

# UNIVERSITY LIBRARY DATABASE SYSTEM PROPOSAL

*Group Members:*

Abdulwadud

Usman-Inenemo,

Alamin Adeleke,

Chukwunonso

Daniel Ekweaga,

Emmanuel

Adeyemi-Kings,

Marvellous Akinola





---

## SYSTEM PURPOSE & KEY GOALS

- Build a reliable database to manage library resources and users.
- Track book borrowing, returns, and fines.
- Simplify user management and enhance efficiency.
- Enable quick search and generate useful reports.

---

# SYSTEM ENTITIES & RELATIONSHIPS

## Entities:

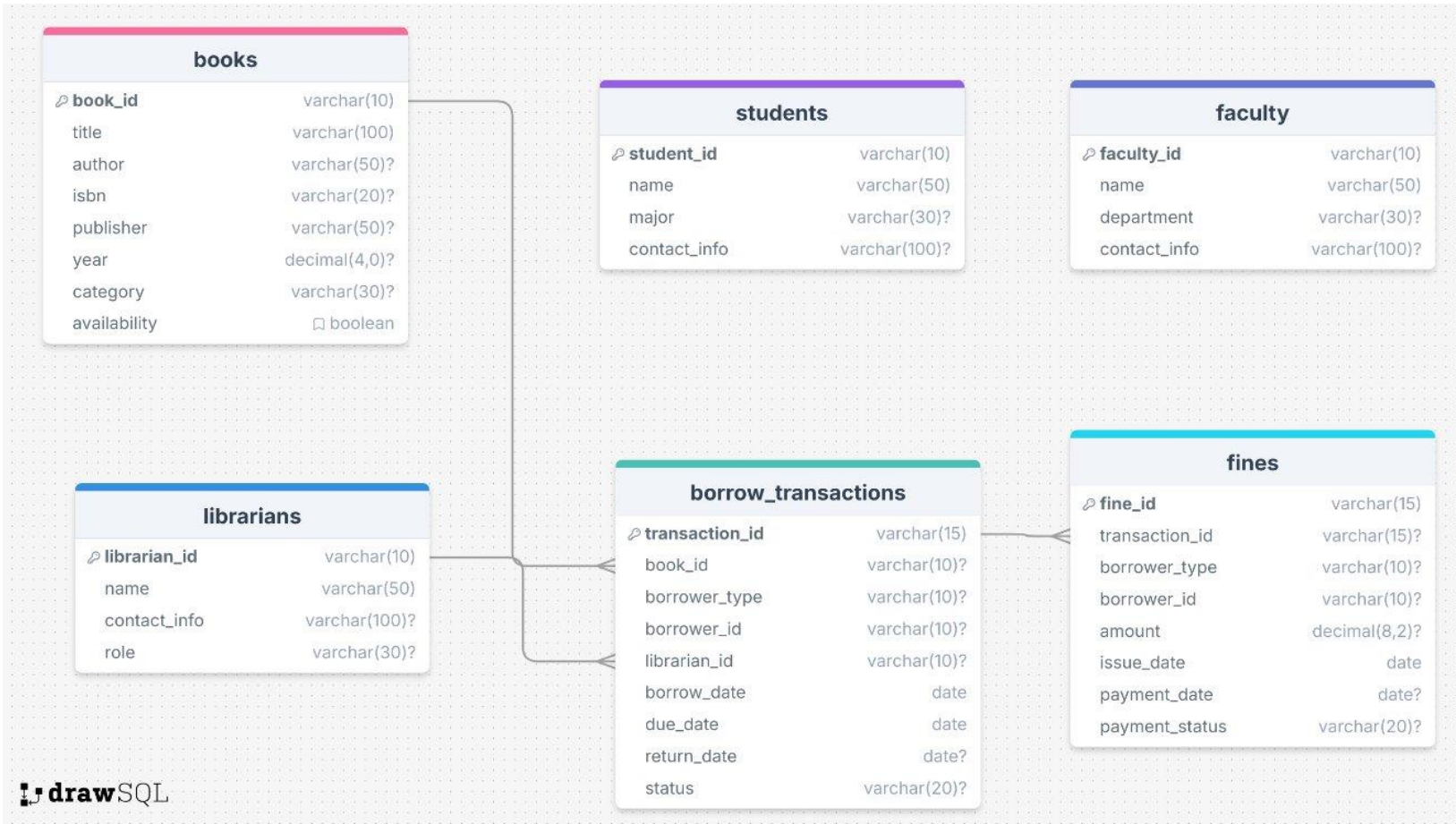
- Books, Students, Faculty, Librarians, Borrow\_Transactions, Fines

## Relationships:

- A student or faculty member can borrow many books.
- A book can only be borrowed by one user at a time.
- Borrow transactions connect users, books, and librarians.
- Fines are linked to specific borrow transactions.



# ENTITY RELATIONSHIP DIAGRAM







---

# BOOK MANAGEMENT FEATURES

- Add new books with full bibliographic data
- Edit or remove outdated or damaged books
- Update availability in real-time upon borrow/return
- Categorize books by genre, year, or publisher

---

# USER MANAGEMENT FEATURES

1

Register and  
update student  
and faculty  
profiles

2

Associate  
borrowing  
privileges and  
limits

3

Track user  
borrowing  
history

4

Distinguish  
between student  
and faculty user  
types

---

# BORROW & RETURN SYSTEM



Enable borrowing with due date assignment



Return process updates book availability



Record timestamps of transactions



Ensure borrowing policy compliance

---

# FINE MANAGEMENT SYSTEM



Detect overdue books automatically



Calculate fines based on delay duration



Update payment status (paid/unpaid)

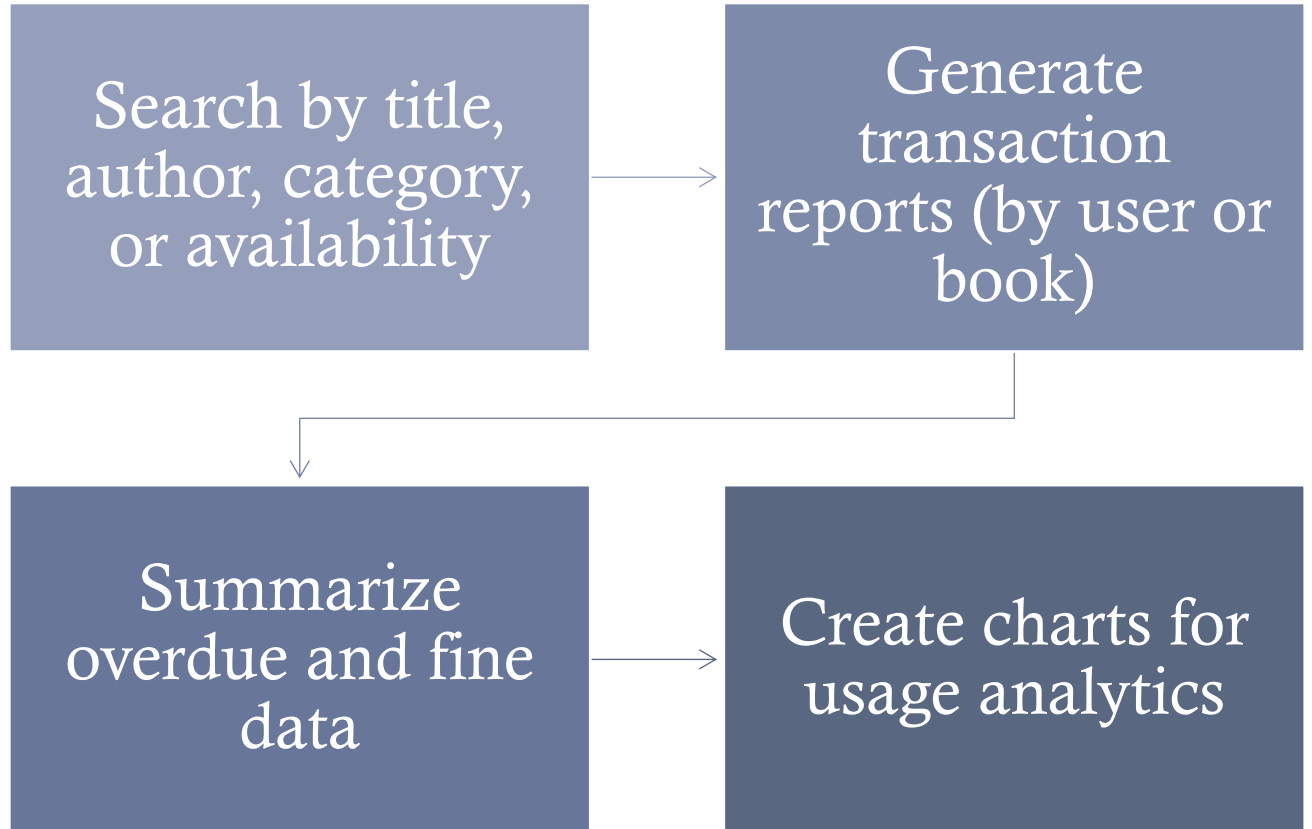


Generate fine summaries per user



---

# SEARCH & REPORTING FEATURES



---

# TECHNICAL IMPLEMENTATION

DBMS: MySQL

Programming Language: Java

Connection Library: JDBC

IDE & Tools: BlueJ , GitHub for  
version control

---

# GITHUB REPOSITORY

Repository:

<https://github.com/danekweaga/Database-Project>

Will include:

- Java source code
- SQL DDL scripts
- Sample data and inserts
- ER diagram
- Documentation and reports



---

# SCHEMA Q&A - STRUCTURE

1. What tables exist in the database?

books, students, faculty, librarians,  
borrow\_transactions, fines

2. Columns in "books" table:

book\_id, title, author, isbn, publisher, year,  
category, availability

3. Primary key for "students" table:

student\_id



---

## SCHEMA Q&A - RELATIONSHIPS

### 4. Table Relationships:

`borrow_transactions.book_id → books.book_id`

`borrow_transactions.borrower_id → students.student_id or  
faculty.faculty_id`

`borrow_transactions.librarian_id → librarians.librarian_id`

`fines.transaction_id → borrow_transactions.transaction_id`

### 5. Data type for "year" in books table:

INT

---

---

# BOOK FUNCTIONALITY Q&A

## Retrieve

Retrieve books sorted by title:

- `SELECT * FROM books ORDER BY title;`

## Find

Find a specific book by ID:

- `SELECT * FROM books WHERE book_id = ?;`





---

# BOOK CRUD Q&A

- Information to add a book:

book\_id, title, author, isbn, publisher, year,  
category, availability

- How is availability tracked?

availability column (boolean)

- Update book availability:

```
UPDATE books SET availability = ? WHERE  
book_id = ?;
```



---

# FINAL SUMMARY

- A complete, scalable system to automate library tasks
  - Structured database with clear entity relationships
  - Real-time transaction handling with fine tracking
  - Strong technical foundation using Java and SQL
  - Clear path for implementation and collaboration on GitHub
-