```cpp
#include <iostream>
#include <vector>
#include <string>
using namespace std;

class Book {
    string title;
    string author;
    string isbn;
    bool available;

public:
    Book(string t, string a, string i) {
        title = t;
        author = a;
        isbn = i;
        available = true;
    }

    void showDetails() {
        cout << "Title: " << title << " | Author: " << author << " | ISBN: " << isbn;
        cout << " | " << (available ? "Available" : "Borrowed") << endl;
    }

    void borrow() {
        if (available) {
            available = false;
        } else {
            cout << "Book already borrowed!\n";
        }
    }

    void giveBack() {
        available = true;
    }

    bool isAvailable() {
        return available;
    }

    string getISBN() {
        return isbn;
    }

    string getTitle() {
        return title;
    }
};

```

```
};

class Patron {
    string name;
    int id;
    vector<Book*> borrowed;

public:
    Patron(string n, int i) {
        name = n;
        id = i;
    }

    void borrowBook(Book &b) {
        if (b.isAvailable()) {
            b.borrow();
            borrowed.push_back(&b);
            cout << name << " borrowed \"" << b.getTitle() << "\"\n";
        } else {
            cout << "Book not available.\n";
        }
    }

    void returnBook(Book &b) {
        bool found = false;
        for (int i = 0; i < borrowed.size(); i++) {
            if (borrowed[i]->getISBN() == b.getISBN()) {
                b.giveBack();
                borrowed.erase(borrowed.begin() + i);
                cout << name << " returned \"" << b.getTitle() << "\"\n";
                found = true;
                break;
            }
        }
        if (!found) {
            cout << "You didn't borrow this book.\n";
        }
    }

    void listBooks() {
        cout << "Books borrowed by " << name << ":\n";
        if (borrowed.size() == 0) {
            cout << "- None -\n";
        } else {
            for (auto b : borrowed) {
                b->showDetails();
            }
```

```
            }
        }
    }

    int getID() {
        return id;
    }

    string getName() {
        return name;
    }
};

Book* getBook(vector<Book> &books, string isbn) {
    for (int i = 0; i < books.size(); i++) {
        if (books[i].getISBN() == isbn)
            return &books[i];
    }
    return nullptr;
}

Patron* getPatron(vector<Patron> &users, int pid) {
    for (int i = 0; i < users.size(); i++) {
        if (users[i].getID() == pid)
            return &users[i];
    }
    return nullptr;
}

int main() {
    vector<Book> books;
    vector<Patron> patrons;

    int opt;

    do {
        cout << "\n--- Library Menu ---\n";
        cout << "1. Add Book\n";
        cout << "2. Register Patron\n";
        cout << "3. Borrow Book\n";
        cout << "4. Return Book\n";
        cout << "5. Show All Books\n";
        cout << "6. Show Patron's Books\n";
        cout << "0. Exit\n";
        cout << "Choice: ";
        cin >> opt;
        cin.ignore();

        if (opt == 1) {
```

```
136         cout << "6. Show Patron's Books\n";
137         cout << "0. Exit\n";
138         cout << "Choice: ";
139         cin >> opt;
140         cin.ignore();
141
142         if (opt == 1) {
143             string t, a, i;
144             cout << "Title: ";
145             getline(cin, t);
146             cout << "Author: ";
147             getline(cin, a);
148             cout << "ISBN: ";
149             getline(cin, i);
150             Book newBook(t, a, i);
151             books.push_back(newBook);
152             cout << "Book added.\n";
153
154         } else if (opt == 2) {
155             string n;
156             int id;
157             cout << "Name: ";
158             getline(cin, n);
159             cout << "ID: ";
160             cin >> id;
161             cin.ignore();
162             Patron p(n, id);
163             patrons.push_back(p);
164             cout << "Patron registered.\n";
165
166         } else if (opt == 3) {
167             int pid;
168             string isbn;
169             cout << "Patron ID: ";
170             cin >> pid;
171             cin.ignore();
172             cout << "ISBN: ";
173             getline(cin, isbn);
174
175             Patron* p = getPatron(patrons, pid);
176             Book* b = getBook(books, isbn);
177
178             if (p && b) {
179                 p->borrowBook(*b);
180             } else {
181                 cout << "Invalid ID or ISBN.\n";
182             }
183
184         } else if (opt == 4) {
```

```
181             cout << "Invalid ID or ISBN.\n";
182         }
183
184     } else if (opt == 4) {
185         int pid;
186         string isbn;
187         cout << "Patron ID: ";
188         cin >> pid;
189         cin.ignore();
190         cout << "ISBN: ";
191         getline(cin, isbn);
192
193         Patron* p = getPatron(patrons, pid);
194         Book* b = getBook(books, isbn);
195
196         if (p && b) {
197             p->returnBook(*b);
198         } else {
199             cout << "Invalid ID or ISBN.\n";
200         }
201
202     } else if (opt == 5) {
203         cout << "All Library Books:\n";
204         for (int i = 0; i < books.size(); i++) {
205             books[i].showDetails();
206         }
207
208     } else if (opt == 6) {
209         int pid;
210         cout << "Enter Patron ID: ";
211         cin >> pid;
212         Patron* p = getPatron(patrons, pid);
213         if (p) {
214             p->listBooks();
215         } else {
216             cout << "Patron not found.\n";
217         }
218
219     } else if (opt == 0) {
220         cout << "Bye!\n";
221     } else {
222         cout << "Invalid option.\n";
223     }
224
225 } while (opt != 0);
226
227 return 0;
228 }
229
```

I designed two main classes for this library system, Book to hold informations like title, author, and availability and Patron to represent library users with their name and ID. The key relationship is managed by the Patron class, which keeps track of borrowed books using a vector containing pointers to Book objects. I decided to use pointers instead of copying the Book objects themselves this was important to ensure that when a patron borrows or returns a book, the status change is reflected in one single Book object in the main library list, keeping everything consistent.