**Library Borrowing System Final Document**

**1. Use Cases**

A diagram of a user

Description automatically generated

**2. Fully-Dressed Use Cases**

**Use Case: Borrow Book**

**Actors:** Library Member, System

**Preconditions:** User must be logged in; book must be available in the catalog.

**Main Flow:**

1. Member selects a book to borrow.
2. System checks availability.
3. Member confirms borrowing.
4. System records transaction and updates book status to "borrowed."

**Postconditions:** Transaction is logged; book status updated.

**Alternate Flows:**

* If the book is unavailable, prompt user to reserve.

**3. Use Case Diagram**

*(Placeholder for Use Case Diagram)*

**4. Vision**

To create an efficient, user-friendly, and secure system for managing book borrowing and returning in libraries, supporting scalability and seamless user interaction.

**5. Supplementary Specs Document**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Description** | **Author** |
| **1.0** | **Dec 30, 2024** | **Initial draft based on system requirements.** | **Haneen ELGhawy** |

**Introduction**

**This document is the repository of all library borrowing system requirements not captured in the use cases.**

**Functionality**

**Logging and Error Handling:**

* **Log all errors to persistent storage.**

**Pluggable Rules:**

* **Various rules for borrowing limits and overdue fines can be configured based on user types or library policies.**

**Security:**

* **All usage requires user authentication.**

**Usability**

**Human Factors:**

* **Text should be easily visible from 1 meter.**
* **Avoid color schemes that could affect users with color blindness.**

**Reliability**

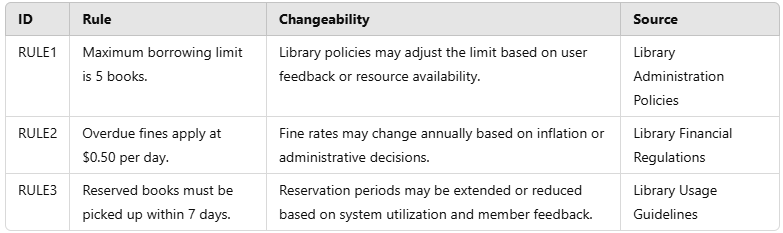
**Recoverability:**

* **If there is a failure in external services , provide a local fallback solution to ensure minimal disruption.**

**Performance**

* **System response time should be under 2 seconds for catalog searches and borrowing transactions.**

**6. Business/Domain Rules**

****

**7. Glossary**

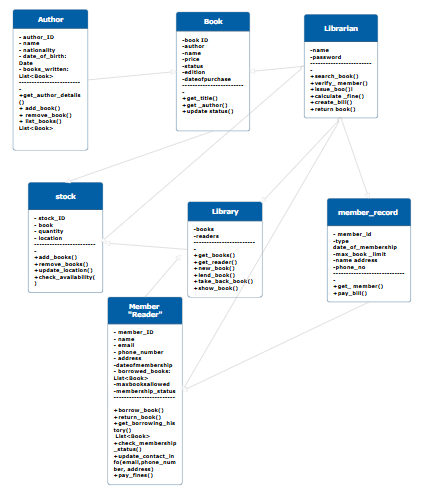
**Definitions**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Term | Definition and Information | Format | |  |  |  | | --- | --- | --- | | **Validation Rules | Aliases** |  |  |  |  | | --- | |  | |
|  | A registered user with borrowing privileges. |  | | |
| Member | A registered user with borrowing privileges. | Alphanumeric ID | Unique ID for each member Library User |
| Overdue | A book not returned within the allowed period. | Date | Must exceed the due date. Late Book |
| Transaction | Record of borrowing or returning a book. | Transaction ID | Unique record per borrow/return. Borrowing Record |

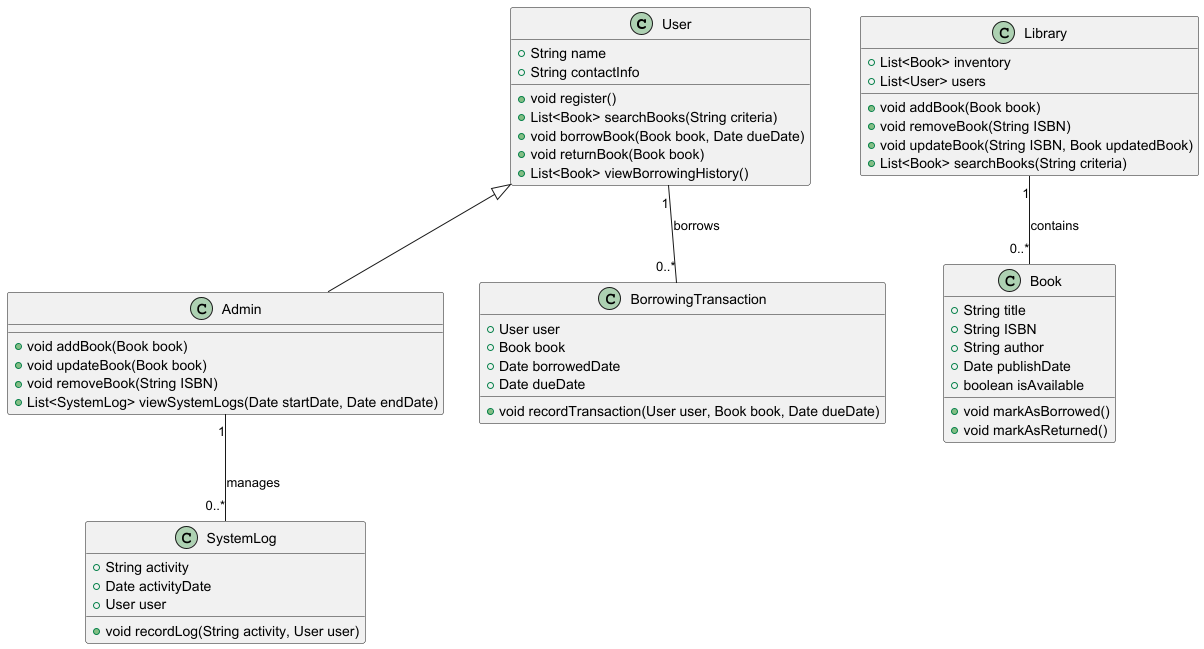
**8. Logical Architecture**

* **Presentation Layer:** GUI for users and Admin.
* **Business Logic Layer:** Core functionality like transactions and Book Browsing.
* **Data Layer:** Relational database for storage.

**9. Domain Model**

*(*

**10. Design Class Diagrams**

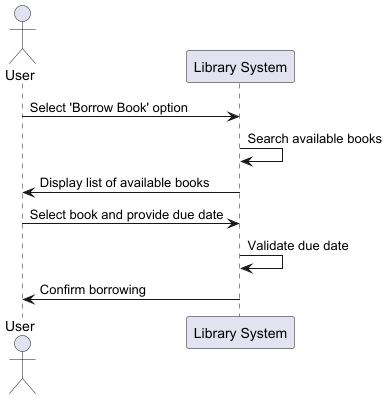
**11. Sequence Diagrams**

***Search book:***

A diagram of a library system

Description automatically generated

***Borrow book:***



**12. Activity Diagrams**

*(Placeholder for Activity Diagrams)*

**13. State Machine Diagrams**

*(Placeholder for State Machine Diagram, focusing on UI Navigation)*

**14. GUI Snapshots**

*(Placeholder for GUI mockups/screenshots)*

**15. Design Patterns**

1. **Singleton:** For database connection.
2. **Observer:** For notifications on book availability.
3. **Factory:** For user role creation (e.g., staff, member).

**Needdsss changing**

**16. Summary Sheet**

**Contributions:**

* **[Ahmed Khaled]:** Worked on Use Cases, Vision, Supplementary Specs, Designed Diagrams. (15 hours).
* **[Yehia Hesham]:** Code Frontend & Backend ,Repo on GitHub, ReadMe file (20 hours).
* **[Haneen ElGhawy]:** Implemented UI, Design Document, Mockup (15 hours).

**17. Lessons Learnt**

**Key Takeaways:**

* Importance of clear communication in a team project.
* Benefits of modular design for scalability.
* Challenges of integrating different components and resolving dependencies.
* Enhanced understanding of UML and design patterns.