```
#include <iostream>
#include <vector>
#include <algorithm>
#include <string>
#include <fstream>
#include <sstream>
#include <ctime>
#include <iomanip>
#include <limits>

using namespace std;

class Book {
public:
    int id;
    string title;
    string author;
    bool isIssued;
    string dueDate;
    string issuedTo;
    int timesIssued;

    Book(int id, const string &title, const string &author)
        : id(id), title(title), author(author), isIssued(false), dueDate("N/A"), issuedTo(""),
timesIssued(0) {}

    void display() const {
        cout << left << setw(6) << id << setw(20) << title << setw(20) << author << setw(10) <<
(isIssued ? "Issued" : "Available")
             << setw(15) << issuedTo << setw(15) << dueDate << setw(10) << timesIssued << endl;
    }

    string getDetailsForFile() const {
        ostringstream details;
        details << id << "," << title << "," << author << "," << isIssued << "," << issuedTo << ","
<< dueDate << "," << timesIssued;
        return details.str();
    }

    static bool compareById(const Book &a, const Book &b) {
        return a.id < b.id;
    }

    static bool compareByTitle(const Book &a, const Book &b) {
        return a.title < b.title;
    }
};

class Library {
private:
    vector<Book> books;

    int findBookIndexById(int id) const {
        for (size_t i = 0; i < books.size(); ++i) {
            if (books[i].id == id) {
                return i;
            }
        }
        return -1;
    }

    string generateDueDate() const {
        time_t now = time(0);
        tm *ltm = localtime(&now);
        ltm->tm_mday += 14; // Assuming 14 days issue period
        mktime(ltm);

        stringstream ss;
        ss << 1900 + ltm->tm_year << "-" << setw(2) << setfill('0') << 1 + ltm->tm_mon << "-" <<
```

```
setw(2) << ltm->tm_mday;
        return ss.str();
    }

public:
    void addBook(int id, const string &title, const string &author) {
        if (findBookIndexById(id) == -1) {
            books.push_back(Book(id, title, author));
            cout << "Book added successfully." << endl;
            saveData();
        } else {
            cout << "Book with this ID already exists." << endl;
        }
    }

    void searchBookById(int id) const {
        int index = findBookIndexById(id);
        if (index != -1) {
            cout << left << setw(6) << "ID" << setw(20) << "Title" << setw(20) << "Author" <<
setw(10) << "Status"
                 << setw(15) << "Issued To" << setw(15) << "Due Date" << setw(10) << "Times Issued"
<< endl;
            cout << string(96, '-') << endl;
            books[index].display();
        } else {
            cout << "Book not found." << endl;
        }
    }

    void searchBookByTitle(const string &title) const {
        bool found = false;
        cout << left << setw(6) << "ID" << setw(20) << "Title" << setw(20) << "Author" << setw(10)
<< "Status"
             << setw(15) << "Issued To" << setw(15) << "Due Date" << setw(10) << "Times Issued" <<
endl;
        cout << string(96, '-') << endl;
        for (const auto &book : books) {
            if (book.title == title) {
                book.display();
                found = true;
            }
        }
        if (!found) {
            cout << "Book with title '" << title << "' not found." << endl;
        }
    }

    void listAllBooks() const {
        if (books.empty()) {
            cout << "No books available in the library." << endl;
        } else {
            char choice;
            cout << "Sort by (I)d or (T)itle: ";
            cin >> choice;
            cin.ignore(numeric_limits<streamsize>::max(), '\n');

            vector<Book> sortedBooks = books;
            if (toupper(choice) == 'I') {
                sort(sortedBooks.begin(), sortedBooks.end(), Book::compareById);
            } else if (toupper(choice) == 'T') {
                sort(sortedBooks.begin(), sortedBooks.end(), Book::compareByTitle);
            } else {
                cout << "Invalid choice. Sorting by ID by default." << endl;
                sort(sortedBooks.begin(), sortedBooks.end(), Book::compareById);
            }

            cout << left << setw(6) << "ID" << setw(20) << "Title" << setw(20) << "Author" <<
setw(10) << "Status"
                 << setw(15) << "Issued To" << setw(15) << "Due Date" << setw(10) << "Times Issued"
```

```
<< endl;
            cout << string(96, '-') << endl;
            for (const auto &book : sortedBooks) {
                book.display();
            }
        }
    }

    void displayAvailableBooks() const {
        cout << "Available Books:" << endl;
        cout << left << setw(6) << "ID" << setw(20) << "Title" << setw(20) << "Author" << setw(10)
<< "Status"
             << setw(15) << "Issued To" << setw(15) << "Due Date" << setw(10) << "Times Issued" <<
endl;
        cout << string(96, '-') << endl;
        for (const auto &book : books) {
            if (!book.isIssued) {
                book.display();
            }
        }
    }

    void issueBook(int id, const string &student) {
        int index = findBookIndexById(id);
        if (index != -1) {
            if (!books[index].isIssued) {
                books[index].isIssued = true;
                books[index].issuedTo = student;
                books[index].dueDate = generateDueDate();
                books[index].timesIssued++;
                cout << "Book issued to " << student << " with due date " << books[index].dueDate <<
"." << endl;
                saveData();
            } else {
                cout << "Book is already issued." << endl;
            }
        } else {
            cout << "Book not found." << endl;
        }
    }

    void returnBook(int id) {
        int index = findBookIndexById(id);
        if (index != -1) {
            if (books[index].isIssued) {
                books[index].isIssued = false;
                books[index].issuedTo = "";
                books[index].dueDate = "N/A";
                cout << "Book returned successfully." << endl;
                saveData();
            } else {
                cout << "Book was not issued." << endl;
            }
        } else {
            cout << "Book not found." << endl;
        }
    }

    void deleteBook(int id) {
        int index = findBookIndexById(id);
        if (index != -1) {
            books.erase(books.begin() + index);
            cout << "Book deleted successfully." << endl;
            saveData();
        } else {
            cout << "Book not found." << endl;
        }
    }
```

```
    void saveData() const {
        ofstream file("library_data.txt");
        if (file.is_open()) {
            for (size_t i = 0; i < books.size(); ++i) {
                file << books[i].getDetailsForFile() << endl;
            }
            file.close();
            cout << "Data saved successfully." << endl;
        } else {
            cout << "Unable to open file for saving data." << endl;
        }
    }

    void loadData() {
        ifstream file("library_data.txt");
        if (file.is_open()) {
            books.clear();
            string line;
            while (getline(file, line)) {
                int id;
                string title, author, issuedTo, dueDate;
                bool isIssued;
                int timesIssued;
                char delimiter;

                istringstream stream(line);
                stream >> id >> delimiter;
                getline(stream, title, ',');
                getline(stream, author, ',');
                stream >> isIssued >> delimiter;
                getline(stream, issuedTo, ',');
                getline(stream, dueDate, ',');
                stream >> timesIssued;

                Book book(id, title, author);
                book.isIssued = isIssued;
                book.issuedTo = issuedTo;
                book.dueDate = dueDate;
                book.timesIssued = timesIssued;
                books.push_back(book);
            }
            file.close();
            cout << "Data loaded successfully." << endl;
        } else {
            cout << "Unable to open file for loading data." << endl;
        }
    }
};

void clearScreen() {
    cout << "\033[2J\033[1;1H"; // ANSI escape code to clear screen
}

void displayMenu() {
    cout << "======================================" << endl;
    cout << "     WELCOME TO THE LIBRARY SYSTEM      " << endl;
    cout << "======================================" << endl;
    cout << "1. Add New Book" << endl;
    cout << "2. Search for a Book by ID" << endl;
    cout << "3. Search for a Book by Title" << endl;
    cout << "4. Issue a Book" << endl;
    cout << "5. Return a Book" << endl;
    cout << "6. List All Books" << endl;
    cout << "7. Delete a Book" << endl;
    cout << "8. Exit" << endl;
    cout << "======================================" << endl;
    cout << "Enter your choice: ";
}
```

```
int main() {
    Library library;
    library.loadData();

    int choice;
    do {
        clearScreen();
        displayMenu();
        cin >> choice;
        cin.ignore(numeric_limits<streamsize>::max(), '\n');

        clearScreen();
        switch (choice) {
            case 1: {
                int id;
                string title, author;
                cout << "Enter Book ID: ";
                cin >> id;
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                cout << "Enter Book Title: ";
                getline(cin, title);
                cout << "Enter Book Author: ";
                getline(cin, author);
                library.addBook(id, title, author);
                break;
            }
            case 2: {
                int id;
                cout << "Enter Book ID to search: ";
                cin >> id;
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                library.searchBookById(id);
                break;
            }
            case 3: {
                string title;
                cout << "Enter Book Title to search: ";
                getline(cin, title);
                library.searchBookByTitle(title);
                break;
            }
            case 4: {
                library.displayAvailableBooks();
                int id;
                string student;
                cout << "Enter Book ID to issue: ";
                cin >> id;
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                cout << "Enter Student Name: ";
                getline(cin, student);
                library.issueBook(id, student);
                break;
            }
            case 5: {
                int id;
                cout << "Enter Book ID to return: ";
                cin >> id;
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                library.returnBook(id);
                break;
            }
            case 6: {
                library.listAllBooks();
                break;
            }
            case 7: {
                int id;
                cout << "Enter Book ID to delete: ";
                cin >> id;
```

```
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                library.deleteBook(id);
                break;
            }
            case 8: {
                cout << "Exiting program..." << endl;
                break;
            }
            default:
                cout << "Invalid choice. Please try again." << endl;
        }

        if (choice != 8) {
            cout << "Press Enter to continue...";
            cin.get();
        }
    } while (choice != 8);

    library.saveData();
    return 0;
}
```