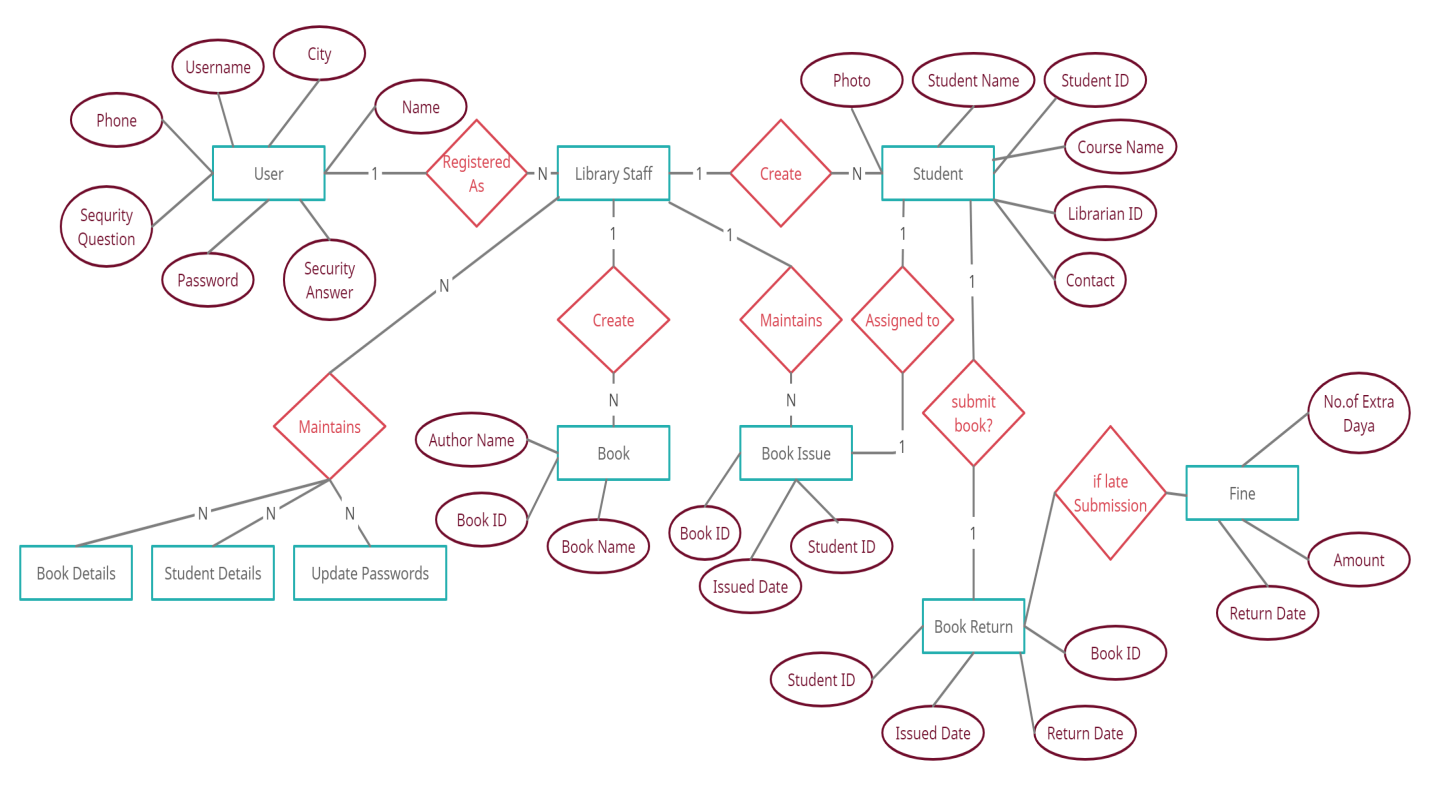
**Task-1: Idea**

Library Management system which handles the data about the books exist in the library, authors, and the candidates of library to whom books were issued, library workforce etc... This is pretty much difficult to categorize manually. It is very difficult to maintain the information manually. Owing to the advancement of technology, organization of an Online Library becomes much simple. The Library Management system has been designed to computerize and automate the operations performed over the information about the members, add/remove books, add/remove students, book renewals, book fine for late submission details, book issues and returns and all other operations. This computerization of library helps in many instances of its maintenances. It reduces the workload of management as most of the manual work done is reduced. The main of Library Management application is to keep the book in the proper format with its complete details including Author and the person who is issuing this book. Everything is managed from the Mysql database. We need to maintain the book cost and to check whether the books are available on the library or not.

**2. Design schema**

**E-R Diagram**



**Database SQL Design Schema**

/\*!40101 SET NAMES utf8 \*/;

/\*!40101 SET SQL\_MODE=''\*/;

create database if not exists `library`;

USE `library`;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*Table structure for table `admin` \*/

CREATE TABLE `admin` (

`id` varchar(200) NOT NULL,

`name` varchar(200) NOT NULL,

`password` varchar(200) NOT NULL,

`secQuestion` varchar(200) NOT NULL,

`secAnswer` varchar(200) NOT NULL,

`Phone` varchar(200) NOT NULL,

`City` varchar(100) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

Graphical user interface, application, Word

Description automatically generated

/\*Table structure for table `books` \*/

CREATE TABLE `books` (

`Book\_Id` int(11) NOT NULL,

`Book\_name` text NOT NULL,

`Author` text NOT NULL,

`Availiability` tinyint(1) NOT NULL,

PRIMARY KEY (`Book\_Id`),

UNIQUE KEY `Book\_Id` (`Book\_Id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

Graphical user interface, application

Description automatically generated

/\*Table structure for table `issue` \*/

CREATE TABLE `issue` (

`BID` int(11) NOT NULL,

`SID` int(11) NOT NULL,

`Issue\_date` date NOT NULL,

`Return\_date` date NOT NULL,

PRIMARY KEY (`BID`,`SID`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

Graphical user interface, application

Description automatically generated

/\*Table structure for table `students` \*/

CREATE TABLE `students` (

`Roll\_no` int(11) NOT NULL,

`name` varchar(120) NOT NULL,

`Student\_Id` varchar(120) NOT NULL,

`class` varchar(200) NOT NULL,

`Phone\_number` varchar(20) NOT NULL,

`Image` blob NOT NULL,

`Books\_Issued` int(11) NOT NULL DEFAULT '0',

`Fine` float NOT NULL DEFAULT '0',

PRIMARY KEY (`Student\_Id`,`Roll\_no`),

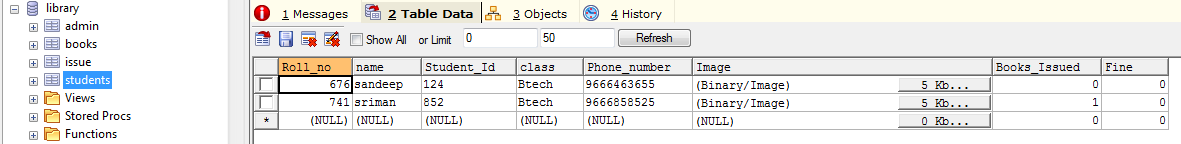
UNIQUE KEY `Roll\_no` (`Roll\_no`),

UNIQUE KEY `Student\_Id` (`Student\_Id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;



/\*Table structure for table `breturns` \*/

CREATE TABLE `breturns` (

`bid` varchar(200) DEFAULT NULL,

`sid` varchar(200) DEFAULT NULL,

`return\_date` varchar(200) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

Graphical user interface, application

Description automatically generated

**Task-3: SQL queries**

**Query 1:**

**SELECT**

**b.book\_id,b.book\_name,b.author,**

**(SELECT COUNT(\*) FROM issue WHERE issue.bid = b.book\_id) AS issue\_count,**

**(SELECT COUNT(\*) FROM breturns WHERE breturns.bid IN (**

**SELECT bid FROM breturns WHERE breturns.bid = b.book\_id**

**)**

**) AS breturns\_count**

**FROM books AS b**

**ORDER BY breturns\_count DESC, issue\_count DESC, author DESC**

In this area, I am creating sub query by following 3 queries having query data from more than one table This SQL Query returns the total number of book issues and book returns for each and every book.

**Query 2:**

**select sid,count(bid) from issue group by sid**

In this area, I am using grouping query which is formed using “group by” command. This SQL Query returns the total number of book issues from each and every student.

**Query 3:**

**Select b.Book\_Id,b.Book\_name,b.Author,b.Availiability,count(iu.sid) from books b,issue iu where iu.BID=%s and b.Book\_Id LIKE %s "**, (mValue.get(),**'%'**+mValue.get()+**'%'**

In this area, I am using Aggregation query which is formed using “count()” command. This SQL Query returns the book details and the total number of students who takes the books from each and every book.

**Query 4:**

**SELECT issue.BID, books.book\_name,books.Author,issue.Issue\_date,issue.Return\_date FROM issue INNER JOIN books ON issue.BID=books.book\_id**

In this area, I am using SQL JOIN query which is formed using “INNER JOIN” command. This SQL Query returns the book details with date of issue and return books.

**Query 5:**

**CREATE VIEW booksavialability AS SELECT book\_name,Availiability FROM books**

In this area, I am using VIEW query which is formed using “CREATE VIEW” command. This SQL Query creates the new table which contains books availability with book details.

**Task-4: Implementation**

This is included as “Implementation” document

**Task-5: Supplemental video**

**Please watch the project demonstration Video by using following link**

<https://youtu.be/1LYdZd79qZ8>

**Git Repo Link**

<https://github.com/gr222ep/Databaseassignment2/>