RMIT International University Vietnam

Assignment Cover Page

Subject Code:	ISYS3414
Subject Name:	Practical Database Concepts
Location & Campus (SGS or HN) where you study:	SGS Campus
Title of Assignment:	Assessment 2 - Group Project Report- Library Management System
Teachers Name:	Truong Nguyen Xuan Vinh
Group Name:	Group 7 - VinhFC
Group Members (names and id numbers):	1. Nguyen Trong Bach (s4044878) 2. Trinh Nguyen Ha (s3981134) 3. Nguyen Mai Huong (s3927244) 4. Ngo Nguyen Phuong Uyen (s3979198)
Assignment due date:	5th of May, 2024
Date of Submission:	5th of May, 2024
Declaration of Authorship	We declare that in submitting all work for this assessment, we have read, understood and agreed to the content and expectations of Assessment Declaration as specified in https://www.rmit.edu.vn/students/my-studies/assessment-and-exams/assessment
Consent to Use:	We give RMIT University permission to use our work as an exemplar and for showcase/exhibition display.

Table of Contents

INTRODUCTION

Topic Overview

Libraries serve as cornerstone institutions in the quest for knowledge, preserving a wealth of information across various media and offering invaluable resources for academic, professional, and personal growth (Rahul 2020). Historically, libraries have facilitated access to a vast array of literature and data, playing a crucial role in education and research. However, as beneficial as these institutions are, traditional libraries face significant operational challenges, especially concerning the management of transactions such as book loans, room reservations, fine collections, and membership services. These challenges primarily arise from outdated manual processes that are often cumbersome and error-prone, leading to issues such as lost books or incomplete records.

Understanding these limitations, our group is committed to developing a project, Library Management System (LMS), modernize traditional libraries by digitizing their core functions.

The first objective of our system is to **reduce paperwork and staffing** in libraries. Previously, traditional libraries required many people to manage books and library facilities, but now this system will help reduce those resources. Secondly, this system is designed to **enhance daily transactions and record management** through a computerized platform, utilized by librarians and library administrators. By leveraging advanced technologies like Oracle Apex and SQL databases, the LMS aims to provide a robust framework for managing library operations, ensuring that issues such as file and record losses are a thing of the past. Furthermore, the system offers powerful search capabilities, enabling users to easily locate books and rooms based on various criteria, including title, author, major/course relevance, category, capacity, and availability. This enhancement not only streamlines library procedures but also significantly improves user experience by facilitating easier access to library resources.

Most notably, one of the features our team is particularly excited about is the "**Personalized Recommendations" feature**, coupled with a **"Ratings and Reviews" system**. As a diverse team of students from business and technology disciplines, we have identified a gap in current digital library offerings, particularly regarding the difficulty of finding appropriate and reliable materials for course studies and major-specific needs. This challenge particularly happens to first-year students who may not be familiar with choosing the best resources for their studies. Hence, our system addresses this issue by providing a more advanced search where students can select their major, course, school, and other criteria. Consequently, our library will suggest tailored book recommendations that match the book requirements students are searching for, enhancing academic outcomes. Additionally, our platform enables previous readers to leave reviews and rate books, enriching the decision-making process for future users and fostering an interactive, supportive library community.

Through these innovations, our LMS not only solves operational inefficiencies but also transforms the library into a dynamic, interactive hub that significantly enhances the educational journey of its users. This project reflects our vision for a future where libraries are not only repositories of information but also vibrant centers of learning and engagement.

People

Member	Personal Interest	Relevant skills	Role description
Nguyen Trong Bach (s4044878) **Email address:** s4044878@rmit.edu.vn	Bach is an enthusiastic software engineer strongly oriented towards a career as a full stack developer in a large-scale software corporation. His passion for programing and software engineering arose when he participated in an academic research project on combinatorics optimization, where the application of computer sciences can bring great economic and environmental benefits to the real world fascinated the high schooler into pursuing a degree in software engineering. Furthermore, an internship as a software engineer helped him discovered the great potential and conveniency that a computer software can bring to the society while reaping great economic benefits which captivated him to one day designing an industry disruptive software that can greatly improve the efficiency and help bring greater good to the society. Understanding the underlying principles on how databases and data storage works will laid a great foundation towards comprehending how software system operates, thus moving him a step closer towards his dream.	Bach took part in various university level computer science courses and programs, as well as participation in a university level academical programming research project, building foundation for great programming knowledge including object oriented programming and data structure and algorithms, and combinatoric optimizations, which help him navigate through new technologies and application platforms seamlessly with little struggles. Moreover, with past experiences as a Front-End Developer at a tech corporation, he is passionate in designing and implementing front end designs and concepts to integrate and articulate the backend components with the UI/UX components, delivering sophisticated user experiences which are critical in design and implementing the application front end for the project. He also had a great communication skill to help bringing people together and finding common voices within each discussion in order to improve the work dynamic and boosting overall efficiency within the team.	- Brainstormed and suggested library system functions and generation. - Participated in developing the ERD. - Primarily responsible for Oracle Apex aspects (database usage, retrieval, and administration) - Finalized the report content. - Finalized the presentation.
Trinh Nguyen Ha (s3981134) **Email address:**	Ha's passion for science, especially about computers, has been a driving force of her life. It goes beyond academics; but rather a way of daily routine and a source of boundless curiosity. Her journey into the world of Information Technology began during	With the strongest academic background and prior experience in database design as well as SQL in particular, she took charge of being the Technical Leader of our whole group. From making final calls to the system scope and design to ensuring	- Brainstormed and outlined general functions and structure of

s3981134@rmit.edu.vn	a summer break in 10th grade when she was captivated by true crime and medical forensics documentaries. It was then that Ha realised the critical role that systematic, organised, and digital records play in solving complex problems. The shortage of such records often hinders the prompt resolution of issues, and she was haunted by the idea that effective data management and information systems are the key to solving these problems. This realisation became a calling, inspiring her to become a data scientist, with the aim of automating and simplifying everyday tasks, thus contributing significantly to society.	that the implementation of the application was on point, to writing up/organizing the documentation. Moreover, as Ha had studied in software project management courses such as BITS, SEF, and SEPM; she also took a bit of a role in sketching out the project timeline and set deadlines for everyone.	the library system. - Participated in developing the ERD. - Generated relational schema translations and database creation codes. - Main developer in Oracle Apex (database usage, retrieval, and administration) - Wrote complicated triggers and queries for the statistical charts. - Finalized report content. - Finalized the presentation.
Nguyen Mai Huong (s3927244)	As a Business student with a deep passion for technology, particularly data analytics, she recognized the crucial role technology plays in modern business development. Observing daily business decisions, she realized that with limited resources such as money	She possesses fundamental technical skills in SQL, enabling her to conceptualize the database schema, write basic queries, and manage databases at a beginner level. Additionally, she is proficient in Microsoft Office, which aids her in	- Brainstormed and suggested functions and generation of

Email address: s3927244@rmit.edu.vn	and opportunities, data is an indispensable part of making informed decisions. This realization prompted her to step outside her comfort zone and enroll in a course vastly different from her major, Practical Database Concepts. She was thankful for the course as it enriched her knowledge about databases and introduced her to SQL for data analysis and Oracle Apex for its capabilities in facilitating rapid web application development with extensive customization options. When she was assigned the Library Management System project, she felt a surge of enthusiasm and commitment. Having consistently utilized the RMIT library to support her studies, she realized she had yet to fully understand the intricacies of how its databases were operated. This assignment offered her a golden opportunity to delve deeper into the mechanics of how digital libraries are established and function. Consequently, she gained extensive knowledge not only about databases but also about the broader aspects of managing a library system, significantly broadening her academic and practical understanding.	preparing reports and presentations. On the soft skills front, her capabilities in problem-solving, flexibility, adaptability, leadership, management, organization, and communication equip her to effectively address challenges and function efficiently in varied situations.	the library system. - Mainly responsible for Background and Scope of the project. - Participated in developing the ERD. - Mainly in charge of Conclusions and Recommendations. - Finalized the report content. - Referencing
Ngo Nguyen Phuong Uyen (s3979198) **Email address:** s3979198@rmit.edu.vn	Uyen's prior experience within the Operations Department of a technology startup fostered a keen understanding of the critical role database construction and organization play in establishing efficient and seamless workflows within an organization. However, limitations in available resources often confined her past work to utilizing Excel and Sheets, which proved inadequate for managing complex datasets.	While Uyen's technical background may not have been as extensive as some of her teammates, she effectively leveraged her strengths to make valuable contributions to the project. Leveraging her flexibility and resourcefulness, she proactively sought out relevant materials to enhance her technical skills, concurrently enriching the team's collective learning resources.	- Brainstormed and outlined general functions and structure of the library system. - Played a key role in developing

	Recognizing her aspirations to evolve into a Business Analyst, Uyen embarked on a journey to explore the realms of Data Analytics and Business Analytics, with an initial focus on understanding Database Organization. Prior to enrolling in the Practical Database Concepts course, Uyen had pursued certifications in Data Analytics and Business Analytics. However, it was the structured and foundational approach of the course that illuminated her path towards data management from its most rudimentary standpoint. Furthermore, her proficiency in SQL was significantly bolstered, equipping her with the ability to query data and manage attribute relationships efficiently. The context of the Library Management System project allowed Uyen to fully appreciate the profound impact of pre-established database organization and the creation of attribute relationships on the overall workflow and functionality of the system. This newfound understanding allowed her to leverage her existing organizational skills, operational experience, and newly acquired knowledge to contribute meaningfully to the project.	Furthermore, her understanding in SQL has equipped her with the ability to identify bugs and propose solutions to her peers during the construction of query schemas. Uyen's organizational and optimization skills were instrumental in crafting a well-designed Entity-Relationship Diagram (ERD). This well-structured ERD laid the foundation for efficient application development by the team. Lastly, Uyen's analytical and detail-oriented approach significantly contributed to the organization, structure, and detailed explanations within the project report, particularly for the Application Features section.	ERD design and identifying relationships. - Primary participant in populating the database. - Handled report formatting and visualization. - Mainly responsible for the Application section of the report. - Finalized the report content.

Project Scope

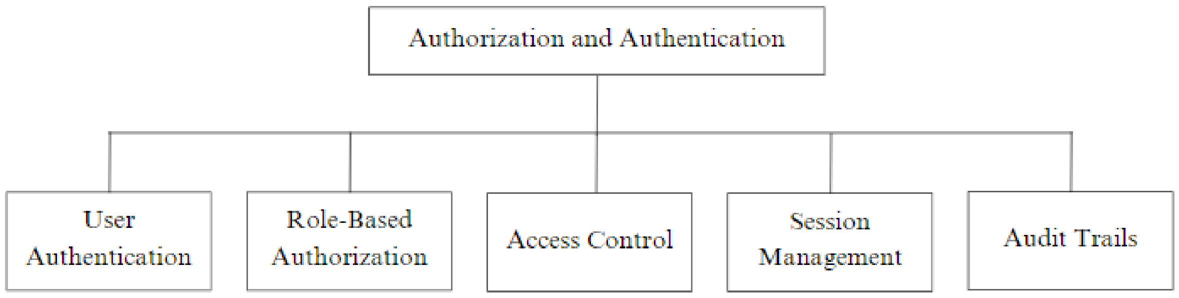
In scope

The LMS is designed to deal with the management of book and room resources within a university setting, catering specifically to the needs of students and staff/faculty. This comprehensive system will enable the efficient handling of books and room bookings, making them easily searchable, sortable, and lendable/bookable. We also introduce Librarian and Admin for higher and more general management and administrative rights.

The front-end tier will be developed using Oracle Apex, a low-code development environment that enables rapid web application development with a high degree of customization (Oracle Apex n.d). This platform is ideal for creating scalable and secure applications, making it particularly suitable for managing the complex data and user interaction needs of the LMS.

Core functionalities

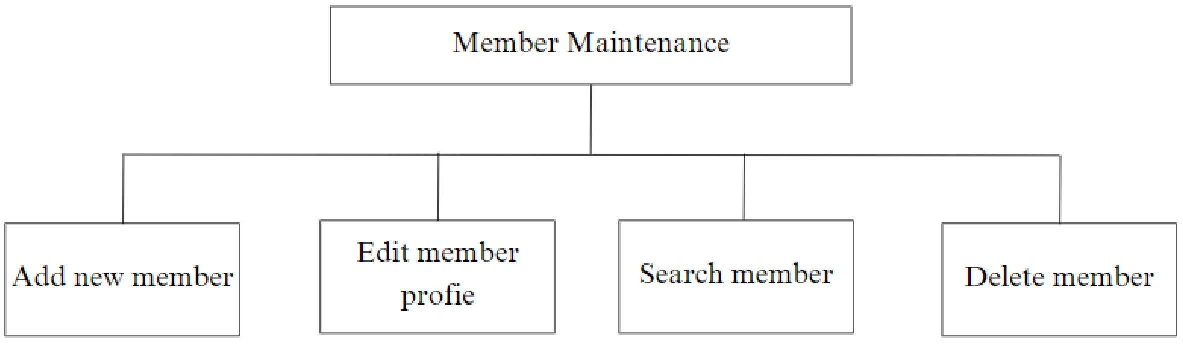
- **Authorization and Authentication**



The Authentication and Authorization mechanism on our library system application is employed by various user groups, including regular members, admins, and librarians, each logging in with unique usernames and passwords. Everyone has the ID format of **'Uxxxxxx'**, however categorized by the **user_type** attribute, which includes 'Admin', 'Student', 'Staff', and 'Librarian'.

Outcomes: This systematic structuring differentiates user privileges and access levels - normal members have certain access, while admins and librarians have broader permissions, such as viewing the complete book transaction history of all members, reflecting their roles' responsibilities and needs.

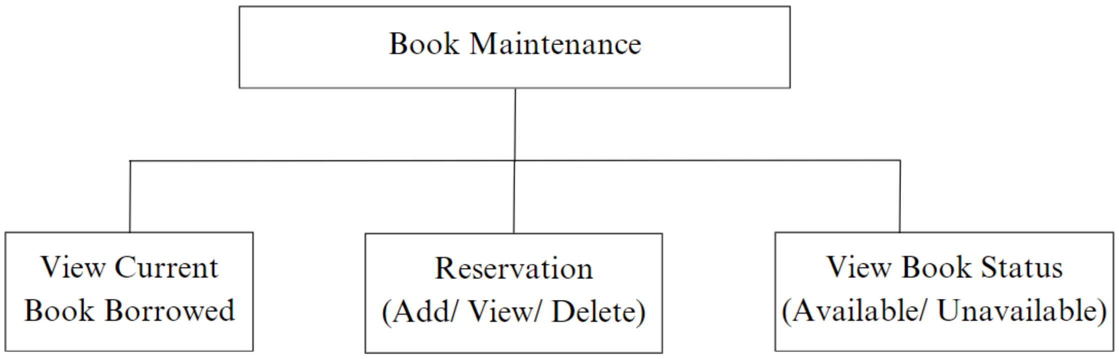
- **Member/Major/Course Maintenance**



This module is mainly for Admin in maintaining member profiles by allowing them to add, edit, search, and delete Librarian/Student/Staff information.

Outcomes: Supporting better organizational management and user service within the library system.

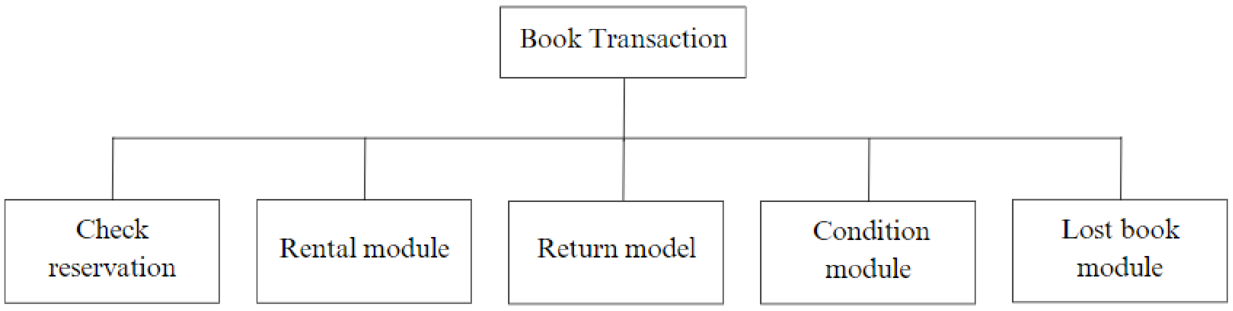
- **Book Maintenance**


Book Maintenance
View Current Book Borrowed
Reservation (Add/ View/ Delete)
View Book Status (Available/ Unavailable)

The module mainly aids librarians in managing the library's book collection by providing options to add new books, edit existing information, and search for books within the database.

Outcomes: Maintaining an accurate and current collection, ensuring that users have access to the latest resources and that the library catalog remains up-to-date and easily navigable.

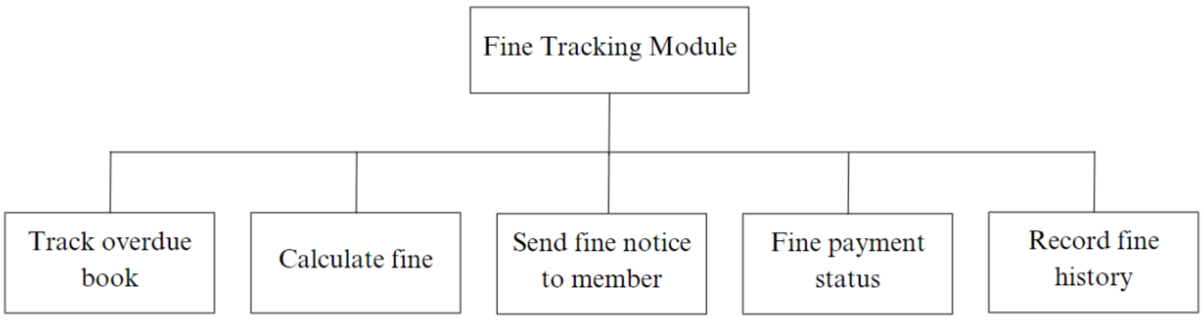
- **Book Transaction**



The book transaction module can be accessed by admin and librarians and is used to manage the number of books borrowed, returned, the current books remaining in the library, and the books' status, including those that are overdue or lost.

Outcomes: Ensuring accurate record-keeping and enhances accountability, providing a reliable framework for managing library resources effectively.

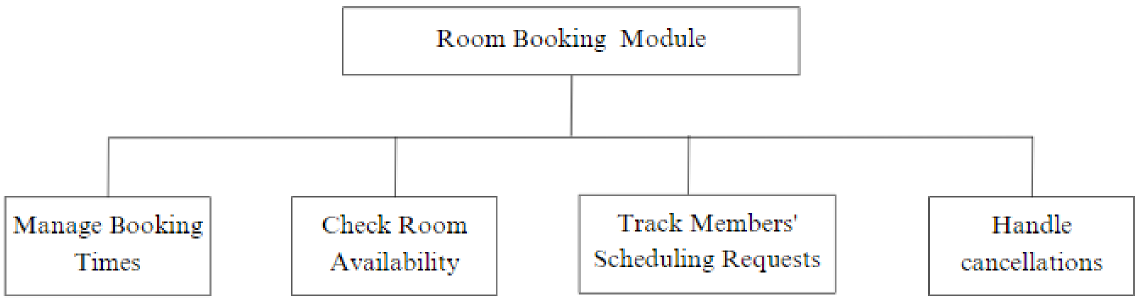
- **Fine Tracking**



Fine tracking module is designed to help admin and librarians manage fines efficiently. It facilitates the monitoring of overdue books, automates fine calculation based on pre-set rules, generates and dispatches fine notices to members, oversees the status of fine payments, and allows the tracking of comprehensive fine histories.

Outcomes: Maintaining a detailed history of all fine transactions, enabling a comprehensive view of financial interactions and helping libraries maintain financial accountability and operational efficiency.

- **Meeting Room Booking**



This module consists of four key functions: "Manage Booking Times" allows staff to set up and modify the timings for room reservations; "Check Room Availability" provides real-time status updates on which rooms are free to book; "Track Members' Scheduling Requests" offers a way to monitor and fulfill booking requests from library members; and "Handle Cancellations" enables the system to process and update the schedule when bookings are canceled.

Outcomes: Streamlining the room reservation process, ensuring an organized approach to managing the library's meeting spaces.

- **Search and Organization**

The search module on our website enables both members and librarians to efficiently find books and rooms by employing a variety of filters, including title, author, relevance to specific majors or courses, category, room capacity, and availability. Additionally, users have the ability to access detailed information about each book, including its rankings and reviews from other readers.

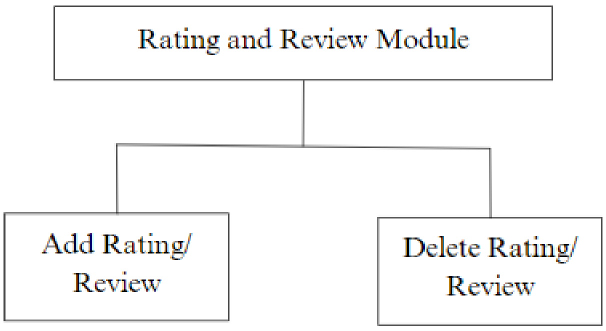
Outcomes: Enhancing the user experience by providing comprehensive insights and facilitating easier access to library resources.

- **Report Module**

The Report Module enables librarians to generate reports related to the collected database, such as the number of students from various majors, top borrowed books, books categorized by schools, and so on.

Outcomes: Providing valuable insights into user trends and resource utilization, aiding in strategic library planning and management.

- **Rating and Review**



The Rating and Review function allows users to rate books and provide feedback, guiding others in their reading choices.

Outcomes: Supporting popular and relevant materials, influencing library acquisitions and ensuring resources align with user needs. By fostering a collaborative environment, it enhances the overall user experience and supports informed decision-making.

Out of Scope:

- A fully functional website made with HTML/CSS/JS technology.

- The implementation's idea of the "External Database Access for Institutional Library Accounts" feature provides libraries with privileged access to a range of external databases, including Statista, World Bank, Passport, LinkedIn Learning, etc. This initiative aims to enhance research capabilities and scholarly activities.

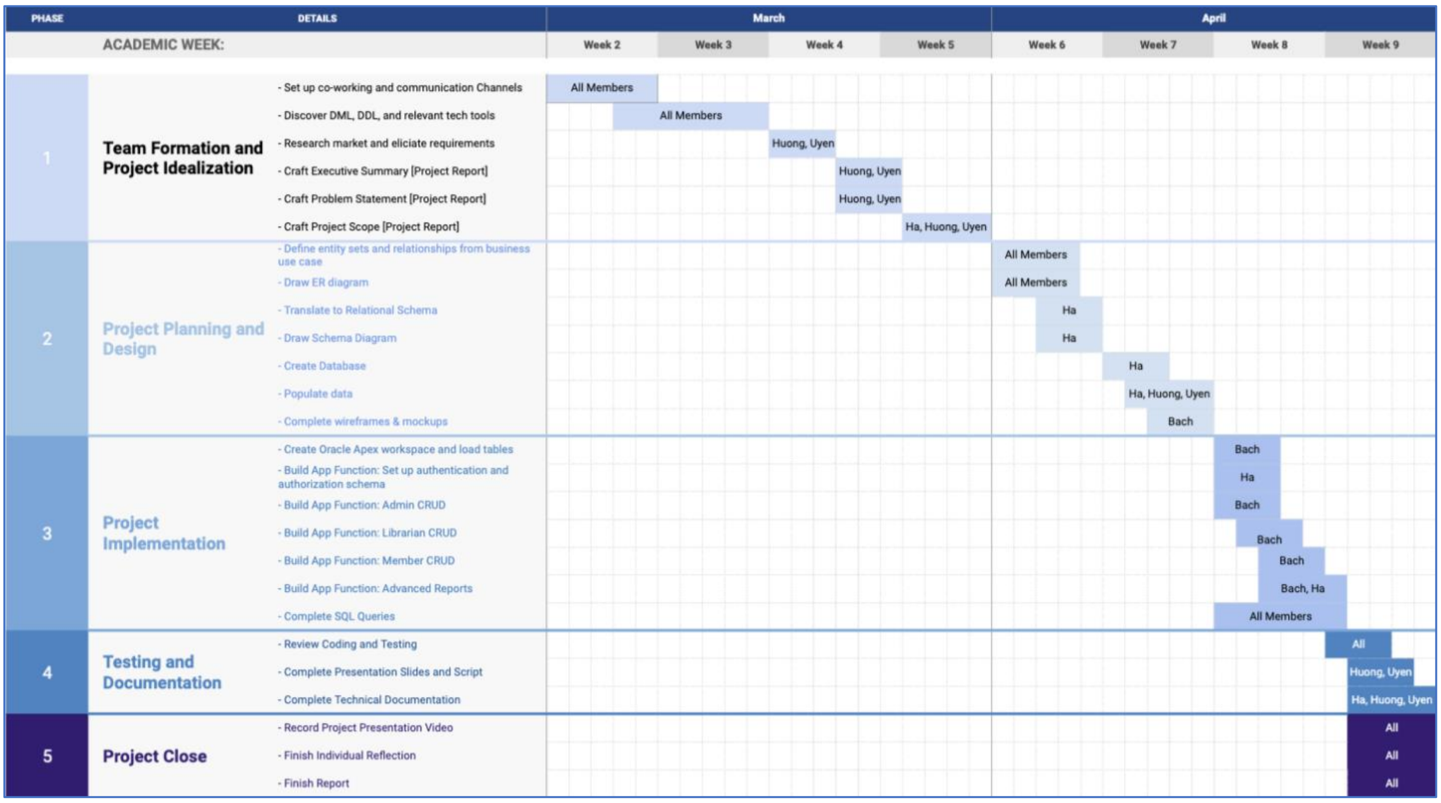
- "Language Localization" enhances system accessibility and inclusivity by providing support for multiple languages beyond the default, accommodating a broader range of users.

- "Integrated Citation Management" incorporates tools within the library system to assist users in organizing, citing, and managing references for their research projects and assignments.

- Common features in other libraries such as 'Chat with Librarian' and 'Book Consultation' have not yet been implemented in our library.

- The idea of using images to display information about books and to feature weekly and monthly library marketing campaigns/ events has not yet been successfully implemented.

Progress



PHASE	DETAILS		March				April			
	ACADEMIC WEEK:		Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9
1	Team Formation and Project Idealization	- Set up co-working and communication Channels	All Members							
		- Discover DML, DDL, and relevant tech tools		All Members						
		- Research market and eliciate requirements			Huong, Uyen					
		- Craft Executive Summary [Project Report]				Huong, Uyen				
		- Craft Problem Statement [Project Report]				Huong, Uyen				
		- Craft Project Scope [Project Report]				Ha, Huong, Uyen				
2	Project Planning and Design	- Define entity sets and relationships from business use case					All Members			
		- Draw ER diagram					All Members			
		- Translate to Relational Schema					Ha			
		- Draw Schema Diagram					Ha			
		- Create Database						Ha		
		- Populate data						Ha, Huong, Uyen		
		- Complete wireframes & mockups						Bach		
3	Project Implementation	- Create Oracle Apex workspace and load tables							Bach	
		- Build App Function: Set up authentication and authorization schema							Ha	
		- Build App Function: Admin CRUD							Bach	
		- Build App Function: Librarian CRUD							Bach	
		- Build App Function: Member CRUD							Bach	
		- Build App Function: Advanced Reports							Bach, Ha	
		- Complete SQL Queries							All Members	
4	Testing and Documentation	- Review Coding and Testing								All
		- Complete Presentation Slides and Script								Huong, Uyen
		- Complete Technical Documentation								Ha, Huong, Uyen
5	Project Close	- Record Project Presentation Video								All
		- Finish Individual Reflection								All
		- Finish Report								All

Figure 1: VinhFC's Project Timeline

The Waterfall project management methodology, with its linear, structured, and phase-by-phase progression - from requirements analysis to maintenance - is particularly suitable for educational settings where team size is small, assignments have tight deadlines, and deliverables are clearly defined upfront in the assessment specification, with little expectation for major changes (Aiden 2019). For VinhFC's LMS, this approach aligns perfectly with the well-defined scope and the project's tight 7-week schedule, minimizing the risks of scope creep. One has specific tasks assigned over two-week periods, aligning perfectly with the linear approach. This is beneficial because it ensures that before moving to the next phase, such as from Design to Project Implementation, all prior phase requirements like drawing ER diagrams and defining schemas are fully completed and signed off.

In contrast, Agile/Scrum methodologies, though excellent for projects requiring flexibility and rapid iteration, may not be ideal in this context. The frequent meetings and adjustments characteristic of Agile could complicate scheduling and increase the workload, potentially leading to delays [Agile vs. waterfall project management (Dan 2023). Moreover, the predictable nature of Waterfall aids in straightforward task allocation, with each team member clearly understanding their roles and deadlines. This makes our chosen process model a practical choice for ensuring that all objectives are met systematically and efficiently.

DATABASE DESIGN AND IMPLEMENTATION

Conceptual Design & ERD

We strictly adhere to real-world business rules, thereby eliminating the need for assumptions. This ensures our operations are both realistic and align with established standards.

Entity Relationship Diagram

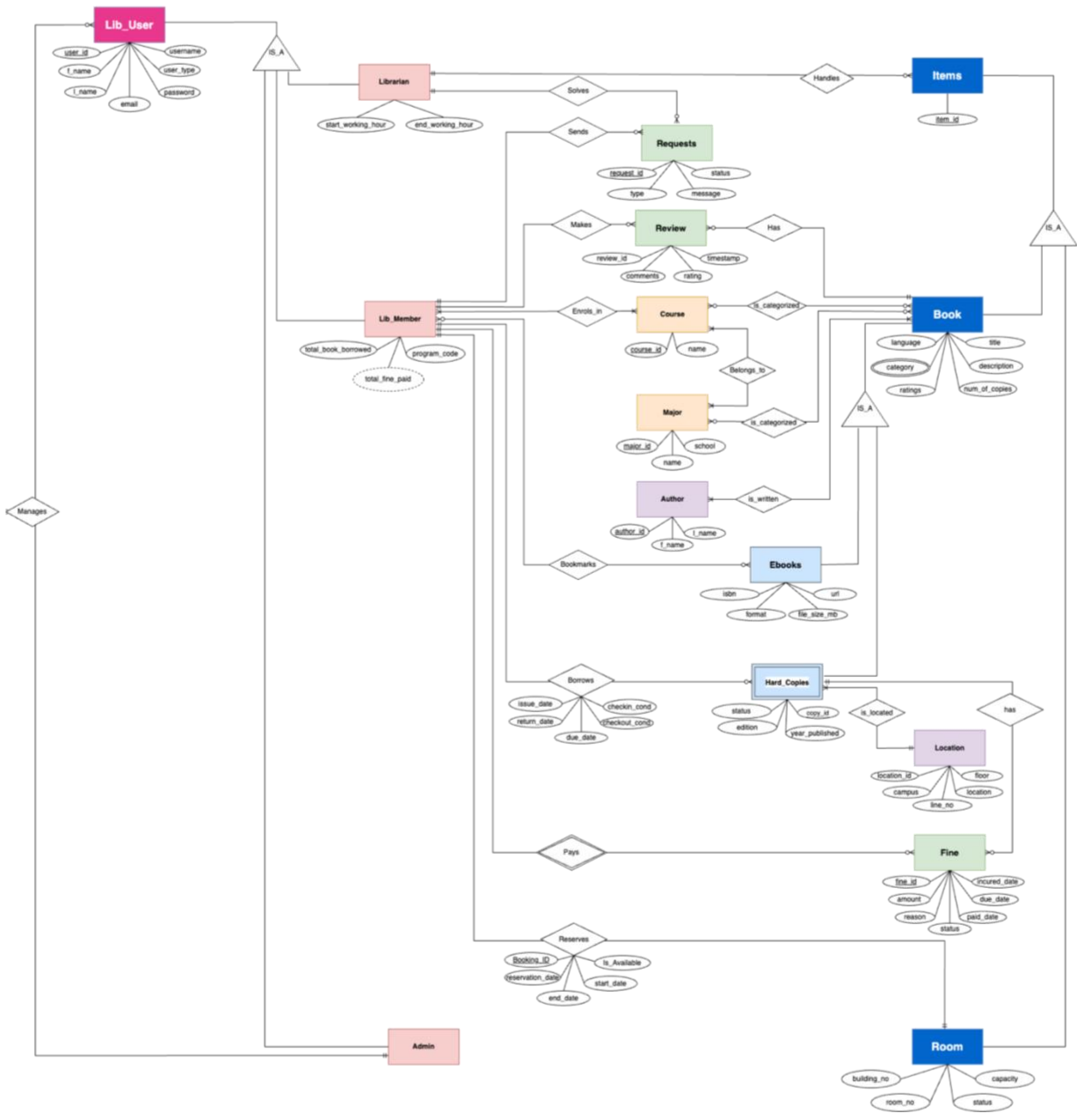


Figure 2: ERD for an Online Library Management System

Description

There is a total of 14 entity types:

- **Major**: Represents academic majors with fields for major_ID, school, and name; used for categorizing books and student management. Major_ids are taken from the RMIT Vietnam website.
- **Course**: Details academic courses available; used for categorizing books and students to enroll in. Courses are taken from a General Electives list publised on RMIT Vietnam website.
- **Category**: Used for categorizing books. There will be approximately 40 categories, both academically and non-academically related.
- **Location**: Specifies the physical location within the library (campus, floor, shelf, line number).
- **Lib_User**, identifiable by primary key user_ID, and other attributes named username, f_name, l_name, email, pwd, created_at, and user_type. Is specialized into Lib_Member (Student/Staff), Lib_Librarian and Lib_Admin.
    - **Lib_Member** will have six main actions: sending Requests, making Review for books, borrowing Hard_Copies, bookmarking Ebooks, paying Fine for late returns/property damage, and finally reserving a meeting room.
    - **Lib_Librarian** includes two main tasks: handling Item and resolving **Requests**.
    - **Lib_Admin** is the one with the most administrative priviledges, which consist of
- **Item** includes Book and Room.
    - **Book** entity:
        - Has one or multiple **Author.** An author can also write more than one book.
        - Determine whether the book is a **Hard_Copies** or electronic (**Ebooks**). If the book is a Hard_Copies, it will be located using Location. If it is an Ebooks, it will include an isbn and an online URL.
        - Contains many **Review** (primary key review_ID) from students/faculty. This includes point ratings ranging from 1 to 5, comments, and timestamp.
    - **Room**: Details about rooms that can be booked for study or meetings. Restrictions are that one room can only be booked by one member at a time, with a maximum time span of 2 hours on the same date.
- A **Fine** record is associated with one library member and a particular book copy.

Relationships

Entity 1	Relationship	Entity 2	Cardinality
Lib_Member	Sends	Requests	One-to-Many
	Makes	Review	One-to-Many
	Borrows	Hard_Copies	One-to-Many
	Bookmarks	Ebooks	Many-to-Many
	Pays	Fine	One-to-Many
	Reserves	Room	One-to-Many
	Enrols_in	Course	Many-to-Many
Lib_Librarian	Manages	Item	One-to-Many
	Solves	Requests	One-to-Many
Lib_Admin	Manages	Lib_User	One-to-Many
	Manages	Item, Course, Major	One-to-Many
Book, Room	IS_A	Item	Hierarchy
Hard_Copies, Ebooks	IS_A	Book	Hierarchy
Book	Has	Review	One-to-Many
	Is_writen_by	Author	Many-to-Many
	Has	Category	Many-to-Many
	Belongs_to	Course, Major	Many-to-Many
Hard_Copies	Is_located_in	Location	Many-to-One
	Has	Fine	Many-to-One
Major	Has	Course	Many-to-Many *
	Has	Lib_Member	One-to-Many

*: *will not be implemented for simplicity*

Logical Design, Relational Schema, and Database Creation

Relational Schema

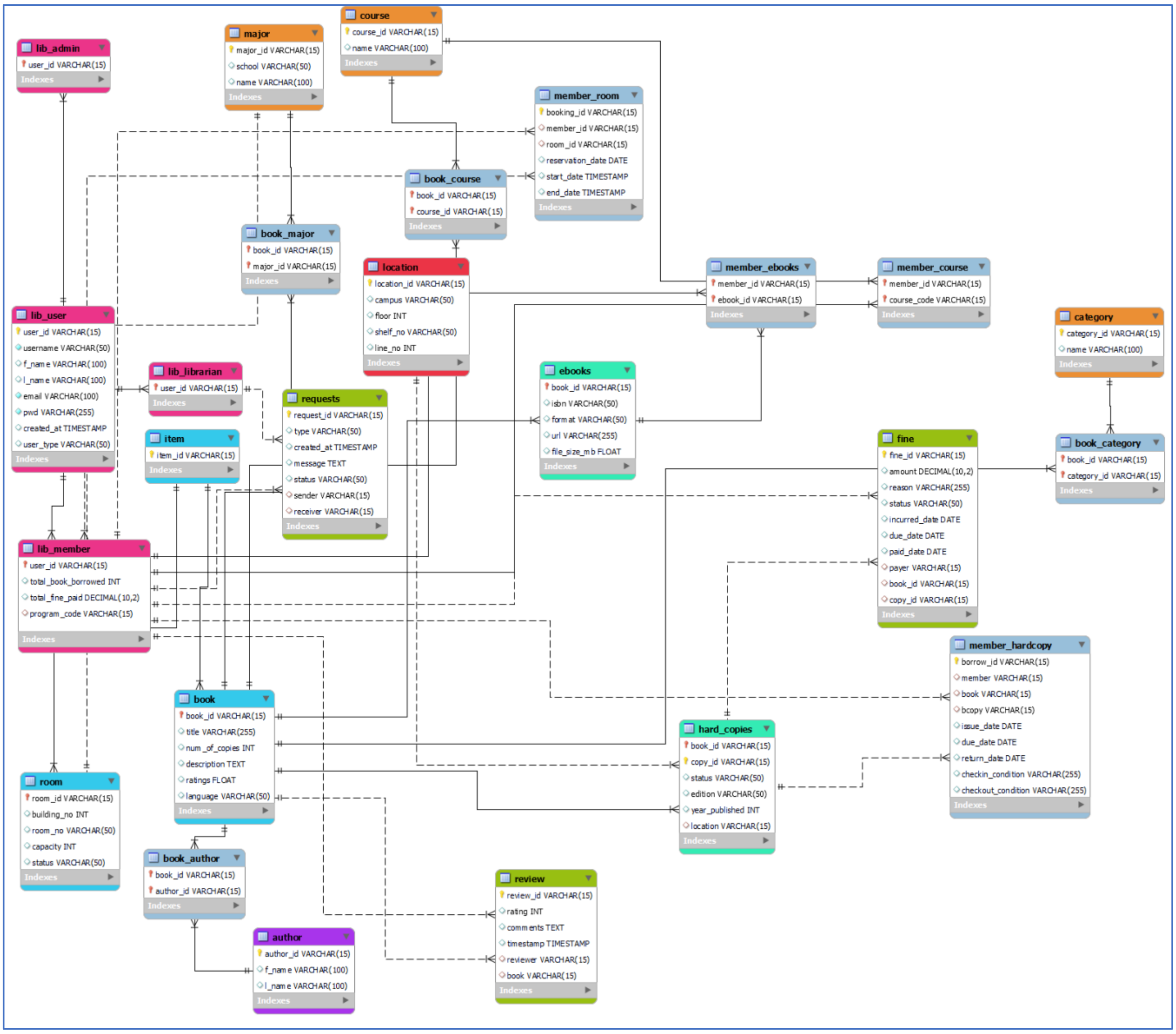


Figure 3: Schema Diagram generated using Reverse Engineer on MySQL Workbench CE

List of schema: Comprises of 25 tables deprived from 17 entities and 6 many-to-many relationships; two exceptions are that: even though the **‘reserves’** relationship between a Lib_Member and a Room and the **‘borrows’** relationship between a Lib_Member and a Hard_Copies are one-to-many (1:N), we would like to add 2 tables Member_Room and Member_HardCopy to keep track of the history of borrowing/booking records.

- **Major** (**major_id**, school, name);
- **Course** (**course_id**, name);
- **Category** (**category_id**, name);
- **Location** (**location_id**, campus, floor, shelf_no, line_no);
- **Lib_User** (**user_id**, username, f_name, l_name, email, pwd, created_at, user_type);
- **Lib_Member** (**Lib_User.user_id user_id**, total_book_borrowed, total_fine_paid, Major.major_id program_code);
- **Lib_Librarian** (**Lib_User.user_id user_id**);
- **Lib_Admin** (**Lib_User.user_id user_id**);
- **Item** (**item_id**);
- **Room** (**Item.item_id room_id**, building_no, room_no, capacity, status);
- **Book** (**Item.item_id book_id**, title, num_of_copies, description, ratings, language);
- **Ebooks** (**Book.book_id book_id**, isbn, format, url, file_size_mb);
- **Hard_Copies** (**Book.book_id book_id**, **copy_id**, status, edition, year_published, Location.location_id location);
- **Author** (**author_id**, f_name, l_name);
- **Review** (**review_id**, rating, comments, timestamp, Lib_Member.user_id reviewer, Book.book_id book);
- **Fine** (**fine_id**, amount, reason, status, incurred_date, due_date, paid_date, Lib_Member.user_id payer, Hard_Copies.(book_id, copy_id) (book_id, copy_id));
- **Requests** (**request_id**, type, created_at, message, status, Lib_Member.user_id sender, Lib_Librarian.user_id receiver);
- **Member_Course** (**Lib_Member.user_id member, Course.course_id course_code**);
- **Member_Ebooks** (**Lib_Member.user_id member, Ebooks.book_id ebook_id**);
- **Member_Room** (**booking_id**, Lib_Member.user_id member_id, Room.room_id room_id, reservation_date, start_date, end_date);
- **Member_HardCopy** (**borrow_id**, Lib_Member.user_id member, Hard_Copies.(book_id, copy_id) (book, bcopy), issue_date, due_date, return_date, checkin_condition, checkout_condition);
- **Book_Author** (**Book.book_id book_id**, **Author.author_id author_id**);
- **Book_Course** (**Book.book_id book_id**, **Course.course_id course_id**);
- **Book_Category** (**Book.book_id**, **Category.category_id category_id**);
- **Book_Major** (**Book.book_id book_id**, **Major.major_id major_id**);

Normalization:

Database normalization is a fundamental design approach used to organize database tables and their relationships to increase the efficiency of the database by eliminating redundancy and inconsistent dependency (Albarak 2020). It involves structuring a database in accordance with a series of so-called "normal forms" in order to reduce data redundancy and improve data integrity. Typically, the most aimed-for level in many practical applications is the Third Normal Form (3NF) (Sorbello 2022).

Examining the schema for our LMS, each table follows these principles rigorously:

- **Entity Integrity and Atomicity (1NF):** Each table has a primary key defined. Moreover, all attributes store single values, maintaining atomicity and compliance with 1NF.
- **No Partial Dependencies (2NF):** Tables with composite primary keys: Member_HardCopy and Book_Author demonstrates that all other attributes in the tables depend on the whole primary key and not on just part of it. For example, in Book_Author, both book_id and author_id is needed to uniquely identify a record, ensuring compliance with 2NF.
- **No Transitive Dependencies (3NF):** Attributes in each table are dependent only on the primary key. For instance, in Lib_Member, attributes like total_book_borrowed and total_fine_paid directly depend on user_id and not through some other attribute. Similarly, in tables like Book or Ebooks, non-key attributes depend directly on their primary key without any transitive dependency, ensuring 3NF compliance.

This adherence helps in minimizing redundancy (no repeated data across multiple tables), reducing the chance of data anomalies, and ensuring a clear and efficient data retrieval path.

Database Creation

Link to our group repository: https://github.com/bleuucheese/vinhfc.

The SQL file ***(ISYS3414_DB_Group7_VinhFC.sql)*** includes the complete schema and data setup necessary for the LMS. It is intended for execution on MySQL Workbench (version 8.0.36 Community Edition) and Oracle (specifically Oracle Apex). The primary programming languages utilized in this project were SQL and PL/SQL.

Step 1: Clean up the database schema, with considerations about safe order to execute (because of checks and foreign key/child constraints).

MySQL

```sql
DROP TABLE IF EXISTS
Member_HardCopy;
DROP TABLE IF EXISTS Member_Room;
DROP TABLE IF EXISTS Member_Ebooks;
DROP TABLE IF EXISTS Member_Course;
DROP TABLE IF EXISTS Requests;
DROP TABLE IF EXISTS Fine;
DROP TABLE IF EXISTS Review;
DROP TABLE IF EXISTS Hard_Copies;
DROP TABLE IF EXISTS Ebooks;
DROP TABLE IF EXISTS Book_Author;
DROP TABLE IF EXISTS Book_Course;
DROP TABLE IF EXISTS Book_Category;
DROP TABLE IF EXISTS Book_Major;
DROP TABLE IF EXISTS Room;
DROP TABLE IF EXISTS Book;
DROP TABLE IF EXISTS Item;
DROP TABLE IF EXISTS Lib_Librarian;
DROP TABLE IF EXISTS Lib_Admin;
DROP TABLE IF EXISTS Lib_Member;
DROP TABLE IF EXISTS Lib_User;
DROP TABLE IF EXISTS Author;
DROP TABLE IF EXISTS Location;
DROP TABLE IF EXISTS Course;
DROP TABLE IF EXISTS Category;
```

Oracle

```sql
BEGIN
  EXECUTE IMMEDIATE 'DROP TABLE
Member_HardCopy';
  EXECUTE IMMEDIATE 'DROP TABLE
Member_Room';
  EXECUTE IMMEDIATE 'DROP TABLE
Member_Ebooks';
  EXECUTE IMMEDIATE 'DROP TABLE
Member_Course';
  EXECUTE IMMEDIATE 'DROP TABLE
Requests';
  EXECUTE IMMEDIATE 'DROP TABLE Fine';
  EXECUTE IMMEDIATE 'DROP TABLE Review';
  EXECUTE IMMEDIATE 'DROP TABLE
Hard_Copies';
  EXECUTE IMMEDIATE 'DROP TABLE Ebooks';
  EXECUTE IMMEDIATE 'DROP TABLE
Book_Author';
  EXECUTE IMMEDIATE 'DROP TABLE
Book_Course';
  EXECUTE IMMEDIATE 'DROP TABLE
Book_Category';
  EXECUTE IMMEDIATE 'DROP TABLE
Book_Major';
  EXECUTE IMMEDIATE 'DROP TABLE Room';
```

`DROP TABLE IF EXISTS Major;`	`EXECUTE IMMEDIATE 'DROP TABLE Book';` `EXECUTE IMMEDIATE 'DROP TABLE Item';` `EXECUTE IMMEDIATE 'DROP TABLE Lib_Librarian';` `EXECUTE IMMEDIATE 'DROP TABLE Lib_Admin';` `EXECUTE IMMEDIATE 'DROP TABLE Lib_Member';` `EXECUTE IMMEDIATE 'DROP TABLE Lib_User';` `EXECUTE IMMEDIATE 'DROP TABLE Author';` `EXECUTE IMMEDIATE 'DROP TABLE Location';` `EXECUTE IMMEDIATE 'DROP TABLE Course';` `EXECUTE IMMEDIATE 'DROP TABLE Category';` `EXECUTE IMMEDIATE 'DROP TABLE Major';` `EXCEPTION` `WHEN OTHERS THEN` `DBMS_OUTPUT.PUT_LINE('Error encountered: ' || SQLCODE || ' - ' || SQLERRM);` `END;`

Step 2: DDL statements

In our schema design, the implementation of ON DELETE CASCADE and ON DELETE SET NULL directives is of paramount importance in maintaining the integrity of the dataset while dissociating the interlinked records, thus offering a degree of flexibility in data retention. The ON DELETE CASCADE ensures the automatic deletion of dependent records within child tables upon the removal of associated rows from the parent table, thereby averting the occurrence of orphaned entries and upholding data consistency. Conversely, the ON DELETE SET NULL provides a mechanism whereby foreign key fields in child tables are assigned a NULL value in the event of the deletion of referenced rows in the parent table.

Table Name	SQL
MAJOR	`CREATE TABLE Major (` `major_id VARCHAR(15) PRIMARY KEY,`

	`school VARCHAR(50) CHECK (school IN ('SSET', 'TBS', 'SCD', 'SEUP')),`
	`name VARCHAR(100)`
	`);`

COURSE	`CREATE TABLE Course (`
	`    course_id VARCHAR(15) PRIMARY KEY,`
	`    name VARCHAR(100)`
	`);`

CATEGORY	`CREATE TABLE Category (`
	`    category_id VARCHAR(15) PRIMARY KEY,`
	`    name VARCHAR(100)`
	`);`

LOCATION	`CREATE TABLE Location (`
	`    location_id VARCHAR(15) PRIMARY KEY,`
	`    campus VARCHAR(50),`
	`    floor INT,`
	`    shelf_no VARCHAR(50),`
	`    line_no INT`
	`);`

LIB_USER	`CREATE TABLE Lib_User (`
	`    user_id VARCHAR(15) PRIMARY KEY,`
	`    username VARCHAR(100) CONSTRAINT username_unique UNIQUE NOT NULL,`
	`    f_name VARCHAR2(100),`
	`    l_name VARCHAR2(100),`
	`    email VARCHAR(100) CONSTRAINT email_unique UNIQUE NOT NULL,`
	`    pwd VARCHAR2(255) NOT NULL,`
	`    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,`
	`    user_type VARCHAR(50) CHECK (user_type IN ('Student', 'Staff', 'Librarian', 'Admin'))`
	`);`

LIB_MEMBER

```
CREATE TABLE Lib_Admin (
    user_id VARCHAR(15) PRIMARY KEY,
    FOREIGN KEY (user_id) REFERENCES Lib_User(user_id) ON
DELETE CASCADE
);
```

LIB_ADMIN

```
CREATE TABLE Lib_Librarian (
    user_id VARCHAR(15) PRIMARY KEY,
    start_working_hour INT CHECK (start_working_hour
BETWEEN 0 AND 23),
    end_working_hour INT CHECK (end_working_hour BETWEEN 0
AND 23),
    FOREIGN KEY (user_id) REFERENCES Lib_User(user_id) ON
DELETE CASCADE
);
```

LIB_LIBRARIAN

```
CREATE TABLE Lib_Librarian (
    user_id VARCHAR(15) PRIMARY KEY,
    start_working_hour INT CHECK (start_working_hour
BETWEEN 0 AND 23),
    end_working_hour INT CHECK (end_working_hour BETWEEN 0
AND 23),
    FOREIGN KEY (user_id) REFERENCES Lib_User(user_id) ON
DELETE CASCADE
);
```

ITEM

```
CREATE TABLE Item (
    item_id VARCHAR(15) PRIMARY KEY
);
```

ROOM

```
CREATE TABLE Room (
    room_id VARCHAR(15) PRIMARY KEY,
    building_no INT,
    room_no VARCHAR(50),
    capacity INT,
    status VARCHAR(50),
    FOREIGN KEY (room_id) REFERENCES Item(item_id) ON
DELETE CASCADE
);
```

HARD_COPIES

```
 CREATE TABLE Hard_Copies (
    book_id VARCHAR(15),
    copy_id VARCHAR(15),
    status VARCHAR(50) CHECK (status IN ('Available',
'Unavailable')),
    edition VARCHAR(50),
    year_published INT,
    location VARCHAR(15),
    PRIMARY KEY (book_id, copy_id),
    FOREIGN KEY (book_id) REFERENCES Book(book_id) ON
DELETE CASCADE,
    FOREIGN KEY (location) REFERENCES
Location(location_id) ON DELETE SET NULL
);
```

EBOOKS

```
 CREATE TABLE Ebooks (
    book_id VARCHAR(15) PRIMARY KEY,
    isbn VARCHAR(50),
    format VARCHAR(50) CHECK (format IN ('PDF', 'EPub')),
    url VARCHAR(255),
    file_size_mb FLOAT,
    FOREIGN KEY (book_id) REFERENCES Book(book_id) ON
DELETE CASCADE
);
```

AUTHOR

```
CREATE TABLE Author (
    author_id VARCHAR(15) PRIMARY KEY,
    f_name VARCHAR2(100),
    l_name VARCHAR2(100)
);
```

REVIEW

```
CREATE TABLE Review (
    review_id VARCHAR(15) PRIMARY KEY,
    rating INT CHECK (rating BETWEEN 1 AND 5),
    comments CLOB,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    reviewer VARCHAR(15),
    book VARCHAR(15),
    FOREIGN KEY (reviewer) REFERENCES Lib_Member(user_id)
ON DELETE SET NULL,
    FOREIGN KEY (book) REFERENCES Book(book_id) ON DELETE
CASCADE
);
```

FINE

```
 CREATE TABLE Fine (
    fine_id VARCHAR(15) PRIMARY KEY,
    amount DECIMAL(10,2),
    reason VARCHAR(255),
    status VARCHAR(50),
    incurred_date DATE,
    due_date DATE,
    paid_date DATE,
    payer VARCHAR(15),
    book_id VARCHAR(15),
    copy_id VARCHAR(15),
    FOREIGN KEY (payer) REFERENCES Lib_Member(user_id) ON
DELETE SET NULL,
    FOREIGN KEY (book_id, copy_id) REFERENCES
Hard_Copies(book_id, copy_id) ) ON DELETE SET NULL
);
```

REQUESTS

```
 CREATE TABLE Requests (
    request_id VARCHAR(15) PRIMARY KEY,
    type VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    message CLOB,
    status VARCHAR(50) CHECK (status IN ('Pending',
'Ongoing', 'Resolved')),
    sender VARCHAR(15),
    receiver VARCHAR(15),
    FOREIGN KEY (sender) REFERENCES Lib_Member(user_id) ON
DELETE SET NULL,
    FOREIGN KEY (receiver) REFERENCES
Lib_Librarian(user_id) ON DELETE SET NULL
);
```

MEMBER_COURSE	CREATE TABLE Member_Course (
	    member_id VARCHAR(15),
	    course_code VARCHAR(15),
	    PRIMARY KEY (member_id, course_code),
	    FOREIGN KEY (member_id) REFERENCES Lib_Member(user_id)
	ON DELETE CASCADE,
	    FOREIGN KEY (course_code) REFERENCES Course(course_id)
	ON DELETE CASCADE
	);

MEMBER_EBOOKS	CREATE TABLE Member_Ebooks (
	    member_id VARCHAR(15),
	    ebook_id VARCHAR(15),
	    PRIMARY KEY (member_id, ebook_id),
	    FOREIGN KEY (member_id) REFERENCES Lib_Member(user_id)
	ON DELETE CASCADE,
	    FOREIGN KEY (ebook_id) REFERENCES Ebooks(book_id) ON
	DELETE CASCADE
	);

MEMBER_ROOM	CREATE TABLE Member_Room (
	    booking_id VARCHAR(15) PRIMARY KEY,
	    member_id VARCHAR(15),
	    room_id VARCHAR(15),
	    reservation_date DATE DEFAULT SYSDATE,
	    start_date TIMESTAMP,
	    end_date TIMESTAMP,
	    FOREIGN KEY (member_id) REFERENCES Lib_Member(user_id)
	ON DELETE CASCADE,
	    FOREIGN KEY (room_id) REFERENCES Room(room_id) ON
	DELETE CASCADE,
	    CHECK ((end_date - start_date) <= INTERVAL '2' HOUR)
	-- Ensures booking duration does not exceed 2 hours
	);

MEMBER_HARDCOPY	CREATE TABLE Member_HardCopy (
	    borrow_id VARCHAR(15) PRIMARY KEY,
	    member VARCHAR(15),
	    book VARCHAR(15),

	```
	    bcopy VARCHAR(15),
	    issue_date DATE DEFAULT SYSDATE,
	    due_date DATE,
	    return_date DATE,
	    checkin_condition VARCHAR(255) CHECK
	(checkin_condition IN ('New', 'Damaged')),  -- Only allows
	'New' or 'Damaged'
	    checkout_condition VARCHAR(255) CHECK
	(checkout_condition IN ('New', 'Damaged')),  -- Only
	allows 'New' or 'Damaged'
	    FOREIGN KEY (member) REFERENCES Lib_Member(user_id) ON
	DELETE CASCADE,
	    FOREIGN KEY (book, bcopy) REFERENCES
	Hard_Copies(book_id, copy_id) ON DELETE CASCADE
	);
	```

BOOK_AUTHOR	
	```
	CREATE TABLE Book_Author (
	    book_id VARCHAR(15),
	    author_id VARCHAR(15),
	    PRIMARY KEY (book_id, author_id),
	    FOREIGN KEY (book_id) REFERENCES Book(book_id) ON
	DELETE CASCADE,
	    FOREIGN KEY (author_id) REFERENCES Author(author_id)
	ON DELETE CASCADE
	);
	```

BOOK_COURSE	
	```
	CREATE TABLE Book_Course (
	    book_id VARCHAR(15),
	    course_id VARCHAR(15),
	    PRIMARY KEY (book_id, course_id),
	    FOREIGN KEY (book_id) REFERENCES Book(book_id) ON
	DELETE CASCADE,
	    FOREIGN KEY (course_id) REFERENCES Course(course_id)
	ON DELETE CASCADE
	);
	```

BOOK_CATEGORY	
	```
	CREATE TABLE Book_Category (
	    book_id VARCHAR(15),
	    category_id VARCHAR(15),
	    PRIMARY KEY (book_id, category_id),
	```

```
    FOREIGN KEY (book_id) REFERENCES Book(book_id) ON
DELETE CASCADE,
    FOREIGN KEY (category_id) REFERENCES
Category(category_id) ON DELETE CASCADE
);
```

BOOK_MAJOR

```
CREATE TABLE Book_Major (
    book_id VARCHAR(15),
    major_id VARCHAR(15),
    PRIMARY KEY (book_id, major_id),
    FOREIGN KEY (book_id) REFERENCES Book(book_id) ON
DELETE CASCADE,
    FOREIGN KEY (major_id) REFERENCES Major(major_id) ON
DELETE CASCADE
);
```

Indexing: It is known that indexes help reduce full-table scans and are optimized for search (Sun 2023). In the context of a small LMS with a limited dataset, applying indexing to frequently queried columns might not be ideal due to a host of reasons. Firstly, indexing adds overhead to the database system, including **additional storage space and computational resources**, which may not be justified for a small-scale system with limited data. Secondly, the benefits of indexing, such as **improved query performance, are more pronounced in larger databases** with extensive datasets and complex queries. In a small LMS with a limited sample of data, the performance gains from indexing **may not be significant enough to outweigh the associated costs**. Additionally, frequent updates or modifications to indexed columns can lead to **performance degradation**, as indexes need to be maintained alongside the data. Therefore, in our scenario, it may be more practical to prioritize simplicity and efficiency in database operations, rather than introducing unnecessary complexity with indexing on frequently queried columns.

Step 3: DML statements (Insert records to populate sample data)

a) Table name: Major

Description: Stores information about academic majors offered at the university. Each major is identified by a unique ID and is associated with a specific school within the university (TBS/SSET/SCD/SEUP).

Example data:

```
-- Populate Major (MySQL)
INSERT INTO Major (major_id, school, name) VALUES
('BP343', 'TBS', 'Business'),
('BP214', 'SCD', 'Game Design'),
('BP162', 'SSET', 'Information Technology'),
('BP317', 'SEUP', 'Languages');
```

***

```
-- Populate Major (Oracle)
INSERT ALL
  INTO Major (major_id, school, name) VALUES ('BP343', 'TBS', 'Business')
  INTO Major (major_id, school, name) VALUES ('BP214', 'SCD', 'Game Design')
  INTO Major (major_id, school, name) VALUES ('BP162', 'SSET', 'Information Technology')
  INTO Major (major_id, school, name) VALUES ('BP317', 'SEUP', 'Languages')
SELECT * FROM dual;
```

b) Table name: Course

Description: Contains details about courses available at the university, including a unique course ID and the course name.

Example data:

```
-- Populate Course
INSERT INTO Course (course_id, name) VALUES
('ISYS3414', 'Practical Database Concepts'),
('MATH2394', 'Engineering Mathematics');
```

c) Table name: Category

Description: Used to categorize books and other library materials into various subject areas or themes, helping with organization and searchability. ID Format: ‘CATxxx’

Example data:

```
-- Populate Category
INSERT INTO Category (category_id, name) VALUES
('CAT001', 'Mathematics'),
('CAT002', 'Physics');
```

d) Table name: Location

Description: Specifies physical locations within the library, such as specific rooms, shelves, or areas where items are stored or used. The campus used here is referenced from RMIT (Melbourne, Beanland – SGS, Hanoi). ID Format: ‘LOCxxx’

Example data:

```
-- Populate Location
INSERT INTO Location (location_id, campus, floor, shelf_no, line_no) VALUES
('LOC001', 'Melbourne', 1, 'C1', 1),
('LOC002', 'Beanland', 2, 'A1', 2),
('LOC003', 'Hanoi', 1, 'A2', 3);
```

e) Table name: Lib_User

Description: General user table that includes all individuals who have access to the library system, such as students, staff, librarians, and administrators, which is indicated by the attribute user_type. It stores usernames, names, emails, passwords, and user roles. All users have the same ID format of ‘Uxxx’.

Example data:

```
-- Populate Lib_User
INSERT INTO Lib_User (user_id, username, f_name, l_name, email, pwd, user_type) VALUES
('U003', 'johnathan.crellin', 'Johnathan', 'Crellin', 'johncrel@gmail.com', '123', 'Staff'),
('U010', 'iris.nguyen', 'Iris', 'Nguyen', 'ngmaihuong@gmail.com', '123', 'Student'),
('U011', 'andrew.tran', 'Andrew', 'Tran', 'litvandrius@gmail.com', '123', 'Librarian'),
('U016', 'vinh.truong', 'Vinh', 'Truong', 'trngxuanvinh@gmail.com', 'HD', 'Admin');
```

f) Table name: Lib_Admin

Description: Specific for library administrators. This table links back to Lib_User and includes administrative roles responsible for the management of the library system.

Example data:

```
-- Populate Lib_Admin
INSERT INTO Lib_Admin (user_id) VALUES
('U016');
```

g) Table name: Lib_Librarian

Description: Contains details of librarians who manage daily operations of the library. This also references back to the Lib_User.

Example data:

```
-- Populate Lib_Librarian
INSERT INTO Lib_Librarian (user_id, start_working_hour, end_working_hour) VALUES
('U011', 7, 15);
```

h) Table name: Lib_Member

Description: Extends Lib_User for library members (typically students and staff/faculty) who utilise items. It tracks the total number of books borrowed and fines paid, linking to their academic program.

Example data:

```
-- Populate Lib_Member
INSERT INTO Lib_Member (user_id, total_book_borrowed, total_fine_paid, program_code)
VALUES
('U003', 0, 25.00, NULL),
('U010', 0, 0.00, 'BP309');
```

i) Table name: Item

Description: An abstract entity used to generalize different items available in the library, in our case: books and rooms.

Example data:

```
-- Populate Item
INSERT INTO Item (item_id) VALUES
('B001'),
('R001');
```

j) Table name: Room

Description: Details rooms available for booking within the library, such as study rooms or meeting rooms. It includes capacity (for how many people) and status (Available/Unavailable). ID Format: ‘Rxxx’

Example data:

```
-- Populate Room
INSERT INTO Room (room_id, building_no, room_no, capacity, status) VALUES
('R001', 1, '101', 5, 'Available'),
('R010', 2, '205', 5, 'Unavailable');
```

k) Table name: Book

Description: Stores detailed information about each book, including title, number of copies, descriptions, ratings, and language. Book ids are marked 'Bxxx'.

Example data:

```
-- Populate Book
INSERT INTO Book (book_id, title, num_of_copies, description, ratings, language) VALUES
('B001', 'Business Dynamics', 5, 'Insights into evolving market trends.', 4.0, 'English'),
('B002', 'Principles of Game Design', 3, 'Foundations of game development and design.', 4.33,
'English');
```

l) Table name: Ebooks

Description: Details about electronic books available in the library, including ISBN, format (PDF, EPub), URL for access, and file size.

Example data:

```
-- Populate Ebooks
INSERT INTO Ebooks (book_id, isbn, format, url, file_size_mb) VALUES
('B001', '978-0-00-000001-2', 'PDF', 'https://example.com/ebooks/business_dynamics.pdf', 2.5),
('B002', '978-0-00-000002-9', 'EPub', 'https://example.com/ebooks/game_design.epub', 1.5);
```

m) Table name: Hard_Copies

Description: Specific details of physical copies of books, including their status (available, unavailable), edition, year published, and physical location within the library. The number of book copy records of one book reflects the num_of_copies attribute in the Book table. The status (Available/Unavailable) will be continuously updated based on the borrowing transactions. Book copy ids are named after their book's id, followed by 'Cxx'.

Example data:

```
-- Populate Hard_Copies
INSERT INTO Hard_Copies (book_id, copy_id, status, edition, year_published, location) VALUES
('B001', 'B001C1', 'Available', 'Seventh', 2009, 'LOC001'),
('B004', 'B004C1', 'Unavailable', 'First', 2021, 'LOC004');
```

n) Table name: Author

Description: Contains information about authors of books available in the library, including first and last names.

Example data:

```
-- Populate Author
INSERT INTO Author (author_id, f_name, l_name) VALUES
('AUTH001', 'John', 'Smith');
```

o) Table name: Fine

Description: Tracks fines incurred by library members for overdue/lost books, including the amount, reasons, and payment status. Fine identifiers are under the ‘FINExxx’ format.

Example data:

```
-- Populate Fine
INSERT INTO Fine (fine_id, amount, reason, status, incurred_date, due_date, paid_date, payer,
book_id, copy_id) VALUES
('FINE001', 50.00, 'Late return', 'Unpaid', DATE '2023-11-11', DATE '2024-1-11', NULL, 'U010',
'B005', 'B005C4');
```

p) Table name: Review

Description: Allows users to post reviews of books, including a rating system, comments, and timestamps for when reviews are posted. ID format: ‘REVxxx’

Example data:

```
-- Populate Review
INSERT INTO Review (review_id, rating, comments, reviewer, book) VALUES
('REV001', 5, 'Absolutely insightful with practical tips on navigating market trends.', 'U001', 'B001');
```

q) Table name: Requests

Description: Manages various types of requests made by users, such as reservations, book requests, or information queries. Includes details about the request type, status (Pending/Ongoing/Resolved), and involved parties. ID format: ‘REQxxx’.

Example data:

```
-- Populate Requests
INSERT INTO Requests (request_id, type, message, status, sender, receiver) VALUES
('REQ001', 'Book Reservation', 'Request to reserve "Business Dynamics" for upcoming coursework.',
'Pending', 'U001', 'U011');
```

r) Table name: Member_Course

Description: Links library members to courses they are enrolled in, facilitating management of course-specific materials.

Example data:

```
-- Populate Member_Course
INSERT INTO Member_Course (member_id, course_code) VALUES
('U005', 'COSC2430'),
('U006', 'COSC2440'),
('U006', 'COSC2652'),
('U007', 'ISYS3414'),
('U007', 'COMM2596');
```

s) Table name: Member_Ebooks

Description: Tracks which ebooks a member has bookmarked, useful for usage tracking.

Example data:

```
-- Populate Member_Ebooks
INSERT INTO Member_Ebooks (member_id, ebook_id) VALUES
('U001', 'B001'),
('U001', 'B019'),
('U002', 'B002');
```

t) Table name: Member_Room

Description: Manages room bookings made by members, including time slots and room details. The interval between the start date and end date should not exceed 2 hours.

Example data:

```
-- Populate Member_Room
INSERT INTO Member_Room (booking_id, member_id, room_id, reservation_date, start_date,
end_date) VALUES
('BKG001', 'U001', 'R001', DATE '2023-10-01', TIMESTAMP '2023-10-01 09:00:00', TIMESTAMP
'2023-10-01 11:00:00'),
('BKG020', 'U009', 'R010', DATE '2024-04-27', TIMESTAMP '2024-10-19 10:00:00', TIMESTAMP
'2024-10-19 12:00:00');
```

u) Table name: Member_HardCopy

Description: Details the borrowing records for physical copies of books by members, including dates and conditions (New/Damanged) of the items when checked out and returned.

Example data:

```
-- Populate Member_HardCopy
INSERT INTO Member_HardCopy (borrow_id, member, book, bcopy, issue_date, due_date,
return_date, checkin_condition, checkout_condition) VALUES
('BRW001', 'U001', 'B001', 'B001C1', DATE '2023-10-01', DATE '2023-10-15', DATE '2023-10-08',
'New', 'New'),
('BRW003', 'U002', 'B002', 'B002C1', DATE '2023-10-03', DATE '2023-10-17', DATE '2023-10-10',
'Damaged', 'Damaged'),
('BRW005', 'U003', 'B004', 'B004C1', DATE '2023-10-05', DATE '2023-10-19', DATE '2023-10-12',
'New', 'Damaged'),
('BRW019', 'U009', 'B004', 'B004C1', DATE '2023-10-19', DATE '2023-11-02', NULL, 'Damaged',
NULL);
```

v) Table name: Book_Author

Description: A junction table that manages the many-to-many relationship between books and their authors.

Example data:

```
-- Populate Book_Author
INSERT INTO Book_Author (book_id, author_id) VALUES
('B005', 'AUTH005'), ('B005', 'AUTH013');
```

w) Table name: Book_Course

Description: Links books to courses that recommend or require them, aiding in resource allocation and accessibility.

Example data:

```
-- Populate Book_Course
INSERT INTO Book_Course (book_id, course_id) VALUES
('B027', 'ISYS3414'), ('B027', 'COSC2440'), ('B028', 'EEET2601'), ('B028', 'EEET2599'),
('B030', 'FOHO1024');
```

x) Table name: Book_Category

Description: Manages the relationship between books and their categories, helping categorize library resources effectively.

Example data:

```
-- Populate Book_Category
INSERT INTO Book_Category (book_id, category_id) VALUES
('B001', 'CAT016');
```

y) Table name: Book_Major

Description: Links books to academic majors, indicating resources that are particularly relevant to specific fields of study.

Example data:

```
-- Populate Book_Major
INSERT INTO Book_Major (book_id, major_id) VALUES
('B026', 'BP309'), ('B027', 'BP162'), ('B028', 'BH073'),
('B027', 'BH120'), ('B028', 'BH123'), ('B030', 'BP199');
```

3.1. Queries

All queries used for the application can be found in the file ***ISYS3414_Query_Group7_VINHFC.sql***.

3.1.1. For in-app custom user authentication and authorization

TYPE	PL/SQL

Authentication scheme: Custom function	`-- Authentication function`
	`FUNCTION user_aut(`
	`p_username IN VARCHAR2, --User_Name`
	`p_password IN VARCHAR2 -- Password`
	`)`
	`RETURN BOOLEAN`
	`AS`
	`lc_pwd_exit VARCHAR2 (1);`
	`BEGIN`
	`-- Validate whether the user exits or not`
	`SELECT 'Y'`
	`INTO lc_pwd_exit`
	`FROM Lib_User`
	`WHERE upper(username) = UPPER (p_username) AND pwd =`
	`p_password;`
	`RETURN TRUE;`
	`EXCEPTION`
	`WHEN NO_DATA_FOUND`
	`THEN`
	`RETURN FALSE;`
	`END user_aut;`

Authorization scheme: PL/SQL Function Returning Boolean
 	
(ADMIN_ONLY)
	```
	DECLARE
	result VARCHAR2(10);
	BEGIN
	SELECT USER_TYPE INTO result FROM Lib_User WHERE
	UPPER(USERNAME)=V('APP USER');
	IF NVL (result, 'x') = 'Admin' THEN
	RETURN TRUE;
	END IF;
	RETURN FALSE;
	END;
	```

Authorization scheme: PL/SQL Function Returning Boolean
 	
(LIBRARIAN_ONLY)
	```
	DECLARE
	    result VARCHAR2(10);
	BEGIN
	    SELECT USER_TYPE INTO result FROM Lib_User WHERE
	UPPER(USERNAME)=V('APP USER');
	    IF NVL (result, 'x') = 'Librarian' THEN
	        RETURN TRUE;
	    END IF;
	    RETURN FALSE;
	END;
	```

Authorization scheme: PL/SQL Function Returning Boolean
 	
(MEMBER_ONLY)
	```
	DECLARE
	    result VARCHAR2(10);
	BEGIN
	    SELECT user_type INTO result FROM Lib_User WHERE
	UPPER(USERNAME) = UPPER(V('APP_USER'));

	    IF NVL(result, 'x') IN ('Student', 'Staff') THEN
	        RETURN TRUE;
	    ELSE
	        RETURN FALSE;
	    END IF;
	END;
	```

3.1.2. Triggers for ID-auto-generation before inserting new records into the tables.

APPLIED TABLE	Oracle	MySQL

Book	CREATE SEQUENCE book_seq
	START WITH 1 INCREMENT BY
	1;
	CREATE OR REPLACE TRIGGER
	trg_before_insert_book
	BEFORE INSERT ON Book
	FOR EACH ROW
	BEGIN
	:NEW.book_id := 'B' ||
	TO_CHAR(book_seq.NEXTVAL,
	'FM000');
	END;
		DELIMITER $$
		CREATE TRIGGER
		trg_before_insert_book
		BEFORE INSERT ON Book
		FOR EACH ROW
		BEGIN
		    SELECT CONCAT('B',
		LPAD(IFNULL(MAX(CAST(SUBSTRING(bo
		ok_id, 2) AS UNSIGNED)), 0) + 1,
		3, '0')) INTO @next_book_id
		    FROM Book;
		    SET NEW.book_id =
		@next_book_id;
		END$$
		DELIMITER ;

Fine	CREATE SEQUENCE fine_seq
	START WITH 1 INCREMENT BY
	1;
	CREATE OR REPLACE TRIGGER
	trg_before_insert_fine
	BEFORE INSERT ON Fine
	FOR EACH ROW
	BEGIN
	:NEW.fine_id := 'FINE' ||
	TO_CHAR(fine_seq.NEXTVAL,
	'FM000');
	END;
		DELIMITER $$
		CREATE TRIGGER
		trg_before_insert_fine
		BEFORE INSERT ON Fine
		FOR EACH ROW
		BEGIN
		    SELECT CONCAT('FINE',
		LPAD(IFNULL(MAX(CAST(SUBSTRING(fi
		ne_id, 5) AS UNSIGNED)), 0) + 1,
		3, '0')) INTO @next_fine_id
		    FROM Fine;
		    SET NEW.fine_id =
		@next_fine_id;
		END$$
		DELIMITER ;

Room	CREATE SEQUENCE room_seq
	START WITH 1 INCREMENT BY
	1;
	CREATE OR REPLACE TRIGGER
	trg_before_insert_room
	BEFORE INSERT ON Room
	FOR EACH ROW
	BEGIN
	:NEW.room_id := 'R' ||
	TO_CHAR(room_seq.NEXTVAL,
	'FM000');
	END;
		DELIMITER $$
		CREATE TRIGGER
		trg_before_insert_room
		BEFORE INSERT ON Room
		FOR EACH ROW
		BEGIN
		    SELECT CONCAT('R',
		LPAD(IFNULL(MAX(CAST(SUBSTRING(ro
		om_id, 2) AS UNSIGNED)), 0) + 1,
		3, '0')) INTO @next_room_id
		    FROM Room;
		    SET NEW.room_id =
		@next_room_id;
		END$$
		DELIMITER ;

Member_HardCopy	CREATE SEQUENCE borrow_seq START WITH 1 INCREMENT BY 1;
	CREATE OR REPLACE TRIGGER trg_before_insert_borrow
	BEFORE INSERT ON Member_HardCopy
	FOR EACH ROW
	BEGIN
	:NEW.borrow_id := 'BRW' || TO_CHAR(borrow_seq.NEXTVAL, 'FM000');
	END;
		DELIMITER $$
		CREATE TRIGGER trg_set_borrow_id
		BEFORE INSERT ON Member_HardCopy
		FOR EACH ROW
		BEGIN
		    DECLARE max_id INT;
		    SELECT IFNULL(MAX(CAST(SUBSTRING(borrow_id, 4) AS UNSIGNED)), 0) INTO max_id FROM Member_HardCopy;
		    SET NEW.borrow_id = CONCAT('BRW', LPAD(max_id + 1, 3, '0'));
		END$$
		DELIMITER ;

Member_Room	CREATE SEQUENCE booking_seq START WITH 1 INCREMENT BY 1;
	CREATE OR REPLACE TRIGGER trg_before_insert_member_room_booking
	BEFORE INSERT ON Member_Room
	FOR EACH ROW
	BEGIN
	:NEW.booking_id := 'BKG' || TO_CHAR(booking_seq.NEXTVAL, 'FM000');
	END;
		DELIMITER $$
		CREATE TRIGGER trg_before_insert_member_room_booking
		BEFORE INSERT ON Member_Room
		FOR EACH ROW
		BEGIN
		    DECLARE max_id INT;
		    DECLARE prefix CHAR(3) DEFAULT 'BKG';
		    DECLARE new_id CHAR(15);
		    SELECT MAX(CAST(SUBSTRING(booking_id, 4) AS UNSIGNED)) INTO max_id FROM Member_Room;
		    SET max_id = IFNULL(max_id, 0) + 1;
		    SET new_id = CONCAT(prefix, LPAD(max_id, 3, '0'));
		    SET NEW.booking_id = new_id;
		END$$
		DELIMITER ;

Request	CREATE SEQUENCE request_seq START WITH 1 INCREMENT BY 1;
	CREATE OR REPLACE TRIGGER trg_before_insert_requests
	BEFORE INSERT ON Requests
	FOR EACH ROW
	BEGIN
	:NEW.request_id := 'REQ' ||
		DELIMITER $$
		CREATE TRIGGER trg_before_insert_requests
		BEFORE INSERT ON Requests
		FOR EACH ROW
		BEGIN
		    DECLARE max_id INT;
		    SELECT IFNULL(MAX(CAST(SUBSTRING(request_id, 4) AS UNSIGNED)), 0) INTO max_id FROM Requests;

	TO_CHAR(request_seq.NEXTVAL, 'FM000');
	END;
		SET NEW.request_id = CONCAT('REQ', LPAD(max_id + 1, 3, '0'));
		END$$
		DELIMITER ;

Review	CREATE SEQUENCE review_seq START WITH 1 INCREMENT BY 1;
	CREATE OR REPLACE TRIGGER trg_before_insert_review
	BEFORE INSERT ON Review
	FOR EACH ROW
	BEGIN
	:NEW.review_id := 'REV' || TO_CHAR(review_seq.NEXTVAL, 'FM000');
	END;
		DELIMITER $$
		CREATE TRIGGER trg_before_insert_review
		BEFORE INSERT ON Review
		FOR EACH ROW
		BEGIN
		    DECLARE max_id INT;
		    SELECT IFNULL(MAX(CAST(SUBSTRING(review_id, 4) AS UNSIGNED)), 0) INTO max_id FROM Review;
		    SET NEW.review_id = CONCAT('REV', LPAD(max_id + 1, 3, '0'));
		END$$
		DELIMITER ;

Location	CREATE SEQUENCE location_seq START WITH 1 INCREMENT BY 1;
	CREATE OR REPLACE TRIGGER trg_before_insert_location
	BEFORE INSERT ON Location
	FOR EACH ROW
	BEGIN
	:NEW.review_id := 'LOC' || TO_CHAR(review_seq.NEXTVAL, 'FM000');
	END;	

Lib_User	CREATE SEQUENCE user_seq START WITH 1 INCREMENT BY 1;
	CREATE OR REPLACE TRIGGER trg_before_insert_user
	BEFORE INSERT ON Lib_User
	FOR EACH ROW
	BEGIN
	:NEW.user_id := 'U' || TO_CHAR(user_seq.NEXTVAL, 'FM000');
	END;
		DELIMITER $$
		CREATE TRIGGER trg_before_insert_lib_user
		BEFORE INSERT ON Lib_User
		FOR EACH ROW
		BEGIN
		    DECLARE max_num INT DEFAULT 0;
		    DECLARE new_id CHAR(10);
		    SELECT MAX(CAST(SUBSTRING(user_id, 2) AS UNSIGNED)) INTO max_num FROM Lib_User;
		    SET max_num = IFNULL(max_num, 0) + 1;

		`SET new_id = CONCAT('U', LPAD(max_num, 3, '0'));`
		`SET NEW.user_id = new_id;`
		`END$$`
		`DELIMITER ;`

3.1.3. Triggers for specific scenarios

a) Borrowing book process

When a member borrows a new book copy, automatically calculate the due_date by 2-month time from the current system date. Then we make that book copy unavailable for other user views by changing the status from ‘Available’ to ‘Unavailable’. Lastly, update the number of books currently being borrowed by the member by incrementing the current value by 1.

```
DELIMITER $$
CREATE TRIGGER trg_borrow_book
BEFORE INSERT ON Member_HardCopy
FOR EACH ROW
BEGIN
    -- Set the due_date to be 2 months after the issue_date directly in
the NEW row
    SET NEW.due_date = DATE_ADD(NEW.issue_date, INTERVAL 2 MONTH);

    -- Update the status of the hard copy to 'Unavailable' in Hard_Copies
table
    UPDATE Hard_Copies
    SET status = 'Unavailable'
    WHERE book_id = NEW.book AND copy_id = NEW.bcopy;

    -- Increment the total_book_borrowed for the user in the Lib_Member
table
    UPDATE Lib_Member
    SET total_book_borrowed = total_book_borrowed + 1
    WHERE user_id = NEW.member;
END$$
DELIMITER ;
```

```
DELIMITER $$
CREATE TRIGGER trg_check_borrow_limit
BEFORE INSERT ON Member_HardCopy
FOR EACH ROW
BEGIN
    -- Variables to store user details
    DECLARE v_user_type VARCHAR(10);
    DECLARE v_total_borrowed INT;

    -- Retrieve the user type and current total books borrowed
    SELECT user_type, total_book_borrowed INTO v_user_type,
v_total_borrowed
    FROM Lib_Member lm, Lib_User lu
    WHERE lm.user_id = lu.user_id and lu.user_id = NEW.member;
```

b) Returning book process

When a member returns their book to the library, automatically get the return_date when updating the checkout_condition to ‘New’/‘Damaged’ in the Member_Hardcopy table. Next, update the status of that hard_copies ID to ‘Available’ again , and decrement the total_book_borrowed of the user_id by 1. After updating, insert a fine record if checkin_condition is not the same as checkout_condition. The type of fine is ‘Damaged book’.

```
DELIMITER $$
CREATE TRIGGER trg_before_update_member_hardcopy
BEFORE UPDATE ON Member_HardCopy
FOR EACH ROW
BEGIN
    -- Set return_date to current date when updating checkout_condition
    IF NEW.checkout_condition IN ('New', 'Damaged') THEN
        SET NEW.return_date = CURRENT_DATE();
    END IF;
END$$
DELIMITER ;
```

```
DELIMITER $$
CREATE TRIGGER trg_after_update_member_hardcopy
AFTER UPDATE ON Member_HardCopy
FOR EACH ROW
BEGIN
    -- Update the status of the hard copy to 'Available' after the
checkout_condition is updated
    IF NEW.checkout_condition IS NOT NULL THEN
        UPDATE Hard_Copies
        SET status = 'Available'
        WHERE book_id = NEW.book AND copy_id = NEW.bcopy;

        -- Decrement total_book_borrowed for the user
        UPDATE Lib_Member
        SET total_book_borrowed = total_book_borrowed - 1
        WHERE user_id = NEW.member;
    END IF;

    -- Insert a fine if the checkin_condition is not the same as
checkout_condition
    IF NEW.checkout_condition IN ('New', 'Damaged') AND
NEW.checkin_condition <> NEW.checkout_condition THEN
```

```
        INSERT INTO Fine (amount, reason, status, incurred_date,
due_date, payer, book_id, copy_id)
        VALUES (50, 'Damaged book', 'Unpaid', NEW.issue_date,
NEW.due_date, NEW.member, NEW.book, NEW.bcopy);
    END IF;
END$$
DELIMITER ;
```

c) Paying fine process

When a member pays their fine, the librarian will update the status of that fine_id from ‘Unpaid’ to ‘Paid’ and the system automatically get the current date as the paid_date attribute. Upon updating the fine table, update the Lib_Member table where the user_id is the Fine.payer and increment the total_fine_paid attribute to the fine amount respectively.

```
DELIMITER $$
CREATE TRIGGER trg_fine_paid
BEFORE UPDATE ON Fine
FOR EACH ROW
BEGIN
    -- Check if the fine status is being updated to 'Paid'
    IF OLD.status <> 'Paid' AND NEW.status = 'Paid' THEN
        -- Set the paid_date to the current date
        SET NEW.paid_date = CURRENT_DATE();

        -- Since we are in a BEFORE trigger, the following update will be
executed
        -- AFTER the current transaction commits, avoiding the problem of
updating
        -- the same table in an AFTER trigger
    END IF;
END$$
DELIMITER ;
```

```
DELIMITER $$
CREATE TRIGGER trg_fine_paid2
AFTER UPDATE ON Fine
FOR EACH ROW
BEGIN
    -- Check if the fine status has been updated to 'Paid'
    IF OLD.status <> 'Paid' AND NEW.status = 'Paid' THEN
        -- Update the total_fine_paid in the Lib_Member table
        UPDATE Lib_Member
        SET total_fine_paid = total_fine_paid + NEW.amount
        WHERE user_id = NEW.payer;

    END IF;
```

```
END$$
DELIMITER ;
```

d) Room booking process

When a member books a room: after inserting a record into the member_room, set the room_id's status in the room table to "Unavailable". Also help me come up with a mechanism so that no member can book more than one room within one timeslot (no overlap). And update the status of the room back to 'available' when the current system time passed the member_room.end_date

```
DELIMITER $$
CREATE TRIGGER trg_after_insert_member_room
AFTER INSERT ON Member_Room
FOR EACH ROW
BEGIN
    -- Set the room status to 'Unavailable'
    UPDATE Room
    SET status = 'Unavailable'
    WHERE room_id = NEW.room_id;
END$$
DELIMITER ;
```

```
DELIMITER $$
CREATE TRIGGER trg_before_insert_member_room
BEFORE INSERT ON Member_Room
FOR EACH ROW
BEGIN
    DECLARE overlap_count INT;
    -- Check for overlapping bookings
    SELECT COUNT(*) INTO overlap_count
    FROM Member_Room
    WHERE member_id = NEW.member_id
    AND room_id = NEW.room_id
    AND (
        (NEW.start_date BETWEEN start_date AND end_date) OR
        (NEW.end_date BETWEEN start_date AND end_date) OR
        (start_date BETWEEN NEW.start_date AND NEW.end_date) OR
        (end_date BETWEEN NEW.start_date AND NEW.end_date)
    );

    -- If there's an overlap, prevent insertion
    IF overlap_count > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot book more than one room within the
same timeslot.';
    END IF;
```

```
END$$
DELIMITER ;
```

```
SET GLOBAL event_scheduler = ON;
DELIMITER $$
CREATE EVENT ev_update_room_status
ON SCHEDULE EVERY 1 MINUTE
DO
    BEGIN
        -- Update the room status based on whether there's an ongoing
booking that overlaps the current timestamp
        UPDATE Room
        LEFT JOIN Member_Room ON Room.room_id = Member_Room.room_id
            AND CURRENT_TIMESTAMP BETWEEN Member_Room.start_date AND
Member_Room.end_date
        SET Room.status = IF(Member_Room.room_id IS NULL, 'Available',
'Unavailable');
    END$$
DELIMITER ;
```

e) Giving book review process

When a user leave a review for a book_id, recalculate the avg(rating) for that book, then update the Book table by updating that value into the ratings attribute into the book_id being reviewed.

```
DELIMITER $$
CREATE TRIGGER trg_after_insert_review
AFTER INSERT ON Review
FOR EACH ROW
BEGIN
    UPDATE Book
    SET ratings = (SELECT AVG(rating) FROM Review WHERE book = NEW.book)
    WHERE book_id = NEW.book;
END$$
DELIMITER ;
```

f) Add a new hard copy record for a book process

When the librarian inserts a new book copy record into hard_copies, update the num_of_copies corresponding to that book_id by plusing 1.

```
DELIMITER $$
CREATE TRIGGER trg_after_insert_hardcopy
AFTER INSERT ON Hard_Copies
FOR EACH ROW
BEGIN
    UPDATE Book
    SET num_of_copies = num_of_copies + 1
    WHERE book_id = NEW.book_id;
```

```
END$$
DELIMITER ;
```

Application Features

a) Application and Accounts

The authentication scheme mentioned above allows application users to log in using their credentials stored in the database, particularly the Lib_User table. Currently our sample dataset has 16 users, and we can choose one among them for signing in.

Application URL:

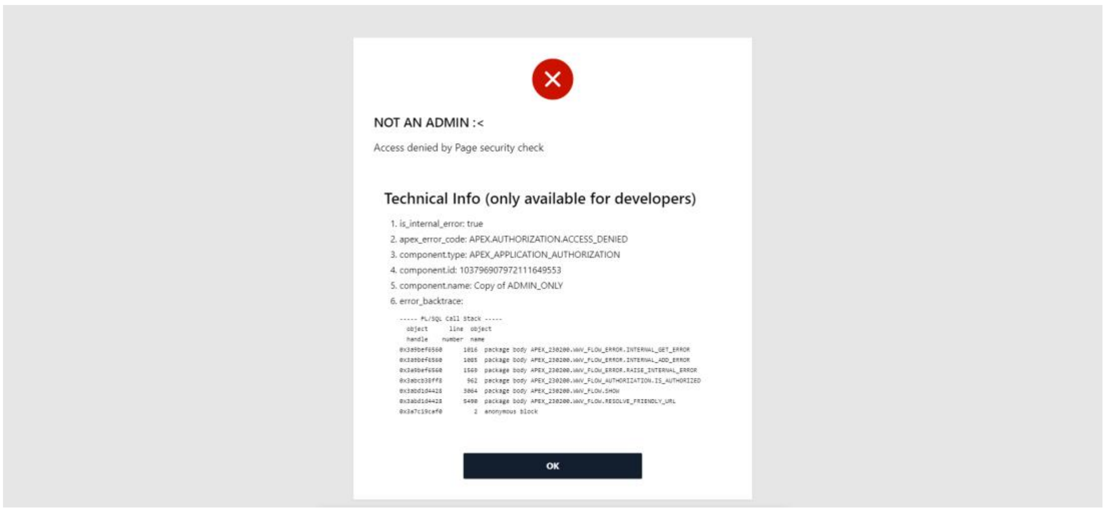
https://apex.oracle.com/pls/apex/r/isys3414_vinhfc_lms/library-management-system140255

Video demonstration URL:

https://rmiteduau-my.sharepoint.com/:v:/g/personal/s3927244_rmit_edu_vn/EdTuOKRZcP9Bqb4HGFgC6V8BGvtmOC-jXJMSU5UUcK_ilg?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJTdHJlYW1XZWJBcHAiLCJyZWZlcnJhbFZpZXciOiJTaGFyZURpYWxvZy1MaW5rIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXcifX0%3D&e=ZZs5CM

List of Application Accounts:

Username	Password	User_type
ha.trinh	pdc	Admin
bach.nguyen	team7	Admin
vinh.truong	HD	Admin
andrew.tran	123	Librarian
chi.dang	123	Librarian
thu.le	123	Librarian
johnathan.crellin	123	Staff
joshua.hansen	123	Staff
anh.lam	123	Student
bao.ho	123	Student
duc.pham	123	Student
evelyn.vo	123	Student
giang.dinh	123	Student
khai.luong	123	Student
vy.kieu	123	Student
iris.nguyen	123	Student


NOT AN ADMIN :<

Access denied by Page security check

Technical Info (only available for developers)

1. is_internal_error: true
2. apex_error_code: APEX.AUTHORIZATION.ACCESS_DENIED
3. component.type: APEX_APPLICATION_AUTHORIZATION
4. component.id: 103796907972111649553
5. component.name: Copy of ADMIN_ONLY
6. error_backtrace:

```
----- PL/SQL Call Stack -----
  object      line  object
  handle    number  name
0x3a9bef6560     1016  package body APEX_230200.WWV_FLOW_ERROR.INTERNAL_GET_ERROR
0x3a9bef6560     1085  package body APEX_230200.WWV_FLOW_ERROR.INTERNAL_ADD_ERROR
0x3a9bef6560     1560  package body APEX_230200.WWV_FLOW_ERROR.RAISE_INTERNAL_ERROR
0x3abcb38ff8      962  package body APEX_230200.WWV_FLOW_AUTHORIZATION.IS_AUTHORIZED
0x3abd1d4428     3864  package body APEX_230200.WWV_FLOW.SHOW
0x3abd1d4428     5490  package body APEX_230200.WWV_FLOW.RESOLVE_FRIENDLY_URL
0x3a7c19caf0        2  anonymous block
```

Figure 4: Page security check for admin authorization

In the security settings for each page/component/item, the authorization scheme can be specified for the views that we want to hide away from the users who have less administrative rights. A pop-up depicted in figure 4 above is presented if a user try to access a page that was not initially tailored to their use cases.

b) Common/General Features

These include basic features that are accessible for all users, regardless of their roles, when accessing our web-based application.

- **Login Interface:** The Login Interface serves as the initial access point to the system, facilitating secure authentication to ensure data confidentiality and integrity. Utilizing Oracle Apex's authentication mechanisms, first-time users must provide valid credentials (including Username and Password) for access. Clicking on 'Remember Username' checkbox allows for time-efficient log-in in the future by autonomous account saving algorithm.

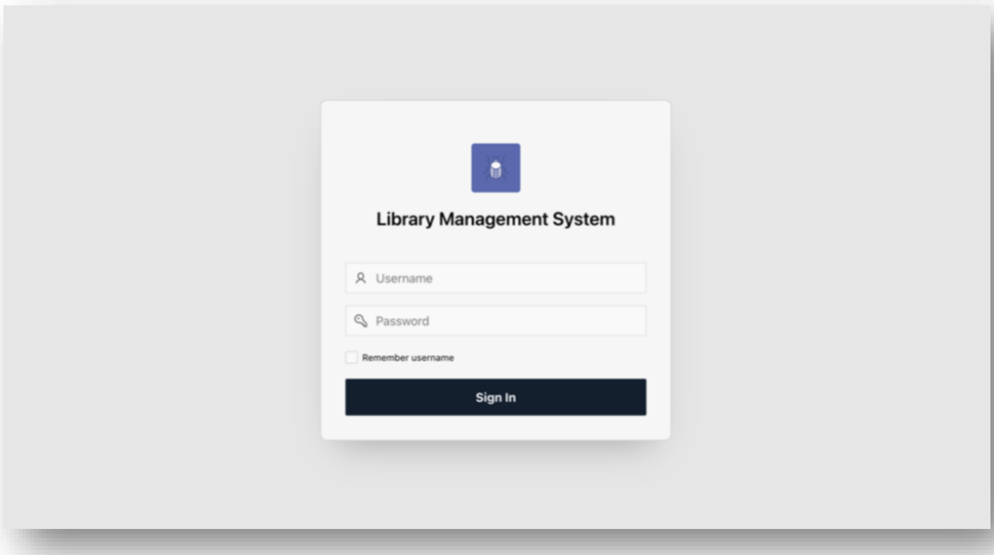


Figure 5 : Log-in Interface

- **Home Page Interface:** Home Page Interface, generally acknowledged as the Central Hub, provides users with all essential information and access to core functionalities. Through a set of clickable icons or buttons, this section may offer shortcuts for users to commonly used features and modules, enabling efficient navigation and task execution.

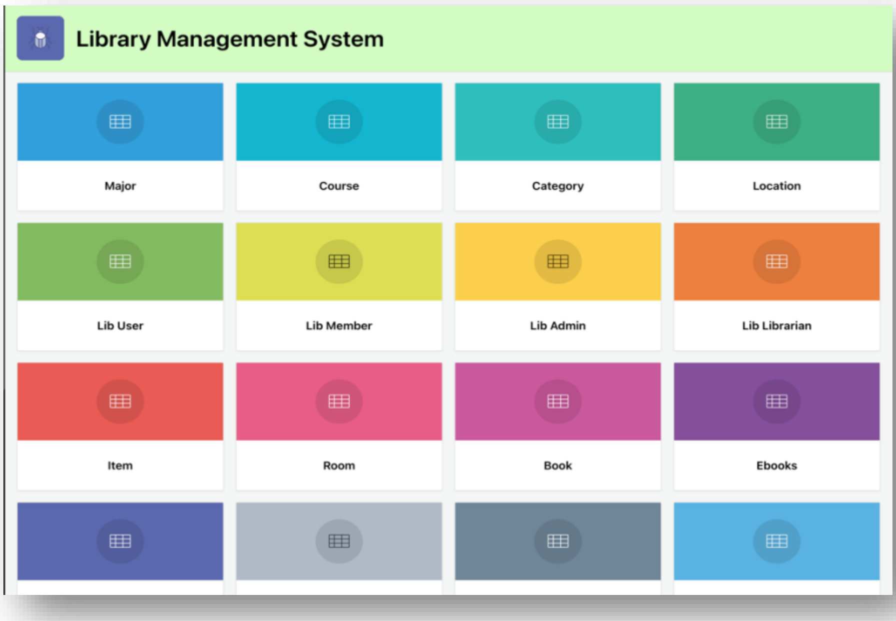


Figure 6 : Homepage Interface

- **Side Navigation Bar:** The Navigation Bar is a menu-driven structure for user, facilitating intuitive navigation and access to various modules within the system. Menu Items are organized and categorized hierachically and logically to enable users to locate desired features quickly, while allowing for nested menus to accommodate diverse range of funcationalities without cluttering the interface.

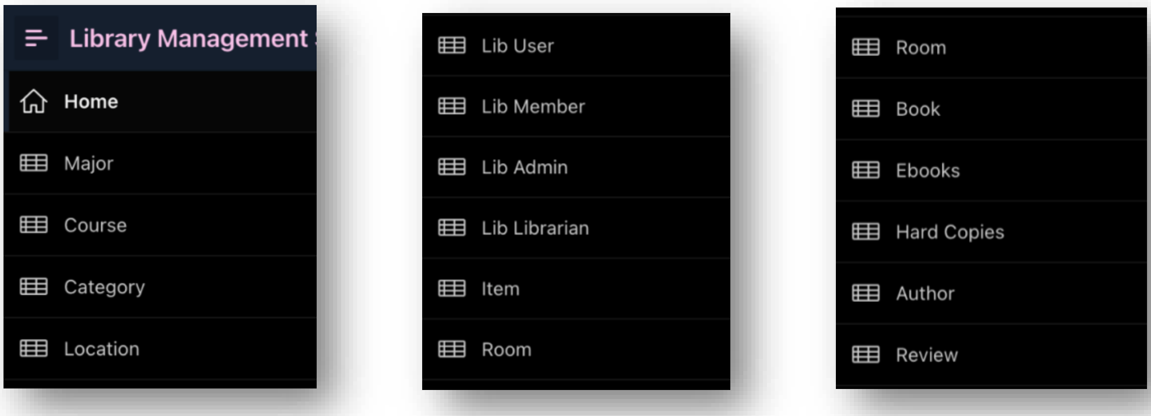


Figure 7: Side Navigation Bar Interface

c) Role-based Features

Leveraging Oracle Apex's role-based access control, our system ensures that users are granted appropriate priviledges based on their assigned roles within the library.

Role	Features
User	• Faceted Search Book • Bookmark Book (E-book) • Borrow Book (Hard Copy) • Room Reservations • Submit Book Review • Submit Request
Librarian	• Manage Book (Ebooks/Hard_Copies) • Manage Author/Category • Manage Fine • Resolve Request • Librarian Report (Statistical reports)
Admin	• Manage Course/Major/Location • View Dashboard

c.1. User's Features

- **Faceted Search Book:** The facted search functionality is enabled in '[Book Cards]' module, allowing users to perform advanced searches for books within the library catalog based on multiple criteria, such as: Author, Book Category, Book Format, Course, Major, School. Users can select one or more value from each facet, and Apex dynamically refines the search results based on the chosen criteria using SQL queries. The results display the cover image of the book, the online book URL, the type of file (PDF/Epub), number of hard copies available, and author names. This was achieved by creating a new View on relevant tables of Book called Book_Info.

User can use this feature by following these steps:

1) Clicking on ‘Book Card’ element within Side Navigation Bar.

2) Choosing one or more values from checkbox lists to narrow down search results.

3) The user interface should reflect the updated search results instantanously.

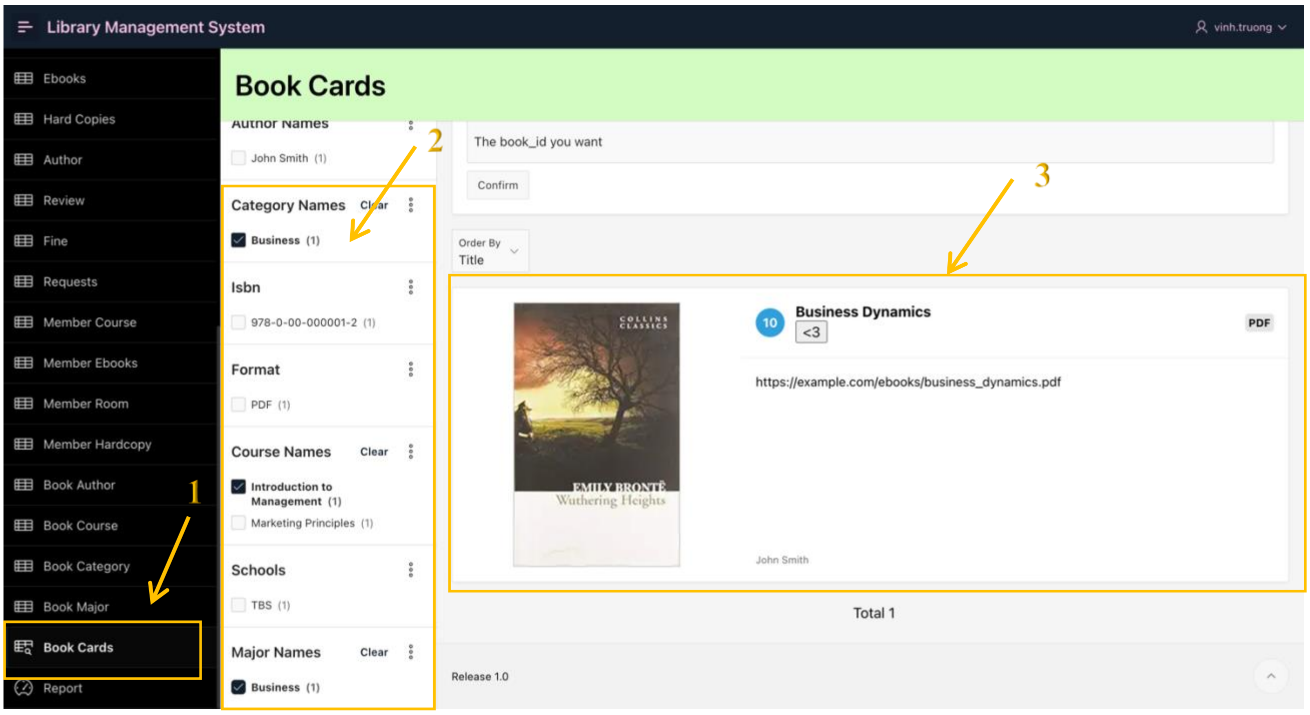


Figure 8: Faceted Search Book Interface

- **Bookmark Book (E-book):** Users can designate specific E-books as bookmarks for future reference, enabling convenient retrieval across multiple sessions and devices. Originally, we intended to make this bookmarking process more intuitive by adding a HTML Heart Button ‘<3’ in each book card results displayed, but due to some technical issues, we cannot obtain the user_id of the current app user, therefore the insert statement could not run. In the end, we resorted to using a form where users can add selected ebook_id and user_id into the Member_Ebooks table, so that they can easily store bookmarks in the database associated with their user accounts.

  The process to bookmark E-books is as follows:

  1) Accessing ‘Book Card’ module

2) Filling in the popped-up form with your user-id and preferred book-id
   a. User ID can be found in the ‘**UserID Appendix**’
   b. Book ID is available in the right corner of the book card (Ex: B001)
3) Clicking ‘Confirm’.

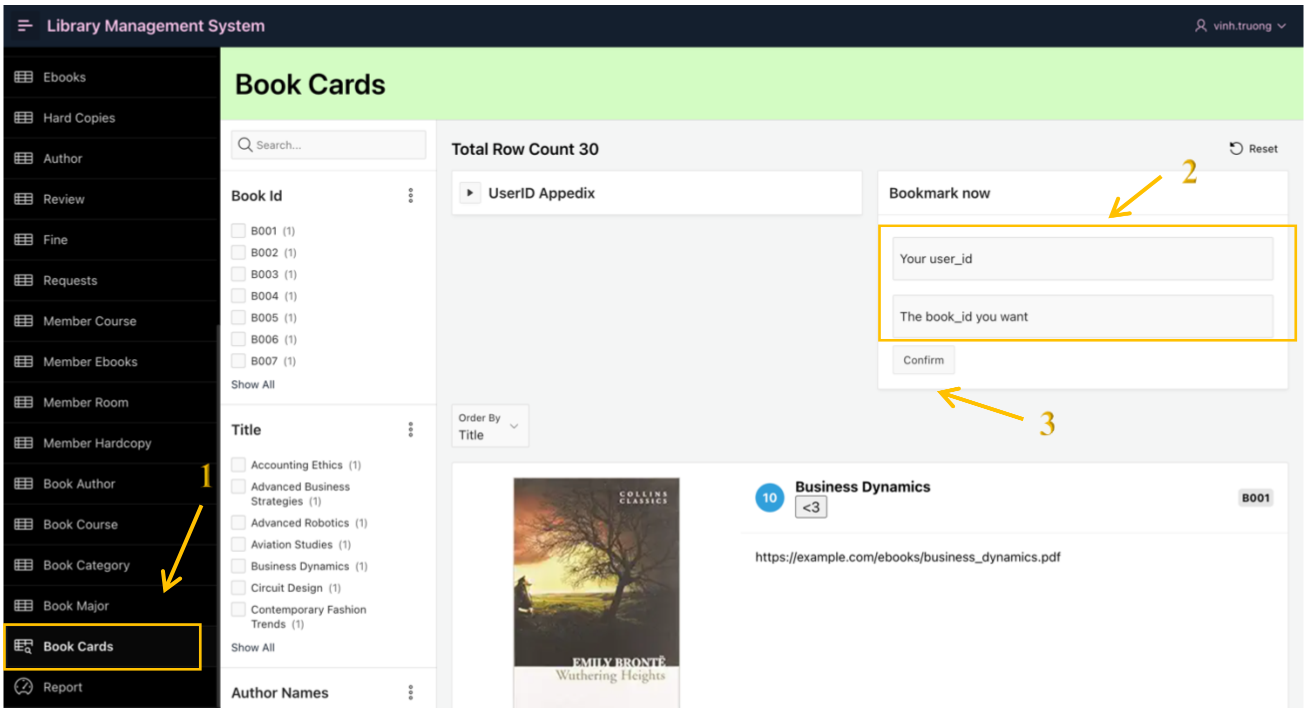
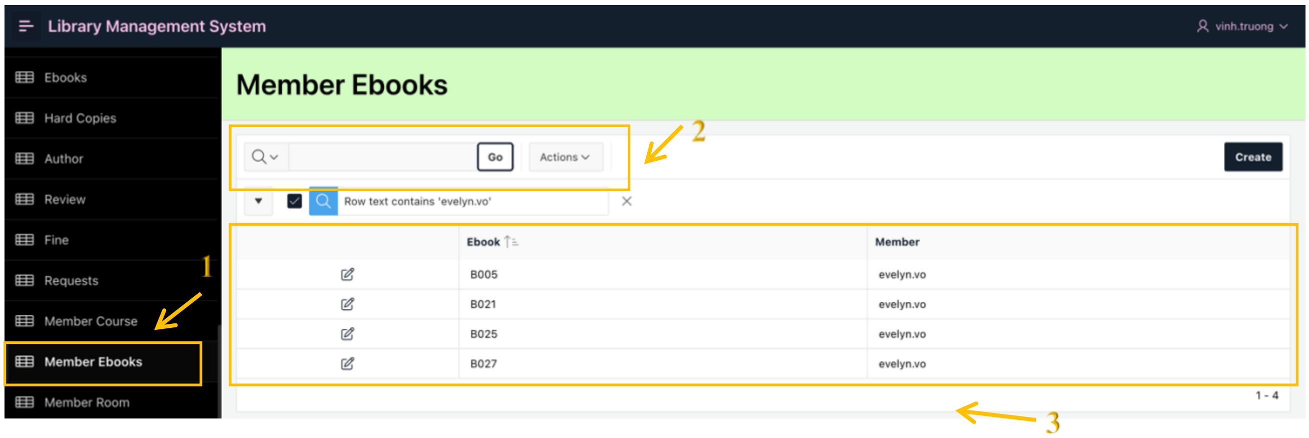


Figure 9: Book Cards Interface

To access ‘Personal Library’ with your collection of bookmarked books:

1) Clicking on ‘Member Ebooks’ Section from Side Navigation Bar
2) Entering your User_ID or Username in the search bar
3) All bookmarked books associated with the user account will appear as-if.

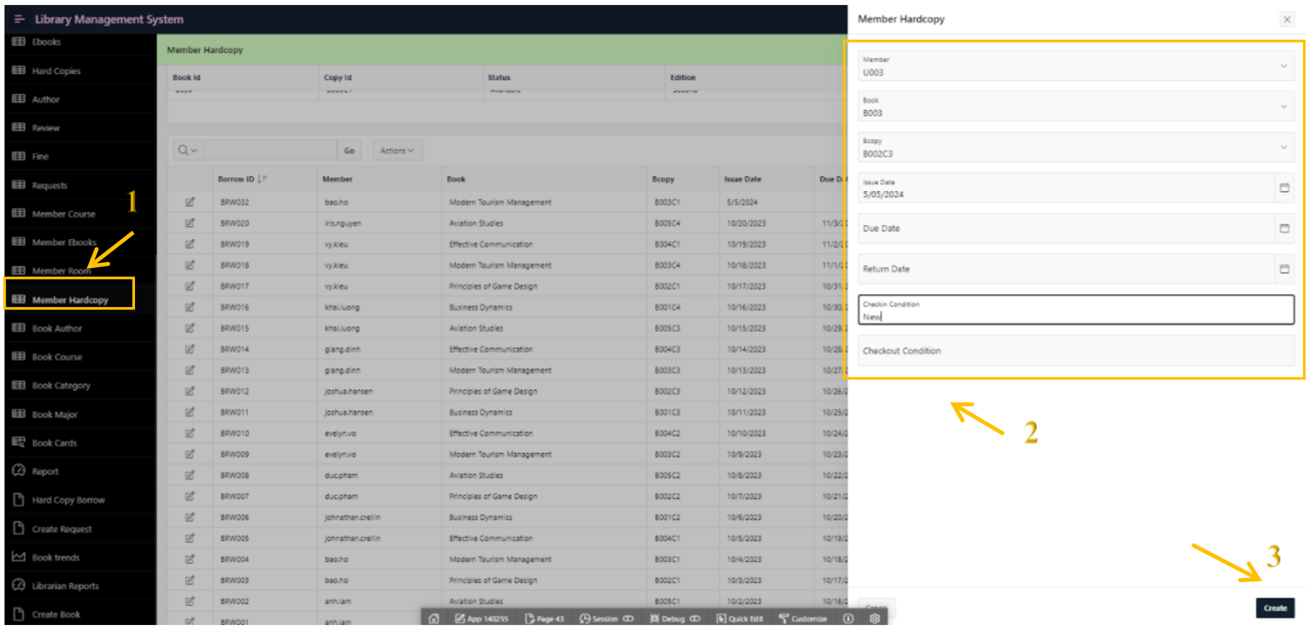


Ebook	Member
B005	evelyn.vo
B021	evelyn.vo
B025	evelyn.vo
B027	evelyn.vo

Figure 10: Personal Library Interface

- **Borrow Book (Hard Copy):** User can borrow physical books from the library collection for a specified duration of up to **2 months**. Through a transactional database model with real-time triggers executed, Apex can integrate with the Member_HardCopy database to check book availability. Notably, different type of users, namely Student and Staff, may be entitled with corresponding book loans limit of 10 and 20 book per month.

  To borrow a Hard Copy, user must:
  1) Navigating to 'Member_HardCopy' module from Side Navigation Bar
  2) Clicking on 'Create' button and filling in Borrowing Form as required
  3) Clicking 'Create' button again to submit borrowing request



Member Hardcopy

Book Id	Copy Id	Status	Edition

Borrow ID	Member	Book	Bcopy	Issue Date	Due D
BRW032	bao.ho	Modern Tourism Management	B003C1	5/5/2024	
BRW020	iris.nguyen	Aviation Studies	B005C4	10/20/2023	11/3/2
BRW019	vy.kieu	Effective Communication	B004C1	10/19/2023	11/2/2
BRW018	vy.kieu	Modern Tourism Management	B003C4	10/18/2023	11/1/2
BRW017	vy.kieu	Principles of Game Design	B002C1	10/17/2023	10/31/
BRW016	khai.luong	Business Dynamics	B001C4	10/16/2023	10/30/
BRW015	khai.luong	Aviation Studies	B005C3	10/15/2023	10/29/
BRW014	giang.dinh	Effective Communication	B004C3	10/14/2023	10/28/
BRW013	giang.dinh	Modern Tourism Management	B003C3	10/13/2023	10/27/
BRW012	joshua.hansen	Principles of Game Design	B002C3	10/12/2023	10/26/2
BRW011	joshua.hansen	Business Dynamics	B001C3	10/11/2023	10/25/2
BRW010	evelyn.vo	Effective Communication	B004C2	10/10/2023	10/24/2
BRW009	evelyn.vo	Modern Tourism Management	B003C2	10/9/2023	10/23/2
BRW008	duc.pham	Aviation Studies	B005C2	10/8/2023	10/22/2
BRW007	duc.pham	Principles of Game Design	B002C2	10/7/2023	10/21/2
BRW006	johnathan.crellin	Business Dynamics	B001C2	10/6/2023	10/20/2
BRW005	johnathan.crellin	Effective Communication	B004C1	10/5/2023	10/19/2
BRW004	bao.ho	Modern Tourism Management	B003C1	10/4/2023	10/18/2
BRW003	bao.ho	Principles of Game Design	B002C1	10/3/2023	10/17/2
BRW002	anh.lam	Aviation Studies	B005C1	10/2/2023	10/16/2
BRW001	anh.lam				

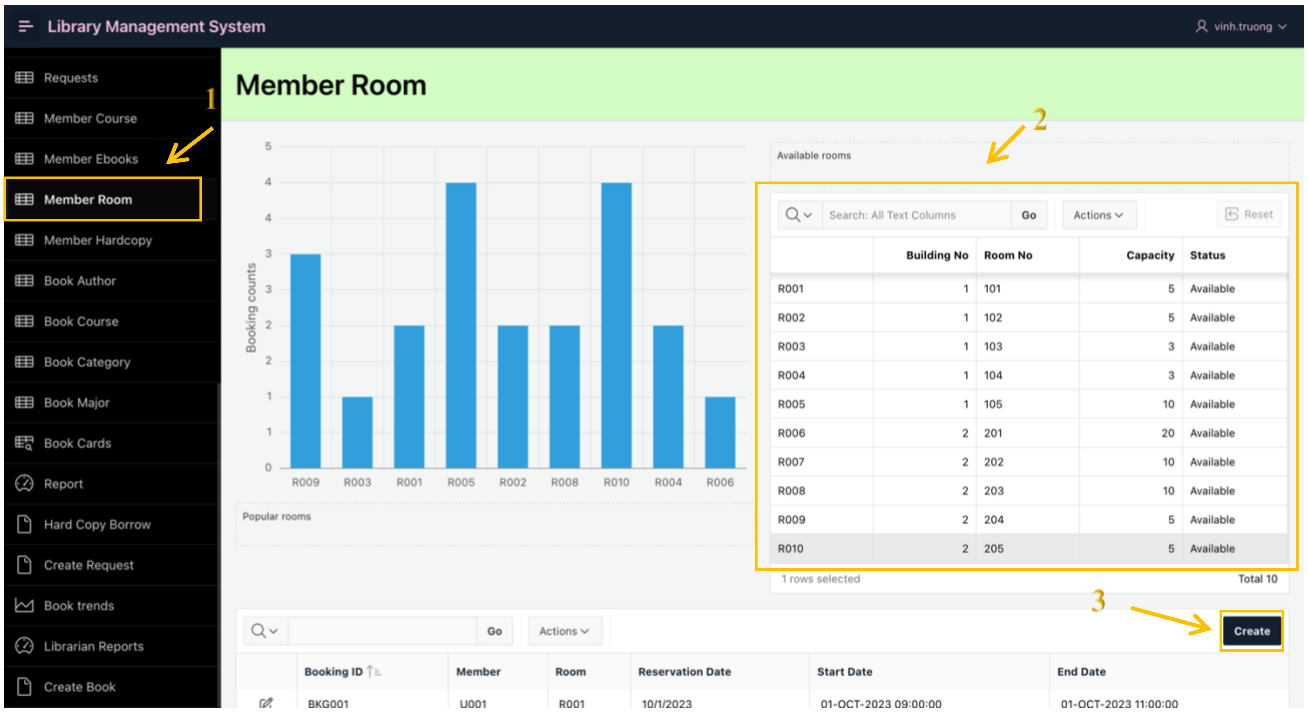
Member Hardcopy

Member	U003
Book	B003
Bcopy	B002C3
Issue Date	5/05/2024
Due Date	
Return Date	
Checkin Condition	New
Checkout Condition	

Figure 11: Borrow Book (Hard Copy) Interface

- **Room Reservations:** Users can reserve library study rooms or meeting spaces for academic or collaborative purposes. Through Oracle Apex's Calender and Reservation Management components, users can view room availability in real-time with the lastest reservation updates. Subsequently, users can utilize an Apex form to submit a room reservation request with details on desired time slots and room. The sýtem may automatically updates to the room booking status in the database if the room is available, and notify users of reservation confirmation or potential conflicts.

  Member can book rooms by following these three steps:
  1) Navigating to 'Member Room' Module in Side Navigation Bar
  2) Checking Room Availability in the table in the right corner
  3) Clicking on 'Create' and filling up information as required on Booking Form



	Building No	Room No	Capacity	Status
R001	1	101	5	Available
R002	1	102	5	Available
R003	1	103	3	Available
R004	1	104	3	Available
R005	1	105	10	Available
R006	2	201	20	Available
R007	2	202	10	Available
R008	2	203	10	Available
R009	2	204	5	Available
R010	2	205	5	Available

1 rows selected Total 10

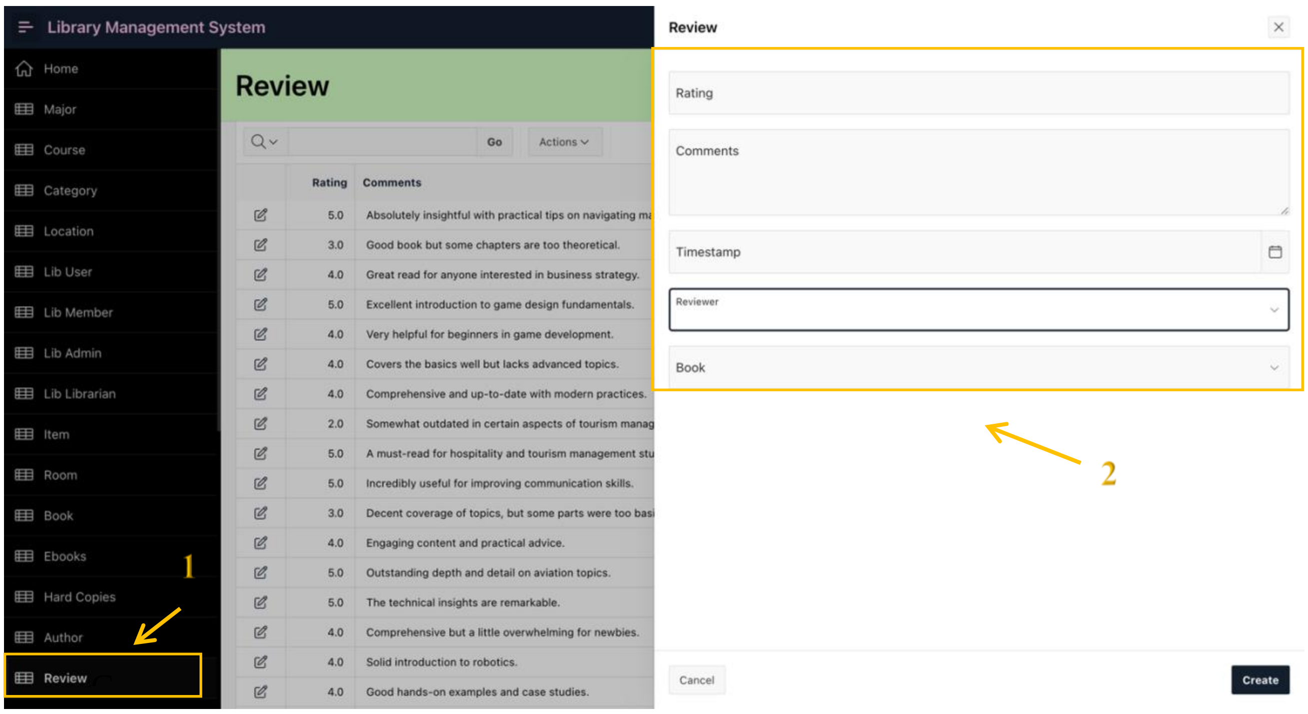
Booking ID	Member	Room	Reservation Date	Start Date	End Date
BKG001	U001	R001	10/1/2023	01-OCT-2023 09:00:00	01-OCT-2023 11:00:00

Figure 12 : Room Reservations Interface

- **Submit Book Review:** User can submit reviews or ratings for Books they have read, providing feedback for other library patrons and enhancing visibility of recommended titles. The Review Form facilitates the capture of review content alongside functionalities for rating the book, bolstering user engagement. Upon submission, these reviews are stored within the 'Review' database, with ratings being displayed alongside relevant book information in 'Book' session, hence enriching user experience through peer insights. Notably, the system is equipped with a trigger mechanism to automatically update a book's rating score as the average of all relevant reviews whenever a new review is submitted.

To submit a Review for any book, User must:

1) Clicking onto 'Review' module from Side Navigation Bar
2) Entering 'Create' button and finishing the Review Form as required



Rating	Comments
5.0	Absolutely insightful with practical tips on navigating ma
3.0	Good book but some chapters are too theoretical.
4.0	Great read for anyone interested in business strategy.
5.0	Excellent introduction to game design fundamentals.
4.0	Very helpful for beginners in game development.
4.0	Covers the basics well but lacks advanced topics.
4.0	Comprehensive and up-to-date with modern practices.
2.0	Somewhat outdated in certain aspects of tourism manag
5.0	A must-read for hospitality and tourism management stu
5.0	Incredibly useful for improving communication skills.
3.0	Decent coverage of topics, but some parts were too basi
4.0	Engaging content and practical advice.
5.0	Outstanding depth and detail on aviation topics.
5.0	The technical insights are remarkable.
4.0	Comprehensive but a little overwhelming for newbies.
4.0	Solid introduction to robotics.
4.0	Good hands-on examples and case studies.

Figure 13 : Submit Book Review Interface

- **Submit Request:** Lastly, users can sumit requests for new book acquisition, reporting damaged books, seeking technical assistance, or any other library services. Thanks to Apex's form-based input and transactional database system, the submit request function allows users to submit detailed information about their requests, which are then routed to appropriate personnel (library staff) for review and processing, with status updated to users through the system
  User can access the Review Form by following these steps:
  1) Choosing 'Create Request' from Side Navigation Bar
  2) Filling-in the Request Form as required
  3) Clicking 'Create' to submit

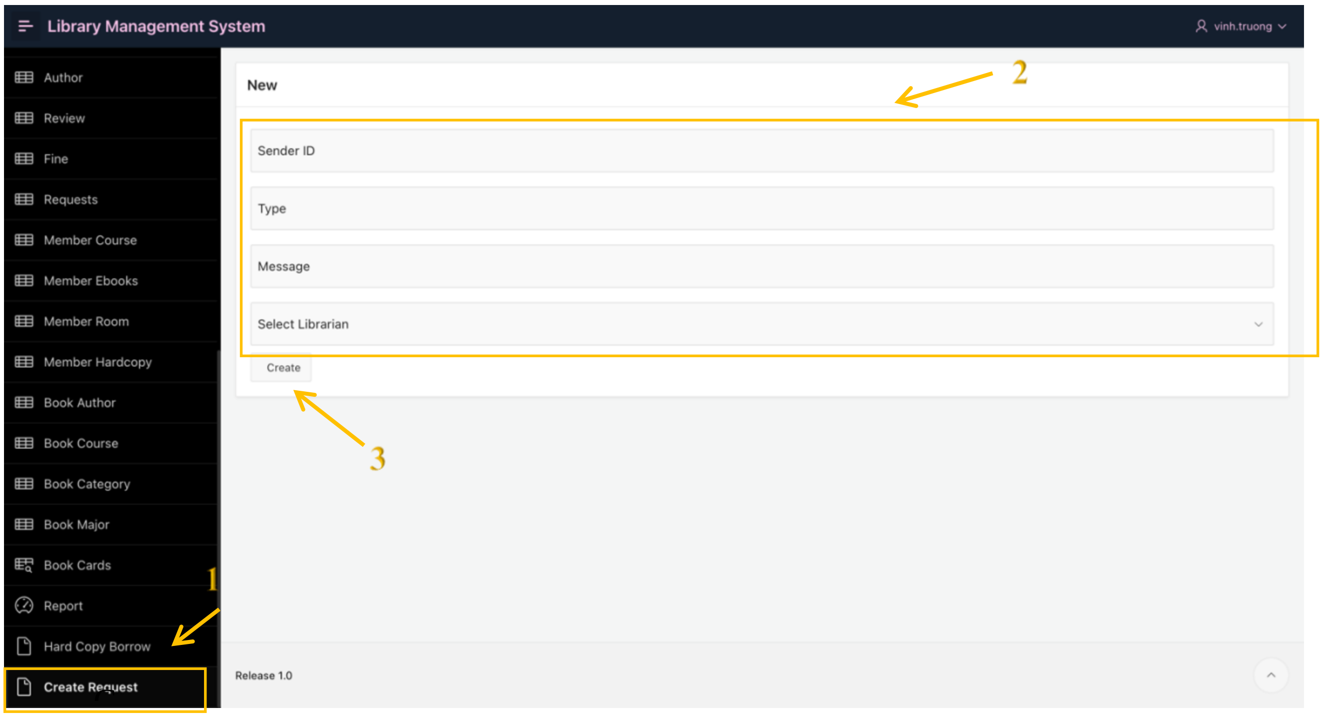


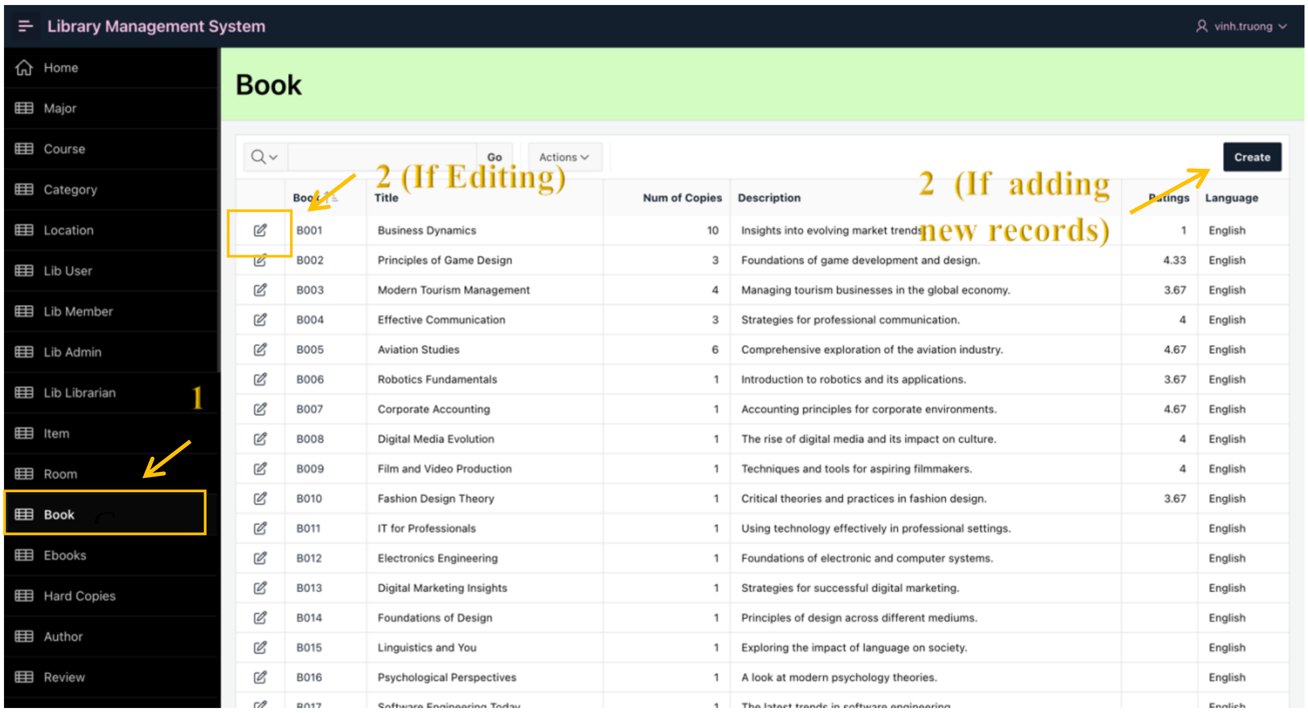
Figure 14: Submit Request Interface

c.2. Librarian's Features

- **Manage Book (Ebooks/Hard_Copies):** Librarians are authorized to add, edit, or remove library's book collection, including both e-books and hard copies. With Oracle Apex's data manipulation capabilities, this manage_book function enables librarians to maintain comprehensive book records, update status to reflect real-time availability, and circulation history, with transactional integrity ensured through database constraints.

  To update/edit/delete information on Book records,

  1) Clicking onto 'Book' module (Note: Changes in Course/Major/Category of Book requires access to Book Course/Book Major/Book Category module)
  2) Clicking onto 'Create' button if adding new records or clicking button if editing

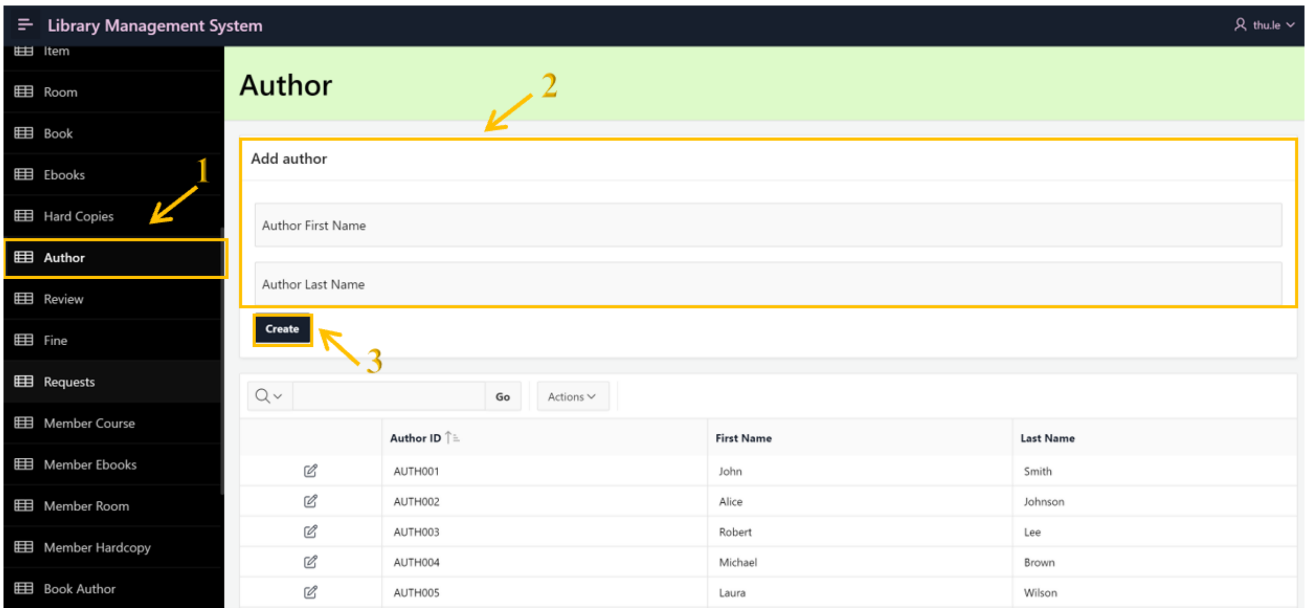


	Book	Title	Num of Copies	Description	Ratings	Language
	B001	Business Dynamics	10	Insights into evolving market trends	1	English
	B002	Principles of Game Design	3	Foundations of game development and design.	4.33	English
	B003	Modern Tourism Management	4	Managing tourism businesses in the global economy.	3.67	English
	B004	Effective Communication	3	Strategies for professional communication.	4	English
	B005	Aviation Studies	6	Comprehensive exploration of the aviation industry.	4.67	English
	B006	Robotics Fundamentals	1	Introduction to robotics and its applications.	3.67	English
	B007	Corporate Accounting	1	Accounting principles for corporate environments.	4.67	English
	B008	Digital Media Evolution	1	The rise of digital media and its impact on culture.	4	English
	B009	Film and Video Production	1	Techniques and tools for aspiring filmmakers.	4	English
	B010	Fashion Design Theory	1	Critical theories and practices in fashion design.	3.67	English
	B011	IT for Professionals	1	Using technology effectively in professional settings.		English
	B012	Electronics Engineering	1	Foundations of electronic and computer systems.		English
	B013	Digital Marketing Insights	1	Strategies for successful digital marketing.		English
	B014	Foundations of Design	1	Principles of design across different mediums.		English
	B015	Linguistics and You	1	Exploring the impact of language on society.		English
	B016	Psychological Perspectives	1	A look at modern psychology theories.		English
	B017	Software Engineering Today	1	The latest trends in software engineering		English

Figure 15: CRUD Operations for Book Interface

- **Manage Author/Category:** This administrative feature allows for adding and removing authors/ category, potentially useful for keeping author data current and organized.

  To update Author/ Category:
  1) Clicking onto 'Authors/ Category' module from Side Navigation Bar
  2) Click the "Create" button to enter
  3) Finalize the edits by clicking "Create" again.



	Author ID	First Name	Last Name
	AUTH001	John	Smith
	AUTH002	Alice	Johnson
	AUTH003	Robert	Lee
	AUTH004	Michael	Brown
	AUTH005	Laura	Wilson

Figure 16: Book Author Updating Interface

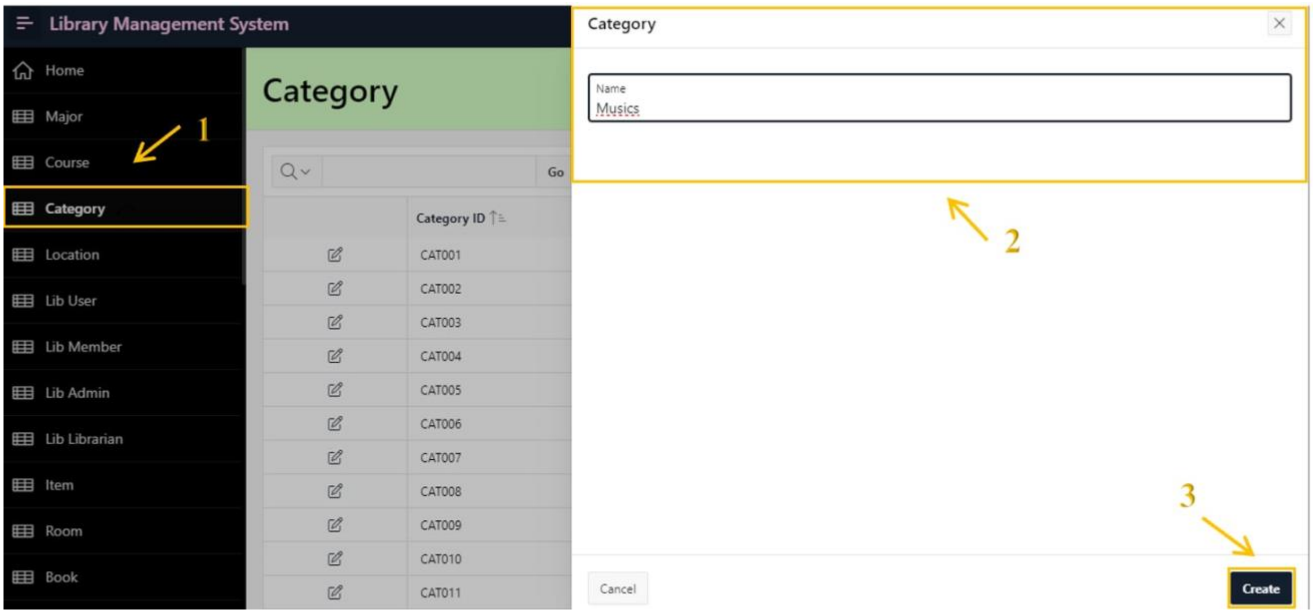
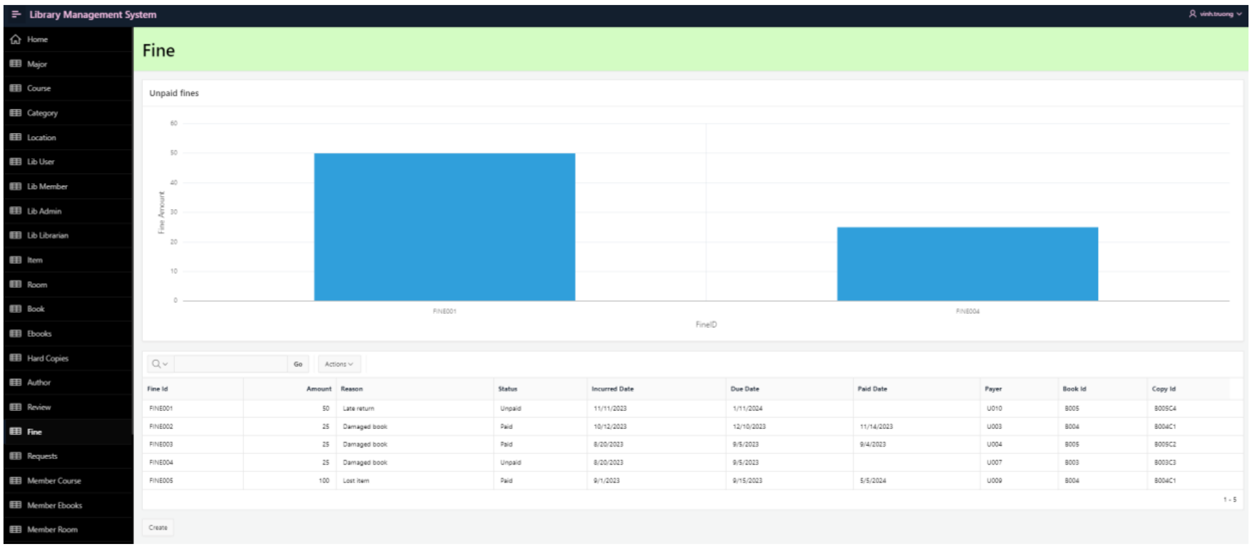


Figure 17: Book Category Updating Interface

- **Manage Fine:** Librarian can manage fines and penalties associated with overdue book returns or other infractions. Apex can integrate with the user borrowing data to calculate potential fines based on pre-defined rules. The system provides functionalities for librarian to view outstanding fines, sending overdue notifications to users, and processing fine payments with updating fine status.

  To manage Fine: Clicking onto ‘Fine’ module from Side Navigation Bar



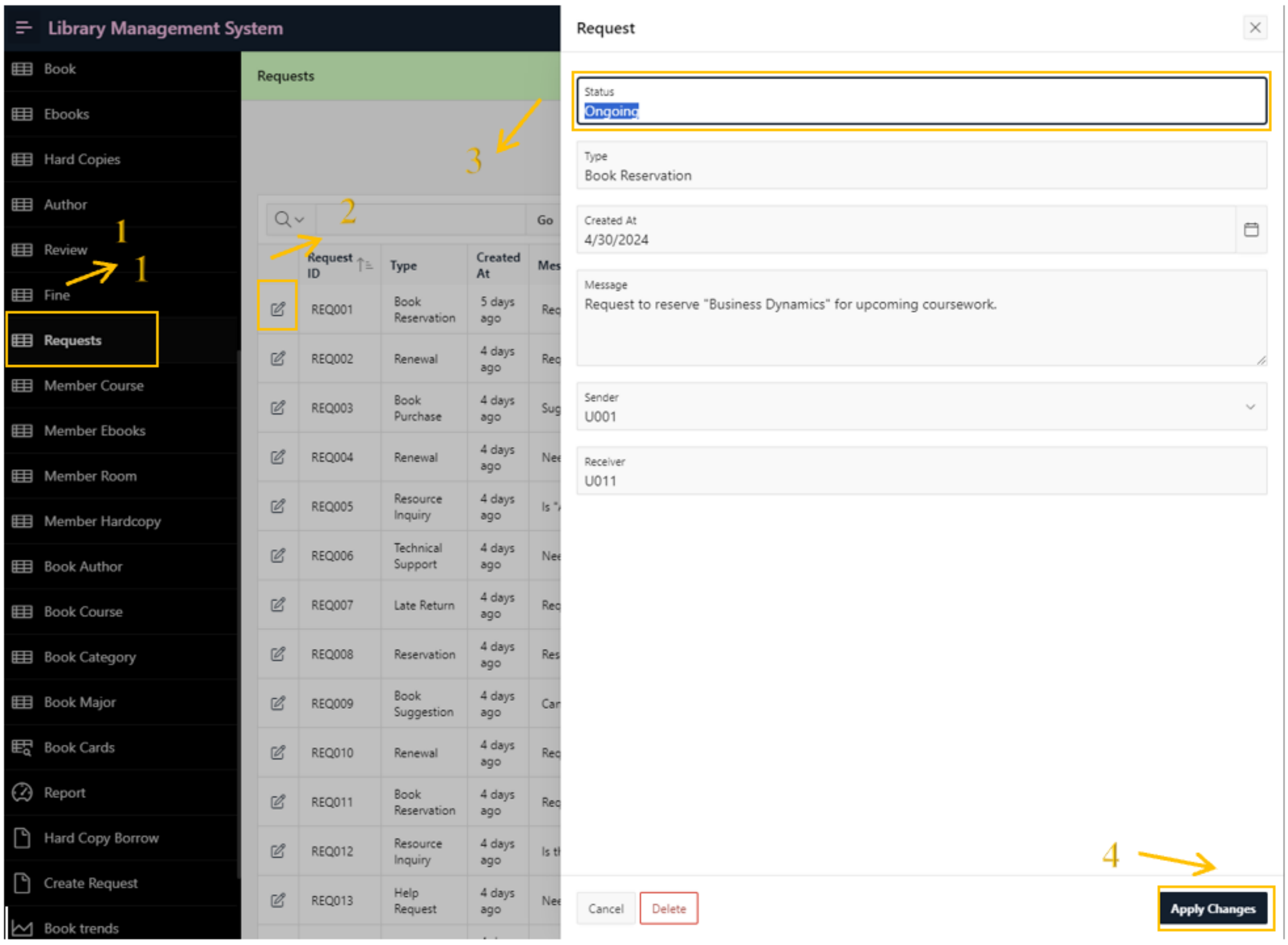
Fine Id	Amount	Reason	Status	Incurred Date	Due Date	Paid Date	Payer	Book Id	Copy Id
FINE001	50	Late return	Unpaid	11/11/2023	1/11/2024		U010	B005	B005C4
FINE002	25	Damaged book	Paid	10/12/2023	12/10/2023	11/14/2023	U003	B004	B004C1
FINE003	25	Damaged book	Paid	8/20/2023	9/5/2023	9/4/2023	U004	B005	B005C2
FINE004	25	Damaged book	Unpaid	8/20/2023	9/5/2023		U007	B003	B003C3
FINE005	100	Lost item	Paid	9/1/2023	9/15/2023	5/5/2024	U009	B004	B004C1

Figure 18: Manage Fine Interface

- **Resolve Request:** users can sumit requests for new book acquisition, reporting damaged books, seeking technical assistance, or any other library services. Thanks to Apex’s form-based input and transactional database system, the submit request function allows users to submit detailed information about their requests, which are then routed to appropriate personnel (library staff) for review and processing, with status updated to users through the system

User can access the Review Form by following these steps:

1) Choosing ‘Request’ from Side Navigation Bar
2) Click on the “Edit Icon”
3) Resolve the Request Status as required
4) Click on ‘Apply Changes’ to submit



Requests

Request ID	Type	Created At	Mes
REQ001	Book Reservation	5 days ago	Req
REQ002	Renewal	4 days ago	Req
REQ003	Book Purchase	4 days ago	Sug
REQ004	Renewal	4 days ago	Nee
REQ005	Resource Inquiry	4 days ago	Is "/
REQ006	Technical Support	4 days ago	Nee
REQ007	Late Return	4 days ago	Req
REQ008	Reservation	4 days ago	Res
REQ009	Book Suggestion	4 days ago	Car
REQ010	Renewal	4 days ago	Req
REQ011	Book Reservation	4 days ago	Req
REQ012	Resource Inquiry	4 days ago	Is tl
REQ013	Help Request	4 days ago	Nee

Status: Ongoing
Type: Book Reservation
Created At: 4/30/2024
Message: Request to reserve "Business Dynamics" for upcoming coursework.
Sender: U001
Receiver: U011

Figure 19: Resolve Request Interface

- **View Librarian Reports (statistical reports)**

To view Librarian Reports: Click on Librarian Reports from Side Navigation Bar to view the report.

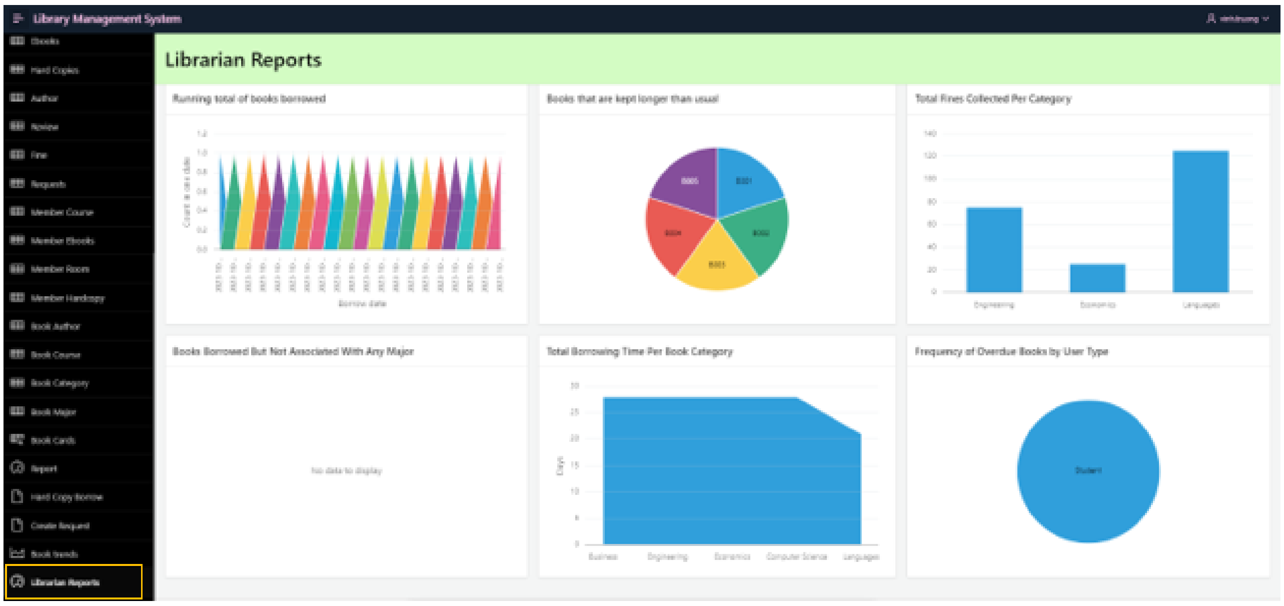


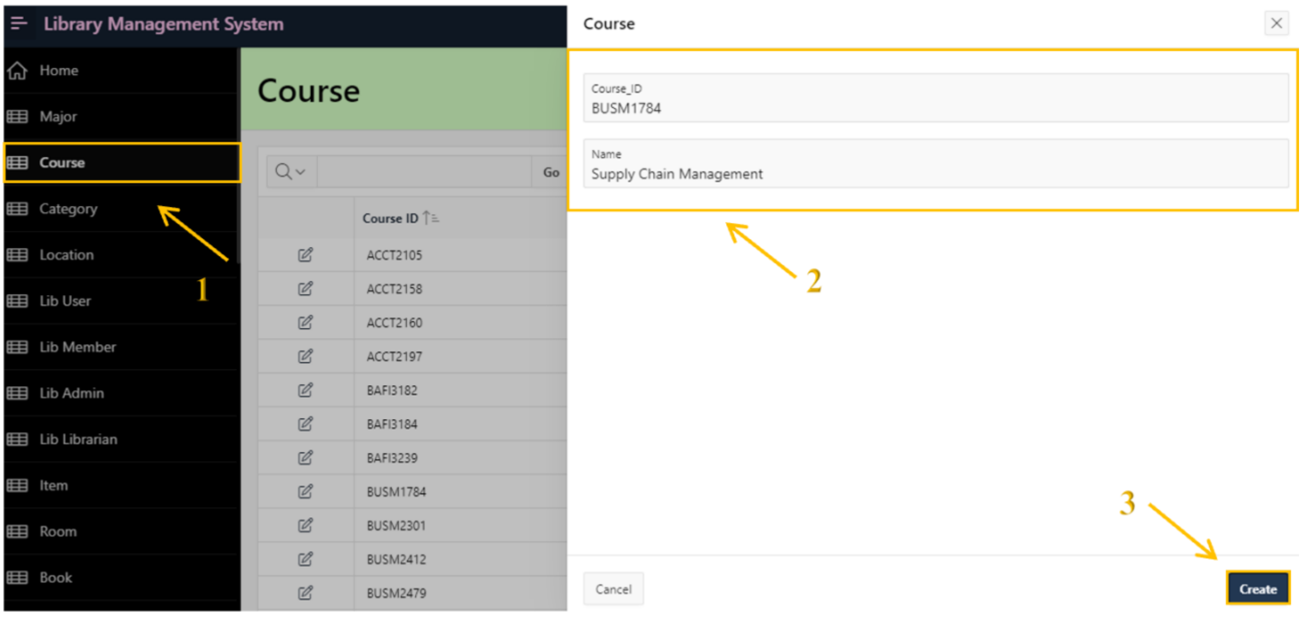
Figure 20: View Dashboard (statistical reports)

c.3. Admin's Features

- **Manage Course/ Major/ Location:** This administrative feature allows for adding and removing Course/ Major/ Location, potentially useful for keeping author data current and organized.

To update Course/ Major/ Location:

1) Clicking onto 'Course/ Major/ Location' module from Side Navigation Bar
2) Click the "Create" button to enter
3) Finalize the edits by clicking "Create" again.



Course

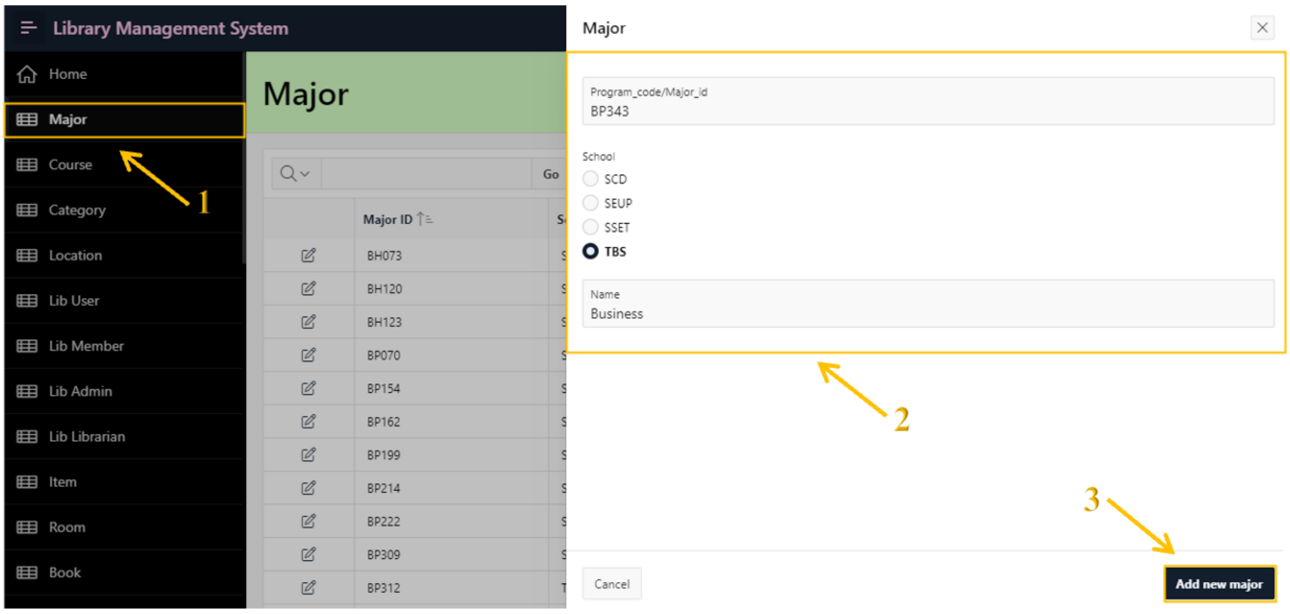
Course ID
ACCT2105
ACCT2158
ACCT2160
ACCT2197
BAFI3182
BAFI3184
BAFI3239
BUSM1784
BUSM2301
BUSM2412
BUSM2479

Course

Course_ID: BUSM1784

Name: Supply Chain Management

Figure 21: Manage Course Interface



Major

Major ID
BH073
BH120
BH123
BP070
BP154
BP162
BP199
BP214
BP222
BP309
BP312

Major

Program_code/Major_id: BP343

School: SCD SEUP SSET [✓ TBS]

Name: Business

Figure 22: Manage Major Interface

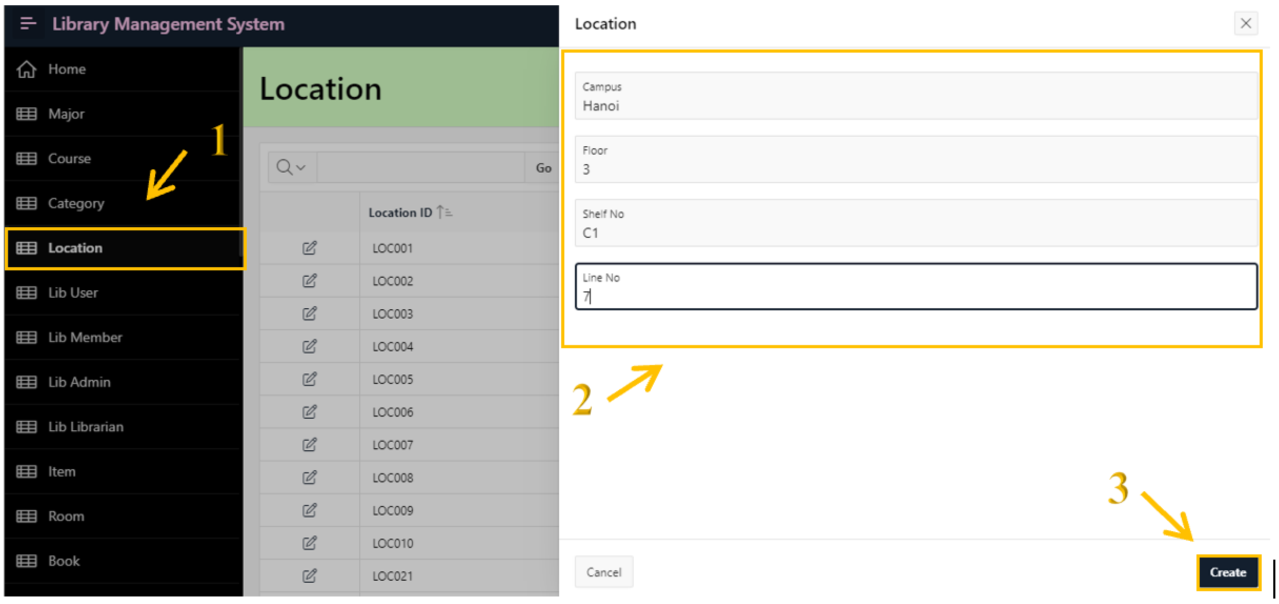


Figure 23: Manage Location Interface

- **View Dashboard**
  To view Admin Dashboard: Click on Reports from Side Navigation Bar to view the dashboard.

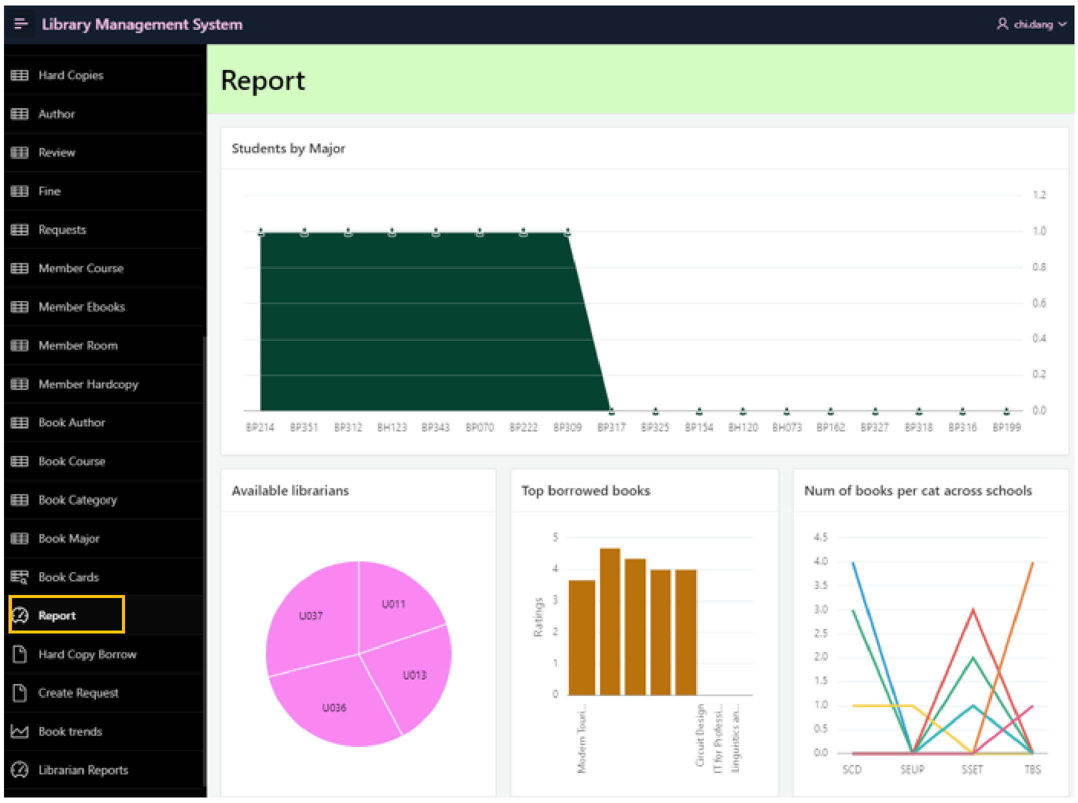


Figure 24: View Dashboard Interface

d) Extra Features

- Each member type (Staff and Student) has a different limit in the number of books they can borrow in a timeframe.
- Auto-trigger for Real-time Records:

Triggers	Scenario	Function
Book-borrowing Trigger	A Librarian adds a new record into ‘Member_HardCopy’ table	• Storing calculated due date by adding two months to the current date, ensuring timely return of the borrowed book • Update the status of borrowed book copy to ‘Unavailable’ to prevent further borrowing. • Incrementing the total number of books curently borrowed by member, facilitating accurate tracking of borrowing activities.
Book-returning Trigger	A member returns their book to the library	• Setting the return date to the curent date while updating the checkout condition of the returned book to ‘New’ or ‘Damaged’. • Updating the status of returned book copy to ‘Available’, indicating its availability to other users. • Decrementing the total number of books borrowed by the users. • Inserting a fine record with details on penalty reasons if the check-in condition differs from the checkout condition.
Fine-paying Trigger	A member pays their fines	• Setting the fine status to ‘Paid’ and setting paid_date to current date. • Incrementing total fine paid amount attribute of the member who paid fine in the database.
Room-booking Trigger	A member books a room	• Updating the corresponding room’s status to ‘Unavailable’. • Checking for conflicts in booking timeslots signalling an error message if detected. • Periodically updating the status of rooms based on ongoing bookings, ensuring real-time availability status.

Book-Reviewing Trigger	A member submits a review for a book	• Recalculating the average rating for the reviewed book based on all submitted reviews. • Updating the Book table accordingly
Book_Copy_Adding Trigger	A Librarian adds a hard copy record of any book.	• Updating the number of copies attribute in the coresponding book record

CONCLUSION

1. Project Objectives & Main design and Implementation Outcomes

Objectives	Functions	Outcomes
REDUCE PAPER WORK AND STAFFING	• Room Reservations • Submit Request • Manage Book (Ebooks/Hard_Copies) • Manage Fine • Resolve Request • Manage User	• The implementation of the LMS effectively reduces paperwork and staffing needs by automating essential processes. • Room Reservations are streamlined, allowing for efficient scheduling without manual oversight. • User requests are managed effectively through the Submit Request function, which reduces response times and administrative workload. • Book management, both for eBooks and hard copies, is automated, ensuring accurate and up-to-date inventory with minimal manual intervention. Fine management is simplified through automated calculations and tracking, enhancing financial accuracy and reducing administrative burden. • Resolving user requests quickly boosts satisfaction and operational efficiency, while centralized user management diminishes the need

		for extensive manual administration and increases system security.
ENHANCE DAILY TRANSACTIONS AND RECORD MANAGEMENT	• Borrow Book (Hard Copy) • Room Reservations • Submit Request • Manage Book (Ebooks/Hard_Copies) • Manage Author/Category • Manage Fine • Resolve Request • View Dashboard (Statistical reports) • Manage Course/Major/Location • Manage User	• The LMS enhances the efficiency of daily transactions and record management through a suite of integrated functions. • The system facilitates the borrowing of hard copies with efficient check-out and return processes, increasing user satisfaction by minimizing wait times. • Room reservation capabilities provide real-time updates of available spaces, preventing double bookings and optimizing resource use. • The ability to manage books, authors, and categories ensures that library resources are accurately categorized and easily retrievable. • Fine management functions streamline the financial aspects of library operations, ensuring accurate and timely accounting. • The system also enables swift resolution of user requests and the management of user profiles, which tailor services to meet specific user needs. • Additionally, the dashboard offers statistical reports that provide insights for strategic decision-making, improving library management effectiveness.
	• Faceted Search Book • Bookmark Book (E-book) • Submit Book Review	• The LMS is designed to support a robust academic library environment by enabling sophisticated search capabilities, personalization, and community engagement. • The Faceted Search Book function allows users to efficiently locate

PROVIDE A SUPPORTIVE ACADEMIC LIBRARY		academic materials based on specific criteria such as major, course, or school, enhancing research productivity. • Users can bookmark ebooks for quick access in future sessions, which facilitates continuous learning and personal study efficiency. • Moreover, the system's ability to accept and display book reviews helps create an engaged and informed user community, aiding students in making well-informed decisions about their reading and research choices, thereby enriching the overall academic experience.

2. Alignment of Design and Implementation

The development of our Online Library Management System adhered closely to the initally-planned Entity Relationship Diagram (ERD). Most tables were generated, linked, and populated with data attributes according to the ERD, ensuring a well-aligned approach between design and implementation.

However, unforeseen technical challenges arose during the implementation phase, specifically related to user role authentication. This, in turn, limited the complete realization of user interface (UI) as initally envisioned. The inability to fully granularize user permissions based on assigned roles restricted the UI's ability to dynamically adapt to different type of users (Ex: Librarians, Members, Admins). This represented some undesirable discourses from the inital design, where UI would ideally present functionalities specific to each user role.

In conclusion, while the design and implementation strive for perfect alignment, the reality of technical constraints neccessitates ônging adaptation and refinement to ensure system functionality.

3. Limitations and Challenges

a. Limited Experience with Library Processes and Oracle Apex

Reflecting on our team's journey, it's evident that our limited familiarity with Oracle Apex and library operational processes poses challenges. We encountered many difficulties in bringing the database from the back end to the front end and many bugs along the way. This gap sometimes slows our progress, moves the team's completion deadline multiple times, and leads to uncertainties in implementing functionalities that align closely with our expectations of an initial website4. Future Recommendations and Plans.

b. Technical Issue of User Role’s Authorization

The first problem directly affects this second limitation, which is that the website can still not be customized for each user role. Instead of limiting which accounts can be accessed and which functions can be viewed, all users can access all functionalities.

c. The System Currently Lacks Real Data and Operates with a Limited Dataset

This limitation is related to the system's current data set, which may not reflect the complexity and scale of real-world operations. A limited data set can impact system testing and development because it may not provide enough scenarios or data volumes to thoroughly test performance and functionality.

4. Recommendations and future plans

Step 1	First, to gain a deeper understanding of LMS and Oracle Apex, we plan to invest more time researching successful library web builds and participating in a detailed training program covering the functions of Oracle Apex tutorials and common bugs. By examining case studies and research on sample library systems, we aim to enhance technical skills and practical knowledge, ensuring our development efforts are both well-informed and effective.
Step 2	We believe that after having more knowledge about Oracle Apex, we will also have more perspectives on solving the "User Role's Authorization" issue. But if we still cannot find a solution, our plan is to ask for advice and guidance from people with experience in this field, such as our lecturer, from whom we can solve the above problem.
Step 3	To overcome the limitations of operating with limited data sets, it is important to integrate a more comprehensive variety of real-world data into the system. For example, partnering with libraries to gain data from their daily transactional and operational data will provide richer data sets that reflect real-life usage patterns and challenges. This expanded data set will enable more thorough testing and tuning of the system, ensuring it is robust and capable of handling diverse real-world scenarios effectively. Additionally, incorporating synthetic data generation techniques can complement real data, allowing more comprehensive testing and evaluating scalability under different simulation conditions.

REFERENCES

Aiden G, Jack D and Peter R (2019) *The Waterfall Model: Advantages, disadvantages, and when you should use it*, IBM website, accessed 1 May 2024. https://developer.ibm.com/articles/waterfall-model-advantages-disadvantages/

Albarak M (2020) ‘Managing Technical Debt in Database Normalization’, *IEEE Xplore*, 48 (3): 755 – 772, doi: 10.1109/TSE.2020.3001339.

Dan R (2023) *Agile vs. waterfall project management*, ATLASSIAN website, accessed 1 May 2024. https://www.atlassian.com/agile/project-management/project-management-intro

Oracle Apex (2023) *Build enterprise apps 20x faster with 100x less code*, Oracle Apex website, accessed 1 May 2024. https://apex.oracle.com/en/

Rahul D (2020) “NEW NATIONAL EDUCATION POLICY & THE ROLE OF LIBRARIAN”, *Department Of Library & Information Science*. https://upa.org.in/adminserver/uploadedImage/reserch/54261.pdf

Sorbello P (2022) *Towards an Algorithm for the Third Normal Form (3NF) Using Relational Algebra*, Oregon State University, accessed 1 May 2024. https://ir.library.oregonstate.edu/concern/honors_college_theses/1v53k5158

Sun Z (2023) ‘Learned Index: A Comprehensive Experimental Evaluation’, *ACM*, 16 (8): 1992 – 2004, doi: 10.14778/3594512.3594528.