NEURONETIX

EXPLORE THE FUTURE

ABDELGHAFOR'S HACKATHON

PYTHON HACKATHON

NEURONETIX
EXPLORE THE FUTURE

Local Library Console Management System

Project Overview:

Create a Python-based console application to manage a small community library. This system will provide basic functionalities like cataloging books, managing members, and handling book borrowing and returning. It will be a straightforward solution that meets the needs of small libraries with limited resources and no graphical user interface.

Key Features:

1. **Book Cataloging:**
    - **Data Structures:** Use dictionaries to store and manage book details. Each book will have attributes like Book ID, title, author, genre, and availability status.
    - **Functions:**
        - **add_book():** Add a new book to the catalog.
        - **update_book():** Update details of an existing book (e.g., when new copies are added).
        - **remove_book():** Remove a book from the catalog.
2. **Member Management:**
    - **Data Structures:** Store member information using a dictionary with attributes like Member ID, name, and borrowed books.
    - **Functions:**
        - **register_member():** Register a new member.
        - **update_member():** Update member details or borrowing history.
        - **remove_member():** Remove a member from the system.
3. **Book Borrowing and Returning:**
    - **Functions:**
        - **borrow_book():** Handle the borrowing process, checking book availability, updating the catalog, and recording the transaction.

        - **return_book()**: Handle the return process, updating book availability, and removing the book from the member's borrowed list.
        - Include a due date feature to manage borrowing periods and calculate late fees.
4. **Simple Reporting:**
    - **Functions:**
        - **list_borrowed_books()**: Display a list of currently borrowed books and their due dates.
        - **list_overdue_books()**: Show books that are overdue and need to be returned.
        - **most_popular_books()**: Generate a list of the most frequently borrowed books.
5. **User Interface:**
    - Provide a command-line menu with options for each feature, allowing users to choose actions by entering commands.
    - Implement a simple text-based interface for interacting with the system, ensuring it's intuitive and easy to navigate.
6. **Error Handling:**
    - Handle common errors such as attempting to borrow a book that is already checked out, invalid member IDs, or issues with file access.
    - Use custom exceptions to provide clear error messages for specific scenarios, such as trying to add a book with a duplicate ID.
7. **File Handling:**
    - **Functions:**
        - **load_data():** Load book and member data from text files at startup.
        - **save_data()**: Save the current state of the catalog and member information to text files before exiting.
        - Implement data persistence to ensure that information is retained between sessions.

Project Requirements:

1. **Book Cataloging:**
    - Implement functions to add, update, and remove books.
    - Store book information (Book ID, title, author, genre, availability) using appropriate data structures.
2. **Member Management:**
    - Implement functions to register, update, and remove members.
    - Track member information, including borrowed books.
3. **Book Borrowing and Returning:**
    - Implement borrowing functionality that updates the availability of books and records the member's transaction.
    - Implement return functionality that updates the availability status and handles due dates.
    - Include a feature for managing borrowing periods and calculating late fees.
4. **Simple Reporting:**
    - Provide a feature to list currently borrowed books with due dates.
    - Implement a feature to identify and list overdue books.
    - Provide a report of the most frequently borrowed books.
5. **User Interface:**
    - Create a text-based menu system that allows users to select options for managing books, members, and transactions.
    - Ensure the interface is intuitive and easy to navigate.
6. **Error Handling:**
    - Implement error handling for invalid inputs, file access issues, and logical errors (e.g., borrowing a non-existent book).
    - Use custom exceptions where appropriate to provide clear and specific error messages.
7. **File Handling:**
    - Implement persistent storage using text files for book and member data.

- Ensure the application loads data from files at startup and saves data before exiting.
- Include functionality for backup and restore to prevent data loss.

Criteria for Evaluation:

1. **Functionality (35%)**
   - How well the application meets the project requirements and performs the expected tasks.
2. **Code Quality (20%)**
   - The organization, readability, and structure of the code.
3. **User Interface and Experience (15%)**
   - The intuitiveness and usability of the console application.
4. **Error Handling (10%)**
   - The effectiveness of error management within the application.
5. **Data Persistence and File Handling (10%)**
   - The reliability and correctness of saving/loading data.
6. **Presentation (10%)**
   - The quality of the project presentation, including content, structure, visuals, and delivery.
7. **Creativity and Additional Features (5%)**
   - Any extra features or innovative approaches implemented in the project.

Bonus Points (Optional):

- **Implementation of Advanced Features:**
  - Handling overdue books with notifications or reminders.
  - Implementing search functionality for books or members.
  - Advanced reporting features, such as monthly borrowing trends or member activity logs.

GOOD LUCK