**NAAN MUDHALVAN HACKATHON SET 1 ANSWERS**

**HACKATHON SET - 1**

**Use Case Title:** Library management system

**Student Name:** RUBAN J

**Register Number:** C2S22922

**Institution:** NPR arts and science college,Natham

**Department:** Bsc.Computer science

**Date of Submission:** 17-03-2025

**1. Problem Statement**

Libraries face challenges in managing their operations, including tracking book loans, monitoring overdue items, and maintaining an organized database of available books. The Library Management System (LMS) aims to create a robust database system that simplifies these processes, ensuring that library staff can efficiently manage book lending and provide a better experience for users.

**2. Proposed Solution**

The proposed solution is a SQL-based Library Management System that includes the following features:

* Book Management: Staff can add new books with details such as title, author, genre, and availability status.
* Loan Tracking: The system tracks book loans, including issue and return dates, allowing for efficient monitoring of borrowed books.
* Overdue Monitoring: Automated notifications for overdue books and the capability to generate reports on late returns.
* User Management: Users (students) can easily borrow and return books, with their borrowing history accessible for review.

The system will utilize a well-defined database schema to ensure data integrity and optimize query performance.

**3. Technologies & Tools Considered**

* Database Management System: MySQL or Oracle Database
* Programming Language: SQL for database operations
* Development Tools: MySQL Workbench or Oracle SQL Developer for database design and management
* Version Control: GitHub for repository management

**4. Database Schema & Data Flow**

**Database Schema**

**The database schema includes the following key tables:**

* **Books Table:**
  + ISBN (VARCHAR, Primary Key)
  + Title (VARCHAR)
  + Author (VARCHAR)
  + Genre (VARCHAR)
  + AvailabilityStatus (VARCHAR)
* **Users Table:**
  + UserID (INT, Primary Key)
  + Name (VARCHAR)
  + Email (VARCHAR, Unique)
* **Transactions Table:**
  + TransactionID (INT, Primary Key)
  + UserID (INT, Foreign Key)
  + ISBN (VARCHAR, Foreign Key)
  + IssueDate (DATE)
  + ReturnDate (DATE)

**Data Flow**

**1.** When a user borrows a book, a new transaction is created in the Transactions table.

2. The availability status of the book is updated in the Books table.

3. Queries can be used to fetch overdue books by comparing the current date with the return date.

4. Reports can be generated to show user borrowing history and overdue items.

**Entity-Relationship Diagram (ERD)**

**5. Feasibility & Challenges**

* Feasibility: The proposed LMS is practical due to its straightforward design and the availability of SQL databases. Implementation is feasible within a short timeframe, given the well-defined scope and existing technologies.
* Challenges: Potential challenges include ensuring data integrity during concurrent transactions and managing user access levels. These can be addressed by implementing proper locking mechanisms and user authentication protocols.

**6. Expected Outcome & Impact**

The expected outcome is a fully functional Library Management System that streamlines library operations. Benefits include:

* Improved efficiency in managing book lending and returns.
* Reduced instances of overdue books through automated tracking.
* Enhanced user experience for students borrowing and returning books.

**7. Future Enhancements**

**Future enhancements could include:**

* A web interface for easier access to the LMS.
* Integration with mobile applications for notifications and book searches.
* Advanced reporting features for library staff to analyze borrowing trends.

**8. SQL Scripts**

Table Creation Scripts

sql

CREATE TABLE Books (

ISBN VARCHAR(13) PRIMARY KEY,

Title VARCHAR(255),

Author VARCHAR(255),

Genre VARCHAR(100),

AvailabilityStatus VARCHAR(20)

);

CREATE TABLE Users (

UserID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(255),

Email VARCHAR(255) UNIQUE

);

CREATE TABLE Transactions (

TransactionID INT PRIMARY KEY AUTO\_INCREMENT,

UserID INT,

ISBN VARCHAR(13),

IssueDate DATE,

ReturnDate DATE,

FOREIGN KEY (UserID) REFERENCES Users(UserID),

FOREIGN KEY (ISBN) REFERENCES Books(ISBN)

);

Sample Data Insertion

sql

INSERT INTO Books (ISBN, Title, Author, Genre, AvailabilityStatus) VALUES

('978-3-16-148410-0', 'The Great Gatsby', 'F. Scott Fitzgerald', 'Fiction', 'Available'),

('978-1-56619-909-4', '1984', 'George Orwell', 'Dystopian', 'Available');

INSERT INTO Users (Name, Email) VALUES

( ‘kumar R', 'kumar123@gmail.com'),

('Jane Smith', thakshithapandian3@gmail.com');

Example Queries and Outputs

* Fetch all available books:

sql

SELECT \* FROM Books WHERE AvailabilityStatus = 'Available';

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ISB** | **Title** | **Author** | **Genre** | **Availability Status** |
| **978-3-16-148410-0** | **The Great Gatsby** | **F. Scott Fitzgerald** | **Fiction** | **Available** |
| **978-1-56619-909-4** | **198** | **George Orwell** | **Dystopian** | **Available** |

**Example Output:**

* Track overdue books:

sql

SELECT \* FROM Transactions WHERE ReturnDate < CURDATE();

**Example Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TransactionID** | **UserID** | **ISBN** | **IssueDate** | **ReturnDate** |
| **1** | **1** | **978-3-16-148410-0** | **2025-03-01** | **2025-03-10** |
| **2** | **2** | **978-1-56619-909-4** | **2025-03-05** | **2025-03-15** |
|  |  |  |  |  |

**9. GitHub Repository Structure**

* /sql\_scripts: Contains all SQL scripts for table creation and data insertion.
* /documentation: Contains this document and any additional documentation.
* README.md: Overview of the project, setup instructions, and usage examples.
* **GitHub link:**

[**https://github.com/rubanjr/Naan-muthalvan-hackathan/upload/main**](https://github.com/rubanjr/Naan-muthalvan-hackathan/upload/main)