UNIVERSITY LIBRARY MANAGEMENT SYSTEM (DBMS PROJECT)

GROUP MEMBERS:
G11:

RAYALA NAVYA HARSHITHA	200003064
CHETANA DHARAVATHU	200005011
PADAMATA KANISHKA SAI	200001058
BATTHULA BHAVANA	200001014
NELAVALLI SRINIKHITHA	200001052

INTRODUCTION:

In the era of modernization, everything is accomplished or thought to be accessed through the internet so that do not need to waste time and face any inconvenience. With the increase of the number of students in Universities, maintaining a library using the old management methods is too complex and time-consuming and hardly possible to adapt to the development of times.

The manual process of keeping student records, book records, account details, managing employees is very difficult. There are various problems also faced by the student in the library such as finding any particular book, information whether the book is available or not, for what time this book will be available, searching of books using ISBN number etc. To eliminate this manual system, a library management system has been developed. Library Management System will handle all the current issues faced by the students and by its admin personnel.

Core Features

- Searching of books
- Issuing and returning books
- Check fines(if any)
- Librarian can read information about any member
- Librarian can track the books issued by a particular student
- A librarian can add/remove any member(student).
- A librarian can add/delete books
- Librarian can update the availability status of the books

Additional Features

Admin Dashboard deals with the following :

1) Log in to Admin Dashboard:

2) check all issues :

   ➢see issues, delete issues

   ➢search issues by student id

   ➢filter issues based on: issued or not, returned or not.

3) accept an issue:

   ➢from the dashboard where the admin has to manually select a return date or

   ➢from the Issue requests page where the return date is automatically calculated.

4) add, delete search books and filter books based on the author

5) add, delete, search author

6) calculates fine by clicking a button,

7) create, delete fine, search fines for student id

8) toggle fine paid status (if paid in cash)

9) search, modify, add, delete students, filter them based on department and check all fines and issues of that student

10) can see the last login, date joined & the student associated with a particular user

can change the password for any user

Students can

11) login/signup,

12)can request book

13) see their own issues and filter them based on:

➢ requested issues,

➢issued books or

➢ all of them together

14)check their own fines

15) can see

➢ the days remaining to return a particular book or

➢ the number of days passed the return date of a particular book on my fines page

16) pay their fines online

Anyone can see

➢all the books on the homepage

➢search books based on author or name of the book or category of the book

➢sort books or author alphabetically

Functional Requirements

- The system must only allow users with valid IDs and passwords to enter the system.
- The user must be able to log out after they finished using the system.
- The system must be able to not allow two books having the same book id.
- The system must be able to search if the book is available or not before issuing books.
- Admin can be able to see the availability of the particular book, they can also be able to see each user data (ie. Which book is issued to which user and the fine amount of the user).

Data requirements

- There will be the data of each book in the database.
- There will be the user name and password of each member and faculty in the database

- The record of the issued books will also be in the database.
- The record of all the members working in the library will be there.

Student app
We need this for writing our authentication views as well as student & department models. The student model has the first+last name, department foreign key and student ID which is a one-to-one field to Django' User model. We use Django's User Model for authentication. There will be 3 views: login, signup and logout. The URLs will have /student/< login or signup or logout >/.

Library app
This is our main app where we will write our library system's main logic. It comprises 4 models:

Author - for storing name & description of the author
Book - for storing name, image, category of a book & connecting to the author
Issue - for tracking each & every issue a student requests. It will also track the book for which issue is requested, issue status (whether issued or not), return status (whether returned or not), return date (the last date to return the book) and more...
Fine - for tracking the fines & calculating fines automatically for each student whose issued book/s is/are not returned and the last date is passed.

Some important logic:

Student ID - Username of Django's User Model serves as our student ID

Signing Up - so every student who signs up creates a new user instance with his/her student id as the username and then a student instance is also created with the names and department and this user we just created.

Calculating Fine - How ?? -
We run a for loop and pass all the issues to this calculate fine function.
Then:
➧ for each issue, check whether the issue is issued or not (if the issue is not issued then no need to calculate fines)
➢ if an issue is issued then check whether the issue is returned or not (if the issue is returned then no need to calculate fines)
➢ if the issue is not returned then check whether the issue's return date is passed or not (if not passed then no calculation of fines is needed)

➢create or get a fine instance with student & issue then calculate the amount and save it to the amount field

Calculating Fine - When ?? -
➢Whenever admin clicks on the "Calculate Fine" button
➢Whenever a student opens his "My Fines" page

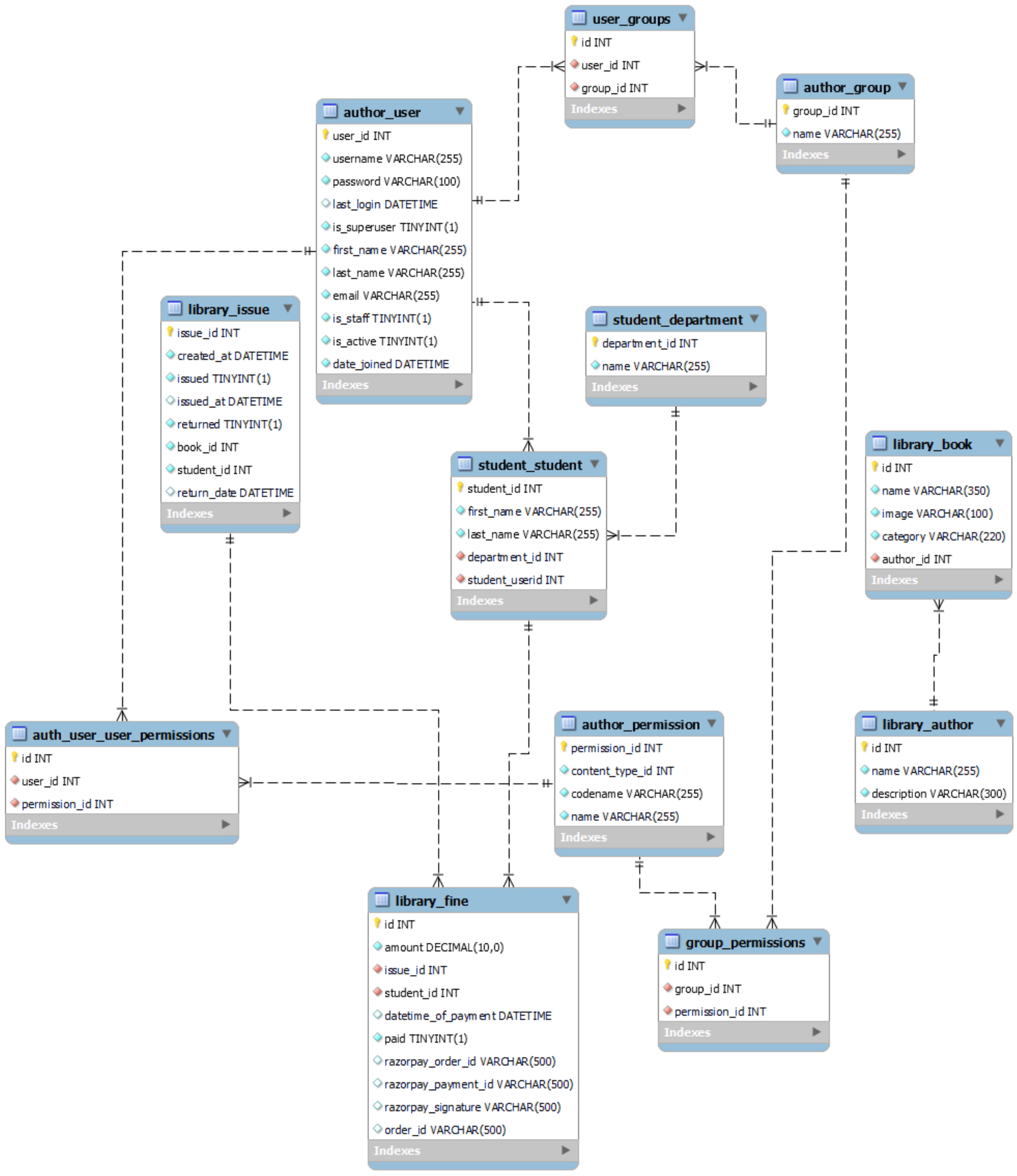
Payment of Fines:

➢ when a student clicks on the pay button (on my fines page)
➢we create a Razorpay order with a dict containing fine amount (converted to int and multiplied by 100 because Razorpay wants in paisa), order_id, currency
➢then we send the user to the pay fines page (payfines.html) with the amount (in paisa) , Razorpay key-id, Razorpay order id & amount (which should be displayed)
➢ The user chooses to proceed to payment online, selects pay mode (Netbanking, Card, Wallet etc.) and pays the amount
➢we verify the payment status whether success or failure
➢then payment is (successful/failure) message is shown on my fine page with (paid status/pay button) beside that fine

Software and Languages required

asgiref==3.3.1
certifi==2020.12.5
chardet==4.0.0
Django==3.1.7
django-environ==0.4.5
idna==2.10
Naked==0.1.31
paytmchecksum==1.7.0
Pillow==8.1.2
pycryptodome==3.10.1
pytz==2021.1
PyYAML==5.4.1
razorpay==1.2.0
requests==2.25.1
shellescape==3.8.1
sqlparse==0.4.1
urllib3==1.26.4

Entity-Relationship (ER) Diagram:


user_groups
id INT
user_id INT
group_id INT
Indexes
author_group
group_id INT
name VARCHAR(255)
Indexes
author_user
user_id INT
username VARCHAR(255)
password VARCHAR(100)
last_login DATETIME
is_superuser TINYINT(1)
first_name VARCHAR(255)
last_name VARCHAR(255)
email VARCHAR(255)
is_staff TINYINT(1)
is_active TINYINT(1)
date_joined DATETIME
Indexes
library_issue
issue_id INT
created_at DATETIME
issued TINYINT(1)
issued_at DATETIME
returned TINYINT(1)
book_id INT
student_id INT
return_date DATETIME
Indexes
student_department
department_id INT
name VARCHAR(255)
Indexes
student_student
student_id INT
first_name VARCHAR(255)
last_name VARCHAR(255)
department_id INT
student_userid INT
Indexes
library_book
id INT
name VARCHAR(350)
image VARCHAR(100)
category VARCHAR(220)
author_id INT
Indexes
auth_user_user_permissions
id INT
user_id INT
permission_id INT
Indexes
author_permission
permission_id INT
content_type_id INT
codename VARCHAR(255)
name VARCHAR(255)
Indexes
library_author
id INT
name VARCHAR(255)
description VARCHAR(300)
Indexes
library_fine
id INT
amount DECIMAL(10,0)
issue_id INT
student_id INT
datetime_of_payment DATETIME
paid TINYINT(1)
razorpay_order_id VARCHAR(500)
razorpay_payment_id VARCHAR(500)
razorpay_signature VARCHAR(500)
order_id VARCHAR(500)
Indexes
group_permissions
id INT
group_id INT
permission_id INT
Indexes