Daily Coding Assignment Wipro NGA .NET Cohort

Coding Assignment: Building a Simple Library Management System

Duration: 90 minutes

Objective: Implement a basic library management system that includes functionality for adding, viewing, and managing books and borrowers. Write unit tests to ensure the system works correctly.

Problem Statement

You are tasked with developing a simple library management system that supports the following operations:

1. **Add a New Book**: Users should be able to add new books to the library with details such as title, author, and ISBN number.
2. **Register a Borrower**: Users should be able to register new borrowers with their name and library card number.
3. **Borrow a Book**: Borrowers should be able to borrow a book from the library.
4. **Return a Book**: Borrowers should be able to return a book to the library.
5. **View Books and Borrowers**: Users should be able to view a list of all books and borrowers.

User Stories and Expectations

User Story 1: Add a New Book

- **As a user**, I want to be able to add a new book to the library.
- **Acceptance Criteria**:
    - The system should accept a book's title, author, and ISBN number.
    - The system should store the book's details in a list.
    - The system should confirm that the book has been added successfully.

User Story 2: Register a Borrower

- **As a user**, I want to be able to register a new borrower.
- **Acceptance Criteria**:
    - The system should accept the borrower's name and library card number.
    - The system should store the borrower's details in a list.
    - The system should confirm that the borrower has been registered successfully.

User Story 3: Borrow a Book

- **As a borrower**, I want to be able to borrow a book from the library.
- **Acceptance Criteria**:
  - The system should allow a borrower to select a book from the library.
  - The system should mark the book as borrowed.
  - The system should associate the borrowed book with the borrower's details.

User Story 4: Return a Book

- **As a borrower**, I want to be able to return a borrowed book to the library.
- **Acceptance Criteria**:
  - The system should allow a borrower to return a book.
  - The system should mark the book as available for borrowing again.
  - The system should update the borrower's record to reflect the returned book.

User Story 5: View Books and Borrowers

- **As a user**, I want to be able to view all books and borrowers.
- **Acceptance Criteria**:
  - The system should provide a list of all books in the library, showing their details and current status (available/borrowed).
  - The system should provide a list of all borrowers, showing their details and borrowed books (if any).

Coding Assignment

Task 1: Implement Classes

Create the following classes:

1. `Book`
   - Properties: `Title`, `Author`, `ISBN`, `IsBorrowed`
   - Methods: `Borrow()`, `Return()`
2. `Borrower`
   - Properties: `Name`, `LibraryCardNumber`, `BorrowedBooks` (a list of books)
   - Methods: `BorrowBook(Book book)`, `ReturnBook(Book book)`
3. `Library`
   - Properties: `Books` (a list of books), `Borrowers` (a list of borrowers)
   - Methods: `AddBook(Book book)`, `RegisterBorrower(Borrower borrower)`, `BorrowBook(string isbn, string libraryCardNumber)`,

`ReturnBook(string isbn, string libraryCardNumber)`, `ViewBooks()`, `ViewBorrowers()`

Task 2: Write Unit Tests

Write unit tests for the following scenarios using NUnit or MSTest:

1. **Adding a Book**:
    - Test that a book is added correctly to the library.
2. **Registering a Borrower**:
    - Test that a borrower is registered correctly.
3. **Borrowing a Book**:
    - Test that a book can be borrowed and is marked as borrowed.
    - Test that the book is correctly associated with the borrower.
4. **Returning a Book**:
    - Test that a book can be returned and is marked as available.
    - Test that the book is correctly removed from the borrower's list.
5. **Viewing Books and Borrowers**:
    - Test that the list of books and borrowers is displayed correctly.

Expectations

- **Code Quality**: Your code should be clean, well-organized, and follow best practices.
- **Unit Tests**: Ensure that all tests are written to cover the scenarios described and are passing.
- **Documentation**: Include comments and documentation where necessary to explain your code and tests.
- **Submission**: Submit your implementation of the classes and the unit tests as a single project.

This assignment will assess your ability to design a basic system, implement functionality, and write comprehensive unit tests.