Console Based Library Management System in C# with SQL Server and Entity Framework

Jakub Zawadzki – M00981994

1. Introduction and description of the project.

The Library Management System is console based application developed in C# using .NET 8 and Entity Framework Core. It enables librarians to manage resources such as books, magazines or journals, by offering CRUD operations, as well as borrowing and returning of library resources.

This application was developed to simulate real world operations of a small library. The console interface provides simplified user experience, while Entity Framework provides effective interaction with SQL Server.

Additionally the project serves as a modular template for expanding its functionality such as user management, and supports future scalability.

2. Justification of selected data structures and algorithms.

Data Structures

Entity Classes (Resource) – Used to model real world library item. Each resource contains key attributes such as Title, Genre and Resource Type. It allows for encapsulation and easy database mapping.

Lists – Used List<T> in conjunction with LINQ to filter and manipulate collections of Resource objects. LINQ provided efficiency while querying collections. The list structure was chosen for simplicity and ease of iteration, which suits our case with relatively small datasets.

With future expansion of Management System I would introduce more complex features such as user accounts, resource reservation and that would require wider variety of data structures e.g. Queue<T> for implementation of resource reservation, Graph for recommendation system or Stack<T> for undo operation.

Algorithms

Search by title, author, genre

Linear search with LINQ .Where() – easy to implement and understand algorithm that works well with small datasets

Borrowing

Check IsAvailable before assigning BorrowedBy and DueDate. I chose it this simple state checking logic to prevent double borrowing (queue not yet implemented) with quick O(1) updates to single record.

3. Time complexity analysis

Search by field (title, author, genre)

```
INPUT: keyword
FOR each resource IN resourceList
        IF resource.field CONTAINS keyword
                ADD resource TO results
END FOR
```

Linear search across all resources

Best Case: O(1) – Match found on first resource

Worst Case: O(n) – No match

Add new resource

```
INPUT: resourceDetails
CREATE newResource WITH resourceDetails
ADD newResource TO database
SAVE
```

Always inserts a single item with constant performance

Time Complexity: O(1)

Generate overdue report

```
GET currentDate
FOR each resource IN resourceList
        IF resourceHasDueDate AND dueDate < currentDate
                ADD to overdueList
DISPLAY overdueList
```

Checks each record regardless of outcome

Time Complexity: O(n) – no overdue items or all items overdue

4. Testing approach and test cases

Testing for Library Management System prioritised unit tests written in xUnit and EF Core InMemory to simulate database operations. It allows to test the behaviour of application without affecting actual SQL database.
I implemented modular testing which means each unit (resource creation, borrowing) was tested independently.

Example Test cases:

AddResource_ShouldAddResourceToDatabase

Case: Add a new resource to the database
Purpose: Verify that a resource can be created and persisted in
Result: Resource is stored and matches the input

UpdateResource_ShouldModifyExistingResource

Case: Modify existing resource
Purpose: Confirm updates are saved correctly
Result: Resource matches updated information after saving

SearchResource_ByTitle_ShouldReturnCorrectResource

Case: Search by a partial match
Purpose: Ensure that search logic returns accurate results
Results: One or more resources with matching input are found

DeleteResource_ShouldRemoveFromDatabase

Case: Delete resource from database
Purpose: Confirm the resource is no longer present in database
Result: context.Resources doesn't contain deleted item

BorrowResource_ShouldSetDueDate

Case: Simulate borrowing a resource
Purpose: Test that borrowing sets IsAvailable to false and sets DueDate
Result: Resource is unavailable to borrow

BorrowNonexistentResource_ShouldFail

Case: Attempt to borrow a non-existent resource
Purpose: Confirm that borrowing fails if resource isn't found
Result: No resource returned and operation skipped

This testing strategy provides high confidence in core functionality. By covering typical and edge cases and using in-memory database the tests remain reliable and easy to maintain

5. Conclusions with Limitations and Reflection

The Library Management System demonstrates how a structured console application can be developed using C#, Entity Framework and SQL Server.
It supports essential library operations, while ensuring data validation. The use of LINQ contributes to clean and maintainable code and automated testing via xUnit makes critical functionalities verifiable and reliable.

This project also highlights the value of step-by-step design. Starting with simple features and gradually adding in validation, searching and tests allowed each future to be developed and improved independently making entire project more manageable.

Limitations
While functional, the system has key limitations, as mentioned in Section 2

- No access roles, which restricts the system to single user operations without differentiated permissions.
- Console interface, which limits the user experience and accessibility compared to GUI or web based application.
- Simplified data model, with no support for more sophisticated features such as reservations or advanced categorisation.
- Static seeding, with initial data being hardcoded and not dynamically imported
- Single user due to lack of concurrency

With all its limitations the system serves as a working tool and a blueprint for further development in the future.

6. References

- Microsoft (2024) *Entity Framework Core Documentation*. Available at: https://learn.microsoft.com/ef/
- Microsoft (2024) *SQL Server Documentation*. Available at: https://learn.microsoft.com/sql/
- xUnit.net (2024) *xUnit Documentation*. Available at: https://xunit.net
- Albahari, J. (2022) *C# 10 in a Nutshell*. O'Reilly Media.
- Stack Overflow (2024) *Entity Framework Core – Questions and Answers*. Available at: https://stackoverflow.com/questions/tagged/entity-framework-core