# ALEXANDRIA

## THE INTEGRATED LIBRARY SYSTEM


DEVELOPED, DESIGNED, DEBUGGED

BY

ABARAJITHAN G.

# Acknowledgement

This Project, although claimed to be developed, designed, debugged and documented by an individual, would never have succeeded without the direct and indirect help, support and encouragement offered by my parents, many of my teachers and friends. I'm greatly indebted to all of them and I'd like to mention the names of few of them here, as a minor attempt to express my gratefulness.

Firstly, I owe my gratitude to Mr. Janaka Madiwaka, the ICT teacher of St. Sylvester's College Kandy. Through the Visual Basic lessons in O/L ICT curriculum, he introduced us programming in an excellent manner which we'd never forget for the rest of our lives. He also recommended us to complete a diploma course in programming at ESOFT, Kandy during the vacation after the O/L exams. Then, I'd like to thank the teachers of ESOFT: Miss K. Janarthanee, Mr. Vajjira Kulathunga, and Mrs. Zeenath. Miss Janarthanee, the teacher in charge of Diploma in Software Engineering, offered her help and support without hesitation during the documentation part of the project, whenever I requested assistance through the phone. The basic programming techniques of Visual C# programming language and Visual Studio software were introduced to us by Mr. Vajjira. Mrs. Zeenath introduced us the SQL and Database management, which were whole new topics for me, in which I had no prior exposure.

I owe my gratitude to my parents, especially to my mother who offered me with access to an excellent environment with a powerful computer during the coding and designing part of the project. I am also thankful to the rest of my family members who has offered their indirect assistance in making this project successful.

I should also mention the websites and forums such as Stack Overflow, MSDN Social and W3Schools from which I could learn a whole lot of new programming techniques and get help and support for errors and bugs found in the software. I'm thankful to Google Image Search and countless web pages from which I could download numerous royalty free images which I combined into new designs and used as User interfaces of Alexandria.

Lastly, I'd like to thank the libraries: D.S. Senanayaka Public Library, Kandy and British Council Library, Kandy which helped me in the observational study of feasibility of prevailing manual library systems. Many of the services offered by British Council, such as email alerts to remind about returns, quick lending and receiving processes, facility to check availability of a book from a digital database inspired many of the special features I've built into Alexandria: The Integrated Library System.

# Contents

# Introducing ALEXANDRIA

Alexandria: The Integrated Library System was named in order to honor The Royal Library of Alexandria, Egypt, the largest library of the ancient world. Alexandria is a library automation system that manages a database of book catalogues, member details, lending details, member activity, administrator preferences and user accounts. It is built as an effective replacement for manual library information systems. It is designed specifically to function with minimum data input from the user by processing all the rest of the information with the aid of data in the database.

This application is developed for the use of public libraries which use an ineffective, slow, buggy manual information system of handling file folders, receipt bills, and book information cards. The librarians of public libraries can save time, labor and money by using automation systems like Alexandria.

# What's wrong with using Manual Information systems in libraries?

Almost all public libraries of Sri Lanka use manual information systems. Manual systems are slow, ineffective, require more human labor and cause inconvenience for both members and librarians. Presence of such drawbacks is quite evident when observing the manual information systems at work.

The libraries which use manual operation systems catalogue details of each book in cards which are then arranged in alphabetical order. A card containing the Title of the book and the Book ID is placed inside each book. Details of new memberships are taken down into forms which are manually arranged into files in alphabetical order. Each member is offered with few cards which should be submitted to the library when a book is borrowed. The librarian then takes the book card from inside the book, clip it together with Member's card and place it within the stacks of such cards arranged according to the member ID. When the book is returned, the librarian manually goes through piles of such cards and discovers the member's card after several minutes. Fines are calculated manually by manually checking the no. of days past the time allowed for borrowings. Then, a receipt bill is written with Book and Member details and given to the member requesting to pay the fine amount. A copy of the receipt is pinned to the collection of hundreds of such receipts and the fine amount is logged into an accounts book.

In my personal experience, a simple process of joining as a member in public libraries takes up to 30 minutes. In such a system, it'd take several minutes to retrieve details of members during renewal or during similar instances. Contact information of the member must be searched and taken from piles of files. To find which books he/she has borrowed, stacks of hundreds of cards must be manually scanned. In such a system, getting details about a book or checking whether a book is available or lent is practically impossible. Also, changing the fine amount, member expiry period, maximum time allowed per issue are very complicated tasks which may take weeks to be fully completed. In addition to that, manual systems are prone to random human errors. For example, a librarian misplacing a member's detail form may result in hours of searching, which is a waste of human labor.

Therefore, it is quite evident that manual information systems are prone to human errors, inconsistent, slow, require more labor and inefficient. Replacement of such systems with integrated library automation systems is an effective solution to several inconveniences caused by manual systems.

## Advantages of an Automation System

In the modern information era, with the widespread use of computers in almost every field, public libraries are in need to switch from such slow, inconsistent manual information systems into fast, efficient and more accurate integrated library systems in order to provide the members with more convenient, easy-to-use, quick service and to introduce new, innovative services.

When using an automation system, weary tasks which take long time to be completed such as searching for a book's or member's details and lending and receiving books can very quickly be done in less than few seconds. For example when using a manual system to receive a book, the librarian needs to scan through piles of members' cards, find the right card, check the date for fines, if there are fines, check the member type and calculate the fines manually, write a receipt bill with member and book details and request to pay the fines. This process takes at least 5 minutes of time. But when using an automated system like Alexandria, a librarian just needs to enter only the book ID and press a button, in order to receive a book. The system scans the database of hundreds of books, checks in background whether the book was lent or not, if lent, whether there any fines, if there are fines, calculates the fines and asks whether to receive the amount now or later, all in less than a fraction of a second. This example clearly shows the advantages of using automation systems over manual systems in libraries.

## Possible Drawbacks of the System ALEXANDRIA

Although automated systems are claimed to be fast, accurate and consistent, bugs and errors, which are inevitable when building such a large system, can exist in the software and may cause a corruption in the data which may result in inconvenience or in loss of a large sum of money for the library management. As mentioned below in the Testing section, a system with 87 pages of coding, developed and tested individually, has all chances for the existence of serious bugs and errors which were not discovered when testing individually. Therefore, if, in case, this project is to be implemented on a real library, it is advisable to use the system in parallel with the manual system for few weeks to find possible bugs, which can later be informed to the developer and can be fixed in the next stable release.

In addition to that, due to the short project duration, few more useful features which could be implemented have not been included in the system. If this project is to be used in real libraries, those and many other features can be included on user request in the next stable release.

# Functionalities of ALEXANDRIA

As a Library Automation System, Alexandria allows the librarians to perform the functions of a library's manual information system faster, easier and in a more advanced manner. Those basic functions include adding new books to catalogue, adding new members, viewing book & member details, removing books and members, lend books to members, receive books from members, calculate fines etc. These operations are explained in detail in the user guide (see pg. 31).

In addition to those basic operations, some more advanced and useful functionalities have been built into Alexandria: The Integrated Library System. Few of those notable features are listed below.

## 1. Preferences

Preferences are settings that are stored in the database and can be modified by an administrator user account. Some of them are:

- Fees to be charged from a new member
- Fees to be charged for a membership renewal
- Maximum no. of days a book is allowed to be borrowed (Book Time)
- Fines per day for a book
- No. of months in which a membership expires
- Maximum no. of books allowed for a member
- No. of days of lending after which an alert should be raised

These settings can be set separately for an adult member and a child member right from the software (from preferences window) by an administrator. The minimum age for an adult membership can also bet set.

## 2. User Accounts

User Accounts can be created with a username and password combination under one of the two user types: administrator and librarian. These accounts can be created, modified or deleted from the preferences window by an administrator. The credentials for these accounts are stored in the database and the database is secured with a master password [masterpwd], which is not needed for a user to use the application.

The application can be opened in three modes. A member can login without a username and password, but can only access the Check-In-Out and BookFinder windows. Librarians and administrators are required to provide their credentials to login. A librarian can manage Books, Members and Book Issues only, while an administrator has access to all of the windows, especially windows for transaction details, preferences, member logbook and statistics.

## 3. Alert Notifications

Notifications are issues that have not been returned past the alert time. (Alert time is usually set longer than the Book Time. By default, Book Time is set as 30 days and alert time is set as 90 days for an adult membership) Members may have forgotten to return these issues or these books may have been stolen. When the application is launched, the database is scanned for such issues and if such records have been found, a table filled with basic details of the book and member is shown to a librarian or administrator. Details of a member, book or title can be viewed by clicking respective cells in the table. The member can be reminded about the issue through phone or email by checking the contact information, or else the book can be removed and member can be added to blacklist with a few clicks from the window itself.

## 4. Blacklist (Blocked Members)

Members who are suspected for a book theft or other security reasons can be added to blacklist and blocked. Whenever these members try to use the library (check-in, borrow, return or extend a book) an alert is raised and the librarian is notified to take necessary actions.

In addition to that, when such members try to apply for a new membership, the database is scanned for an NIC or email ID match and an alert is raised. Also, when somebody registers with same address, phone no. or work address the librarian is notified and the librarian may inquire the new member about the whereabouts of the blocked member.

## 5. Statistics

In addition to general statistics, the most active members and most popular titles, authors and genres are also listed in the statistics window. This may be less useful for traditional libraries, but innovative libraries may award the top members and encourage them and consider the trending statistics when choosing new books for the library.

Other minor features are:

- Account Fines: Storing unpaid book fines to members account (see pg. 34)
- Member Check In, Check Out.
- Ability to add several books under one title
- Allow members to report a lost book and pay certain times the price of the book.

Only the special functionalities have been briefly explained above. All the functions of the system, both basic and advanced functions have been explained in detail with screenshots, in the user guide attached at the end of the documentation.

# Project Plan

Project Duration: 20 days (8th to 27th August 2012)

| Steps of Development | Duration |
|---|---|
| **Requirement Specification** | Few days |
| **System Design** | 2 days |
| **Implementation (Coding)** | 12 days |
| **Testing** | 2 days |
| **User Interface Design** | 7 days |
| Documentation | 3 days |

| | | |
|---|---|---|
| **Software Requirements** | Microsoft Visual Studio 2010 | To Code the Application in C# |
| | Microsoft Access 2010 | To Design the Oledb Database |
| | Adobe Photoshop | To Design User Interface |
| | yEd Graph Editor | To Create Flowcharts |
| | Microsoft Word 2010 | For Documentation |
| | Fonts Used: Trebuchet MS, Georgia, Book Antiqua, Palatino Linotype | |
| **Network Requirements** | Internet Access | For help regarding advanced C# |

# Development Process of Project ALEXANDRIA

To submit as the final project of *Diploma in Software Engineering* Course (January – August 2012) done in ESOFT, Kandy, we, students were asked to select an automation system, to build it and to submit the project with necessary documents. Since I was busy with my first year A/L lessons and term exams at the school, I had to start doing the project during the school vacation in August, individually and well behind the other project teams of our class. When starting the project, I had a difficulty in choosing an automation system, which took more than three months! Since I was not much familiar with business systems such as systems for shops, I couldn't have completed such projects successfully. I once considered an automation system for Chemical Laboratories at schools, but I found it too easy, since it was not a challenge to build such a system.

Therefore, finally, I had to select a Library System, which was very familiar to me, a member in both D.S. Senanayaka Memorial Public Library of Kandy, which uses a manual information system and in British Council Library, Kandy where a very advanced automated library system is used. Many of the ideas I've implemented in Alexandria, such as BookFinder, Notifications, the simplicity in lending and receiving processes etc. are already being used effectively in British Council Library. However, a library system is always considered as a primitive system which is easy to develop and to implement. But on the other hand, building an automated system for lending libraries requires all the knowledge gained and programming techniques learnt in the Diploma course. Therefore, in order to make my project more complicated and impressive, I had to add few complicated functionalities (see pg. 7) which occasionally required knowledge well beyond the C# and SQL lessons taught in the course.

Considering the technical details of the project, this application was built using Visual C# programming language and Microsoft access databases. The project was started on 8$^{th}$ of August and completed on 30$^{th}$ of August 2012. Firstly, the features the system should possess were listed. Then, databases were designed and flowcharts were drawn (see: System Design, pg. 12) for each activity in the program. The coding process, in which 87 pages of coding and necessary comments to understand the coding were typed to make the software work according to the flowchart blueprints, took more than two weeks. Then, user interfaces for each window were designed using appropriate software. Finally, the system was tested for possible bugs and then the project was documented and submitted.

# System Development Life Cycle

## 1. Requirement Specifications

The inconsistencies of the manual information systems and inconveniences caused by the use of them are explained in detail in the '*What's wrong with using Manual Information systems in libraries?*' section above. Since I had a chance of using services of two libraries: one with a manual system and the other with an automated system, I was able to work on my study of feasibility of the prevailing manual system with ease. Comparing in contrast my personal experiences in both libraries helped me to list out the inconveniences in using manual system and advantages of using an automated system. As I mentioned earlier, in a public library, joining as a new member took about half an hour, returning a book with fines takes at least five minutes and as a member, it was impossible to get details about availability of a book. On the other hand, in a library that uses an automated system, a new member can join in less than five minutes, a book with fines can be returned in less than few seconds with more options to pay the fines and a member can check for a book's availability by logging in as a member in the library's online database. These details were taken into consideration when building Alexandria: The Integrated Library System.

# 3. Implementation (Coding)

Coding was the hardest process in the system development life cycle which took about two weeks of day and night work. 87 pages of coding were typed into the MS Visual Studio 2010 with the aid of IntelliSense and Error list technologies of the software. While developing this project, I had to use several programming techniques which were not in the syllabus of the diploma course. Some of them are:

- Calling public methods, passing values into and getting values out of them.
- Workarounds for SQL commands which are not allowed in Access [COUNT (Distinct..)]
- Use of public variables to store preferences.
- Designing datatables programically and using DataGridViews to display data from database.
- goto statements which helped to get rid of several complicated nested if commands.
- The following workaround to eliminate single and double quotes in inputs, which caused a crash.

```
foreach (Control c in this.Controls)
{
if (c is TextBox || c is ComboBox)
{ c.Text = c.Text.Replace("'", ""); c.Text = c.Text.Replace("\"", ""); }
}
```

- 'If data connection is open…' clauses to overcome data connection closing and opening errors.

A small piece of coding which is executed when a new book is added is presented in the next page. The complete coding of the system is presented as a PDF document in the project disk and it can also be opened from the software's menu bar.

A piece of coding typed into the Add new books form.

```
//Add Values to Relation: Title

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

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

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

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

            // Adding books in a loop.

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

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

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

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

            Maxbook = ""; BookID = "";


            // Adding Transaction details

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

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

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

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

# 4. Testing Process

This software was tested in a three step process for possible errors and bugs in the application. Often it was interesting when trying to understand these errors and bugs and to figure out a way to solve them, which was similar to solving a puzzle. However, on the other hand, when struggling with bugs for several hours, debugging was a frustrating process that slowed down the development of the project. The process of testing I used and few of these interesting and frustrating errors I came across are explained in this section.

Firstly, as soon as each piece of coding were successfully written at Visual Studio without any syntax errors, that piece of coding was tested by quickly comparing some test data and the output produced. New types of coding were first tested on a Test form (which is then deleted) and then were brought into the main coding.

While writing the code, I met with several frustrating SQL errors which are very hard to be understood and to be solved. For example, when writing a really long SQL command to add members to database, an SQL exception was raised again and again. I had to struggle with it more than three hours to discover that referring to a column named 'Work' was the problem! I never understood why, but just changed the column name to 'MWork' to make it work. Once again when working on an SQL query as COUNT (DISTINCT…), though the query was perfect, an exception was raised. Later, from C# support websites, I learnt that MS Access does not support COUNT (DISTINCT…) queries which are supported in MYSQL, on which I was taught the SQL.

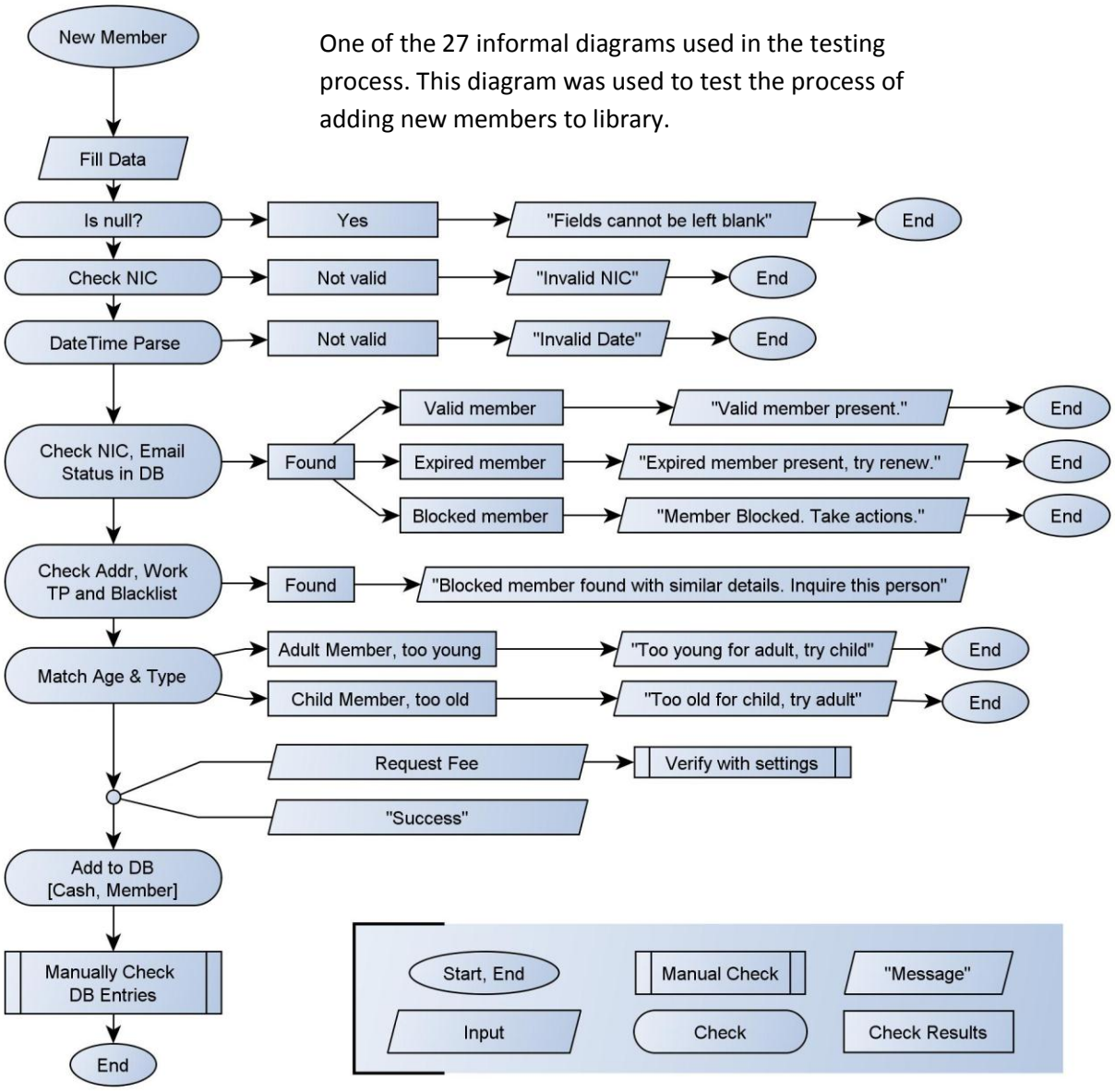
After completing the coding session, for the second step of the testing process, I initially used a set of informal diagrams (a sample is shown in the next page) to test Alexandria for possible application errors and bugs. These informal diagrams were drawn similar to flowcharts and displayed each and every possible combination of user inputs and the messages to be shown at each instance. Later when an action is successfully completed, database was checked for entries and made sure the system processed the user inputs in a right manner and each piece of correct and corrected coding was ticked away.

Third step testing was unintentionally done after completing the design process, when entering some decent data into the system in order to submit with the software. Few interesting, but very dangerous bugs and errors were discovered here for my surprise. Some of them are explained below.

When trying to pay the book fines of 40/- in the receive form, the software cleared all the unpaid account fines of the Member which was more than Rs. 2000! Then, when adding a new book under the title of "Aesop's Fables" and when adding a member with school's name: "St. Sylvester's College, Kandy" the system quickly crashed with an SQL error. I then realized that it

was due to the single quotes in the text and I had to include a command to remove single and double quotes from textboxes, rich text and combo boxes whenever an action is performed! Then when I emptied the database and started adding new books to it, I met an error with the method which was called to calculate the new Book ID from the highest Book ID present in the database. These bugs were then fixed with more coding or with minor alterations in the coding.

Therefore, time to time, countless bugs and errors were found and were fixed in the application. Since this project was being developed individually, I had to do the testing process myself, which was less effective and a time consuming task. I believe that there could be many bugs and errors still left in the program which were, unfortunately, not yet found and fixed. If, in case, this project is to be implemented on a real library, with a few days of practical testing and feedback from the users, more bugs can be eliminated.




One of the 27 informal diagrams used in the testing process. This diagram was used to test the process of adding new members to library.