Mini Project 1

SYSTEM OVERVIEW:

Library System User Guide:

This system offers a comprehensive library management system that allows members to interact with a vast collection of books, manage borrowings, and handle penalties seamlessly. Users can log in to their accounts to view their profiles, borrow books, return them, and pay any outstanding penalties. New users have the option to sign up by providing necessary personal information.

Upon logging in, members must provide their email (case-insensitive for login purposes but stored case-sensitively) and password (case-sensitive) correctly. To sign up, users must enter an email, name, birth year (optional), faculty (optional), and a password. The system ensures the uniqueness of the email address.

Once inside the system, users can navigate through various functionalities based on their membership. They can view their personal information, borrowing history, current borrowings, overdue items, and unpaid penalties. Additionally, the system allows members to search for books using keywords, borrow available books, and return borrowed ones along with the option to write a review.

AVAILABLE COMMANDS:

- **View Profile:** Access personal information, borrowing history, and penalty details.
- **Return a Book:** List current borrowings and process returns with automatic penalty calculation for overdue returns. Optionally, write a review.
- **Search and Borrow Books:** Look for books by title or author and initiate a borrowing if the book is available.
- **Pay a Penalty:** View and pay off outstanding penalties partially or in full.
- **Logout:** Safely log out of the system.
- **Exit Program:** Terminate the application.

implementation logic :

connect_to_database(db_path)

- **Functionality:** Establishes a connection to the SQLite database specified by **db_path**. It also sets up the database to enforce foreign key constraints, ensuring referential integrity across the database tables.
- **Implementation Logic:**
  - Uses the **sqlite3.connect** method to connect to the database file specified by **db_path**.
  - Creates a cursor object for executing SQL queries.
  - Enables foreign key constraints by executing the **PRAGMA foreign_keys=ON;** SQL command.

login()

- **Functionality:** Prompts the user for their email and password, and attempts to log them in by validating their credentials against the **members** table in the database.
- **Implementation Logic:**
    - Collects user input for email and password. The password input is obscured using **getpass**.
    - Calls **check_credentials(email, pwd)** to verify if the provided credentials match an entry in the **members** table.
    - If successful, returns the user's email as an acknowledgment of successful login.

check_credentials(email, pwd)

- **Functionality:** Validates a user's login credentials.
- **Implementation Logic:**
    - Executes an SQL query to search for a member whose email and password match the provided credentials. The email comparison is made case-insensitive by applying **lower()** to both the email in the database and the input email.
    - If a matching member is found, the function confirms successful login; otherwise, it indicates failure.

register()

- **Functionality:** Allows new users to create an account by providing their personal information.
- **Implementation Logic:**
    - Collects user inputs for email, name, birth year, faculty, and password. It includes basic validation, such as checking the email format.
    - Inserts a new row into the **members** table with the provided information. Uses parameterized queries to prevent SQL injection.
    - If the email is already registered, it catches a **sqlite3.IntegrityError** and informs the user.

get_member_profile(email)

- **Functionality:** Displays the profile of the logged-in member, including personal information, borrowing history, and unpaid penalties.
- **Implementation Logic:**
    - Executes several SQL queries to retrieve and display:
        - Personal information from the **members** table.
        - Counts of previous, current, and overdue borrowings from the **borrowings** table.
        - Information on unpaid penalties from the **penalties** table, including a total debt amount calculation.

return_book(email)

- **Functionality:** Processes the return of a borrowed book, calculates any overdue penalties, and offers the option to write a book review.
- **Implementation Logic:**

        - Lists current borrowings for the user and prompts for the ID of the book to return.
        - Calculates overdue days and applies a penalty if the book is returned late.
        - Inserts a new penalty record if applicable and updates the **borrowings** table to set the **end_date** for the returned book.
        - Offers the option to write a review and inserts it into the **reviews** table if the user chooses to do so.

search_and_borrow_books(cursor, email)

- **Functionality:** Allows users to search for books by title or author and borrow an available book.
- **Implementation Logic:**
    - Prompts for a keyword and performs a search in the **books** table, displaying results with pagination.
    - If a book is available (not currently borrowed), allows the user to initiate a borrowing by inserting a record into the **borrowings** table with a unique **bid** and the current date as **start_date**.

pay_penalty(email)

- **Functionality:** Allows users to view and pay off their unpaid penalties.
- **Implementation Logic:**
    - Lists all unpaid penalties for the user, including each penalty's ID, associated borrowing ID, and the unpaid amount.
    - Prompts for the ID of the penalty to pay and the amount to pay towards it.
    - Updates the **paid_amount** for the selected penalty in the **penalties** table and recalculates the user's total unpaid debt.

main()

- **Functionality:** The entry point of the application. It orchestrates user login/signup and navigates through various functionalities based on user input.
- **Implementation Logic:**
    - Establishes a database connection and then enters a loop where it presents different options to the user based on whether they are logged in.
    - Processes user choices to invoke the appropriate functionalities like viewing profiles, borrowing or returning books, paying penalties, or logging out.