**Project Title: Advanced Digital and Physical Library Management System**

by:

Shri Sudhan Vijayaraj 2310110293

Hemanth M S 2310110122

Tuhina Agarwal 2310110331

**Objective:**

The objective of this project is to design and develop a comprehensive full-stack Library Management System that efficiently manages both physical and digital resources, as well as study room bookings. This system aims to streamline key library workflows such as book issue and return, penalty tracking, digital engagement monitoring, and study room reservation. It is built to enhance accessibility, improve user experience, and empower administrators with real-time tracking, automated status updates, and robust user role management.

A diagram of a diagram of a library management system

AI-generated content may be incorrect.

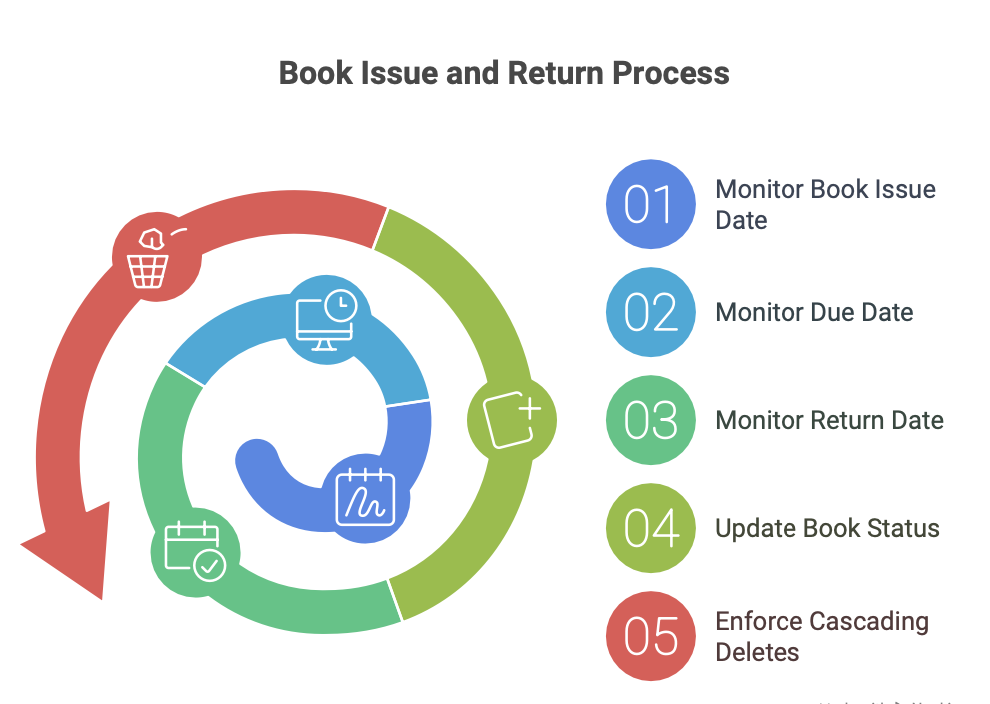
**Key Features:**

**1. Resource Management**

* + **Digital Resources:**
    - Store and manage digital content such as e-books, PDFs, and research articles.
    - Maintain metadata including file format, file size, access URL, subcategory, and downloadability.
    - Track member-wise digital interactions including views and downloads.
  + **Physical Resources:**
    - Maintain records of physical books including ISBN, title, author, genre, subcategory, and shelf location.
    - Track availability and allow registered members to reserve and borrow books.

**2. Book Issue and Return System**

* + Monitor and record book issue dates, due dates, return dates, and update status to 'Issued', 'Returned', or 'Overdue'.
  + Automatically enforce cascading deletes for dependent records when a member or staff is removed.



**3. Penalty Management**

* + Automatically generate penalty records for overdue book returns.
  + Track payment status of penalties categorized as 'Paid', 'Unpaid', or 'Waived'.
  + Maintain a history of penalties issued to each member for administrative oversight.

**4. Digital Engagement Tracking**

* + Log each interaction with digital resources, specifying the action (View or Download), date, time, and duration.
  + Enable analytical reporting for engagement metrics.

**5. Study Room Booking System**

* + Allow members to book study rooms by specifying room number and desired date.
  + Manage room capacity, location, and availability in real-time.
  + Update and track booking statuses as 'Booked', 'Cancelled', or 'Completed'.

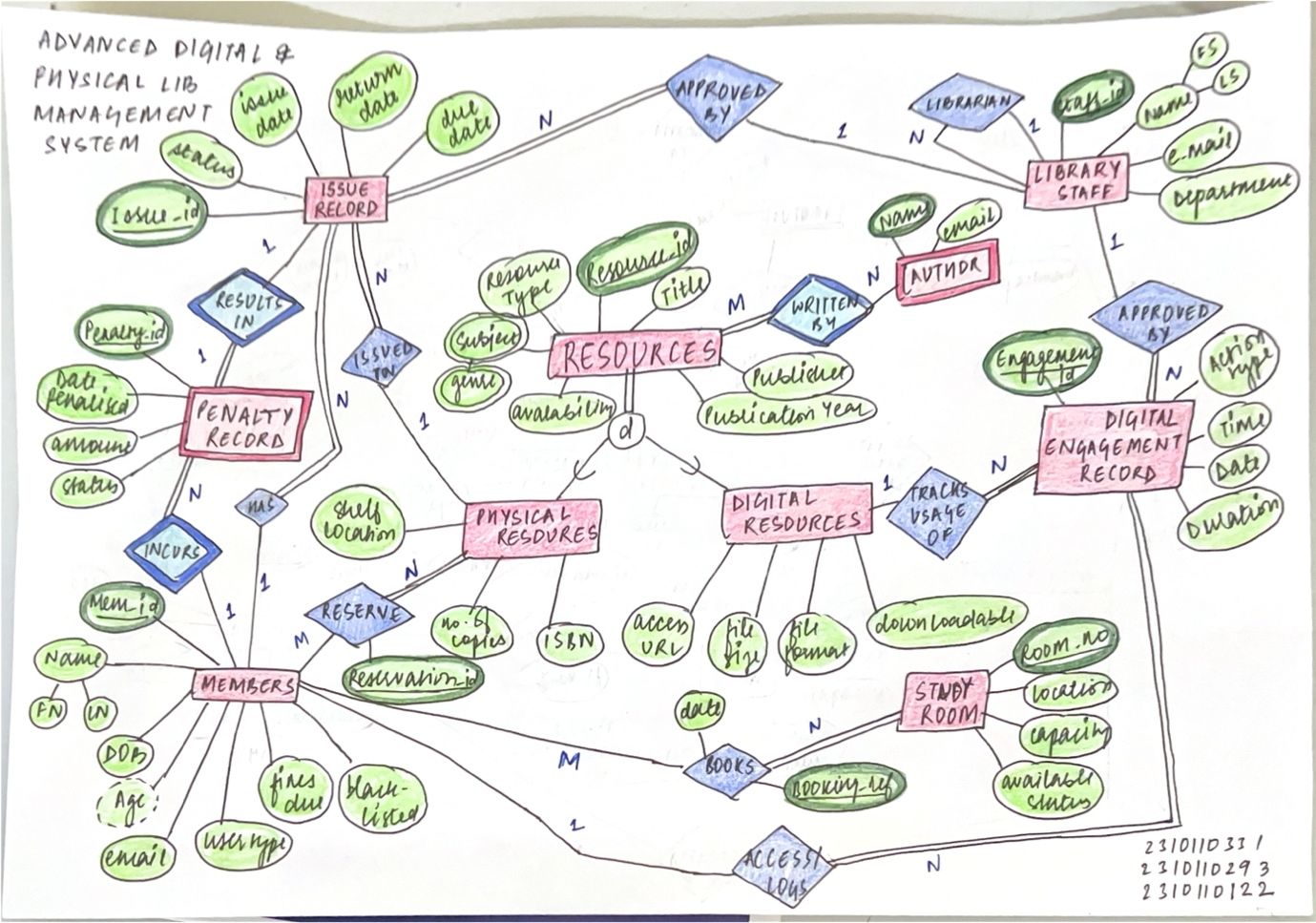
**6. User Roles and Access Control**

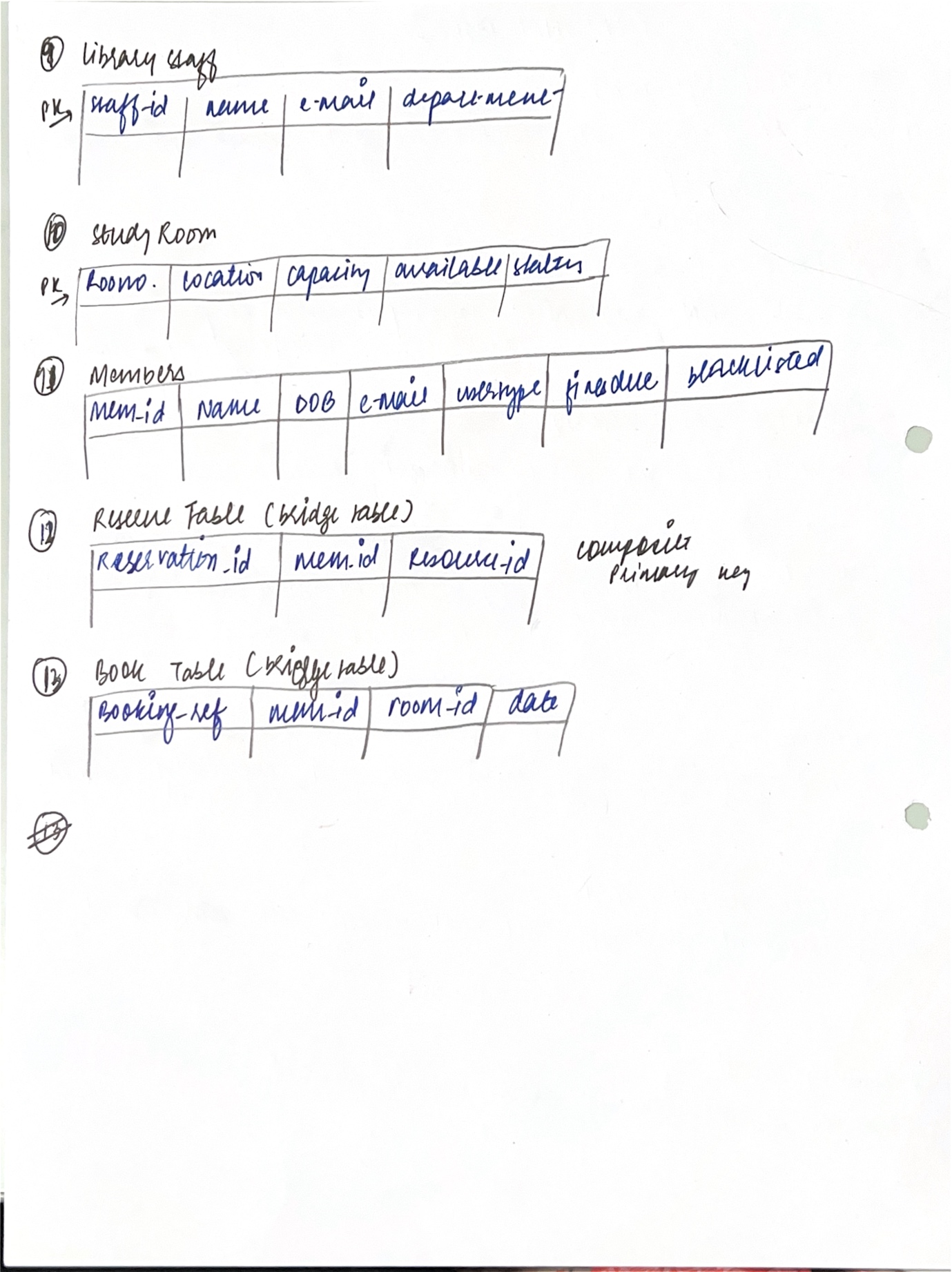
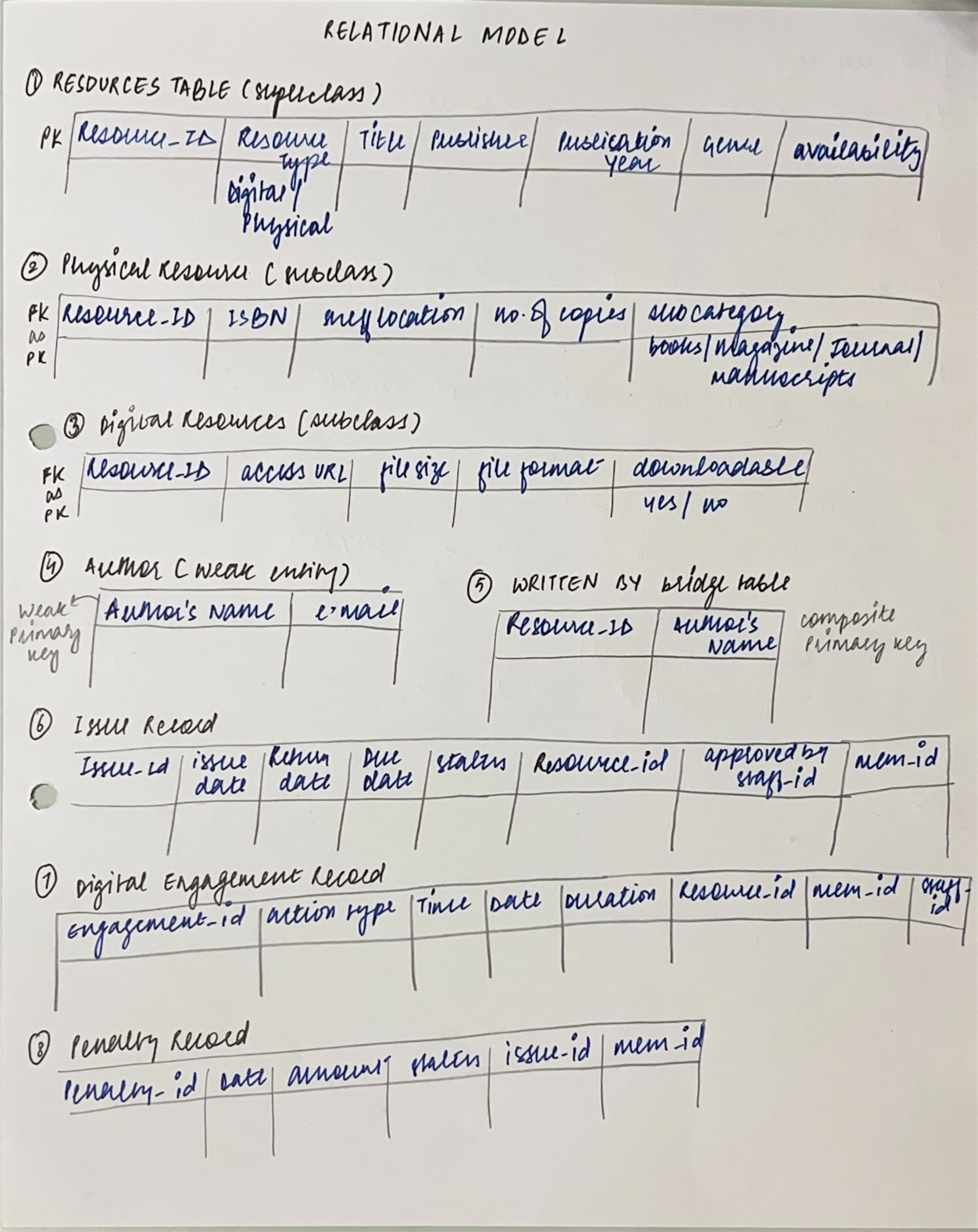
* + **Members (Students, Faculty, Guests):**
    - Borrow and renew books (renewal restricted if future reservations exist).
    - Access, view, and download digital content.
    - Reserve study rooms.
    - Search for resources using advanced filters.
    - Edit their profile information.
    - Submit book requests and check availability in real-time.
  + **Library Staff (Administrators):**
    - Manage issue records, penalties, and monitor digital engagement.
    - Add, update, or delete physical and digital resources.
    - Oversee member profiles and assign roles or permissions.
    - Handle book requests including acceptance and fulfilment.
    - Manage study room bookings and schedules.



This system is designed to offer a scalable, secure, and intuitive interface that meets the evolving needs of modern libraries, ensuring seamless access to educational resources and efficient library operations.

**ER MODEL**



**RELATIONAL MODEL** 

**DATABASE SCHEMA**  
  
CREATE TABLE DigitalResource (

    Resource\_ID VARCHAR(7) PRIMARY KEY,

    Title VARCHAR(255) NOT NULL,

    Publication VARCHAR(255),

    Published\_Year INT,

    Genre VARCHAR(100),

    Author varchar(30),

    Availability BOOLEAN DEFAULT TRUE,

    Subcategory VARCHAR(25),

    Access\_URL VARCHAR(500) NOT NULL UNIQUE,

    File\_Size DECIMAL(10,2), -- Size in MB or GB

    File\_Format VARCHAR(20),

    Downloadable BOOLEAN DEFAULT FALSE

);CREATE TABLE Book (

    ISBN VARCHAR(20) PRIMARY KEY,

    Title VARCHAR(255) NOT NULL,

    Publication VARCHAR(255),

    Published\_Year INT,

    Genre VARCHAR(100),

    Subcategory VARCHAR(25)

);

CREATE TABLE PhysicalCopy (

    Resource\_ID VARCHAR(7) PRIMARY KEY,

    ISBN VARCHAR(20),

    Author varchar(30),

    Shelf\_Location VARCHAR(50),

    Availability BOOLEAN DEFAULT TRUE,

    FOREIGN KEY (ISBN) REFERENCES Book(ISBN)

);

CREATE TABLE IssueRecord (

    Issue\_ID INT PRIMARY KEY AUTO\_INCREMENT, -- Unique issue record

    Issue\_Date DATE NOT NULL, -- Date when the resource is issued

    Due\_Date DATE NOT NULL, -- Due date for return

    Return\_Date DATE, -- Date when the resource is returned (nullable)

    Status VARCHAR(20) CHECK (Status IN ('Issued', '-Returned', 'Overdue')), -- Issue status

    Resource\_ID VARCHAR(7), -- References PhysicalResource table

    Member\_ID VARCHAR(10), -- References Members table

    Staff\_ID VARCHAR(10), -- References LibraryEmployee table

    -- Foreign Key Constraints

    FOREIGN KEY (Resource\_ID) REFERENCES PhysicalCopy(Resource\_ID) ON DELETE CASCADE,

    FOREIGN KEY (Member\_ID) REFERENCES Members(Member\_ID) ON DELETE CASCADE,

    FOREIGN KEY (Staff\_ID) REFERENCES LibraryEmployee(Staff\_ID) ON DELETE SET NULL

);

CREATE TABLE DigitalEngagementRecord (

    Engagement\_ID VARCHAR(10) PRIMARY KEY, -- Unique engagement record ID

    Action\_Type ENUM('Download', 'View') NOT NULL, -- Type of engagement

    Engagement\_Time TIME NOT NULL, -- Time of the engagement

    Engagement\_Date DATE NOT NULL, -- Date of the engagement

    Duration INT CHECK (Duration >= 0), -- Duration in seconds or minutes (nullable)

    Resource\_ID VARCHAR(7), -- References DigitalResource(Resource\_ID)

    Member\_ID VARCHAR(10), -- References Members(Member\_ID)

    Staff\_ID VARCHAR(10), -- References LibraryEmployee(Staff\_ID)

    -- Foreign Key Constraints

    FOREIGN KEY (Resource\_ID) REFERENCES DigitalResource(Resource\_ID) ON DELETE CASCADE,

    FOREIGN KEY (Member\_ID) REFERENCES Members(Member\_ID) ON DELETE CASCADE,

    FOREIGN KEY (Staff\_ID) REFERENCES LibraryEmployee(Staff\_ID) ON DELETE SET NULL

);

CREATE TABLE PenaltyRecord (

    Penalty\_ID INT PRIMARY KEY AUTO\_INCREMENT, -- Unique penalty record ID

    Penalty\_Date DATE NOT NULL, -- Date when penalty is recorded

    Amount\_Due DECIMAL(10,2) CHECK (Amount\_Due >= 0), -- Penalty amount (ensuring non-negative)

    Status VARCHAR(20) CHECK (Status IN ('Unpaid', 'Paid', 'Waived')), -- Payment status

    Issue\_ID INT, -- Reference to IssueRecord table

    Member\_ID VARCHAR(10), -- Reference to Members table

    -- Foreign Key Constraints

    FOREIGN KEY (Issue\_ID) REFERENCES IssueRecord(Issue\_ID) ON DELETE CASCADE,

    FOREIGN KEY (Member\_ID) REFERENCES Members(Member\_ID) ON DELETE CASCADE

);

CREATE TABLE LibraryEmployee (

    Staff\_ID VARCHAR(10) PRIMARY KEY, -- Unique Staff ID

    Name VARCHAR(100) NOT NULL, -- Employee name

    Email VARCHAR(255) UNIQUE NOT NULL, -- Unique Email (Candidate Key)

    Department VARCHAR(30) NOT NULL -- Department of employee

);

CREATE TABLE StudyRoom (

    Room\_No VARCHAR(10) PRIMARY KEY, -- Unique Room Number

    Location VARCHAR(100) NOT NULL, -- Location of the study room

    Capacity INT CHECK (Capacity > 0) -- Ensures capacity is positive

);

CREATE TABLE Members (

    Member\_ID VARCHAR(10) PRIMARY KEY, -- Unique Member ID

    Name VARCHAR(100) NOT NULL, -- Full Name

    DOB DATE NOT NULL, -- Date of Birth

    Email VARCHAR(255) UNIQUE NOT NULL, -- Unique Email for each member

    UserType ENUM('Student', 'Faculty', 'Guest') NOT NULL, -- Type of user

    Fines\_Due DECIMAL(10,2) DEFAULT 0 CHECK (Fines\_Due >= 0), -- Total fines due

    Blacklisted BOOLEAN DEFAULT FALSE -- Whether the member is blacklisted

);

CREATE TABLE ReserveBook (

    Reservation\_ID INT PRIMARY KEY AUTO\_INCREMENT, -- Unique reservation ID

    Member\_ID VARCHAR(10), -- References Members table

    Resource\_ID VARCHAR(7), -- References PhysicalCopy table

    Reservation\_Date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, -- Auto-set reservation timestamp

    Status ENUM('Reserved', 'Cancelled', 'Fulfilled') DEFAULT 'Reserved', -- Reservation status

    -- Foreign Key Constraints

    FOREIGN KEY (Member\_ID) REFERENCES Members(Member\_ID) ON DELETE CASCADE,

    FOREIGN KEY (Resource\_ID) REFERENCES PhysicalCopy(Resource\_ID) ON DELETE CASCADE

);

CREATE TABLE BookRoom (

    Booking\_Ref INT PRIMARY KEY AUTO\_INCREMENT, -- Unique booking reference

    Member\_ID VARCHAR(10), -- References Members table

    Room\_ID VARCHAR(10), -- References StudyRoom table

    Booking\_Date DATE NOT NULL, -- Date of booking

    Status ENUM('Booked', 'Cancelled', 'Completed') DEFAULT 'Booked', -- Booking status

    -- Foreign Key Constraints

    FOREIGN KEY (Member\_ID) REFERENCES Members(Member\_ID) ON DELETE CASCADE,

    FOREIGN KEY (Room\_ID) REFERENCES StudyRoom(Room\_No) ON DELETE CASCADE

);

**Functional Dependencies and BCNF Normalization**

Initially, we had a **PhysicalResource** table similar to **DigitalResource**, which combined attributes of the current **Book** and **PhysicalCopy** tables. This structure led to transitive dependencies:

**ISBN** → {**Title, Publication, Published\_Year, Genre, Subcategory**} (Book attributes)

Since **ISBN** determined all book attributes---including **Title, Publication, Published\_Year, Genre, and Subcategory**---storing these in the **PhysicalResource** table created redundancy and a transitive dependency.

To eliminate this transitive dependency and achieve BCNF compliance, we decomposed the table into **Book** (storing book details) and **PhysicalCopy** (storing physical copies with shelf location and availability). This ensured a lossless decomposition while maintaining data integrity and preserving dependencies.

**1. Book Table**

**Primary Key:** ISBN

**Functional Dependencies:**

ISBN → {Title, Publication, Published\_Year, Genre, Subcategory}

**Normalization Analysis:**

**1NF:** Ensures atomicity of attributes.

**2NF:** No partial dependency since ISBN is the only key.

**3NF:** No transitive dependencies.

**BCNF:** Every determinant is a candidate key.

**Highest Normal Form:** BCNF

**2. PhysicalCopy Table**

**Primary Key:** Resource\_ID

**Foreign Key:** ISBN references Book(ISBN)

**Functional Dependencies:**

Resource\_ID → {ISBN, Shelf\_Location, No\_of\_Copies, Availability}

ISBN → {Title, Publication, Published\_Year, Genre, Subcategory} (from Book table)

**Normalization Analysis:**

**1NF:** Attributes are atomic.

**2NF:** No partial dependencies.

**3NF:** No transitive dependencies.

**BCNF:** Every determinant is a candidate key.

**Highest Normal Form:** BCNF

**3. DigitalResource Table**

**Primary Key:** Resource\_ID

**Candidate Key(s):** Access\_URL (unique constraint)

**Functional Dependencies:**

Resource\_ID → {Title, Publication, Published\_Year, Genre, Author, Availability, Subcategory, Access\_URL, File\_Size, File\_Format, Downloadable}

Access\_URL → {Title, Publication, Published\_Year, Genre, Author, Availability, Subcategory, File\_Size, File\_Format, Downloadable}

**Normalization Analysis:**

**1NF:** Ensures atomicity.

**2NF:** No partial dependency.

**3NF:** No transitive dependency.

**BCNF:** Every determinant is a candidate key.

**Highest Normal Form:** BCNF

**4. IssueRecord Table**

**Primary Key:** Issue\_ID

**Foreign Keys:** Resource\_ID (PhysicalCopy), Member\_ID (Members), Staff\_ID (LibraryEmployee)

**Functional Dependencies:**

Issue\_ID → {Issue\_Date, Due\_Date, Return\_Date, Status, Resource\_ID, Member\_ID, Staff\_ID}

**Normalization Analysis:**

**1NF:** Attributes are atomic.

**2NF:** No partial dependency.

**3NF:** No transitive dependency.

**BCNF:** Every determinant is a candidate key.

**Highest Normal Form:** BCNF

**5. DigitalEngagementRecord Table**

**Primary Key:** Engagement\_ID

**Foreign Keys:** Resource\_ID (DigitalResource), Member\_ID (Members), Staff\_ID (LibraryEmployee)

**Functional Dependencies:**

Engagement\_ID → {Action\_Type, Engagement\_Time, Engagement\_Date, Duration, Resource\_ID, Member\_ID, Staff\_ID}

**Normalization Analysis:**

**1NF:** Attributes are atomic.

**2NF:** No partial dependency.

**3NF:** No transitive dependency.

**BCNF:** Every determinant is a candidate key.

**Highest Normal Form:** BCNF

**6. PenaltyRecord Table**

**Primary Key:** Penalty\_ID

**Foreign Keys:** Issue\_ID (IssueRecord), Member\_ID (Members)

**Functional Dependencies:**

Penalty\_ID → {Penalty\_Date, Amount\_Due, Status, Issue\_ID, Member\_ID}

**Normalization Analysis:**

**1NF:** Attributes are atomic.

**2NF:** No partial dependency.

**3NF:** No transitive dependency.

**BCNF:** Every determinant is a candidate key.

**Highest Normal Form:** BCNF

**7. ReserveBook Table**

**Primary Key:** Reservation\_ID

**Foreign Keys:** Member\_ID (Members), Resource\_ID (PhysicalCopy)

**Functional Dependencies:**

Reservation\_ID → {Member\_ID, Resource\_ID, Reservation\_Date, Status}

**Normalization Analysis:**

**1NF:** Ensures atomicity.

**2NF:** With a single-attribute primary key, no partial dependencies exist.

**3NF:** No transitive dependencies among non-key attributes.

**BCNF:** Every determinant is a candidate key.

**Highest Normal Form:** BCNF

**8. BookRoom Table**

**Primary Key:** Booking\_Ref

**Foreign Keys:** Member\_ID (Members), Room\_ID (StudyRoom)

**Functional Dependencies:**

Booking\_Ref → {Member\_ID, Room\_ID, Booking\_Date, Status}

**Normalization Analysis:**

**1NF:** Attributes are atomic.

**2NF:** No partial dependency.

**3NF:** No transitive dependency.

**BCNF:** Every determinant is a candidate key.

**Highest Normal Form:** BCNF

**9. LibraryEmployee Table**

**Primary Key:** Staff\_ID

**Candidate Key(s):** Email (Unique Constraint)

**Functional Dependencies:**

Staff\_ID → {Name, Email, Department}

Email → {Name, Staff\_ID, Department}

**Normalization Analysis:**

**1NF:** Ensures atomicity.

**2NF:** No partial dependency.

**3NF:** No transitive dependency.

**BCNF:** Every determinant is a candidate key.

**Highest Normal Form:** BCNF

**10. StudyRoom Table**

**Primary Key:** Room\_No

**Functional Dependencies:**

Room\_No → {Location, Capacity}

**Normalization Analysis:**

**1NF:** Attributes are atomic.

**2NF:** No partial dependency.

**3NF:** No transitive dependency.

**BCNF:** Every determinant is a candidate key.

**Highest Normal Form:** BCNF

**11. Members Table**

**Primary Key:** Member\_ID

**Candidate Key(s):** Email (Unique Constraint)

**Functional Dependencies:**

Member\_ID → {Name, DOB, Email, UserType, Fines\_Due, Blacklisted}

Email → {Member\_ID, Name, DOB, UserType, Fines\_Due, Blacklisted}

**Normalization Analysis:**

**1NF:** Attributes are atomic.

**2NF:** With a single-attribute primary key, no partial dependencies exist.

**3NF:** No transitive dependencies among non-key attributes.

**BCNF:** Every determinant is a candidate key.

**Highest Normal Form:** BCNF

**SQL COMMANDS**

# Library Management System - Database Documentation

## 1. Database Configuration

The project is configured to use SQLite3 by default, but the models are compatible with MySQL. The database configuration is defined in `settings.py`:

```python

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),

}

}

```

To use MySQL, you would modify this configuration to:

```python

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'library\_db',

'USER': 'your\_mysql\_user',

'PASSWORD': 'your\_mysql\_password',

'HOST': 'localhost',

'PORT': '3306',

}

}

```

## 2. Data Models and Relationships

### 2.1 User Management

#### Student Model

```python

class Student(models.Model):

user = models.OneToOneField(User, *on\_delete*=models.CASCADE)

phone\_no = models.CharField(*max\_length*=15)

department = models.CharField(*max\_length*=400)

roll\_number = models.CharField(*max\_length*=10)

registered\_id = models.CharField(*max\_length*=10)

college\_name = models.CharField(*max\_length*=100)

```

- One-to-One relationship with Django's built-in User model

- Stores additional student information

### 2.2 Book Management

#### Book Model

```python

class Book(models.Model):

book\_name = models.CharField(*max\_length*=100)

author = models.CharField(*max\_length*=100)

book\_id = models.CharField(*max\_length*=20, *unique*=True)

description = models.TextField()

book\_image = models.ImageField(*upload\_to*='book\_images/')

available\_copies = models.PositiveIntegerField(*default*=1)

```

- Core model for book information

- Tracks book availability

### 2.3 Book Borrowing System

#### BookRequest Model

```python

class BookRequest(models.Model):

user = models.ForeignKey(User, *on\_delete*=models.CASCADE)

book = models.ForeignKey(Book, *on\_delete*=models.CASCADE)

request\_date = models.DateTimeField(*default*=timezone.now)

```

- Handles book borrowing requests

- Many-to-One relationships with User and Book

#### AcceptedBookRequest Model

```python

class AcceptedBookRequest(models.Model):

user = models.ForeignKey(User, *on\_delete*=models.CASCADE)

book = models.ForeignKey(Book, *on\_delete*=models.CASCADE)

details = models.ForeignKey(BookRequest, *on\_delete*=models.CASCADE)

accepted\_date = models.DateTimeField(*auto\_now\_add*=True)

fine = models.DecimalField(*max\_digits*=10, *decimal\_places*=2)

return\_date = models.DateTimeField()

is\_returned = models.BooleanField(*default*=False)

returned\_date = models.DateTimeField(*null*=True)

```

- Tracks approved book requests

- Handles book returns and fines

### 2.4 Study Room Management

#### StudyRoom Model

```python

class StudyRoom(models.Model):

room\_id = models.CharField(*max\_length*=10, *unique*=True)

room\_name = models.CharField(*max\_length*=100)

room\_capacity = models.IntegerField()

description = models.TextField()

```

#### RoomBooking Model

```python

class RoomBooking(models.Model):

user = models.ForeignKey(User, *on\_delete*=models.CASCADE)

room = models.ForeignKey(StudyRoom, *on\_delete*=models.CASCADE)

booking\_date = models.DateField()

status = models.CharField(*max\_length*=10)

```

### 2.5 Digital Resources

#### DigitalResource Model

```python

class DigitalResource(models.Model):

resource\_number = models.AutoField(*primary\_key*=True)

name = models.CharField(*max\_length*=200)

author = models.CharField(*max\_length*=200)

type = models.CharField(*max\_length*=20)

file = models.FileField(*upload\_to*='digital\_resources/')

```

#### DigitalEngagementRecord Model

```python

class DigitalEngagementRecord(models.Model):

user = models.ForeignKey(User, *on\_delete*=models.CASCADE)

resource = models.ForeignKey(DigitalResource, *on\_delete*=models.CASCADE)

download\_date = models.DateTimeField(*auto\_now\_add*=True)

```

## 3. Key Database Operations

### 3.1 Book Management

```python

*# Adding a new book*

Book.objects.create(

*book\_name*="Sample Book",

*author*="Author Name",

*book\_id*="BK001",

*available\_copies*=5

)

*# Updating book availability*

book.available\_copies -= 1

book.save()

```

### 3.2 Book Borrowing

```python

*# Creating a book request*

BookRequest.objects.create(

*user*=request.user,

*book*=book,

*request\_date*=timezone.now()

)

*# Accepting a book request*

AcceptedBookRequest.objects.create(

*user*=book\_request.user,

*book*=book\_request.book,

*details*=book\_request,

*return\_date*=timezone.now() + timedelta(*days*=14)

)

```

### 3.3 Fine Calculation

```python

def calculate\_fine(*self*):

*if* not self.is\_returned:

current\_date = timezone.now()

*if* current\_date > self.return\_date:

days\_late = (current\_date - self.return\_date).days

self.fine = days\_late \* 10 *# ₹10 per day*

self.save()

*return* self.fine

```

## 4. Database Schema and Relationships

### 4.1 Key Relationships

- User ←→ Student (One-to-One)

- User ←→ BookRequest (One-to-Many)

- Book ←→ BookRequest (One-to-Many)

- BookRequest ←→ AcceptedBookRequest (One-to-One)

- User ←→ RoomBooking (One-to-Many)

- StudyRoom ←→ RoomBooking (One-to-Many)

- User ←→ DigitalEngagementRecord (One-to-Many)

- DigitalResource ←→ DigitalEngagementRecord (One-to-Many)

### 4.2 Constraints and Indexes

- Unique constraints: `book\_id`, `room\_id`, `[room, booking\_date]`

- Foreign key constraints with CASCADE deletion

- Auto-incrementing fields: `resource\_number`, `download\_number`

- Default ordering by dates in relevant models

## 5. Important Queries and Operations

### 5.1 Book Availability

```python

*# Check available books*

available\_books = Book.objects.filter(*available\_copies\_\_gt*=0)

*# Check borrowed books for a user*

borrowed\_books = AcceptedBookRequest.objects.filter(

*user*=user,

*is\_returned*=False

)

```

### 5.2 Room Booking

```python

*# Check room availability*

available\_rooms = StudyRoom.objects.exclude(

*roombooking\_\_booking\_date*=date,

*roombooking\_\_status*='approved'

)

```

### 5.3 Digital Resource Access

```python

*# Track resource downloads*

DigitalEngagementRecord.objects.create(

*user*=user,

*resource*=digital\_resource,

*ip\_address*=request.META.get('REMOTE\_ADDR')

)

```

### 5.4 Fine Management

```python

*# Get users with pending fines*

users\_with\_fines = AcceptedBookRequest.objects.filter(

*is\_returned*=False,

*fine\_\_gt*=0

)

```

## 6. Database Maintenance

### 6.1 Regular Tasks

- Calculate fines for overdue books

- Update book availability status

- Clean up expired room bookings

- Track digital resource usage

### 6.2 Data Integrity

- Cascade deletion for related records

- Null handling for optional fields

- Unique constraints for critical fields

- Default values for required fields

**TRIGGERS, PROCEDURES, AND LIMITATIONS**

## 1. Implicit Triggers in Django Models

### 1.1 Book Availability Trigger

This trigger is implemented in the `AcceptedBookRequest` model through the `return\_book` method:

```python

def return\_book(*self*):

*if* not self.is\_returned:

self.is\_returned = True

self.returned\_date = timezone.now()

*if* self.book:

self.book.available\_copies += 1 *# Trigger: Automatically updates book availability*

self.book.save()

*elif* self.details:

self.details.book.available\_copies += 1

self.details.book.save()

self.save()

```

\*\*Purpose\*\*: Automatically updates book availability when a book is returned.

### 1.2 Fine Calculation Trigger

Implemented in the `AcceptedBookRequest` model:

```python

def calculate\_fine(*self*):

*if* not self.is\_returned:

current\_date = timezone.now()

*if* current\_date > self.return\_date:

days\_late = (current\_date - self.return\_date).days

self.fine = days\_late \* 10 *# ₹10 per day*

self.save()

*return* self.fine

```

\*\*Purpose\*\*: Automatically calculates fines for overdue books.

### 1.3 Save Method Trigger

In the `AcceptedBookRequest` model:

```python

def save(*self*, \**args*, \*\**kwargs*):

*# Trigger: Auto-populate user and book from details*

*if* self.details and not self.user and not self.book:

self.user = self.details.user

self.book = self.details.book

super().save(\*args, \*\*kwargs)

```

\*\*Purpose\*\*: Automatically populates user and book information when saving an accepted request.

## 2. Database Procedures (Django Methods)

### 2.1 Book Request Processing

```python

class BookRequest(models.Model):

@classmethod

def create\_request(*cls*, *user*, *book*):

"""

Procedure to create a new book request with validation

"""

*if* book.available\_copies > 0:

request = cls.objects.create(

*user*=user,

*book*=book,

*request\_date*=timezone.now()

)

*return* request

*return* None

@classmethod

def approve\_request(*cls*, *request\_id*):

"""

Procedure to approve a book request

"""

*try*:

request = cls.objects.get(*id*=request\_id)

*if* request.book.available\_copies > 0:

accepted = AcceptedBookRequest.objects.create(

*user*=request.user,

*book*=request.book,

*details*=request,

*return\_date*=timezone.now() + timedelta(*days*=14)

)

request.book.available\_copies -= 1

request.book.save()

*return* accepted

*except* cls.DoesNotExist:

*return* None

```

### 2.2 Room Booking Procedure

```python

class RoomBooking(models.Model):

@classmethod

def book\_room(*cls*, *user*, *room*, *date*):

"""

Procedure to book a study room with validation

"""

*# Check if room is already booked*

existing\_booking = cls.objects.filter(

*room*=room,

*booking\_date*=date,

*status*='approved'

).exists()

*if* not existing\_booking:

booking = cls.objects.create(

*user*=user,

*room*=room,

*booking\_date*=date,

*status*='pending'

)

*return* booking

*return* None

```

## 3. Database Limitations and Constraints

### 3.1 Model Field Constraints

#### Book Model

```python

class Book(models.Model):

book\_id = models.CharField(*max\_length*=20, *unique*=True) *# Unique constraint*

available\_copies = models.PositiveIntegerField(*default*=1) *# Non-negative constraint*

```

#### StudyRoom Model

```python

class StudyRoom(models.Model):

room\_id = models.CharField(*max\_length*=10, *unique*=True) *# Unique constraint*

room\_capacity = models.IntegerField() *# Required field*

```

#### RoomBooking Model

```python

class RoomBooking(models.Model):

class Meta:

unique\_together = ['room', 'booking\_date'] *# Composite unique constraint*

```

### 3.2 Business Logic Constraints

1. \*\*Book Availability\*\*

- Books cannot have negative available copies

- Books cannot be borrowed if no copies are available

2. \*\*Fine System\*\*

- Fines are calculated at ₹10 per day

- Fines only apply to overdue books

- Fines are non-negative

3. \*\*Room Booking\*\*

- A room cannot be double-booked for the same date

- Room capacity cannot be exceeded

- Booking dates must be in the future

### 3.3 Data Integrity Constraints

1. \*\*Cascade Deletion\*\*

```python

user = models.ForeignKey(User, *on\_delete*=models.CASCADE)

book = models.ForeignKey(Book, *on\_delete*=models.CASCADE)

```

- When a user or book is deleted, all related records are automatically deleted

2. \*\*Null Handling\*\*

```python

returned\_date = models.DateTimeField(*null*=True, *blank*=True)

description = models.TextField(*null*=True, *blank*=True)

```

- Specific fields are allowed to be null based on business requirements

## 4. Recommended Additional Triggers and Procedures

### 4.1 Automated Fine Collection Trigger

```sql

CREATE TRIGGER calculate\_daily\_fines

AFTER UPDATE ON libapp\_acceptedbookrequest

FOR EACH ROW

BEGIN

IF NOT NEW.is\_returned AND NEW.return\_date < CURRENT\_TIMESTAMP THEN

SET NEW.fine = DATEDIFF(CURRENT\_TIMESTAMP, NEW.return\_date) \* 10;

END IF;

END;

```

### 4.2 Book Status Update Procedure

```sql

DELIMITER //

CREATE PROCEDURE update\_book\_status()

BEGIN

UPDATE libapp\_book b

SET available\_copies = (

SELECT COUNT(\*)

FROM libapp\_acceptedbookrequest ar

WHERE ar.book\_id = b.id AND ar.is\_returned = FALSE

);

END //

DELIMITER ;

```

### 4.3 Room Booking Validation Procedure

```sql

DELIMITER //

CREATE PROCEDURE validate\_room\_booking(

IN p\_room\_id INT,

IN p\_booking\_date DATE,

OUT p\_is\_available BOOLEAN

)

BEGIN

SELECT NOT EXISTS(

SELECT 1 FROM libapp\_roombooking

WHERE room\_id = p\_room\_id

AND booking\_date = p\_booking\_date

AND status = 'approved'

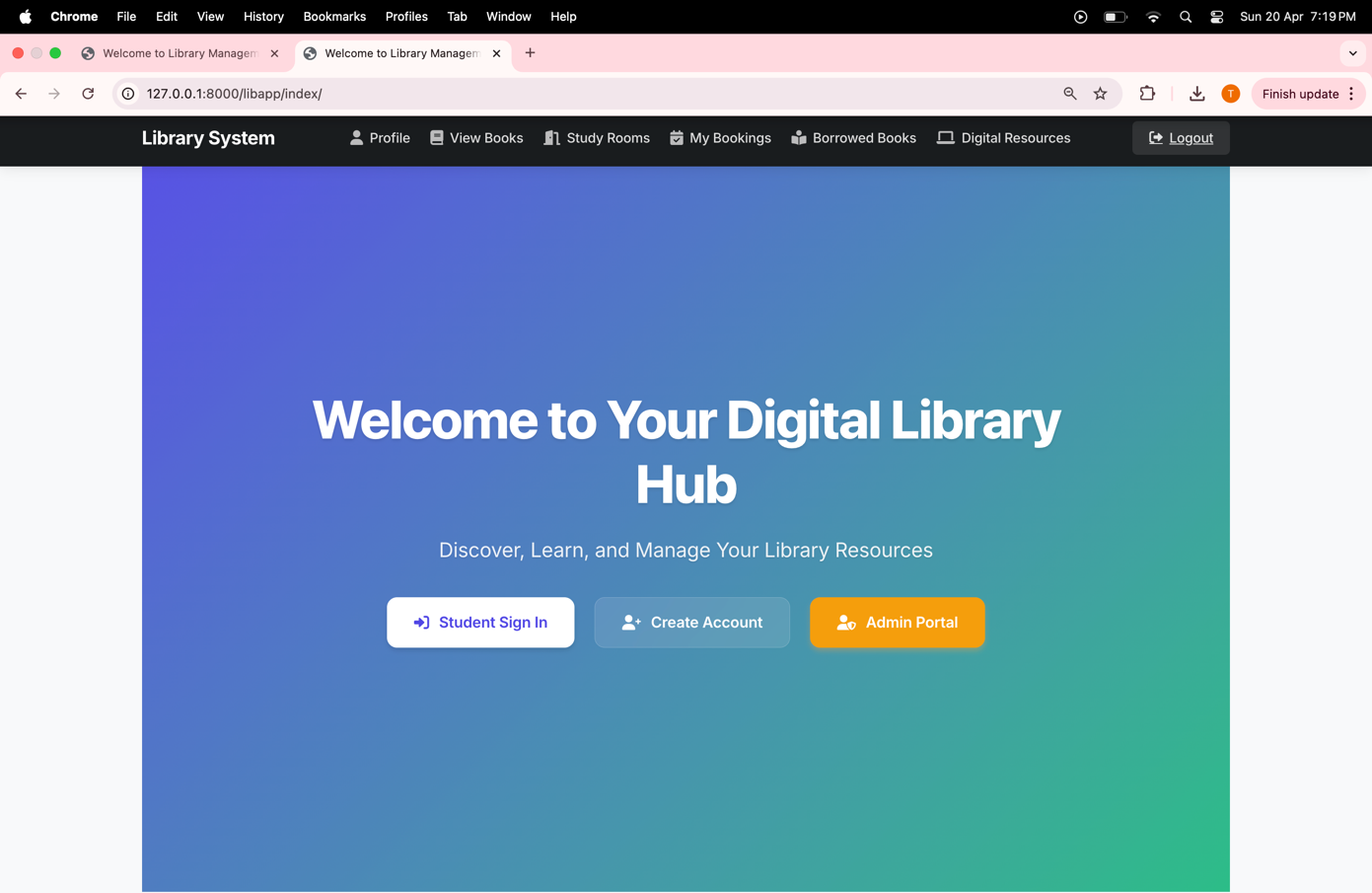
) INTO p\_is\_available;

END //

DELIMITER ;

```

**SCREENSHOTS OF THE PROJECT**

****

**A screenshot of a computer

AI-generated content may be incorrect.**

Home page with Student Sign In, Create Account, and Admin Portal buttons, along with a top navigation bar for Profile, View Books, Study Rooms, My Bookings, Borrowed Books, Digital Resources, and Logout.

**A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.**Profile page with user details, quick actions for booking and viewing study rooms, browsing books, checking borrowed items, and a section for pending fines.

**A screenshot of a computer

AI-generated content may be incorrect.**

view all books page with name, image and author and number of copies available

**A screenshot of a computer

AI-generated content may be incorrect.**

view all study rooms with capacity location and availability

**A screenshot of a computer

AI-generated content may be incorrect.**

view your study room bookings (approved, rejected and pending)

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

see all the available digital resources and download them

**A screenshot of a computer

AI-generated content may be incorrect.**

admin login page**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

admin can add users

**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**

admin can approve/ reject book requests**A computer screen shot of a computer

AI-generated content may be incorrect.**

admin can add books and digital resources in the library**A screenshot of a computer

AI-generated content may be incorrect.**

admin can see digital engagement record**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**

admin can approve/reject room booking requests**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**

THANKYOU