­

**DATABASE SYSTEMS**

**MIDTERM PROJECT**

LIBRARY SYSTEM

SQLovers2 </3

Chua, John Patrick G.

Coroza, Christian Kyle M.

Guerrero, Drennix Jean-Roe E.

Rizan, Hezekiah John V.

CS0005 DATABASE SYSTEMS

FINAL PROJECT SPECIFICATIONS

Project: Library System

Using the EERD that you have created on your midterm project, create a working Library database system using any programming language that supports Oracle database. The Library system should be able to:

* add, delete and edit users,
* let users borrow, return and compute and pay dues for overdue books (for rules refer below),
* and can search for books in the system showing their status.

The following are the requirements for the database:

1. The database should contain all the tables stated on the EERD. It should be complete with attributes, primary and foreign key, and it should be normalized
2. Create a stored procedure to:
   * Add, edit, delete a user (this option is available for librarian, patrons are not allowed to change information except the librarian)
   * Add, edit details and delete a book (see detail number 7)
   * Withdraw or return a book (Status should be shown as in shelf, withdrawn and other applicable attributes)
   * Return the book (If the book is overdue, it should show the fine applicable for the overdue)
3. The database should contain information about each valid patron of the system: loginid, name, password, address, unpaid fines and the identity of each book the patron has currently withdrawn. The system shall contain information about each librarian: loginid, name, password and address.
4. The database should contain information about each book in the library: isbn number, title, author name(s), year (of publication), copy number, shelfid, current status, the date on which the book entered loan and/or hold status.
5. The database should contain information about each shelf in the library: shelfId, *capacity* (the maximum number of books that the shelf can hold).
6. The database should contain information about the author of the book: authorid, authorname.
7. A user can either be a librarian or patron, but a librarian can be a patron at the same time. A patron can borrow a maximum of two books per transaction and only when the books are return, he/she can have another transaction. The maximum days a patron can borrow a book is 1 week. It will incur a penalty charge when the books are borrowed beyond 1 week. The penalty is P20 per day of delay.
8. The books should have a designated bookshelf. Each book can be authored by one or more authors.

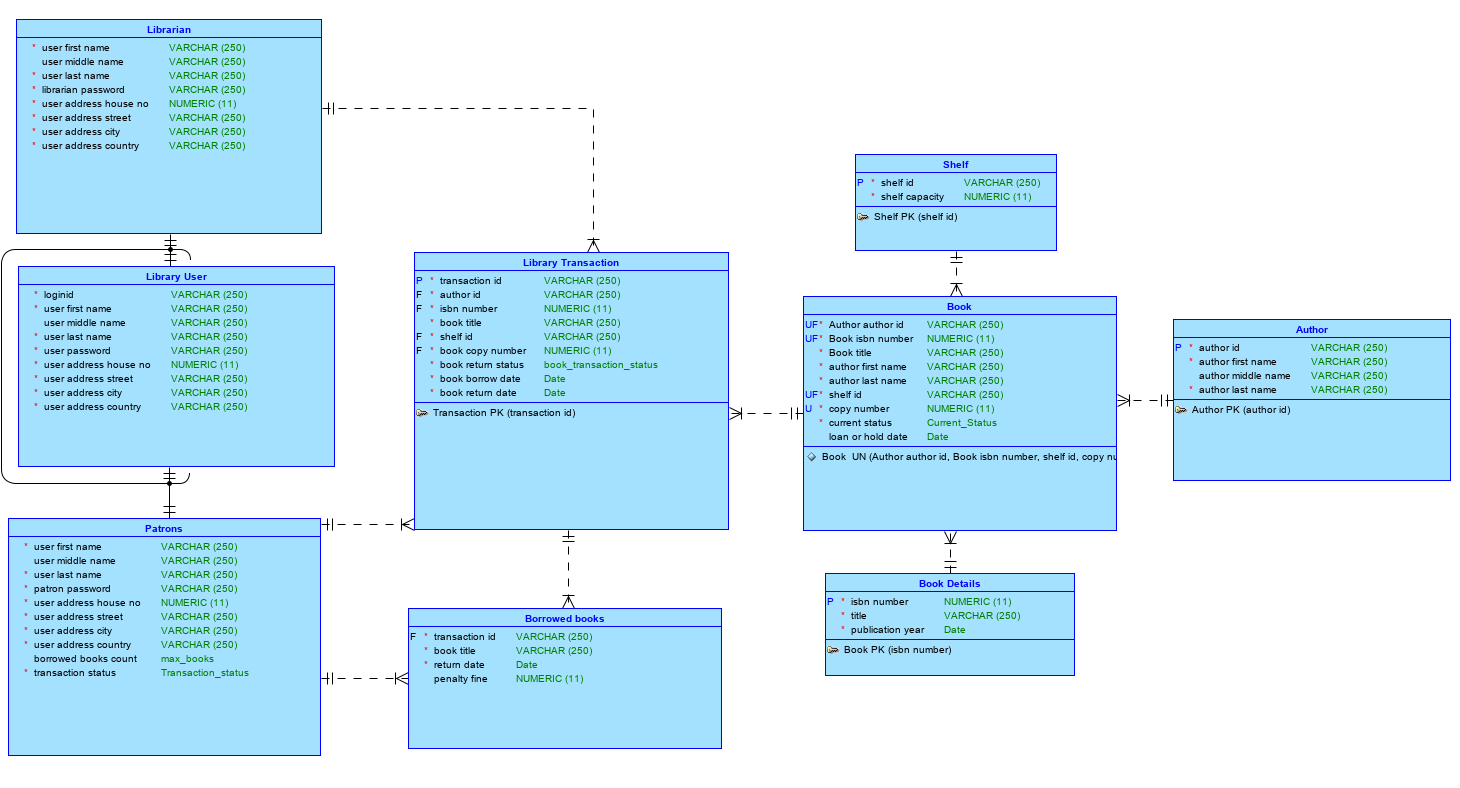
Requirements:

Documentation (This should be submitted on Canvas)

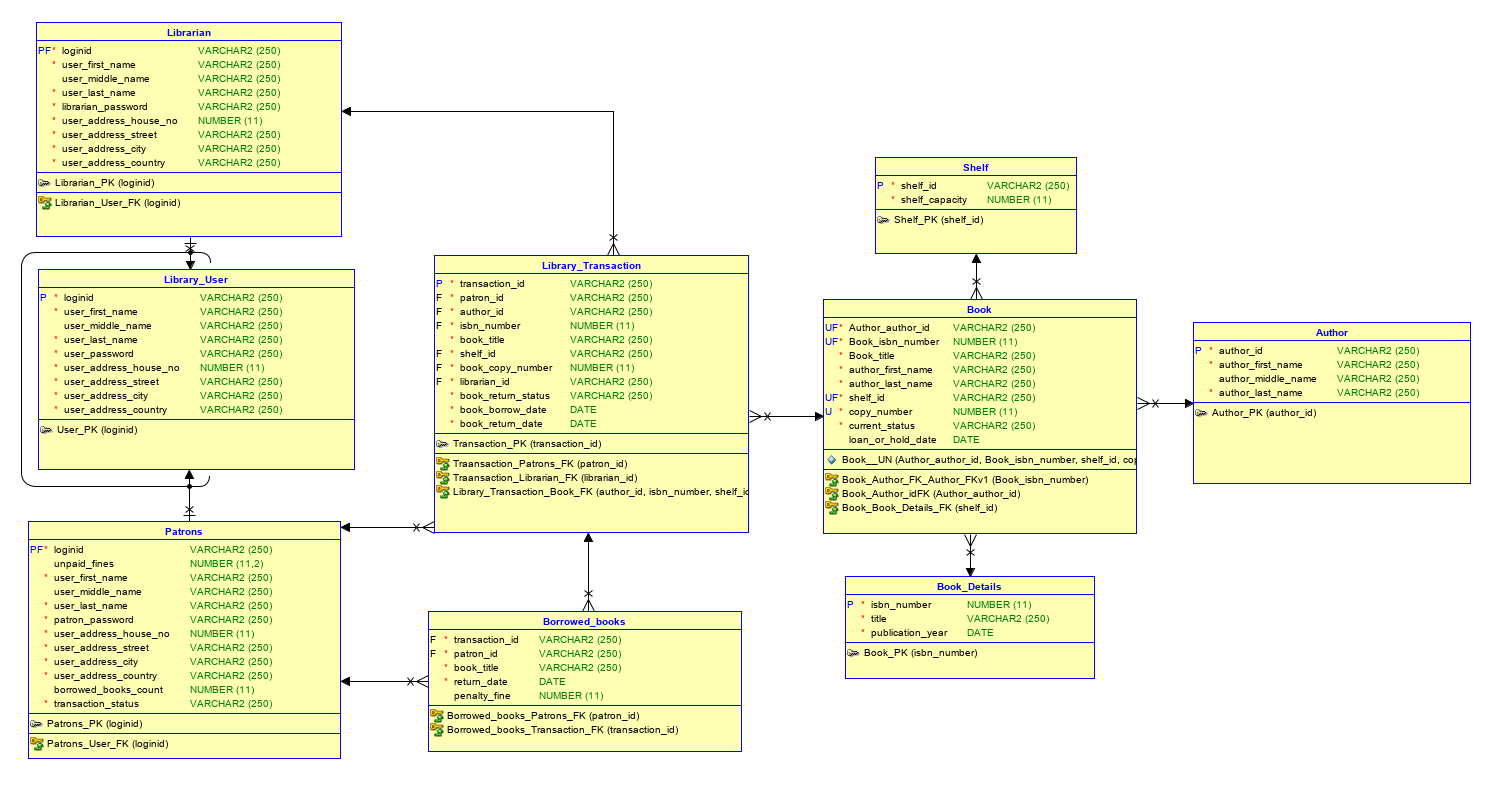
* The EERD you have created
* Screenshots of the tables and records inside the tables
* Source codes of the procedures

Presentation

* Present the database that you have created in class



EERD Logical Model



EERD Relational Model

--add user procedure

create or replace PROCEDURE add\_user(user\_name IN library\_user.loginid%TYPE,

fName IN library\_user.user\_first\_name%TYPE,

mName IN library\_user.user\_middle\_name%TYPE,

lName IN library\_user.user\_last\_name%TYPE,

user\_pass IN library\_user.user\_password%TYPE,

hNo IN library\_user.user\_address\_house\_no%TYPE,

hStreet IN library\_user.user\_address\_street%TYPE,

hCity IN library\_user.user\_address\_city%TYPE,

hCountry IN library\_user.user\_address\_country%TYPE,

isLib IN NUMBER)

IS

BEGIN

INSERT INTO Library\_user VALUES(user\_name,fName, mName, lName, user\_pass,

hNo, hStreet, hCity, hCountry);

IF isLib = 1 THEN

INSERT INTO Librarian VALUES(user\_name,fName, mName, lName, user\_pass,

hNo, hStreet, hCity, hCountry);

END IF;

INSERT INTO Patrons VALUES(user\_name, 0, fName, mName, lName, user\_pass,

hNo, hStreet, hCity, hCountry, 0, 'Allowed');

END;

--change shelf procedure

create or replace PROCEDURE change\_shelf(book\_isbn IN book\_details.isbn\_number%TYPE,

cpyNo IN book.copy\_number%TYPE, newshelf IN book.shelf\_id%TYPE)

IS

oldshelf book.shelf\_id%TYPE;

max\_shelf\_cap shelf.shelf\_capacity%TYPE;

e\_max\_shelf\_cap EXCEPTION;

BEGIN

SELECT shelf\_capacity INTO max\_shelf\_cap FROM shelf WHERE shelf\_id = newshelf;

IF max\_shelf\_cap = 20 THEN

RAISE e\_max\_shelf\_cap;

ELSE

SELECT shelf\_id INTO oldshelf FROM book

WHERE book.book\_isbn\_number = book\_isbn

AND book.copy\_number = cpyNo

AND ROWNUM = 1;

UPDATE shelf SET shelf\_capacity = shelf\_capacity + 1 WHERE shelf\_id = oldshelf;

UPDATE shelf SET shelf\_capacity = shelf\_capacity - 1 WHERE shelf\_id = newshelf;

UPDATE book SET shelf\_id = newshelf

WHERE book.book\_isbn\_number = book\_isbn

AND book.copy\_number = cpyNo;

UPDATE library\_transaction

SET shelf\_id = newshelf

WHERE isbn\_number = book\_isbn

AND book\_copy\_number = cpyNo;

END IF;

END;

--change title procedure

create or replace PROCEDURE change\_title(book\_isbn IN book\_details.isbn\_number%TYPE,

new\_title IN book\_details.title%TYPE)

IS

e\_no\_book\_exist EXCEPTION;

book\_count NUMBER;

BEGIN

SELECT COUNT(\*) INTO book\_count FROM book\_details WHERE isbn\_number = book\_isbn;

IF book\_count = 0 THEN RAISE e\_no\_book\_exist;

END IF;

UPDATE book\_details SET title = new\_title

WHERE book\_details.isbn\_number = book\_isbn;

UPDATE book SET book\_title = new\_title

WHERE book.book\_isbn\_number = book\_isbn;

UPDATE library\_transaction

SET book\_title = new\_title

WHERE isbn\_number = book\_isbn;

UPDATE borrowed\_books

SET book\_title = new\_title

WHERE book\_title = new\_title;

END;

--check reserve procedure

create or replace PROCEDURE check\_reserve

IS

patID library\_transaction.patron\_id%TYPE;

isbn\_num library\_transaction.isbn\_number%TYPE;

shlf\_id library\_transaction.shelf\_id%TYPE;

cpy\_no library\_transaction.book\_copy\_number%TYPE;

BEGIN

SELECT patron\_id, isbn\_number, shelf\_id, book\_copy\_number

INTO patID, isbn\_num, shlf\_id, cpy\_no

FROM library\_transaction

WHERE SYSDATE > book\_return\_date

AND book\_return\_status = 'on-loan';

UPDATE library\_transaction

SET book\_return\_status = 'returned'

WHERE SYSDATE > book\_return\_date

AND book\_return\_status = 'on-loan';

UPDATE book

SET current\_status = 'on-shelf',

loan\_or\_hold\_date = null

WHERE book\_isbn\_number = isbn\_num

AND shelf\_id = shlf\_id

AND copy\_number = cpy\_no;

UPDATE patrons

SET borrowed\_books\_count = borrowed\_books\_count - 1

WHERE loginid = patID;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Reservation checked.');

END;

--check user procedure

create or replace PROCEDURE checkUser

IS

usercount NUMBER;

e\_no\_user EXCEPTION;

BEGIN

SELECT COUNT(\*) INTO usercount FROM library\_user;

IF usercount = 0 THEN

RAISE e\_no\_user;

END IF;

END;

--delete books procedure

create or replace PROCEDURE delete\_books(book\_isbn IN book.book\_isbn\_number%TYPE,

title IN book.book\_title%TYPE, cpyNo IN book.copy\_number%TYPE)

IS

shlf\_id shelf.shelf\_id%TYPE;

book\_in\_trans NUMBER;

e\_book\_in\_transaction EXCEPTION;

book\_status book.current\_status%TYPE;

BEGIN

SELECT COUNT(\*) INTO book\_in\_trans FROM library\_transaction

WHERE isbn\_number = book\_isbn

AND book\_copy\_number = cpyNo

AND book\_return\_status <> 'returned';

IF book\_in\_trans > 0 THEN RAISE e\_book\_in\_transaction;

END IF;

SELECT current\_status INTO book\_status FROM book

WHERE book\_isbn\_number = book\_isbn

AND copy\_number = cpyNo;

IF book\_status = 'on-loan' OR book\_status = 'on-hold' THEN

RAISE e\_book\_in\_transaction;

END IF;

SELECT shelf\_id INTO shlf\_id

FROM book WHERE book.book\_isbn\_number = book\_isbn

AND book.book\_title = title

AND book.copy\_number = cpyNo

AND ROWNUM = 1;

UPDATE shelf SET shelf\_capacity = shelf\_capacity + 1

WHERE shelf\_id = shlf\_id;

DELETE FROM book WHERE book.book\_isbn\_number = book\_isbn

AND book.book\_title = title AND book.copy\_number = cpyNo;

END;

--delete user procedure

create or replace PROCEDURE delete\_user(userID IN library\_user.loginid%TYPE)

IS

has\_books NUMBER;

e\_user\_hasBooks EXCEPTION;

BEGIN

SELECT borrowed\_books\_count INTO has\_books FROM patrons

WHERE loginid = userID;

IF has\_books > 0 THEN RAISE e\_user\_hasBooks;

END IF;

DELETE FROM library\_user WHERE loginid = userID;

END;

--insert author procedure

create or replace PROCEDURE insert\_author(author\_id IN author.author\_id%TYPE,

fName IN author.author\_first\_name%TYPE,

mNane IN author.author\_middle\_name%TYPE,

lName IN author.author\_last\_name%TYPE) IS

BEGIN

INSERT INTO author VALUES(author\_id, fName, mNane, lName);

END;

--insert book details procedure

create or replace PROCEDURE insert\_book\_details(isbn\_num IN book\_details.isbn\_number%TYPE,

bkTitle IN book\_details.title%TYPE,

pub\_date IN book\_details.publication\_year%TYPE) IS

BEGIN

INSERT INTO book\_details(isbn\_number, title, publication\_year)

VALUES(isbn\_num, bkTitle, pub\_date);

END;

--insert books procedure

create or replace PROCEDURE insert\_books(auth\_id IN book.author\_author\_id%TYPE,

book\_isbn IN book.book\_isbn\_number%TYPE,

shf\_id IN book.shelf\_id%TYPE,

cpyNo IN book.copy\_number%TYPE)

IS

book\_title book.book\_title%TYPE;

auth\_fName book.author\_first\_name%TYPE;

auth\_lName book.author\_last\_name%TYPE;

max\_shelf\_cap shelf.shelf\_capacity%TYPE;

book\_isThere NUMBER;

e\_max\_shelf\_cap EXCEPTION;

exist\_shelf book.shelf\_id%TYPE;

inp\_shelf book.shelf\_id%TYPE;

anyrows NUMBER;

BEGIN

SELECT COUNT(\*) INTO anyrows FROM book WHERE book.book\_isbn\_number = book\_isbn

AND book.copy\_number = cpyNo;

IF anyrows <> 0 THEN

SELECT shelf\_id INTO exist\_shelf FROM book WHERE book.book\_isbn\_number = book\_isbn

AND book.copy\_number = cpyNo;

inp\_shelf := exist\_shelf;

ELSE

SELECT shelf\_capacity INTO max\_shelf\_cap FROM shelf WHERE shelf\_id = shf\_id;

IF max\_shelf\_cap = 0 THEN

RAISE e\_max\_shelf\_cap;

ELSE

UPDATE shelf SET shelf\_capacity = shelf\_capacity - 1

WHERE shelf\_id = shf\_id;

END IF;

inp\_shelf := shf\_id;

END IF;

SELECT title INTO book\_title FROM book\_details

WHERE book\_details.isbn\_number = book\_isbn;

SELECT author\_first\_name, author\_last\_name INTO auth\_fName, auth\_lName

FROM author WHERE author.author\_id = auth\_id;

INSERT INTO book (author\_author\_id, book\_isbn\_number,

book\_title, author\_first\_name, author\_last\_name,

shelf\_id, copy\_number, current\_status)

VALUES(auth\_id, book\_isbn, book\_title, auth\_fName,

auth\_lName, inp\_shelf, cpyNo, 'on-shelf');

END;

--insert shelf

create or replace PROCEDURE insert\_shelf(shelf\_id IN shelf.shelf\_id%TYPE,

shelf\_capacity IN shelf.shelf\_capacity%TYPE) IS

BEGIN

INSERT INTO shelf VALUES (shelf\_id, shelf\_capacity);

END;

--pay fines procedure

create or replace PROCEDURE pay\_fines(patID patrons.loginid%TYPE,

paid\_fine patrons.unpaid\_fines%TYPE)

IS

userFine patrons.unpaid\_fines%TYPE;

e\_no\_fine EXCEPTION;

BEGIN

SELECT unpaid\_fines INTO userFine FROM patrons

WHERE loginid = patID;

IF userFine = 0 OR userFine IS NULL THEN

RAISE e\_no\_fine;

END IF;

UPDATE patrons SET

unpaid\_fines = unpaid\_fines - paid\_fine

WHERE loginid = patID;

END;

--reserve book procedure

create or replace PROCEDURE reserve\_book

( patID IN patrons.loginid%TYPE,

isbnNum IN book.book\_isbn\_number%TYPE,

shf\_ID IN book.shelf\_id%TYPE,

cpyNo IN book.copy\_number%TYPE

)

IS

pat\_books patrons.borrowed\_books\_count%TYPE;

book\_status book.current\_status%TYPE;

bkTitle book.book\_title%TYPE;

auth\_id book.author\_author\_id%TYPE;

e\_book\_on\_loan EXCEPTION;

e\_max\_books EXCEPTION;

BEGIN

SELECT current\_status INTO book\_status FROM book

WHERE book\_isbn\_number = isbnNum

AND copy\_number = cpyNo

AND shelf\_id = shf\_ID;

IF book\_status = 'on-loan' OR book\_status = 'on-hold' THEN

RAISE e\_book\_on\_loan;

ELSE

SELECT borrowed\_books\_count INTO pat\_books FROM patrons WHERE patrons.loginid = patID;

IF pat\_books <> 2 THEN

UPDATE book

SET current\_status = 'on-hold',

loan\_or\_hold\_date = SYSDATE

WHERE book\_isbn\_number = isbnNum

AND copy\_number = cpyNo

AND shelf\_id = shf\_ID;

SELECT author\_author\_id, book\_title INTO auth\_id, bkTitle

FROM book

WHERE book\_isbn\_number = isbnNum

AND shelf\_id = shf\_ID

AND copy\_number = cpyNo

AND ROWNUM = 1;

INSERT INTO library\_transaction VALUES(transaction\_id.nextval, patID, auth\_id,

isbnNum, bkTitle, shf\_ID, cpyNo, 'admin', 'on-hold', SYSDATE, SYSDATE+7);

UPDATE patrons

SET borrowed\_books\_count = borrowed\_books\_count + 1

WHERE loginid = patID;

IF pat\_books = 2 THEN

UPDATE patrons

SET transaction\_status = 'Not Allowed'

WHERE patrons.loginid = patID;

END IF;

ELSE

RAISE e\_max\_books;

END IF;

END IF;

END;

--return book procedure

create or replace PROCEDURE return\_book

( patID IN library\_transaction.patron\_id%TYPE,

isbn\_num IN library\_transaction.isbn\_number%TYPE,

cpyNo IN library\_transaction.book\_copy\_number%TYPE,

libID IN library\_transaction.librarian\_id%TYPE

)

IS

isLib NUMBER;

noOfDays NUMBER;

e\_not\_librarian EXCEPTION;

transID library\_transaction.transaction\_id%TYPE;

ret\_date library\_transaction.book\_return\_date%TYPE;

pen\_fine borrowed\_books.penalty\_fine%TYPE;

BEGIN

SELECT COUNT(\*) INTO isLib FROM librarian

WHERE loginid = libID;

IF isLib <> 0 THEN

SELECT transaction\_id INTO transID FROM library\_transaction

WHERE patron\_id = patID

AND isbn\_number = isbn\_num

AND book\_copy\_number = cpyno

AND book\_return\_status <> 'returned';

UPDATE library\_transaction

SET book\_return\_date = SYSDATE,

book\_return\_status = 'returned'

WHERE transaction\_id = transID;

SELECT return\_date INTO ret\_date FROM borrowed\_books WHERE patron\_id = patID AND

transaction\_id = transID;

IF SYSDATE > ret\_date THEN

UPDATE borrowed\_books

SET penalty\_fine = (SYSDATE - return\_date) \*20,

return\_date = SYSDATE

WHERE patron\_id = patID

AND transaction\_id = transID;

END IF;

SELECT penalty\_fine INTO pen\_fine FROM borrowed\_books

WHERE transaction\_id = transID;

UPDATE Patrons

SET borrowed\_books\_count = borrowed\_books\_count - 1,

transaction\_status = 'Allowed',

unpaid\_fines = unpaid\_fines + pen\_fine

WHERE loginid = patID;

UPDATE book

SET current\_status = 'on-shelf',

loan\_or\_hold\_date = null

WHERE book\_isbn\_number = isbn\_num

AND copy\_number = cpyNo;

ELSE

RAISE e\_not\_librarian;

END IF;

END;

--update user procedure

create or replace PROCEDURE update\_user(uname IN library\_user.loginid%TYPE,

firName IN library\_user.user\_first\_name%TYPE,

midName IN library\_user.user\_middle\_name%TYPE,

lastName IN library\_user.user\_last\_name%TYPE,

upass IN library\_user.user\_password%TYPE,

houseNo IN library\_user.user\_address\_house\_no%TYPE,

houseStreet IN library\_user.user\_address\_street%TYPE,

houseCity IN library\_user.user\_address\_city%TYPE,

houseCountry IN library\_user.user\_address\_country%TYPE)

IS

any\_rows NUMBER;

BEGIN

SELECT COUNT(\*) INTO any\_rows FROM Librarian WHERE loginid = uname;

UPDATE library\_user SET

user\_first\_name = firName,

user\_middle\_name = midName,

user\_last\_name = lastName,

user\_password = upass,

user\_address\_house\_no = houseNo,

user\_address\_street = houseStreet,

user\_address\_city = houseCity,

user\_address\_country = houseCountry

WHERE loginid = uname;

IF any\_rows <> 0 THEN

UPDATE librarian SET

user\_first\_name = firName,

user\_middle\_name = midName,

user\_last\_name = lastName,

librarian\_password = upass,

user\_address\_house\_no = houseNo,

user\_address\_street = houseStreet,

user\_address\_city = houseCity,

user\_address\_country = houseCountry

WHERE loginid = uname;

END IF;

UPDATE patrons SET

user\_first\_name = firName,

user\_middle\_name = midName,

user\_last\_name = lastName,

patron\_password = upass,

user\_address\_house\_no = houseNo,

user\_address\_street = houseStreet,

user\_address\_city = houseCity,

user\_address\_country = houseCountry

WHERE loginid = uname;

END;

--withdraw book procedure

create or replace PROCEDURE withdraw\_book(patID IN patrons.loginid%TYPE,

libId IN librarian.loginid%TYPE,

bkTitle library\_transaction.book\_title%TYPE,

shlfID IN library\_transaction.shelf\_id%TYPE,

cpyNo IN book.copy\_number%TYPE,

bkStatus IN library\_transaction.book\_return\_status%TYPE)

IS

auth\_id library\_transaction.author\_id%TYPE;

isbn\_num library\_transaction.isbn\_number%TYPE;

pat\_books patrons.borrowed\_books\_count%TYPE;

book\_status book.current\_status%TYPE;

trans\_ID library\_transaction.transaction\_id%TYPE;

e\_no\_reserved EXCEPTION;

isAvailable NUMBER;

BEGIN

SELECT COUNT(\*) INTO isAvailable FROM library\_transaction

WHERE patron\_id = patID

AND book\_title = bkTitle

AND shelf\_id = shlfID

AND book\_copy\_number = cpyNo

AND book\_return\_status = 'on-hold';

SELECT transaction\_id INTO trans\_ID FROM library\_transaction

WHERE patron\_id = patID

AND book\_title = bkTitle

AND shelf\_id = shlfID

AND book\_copy\_number = cpyNo

AND book\_return\_status = 'on-hold';

IF isAvailable = 1 THEN

UPDATE library\_transaction

SET book\_return\_status = 'on-loan',

book\_borrow\_date = SYSDATE,

book\_return\_date = SYSDATE + 7,

librarian\_id = libID

WHERE patron\_id = patID

AND book\_title = bkTitle

AND shelf\_id = shlfID

AND book\_copy\_number = cpyNo

AND book\_return\_status = 'on-hold';

UPDATE book

SET current\_status = 'on-loan',

loan\_or\_hold\_date = SYSDATE

WHERE book\_title = bkTitle

AND copy\_number = cpyNo

AND shelf\_id = shlfID;

INSERT INTO borrowed\_books(transaction\_id, patron\_id, book\_title, return\_date)

VALUES(trans\_ID, patID, bkTitle, SYSDATE+7);

ELSE

RAISE e\_no\_reserved;

END IF;

END;

Rubrics

|  |  |  |
| --- | --- | --- |
| Choice of system | The system should be made from scratch and are not an previous existing system | 10% |
| Entity Relationship Diagram/Enhanced Entity Relationship Diagram | The ERD/EERD of the database should be complete and should match with the existing database chosen | 20% |
| Tables and sample records | The sample table should be at least 6 with at least 6 records inside. | 10% |
| GUI | The GUI of the system should be working, complete and appropriate for the system chosen | 20% |
| Working System | The system should be properly working | 20% |
| Errors | The system should have been thoroughly tested, and minimal errors are found during testing of the system | 10% |
| Documentation | The document should show what the system is, how does it work and the screen shots | 10% |
| Total |  | 100% |