```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <algorithm>

using namespace std;

// =============== Book Class (basic book data holder) ===============
class Book {
private:
    string title;
    string author;
    string isbn;
    bool available;

public:
    Book() : title(""), author(""), isbn(""), available(true) {}
    Book(string t, string a, string i) {
        title = t;
        author = a;
        isbn = i;
        available = true;   // by default every new book is available
    }

    string getTitle() const { return title; }
    string getAuthor() const { return author; }
    string getISBN() const { return isbn; }
    bool isAvailable() const { return available; }

    void setTitle(string t) { title = t; }
    void setAuthor(string a) { author = a; }
    void setISBN(string i) { isbn = i; }
    void setAvailability(bool status) { available = status; }

    string serialize() const {
        return title + "|" + author + "|" + isbn + "|" + (available ? "1" : "0");
    }

    void deserialize(const string& line) {
        size_t p1 = line.find('|');
        size_t p2 = line.find('|', p1 + 1);
        size_t p3 = line.find('|', p2 + 1);

        title = line.substr(0, p1);
        author = line.substr(p1 + 1, p2 - p1 - 1);
        isbn = line.substr(p2 + 1, p3 - p2 - 1);

        string availStr = line.substr(p3 + 1);
        available = (availStr == "1");
    }
};
```

```
// =============== User Class (library member) ===============
class LibraryUser {
private:
    string userID;
    string fullName;
    vector<string> borrowedList;

public:
    LibraryUser() : userID(""), fullName("") {}
    LibraryUser(string id, string n) {
        userID = id;
        fullName = n;
    }

    string getUserID() const { return userID; }
    string getName() const { return fullName; }
    vector<string> getBorrowedBooks() const { return borrowedList; }

    void borrowBook(string isbn) {
        borrowedList.push_back(isbn);
    }

    void returnBook(string isbn) {
        auto it = find(borrowedList.begin(), borrowedList.end(), isbn);
        if (it != borrowedList.end()) {
            borrowedList.erase(it);
        }
    }

    void displayBorrowedBooks() const {
        cout << "Borrowed Books by " << fullName << ": ";
        if (borrowedList.empty()) {
            cout << "None";
        } else {
            for (auto& bookIsbn : borrowedList) {
                cout << bookIsbn << " ";
            }
        }
        cout << endl;
    }

    string serialize() const {
        string data = userID + "|" + fullName + "|";
        for (size_t i = 0; i < borrowedList.size(); i++) {
            data += borrowedList[i];
            if (i < borrowedList.size() - 1) data += ",";
        }
        return data;
    }

    void deserialize(const string& line) {
        size_t p1 = line.find('|');
```

```
        size_t p2 = line.find('|', p1 + 1);

        userID = line.substr(0, p1);
        fullName = line.substr(p1 + 1, p2 - p1 - 1);

        borrowedList.clear();
        string borrowed = line.substr(p2 + 1);

        size_t pos = 0, next;
        while ((next = borrowed.find(',', pos)) != string::npos) {
            borrowedList.push_back(borrowed.substr(pos, next - pos));
            pos = next + 1;
        }
        if (!borrowed.empty()) {
            borrowedList.push_back(borrowed.substr(pos));
        }
    }
};

// =============== Library Class (the "manager") ===============
class Library {
private:
    vector<Book> books;
    vector<LibraryUser> users;
    string bookFile = "books.txt";
    string userFile = "users.txt";

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

    void loadBooks() {
        ifstream in(bookFile);
        string line;
        while (getline(in, line)) {
            if (line.empty()) continue;
            Book b;
            b.deserialize(line);
            books.push_back(b);
        }
        in.close();
    }

    void saveBooks() {
        ofstream out(bookFile);
        for (auto& b : books) {
            out << b.serialize() << endl;
        }
        out.close();
    }
```

```
void loadUsers() {
    ifstream in(userFile);
    string line;
    while (getline(in, line)) {
        if (line.empty()) continue;
        LibraryUser u;
        u.deserialize(line);
        users.push_back(u);
    }
    in.close();
}

void saveUsers() {
    ofstream out(userFile);
    for (auto& u : users) {
        out << u.serialize() << endl;
    }
    out.close();
}

void addBook() {
    string title, author, isbn;
    cout << "Enter book title: "; getline(cin, title);
    cout << "Enter book author: "; getline(cin, author);
    cout << "Enter book ISBN: "; getline(cin, isbn);

    Book b(title, author, isbn);
    books.push_back(b);
    saveBooks();

    cout << "Book added successfully." << endl;
}

void removeBook() {
    string isbn;
    cout << "Enter ISBN of book to remove: "; getline(cin, isbn);

    auto it = remove_if(books.begin(), books.end(), [&](Book& b) {
        return b.getISBN() == isbn;
    });

    if (it != books.end()) {
        books.erase(it, books.end());
        saveBooks();
        cout << "Book removed." << endl;
    } else {
        cout << "Book not found." << endl;
    }
}

void displayBooks() const {
    cout << "\n--- Books in Library ---\n";
    for (auto& b : books) {
```

```
        cout << "Title: " << b.getTitle()
             << " | Author: " << b.getAuthor()
             << " | ISBN: " << b.getISBN()
             << " | Status: " << (b.isAvailable() ? "Available" : "Borrowed")
             << endl;
    }
}

void registerUser() {
    string id, name;
    cout << "Enter user ID: "; getline(cin, id);
    cout << "Enter user name: "; getline(cin, name);

    users.push_back(LibraryUser(id, name));
    saveUsers();
    cout << "User registered successfully." << endl;
}

void removeUser() {
    string id;
    cout << "Enter User ID to remove: "; getline(cin, id);

    auto it = remove_if(users.begin(), users.end(), [&](LibraryUser& u) {
        return u.getUserID() == id;
    });

    if (it != users.end()) {
        users.erase(it, users.end());
        saveUsers();
        cout << "User removed." << endl;
    } else {
        cout << "User not found." << endl;
    }
}

void displayUsers() const {
    cout << "\n--- Library Users ---\n";
    for (auto& u : users) {
        cout << "ID: " << u.getUserID() << " | Name: " << u.getName() << endl;
        u.displayBorrowedBooks();
    }
}

void borrowBook() {
    string userID, isbn;
    cout << "Enter User ID: "; getline(cin, userID);
    cout << "Enter ISBN to borrow: "; getline(cin, isbn);

    auto userIt = find_if(users.begin(), users.end(), [&](LibraryUser& u) {
        return u.getUserID() == userID;
    });

    auto bookIt = find_if(books.begin(), books.end(), [&](Book& b) {
```

```
            return b.getISBN() == isbn;
        });

        if (userIt != users.end() && bookIt != books.end()) {
            if (bookIt->isAvailable()) {
                bookIt->setAvailability(false);
                userIt->borrowBook(isbn);
                saveBooks();
                saveUsers();
                cout << "Book borrowed successfully." << endl;
            } else {
                cout << "Book is already borrowed." << endl;
            }
        } else {
            cout << "Invalid user or book." << endl;
        }
    }

    void returnBook() {
        string userID, isbn;
        cout << "Enter User ID: "; getline(cin, userID);
        cout << "Enter ISBN to return: "; getline(cin, isbn);

        auto userIt = find_if(users.begin(), users.end(), [&](LibraryUser& u) {
            return u.getUserID() == userID;
        });

        auto bookIt = find_if(books.begin(), books.end(), [&](Book& b) {
            return b.getISBN() == isbn;
        });

        if (userIt != users.end() && bookIt != books.end()) {
            userIt->returnBook(isbn);
            bookIt->setAvailability(true);
            saveBooks();
            saveUsers();
            cout << "Book returned successfully." << endl;
        } else {
            cout << "Invalid user or book." << endl;
        }
    }
};

int main() {
    Library lib;
    int choice;

    while (true) {
        cout << "\n===== Library Management System =====\n";
        cout << "1. Add Book\n2. Remove Book\n3. Display Books\n";
        cout << "4. Register User\n5. Remove User\n6. Display Users\n";
        cout << "7. Borrow Book\n8. Return Book\n9. Exit\n";
        cout << "Enter choice: ";
```

```
        cin >> choice;
        cin.ignore();

        switch (choice) {
            case 1: lib.addBook(); break;
            case 2: lib.removeBook(); break;
            case 3: lib.displayBooks(); break;
            case 4: lib.registerUser(); break;
            case 5: lib.removeUser(); break;
            case 6: lib.displayUsers(); break;
            case 7: lib.borrowBook(); break;
            case 8: lib.returnBook(); break;
            case 9: cout << "Exiting..." << endl; return 0;
            default: cout << "Invalid choice." << endl;
        }
    }
}
```