Project Title: Advanced Library Management System

Project Description: You have been assigned to develop an advanced Library Management System using .NET and C#. The system should provide enhanced features for librarians and patrons, including book reservations, fine calculation, and advanced search capabilities. This project will challenge your skills in object-oriented programming, data manipulation, file handling, and advanced concepts.

Project Requirements:

1. Data Model:
   * Create classes for the following entities:
     + Book: Represents a book in the library. Include properties such as BookId, Title, Author, ISBN, Genre, Availability, and ReservedBy.
     + Patron: Represents a library patron. Include properties such as PatronId, Name, Address, Email, and Phone.
     + Library: Represents the library itself and contains collections of books and patrons. Implement methods to add, remove, and retrieve books and patrons.
     + Loan: Represents a book loan transaction between a patron and the library. Include properties such as LoanId, Book, Patron, LoanDate, DueDate, and ReturnDate.
     + Fine: Represents a fine incurred by a patron for late return of books. Include properties such as FineId, Patron, Book, FineAmount, and PaymentStatus.
2. Menu Interface:
   * Design an interactive menu-driven interface for librarians and patrons to perform various operations.
   * The menu should include options for the following operations: a) Add a book b) Add a patron c) Borrow a book d) Return a book e) Reserve a book f) Pay fine g) Search for a book by title, author, or genre h) Search for a patron by name or ID i) Generate overdue book report j) Generate fine report k) Exit the program
   * Implement error handling for invalid menu choices and input validation for user interactions.
3. Book Management:
   * Implement functionality to add new books to the library, including input validation for book details.
   * Allow librarians to search for books by title, author, or genre, providing relevant search results.
   * Enable librarians to mark a book as borrowed, returned, or reserved, updating its availability and reservation status accordingly.
4. Patron Management:
   * Implement functionality to add new patrons to the library, including input validation for patron details.
   * Allow librarians to search for patrons by name or ID, providing relevant search results.
5. Loan Management:
   * Implement functionality to handle book borrowing and returning operations.
   * When a book is borrowed, create a loan record with the associated book, patron, loan date, and due date.
   * When a book is returned, update the loan record accordingly and mark the book as available.
   * Provide appropriate error handling for invalid borrowing or returning actions.
   * Calculate and update fines for late returns based on a predefined fine calculation formula.
6. Book Reservation:
   * Allow patrons to reserve a book that is currently unavailable.
   * Implement functionality to manage book reservations, including tracking the patron who reserved the book.
   * When a reserved book becomes available, notify the patron who made the reservation.
7. Fine Management:
   * Calculate and track fines for patrons who fail to return books on time.
   * Implement functionality for patrons to pay fines, updating the payment status accordingly.
   * Generate a fine report displaying unpaid fines and total fines collected.
8. File Operations:
   * Implement functionality to read and write book, patron, loan, and fine data from/to files.
   * On program startup, load existing data from files into the library management system.
   * When the program exits, save any changes made to the data back to the files.
9. Git Integration:
   * Set up a Git repository for the project.
   * Utilize branching and merging to manage features, bug fixes, and refinements.
   * Demonstrate your understanding of version control by maintaining a clean and organized commit history.
10. Code Quality and Best Practices:
    * Write clean, modular, and well-documented code following naming conventions and best practices.
    * Implement appropriate exception handling and error checking.
    * Utilize design patterns and advanced object-oriented programming techniques where applicable.
    * Implement efficient algorithms and data structures for optimal performance.
11. Testing and Validation:
    * Write unit tests to validate the functionality of your code.
    * Conduct thorough testing, including boundary cases and edge scenarios.
    * Ensure all requirements are met and the system works as expected.
12. Documentation:
    * Provide clear and comprehensive documentation explaining the purpose, functionality, and usage of the system.
    * Include instructions on how to build, configure, and run the program.

Note: This project challenges you to implement advanced features such as book reservations, fine calculations, and advanced search capabilities. It does not include web-related technologies, databases, or advanced topics such as ASP.NET.