420-PZ1-GX Introduction to Programming class 00421

Final Project

Sakkaravarthi Ramanathan

Objective:

The aim is to design and implement The Library Management System that caters to librarians, publishers, and users, providing features for adding, editing, and deleting information related to books, librarians, publishers, and users. The primary goal is to streamline book transactions, user management, and overall library administration.

System Requirements:

User Roles:

1. Administrator (Admin):
    - Can add, edit, and delete librarians, publishers, and users.
    - Manages overall system configuration.
2. Librarian:
    - Can add, edit, and delete books.
    - Handles book transactions (check-in, check-out).
    - Manages user information.
3. User:
    - Can search for books, check out books, and view transaction history.
    - Receives notifications for overdue books.

Librarian Management:

4. Add Librarian:
    - Username, password, name, contact information.
5. Edit Librarian:
    - Modify contact information or password.
6. Delete Librarian:
    - Remove a librarian from the system.

Publisher Management:

7. Add Publisher:
    - Publisher name, address, contact details.
8. Edit Publisher:
    - Modify publisher details.
9. Delete Publisher:
    - Remove a publisher from the system.

Book Management:

1. Add Book:
    - Title, author, genre, ISBN, quantity, publication year, etc.
2. Edit Book:
    - Modify book details, update quantity, change genre, etc.
3. Delete Book:
    - Remove a book from the library inventory.

Transaction Management:

1. Check Out Book:
    - Record user borrowing details (user, book, due date, status).
2. Check In Book:
    - Mark a book as returned, update status, check for overdue fines.
3. View Transaction History:
    - See a history of book transactions, including checkouts and returns.

User Management:

4. Add User:
    - Username, password, name, contact information.
5. Edit User:
    - Modify user details.
6. Delete User:
    - Remove a user from the system.

Search and Sorting:

7. Search Books:
    - Search based on title, author, genre, or ISBN.
8. Sort Books:
    - Sort books by title, author, or publication year.

<u>Classes and Objects:</u>

9. Librarian Class:
    - Attributes: username, password, name, contact information.
    - Methods: add_librarian(), edit_librarian(), delete_librarian().
10. Publisher Class:
    - Attributes: publisher name, address, contact details.
    - Methods: add_publisher(), edit_publisher(), delete_publisher().
11. Book Class:
    - Attributes: title, author, genre, ISBN, quantity, publication year, etc.
    - Methods: add_book(), edit_book(), delete_book().
12. User Class:

- Attributes: username, password, name, contact information.
- Methods: add_user(), edit_user(), delete_user().

13. Transaction Class:
    - Attributes: user, book, due date, status.
    - Methods: check_out_book(), check_in_book(), view_transaction_history().

Data Structures:

14. Lists and Tuples:
    - Lists can be used to store multiple books, users, librarians, etc.
    - Tuples may represent fixed details like ISBN or a specific transaction record.
15. Dictionaries:
    - Used to store key-value pairs.
    - Example: A dictionary to store book details where ISBN is the key.
16. Collections Module:
    - Utilize the collections module for more advanced data structures like defaultdict or namedtuple if needed.

OOP Concepts:

17. Encapsulation:
    - Class methods encapsulate related functionality.
    - Each class represents a specific entity (Librarian, Publisher, Book, User).
18. Inheritance:
    - Potential use of inheritance for common attributes/methods among classes (e.g., User and Librarian inheriting from a common User class).
19. Polymorphism:
    - Methods like add() can be polymorphic, allowing different types of entities to be added.

CRUD Operations:

- Create: add_*() methods for adding librarians, publishers, books, and users.
- Read: Methods for searching and viewing details, e.g., search_books() or view_transaction_history().
- Update: edit_*() methods for modifying librarian, publisher, book, or user details.
- Delete: delete_*() methods for removing librarians, publishers, books, and users.

SQL Foreign Keys:

- Database Design: Use foreign keys to establish relationships between tables (e.g., a foreign key in the Transaction table referencing the Book and User tables).
- CRUD Operations in SQL: SQL queries for CRUD operations are embedded in the methods of the classes (e.g., INSERT INTO, UPDATE, DELETE).

Additional SQL Methods in Library Management System:

SQL Search:

1. Search by Title: Method: search_books_by_title(title)
2. Search by Author:Method: search_books_by_author(author)
3. Search by User: Method: search_user(username)

SQL Filter:

4. Filter Books by Genre:Method: filter_books_by_genre(genre)
5. Filter Users by Name:Method: filter_users_by_name(name)

SQL Range:

1. Filter Books by Publication Year Range:
   Method: filter_books_by_year_range(start_year, end_year)
2. Filter Users by Registration Year Range:
   Method: filter_users_by_registration_year_range(start_year, end_year)

SQL Find:

1. Find User's Checked-Out Books: Method: find_checked_out_books(username)
2. Find Overdue Books: Method: find_overdue_books()

Implementation:

- Create tables for Librarians, Publishers, Books, Users, and Transactions.
- Classes should initialize with necessary attributes and have methods to interact with the database.
- Incorporate SQL queries within methods to perform CRUD operations on the database.
- Utilize lists, dictionaries, and tuples for efficient data storage and retrieval.
- Implement proper error handling for database operations and user inputs.
- Incorporate these methods into the user interface to allow users and librarians to perform advanced searches.

Important Note: Two people can collaborate to implement the project, and the primary objective is to apply the concepts learned in class. Focus on strategies for applying data structures, OOP concepts, lists, tuples, and dictionaries in this project to enhance its efficiency.

GIT: Each team should create two branches, and each student from the team should use one branch to push their code to the main branch. Other team members should resolve any conflicts before committing to the main branch. All code from both students in the team should be pushed, merged, and committed to the main branch. Remember, this is also a crucial part of the learning process.

Demo: On the upcoming Tuesday, you are required to present a demonstration of the project to me and be prepared to answer my questions.