Library Mangenment System

```
import os
import json
from datetime import datetime

# File paths
BOOKS_FILE = "books.json"
MEMBERS_FILE = "members.json"
TRANSACTIONS_FILE = "transactions.json"

def initialize_files():
    """Create empty files if they don't exist"""
    for file in [BOOKS_FILE, MEMBERS_FILE, TRANSACTIONS_FILE]:
        if not os.path.exists(file):
            with open(file, 'w') as f:
                json.dump([], f)

def load_data(file_name):
    """Load data from JSON file"""
    try:
        with open(file_name, 'r') as f:
            return json.load(f)
    except:
        return []

def save_data(file_name, data):
    """Save data to JSON file"""
    with open(file_name, 'w') as f:
        json.dump(data, f, indent=2)

def add_book():
    """Add a new book to the library"""
    print("\n--- Add New Book ---")
    title = input("Enter book title: ")
    author = input("Enter author name: ")
    isbn = input("Enter ISBN: ")
```

```
    books = load_data(BOOKS_FILE)

    # Check if book already exists
    for book in books:
        if book['isbn'] == isbn:
            print("Error: Book with this ISBN already exists!")
            return

    new_book = {
        'title': title,
        'author': author,
        'isbn': isbn,
        'available': True
    }

    books.append(new_book)
    save_data(BOOKS_FILE, books)
    print("Book added successfully!")

def list_books():
    """Display all books in the library"""
    books = load_data(BOOKS_FILE)

    print("\n--- Library Books ---")
    if not books:
        print("No books in the library.")
        return

    for i, book in enumerate(books, 1):
        status = "Available" if book['available'] else "Checked
Out"
        print(f"{i}. {book['title']} by {book['author']} (ISBN:
{book['isbn']}) - {status}")

def add_member():
    """Add a new library member"""
```

```
    print("\n--- Add New Member ---")
    name = input("Enter member name: ")
    member_id = input("Enter member ID: ")

    members = load_data(MEMBERS_FILE)

    # Check if member already exists
    for member in members:
        if member['id'] == member_id:
            print("Error: Member with this ID already exists!")
            return

    new_member = {
        'name': name,
        'id': member_id,
        'books_borrowed': []
    }

    members.append(new_member)
    save_data(MEMBERS_FILE, members)
    print("Member added successfully!")

def list_members():
    """Display all library members"""
    members = load_data(MEMBERS_FILE)

    print("\n--- Library Members ---")
    if not members:
        print("No members registered.")
        return

    for i, member in enumerate(members, 1):
        print(f"{i}. {member['name']} (ID: {member['id']}) -
Books borrowed: {len(member['books_borrowed'])}")

def checkout_book():
```

```
    """Checkout a book to a member"""
    print("\n--- Checkout Book ---")

    # Display available books
    books = load_data(BOOKS_FILE)
    available_books = [b for b in books if b['available']]

    if not available_books:
        print("No books available for checkout.")
        return

    print("\nAvailable Books:")
    for i, book in enumerate(available_books, 1):
        print(f"{i}. {book['title']} by {book['author']} (ISBN:
{book['isbn']})")

    # Select book
    try:
        book_choice = int(input("\nEnter book number to
checkout: ")) - 1
        selected_book = available_books[book_choice]
    except:
        print("Invalid book selection!")
        return

    # Display members
    members = load_data(MEMBERS_FILE)
    if not members:
        print("No members registered.")
        return

    print("\nMembers:")
    for i, member in enumerate(members, 1):
        print(f"{i}. {member['name']} (ID: {member['id']})")

    # Select member
```

```
    try:
        member_choice = int(input("\nEnter member number: ")) -
1
        selected_member = members[member_choice]
    except:
        print("Invalid member selection!")
        return

    # Update book status
    for book in books:
        if book['isbn'] == selected_book['isbn']:
            book['available'] = False
            break

    # Update member's borrowed books
    for member in members:
        if member['id'] == selected_member['id']:
            member['books_borrowed'].append(selected_book['isbn
'])
            break

    # Record transaction
    transactions = load_data(TRANSACTIONS_FILE)
    transactions.append({
        'type': 'checkout',
        'isbn': selected_book['isbn'],
        'member_id': selected_member['id'],
        'date': datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    })

    # Save all changes
    save_data(BOOKS_FILE, books)
    save_data(MEMBERS_FILE, members)
    save_data(TRANSACTIONS_FILE, transactions)
```

```
    print(f"Book '{selected_book['title']}' checked out to
{selected_member['name']} successfully!")

def return_book():
    """Return a book to the library"""
    print("\n--- Return Book ---")

    # Get member ID
    member_id = input("Enter member ID: ")

    members = load_data(MEMBERS_FILE)
    books = load_data(BOOKS_FILE)

    # Find member
    member = None
    for m in members:
        if m['id'] == member_id:
            member = m
            break

    if not member:
        print("Member not found!")
        return

    if not member['books_borrowed']:
        print("This member has no books to return.")
        return

    print("\nBooks borrowed by this member:")
    borrowed_isbns = member['books_borrowed']
    borrowed_books = []

    for isbn in borrowed_isbns:
        for book in books:
            if book['isbn'] == isbn and not book['available']:
                borrowed_books.append(book)
```

```
                print(f"- {book['title']} by {book['author']}
(ISBN: {book['isbn']})")

    # Select book to return
    isbn_to_return = input("\nEnter ISBN of book to return: ")

    if isbn_to_return not in borrowed_isbns:
        print("This member hasn't borrowed a book with that
ISBN.")
        return

    # Update book status
    for book in books:
        if book['isbn'] == isbn_to_return:
            book['available'] = True
            break

    # Update member's borrowed books
    for m in members:
        if m['id'] == member_id:
            m['books_borrowed'].remove(isbn_to_return)
            break

    # Record transaction
    transactions = load_data(TRANSACTIONS_FILE)
    transactions.append({
        'type': 'return',
        'isbn': isbn_to_return,
        'member_id': member_id,
        'date': datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    })

    # Save all changes
    save_data(BOOKS_FILE, books)
    save_data(MEMBERS_FILE, members)
    save_data(TRANSACTIONS_FILE, transactions)
```

```
        print("Book returned successfully!")

    def show_menu():
        """Display the main menu"""
        print("\nLibrary Management System")
        print("1. Add Book")
        print("2. List All Books")
        print("3. Add Member")
        print("4. List All Members")
        print("5. Checkout Book")
        print("6. Return Book")
        print("7. Exit")

    def main():
        initialize_files()

        while True:
            show_menu()
            choice = input("Enter your choice (1-7): ")

            if choice == '1':
                add_book()
            elif choice == '2':
                list_books()
            elif choice == '3':
                add_member()
            elif choice == '4':
                list_members()
            elif choice == '5':
                checkout_book()
            elif choice == '6':
                return_book()
            elif choice == '7':
                print("Exiting the system. Goodbye!")
                break
```

```
        else:
            print("Invalid choice. Please try again.")

if _name_ == "_main_":
    main()
```