Name: Albert jr. L. Baladia.   Year/Section: BSIT-2D

```
#include <iostream>

#include <vector>

#include <string>

#include <fstream>

#include <algorithm>

#include <limits>


using namespace std;



class Book {

private:

    string title;

    string author;

    string isbn;

    bool isAvailable;


public:

    Book(string t = "", string a = "", string i = "", bool avail = true)

        : title(t), author(a), isbn(i), isAvailable(avail) {}


    string getTitle() const { return title; }

    string getAuthor() const { return author; }
    string getISBN() const { return isbn; }

    bool getAvailability() const { return isAvailable; }


    void setAvailability(bool avail) { isAvailable = avail; }


    string toFileString() const {

        return title + "|" + author + "|" + isbn + "|" + (isAvailable ? "1" : "0");

    }

    static Book fromFileString(const string &line) {

        Book b;

        size_t pos1 = line.find("|");

        size_t pos2 = line.find("|", pos1 + 1);

        size_t pos3 = line.find("|", pos2 + 1);


        b.title = line.substr(0, pos1);

        b.author = line.substr(pos1 + 1, pos2 - pos1 - 1);

        b.isbn = line.substr(pos2 + 1, pos3 - pos2 - 1);

        b.isAvailable = (line.substr(pos3 + 1) == "1");

        return b;

    }

};



class LibraryUser {

private:
    string userID;

    string name;

    vector<string> borrowedBooks;


public:

    LibraryUser(string id = "", string n = "") : userID(id), name(n) {}


    string getUserID() const { return userID; }

    string getName() const { return name; }

    vector<string> getBorrowedBooks() const { return borrowedBooks; }


    void borrowBook(const string &isbn) { borrowedBooks.push_back(isbn); }

    void returnBook(const string &isbn) {

        borrowedBooks.erase(remove(borrowedBooks.begin(), borrowedBooks.end(), isbn),
borrowedBooks.end());

    }




    void displayBorrowedBooks(const vector<Book> &books) const {

        if (borrowedBooks.empty()) {

            cout << "No borrowed books.\n";

            return;

        }

        for (const auto &isbn : borrowedBooks) {

            auto it = find_if(books.begin(), books.end(),

                              [&](const Book &b){ return b.getISBN() == isbn; });
            if (it != books.end())
                cout << "- " << it->getTitle() << " by " << it->getAuthor() << endl;
            else
                cout << "- Unknown Book (ISBN: " << isbn << ")\n";
        }
    }

    string toFileString() const {
        string line = userID + "|" + name + "|";
        for (size_t i = 0; i < borrowedBooks.size(); i++) {
            line += borrowedBooks[i];
            if (i < borrowedBooks.size() - 1) line += ",";
        }
        return line;
    }
    static LibraryUser fromFileString(const string &line) {
        LibraryUser u;
        size_t pos1 = line.find("|");
        size_t pos2 = line.find("|", pos1 + 1);

        u.userID = line.substr(0, pos1);
        u.name = line.substr(pos1 + 1, pos2 - pos1 - 1);
        string booksStr = line.substr(pos2 + 1);

        size_t start = 0, end;
```

```
        while ((end = booksStr.find(",", start)) != string::npos) {
            u.borrowedBooks.push_back(booksStr.substr(start, end - start));
            start = end + 1;
        }
        if (!booksStr.empty())
            u.borrowedBooks.push_back(booksStr.substr(start));

        return u;
    }
};



class Library {
private:
    vector<Book> books;
    vector<LibraryUser> users;

    const string booksFile = "books.txt";
    const string usersFile = "users.txt";

    void loadBooks() {
        ifstream fin(booksFile);
        string line;
        while (getline(fin, line)) {
            if (!line.empty()) books.push_back(Book::fromFileString(line));
    }
    fin.close();
}

void loadUsers() {
    ifstream fin(usersFile);
    string line;
    while (getline(fin, line)) {
        if (!line.empty()) users.push_back(LibraryUser::fromFileString(line));
    }
    fin.close();
}

void saveBooks() const {
    ofstream fout(booksFile);
    for (const auto &b : books) fout << b.toFileString() << endl;
    fout.close();
}

void saveUsers() const {
    ofstream fout(usersFile);
    for (const auto &u : users) fout << u.toFileString() << endl;
    fout.close();
}
```

```
public:
    Library() {
        loadBooks();
        loadUsers();
    }

    ~Library() {
        saveBooks();
        saveUsers();
    }

    void addBook() {
        string title, author, isbn;
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cout << "Enter Title: "; getline(cin, title);
        cout << "Enter Author: "; getline(cin, author);
        cout << "Enter ISBN: "; getline(cin, isbn);
        books.push_back(Book(title, author, isbn, true));
        cout << "Book added successfully.\n";
    }

    void removeBook() {
        string isbn;
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cout << "Enter ISBN to remove: "; getline(cin, isbn);
        books.erase(remove_if(books.begin(), books.end(),
                              [&](const Book &b) { return b.getISBN() == isbn; }),
                    books.end());
        cout << "Book removed if found.\n";
    }

    void registerUser() {
        string id, name;
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cout << "Enter User ID: "; getline(cin, id);
        cout << "Enter Name: "; getline(cin, name);
        users.push_back(LibraryUser(id, name));
        cout << "User registered successfully.\n";
    }

    void removeUser() {
        string id;
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cout << "Enter User ID to remove: "; getline(cin, id);
        users.erase(remove_if(users.begin(), users.end(),
                              [&](const LibraryUser &u) { return u.getUserID() == id; }),
                    users.end());
        cout << "User removed if found.\n";
    }
```

```
void displayAllBooks() const {
    cout << "\n--- All Books ---\n";
    for (const auto &b : books) {
        cout << b.getTitle() << " by " << b.getAuthor()
             << " (ISBN: " << b.getISBN() << ") - "
             << (b.getAvailability() ? "Available" : "Borrowed") << endl;
    }
}

void displayAllUsers() const {
    cout << "\n--- All Users ---\n";
    for (const auto &u : users) {
        cout << u.getUserID() << ": " << u.getName() << endl;
    }
}

void borrowBook() {
    string isbn, userID;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cout << "Enter ISBN: "; getline(cin, isbn);
    cout << "Enter User ID: "; getline(cin, userID);

    for (auto &b : books) {
        if (b.getISBN() == isbn) {
            if (!b.getAvailability()) {
                cout << "Book already borrowed.\n";
                return;
            }
            b.setAvailability(false);
            for (auto &u : users) {
                if (u.getUserID() == userID) {
                    u.borrowBook(isbn);
                    cout << "Book borrowed successfully.\n";
                    return;
                }
            }
        }
    }
    cout << "Book or user not found.\n";
}

void returnBook() {
    string isbn, userID;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cout << "Enter ISBN: "; getline(cin, isbn);
    cout << "Enter User ID: "; getline(cin, userID);

    for (auto &u : users) {
        if (u.getUserID() == userID) {
            u.returnBook(isbn);
            for (auto &b : books) {
                if (b.getISBN() == isbn) {
                    b.setAvailability(true);
                    cout << "Book returned successfully.\n";
                    return;
                }
            }
        }
    }
    cout << "Book or user not found.\n";
}

void displayBorrowedBooks() const {
    string userID;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cout << "Enter User ID: "; getline(cin, userID);

    for (const auto &u : users) {
        if (u.getUserID() == userID) {
            cout << "\nBorrowed Books by " << u.getName() << ":\n";
            u.displayBorrowedBooks(books);
            return;
        }
    }
    cout << "User not found.\n";
    }
};


int main() {
    Library library;
    int choice;

    do {
        cout << "\n===== Library Management Menu =====\n";
        cout << "1. Add Book\n";
        cout << "2. Remove Book\n";
        cout << "3. Register User\n";
        cout << "4. Remove User\n";
        cout << "5. Display All Books\n";
        cout << "6. Display All Users\n";
        cout << "7. Borrow Book\n";
        cout << "8. Return Book\n";
        cout << "9. Display Borrowed Books\n";
        cout << "0. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
            case 1: library.addBook(); break;
            case 2: library.removeBook(); break;

            case 3: library.registerUser(); break;

            case 4: library.removeUser(); break;

            case 5: library.displayAllBooks(); break;

            case 6: library.displayAllUsers(); break;

            case 7: library.borrowBook(); break;

            case 8: library.returnBook(); break;

            case 9: library.displayBorrowedBooks(); break;

            case 0: cout << "Exiting program...\n"; break;

            default: cout << "Invalid choice.\n";

        }

    } while (choice != 0);


    return 0;

}
```