Library Management System

The Library Management System is a software application designed to manage the operations of a library. It provides functionalities for adding, updating, and deleting books and users, as well as for checking out and checking in books.

Features:

- **Add Book**: Allows adding a new book to the library.
- **Update Book**: Allows updating information about a book in the library.
- **Delete Book**: Allows deleting a book from the library.
- **List Books**: Displays a list of all books in the library.
- **Search Book**: Allows searching for a book by title, author, or ISBN.
- **Add User**: Allows adding a new user to the library system.
- **Update User**: Allows updating information about a user in the library system.
- **Delete User**: Allows deleting a user from the library system.
- **List Users**: Displays a list of all users in the library system.
- **Search User**: Allows searching for a user by ID, name, or email.
- **Checkout Book**: Allows a user to checkout a book from the library.
- **Checkin Book**: Allows a user to return a book to the library.
- **List Checkouts**: Displays a list of all book checkouts.
- **Search Checkout**: Allows searching for a checkout transaction by user ID, book ISBN, or checkout date.
- **List Checkins**: Displays a list of all book checkins.
- **Search Checkin**: Allows searching for a checkin transaction by user ID, book ISBN, or checkin date.

Project Structure:

├── book.py

├── check.py

├── main.py

├── models.py

├── storage.py

└── user.py

The project consists of the following files:

1. **book.py**: Contains the `BookManager` class for managing books in the library.
2. **check.py**: Contains the `CheckManager` class for managing checkouts and checkins of books.
3. **main.py**: The main script that serves as the entry point for the application, handling user interactions and the overall flow of the program.
4. **models.py**: Contains data models or classes representing various entities in the library system, such as users, books, and checkouts.
5. **storage.py**: Contains functions for storing and retrieving data, using JSON format.
6. **user.py**: Contains the `UserManager` class for managing users in the library system.

book.py:

The `BookManager` class in `book.py` manages the operations related to books in the library. It provides functionalities for adding, updating, deleting, listing, searching, and retrieving book information. The class interacts with a storage mechanism to save and load book data. Here's a brief description of each method:

- `__init__(self, storage_file)`: Initializes the `BookManager` with a storage file path.
- `save_books(self)`: Saves the current book data to the storage file.
- `is_isbn_unique(self, isbn)`: Checks if a given ISBN is unique in the library.
- `add_book(self, title, author, isbn)`: Adds a new book to the library.
- `update_book(self, isbn, **kwargs)`: Updates the information of a book based on the ISBN.
- `delete_book(self, isbn)`: Deletes a book from the library based on the ISBN.
- `list_books(self)`: Lists all books in the library.
- `search_book(self, key, value)`: Searches for books based on a specific attribute (e.g., title, author, ISBN).
- `get_book(self, isbn)`: Retrieves the book information based on the ISBN.

check.py:

The `CheckManager` class in `check.py` manages the checkout and checkin operations of books in the library. It interacts with a storage mechanism to load and save checkout and checkin data to files. Here's a brief description of each method:

- `__init__(self, filename)`: Initializes the `CheckManager` with the given file path for storing checkouts data.
- `save_checkouts(self)`: Saves the checkout data to a file.
- `save_checkins(self)`: Saves the checkin data to a file.
- `load_checkouts(self)`: Loads the checkout data from a file.
- `load_checkins(self)`: Loads the checkin data from a file.
- `list_checkouts(self)`: Lists all checkout transactions.
- `search_checkout(self, key, value)`: Searches for a checkout transaction based on a key and value (e.g., user ID, ISBN, checkout date).
- `list_checkins(self)`: Lists all checkin transactions.
- `checkout_book(self, user_id, isbn)`: Checks out a book for a user and updates its availability.
- `checkin_book(self, user_id, isbn)`: Checks in a book for a user and updates its availability.
- `exit(self)`: Saves checkout and checkin data to files before exiting.

main.py:

This code defines a library management system that allows users to manage books, users, checkouts, and checkins. Here's a brief description of the functionality:

- `BookManager`, `UserManager`, and `CheckManager` classes manage books, users, and checkouts/checkins, respectively.
- Users can add, update, delete, list, and search for books and users.
- Users can also checkout and checkin books, list checkout transactions, and search for checkout and checkin transactions.
- The main function provides a text-based menu for users to interact with the system.

models.py:

The `models.py` file defines three classes to represent entities in the library management system:

1. `Book`: Represents a book in the library, with attributes for title, author, ISBN, and availability status.
2. `User`: Represents a user of the library, with attributes for name and user ID.
3. `Check`: Represents a checkout transaction, with attributes for user ID, ISBN of the checked-out book, and checkout date.

storage.py:

The `storage.py` module provides functions for saving, loading, and updating data using JSON format. It includes the following functions:

1. `save_data(filename, data)`: Saves data to a file in JSON format.
2. `load_data(filename)`: Loads data from a file in JSON format. If the file does not exist, it creates an empty list.
3. `update_data(filename, data)`: Updates existing data in a file with new data.
4. `save_checkins(data)`: Saves check-in data to a JSON file.
5. `load_checkins()`: Loads check-in data from a JSON file.

user.py:

The `storage.py` module provides functions for handling data storage using JSON format. These functions are designed to save, load, and update data efficiently. Here's a brief description of each function:

1. `save_data(filename, data)`: Saves the provided `data` (which can be a dictionary or a list) to a file specified by `filename` in JSON format.
2. `load_data(filename)`: Loads data from the specified `filename` in JSON format. If the file does not exist, it creates an empty list and returns it.
3. `update_data(filename, data)`: Updates the existing data in the file specified by `filename` with the new `data` provided. It appends the new data to the existing data.
4. `save_checkins(data)`: Saves the check-in data (provided as a list) to a JSON file named `checkins.json`.
5. `load_checkins()`: Loads the check-in data from the `checkins.json` file. If the file does not exist, it returns an empty list.

To use the Library Management System from the GitHub repository you provided, follow these steps:

1. **Clone the Repository:** Clone the repository to your local machine using Git. Open a terminal and run the following command:

   ```
   git clone https://github.com/Kirtana-P/Library-Management-System.git
   ```

2. **Navigate to the Project Directory:** Change to the project directory:

   ```
   cd Library-Management-System
   ```

3. **Install Dependencies:** If the project has any dependencies, install them using pip:

   ```
   pip install -r requirements.txt
   ```

4. **Run the Application:** Run the `main.py` file to start the Library Management System:

   ```
   python main.py
   ```

This will start the application and display the main menu in the terminal.