m = new Methods();
                m.RemBook(BookID, Reason, false);

                n = n + 1;
            }

            //Success
            MessageBox.Show(string.Format("{0} books under this title ha(ve)s been
removed and details reason have been logged successfully.", n), "Removal Successful",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            Reset();

            End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void btnReset_Click(object sender, EventArgs e)
        {
            Reset();
        }
        private void Reset()
        {
            this.Close(); frmBookDetail f = new frmBookDetail(); f.Show();
        }

        private void MakeTaller()
        {
            this.Height = 617;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }
```

```
        private void lblLentTo_Click(object sender, EventArgs e)
        {
            frmMemDetails m = new frmMemDetails();
            m.txtMemID.Text = lblLentTo.Text; m.Show();
        }
    }
}
```

Remove Books Form (frmRemBook)

```
namespace Alexandria
{
    public partial class frmRemBook : Form
    {
        bool IsValidID; string TitleID, BookID; int N, TotN; List<string> AvailBooks =
new List<string>();

        public frmRemBook()
        {
            InitializeComponent();
        }

        private void btnRBook_Click(object sender, EventArgs e)
        {
            // Disclaimers
            if (txtBookID.Text == "") goto End;
            //Remove single & double quoteswhich cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }


            //Declarations
            BookID = txtBookID.Text;

            //Check Book Table
            string sqlBook = string.Format("SELECT TitleID FROM Book WHERE BookID =
'{0}'", BookID);
            OleDbCommand cmdBook = new OleDbCommand(sqlBook, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drBook = cmdBook.ExecuteReader(); drBook.Read();

            //Disclaimer: If Invalid BookID
            if (drBook.HasRows) { IsValidID = true; TitleID =
drBook["TitleID"].ToString(); } else { MessageBox.Show("Please enter a valid BookID",
"Invalid BookID", MessageBoxButtons.OK, MessageBoxIcon.Error); IsValidID = false; goto
End; }

            // Disclaimer for sure
            DialogResult sure = MessageBox.Show("Are you sure you want to remove this
book? This cannot be undone.", "Are you sure?", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);
            if (sure == DialogResult.No) goto End;
            //If sure, continue
            txtTitleID.Clear();

            //Calling VisualBasic inputBox for Reason
            string Reason = Microsoft.VisualBasic.Interaction.InputBox("Please enter
the reasons for this removal:", "Give Reasons");

            //Call method to delete book
            Methods m = new Methods();
            m.RemBook(BookID, Reason, true);
```

```
            //Success
            MessageBox.Show("Book has been removed and details of cash and reason have
been logged successfully.", "Removal Successful", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            txtBookID.Clear();


        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void btnRTitle_Click(object sender, EventArgs e)
        {
            // Disclaimers
            if (txtTitleID.Text == "" || cboxNo.Text == "") goto End;
            //Remove single & double quoteswhich cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }
            if (cboxNo.Text != "All")
            {
                bool isN = int.TryParse(cboxNo.Text, out N);
                if (isN == false) { MessageBox.Show("Enter a valid number for no. of
books", "Invalid number", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }
            }


            //Declarations
            TitleID = txtTitleID.Text;

            //Check Book Table
            string sqlTitle = string.Format("SELECT BookID FROM Book WHERE TitleID =
'{0}'", TitleID);
            OleDbCommand cmdTitle = new OleDbCommand(sqlTitle, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drTitle = cmdTitle.ExecuteReader();

            //Disclaimer: If Invalid BookID
            if (drTitle.HasRows) { IsValidID = true; } else { MessageBox.Show("Please
enter a valid TitleID", "Invalid TitleID", MessageBoxButtons.OK, MessageBoxIcon.Error);
IsValidID = false; goto End; }

            //Count all books.
            string sqlBookCnt = string.Format("SELECT COUNT(BookID) AS BookCnt FROM
Book WHERE TitleID = '{0}'", TitleID);
            OleDbCommand cmdBookCnt = new OleDbCommand(sqlBookCnt, db.con);
            OleDbDataReader drBookCnt = cmdBookCnt.ExecuteReader(); drBookCnt.Read();
            TotN = int.Parse(drBookCnt["BookCnt"].ToString());



            //Substract Lent Books
            while (drTitle.Read())
            {
                BookID = drTitle["BookID"].ToString();
```

```
                //Check each bookID in LendStatus
                string sqlLentD = string.Format("SELECT BookID FROM LendStatus WHERE
BookID = '{0}'", BookID);
                OleDbCommand cmdLentD = new OleDbCommand(sqlLentD, db.con);
                OleDbDataReader drLentD = cmdLentD.ExecuteReader();

                //If Lent, remove
                if (drLentD.HasRows == false) AvailBooks.Add(BookID);

            }
            TotN = AvailBooks.Count;

            string ShowNo;
            if (TotN == 0) ShowNo = "no books"; else ShowNo = "only " + TotN.ToString()
+ " books";

            if (cboxNo.Text != "All" && TotN < N ) {
MessageBox.Show(string.Format("There are {0} available to remove at the moment.",
ShowNo), "Invalid number", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }

            // Disclaimer for sure
            DialogResult sure = MessageBox.Show("Are you sure you want to remove these
books under this title? This cannot be undone.", "Are you sure?",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (sure == DialogResult.No) goto End;
            //If sure, continue


            //Calling VisualBasic inputBox for Reason
            string Reason = Microsoft.VisualBasic.Interaction.InputBox("Please enter
the reasons for this removal:", "Give Reasons");

            if(cboxNo.Text == "All") // To remove all books under title
            {
                //Read Book table for all book IDs
                string sqlChkTitle = string.Format("SELECT BookID FROM Book WHERE
TitleID = '{0}'", TitleID);
                OleDbCommand cmdChkTitle = new OleDbCommand(sqlChkTitle, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drChkTitle = cmdChkTitle.ExecuteReader();

                while (drChkTitle.Read())
                {
                    BookID = drChkTitle["BookID"].ToString();

                    //Call method to remove
                    Methods m = new Methods();
                    m.RemBook(BookID, Reason, false);
                }
                goto Success;
            }

            for(int i = 0; i < N; i++)
            {
                //Call method to remove
```

```
                Methods m = new Methods();
                m.RemBook(AvailBooks[i], Reason, false);
            }


        Success:
            //Success

            string s;
        if (cboxNo.Text == "All") s = "All books have"; else if (cboxNo.Text == "1") s
= N.ToString() + " Book has"; else s = N.ToString() + " Books have";
            MessageBox.Show(string.Format("{0} been removed and details of reason have
been logged successfully.", s), "Removal Successful", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            txtBookID.Clear(); txtTitleID.Clear(); cboxNo.Text = "";

        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }
    }
}
```

Add Member Form (frmMemNew)

```
namespace Alexandria
{
    public partial class frmMemNew : Form
    {
        // Common Variable Declaration.

        string MemID, FName, LName, MType, Guard, NIC, Addr, Work, TP, Email, Status;
        int MTi; DateTime DateJ, DoB; double Fine, fee;

        public frmMemNew()
        {
            InitializeComponent();
        }


        private void btnNew_Click(object sender, EventArgs e)
        {
            // Disclaimers

            //Remove single & double quotes which cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox ||
c is RichTextBox) { c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"",
""); }}

            if (txtFName.Text == "" || txtLName.Text == "" || cboxType.Text == "" ||
txtAddr.Text == "" || txtNIC.Text == "" || txtWork.Text == "" || txtTP.Text == "" ||
txtDobDate.Text=="" || txtDobMon.Text == ""|| txtDobYear.Text == "" ||
txtDobDate.Text=="Date" || txtDobMon.Text == "Month"|| txtDobYear.Text == "Month")
            { MessageBox.Show("One or many of the required fields are left blank.
Please fill them and try again", "Field(s) left blank", MessageBoxButtons.OK,
MessageBoxIcon.Error); goto End; }
            if (txtNIC.Text.Length != 10) { MessageBox.Show("NIC number invalid",
"Invalid NIC", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }
            if (cboxType.Text == "Child Membership" && txtGuardian.Text == "") {
MessageBox.Show("A guardian should be specified for a Child Member", "Specify
Guardian", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }
            bool isDate = DateTime.TryParse((txtDobMon.Text + "/" + txtDobDate.Text +
"/" + txtDobYear.Text), out DoB);
            if(!isDate) { MessageBox.Show("Date of Birth is invalid", "Invalid Date of
Birth", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }

            //Declarations
            FName = txtFName.Text; LName = txtLName.Text; Guard = txtGuardian.Text; NIC
= txtNIC.Text; Work = txtWork.Text; Addr = txtAddr.Text; TP = txtTP.Text; Email =
txtEmail.Text; Status = "Valid";
            DateJ = DateTime.Today.Date; Fine = 0;
            if (cboxType.Text == "Child Membership") MType = "Child"; else if
(cboxType.Text == "Adult Membership") MType = "Adult";

            //Check age and type
            DateTime now = DateTime.Today;
            int age = now.Year - DoB.Year; if (DoB > now.AddYears(-age)) age--;
```

```
            //Disclaim age and type
            if (age < set.AgeMin && MType == "Adult") { MessageBox.Show("This person is
not old enough for an adult membership.", "Age not enough", MessageBoxButtons.OK,
MessageBoxIcon.Error); goto End; }
            if (age >= set.AgeMin && MType == "Child") { MessageBox.Show("This person
is old enough for an adult membership. Therefore, child membership cannot be created",
"Invalid Age", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }

            //Check for previous NIC, address and Blacklist
            string sqlDupl = string.Format("SELECT MemberID, FName, LName, MType,
MStatus, NIC, Address, MWork, Email, TP FROM Member WHERE NIC = '{0}' OR Address =
'{1}' OR MWork = '{2}' OR Email = '{3}' OR TP = '{4}'", NIC, Addr, Work, Email, TP);
OleDbCommand cmdDupl = new OleDbCommand(sqlDupl, db.con);
            if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
OleDbDataReader drDupl = cmdDupl.ExecuteReader(); drDupl.Read();

            if (drDupl.HasRows && Email != "")
            {
                string tMemID = drDupl["MemberID"].ToString(), tMType =
drDupl["MType"].ToString(), tFName = drDupl["FName"].ToString(), tLName =
drDupl["LName"].ToString(), tNIC = drDupl["NIC"].ToString(), tStatus =
drDupl["MStatus"].ToString(), tAddr = drDupl["Address"].ToString(), tMWork =
drDupl["MWork"].ToString(), tEmail = drDupl["Email"].ToString(), tTP =
drDupl["TP"].ToString();


                if (tStatus == "Blocked" && (tNIC == NIC || tEmail == Email) && MType
== tMType) { MessageBox.Show(string.Format("A person with such personal credentials,
MemberID '{0}' and in the name name: '{1} {2}' is blocked and marked in Blacklist,
maybe due to book theft or any other illegal activity. \r\n\r\nPlease take nessasary
actions.", tMemID, tFName, tLName), "Person in Blacklist", MessageBoxButtons.OK,
MessageBoxIcon.Warning); goto End; }
                else if (tStatus == "Valid" && (tNIC == NIC || tEmail == Email) &&
MType == tMType && MType == "Adult") { MessageBox.Show(string.Format("This person is
already a valid member in the library with the name: '{0} {1}'\r\n\r\nIf not, please
change the personal credentials, such as Email or NIC number anew", tFName, tLName),
"Valid member", MessageBoxButtons.OK, MessageBoxIcon.Warning); goto End; }
                else if (tStatus == "Expired" && (tNIC == NIC || tEmail == Email) &&
MType == tMType && MType == "Adult") { MessageBox.Show(string.Format("This person holds
an expired membership with MemberID '{0}' in the library with the name: '{1}
{2}'\r\n\r\nIf not, please change the personal credentials, such as Email or NIC number
anew", tMemID, tFName, tLName), "Expired member", MessageBoxButtons.OK,
MessageBoxIcon.Warning); goto End; }
                else if (tStatus == "Blocked" && (tAddr == Addr || tMWork == Work ||
tTP == TP)) { MessageBox.Show(string.Format("A person with these home address, work
address or telephone number with MemberID '{0}' and in the name: '{1} {2}' is blocked
and marked in Blacklist, maybe due to book theft or any other illegal activity.
\r\n\r\nYou may inquire about the blocked member from this person. This person will be
added as a Member", tMemID, tFName, tLName), "Person in Blacklist",
MessageBoxButtons.OK, MessageBoxIcon.Information); }

            }

            // Declaring MemID
```

```
            {
                string MaxID;

                // Getting the MaxID from DB
                string sqlMaxID = "SELECT MAX(MemberID) as MaxID, COUNT(MemberID) as
CountID FROM Member"; OleDbCommand cmdMaxID = new OleDbCommand(sqlMaxID, db.con);
                if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
OleDbDataReader drMaxID = cmdMaxID.ExecuteReader(); drMaxID.Read();

                if (drMaxID["CountID"].ToString() != "0")
                {
                    MaxID = drMaxID["MaxID"].ToString();
                    Methods meth = new Methods();
                    MemID = meth.NewID(MaxID);
                }
                else MemID = "M0001";

            }
            if (MType == "Child") MTi = 1; else MTi = 0;
            // Declarations End

            //Request paying
            fee = set.NewC[MTi];
            DialogResult receive = MessageBox.Show(string.Format("Rs. {0}/- should be
received for creation of a new {1} membership. Receive the amount and continue.", fee,
MType), "Receive Fees", MessageBoxButtons.OKCancel, MessageBoxIcon.Information);
            if (receive == DialogResult.Cancel) goto End;

            // Adding Transaction details

            string sqlTrans = string.Format("INSERT INTO Cash (TDate, Amount, TDetail,
Event) VALUES('{0}', {1}, '{2}', 'NewMem')", DateTime.Now.ToString(), fee, "MemberID =
"+MemID);
            OleDbCommand cmdTrans = new OleDbCommand(sqlTrans, db.con);
            cmdTrans.ExecuteNonQuery();
            // Transactions added.

            // Add all to Relation 'Member' in DB.

            string sqlNew = string.Format("INSERT INTO Member (MemberID, FName, LName,
MType, MStatus, DateJoined, Email, DateofBirth, TP, Guardian, NIC, Address, MWork)
VALUES ('{0}', '{1}', '{2}', '{3}', '{4}', '{5}', '{6}', '{7}', '{8}', '{9}', '{10}',
'{11}', '{12}')", MemID, FName, LName, MType, Status, DateJ.ToShortDateString(), Email,
DoB.ToShortDateString(), TP, Guard, NIC, Addr, Work);
            OleDbCommand cmdNew = new OleDbCommand(sqlNew, db.con);
cmdNew.ExecuteNonQuery();
            MessageBox.Show(string.Format("New Member '{0}' has been added successfully
and transactions have been recorded.", MemID), "Adding Successful",
MessageBoxButtons.OK, MessageBoxIcon.Information); // Message

            // Reset Form
            txtFName.Text = txtLName.Text = cboxType.Text = txtGuardian.Text =
txtDobDate.Text = txtDobMon.Text = txtDobYear.Text = txtNIC.Text = txtAddr.Text =
txtWork.Text = txtTP.Text = txtEmail.Text = "";
```

```
        End: if(db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }


        private void cboxType_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (cboxType.SelectedIndex == 0) { txtGuardian.Visible = label9.Visible =
true; lblLLNIC.Text = "Guardian's NIC"; }
            else if (cboxType.SelectedIndex == 1) { txtGuardian.Visible =
label9.Visible = false; lblLLNIC.Text = "Member's NIC"; }
        }

        private void btnReset_Click(object sender, EventArgs e)
        {
            this.Close(); frmMemNew f = new frmMemNew(); f.Show();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }


    }
}
```

Member Details / Update / Remove / Add-Remove from Blacklist Form (frmMemDetails)

```
namespace Alexandria
{
    public partial class frmMemDetails : Form
    {
        // Common Variable Declaration.

        string MemID, NewMemID, FName, LName, FullDate, MType, Guard, NIC, Addr, Work,
TP, Email, Status;
        int TimesBorrow, Renewed, MTi; DateTime DateJ, DateR, DoB; double Stars,
fullStars, fee, Fine;


        public frmMemDetails()
        {
            InitializeComponent();
        }

        private void btnGet_Click(object sender, EventArgs e)
        {
            //Remove single & double quotes which cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox ||
c is RichTextBox) { c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"",
""); } }

            //Declare MemID based on Input Type
            if (cvar.InsertID && txtMemID.Text == "") { MemID = cvar.MemID;
txtMemID.Text = MemID; } // If input from outside.
            else if (txtMemID.Text == "") goto End; // If Form opened anew.
            else MemID = txtMemID.Text; // If Value entered in textbox.

            //Access Database for Details
            string sqlMemD = string.Format("SELECT * FROM Member WHERE MemberID =
'{0}'", MemID);
            OleDbCommand cmdMemD = new OleDbCommand(sqlMemD, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drMemD = cmdMemD.ExecuteReader();

            if (drMemD.Read())
            {
                txtMemID.Enabled = false; MakeWide(); //Make the form wider.

                txtFName.Text = drMemD["FName"].ToString();
                txtLName.Text = drMemD["LName"].ToString();
                txtAddr.Text = drMemD["Address"].ToString();
                txtEmail.Text = drMemD["Email"].ToString();
                txtGuardian.Text = drMemD["Guardian"].ToString();
                txtNIC.Text = drMemD["NIC"].ToString();
                txtTP.Text = drMemD["TP"].ToString();
                txtWork.Text = drMemD["MWork"].ToString();
                lblStatus.Text = drMemD["MStatus"].ToString();
```

```
                Fine = double.Parse(drMemD["Fine"].ToString());
                lblFine.Text = Fine.ToString() + "/-";
                lblRenew.Text = drMemD["Renewed"].ToString();
                Popul p = new Popul();
                TimesBorrow = int.Parse(drMemD["TimesBorrow"].ToString());

                if (Fine == 0) btnFine.Enabled = false; else btnFine.Enabled = true;


                DateTime now = DateTime.Today;
                DateTime Dob = DateTime.Parse(drMemD["DateOfBirth"].ToString());
                int age = now.Year - Dob.Year; if (Dob > now.AddYears(-age)) age--;
                lblAge.Text = age.ToString();

                MType = drMemD["MType"].ToString();
                if (MType == "Child") { cboxType.SelectedIndex = 0; lblLLg.Visible =
txtGuardian.Visible = true; lblLLNic.Text = "Guardian's NIC"; }
                else if (MType == "Adult") { cboxType.SelectedIndex = 1; lblLLg.Visible
= txtGuardian.Visible = false; lblLLNic.Text = "NIC Number"; cboxType.Enabled = false;
}

                if (lblStatus.Text == "Blocked") btnBlack.Text = "Remove from
Blacklist";
                else btnBlack.Text = "Add to Blacklist";

                //Split & Use Date of Birth
                string Fulldob = drMemD["DateOfBirth"].ToString();
                string[] DoBSpace = Fulldob.Split(' '); // Split Date from Time
                string[] DoBSlash = DoBSpace[0].Split('/'); // SpilitDate components

                txtDobMon.Text = DoBSlash[0]; txtDobDate.Text = DoBSlash[1];
txtDobYear.Text = DoBSlash[2];

                //Split & Use Date Joined
                DateJ = DateTime.Parse(drMemD["DateJoined"].ToString());
                FullDate = DateJ.ToString();
                string[] DateSpace = FullDate.Split(' '); // Split Date from Time
                string[] DateSlash = DateSpace[0].Split('/'); // SpilitDate components

                lblDate.Text = DateSlash[1] + "-" + DateSlash[0] + "-" + DateSlash[2];

                // To Retrive BookIDs

                string sqlBookID = string.Format("SELECT BookID FROM LendStatus WHERE
MemberID = '{0}'", MemID);
                OleDbCommand cmdBookID = new OleDbCommand(sqlBookID, db.con);
OleDbDataReader drBookID = cmdBookID.ExecuteReader();

                while (drBookID.Read()) // For Each bookID
                {
                    string BookID = drBookID["BookID"].ToString();

                    string sqlBook = string.Format("SELECT TitleID FROM Book WHERE
BookID = '{0}'", BookID); //Get TitleID from BookID
```

```
                    OleDbCommand cmdBook = new OleDbCommand(sqlBook, db.con);
OleDbDataReader drBook = cmdBook.ExecuteReader();

                    if (drBook.Read())
                    {
                        string TitleID = drBook["TitleID"].ToString();

                        string sqlTitle = string.Format("SELECT BTitle, Author, Genre
FROM Title WHERE TitleID = '{0}'", TitleID); // Get Title info from TitleID
                        OleDbCommand cmdTitle = new OleDbCommand(sqlTitle, db.con);
OleDbDataReader drTitle = cmdTitle.ExecuteReader();

                        if (drTitle.Read()) { lboxBookID.Items.Add(BookID);
lboxTitle.Items.Add(drTitle["BTitle"].ToString());
lboxAuthor.Items.Add(drTitle["Author"].ToString());
lboxGenre.Items.Add(drTitle["Genre"].ToString()); }
                    }
                }

                //To Get last Check In

                string sqlCheckIn = string.Format("SELECT Max(CDate) as MaxCheckIn FROM
CheckInOut WHERE MemberID = '{0}' AND EVENT = 'In' GROUP BY MemberID, Event", MemID);
                OleDbCommand cmdCheckIn = new OleDbCommand(sqlCheckIn, db.con);
OleDbDataReader drCheckIn = cmdCheckIn.ExecuteReader();

                if (drCheckIn.Read() & drCheckIn.HasRows) { DateTime CheckIn =
DateTime.Parse(drCheckIn["MaxCheckIn"].ToString()); lblCheckIn.Text =
CheckIn.ToString("dd-MM-yyyy"); }



                //Calculate Recent Star Points Via method & Display
                Popul star = new Popul();
                fullStars = star.MemCalc(MemID, false);
                Stars = double.Parse(Math.Round(decimal.Parse(fullStars.ToString()),
2).ToString());
                lblStars.Text = Stars.ToString();

                btnGet.Enabled = false;

            }
            else MessageBox.Show("MemberID does not match with any of the members. Try
again", "Invalid MemberID", MessageBoxButtons.OK, MessageBoxIcon.Error);

            cvar.MemID = ""; cvar.InsertID = false;
            if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();


        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
```

```
            //Verify
            DialogResult one = MessageBox.Show(string.Format("Are you sure you want to
update these details to the member '{0}'?", MemID), "Are you sure?",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if(one ==DialogResult.No) goto End;

            // Disclaimers

            //Remove single & double quotes which cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox ||
c is RichTextBox) { c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"",
""); } }

            if (txtFName.Text == "" || txtLName.Text == "" || txtAddr.Text == "" ||
txtNIC.Text == "" || txtWork.Text == "" || txtTP.Text == "" || txtDobDate.Text == "" ||
txtDobMon.Text == "" || txtDobYear.Text == "" || txtDobDate.Text == "Date" ||
txtDobMon.Text == "Month" || txtDobYear.Text == "Month")
            { MessageBox.Show("One or many of the required fields are left blank.
Please fill them and try again", "Field(s) left blank", MessageBoxButtons.OK,
MessageBoxIcon.Error); goto End; }
            if (txtNIC.Text.Length != 10) { MessageBox.Show("NIC number invalid",
"Invalid NIC", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }
            if ((MType == "Child" && cboxType.Text == "Adult Member") || (MType ==
"Adult" && cboxType.Text == "Child Member")) { MessageBox.Show("Member type cannot be
changed", "Can't change member type", MessageBoxButtons.OK, MessageBoxIcon.Error); goto
End; }
            if (cboxType.Text == "Child Member" && txtGuardian.Text == "") {
MessageBox.Show("A guardian must be specified for a child member.", "Specify Guardian",
MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }


            // Disclaimers End

            //Declarations
            FName = txtFName.Text; LName = txtLName.Text; Guard = txtGuardian.Text; NIC
= txtNIC.Text; Work = txtWork.Text; Addr = txtAddr.Text; TP = txtTP.Text; Email =
txtEmail.Text; Status = "Valid";
            Fine = 0; DoB = DateTime.Parse(txtDobMon.Text + "/" + txtDobDate.Text + "/"
+ txtDobYear.Text);

            // Declarations End

            // Update all to Relation 'Member' in DB.

            string sqlUpd = string.Format("UPDATE Member SET FName ='{0}', LName
='{1}', Email ='{2}', DateofBirth ='{3}', TP ='{4}', Guardian ='{5}', NIC ='{6}',
Address ='{7}', MWork ='{8}' WHERE MemberID = '{9}'", FName, LName, Email,
DoB.ToShortDateString(), TP, Guard, NIC, Addr, Work, MemID);
            OleDbCommand cmdUpd = new OleDbCommand(sqlUpd, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open(); cmdUpd.ExecuteNonQuery();
            MessageBox.Show(string.Format("Details of Member({0}) have been updated
successfully", MemID), "Update Successful", MessageBoxButtons.OK,
MessageBoxIcon.Information); // Message

            // Reset Form
```

```
            Reset();

            End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void btnRenew_Click(object sender, EventArgs e)
        {
            //Remove single & double quotes which cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox ||
c is RichTextBox) { c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"",
""); } }


            // Check if any books borrowed
            DialogResult two = MessageBox.Show("Are you sure you want to renew this
membership?", "Are you sure?", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (two == DialogResult.No) goto End;
            if (lboxBookID.Items.Count != 0 || lblFine.Text != "0/-") {
MessageBox.Show("The books borrowed by the member must be returned and all fines should
be paid before attempting a membership renewal", "Return books and pay fines",
MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }
            if (txtFName.Text == "" || txtLName.Text == "" || cboxType.Text == "" ||
txtAddr.Text == "" || txtNIC.Text == "" || txtWork.Text == "" || txtTP.Text == "" ||
txtDobDate.Text == "" || txtDobMon.Text == "" || txtDobYear.Text == "" ||
txtDobDate.Text == "Date" || txtDobMon.Text == "Month" || txtDobYear.Text == "Month")
            { MessageBox.Show("One or many of the required fields are left blank.
Please fill them and try again", "Field(s) left blank", MessageBoxButtons.OK,
MessageBoxIcon.Error); goto End; }
            if (txtNIC.Text.Length != 10) { MessageBox.Show("NIC number invalid",
"Invalid NIC", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }
            if (cboxType.Text == "Child Membership" && txtGuardian.Text == "") {
MessageBox.Show("A guardian should be specified for a Child Member", "Specify
Guardian", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }
            bool isDate = DateTime.TryParse((txtDobMon.Text + "/" + txtDobDate.Text +
"/" + txtDobYear.Text), out DoB);
            if (!isDate) { MessageBox.Show("Date of Birth is invalid", "Invalid Date of
Birth", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }


            if (lblStatus.Text == "Valid")
            {
                DialogResult one = MessageBox.Show("The membership is not yet expired.
Beginning of new membership will be calculated from the end of present membership. Do
you want to continue?", "Membership still valid", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);
                if (one == DialogResult.No) goto End;
            }
            DateR = DateTime.Now;

            if (lblStatus.Text == "Blocked") { MessageBox.Show("This member is blocked
and marked in Blacklist, maybe due to book theft or any other illegal activity. This
membership cannot be renewed. \r\n\r\nPlease take nessasary actions.", "Member in
Blacklist", MessageBoxButtons.OK, MessageBoxIcon.Warning); goto End; }
            // Disclaimers End
```

```
            //Declarations
            FName = txtFName.Text; LName = txtLName.Text; Guard = txtGuardian.Text; NIC
= txtNIC.Text; Work = txtWork.Text; Addr = txtAddr.Text; TP = txtTP.Text; Email =
txtEmail.Text; Status = "Valid";
            Fine = 0; DoB = DateTime.Parse(txtDobMon.Text + "/" + txtDobDate.Text + "/"
+ txtDobYear.Text); Popul p = new Popul(); Stars = p.MemCalc(MemID, false);
            if (cboxType.Text == "Child Member") MType = "Child"; else if
(cboxType.Text == "Adult Member") MType = "Adult";

            if (lblStatus.Text == "Valid") DateR = DateJ.AddMonths(set.Expire[MTi]);

            //Check age and type
            DateTime now = DateTime.Today;
            int age = now.Year - DoB.Year; if (DoB > now.AddYears(-age)) age--;

            if (age >= set.AgeMin && MType == "Child")
            {
                DialogResult r = MessageBox.Show("This Child member is now old enough
to be an Adult Member. Continue creating an adult membership?", "Adult Member",
MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
                if (r == DialogResult.Yes) { MType = "Adult"; cboxType.Text = "Adult
Member"; } else goto End;
            }
            if (MType == "Child") MTi = 0; else MTi = 1;

            Renewed = int.Parse(lblRenew.Text) + 1;

            // Declaring MemID
            {
                string MaxID;

                // Getting the MaxID from DB
                string sqlMaxID = "SELECT MAX(MemberID) as MaxID FROM Member";
OleDbCommand cmdMaxID = new OleDbCommand(sqlMaxID, db.con);
                if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
OleDbDataReader drMaxID = cmdMaxID.ExecuteReader();
                if (drMaxID.Read()) MaxID = drMaxID["MaxID"].ToString().Substring(1,
4); else goto End;

                // Increment MaxID to PresentID
                int iID = int.Parse(MaxID) + 1;
                string sID = iID.ToString();

                //Loop to add enough zeros for the 5-character format (M0001)

                for (int i = sID.Length; i != 4; i++) { sID = "0" + sID; }
                NewMemID = "M" + sID;
            }
            // Declarations End

            //Request paying

            fee = set.NewC[MTi];
```

```
            DialogResult receive = MessageBox.Show(string.Format("Rs. {0}/- should be
received for renewal of a {1} membership. Receive the amount and continue.", fee,
MType), "Receive Fees", MessageBoxButtons.OKCancel, MessageBoxIcon.Information);
            if (receive == DialogResult.Cancel) goto End;

            // Adding Transaction details

            string sqlTrans = string.Format("INSERT INTO Cash (TDate, Amount, TDetail,
Event) VALUES('{0}', {1}, '{2}', 'RenewMem')", DateTime.Now.ToString(), fee, "MemberID
= " + MemID);
            OleDbCommand cmdTrans = new OleDbCommand(sqlTrans, db.con);
            cmdTrans.ExecuteNonQuery();
            // Transactions added.

            // Add all to Relation 'Member' in DB.

            string sqlNew = string.Format("INSERT INTO Member (MemberID, FName, LName,
MType, MStatus, DateJoined, Email, DateofBirth, TP, Guardian, NIC, Address, MWork,
Renewed, TimesBorrow, Stars) VALUES ('{0}', '{1}', '{2}', '{3}', '{4}', '{5}', '{6}',
'{7}', '{8}', '{9}', '{10}', '{11}', '{12}', {13}, {14}, {15})", NewMemID, FName,
LName, MType, Status, DateR.ToShortDateString(), Email, DoB.ToShortDateString(), TP,
Guard, NIC, Addr, Work, Renewed, TimesBorrow, Stars);
            OleDbCommand cmdNew = new OleDbCommand(sqlNew, db.con);
cmdNew.ExecuteNonQuery();

            //Delete Old Member
            string sqlDel = string.Format("DELETE FROM Member WHERE MemberID = '{0}'",
MemID);
            OleDbCommand cmdDel = new OleDbCommand(sqlDel, db.con);
cmdDel.ExecuteNonQuery();

            MessageBox.Show(string.Format("Membership of Member({0}) has been
successfully renewed as ID '{1}'", MemID, NewMemID), "Renewal Successful",
MessageBoxButtons.OK, MessageBoxIcon.Information); // Message

            // Reset Form
            Reset();



        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void btnBlack_Click(object sender, EventArgs e)
        {
            //Remove single & double quotes which cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox ||
c is RichTextBox) { c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"",
""); } }
            if (btnBlack.Text == "Add to Blacklist")
            {
                DialogResult r = MessageBox.Show("Do you really want to block this
member and add to blacklist? \r\n\r\nAn alert will be raised when this member uses the
library again.", "Are you sure", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
```

```
                if (r == DialogResult.Yes)
                {
                    string sqlBlack = string.Format("UPDATE Member SET MStatus =
'Blocked' WHERE MemberID = '{0}'", MemID);
                    OleDbCommand cmdBlack = new OleDbCommand(sqlBlack, db.con);
                    if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                    cmdBlack.ExecuteNonQuery();

                    //Remove taken books if stolen
                    Methods m = new Methods();
                    if(lboxBookID.Items.Count !=0)
                    {
                    DialogResult b = MessageBox.Show("This member has not returned some
books. Do you want to mark them stolen and remove from library?", "Remove books",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);
                    if (b == DialogResult.Yes)
                    {
                        //Pass to method in loop
                        for (int i = 0; i < lboxBookID.Items.Count; i++)
                        {
                            string BookID = lboxBookID.Items[i].ToString();
                            m.RemBook(BookID, string.Format("Stolen by MemberID = {0}",
MemID), false);
                        }
                    }
                    }


                    MessageBox.Show("Member has been added to blacklist successfully",
"Successful", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    Reset();
                }

            }
            else if (btnBlack.Text == "Remove from Blacklist")
            {
                DialogResult r = MessageBox.Show("Do you really want to unblock this
member and remove from blacklist?", "Are you sure", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);
                if (r == DialogResult.Yes)
                {
                    if (MType == "Child") MTi = 1; else MTi = 0;
                    DateTime DJ = DateTime.Parse(FullDate).AddMonths(set.Expire[MTi]);

                    if (DJ.CompareTo(DateTime.Today) < 0) { MessageBox.Show("This
membership has expired and will be marked as expired", "Expired Membership"); Status =
lblStatus.Text = "Expired"; }
                    else Status = lblStatus.Text = "Valid";

                    string sqlBlack = string.Format("UPDATE Member SET MStatus = '{0}'
WHERE MemberID = '{1}'", Status, MemID);
                    OleDbCommand cmdBlack = new OleDbCommand(sqlBlack, db.con);
                    if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                    cmdBlack.ExecuteNonQuery();
```

```
                    MessageBox.Show("Member has been removed from blacklist
successfully", "Successful", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    Reset();
                }

            }
            else { }

        if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void btnFine_Click(object sender, EventArgs e)
        {
            Methods m = new Methods();
            m.PayAccFines(MemID, Fine);
            lblFine.Text = "0/-"; btnFine.Enabled = false;
        }

        private void lboxBookID_DoubleClick(object sender, EventArgs e)
        {
            frmBookDetail f = new frmBookDetail();
            f.txtBookID.Text = lboxBookID.Items[lboxBookID.SelectedIndex].ToString();
            f.btnGetBook_Click(sender, e);
            f.ShowDialog();
        }

        private void lboxTitle_DoubleClick(object sender, EventArgs e)
        {
            frmBookDetail f = new frmBookDetail();
            f.txtBookID.Text = lboxBookID.Items[lboxTitle.SelectedIndex].ToString();
            f.btnGetBook_Click(sender, e);
            f.ShowDialog();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Reset();
        }

        private void btnRemove_Click(object sender, EventArgs e)
        {
            //Remove single & double quotes which cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox ||
c is RichTextBox) { c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"",
""); } }


            DialogResult r = MessageBox.Show("Do you really want to remove this member?
\r\n\r\nThis action cannot be undone.", "Are you sure", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);
            if (r == DialogResult.Yes)
            {
                string Isrembook = "";

                //Check for lent books
```

```
                if (lboxBookID.Items.Count != 0)
                {
                    DialogResult b = MessageBox.Show("This member has not returned few
books. Do you want to mark them as returned?", "Return Books", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);
                    if (b == DialogResult.Yes)
                    {
                        Methods m = new Methods();

                        //Pass to method in loop
                        for (int i = 0; i < lboxBookID.Items.Count; i++)
                        {
                            string BookID = lboxBookID.Items[i].ToString();
                            string sqlRemBook = string.Format("DELETE FROM LendStatus
WHERE BookID = '{0}'", BookID);
                            OleDbCommand cmdRemBook = new OleDbCommand(sqlRemBook,
db.con);
                            if (db.con.State.Equals(ConnectionState.Closed))
db.con.Open();
                            cmdRemBook.ExecuteNonQuery();
                        }
                        Isrembook = " and all the books borrowed have been marked as
returned.";
                    }
                    else goto End;
                }
                    string sqlRem = string.Format("DELETE FROM Member WHERE MemberID =
'{0}'", MemID);

                    OleDbCommand cmdRem = new OleDbCommand(sqlRem, db.con);
                    if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                    cmdRem.ExecuteNonQuery();

                MessageBox.Show(string.Format("Member has been removed successfully",
Isrembook), "Removal Successful", MessageBoxButtons.OK, MessageBoxIcon.Information);
                Reset();
            }
        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void Reset()
        {
            this.Close();
            frmMemDetails f = new frmMemDetails();
            f.Show();
        }

        private void frmMemDetails_Load(object sender, EventArgs e)
        {
            if (txtMemID.Text == "") this.Width = 218;
            else btnGet_Click(sender, e);
        }

        private void MakeWide()
        { this.Width = 1055; }
```

```
        private void lboxBookID_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (lboxBookID.SelectedItems.Count != 0)
            {
                frmBookDetail b = new frmBookDetail(); b.txtBookID.Text =
lboxBookID.Items[lboxBookID.SelectedIndex].ToString(); b.btnGetBook_Click(sender, e);
b.ShowDialog();
            }
        }

    }
}
```

Lend Books Form (frmIssues)

```
namespace Alexandria
{
    public partial class frmIssues : Form
    {
        string MemID, MStatus, MType, BookID, BType, accFine; int BooksAllowed, MTi;

        public frmIssues()
        {
            InitializeComponent();
        }

        private void btnSet_Click(object sender, EventArgs e)
        {
            //Disclaimer
            if (txtMemID.Text == "") goto End;
            //Remove single & double quoteswhich cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }


            MemID = txtMemID.Text;

            //Get info about Member
            string sqlMem = string.Format("SELECT MType, Fine, MStatus FROM Member
WHERE MemberID = '{0}'", MemID);
            OleDbCommand cmdMem = new OleDbCommand(sqlMem, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drMem = cmdMem.ExecuteReader();

            if (drMem.Read()) //If memeber is present
            {
                MStatus = drMem["MStatus"].ToString(); accFine =
drMem["Fine"].ToString(); MType = drMem["MType"].ToString(); int NoBooks;

                //Disclaimers and Percautions

                //Check Membership Status
                if (MStatus == "Expired") { MessageBox.Show("Books cannot be lent to
this Member since the membership has been expired.", "Membership Expired",
MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }
                else if (MStatus == "Blocked") { MessageBox.Show("This Membership has
been Blocked, maybe due to behavior or security reasons. Please take nessasary
actions.", "MEMBER BLOCKED", MessageBoxButtons.OK, MessageBoxIcon.Warning); goto End; }

                // ALLOW TO PAY FINES FROM HERE ITSELF.
                if (accFine != "0") MessageBox.Show(string.Format("This member has
{0}/- balance of unpaid fines in his account. These fines may be recieved now and/or
new books can be recieved.", accFine), "Member has Fines", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                if (MType == "Adult") MTi = 0; else if (MType == "Child") MTi = 1;

                //Count Number of books he borrowed
```

```
                string sqlNoBooks = string.Format("SELECT COUNT(BookID) as NoBooks FROM
LendStatus WHERE MemberID = '{0}'", MemID);
                OleDbCommand cmdNoBooks = new OleDbCommand(sqlNoBooks, db.con);
OleDbDataReader drNoBooks = cmdNoBooks.ExecuteReader();
                if (drNoBooks.Read()) NoBooks =
int.Parse(drNoBooks["NoBooks"].ToString()); else NoBooks = 0;

                BooksAllowed = set.NoBooks[MTi] - NoBooks; // Set the no. of books he
can borrow not.

                //Check Maximum books limit exceepded
                if (BooksAllowed <= 0) { MessageBox.Show(string.Format("This Member has
borrowed {0}, the maximum number of books allowed for this type of membership. No more
books can be Lent to this member.", set.NoBooks[MTi]), "Lending limit exceeded",
MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }

                //If all right show other controls
                        //Now Height 109
                btnSet.Enabled = txtMemID.Enabled = false; btnIssue.Enabled =
txtBookID.Enabled = true; btnIssue.Visible = txtBookID.Visible = lblBookID.Visible =
true;

            }
            else MessageBox.Show(string.Format("No Membership with Member Id '{0}'
exists.", MemID), "Invalid Member ID", MessageBoxButtons.OK, MessageBoxIcon.Error);


        End: if(db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }


        private void btnIssue_Click(object sender, EventArgs e)
        {
            //Disclaimer
            if (BooksAllowed <= 0) { MessageBox.Show("This member has borrowed maximum
number of books.", "Lending Limit Exceeded", MessageBoxButtons.OK,
MessageBoxIcon.Error); goto End; }
            if (txtBookID.Text == "") goto End;
            //Remove single & double quoteswhich cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }
            BookID = txtBookID.Text;

            //Check validity of BookID
            string sqlBookIDV = string.Format("SELECT TitleID FROM Book WHERE BookID =
'{0}'", BookID);
            OleDbCommand cmdBookIDV = new OleDbCommand(sqlBookIDV, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drBookIDV = cmdBookIDV.ExecuteReader(); drBookIDV.Read();

            if (drBookIDV.HasRows) // If valid BookID
            {
                //Check Book Type
```

```
                string sqlBType = string.Format("SELECT BType FROM Title WHERE TitleID
= '{0}'", drBookIDV["TitleID"]);
                OleDbCommand cmdBType = new OleDbCommand(sqlBType, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drBType = cmdBType.ExecuteReader(); drBType.Read();
BType = drBType["BType"].ToString();

                //Disclaim for invalid book Types
                if (BType == "Ref") {MessageBox.Show("This book is for reference only,
not for borrowal", "Reference book", MessageBoxButtons.OK, MessageBoxIcon.Error); goto
End;}
                else if (BType == "ALend" && MTi == 1) { MessageBox.Show("Adult books
cannot be borrowed using child memberships.", "Adult Book", MessageBoxButtons.OK,
MessageBoxIcon.Error); goto End; }


                // Check Book Availability
                string sqlIsLentBook = string.Format("SELECT COUNT(BookID) as
IsLentBook FROM LendStatus WHERE BookID = '{0}'", BookID);
                OleDbCommand cmdIsLentBook = new OleDbCommand(sqlIsLentBook, db.con);
if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drIsLentBook = cmdIsLentBook.ExecuteReader();

                if (drIsLentBook.Read() && drIsLentBook["IsLentBook"].ToString() !=
"0") { MessageBox.Show("This book has already been lent and not yet received. Receive
the book before lending it again.", "Book Unavailable", MessageBoxButtons.OK,
MessageBoxIcon.Error); goto End; }
                //Book Available to lend

                //LEND THE BOOK. Log Lending to Database
                {
                    //Log to LendStatus
                    string sqlLendBook = string.Format("INSERT INTO LendStatus (BookID,
MemberID, LendDate, Extend) VALUES ('{0}', '{1}', '{2}', 0)", BookID, MemID,
DateTime.Now.ToString());
                    OleDbCommand cmdLendBook = new OleDbCommand(sqlLendBook, db.con);
if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                    cmdLendBook.ExecuteNonQuery();

                    //Log to Member Timesborrow
                    string sqlMemTB = string.Format("UPDATE Member SET TimesBorrow =
TimesBorrow + 1 WHERE MemberID = '{0}'", MemID);
                    OleDbCommand cmdMemTB = new OleDbCommand(sqlMemTB, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                    cmdMemTB.ExecuteNonQuery();

                    //Log to Title Timesborrow
                    string TitleID = drBookIDV["TitleID"].ToString();
                    string sqlTitTB = string.Format("UPDATE Title SET TimesBorrowed =
TimesBorrowed + 1 WHERE TitleID = '{0}'", TitleID);
                    OleDbCommand cmdTitTB = new OleDbCommand(sqlTitTB, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                    cmdTitTB.ExecuteNonQuery();

                }
```

```
                //Lending Successful
                MessageBox.Show("Book Lending has been recorded successfully", "Lending
Successful", MessageBoxButtons.OK, MessageBoxIcon.Information);
                BooksAllowed = BooksAllowed - 1; txtBookID.Clear();

                //Block lending more books, if limits exceeds
                if (BooksAllowed == 0) { txtBookID.Enabled = btnIssue.Enabled = false;
}

            }
            else MessageBox.Show(string.Format("No book with BookID '{0}' exists.",
BookID), "Invalid BookID", MessageBoxButtons.OK, MessageBoxIcon.Error);



        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void btnReset_Click(object sender, EventArgs e)
        {
            this.Close(); frmIssues f = new frmIssues(); f.Show();
        }
    }
}
```

Receive / Extend / Report Loss Form (frmReturnExt)

```
namespace Alexandria
{
    public partial class frmReturnExt : Form
    {
        string BookID = "", MemID; double Price, AmountOnly, Amount, Fine, accFine;

        public frmReturnExt()
        {
            InitializeComponent();
        }

        private void btnSet_Click(object sender, EventArgs e)
        {

            if (txtBookID.Text == "") goto End;
            //Remove single & double quoteswhich cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }

            BookID = txtBookID.Text;


            // Check Book Availability in LendStatus table
            string sqlIsLentBook = string.Format("SELECT COUNT(BookID) as IsLentBook
FROM LendStatus WHERE BookID = '{0}'", BookID);
            OleDbCommand cmdIsLentBook = new OleDbCommand(sqlIsLentBook, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drIsLentBook = cmdIsLentBook.ExecuteReader();

            //If book is not present in Lent table
            drIsLentBook.Read();
            if (drIsLentBook["IsLentBook"].ToString() == "0")
            {
                // Check the ID in BookID table
                string sqlIsValid = string.Format("SELECT COUNT(BookID) as Books FROM
Book WHERE BookID = '{0}'", BookID);
                OleDbCommand cmdIsValid = new OleDbCommand(sqlIsValid, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drIsValid = cmdIsValid.ExecuteReader();


                drIsValid.Read();
                if (drIsValid["Books"].ToString() == "0") // If BookID not present in
Book Table, say Invalid ID
                { MessageBox.Show("No book with given BookID exists. Please enter a
valid ID", "Invalid BookID", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }

                else // If BookID present in Book table (and not in Lend Table), then
book not lent.
```

```
                { MessageBox.Show("This book has not been lent or has been returned
already.", "Book not Lent.", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }
            }

            //If book is present in Lent table, continue.
            btnSet.Enabled = txtBookID.Enabled = false; btnExtend.Visible =
btnLoss.Visible = btnReset.Visible = btnReturn.Visible = true;

        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();

        }

        private void btnReturn_Click(object sender, EventArgs e)
        {

            Methods m = new Methods();

            // Decalre varibales and call method to check and calculate book fines

            bool isBFine, IsBlocked; int Days; DateTime iDate; int MTi;
            m.FindBookFine(BookID, out isBFine, out Fine, out Days, out iDate, out
MemID, out IsBlocked, out MTi);

            //Find Blocked Member
            if (IsBlocked) MessageBox.Show("This Membership has been Blocked, maybe due
to behavior or security reasons. Please take nessasary actions.", "MEMBER BLOCKED",
MessageBoxButtons.OK, MessageBoxIcon.Warning);

            //For member's fines in account
            accFine = 0; bool isFine = false;
            m.FindAccFines(MemID, ref isFine, ref accFine);

            //Tak actions for fines and non fines
            {
                if (isBFine) //If Fines [ideal date is passed]
                {

                    //Whether to pay today or later
                    DialogResult now = MessageBox.Show(string.Format("This Book should
have been returned {0} day(s) ago. Now, Rs {1}/- should be paid as fine. \r\n\r\nFine
amount can be received now or the amount will be added to Member's account and a
notification will be raised when he uses his Membership again. \r\n\r\nHave you
received the amount today?", Days, Fine), "Received the fine?",
MessageBoxButtons.YesNo, MessageBoxIcon.Information);

                    if (now == DialogResult.Yes) // If paying today.
                    {
                        //Add details to Cash table

                        string sqlPayRem = string.Format("INSERT INTO Cash(Event,
TDetail, TDate, Amount) VALUES ('FinePaid', '{0}', '{1}', '{2}')",
string.Format("MemberID = {0} | BookID = {1}", MemID, BookID), DateTime.Now.ToString(),
Fine);
                        OleDbCommand cmdPayRem = new OleDbCommand(sqlPayRem, db.con);
                        cmdPayRem.ExecuteNonQuery();
```

```
                        MessageBox.Show(string.Format("Transaction of fine amount of
Rs.{0}/- has beeen successfully recorded.", Fine), "Transaction Successful",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                    }
                    else // If to be paid later
                    {
                        //Add the fine amount to old account fines.

                        string sqlFineLater = string.Format("UPDATE Member SET Fine =
Fine + {0} WHERE MemberID = '{1}'", Fine, MemID);
                        OleDbCommand cmdFineLater = new OleDbCommand(sqlFineLater,
db.con);
                        cmdFineLater.ExecuteNonQuery();

                        MessageBox.Show(string.Format("Fine amount of Rs.{0}/- has
beeen successfully added to Member's account to be received later.", Fine), "Fines
successfully added", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                }

                else //If No fines, show message
                { MessageBox.Show("No fines will be charged for this return", "No
Fines!", MessageBoxButtons.OK, MessageBoxIcon.Information); }

            }
            // End taking actions on book fines


            //Check for Previous Account Fines Using Method FindAccFines
            {


                if (isFine) // If there's fine
                {
                    //Ask whether to pay them today.
                    DialogResult accNow = MessageBox.Show(string.Format("The Member {0}
has unpaid fines of Rs. {1} stored in his/her account. Would you like to receive them
now too? \r\n\r\nIf yes, receive them and continue.", MemID, accFine), "Receive the
unpaid fines?", MessageBoxButtons.YesNo, MessageBoxIcon.Information);
                    if (accNow == DialogResult.Yes) { m.PayAccFines(MemID, accFine); }
                }
            }

            //Remove record Book from LendStatus

            string sqlRemBook = string.Format("DELETE FROM LendStatus WHERE BookID =
'{0}'", BookID);
            OleDbCommand cmdRemBook = new OleDbCommand(sqlRemBook, db.con);
            cmdRemBook.ExecuteNonQuery();

            //SUCCESS
            MessageBox.Show("Book return has been recorded successfully", "Book Return
Successful", MessageBoxButtons.OK, MessageBoxIcon.Information);
```

```
            //Reset Form
            Reset();


            End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void btnExtend_Click(object sender, EventArgs e)
        {
            Methods m = new Methods();

            // Decalre varibales and call method to check and calculate book fines

            bool isBFine, IsBlocked; int Days; DateTime iDate; int MTi;
            m.FindBookFine(BookID, out isBFine, out Fine, out Days, out iDate, out
MemID, out IsBlocked, out MTi);


            //Get the Extension detail from LendStatus relation
            string sqlLDetail = string.Format("SELECT Extend FROM LendStatus WHERE
BookID = '{0}'", BookID);
            OleDbCommand cmdLDetail = new OleDbCommand(sqlLDetail, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drLDetail = cmdLDetail.ExecuteReader(); drLDetail.Read();

            //Alert Blocked Member
            if (IsBlocked) { MessageBox.Show("This Membership has been Blocked, maybe
due to behavior or security reasons. Please take nessasary actions.", "MEMBER BLOCKED",
MessageBoxButtons.OK, MessageBoxIcon.Warning); goto End; }

            //Disclaim for already Extended books.
            if (drLDetail["Extend"].ToString() == "-1") { MessageBox.Show("This issue
has already been extended once. It cannot be extended again.", "Issue already
extended", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }

            //Tak actions for fines and non fines
            {
                if (isBFine) //If Fines [ideal date is passed]
                    { MessageBox.Show(string.Format("This Book should've been returned
{0} day(s) ago. Issues that have passed the return date cannot be extended.", Days),
"Issue Cannot be Extended", MessageBoxButtons.OK, MessageBoxIcon.Error); }

                // If No Fines
                else
                {
                    //Modify Days: Days allowed per book
                    //Modify extended ideal Date: Date to be returned + days allowed
per book.
                    Days = set.Btime[MTi];
                    iDate = iDate.AddDays(Days);

                    string sqlExt = string.Format("UPDATE LendStatus SET Extend = -1,
LendDate = '{0}' WHERE BookID = '{1}'", iDate, BookID);
                    OleDbCommand cmdExt = new OleDbCommand(sqlExt, db.con);
```

```
                    cmdExt.ExecuteNonQuery();

                    MessageBox.Show(string.Format("This issue has been successfully
extended by {0} days, upto {1}.", Days, iDate.ToLongDateString()), "Extension
SUccessful", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }

            }
            //Reset Form
            Reset();

        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void btnLoss_Click(object sender, EventArgs e)
        {
            DialogResult sure = MessageBox.Show("Are you sure you want to mark this
book as lost?", "Are you sure?", MessageBoxButtons.YesNo, MessageBoxIcon.Information);
            if (sure == DialogResult.No) goto End;

            //Get TitleID from Book table
            string sqlBook = string.Format("SELECT TitleID FROM Book WHERE BookID =
'{0}'", BookID);
            OleDbCommand cmdBook = new OleDbCommand(sqlBook, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drBook = cmdBook.ExecuteReader(); drBook.Read();
            string TitleID = drBook["TitleID"].ToString();

            //Select Price from Title Table with BookID
            string sqlTitle = string.Format("SELECT Price FROM Title WHERE TitleID =
'{0}'", TitleID);
            OleDbCommand cmdTitle = new OleDbCommand(sqlTitle, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drTitle = cmdTitle.ExecuteReader(); drTitle.Read();
            Price = double.Parse(drTitle["Price"].ToString());


            //Call method to find fines for this book & member Tyoe
            bool isBFine, IsBlocked; double Fine; int Days; DateTime iDate; int MTi;
            Methods m = new Methods();
            m.FindBookFine(BookID, out isBFine, out Fine, out Days, out iDate, out
MemID, out IsBlocked, out MTi);

            //Remove member from Blacklist by shaowing member details
            if (IsBlocked)
            {
                DialogResult black = MessageBox.Show("This Member was blocked due to
possible book theft or other security reasons. Since the member pays for the lost book,
do you want to check details to remove the member from blacklist?", "Remove from
Blacklist?", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
                if (black == DialogResult.Yes)
                {
                    frmMemDetails n = new frmMemDetails(); n.txtMemID.Text = MemID;
n.ShowDialog();
                }
```

```
            }
            //Calculate Amount: Amount = (Price x Defined Times) + Present fine.
            AmountOnly = (Price * set.LostBookTimes[MTi]);
            Amount = AmountOnly + Fine;

            // If there's fine, receive the fine also.
            string FineText = ""; if(isBFine) { FineText = string.Format(" and fines of
Rs. {0}/- ", Fine); }

            //Show messagebox and verify transaction
            DialogResult isok = MessageBox.Show(string.Format("This Book costs Rs.
{0}/-.\r\n\r\nWith {1} times the price of the book{2}, amount to be paid in total is
Rs. {3}/-. \r\n\r\nReceive the amount and press OK.", Price, set.LostBookTimes[MTi],
FineText, Amount ), string.Format("Rs. {0}/- must be paid Now", Amount),
MessageBoxButtons.OKCancel, MessageBoxIcon.Information);
            if (isok == DialogResult.Cancel) goto End;

            //If Okay, continue.

            if (isBFine)
            {
                //Add details of Pay Fine and Pay for book to Cash table
                string sqlPayFine = string.Format("INSERT INTO Cash (Event, TDetail,
TDate, Amount) VALUES ('FinePaid', '{0}', '{1}', {2})", string.Format("BookID = {0} |
MemberID = {1}", BookID, MemID), DateTime.Now.ToString(), Fine);
                OleDbCommand cmdPayFine = new OleDbCommand(sqlPayFine, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                cmdPayFine.ExecuteNonQuery();
            }

            string sqlRemCash = string.Format("INSERT INTO Cash (Event, TDetail, TDate,
Amount) VALUES ('LostPaid', '{0}', '{1}', {2})", string.Format("MemberID = {0}",
MemID), DateTime.Now.ToString(), AmountOnly);
            OleDbCommand cmdRemCash = new OleDbCommand(sqlRemCash, db.con);
            cmdRemCash.ExecuteNonQuery();

            //Call method to remove Book from database
            m.RemBook(BookID, string.Format("Book Lost and Paid by Member = {0}",
MemID), false);

            //Success
            MessageBox.Show(string.Format("Transaction of fine amount of Rs.{0}/- has
beeen successfully recorded and Book has been successfully removed.", Amount),
"Transaction & Removal Successful", MessageBoxButtons.OK, MessageBoxIcon.Information);

            Reset();



        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void btnReset_Click(object sender, EventArgs e)
        {
            Reset();
```

```
        }

        private void Reset()
        {
            this.Close(); frmReturnExt f = new frmReturnExt(); f.Show();
        }


    }
}
```

Check In Check Out Form (frmCheckIO)

```
namespace Alexandria
{
    public partial class frmCheckIO : Form
    {
        //Declarations
        string MemID, accFine, BookID; double FineAll = 0;

        public frmCheckIO()
        {
            InitializeComponent();
        }

        private void btnOut_Click(object sender, EventArgs e)
        {
            //Disclaimer
            if (txtMemID.Text == "") goto End;
            //Remove single & double quoteswhich cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }


            MemID = txtMemID.Text;
            string sqlMemDD = string.Format("SELECT MStatus, Fine FROM Member WHERE
MemberID = '{0}'", MemID);
            OleDbCommand cmdMemDD = new OleDbCommand(sqlMemDD, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drMemDD = cmdMemDD.ExecuteReader(); drMemDD.Read();

            //Disclaim
            if (!drMemDD.HasRows) { MessageBox.Show("Invalid Member ID", "Invalid ID",
MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }
            else if (drMemDD["MStatus"].ToString() == "Valid") goto Continue;
            else if (drMemDD["MStatus"].ToString() == "Expired") {
MessageBox.Show("This membership has expired. You cannot use the library until you
renew this membership.", "Membership Expired", MessageBoxButtons.OK,
MessageBoxIcon.Error); goto End; }
            else if (drMemDD["MStatus"].ToString() == "Blocked") {
MessageBox.Show("This membership has been blocked. Please contact the Librarian for
more details.", "Membership BLOCKED", MessageBoxButtons.OK, MessageBoxIcon.Warning);
goto End; }
            else { MessageBox.Show("This Membership ID is invalid.", "Invalid ID",
MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }

        Continue:
            {
                //Check for Account fines
                accFine = drMemDD["Fine"].ToString();
                if (accFine != "0") MessageBox.Show(string.Format("You have Rs. {0}/-
of unpaid fines stored in your account. Please pay them as soon as possible", accFine),
"You have fines", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

                //Check LendStatus for Present Fines
```

```
                string sqlFineC = string.Format("SELECT BookID FROM LendStatus WHERE
MemberID = '{0}'", MemID);
                OleDbCommand cmdFineC = new OleDbCommand(sqlFineC, db.con);
                OleDbDataReader drFineC = cmdFineC.ExecuteReader();

                while (drFineC.Read() && drFineC.HasRows)
                {
                    //Call Method to calculate present fines
                    BookID = drFineC["BookID"].ToString(); bool isBFine, IsBlocked;
double Fine; int Days; DateTime iDate; int MTi;
                    Methods m = new Methods();
                    m.FindBookFine(BookID, out isBFine, out Fine, out Days, out iDate,
out MemID, out IsBlocked, out MTi);
                    if (isBFine) FineAll = FineAll + Fine;
                }

                //Remind Present Fines
                if (FineAll != 0) MessageBox.Show(string.Format("You have Rs. {0}/- of
fines to be paid for the books you've borrowed. Please return the books and pay the
fines as soon as possible", FineAll), "You have fines", MessageBoxButtons.OK,
MessageBoxIcon.Asterisk);

                //Check whether last event was recorded.
                string sqlChk = string.Format("SELECT Event FROM CheckInOut WHERE
MemberID = '{0}' ORDER BY CDate DESC", MemID);
                OleDbCommand cmdChk = new OleDbCommand(sqlChk, db.con);
                OleDbDataReader drChk = cmdChk.ExecuteReader(); drChk.Read();

                if (!drChk.HasRows || drChk["Event"].ToString() == "Out")
                {
                    //If last checkout was not recorded, mark it as NotRec
                    string sqlNotRec = string.Format("INSERT INTO CheckInOut (MemberID,
Event) VALUES ('{0}', 'NotRec')", MemID);
                    OleDbCommand cmdNotRec = new OleDbCommand(sqlNotRec, db.con);
                    cmdNotRec.ExecuteNonQuery();
                }

                //Log CheckIn
                string sqlCin = string.Format("INSERT INTO CheckInOut (MemberID, Event,
CDate) VALUES ('{0}', 'Out', '{1}')", MemID, DateTime.Now);
                OleDbCommand cmdCin = new OleDbCommand(sqlCin, db.con);
                cmdCin.ExecuteNonQuery();

                //Success
                MessageBox.Show("Thank you for visiting the library. Your departure has
been logged. Please come again.", "Thank You", MessageBoxButtons.OK,
MessageBoxIcon.Asterisk);

                //Reset Form
                txtMemID.Clear();
            }


        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
```

```
        }

        private void btnIn_Click(object sender, EventArgs e)
        {
            //Disclaimer
            if (txtMemID.Text == "") goto End;
            //Remove single & double quoteswhich cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }


            MemID = txtMemID.Text;
            string sqlMemDD = string.Format("SELECT MStatus, Fine FROM Member WHERE
MemberID = '{0}'", MemID);
            OleDbCommand cmdMemDD = new OleDbCommand(sqlMemDD, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drMemDD = cmdMemDD.ExecuteReader(); drMemDD.Read();

            //Disclaim
            if(!drMemDD.HasRows) { MessageBox.Show("Invalid Member ID", "Invalid ID",
MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }
            else if (drMemDD["MStatus"].ToString() == "Valid") goto Continue;
            else if (drMemDD["MStatus"].ToString() == "Expired") {
MessageBox.Show("This membership has expired. You cannot use the library until you
renew this membership.", "Membership Expired", MessageBoxButtons.OK,
MessageBoxIcon.Error); goto End; }
            else if (drMemDD["MStatus"].ToString() == "Blocked") {
MessageBox.Show("This membership has been blocked. Please contact the Librarian for
more details.", "Membership BLOCKED", MessageBoxButtons.OK, MessageBoxIcon.Warning);
goto End; }
            else { MessageBox.Show("This Membership ID is invalid.", "Invalid ID",
MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }

            Continue:
            {
                //Check for Account fines
                accFine = drMemDD["Fine"].ToString();
                if (accFine != "0") MessageBox.Show(string.Format("You have Rs. {0}/-
of unpaid fines stored in your account. Please pay them as soon as possible", accFine),
"You have fines", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

                //Check LendStatus for Present Fines

                string sqlFineC = string.Format("SELECT BookID FROM LendStatus WHERE
MemberID = '{0}'", MemID);
                OleDbCommand cmdFineC = new OleDbCommand(sqlFineC, db.con);
                OleDbDataReader drFineC = cmdFineC.ExecuteReader();

                while (drFineC.Read() && drFineC.HasRows)
                {
                    //Call Method to calculate present fines
                    BookID = drFineC["BookID"].ToString(); bool isBFine, isBlocked;
double Fine; int Days ; DateTime iDate; int MTi;
                    Methods m = new Methods();
```

```
                    m.FindBookFine(BookID, out isBFine, out Fine, out Days, out iDate,
out MemID, out isBlocked, out MTi);
                    if (isBFine) FineAll = FineAll + Fine;
                }

                //Remind Present Fines
                if (FineAll != 0) MessageBox.Show(string.Format("You have Rs. {0}/- of
fines to be paid for the books you've borrowed. Please return the books and pay the
fines as soon as possible", FineAll), "You have fines", MessageBoxButtons.OK,
MessageBoxIcon.Asterisk);

                //Check whether last event was recorded.
                string sqlChk = string.Format("SELECT Event FROM CheckInOut WHERE
MemberID = '{0}' ORDER BY CDate DESC", MemID);
                OleDbCommand cmdChk = new OleDbCommand(sqlChk, db.con);
                OleDbDataReader drChk = cmdChk.ExecuteReader(); drChk.Read();

                if (!drChk.HasRows) { }
                else if (drChk["Event"].ToString() == "In")
                {
                    //If last checkout was not recorded, mark it as NotRec
                    string sqlNotRec = string.Format("INSERT INTO CheckInOut (MemberID,
Event) VALUES ('{0}', 'NotRec')", MemID);
                    OleDbCommand cmdNotRec = new OleDbCommand(sqlNotRec, db.con);
                    cmdNotRec.ExecuteNonQuery();
                }

                //Log CheckIn
                string sqlCin = string.Format("INSERT INTO CheckInOut (MemberID, Event,
CDate) VALUES ('{0}', 'In', '{1}')", MemID, DateTime.Now);
                OleDbCommand cmdCin = new OleDbCommand(sqlCin, db.con);
                cmdCin.ExecuteNonQuery();

                //Success
                MessageBox.Show("Welcome to the Library. Your arrival has been logged",
"Welcome", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);


                //Reset Form
                txtMemID.Clear();
            }


        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void frmCheckIO_Load(object sender, EventArgs e)
        {
            label1.BackColor = System.Drawing.ColorTranslator.FromHtml("#f6ebd7");
        }

        private void pboxClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }
```

```
        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

BookFinder Form (frmBSearch)

```
namespace Alexandria
{
    public partial class frmBSearch : Form
    {
        //Commom Variable Declaration

        string OrderBy = null, OrderIn = "ASC"; int[] Pg = new int[2] { 0, 0 };

        //Commom variable declaration ends

        public frmBSearch()
        {
            InitializeComponent();
        }

        private void frmBSearch_Load(object sender, EventArgs e)
        {
            SearchBooks();
        }

        private void SearchBooks()
        {
            //Declarations
            string Title, Author, Genre, BType, Publisher, OrderSQL, PageSQL;
            string TitleID, BookID, rTitle, rAuthor, rGenre, rBType, rPublisher, rPg,
rPop;
            int TotBooks, AvailBooks;

            //Disclaim for pages
            if (txtPg1.Text == "" && txtPg2.Text == "") PageSQL = "";
            else
            {
                bool isNum1 = int.TryParse(txtPg1.Text, out Pg[0]);
                bool isNum2 = int.TryParse(txtPg2.Text, out Pg[1]);
                if (!(isNum1 || isNum1)) { MessageBox.Show("Please insert valid numbers
into number of pages or leave the fields blank.", "Invalid Pages",
MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }

                //Design the SQL for filter by pages.
                PageSQL = string.Format(" AND Pg BETWEEN {0} AND {1} ", Pg[0], Pg[1]);
            }

            //Remove single & double quoteswhich cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }

            // Disclaimers end



            //Declarations
```

```
            Title = txtTitle.Text; Author = txtAuthor.Text; Genre = txtGenre.Text;
Publisher = txtPub.Text; BType = cboxType.Text; OrderBy = cboxSort.Text;
            if (BType == "Adult Lending") BType = "ALend"; else if (BType == "Child
Lending") BType = "CLend"; else if (BType == "Reference") BType = "Ref"; else if (BType
== "All") BType = ""; else BType = "";

            //Design the SQL for sort by.
            switch (OrderBy)
            {
                case "Author":
                    OrderSQL = " ORDER BY Author ";
                    break;
                case "Genre":
                    OrderSQL = " ORDER BY Genre ";
                    break;
                case "Popularity":
                    OrderSQL = " ORDER BY Popularity ";
                    break;
                case "Pages":
                    OrderSQL = " ORDER BY Pg ";
                    break;
                case "Book Type":
                    OrderSQL = " ORDER BY BType ";
                    break;
                case "Publisher":
                    OrderSQL = " ORDER BY Publisher ";
                    break;
                default: OrderSQL = " ORDER BY BTitle ";
                    break;
            }
            OrderSQL = OrderSQL + OrderIn;

            //Create a datatable

                DataTable Results = new DataTable("Results");
                Results.Columns.Add("Title");
                Results.Columns.Add("Author");
                Results.Columns.Add("Genre");
                Results.Columns.Add("Popularity");
                Results.Columns.Add("Book Type");
                Results.Columns.Add("Pages");
                Results.Columns.Add("Publisher");
                Results.Columns.Add("Available");
                Results.Columns.Add("Total");


            //End Declarations

            //*************** Call to calculate Popularity

            //Write the main sql command
            string sqlMain = string.Format("SELECT * FROM Title WHERE BTitle LIKE
'%{0}%' AND Author LIKE '%{1}%' AND Genre LIKE '%{2}%' AND BType LIKE '%{3}%' AND
Publisher LIKE '%{4}%'{5}{6}", Title, Author, Genre, BType, Publisher, PageSQL,
OrderSQL);
```

```
            OleDbCommand cmdMain = new OleDbCommand(sqlMain, db.con);
            if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drMain = cmdMain.ExecuteReader();

            //For each row in result
            while (drMain.Read())
            {
                //Load results to variables
                TitleID = drMain["TitleID"].ToString(); rTitle =
drMain["BTitle"].ToString(); rAuthor = drMain["Author"].ToString(); rGenre =
drMain["Genre"].ToString(); rPop = drMain["Popularity"].ToString(); rPg =
drMain["Pg"].ToString(); rPublisher = drMain["Publisher"].ToString(); rBType =
drMain["BType"].ToString();
                if (rBType == "CLend") rBType = "Child Lending"; else if (rBType ==
"ALend") rBType = "Adult Lending"; else rBType = "Reference";

                //Count BookIDs for given TitleID
                string sqlBookCnt = string.Format("SELECT COUNT(BookID) AS BookCnt FROM
Book WHERE TitleID = '{0}'", TitleID);
                OleDbCommand cmdBookCnt = new OleDbCommand(sqlBookCnt, db.con);
OleDbDataReader drBookCnt = cmdBookCnt.ExecuteReader(); drBookCnt.Read();
                TotBooks = int.Parse(drBookCnt["BookCnt"].ToString());
                AvailBooks = TotBooks;

                //SQL to get bookIDs for title
                string sqlBookID = string.Format("SELECT BookID FROM Book WHERE TitleID
= '{0}'", TitleID);
                OleDbCommand cmdBookID = new OleDbCommand(sqlBookID, db.con);
OleDbDataReader drBookID = cmdBookID.ExecuteReader();

                while (drBookID.Read())
                {
                    BookID = drBookID["BookID"].ToString();

                    //SQL to check each bookID in Lendstatus
                    string sqlLendD = string.Format("SELECT MemberID AS TempD FROM
LendStatus WHERE BookID = '{0}'", BookID);
                    OleDbCommand cmdLendD = new OleDbCommand(sqlLendD, db.con);
OleDbDataReader drLendD = cmdLendD.ExecuteReader(); drLendD.Read();

                    if (drLendD.HasRows) AvailBooks = AvailBooks - 1;
                }

                //Add Details to Data Table
                Results.Rows.Add(rTitle, rAuthor, rGenre, rPop, rBType, rPg,
rPublisher, AvailBooks, TotBooks);
            }

            //Display results in data grid

            if (Results.Rows.Count == 0) MessageBox.Show("There are no titles with the
specified details. Try removing or modifying few filters.", "No Results",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
            dtResults.DataSource = Results;
```

```
        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void cboxSort_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (OrderBy == cboxSort.SelectedItem.ToString())
            {
                if (OrderIn == "ASC") OrderIn = "DESC";
                else OrderIn = "ASC";
            }

            OrderBy = cboxSort.SelectedItem.ToString();
            SearchBooks();
        }

        private void btnSearch_Click(object sender, EventArgs e)
        {
            //Call method
            SearchBooks();
        }

    }
}
```

Transactions Form (frmCash)

```
namespace Alexandria
{
    public partial class frmCash : Form
    {
        public frmCash()
        {
            InitializeComponent();
        }

        private void btnSearch_Click(object sender, EventArgs e)
        {
            //Remove single & double quoteswhich cause sql troubles
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }

            //Call method
            CashSearch();
        }

        private void CashSearch()
        {
            //Declarations
            string TransID, Time, TDate, TTime, TAmount, TDetail, Event, sEvent,
AmntSQL; double[] Amnt = new double[2]; DateTime iDate;

            //Disclaim for Amount
            if (txtAmnt1.Text == "" && txtAmnt2.Text == "") AmntSQL = "";
            else
            {
                bool isNum1 = double.TryParse(txtAmnt1.Text, out Amnt[0]);
                bool isNum2 = double.TryParse(txtAmnt2.Text, out Amnt[1]);
                if (!(isNum1 || isNum1)) { MessageBox.Show("Please insert valid numbers
into amounts or leave the fields blank.", "Invalid Amounts", MessageBoxButtons.OK,
MessageBoxIcon.Error); goto End; }

                //Design the SQL for filter by pages.
                AmntSQL = string.Format(" AND Amount BETWEEN {0} AND {1} ", Amnt[0],
Amnt[1]);
            }

            // Disclaimers end

            //Declarations
            Time = cboxDate.Text; sEvent = cboxEvent.Text;

            if (Time == "Today") iDate =
DateTime.Parse(DateTime.Today.ToShortDateString());
            else if (Time == "Past Week") iDate = DateTime.Today.AddDays(-7);
            else if (Time == "Last Two Weeks") iDate = DateTime.Today.AddDays(-14);
            else if (Time == "Past Month") iDate = DateTime.Today.AddMonths(-1);
```

```
            else if (Time == "Last Two Months") iDate = DateTime.Today.AddMonths(-2);
            else if (Time == "Last Six Months") iDate = DateTime.Today.AddMonths(-6);
            else if (Time == "Past Year") iDate = DateTime.Today.AddYears(-1);
            else iDate = DateTime.Today;

            //Write Date sql
            string DateSQL;
            if (Time == "Today") DateSQL = string.Format(" AND TDate >= #{0}#", iDate);
            else if(iDate == DateTime.Today) DateSQL = "";
            else DateSQL = string.Format(" AND TDate >= #{0}#", iDate);

            //Modify Event to suit Database
            if (sEvent == "New Stock") sEvent = "NewStock";
            else if (sEvent == "Remove Books") sEvent = "RemBook";
            else if (sEvent == "New Membership") sEvent = "NewMem";
            else if (sEvent == "Membership Renewal") sEvent = "MemRenew";
            else if (sEvent == "Fine") sEvent = "Fine";
            else if (sEvent == "Paid for Lost Books") sEvent = "LostPaid";
            else sEvent = "";

            //Datatable
            DataTable Results = new DataTable();
            Results.Columns.Add("ID");
            Results.Columns.Add("Event");
            Results.Columns.Add("Amount (Rs)");
            Results.Columns.Add("Date");
            Results.Columns.Add("Time");
            Results.Columns.Add("Detail");


            //Declarations end


            //Write Main SQL

            string sqlMain = string.Format("SELECT * FROM Cash WHERE Event LIKE
'%{0}%'{1}{2}", sEvent, DateSQL, AmntSQL);
            OleDbCommand cmdMain = new OleDbCommand(sqlMain, db.con);
            if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drMain = cmdMain.ExecuteReader();

            //For each row in result
            while (drMain.Read())
            {
                TransID = drMain["TransID"].ToString(); TDate =
drMain["TDate"].ToString(); TAmount = drMain["Amount"].ToString(); TDetail =
drMain["TDetail"].ToString(); Event = drMain["Event"].ToString();
                TTime = DateTime.Parse(TDate).ToShortTimeString(); TDate =
DateTime.Parse(TDate).ToString("dd MMMM yyyy"); ;

                if (Event == "NewStock") Event = "New Stock";
                else if (Event == "RemBook") Event = "Book Removed";
                else if (Event == "NewMem") Event = "New Membership";
                else if (Event == "MemRenew") Event = "Membership Renewal";
                else if (Event == "Fine") Event = "Fine";
```

```
                else if (Event == "LostPaid") Event = "Paid for Lost Book";
                else if (Event == "AccFinePaid") Event = "Paid for Fines in Account";

                Results.Rows.Add(TransID, Event, TAmount, TDate, TTime, TDetail);
            }

            //Find total Transaction
            double sum = 0;
            for (int i = 0; i < Results.Rows.Count; i++)
            {
                sum = sum + double.Parse(Results.Rows[i][2].ToString());
            }

            //Show total
           lblTot.Text = sum.ToString();


            //Show results in Data Grid
            if (Results.Rows.Count == 0) MessageBox.Show("There are no transactions
with the specified details. Try removing or modifying few filters.", "No Results",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
            dtResults.DataSource = Results;


        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void frmCash_Load(object sender, EventArgs e)
        {
            //Call method
            CashSearch();
            cboxDate.SelectedIndex = 7; cboxEvent.SelectedIndex = 6;
        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Member Logbook Form (frmViewCheckIO)

```
namespace Alexandria
{
    public partial class frmViewCheckIO : Form
    {

        public frmViewCheckIO()
        {
            InitializeComponent();
        }

        private void btnFilter_Click(object sender, EventArgs e)
        {

            //Declarations
            string MemID = txtMemID.Text, MStatus = cboxStatus.Text, Event =
cboxEvent.Text, Time = cboxDate.Text, MType = cboxMType.Text; DateTime iDate;
            //Remove aingle double quotes in textboxes
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }

            //Check MemID Validity
            if (MemID != "")
            {
                string sqlIsMem = string.Format("SELECT MemberID FROM Member WHERE
MemberID = '{0}'", MemID);
                OleDbCommand cmdIsMem = new OleDbCommand(sqlIsMem, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drIsMem = cmdIsMem.ExecuteReader(); drIsMem.Read();

                //Disclaimer
                if (!drIsMem.HasRows) { MessageBox.Show("No member is found with such
Member ID", "Invalid Member ID", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End;
}
            }

            //Declare MemType
            if (MType == "Child Member") MType = "Child"; else if (MType == "Adult
Member") MType = "Adult"; else MType = "";

            //Declare Date
            if (Time == "Today") iDate =
DateTime.Parse(DateTime.Today.ToShortDateString());
            else if (Time == "Past Week") iDate = DateTime.Today.AddDays(-7);
            else if (Time == "Last Two Weeks") iDate = DateTime.Today.AddDays(-14);
            else if (Time == "Past Month") iDate = DateTime.Today.AddMonths(-1);
            else if (Time == "Last Two Months") iDate = DateTime.Today.AddMonths(-2);
            else if (Time == "Last Six Months") iDate = DateTime.Today.AddMonths(-6);
            else if (Time == "Past Year") iDate = DateTime.Today.AddYears(-1);
            else iDate = DateTime.Today;

            //Write Date sql
            string DateSQL;
```

```
            if (Time == "Today") DateSQL = string.Format(" AND CDate >= #{0}#",
DateTime.Today.Date);
            else if (iDate == DateTime.Today) DateSQL = "";
            else DateSQL = string.Format(" AND CDate >= #{0}#", iDate);

            //Declare Event
            if (Event == "Check In") Event = "In";
            else if (Event == "Check Out") Event = "Out";
            else if (Event == "Not Logged") Event = "NotRec";
            else Event = "";

            //Declare MStatus
            if (MStatus == "All") MStatus = "";

            //Call Method
            GetLog(MemID, MType, MStatus, DateSQL, Event);

        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();

        }

        private void GetLog(string MemID, string MType, string MStatus, string DateSQL,
string Event)
        {
            //Declarations
            string rEvent, rmMemID, rMemID, rDate, rTime, rMType, rMStatus, rMName;

            //Create Datatable
            DataTable dtAllLog = new DataTable();

            //Design Datatable
            dtAllLog.Columns.Add("Event");
            dtAllLog.Columns.Add("Date");
            dtAllLog.Columns.Add("Time");
            dtAllLog.Columns.Add("Member ID");
            dtAllLog.Columns.Add("Member Name");
            dtAllLog.Columns.Add("Type");
            dtAllLog.Columns.Add("Status");

            //Write Main SQL
            string sqlMain = string.Format("SELECT * FROM CheckInOut WHERE MemberID
LIKE '%{0}%' AND Event LIKE '%{1}%'{2}", MemID, Event, DateSQL);
            OleDbCommand cmdMain = new OleDbCommand(sqlMain, db.con);
            if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drMain = cmdMain.ExecuteReader();

            //For each row in result
            while (drMain.Read())
            {
                //Temp. Declarations
                rMemID = drMain["MemberID"].ToString(); rDate =
drMain["CDate"].ToString(); rEvent = drMain["Event"].ToString();

                if (rDate != "")
```

```
                {
                    rTime = DateTime.Parse(rDate).ToShortTimeString();
                    rDate = DateTime.Parse(rDate).ToString("dd MMMM yyyy");
                }
                else rTime = "";

                if (rEvent == "In") rEvent = "Check In"; else if (rEvent == "Out")
rEvent = "Check Out"; else rEvent = "Not Logged";

                //Get Member Details
                string sqlMemD = string.Format("SELECT MType, MStatus, FName, LName
FROM Member WHERE MemberID = '{0}' AND MStatus LIKE '%{1}%' AND MType LIKE '%{2}%'",
rMemID, MStatus, MType);
                OleDbCommand cmdMemD = new OleDbCommand(sqlMemD, db.con);
                if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drMemD = cmdMemD.ExecuteReader();

                //For each row in Member result
                while (drMemD.Read())
                {
                    rmMemID = rMemID; rMName = drMemD["FName"].ToString() + " " +
drMemD["LName"].ToString(); rMStatus = drMemD["MStatus"].ToString(); rMType =
drMemD["MType"].ToString();

                    dtAllLog.Rows.Add(rEvent, rDate, rTime, rmMemID, rMName, rMType,
rMStatus);
                }
            }

            //Show results in Data Grid
            if (dtAllLog.Rows.Count == 0) MessageBox.Show("There are no Member Logs
with the specified details. Try removing or modifying few filters.", "No Results",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
            dtResults.DataSource = dtAllLog;

        if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void frmViewCheckIO_Load(object sender, EventArgs e)
        {
            //Call Method
            GetLog("", "", "", "", "");
        }

        private void dtResults_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            if (dtResults.SelectedCells.Count != 0)
            {
                int NowCol = dtResults.SelectedCells[0].ColumnIndex;
                string NowValue =
dtResults.Rows[dtResults.CurrentCell.RowIndex].Cells[3].Value.ToString();

                frmMemDetails f = new frmMemDetails();
                if (NowCol == 3 || NowCol == 4)
                { f.txtMemID.Text = NowValue; f.ShowDialog(); }
```

```
            }
        }

    }
}
```

Statistics Form (frmExtra)

```
namespace Alexandria
{
    public partial class frmExtra : Form
    {
        public frmExtra()
        {
            InitializeComponent();
        }

        List<string> Title = new List<string>();

        private void frmExtra_Load(object sender, EventArgs e)
        {

            //Calculate popularity and stars
            Popul p = new Popul();
            p.BookCalcAll();
            p.MemCalcAll();



            //Adding Top Members
            {
                //Declarations
                string topMemID, topMPop;

                //Count Members
                string sqlCntMem = string.Format("SELECT COUNT(MemberID) as CntMem FROM
Member");
                OleDbCommand cmdCntMem = new OleDbCommand(sqlCntMem, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drCntMem = cmdCntMem.ExecuteReader(); drCntMem.Read();
                int No = int.Parse(drCntMem["CntMem"].ToString());

                //Write SQL for top members
                string sqlTopMem = string.Format("SELECT MemberID, Stars FROM Member
WHERE Stars NOT IN (0) ORDER BY Stars DESC");
                OleDbCommand cmdTopMem = new OleDbCommand(sqlTopMem, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drTopMem = cmdTopMem.ExecuteReader(); drTopMem.Read();

                int n = 5; int i = 0;
                if (No < n) n = No;

                while (drTopMem.Read() && i < n)
```

```
                {
                    topMemID = drTopMem["MemberID"].ToString(); topMPop =
drTopMem["Stars"].ToString();
                    topMPop = Math.Round(decimal.Parse(topMPop.ToString()),
2).ToString();
                    lboxTopMem.Items.Add(string.Format("{0} [{1}]", topMemID,
topMPop));
                    i++;
                }
            }

            //Adding Top Titles
            {
                //Declarations
                string topTitleID, topTPop;

                //Count Titles
                string sqlCntTitle = string.Format("SELECT COUNT(TitleID) as CntTitle
FROM Title");
                OleDbCommand cmdCntTitle = new OleDbCommand(sqlCntTitle, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drCntTitle = cmdCntTitle.ExecuteReader();
drCntTitle.Read();
                int No = int.Parse(drCntTitle["CntTitle"].ToString());

                //Write SQL for top Titles
                string sqlTopTitle = string.Format("SELECT TitleID, BTitle, Popularity
FROM Title WHERE Popularity ORDER BY Popularity DESC");
                OleDbCommand cmdTopTitle = new OleDbCommand(sqlTopTitle, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drTopTitle = cmdTopTitle.ExecuteReader();

                int n = 5; int i = 0;
                if (No < n) n = No;

                while (drTopTitle.Read() && i < n)
                {
                    topTitleID = drTopTitle["BTitle"].ToString(); topTPop =
drTopTitle["Popularity"].ToString();
                    topTPop = Math.Round(decimal.Parse(topTPop.ToString()),
2).ToString();
                    lboxTitlePop.Items.Add(string.Format("[{0}] {1}", topTPop,
topTitleID));
                    Title.Add(drTopTitle["TitleID"].ToString());
                    i++;
                }
            }

            //Adding Top Authors
            {
                //Declarations
                string topAuthor, topAPop;

                //Count Authors
```

```
                string sqlCntAuthor = "SELECT COUNT(Author) AS CntAuthor FROM (SELECT
DISTINCT Author FROM Title)";
                OleDbCommand cmdCntAuthor = new OleDbCommand(sqlCntAuthor, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drCntAuthor = cmdCntAuthor.ExecuteReader();
drCntAuthor.Read();
                int No = int.Parse(drCntAuthor["CntAuthor"].ToString());

                //Write SQL for top Titles
                string sqlTopAuthor = string.Format("SELECT Author, SUM(Popularity) AS
TotPop FROM Title GROUP BY Author ORDER BY SUM(Popularity) DESC");
                OleDbCommand cmdTopAuthor = new OleDbCommand(sqlTopAuthor, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drTopAuthor = cmdTopAuthor.ExecuteReader();

                int n = 5; int i = 0;
                if (No < n) n = No;

                while (drTopAuthor.Read() && i < n)
                {
                    topAuthor = drTopAuthor["Author"].ToString(); topAPop =
drTopAuthor["TotPop"].ToString();
                    topAPop = Math.Round(decimal.Parse(topAPop.ToString()),
2).ToString();
                    if(topAPop != "0") lboxAuthorPop.Items.Add(string.Format("[{0}]
{1}", topAPop, topAuthor));

                    i++;
                }
            }

            //Adding Top Genres
            {
                //Declarations
                string TopGenre, topGPop;

                //Count Genres
                string sqlCntGenre = string.Format("SELECT COUNT(Genre) as CntGenre
FROM (SELECT DISTINCT Genre FROM Title)");
                OleDbCommand cmdCntGenre = new OleDbCommand(sqlCntGenre, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drCntGenre = cmdCntGenre.ExecuteReader();
drCntGenre.Read();
                int No = int.Parse(drCntGenre["CntGenre"].ToString());

                //Write SQL for top Titles
                string sqlTopGenre = string.Format("SELECT Genre, SUM(Popularity) AS
TotPop FROM Title GROUP BY Genre ORDER BY SUM(Popularity) DESC");
                OleDbCommand cmdTopGenre = new OleDbCommand(sqlTopGenre, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drTopGenre = cmdTopGenre.ExecuteReader();

                int n = 5; int i = 0;
                if (No < n) n = No;
```

```
                while (drTopGenre.Read() && i < n)
                {

                    TopGenre = drTopGenre["Genre"].ToString(); topGPop =
drTopGenre["TotPop"].ToString();
                    topGPop = Math.Round(decimal.Parse(topGPop.ToString()),
2).ToString();
                    if (topGPop != "0") lboxGenrePop.Items.Add(string.Format("[{0}]
{1}", topGPop, TopGenre));

                    i++;
                }
            }

            //Fill Blacklist
            {
                //Write SQL for top Titles
                string sqlBlack = "SELECT MStatus, MemberID FROM Member";
                OleDbCommand cmdBlack = new OleDbCommand(sqlBlack, db.con); if
(db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drBlack = cmdBlack.ExecuteReader(); drBlack.Read();

                while (drBlack.Read())
                {
                    if(drBlack["MStatus"].ToString() == "Blocked")
lboxBlack.Items.Add(drBlack["MemberID"].ToString());
                }
            }

            //Fill all common stats in method
            FillAllStats();

            if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();

        }

        private void lboxTopMem_DoubleClick(object sender, EventArgs e)
        {
            if (lboxTopMem.SelectedItems.Count == 1)
            {
                frmMemDetails f = new frmMemDetails();
                f.txtMemID.Text = lboxTopMem.SelectedItem.ToString().Substring(0, 5);
                f.ShowDialog();
            }
        }

        private void lboxBlack_DoubleClick(object sender, EventArgs e)
        {
            if (lboxBlack.SelectedItems.Count == 1)
            {
                frmMemDetails f = new frmMemDetails();
                f.txtMemID.Text = lboxBlack.SelectedItem.ToString();
                f.ShowDialog();
            }
        }
```

```
        private void lboxTitlePop_DoubleClick(object sender, EventArgs e)
        {
            if (lboxTitlePop.SelectedItems.Count == 1)
            {
                int i = lboxTitlePop.SelectedIndex;

                frmBookDetail f = new frmBookDetail();
                f.FillTitleInfo(Title[i]);
                f.txtTitleID.Text = Title[i];
                f.btnGetTitle_Click(sender, e);
            }
        }

        private void FillAllStats()
        {

            //Array to fill Member Stats
            string[] sqlCountMem = new string[] { "MemberID LIKE '%'", "MType =
'Adult'", "MType = 'Child'", "MStatus = 'Valid'", "MStatus = 'Expired'", "MStatus =
'Blocked'" };
            string[] CountMem = new string[6];

            //Loop to count MemberID of each type and add to array
            for (int i = 0; i < 6; i++)
            {
                string sqlTotM = string.Format("SELECT COUNT(MemberID) AS MemID FROM
Member WHERE {0}", sqlCountMem[i]);
                OleDbCommand cmdTotM = new OleDbCommand(sqlTotM, db.con);
                if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
                OleDbDataReader drTotM = cmdTotM.ExecuteReader(); drTotM.Read();

                CountMem[i] = drTotM["MemID"].ToString();
            }

            //Show data in labels
            lblTotMem.Text = CountMem[0]; lblAdultM.Text = CountMem[1]; lblChildM.Text
= CountMem[2]; lblValidM.Text = CountMem[3]; lblExpiredM.Text = CountMem[4];
lblBlockedM.Text = CountMem[5];




            //Arrays to fill TotBooks and BType stats
            string[] sqlCountTit1 = new string[] { "TitleID LIKE '%'", "BType =
'ALend'", "BType = 'CLend'", "BType = 'Ref'" };
            string[] CountBook1 = new string[4]; int TotBooks;

            //Loop to fill each of 4 info.
            for (int i = 0; i < 4; i++)
            {
                string sqlEachT = string.Format("SELECT TitleID FROM Title WHERE {0}",
sqlCountTit1[i]);
                OleDbCommand cmdEachT = new OleDbCommand(sqlEachT, db.con);
```

```
                OleDbDataReader drEachT = cmdEachT.ExecuteReader();

                int tNoBooks = 0;

                //For each TitleID of specific type, no. of books are calculated
                while (drEachT.Read())
                {
                    string tTitleID = drEachT["TitleID"].ToString();
                    string sqlEachBT = string.Format("SELECT COUNT(BookID) as Book FROM
Book WHERE TitleID = '{0}'", tTitleID);
                    OleDbCommand cmdEachBT = new OleDbCommand(sqlEachBT, db.con);
                    OleDbDataReader drEachBT = cmdEachBT.ExecuteReader();
drEachBT.Read();

                    //No. of books for each titleid are added to a common interger
represents the no of books in that type.
                    tNoBooks = tNoBooks + int.Parse(drEachBT["Book"].ToString());
                }

                //common integer is added to array
                CountBook1[i] = tNoBooks.ToString();
            }
            //Get total no. of books
            TotBooks = int.Parse(CountBook1[0]);

            //Show data in labels
            lblTotbook.Text = CountBook1[0]; lblAdultB.Text = CountBook1[1];
lblChildB.Text = CountBook1[2]; lblRefB.Text = CountBook1[3];



            //Arrays to find no of title, authors and genres
            string[] sqlTitCommon = new string[] { "TitleID", "Author", "Genre" };
            string[] TitCommon = new string[3];

            //Loop to fill each 3 info
            for (int i = 0; i < 3; i++)
            {
                string sqlCommon = string.Format("SELECT COUNT({0}) as Common FROM
(SELECT DISTINCT {0} FROM Title)", sqlTitCommon[i]);
                OleDbCommand cmdCommon = new OleDbCommand(sqlCommon, db.con);
                OleDbDataReader drCommon = cmdCommon.ExecuteReader(); drCommon.Read();

                TitCommon[i] = drCommon["Common"].ToString();
            }

            //Show data in labels
            lblTotTitle.Text = TitCommon[0]; lblAuthor.Text = TitCommon[1];
lblGenre.Text = TitCommon[2];



            //Count no. of bookID in Lendstatus as the no. of lent books.
```

```
            string sqlIsLent = string.Format("SELECT COUNT(BookID) as LentB FROM
LendStatus");
            OleDbCommand cmdIsLent = new OleDbCommand(sqlIsLent, db.con);
            OleDbDataReader drIsLent = cmdIsLent.ExecuteReader(); drIsLent.Read();

            int LentB = int.Parse(drIsLent["LentB"].ToString());
            int AvailB = TotBooks - LentB; //Avail books are calculated.

            //Show data in labels
            lblLentB.Text = LentB.ToString();
            lblAvailB.Text = AvailB.ToString();
        }

    }
}
```

Preferences Form (frmSettings)

```
namespace Alexandria
{
    public partial class frmSetting : Form
    {
        bool isSave = false;


        public frmSetting()
        {
            InitializeComponent();
        }

        private void btnSave_Click(object sender, EventArgs e)
        {
            if (!isSave)
            {
                //Call method to save
                setSave();
            }
        }

        private void setSave()
        {
            //Disclaim free boxes
            if (AtxtBTime.Text == "" || CtxtBTime.Text == "" || AtxtNoBooks.Text == ""
|| CtxtNoBooks.Text == "" || AtxtExpire.Text == "" || CtxtExpire.Text == "" ||
AtxtAlert.Text == "" || CtxtAlert.Text == "" || AtxtLostBookTimes.Text == "" ||
CtxtLostBookTimes.Text == "" || AtxtNewC.Text == "" || CtxtNewC.Text == "" ||
AtxtRenewC.Text == "" || CtxtRenewC.Text == "")
            { MessageBox.Show("No fields should be left blank. Fill all fields with
numbers and try again", "Blank values", MessageBoxButtons.OK, MessageBoxIcon.Error);
goto End; }
            //Remove aingle double quotes in textboxes
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }

            //Declarations, move all text to two string arrays and variable
            string[,] SValueSI = new string[,] { { AtxtBTime.Text, CtxtBTime.Text }, {
AtxtNoBooks.Text, CtxtNoBooks.Text }, { AtxtExpire.Text, CtxtExpire.Text }, {
AtxtAlert.Text, CtxtAlert.Text }, { AtxtLostBookTimes.Text, CtxtLostBookTimes.Text } };
            string[,] SValueSD = new string[,] { { AtxtFine.Text, CtxtFine.Text }, {
AtxtRenewC.Text, CtxtRenewC.Text }, { AtxtNewC.Text, CtxtNewC.Text } };
            string AgeMinS = AtxtAgeMin.Text;

            //Declare arrays of column names of Setting table for DB purposes.
            string[] SType = new string[] { "A", "C" };
            string[] SNameI = new string[] { "Btime", "NoBooks", "Expire", "Alert",
"LostBookTimes" };
            string[] SNameD = new string[] { "Fine", "RenewC", "NewC" };
```

```
            //Create a double and int arrays and variable
            int[,] SValueI = new int[,] { { 0, 0 }, { 0, 0 }, { 0, 0 }, { 0, 0 }, { 0,
0 } };
            double[,] SValueD = new double[,] { { 0, 0 }, { 0, 0 }, { 0, 0 } };
            int AgeMinI;

            //Try Parsing in Main for loop for each dimension set in SValueSI
            for (int i = 0; i < SValueSI.Length / 2; i++)
            {
                //Try parsing each item of each dimension set of SValueSI
                for (int j = 0; j < 2; j++)
                {
                    //Try to parse respective values to int and move them to integer
array.
                    bool isNum = int.TryParse(SValueSI[i, j], out SValueI[i, j]);

                    //If not an integer, raise message and stop.
                    if (isNum == false) { MessageBox.Show("One or many of the values
you entered is invalid. Please correct them and and try saving agian.", "Invalid
Values", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }
                }
            }

            //Try Parsing in Main for loop for each dimension set in SValueSI
            for (int i = 0; i < SValueSD.Length / 2; i++)
            {
                //Try parsing each item of each dimension set of SValueSI
                for (int j = 0; j < 2; j++)
                {
                    //Try to parse respective values to int and move them to double
array.
                    bool isNum = double.TryParse(SValueSD[i, j], out SValueD[i, j]);

                    //If not a double, raise message and stop.
                    if (isNum == false) { MessageBox.Show("One or many of the values
you entered is invalid. Please correct them and and try saving agian.", "Invalid
Values", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }
                }
            }
            //Try Parsing AgeMin
            bool isNum1 = int.TryParse(AgeMinS, out AgeMinI);
            if (isNum1 == false) { MessageBox.Show("The mimum age value you entered is
not a valid number. Please correct it and and try saving agian.", "Invalid Age",
MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }

            //Disclaimer
            DialogResult sure = MessageBox.Show("Are you sure you want to update the
preference data with these values?", "Are you sure?", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);
            if (sure == DialogResult.No) goto End; // If disagree, stop.



            //     If agree, continue, add values to database
```

```
            //Open connection
            if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();

            // Main For loop of Int to repeatedly add interger values to DB (using
SNameI) and add values from SValue[*,_] 10 times.
            for (int iSN = 0; iSN < SNameI.Length; iSN++)
            {
                // Sub for loop to repeatedly add values to DB (using SType) and add
from SValue[_,*] twice during each main loop.
                for (int iST = 0; iST < SType.Length; iST++)
                {
                    string sqlSet = string.Format("UPDATE Setting SET SValue = {0}
WHERE SName = '{1}' AND SType = '{2}'", SValueI[iSN, iST], SNameI[iSN], SType[iST]);
                    OleDbCommand cmdSet = new OleDbCommand(sqlSet, db.con);
cmdSet.ExecuteNonQuery();
                }
            }

            // Main For loop of double to repeatedly retrive double values from DB
(using SNameD) and add values to SValue[*,_] 15 times.
            for (int iSN = 0; iSN < SNameD.Length; iSN++)
            {
                // Sub for loop to repeatedly add values from DB (using SType) and add
to SValue[_,*] twice during each main loop.
                for (int iST = 0; iST < SType.Length; iST++)
                {
                    string sqlSet = string.Format("UPDATE Setting SET SValue = {0}
WHERE SName = '{1}' AND SType = '{2}'", SValueD[iSN, iST], SNameD[iSN], SType[iST]);
                    OleDbCommand cmdSet = new OleDbCommand(sqlSet, db.con);
cmdSet.ExecuteNonQuery();
                }
            }

            string sqlSetAge = string.Format("UPDATE Setting SET SValue = {0} WHERE
SName = 'AgeMin'", AgeMinI);
            OleDbCommand cmdSetAge = new OleDbCommand(sqlSetAge, db.con);
cmdSetAge.ExecuteNonQuery();

            //Recall The new settings
            set s = new set(); s.recall();

            //All successfull
            MessageBox.Show("Preferences have been saved succesfully", "Saving
successful", MessageBoxButtons.OK, MessageBoxIcon.Information);
            isSave = true;

        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close(); ;
        }

        private void frmSetting_Load(object sender, EventArgs e)
        {
            //Load settings from set vaiables to textboxes

            AtxtAgeMin.Text = set.AgeMin.ToString();
```

```
            AtxtAlert.Text = set.Alert[0].ToString();
            AtxtBTime.Text = set.Btime[0].ToString();
            AtxtExpire.Text = set.Expire[0].ToString();
            AtxtFine.Text = set.Fine[0].ToString();
            AtxtLostBookTimes.Text = set.LostBookTimes[0].ToString();
            AtxtNewC.Text = set.NewC[0].ToString();
            AtxtNoBooks.Text = set.NoBooks[0].ToString();
            AtxtRenewC.Text = set.RenewC[0].ToString();

            CtxtAlert.Text = set.Alert[1].ToString();
            CtxtBTime.Text = set.Btime[1].ToString();
            CtxtExpire.Text = set.Expire[1].ToString();
            CtxtFine.Text = set.Fine[1].ToString();
            CtxtLostBookTimes.Text = set.LostBookTimes[1].ToString();
            CtxtNewC.Text = set.NewC[1].ToString();
            CtxtNoBooks.Text = set.NoBooks[1].ToString();
            CtxtRenewC.Text = set.RenewC[1].ToString();

            isSave = true;

        }

        private void frmSetting_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (isSave == false)
            {
                //Disclaimer
                DialogResult sure = MessageBox.Show("Do you want to exit without saving
these preferences?", "Are you sure?", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);
                if (sure == DialogResult.Yes){ }
                else e.Cancel = true; // If disagree, stop closing.
            }
        if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void falseSave(object sender, EventArgs e)
        {
            isSave = false;
        }

        private void btnExit_Close(object sender, EventArgs e)
        {
            this.Close();
        }

        private void btnRem_Click(object sender, EventArgs e)
        {
            //Check Null
            if(cboxUTy.Text =="" || txtPW.Text == "" || txtUN.Text == "")
            { MessageBox.Show("Required fields cannot not be left blank.", "Blank
Fields", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }
            //Remove aingle double quotes in textboxes
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }
```

```
            //Declarations
            string UN = txtUN.Text, PW = txtPW.Text, Ty = cboxUTy.Text;
            if (Ty == "Administrator") Ty = "Admin";

            //Check Valid
            string sqlCheck = string.Format("SELECT * FROM UserAccount WHERE UName =
'{0}' AND UType = '{1}' AND Pwd = '{2}'", UN, Ty, PW);
            OleDbCommand cmdCheck = new OleDbCommand(sqlCheck, db.con);
            if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drCheck = cmdCheck.ExecuteReader(); drCheck.Read();

            if (!drCheck.HasRows) { MessageBox.Show("No User account with such
combination exists.", "Invalid Credentials", MessageBoxButtons.OK,
MessageBoxIcon.Error); goto End; }

            //Verify Delete
            DialogResult sure = MessageBox.Show("Are you sure you want to remove this
account? This action cannot be undone.", "Are you sure?", MessageBoxButtons.YesNo,
MessageBoxIcon.Warning);
            if (sure != DialogResult.Yes) goto End;

            string sqlRem = string.Format("DELETE FROM UserAccount WHERE UName = '{0}'
AND UType = '{1}' AND Pwd = '{2}'", UN, Ty, PW);
            OleDbCommand cmdRem = new OleDbCommand(sqlRem, db.con);
cmdRem.ExecuteNonQuery();

            //Success
            MessageBox.Show("The user account has been deleted successfully.", "Removal
Successful", MessageBoxButtons.OK, MessageBoxIcon.Information);

            //Reset Controls
            txtPW.Text = txtUN.Text = cboxUTy.Text = "";

        End: if(db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void btnCreate_Click(object sender, EventArgs e)
        {
            //Check Null
            if (cboxUTy.Text == "" || txtPW.Text == "" || txtUN.Text == "")
            { MessageBox.Show("Required fields cannot not be left blank.", "Blank
Fields", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }
            //Remove aingle double quotes in textboxes
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }

            //Declarations
            string UN = txtUN.Text, PW = txtPW.Text, Ty = cboxUTy.Text;
            if (Ty == "Administrator") Ty = "Admin";

            //Check If already present
            string sqlCheck = string.Format("SELECT UName FROM UserAccount WHERE UName
= '{0}'", UN);
            OleDbCommand cmdCheck = new OleDbCommand(sqlCheck, db.con);
```

```
            if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drCheck = cmdCheck.ExecuteReader(); drCheck.Read();

            if (drCheck.HasRows) { MessageBox.Show("This username is already in use.
Please choose a new username", "Username in use.", MessageBoxButtons.OK,
MessageBoxIcon.Error); goto End; }


            string sqlNew = string.Format("INSERT INTO UserAccount (UName, UType, Pwd)
VALUES ('{0}', '{1}', '{2}')", UN, Ty, PW);
            OleDbCommand cmdNew = new OleDbCommand(sqlNew, db.con);
cmdNew.ExecuteNonQuery();

            //Success
            MessageBox.Show("New User account has been created successfully.",
"Creation Successful", MessageBoxButtons.OK, MessageBoxIcon.Information);

            //Reset Controls
            txtPW.Text = txtUN.Text = cboxUTy.Text = "";


        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void btnSet_Click(object sender, EventArgs e)
        {
            //Check Null
            if (cboxUTyCh.Text == "" || txtPWOld.Text == "" || txtUNOld.Text == "")
            { MessageBox.Show("Required fields cannot not be left blank.", "Blank
Fields", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }
            //Remove aingle double quotes in textboxes
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }

            //Declarations
            string UNold = txtUNOld.Text, PWold = txtPWOld.Text, Ty = cboxUTyCh.Text;
            if (Ty == "Administrator") Ty = "Admin";

            //Check Valid
            string sqlCheck = string.Format("SELECT UName FROM UserAccount WHERE UName
= '{0}' AND UType = '{1}' AND Pwd = '{2}'", UNold, Ty, PWold);
            OleDbCommand cmdCheck = new OleDbCommand(sqlCheck, db.con);
            if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
            OleDbDataReader drCheck = cmdCheck.ExecuteReader(); drCheck.Read();

            if (!drCheck.HasRows) { MessageBox.Show("No User account with such
combination exists.", "Invalid Credentials", MessageBoxButtons.OK,
MessageBoxIcon.Error); goto End; }
            //If valid, continue..

            btnSet.Enabled = txtPWOld.Enabled = txtUNOld.Enabled = cboxUTyCh.Enabled =
false;
            btnUpdate.Enabled = txtUNNew.Enabled = txtPWNew1.Enabled =
txtPWNew2.Enabled = true;
            txtUNNew.Text = UNold;
```

```
        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            //Check Null
            if (txtPWNew1.Text == "" || txtUNNew.Text == "" || txtPWNew2.Text == "")
            { MessageBox.Show("Required fields cannot not be left blank.", "Blank
Fields", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }
            //Remove aingle double quotes in textboxes
            foreach (Control c in this.Controls) { if (c is TextBox || c is ComboBox) {
c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); } }

            //Check both new passwords
            if (txtPWNew1.Text != txtPWNew2.Text) { MessageBox.Show("Passwords
mismatch. Repeated Password should be Identical to the new password. Try again.",
"Passwords Mismatch", MessageBoxButtons.OK, MessageBoxIcon.Error); goto End; }

            //Declarations
            string PW = txtPWNew1.Text, UNnew = txtUNNew.Text, UNold = txtUNOld.Text;

            string sqlUpdate = string.Format("UPDATE UserAccount SET UName = '{0}', Pwd
= '{1}' WHERE UName = '{2}'", UNnew, PW, UNold);
            OleDbCommand cmdUpdate = new OleDbCommand(sqlUpdate, db.con);
            if (db.con.State.Equals(ConnectionState.Closed)) db.con.Open();
cmdUpdate.ExecuteNonQuery();

            //Success
            MessageBox.Show("User account has been updated successfully.", "Update
Successful", MessageBoxButtons.OK, MessageBoxIcon.Information);

            //Reset Form
            cboxUTyCh.Text = txtPWNew1.Text = txtPWNew2.Text = txtPWOld.Text =
txtUNNew.Text = txtUNOld.Text = "";
            btnSet.Enabled = txtPWOld.Enabled = txtUNOld.Enabled = cboxUTyCh.Enabled =
true;
            btnUpdate.Enabled = txtUNNew.Enabled = txtPWNew1.Enabled =
txtPWNew2.Enabled = false;

        End: if (db.con.State.Equals(ConnectionState.Open)) db.con.Close();
        }


    }
}
```