Luna's Library System

Documentation and Code Overview

Your Name

June 12, 2025

Contents

1 Introduction

Luna's Library System is a C++ console application designed to manage a digital library. It allows users to register, log in, borrow and return books, theses, and magazines, and view their borrowing history. The system uses CSV files for persistent storage of users and library items.

2 Project Structure

- `main.cpp` – Main program logic and menu handling
- `bibliofiles.hpp` – Base and derived classes for library items (Book, Thesis, Magazine)
- `user.csv` – Stores user data
- `books.csv`, `thesis.csv`, `mags.csv` – Store library items

3 Main Features

- User registration and login
- Borrowing and returning files
- Viewing file information and fragments
- Persistent storage using CSV files
- User history tracking

4 Class Overview

4.1 BiblioFiles and Derived Classes

```
class BiblioFiles {
protected:
    std::string idfile, title, author, filetype, fragment;
    int publicationyear;
    bool availability;
public:
    // Methods for getting info, showing fragments, etc.
};
```

Listing 1: BiblioFiles Base Class

Book, **Thesis**, and **Magazine** inherit from `BiblioFiles` and add specific fields and methods.

4.2 User Class

```
class User {
protected:
    std::string history;
    std::string name, password;
    std::string borrowedfiles;
public:
    void borrowfile(BiblioFiles* file);
    void returnfile(BiblioFiles* file);
    // Other user-related methods
};
```

Listing 2: User Class

5 Program Flow

5.1 Startup

1. Loads user data from `user.csv`.
2. Displays the main menu: Login, Register, Exit.

5.2 Login and Registration

- On login, verifies username and password.
- On registration, checks for unique username and appends new user to `user.csv`.

5.3 Library Data Loading

After successful login, the program loads library data from `books.csv`, `thesis.csv`, and `mags.csv`.

5.4 Main Application Loop

1. Shows user menu: View files, Search, Borrow, Return, History, User Settings, Exit.
2. Handles user actions with nested menus and input validation.
3. Updates user and library data in memory and in CSV files.

6 File Operations

- Reading: Uses `std::ifstream` and `std::getline` to load data.
- Writing: Uses `std::ofstream` (with `std::ios::app` for appending or `std::ios::trunc` for overwriting).
- Updates: Reads all lines, modifies in memory, then rewrites the file.

7 Example: Borrowing a File

```
void User::borrowfile(BiblioFiles* file) {
    if (!history.empty()) history += ";";
    history += file->getidfile();
    // Update user.csv with new borrowed file and history
}
```

8 Error Handling

- The program checks for file open errors and invalid data.
- On fatal errors (e.g., missing CSV files), the program prints an error and exits.
- Input is validated at each menu to prevent invalid actions.

9 Data Consistency

The system reads the entire CSV file into memory, updates the relevant records, and rewrites the file. This approach ensures that changes are atomic from the user's perspective, but may not be safe for concurrent access by multiple users or processes. For multi-user environments, consider using a database or file-locking mechanisms.

10 Extending the Program

To add new features (e.g., new file types or user roles), create new classes inheriting from `BiblioFiles` or `User`, and update the menu logic in `main.cpp`.

11 Menu Navigation and User Experience

The program uses a text-based menu system. After launching, users are presented with options to log in, register, or exit. Once logged in, users can navigate through nested menus to view, borrow, or return files, and access their history or settings. Each menu validates input and provides feedback for invalid choices, ensuring a robust and user-friendly experience.

12 User Manual

12.1 Starting the Program

To start Luna's Library System, compile and run the C++ program. The main menu will appear in your terminal.

12.2 Main Menu Options

- **[L] Login**: Enter your username and password to access your account.
- **[R] Register**: Create a new user account by providing a unique username and password.
- **[E] Exit**: Close the program.

12.3 After Login: User Menu

Once logged in, you will see a menu with options such as:

- **[V] View Files**: Browse books, theses, or magazines. Select an item to see details or perform actions.
- **[L] Search**: Search for files by title, author, or other criteria (if implemented).
- **[B] Borrow**: Borrow a file by entering its ID.
- **[R] Return**: Return a borrowed file by entering its ID.
- **[S] History**: View your borrowing history.
- **[U] User Settings**: Change your username or delete your account.
- **[E] Exit**: Log out and return to the main menu.

12.4 Borrowing and Returning Files

- To borrow a file, navigate to the desired item and select the borrow option. The file will be added to your borrowed list and history.
- To return a file, select the return option and specify the file ID.

12.5 User Settings

In the user settings menu, you can:

- Change your username.
- Delete your account (this action cannot be undone).
- Return to the main menu.

12.6 Input Guidelines

- Enter the letter corresponding to your menu choice (e.g., `L` for Login).
- When prompted for IDs or text, type the required information and press Enter.
- If you enter an invalid option, the program will prompt you to try again.

12.7 Exiting the Program

You can exit the program at any time by selecting the `E` option in the current menu.

13 Conclusion

This system demonstrates a modular approach to C++ application design, using classes, file I/O, and menu-driven user interaction. For further improvements, consider adding unit tests, using a database for storage, or implementing a graphical interface.

14 CSV File Formats

14.1 User File (`user.csv`)

Each line represents a user:

```
username,password,borrowedfiles,history
```

14.2 Book/Thesis/Magazine Files

Each line represents a library item, with fields such as ID, title, author, year, and type-specific attributes. For example:

```
B001,The C++ Programming Language,Bjarne Stroustrup,2013,book,...
```

15 Sample Menu Output

```
+-----------------------------------------+
| Luna's Library System                    |
+-----------------------------------------+
| [L] Login                                |
| [R] Register                             |
```

```
| [E] Exit                                    |
+---------------------------------------------+
:
```

16 Limitations and Future Work

- The system does not support concurrent users.
- Passwords are stored in plain text for simplicity.
- No advanced search or filtering features.
- Future improvements could include password hashing, a graphical interface, or migration to a database backend.

17 Developer Guide: Modifying the Code and CSV Structure

17.1 Modifying Header Files or `main.cpp`

If you wish to extend or change the functionality of Luna's Library System, you will likely need to modify the header files (such as `bibliofiles.hpp`) or the main program file (`main.cpp`). Here are some tips:

- **Adding a New File Type:**
  Create a new class (e.g., `class Newspaper`) that inherits from `BiblioFiles`. Implement any new attributes or methods specific to your file type. Update the logic in `main.cpp` to recognize and handle this new type, including reading from and writing to a new CSV file if needed.
- **Changing User or File Attributes:**
  If you add or remove attributes (fields) in the `User` or `BiblioFiles` classes, make sure to update all code that reads from or writes to the corresponding CSV files. Adjust the parsing logic (e.g., `parseCSVLine`) and the code that constructs objects from CSV data.
- **Updating Menus:**
  If you add new features, update the menu functions (such as `showmenu()`, `showfilemenu()`, etc.) to include new options, and handle the new user input in the main application loop.

17.2 Adding or Modifying Text in CSV Files

- **Adding a New Column:**
  If you want to add a new field (e.g., "email" for users), update the CSV header and all user records. Then, update the code that reads and writes user data to handle the new field. This includes modifying the constructor and any functions that parse or output user data.

- **Changing the CSV Format:**
  Always keep the order of fields consistent between the code and the CSV files. If you change the order or add/remove fields, update both the CSV files and the code that reads/writes them.

- **Adding New Records Manually:**
  You can add new users or files directly in the CSV files using a text editor. Make sure each field is separated by a comma, and that the number and order of fields matches what the program expects.

- **Handling Special Characters:**
  Avoid using commas within fields, as this may break the CSV parsing logic. If you need to store text with commas, consider updating the parsing logic to handle quoted fields.

17.3 Best Practices

- Always back up your CSV files before making bulk changes.
- Test your changes with a few records before applying them to the entire dataset.
- If you add new features, update the documentation and user manual accordingly.

References

[1] 2024. C++ reference. Accessed: 2024-06-12. https://en.cppreference.com/w/.

[2] 2024. Cplusplus.com. Accessed: 2024-06-12. https://www.cplusplus.com/.

[3] 2024. The LaTeX project. Accessed: 2024-06-12. https://www.latex-project.org/.