CST2550
Library Management system Project

Presented by M00737611

The overview

- Introduction to the project
- Discussing the design of the management system
- All things implementation
- Testing approach chosen
- Conclusion

Project brief

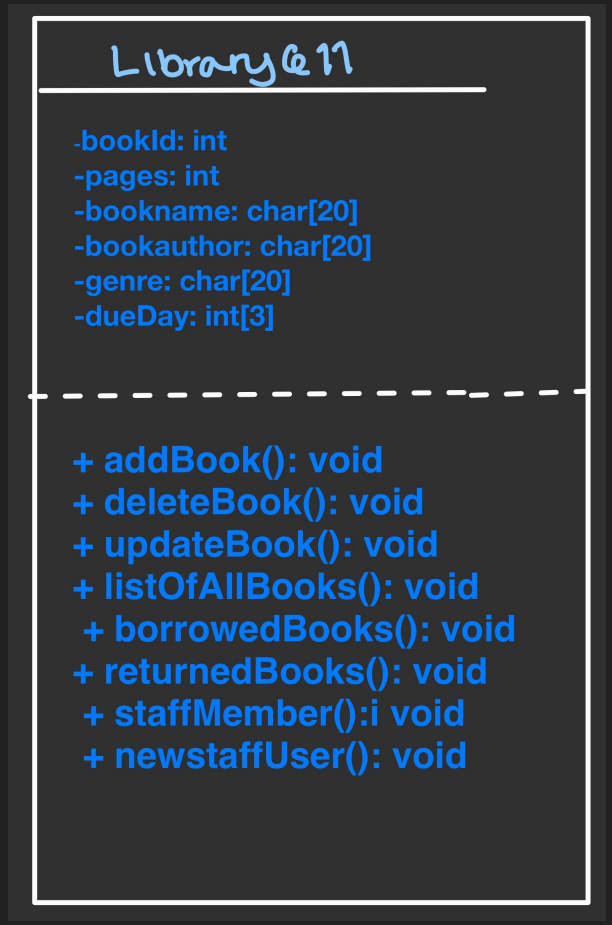
The Library Management System in request is a software application developed in C++ to efficiently manage the day-to-day operations of a small library. The system encompasses essential functions such as adding books, issuing books to members, adding new members, displaying all available books, and calculating fines for overdue books. It caters to a diverse collection of books across various genres.

Project brief
Key requirements for LMS

- **Add Book:**
  - Functionality to input and store information about new books.
  - Details include book ID, title, author, genre, and page count.
- **Issue Book to Member:**
  - Capability to assign a book to a library member.
  - Records the member's details, issue date and due date.
- **Add Member:**
  - Ability to register new library staff members , they are required to make a username and password limited by 15 characters each
- **Display All Books:**
  - A comprehensive list of all available books in the library.
  - Organized by genres for easy navigation.
- **Calculate Fine:**
  - Automated fine calculation for overdue books.
  - Using the due date to calculate how many days over the borrower is. It’s a £1 a day over the due date fine

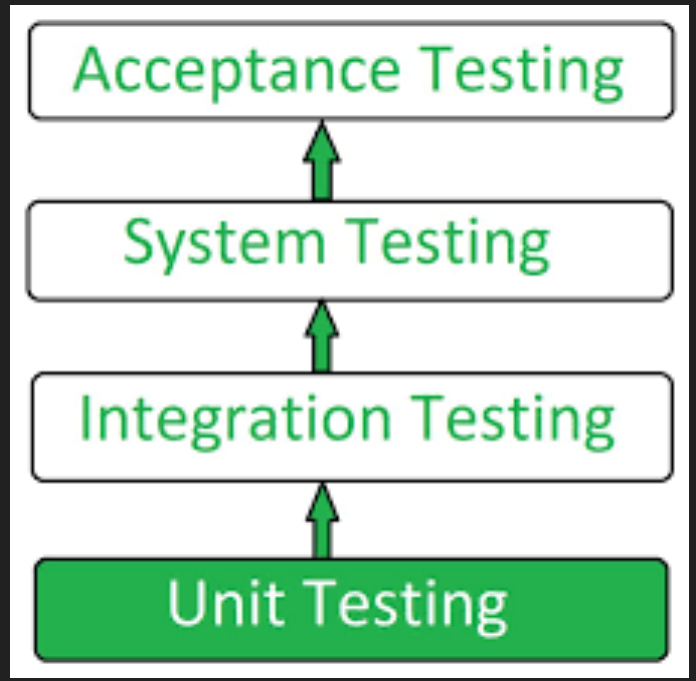
My software's current abilities

UML Diagram



1. **User Interaction**: The application begins by presenting a menu to the user, who can choose to log in as a staff member, add a new staff member, or exit.
2. **Authentication**: If a user chooses to log in, they are prompted for a username and password. If the credentials match predefined values, the user is logged in.
3. **Staff Functionality**: Once logged in, the user can add, update, delete, or list all books. This functionality is not fully implemented yet, but there are placeholder functions ready for you to fill in.
4. **Book Management**: In the addBook()function, the user can enter details for a new book, which are then saved to a file. The filename is based on the book's ID.

Ideal Testing Approach



Unit testing is a method where individual parts of the code are tested to determine if they work as expected. In the context of the library management system, here's how unit testing would be applied :

1. **addBook() Function**: A test could be written that calls addBook() with a specific set of inputs, then checks if the new book was correctly added to your system (e.g., a new file was created with the correct content).
2. **deleteBook() Function**: Similarly, a test for deleteBook() could try deleting a book and then check if it was correctly removed from your system.
3. **updateBook() Function**: A test for updateBook() might involve updating a book's details, then checking if the changes were correctly saved.
4. **listOfAllBooks() Function**: A test for this function could check if the returned list of books matches the expected list.
5. **Login Functionality**: writing tests to check if the login functionality works correctly. For example, a test might try logging in with correct credentials and expect a successful login message, while another test might try incorrect credentials and expect an error message.

Testing Approach

I took on the system testing approach under the waterfall development methodology. Instead of taking on an agile approach and testing each unit once complete I tested the project at the very end. This can have a few advantages like : it simplicity in being easy to understand and its ideal for smaller projects like this one.

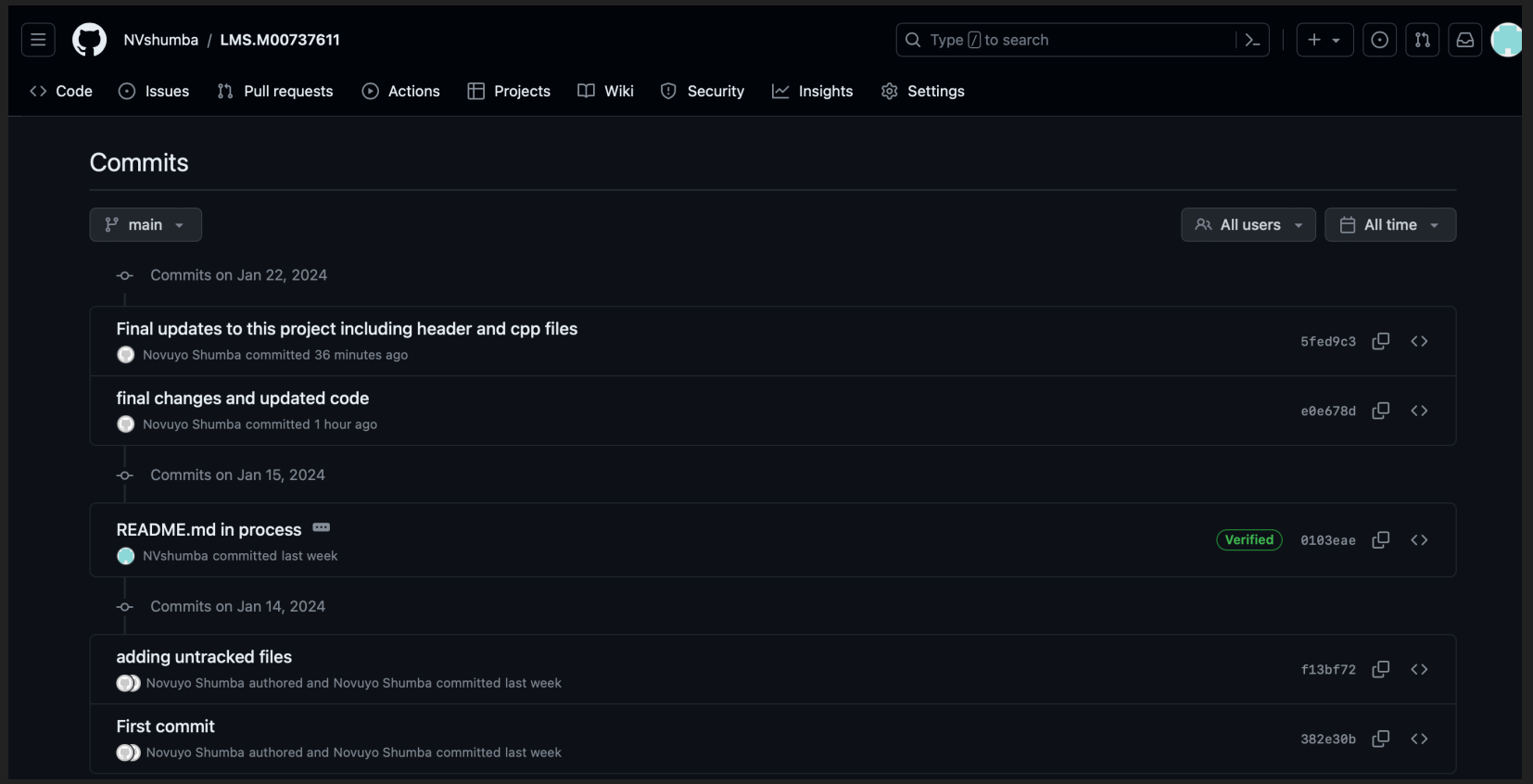
And disadvantages like: its difficult to make changes on previous stages of the project once completed and the late testing of the system can lead to issues being discovered at a later more crutial stage of the project

Implementation

- My original approach to this project began with me doing research on what an ideal project like this would function like. I then moved on to reading over the specifications and did my best to translate that into the code I have presented here.
- I next went into bringing the project idea into life and applied all that I had learnt during the research part throughout the project.
- Version control in this project was important, the basic purpose of it is to be able to save new changes of files whilst also being able to view previous versions . In this project it helps track progress and in case of any mistake I can return back to older versions and see what could be the cause of current errors. On a industry level I can understand how learning a skill like this could majorly help as a developer working as a part of a team. For the benefits such as isolated collaborative work (all working on the same project individually) it helps with testing and integration of work and is a transparent and controllable way to manage the project

Git Commits

These are the commits pushed over the course of me working on the project. In retrospect I could've made a lot more commits for this project as I was working on it daily but because I didn't push new changes to my repository as often as I made had changes, I removed the opportunity to use version control systems like this to its advantages.



Software Demonstration

In Conclusion

How would I approach a similar project :

I would most likely seek to apply a more industry perspective to it , considering elements like formal process , process engineering and quality control

I believe my approach could have been clearer following each process in order and completing it before moving onto the next .

Limitations :

Time management, not testing as I go along and minimal use of github inline with the specification