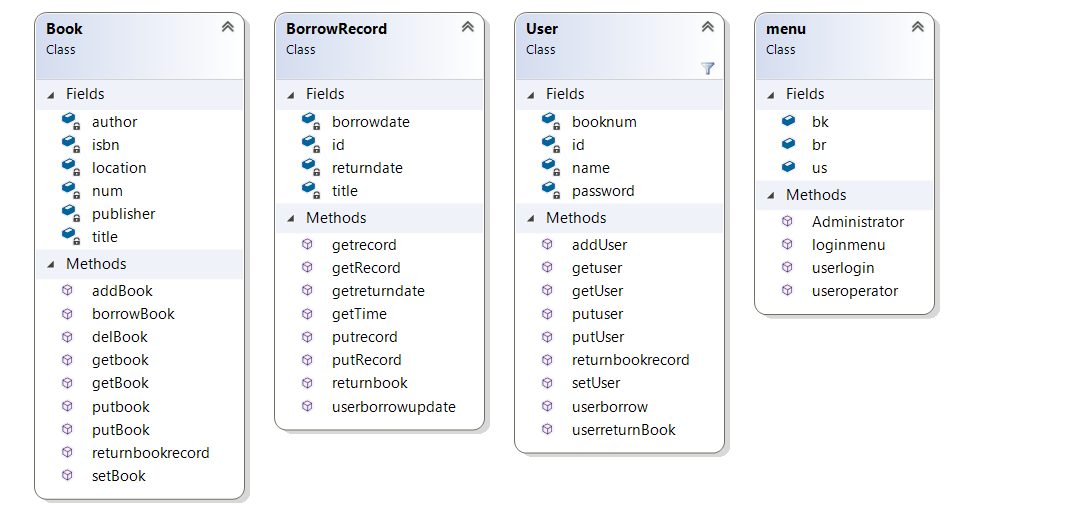
C++ Design Specification

**Assignment Topic:** School Library Management System

**Class Design:** In this big assignment, I designed four classes: Book (book class), User (user class), and BookRecord (record class). Menu, the class diagram is as follows:



**Book Class:**

**Class Data Members:**

**string author** : This data is used to store the author name, which can be called directly when modifying or adding book library records and outputting book materials.

**int ISBN** : This data is used to store the international book number of the book, which can be directly recalled when modifying or adding book library records and outputting book materials, and ensuring that all input is digital.

**string location** : This data is used as the storage location of books, and can be directly called when modifying or adding book library records and outputting book materials.

**int num** : This data is used as the international book number of the stored book, which can be directly called and used when modifying or adding book library records and outputting book information, ensuring that the input is a number, and directly adding or subtracting it when returning or borrowing books.

**string publisher** : This data is used to store book publishers, and can be directly called when modifying or adding book library records and outputting book materials.

**string title**  : This data is used as the storage book title, which can be directly called when modifying or adding book library records and outputting book materials.

**Class function members:**

**getbook**: Before using this function, you need to enter a full line of data (style: book code) obtained directly from the book .csv file Book title, Author's name, ISBN, publisher, location, number of collections), and use find\_first\_of(",") and substr() to separate the data. And use stringstream to convert the international book number and the number of books in the collection from string to int, and convert the title, author name, international book number, publisher, location, The number of books in the collection is stored in that order: title, author, isbn, publisher, location, num.

**getBook**  : Enter numbers representing the new data to be added before using this function ([1] book title, [2] author name, [3] international book number,[4] publisher, [5] location,[6] number of books), using getline(cin,) to ensure that data containing spaces can be included 。 And use if (!( cin >> i) {cin.clear(); cin.ignore(10000, '\n'); } Ensure that there is no error when entering the number of international books and collections without numbers, and store the book code, title, author name, international book number, publisher, location, and number of collections in the title author, isbn, publisher, location, num.

**putbook**  : Enter the book code of the book before using this function, and use setw() to adjust the distance between the data. When outputting books symbol on the far left, and the book information (title, author name, international book number, publisher, location, number of collections) is output on the right, and the data is directly using title, author, ISBN, publisher，location，num。 Therefore, it must be used with getbook or getBook before use .

**putBook**  : Enter the book code of the book before using this function, and use setw() to adjust the distance between the data. Single in each row Independently output the title, author's name, international book number, publisher, location, and number of collections. Data is directly available using title, author, ISBN, publisher, location, num。 Therefore, it must be used with getbook or getBook before use .

**addBook**  : Before using this function, you need to enter an entire line of data obtained directly from the book .csv file (style: book code, book title,). Author name, ISBN, publisher, location, number of collections), the number of columns and book codes corresponding to the data to be changed. This function splits the input row data according to the position of ","" and puts it separately into the vector (the data is placed in the same vector with ","), and directly uses title, author , ISBN, publisher, location, num change data into the corresponding vector Replace the original data, and finally merge the vector into a string to change the initial row data into new row data. Therefore, it must be used with getbook or getBook before use .

**delBook: Before** using this function, you need to enter a whole line of data directly from the book .csv file (style: book code, book title,). Author name, ISBN, publisher, location, number of collections). This function splits the input row data according to the position of ","" and puts it in the vector separately (the data is placed in the same vector with ","), using stringstream The input string line data is converted to int, and only the first "," is output due to the stringstream feature The number before it, that is, the book code. And since the book code is to fill the book code of the deleted book, the book code is subtracted by one and put in the first parameter of the vector instead of the original book code book code 。 Finally, merge the vector into a string, and change the original row data to a new row data.

**setBook: Before** using this function, you need to enter a whole line of data directly from the book.csv file (style: book code, book title,). Author name, ISBN, publisher, location, number of collections) and the number of columns corresponding to the data to be changed. This function splits the input row data according to the position of ","" and puts it in the vector separately (the data is placed in the same vector with ","), and uses it directly according to the number of columns title， author，ISBN，publisher，location，num Put the change data into the corresponding vector instead of the original data, and finally merge the vector into a string to change the initial row data to new row data. Therefore, it must be used with getbook or getBook before use .

**borrowBook : Before**  using this function, you need to enter a whole line of data directly obtained from the book.csv file (style: book code, book title,). Author name, ISBN, publisher, location, number of collections). This function splits the input row data according to the position of ","" and puts it separately into the vector (the data is placed in the same vector with ","), and uses num directly, and subtracts one from it to represent when borrowing a book The number of books in the collection is subtracted by one, and the data is placed at the end of the vector instead of the original number of books, and finally the vector is merged into a string to change the original row data to new row data. Therefore, it must be used with getbook or getBook before use .

**returnbookrecord : Enter** an entire line of data (style: book code, book title, etc.) obtained directly from the book.csv file before using this function

Author name, ISBN, publisher, location, number of collections). This function splits the input row data according to the position of ","" and places it in the vector separately (the data is placed in the same vector with ","), and uses num directly, and adds a word to it to return the book When the number of books in the collection is increased by one, the data is placed at the end of the vector to replace the original number of books, and finally the vector is merged into a string to change the initial row data to new row data. Therefore, it must be used with getbook or getBook before use .

**BorrowRecord :**

**Class Data Members:**

**string borrowdate**  : This data is used to store the borrowing time, which can be directly adjusted when modifying or adding borrowing library records and outputting borrowing materials

Use with.

**int id**  : This data is used as a storage number, which can be directly recalled when modifying or adding library records and outputting borrowing materials

, and make sure all input is 1 0 digit number.

**string returndate** : This data is used to store the borrowing time, which can be directly adjusted when modifying or adding library records and outputting borrowing materials

Use with.

**string title**  : This data is used as the storage title of the book, and can be called directly when modifying or adding library records and outputting borrowing materials

Use.

**Class function members:**

**getTime**: Use localtime() to get the date of the day as the borrowing date.

**getreturndate**  : Use localtime() and tm\_mon +=2 to get the date two months from now as the return date.

**getrecord : Before**  using this function, you need to enter an entire line of data obtained directly from the userborrow .csv file (style: Borrow code, student number, book code, title, borrow date, return date), and use find\_first\_of(",") and substr() to separate the data. And use stringstream to convert the student number from string to int, and finally store the student number, book title, borrowing date, and return date in id, title, borrowdate, returndate.

**getRecord**  : Before using this function, you need to enter the student number and book title, and use stringstream to change the student number from string to int, and getTime() and getReturnDate() get the date of the day and the date 2 months later as the borrowing date and return date, and finally store the student number, book title, borrowing date, and return date in id, title , borrowdate, returndate.

**putrecord**  : This function uses setw() to adjust the distance between the data, and output the book title, borrowing date, and in the same line Return date, data directly using title, borrowdate, returndate. Therefore, it must be used with get record or getrecord before use .

**putRecord**  : Enter the borrow code before using this function, which uses setw() to adjust the distance between the data Output the borrowing code, book title, borrowing date and return date separately in each line, and the data is directly using title, borrowdate, returndate. Therefore, it must be used with getRecord or getRecord before use .

**userborrowupdate: Before** using this function, you need to enter an entire row of data obtained directly from the userborrow .csv file (style: Borrowing code, student number, book code, book title, borrowing date, return date), the number of columns, borrowing code and book code corresponding to the data to be changed. This function splits the input row data according to the position of ","" and puts it in the vector separately (the data is placed in the same vector with ","), and directly uses id, title, BorrowDate, returndate, and the input borrowing code and book code change data are put into the corresponding vector instead of the original data, and finally the vector is replaced Merging into a string changes the initial row data into new row data. Therefore, it must be used with get record or getrecord before use .

**returnbook: Before** using this function, you need to enter an entire row of data obtained directly from the userborrow .csv file (style: Borrowing code, student number, book code, book title, borrowing date, return date). This function splits the input row data according to the position of ","" and puts it in the vector separately (the data is placed in the same vector with ","), and uses the stringstream to put the input string The row data is converted into int, and due to the stringstream feature, only the number before the first "," is output, that is, the borrowing code. And since the borrowing code fills the borrowing code of the deleted borrowing record, the borrowing code is subtracted and placed in the first parameter of the vector instead of the original borrowing code. Finally, merge the vector into a string, and change the original row data to a new row data.

**User class:**

**Class Data Members:**

**string name** : This data is used to store user names, which can be used when modifying or adding user library records and exporting user information Subsequent calls are used.

**int id**  : This data is used to store user numbers, which can be used when modifying or adding user library records and exporting user data Then call to use, and make sure all entries are numeric.

**string password**  : This data is used to store user passwords, which can be used when modifying or adding user library records and outputting user information Subsequent calls are used.

**int booknum**  : This data is used to store the number of books borrowed, and can modify or add user library records and output user information When using the direct call, make sure that the input is a number, and add or subtract it directly when returning or borrowing a book.

**Class function members:**

**getuser: Before** using this function, you need to enter an entire row of data obtained directly from the user .csv file (style: User code, student number, user name, user password, number of borrowed), and use find\_first\_of(",") and substr() to separate the data. And use stringstream to convert the student number and borrowing quantity from string to int, and finally store the student number, user name, user password, and borrowing quantity in id and name , password, booknum.

**getUser**: Enter numbers representing the new data to be added before using this function ([1] student number, [2] user name, [3] user password code,[4] number of borrowed),. Use getline(cin,) when entering information to ensure that data containing spaces can be added, and use if (!( cin >> i) {cin.clear(); cin.ignore(10000, '\n'); } Make sure that there will be no error when the student number and the number of borrowed books are not entered as numbers, and store the student number, user name, user password, and borrowing quantity in id, name, password, booknum.

**putuser**: Before using this function, enter the user's user code and use setw() to adjust the distance between the data. When output, the user symbol on the far left, and user information (user name, student number, user password, number of borrowing) is output on the right, and the data is directly using name, id, password, booknum。 Therefore, it must be used with get user or getuser before use .

**putUser**: Enter the user's user code before using this function and use setw() to adjust the distance between the data. Single in each row The user name, student number, user password, and number of borrowed books are output alone, and the data is directly used name, id, password, booknum. Therefore, it must be used with get user or getuser before use .

**addUser: Before** using this function, you need to enter a whole line of data directly obtained from the user .csv file (in the style: user code, student number, etc.). User name, user password, number of borrowed books), the number of columns and user codes corresponding to the data to be changed. This function splits the input row data according to the position of ","" and puts it in the vector separately (the data is placed in the same vector with ","), and directly uses id, name, password, booknum change the data into the corresponding vector instead of the original data, and finally merge the vector into a string Change the initial row data to the new row data. Therefore, it must be used with getbook or getBook before use .

**setUser : Before**  using this function, you need to enter a whole line of data obtained directly from the user .csv file (in the style: user code, student number, etc.). User name, user password, number of borrowed books) and the number of columns corresponding to the data to be changed. This function splits the input row data according to the position of ","" and puts it in the vector separately (the data is placed in the same vector with ","), and directly uses id, name,, according to the number of columns password, booknum change the data into the corresponding vector instead of the original data, and finally merge the vector into a string Change the initial row data to the new row data. Therefore, it must be used with getbook or getBook before use .

**userborrow : Before**  using this function, you need to enter a whole line of data directly obtained from the user .csv file (style: user code, student number, etc.). User name, user password, number of borrowed books). This function splits the input row data according to the position of ","" and places it in the vector separately (the data is placed in the same vector with ","), and directly uses booknum and adds to it One represents the number of borrowed books when borrowing books, plus one, and the data is placed at the end of the vector instead of the original number of books, and finally the vector is merged into one string changes the initial row data to the new row data. Therefore, it must be used with get user or getuser before use .

**userreturnBook: Before** using this function, you need to enter a whole line of data obtained directly from the user .csv file (in the style: user code, student number, etc.). User name, user password, number of borrowed books). This function splits the input row data according to the position of ","" and puts it in the vector separately (the data is placed in the same vector with ","), and uses booknum directly, and subtracts one from it to borrow the book when returning it The number is subtracted by one and the data is placed at the end of the vector instead of the original number of books, and finally the vector is merged into a string to change the original row data to new row data. Therefore, it must be used with get user or getuser before use .

**Menu class:**

**Class function members:**

**LoginMenu :**  Allows users to choose two login methods : user login and administrator login, and reflects two choices through the switch, if you select a user Login outputs the userlogin function, and administrator login requires a password to enter the administrator interface.

**userlogin**  : First ask the user to enter the student number, and then look for the entire line of data corresponding to the student number from the user .csv file, and look for the secret Yard. Then ask the user to enter the password to match, if the password is consistent, the login is successfully output useroperator.

**Administrator** : Let the administrator choose the function that needs to be executed, implement it through the switch, and call out the function of other classes in the function Now functional.

**useroperator**  : Before using this function, you need to enter the student number in userlogin, and the functions obtained in this function are based on this student number Yes. The switch implementation allows the user to select the function that needs to be executed, and calls out the function reals of other classes in the function Now functional.

**The main technical difficulties and implementation solutions:**

**.csv File handling**: At first, I intended to use .txt documents when choosing how to store data, but when I accidentally looked through the data, I found that I could use Excel to save files through .csv documents. The advantage of .csv file is that it can write the related data or information on the same line, and the related data can be output together at one time. Since .csv documents are rendered when going to C++ ( XXX, YYY , ZZZ , ... , AAA) (the comma represents the next column in excel), so this data cannot be called directly, but by splitting it again and again Make information usable. However, my first difficulty in working with this file was how to add new information to the .csv file repository. At first, I couldn't find a reason why I couldn't successfully add new materials, only to find out later that when we loaded .csv documentation into a vector, he would put a completely blank vector in the last box of the vector That is to say, if we directly use push\_back put the new data last, then there will be a blank data in the middle, and the numbering will become unsmooth, so we need to use the pop\_back to delete the gap in the middle before creating new data push\_back add new material to the database. Then the second difficulty encountered when working with .csv documents is when deleting or modifying materials. As mentioned earlier.csv when entering C++ will have an extra blank line of data at the end, so if we don't pay attention when putting the data back into the .csv file, we will .csv the next input The document will have two blank rows of data. Since our numbering of data is based on the number of rows, the occurrence of these empty data will not only cause compilation errors but also cause errors in the numbering of book codes, user codes, etc. Therefore, after we finish modifying or deleting data, we need to use pop\_back to delete the trailing blank line data to ensure that the program can compile normally.

**Data input errors:** In this program, many data can cause infinite loops or compilation errors after entering errors, including entering letters into int data and entering numbers greater than 2147483647 to int cin>>string does not enter spaces, etc. First of all, it's like int data input letters, which will cause constant garbled characters, so that the program will keep outputting strange numbers, and finally I searched for more than one to find out that we could use if (!( cin >>)) {cin.clear(); cin.ignore(10000, '\n'); } allows the system to ignore garbled characters and ask for re-entry when it sees the wrong output. The second is to enter a number greater than 2147483647 to int, and the closest value in this program is the student number, but it is negligible because the first four digits of the student number are the year of entry (less than 2019). If a number greater than 2147483647 is entered when entering a student number, the system will output garbled characters and count them as if (!( cin >>)) {cin.clear(); cin.ignore(10000, '\n'); } processing. In the problem that you cannot enter spaces when entering cin>>string, many materials such as book titles and author names need to use spaces, so we need to use getline(cin,) when entering However, this also raises other problems, if the cin >> has been executed before using getline(cin,), it will be omitted input for getline(cin,). So I need to initialize the string parameter first and then execute getline(cin,) twice. However, if the front is not cin >> input, two getline(cin,) will require the user to enter data twice. So I chose to use getline(cin,) only once, but I added if (x= ="") to the rear {getline(cin,x);)}, so that because of the front cin, The getline(cin,) (also meaning that there is no input, i.e. x= =") that will be omitted will have the opportunity to enter again, and it will not appear if it has been successfully entered.

**Implementation of fuzzy search:** In this program, the search book title, author name, and user name are all fuzzy searched. The implementation of fuzzy retrieval is to first convert the input string into c har[] storage, and then convert the book title, author or user name in the database into c har[], and use it If (strstr(char2, char1) != NULL) for comparison, output if there are overlapping letters, output if we do not enter any input, output all data in the database, so to prevent this from happening I set up a program that requires you to enter it again if the input is empty.

**User experience:** In order to log in to the user, you need to enter your student number and user password. The presentation of this function also relies on the data arrangement method in the .csv, first after the user enters the student number, the user .csv is entered line by line, knowing that the data (that is, the student number) that can read the second column of a row is consistent with the entered student number. Then we get the fourth column of data in this row, which is the user's user password, and then compare it with the user input, if it is a consistent successful login to the user interface, and enter the student number into the interface at the same time. After the user logs in to the user interface, all functions do not need to enter their own student number, the system has obtained the student number information from the login interface, and automatically enter the student number in all functions. In this programming, I try to pursue the convenience of the user, not the convenience of programming, in the program I can ask the user to enter all the information into the program one by one But in the end, I still reduce the user's input through the program itself, many materials such as book codes, user codes, etc. are automatically generated, and various materials also need to be updated before users need to enter.

**Summary report**

Through this big assignment, I think I have a deeper understanding of programming. From MOOCs and lectures, I can only understand the small role of functions or programming through one small experimental area after another, but when the amount of programming increases, the difficulty of programming and the perception of programming have a different feeling. The programming method I took in this big assignment was to write out one function after another independently and piece it together one by one, and the problem I often encountered was that there was no problem after running independently, but when I put everything together, I found all kinds of bugs, and these bugs were difficult to detect, and could only be modified by modifying various parameters, inputs, and logic Derivation is the only way to find a solution. I feel that I have a deeper understanding of programming after experiencing this big homework, such as various ways of dealing with data input and other problems and the mentality and ideas for dealing with problems, maybe this big homework is just a helpless choice under the epidemic, but I think the practical teaching of big homework is more suitable for learning programming, and I hope to become the norm in future programming courses.