DELHI TECHNOLOGICAL UNIVERSITY

SOFTWARE ENGINEERING (CO 301)

PRACTICAL FILE

Submitted to:

Dr. Rohit Beniwal
Assistant Professor
Department of COE

Submitted by:

Kunal Sinha
2K17/CO/A3/164

INDEX

S.No	EXPERIMENT	DATE	SIGNATURE
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			

Program 1

OBJECTIVE:

Write a program to count lines of code excluding blank lines and comments.

DESCRIPTION:

Lines of code is a quantitative measure of the size of a software program. It can be used to estimate the amount of development effort behind the software project. Calculation of lines of code is not as simple as just counting the number of lines in a software project. Code also includes comments as well as blank lines for whitespace to enhance readability of the code. These factors have to be taken into account while calculating the LOC of a software project.

PROGRAM:

```
# include <iostream>
# include <fstream>
# include <string>

using namespace std;

void get_lines_of_code(string);
bool is_comment(string);
bool is_blank(string);

int main() {
	cout << "Enter file name:" << endl;
	string filename;
	cin >> filename;
	get_lines_of_code(filename);
	return 0;
}
void get_lines_of_code(string filename) {
	ifstream file(filename);
	string line;
	int comments = 0;
	int blanks = 0;
	int count = 0;
	while (getline(file, line)) {
	if (is_comment(line)) {
	comments++;
	} else if (is_blank(line)) {
	blanks++;
	}
	count++;
```

```
        }
        file.close();
        int lines = count - comments - blanks;
        cout << "LOC: " << lines << endl;
        cout << "Comments: " << comments << endl;
        cout << "Blanks: " << blanks << endl;
}
// Checks if the given line is a comment
bool is_comment(string line) {
        if (line.length() < 2)
        return false;

        if (line.at(0) == '/' && line.at(1) == '/')
        return true;
        return false;
}
// Checks if the given line is an empty line
bool is_blank(string line) {
        return line.length() == 0;
}
```

OUTPUT:

```
kshitijchauhan ~ > Labs > Software Engineering ./loc.out
Enter file name:
lines_of_code.cpp
LOC: 44
Comments: 2
Blanks: 8
```

FINDIND AND LEARNINGS:

Estimation of the size of software is an essential part of Software Project Management. It helps the project manager to further predict the effort and time which will be needed to build the project. Lines of Code is one such measure.

Advantages:

- Universally accepted and is used in many models like COCOMO.
- Estimation is closer to developer's perspective.
- Simple to use.

Disadvantages:

- Different programming languages contains different number of lines.
- No proper industry standard exists for this technique.
- It is difficult to estimate the size using this technique in early stages of the project.

Program 2

OBJECTIVE:

Write a program implementing the COCOMO (Constructive Cost Model) model in C++.

DESCRIPTION:

The Constructive Cost Model (COCOMO) is a procedural software cost estimation model developed by Barry W. Boehm. The model parameters are derived from fitting a regression formula using data from historical projects (61 projects for COCOMO 81 and 163 projects for COCOMO II).

COCOMO is used to estimate size, effort and duration based on the cost of the software.
Basic COCOMO computes software development effort (and cost) as a function of program size. Program size is expressed in estimated thousands of source lines of code (SLOC, KLOC).

COCOMO applies to three classes of software projects:

- Organic projects - "small" teams with "good" experience working with "less than rigid" requirements
- Semi-detached projects - "medium" teams with mixed experience working with a mix of rigid and less than rigid requirements
- Embedded projects - developed within a set of "tight" constraints. It is also combination of organic and semi-detached projects. (hardware, software, operational)

Software Project	Ab	Bb	Cb	Db
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Basic COMOMO Model (Table)

The basic equation of COCOMO takes the form:

$$E = a_b KLOC^{bb}$$

$$D = CbEbb$$

where E is the effort applied in person-months, D is the development time in chronological months and KLOC is the estimated number of delivered lines of code for the project (express in thousands). The coefficients ab and cb and the exponents bb and db are given in Table 1.

The basic model is extended to consider a set of "cost driver attributes" [BOE81] that can be grouped into four major categories:

1. Product attributes
    a. required software reliability
    b. size of application data base
    c. complexity of the product
2. Hardware attributes
    a. run-time performance constraints
    b. memory constraints

c. volatility of the virtual machine environment
d. required turnaround time

3. Personnel attributes
a. analyst capability
b. software engineer capability
Applications experience
d. virtual machine experience
e. programming language experience

4. Project attributes
a. use of software tools
b. application of software engineering methods
c. required development schedule

PROGRAM:

```
#include <iostream>
#include <cmath>
using namespace std;
float COCOMO[][4] = {
                                2.4, 1.05, 2.5, 0.38,
                                3.0, 1.12, 2.5, 0.35,
                                3.6, 1.202, 2.5,0.32
                              };
int main() {
float loc;
int type = 0;
float E, D;
cout<<"Enter lines of PROGRAM in 1000s: ";
cin>> loc;
if(loc >50)
type = 1;
if(loc >300)
type = 2;

   E = COCOMO[type][0] * pow(loc, COCOMO[type][1]);
   D = COCOMO[type][2] * pow(E, COCOMO[type][3]);

cout<<"\nProject is ";
switch(type) {
case0: {
          cout<<"Organic";
          break;
       }
case1: {
          cout<<"Semi detached";
          break;
       }
case2: {
          cout<<"Embedded";
          break;
       }    }
 std::cout<<"\nE = "<< E <<"\nD = "<< D << std::endl;
```

```
return0;
}
```

OUTPUT:

```
$ ./COCOMO.exe
Enter lines of code in 1000s: 1200

Project is Embedded
E = 17836.9
D = 57.3274
```

FINDING AND LEARNING:

We implemented COCOMO model by taking the number of lines of code as an input and producing the metrics. COCOMO is well defined, and because it doesn't rely upon proprietary estimation algorithms, it offers these advantages to its users: estimates are more objective and repeatable than estimates made by methods relying on proprietary models and it can be calibrated to reflect your software development environment, and to produce more accurate estimates.

Program 3

OBJECTIVE:

Write a program to implement function point method.

DESCRIPTION:

A function point is a unit of measurement used to express the amount of business functionality an information system provides to a user. The values of function units i.e. User Inputs, User Outputs, User Enquiries, User Files, External Interfaces are taken as an input from the user.

Unadjusted Function Point is calculated by multiplying respective weights with inputs taken. Then accept the ratings(0-5) for 14 factors and calculate the value of CAF (Complexity Adjustment Factor). The function point is then calculated using the values of UFP and CAF.

PROGRAM :

```
#include<iostream>
using namespace std;
int fround(float x)
{
   int a;
   x=x+0.5;     a=x;     return(a);
}
 int main()
{
     int weights[5]={4,5,4,10,7};
     int UFP=0,F=0,rating,i,j;
     char functionunits[][30]={"UserInputs","UserOutputs",
                               "UserEnquiries","UserFiles","External Interfaces"};
     int input[5];
    float FP,CAF;
    for(i=0;i<5;i++)
          {
                cout<<"Enter number of "<<functionunits[i]<<endl;
                cin>>input[i];
          }
   for(i=0;i<5;i++)
          {
                UFP=UFP+(input[i]*weights[i]);
          }
          cout<<"Unadjusted Function Point(UFP) ="<<UFP;
          cout<<"Enter Rating of 14 factors on the scale of 0-5:"<<endl;
          cout<<" 0 - No Influence \n 1 – Incidental \n 2 - Moderate"<<endl;
          cout<<" 3 – Average \n 4 – Significant \n 5 - Essential"<<endl;
 for(i=0;i<14;i++)
   {
          cin>>rating;
           F=F+rating;
   }
   CAF=0.65+0.01*F;
```

```
  FP=UFP*CAF;
  cout<<"Adjusted Function Point"<<FP<<endl;
  return 0;
}
```

OUTPUT :

```
Enter number of User Inputs
4
Enter number of User Outputs
5
Enter number of User Enquiries
4
Enter number of User Files
6
Enter number of External Interfaces
6
Unadjusted Function Point(UFP) =159Enter Rating of 14 factors on the scale of 0-5:
 0 - No Influence 1 - Incidental2 - Moderate 3 - Average 4 - Significant 5 - Essential
3
3
5
4
2
1
3
2
4
5
3
4
5
2
Adjusted Function Point176.49
```

FINDING AND LEARNING:

A Function Point (FP) is a unit of measurement to express the amount of business functionality, an information system (as a product) provides to a user. FPs measure software size.
They are widely accepted as an industry standard for functional sizing.

Program 4A

OBJECTIVE:

Write the problem statement of Library Management System.

PROBLEM STATEMENT:

The Library management system is basically updating the manual library system into an internet-based application so that the users can know the details of their accounts, availability of books and maximum limit for borrowing and librarian can manage the library by entering the record of books available in library. Librarian can also retrieve the details of book available in library and can issue the books to the student/staff , maintains their records and maintain the late fine of the student who returns the diagram after due date. Student can login the system and check how many books are issued to them and stock available in library.

Earlier the library management is done manually by the librarian. where all the transaction of books is done manually. So making more time for a transaction like borrowing a book or returning a book and for searching of members and books. Another major disadvantage is that to preparing the list of books borrowed and the available books in the library will take more time. Currently it is doing as a one day process for verifying all records. So. we will be converting this manual system to automatic management system to ease the management of data.

The project is specifically designed for the use of librarians and library users. The product will work as a complete user interface for library management process and library usage from ordinary users. Library management system can he used by an existing or new library to manage its books and book borrowing, insertion and monitoring. It is especially useful for any educational institute where modifications in the content can be done easily according to requirements.

In automated, library management system. user can add members add books. search members, search books, update information, edit information. borrow and return books in quick time

The basic operations the system should perform:

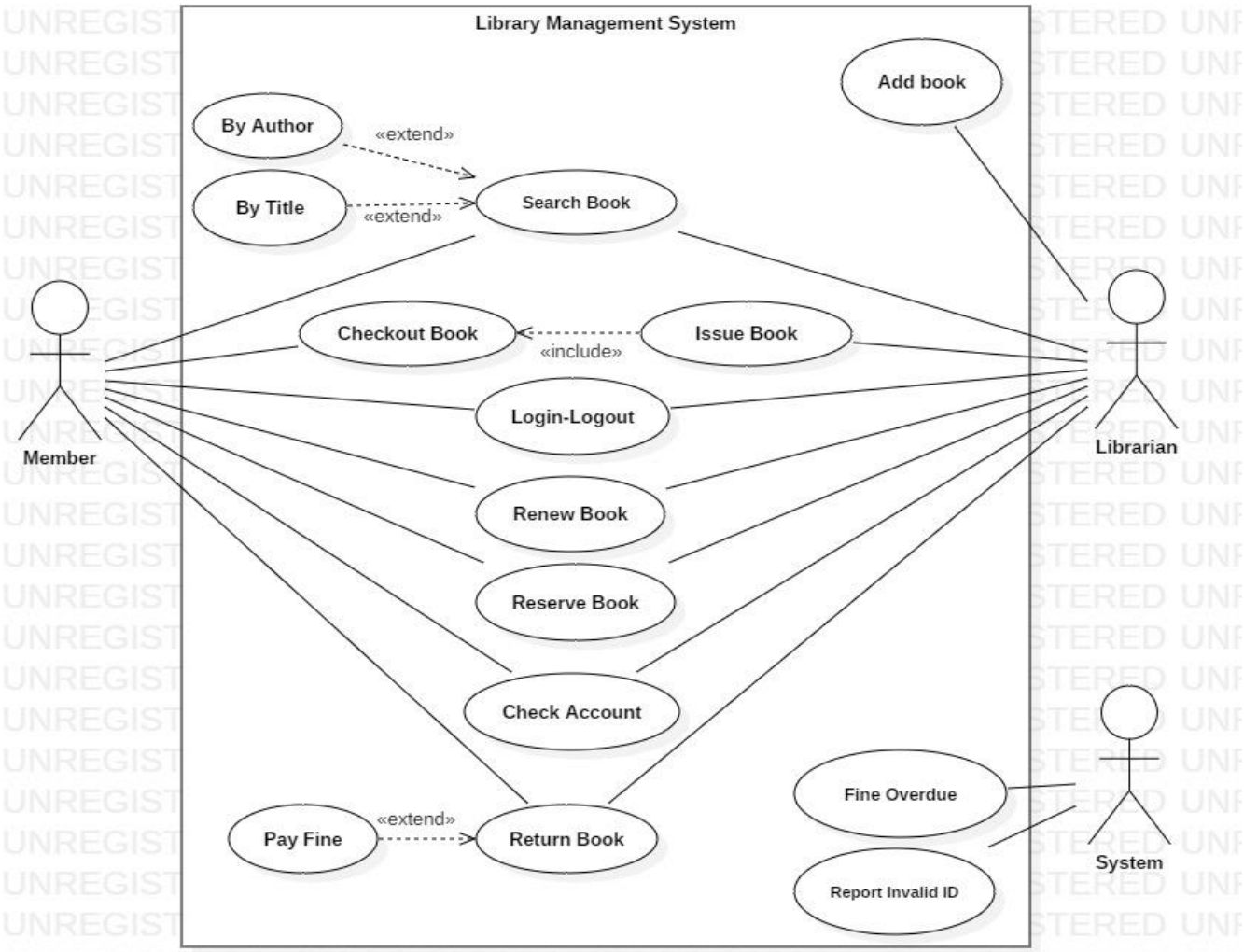
- Login
- Renew Book
- Search for Book
- Check Out Book
- Reserve Book
- Add Book
- Return Book
- Check Account
- Report invalid ID's
- Fine payment

Program 5A

OBJECTIVE:

Design the use case diagram of Library Management System.

USE CASE DIAGRAM:



Program 6A

OBJECTIVE:

Write use case description of Library Management System.

USE CASE DESCRIPTION:

We have three main actors in our system:

- **Librarian:** Mainly responsible for adding and modifying books, book items, and users. The Librarian can also issue, reserve, and return book items.
- **Member:** All members can search the catalog, as well as check-out, reserve, renew, and return a book.
- **System:** Mainly responsible for sending notifications for overdue books, canceled reservations, etc.

Here are the top use cases of the Library Management System:

- **Add/Remove/Edit book:** To add, remove or modify a book or book item.
- **Search catalog:** To search books by title, author, subject or publication date.
- **Register new account/cancel membership:** To add a new member or cancel the membership of an existing member.
- **Issue book:** To issue a book to a member.
- **Login-Logout**: To login into the system and logout from the system.
- **Report Invalid ID:** To authenticate a user.
- **Check-out book:** To borrow a book from the library.
- **Reserve book:** To reserve a book which is not currently available.
- **Renew a book:** To reborrow an already checked-out book.
- **Return a book:** To return a book to the library which was issued to a member.

Here are the main classes of our Library Management System:

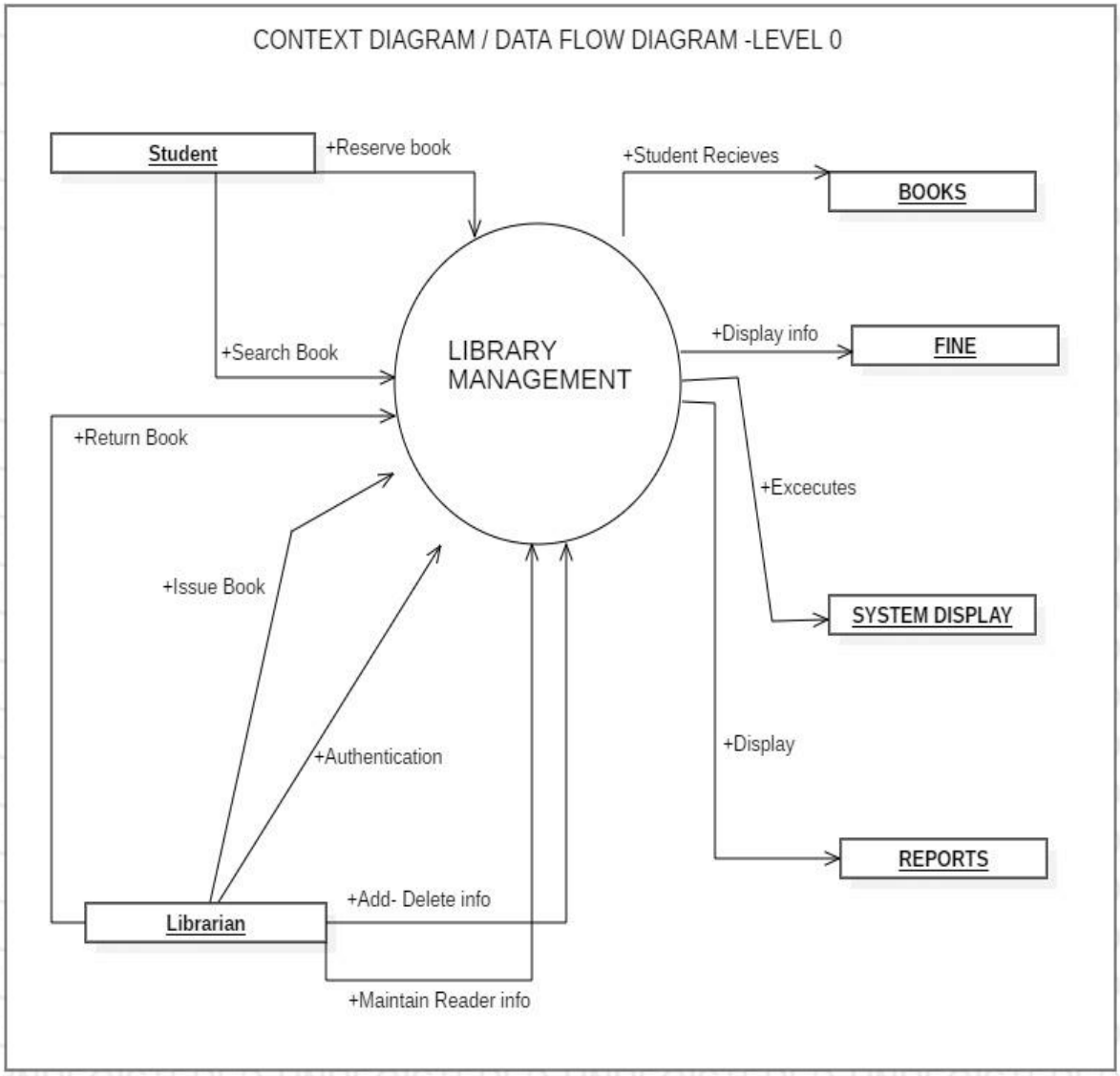
- **Library:** The central part of the organization for which this software has been designed. It has attributes like 'Name' to distinguish it from any other libraries and 'Address' to describe its location.
- **Book:** The basic building block of the system. Every book will have ISBN, Title, Subject, Publishers, etc.
- **BookItem:** Any book can have multiple copies, each copy will be considered a book item in our system. Each book item will have a unique barcode.
- **Account:** We will have two types of accounts in the system, one will be a general member, and the other will be a librarian.
- **LibraryCard:** Each library user will be issued a library card, which will be used to identify users while issuing or returning books.
- **BookReservation:** Responsible for managing reservations against book items.
- **BookLending:** Manage the checking-out of book items.
- **Catalog:** Catalogs contain list of books sorted on certain criteria. Our system will support searching through four catalogs: Title, Author, Subject, and Publish-date.
- **Fine:** This class will be responsible for calculating and collecting fines from library members.
- **Author:** This class will encapsulate a book author.
- **Rack:** Books will be placed on racks. Each rack will be identified by a rack number and will have a location identifier to describe the physical location of the rack in the library.
- **Notification:** This class will take care of sending notifications to library members.

Program 7A

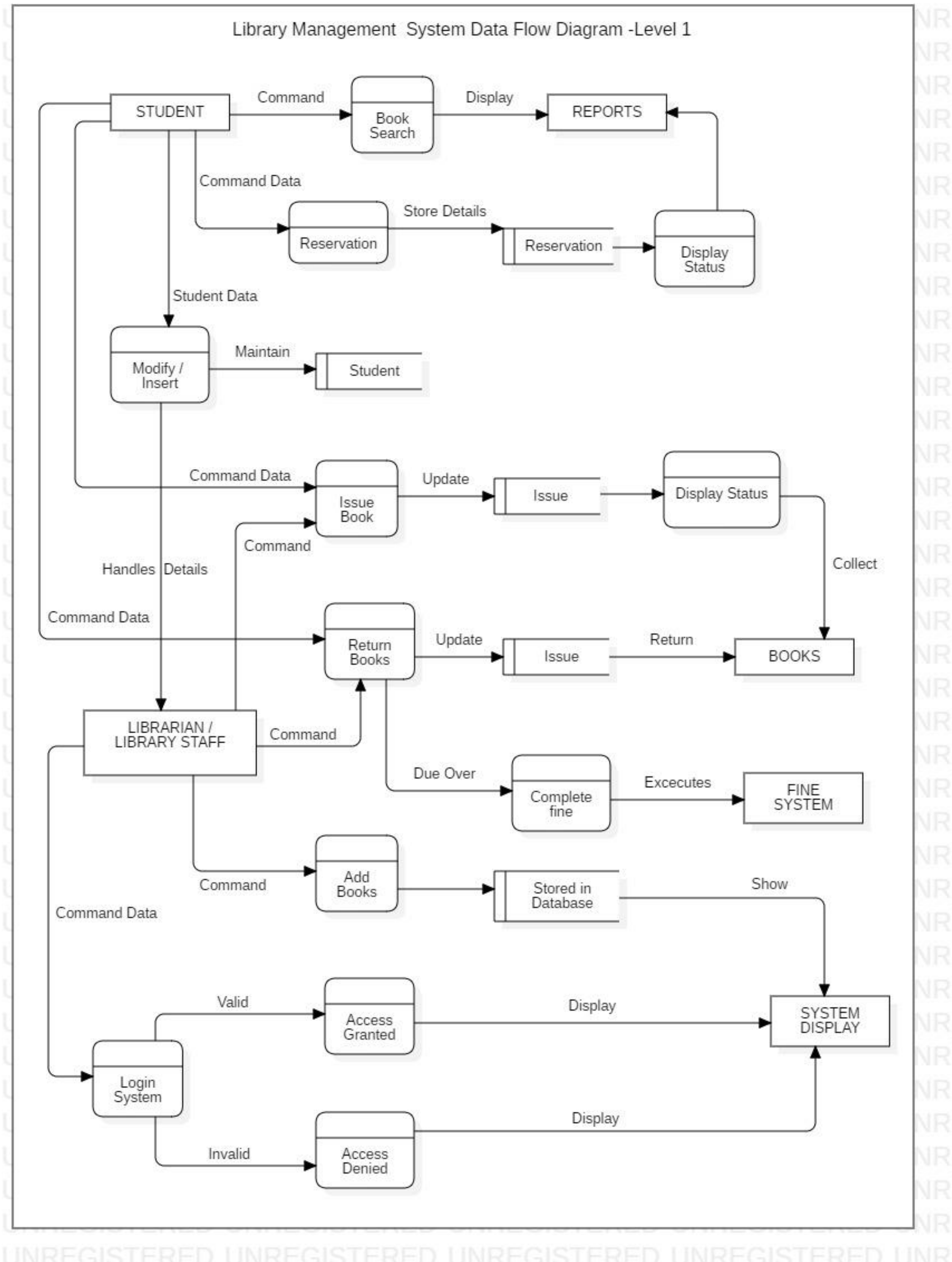
OBJECTIVE:

Draw Data Flow Diagram (DFD) of Library Management System.

CONTEXT DIAGRAM:



DATA FLOW DIAGRAM:



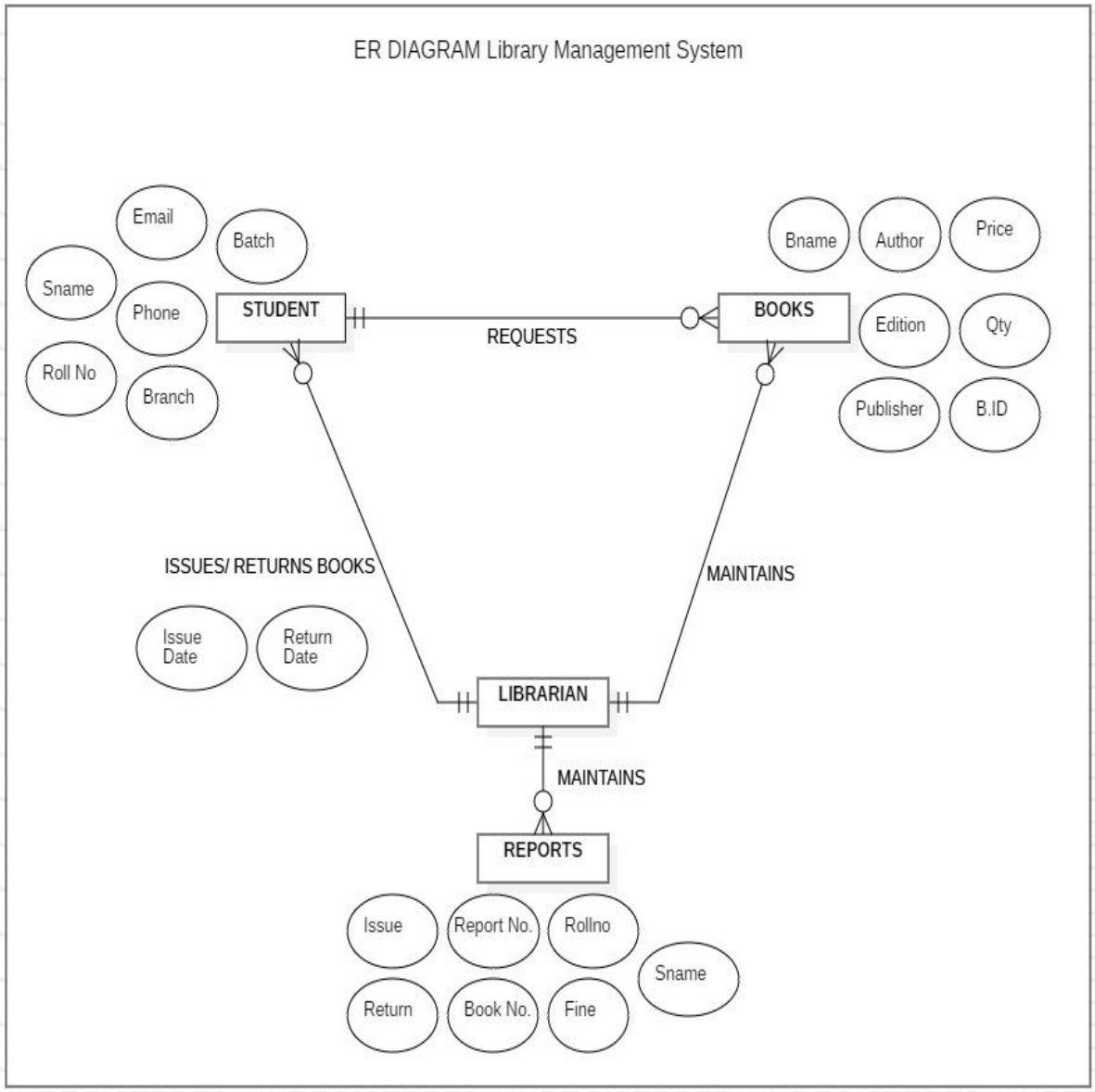
‘

Program 8A

OBJECTIVE:

Draw the ER Diagram of Library Management System.

ER DIAGRAM:



Program 9A

OBJECTIVE:

Design Software Requirement Specification according to IEEE standard for Library Management System.

Software Requirement Specification (SRS):

1.1 Purpose

The purpose of this document is to describe the Web Library Management System (WLMS) product with the release number 0.1. This document contains the functional and non-functional requirements of the project. This document contains the guidelines for website developers system engineers and designers to start working the project.

1.2 Scope

WLMS product is basically updating the manual library system into a internet-based application so that the users can know the details of their accounts, availability of books and remaining time for borrowing.
The project is specifically designed for the use of librarians and library users. The product will work as a complete user interface for library management process and library usage from ordinary users. WLMS can be used by any existing or new library to manage its books and book borrowing, insertion and monitoring .
WLMS can work as a powerful library management system for big libraries, and can provide a free easy-to-use system for rising libraries.

1.3 Audience Definitions, Acronyms and Abbreviations

1.3.1 Audience Definitions

The intended readers of this document are the developers of the site, testers, library owners and managers and coordinators.
Any suggested changes on the requirements listed on this document should be included in the last version of it so it can be a reference to developing and validating teams.

1.3.2 Acronyms and Abbreviations

Acronym	Meaning
WLMS	Web Library Management System
MS SQL	Microsoft Structured Query Language
ASP	Active Server Pages
ISBN	International Standard Book Number
DVD	Digital Video Disc
IEEE	Institute of Electrical and Electronics Engineers

1.4 References

- IEEE 830-1998 standard for writing SRS document.
- I. Sommerville, *Software Engineering*, 8th ed. England: Addison-Wesley, 2007.

1.5 Overview

Section 2 defines the general functions of WLMS, operating environment and user constraints along with our assumptions.

Section 3 specifies functional and nonfunctional requirements; all of them are described to a level of detail sufficient for designers to design a system.

Section 4 illustrates interfaces and its possible scenarios along with some screenshots to make a general idea about the interfaces.

Section 5 specifies all stored information that we are concerned about for every entity in the website .

2. Overall Description

2.1 Product Perspective

WLMS is a replacement for the ordinary library management systems which depend on paper work for recording book and users' information.

WLMS will provide an advanced book search mechanism and will make it easy to borrow, insert and index a book in the library.

2.2 Product Functions

2.2.1 Administrators

- Admin should be able to insert, modify and delete books.
- Can accept or reject a new user according to the library policy or payment methods.
- Increase the period for borrowing a book for specific type or group of users.
- Can get the information (status report) of any member who has borrowed a book.
- Add and edit book categories and arrange books by categories.
- Add and edit authors and publishers information.
- Can send lateness warnings to people who have exceeded deadline date.
- Can record books returned by users.

2.2.2 Normal Users (Library Members)

- The member should be provided with the updated information about the books catalog.
- Members are given a provision to check their account's information and change it.
- Members have the ability to search through books by subject, title, authors or any information related to the book.

- Can extend the period of borrowing books according to the library policy.
- The customer may suggest a book to be brought to the library book collection.

2.3 Operating Environment

The WLMS is a website and shall operate in all famous browsers, for a model we are taking Microsoft Internet Explorer versions 7.0, 8.0 and 9.0, with Flash Player 9 and JavaScript.

2.4 User Characteristics

Users of the website are members, librarians and the administrators who maintain the website. Members and librarians are assumed to have basic knowledge of computers and Internet browsing. Administrators of the system should have more knowledge of internal modules of the system and are able to rectify small problems that may arise due to disk crashes, power failures and other catastrophes. Friendly user interface, online help and user guide must be sufficient to educate the users on how to use this product without any problems or difficulties.

2.5 Design and Implementation Constraints

- The information of all users, books and libraries must be stored in a database that is accessible by the website.
- MS SQL Server will be used as SQL engine and database.
- The Online Library System is running 24 hours a day.
- Users may access WLMS from any computer that has Internet browsing capabilities and an Internet connection.
- Users must have their correct usernames and passwords to enter into their online accounts and do actions.

2.6 Assumptions and Dependencies

The product needs the following third party products.

- Microsoft SQL server to store the database.
- ASP.net to develop the Product.

The success of this system depends on

- Existence of an Internet service to all people in Gaza Strip.
- Are librarians and users comfortable with computers and have enough conation to work with the product?
- Website interface must be friendly and easy-to-use.
- The search mechanism should be simple and fast.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 Librarian

Prerequisite (admin signed in) for all requirements below

Requirement ID	R1.01.01
Title	insert book
Description	This action is done to add new book to library book collection.
Priority	2

Requirement ID	R1.01.02
Title	delete / modify book
Description	this event is to delete an existing book or modify its information.
Priority	2

Requirement ID	R1.01.03
Title	Validate user account
Description	when a new member sign up then he should wait for acceptance by Administrator according to library policies (e.g. fees required).
Priority	1

Requirement ID	R1.01.04
Title	delete member
Description	Admin can delete a member due to some specific rules.
Priority	2

Requirement ID	R1.01.05
Title	modify member rank
Description	Admin can extend the borrowing time or number of book borrowed simultaneity to a user.
Priority	2

Requirement ID	R1.01.06
Title	return book
Description	Admin should confirm the return of books borrowed by users.
Priority	1

3.1.2 Normal User

Requirement ID	R1.02.01
Title	register
Description	when new user enters WLMS for the first time then he has to register
Priority	3

Requirement ID	R1.02.02
Title	extending borrowing deadline.
Description	member can extend the borrowing time to some limit decided by Admin
Priority	2
Requirement ID	R1.02.03
Title	reset password
Description	when a member forgets his password he can claim it back via e-mail.

Priority	1

Requirement ID	R1.02.04
Title	edit personal information
Description	if some user changes for example his mobile number, he can modify it.
Priority	2

Requirement ID	R1.02.05
Title	reset password
Description	when a member forgets his password he can claim it back via e-mail.
Priority	1

3.1.3 Common Functions

Requirement ID	R1.03.01
Title	login
Description	both Admin and members must be logged in before they modify any information
Priority	1

Requirement ID	R1.03.02
Title	search for book
Description	when user or admin wants to search on some book by name, author or subject etc.
Priority	1

3.2 Non-functional Requirements

3.2.1 Error handling

- WLMS product shall handle expected and non-expected errors in ways that prevent loss in information and long downtime period.

3.2.2 Performance Requirements

- The system shall accommodate high number of books and users without any fault.
- Responses to view information shall take no longer than 5 seconds to appear on the screen.

3.2.3 Safety Requirements

- System use shall not cause any harm to human users.

3.2.4 Security Requirements

- System will use secured database
- Normal users can just read information but they cannot edit or modify anything except their personal and some other information.
- System will have different types of users and every user has access constraints.

Program 10A

OBJECTIVE:

Design Test cases of Library Management System using boundary value method.

DESCRIPTION:

Boundary Value Analysis

Consider a program with two input variables x and y. These input variables have specified boundaries as:

$$a \leq x \leq b$$
$$c \leq y \leq d$$

The boundary value analysis test cases for our program with two inputs variables (x and y) that may have any value from 100 to 300 are: (200,100), (200,101), (200,200), (200,299), (200,300), (100,200), (101,200), (299,200) and (300,200). This input domain is shown in Fig. 8.5. Each dot represents a test case and inner rectangle is the domain of legitimate inputs. Thus, for a program of n variables, boundary value analysis yield **4n + 1** test cases.

The test cases for library management system is an application that explains the test cases for library management system. Software testing is a critical part that is involved in the overall development of the application. This will be one of the interesting projects that one can work on and implement in real time world.

Quality assurance is the review of the software product that checks for the correctness, reliability, completeness and maintainability. The different sections under quality assurance are unit testing, integrated testing, validation testing, output testing and user acceptance testing. Test cases gives an idea like on perform of some tasks what will be the predicted output or result. It will help in predicting the result on perform of certain tasks. The test cases below gives an idea of what result must be obtained on performing a particular task.

- **Login form**: The test cases involved are whether valid password and name are entered or invalid name and password entered.
- **Book entry form**: The test cases included are-on the click of add button, delete button, update button, search button, clear button, exit button and next button.
- **User account form**: The test cases included are-on the click of add button, delete button, update button, search button, clear button, exit button and next button.
- **Book return form**: The test cases included are-on the click of add button, delete button, update button, search button, clear button, exit button and next button.

Test Cases for Library Management System

Login Form:

SL.No	Test Case	Excepted Result	Test Result
1	Enter valid name and password & click on login button	Software should display main window	Successful
2	Enter invalid	Software should not display main window	successful

Book Entry Form:

SL.No	Test Case	Excepted Result	Test Result
1	On the click of ADD button	At first user have to fill all fields with proper data, if any Error like entering text data instead of number or entering number instead of text..is found then it gives proper message otherwise Adds Record To the Database	successful
2.	On the Click of DELETE Button	This deletes the details of book by using Accession no.	Successful
3.	On the Click of UPDATE Button	Modified records are Updated in database by clicking UPDATE button.	Successful
4.	On the Click of SEARCH Button	Displays the Details of book for entered Accession no. Otherwise gives proper Error message.	Successful
5.	On the Click of CLEAR Button	Clears all fields	Successful
6.	On the Click of EXIT button	Exit the current book details form	successful
7.	On the Click of NEXT button	Display the next form	successful

User Account Form:

SL.No	Test Case	Excepted Result	Test Result
1	On the click of ADD button	At first user have to fill all fields with proper data , if any Error like entering text data instead of number or entering number instead of text..is found then it gives proper message otherwise Adds Record To the Database	successful
2.	On the Click of DELETE Button	This deletes the details of student by using Register no.	Successful
3.	On the Click of UPDATE Button	Modified records are Updated in database by clicking UPDATE button.	Successful
4.	On the Click of SEARCH Button	Displays the Details of book for entered Register no. Otherwise gives proper Error message.	Successful
5.	On the Click of CLEAR Button	Clears all fields	Successful
6.	On the Click of EXIT button	Exit the current book details form	successful
7.	On the Click of NEXT button	Display the next form	successful

Book Issue Form:

SL.No	Test Case	Excepted Result	Test Result
1	On the click of ADD button	At first user have to fill all fields with proper data ,if the accession number book is already issued then it will giving proper msg.	successful
2.	On the Click of DELETE Button	This deletes the details of book by using Register no.	Successful
3.	On the Click of UPDATE Button	Modified records are Updated in database by clicking UPDATE button.	Successful
4.	On the Click of SEARCH Button	Displays the Details of issued book..Otherwise gives proper Error message.	Successful
5.	On the Click of CLEAR Button	Clears all fields	Successful
6.	On the Click of EXIT button	Exit the current book details form	successful
7.	On the Click of NEXT button	Display the next form	successful

Book Return Form:

SL.No	Test Case	Excepted Result	Test Result
1	On the click of ADD button	At first user have to fill all fields with proper data , if any Error like entering text data instead of number or entering number instead of text..is found then it gives proper message otherwise Adds Record To the Database	successful
2.	On the Click of DELETE Button	Which deletes the details of book by using Register no.	Successful
3.	On the Click of UPDATE Button	Modified records are Updated in database by clicking UPDATE button.	Successful
4.	On the Click of SEARCH Button	Displays the Details of returned book … Otherwise gives proper Error message.	Successful
5.	On the Click of CLEAR Button	Clears all fields	Successful
6.	On the Click of EXIT button	Exit the current book details form	successful
7.	On the Click of NEXT button	Display the next form	successful