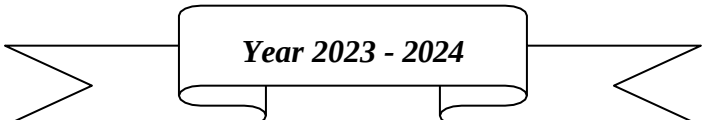
INTERNATIONAL SCHOOL – VIETNAM NATIONAL UNIVERSITY
Informatics and Computer Engineering

FINAL EXAM

TOPIC: C++ Library Management System Using CSV As Local Storage

Mentor: M.Nguyen Tran Dinh Long
Member: Phung Hoang (20070832)
Uong Thi Nga (21070270)
Bui Linh Dan (20070815)
Group: 6
Class: INS2073-02



1. Member Contribution:

Team Leader: Phung Hoang (20070832)
Contribution: 100%
Work: Main coder, design all algorithm, handle logic and data processing in this project, review team member code and merge them into the main branch.

Team Member 1: Uong Thi Nga (21070270)
Contribution: 80%
Work: Handle the data input and output from class method and display it (Responsible for UI) with leader support. Testing and Do report and prepare slide for the presentation.

Team Member 2: Bui Linh Dan (20070815)
Contribution: 80%
Work: Handle the data input and output from class method and display it (Responsible for UI) with leader support. Testing and Do report and prepare slide for the presentation.

Git Project: phunghoangvnuit/LibMS (github.com)
Note: My team member not familiar working with Git so they share me code on Zalo then I review and merge their code into main branch on Github remote respository. That why you don't see any contributer except "Phung Hoang" – me on GitHub.

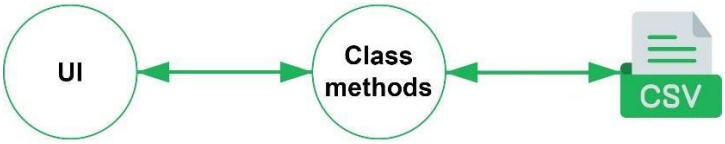
2. Program Overview:

Objectives:
The purpose of our "library management system" is a build a tool for librarian. They can manage books, categories, patrons, loans (included "return" and "borrow"). Only librarian can use it.

Technologies & tool:
Language: C++
Dev tool: Visual Studio 2022
Version control system: Github

3. Program Design:

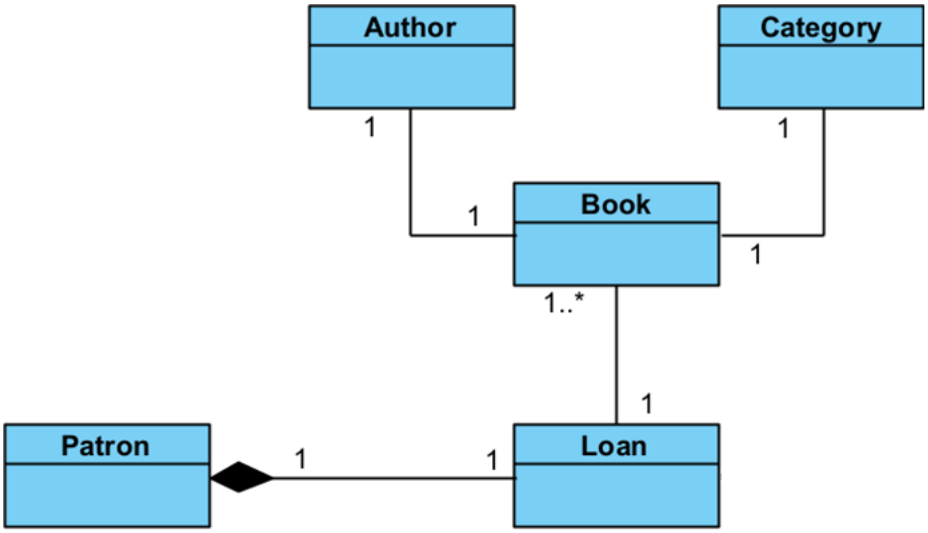
Architecture:



We create forms (create/get/update/delete…) for user to works with library resoureces. Through these forms, users can create requests for the program, class method will take the requests and processed then implement C.R.U.D on the CSV, finally notify the results to the user.

UML Classes Diagram:

Each object class will have corresponding attributes and methods. **We would like to not list them here (attribute and method)**, but only show the relationships between classes.



4. Program Implementation:

Coding Standards:

- Naming Convention variable with “camelCase”, and class with “PascalCase”.
- We also used comment to describe method in classes and some important notices related to program. Comment “NOT” is stand for “Not complete – in developing process”.

Key Implementations:

- The core of this project is handling data in CSV files. This is a crucial concept of database connectivity for future programming subjects.
- In our program, we not only create C.R.U.D but also succeed establish relationship between csv files like between tables in SQL Database.

- Below is snippet code for C.R.U.D in csv:

```
// Check BookId If Existed
bool checkIdExisted(int id) {
    fstream bookCSV;
    bookCSV.open("book.csv", ios::in);

    if (bookCSV.is_open()) {
        string line;
        while (getline(bookCSV, line)) {
            vector<string> vectVal;
            stringstream ss(line);

            while (ss.good()) {
                string word;
```

```
            getline(ss, word, ',');
            vectVal.push_back(word);
        }

        if (stoi(vectVal[0]) == id) {
            return true;
        }
    }
    bookCSV.close();
    return false;
  }
}

// Insert Book Into CSV
void insertIntoCSV()
{
    fstream bookCSV;
    bookCSV.open("book.csv", ios::app);
    if (bookCSV.is_open()) {
        bookCSV << this->id << ',';
        bookCSV << this->title << ',';
        bookCSV << this->isbn << ',';
        bookCSV << this->quantity << ',';
        bookCSV << this->categoryId << ',';
        bookCSV << this->authorId << endl;
        bookCSV.close();
    }
}

// Update Book In CSV
void updateInCSV(int id, Book book) {
    fstream bookCSV;
    bookCSV.open("book.csv", ios::in);

    if (bookCSV.is_open()) {
        vector<string> vectObj;
        string line;
        while (getline(bookCSV, line)) {
            vector<string> vectVal;
            stringstream ss(line);

            while (ss.good()) {
                string word;
                getline(ss, word, ',');
                vectVal.push_back(word);
            }

            if (stoi(vectVal[0]) == id) {
                vectVal[0] = to_string(book.getId());
                vectVal[1] = book.getTitle();
                vectVal[2] = book.getIsbn();
```

```
                vectVal[3] = to_string(book.getQuantity());
                vectVal[4] = to_string(book.getCategoryId());
                vectVal[5] = to_string(book.getAuthorId());
                string altLine;
                altLine = vectVal[0] + ','
                        + vectVal[1] + ','
                        + vectVal[2] + ','
                        + vectVal[3] + ','
                        + vectVal[4] + ','
                        + vectVal[5];
                vectObj.push_back(altLine);
            }
            else {
                vectObj.push_back(line);
            }
        }
        bookCSV.close();

        fstream bookCSV;
        bookCSV.open("book.csv", ios::out);
        for (int i = 0; i < vectObj.size(); i++) {

            if (bookCSV.is_open()) {
                bookCSV << vectObj[i] << "\n";
            }
        }
        bookCSV.close();
    }
}

// Get Book From CSV
void getFromCSV() {
    fstream bookCSV;
    bookCSV.open("book.csv", ios::in);

    if (bookCSV.is_open()) {
        string line;
        while (getline(bookCSV, line)) {
            vector<string> vectVal;
            stringstream ss(line);

            while (ss.good()) {
                string word;
                getline(ss, word, ',');
                vectVal.push_back(word);
            }
            Book book;
            vectVal[4] = book.getCategoryById(stoi(vectVal[4]));
            vectVal[5] = book.getAuthorById(stoi(vectVal[5]));
            cout << " ----------------------------------------------------\n";
            cout << "| BookId     :  " << vectVal[0] << " (Current Version)\n";
```

```
            cout << "| Title       :  " << vectVal[1] << "\n";
            cout << "| ISBN        :  " << vectVal[2] << "\n";
            cout << "| Quantity    :  " << vectVal[3] << " (CurrentQty)\n";
            cout << "| Category    :  " << vectVal[4] << "\n";
            cout << "| Author      :  " << vectVal[5] << "\n";
            cout << "-------------------------------------------------------\n";
        }
        bookCSV.close();
    }
}

// Get Book From CSV
void getFromCSV(int id) {
    fstream bookCSV;
    bookCSV.open("book.csv", ios::in);

    if (bookCSV.is_open()) {
        string line;
        while (getline(bookCSV, line)) {
            vector<string> vectVal;
            stringstream ss(line);

            while (ss.good()) {
                string word;
                getline(ss, word, ',');
                vectVal.push_back(word);
            }

            if (stoi(vectVal[0]) == id) {
            Book book;
            vectVal[4] = book.getCategoryById(stoi(vectVal[4]));
            vectVal[5] = book.getAuthorById(stoi(vectVal[5]));
            cout << " -------------------------------------------------------\n";
            cout << "| BookId      :  " << vectVal[0] << " (Current Version)\n";
            cout << "| Title       :  " << vectVal[1] << "\n";
            cout << "| ISBN        :  " << vectVal[2] << "\n";
            cout << "| Quantity    :  " << vectVal[3] << " (CurrentQty)\n";
            cout << "| Category    :  " << vectVal[4] << "\n";
            cout << "| Author      :  " << vectVal[5] << "\n";
            cout << "-------------------------------------------------------\n";
            return;
            }
        }
        bookCSV.close();
    }
}

// Remove Book From CSV
void removeFromCSV(int id) {
    fstream bookCSV;
    bookCSV.open("book.csv", ios::in);
```

```
    if (bookCSV.is_open()) {
        vector<string> vectObj;
        string line;
        while (getline(bookCSV, line)) {
            vector<string> vectVal;
            stringstream ss(line);

            while (ss.good()) {
                string word;
                getline(ss, word, ',');
                vectVal.push_back(word);
            }

            if (stoi(vectVal[0]) != id) {
                vectObj.push_back(line);
            }
        }
        bookCSV.close();

        fstream bookCSV;
        bookCSV.open("book.csv", ios::out);
        for (int i = 0; i < vectObj.size(); i++) {
            if (bookCSV.is_open()) {
                bookCSV << vectObj[i] << "\n";
            }
        }
        bookCSV.close();
    }
}
```

5. Program Evaluation:

Advantages/Strengths:

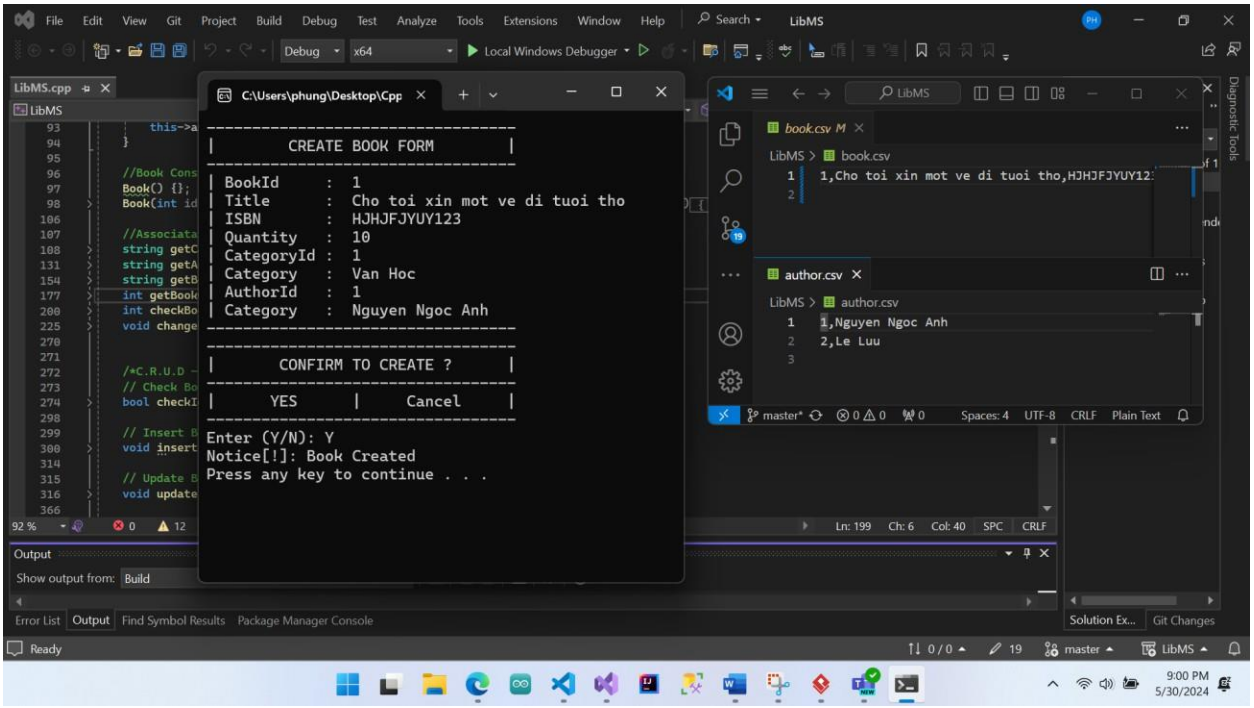
- Multi files handling, simulate the relationship between table like in SQL Database.
- Using csv like a local storage, librarian can save data after they close the program and reused it later.

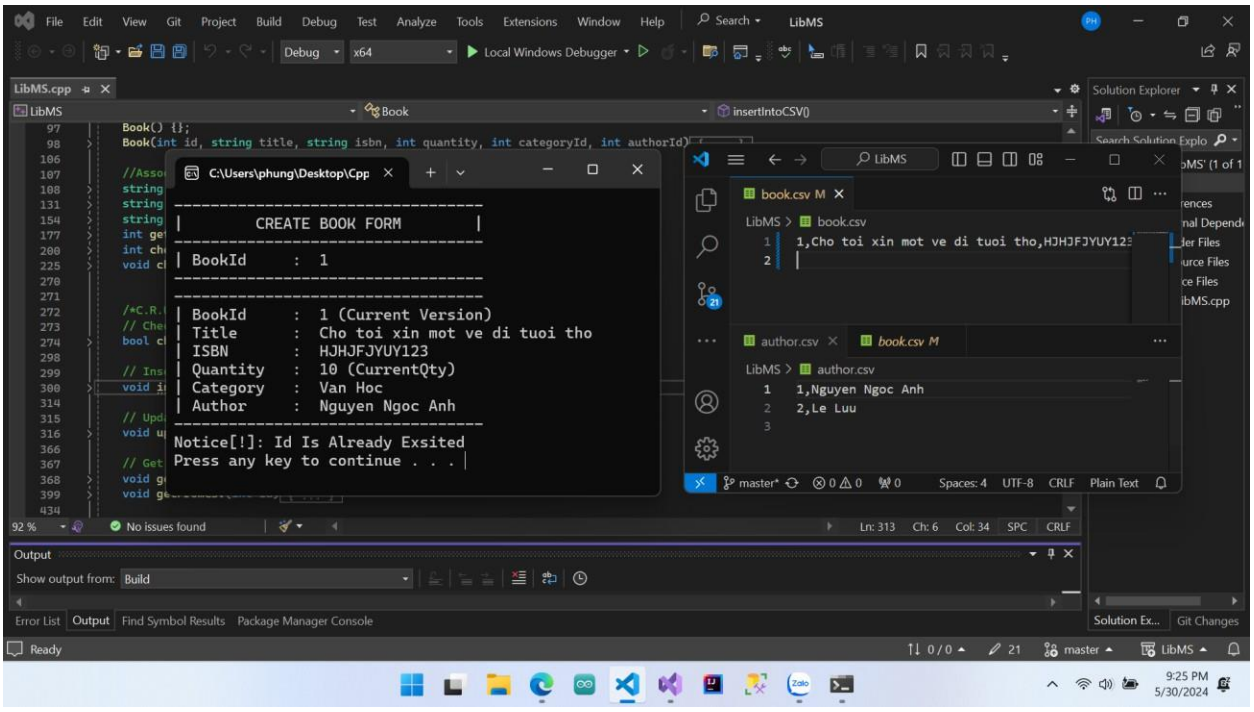
Disadvantages/Weaknesses:

- Not fully optimized, a lot of loops, less reused code.
- There are some bugs related to exception handling.

Future/Possible improvements:

- Sort by keyword. For example: we search patron by keyword “Nguyen”, the system returns results all patron’s names include “Nguyen” -> Nguyen Van A, Nguyen Van B…
- Over due date and fine. If patron not return book in due date, they will be fined. We can create a function to compare borrow date and due date to do that.


C:\Users\phung\Desktop\Cpp
CREATE BOOK FORM
BookId : 1
Title : Cho toi xin mot ve di tuoi tho
ISBN : HJHJFJYUY123
Quantity : 10
CategoryId : 1
Category : Van Hoc
AuthorId : 1
Category : Nguyen Ngoc Anh
CONFIRM TO CREATE ?
YES
Cancel
Enter (Y/N): Y
Notice[!]: Book Created
Press any key to continue . . .
book.csv
1,Cho toi xin mot ve di tuoi tho,HJHJFJYUY12
author.csv
1,Nguyen Ngoc Anh
2,Le Luu


C:\Users\phung\Desktop\Cpp
CREATE BOOK FORM
BookId : 1
BookId : 1 (Current Version)
Title : Cho toi xin mot ve di tuoi tho
ISBN : HJHJFJYUY123
Quantity : 10 (CurrentQty)
Category : Van Hoc
Author : Nguyen Ngoc Anh
Notice[!]: Id Is Already Exsited
Press any key to continue . . .
book.csv
1,Cho toi xin mot ve di tuoi tho,HJHJFJYUY12
author.csv
1,Nguyen Ngoc Anh
2,Le Luu