1. Introduction:

1.1 Synopsis

Title of the Project: “Library Automation System”

Abstract:

* The Project entitled Library Automation System which automates day to day Transaction of the Library.
* A System with process of Issue, Renew and Return of books, Staff details and Member/Student management with the usage of Database to store such details.
* A Search module with quick and advanced search.
* Stock and Vendor Management.
* Generation of Reports and management of reports based on specified criteria

Objectives:

* Computerize and Automate Library Management by replacing the Existing manual system in order to meet the necessary requirements.
* To create an Attractive UI which is easy to Understand and operate for the librarians.
* To Easily Manage Issue, Renew and Return of the Books and display the list of records, copies each members has.
* To better Manage Human resources such that there exists separate Login credentials for Administrator and Staff.
* To provide Quick and Advanced Search based on Keywords, Title, and Author-Name etc.
* To export reports as a document or Spreadsheet files.

Project Category:

* This software is a Client-side Database Application.
* This can also be categorized to as RDBMS application.

Tools to be used:

* Requirements Analysis Tools:
  + Questionnaires to the librarian , staff and users
    - Vendors and Publishers details.
    - Books and periodical purchase details.
    - Fine Payment for the delay of books.
    - Reports of the books purchased and issued.
* Testing Tools: Manual testing with real-time data.
  + Hardware Requirements:
    - Standard Input / Output devices
    - Intel Pentium or above
    - 512 MB RAM or above
    - 40 GB Disk space or above
  + Software Requirements:
    - Microsoft Windows XP or above.
* Programming Tools:
  + Frontend: Microsoft Visual C# with .NET framework using Visual Studio IDE.

*An Extension to Visual Programming title by Microsoft which provides web based and windows based application. This also features Inheritance, Overloading and Interfaces.*

* + Backend: Microsoft SQL Server 2005.

*SQL is similar to relational database management system (RDBMS). It can be made to run from Laptops to Multiprocessor Servers and operated by concurrent users.*

This project is a Client-side Database Application.

Structure of the Project:

1. Administration module
   1. Login
   2. Add User
   3. Edit User
   4. Delete User
2. Registration module
   1. Book Registration
   2. Member Registration
3. Book Catalog module
   * Book Entry
   * Book Edit / Delete
4. Circulation module
   * Book Transaction
     + Student Details
     + Member Details
     + Staff Details
5. Stock module
   * Book Stock
   * Periodical Stock
6. Search module
7. Vendor module
   * Vendor Details
8. Fine module
   * View - Fine
   * View – Student / Member Details
9. Purchase module
   * Purchase Order
     + Create
     + View
   * Purchase Bill Save
   * Automatic Stock Update
10. Report module
    * Book Transaction Report
    * Book Report
    * Purchase Report
    * Member details Report
    * Fine Report
11. Settings module
    * Change Password
    * Logout
12. Administration Module: This module checks for Valid User when the user enters his User-Name and Password.
    * Login: Login to Software with valid User-name and Password.
    * Add User: Add a User account, who will work with Software
    * Edit User: To Edit some User attributes and details
    * Delete User: Remove User Account.
13. Registration Module:
    * Book Registration: Allows Administrator / Staff to register the information about new books that is bought to the library.
    * Student Registration: Enables the Admin / Staff to register student details such as name, roll no, address etc and generate a Student-ID.
14. Catalog Module: For the Storage and management of the records of each title in the Library,

Functions Includes

* + Book Entry: Enter of the Book Title and corresponding details like Vendor from which the book was purchased from, Author name, Call no, and Publisher details. Also enables entries of Periodicals like Magazine and Journals.
  + Category Entry: Adds a Category to sort the book into a general Category.
  + Subject Entry: Adds a subject for the selected Category.

1. Circulation Module:To Track the Movement of Books / periodicals through Borrow, Renew and Return and generate notice and circulation fees.
   * Book Transactions: A Student / Member and Staff can borrow books from the library.
     + Student/Member Details: Students and Members can issue and then return books from the library. They should return books within their due date. Particular books are issued only if its status is available and not if status is not available. Students and Members can also renew a book. One Student can have only One book. Members can have constraints based on their membership.
     + Staff Members: Staff can issue books and return them back to the library. All the books that respective staff has, must be returned by the end of the last working day. A particular book can be issued if only if the status is available if not the book cannot be issued. Staff members get the highest privilege that can be given to members with additional benefits.
2. Stock Module: Automatically Update Stock based on Purchase Module and Billing Criteria.
   * Store Book Stock
   * Edit / Update Book Stock
   * Delete Book Stock
3. Search Module: To Search a Specific book and Periodical based on the Book Title and Author. Advanced Search includes search of book using Accession Number, Keywords, Year, Language,Section or Subjects.
4. Vendor Module: To monitor information about the supply of the books and periodicals.
   * Vendor Details : To register Vendor details like name, address, contact no, mail ID etc.
5. Purchase Module: A module to place orders and deals with bill entry.
   * Create Purchase Order.
     + Book Info with Number of copies to be sent.
   * Place Order to different vendors.
   * View Purchase Orders and display date-wise
   * Purchase Bill Entry: Once billed stock is automatically updated.
6. Report Module: A Module designed to view and generate reports based on different criteria. Generated reports can be exported to document and Spreadsheet formats.
   * Book Transaction report.
   * Members Report
   * Purchase Report
   * Fine Report
7. Fine Module: To Find Members with Fine, and Reset the Fine as of the Payment of the Fine.
   * View Student details
   * View Fine
8. Settings: A Module to manage the settings of Administrator and Users.
   * To Change password based on Securtiy Question.
   * To Reset Password and Security question

Limitations:

* Unavailability of Online Services.
* No Support for Optical Character Lookup.
* No Support for Bio-metric Login

Future Scope and Enhancement of the project:

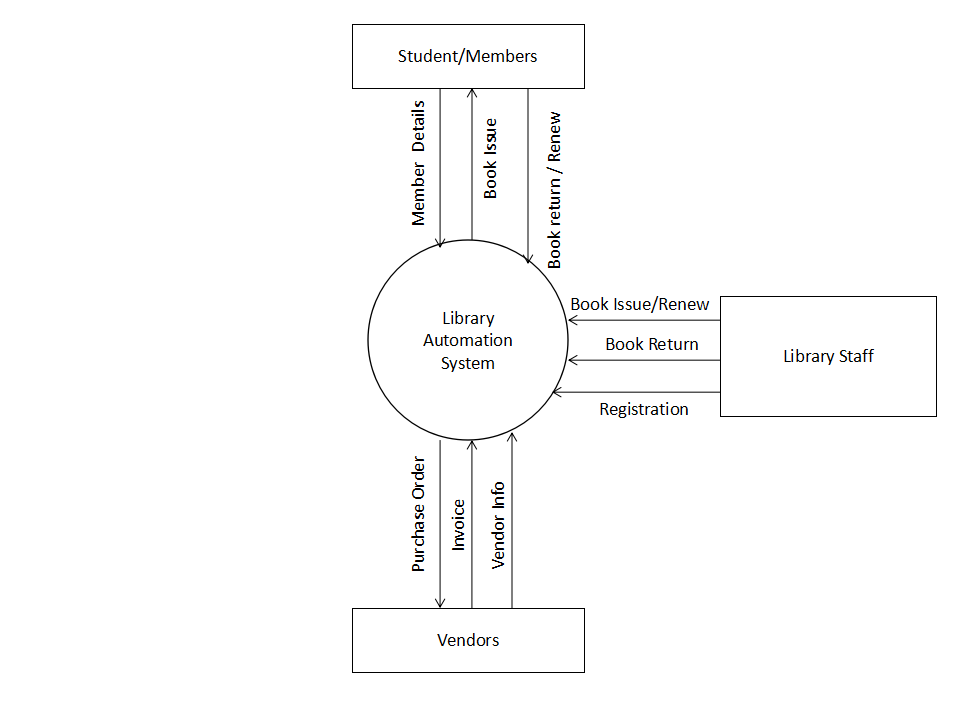
* Online uses such that it can be implemented as a digital Library.
* Optical Character lookup for Book ISBN vs. Database ISBN can be implemented.
* Biometric Login can be implemented with a proper interface using a sensor based recognition system.
* A Module can be implemented to convert existing database of book catalog, to convert and work with this Software.

2. Structure of the System Design:

2.1 Context Flow Diagram (CFD)

In Context Flow Diagram, the entire System is treated as a single process and all input, output, sinks and sources of the processes are shown .Using CFD design methodology DFE of the system is represented.

The environment in which (the context) the Software is used is depicted in CFD. The CFD shows the external entity acting on the software



Context Flow Diagram (Level - 0)

2.2 Data Flow Diagrams (DFD)

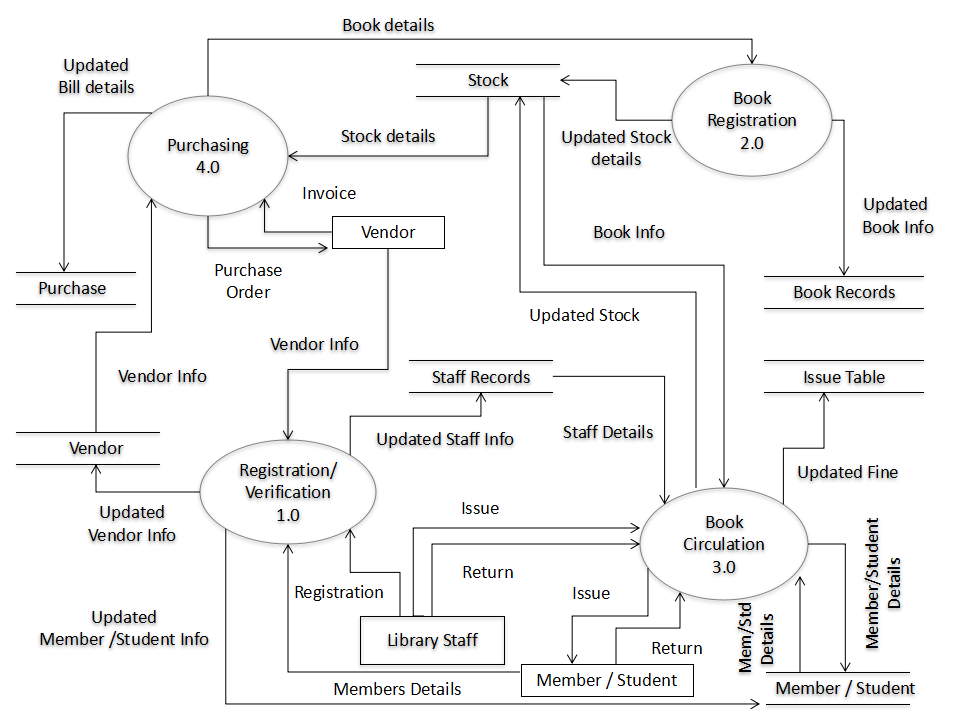
Data Flow Diagrams are also called Data Flow Graphs or Bubble Chart Which are commonly used during problem analysis. A DFD shows the flow of data through a System. It views a System as a function that transforms the inputs into desired outputs.

The processes in DFD are represented by named circles and data flows are represented by named arrows entering or leaving the bubbles.

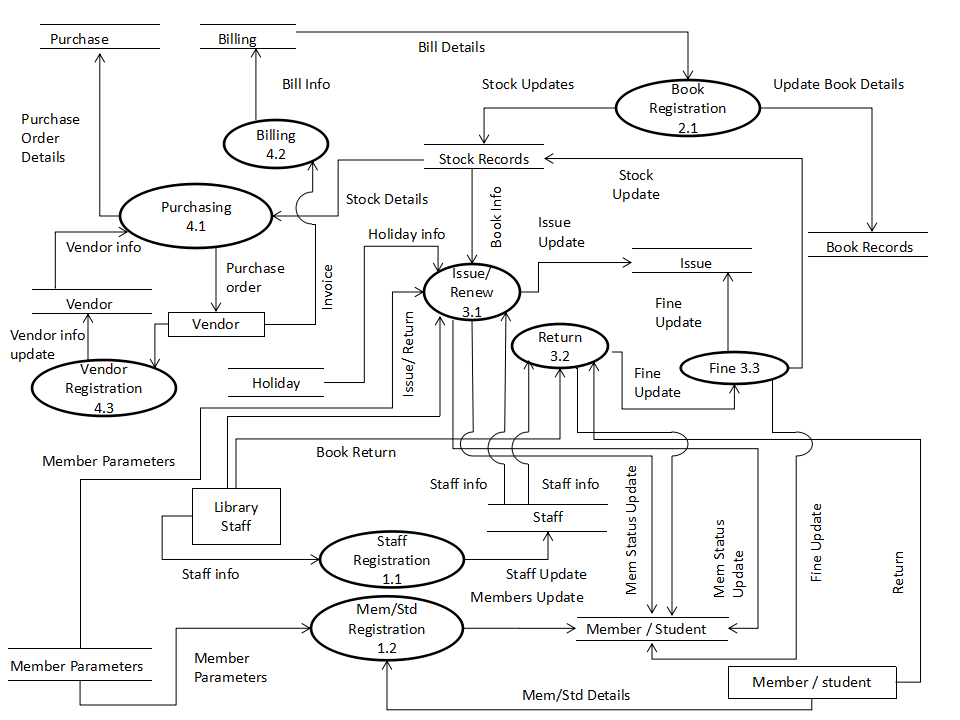
A rectangle represented a source or sinks and is a net originator or consumer of data, source or sink is typically outside the main system of study.

**DFD Notation**

|  |  |
| --- | --- |
| Notation | Description |
|  | A circle represents a process or transform that is applied to the data or control and changes it in some way. |
|  | A rectangle is used to represents an external entity that produces information for transformation by the software or receives information produced by the software. |
|  | An arrow represents one or more data items or data objects |
|  | The open box represents data-store to store information that is used by the software |



Data Flow Diagram (Level - 1)



Data Flow Diagram (Level - 2)

2.3 Entity Relationship Diagram :

In software engineering, an entity-relationship model (ER model for short) is an abstract and conceptual representation of data. Entity-relationship modelling is a database modelling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion. Diagrams created by this process are called entity-relationship diagrams or ER diagrams. The first stage of information system design uses these models during the requirements analysis to describe information needs or the type of information that is to be stored in a database.

The data modelling technique can be used to describe any ontology (i.e. an overview and classifications of used terms and their relationships) for a certain area of interest. In the case of the design of an information system that is based on a database, the conceptual data model is, at a later stage (usually called logical design), mapped to a logical data model, such as the relational model; this in turn is mapped to a physical model during physical design. Note that sometimes, both of these phases are referred to as "physical design".

An entity may be defined as a thing which is recognized as being capable of an independent existence and which can be uniquely identified. An entity is an abstraction from the complexities of some domain. When we speak of an entity we normally speak of some aspect of the real world which can be distinguished from other aspects of the real world.

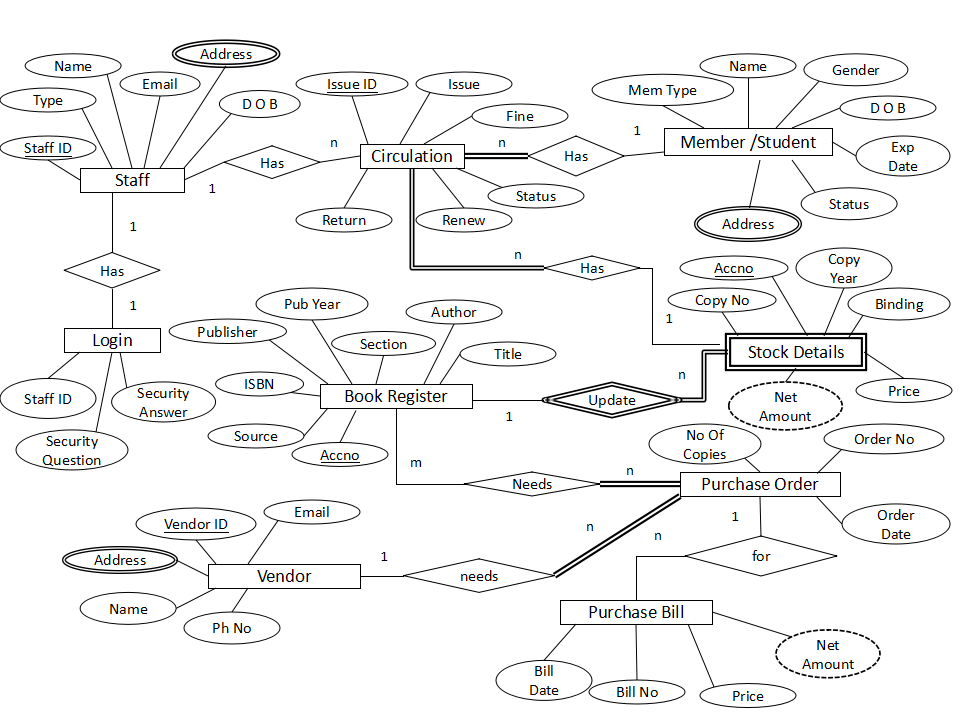
An entity may be a physical object such as a house or a car, an event such as a house sale or a car service, or a concept such as a customer transaction or order. Although the term entity is the one most commonly used, following Chen we should really distinguish between an entity and an entity-type. An entity-type is a category. An entity, strictly speaking, is an instance of a given entity-type. There are usually many instances of an entity-type. Because the term entity-type is somewhat cumbersome, most people tend to use the term entity as a synonym for this term.Entities can be thought of as nouns. Examples: a computer, an employee, a song, a mathematical theorem.

A relationship captures how entities are related to one another. Relationships can be thought of as verbs, linking two or more nouns. Examples: a known relationship between a company and a computer, a supervises relationship between an employee and a department, a performs relationship between an artist and a song, a proved relationship between a mathematician and a theorem.

The model's linguistic aspect described above is utilized in the declarative database query

**ER Diagram Notation**

|  |  |
| --- | --- |
| Symbols | Meaning |
|  | Entity |
|  | Weak Entity |
|  | Relationship |
|  | Identifying Attribute |
|  | Attribute |
|  | Key Attribute |
|  | Composite Attribute |
|  | DERIVED ATTRIBUTE |
| E2  R  E1 | TOTAL PARTICIPATION OF  E1 E2 IN R |
| R  E2  E1 | CARDINALITY RATIO 1:N FOR E1:E2 INR |
| R  E | STRUCTURAL CONSTRAINT (min,max)ON ARTICIPATION OF E IN R |



Entity Relationship Diagram

**Limitations:**

* Microsoft SQL Server has to be installed
* Microsoft .NET Framework 2.0 has to be installed.

2.4 DATABASE DIAGRAM

A database is inherent collection of data with some inherent meaning designed, built populated with data for a specific purpose. The following guidance have been followed during the database design.

1. Descriptive name for the tables, columns and index.
2. Distinct names for table and columns.
3. Proper data type for each column.

S-Designer:

S-designer is a database tool used foe entity relationship diagrams. It is basically a developmental tool, which helps the developer to create relations between table or entities easily.

Conceptual Data Model:

with S-designer process starts on the conceptual where level where there is no need to consider the details of actual physical implementation.

A CDM represents the overall logical structure of a database,which is independent of any software or data storage. A conceptual model often contains objects not yet implemented in the physical databse.It gives formal representation of the data needed to run an enterprise or a business activity.

CDM Roles:

CDM plays very important role in the design of database.

* Represents the organization of the data in a graphic format.
* Verify the validity of data design.
* Generate the PDM, Which specific the physical implementation of the database.

Building conceptual data model is the building ER-model, which, mainly comprises of:

* Entities: An entity represents an object defined within the information system.
* Entity attributes: Attributes are elementary pieces of information attached to an entity.
* Entity relationship: A relationship is a named collection or associated between entities.

Database Diagram:



2.5 Data Dictionary :

1. \_CategoryTable = @list

list={A-Z|a-z|0-9}

1. \_CurrencyTable = @list

list={A-Z|a-z|0-9}

1. \_DeptTable = @list

list={A-Z|a-z|0-9}

1. \_LanguageTable = @list

list={A-Z|a-z|0-9}

1. \_SourceTable = @list

list={A-Z|a-z|0-9}

1. \_SubjectTable = @list

list={A-Z|a-z|0-9}

1. AccRegIssues=@issueid+accno+section+type+title+subtitle+author+editor+volume+series+isbn+language+keyword+remarks+subject+subject2+callno+classno+pubyear+place+publisher

Issueid={0-9}

Accno={A-Z|a-z|0-9}

section={General| UG Library | PG Library}

type={Book|Book + CD|Magazine|Magazine + CD|Journal|Journal + CD}

title={legal characters}

subtitle={legal characters}

author={A-Z|a-z}

editor={A-Z|a-z}

volume={A-Z|a-z|0-9}

series={legal characters}

isbn={A-Z|a-z|0-9}

language={A-Z|a-z}

keyword={legal characters}

remarks={A-Z|a-z}

subject={A-Z|a-z}

subject2={A-Z|a-z}

callno={legal characters}

classno={legal characters}

Pubyear={0-9}

Place={legal characters}

Publisher={legal characters}

1. AccRegTable=@accno+section+type+title+subtitle+author+editor+volume+series+isbn+language+keyword+remarks+subject+subject2+callno+classno+pubyear+place+publisher

Accno={A-Z|a-z|0-9}

section={General| UG Library | PG Library}

type={Book|Book + CD|Magazine|Magazine + CD|Journal|Journal + CD}

title={legal characters}

subtitle={legal characters}

author={A-Z|a-z}

editor={A-Z|a-z}

volume={A-Z|a-z|0-9}

series={legal characters}

isbn={A-Z|a-z|0-9}

language={A-Z|a-z}

keyword={legal characters}

remarks={A-Z|a-z}

subject={A-Z|a-z}

subject2={A-Z|a-z}

callno={legal characters}

classno={legal characters}

Pubyear={0-9}

Place={legal characters}

Publisher={legal characters}

1. CreateOrderDummy=orderno+orderdate+accno+vendorid+name+email+address+phoneno+title+author+publisher+noofcopies

Orderno={0-9}

Orderdate={DD-MM-YYYY}

accno={A-Z|a-z|0-9}

vendorid={legal characters}

Name={A-Z|a-z}

email={A-Z|a-z|0-9|\_+@+.}

address={legal characters}

phoneno={0-9}

title={legal characters}

author={A-Z|a-z}

publisher={legal characters}

noofcopies={0-9}

1. CreateOrderTable=orderno+orderdate+accno+vendorid+name+email+address+phoneno+title+author+publisher+noofcopies

Orderno={0-9}

Orderdate={DD-MM-YYYY}

accno={A-Z|a-z|0-9}

vendorid={legal characters}

Name={A-Z|a-z}

Email={A-Z|a-z|0-9|\_+@+.}

address={legal characters}

phoneno={0-9}

title={legal characters}

author={A-Z|a-z}

publisher={legal characters}

noofcopies={0-9}

1. GenerateBillTable=billno+billdate+orderno+orderdate+vendorid+name+accno+title+author+publisher+noofcopies+price+totamt

billno={0-9}

billdate={DD-MM-YYYY}

orderno={0-9}

orderdate={DD-MM-YYYY}

vendorid={legal characters}

name={A-Z|a-z}

accno={A-Z|a-z|0-9}

title={legal characters}

author={A-Z|a-z}

publisher={legal characters}

noofcopies={0-9}

price={0-9}

Totamt={0-9}

1. HolidayTable=@date+reason

date={DD-MM-YYYY}

Reason={legal characters}

1. IssueTable=@issueid+memberid+accno+copyno+issuedate+duedate+returndate+fine+status+renewdate+staffIDIssue+staffIDRenew+staffIDReturn

issueid={0-9}

memberid={A-Z|a-z|0-9}

accno={A-Z|a-z|0-9}

copyno={0-9}

issuedate={DD-MM-YYYY}

duedate={DD-MM-YYYY}

returndate={DD-MM-YYYY}

fine={0-9}

Status={Issued | Returned}

renewdate={DD-MM-YYYY}

staffIDIssue={A-Z|a-z|0-9}

staffIDRenew={A-Z|a-z|0-9}

staffIDReturn={A-Z|a-z|0-9}

1. LoginTable=@staffid+password+usertype+resetpassword+sQuestion+sAnswer

staffid={A-Z|a-z|0-9}

password={legal characters}

usertype={A-Z|a-z}

resetpassword={A-Z|a-z}

sQuestion={legal characters}

sAnswer={legal characters}

1. MemberParameters=@memtype+mtypeidentifier+maxissuedays+fineperday

memtype={A-Z|a-z}

mtypeidentifier={A-Z|a-z}

maxissuedays={0-9}

Fineperday={0-9}

1. MembersIssues=@issueid+membertype+memberid+firstname+lastname+course+branch+dept+dob+gender+status+subsdate+expdate+fathername+occupation+phoneno+email+address1+address2+pincode+state+fine

issueid={0-9}

membertype={A-Z|a-z}

memberid={A-Z|a-z|0-9}

firstname={A-Z|a-z}

lastname={A-Z|a-z}

course={A-Z|a-z}

branch={A-Z|a-z}

dept={A-Z|a-z}

dob={DD-MM-YYYY}

Gender={M|F}

Status={Active | Inactive | With Issue}

subsdate={DD-MM-YYYY}

expdate={DD-MM-YYYY}

fathername={A-Z|a-z}

occupation={A-Z|a-z}

phoneno={0-9}

email={A-Z|a-z|0-9|\_+@+.}

address1={legal Characters}

address2={legal Characters}

Pincode={0-9}

state={A-Z|a-z}

Fine={0-9}

1. MembersTable=membertype+@memberid+firstname+lastname+course+branch+dept+dob+gender+status+subsdate+expdate+fathername+occupation+phoneno+email+address1+address2+pincode+state+fine

membertype={A-Z|a-z}

memberid={A-Z|a-z|0-9}

firstname={A-Z|a-z}

lastname={A-Z|a-z}

course={A-Z|a-z}

branch={A-Z|a-z}

dept={A-Z|a-z}

dob={DD-MM-YYYY}

Gender={M|F}

Status={Active | Inactive | With Issue}

subsdate={DD-MM-YYYY}

expdate={DD-MM-YYYY}

fathername={A-Z|a-z}

occupation={A-Z|a-z}

phoneno={0-9}

email={A-Z|a-z|0-9|\_+@+.}

address1={legal Characters}

address2={legal Characters}

Pincode={0-9}

state={A-Z|a-z}

Fine={0-9}

1. StaffTable=@staffid+stafftype+fname+lname+gender+dob+address1+address2+pin+state+email+phoneno+doj+status

staffid={A-Z|a-z|0-9}

Stafftype={Administrator | Library Staff}

Fname={A-Z|a-z}

lname={A-Z|a-z}

Gender={M|F}

dob={DD-MM-YYYY}

address1={legal Characters}

address2={legal Characters}

pin={0-9}

state={A-Z|a-z}

email={A-Z|a-z|0-9|\_+@+.}

phoneno={0-9}

doj={DD-MM-YYYY}

Status={A-Z|a-z}

1. StockIssues=@issueid+accno+copyno+vendor+source+currency+dept+edition+billno+billdate+discount+status+category+price+netcost+pages+location+binding+copyyear

Issueid={0-9}

accno={A-Z|a-z|0-9}

Copyno={0-9}

Vendor={legal Characters}

Source={A-Z|a-z}

Currency={legal Characters}

dept={A-Z|a-z}

edition={A-Z|a-z|0-9}

Billno={0-9}

Billdate={DD-MM-YYYY}

discount={0-9}

Status={Available|Unavailable}

category={A-Z|a-z}

price={0-9}

netcost={0-9}

pages={0-9}

location={A-Z|a-z}

Binding={paperback|hardcover|unidentifiable}

Copyyear={0-9}

1. StockTable=@accno+@copyno+vendor+source+currency+dept+edition+billno+billdate+discount+status+category+price+netcost+pages+location+binding+copyyear

accno={A-Z|a-z|0-9}

Copyno={0-9}

Vendor={legal Characters}

Source={A-Z|a-z}

Currency={legal Characters}

dept={A-Z|a-z}

edition={A-Z|a-z|0-9}

Billno={0-9}

Billdate={DD-MM-YYYY}

discount={0-9}

Status={Available|Unavailable}

category={A-Z|a-z}

price={0-9}

netcost={0-9}

pages={0-9}

location={A-Z|a-z}

Binding={paperback|hardcover|unidentifiable}

Copyyear={0-9}

1. VendorTable=@vendorid+name+address+phoneno+email

vendorid={A-Z|a-z}

Name={A-Z|a-z}

address={legal Characters}

phoneno={0-9}

email={A-Z|a-z|0-9|\_+@+.}

2.6 Table Structure :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table Name | Column Name | Data type | Constraint type | Allow Null |
|  |  |  |  |  |
| \_CategoryTable | list | varchar(50) | PRIMARY KEY | NO |
| \_CurrencyTable | list | varchar(50) | PRIMARY KEY | NO |
| \_DeptTable | list | varchar(50) | PRIMARY KEY | NO |
| \_LanguageTable | list | varchar(50) | PRIMARY KEY | NO |
| \_SourceTable | list | varchar(50) | PRIMARY KEY | NO |
| \_SubjectTable | list | varchar(50) | PRIMARY KEY | NO |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AccRegIssues | issueid | int | FOREIGN KEY | NO |
| AccRegIssues | accno | varchar(50) |  | NO |
| AccRegIssues | section | varchar(50) |  | YES |
| AccRegIssues | type | varchar(50) |  | YES |
| AccRegIssues | title | varchar(50) |  | YES |
| AccRegIssues | subtitle | varchar(50) |  | YES |
| AccRegIssues | author | varchar(50) |  | YES |
| AccRegIssues | editor | varchar(50) |  | YES |
| AccRegIssues | volume | varchar(50) |  | YES |
| AccRegIssues | series | varchar(50) |  | YES |
| AccRegIssues | isbn | varchar(50) |  | YES |
| AccRegIssues | language | varchar(50) |  | YES |
| AccRegIssues | keyword | varchar(50) |  | YES |
| AccRegIssues | remarks | varchar(50) |  | YES |
| AccRegIssues | subject | varchar(50) |  | YES |
| AccRegIssues | subject2 | varchar(50) |  | YES |
| AccRegIssues | callno | varchar(50) |  | YES |
| AccRegIssues | classno | varchar(50) |  | YES |
| AccRegIssues | pubyear | varchar(50) |  | YES |
| AccRegIssues | place | varchar(50) |  | YES |
| AccRegIssues | publisher | varchar(50) |  | YES |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AccRegTable | accno | varchar(50) | PRIMARY KEY | NO |
| AccRegTable | section | varchar(50) |  | YES |
| AccRegTable | type | varchar(50) |  | YES |
| AccRegTable | title | varchar(50) |  | YES |
| AccRegTable | subtitle | varchar(50) |  | YES |
| AccRegTable | author | varchar(50) |  | YES |
| AccRegTable | editor | varchar(50) |  | YES |
| AccRegTable | volume | varchar(50) |  | YES |
| AccRegTable | series | varchar(50) |  | YES |
| AccRegTable | isbn | varchar(50) |  | YES |
| AccRegTable | language | varchar(50) |  | YES |
| AccRegTable | keyword | varchar(50) |  | YES |
| AccRegTable | remarks | varchar(50) |  | YES |
| AccRegTable | subject | varchar(50) |  | YES |
| AccRegTable | subject2 | varchar(50) |  | YES |
| AccRegTable | callno | varchar(50) |  | YES |
| AccRegTable | classno | varchar(50) |  | YES |
| AccRegTable | pubyear | varchar(50) |  | YES |
| AccRegTable | place | varchar(50) |  | YES |
| AccRegTable | publisher | varchar(50) |  | YES |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CreateOrderDummy | orderno | int |  | NO |
| CreateOrderDummy | orderdate | datetime |  | YES |
| CreateOrderDummy | accno | varchar(50) |  | YES |
| CreateOrderDummy | vendorid | varchar(50) |  | YES |
| CreateOrderDummy | name | varchar(50) |  | YES |
| CreateOrderDummy | email | varchar(50) |  | YES |
| CreateOrderDummy | address | varchar(50) |  | YES |
| CreateOrderDummy | phoneno | varchar(50) |  | YES |
| CreateOrderDummy | title | varchar(50) |  | YES |
| CreateOrderDummy | author | varchar(50) |  | YES |
| CreateOrderDummy | publisher | varchar(50) |  | YES |
| CreateOrderDummy | noofcopies | int |  | YES |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CreateOrderTable | orderno | int |  | NO |
| CreateOrderTable | orderdate | datetime |  | YES |
| CreateOrderTable | accno | varchar(50) |  | YES |
| CreateOrderTable | vendorid | varchar(50) |  | YES |
| CreateOrderTable | name | varchar(50) |  | YES |
| CreateOrderTable | email | varchar(50) |  | YES |
| CreateOrderTable | address | varchar(50) |  | YES |
| CreateOrderTable | phoneno | varchar(50) |  | YES |
| CreateOrderTable | title | varchar(50) |  | YES |
| CreateOrderTable | author | varchar(50) |  | YES |
| CreateOrderTable | publisher | varchar(50) |  | YES |
| CreateOrderTable | noofcopies | int |  | YES |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GenerateBillTable | billno | nvarchar(50) |  | YES |
| GenerateBillTable | billdate | nvarchar(50) |  | YES |
| GenerateBillTable | orderno | nvarchar(50) |  | YES |
| GenerateBillTable | orderdate | nvarchar(50) |  | YES |
| GenerateBillTable | vendorid | nvarchar(50) |  | YES |
| GenerateBillTable | name | nvarchar(50) |  | YES |
| GenerateBillTable | accno | nvarchar(50) |  | YES |
| GenerateBillTable | title | nvarchar(50) |  | YES |
| GenerateBillTable | author | nvarchar(50) |  | YES |
| GenerateBillTable | publisher | nvarchar(50) |  | YES |
| GenerateBillTable | noofcopies | nvarchar(50) |  | YES |
| GenerateBillTable | price | nvarchar(50) |  | YES |
| GenerateBillTable | totamt | nvarchar(50) |  | YES |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| HolidayTable | date | datetime | PRIMARY KEY | NO |
| HolidayTable | reason | varchar(50) |  | YES |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IssueTable | issueid | int | PRIMARY KEY | NO |
| IssueTable | memberid | varchar(50) |  | NO |
| IssueTable | accno | varchar(50) |  | NO |
| IssueTable | copyno | varchar(50) |  | NO |
| IssueTable | issuedate | datetime |  | NO |
| IssueTable | duedate | datetime |  | NO |
| IssueTable | returndate | datetime |  | YES |
| IssueTable | fine | int |  | YES |
| IssueTable | status | varchar(50) |  | NO |
| IssueTable | renewdate | datetime |  | YES |
| IssueTable | staffIDIssue | varchar(50) |  | YES |
| IssueTable | staffIDRenew | varchar(50) |  | YES |
| IssueTable | staffIDReturn | varchar(50) |  | YES |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| LoginTable | staffid | varchar(50) | PRIMARY KEY | NO |
| LoginTable | password | varbinary |  | YES |
| LoginTable | usertype | varchar(50) |  | YES |
| LoginTable | resetpassword | varchar(50) |  | YES |
| LoginTable | sQuestion | varbinary |  | YES |
| LoginTable | sAnswer | varbinary |  | YES |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MemberParameters | memtype | varchar(50) | PRIMARY KEY | NO |
| MemberParameters | mtypeidentifier | varchar(50) |  | YES |
| MemberParameters | maxissuedays | int |  | NO |
| MemberParameters | fineperday | int |  | NO |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MembersIssues | issueid | int | FOREIGN KEY | NO |
| MembersIssues | membertype | varchar(50) |  | NO |
| MembersIssues | memberid | varchar(50) |  | NO |
| MembersIssues | firstname | varchar(50) |  | NO |
| MembersIssues | lastname | varchar(50) |  | NO |
| MembersIssues | course | varchar(50) |  | YES |
| MembersIssues | branch | varchar(50) |  | YES |
| MembersIssues | dept | varchar(50) |  | YES |
| MembersIssues | dob | varchar(50) |  | NO |
| MembersIssues | gender | varchar(50) |  | NO |
| MembersIssues | status | varchar(50) |  | NO |
| MembersIssues | subsdate | datetime |  | NO |
| MembersIssues | expdate | datetime |  | NO |
| MembersIssues | fathername | varchar(50) |  | YES |
| MembersIssues | occupation | varchar(50) |  | YES |
| MembersIssues | phoneno | varchar(50) |  | NO |
| MembersIssues | email | varchar(50) |  | NO |
| MembersIssues | address1 | varchar(50) |  | NO |
| MembersIssues | address2 | varchar(50) |  | YES |
| MembersIssues | pincode | varchar(50) |  | YES |
| MembersIssues | state | varchar(50) |  | YES |
| MembersIssues | fine | int |  | YES |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MembersTable | membertype | varchar(50) |  | NO |
| MembersTable | memberid | varchar(50) | PRIMARY KEY | NO |
| MembersTable | firstname | varchar(50) |  | NO |
| MembersTable | lastname | varchar(50) |  | NO |
| MembersTable | course | varchar(50) |  | YES |
| MembersTable | branch | varchar(50) |  | YES |
| MembersTable | dept | varchar(50) |  | YES |
| MembersTable | dob | varchar(50) |  | NO |
| MembersTable | gender | varchar(50) |  | NO |
| MembersTable | status | varchar(50) |  | NO |
| MembersTable | subsdate | datetime |  | NO |
| MembersTable | expdate | datetime |  | NO |
| MembersTable | fathername | varchar(50) |  | YES |
| MembersTable | occupation | varchar(50) |  | YES |
| MembersTable | phoneno | varchar(50) |  | NO |
| MembersTable | email | varchar(50) |  | NO |
| MembersTable | address1 | varchar(50) |  | NO |
| MembersTable | address2 | varchar(50) |  | YES |
| MembersTable | pincode | varchar(50) |  | YES |
| MembersTable | state | varchar(50) |  | YES |
| MembersTable | fine | int |  | YES |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| StaffTable | staffid | varchar(50) | PRIMARY KEY | NO |
| StaffTable | stafftype | varchar(50) |  | NO |
| StaffTable | fname | varchar(50) |  | NO |
| StaffTable | lname | varchar(50) |  | YES |
| StaffTable | gender | varchar(50) |  | NO |
| StaffTable | dob | varchar(50) |  | NO |
| StaffTable | address1 | varchar(50) |  | NO |
| StaffTable | address2 | varchar(50) |  | YES |
| StaffTable | pin | varchar(50) |  | YES |
| StaffTable | state | varchar(50) |  | YES |
| StaffTable | email | varchar(50) |  | NO |
| StaffTable | phoneno | varchar(50) |  | NO |
| StaffTable | doj | varchar(50) |  | NO |
| StaffTable | status | varchar(50) |  | NO |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| StockIssues | issueid | int | FOREIGN KEY | NO |
| StockIssues | accno | varchar(50) |  | NO |
| StockIssues | copyno | int |  | NO |
| StockIssues | vendor | varchar(50) |  | YES |
| StockIssues | source | varchar(50) |  | YES |
| StockIssues | currency | varchar(50) |  | YES |
| StockIssues | dept | varchar(50) |  | YES |
| StockIssues | edition | varchar(50) |  | YES |
| StockIssues | billno | varchar(50) |  | YES |
| StockIssues | billdate | datetime |  | YES |
| StockIssues | discount | varchar(50) |  | YES |
| StockIssues | status | varchar(50) |  | YES |
| StockIssues | category | varchar(50) |  | YES |
| StockIssues | price | varchar(50) |  | YES |
| StockIssues | netcost | varchar(50) |  | YES |
| StockIssues | pages | varchar(50) |  | YES |
| StockIssues | location | varchar(50) |  | YES |
| StockIssues | binding | varchar(50) |  | YES |
| StockIssues | copyyear | varchar(50) |  | YES |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| StockTable | accno | varchar(50) | FOREIGN KEY | NO |
| StockTable | copyno | int |  | NO |
| StockTable | vendor | varchar(50) |  | YES |
| StockTable | source | varchar(50) |  | YES |
| StockTable | currency | varchar(50) |  | YES |
| StockTable | dept | varchar(50) |  | YES |
| StockTable | edition | varchar(50) |  | YES |
| StockTable | billno | varchar(50) |  | YES |
| StockTable | billdate | datetime |  | YES |
| StockTable | discount | varchar(50) |  | YES |
| StockTable | status | varchar(50) |  | YES |
| StockTable | category | varchar(50) |  | YES |
| StockTable | price | varchar(50) |  | YES |
| StockTable | netcost | varchar(50) |  | YES |
| StockTable | pages | varchar(50) |  | YES |
| StockTable | location | varchar(50) |  | YES |
| StockTable | binding | varchar(50) |  | YES |
| StockTable | copyyear | varchar(50) |  | YES |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| VendorTable | vendorid | varchar(50) | PRIMARY KEY | NO |
| VendorTable | name | varchar(50) |  | NO |
| VendorTable | address | varchar(50) |  | NO |
| VendorTable | phoneno | varchar(50) |  | NO |
| VendorTable | email | varchar(50) |  | NO |

3 Source Code:

00 Connect.cs:

using System;

using System.Collections.Generic;

using System.Text;

using System.Windows.Forms;

using System.Data.Sql;

using System.Data.SqlClient;

namespace Library\_Auto

{

classConnect

{

publicSqlCommand cmd = newSqlCommand();

publicSqlConnection cnn = newSqlConnection();

public Connect()

{

try

{

//cnn.ConnectionString = "data source=mgm-server; initial catalog=bgroup11; integrated security=true";

cnn.ConnectionString = "data source=ONRA\\SQLEXPRESS; initial catalog=bgroup11; integrated security=true";

cnn.Open();

cmd.Connection = cnn;

}

catch (SqlException)

{

string msg1, msg2;

msg1 = "An Error Occured!\nCould not create a connection to the Server, \nPlease check that you have an Active Internet Connection.";

msg2 = "\nIf the problem persists, restart the Software.";

MessageBox.Show(msg1 + msg2, "SQL Exception: Database Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

01 Search:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace Library\_Auto

{

publicpartialclassSearch\_01 : Form

{

publicAdministration\_03 adminInstance;

publicbool isLoginComplete = true;

Connect c = newConnect();

SqlDataAdapter adp = newSqlDataAdapter();

DataTable dt = newDataTable();

public Search\_01(){InitializeComponent();}

privatevoid Search\_01\_Load(object sender, EventArgs e){if (isLoginComplete){lblOPAC.Text = "Search";}}

protectedoverridevoid OnFormClosing(FormClosingEventArgs e)

{

base.OnFormClosing(e);

switch (e.CloseReason)

{

caseCloseReason.WindowsShutDown:

caseCloseReason.TaskManagerClosing:

return;

default: break;

}

if (adminInstance != null)

adminInstance.Show();

}

privatevoid btnClear\_Click(object sender, EventArgs e)

{

dt.Clear();

foreach (Control ctrl in grpSimple.Controls)

{

if (ctrl isTextBox){(ctrl asTextBox).Clear();}

if (ctrl isComboBox)

{

(ctrl asComboBox).SelectedItem = null;

(ctrl asComboBox).Text = "";

}

}

foreach (Control ctrl in grpAdv.Controls)

{

if (ctrl isTextBox)

{

(ctrl asTextBox).Clear();

}

if (ctrl isComboBox)

{

(ctrl asComboBox).SelectedItem = null;

(ctrl asComboBox).Text = "";

}

}

foreach (Control ctrl in grpTxtFormat.Controls)

{

if (ctrl isRadioButton)

{

(ctrl asRadioButton).Checked = false;

}

}

radoAll.Checked = true;

radoFull.Checked = true;

}

privatevoid btnSearch\_Click(object sender, EventArgs e)

{

dt.Clear();

dataGridView1.Refresh();

if (cbAccno.Checked)

{

c.cmd.CommandText = "select \* from AccRegTable where accno='" + txtAccno.Text + "'";

FillGrid();

return;

}

if (cbKeyword.Checked)

{

string keyword = (txtKeyword.Text == "") ? null : SearchType(txtKeyword.Text);

if (keyword != null)

{

c.cmd.CommandText = "select \* from AccRegTable where keyword like '" + keyword + "'" + SetBookType();

FillGrid();

return;

}

elsereturn;

}

if (cbYear.Checked)

{

string year = (txtYear.Text == "") ? null : txtYear.Text;

if (year != null)

{

c.cmd.CommandText = "select \* from AccRegTable where pubyear like '" + year + "'" + SetBookType();

FillGrid();

return;

}

elsereturn;

}

string title, author;

title = (txtTitle1.Text == "") ? null : SearchType(txtTitle1.Text);

author = (txtAuthor1.Text == "") ? null : SearchType(txtAuthor1.Text);

if (title == null)

{

c.cmd.CommandText = "select \* from AccRegTable where author like '" + author + "'" + SetBookType();

FillGrid();

return;

}

elseif (author == null)

{

c.cmd.CommandText = "select \* from AccRegTable where title like '" + title + "'" + SetBookType();

FillGrid();

return;

}

if (ddlLogical.SelectedItem == null)

{

return;

}

c.cmd.CommandText = "select \* from AccRegTable where title like '" + title + "' " + ddlLogical.SelectedItem + " author like '" + author + "'" + SetBookType();

FillGrid();

return;

}

privatestring SetBookType()

{

if (radoBook.Checked) return" and (type ='Book' or type ='Book + CD')";

if (radoMagazine.Checked) return" and (type = 'Magazine' or type = 'Magazine + CD')";

if (radoJournal.Checked) return" and (type = 'Journal' or type = 'Journal + CD')";

if (radoAll.Checked) return"";

return"";

}

privatestring SearchType(string str)

{

if (radoLetter.Checked)

{

str = str[0] + "%";

return str;

}

if (radoWordBeg.Checked)

{

try

{

str = str.Substring(0, str.IndexOf(""));

str = str + "%";

return str;

}

catch (ArgumentOutOfRangeException)

{

str = str + "%";

return str;

}

}

if (radoFull.Checked)

{

return str;

}

return str;

}

privatevoid FillGrid()

{

//Fills The grid based on the Command Text

adp.SelectCommand = c.cmd;

adp.Fill(dt);

dataGridView1.DataSource = dt;

}

privatevoid btnAll\_Click(object sender, EventArgs e)

{

c.cmd.CommandText = "select \* from AccRegTable";

adp.SelectCommand = c.cmd;

dt.Clear();

adp.Fill(dt);

dataGridView1.DataSource = dt;

}

privatevoid cbAdvSearch\_CheckedChanged(object sender, EventArgs e)

{

if (cbAdvSearch.Checked)

{

grpAdv.Enabled = true;

grpSimple.Enabled = false;

}

else

{

grpAdv.Enabled = false;

grpSimple.Enabled = true;

foreach (Control rados in grpAdv.Controls)

{

if (rados isRadioButton)

{

(rados asRadioButton).Checked = false;

}

}

}

}

privatevoid cbAccno\_CheckedChanged(object sender, EventArgs e)

{

dt.Clear();

if (cbAccno.Checked)

{

txtAccno.Enabled = true;

}

else

{

txtAccno.Enabled = false;

txtAccno.Clear();

}

}

privatevoid chSection\_CheckedChanged(object sender, EventArgs e)

{

dt.Clear();

if (cbSection.Checked)

{

ddlSection.Enabled = true;

}

else

{

ddlSection.Enabled = false;

ddlSection.SelectedItem = null;

}

}

privatevoid ddlSection\_SelectionChangeCommitted(object sender, EventArgs e)

{

dt.Clear();

c.cmd.CommandText = "select \* from AccRegTable where section ='" + ddlSection.SelectedItem.ToString() + "'" + SetBookType();

FillGrid();

}

privatevoid cbLanguage\_CheckedChanged(object sender, EventArgs e)

{

dt.Clear();

if (cbLanguage.Checked)

{

ddlLanguage.Enabled = true;

}

else

{

ddlLanguage.Enabled = false;

ddlLanguage.SelectedItem = null;

}

DdlUpdate("select language from AccRegTable", ddlLanguage);

}

privatevoid ddlLanguage\_SelectionChangeCommitted(object sender, EventArgs e)

{

dt.Clear();

c.cmd.CommandText = "select \* from AccRegTable where language ='" + ddlLanguage.SelectedItem.ToString() + "'" + SetBookType();

FillGrid();

}

privatevoid cbSubject\_CheckedChanged(object sender, EventArgs e)

{

dt.Clear();

if (cbSubject.Checked)

{

ddlSubject.Enabled = true;

}

else

{

ddlSubject.Enabled = false;

ddlSubject.SelectedItem = null;

}

DdlUpdate("select subject from AccRegTable", ddlSubject);

}

privatevoid ddlSubject\_SelectionChangeCommitted(object sender, EventArgs e)

{

dt.Clear();

c.cmd.CommandText = "select \* from AccRegTable where subject ='" + ddlSubject.SelectedItem.ToString() + "'" + SetBookType();

FillGrid();

}

privatevoid DdlUpdate(string cmdText, ComboBox ddlToFill)

{

DataTable dtDummy = newDataTable();

string itemToAdd = "";

try

{

ddlToFill.Items.Clear();

c.cmd.CommandText = cmdText;

adp.SelectCommand = c.cmd;

adp.Fill(dtDummy);

for (int i = 0; i < dtDummy.Rows.Count; i++)

{

itemToAdd = "";

itemToAdd += dtDummy.Rows[i].ItemArray[0];

if (!ddlToFill.Items.Contains(itemToAdd))

{

ddlToFill.Items.Add(itemToAdd);

}

}

}

catch (Exception ex)

{

if (adminInstance != null)

{

adminInstance.errObj.txtException.Text = ex.ToString();

adminInstance.errObj.Show();

}

}

}

privatevoid cbKeyword\_CheckedChanged(object sender, EventArgs e)

{

dt.Clear();

if (cbKeyword.Checked){txtKeyword.Enabled = true;}

else

{

txtKeyword.Enabled = false;

txtKeyword.Clear();

}

}

privatevoid cbYear\_CheckedChanged(object sender, EventArgs e)

{

dt.Clear();

if (cbYear.Checked){txtYear.Enabled = true;}

Else{txtYear.Enabled = false;txtYear.Clear();}

}

}

}

02 LOGIN:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Reflection;

using Microsoft.VisualBasic;

namespace Library\_Auto

{

publicpartialclassLogin\_02 : Form

{

publicstaticAdministration\_03 adminMaster = newAdministration\_03();

Connect c = newConnect();

SqlDataAdapter adp = newSqlDataAdapter();

DataTable dt = newDataTable();

public Login\_02()

{

InitializeComponent();

}

privatevoid btnLogin\_Click(object sender, EventArgs e)

{

adminMaster.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

try

{

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

c.cmd.CommandText = "select count(\*) from LoginTable where staffid='" + txtUsername.Text + "'";

if ((int)c.cmd.ExecuteScalar() <= 0)

{

txtUsername.Focus();

lblError.Text = "Username not found!";

Interaction.Beep();

return;

}

else

{

c.cmd.CommandText = "select resetpassword from LoginTable where staffid=@staffid";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@staffid", SqlDbType.VarChar).Value = txtUsername.Text;

string resetpassword = (string) c.cmd.ExecuteScalar();

switch (resetpassword)

{

case"Reset\_Full":

Login\_CP\_02b objCPFull = newLogin\_CP\_02b();

objCPFull.state = "Reset\_Full";

objCPFull.txtUsername.Text = txtUsername.Text;

objCPFull.Show();

break;

case"Reset\_Password":

Login\_CP\_02b objChangePassword = newLogin\_CP\_02b();

objChangePassword.state = "Reset\_Password";

objChangePassword.txtUsername.Text = txtUsername.Text;

objChangePassword.Show();

break;

case"Not\_Required":

c.cmd.CommandText = "select count(\*) from LoginTable where staffid=@staffid and password=@password";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@staffid", SqlDbType.VarChar).Value = txtUsername.Text;

c.cmd.Parameters.Add("@password", SqlDbType.VarBinary).Value = ConvertToByteArray(txtpword.Text);

if ((int)c.cmd.ExecuteScalar() > 0)

{

this.Hide();

adminMaster.staffID = txtUsername.Text;

adminMaster.Show();

adminMaster.Closed += newEventHandler(adminMaster\_Closed);

}

else

{

txtpword.Clear();

lblError.Text = "Wrong Password!";

}

break;

default: MessageBox.Show(""); break;

}

}

}

catch (Exception ex)

{

this.Hide();

adminMaster.errObj.txtException.Text += ex.ToString();

adminMaster.errObj.Show();

}

Finally { c.cnn.Close(); }

}

void adminMaster\_Closed(object sender, EventArgs e){ this.Close(); }

privatevoid cbShowPw\_CheckedChanged(object sender, EventArgs e)

{

if (cbShowPw.Checked)

{ txtpword.UseSystemPasswordChar = false; }

elseif (!cbShowPw.Checked)

{ txtpword.UseSystemPasswordChar = true; }

}

privatestaticbyte[] ConvertToByteArray(string password)

{

ASCIIEncoding encoding = newASCIIEncoding();

return encoding.GetBytes(password);

}

privatevoid linkForgotPW\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

adminMaster.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

c.cmd.CommandText = "select count(\*) from LoginTable where staffid='" + txtUsername.Text + "'";

if ((int)c.cmd.ExecuteScalar() <= 0)

{

txtUsername.Focus();

lblError.Text = "Username not found!";

Interaction.Beep();

return;

}

Login\_CP\_02b objChangePassword = newLogin\_CP\_02b();

objChangePassword.state = "Reset\_Password";

objChangePassword.txtUsername.Text = txtUsername.Text;

objChangePassword.Show();

}

}

}

02a LOGIN\_Change Password :

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace Library\_Auto

{

publicpartialclassLogin\_CP\_02b : Form

{

Connect c = newConnect();

SqlDataAdapter adp = newSqlDataAdapter();

publicstring state;

publicbool saveClicked = false;

public Login\_CP\_02b() { InitializeComponent(); }

privatevoid btnSave\_Click(object sender, EventArgs e)

{

if ((txtpword.Text asstring).Length <= 5)

{

txtpword.Clear();

txtConfirmPW.Clear();

MessageBox.Show("Password should be 6 Characters or more!");

txtpword.Focus();

return;

}

if (txtpword.Text != txtConfirmPW.Text)

{

txtpword.Clear();

txtConfirmPW.Clear();

lblError.Text = "Password doesn't match!";

txtpword.Focus();

return;

}

switch (state)

{

case"Reset\_Password":

c.cmd.CommandText = "select count(\*) from LoginTable where staffid = " + txtUsername.Text +" and sAnswer=@sAnswer";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@sAnswer", SqlDbType.VarBinary).Value = ConvertToByteArray(txtSAnswer.Text);

if ((int)c.cmd.ExecuteScalar() <= 0)

{ MessageBox.Show("Wrong Security Answer!"); }

else

{

c.cmd.CommandText = "update LoginTable set password=@password, resetpassword=@resetpassword where staffid=@staffid";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@staffid", SqlDbType.VarChar).Value = txtUsername.Text;

c.cmd.Parameters.Add("@password", SqlDbType.VarBinary).Value = ConvertToByteArray(txtpword.Text);

c.cmd.Parameters.Add("@resetpassword", SqlDbType.VarChar).Value = "Not\_Required";

c.cmd.ExecuteNonQuery();

MessageBox.Show("Password Changed!");

saveClicked = true;

this.Close();

}

break;

case"Reset\_Full":

c.cmd.CommandText = "update LoginTable set password=@password, resetpassword=@resetpassword, sQuestion=@sQuestion, sAnswer=@sAnswer where staffid=@staffid";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@staffid", SqlDbType.VarChar).Value = txtUsername.Text;

c.cmd.Parameters.Add("@password", SqlDbType.VarBinary).Value = ConvertToByteArray(txtpword.Text);

c.cmd.Parameters.Add("@resetpassword", SqlDbType.VarChar).Value = "Not\_Required";

c.cmd.Parameters.Add("@sQuestion", SqlDbType.VarBinary).Value = ConvertToByteArray(txtSQuestion.Text);

c.cmd.Parameters.Add("@sAnswer", SqlDbType.VarBinary).Value = ConvertToByteArray(txtSAnswer.Text);

c.cmd.ExecuteNonQuery();

MessageBox.Show("User details Reset!");

saveClicked = true;

this.Close();

break;

}

}

privatestaticbyte[] ConvertToByteArray(string str)

{

ASCIIEncoding encoding = newASCIIEncoding();

return encoding.GetBytes(str);

}

privatestaticstring ConvertToASCIIString(byte[] byteArray)

{

ASCIIEncoding encoding = newASCIIEncoding();

return encoding.GetString(byteArray);

}

privatevoid Login\_CP\_02b\_Load(object sender, EventArgs e)

{

if (state == "Reset\_Password") { ResetPasswordPrompt(); }

}

privatevoid ResetPasswordPrompt()

{

c.cmd.CommandText = "select sQuestion from LoginTable where staffid ='" + txtUsername.Text + "'";

lblSecQuestion.Visible = false;

txtSQuestion.BackColor = this.BackColor;

txtSQuestion.BorderStyle = BorderStyle.None;

txtSQuestion.TextAlign = HorizontalAlignment.Center;

txtUsername.ReadOnly = true;

string sQuestionString = ConvertToASCIIString((byte[]) c.cmd.ExecuteScalar());

sQuestionString += (sQuestionString[sQuestionString.Length - 1].Equals('?')) ? "" : "?";

txtSQuestion.Text = "" + sQuestionString;

txtSQuestion.ReadOnly = true;

}

}

}

03 Administration :

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using Library\_Auto.Modules;

using Library\_Auto.Crystal\_Reports;

namespace Library\_Auto

{

publicpartialclassAdministration\_03 : Form

{

publicError\_Handler errObj = newError\_Handler();

Login\_CP\_02b objCP = newLogin\_CP\_02b();

Connect c = newConnect();

SqlDataAdapter adp = newSqlDataAdapter();

DataTable dt = newDataTable();

BindingSource bds = newBindingSource();

publicbool isAdmin;

publicstring staffID;

public Administration\_03() { InitializeComponent(); }

privatevoid Administration\_03\_Load(object sender, EventArgs e)

{

SetUserDetails();

if (!isAdmin)

{

staffToolStripMenuItem1.Enabled = false;

staffToolStripMenuItem1.Visible = false;

purchaseToolStripMenuItem1.Enabled = false;

purchaseToolStripMenuItem1.Visible = false;

}

}

void SetUserDetails()

{

//For testing purpose

try

{

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

c.cmd.CommandText = "select \* from StaffTable where staffid='" + staffID + "'";

adp.SelectCommand = c.cmd;

adp.Fill(dt);

lblID.Text += "" + dt.Rows[0].ItemArray[0];

lblName.Text += "" + dt.Rows[0].ItemArray[2] + "" + dt.Rows[0].ItemArray[3];

lblDesg.Text += "" + dt.Rows[0].ItemArray[1];

txtAdd.Text = "" + dt.Rows[0].ItemArray[6];

lblEmail.Text += "" + dt.Rows[0].ItemArray[10];

lblPhno.Text += "" + dt.Rows[0].ItemArray[11];

DateTime dtDOJ = Convert.ToDateTime(dt.Rows[0].ItemArray[12]);

lblDOJ.Text += "" + dtDOJ.ToShortDateString();

lblStatus.Text += "" + dt.Rows[0].ItemArray[13];

if ((string)dt.Rows[0].ItemArray[13] != "Active")

{

menuStrip1.Enabled = false;

lblStatus.ForeColor = Color.Red;

}

if ((string)dt.Rows[0].ItemArray[1] == "Administrator")

{

isAdmin = true;

}

else isAdmin = false;

grpUserInfo.Focus();

}

catch (IndexOutOfRangeException){}

}

//Circulation Actions

privatevoid CreateCirculationObject()

{

Circulation\_06 objCirc = newCirculation\_06();

objCirc.adminInstance = this;

this.Hide();

objCirc.Show();

}

privatevoid issueToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateCirculationObject();

}

privatevoid reportsToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateCirculation\_rpt();

}

privatevoid fineToolStripMenuItem\_Click(object sender, EventArgs e)

{

Fine\_06b objFine = newFine\_06b();

this.Hide();

objFine.adminInstance = this;

objFine.Show();

}

//Accession Register Actions

privatevoid CreateAccRegObject(string action)

{

AccessionRegister\_04 objAccReg = newAccessionRegister\_04();

this.Hide();

objAccReg.adminInstance = this;

objAccReg.Show();

objAccReg.AccRegButtonActions(action);

}

privatevoid viewCatalogToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateAccRegObject("View");

}

privatevoid addCatalogToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateAccRegObject("Add");

}

privatevoid editCatalogToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateAccRegObject("Edit");

}

privatevoid deleteCatalogToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateAccRegObject("Delete");

}

privatevoid searchBookToolStripMenuItem\_Click(object sender, EventArgs e)

{

CreateAccRegObject("Search");

}

privatevoid oPACToolStripMenuItem\_Click(object sender, EventArgs e)

{

Search\_01 objSearch = newSearch\_01();

this.Hide();

objSearch.adminInstance = this;

objSearch.Show();

}

//Staff Actions

publicvoid CreateStaffObjects(string action)

{

Staff\_07 objStaff = newStaff\_07();

objStaff.adminInstance = this;

this.Hide();

objStaff.Show();

objStaff.StaffButtonActions(action);

}

privatevoid viewStaffToolStripMenuItem1\_Click(object sender, EventArgs e)

{ CreateStaffObjects("View");

}

privatevoid addStaffToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateStaffObjects("Add");

}

privatevoid editStaffToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateStaffObjects("Edit");

}

privatevoid deleteStaffToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateStaffObjects("Delete");

}

//Member Actions

publicvoid CreateMembersObject(string action)

{

Members\_05 objMembers = newMembers\_05();

objMembers.adminInstance = this;

this.Hide();

objMembers.Show();

objMembers.MembersButtonActions(action);

}

privatevoid viewMemberToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateMembersObject("View");

}

privatevoid addMemberToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateMembersObject("Add");

}

privatevoid editMemberToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateMembersObject("Edit");

}

privatevoid deleteMemberToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateMembersObject("Delete");

}

privatevoid searchMemberToolStripMenuItem\_Click(object sender, EventArgs e)

{

CreateMembersObject("Search");

}

privatevoid parametersToolStripMenuItem1\_Click(object sender, EventArgs e)

{

MemberParameters\_05b objMemP = newMemberParameters\_05b();

objMemP.adminInstance = this;

this.Hide();

objMemP.Show();

}

//Stock Button Actions

publicvoid CreateStockObjects()

{

Stock\_08 objStock = newStock\_08();

objStock.adminInstance = this;

this.Hide();

objStock.Show();

}

privatevoid viewStocksToolStripMenuItem\_Click(object sender, EventArgs e)

{

CreateStockObjects();

}

privatevoid editStocksToolStripMenuItem\_Click(object sender, EventArgs e)

{

CreateAccRegObject("Edit");

}

//Reports Button Clicks

void CreateReportObjects(string reName)

{

Report\_09 objReport = newReport\_09();

objReport.adminInstance = this;

this.Hide();

objReport.Show();

objReport.ShowReport(reName);

}

privatevoid CreateCirculation\_rpt()

{

Circulation\_rpt objCircRpt = newCirculation\_rpt();

objCircRpt.adminInstance = this;

this.Hide();

objCircRpt.Show();

}

privatevoid CreateMembers\_rpt()

{

Members\_rpt objMembersRpt = newMembers\_rpt();

objMembersRpt.adminInstance = this;

this.Hide();

objMembersRpt.Show();

}

privatevoid CreateStock\_rpt()

{

Stock\_rpt objStock\_rpt = newStock\_rpt();

objStock\_rpt.adminInstance = this;

this.Hide();

objStock\_rpt.Show();

}

privatevoid circulationReportToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateCirculation\_rpt();

}

privatevoid stockReportToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateStock\_rpt();

}

privatevoid membersReportToolStripMenuItem\_Click(object sender, EventArgs e)

{

CreateMembers\_rpt();

}

privatevoid purchaseOrderReportToolStripMenuItem1\_Click(object sender, EventArgs e)

{

Purchase\_Order\_rpt objPurOrd = newPurchase\_Order\_rpt();

objPurOrd.adminInstance = this;

this.Hide();

objPurOrd.Show();

}

privatevoid purchaseBillToolStripMenuItem\_Click(object sender, EventArgs e)

{

Create\_Bill\_rpt objCreateBill = newCreate\_Bill\_rpt();

objCreateBill.adminInstance = this;

this.Hide();

objCreateBill.Show();

}

privatevoid defaultersReportToolStripMenuItem1\_Click(object sender, EventArgs e)

{

Fine\_rpt objFine = newFine\_rpt();

objFine.adminInstance = this;

this.Hide();

objFine.Show();

}

privatevoid bookReportToolStripMenuItem\_Click(object sender, EventArgs e)

{

AccReg\_rpt objAccreg = newAccReg\_rpt();

objAccreg.adminInstance = this;

this.Hide();

objAccreg.Show();

}

//Holiday Objects

privatevoid holdiaysSetupToolStripMenuItem\_Click(object sender, EventArgs e)

{

HolidaySetup\_06c objHoliday = newHolidaySetup\_06c();

objHoliday.adminInstance = this;

this.Hide();

objHoliday.Show();

}

//Vendors Object

void CreateVendorsObject()

{

Vendor\_10b objVendor = newVendor\_10b();

objVendor.adminInstance = this;

this.Hide();

objVendor.Show();

}

privatevoid viewVendorToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateVendorsObject();

}

privatevoid addVendorToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateVendorsObject();

}

privatevoid editVendorToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateVendorsObject();

}

privatevoid deleteVendorToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateVendorsObject();

}

//Create Orders

publicvoid createOrderToolStripMenuItem1\_Click(object sender, EventArgs e)

{

CreateOrder\_10c objCreateOrder = newCreateOrder\_10c();

objCreateOrder.adminInstance = this;

this.Hide();

objCreateOrder.Show();

}

publicvoid generateBillToolStripMenuItem1\_Click(object sender, EventArgs e)

{

GenerateBill\_10d objGenBill = newGenerateBill\_10d();

objGenBill.adminInstance = this;

this.Hide();

objGenBill.Show();

}

//Password

privatevoid ChangePasswordToolStripMenuItem1\_Click(object sender, EventArgs e)

{

objCP = newLogin\_CP\_02b();

objCP.txtUsername.Text = staffID;

objCP.state = "Reset\_Full";

objCP.Closed += newEventHandler(objCP\_Closed);

objCP.Show();

}

privatevoid changePasswordToolStripMenuItem\_Click(object sender, EventArgs e)

{

objCP = newLogin\_CP\_02b();

objCP.txtUsername.Text = staffID;

objCP.state = "Reset\_Password";

objCP.Closed += newEventHandler(objCP\_Closed);

objCP.Show();

}

void objCP\_Closed(object sender, EventArgs e)

{

if (objCP.saveClicked == true) { Application.Restart(); }

}

privatevoid logoutToolStripMenuItem\_Click(object sender, EventArgs e)

{

Application.Restart();

}

//End Times

}

}

04 Accession Register :

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using Microsoft.VisualBasic;

namespace Library\_Auto

{

publicpartialclassAccessionRegister\_04 : Form

{

publicAdministration\_03 adminInstance;

Connect c = newConnect();

Connect c2 = newConnect();

DataTable dt = newDataTable();

DataTable dt2 = newDataTable();

SqlDataAdapter adp = newSqlDataAdapter();

SqlDataAdapter adp2 = newSqlDataAdapter();

BindingSource bds = newBindingSource();

BindingSource bds2 = newBindingSource();

string stockCommandText;

string state = "";

//Circulation Variables

publicbool sentByCirculation = false;

publicstring circAccno, circCopyNo;

public AccessionRegister\_04() { InitializeComponent(); }

privatevoid Catalog\_04\_Load(object sender, EventArgs e)

{

this.WindowState = FormWindowState.Maximized;

state = "";

grpNav.Enabled = false;

grpCopyNav.Enabled = false;

btnSubmit.Enabled = false;

btnCancel.Enabled = false;

DisableMainGroupbox();

ttBtnNet.SetToolTip(btnNet, "Calculate Net Amount using Price and Discount");

if (sentByCirculation) { btnSearch\_Click(this, EventArgs.Empty); }

try

{

c.cmd.CommandText = "select count(\*) from AccRegTable";

lblTotal.Text = "Total Records: " + (int)c.cmd.ExecuteScalar();

}

catch (InvalidOperationException) { }

catch (SqlException) { }

}

publicvoid AccRegButtonActions(string actionText)

{

switch (actionText)

{

case"Add": btnAdd\_Click(this, EventArgs.Empty); break;

case"View": btnView\_Click(this, EventArgs.Empty); break;

case"Edit": btnEdit\_Click(this, EventArgs.Empty); break;

case"Delete": btnDelete\_Click(this, EventArgs.Empty); break;

case"Search": btnSearch\_Click(this, EventArgs.Empty); break;

default: return;

}

}

protectedoverridevoid OnFormClosing(FormClosingEventArgs e)

{

base.OnFormClosing(e);

switch (e.CloseReason)

{

caseCloseReason.TaskManagerClosing:

caseCloseReason.WindowsShutDown:

return;

default: break;

}

if (!sentByCirculation && adminInstance != null) { adminInstance.Show(); }

}

privatevoid btnClear\_Click(object sender, EventArgs e)

{

bool clearStateValue;

bool addCopyClearOnly = false;

switch (state)

{

case"":

clearStateValue = true;

DisableMainGroupbox();

ddlSection.SelectedItem = null;

ddlType.SelectedItem = null;

txtAccNo.Text = "";

grpNav.Enabled = false;

grpCopyNav.Enabled = false;

break;

case"BeginView":

state = "View";

DisableMainGroupbox();

clearStateValue = false;

ddlSection.SelectedItem = null;

ddlType.SelectedItem = null;

txtAccNo.Text = "";

grpNav.Enabled = false;

grpCopyNav.Enabled = false;

break;

case"View":

DisableMainGroupbox();

clearStateValue = true;

ddlSection.SelectedItem = null;

ddlType.SelectedItem = null;

txtAccNo.Text = "";

grpNav.Enabled = false;

grpCopyNav.Enabled = false;

break;

case"Add":

clearStateValue = false;

break;

case"EditPrompt":

DisableMainGroupbox();

state = "Edit";

clearStateValue = false;

break;

case"Edit":

DisableMainGroupbox();

clearStateValue = true;

ddlSection.Text = "";

ddlType.Text = "";

txtAccNo.Text = "";

break;

case"Add Copy":

clearStateValue = false;

addCopyClearOnly = true;

break;

case"Delete Copy":

clearStateValue = true;

break;

default: return;

}

if (clearStateValue)

{

state = "";

stripCurrOps.Text = "Current Operation:" + state;

}

if (!addCopyClearOnly) dt.Clear();

dt2.Clear();

foreach (Control ctr in grpCatalogDetails.Controls)

{

if (addCopyClearOnly) break;

if (ctr isTextBox)

{

(ctr asTextBox).Clear();

(ctr asTextBox).ReadOnly = false;

}

if (ctr isComboBox) { (ctr asComboBox).Text = ""; }

}

foreach (Control ctr in grpCopyDetails.Controls)

{

if (ctr isTextBox)

{

(ctr asTextBox).Clear();

(ctr asTextBox).ReadOnly = false;

}

if (ctr isComboBox)

{

(ctr asComboBox).Text = "";

}

if (ctr isDateTimePicker)

{

(ctr asDateTimePicker).Value = DateTime.Today;

}

}

}

void DisableMainGroupbox()

{

ddlSection.Enabled = false;

ddlType.Enabled = false;

txtAccNo.Enabled = false;

grpCatalogDetails.Enabled = false;

grpCopyDetails.Enabled = false;

}

void EnableMainGroupBox()

{

ddlSection.Enabled = true;

ddlType.Enabled = true;

txtAccNo.Enabled = true;

grpCatalogDetails.Enabled = true;

grpCopyDetails.Enabled = true;

}

void EnableControls(string \_state)

{

switch (\_state)

{

case"Add" :

grpNav.Enabled = false;

grpCopyNav.Enabled = false;

EnableMainGroupBox();

txtAccNo.ReadOnly = false;

txtCopyNo.ReadOnly = false;

btnSubmit.Enabled = true;

btnCancel.Enabled = true;

break;

case"View":

DisableMainGroupbox();

grpNav.Enabled = true;

grpCopyNav.Enabled = true;

txtAccNo.ReadOnly = false;

txtCopyNo.ReadOnly = false;

btnSubmit.Enabled = false;

btnCancel.Enabled = false;

break;

case"Edit":

//Only called for Prompt.

EnableMainGroupBox();

grpNav.Enabled = false;

grpCopyNav.Enabled = false;

txtAccNo.ReadOnly = true;

txtCopyNo.ReadOnly = true;

btnSubmit.Enabled = true;

btnCancel.Enabled = true;

break;

case"Delete":

DisableMainGroupbox();

break;

case"Add Copy":

ddlSection.Enabled = false;

ddlType.Enabled = false;

txtAccNo.Enabled = false;

grpCatalogDetails.Enabled = false;

grpCopyDetails.Enabled = true;

grpNav.Enabled = false;

grpCopyNav.Enabled = false;

txtCopyNo.ReadOnly = false;

txtAccNo.ReadOnly = false;

btnSubmit.Enabled = true;

btnCancel.Enabled = true;

break;

default: return;

}

}

privatevoid btnAdd\_Click(object sender, EventArgs e)

{

state = "Add";

stripCurrOps.Text = "Current Operation: " + state;

btnClear\_Click(this, EventArgs.Empty);

EnableControls(state);

ListUpdates();

}

privatevoid DdlUpdate(string cmdText, ComboBox ddlToFill)

{

DataTable dtDummy = newDataTable();

string itemToAdd = "";

try

{

ddlToFill.Items.Clear();

if (!ddlToFill.Items.Contains("(None)"))

ddlToFill.Items.Add("(None)");

c.cmd.CommandText = cmdText;

adp.SelectCommand = c.cmd;

adp.Fill(dtDummy);

for (int i = 0; i < dtDummy.Rows.Count; i++)

{

itemToAdd = "";

for (int j = 0; j < dtDummy.Columns.Count; j++)

{

itemToAdd += dtDummy.Rows[i].ItemArray[j] + "";

}

if (!ddlToFill.Items.Contains(itemToAdd))

{

ddlToFill.Items.Add(itemToAdd);

}

}

}

catch (Exception ex)

{

adminInstance.errObj.txtException.Text = ex.ToString();

adminInstance.errObj.Show();

}

}

privatevoid DdlListUpdate(string cmdText, ComboBox ddlToFill)

{

DataTable dtDummy = newDataTable();

string itemToAdd = "";

ddlToFill.Items.Clear();

try

{

Connect c = newConnect();

if (!ddlToFill.Items.Contains("(None)"))

ddlToFill.Items.Insert(0, "(None)");

c.cmd.CommandText = cmdText;

adp.SelectCommand = c.cmd;

adp.Fill(dtDummy);

for (int i = 0; i < dtDummy.Rows.Count; i++)

{

itemToAdd = "" + dtDummy.Rows[i].ItemArray[0];

if (!ddlToFill.Items.Contains(itemToAdd))

{ ddlToFill.Items.Add(itemToAdd); }

}

}

catch (Exception ex) {}

}

privatevoid btnView\_Click(object sender, EventArgs e)

{

state = "BeginView";

btnClear\_Click(this, EventArgs.Empty);

EnableControls(state);

//The State is Set to "View" in Enable Controls

stripCurrOps.Text = "Current Operation: " + state;

try

{

if (c.cnn.State != ConnectionState.Open)

{

c.cnn.Close();

c.cnn.Open();

}

stockCommandText = "";

c.cmd.CommandText = "select \* from AccRegTable";

adp.SelectCommand = c.cmd;

adp.Fill(dt);

bds.DataSource = dt;

bds.Sort = "accno ASC";

ClearAccRegDataBindings();

AddAccRegDataBindings();

StockBindingUpdate();

ClearStockDataBindings();

AddStockDataBindings();

}

catch (SqlException) { throw; }

finally

{

c.cnn.Close();

c2.cnn.Close();

}

}

privatevoid btnSearch\_Click(object sender, EventArgs e)

{

state = "View";

stripCurrOps.Text = "Current Operation: " + "Search";

btnClear\_Click(this, EventArgs.Empty);

state = "View";

stripCurrOps.Text = "Current Operation: " + "Search";

//Enable Controls sent down to work with only Found values

try

{

if (c.cnn.State != ConnectionState.Open)

{

c.cnn.Close();

c.cnn.Open();

}

string accno;

if (sentByCirculation)

{

accno = circAccno;

lblAccReg.Text = "Searching Record | Accession Number : " + accno;

}

else

{ accno = Interaction.InputBox("Enter the Accession Number:", "Search using Accno", "", -1, -1); }

c.cmd.CommandText = "select count(\*) from AccRegTable where accno ='" + accno + "'";

if ((int)c.cmd.ExecuteScalar() <= 0)

{

Interaction.Beep();

MessageBox.Show("Record not found!", "Search");

state = "";

stripCurrOps.Text = "Current Operation: " + state;

return;

}

stockCommandText = "";

c.cmd.CommandText = "select \* from AccRegTable where accno ='" + accno + "'";

EnableControls(state);

grpNav.Enabled = false;

txtAccNo.Text = accno;

dt.Clear();

adp.SelectCommand = c.cmd;

adp.Fill(dt);

bds.DataSource = dt;

ClearAccRegDataBindings();

AddAccRegDataBindings();

StockBindingUpdate();

ClearStockDataBindings();

AddStockDataBindings();

//Set the State to Null here if Problem persists

}

catch (SqlException){ throw; }

finally

{

c.cnn.Close();

c2.cnn.Close();

}

}

void StockBindingUpdate()

{

//updates the StockDatabindings on each >>, >, <<, <, of Catalog.

try

{

if (c2.cnn.State != ConnectionState.Open)

{

c2.cnn.Close();

c2.cnn.Open();

}

if (sentByCirculation)

{

stockCommandText = "select \* from StockTable where accno='" + circAccno + "' and copyno='" + circCopyNo + "'";

//Invalidate Everything

DisableMainGroupbox();

panelMain.Enabled = false;

btnClear.Enabled = false;

grpCopyNav.Enabled = false;

}

else stockCommandText = "select \* from StockTable where accno='" + txtAccNo.Text + "'";

dt2.Clear();

c2.cmd.CommandText = stockCommandText;

adp2.SelectCommand = c2.cmd;

adp2.Fill(dt2);

for (int i = 0; i < dt2.Rows.Count; i++)

{

if (!ddlVendor.Items.Contains(dt2.Rows[i].ItemArray[2].ToString()))

{

ddlVendor.Items.Add((string)dt2.Rows[i].ItemArray[2]);

}

}

bds2.DataSource = dt2;

bds2.Sort = "copyno ASC";

}

catch (SqlException){throw;}

}

privatevoid btnEdit\_Click(object sender, EventArgs e)

{

try

{

if (c.cnn.State != ConnectionState.Open)

{

c.cnn.Close();

c.cnn.Open();

}

if (state == "View")

{

state = "Edit";

stripCurrOps.Text = "Current Operation: " + state;

c.cmd.CommandText = "select count(\*) from StockTable where accno='" + txtAccNo.Text + "'";

if ((int)c.cmd.ExecuteScalar() == 0)

{

MessageBox.Show("No Copy found! Redirecting to Add Copy!", "Book Stock Empty");

state = "View";

stripCurrOps.Text = "Current Operation: " + state;

btnAddCopy\_Click(this, EventArgs.Empty);

return;

}

EnableControls(state);

return;

}

else

{

state = "EditPrompt";

btnClear\_Click(this, EventArgs.Empty);

EnableControls(state);

//Enable Controls sets the State to Edit

stripCurrOps.Text = "Current Operation: " + state;

string accno = Interaction.InputBox("Enter the Accno:", "Edit Record", "", -1, -1);

c.cmd.CommandText = "select count(\*) from AccRegTable where accno='" + accno + "'";

if ((int)c.cmd.ExecuteScalar() > 0)

{

string copyno = Interaction.InputBox("Search for Specific book using copy Number: \nEnter the Copy Number:", "Enter Copy Number", "", -1, -1);

c.cmd.CommandText = "select count(\*) from StockTable where accno='" + accno + "'";

if ((int)c.cmd.ExecuteScalar() == 0)

{

MessageBox.Show("No Copy found! Redirecting to Add Copy!", "Book Stock Empty");

btnAddCopy\_Click(this, EventArgs.Empty);

return;

}

c.cmd.CommandText = "select count(\*) from StockTable where accno='" + accno + "' and copyno='" + copyno + "'";

if ((int)c.cmd.ExecuteScalar() <= 0)

{

Interaction.Beep();

MessageBox.Show("Book with Accno:" + accno + " and Copy Number:" + copyno + " not found!", "Edit");

btnCancel\_Click(this, EventArgs.Empty);

return;

}

c.cmd.CommandText = "select \* from AccRegTable where accno='" + accno + "'";

adp.SelectCommand = c.cmd;

dt.Clear();

adp.Fill(dt);

bds.DataSource = dt;

ClearAccRegDataBindings();

AddAccRegDataBindings();

stockCommandText = "select \* from StockTable where accno='" + accno + "' and copyno='" + copyno + "'";

c2.cmd.CommandText = stockCommandText;

adp2.SelectCommand = c2.cmd;

dt2.Clear();

adp2.Fill(dt2);

bds2.DataSource = dt2;

ClearStockDataBindings();

AddStockDataBindings();

}

else

{

Interaction.Beep();

MessageBox.Show("Record not found!", "Edit");

state = "Edit";

stripCurrOps.Text = "Current Operation: " + state;

btnCancel\_Click(this, EventArgs.Empty);

}

}

}

catch (SqlException){throw;}

}

privatevoid btnDelete\_Click(object sender, EventArgs e)

{

try

{

if (c.cnn.State != ConnectionState.Open)

{

c.cnn.Close();

c.cnn.Open();

}

if (state == "View")

{

DialogResult dolores = MessageBox.Show("Delete this record? Accno = "

+ txtAccNo.Text + "\nAll Stock information of this book will be deleted!",

"Delete Records", MessageBoxButtons.OKCancel, MessageBoxIcon.Information);

if (dolores == DialogResult.Cancel)

{

Interaction.Beep();

return;

}

elseif (dolores == DialogResult.OK)

{

DeleteRecords(txtAccNo.Text);

}

}

else

{

string inputText = Interaction.InputBox("Enter the Accno:", "Delete Records", "", -1, -1);

c.cmd.CommandText = "select count(\*) from AccRegTable where accno='" + inputText + "'";

if ((int)c.cmd.ExecuteScalar() > 0)

{

c.cmd.CommandText = "select \* from AccRegTable where accno='" + inputText + "'";

c2.cmd.CommandText = "select \* from StockTable where accno='" + inputText + "'";

adp.SelectCommand = c.cmd;

adp2.SelectCommand = c2.cmd;

dt.Clear();

dt2.Clear();

adp.Fill(dt);

adp2.Fill(dt2);

bds.DataSource = dt;

bds2.DataSource = dt2;

ClearAccRegDataBindings();

ClearStockDataBindings();

AddAccRegDataBindings();

AddStockDataBindings();

DeleteRecords(inputText);

}

else

{

Interaction.Beep();

MessageBox.Show("Invalid!", "Delete Records");

}

}

}

catch (SqlException){throw;}

}

void DeleteRecords(string accno)

{

c.cmd.CommandText = "delete from StockTable where accno='" + accno + "'";

c.cmd.ExecuteNonQuery();

string accCommandText = "delete from AccRegTable where accno='" + accno + "'";

c.cmd.CommandText = accCommandText;

c.cmd.ExecuteNonQuery();

state = "";

stripCurrOps.Text = "Current Operation: " + "Delete";

MessageBox.Show("Records deleted!", "Delete Records");

}

bool EmptyValidations()

{

if (txtAccNo.Text == "" || ddlSection.SelectedItem == null

|| ddlType.SelectedItem == null)

{

MessageBox.Show("Main Fields Empty! \nPlease Check for Accession Number, Section or Type.", "Empty Fields", MessageBoxButtons.OK, MessageBoxIcon.Stop);

txtAccNo.Focus();

returntrue;

}

if (txtPubYear.Text != "")

{

if (Convert.ToInt16(txtPubYear.Text) >DateTime.Now.Year + 1)

{

stripError.Text = "Error: Invalid Year!";

txtPubYear.Focus();

returntrue;

}

}

btnNet\_Click(this, EventArgs.Empty);

foreach (Control ctrl in grpCatalogDetails.Controls)

{

if (ctrl isTextBox)

{

if ((ctrl asTextBox).Text == "")

{

stripError.Text = "Error: Empty Textbox";

(ctrl asTextBox).Focus();

returntrue;

}

}

elseif (ctrl isComboBox)

{

if ((ctrl asComboBox).SelectedItem == null)

{

stripError.Text = "Error: Unselected Item";

(ctrl asComboBox).Focus();

returntrue;

}

}

}

foreach (Control ctrl in grpCopyDetails.Controls)

{

if (ctrl isTextBox)

{

if ((ctrl asTextBox).Text == "")

{

stripError.Text = "Error: Empty Textbox";

(ctrl asTextBox).Focus();

returntrue;

}

}

elseif (ctrl isComboBox)

{

if ((ctrl asComboBox).SelectedItem == null)

{

stripError.Text = "Error: Unselected Item";

(ctrl asComboBox).Focus();

returntrue;

}

}

}

returnfalse;

}

privatevoid btnSubmit\_Click(object sender, EventArgs e)

{

try

{

if (c.cnn.State != ConnectionState.Open)

{

c.cnn.Close();

c.cnn.Open();

}

string process = state;

bool isInvalid;

isInvalid = EmptyValidations();

if (isInvalid) {return;}

else stripError.Text = "...";

switch (state)

{

case"Add":

c.cmd.CommandText = "select count(\*) from AccRegTable where accno='" + txtAccNo.Text + "'";

if ((int)c.cmd.ExecuteScalar() > 0)

{

Interaction.Beep();

MessageBox.Show("Book already present! \nAccno Match found. \n\nEnter a New Accno!", "Add Book", MessageBoxButtons.OK, MessageBoxIcon.Stop);

txtAccNo.Focus();

return;

}

c.cmd.CommandText = "insert into AccRegTable values (@accno, @section, @type, @title, @subtitle, @author, @editor, @volume, @series, @isbn, @language, @keyword, @remarks, @subject, @subject2, @callno, @classno, @pubyear, @place, @publisher)";

stockCommandText = "insert into StockTable values (@accno, @copyno, @vendor, @source, @currency, @dept, @edition, @billno, @billdate, @discount, @status, @category, @price, @netcost, @pages, @location, @binding, @copyyear)";

state = "View";

stripCurrOps.Text = "Current Operation: " + state;

break;

case"Edit":

txtAccNo.ReadOnly = false;

txtCopyNo.ReadOnly = false;

c.cmd.CommandText = "update AccRegTable set section=@section, type=@type, title=@title, subtitle=@subtitle, author=@author, editor=@editor, volume=@volume, series=@series, isbn=@isbn, language=@language, keyword=@keyword, remarks=@remarks, subject=@subject, subject2=@subject2, callno=@callno, classno=@classno, pubyear=@pubyear, place=@place, publisher=@publisher where accno=@accno";

stockCommandText = "update StockTable set vendor=@vendor, source=@source, currency=@currency, dept=@dept, edition=@edition, billno=@billno, billdate=@billdate, discount=@discount, status=@status, category=@category, price=@price, netcost=@netcost, pages=@pages, location=@location, binding=@binding, copyyear=@copyyear where accno=@accno and copyno=@copyno";

state = "View";

stripCurrOps.Text = "Current Operation: " + state;

break;

case"Add Copy":

c.cmd.CommandText = "select count(\*) from StockTable where accno='" + txtAccNo.Text + "' and copyno='" + txtCopyNo.Text + "'";

int tempCount = (int)c.cmd.ExecuteScalar();

if (tempCount > 0)

{

MessageBox.Show("A Copy Number match has been found!, Insert a new copy Number!", "Add a Copy");

txtCopyNo.Focus();

return;

}

c.cmd.CommandText = "AddCopySkipAccRegTable";

stockCommandText = "insert into StockTable values (@accno, @copyno, @vendor, @source, @currency, @dept, @edition, @billno, @billdate, @discount, @status, @category, @price, @netcost, @pages, @location, @binding, @copyyear)";

grpNav.Enabled = false;

grpCopyNav.Enabled = false;

state = "View";

stripCurrOps.Text = "Current Operation: " + state;

break;

default: return;

}

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@accno", SqlDbType.VarChar).Value = txtAccNo.Text;

c.cmd.Parameters.Add("@section", SqlDbType.VarChar).Value = ddlSection.Text;

c.cmd.Parameters.Add("@type", SqlDbType.VarChar).Value = ddlType.Text;

c.cmd.Parameters.Add("@title", SqlDbType.VarChar).Value = txtTitle.Text;

c.cmd.Parameters.Add("@subtitle", SqlDbType.VarChar).Value = txtSubTitle.Text;

c.cmd.Parameters.Add("@author", SqlDbType.VarChar).Value = txtAuthor.Text;

c.cmd.Parameters.Add("@editor", SqlDbType.VarChar).Value = txtEditor.Text;

c.cmd.Parameters.Add("@volume", SqlDbType.VarChar).Value = txtVolume.Text;

c.cmd.Parameters.Add("@series", SqlDbType.VarChar).Value = txtSeries.Text;

c.cmd.Parameters.Add("@isbn", SqlDbType.VarChar).Value = txtISBN.Text;

c.cmd.Parameters.Add("@language", SqlDbType.VarChar).Value = ddlLanguage.Text;

c.cmd.Parameters.Add("@keyword", SqlDbType.VarChar).Value = txtKeyword.Text;

c.cmd.Parameters.Add("@remarks", SqlDbType.VarChar).Value = txtRemark.Text;

c.cmd.Parameters.Add("@subject", SqlDbType.VarChar).Value = ddlSub1.Text;

c.cmd.Parameters.Add("@subject2", SqlDbType.VarChar).Value = ddlSub2.Text;

c.cmd.Parameters.Add("@callno", SqlDbType.VarChar).Value = txtCallNo.Text;

c.cmd.Parameters.Add("@classno", SqlDbType.VarChar).Value = txtClsNo.Text;

c.cmd.Parameters.Add("@pubyear", SqlDbType.VarChar).Value = txtPubYear.Text;

c.cmd.Parameters.Add("@place", SqlDbType.VarChar).Value = txtPlace.Text;

c.cmd.Parameters.Add("@publisher", SqlDbType.VarChar).Value = txtPublisher.Text;

if (!c.cmd.CommandText.Equals("AddCopySkipAccRegTable"))

{

c.cmd.ExecuteNonQuery();

}

c.cmd.Parameters.Clear();

c.cmd.CommandText = stockCommandText;

c.cmd.Parameters.Add("@accno", SqlDbType.VarChar).Value = txtAccNo.Text;

c.cmd.Parameters.Add("@copyno", SqlDbType.Int).Value = Convert.ToInt32(txtCopyNo.Text);

c.cmd.Parameters.Add("@vendor", SqlDbType.VarChar).Value = ddlVendor.Text;

c.cmd.Parameters.Add("@source", SqlDbType.VarChar).Value = ddlSource.Text;

c.cmd.Parameters.Add("@currency", SqlDbType.VarChar).Value = ddlCurency.Text;

c.cmd.Parameters.Add("@dept", SqlDbType.VarChar).Value = ddlDept.Text;

c.cmd.Parameters.Add("@edition", SqlDbType.VarChar).Value = txtEdition.Text;

c.cmd.Parameters.Add("@billno", SqlDbType.VarChar).Value = txtBillNo.Text;

c.cmd.Parameters.Add("@billdate", SqlDbType.VarChar).Value = dtBilldate.Value;

c.cmd.Parameters.Add("@discount", SqlDbType.VarChar).Value = txtDiscount.Text;

c.cmd.Parameters.Add("@status", SqlDbType.VarChar).Value = ddlStatus.Text;

c.cmd.Parameters.Add("@category", SqlDbType.VarChar).Value = ddlCategory.Text;

c.cmd.Parameters.Add("@price", SqlDbType.VarChar).Value = txtPrice.Text;

c.cmd.Parameters.Add("@netcost", SqlDbType.VarChar).Value = txtNetCost.Text;

c.cmd.Parameters.Add("@pages", SqlDbType.VarChar).Value = txtPages.Text;

c.cmd.Parameters.Add("@location", SqlDbType.VarChar).Value = txtLocation.Text;

c.cmd.Parameters.Add("@binding", SqlDbType.VarChar).Value = ddlBinding.Text;

c.cmd.Parameters.Add("@copyyear", SqlDbType.VarChar).Value = txtCopyYear.Text;

c.cmd.ExecuteNonQuery();

btnSubmit.Enabled = false;

btnCancel.Enabled = false;

DisableMainGroupbox();

c.cmd.CommandText = "select count(\*) from AccRegTable";

lblTotal.Text = "Total Records: " + (int)c.cmd.ExecuteScalar();

MessageBox.Show(process + " Complete!", process, MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (SqlException) {throw; }

}

privatevoid btnAddCopy\_Click(object sender, EventArgs e)

{

try

{

if (c.cnn.State != ConnectionState.Open)

{

c.cnn.Close();

c.cnn.Open();

}

if (state == "View")

{

state = "Add Copy";

stripCurrOps.Text = "Current Operation: " + state;

EnableControls(state);

txtCopyNo.Focus();

}

else

{

string accno = Interaction.InputBox("Enter the Accno:", "Add a copy of the Book", "", -1, -1);

c.cmd.CommandText = "select count(\*) from AccRegTable where accno='" + accno + "'";

if ((int)c.cmd.ExecuteScalar() > 0)

{

c.cmd.CommandText = "select \* from AccRegTable where accno='" + accno + "'";

adp.SelectCommand = c.cmd;

dt.Clear();

adp.Fill(dt);

bds.DataSource = dt;

ClearAccRegDataBindings();

AddAccRegDataBindings();

ClearStockDataBindings();

state = "Add Copy";

stripCurrOps.Text = "Current Operation: " + state;

EnableControls(state);

txtCopyNo.Focus();

}

else

{

MessageBox.Show("Entry with accno:" + accno + " not found!", "Add Copy");

state = "";

stripCurrOps.Text = "Current Operation: " + state;

btnCancel\_Click(this, EventArgs.Empty);

}

}

}

catch (SqlException){throw;}

}

privatevoid btnDeleteCopy\_Click(object sender, EventArgs e)

{

try

{

if (c.cnn.State != ConnectionState.Open)

{

c.cnn.Close();

c.cnn.Open();

}

if (state == "View")

{

if (txtCopyNo.Text == "")

{

MessageBox.Show("No Copy found!", "Delete Copy");

return;

}

DialogResult diaRes = MessageBox.Show("Delete this Copy?\n\n Accno:"

+ txtAccNo.Text + "\n Copy Number:" + txtCopyNo.Text,

"Delete Copy", MessageBoxButtons.OKCancel);

if (diaRes == DialogResult.Cancel)

{

return;

}

elseif (diaRes == DialogResult.OK)

{

c.cmd.CommandText = "delete from StockTable where accno='"

+ txtAccNo.Text + "' and copyno='" + txtCopyNo.Text + "'";

c.cmd.ExecuteNonQuery();

MessageBox.Show("Records Deleted!");

grpNav.Enabled = false;

grpCopyNav.Enabled = false;

DisableMainGroupbox();

state = "";

stripCurrOps.Text = "Current Operation: " + "Delete Copy";

}

}

else

{

string accno = Interaction.InputBox("Enter the Accno:", "Enter Accno", "", -1, -1);

c.cmd.CommandText = "select count(\*) from AccRegTable where accno='" + accno + "'";

if ((int)c.cmd.ExecuteScalar() > 0)

{

string copyno = Interaction.InputBox("Enter the Copy Number:", "Copy Number", "", -1, -1);

c.cmd.CommandText = "select count(\*) from StockTable where accno ='" + accno + "' and copyno='" + copyno + "'";

if ((int)c.cmd.ExecuteScalar() > 0)

{

c.cmd.CommandText = "select \* from AccRegTable where accno='" + accno + "'";

c2.cmd.CommandText = "select \* from StockTable where accno ='" + accno + "' and copyno ='" + copyno + "'";

dt.Clear();

dt2.Clear();

adp.SelectCommand = c.cmd;

adp.Fill(dt);

bds.DataSource = dt;

adp2.SelectCommand = c2.cmd;

adp2.Fill(dt2);

bds2.DataSource = dt2;

ClearAccRegDataBindings();

AddAccRegDataBindings();

ClearStockDataBindings();

AddStockDataBindings();

//Delete Copy starts here

c.cmd.CommandText = "delete from StockTable where accno ='" + accno + "' and copyno ='" + copyno + "'";

c.cmd.ExecuteNonQuery();

MessageBox.Show("Records Deleted! \nAccno:" + txtAccNo.Text + "\nCopy Number:" + copyno, "Delete Copy");

grpNav.Enabled = false;

grpCopyNav.Enabled = false;

DisableMainGroupbox();

//EnableControls();

state = "View";

stripCurrOps.Text = "Current Operation: " + "Delete Copy";

}

elseMessageBox.Show("Invalid Copy Number!", "Delete Copy", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

elseMessageBox.Show("Record with Accno:" + accno + " is Unavailable", "Delete Copy", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

catch (SqlException){throw;}

}

privatevoid btnCancel\_Click(object sender, EventArgs e)

{

state = "";

btnClear\_Click(this, EventArgs.Empty);

DisableMainGroupbox();

txtAccNo.ReadOnly = false;

txtCopyNo.ReadOnly = false;

btnSubmit.Enabled = false;

btnCancel.Enabled = false;

}

void ClearAccRegDataBindings()

{

ddlSection.DataBindings.Clear();

ddlType.DataBindings.Clear();

txtAccNo.DataBindings.Clear();

foreach (Control c in grpCatalogDetails.Controls)

{

if (c isTextBox) {(c asTextBox).DataBindings.Clear(); }

if (c isComboBox) { (c asComboBox).DataBindings.Clear(); }

}

}

void ClearStockDataBindings()

{

foreach (Control c in grpCopyDetails.Controls)

{

if (c isTextBox) { (c asTextBox).DataBindings.Clear(); }

if (c isComboBox) { (c asComboBox).DataBindings.Clear(); }

if (c isDateTimePicker)

(c asDateTimePicker).DataBindings.Clear();

}

}

void AddAccRegDataBindings()

{

txtAccNo.DataBindings.Add("text", bds, "accno");

ddlSection.DataBindings.Add("text", bds, "section");

ddlType.DataBindings.Add("text", bds, "type");

txtTitle.DataBindings.Add("text", bds, "title");

txtSubTitle.DataBindings.Add("text", bds, "subtitle");

txtAuthor.DataBindings.Add("text", bds, "author");

txtEditor.DataBindings.Add("text", bds, "editor");

txtVolume.DataBindings.Add("text", bds, "volume");

txtSeries.DataBindings.Add("text", bds, "series");

txtISBN.DataBindings.Add("text", bds, "isbn");

ddlLanguage.DataBindings.Add("text", bds, "language");

txtKeyword.DataBindings.Add("text", bds, "keyword");

txtRemark.DataBindings.Add("text", bds, "remarks");

ddlSub1.DataBindings.Add("text", bds, "subject");

ddlSub2.DataBindings.Add("text", bds, "subject2");

txtCallNo.DataBindings.Add("text", bds, "callno");

txtClsNo.DataBindings.Add("text", bds, "classno");

txtPubYear.DataBindings.Add("text", bds, "pubyear");

txtPlace.DataBindings.Add("text", bds, "place");

txtPublisher.DataBindings.Add("text", bds, "publisher");

}

void AddStockDataBindings()

{

txtCopyNo.DataBindings.Add("text", bds2, "copyno");

ddlVendor.DataBindings.Add("text", bds2, "vendor");

ddlSource.DataBindings.Add("text", bds2, "source");

ddlCurency.DataBindings.Add("text", bds2, "currency");

ddlDept.DataBindings.Add("text", bds2, "dept");

txtEdition.DataBindings.Add("text", bds2, "edition");

txtBillNo.DataBindings.Add("text", bds2, "billno");

dtBilldate.DataBindings.Add("text", bds2, "billdate");

txtDiscount.DataBindings.Add("text", bds2, "discount");

ddlStatus.DataBindings.Add("text", bds2, "status");

ddlCategory.DataBindings.Add("text", bds2, "category");

txtPrice.DataBindings.Add("text", bds2, "price");

txtNetCost.DataBindings.Add("text", bds2, "netcost");

txtPages.DataBindings.Add("text", bds2, "pages");

txtLocation.DataBindings.Add("text", bds2, "location");

ddlBinding.DataBindings.Add("text", bds2, "binding");

txtCopyYear.DataBindings.Add("text", bds2, "copyyear");

}

privatevoid NumValidations(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar)

&& e.KeyChar != 0)

{

stripError.Text = "Error: Enter a number!";

e.Handled = true;

return;

}

stripError.Text = "...";

}

privatevoid AlphaValidations(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsLetter(e.KeyChar)

&& !char.IsWhiteSpace(e.KeyChar))

{

stripError.Text = "Enter only Letters!";

e.Handled = true;

return;

}

stripError.Text = "...";

}

privatevoid AlphaNumValidations(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsLetterOrDigit(e.KeyChar)

&& !char.Equals(e.KeyChar, '-'))

{

stripError.Text = "Error: Enter only Letters or Digits";

e.Handled = true;

return;

}

stripError.Text = "...";

}

privatevoid btnStockLookup\_Click(object sender, EventArgs e)

{

Stock\_08 objStockLookup = newStock\_08();

if (state == "View")

{

objStockLookup.accno = txtAccNo.Text;

}

objStockLookup.Show();

}

privatevoid btnFirst\_Click(object sender, EventArgs e)

{

bds.MoveFirst();

StockBindingUpdate();

}

privatevoid btnPrev\_Click(object sender, EventArgs e)

{

bds.MovePrevious();

StockBindingUpdate();

}

privatevoid btnNext\_Click(object sender, EventArgs e)

{

bds.MoveNext();

StockBindingUpdate();

}

privatevoid btnLast\_Click(object sender, EventArgs e)

{

bds.MoveLast();

StockBindingUpdate();

}

privatevoid btnCopyFirst\_Click(object sender, EventArgs e)

{

bds2.MoveFirst();

}

privatevoid btnCopyPrev\_Click(object sender, EventArgs e)

{

bds2.MovePrevious();

}

privatevoid btnCopyNext\_Click(object sender, EventArgs e)

{

bds2.MoveNext();

}

privatevoid btnCopyLast\_Click(object sender, EventArgs e)

{

bds2.MoveLast();

}

privatevoid btnNet\_Click(object sender, EventArgs e)

{

if (txtDiscount.Text == "" || txtPrice.Text == "")

{

txtNetCost.Text = "";

return;

}

int cost = Convert.ToInt16(txtPrice.Text);

int discount = Convert.ToInt16(txtDiscount.Text);

if (discount > 100)

{

stripError.Text = "Discount is Greater than 100%";

txtNetCost.Text = "";

return;

}

int net = (cost - (cost \* discount / 100));

txtNetCost.Text = net.ToString();

}

void ListUpdates()

{

DdlUpdate("select name, address from VendorTable", ddlVendor);

DdlListUpdate("select list from \_LanguageTable", ddlLanguage);

DdlListUpdate("select list from \_SubjectTable", ddlSub1);

DdlListUpdate("select list from \_SubjectTable", ddlSub2);

DdlListUpdate("select list from \_SourceTable", ddlSource);

DdlListUpdate("select list from \_CategoryTable", ddlCategory);

DdlListUpdate("select list from \_CurrencyTable", ddlCurency);

DdlListUpdate("select list from \_DeptTable", ddlDept);

}

privatevoid btnLang\_Click(object sender, EventArgs e)

{

AccRegButtons\_04a objButton = newAccRegButtons\_04a();

objButton.currentTable = "\_LanguageTable";

objButton.Closed += newEventHandler(objButton\_Closed);

objButton.Show();

}

void objButton\_Closed(object sender, EventArgs e)

{

ListUpdates();

}

privatevoid btnSubject\_Click(object sender, EventArgs e)

{

AccRegButtons\_04a objButton = newAccRegButtons\_04a();

objButton.currentTable = "\_SubjectTable";

objButton.Closed += newEventHandler(objButton\_Closed);

objButton.Show();

}

privatevoid btnSource\_Click(object sender, EventArgs e)

{

AccRegButtons\_04a objButton = newAccRegButtons\_04a();

objButton.currentTable = "\_SourceTable";

objButton.Closed += newEventHandler(objButton\_Closed);

objButton.Show();

}

privatevoid btnCurrency\_Click(object sender, EventArgs e)

{

AccRegButtons\_04a objButton = newAccRegButtons\_04a();

objButton.currentTable = "\_CurrencyTable";

objButton.Closed += newEventHandler(objButton\_Closed);

objButton.Show();

}

privatevoid btnCategory\_Click(object sender, EventArgs e)

{

AccRegButtons\_04a objButton = newAccRegButtons\_04a();

objButton.currentTable = "\_CategoryTable";

objButton.Closed += newEventHandler(objButton\_Closed);

objButton.Show();

}

privatevoid btnDept\_Click(object sender, EventArgs e)

{

AccRegButtons\_04a objButton = newAccRegButtons\_04a();

objButton.currentTable = "\_DeptTable";

objButton.Closed += newEventHandler(objButton\_Closed);

objButton.Show();

}

}

}

04a Acc Reg Buttons :

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using Microsoft.VisualBasic;

using System.Data.SqlClient;

namespace Library\_Auto

{

publicpartialclassAccRegButtons\_04a : Form

{

publicstring currentTable;

Connect c = newConnect();

SqlDataAdapter adp = newSqlDataAdapter();

DataTable dt = newDataTable();

public AccRegButtons\_04a(){InitializeComponent();}

privatevoid AccRegButtons\_04a\_Load(object sender, EventArgs e) {SelectList();}

privatevoid SelectList()

{

string itemToAdd;

c.cmd.CommandText = "select list from " + currentTable;

adp.SelectCommand = c.cmd;

dt.Clear();

adp.Fill(dt);

for (int i = 0; i < dt.Rows.Count; i++)

{

itemToAdd = (string)dt.Rows[i].ItemArray[0];

if (!adderList.Items.Contains(itemToAdd))

adderList.Items.Add(itemToAdd);

}

}

privatevoid btnAdd\_Click(object sender, EventArgs e)

{

if (txtAdd.Text == "") return;

c.cmd.CommandText = "select count(\*) from " + currentTable + " where list = '" + txtAdd.Text + "'";

if ((int)c.cmd.ExecuteScalar() <= 0)

{

c.cmd.CommandText = "insert into " + currentTable + " values(@list)";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@list", SqlDbType.VarChar).Value = txtAdd.Text;

c.cmd.ExecuteNonQuery();

}

txtAdd.Clear();

txtAdd.Focus();

SelectList();

}

privatevoid btnDelete\_Click(object sender, EventArgs e)

{

c.cmd.CommandText = "select count(\*) from " + currentTable + " where list = '" + adderList.SelectedItem.ToString() +"'";

if ((int)c.cmd.ExecuteScalar() > 0)

{

c.cmd.CommandText = "delete from " + currentTable + " where list=@list";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@list", SqlDbType.VarChar).Value = adderList.SelectedItem.ToString();

c.cmd.ExecuteNonQuery();

adderList.Items.Remove(adderList.SelectedItem.ToString());

}

SelectList();

}

}

}

05 Members :

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using Microsoft.VisualBasic;

using System.Data.SqlClient;

using System.Text.RegularExpressions;

using System.Reflection;

namespace Library\_Auto

{

publicpartialclassMembers\_05 : Form

{

publicAdministration\_03 adminInstance;

privateConnect c = newConnect();

privateSqlDataAdapter adp = newSqlDataAdapter();

privateDataTable dt = newDataTable();

privateBindingSource bds = newBindingSource();

string state, searchHandler;

//Circulation to MemberDetails

publicbool sentByCirculation;

publicstring memIDCirc;

public Members\_05() { InitializeComponent(); }

publicvoid MembersButtonActions(string actionText)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

switch (actionText)

{

case"Add": btnAdd\_Click(this, null); break;

case"View": btnView\_Click(this, null); break;

case"Edit": btnEdit\_Click(this, null); break;

case"Delete": btnDelete\_Click(this, null); break;

case"Search": btnSearch\_Click(this, null); break;

default: return;

}

}

protectedoverridevoid OnFormClosing(FormClosingEventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

base.OnFormClosing(e);

switch (e.CloseReason)

{

caseCloseReason.TaskManagerClosing:

caseCloseReason.WindowsShutDown:

return;

default: break;

}

if (!sentByCirculation && adminInstance != null) { adminInstance.Show(); }

}

privatevoid btnClear\_Click(object sender, EventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

dt.Clear();

switch (state)

{

case"View":

state = "";

lblCurrOps.Text = "Current Operation: " + state;

break;

case"Edit":

state = "";

lblCurrOps.Text = "Current Operation: " + state;

grpMemInfo.Enabled = false;

grpSubsInfo.Enabled = false;

break;

case"Add":

state = "Add";

lblCurrOps.Text = "Current Operation: " + state;

break;

case"Delete":

state = "";

lblCurrOps.Text = "Current Operation: " + state;

break;

}

SetControlsToDefault(grpMemInfo);

SetControlsToDefault(grpSubsInfo);

}

privatevoid btnAdd\_Click(object sender, EventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

try

{

grpMemInfo.Enabled = true;

grpSubsInfo.Enabled = true;

txtMemID.ReadOnly = false;

ddlMemberType.Enabled = true;

grpNav.Enabled = false;

btnClear\_Click(this, EventArgs.Empty);

dtDOB.Value = newDateTime(2000, 01, 01);

dtExp.Value = DateTime.Today.AddYears(1);

state = "Add";

lblCurrOps.Text = "Current Operation: " + state;

txtFine.Text = "0";

btnSubmit.Enabled = true;

}

catch (Exception ex)

{

adminInstance.errObj.txtException.Text = ex.ToString();

adminInstance.errObj.Show();

this.Close();

}

}

privatevoid btnView\_Click(object sender, EventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

try

{

grpMemInfo.Enabled = false;

grpSubsInfo.Enabled = false;

btnSubmit.Enabled = false;

state = "View";

lblCurrOps.Text = "Current Operation: " + state;

grpNav.Enabled = true;

c.cmd.CommandText = "select \* from MembersTable";

adp.SelectCommand = c.cmd;

dt.Clear();

adp.Fill(dt);

bds.DataSource = dt;

bds.Sort = "memberid ASC";

ClearMembersDataBindings();

AddMembersDataBindings();

}

catch (Exception ex)

{

adminInstance.errObj.txtException.Text = ex.ToString();

adminInstance.errObj.Show();

this.Close();

}

}

privatevoid btnSearch\_Click(object sender, EventArgs e)

{

string memberid;

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

try

{

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

btnClear\_Click(this, EventArgs.Empty);

if (sentByCirculation)

{

memberid = memIDCirc;

}

else memberid = Interaction.InputBox("Enter the Member ID:", "Search using Member ID", "", -1, -1);

c.cmd.CommandText = "select count(\*) from MembersTable where memberid ='" + memberid + "'";

if ((int)c.cmd.ExecuteScalar() <= 0)

{

MessageBox.Show("Record not found!", "Search", MessageBoxButtons.OK, MessageBoxIcon.Warning);

grpMemInfo.Enabled = false;

grpSubsInfo.Enabled = false;

searchHandler = "Not Found";

return;

}

state = "View";

lblCurrOps.Text = "Current Operation: " + "Search";

grpSubsInfo.Enabled = false;

grpMemInfo.Enabled = false;

grpNav.Enabled = false;

c.cmd.CommandText = "select \* from MembersTable where memberid ='" + memberid + "'";

adp.SelectCommand = c.cmd;

dt.Clear();

adp.Fill(dt);

bds.DataSource = dt;

ClearMembersDataBindings();

AddMembersDataBindings();

if (sentByCirculation)

{

foreach (Control ctrl inthis.Controls) { if (ctrl isButton) { (ctrl asButton).Enabled = false; } }

}

}

catch (SqlException) { this.Close(); }

}

privatevoid btnEdit\_Click(object sender, EventArgs e)

{

searchHandler = "";

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

if (state != "View")

{

btnSearch\_Click(this, EventArgs.Empty);

if (searchHandler == "Not Found") return;

}

txtMemID.ReadOnly = true;

ddlMemberType.Enabled = false;

grpMemInfo.Enabled = true;

grpSubsInfo.Enabled = true;

grpNav.Enabled = false;

btnSubmit.Enabled = true;

state = "Edit";

lblCurrOps.Text = "Current Operation: " + state;

}

privatevoid btnDelete\_Click(object sender, EventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

try

{

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

if (state == "View")

{

if (ddlStatus.Text == "With Issue")

{

MessageBox.Show("Cannot delete a Member with an Issue!");

return;

}

DialogResult dRes = MessageBox.Show("Delete this record? Member ID = " + txtMemID.Text, "Delete Records", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);

if (dRes == DialogResult.Cancel)

{

Interaction.Beep();

return;

}

elseif (dRes == DialogResult.OK)

{

c.cmd.CommandText = "delete from MembersTable where memberid ='" + txtMemID.Text + "'";

c.cmd.ExecuteNonQuery();

MessageBox.Show("Record deleted!", "Delete Records", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

UpdateStats();

state = "";

}

}

else

{

state = "View";

lblCurrOps.Text = "Current Operation: " + "Delete";

grpSubsInfo.Enabled = false;

grpMemInfo.Enabled = false;

grpNav.Enabled = false;

string memberid = Interaction.InputBox("Enter the Member ID:", "Delete using Member ID", "", -1, -1);

c.cmd.CommandText = "select count(\*) from MembersTable where memberid ='" + memberid + "'";

if ((int)c.cmd.ExecuteScalar() <= 0)

{

MessageBox.Show("Record not found!", "Delete", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

c.cmd.CommandText = "select \* from MembersTable where memberid ='" + memberid + "'";

adp.SelectCommand = c.cmd;

adp.Fill(dt);

bds.DataSource = dt;

ClearMembersDataBindings();

AddMembersDataBindings();

//Do not delete members with Issue

if (ddlStatus.Text == "With Issue")

{

MessageBox.Show("Cannot delete a Member with an Issue!");

return;

}

c.cmd.CommandText = "delete from MembersTable where memberid ='" + memberid + "'";

c.cmd.ExecuteNonQuery();

MessageBox.Show("Record deleted!", "Delete Records", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

UpdateStats();

state = "";

}

}

catch (SqlException) { this.Close(); }

}

void ClearMembersDataBindings()

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

foreach (Control ctrl in grpMemInfo.Controls) { ctrl.DataBindings.Clear();}

foreach (Control ctrl in grpSubsInfo.Controls) { ctrl.DataBindings.Clear(); }

}

void AddMembersDataBindings()

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

try

{

ddlMemberType.DataBindings.Add("text", bds, "membertype");

txtMemID.DataBindings.Add("text", bds, "memberid");

ddlStatus.DataBindings.Add("text", bds, "status");

txtFName.DataBindings.Add("text", bds, "firstname");

txtLName.DataBindings.Add("text", bds, "lastname");

lblGender.DataBindings.Add("text", bds, "gender");

SetRadoGender();

dtDOB.DataBindings.Add("text", bds, "dob");

txtCourse.DataBindings.Add("text", bds, "course");

txtBranch.DataBindings.Add("text", bds, "branch");

txtDept.DataBindings.Add("text", bds, "dept");

dtSubsciption.DataBindings.Add("text", bds, "subsdate");

dtExp.DataBindings.Add("text", bds, "expdate");

txtFatherName.DataBindings.Add("text", bds, "fathername");

txtOcupation.DataBindings.Add("text", bds, "occupation");

txtPhNo.DataBindings.Add("text", bds, "phoneno");

txtEmail.DataBindings.Add("text", bds, "email");

txtAd1.DataBindings.Add("text", bds, "address1");

txtAd2.DataBindings.Add("text", bds, "address2");

txtPin.DataBindings.Add("text", bds, "pincode");

txtState.DataBindings.Add("text", bds, "state");

txtFine.DataBindings.Add("text", bds, "fine");

}

catch (SqlException) { throw; }

}

void SetRadoGender()

{

radioFem.Checked = (lblGender.Text == "F") ? true : false;

radioMale.Checked = (lblGender.Text == "M") ? true : false;

}

privatevoid btnFirst\_Click(object sender, EventArgs e) {bds.MoveFirst();SetRadoGender();}

privatevoid btnPrev\_Click(object sender, EventArgs e) {bds.MovePrevious();SetRadoGender();}

privatevoid btnNext\_Click(object sender, EventArgs e){bds.MoveNext();SetRadoGender();}

privatevoid btnLast\_Click(object sender, EventArgs e){bds.MoveLast();SetRadoGender();}

publicvoid SetControlsToDefault(GroupBox dummy)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

foreach (Control ctrl in dummy.Controls)

{

if (ctrl isTextBox)

{

if ((ctrl asTextBox).Name == txtMemID.Name && state == "Edit") { }

else

{

(ctrl asTextBox).Clear();

(ctrl asTextBox).Enabled = true;

}

}

if (ctrl isComboBox)

{

if ((ctrl asComboBox).Name == ddlMemberType.Name && state == "Edit") { }

else

{

(ctrl asComboBox).SelectedItem = null;

(ctrl asComboBox).Text = "";

}

}

if (ctrl isDateTimePicker)

(ctrl asDateTimePicker).Value = DateTime.Today;

if (ctrl isRadioButton)

(ctrl asRadioButton).Checked = false;

}

}

void MemTypeLibraryStaffControls()

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

string staffID = Interaction.InputBox("Enter Staff ID:", "Load Staff Info from Server", "", -1, -1);

if (staffID == "") return;

c.cmd.CommandText = "select \* from StaffTable where staffid='" + staffID + "'";

adp.SelectCommand = c.cmd;

dt.Clear();

adp.Fill(dt);

}

privatevoid btnSubmit\_Click(object sender, EventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

bool isInvalid= false;

string process = state;

bool isEmailInvalid = false;

isEmailInvalid = InvalidEmail();

isInvalid = EmptyValidations();

if (isInvalid || isEmailInvalid)

{

return;

}

else lblError.Text = "...";

//Check for SubsDate > ExpDate

if (dtSubsciption.Value > dtExp.Value)

{

MessageBox.Show("Subscription date cannot be greater than Expiry date!");

return;

}

if (dtExp.Value <DateTime.Today)

{

MessageBox.Show("Expiry date must be greater than Today's date");

return;

}

try

{

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

switch (state)

{

case"Add":

c.cmd.CommandText = "select count(\*) from MembersTable where memberid ='" + txtMemID.Text + "'";

if ((int)c.cmd.ExecuteScalar() > 0)

{

Interaction.Beep();

MessageBox.Show("Member with same Member ID found!", "Add Member");

txtMemID.Focus();

return;

}

c.cmd.CommandText = "insert into MembersTable values (@membertype, @memberid, @firstname, @lastname,";

c.cmd.CommandText += " @course, @branch, @dept, @dob, @gender, @status, @subsdate, @expdate, @fathername,";

c.cmd.CommandText += " @occupation, @phoneno, @email, @address1, @address2, @pincode, @state, @fine)";

break;

case"Edit":

c.cmd.CommandText = "update MembersTable set firstname=@firstname, lastname=@lastname, course=@course, branch=@branch, dept=@dept,";

c.cmd.CommandText += " dob=@dob, gender=@gender, status=@status, subsdate=@subsdate, expdate=@expdate,";

c.cmd.CommandText += " fathername=@fathername, occupation=@occupation, phoneno=@phoneno, email=@email,";

c.cmd.CommandText += " address1=@address1, address2=@address2, pincode=@pincode, state=@state";

c.cmd.CommandText += " where membertype=@membertype and memberid=@memberid";

txtMemID.ReadOnly = false;

ddlMemberType.Enabled = true;

break;

default: return;

}

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@membertype", SqlDbType.VarChar).Value = ddlMemberType.Text;

c.cmd.Parameters.Add("@memberid", SqlDbType.VarChar).Value = txtMemID.Text;

c.cmd.Parameters.Add("@firstname", SqlDbType.VarChar).Value = txtFName.Text;

c.cmd.Parameters.Add("@lastname", SqlDbType.VarChar).Value = txtLName.Text;

c.cmd.Parameters.Add("@course", SqlDbType.VarChar).Value = txtCourse.Text;

c.cmd.Parameters.Add("@branch", SqlDbType.VarChar).Value = txtBranch.Text;

c.cmd.Parameters.Add("@dept", SqlDbType.VarChar).Value = txtDept.Text;

c.cmd.Parameters.Add("@dob", SqlDbType.VarChar).Value = dtDOB.Value.ToString();

string genderStr = (radioMale.Checked == true) ? "M" : "F";

c.cmd.Parameters.Add("@gender", SqlDbType.VarChar).Value = genderStr;

c.cmd.Parameters.Add("@status", SqlDbType.VarChar).Value = ddlStatus.Text;

c.cmd.Parameters.Add("@subsdate", SqlDbType.DateTime).Value = dtSubsciption.Value.ToString();

c.cmd.Parameters.Add("@expdate", SqlDbType.DateTime).Value = dtExp.Value.ToString();

c.cmd.Parameters.Add("@fathername", SqlDbType.VarChar).Value = txtFatherName.Text;

c.cmd.Parameters.Add("@occupation", SqlDbType.VarChar).Value = txtOcupation.Text;

c.cmd.Parameters.Add("@phoneno", SqlDbType.VarChar).Value = txtPhNo.Text;

c.cmd.Parameters.Add("email", SqlDbType.VarChar).Value = txtEmail.Text;

c.cmd.Parameters.Add("address1", SqlDbType.VarChar).Value = txtAd1.Text;

c.cmd.Parameters.Add("address2", SqlDbType.VarChar).Value = txtAd2.Text;

c.cmd.Parameters.Add("pincode", SqlDbType.VarChar).Value = txtPin.Text;

c.cmd.Parameters.Add("state", SqlDbType.VarChar).Value = txtState.Text;

c.cmd.Parameters.Add("fine", SqlDbType.Int).Value = Convert.ToInt32(txtFine.Text);

c.cmd.ExecuteNonQuery();

UpdateStats();

MessageBox.Show(process + " Complete!", process);

grpMemInfo.Enabled = false;

grpSubsInfo.Enabled = false;

btnSubmit.Enabled = false;

}

catch (SqlException) { this.Close(); }

}

privatebool EmptyValidations()

{

foreach (Control ctrl in grpMemInfo.Controls)

{

if (ctrl isTextBox)

{

if ((ctrl asTextBox).Text == "")

{

lblError.Text = "Error: Empty Textbox";

(ctrl asTextBox).Focus();

returntrue;

}

else lblError.Text = "...";

}

if (ctrl isComboBox)

{

if ((ctrl asComboBox).SelectedItem == null)

{

lblError.Text = "Error: Unselected Item";

(ctrl asComboBox).Focus();

returntrue;

}

else lblError.Text = "...";

}

}

foreach (Control ctrl in grpSubsInfo.Controls)

{

if (ctrl isTextBox)

{

if ((ctrl asTextBox).Text == "")

{

lblError.Text = "Error: Empty Textbox";

(ctrl asTextBox).Focus();

returntrue;

}

else lblError.Text = "...";

}

if (ctrl isComboBox)

{

if ((ctrl asComboBox).SelectedItem == null)

{

lblError.Text = "Error: Unselected Item";

(ctrl asComboBox).Focus();

returntrue;

}

else lblError.Text = "...";

}

}

lblError.Text = "...";

returnfalse;

}

void UpdateStats()

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

try

{

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

c.cmd.CommandText = "select count(\*) from MembersTable";

lblTotalMembers.Text = "Total Member :" + (int)c.cmd.ExecuteScalar();

c.cmd.CommandText = "select count(\*) from MembersTable where status ='Active'";

lblActiveMembers.Text = "Members Active :" + (int)c.cmd.ExecuteScalar();

c.cmd.CommandText = "select count(\*) from MembersTable where status ='Inactive'";

lblMembersInactive.Text = "Members Inactive :" + (int)c.cmd.ExecuteScalar();

c.cmd.CommandText = "select count(\*) from MembersTable where status ='With Issue'";

lblIssues.Text = "Members with Issues :" + (int)c.cmd.ExecuteScalar();

}

catch (SqlException) { }

}

privatevoid Members\_05\_Load(object sender, EventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

grpNav.Enabled = false;

UpdateStats();

DdlLoad();

if (sentByCirculation)

{

btnSearch\_Click(this, EventArgs.Empty);

}

}

privatevoid DdlLoad()

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

c.cmd.CommandText = "select memtype from MemberParameters";

DataTable memParaTable = newDataTable();

adp.SelectCommand = c.cmd;

adp.Fill(memParaTable);

for (int i = 0; i < memParaTable.Rows.Count; i++)

{

if (!ddlMemberType.Items.Contains((string)memParaTable.Rows[i].ItemArray[0]))

ddlMemberType.Items.Add((string)memParaTable.Rows[i].ItemArray[0]);

}

}

privatevoid ddlMemType\_SelectionChangeCommited(object sender, EventArgs e)

{

if (state != "View")

{

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

c.cmd.CommandText = "select mtypeidentifier from MemberParameters where memtype ='" + ddlMemberType.SelectedItem.ToString() + "'";

txtMemID.Text = (string)c.cmd.ExecuteScalar();

if (state == "Add"&& ddlMemberType.SelectedItem.ToString() != "Student")

{

txtCourse.Text = "-";

txtBranch.Text = "-";

txtDept.Text = "-";

}

else

{

txtCourse.Text = "";

txtBranch.Text = "";

txtDept.Text = "";

}

if (state == "Add"&& ddlMemberType.SelectedItem.ToString() == "Library Staff")

{

txtMemID.Clear();

string staffID = Interaction.InputBox("MemberType is set to Staff!"

+ "\nEnter the Staff Username to automatically insert the Staff Details", "Enter StaffID", "", -1, -1);

c.cmd.CommandText = "select count(\*) from StaffTable where staffid='" + staffID + "'";

if ((int)c.cmd.ExecuteScalar() <= 0)

{

MessageBox.Show("Staff Not found!");

btnAdd\_Click(this, EventArgs.Empty);

return;

}

else

{

DataTable dtStaff = newDataTable();

c.cmd.CommandText = "select \* from StaffTable where staffid ='" + staffID + "'";

adp.SelectCommand = c.cmd;

adp.Fill(dtStaff);

txtMemID.Text = (string) dtStaff.Rows[0].ItemArray[0];

txtFName.Text = (string)dtStaff.Rows[0].ItemArray[2];

txtLName.Text = (string)dtStaff.Rows[0].ItemArray[3];

radioFem.Checked = ((string)dtStaff.Rows[0].ItemArray[4] == "F") ? true : false;

radioMale.Checked = ((string)dtStaff.Rows[0].ItemArray[4] == "M") ? true : false;

if (dtStaff.Rows[0].ItemArray[5] != DBNull.Value)

dtDOB.Value = Convert.ToDateTime(dtStaff.Rows[0].ItemArray[5]);

txtAd1.Text = (string)dtStaff.Rows[0].ItemArray[6];

txtAd2.Text = (string)dtStaff.Rows[0].ItemArray[7];

txtPin.Text = (string)dtStaff.Rows[0].ItemArray[8];

txtState.Text = (string)dtStaff.Rows[0].ItemArray[9];

txtEmail.Text = (string)dtStaff.Rows[0].ItemArray[10];

txtPhNo.Text = (string)dtStaff.Rows[0].ItemArray[11];

}

}

}

}

//Number Validations

privatevoid NumValidations(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar)

&& e.KeyChar != 0)

{

lblError.Text = "Enter only numbers!";

e.Handled = true;

return;

}

lblError.Text = "...";

}

//Alphabets Validations

privatevoid AlphaValidations(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsLetter(e.KeyChar))

{

lblError.Text = "Enter only Letters!";

e.Handled = true;

return;

}

lblError.Text = "...";

}

//Alpha Num Validations

privatevoid AlphaNumValidations(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsLetterOrDigit(e.KeyChar)

&& !char.IsWhiteSpace(e.KeyChar)

&& !char.Equals(e.KeyChar, '-'))

{

lblError.Text = "Enter only Letters or Digits";

e.Handled = true;

return;

}

lblError.Text = "...";

}

//Email Validations

publicbool InvalidEmail()

{

string emailExpr = null;

emailExpr = "^([0-9a-zA-Z]([-\\.\\w]\*[0-9a-zA-Z])\*@([0-9a-zA-Z][-\\w]\*[0-9a-zA-Z]\\.)+[a-zA-Z]{2,9})$";

if (Regex.IsMatch(txtEmail.Text, emailExpr))

{

//Since the Email is Valid, retutn IsInvalidEmail as False.

lblError.Text = "...";

returnfalse;

}

lblError.Text = "Invalid Email!";

returntrue;

}

}

}

05b MemberParameters:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace Library\_Auto

{

publicpartialclassMemberParameters\_05b : Form

{

publicAdministration\_03 adminInstance;

Connect c = newConnect();

SqlDataAdapter adp = newSqlDataAdapter();

DataTable dt = newDataTable();

public MemberParameters\_05b()

{

InitializeComponent();

}

protectedoverridevoid OnFormClosing(FormClosingEventArgs e)

{

base.OnFormClosing(e);

switch (e.CloseReason)

{

caseCloseReason.WindowsShutDown:

caseCloseReason.TaskManagerClosing:

return;

default: break;

}

if (adminInstance != null) adminInstance.Show();

}

privatevoid btnMaster\_Click(object sender, EventArgs e)

{

try

{

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

c.cmd.CommandText = "select count(\*) from MemberParameters where memtype='" + txtMemtype.Text + "'";

if ((int)c.cmd.ExecuteScalar() <= 0)

{

DialogResult diaRes = MessageBox.Show("Enter this Member parameter?\nMember Parameters cannot be deleted!", "Add Member Parameter",

MessageBoxButtons.OKCancel, MessageBoxIcon.Information);

if (diaRes == DialogResult.Cancel) return;

c.cmd.CommandText = "insert into MemberParameters values(@memtype, @mtypeidentifier, @maxissuedays, @fineperday)";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@memtype", SqlDbType.VarChar).Value = txtMemtype.Text;

c.cmd.Parameters.Add("@mtypeidentifier", SqlDbType.VarChar).Value = txtMtypeID.Text;

c.cmd.Parameters.Add("@maxissuedays", SqlDbType.Int).Value = Convert.ToInt32(txtMaxIssuedays.Text);

c.cmd.Parameters.Add("@fineperday", SqlDbType.Int).Value = Convert.ToInt32(txtFinePD.Text);

c.cmd.ExecuteNonQuery();

MessageBox.Show("Member Parameter Added! \nMembertype : " + txtMemtype.Text);

}

elseMessageBox.Show("Parameter is already present! \nMembertype : " + txtMemtype.Text);

}

catch (SqlException) { throw; }

}

privatevoid checkUnlock\_CheckedChanged(object sender, EventArgs e)

{

if (checkUnlock.Checked == true)

{

btnAdd.Enabled = true;

}

elseif (checkUnlock.Checked == false)

{

btnAdd.Enabled = false;

}

}

privatevoid NumValidations(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar)

&& e.KeyChar != 0)

{

lblValidation.Text = "Enter only numbers!";

e.Handled = true;

return;

}

lblValidation.Text = "...";

}

privatevoid AlphaValidations(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar)

&& !char.IsLetter(e.KeyChar)

&& !char.IsWhiteSpace(e.KeyChar))

{

lblValidation.Text = "Enter only Letters!";

e.Handled = true;

return;

}

lblValidation.Text = "...";

}

}

}

06 Circulation :

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using Microsoft.VisualBasic;

using System.Reflection;

namespace Library\_Auto

{

publicpartialclassCirculation\_06 : Form

{

publicAdministration\_03 adminInstance;

Connect c = newConnect();

SqlDataAdapter adp = newSqlDataAdapter();

DataTable DTIssue = newDataTable();

DataTable DTMem = newDataTable();

DataTable DTAccReg = newDataTable();

DataTable DTStock = newDataTable();

string memStatus, bookStatus;

int index;

bool isViewing = false;

string issueIDBookMemInfo;

public Circulation\_06() { InitializeComponent(); }

privatevoid Circulation\_06\_Load(object sender, EventArgs e)

{

dtToday.Value = DateTime.Today;

try

{

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

c.cmd.CommandText = "select count(\*) from IssueTable";

lblStats.Text += "" + (int)c.cmd.ExecuteScalar();

}

catch (SqlException) { }

}

protectedoverridevoid OnFormClosing(FormClosingEventArgs e)

{

base.OnFormClosing(e);

switch (e.CloseReason)

{

caseCloseReason.WindowsShutDown:

caseCloseReason.TaskManagerClosing:

return;

default: break;

}

if (adminInstance != null) adminInstance.Show();

}

privatevoid btnClear\_Click(object sender, EventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

DTIssue.Reset();

DTMem.Reset();

DTAccReg.Reset();

DTStock.Reset();

txtAccno.Clear();

txtMemID.Clear();

txtCopy.Clear();

txtAccno.ReadOnly = false;

txtMemID.ReadOnly = false;

txtCopy.ReadOnly = false;

ClearGroupBoxContent(grpBookInfo);

ClearGroupBoxContent(grpMemberInfo);

ClearGroupBoxContent(grpIssueDetails);

lblError.Text = "...";

checkBoxOvernightCirc.Checked = false;

}

privatevoid StopViewingClear\_MouseClick(object sender, MouseEventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

if (isViewing)

{

isViewing = false;

grpNavigator.Enabled = false;

btnIssue.Enabled = true;

btnReturn.Enabled = true;

btnRenew.Enabled = true;

ClearGroupBoxContent(grpIssueDetails);

}

}

privatevoid btnIssue\_Click(object sender, EventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

try

{

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

//Empty Validaitions

bool isInvalid = EmptyValidation();

if (isInvalid)

{

lblError.Text = "Error : Empty Textbox!";

Interaction.Beep();

return;

}

else lblError.Text = "...";

btnBookInfo\_Click(this, EventArgs.Empty);

btnMemberInfo\_Click(this, EventArgs.Empty);

//Readonly check to proceed furthur

if (txtAccno.ReadOnly == false

|| txtCopy.ReadOnly == false

|| txtMemID.ReadOnly == false)

{

MessageBox.Show("Invalidated Issue Status!"

+ "\nCheck the Accno, Copy No and MemberID", "Invalid Status!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

//MemberStatus Validations

bool memReturn = MemStatusValidations();

if (memReturn) return;

//BookStatus Validations

if (bookStatus == "Unavailable")

{

string msg1, msg2, msg3;

msg1 = "Book is Unavailable!";

msg2 = "\nThe Book might be already Issued.";

msg3 = "\n\nPlease check the Copy Number of the Book.";

MessageBox.Show(msg1 + msg2 + msg3, "Book Unavailable",

MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

//DueDate Validations

int daysFinal = SetDueDays(txtMemID.Text, DateTime.Today);

if (daysFinal == -1) return;

//Issue Master

int issueID = -1;

c.cmd.CommandText = "insert into IssueTable values(@memberid, @accno, @copyno, @issuedate, @duedate, @returndate, @fine, @status, @renewdate, @staffIDIssue, @staffIDRenew, @staffIDReturn)"

+ " set @issueid = SCOPE\_IDENTITY()";

c.cmd.Parameters.Clear();

SqlParameter outIssueID = newSqlParameter("@issueid", SqlDbType.Int);

outIssueID.Direction = ParameterDirection.Output;

c.cmd.Parameters.Add(outIssueID);

c.cmd.Parameters.Add("@memberid", SqlDbType.VarChar).Value = txtMemID.Text;

c.cmd.Parameters.Add("@accno", SqlDbType.VarChar).Value = txtAccno.Text;

c.cmd.Parameters.Add("@copyno", SqlDbType.VarChar).Value = txtCopy.Text;

DateTime today = DateTime.Today;

c.cmd.Parameters.Add("@issuedate", SqlDbType.DateTime).Value = today;

c.cmd.Parameters.Add("@duedate", SqlDbType.DateTime).Value = today.AddDays(daysFinal);

c.cmd.Parameters.Add("@returndate", SqlDbType.DateTime).Value = DBNull.Value;

c.cmd.Parameters.Add("@fine", SqlDbType.Int).Value = DBNull.Value;

c.cmd.Parameters.Add("@status", SqlDbType.VarChar).Value = "Issued";

c.cmd.Parameters.Add("@renewdate", SqlDbType.DateTime).Value = DBNull.Value;

c.cmd.Parameters.Add("@staffIDIssue", SqlDbType.VarChar).Value = adminInstance.staffID;

c.cmd.Parameters.Add("@staffIDRenew", SqlDbType.VarChar).Value = DBNull.Value;

c.cmd.Parameters.Add("@staffIDReturn", SqlDbType.VarChar).Value = DBNull.Value;

c.cmd.ExecuteNonQuery();

if (outIssueID.Value != DBNull.Value) { issueID = (int)outIssueID.Value; }

//MembersTable and MembersIssues

MembersTableUpdate();

//AccRegIssues, StockTable and StockIssues

StockTableUpdate();

//GroupBox Updates

UpdateGrpIssue(issueID);

c.cmd.CommandText = "select MAX(issueid) from issueTable";

lblStats.Text = "Total Book Issue :" + (int)c.cmd.ExecuteScalar();

MessageBox.Show("Book Issued! \nIssue ID:" + issueID, "Issue Book", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

btnBookInfo\_Click(this, null);

btnMemberInfo\_Click(this, null);

}

catch (Exception ex)

{

adminInstance.errObj.txtException.Text = ex.ToString();

adminInstance.errObj.Show();

this.Close();

}

}

void MembersTableUpdate()

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

//MembersTable Update

c.cmd.Parameters.Clear();

c.cmd.CommandText = "update MembersTable set status=@status where memberid ='" + txtMemID.Text + "'";

c.cmd.Parameters.Add("@status", SqlDbType.VarChar).Value = "With Issue";

c.cmd.ExecuteNonQuery();

//MembersIssues Insert

//MembersIssues has a additional column as issueid with identity specification.

//This Table is used to prevent changes inside of MembersTable

//which could render inaccuracies in IssueDetails.

c.cmd.CommandText = "insert into MembersIssues"

+ " Select \* from MembersTable where memberid ='" + txtMemID.Text + "'";

c.cmd.ExecuteNonQuery();

}

void StockTableUpdate()

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

//StockTable Update

c.cmd.Parameters.Clear();

c.cmd.CommandText = "update StockTable set status=@status where accno ='" + txtAccno.Text + "' and copyno ='" + txtCopy.Text + "'";

c.cmd.Parameters.Add("@status", SqlDbType.VarChar).Value = "Unavailable";

c.cmd.ExecuteNonQuery();

//AccRegIssues Insert

c.cmd.CommandText = "insert into AccRegIssues"

+ " Select \* from AccRegTable where accno ='" + txtAccno.Text + "'";

c.cmd.ExecuteNonQuery();

//StockIssues Insert

c.cmd.CommandText = "insert into StockIssues"

+ " Select \* from StockTable where accno='" + txtAccno.Text + "'and copyno='" + txtCopy.Text + "'";

c.cmd.ExecuteNonQuery();

}

//Does the Same as SetValues()

void UpdateGrpIssue(int issueID)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

ClearGroupBoxContent(grpIssueDetails);

c.cmd.CommandText = "select \* from IssueTable where issueid ='" + issueID + "'";

adp.SelectCommand = c.cmd;

DTIssue.Clear();

adp.Fill(DTIssue);

txtIssueID.Text = "" + DTIssue.Rows[0].ItemArray[0];

IssueDatePicker.Value = (DateTime)DTIssue.Rows[0].ItemArray[4];

DueDatePicker.Value = (DateTime)DTIssue.Rows[0].ItemArray[5];

if (DTIssue.Rows[0].ItemArray[6] != DBNull.Value)

{

ReturnDatePicker.Value = (DateTime)DTIssue.Rows[0].ItemArray[6];

ReturnDatePicker.Visible = true;

lblReturnDate.Visible = true;

}

else

{

ReturnDatePicker.Visible = false;

lblReturnDate.Visible = false;

}

txtIssueFine.Text = "" + DTIssue.Rows[0].ItemArray[7];

txtIssueStatus.Text = "" + DTIssue.Rows[0].ItemArray[8];

if (DTIssue.Rows[0].ItemArray[9] != DBNull.Value)

{

renewDatePicker.Value = (DateTime)DTIssue.Rows[0].ItemArray[9];

renewDatePicker.Visible = true;

lblRenewDate.Visible = true;

}

else

{

renewDatePicker.Visible = false;

lblRenewDate.Visible = false;

}

txtStaffIDIssue.Text = "" + DTIssue.Rows[0].ItemArray[10];

txtStaffIDRenew.Text = "" + DTIssue.Rows[0].ItemArray[11];

txtStaffIDReturn.Text = "" + DTIssue.Rows[0].ItemArray[12];

}

privatevoid btnFine\_Click(object sender, EventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

Fine\_06b objFine = newFine\_06b();

objFine.Show();

}

privatevoid btnBookInfo\_Click(object sender, EventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

try

{

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open();}

ClearGroupBoxContent(grpBookInfo);

if (isViewing)

{

string accRegCText, stockCText;

accRegCText = "select \* from AccRegIssues where issueid ='" + issueIDBookMemInfo + "'";

stockCText = "select \* from StockIssues where issueid ='" + issueIDBookMemInfo + "'";

BookInfoSetup(accRegCText, stockCText);

}

else

{

c.cmd.CommandText = "select count(\*) from AccRegTable where accno ='" + txtAccno.Text + "'";

if ((int)c.cmd.ExecuteScalar() <= 0)

{

MessageBox.Show("Book not found! \nPlease register the book using the Accession Register!", "Book Information",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else

{

c.cmd.CommandText = "select count(\*) from StockTable where accno='" + txtAccno.Text + "' and copyno='" + txtCopy.Text + "'";

if ((int)c.cmd.ExecuteScalar() <= 0)

{

MessageBox.Show("Copy not found! \nPlease Recheck the Copy Number",

"Book Information", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else

{

string accRegCText, stockCText;

accRegCText = "select \* from AccRegTable where accno ='" + txtAccno.Text + "'";

stockCText = "select \* from StockTable where accno ='" + txtAccno.Text + "' and copyno ='" + txtCopy.Text + "'";

BookInfoSetup(accRegCText, stockCText);

if (bookStatus == "Available")

{

txtAccno.ReadOnly = true;

txtCopy.ReadOnly = true;

}

}

}

}

}

catch (Exception ex)

{

adminInstance.errObj.txtException.Text = ex.ToString();

adminInstance.errObj.Show();

this.Close();

}

}

void BookInfoSetup(string accRegCText, string stockCText)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

//Sets the AccReg/AccRegIssues and Stock/StockIssues

int i;

i = (isViewing) ? 1 : 0;

lblError.Text = "...";

c.cmd.CommandText = accRegCText;

adp.SelectCommand = c.cmd;

DTAccReg.Reset();

adp.Fill(DTAccReg);

txtBookType.Text = (string)DTAccReg.Rows[0].ItemArray[2 + i];

txtTitle.Text = (string)DTAccReg.Rows[0].ItemArray[3 + i];

txtAuthor.Text = (string)DTAccReg.Rows[0].ItemArray[5 + i];

txtYear.Text = (string)DTAccReg.Rows[0].ItemArray[17 + i];

txtLanguage.Text = (string)DTAccReg.Rows[0].ItemArray[10 + i];

txtPublisher.Text = (string)DTAccReg.Rows[0].ItemArray[19 + i];

c.cmd.CommandText = stockCText;

adp.SelectCommand = c.cmd;

DTStock.Reset();

adp.Fill(DTStock);

txtEdn.Text = (string)DTStock.Rows[0].ItemArray[6 + i];

txtPages.Text = (string)DTStock.Rows[0].ItemArray[14 + i];

txtBinding.Text = (string)DTStock.Rows[0].ItemArray[16 + i];

txtBookStatus.Text = (string)DTStock.Rows[0].ItemArray[10 + i];

bookStatus = (string)DTStock.Rows[0].ItemArray[10 + i];

}

privatevoid btnMemberInfo\_Click(object sender, EventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

try

{

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

ClearGroupBoxContent(grpMemberInfo);

if (isViewing)

{

string memCText = "select \* from MembersIssues where issueid ='" + issueIDBookMemInfo + "'";

MemInfoSetup(memCText);

}

else

{

c.cmd.CommandText = "select count(\*) from MembersTable where memberid='" + txtMemID.Text + "'";

if ((int)c.cmd.ExecuteScalar() <= 0)

{

MessageBox.Show("Member not found! \nPlease register the Member",

"Member Information", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

string memCText = "select \* from MembersTable where memberid='" + txtMemID.Text + "'";

MemInfoSetup(memCText);

if (memStatus == "Active") txtMemID.ReadOnly = true;

}

}

}

catch (Exception ex)

{

adminInstance.errObj.txtException.Text = ex.ToString();

adminInstance.errObj.Show();

this.Close();

}

}

void MemInfoSetup(string memCText)

{

//Sets up the Member Info with MembersTable/MembersIssues

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

int i;

i = (isViewing) ? 1 : 0;

lblError.Text = "...";

DTMem.Reset();

c.cmd.CommandText = memCText;

adp.SelectCommand = c.cmd;

adp.Fill(DTMem);

txtMType.Text = (string)DTMem.Rows[0].ItemArray[0 + i];

txtName.Text = (string)DTMem.Rows[0].ItemArray[2 + i];

txtEmail.Text = (string)DTMem.Rows[0].ItemArray[15 + i];

txtPhNo.Text = (string)DTMem.Rows[0].ItemArray[14 + i];

dtSubs.Text = Convert.ToString(DTMem.Rows[0].ItemArray[10 + i]);

dtExp.Text = Convert.ToString(DTMem.Rows[0].ItemArray[11 + i]);

txtAddress.Text = (string)DTMem.Rows[0].ItemArray[16 + i];

txtFine.Text = DTMem.Rows[0].ItemArray[20 + i].ToString();

txtMemStatus.Text = (string)DTMem.Rows[0].ItemArray[9 + i];

memStatus = DTMem.Rows[0].ItemArray[9 + i].ToString();

}

void ClearGroupBoxContent(GroupBox grpBx)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

foreach (Control ctrl in grpBx.Controls)

{

if (ctrl isTextBox) { (ctrl asTextBox).Clear(); }

if (ctrl isDateTimePicker) { (ctrl asDateTimePicker).Value = DateTime.Today; }

}

}

privatevoid btnItem\_Click(object sender, EventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

try

{

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

c.cmd.CommandText = "select count(\*) from StockTable where accno ='" + txtAccno.Text + "' and copyno ='" + txtCopy.Text + "'";

if ((int)c.cmd.ExecuteScalar() < 1)

{

lblError.Text = "Record not found!";

return;

}

else lblError.Text = "...";

AccessionRegister\_04 objAccReg = newAccessionRegister\_04();

objAccReg.sentByCirculation = true;

objAccReg.adminInstance = adminInstance;

objAccReg.circAccno = txtAccno.Text;

objAccReg.circCopyNo = txtCopy.Text;

objAccReg.Show();

}

catch (Exception ex)

{

adminInstance.errObj.txtException.Text = ex.ToString();

adminInstance.errObj.Show();

this.Close();

}

}

privatevoid btnMemberDetails\_Click(object sender, EventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

try

{

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

c.cmd.CommandText = "select count(\*) from MembersTable where memberid ='" + txtMemID.Text + "'";

if ((int)c.cmd.ExecuteScalar() < 1)

{

lblError.Text = "Record not found!";

return;

}

else lblError.Text = "...";

Members\_05 objMem = newMembers\_05();

objMem.sentByCirculation = true;

objMem.adminInstance = adminInstance;

objMem.memIDCirc = txtMemID.Text;

objMem.Show();

}

catch (Exception ex)

{

adminInstance.errObj.txtException.Text = ex.ToString();

adminInstance.errObj.Show();

this.Close();

}

}

privatevoid btnBookStock\_Click(object sender, EventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

Stock\_08 objStock = newStock\_08();

if (txtAccno.Text != "")

{

objStock.accno = txtAccno.Text;

}

objStock.Show();

}

privatevoid btnView\_Click(object sender, EventArgs e)

{

//No Issue State check

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

c.cmd.CommandText = "Select count(\*) from IssueTable";

if ((int)c.cmd.ExecuteScalar() <= 0)

{

MessageBox.Show("No Book-Issues found!", "View Issued", MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

grpNavigator.Enabled = true;

btnIssue.Enabled = false;

btnRenew.Enabled = false;

btnClear\_Click(this, null);

txtAccno.ReadOnly = true;

txtCopy.ReadOnly = true;

txtMemID.ReadOnly = true;

try

{

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

c.cmd.CommandText = "select \* from IssueTable";

adp.SelectCommand = c.cmd;

DTIssue.Clear();

adp.Fill(DTIssue);

index = 0;

isViewing = true;

SetIssueValues(index);

}

catch (Exception ex)

{

adminInstance.errObj.txtException.Text = ex.ToString();

adminInstance.errObj.Show();

this.Close();

}

}

void SetIssueValues(int i)

{

try

{

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

issueIDBookMemInfo = "" + (i + 1);

txtIssueID.Text = "" + DTIssue.Rows[i].ItemArray[0];

IssueDatePicker.Value = (DateTime) DTIssue.Rows[i].ItemArray[4];

DueDatePicker.Value = (DateTime) DTIssue.Rows[i].ItemArray[5];

txtIssueFine.Text = "" + DTIssue.Rows[i].ItemArray[7];

txtIssueStatus.Text = "" + DTIssue.Rows[i].ItemArray[8];

if (DTIssue.Rows[i].ItemArray[6] == DBNull.Value)

{

ReturnDatePicker.Visible = false;

lblReturnDate.Visible = false;

}

else

{

ReturnDatePicker.Value = (DateTime)DTIssue.Rows[i].ItemArray[6];

ReturnDatePicker.Visible = true;

lblReturnDate.Visible = true;

}

txtIssueFine.Text = "" + DTIssue.Rows[i].ItemArray[7];

txtIssueStatus.Text = "" + DTIssue.Rows[i].ItemArray[8];

if (DTIssue.Rows[i].ItemArray[9] == DBNull.Value)

{

renewDatePicker.Visible = false;

lblRenewDate.Visible = false;

}

else

{

renewDatePicker.Value = (DateTime)DTIssue.Rows[i].ItemArray[9];

renewDatePicker.Visible = true;

lblRenewDate.Visible = true;

}

txtStaffIDIssue.Text = "" + DTIssue.Rows[i].ItemArray[10];

txtStaffIDRenew.Text = "" + DTIssue.Rows[i].ItemArray[11];

txtStaffIDReturn.Text = "" + DTIssue.Rows[i].ItemArray[12];

//Setup Buttons

if ((string)DTIssue.Rows[i].ItemArray[8] == "Issued")

{

btnReturn.Enabled = true;

btnRenew.Enabled = true;

}

else

{

btnReturn.Enabled = false;

btnRenew.Enabled = false;

}

txtMemID.Text = "" + DTIssue.Rows[i].ItemArray[1];

txtAccno.Text = "" + DTIssue.Rows[i].ItemArray[2];

txtCopy.Text = "" + DTIssue.Rows[i].ItemArray[3];

btnBookInfo\_Click(this, null);

btnMemberInfo\_Click(this, null);

}

catch (Exception e)

{

adminInstance.errObj.txtException.Text = e.ToString();

adminInstance.errObj.Show();

this.Close();

}

}

privatevoid btnFirst\_Click(object sender, EventArgs e)

{

index = 0;

SetIssueValues(index);

}

privatevoid btnPrev\_Click(object sender, EventArgs e)

{

index--;

if (index < 0) index = 0;

SetIssueValues(index);

}

privatevoid btnNext\_Click(object sender, EventArgs e)

{

index++;

if (index > DTIssue.Rows.Count - 1) index = DTIssue.Rows.Count - 1;

SetIssueValues(index);

}

privatevoid btnLast\_Click(object sender, EventArgs e)

{

index = DTIssue.Rows.Count - 1;

SetIssueValues(index);

}

bool EmptyValidation()

{

foreach (Control ctrl inthis.Controls)

{

if (ctrl isTextBox) { if ((ctrl asTextBox).Text == "") returntrue; }

}

returnfalse;

}

bool MemStatusValidations()

{

if (memStatus == "Inactive")

{

string msg1, msg2, msg3;

msg1 = "Member's Status is set to Inactive!";

msg2 = "\nThe Member's Subscription has ended! ";

msg3 = "\n\nPlease setup the Member's Subscription to proceed";

MessageBox.Show(msg1 + msg2 + msg3, "Member Inactive", MessageBoxButtons.OK, MessageBoxIcon.Stop);

returntrue;

}

if (memStatus == "With Issue")

{

string msg1, msg2, msg3;

msg1 = "Member's Status is set to \"With Issue\"";

msg2 = "\nThis member has already issued a Book";

msg3 = "\n\nThis member can Re-issue only after returning the Book";

MessageBox.Show(msg1 + msg2 + msg3, "Member Status : With Issue", MessageBoxButtons.OK, MessageBoxIcon.Stop);

returntrue;

}

returnfalse;

}

privatevoid btnReturn\_Click(object sender, EventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

string issueID;

if (isViewing)

{

issueID = txtIssueID.Text;

grpNavigator.Enabled = false;

btnReturn.Enabled = false;

btnRenew.Enabled = false;

}

else

{

try

{

issueID = Interaction.InputBox("Enter the Issue ID:", "Book Return", "", -1, -1);

c.cmd.CommandText = "select count(\*) from IssueTable where issueid ='" + issueID + "'";

if ((int)c.cmd.ExecuteScalar() <= 0)

{

MessageBox.Show("Issue with Issue ID:" + issueID + " not found!",

"Book Return", MessageBoxButtons.OK, MessageBoxIcon.Stop);

return;

}

}

catch (SqlException)

{

MessageBox.Show("Incorrect Issue ID!",

"Book Renew", MessageBoxButtons.OK, MessageBoxIcon.Stop);

return;

}

}

c.cmd.CommandText = "select \* from IssueTable where issueid ='" + issueID + "'";

adp.SelectCommand = c.cmd;

DTIssue.Clear();

adp.Fill(DTIssue);

issueIDBookMemInfo = issueID;

txtMemID.Text = "" + DTIssue.Rows[0].ItemArray[1];

txtAccno.Text = "" + DTIssue.Rows[0].ItemArray[2];

txtCopy.Text = "" + DTIssue.Rows[0].ItemArray[3];

isViewing = true;

btnBookInfo\_Click(this, null);

btnMemberInfo\_Click(this, null);

UpdateGrpIssue(Convert.ToInt16(issueIDBookMemInfo));

//I don't think this is really required

txtMemID.Text = "" + DTIssue.Rows[0].ItemArray[1];

txtAccno.Text = "" + DTIssue.Rows[0].ItemArray[2];

txtCopy.Text = "" + DTIssue.Rows[0].ItemArray[3];

if (DTIssue.Rows[0].ItemArray[8].ToString() != "Issued")

{

MessageBox.Show("The issue status is " + DTIssue.Rows[0].ItemArray[8]

+ "\n\n To Return a Book, the Issue Status must be \"Issued\"",

"Book Return", MessageBoxButtons.OK, MessageBoxIcon.Stop);

btnIssue.Enabled = false;

btnRenew.Enabled = false;

btnReturn.Enabled = false;

return;

}

DateTime dueDate = (DateTime) DTIssue.Rows[0].ItemArray[5];

if (DateTime.Today > dueDate) { CalculateFine(DTIssue, dueDate); }

c.cmd.CommandText = "update IssueTable set status=@status, returndate=@returndate, fine=@fine, staffIDReturn=@staffIDReturn where issueid ='" + DTIssue.Rows[0].ItemArray[0] + "'";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@status", SqlDbType.VarChar).Value = "Returned";

c.cmd.Parameters.Add("@returndate", SqlDbType.DateTime).Value = DateTime.Today;

if (txtIssueFine.Text == "") txtIssueFine.Text = "0";

c.cmd.Parameters.Add("@fine", SqlDbType.Int).Value = Convert.ToInt32(txtIssueFine.Text);

c.cmd.Parameters.Add("@staffIDReturn", SqlDbType.VarChar).Value = adminInstance.staffID;

c.cmd.ExecuteNonQuery();

//Members Update

c.cmd.CommandText = "select expdate from MembersTable where memberid ='" + DTIssue.Rows[0].ItemArray[1] + "'";

adp.SelectCommand = c.cmd;

DTMem.Reset();

adp.Fill(DTMem);

DateTime expdate = (DateTime) DTMem.Rows[0].ItemArray[0];

MemberStatusUpdate(expdate);

AccRegStatusUpdate();

MessageBox.Show("Book Returned \nIssue ID:" + issueID, "Return Book", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

UpdateGrpIssue(Convert.ToInt16(issueID));

txtMemID.Text = "" + DTIssue.Rows[0].ItemArray[1];

txtAccno.Text = "" + DTIssue.Rows[0].ItemArray[2];

txtCopy.Text = "" + DTIssue.Rows[0].ItemArray[3];

btnBookInfo\_Click(this, null);

btnMemberInfo\_Click(this, null);

}

void MemberStatusUpdate(DateTime expdate)

{

string status = (DateTime.Today > expdate) ? "Inactive" : "Active";

c.cmd.CommandText = "update MembersTable set status=@status where memberid ='" + DTIssue.Rows[0].ItemArray[1] + "'";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@status", SqlDbType.VarChar).Value = status;

c.cmd.ExecuteNonQuery();

c.cmd.CommandText = "update MembersIssues set status=@status where issueid ='" + DTIssue.Rows[0].ItemArray[0] + "'";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@status", SqlDbType.VarChar).Value = status;

c.cmd.ExecuteNonQuery();

}

void AccRegStatusUpdate()

{

c.cmd.CommandText = "update StockTable set status=@status where accno ='" + DTIssue.Rows[0].ItemArray[2] + "' and copyno ='" + DTIssue.Rows[0].ItemArray[3] + "'";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@status", SqlDbType.VarChar).Value = "Available";

c.cmd.ExecuteNonQuery();

c.cmd.CommandText = "update StockIssues set status=@status where issueid ='" + DTIssue.Rows[0].ItemArray[0] + "'";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@status", SqlDbType.VarChar).Value = "Available";

c.cmd.ExecuteNonQuery();

}

privatevoid btnRenew\_Click(object sender, EventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

try

{

checkBoxOvernightCirc.Checked = false;

string issueID;

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

if (isViewing)

{

issueID = txtIssueID.Text;

grpNavigator.Enabled = false;

btnReturn.Enabled = false;

btnRenew.Enabled = false;

}

else

{

try

{

issueID = Interaction.InputBox("Enter the Issue ID:", "Book Return", "", -1, -1);

c.cmd.CommandText = "select count(\*) from IssueTable where issueid ='" + issueID + "'";

if ((int)c.cmd.ExecuteScalar() <= 0)

{

MessageBox.Show("Issue with Issue ID:" + issueID + " not found!",

"Book Renew", MessageBoxButtons.OK, MessageBoxIcon.Stop);

return;

}

}

catch (SqlException)

{

MessageBox.Show("Incorrect Issue ID!",

"Book Renew", MessageBoxButtons.OK, MessageBoxIcon.Stop);

return;

}

}

c.cmd.CommandText = "select \* from IssueTable where issueid ='" + issueID + "'";

adp.SelectCommand = c.cmd;

DTIssue.Clear();

adp.Fill(DTIssue);

issueIDBookMemInfo = issueID;

txtMemID.Text = "" + DTIssue.Rows[0].ItemArray[1];

txtAccno.Text = "" + DTIssue.Rows[0].ItemArray[2];

txtCopy.Text = "" + DTIssue.Rows[0].ItemArray[3];

isViewing = true;

btnBookInfo\_Click(this, null);

btnMemberInfo\_Click(this, null);

UpdateGrpIssue(Convert.ToInt16(issueIDBookMemInfo));

//Do I really need this?

txtMemID.Text = "" + DTIssue.Rows[0].ItemArray[1];

txtAccno.Text = "" + DTIssue.Rows[0].ItemArray[2];

txtCopy.Text = "" + DTIssue.Rows[0].ItemArray[3];

if (DTIssue.Rows[0].ItemArray[8].ToString() != "Issued")

{

MessageBox.Show("The issue status is " + DTIssue.Rows[0].ItemArray[8]

+ "\n\n To Renew a Book, the Issue Status must be \"Issued\"",

"Book Renew", MessageBoxButtons.OK, MessageBoxIcon.Stop);

btnIssue.Enabled = false;

btnRenew.Enabled = false;

btnReturn.Enabled = false;

return;

}

DateTime issueDate = (DateTime)DTIssue.Rows[0].ItemArray[4];

int numOfDays = SetDueDays(txtMemID.Text, issueDate);

DateTime dueDateO = issueDate.AddDays(numOfDays);

DateTime dueDateNow = (DateTime)DTIssue.Rows[0].ItemArray[5];

if (issueDate == DateTime.Today)

{

MessageBox.Show("The Book was issued today!");

btnRenew.Enabled = true;

btnReturn.Enabled = true;

isViewing = false;

return;

}

if (dueDateNow == issueDate.AddDays(1))

{

MessageBox.Show("Cannot Renew a Overnight Circulation!");

btnRenew.Enabled = true;

btnReturn.Enabled = true;

isViewing = false;

return;

}

if (DateTime.Today > dueDateNow)

{

MessageBox.Show("Cannot Renew! \nThe Due-date has passed");

btnRenew.Enabled = true;

btnReturn.Enabled = true;

isViewing = false;

return;

}

if (dueDateNow > dueDateO)

{

MessageBox.Show("The Book has already been renewed once,"

+ "\nPress Return to return the book");

btnRenew.Enabled = true;

btnReturn.Enabled = true;

isViewing = false;

return;

}

else

{

numOfDays = SetDueDays(txtMemID.Text, DateTime.Today);

c.cmd.Parameters.Clear();

c.cmd.CommandText = "update IssueTable set duedate=@duedate, renewdate=@renewdate, staffIDRenew=@staffIDRenew where issueid ='" + issueID + "'";

c.cmd.Parameters.Add("@duedate", SqlDbType.DateTime).Value = DateTime.Today.AddDays(numOfDays);

c.cmd.Parameters.Add("@renewdate", SqlDbType.DateTime).Value = DateTime.Today;

c.cmd.Parameters.Add("@staffIDRenew", SqlDbType.VarChar).Value = adminInstance.staffID;

c.cmd.ExecuteNonQuery();

MessageBox.Show("Book Renewed! \nIssueId :" + txtIssueID.Text);

}

UpdateGrpIssue(Convert.ToInt16(issueID));

btnIssue.Enabled = true;

btnRenew.Enabled = true;

btnReturn.Enabled = true;

}

catch (Exception ex)

{

adminInstance.errObj.txtException.Text = ex.ToString();

adminInstance.errObj.Show();

this.Close();

}

}

int SetDueDays(string memID, DateTime today)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

Connect x = newConnect();

DateTime tomorrow = DateTime.Today.AddDays(1);

DataTable memTable = newDataTable();

DataTable memParaTable = newDataTable();

if (checkBoxOvernightCirc.Checked)

{

x.cmd.CommandText = "select count(\*) from HolidayTable where date='" + tomorrow + "'";

if ((int)x.cmd.ExecuteScalar() <= 0)

{

return 1;

}

else

{

MessageBox.Show("Cannot set Due-date to " + tomorrow.ToShortDateString()

+ "\nAs Tomorrow is a Holiday!"

+ "\nAborting!", "Overnight-Circulation",

MessageBoxButtons.OK, MessageBoxIcon.Stop);

return -1;

}

}

//Bring values from Member parameters

x.cmd.CommandText = "select membertype from MembersTable where memberid ='" + memID + "'";

string memType = (string)x.cmd.ExecuteScalar();

x.cmd.CommandText = "select maxissuedays from MemberParameters where memtype ='" + memType +"'";

int numOfDays = (int) x.cmd.ExecuteScalar();

DateTime dueDate = DateTime.Today.AddDays(numOfDays);

bool isHoliday = true;

do

{

x.cmd.CommandText = "select count(\*) from HolidayTable where date='" + dueDate + "'";

if ((int)x.cmd.ExecuteScalar() <= 0)

{

isHoliday = false;

}

else

{

numOfDays++;

dueDate = DateTime.Today.AddDays(numOfDays);

}

}

while (isHoliday);

return numOfDays;

}

void CalculateFine(DataTable issueTable, DateTime dueDate)

{

string memID, accno, copyno, memType;

DataTable memTable = newDataTable();

DataTable memParaTable = newDataTable();

memID = "" + issueTable.Rows[0].ItemArray[1];

accno = "" + issueTable.Rows[0].ItemArray[2];

copyno = "" + issueTable.Rows[0].ItemArray[3];

c.cmd.CommandText = "select membertype, fine from MembersTable where memberid ='" + memID + "'";

adp.SelectCommand = c.cmd;

memTable.Clear();

adp.Fill(memTable);

memType = (string)memTable.Rows[0].ItemArray[0];

int currentFine = (int)memTable.Rows[0].ItemArray[1];

c.cmd.CommandText = "select fineperday from MemberParameters where memtype ='" + memType + "'";

adp.SelectCommand = c.cmd;

memParaTable.Clear();

adp.Fill(memParaTable);

int finePD = Convert.ToInt16(memParaTable.Rows[0].ItemArray[0]);

int days = (int)(DateTime.Today - dueDate).TotalDays;

txtIssueFine.Text =Convert.ToString(days \* finePD);

int fine = currentFine + (days \* finePD);

c.cmd.CommandText = "update MembersTable set fine=@fine where memberid ='" + memID + "'";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@fine", SqlDbType.Int).Value = fine;

c.cmd.ExecuteNonQuery();

c.cmd.CommandText = "update MembersIssues set fine=@fine where issueid ='" + txtIssueID.Text + "'";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@fine", SqlDbType.Int).Value = fine;

c.cmd.ExecuteNonQuery();

MessageBox.Show("Fine has been added! \nMember ID:" + memID,

"Return Book", MessageBoxButtons.OK,

MessageBoxIcon.Information);

}

privatevoid btnBookLost\_Click(object sender, EventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

try

{

string memID;

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

try

{

memID = Interaction.InputBox("Enter the Member ID:", "Book Lost", "", -1, -1);

c.cmd.CommandText = "select count(\*) from IssueTable where memberid ='" + memID + "'";

if ((int)c.cmd.ExecuteScalar() <= 0)

{

MessageBox.Show("Member with Member ID:" + memID + " not found!",

"Book Lost", MessageBoxButtons.OK, MessageBoxIcon.Stop);

return;

}

ViewIssued\_11 objBookLost = newViewIssued\_11();

objBookLost.txtMemID.Text = memID;

objBookLost.Show();

}

catch (Exception)

{

MessageBox.Show("Incorrect Member ID!",

"Book Lost", MessageBoxButtons.OK, MessageBoxIcon.Stop);

return;

}

}

catch (Exception ex) { throw; }

}

privatevoid btnViewIssuedGrid\_Click(object sender, EventArgs e)

{

adminInstance.errObj.txtBtnClicks.Text += "\n" + MethodBase.GetCurrentMethod().Name;

ViewIssued\_11 objBL = newViewIssued\_11();

objBL.state = "View Issued";

objBL.Show();

}

privatevoid btnViewIssuedDetails\_Click(object sender, EventArgs e)

{

string infoText = "View Issued : The Information stored is only accurate to the time of Issue/Renew/Return."

+ "For the latest Updated details please check the Members-Management/Accession-Register.";

MessageBox.Show(infoText, "View Issued",

MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

}

06b Fine:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using Microsoft.VisualBasic;

namespace Library\_Auto

{

publicpartialclassFine\_06b : Form

{

Connect c = newConnect();

DataTable dt = newDataTable();

SqlDataAdapter adp = newSqlDataAdapter();

publicAdministration\_03 adminInstance;

public Fine\_06b()

{

InitializeComponent();

}

protectedoverridevoid OnFormClosing(FormClosingEventArgs e)

{

base.OnFormClosing(e);

switch (e.CloseReason)

{

caseCloseReason.TaskManagerClosing:

caseCloseReason.WindowsShutDown:

return;

default: break;

}

if (adminInstance != null) { adminInstance.Show(); }

}

privatevoid btnAll\_Click(object sender, EventArgs e)

{

btnPay.Enabled = false;

dt.Clear();

c.cmd.CommandText = "select membertype, memberid, firstname, lastname, course, fine, phoneno, email, status"

+ " from MembersTable where fine > 0";

adp.SelectCommand = c.cmd;

adp.Fill(dt);

dataGridView1.DataSource = dt;

}

privatevoid btnFind\_Click(object sender, EventArgs e)

{

dt.Clear();

c.cmd.CommandText = "select count(\*) from MembersTable where memberid = '" + txtMemID.Text + "' and fine > 0";

if ((int)c.cmd.ExecuteScalar() > 0)

{

btnPay.Enabled = true;

}

else btnPay.Enabled = false;

c.cmd.CommandText = "select membertype, memberid, firstname, lastname, course, fine, phoneno, email, status"

+ " from MembersTable where memberid = '" + txtMemID.Text + "'";

adp.SelectCommand = c.cmd;

adp.Fill(dt);

dataGridView1.DataSource = dt;

}

privatevoid btnPay\_Click(object sender, EventArgs e)

{

try

{

c.cmd.CommandText = "select fine from MembersTable where memberid ='" + txtMemID.Text + "'";

int fine = (int)c.cmd.ExecuteScalar();

string inputBoxValue = Interaction.InputBox("Current Fine:" + fine

+ "\nEnter the Fine Amount to deduct.", "Pay Fine", "", -1, -1);

int fineToDeduct = Convert.ToInt32(inputBoxValue);

if (fineToDeduct > fine || fineToDeduct <= 0)

{

DialogResult diaRes = MessageBox.Show("Incorrect Deduction Amount!"

+ "\nDo you want to Set the Fine to 0 ?", "Pay Fine", MessageBoxButtons.OKCancel, MessageBoxIcon.Information);

if (diaRes == DialogResult.Cancel)

{

return;

}

else fineToDeduct = fine;

}

c.cmd.CommandText = "update MembersTable set fine=@fine where memberid ='" + txtMemID.Text + "'";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@fine", SqlDbType.Int).Value = (fine - fineToDeduct);

c.cmd.ExecuteNonQuery();

c.cmd.CommandText = "select membertype, memberid, firstname, lastname, course, fine, phoneno, email, status"

+ " from MembersTable where memberid = '" + txtMemID.Text + "'";

dt.Clear();

adp.SelectCommand = c.cmd;

adp.Fill(dt);

dataGridView1.DataSource = dt;

dataGridView1.Refresh();

}

catch (FormatException)

{

MessageBox.Show("Incorrect Deduction Amount!");

return;

}

catch (Exception ex)

{

if (adminInstance != null)

{

adminInstance.errObj.txtException.Text = ex.ToString();

adminInstance.errObj.Show();

}

}

}

privatevoid SetMemID\_Click(object sender, DataGridViewCellEventArgs e)

{

try

{

if (dataGridView1.SelectedCells[0].Value == DBNull.Value)

{

return;

}

txtMemID.Text = (string)dt.Rows[dataGridView1.SelectedCells[0].RowIndex].ItemArray[1];

}

catch (FormatException) { }

}

}

}

06c Holidays Setup:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using Microsoft.VisualBasic;

namespace Library\_Auto

{

publicpartialclassHolidaySetup\_06c : Form

{

publicAdministration\_03 adminInstance;

Connect c = newConnect();

SqlDataAdapter adp = newSqlDataAdapter();

DataTable dt = newDataTable();

public HolidaySetup\_06c()

{

InitializeComponent();

}

protectedoverridevoid OnFormClosing(FormClosingEventArgs e)

{

base.OnFormClosing(e);

switch (e.CloseReason)

{

caseCloseReason.TaskManagerClosing:

caseCloseReason.WindowsShutDown:

return;

default: break;

}

if (adminInstance != null) { adminInstance.Show(); }

}

privatevoid btnGotoYear\_Click(object sender, EventArgs e)

{

try

{

if (txtYear.Text != "")

{

int year = Convert.ToInt16(txtYear.Text);

if (year > calHoliday.MaxDate.Year) year = calHoliday.MaxDate.Year;

if (year < calHoliday.MinDate.Year) year = calHoliday.MinDate.Year;

DateTime dtYear = newDateTime(year, 01, 01);

calHoliday.SetDate(dtYear);

}

}

catch (FormatException) { txtYear.Focus(); }

}

privatevoid btnSetHoliday\_Click(object sender, EventArgs e)

{

if (c.cnn.State != ConnectionState.Open)

{

c.cnn.Close();

c.cnn.Open();

}

DateTime selectedDate = calHoliday.SelectionStart;

c.cmd.CommandText = "select count(\*) from HolidayTable where date='" + selectedDate.ToShortDateString() + "'";

if ((int)c.cmd.ExecuteScalar() == 0)

{

c.cmd.CommandText = "insert into HolidayTable values(@date, @reason)";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@date", SqlDbType.DateTime).Value = selectedDate.ToShortDateString();

c.cmd.Parameters.Add("@reason", SqlDbType.VarChar).Value = txtReason.Text;

c.cmd.ExecuteNonQuery();

RefreshGrid();

Interaction.Beep();

}

}

void RefreshGrid()

{

c.cmd.CommandText = "select \* from HolidayTable";

adp.SelectCommand = c.cmd;

dt.Clear();

adp.Fill(dt);

DataView dView = newDataView(dt);

dView.Sort = "date ASC";

dgHoliday.DataSource = dView;

}

privatevoid btnDelete\_Click(object sender, EventArgs e)

{

c.cmd.CommandText = "delete from HolidayTable where date=@date";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@date", SqlDbType.DateTime).Value = calHoliday.SelectionStart.ToShortDateString();

c.cmd.ExecuteNonQuery();

RefreshGrid();

}

privatevoid GridToCal\_Click(object sender, DataGridViewCellEventArgs e)

{

try

{

if (dgHoliday.SelectedCells[0].Value == DBNull.Value) return;

DateTime dtGrid = Convert.ToDateTime(dgHoliday.SelectedCells[0].Value);

if (dtGrid != null) { calHoliday.SetDate(dtGrid); }

}

catch (FormatException) { txtReason.Text = dgHoliday.SelectedCells[0].Value.ToString(); }

}

privatevoid btnInsertSundays\_Click(object sender, EventArgs e)

{

try

{

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

DateTime thisYear = newDateTime(DateTime.Now.Year, 01, 01);

DateTime nextYearPlusTwo = thisYear.AddYears(1);

while (thisYear < nextYearPlusTwo)

{

if (thisYear.DayOfWeek == DayOfWeek.Sunday)

{

c.cmd.CommandText = "select count(\*) from HolidayTable where date='" + thisYear + "'";

if ((int)c.cmd.ExecuteScalar() <= 0)

{

c.cmd.CommandText = "insert into HolidayTable values (@date, @reason)";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@date", SqlDbType.DateTime).Value = thisYear;

c.cmd.Parameters.Add("@reason", SqlDbType.VarChar).Value = "Sunday";

c.cmd.ExecuteNonQuery();

}

}

thisYear = thisYear.AddDays(1);

}

RefreshGrid();

}

catch (Exception ex)

{

adminInstance.errObj.txtException.Text = ex.ToString();

adminInstance.errObj.Show();

this.Close();

}

}

privatevoid btnDelPrev\_Click(object sender, EventArgs e)

{

DateTime prevYear = newDateTime(DateTime.Now.Year - 1, 06, 01);

DialogResult diaRes = MessageBox.Show("Current Year :" + DateTime.Now.Year

+ "\nThis will delete all the previous years results."

+ "\nThe Year " + (DateTime.Now.Year - 1) + " will have upto 6 months of details.",

"Delete Previous Years", MessageBoxButtons.OKCancel, MessageBoxIcon.Information);

if (diaRes == DialogResult.Cancel) return;

c.cmd.CommandText = "delete from HolidayTable where date <= @date";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@date", SqlDbType.DateTime).Value = prevYear;

c.cmd.ExecuteNonQuery();

RefreshGrid();

}

}

}

07 Staff:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using Microsoft.VisualBasic;

using System.Text.RegularExpressions;

namespace Library\_Auto

{

publicpartialclassStaff\_07 : Form

{

//Variable Declarations

publicAdministration\_03 adminInstance;

Connect c = newConnect();

SqlDataAdapter adp = newSqlDataAdapter();

BindingSource bds = newBindingSource();

DataTable dt = newDataTable();

string state;

public Staff\_07()

{

InitializeComponent();

panelNavigation.Enabled = false;

grpStaffdetails.Enabled = false;

}

publicvoid StaffButtonActions(string action)

{

switch (action)

{

case"View": btnView\_Click(this, EventArgs.Empty); break;

case"Add": btnAdd\_Click(this, EventArgs.Empty); break;

case"Edit": btnEdit\_Click(this, EventArgs.Empty); break;

case"Delete": btnDelete\_Click(this, EventArgs.Empty); break;

default: return;

}

}

protectedoverridevoid OnFormClosing(FormClosingEventArgs e)

{

base.OnFormClosing(e);

switch (e.CloseReason)

{

caseCloseReason.TaskManagerClosing:

caseCloseReason.WindowsShutDown:

return;

default: break;

}

if (adminInstance != null) adminInstance.Show();

}

privatevoid btnClear\_Click(object sender, EventArgs e)

{

EnableAllMainButtons();

txtID.Clear();

txtID.ReadOnly = false;

ddlUserType.Text = "";

txtFName.Clear();

txtLName.Clear();

radioFMale.Checked= false;

radioMale.Checked= false;

dtDOB.Value = DateTime.Today;

txtAdd1 .Clear();

txtAdd2.Clear();

txtPincode.Clear();

txtState.Clear();

txtEmail.Clear();

txtPhNo.Clear();

dtDOJ.Value = DateTime.Today;

ddlStatus.Text = "";

grpStaffdetails.Enabled = false;

panelNavigation.Enabled = false;

lblError.Text = "...";

state = "";

}

privatevoid btnView\_Click(object sender, EventArgs e)

{

EnableAllMainButtons();

btnClear\_Click(this, EventArgs.Empty);

grpStaffdetails.Enabled = false;

panelNavigation.Enabled = true;

c.cmd.CommandText = "select \* from StaffTable";

adp.SelectCommand = c.cmd;

dt.Clear();

adp.Fill(dt);

bds.DataSource = dt;

ClearDataBindings();

AddDataBindings();

SetRadoGender();

state = "View";

}

privatevoid btnAdd\_Click(object sender, EventArgs e)

{

EnableAllMainButtons();

txtID.Focus();

panelNavigation.Enabled = false;

btnClear\_Click(this, EventArgs.Empty);

btnView.Enabled = false;

btnEdit.Enabled = false;

btnDelete.Enabled = false;

grpStaffdetails.Enabled = true;

state = "Add";

}

privatevoid btnEdit\_Click(object sender, EventArgs e)

{

EnableAllMainButtons();

txtID.ReadOnly = true;

panelNavigation.Enabled = false;

grpStaffdetails.Enabled = true;

if (state == "View")

{

txtFName.Focus();

state = "Edit";

return;

}

else

{

string staffID = Interaction.InputBox("Enter the Staff ID:", "Edit Staff Details", "", -1, -1);

if (adminInstance.staffID == staffID

|| (adminInstance.staffID asstring).ToUpper() == staffID)

{

panelNavigation.Enabled = false;

grpStaffdetails.Enabled = false;

btnClear\_Click(this, EventArgs.Empty);

MessageBox.Show("Cannot Edit self!");

return;

}

int counter;

c.cmd.CommandText = "select count(\*) from StaffTable where staffid='" + staffID + "'";

counter = (int)c.cmd.ExecuteScalar();

if (counter > 0)

{

c.cmd.CommandText = "select \* from StaffTable where staffid='" + staffID + "'";

adp.SelectCommand = c.cmd;

dt.Clear();

adp.Fill(dt);

bds.DataSource = dt;

ClearDataBindings();

AddDataBindings();

SetRadoGender();

}

else

{

panelNavigation.Enabled = false;

grpStaffdetails.Enabled = false;

MessageBox.Show("Staff Member not found!");

}

}

state = "Edit";

}

privatevoid btnDelete\_Click(object sender, EventArgs e)

{

EnableAllMainButtons();

panelNavigation.Enabled = false;

grpStaffdetails.Enabled = false;

if (state == "View")

{

DeleteRecords();

state = "Delete";

return;

}

else

{

string staffID = Interaction.InputBox("Enter the Staff ID:", "Delete Staff Details", "", -1, -1);

int counter;

if (adminInstance.staffID == staffID

|| (adminInstance.staffID asstring).ToUpper() == staffID)

{

panelNavigation.Enabled = false;

grpStaffdetails.Enabled = false;

btnClear\_Click(this, EventArgs.Empty);

MessageBox.Show("Cannot Delete self!");

return;

}

c.cmd.CommandText = "select count(\*) from StaffTable where staffid='" + staffID + "'";

counter = (int)c.cmd.ExecuteScalar();

if (counter > 0)

{

c.cmd.CommandText = "select \* from StaffTable where staffid='" + staffID + "'";

adp.SelectCommand = c.cmd;

dt.Clear();

adp.Fill(dt);

bds.DataSource = dt;

ClearDataBindings();

AddDataBindings();

DeleteRecords();

panelNavigation.Enabled = false;

}

else

{

panelNavigation.Enabled = false;

grpStaffdetails.Enabled = false;

MessageBox.Show("Invalid ID!");

}

}

state = "Delete";

}

void DeleteRecords()

{

DialogResult dialogRes = MessageBox.Show("Delete this record?", "Delete Staff", MessageBoxButtons.OKCancel);

if (dialogRes == DialogResult.OK)

{

c.cmd.CommandText = "delete from LoginTable where staffid='" + txtID.Text + "'";

c.cmd.ExecuteNonQuery();

c.cmd.CommandText = "delete from StaffTable where staffid='" + txtID.Text + "'";

c.cmd.ExecuteNonQuery();

MessageBox.Show("Record Deleted!");

}

elseif (dialogRes == DialogResult.Cancel) return;

}

//DataBindings.functions()

void ClearDataBindings()

{

txtID.DataBindings.Clear();

ddlUserType.DataBindings.Clear();

txtFName.DataBindings.Clear();

txtLName.DataBindings.Clear();

txtGender.DataBindings.Clear();

radioFMale.Checked = false;

radioMale.Checked = false;

dtDOB.DataBindings.Clear();

dtDOJ.DataBindings.Clear();

txtAdd1.DataBindings.Clear();

txtAdd2.DataBindings.Clear();

txtPincode.DataBindings.Clear();

txtState.DataBindings.Clear();

txtPhNo.DataBindings.Clear();

txtEmail.DataBindings.Clear();

ddlStatus.DataBindings.Clear();

}

void AddDataBindings()

{

txtID.DataBindings.Add("text", bds, "staffid");

ddlUserType.DataBindings.Add("text", bds, "stafftype");

txtFName.DataBindings.Add("text", bds, "fname");

txtLName.DataBindings.Add("text", bds, "lname");

txtGender.DataBindings.Add("text", bds, "gender");

dtDOB.DataBindings.Add("text", bds, "dob");

dtDOJ.DataBindings.Add("text", bds, "doj");

txtAdd1.DataBindings.Add("text", bds, "address1");

txtAdd2.DataBindings.Add("text", bds, "address2");

txtPincode.DataBindings.Add("text", bds, "pin");

txtState.DataBindings.Add("text", bds, "state");

txtEmail.DataBindings.Add("text", bds, "email");

txtPhNo.DataBindings.Add("text", bds, "phoneno");

ddlStatus.DataBindings.Add("text", bds, "status");

//Test the Dropdownlist to check if the databindings match!

}

void SetRadoGender()

{

radioFMale.Checked = (txtGender.Text.Equals("F")) ? true : false;

radioMale.Checked = (txtGender.Text.Equals("M")) ? true : false;

if (adminInstance.staffID == txtID.Text

|| (adminInstance.staffID asstring).ToUpper() == txtID.Text)

{

btnEdit.Enabled = false;

btnDelete.Enabled = false;

}

else

{

btnEdit.Enabled = true;

btnDelete.Enabled = true;

}

}

// \_DataBindings.functions().end

privatevoid btnFirst\_Click(object sender, EventArgs e) { bds.MoveFirst(); SetRadoGender(); }

privatevoid btnLast\_Click(object sender, EventArgs e){bds.MoveLast(); SetRadoGender(); }

privatevoid btnPrev\_Click(object sender, EventArgs e) { bds.MovePrevious(); SetRadoGender(); }

privatevoid btnNext\_Click(object sender, EventArgs e) { bds.MoveNext(); SetRadoGender();}

privatevoid btnSubmit\_Click(object sender, EventArgs e)

{

bool isInvalid;

isInvalid = InputValidations();

if (isInvalid){ return; }

switch (state)

{

case"Add":

c.cmd.CommandText = "select count(\*) from StaffTable where staffid='" + txtID.Text + "'";

int count = (int) c.cmd.ExecuteScalar();

if (count > 0)

{

MessageBox.Show("A Record with the same StaffID found!");

txtID.Focus();

return;

}

c.cmd.CommandText = "insert into StaffTable values(@staffid,@stafftype,@fname,@lname,@gender,@dob,@address1,@address2,@pin,@state,@email,@phoneno,@doj,@status)";

break;

case"Edit":

txtID.ReadOnly = false;

c.cmd.CommandText = "update StaffTable set stafftype=@stafftype, fname=@fname, lname=@lname, gender=@gender, dob=@dob, address1=@address1, address2=@address2, pin=@pin, state=@state, email=@email, phoneno=@phoneno, doj=@doj, status=@status where staffid=@staffid";

break;

default: return;

}

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@staffid", SqlDbType.VarChar).Value = txtID.Text;

c.cmd.Parameters.Add("@stafftype", SqlDbType.VarChar).Value = ddlUserType.Text;

c.cmd.Parameters.Add("@fname", SqlDbType.VarChar).Value = txtFName.Text;

c.cmd.Parameters.Add("@lname", SqlDbType.VarChar).Value = txtLName.Text;

c.cmd.Parameters.Add("@gender", SqlDbType.VarChar).Value = (radioFMale.Checked) ? "F" : "M";

c.cmd.Parameters.Add("@dob", SqlDbType.DateTime).Value = dtDOB.Value.ToString();

c.cmd.Parameters.Add("@address1", SqlDbType.VarChar).Value = txtAdd1.Text;

c.cmd.Parameters.Add("@address2", SqlDbType.VarChar).Value = txtAdd2.Text;

c.cmd.Parameters.Add("@pin", SqlDbType.VarChar).Value = txtPincode.Text;

c.cmd.Parameters.Add("@state", SqlDbType.VarChar).Value = txtState.Text;

c.cmd.Parameters.Add("@email", SqlDbType.VarChar).Value = txtEmail.Text;

c.cmd.Parameters.Add("@phoneno", SqlDbType.VarChar).Value = txtPhNo.Text;

c.cmd.Parameters.Add("@doj", SqlDbType.DateTime).Value = dtDOJ.Value;

c.cmd.Parameters.Add("@status", SqlDbType.VarChar).Value = ddlStatus.Text;

c.cmd.ExecuteNonQuery();

switch (state)

{

case"Add":

c.cmd.CommandText = "insert into LoginTable values(@staffid, @password, @usertype, @resetpassword, @sQuestion, @sAnswer)";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@staffid", SqlDbType.VarChar).Value = txtID.Text;

c.cmd.Parameters.Add("@password", SqlDbType.VarBinary).Value = DBNull.Value;

c.cmd.Parameters.Add("@usertype", SqlDbType.VarChar).Value = ddlUserType.Text;

c.cmd.Parameters.Add("@resetpassword", SqlDbType.VarChar).Value = "Reset\_Full";

c.cmd.Parameters.Add("@sQuestion", SqlDbType.VarBinary).Value = DBNull.Value;

c.cmd.Parameters.Add("@sAnswer", SqlDbType.VarBinary).Value = DBNull.Value;

c.cmd.ExecuteNonQuery();

break;

}

MessageBox.Show("Process Complete!");

btnView\_Click(this, EventArgs.Empty);

}

privatevoid btnCancel\_Click(object sender, EventArgs e)

{

switch (state)

{

case"Add":

panelNavigation.Enabled = true;

btnClear\_Click(this, EventArgs.Empty);

btnView.Enabled = true;

btnEdit.Enabled = true;

btnDelete.Enabled = true;

grpStaffdetails.Enabled = false;

break;

case"View":

btnClear\_Click(this, EventArgs.Empty);

panelNavigation.Enabled = false;

grpStaffdetails.Enabled = false;

break;

case"Edit":

btnClear\_Click(this, EventArgs.Empty);

break;

default: btnClear\_Click(this, EventArgs.Empty); break;

}

}

void EnableAllMainButtons()

{

btnAdd.Enabled = true;

btnEdit.Enabled = true;

btnView.Enabled = true;

btnDelete.Enabled = true;

btnPrev.Enabled = true;

btnNext.Enabled = true;

btnFirst.Enabled = true;

btnLast.Enabled = true;

btnClear.Enabled = true;

}

privatevoid NumValidations(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar)

&& e.KeyChar != 0)

{

lblError.Text = "Enter only numbers!";

e.Handled = true;

return;

}

ResetLblValidation();

}

privatevoid AlphaValidations(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsLetter(e.KeyChar))

{

lblError.Text = "Enter only Letters!";

e.Handled = true;

return;

}

ResetLblValidation();

}

privatevoid AlphaNumValidations(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsLetterOrDigit(e.KeyChar)

&& !char.IsWhiteSpace(e.KeyChar)

&& !char.Equals(e.KeyChar, '-'))

{

lblError.Text = "Enter only Letters or Digits";

e.Handled = true;

return;

}

ResetLblValidation();

}

privatevoid ResetLblValidation()

{

lblError.Text = "...";

}

privatebool InputValidations()

{

bool isRadioChecked = false;

foreach (Control ctrl in grpStaffdetails.Controls)

{

if (ctrl isTextBox)

{

if ((ctrl asTextBox).Name != "txtGender"

&& (ctrl asTextBox).Text == "")

{

lblError.Text = "Empty TextBox detected!";

returntrue;

}

}

elseif (ctrl isComboBox)

{

if ((ctrl asComboBox).SelectedItem == null)

{

lblError.Text = "Unselected Item detected!";

returntrue;

}

}

elseif (ctrl isDateTimePicker)

{

DateTime checkDate = DateTime.Today.AddYears(-18);

if ((ctrl asDateTimePicker).Name == dtDOB.Name

&& (ctrl asDateTimePicker).Value > checkDate)

{

lblError.Text = "DOB < 18 years!";

returntrue;

}

if ((ctrl asDateTimePicker).Name == dtDOJ.Name

&& (ctrl asDateTimePicker).Value < dtDOB.Value)

{

lblError.Text = "DOJ < DOB!";

returntrue;

}

}

}

foreach (Control ctrl in grpStaffdetails.Controls)

{

if (ctrl isRadioButton)

{

if ((ctrl asRadioButton).Checked == true) { isRadioChecked = true; }

}

}

if (!isRadioChecked)

{

lblError.Text = "Select Gender!";

returntrue;

}

//Set the bool isInvalid to True if the Email is Invalid.

if (InvalidEmail()) returntrue;

ResetLblValidation();

returnfalse;

}

publicbool InvalidEmail()

{

string emailExpr = null;

emailExpr = "^([0-9a-zA-Z]([-\\.\\w]\*[0-9a-zA-Z])\*@([0-9a-zA-Z][-\\w]\*[0-9a-zA-Z]\\.)+[a-zA-Z]{2,9})$";

if (Regex.IsMatch(txtEmail.Text, emailExpr))

{

//Since the Email is Valid, retutn IsInvalidEmail as False.

returnfalse;

}

lblError.Text = "Invalid Email!";

returntrue;

}

}

}

08 Stock Lookup:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using Microsoft.VisualBasic;

namespace Library\_Auto

{

publicpartialclassStock\_08 : Form

{

publicAdministration\_03 adminInstance;

Connect c = newConnect();

DataTable dt = newDataTable();

SqlDataAdapter adp = newSqlDataAdapter();

publicstring accno;

public Stock\_08()

{

InitializeComponent();

}

privatevoid btnShow\_Click(object sender, EventArgs e)

{

string commandText = "select \* from StockTable where accno='" + txtAccno.Text + "'";

FillGrid(commandText);

}

privatevoid btnClear\_Click(object sender, EventArgs e)

{

dt.Clear();

}

privatevoid StockLookup\_04b\_Load(object sender, EventArgs e)

{

this.WindowState = FormWindowState.Maximized;

if (accno != null) txtAccno.Text = accno;

}

privatevoid btnShowAll\_Click(object sender, EventArgs e)

{

string commandText = "select \* from StockTable ORDER BY LEN(accno) ASC, accno ASC, copyno ASC";

FillGrid(commandText);

}

privatevoid btnDelCopy\_Click(object sender, EventArgs e)

{

c.cmd.CommandText = "select count(\*) from StockTable where accno ='" + txtAccno.Text + "'";

int count = (int) c.cmd.ExecuteScalar();

if (count > 0)

{

string copyno = Interaction.InputBox("Enter the CopyNo:", "Delete Copy", "", -1, -1);

string commandText = "select \* from StockTable where accno ='" + txtAccno.Text + "' and copyno ='" + copyno + "'";

FillGrid(commandText);

if (dt.Rows.Count > 0)

{

c.cmd.CommandText = "delete from StockTable where accno ='" + txtAccno.Text + "' and copyno ='" + copyno + "'";

c.cmd.ExecuteNonQuery();

MessageBox.Show("Record Deleted!");

}

elseMessageBox.Show("\nAccno :" + txtAccno.Text + "\nCopy Number :" + copyno + "\n\nNot found!", "Delete Copy");

}

else

{

MessageBox.Show("Not Found!");

Interaction.Beep();

txtAccno.Focus();

return;

}

}

void FillGrid(string \_commandText)

{

c.cmd.CommandText = \_commandText;

adp.SelectCommand = c.cmd;

dt.Clear();

adp.Fill(dt);

DataView dView = newDataView(dt);

dataGridView1.DataSource = dView;

lblCurrentStock.Text = "Current Stock (Count): " + dView.Count;

}

protectedoverridevoid OnFormClosing(FormClosingEventArgs e)

{

base.OnFormClosing(e);

switch (e.CloseReason)

{

caseCloseReason.WindowsShutDown:

caseCloseReason.TaskManagerClosing:

return;

default: break;

}

if (adminInstance != null) adminInstance.Show();

}

privatevoid AlphaNumValidations(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsLetterOrDigit(e.KeyChar)

&& !char.Equals(e.KeyChar, '-'))

{

stripError.Text = "Enter only Letters or Digits";

e.Handled = true;

return;

}

stripError.Text = "...";

}

}

}

10b Vendor :

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using Microsoft.VisualBasic;

namespace Library\_Auto

{

publicpartialclassVendor\_10b : Form

{

publicAdministration\_03 adminInstance;

Connect c = newConnect();

SqlDataAdapter adp = newSqlDataAdapter();

BindingSource bds = newBindingSource();

DataTable dt = newDataTable();

public Vendor\_10b()

{

InitializeComponent();

}

protectedoverridevoid OnFormClosing(FormClosingEventArgs e)

{

base.OnFormClosing(e);

switch (e.CloseReason)

{

caseCloseReason.WindowsShutDown:

caseCloseReason.TaskManagerClosing:

return;

default: break;

}

if (adminInstance != null) adminInstance.Show();

}

privatevoid Vendor\_10b\_Load(object sender, EventArgs e)

{

foreach (Control ctr in grpVendorDetail.Controls)

{

if (ctr isTextBox)

{

(ctr asTextBox).ReadOnly = false;

}

}

}

privatevoid btnView\_Click(object sender, EventArgs e)

{

foreach (Control ctr in grpVendorDetail.Controls)

{

if (ctr isTextBox)

{

(ctr asTextBox).ReadOnly = true;

}

}

dt.Clear();

try

{

if (c.cnn.State != ConnectionState.Open)

{

c.cnn.Close();

c.cnn.Open();

}

c.cmd.CommandText = "select \* from VendorTable";

adp.SelectCommand = c.cmd;

adp.Fill(dt);

bds.DataSource = dt;

ClearDataBindings();

AddDataBindings();

}

catch

{

throw;

}

finally

{

c.cnn.Close();

}

}

privatevoid AddDataBindings()

{

txtVenId.DataBindings.Add("text", bds, "vendorid");

txtVenName.DataBindings.Add("text", bds, "name");

txtAddress.DataBindings.Add("text", bds, "address");

txtPhno.DataBindings.Add("text", bds, "phoneno");

txtEmail.DataBindings.Add("text", bds, "email");

}

privatevoid ClearDataBindings()

{

foreach (Control ctr in grpVendorDetail.Controls)

{

if (ctr isTextBox)

{

(ctr asTextBox).DataBindings.Clear();

}

}

}

privatevoid btnAdd\_Click(object sender, EventArgs e)

{

grpVendorDetail.Enabled = true;

dt.Clear();

foreach (Control ctr in grpVendorDetail.Controls)

{

if (ctr isTextBox)

{

(ctr asTextBox).ReadOnly = false;

}

}

foreach (Control ctr in grpVendorDetail.Controls)

{

if (ctr isTextBox)

{

(ctr asTextBox).Clear();

}

}

}

privatevoid btnSubmit\_Click(object sender, EventArgs e)

{

try

{

if (c.cnn.State != ConnectionState.Open)

{

c.cnn.Close();

c.cnn.Open();

}

c.cmd.CommandText = "select count(\*) from VendorTable where vendorid ='" +txtVenId.Text + "'";

if ((int)c.cmd.ExecuteScalar() > 0)

{

//Interaction.Beep();

MessageBox.Show("Vendor with same Vendor ID found!", "Add Member");

txtVenId.Focus();

return;

}

c.cmd.CommandText = "insert into VendorTable values(@vendorid, @name, @address, @phoneno,@email)";

c.cmd.Parameters.Clear();

c.cmd.Parameters.Add("@vendorid", SqlDbType.VarChar).Value = txtVenId.Text;

c.cmd.Parameters.Add("@name", SqlDbType.VarChar).Value = txtVenName.Text;

c.cmd.Parameters.Add("@address", SqlDbType.VarChar).Value = txtAddress.Text;

c.cmd.Parameters.Add("@phoneno", SqlDbType.VarChar).Value = txtPhno.Text;

c.cmd.Parameters.Add("@email", SqlDbType.VarChar).Value = txtEmail.Text;

c.cmd.ExecuteNonQuery();

MessageBox.Show("vendor Added!");

foreach (Control ctr in grpVendorDetail.Controls)

{

if (ctr isTextBox)

{

(ctr asTextBox).ReadOnly = true;

}

}

}

catch (SqlException)

{

this.Close();

}

finally

{

c.cnn.Close();

}

}

privatevoid btnDelete\_Click(object sender, EventArgs e)

{

try

{

if (c.cnn.State != ConnectionState.Open)

{

c.cnn.Close();

c.cnn.Open();

}

if (txtVenId.Text == "")

{

string vendorid =Interaction.InputBox("Enter the Vendor ID:", "Delete using Vendor Id", "", -1, -1);

c.cmd.CommandText = "select count(\*) from VendorTable where vendorid ='" + vendorid + "'";

if ((int)c.cmd.ExecuteScalar() <= 0)

{

MessageBox.Show("Record not found!", "Delete", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else

{

c.cmd.CommandText = "delete from VendorTable where vendorid ='" + vendorid + "'";

c.cmd.ExecuteNonQuery();

MessageBox.Show("Record deleted!", "Delete Records", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

else

{

DialogResult dRes = MessageBox.Show("Delete this record? Member ID = " + txtVenId.Text, "Delete Records", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);

if (dRes == DialogResult.Cancel)

{

return;

}

elseif (dRes == DialogResult.OK)

{

c.cmd.CommandText = "delete from VendorTable where vendorid ='" + txtVenId.Text + "'";

c.cmd.ExecuteNonQuery();

MessageBox.Show("Record deleted!", "Delete Records", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

// UpdateStats();

}

}

}

catch (SqlException)

{

this.Close();

}

finally

{

c.cnn.Close();

}

}

privatevoid btnClear\_Click(object sender, EventArgs e)

{

grpVendorDetail.Enabled = false;

foreach (Control ctr in grpVendorDetail.Controls)

{

if (ctr isTextBox)

{

(ctr asTextBox).Clear();

}

}

ClearDataBindings();

}

privatevoid btnFirst\_Click(object sender, EventArgs e)

{

bds.MoveFirst();

}

privatevoid btnPrev\_Click(object sender, EventArgs e)

{

bds.MovePrevious();

}

privatevoid btnNext\_Click(object sender, EventArgs e)

{

bds.MoveNext();

}

privatevoid btnLast\_Click(object sender, EventArgs e)

{

bds.MoveLast();

}

}

}

10b Create Order:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using Microsoft.VisualBasic;

namespace Library\_Auto

{

publicpartialclassCreateOrder\_10c : Form

{

publicAdministration\_03 adminInstance;

Connect c = newConnect();

DataTable dt = newDataTable();

DataTable dt2 = newDataTable();

SqlDataAdapter adp = newSqlDataAdapter();

int orderNo;

string accno, vendorID;

bool clearSkip = false;

public CreateOrder\_10c()

{

InitializeComponent();

}

protectedoverridevoid OnFormClosing(FormClosingEventArgs e)

{

base.OnFormClosing(e);

switch (e.CloseReason)

{

caseCloseReason.WindowsShutDown:

caseCloseReason.TaskManagerClosing:

return;

default: break;

}

if (clearSkip) return;

if (adminInstance != null) adminInstance.Show();

}

privatevoid btnClear\_Click(object sender, EventArgs e)

{

c.cmd.CommandText = "delete from CreateOrderDummy";

c.cmd.ExecuteNonQuery();

clearSkip = true;

this.Close();

adminInstance.createOrderToolStripMenuItem1\_Click(this, EventArgs.Empty);

}

privatevoid btnNext\_Click(object sender, EventArgs e)

{

btnAdd.Enabled = true;

btnNext.Enabled = false;

grpBookInfo.Enabled = true;

foreach (Control ctr in grpVendorInfo.Controls)

{

if (ctr isTextBox)

{

(ctr asTextBox).ReadOnly = true;

}

}

foreach (Control ctr in grpBookInfo.Controls)

{

if (ctr isTextBox)

{

(ctr asTextBox).ReadOnly = false;

}

}

txtOrderNo.ReadOnly = true;

ddlAccno.SelectedItem = null;

foreach (Control ctr in grpBookInfo.Controls)

{

if (ctr isTextBox)

{

if ((ctr asTextBox).Name != txtOrderNo.Name)

(ctr asTextBox).Clear();

}

}

}

privatevoid DdlUpdate(string cmdText, ComboBox ddlToFill)

{

DataTable dtDummy = newDataTable();

string itemToAdd = "";

try

{

c.cmd.CommandText = cmdText;

adp.SelectCommand = c.cmd;

adp.Fill(dtDummy);

for (int i = 0; i < dtDummy.Rows.Count; i++)

{

itemToAdd = (string)dtDummy.Rows[i].ItemArray[0] ;

if (!ddlToFill.Items.Contains(itemToAdd))

{

ddlToFill.Items.Add(itemToAdd);

}

}

}

catch (Exception ex)

{

adminInstance.errObj.txtException.Text += ex.ToString();

adminInstance.errObj.Show();

this.Close();

}

}

privatevoid CreateOrder\_10c\_Load(object sender, EventArgs e)

{

foreach (Control ctr in grpVendorInfo.Controls)

{

if (ctr isTextBox)

{

(ctr asTextBox).ReadOnly = true;

}

}

ddlVenId.Enabled = false;

OrderDatePicker.Enabled = false;

ddlAccno.Enabled = false;

foreach (Control ctr in grpBookInfo.Controls)

{

if (ctr isTextBox)

{

(ctr asTextBox).ReadOnly = true;

}

}

btnNext.Enabled = false;

btnClear.Enabled = false;

DdlUpdate("select \* from VendorTable", ddlVenId);

DdlUpdate2("select \* from AccRegTable", ddlAccno);

}

privatevoid DdlUpdate2(string p, ComboBox ddlAccno)

{

DataTable dtDummy2 = newDataTable();

string ItemToAdd = "";

try

{

//if (!ddlToFill.Items.Contains("(None)"))

// ddlToFill.Items.Add("(None)");

c.cmd.CommandText = p;

adp.SelectCommand = c.cmd;

adp.Fill(dtDummy2);

for (int i = 0; i < dtDummy2.Rows.Count; i++)

{

ItemToAdd = (string)dtDummy2.Rows[i].ItemArray[0];

if (!ddlAccno.Items.Contains(ItemToAdd))

{

ddlAccno.Items.Add(ItemToAdd);

}

}

}

catch (Exception )

{

//adminInstance.errObj.txtException.Text += ex.ToString();

//adminInstance.errObj.Show();

//this.Close();

throw;

}

}

privatevoid UpdateVendorInfo\_SelectionChangeCommitted(object sender, EventArgs e)

{

DataTable dtVendor = newDataTable();

c.cmd.CommandText = "select name, address, phoneno, email from VendorTable where vendorid ='" + ddlVenId.SelectedItem + "'";

adp.SelectCommand = c.cmd;

adp.Fill(dtVendor);

txtName.Text = "" + dtVendor.Rows[0].ItemArray[0];

txtAddress.Text = "" + dtVendor.Rows[0].ItemArray[1];

txtPhno.Text = "" + dtVendor.Rows[0].ItemArray[2];

txtEmail.Text = "" + dtVendor.Rows[0].ItemArray[3];

}

privatevoid UpdateAccRegTableInfo\_SelectionChangeCommitted(object sender, EventArgs e)

{

txtTitle.ReadOnly = true;

txtAuthor.ReadOnly = true;

txtPublisher.ReadOnly = true;

if (ddlAccno.SelectedItem == null) return;

for (int i = 0; i < dt.Rows.Count; i++)

{

if (ddlAccno.SelectedItem.ToString() == (string) dt.Rows[i].ItemArray[2])

{

lblError.Text = "Error : Item already selected!";

ddlAccno.SelectedItem = null;

foreach (Control ctrl in grpBookInfo.Controls) { if (ctrl isTextBox) (ctrl asTextBox).Clear(); }

return;

}

}

lblError.Text = "...";

DataTable dtAccReg = newDataTable();

c.cmd.CommandText = "select title, author, publisher from AccRegTable where accno ='" + ddlAccno.SelectedItem + "'";

adp.SelectCommand = c.cmd;

adp.Fill(dtAccReg);

txtTitle.Text = "" + dtAccReg.Rows[0].ItemArray[0];

txtAuthor.Text = "" + dtAccReg.Rows[0].ItemArray[1];

txtPublisher.Text = "" + dtAccReg.Rows[0].ItemArray[2];

}

privatevoid btnNew\_Click(object sender, EventArgs e)

{

ddlVenId.Enabled = true;

ddlAccno.Enabled = true;

ddlVenId.SelectedItem = null;

ddlAccno.SelectedItem = null;

txtTitle.ReadOnly = false;

txtAuthor.ReadOnly = false;

txtPublisher.ReadOnly = false;

txtNoCopy.ReadOnly = false;

btnAdd.Enabled = true;

btnNext.Enabled = false;

btnClear.Enabled = true;

btnSubmit.Enabled = false;

grpBookInfo.Enabled = true;

//Empty CreateOrderDummy

c.cmd.CommandText = "delete from CreateOrderDummy";

c.cmd.ExecuteNonQuery();

dt.Clear();

foreach (Control ctrl in grpBookInfo.Controls) { if (ctrl isTextBox) (ctrl asTextBox).Clear(); }

foreach (Control ctrl in grpVendorInfo.Controls) { if (ctrl isTextBox) (ctrl asTextBox).Clear(); }

int orderNo;

c.cmd.CommandText = "select max(orderno) from CreateOrderTable";

if (c.cmd.ExecuteScalar() == DBNull.Value)

{

orderNo = 1;

}

else orderNo = (int)c.cmd.ExecuteScalar() + 1;

txtOrderNo.Text = "" + orderNo;

txtOrderNo.ReadOnly = true;

}

privatevoid btnAdd\_Click(object sender, EventArgs e)

{

if (ddlVenId.SelectedItem == null || ddlAccno.SelectedItem == null)

{

return;

}

if (txtNoCopy.Text == "")

{

lblError.Text = "Error : Enter the number of Copies!";

return;

}

btnAdd.Enabled = false;

btnNext.Enabled = true;

btnSubmit.Enabled = true;

c.cmd.Parameters.Clear();

c.cmd.CommandText = "insert into CreateOrderDummy values (@orderno,@orderdate,@accno,@vendorid,@name,@email,@address,@phoneno,@title,@author,@publisher,@noofcopies)";

c.cmd.Parameters.Add("@orderno", SqlDbType.Int).Value = Convert.ToInt32(txtOrderNo.Text);

c.cmd.Parameters.Add("@orderdate", SqlDbType.DateTime).Value = OrderDatePicker.Value.ToShortDateString();

c.cmd.Parameters.Add("@accno", SqlDbType.VarChar).Value = ddlAccno.Text;

c.cmd.Parameters.Add("@vendorid", SqlDbType.VarChar).Value = ddlVenId.Text;

c.cmd.Parameters.Add("@name", SqlDbType.VarChar).Value = txtName.Text;

c.cmd.Parameters.Add("@email", SqlDbType.VarChar).Value = txtEmail.Text;

c.cmd.Parameters.Add("@address", SqlDbType.VarChar).Value = txtAddress.Text;

c.cmd.Parameters.Add("@phoneno", SqlDbType.VarChar).Value = txtPhno.Text;

c.cmd.Parameters.Add("@title", SqlDbType.VarChar).Value = txtTitle.Text;

c.cmd.Parameters.Add("@author", SqlDbType.VarChar).Value = txtAuthor.Text;

c.cmd.Parameters.Add("@publisher", SqlDbType.VarChar).Value = txtPublisher.Text;

c.cmd.Parameters.Add("@noofcopies", SqlDbType.Int).Value = Convert.ToInt32(txtNoCopy.Text);

c.cmd.ExecuteNonQuery();

MessageBox.Show("Books Inserted");

ddlVenId.Enabled = false;

grpBookInfo.Enabled = false;

c.cmd.CommandText = "select \* from CreateOrderDummy where orderno='" + txtOrderNo.Text + "'";

adp.SelectCommand = c.cmd;

dt.Clear();

adp.Fill(dt);

dgCreateOrder.DataSource = dt;

}

privatevoid btnSubmit\_Click(object sender, EventArgs e)

{

c.cmd.CommandText = "insert into CreateOrderTable"

+ " Select \* from CreateOrderDummy";

c.cmd.ExecuteNonQuery();

c.cmd.CommandText = "delete from CreateOrderDummy";

c.cmd.ExecuteNonQuery();

MessageBox.Show("Order Created!");

c.cmd.CommandText = "delete from CreateOrderDummy";

c.cmd.ExecuteNonQuery();

btnClear\_Click(this, EventArgs.Empty);

}

privatevoid NumValidations(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar)

&& e.KeyChar != 0)

{

lblError.Text = "Enter only numbers!";

e.Handled = true;

return;

}

lblError.Text = "...";

}

}

}

10c Generate Bill :

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace Library\_Auto

{

publicpartialclassGenerateBill\_10d : Form

{

publicAdministration\_03 adminInstance = newAdministration\_03();

Connect c = newConnect();

DataTable dt = newDataTable();

DataTable dtAcc = newDataTable();

SqlDataAdapter adp = newSqlDataAdapter();

SqlDataAdapter adp2 = newSqlDataAdapter();

BindingSource bds = newBindingSource();

BindingSource bds2 = newBindingSource();

bool clearSkip = false;

public GenerateBill\_10d()

{

InitializeComponent();

}

protectedoverridevoid OnFormClosing(FormClosingEventArgs e)

{

base.OnFormClosing(e);

switch (e.CloseReason)

{

caseCloseReason.TaskManagerClosing:

caseCloseReason.WindowsShutDown:

return;

default: break;

}

if (clearSkip) return;

if (adminInstance != null)

{

adminInstance.Show();

}

}

privatevoid btnClear\_Click(object sender, EventArgs e)

{

clearSkip = true;

this.Close();

adminInstance.generateBillToolStripMenuItem1\_Click(this, null);

}

privatevoid GenerateBill\_10d\_Load(object sender, EventArgs e)

{

BillDatePicker.Value = DateTime.Today;

grpBookInfo.Enabled = false;

btnSave.Enabled = false;

btnNext.Enabled = false;

btnClear.Enabled = false;

c.cmd.CommandText = "select distinct orderno from CreateOrderTable";

adp.SelectCommand = c.cmd;

dt.Clear();

adp.Fill(dt);

for (int i = 0; i <= dt.Rows.Count - 1; i++)

{

ddlOrderNo.Items.Add(dt.Rows[i].ItemArray[0]);

}

}

privatevoid btnNew\_Click(object sender, EventArgs e)

{

grpBookInfo.Enabled = true;

grpBillDetails.Enabled = true;

grpOrderInfo.Enabled = true;

BillDatePicker.Text = DateTime.Today.Date.ToString();

ddlOrderNo.SelectedItem = null;

ddlAccno.SelectedItem = null;

dtODate.Enabled = false;

txtVenId.Enabled = false;

txtName.Enabled = false;

txtEmail.Enabled = false;

txtAddress.Enabled = false;

txtTitle.Enabled = false;

txtAuthor.Enabled = false;

txtPublisher.Enabled = false;

btnSave.Enabled = true;

btnNext.Enabled = false;

btnClear.Enabled = true;

c.cmd.CommandText = "select max(billno) from GenerateBillTable";

txtBillNo.Text = (c.cmd.ExecuteScalar() == DBNull.Value) ? "1" : Convert.ToString(1 + (int)c.cmd.ExecuteScalar());

txtBillNo.ReadOnly = true;

}

privatevoid btnNext\_Click(object sender, EventArgs e)

{

ddlAccno.SelectedItem = null;

ddlAccno.Enabled = true;

txtTitle.Clear();

txtAuthor.Clear();

txtPublisher.Clear();

txtNoCopy.Clear();

txtPrice.Clear();

txtDiscount.Clear();

txtNetCost.Clear();

txtTAmount.Clear();

btnSave.Enabled = true;

btnNext.Enabled = false;

}

privatevoid ddlOrderNo\_SelectionChangeCommitted(object sender, EventArgs e)

{

string orderNo;

BindingSource bds = newBindingSource();

orderNo = ddlOrderNo.SelectedItem.ToString();

c.cmd.Parameters.Clear();

c.cmd.CommandText = "select orderdate, vendorid, name, email, address from CreateOrderTable where orderno='" + orderNo + "'";

adp.SelectCommand = c.cmd;

dt.Clear();

adp.Fill(dt);

bds.Clear();

bds.DataSource = dt;

dtODate.DataBindings.Clear();

txtVenId.DataBindings.Clear();

txtName.DataBindings.Clear();

txtEmail.DataBindings.Clear();

txtAddress.DataBindings.Clear();

dtODate.DataBindings.Add("text", bds, "orderdate");

txtVenId.DataBindings.Add("text", bds, "vendorid");

txtName.DataBindings.Add("text", bds, "name");

txtEmail.DataBindings.Add("text", bds, "email");

txtAddress.DataBindings.Add("text", bds, "address");

ddlAccno.Items.Clear();

c.cmd.CommandText="select accno from CreateOrderTable where orderno ='" + orderNo + "'";

adp.SelectCommand=c.cmd;

dtAcc.Clear();

adp.Fill(dtAcc);

DataTable dtAccBill = newDataTable();

c.cmd.CommandText = "select accno from GenerateBillTable where orderno ='" + orderNo + "'";

adp.SelectCommand = c.cmd;

adp.Fill(dtAccBill);

for(int i = 0; i < dtAcc.Rows.Count; i++)

{

ddlAccno.Items.Add(dtAcc.Rows[i].ItemArray[0]);

}

for (int i = 0; i < dtAccBill.Rows.Count; i++)

{

if (ddlAccno.Items.Contains((string)dtAccBill.Rows[i].ItemArray[0]))

{

ddlAccno.Items.Remove((string)dtAccBill.Rows[i].ItemArray[0]);

}

}

DataTable dtGrid = newDataTable();

c.cmd.CommandText = "select \* from GenerateBillTable where orderno ='" + orderNo + "'";

adp.SelectCommand = c.cmd;

adp.Fill(dtGrid);

dataGridView1.DataSource = dtGrid;

}

privatevoid ddlAccno\_SelectionChangeCommitted(object sender, EventArgs e)

{

//SqlDataAdapter adp =new SqlDataAdapter();

//DataTable dt=new DataTable();

//BindingSource bds =new BindingSource();

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

c.cmd.CommandText="select count(\*) from GenerateBillTable where accno='" + ddlAccno.SelectedItem.ToString() + "' and orderno='"+ddlOrderNo.Text+"'";

int a = (int) c.cmd.ExecuteScalar();

if (a > 0)

{

MessageBox.Show("Invoice already generated for this Book", "Information", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

DataTable dtDummy = newDataTable();

c.cmd.CommandText = "select title, author, publisher, noofcopies from CreateOrderTable where orderno ='"+ddlOrderNo.SelectedItem.ToString()+"' and accno ='"+ddlAccno.SelectedItem.ToString()+"'";

adp.SelectCommand = c.cmd;

adp.Fill(dtDummy);

bds2.DataSource = dtDummy;

txtTitle.DataBindings.Clear();

txtAuthor.DataBindings.Clear();

txtPublisher.DataBindings.Clear();

txtNoCopy.DataBindings.Clear();

txtTitle.DataBindings.Add("text", bds2, "title");

txtAuthor.DataBindings.Add("text", bds2, "author");

txtPublisher .DataBindings.Add("text", bds2, "publisher");

txtNoCopy.DataBindings.Add("text", bds2, "noofcopies");

}

}

privatevoid btnSave\_Click(object sender, EventArgs e)

{

if (ddlAccno.SelectedItem == null)

{

MessageBox.Show("Select an Item!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (txtPrice.Text == "" || txtDiscount.Text == "")

{

MessageBox.Show("Enter the price and Discount!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else

{

c.cmd.Parameters.Clear();

c.cmd.CommandText="insert into GenerateBilltable values(@billno,@billdate,@orderno,@orderdate,@vendorid,@name,@accno,@title,@author,@publisher,@noofcopies,@price,@totamt)";

c.cmd.Parameters.Add("@billno", SqlDbType.VarChar).Value = txtBillNo.Text;

c.cmd.Parameters.Add("@billdate",SqlDbType.VarChar).Value=BillDatePicker.Text;

c.cmd.Parameters.Add("@orderno",SqlDbType.VarChar).Value=ddlOrderNo.Text;

c.cmd.Parameters.Add("@orderdate",SqlDbType.VarChar).Value=dtODate.Text;

c.cmd.Parameters.Add("@vendorid",SqlDbType.VarChar).Value=txtVenId.Text;

c.cmd.Parameters.Add("@name",SqlDbType.VarChar).Value=txtName.Text;

c.cmd.Parameters.Add("@accno",SqlDbType.VarChar).Value=ddlAccno.Text;

c.cmd.Parameters.Add("@title",SqlDbType.VarChar).Value=txtTitle.Text;

c.cmd.Parameters.Add("@author",SqlDbType.VarChar).Value=txtAuthor.Text;

c.cmd.Parameters.Add("@publisher",SqlDbType.VarChar).Value=txtPublisher.Text;

c.cmd.Parameters.Add("@noofcopies",SqlDbType.VarChar).Value=txtNoCopy.Text;

c.cmd.Parameters.Add("@price", SqlDbType.VarChar).Value = txtPrice.Text;

int noofcopies, price;

noofcopies = Convert.ToInt16(txtNoCopy.Text);

price = Convert.ToInt16(txtPrice.Text);

txtTAmount.Text = "" + noofcopies \* price;

c.cmd.Parameters.Add("@totamt",SqlDbType.VarChar).Value=txtTAmount.Text;

c.cmd.ExecuteNonQuery();

MessageBox.Show("Invoice information saved", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

btnSave.Enabled = false;

btnNext.Enabled = true;

DataTable dtBill = newDataTable();

c.cmd.CommandText = "select \* from GenerateBillTable ORDER BY billno ASC";

adp.SelectCommand = c.cmd;

adp.Fill(dtBill);

dataGridView1.DataSource = dtBill;

//Auto Insert to StockTable

c.cmd.CommandText = "select count(\*) from StockTable where accno = '" + ddlAccno.Text + "'";

if ((int)c.cmd.ExecuteScalar() <= 0)

{

MessageBox.Show("Could not find copy details of the Book!"

+ " \nManually insert the Book details in AccRegister");

return;

}

string msg;

msg = noofcopies + " copies added to Stock, \n Copy Numbers are :";

for (int i = 0; i < noofcopies; i++)

{

c.cmd.CommandText = "select max(copyno) from StockTable where accno = '" + ddlAccno.Text + "'";

int copyno = (c.cmd.ExecuteScalar() == DBNull.Value) ? 1 : 1 + (int)c.cmd.ExecuteScalar();

msg += "" + copyno;

c.cmd.CommandText = "select \* from StockTable where accno ='" + ddlAccno.Text + "' and copyno =" + (copyno - 1);

DataTable dtStock = newDataTable();

adp.SelectCommand = c.cmd;

adp.Fill(dtStock);

c.cmd.CommandText = "insert into StockTable values(@accno, @copyno, @vendor, @source, @currency, @dept, @edition, @billno, @billdate, @discount, @status, @category, @price, @netcost, @pages, @location, @binding, @copyyear)";

c.cmd.Parameters.Clear();

c.cmd.Parameters.AddWithValue("@accno", ddlAccno.Text);

c.cmd.Parameters.AddWithValue("@copyno", copyno);

c.cmd.Parameters.AddWithValue("@vendor", txtName.Text + "" + txtAddress.Text + "");

c.cmd.Parameters.AddWithValue("@source", (string)dtStock.Rows[0].ItemArray[3]);

c.cmd.Parameters.AddWithValue("@currency", (string)dtStock.Rows[0].ItemArray[4]);

c.cmd.Parameters.AddWithValue("@dept", (string)dtStock.Rows[0].ItemArray[5]);

c.cmd.Parameters.AddWithValue("@edition", (string)dtStock.Rows[0].ItemArray[6]);

c.cmd.Parameters.AddWithValue("@billno", txtBillNo.Text);

c.cmd.Parameters.AddWithValue("@billdate", BillDatePicker.Value);

c.cmd.Parameters.AddWithValue("@discount", txtDiscount.Text);

c.cmd.Parameters.AddWithValue("@status", "Available");

c.cmd.Parameters.AddWithValue("@category", (string)dtStock.Rows[0].ItemArray[11]);

c.cmd.Parameters.AddWithValue("@price", txtPrice.Text);

c.cmd.Parameters.AddWithValue("@netcost", txtNetCost.Text);

c.cmd.Parameters.AddWithValue("@pages", (string)dtStock.Rows[0].ItemArray[14]);

c.cmd.Parameters.AddWithValue("@location", (string)dtStock.Rows[0].ItemArray[15]);

c.cmd.Parameters.AddWithValue("@binding", (string)dtStock.Rows[0].ItemArray[16]);

c.cmd.Parameters.AddWithValue("@copyyear", (string)dtStock.Rows[0].ItemArray[17]);

c.cmd.ExecuteNonQuery();

}

MessageBox.Show(msg, "Copies Added", MessageBoxButtons.OK, MessageBoxIcon.Information);

ddlAccno.Items.Remove(ddlAccno.SelectedItem.ToString());

ddlAccno.Enabled = false;

}

}

privatevoid txtPrice\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) && (e.KeyChar != '.'))

{

MessageBox.Show("Enter Only Numbers", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);

e.Handled = true;

}

}

privatevoid txtBillNo\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) && (e.KeyChar != '.'))

{

MessageBox.Show("Enter Only Numbers", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);

e.Handled = true;

}

}

privatevoid txtDiscount\_TextChanged(object sender, EventArgs e)

{

try

{

int price, discount, netCost, copies;

price = Convert.ToInt32(txtPrice.Text);

discount = Convert.ToInt32(txtDiscount.Text);

copies = Convert.ToInt32(txtNoCopy.Text);

netCost = price - (price \* discount / 100);

txtNetCost.Text = "" + netCost;

txtTAmount.Text = "" + copies \* netCost;

}

catch (FormatException)

{

btnSave.Enabled = false;

}

finally { btnSave.Enabled = true; }

}

//Number Validations

privatevoid NumValidations(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar)

&& e.KeyChar != 0)

{

MessageBox.Show("Enter only numbers!");

e.Handled = true;

return;

}

}

}

}

11 View Issued:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using Microsoft.VisualBasic;

namespace Library\_Auto

{

publicpartialclassViewIssued\_11 : Form

{

Connect c = newConnect();

SqlDataAdapter adp = newSqlDataAdapter();

DataTable dt = newDataTable();

publicstring state = "";

public ViewIssued\_11()

{

InitializeComponent();

}

privatevoid btnAll\_Click(object sender, EventArgs e)

{

DataTable dt = newDataTable();

c.cmd.CommandText = "select \* from IssueTable";

adp.SelectCommand = c.cmd;

adp.Fill(dt);

dataGridView1.DataSource = dt;

}

privatevoid btnLost\_Click(object sender, EventArgs e)

{

int issueIDMax;

string status;

if (c.cnn.State != ConnectionState.Open) { c.cnn.Close(); c.cnn.Open(); }

c.cmd.CommandText = "select max(issueid) from issueTable where memberid ='" + txtMemID.Text + "'";

issueIDMax = (int) c.cmd.ExecuteScalar();

c.cmd.CommandText = "select status from issueTable where issueID='" + issueIDMax + "'";

status = (string) c.cmd.ExecuteScalar();

if (status != "Issued")

{

MessageBox.Show("The issue status is " + status

+ "\n\n To set Book Status to Lost the Issue Status must be \"Issued\"",

"Book Lost", MessageBoxButtons.OK, MessageBoxIcon.Stop);

return;

}

}

privatevoid btnFind\_Click(object sender, EventArgs e)

{

dt.Clear();

c.cmd.CommandText = "select count(\*) from IssueTable where memberid = '" + txtMemID.Text + "'";

if ((int)c.cmd.ExecuteScalar() > 0)

{

btnLost.Enabled = true;

}

else btnLost.Enabled = false;

c.cmd.CommandText = "select \* from IssueTable where memberid = '" + txtMemID.Text + "'";

adp.SelectCommand = c.cmd;

adp.Fill(dt);

dataGridView1.DataSource = dt;

}

privatevoid BookLost\_11\_Load(object sender, EventArgs e)

{

if (state == "View Issued")

{

lblLostBook.Text = "View Issued";

this.Text = "View Issued";

this.Controls.Remove(btnLost);

this.Controls.Remove(txtMemID);

this.Controls.Remove(btnFind);

this.Controls.Remove(lblIssueId);

} }

}

}

09a Accession Report:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using CrystalDecisions.CrystalReports.Engine;

using System.Data.SqlClient;

using Microsoft.VisualBasic;

namespace Library\_Auto.Crystal\_Reports

{

publicpartialclassAccReg\_rpt : Form

{

publicAdministration\_03 adminInstance;

Connect c = newConnect();

SqlDataAdapter adp = newSqlDataAdapter();

string selectionType;

public AccReg\_rpt()

{

InitializeComponent();

}

privatevoid Reports\_Base\_Load(object sender, EventArgs e)

{

}

protectedoverridevoid OnFormClosing(FormClosingEventArgs e)

{

base.OnFormClosing(e);

switch (e.CloseReason)

{

caseCloseReason.WindowsShutDown:

caseCloseReason.TaskManagerClosing:

return;

default: break;

}

if (adminInstance != null) adminInstance.Show();

}

privatevoid ddlType\_SelectionChangeCommitted(object sender, EventArgs e)

{

switch (ddlType.SelectedItem.ToString())

{

case"Section":

DdlUpdate("select section from AccRegTable", ddlRpt);

selectionType = "{AccRegTable.section}";

break;

case"Type":

DdlUpdate("select type from AccRegTable", ddlRpt);

selectionType = "{AccRegTable.type}";

break;

case"Language":

DdlUpdate("select language from AccRegTable", ddlRpt);

selectionType = "{AccRegTable.language}";

break;

case"Subject":

DdlUpdate("select subject from AccRegTable", ddlRpt);

selectionType = "{AccRegTable.subject}";

break;

case"Publisher":

DdlUpdate("select publisher from AccRegTable", ddlRpt);

selectionType = "{AccRegTable.publisher}";

break;

default: Interaction.Beep(); break;

}

}

privatevoid DdlUpdate(string cmdText, ComboBox ddlToFill)

{

DataTable dtDummy = newDataTable();

string itemToAdd = "";

try

{

ddlToFill.Items.Clear();

c.cmd.CommandText = cmdText;

adp.SelectCommand = c.cmd;

adp.Fill(dtDummy);

for (int i = 0; i < dtDummy.Rows.Count; i++)

{

itemToAdd = "";

itemToAdd += dtDummy.Rows[i].ItemArray[0];

if (!ddlToFill.Items.Contains(itemToAdd))

{

ddlToFill.Items.Add(itemToAdd);

}

}

}

catch (Exception ex)

{

adminInstance.errObj.txtException.Text = ex.ToString();

adminInstance.errObj.Show();

}

}

privatevoid btnDisplayType\_Click(object sender, EventArgs e)

{

try

{

cryRptViewer.SelectionFormula = selectionType + "='" + ddlRpt.SelectedItem.ToString() + "'";

cryRptViewer.RefreshReport();

}

catch (Exception)

{

MessageBox.Show("Select an Item!");

}

}

privatevoid btnDisplayTextbased\_Click(object sender, EventArgs e)

{

string selectionType;

//No Format Exceptions to catch in this form.

switch (ddlTextType.SelectedItem.ToString())

{

case"Accno": selectionType = "{AccRegTable.accno}"; break;

case"Title": selectionType = "{AccRegTable.title}"; break;

case"Author": selectionType = "{AccRegTable.author}"; break;

default: Interaction.Beep(); return;

}

cryRptViewer.SelectionFormula = selectionType + " ='" + txtSearch.Text + "'";

cryRptViewer.RefreshReport();

}

privatevoid btnReset\_Click(object sender, EventArgs e)

{

cryRptViewer.SelectionFormula = "1=1";

cryRptViewer.RefreshReport();

}

}

}

09b Circulation Report:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using CrystalDecisions.CrystalReports.Engine;

using Microsoft.VisualBasic;

namespace Library\_Auto.Crystal\_Reports

{

publicpartialclassCirculation\_rpt : Form

{

publicAdministration\_03 adminInstance;

public Circulation\_rpt(){InitializeComponent();}

protectedoverridevoid OnFormClosing(FormClosingEventArgs e)

{

base.OnFormClosing(e);

switch (e.CloseReason)

{

caseCloseReason.WindowsShutDown:

caseCloseReason.TaskManagerClosing:

return;

default: break;

}

if (adminInstance != null) adminInstance.Show();

}

privatevoid Reports\_Base\_Load(object sender, EventArgs e)

{

ddlDateType.SelectedIndex = 0;

ddlTextType.SelectedIndex = 0;

}

privatevoid btnDisplay\_Click(object sender, EventArgs e)

{

if (dtStart.Value > dtEnd.Value)

{

MessageBox.Show("'From Date' should be smaller than 'To Date'!");

return;

}

string dtStartYear, dtStartMonth, dtStartDay;

dtStartYear = dtStart.Value.Year.ToString();

dtStartMonth = dtStart.Value.Month.ToString();

dtStartDay = dtStart.Value.Day.ToString();

string dtEndYear, dtEndMonth, dtEndDay;

dtEndYear = dtEnd.Value.Year.ToString();

dtEndMonth = dtEnd.Value.Month.ToString();

dtEndDay = dtEnd.Value.Day.ToString();

string selectionType;

switch (ddlDateType.SelectedItem.ToString())

{

case"Issue-Date": selectionType = "{IssueTable.issuedate}"; break;

case"Renew-Date": selectionType = "{IssueTable.renewdate}"; break;

case"Return-Date": selectionType = "{IssueTable.returndate}"; break;

case"Due-Date": selectionType = "{IssueTable.duedate}"; break;

default: Interaction.Beep(); return;

}

cryRptViewer.SelectionFormula =

selectionType + ">= DateTime (" + dtStartYear + "," + dtStartMonth + "," + dtStartDay + ") and "

+ selectionType + "<= DateTime (" + dtEndYear + "," + dtEndMonth + "," + dtEndDay + ")";

cryRptViewer.RefreshReport();

}

privatebool CatchFormatExceptions()

{

try

{

switch (ddlTextType.SelectedItem.ToString())

{

case"Issue ID":

case"Fine > [Number]":

case"Fine < [Number]":

Convert.ToUInt16(txtSearch.Text);

break;

default: break;

}

}

catch (FormatException)

{

MessageBox.Show("Enter only Number!");

returntrue;

} returnfalse;

}

privatevoid btnDisplayTextbased\_Click(object sender, EventArgs e)

{

string selectionType;

bool caughtFormatException = CatchFormatExceptions();

if (caughtFormatException) return;

switch (ddlTextType.SelectedItem.ToString())

{

case"Issue ID":

selectionType = "{IssueTable.issueid}";

cryRptViewer.SelectionFormula = selectionType + "=" + txtSearch.Text;

cryRptViewer.RefreshReport();

return;

case"Fine > [Number]":

selectionType = "{IssueTable.fine}";

cryRptViewer.SelectionFormula = selectionType + ">" + txtSearch.Text;

cryRptViewer.RefreshReport();

return;

case"Fine < [Number]":

selectionType = "{IssueTable.fine}";

cryRptViewer.SelectionFormula = selectionType + "<" + txtSearch.Text;

cryRptViewer.RefreshReport();

return;

case"Member ID": selectionType = "{IssueTable.memberid}"; break;

case"Accession No": selectionType = "{IssueTable.accno}"; break;

case"Status = Issued": selectionType = "{IssueTable.status}"; break;

case"Status = Returned": selectionType = "{IssueTable.status}"; break;

default: Interaction.Beep(); return;

}

cryRptViewer.SelectionFormula= selectionType + " ='" + txtSearch.Text + "'";

cryRptViewer.RefreshReport();

}

privatevoid btnReset\_Click(object sender, EventArgs e)

{

txtSearch.Clear();

cryRptViewer.SelectionFormula = "1=1";

cryRptViewer.RefreshReport();

}

privatevoid ddlTextType\_SelectionChangeCommitted(object sender, EventArgs e){

if (ddlTextType.SelectedItem.Equals("Status = Issued")){txtSearch.Text = "Issued";}

if (ddlTextType.SelectedItem.Equals("Status = Returned")){txtSearch.Text = "Returned";}

}

}

}

09c Members Report:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using CrystalDecisions.CrystalReports.Engine;

using Microsoft.VisualBasic;

using System.Data.SqlClient;

namespace Library\_Auto.Crystal\_Reports

{

publicpartialclassMembers\_rpt : Form

{

publicAdministration\_03 adminInstance;

Connect c = newConnect();

SqlDataAdapter adp = newSqlDataAdapter();

string selectionType;

public Members\_rpt() { InitializeComponent(); }

protectedoverridevoid OnFormClosing(FormClosingEventArgs e)

{

base.OnFormClosing(e);

switch (e.CloseReason)

{

caseCloseReason.WindowsShutDown:

caseCloseReason.TaskManagerClosing:

return;

default: break;

}

if (adminInstance != null) adminInstance.Show();

}

privatevoid Reports\_Base\_Load(object sender, EventArgs e)

{

ddlDateType.SelectedIndex = 0;

}

privatevoid btnDisplayDate\_Click(object sender, EventArgs e)

{

if (dtStart.Value > dtEnd.Value)

{

MessageBox.Show("'From Date' should be smaller than 'To Date'!");

return;

}

string dtStartYear, dtStartMonth, dtStartDay;

dtStartYear = dtStart.Value.Year.ToString();

dtStartMonth = dtStart.Value.Month.ToString();

dtStartDay = dtStart.Value.Day.ToString();

string dtEndYear, dtEndMonth, dtEndDay;

dtEndYear = dtEnd.Value.Year.ToString();

dtEndMonth = dtEnd.Value.Month.ToString();

dtEndDay = dtEnd.Value.Day.ToString();

string selectionType;

switch (ddlDateType.SelectedItem.ToString())

{

case"Subscription Date": selectionType = "{MembersTable.subsdate}"; break;

case"Expiry Date": selectionType = "{MembersTable.expdate}"; break;

default: Interaction.Beep(); return;

}

cryRptViewer.SelectionFormula =

selectionType + ">= DateTime (" + dtStartYear + "," + dtStartMonth + "," + dtStartDay + ") and "

+ selectionType + "<= DateTime (" + dtEndYear + "," + dtEndMonth + "," + dtEndDay + ")";

cryRptViewer.RefreshReport();

}

privatevoid DdlUpdate(string cmdText, ComboBox ddlToFill)

{

DataTable dtDummy = newDataTable();

string itemToAdd = "";

try

{

ddlToFill.Items.Clear();

c.cmd.CommandText = cmdText;

adp.SelectCommand = c.cmd;

adp.Fill(dtDummy);

for (int i = 0; i < dtDummy.Rows.Count; i++)

{

itemToAdd = "";

itemToAdd += dtDummy.Rows[i].ItemArray[0];

if (!ddlToFill.Items.Contains(itemToAdd))

{

ddlToFill.Items.Add(itemToAdd);

}

}

}

catch (Exception ex)

{

adminInstance.errObj.txtException.Text = ex.ToString();

adminInstance.errObj.Show();

}

}

privatevoid cbType\_SelectionChangeCommitted(object sender, EventArgs e)

{

switch (ddlType.SelectedItem.ToString())

{

case"Member Type":

DdlUpdate("select memtype from MemberParameters", ddlRpt);

selectionType = "{MembersTable.membertype}";

break;

case"Course":

DdlUpdate("select course from MembersTable", ddlRpt);

selectionType = "{MembersTable.course}";

break;

case"Status":

DdlUpdate("select status from MembersTable", ddlRpt);

selectionType = "{MembersTable.status}";

break;

default: MessageBox.Show(""); break;

}

}

privatevoid btnReset\_Click(object sender, EventArgs e)

{

cryRptViewer.SelectionFormula = "1=1";

cryRptViewer.RefreshReport();

}

privatevoid btnDisplayType\_Click(object sender, EventArgs e)

{

try

{

cryRptViewer.SelectionFormula = selectionType + "='" + ddlRpt.SelectedItem.ToString() + "'";

cryRptViewer.RefreshReport();

}

catch (Exception)

{

MessageBox.Show("Select an Item!");

}

}

privatebool CatchFormatExceptions()

{

try

{

switch (ddlTextType.SelectedItem.ToString())

{

case"Fine > [Number]":

case"Fine < [Number]":

Convert.ToInt16(txtSearch.Text);

break;

default: break;

}

}

catch (FormatException)

{

MessageBox.Show("Enter only Numeric Value!");

returntrue;

}

returnfalse;

}

privatevoid btnDisplayTextbased\_Click(object sender, EventArgs e)

{

string selectionType;

bool caughtFormatException = CatchFormatExceptions();

if (caughtFormatException) return;

switch (ddlTextType.SelectedItem.ToString())

{

case"Fine > [Number]":

selectionType = "{MembersTable.fine}";

cryRptViewer.SelectionFormula = selectionType + ">" + txtSearch.Text + " and " + selectionType + "<> 0";

cryRptViewer.RefreshReport();

return;

case"Fine < [Number]":

selectionType = "{MembersTable.fine}";

cryRptViewer.SelectionFormula = selectionType + "<" + txtSearch.Text + " and " + selectionType + "<> 0";

cryRptViewer.RefreshReport();

return;

case"Member ID": selectionType = "{MembersTable.memberid}"; break;

case"Firstname": selectionType = "{MembersTable.firstname}"; break;

case"Lastname": selectionType = "{MembersTable.lastname}"; break;

case"Email": selectionType = "{MembersTable.email}"; break;

default: Interaction.Beep(); return;

}

cryRptViewer.SelectionFormula = selectionType + " ='" + txtSearch.Text + "'";

cryRptViewer.RefreshReport();

}

}

}

09d Stock Report:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using CrystalDecisions.CrystalReports.Engine;

using Microsoft.VisualBasic;

using System.Data.SqlClient;

namespace Library\_Auto.Crystal\_Reports

{

publicpartialclassStock\_rpt : Form

{

publicAdministration\_03 adminInstance;

Connect c = newConnect();

SqlDataAdapter adp = newSqlDataAdapter();

string selectionType;

public Stock\_rpt() { InitializeComponent(); }

protectedoverridevoid OnFormClosing(FormClosingEventArgs e)

{

base.OnFormClosing(e);

switch (e.CloseReason)

{

caseCloseReason.WindowsShutDown:

caseCloseReason.TaskManagerClosing:

return;

default: break;

}

if (adminInstance != null) adminInstance.Show();

}

privatevoid Stock\_rpt\_Load(object sender, EventArgs e)

{

ddlDateType.SelectedIndex = 0;

}

privatevoid btnDisplayDate\_Click(object sender, EventArgs e)

{

if (dtStart.Value > dtEnd.Value)

{

MessageBox.Show("'From Date' should be smaller than 'To Date'!");

return;

}

string dtStartYear, dtStartMonth, dtStartDay;

dtStartYear = dtStart.Value.Year.ToString();

dtStartMonth = dtStart.Value.Month.ToString();

dtStartDay = dtStart.Value.Day.ToString();

string dtEndYear, dtEndMonth, dtEndDay;

dtEndYear = dtEnd.Value.Year.ToString();

dtEndMonth = dtEnd.Value.Month.ToString();

dtEndDay = dtEnd.Value.Day.ToString();

string selectionType;

switch (ddlDateType.SelectedItem.ToString())

{

case"Bill Date": selectionType = "{StockTable.billdate}"; break;

default: Interaction.Beep(); return;

}

cryRptViewer.SelectionFormula =

selectionType + ">= DateTime (" + dtStartYear + "," + dtStartMonth + "," + dtStartDay + ") and "

+ selectionType + "<= DateTime (" + dtEndYear + "," + dtEndMonth + "," + dtEndDay + ")";

cryRptViewer.RefreshReport();

}

privatevoid btnReset\_Click(object sender, EventArgs e)

{

cryRptViewer.SelectionFormula = "1=1";

cryRptViewer.RefreshReport();

}

privatevoid btnDisplayType\_Click(object sender, EventArgs e)

{

try

{

cryRptViewer.SelectionFormula = selectionType + "='" + ddlRpt.SelectedItem.ToString() + "'";

cryRptViewer.RefreshReport();

}

catch (Exception) { MessageBox.Show("Select an Item!"); }

}

privatevoid ddlType\_SelectionChangeCommitted(object sender, EventArgs e)

{

switch (ddlType.SelectedItem.ToString())

{

case"Vendor":

DdlUpdate("select vendor from StockTable", ddlRpt);

selectionType = "{StockTable.vendor}";

break;

case"Status":

DdlUpdate("select status from StockTable", ddlRpt);

selectionType = "{StockTable.status}";

break;

case"Binding":

DdlUpdate("select binding from StockTable", ddlRpt);

selectionType = "{StockTable.binding}";

break;

default: Interaction.Beep(); break;

}

}

privatevoid DdlUpdate(string cmdText, ComboBox ddlToFill)

{

DataTable dtDummy = newDataTable();

string itemToAdd = "";

try

{

ddlToFill.Items.Clear();

c.cmd.CommandText = cmdText;

adp.SelectCommand = c.cmd;

adp.Fill(dtDummy);

for (int i = 0; i < dtDummy.Rows.Count; i++)

{

itemToAdd = "";

itemToAdd += dtDummy.Rows[i].ItemArray[0];

if (!ddlToFill.Items.Contains(itemToAdd))

{

ddlToFill.Items.Add(itemToAdd);

}

}

}

catch (Exception ex)

{

adminInstance.errObj.txtException.Text = ex.ToString();

adminInstance.errObj.Show();

}

}

privatevoid btnDisplayTextbased\_Click(object sender, EventArgs e)

{

string selectionType;

switch (ddlTextType.SelectedItem.ToString())

{

case"Accno": selectionType = "{StockTable.accno}"; break;

case"Title": selectionType = "{AccRegTable.title}"; break;

case"Author": selectionType = "{AccRegTable.author}"; break;

case"Billno": selectionType = "{StockTable.billno}"; break;

case"Year": selectionType = "{StockTable.copyyear}"; break;

default: Interaction.Beep(); return;

}

cryRptViewer.SelectionFormula = selectionType + " ='" + txtSearch.Text + "'";

cryRptViewer.RefreshReport();

}

}

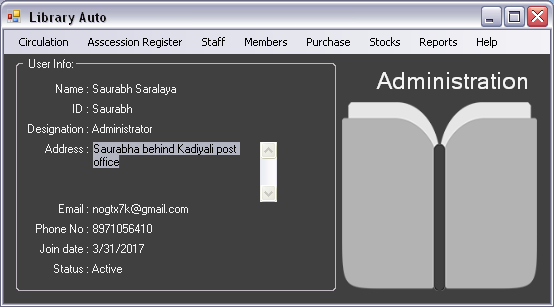
}

4 User Interface:

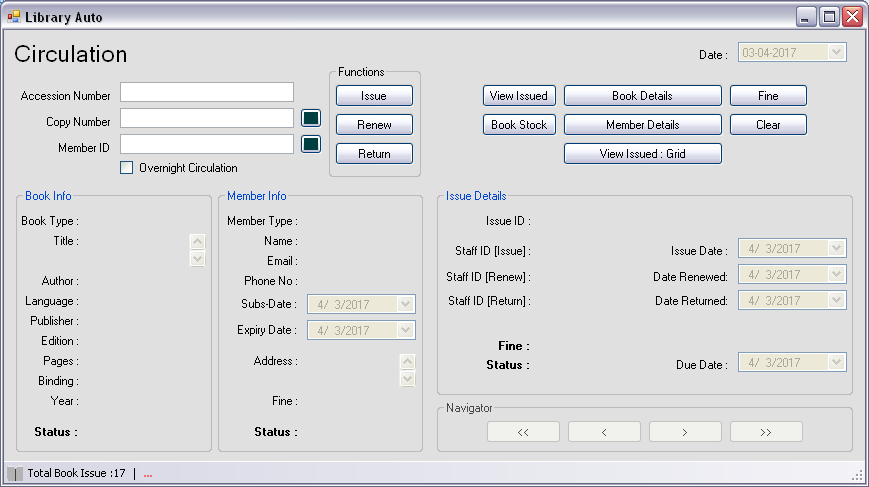
Login:



Administration:

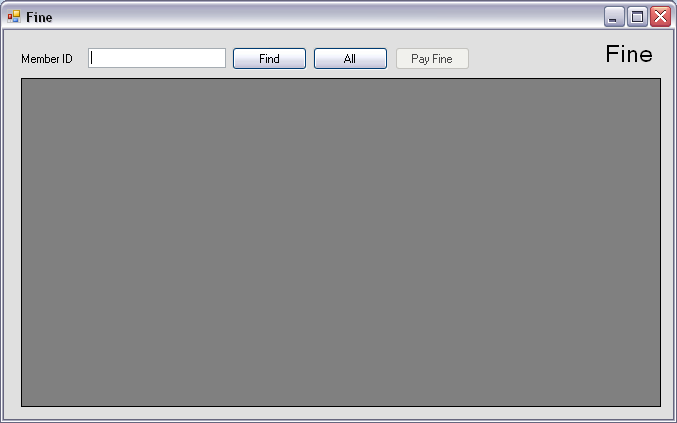


Circulation:

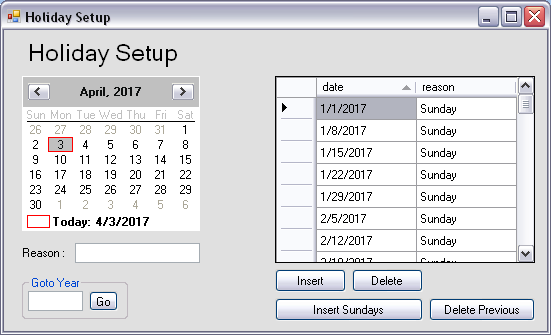


View Issued:

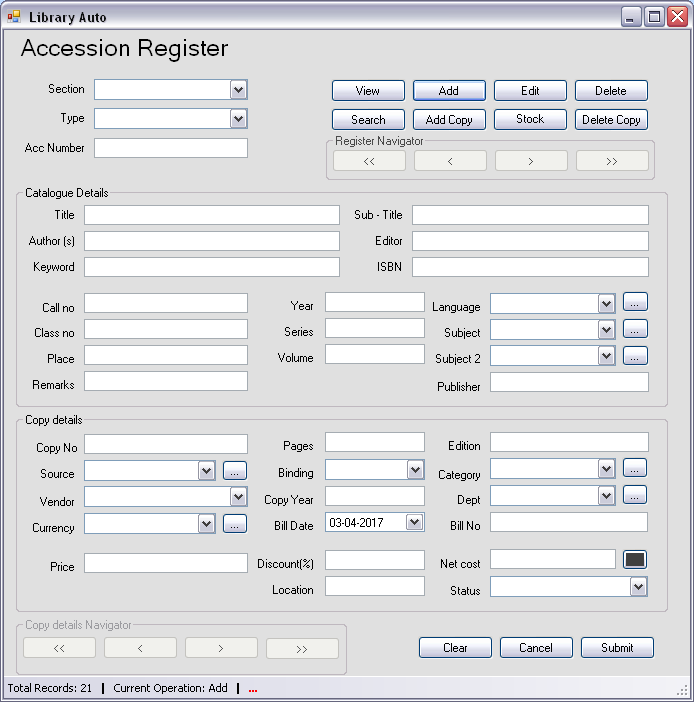
Fine:



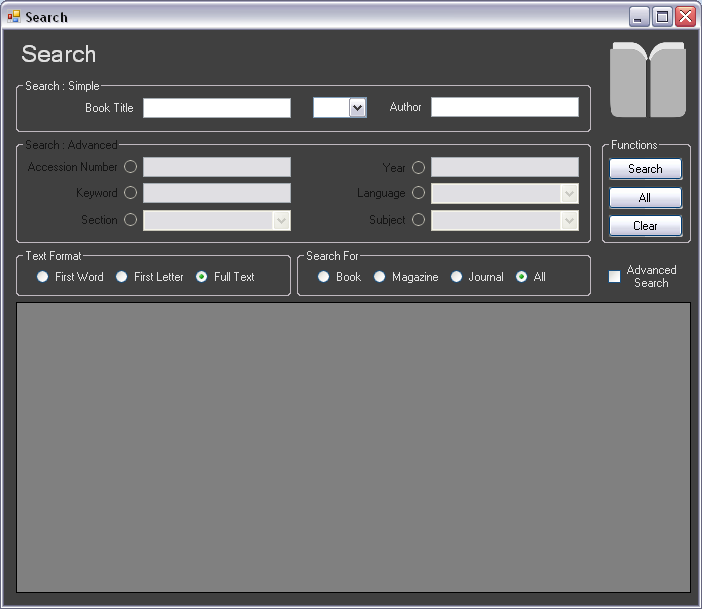
Holiday Management:



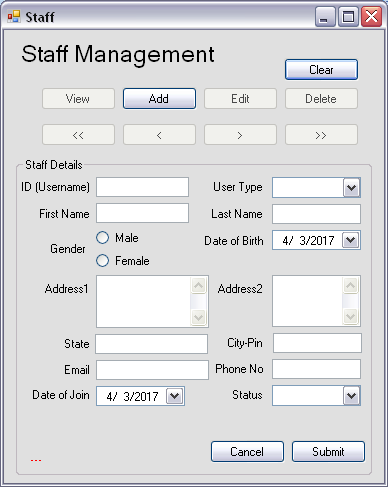
Accession Register:



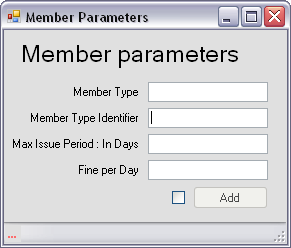
Search/ OPAC(Online Public Access Catalog):



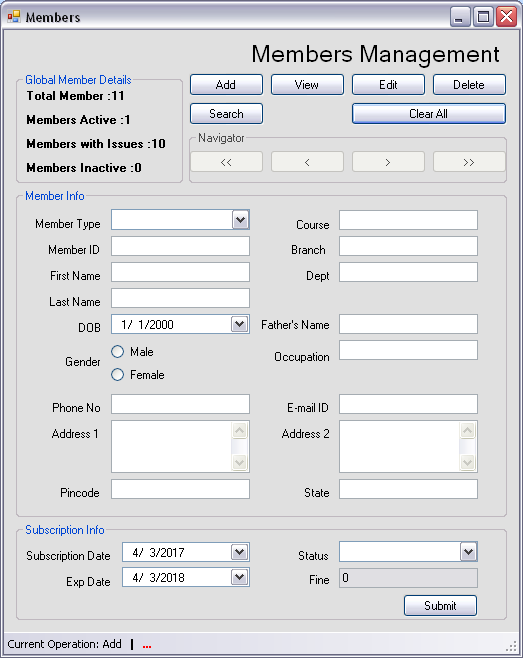
Staff Management:



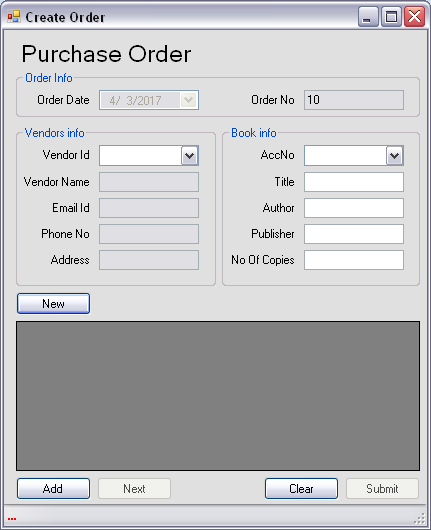
Members parameter:



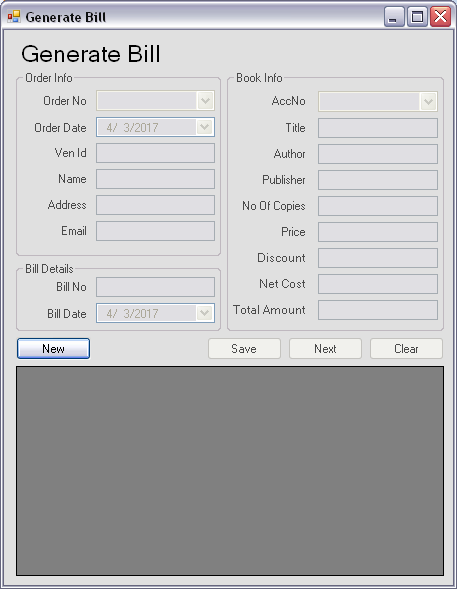
Members Management:



Purchase Order:

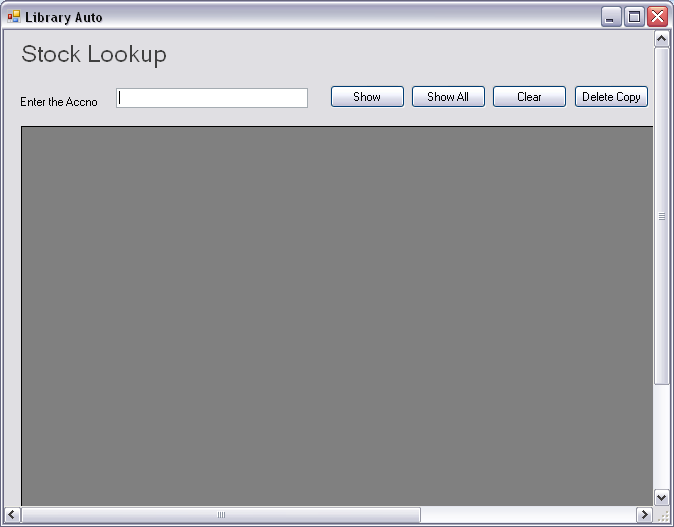


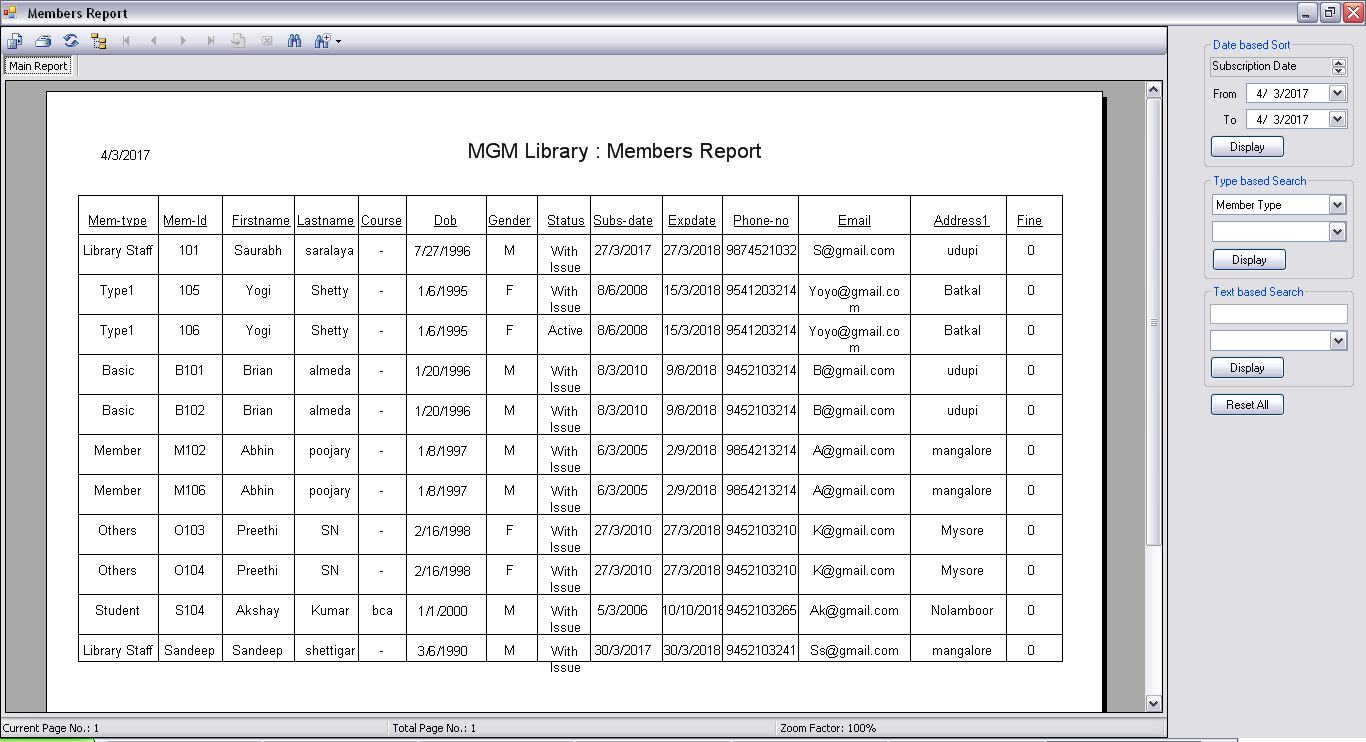
Generate bill:

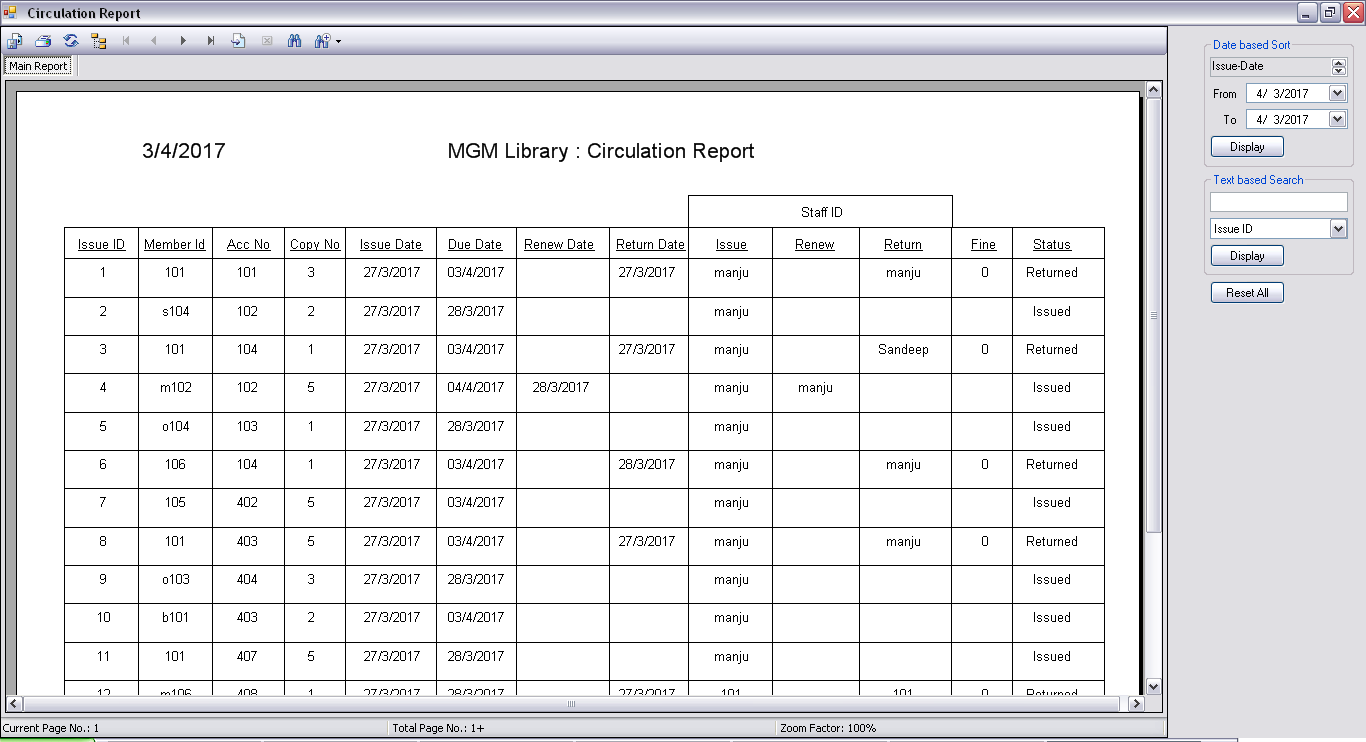


Vendor Management:

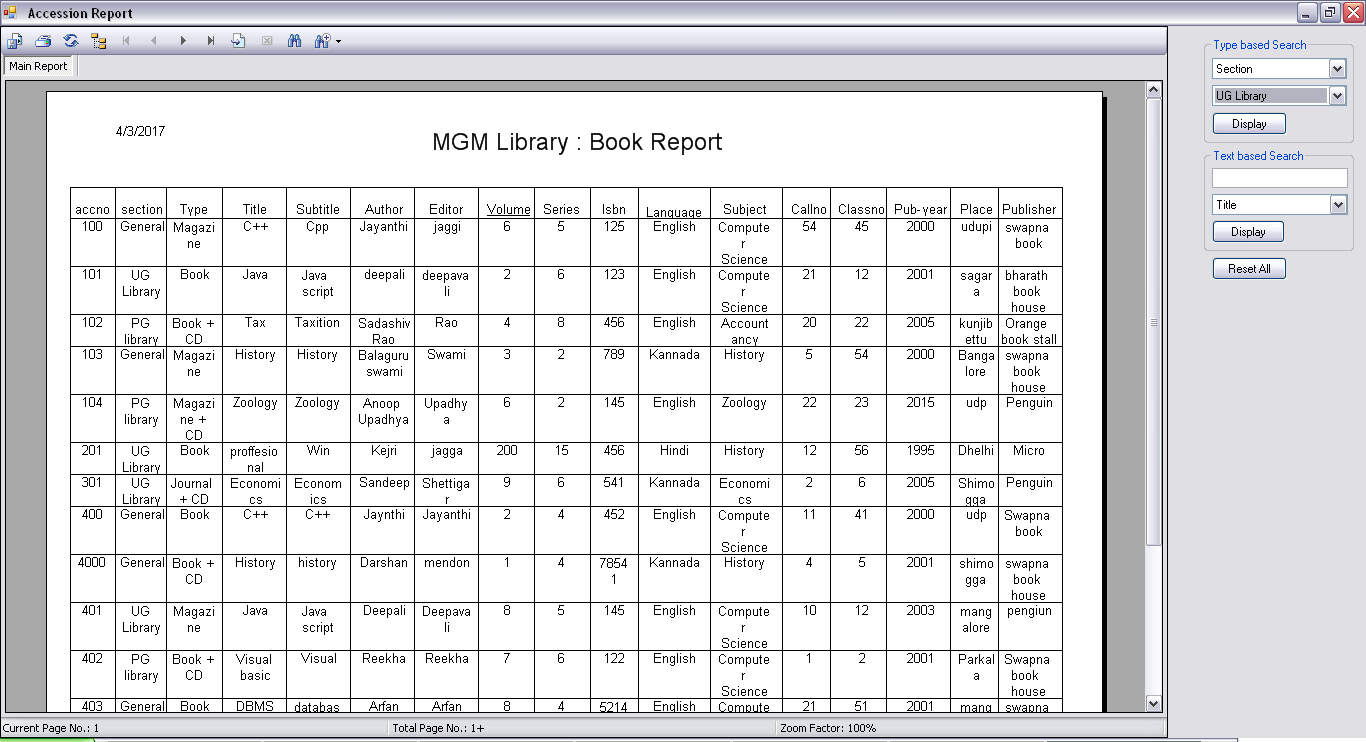


Stocklookup:

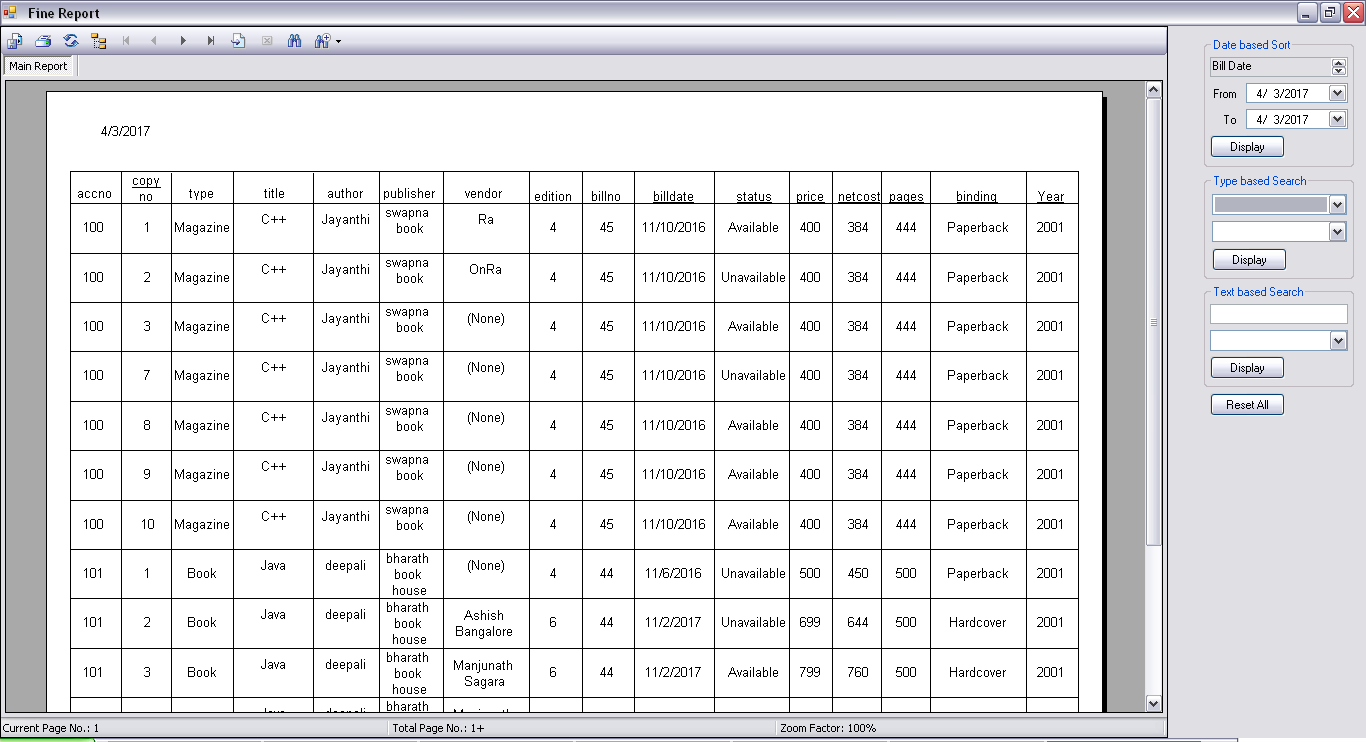
MembersReport:

CirculationReport:

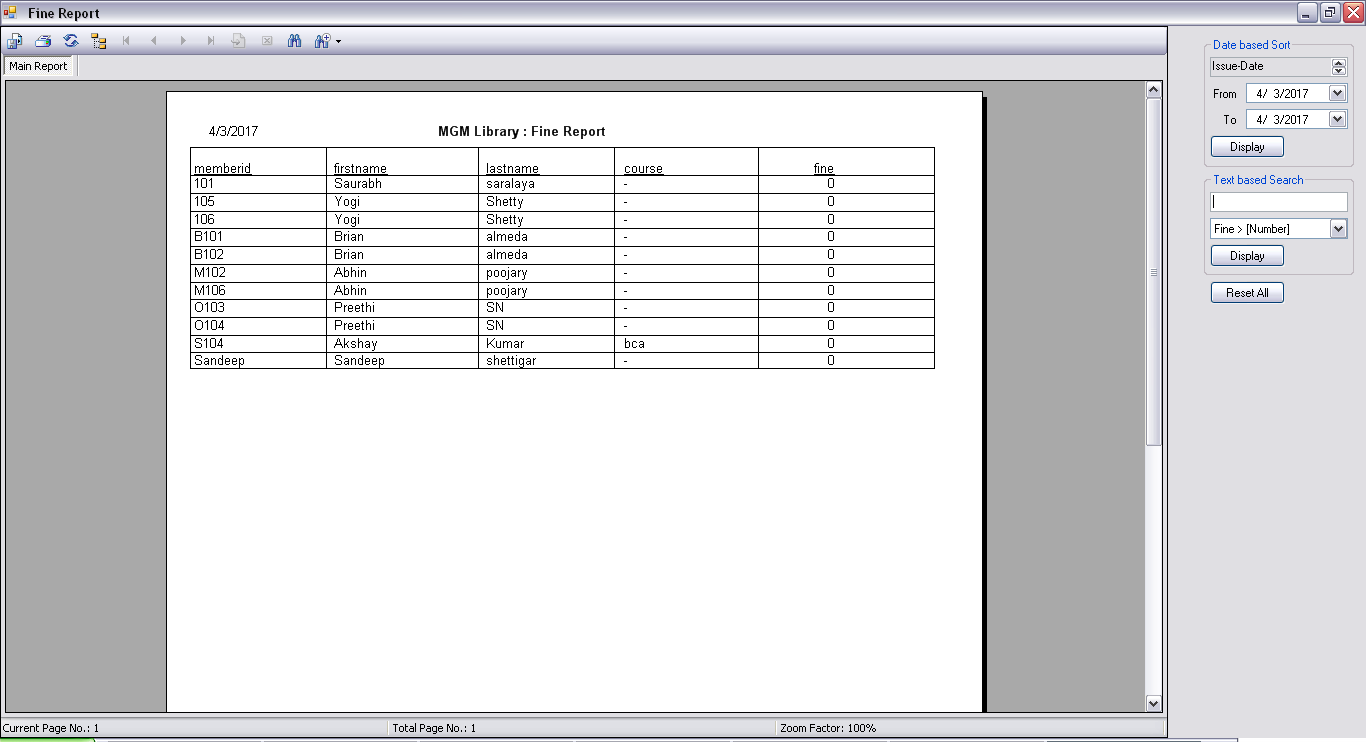
BookReport:



StockReport:



Fine Report:



5 TESTING:

Testing is process of detecting errors. Testing plays a very important role in quality assurance and for ensuring the reliability of the software. the result of testing is mainly used during maintenance.

**Psychology of testing:**

The main of testing of demonstrate that a program should not contain any error.the basic purpose of testing phase is to detect errors that may be presented in the program.hence one should not start testing with the intent to show that a program does not work testing is the process of executing a program with the intent of finding an error

**Testing objectives:**

The main objective of testing is to uncover the host of errors, systematically and with minimum efforts.

* A successful test is one that uncovers an error as yet undiscovered error.
* A good test case is one that has a high probability of finding error, if it exists.
* The tests are inadequate to detect possibility present errors.
* The software confirms the quality and reliability.

**System testing:**

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. the testing phase involves the testing of system using various test data; preparation of test data place a vital role in the system testing. After preparing the test data, the system is tested.

Those test data, errors were found and corrected by following testing steps and corrections are recorded for future references.thus a series testing is performed on the system before it is ready for implementation. the various types of testing on the system are:

* Unit testing
* Integration testing
* Validation testing
* User acceptance testing

**Unit testing:**

Unit testing is the smallest unit testing that focuses on verification. using the concept of detailed design, the process of specifications testing is carried out to uncover the errors within the unit testing before the integration test begins.

In this application developer test the programs as system. Software unit in a System are the modules and routine that are assembled and integrated to from a specific function.unit testing is first done on modules, independent of another to locate errors. This enables to detect error. though these error from the interaction between the modules are initially avoided.

**Integration testing:**

Integration testing is done after unit testing phase. The basic goal of this testing is to see if module are integrated properly, the emphasis being on testing interface between the modules. this testing activity can be considered as testing the design and hence the emphasis on testing module interaction. In this project integrating all the modules effects on the working of any services by giving different combination of inputs.

**Validation testing:**

At the culmination of the integration testing, the software was completely taken as a package. Entire interfacing error have been uncovered and corrected and a final series of software validation testing stage begins. Here the system will be tested in a manner that can be reasonably expected by the customer, the system was tested against the requirement specification.

**Output testing:**

After performing of the validation test, the next phase is output test of the system, since no system could be useful if it does not produce the desired output in desired format. By considering the format of the report / output/report is generated or displayed and is tested.

Here the output format is given in two different ways:

* On the screen
* As a print

**User acceptance testing:**

Acceptance testing is performed on the data of the client to demonstrate that software is working satisfactorily. Testing here is focused on external behavior of the system; not on the internal logic of the program. Test case should be selected so that the largest number of attributes of an equivalence class is exercised at once. the testing phase is an important part of the software developed. it is the process of finding and missing operation.

**PROGRAM TESTING**

Test cases should always be selected so that the largest number of attributes of an equivalence class is exercised at once. the testing phase is an important part of the developed software. It is the process of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied.

**TEST PLAN**

|  |  |
| --- | --- |
| **TEST CASE** | **TEST OBJECTIVES** |
| 1 | Test for user name and password |
| 2 | Test for changing the password |
| 3 | Test for adding Book Records |
| 4 | Test for modifying Book Records |
| 5 | Test for searching Book Records |
| 6 | Test for deleting Book Records |
| 7 | Test for adding Book Copy |
| 8 | Test for deleting Book Copy |
| 9 | Test for adding Staff Record |
| 10 | Test for deleting Staff Record |
| 11 | Test for modifying Staff Record |
| 12 | Test for adding Member Details |
| 13 | Test for modifying Member Details |
| 14 | Test for deleting Member Details |
| 15 | Test for searching Member Details |
| 16 | Test for Creating purchase Order |
| 17 | Test for Generating Purchase Bill |
| 18 | Test on Circulation : Issuing a Book |
| 19 | Test on Circulation : Renewing a Book |
| 20 | Test on Circulation : Returning a Book |

Test case 1

Objectives: Test for user name and password entry.

Test data: **Valid**: Valid user name and password to be entered in the program.

**Invalid**:use of invalid user name and password in the program

Output: **Valid**: Entering into the software.

**Invalid**: Displaying error message.

Result: **valid**: the user is allowed to enter the software.

**Invalid**: Display of error message which restrict the user to enter the Conclusion Both the valid and invalid result are tested. Output matches with the required result. Hence Test case is successful.

Test case 2

Objective: Test for changing the password

Test data: **Valid**: Valid Security Answer to be entered in the program.

**Invalid**: Security Answer is blank or use of incorrect Security Answer.

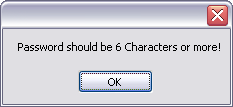
Output: **Valid:** Allows changing the password.

**Invalid**: prompting error message to the user.

Result: **Valid**: The password has been changed.

**Invalid**: the record is not updated to the database.

Conclusion: Both the valid and invalid results are tested.



Test Case: 3

Objectives Test for adding Book Records

Test data **valid**: All the required fields in the Book entry are entered.

**Invalid**: some of the fields are in incorrect format.pincode cannot be greater than 6 and contact number cannot exceed 12digit.

Output: **Valid**: Allow all the records to be added to the database.

**Invalid**: Displays the information message box to help the user know the problem

Result: **Valid**: The record will be saved in the database .

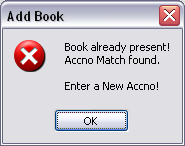
**Invalid**: The record will not be saved in the database.

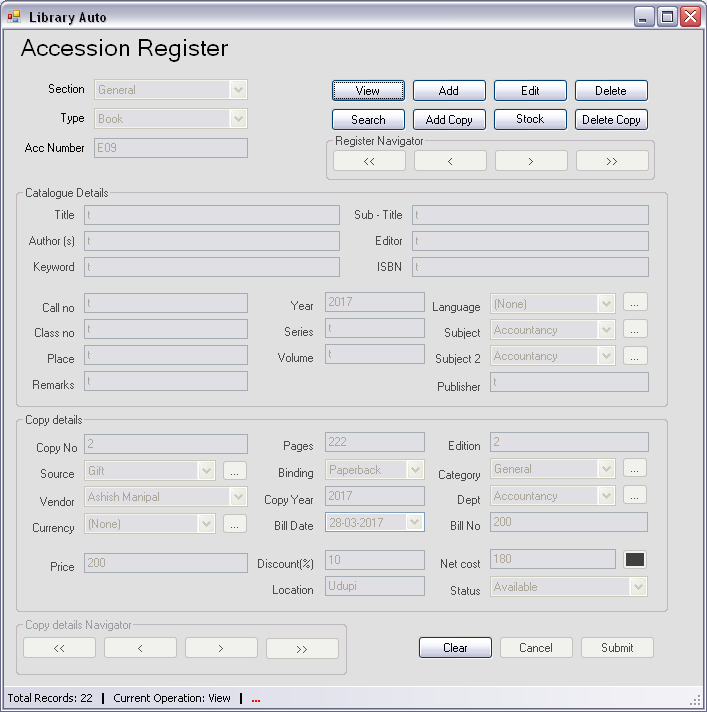
Conclusion: Both the valid and invalid result are tested. output matches with the desired result. Hence Test case is successful.









Test case: 4

Objectives Test for modifying Book Records

Test data: **Valid**: All the required fields in the Book entry are entered.

**Invalid**: some of the fields are in incorrect format. Numbers should be entered in contact and pin code fields. Only characters must be typed in name field.

Output: **Valid**: Allow all the records to be modified in the database.

**Invalid**: Display the information message box to help the user know the problem.

Result: **Valid**: The record has been updated to the database.

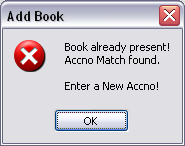
**Invalid**: The record is not updated to the database.

Conclusion: Both the valid and invalid results are tested output matches with the desired result. Hence Test case is successful.









Test case 5

Objectives Test for Searching Book Records

Test data valid: all the required fields in the Book entry are searched.

Invalid: some of the fields are in incorrect format and cannot be searched.

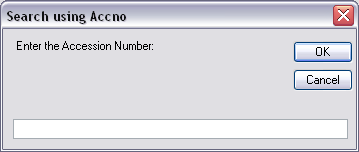
Output: Valid: Allows all the records to be searched in the database.

Invalid: Prompts error message to the user so as to know the problem.

Result: Valid: the record is searched.

Invalid: the record is not searched.

Conclusion: Both the valid and invalid result is tested. Output matches with the desired result .Hence test case is successful.





Test case 6

Objectives Test for deleting the Book records.

Test data Valid: the required fields in the Book entry are deleted.

Invalid: some of the fields are in incorrect format. once deleted records cannot be attempted to deleted again.

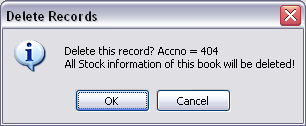
Output: Valid: allows the record to be deleted in the databse.

Invalid: prompts an error message to the user so as to know the problem.

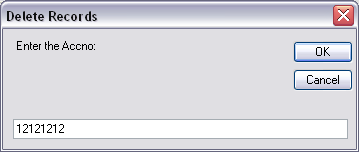
Result: Valid: The records is deleted.

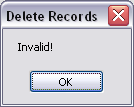
Invalid: The records is not deleted.

Conclusion Both the valid and invalid results are tested.









Test case 7

Objectives Test for adding Book Copy

Test data **valid**: All the required fields in the Book Copy are entered.

**Invalid**: some of the fields are in incorrect format.pincode cannot be greater than 6 and contact number cannot exceed 12digit.

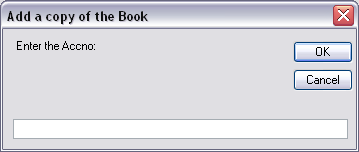
Output: **Valid**: Allow all the records to be added to the database.

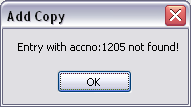
**Invalid**: Displays the information message box to help the user know the problem

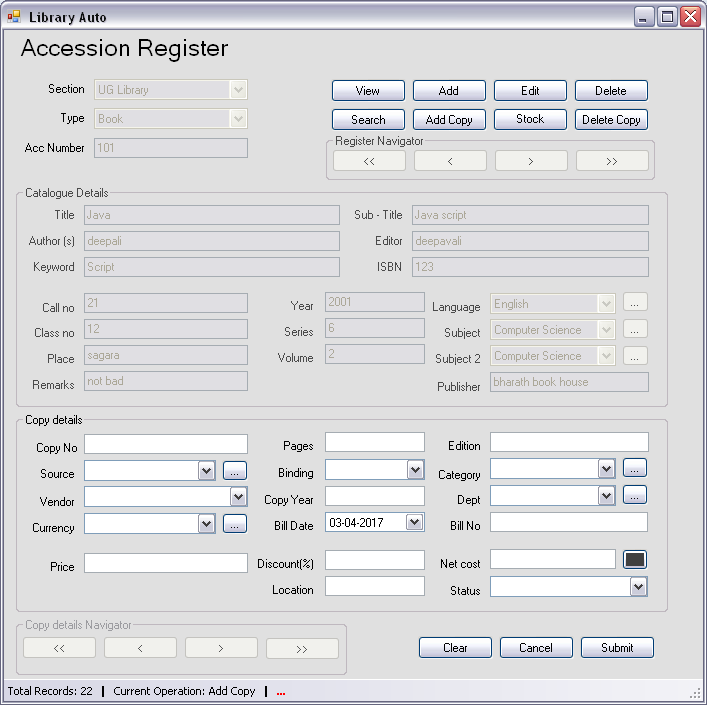
Result: **Valid**: The record will be saved in the database .

**Invalid**: The record will not be saved in the database.

Conclusion: Both the valid and invalid result are tested. output matches with the desired result. Hence Test case is successful.







Test case 8

Objectives Test for deleting Book Copy

Test data Valid: the required fields in the Book Copy are deleted.

Invalid: some of the fields are in incorrect format. once deleted records cannot be attempted to deleted again.

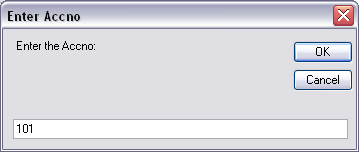
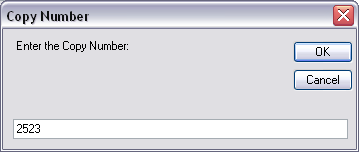
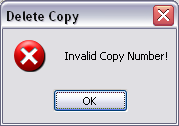
Output: Valid: allows the record to be deleted in the databse.

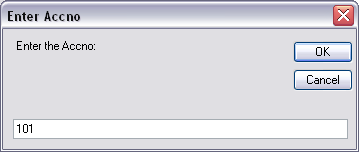
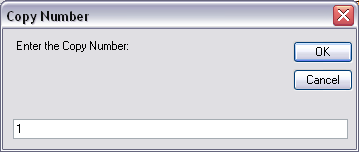
Invalid: prompts an error message to the user so as to know the problem.

Result: Valid: The records is deleted.

Invalid: The records is not deleted.

Conclusion Both the valid and invalid results are tested.

Test Case: 9

Objectives Test for adding Staff Record

Test data Valid: All the required fields for the Staff database are entered.

All the fields such as name, address, contact details and email-id are entered with specific validation

Invalid: some of the fields are in incorrect format. Once deleted records cannot be attempted to delete again.

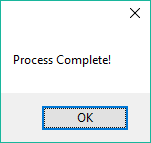
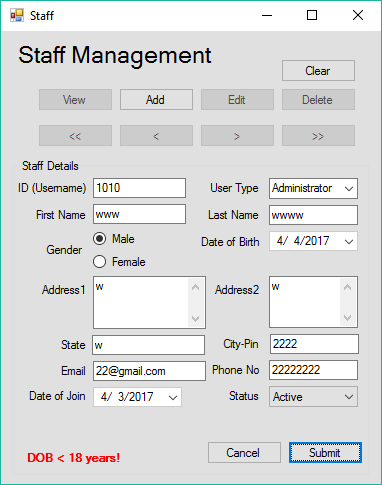
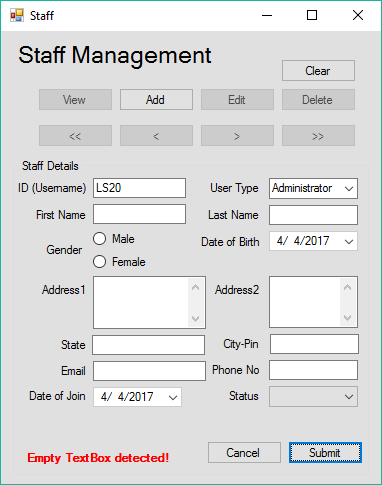
Output: Valid: Allow all the records to be inserted to the database.

Invalid: prompts an error message to the user so as to solve the problem.

Result:Valid: The record is inserted.

Invalid: The record is not inserted.

Conclusion: Both the valid and invalid result is tested. Output matches with the desired result. Hence Test case is successful.



Test case 10

Objectives Test for deleting Staff Record

Test data: Valid:Specific Staff ID given to be deleted.

Invalid: Incorrect Staff ID given to be deleted.

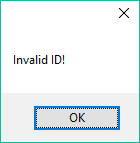
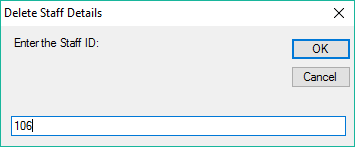
Output: Valid: Delete the specific staff record from the database.

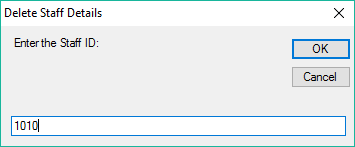
Invalid: Displays an error message to the user.

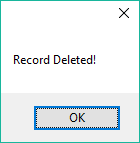
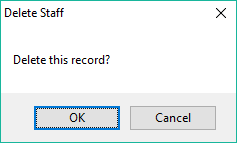
Result: Valid: The record is deleted.

Invalid: The record is not deleted.

Conclusion: Both the valid and invalid result is tested. Output matches with the desired result. Hence Test case is successful.







Test case 11

Objectives Test for editing the Staff Record

Test data: Valid: All the required fields in the Staff database are entered.

All the fields such as name,address,contact details and email-id can be edited.

Invalid: some of the fields are in incorrect format. After editing, the specific records can be updated.

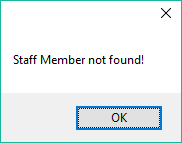
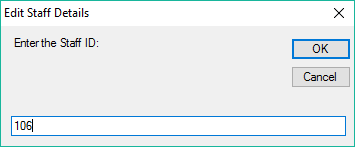
Output: Valid: Allow all the records to be inserted to the database.

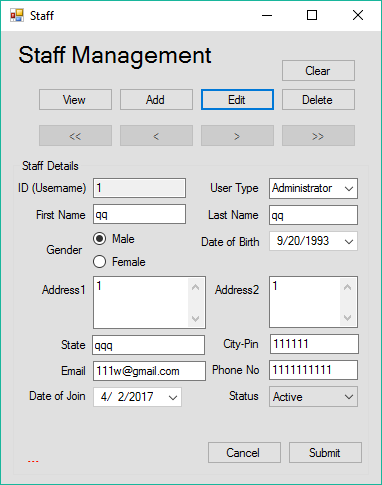
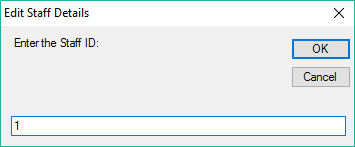
Invalid: Displays an error message to the user.

Result: Valid: The record is modified.

Invalid: The record is not modified.

Conclusion: Both the valid and invalid result is tested. Output matches with the desired result. Hence Test case is successful.





Test Case: 12

Objectives Test for adding Member Record

Test data Valid: All the required fields for the Member database are entered.

All the fields such as name, address, contact details and email-id are entered with specific validation

Invalid: some of the fields are in incorrect format. Once deleted records cannot be attempted to delete again.

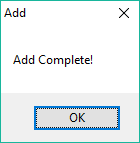
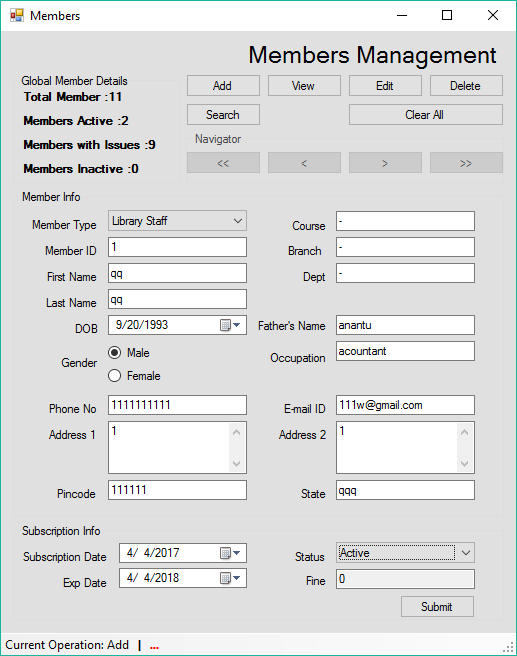
Output: Valid: Allow all the records to be inserted to the database.

Invalid: prompts an error message to the user so as to solve the problem.

Result:Valid: The record is inserted.

Invalid: The record is not inserted.

Conclusion: Both the valid and invalid result is tested. Output matches with the desired result. Hence Test case is successful.



Test case 13

Objectives Test for editing the Member Record

Test data: Valid: All the required fields in the Member database are entered.

All the fields such as name,address,contact details and email-id can be edited.

Invalid: some of the fields are in incorrect format. After editing, the specific records can be updated.

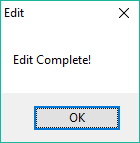
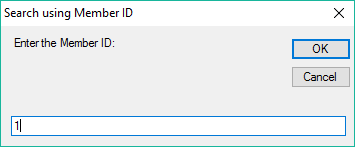
Output: Valid: Allow all the records to be inserted to the database.

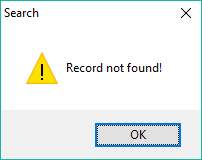
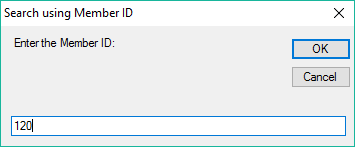
Invalid: Displays an error message to the user.

Result: Valid: The record is modified.

Invalid: The record is not modified.

Conclusion: Both the valid and invalid result is tested. Output matches with the desired result. Hence Test case is successful.





Test case 14

Objectives Test for deleting Member Record

Test data: Valid:Specific Member ID given to be deleted.

Invalid: Incorrect Member ID given to be deleted.

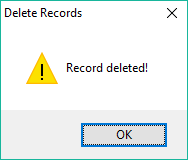
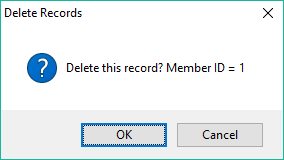
Output: Valid: Delete the specific staff record from the database.

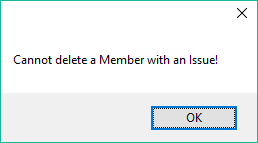
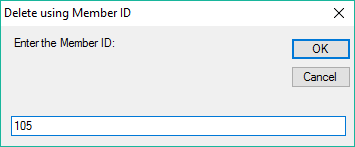
Invalid: Displays an error message to the user.

Result: Valid: The record is deleted.

Invalid: The record is not deleted.

Conclusion: Both the valid and invalid result is tested. Output matches with the desired result. Hence Test case is successful.





Test case 15

Objectives Test for searching Member Details

Test data Valid: all the required fields in the Member entry are searched.

Invalid: some of the fields are in incorrect format and cannot be searched.

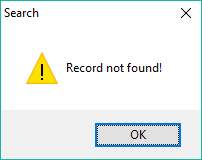
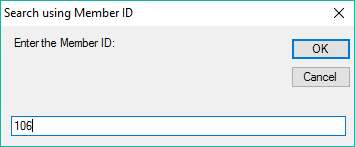
Output: Valid: Allows all the records to be searched in the database.

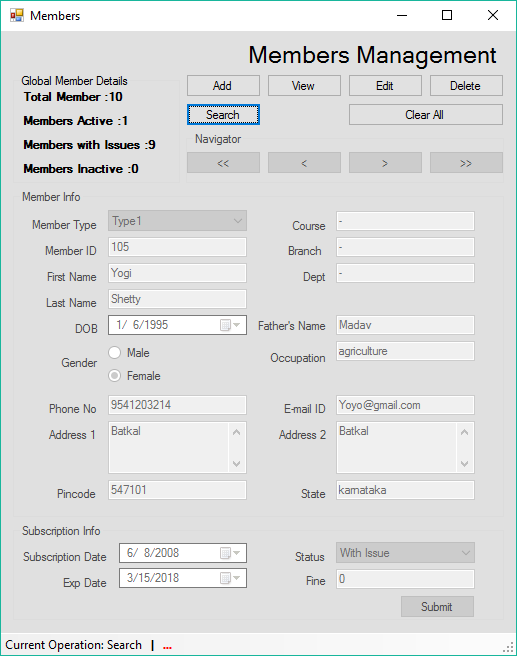
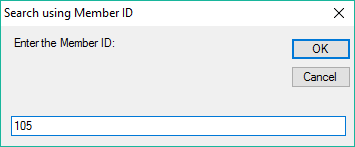
Invalid: Prompts error message to the user so as to know the problem.

Result: Valid: the record is searched.

Invalid: the record is not searched.

Conclusion: Both the valid and invalid result is tested. Output matches with the desired result .Hence test case is successful.





Test Case: 16

Objectives Test for Creating a Purchase Order

Test data Valid: The Vendor ID and Accno are Selected, number of Copies Entered.

Invalid: some of the fields are in incorrect format and cannot be added to the database.

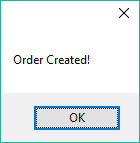
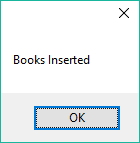
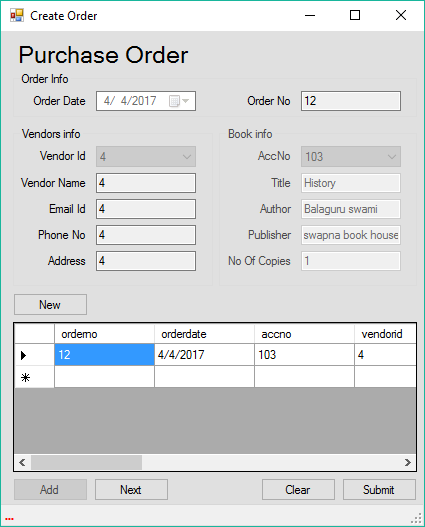
Output: Valid: Allows item which has to be added.

Invalid: prompts the user with the error message.

Result: Valid: The record is added to the database and Purchase Order is created.

Invalid: The record is not saved.

Conclusion: Both the valid and invalid result is tested. Output matches with the desired result. Hence Test case is successful.



Test Case: 17

Objectives Test for Generating Purchase Bill

Test data Valid: The Order No and Accno are Selected, price and discount Entered.

Invalid: some of the fields are in incorrect format and cannot be added to the database.

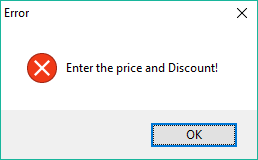
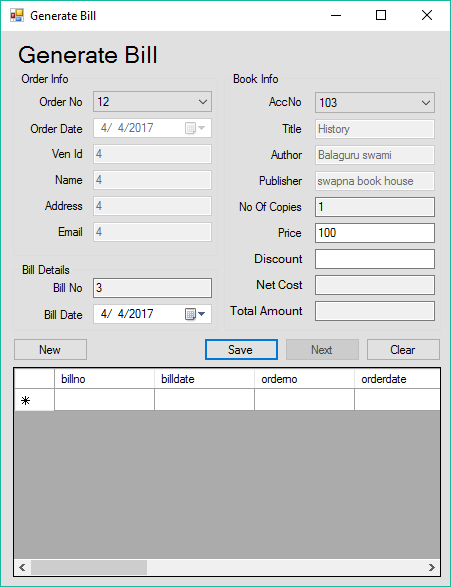
Output: Valid: Allows item which has to be added.

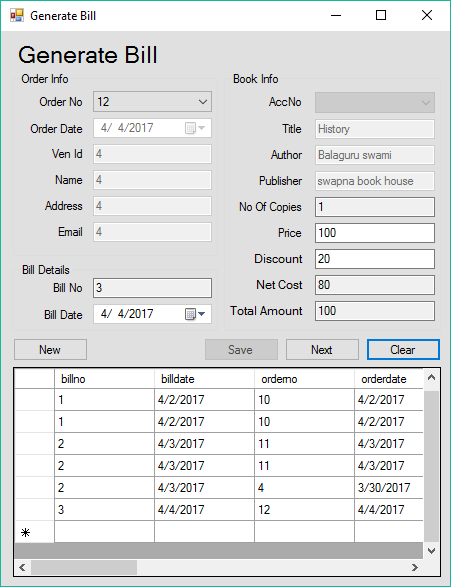
Invalid: prompts the user with the error message.

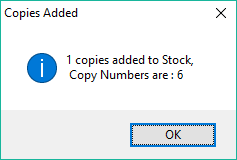
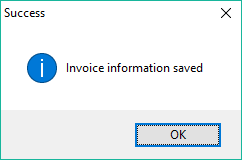
Result: Valid: The record is added to the database and Bill is Generated, Stock is updated.

Invalid: The record is not saved.

Conclusion: Both the valid and invalid result is tested. Output matches with the desired result. Hence Test case is successful.







Test Case: 18

Objectives Test on Circulation : Issuing a Book

Test data Valid: Valid Accno,Copyno and Member ID is entered.

Invalid: Incorrect Accno,Copyno and Member ID is entered or with Unavailable Status.

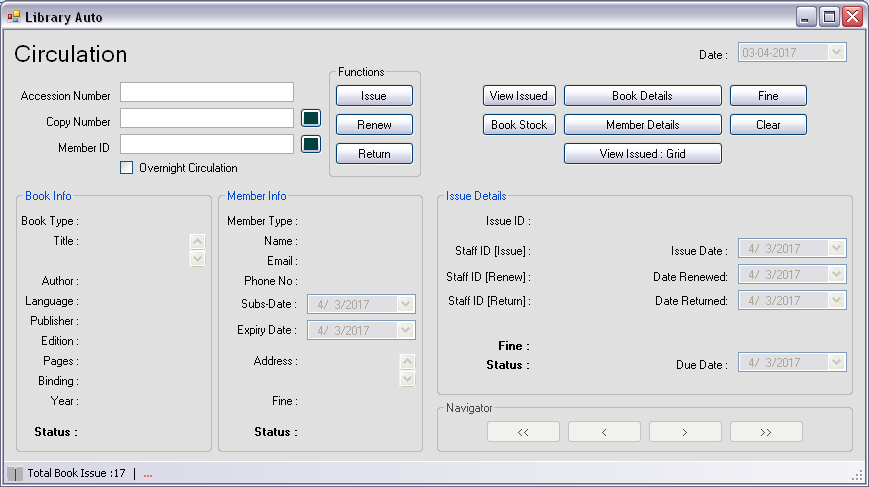
Output: Valid: Issues the book and generates the Issue ID.

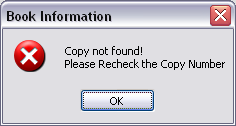
Invalid: prompts the user with the error message.

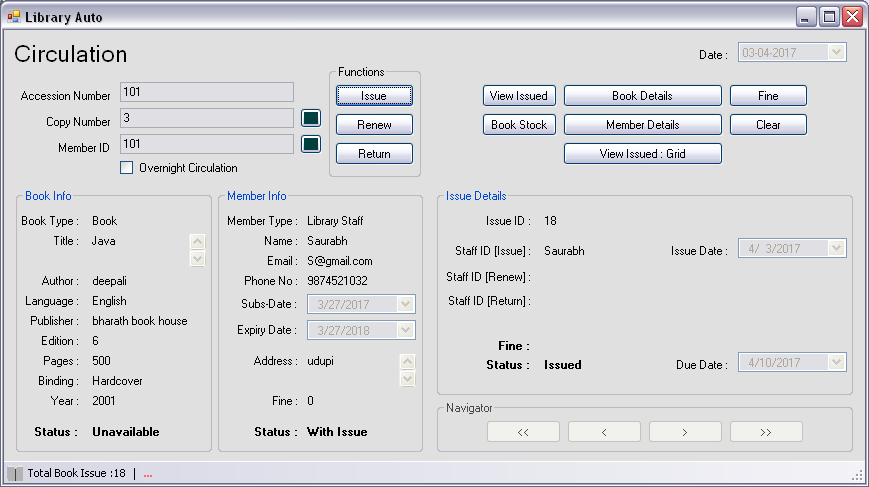
Result: Valid: The record is added to the database and Member and Stock Table is Updated

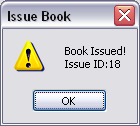
Invalid: The record is not saved.

Conclusion: Both the valid and invalid result is tested. Output matches with the desired result. Hence Test case is successful.









Test Case: 19

Objectives Test on Circulation : Renewing a Book

Test data Valid: Valid Issue ID is entered.

Invalid: Incorrect Issue ID entered or with Unavailable Status.

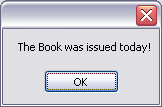
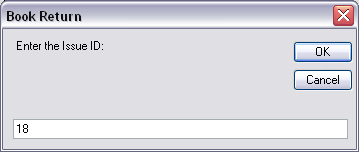
Output: Valid: Resets the Status and Date to renew the book.

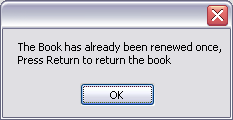
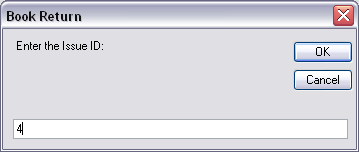
Invalid: prompts the user with the error message.

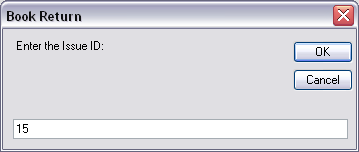
Result: Valid: The record is updated in the database

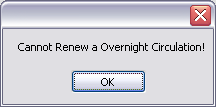
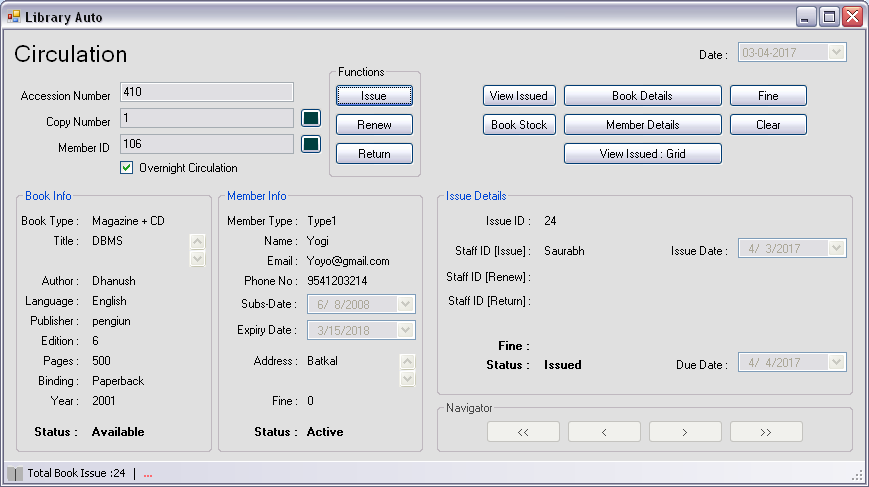
Invalid: The record is not saved.

Conclusion: Both the valid and invalid result is tested. Output matches with the desired result. Hence Test case is successful.









Test Case: 20

Objectives Test on Circulation : Returning a Book

Test data Valid: Valid Issue ID is entered.

Invalid: Incorrect Issue ID entered or with Returned Status.

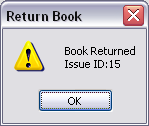
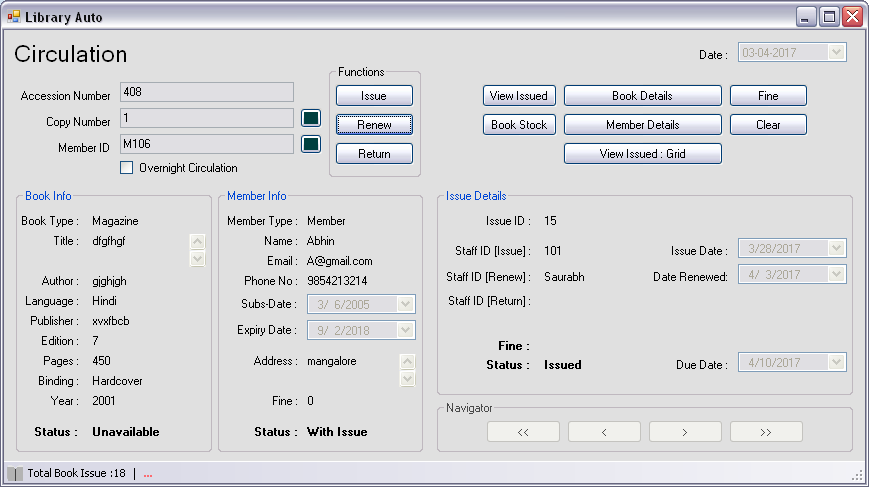
Output: Valid: Resets the Status and Date to Return the book.

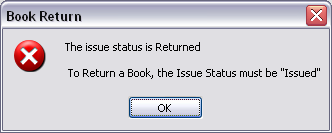
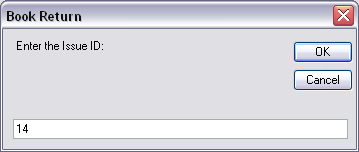
Invalid: prompts the user with the error message.

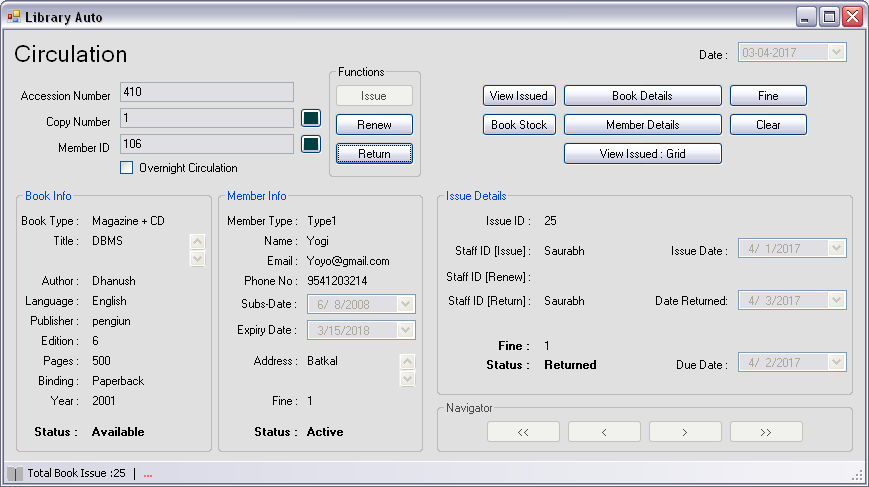
Result: Valid: The record is updated in the database

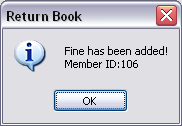
Invalid: The record is not saved.

Conclusion: Both the valid and invalid result is tested. Output matches with the desired result. Hence Test case is successful.









6 DEFINTIONS,ACRONYMS AND ABBREVATIONS

This software makes use of connection string for connecting the SQL server. Connection string defines the following constrains namely

* Data source
* Initial catalogue
* Integrated security

**Data source:** this defines the sources from where the data needs to be extracted.

**Initial catalogue:** this defines the name of the database to be used.

**Integrated security:** The database can be accessed using an authorized passwords and user name.

Following acronyms and abbreviations are being used in the proposed software:

SQl - Structured Query language.

C#- Visual C#.

DFD - Data Flow Diagram.

CFD - Context Flow Diagram.

ER - Entity Relationship

7 Bibliography

An integrated Approach to Software Engineering by Pankaj jalote.

Visual C# by Balaguru Swamy.

The Userbase of StackOverflow and MSDN.

//End