LIBRARY SYSTEM

CST2550 course work 1



M00579249

### Introduction:

For this project, I worked on developing a library management system using C++. The main aim was to create a software solution that simulates typical library operations, including managing books, members, issuing and returning books, and calculating fines for overdue books. The project employed object-oriented principles, with different classes representing various aspects of the library system.

Software Design:

### Class Diagram:

A diagram of a computer

Description automatically generated

I created a class diagram to visually represent the structure of the software. It highlights the relationships and attributes of classes, providing a comprehensive view of the system.

### Use Case Diagram:

A diagram of a library

Description automatically generated

The use case diagram illustrates the interactions between the Liberian and the library system. It helped in understanding how users (actors) would interact with the software.

**Activity Diagrams:**

A diagram of a library

Description automatically generatedA diagram of a member options

Description automatically generatedA diagram of a group of people

Description automatically generated

To break down the workflow for different functionalities, I created activity diagrams. These diagrams detailed the step-by-step processes for loading books, adding a member, issuing and returning books, displaying borrowed books, and calculating fines.

Software Testing:

Testing Approach:

I employed a testing approach that included both unit testing and integration testing. Unit tests focused on individual functions, ensuring their correctness, while integration tests checked how different components collaborated.

Evidence of Testing:

I documented various test cases and their outcomes. This evidence was crucial for tracking the reliability of the system and confirming that all functionalities were working as expected.

Implementation:

Approach:

In terms of implementation, I followed object-oriented principles, such as encapsulation and polymorphism. The Makefile was used to automate the build process, making compilation and execution smoother. Git was the version control tool, helping in managing code changes and facilitating collaboration.

Makefile and Version Control:

The Makefile contained rules for compiling the code and managing dependencies. Git allowed for version control, with branches for features and bug fixes. This approach made it easier to collaborate and revert to previous versions if needed.

Conclusion:

Summary of Work Done:

The project successfully implemented a functional library management system. It covered key functionalities and was designed with modularity and maintainability in mind.

Limitations and Critical Reflection:

One limitation was the use of raw pointers, which might lead to memory management issues. In future projects, I would consider smart pointers for improved memory safety. Additionally, a more robust error-handling mechanism could enhance the software's resilience.

Future Approach:

For similar tasks in the future, I plan to implement more thorough testing, including edge cases. I also aim to explore design patterns and further decouple components to enhance scalability and maintainability.

The project provided valuable insights into software development, testing, and version control. It was a learning experience that has inspired me to continuously improve my approach in future projects.