CCD - Checklist

1. Use logical names for variables, functions, and classes. The purpose and aim of the code should be clearly stated in the names.

2. Divide your code into more manageable, modular classes or functions. To make the code easier to maintain and comprehend, each module should have a single purpose.

3. Use comments sparingly and concentrate on elucidating the purpose of the code rather than just what it is doing. Although well-written code ought to be self-explanatory, comments might give further background information.

4. Maintain a unified formatting and coding style across your whole codebase. This covers name conventions, space, and indentation. Uniformity enhances the readability.

5. Put in place appropriate error handling to make sure your system responds to unforeseen circumstances politely. Give helpful error messages to facilitate troubleshooting.

6. Verify user input to avoid unforeseen actions or security flaws. Make sure the system can elegantly handle a range of inputs.

7. Create unit tests for your library management system's essential parts. Automated tests facilitate future refactoring by assisting in ensuring that your code operates as intended.

8. Especially for intricate logic or algorithms, document your code. Understanding and maintaining the system is made simpler for both you and other users with a well-documented codebase.

9. Follow standardized naming guidelines for classes, variables, and functions. By doing this, a standard is established that developers may adhere to when reading and editing the code.

10. To keep track of code modifications, use version control systems (like Git). Make frequent commits with insightful commit messages. This aids in monitoring the development of the