.NET Core Assignment

Installation, Development, Deployment

13 Mar 2025
Version 0.1

Prepared For:
Linkers

Hashedin

Hashedin University 25.02.1

1. Objective

PYM Technologies has collaborated with HashedIn University to build a platform to build an online books library. They are looking for a centralized system that can be accessed through APIs. This document describes the business context of the application that needs to be developed for them.

1.1 Business Context

Mr. Hank PYM, President and CEO of PYM Technologies thought of maintaining a centralized book library for books in PYM Technologies. As a result, PYM Technologies started project WordWell. They have developed Frontend for WordWell. As it is affecting their allocations they contacted HashedIn University to come up with a platform that helps their Frontend through APIs.

2. Project Plan

Mr. PYM has a critical presentation on Thursday **Evening at 5.00 pm**. We have until Wednesday evening to complete all the deliverables. The delivery is expected to be high quality, iterative and showing progressive enhancements without waiting for a last-minute integration. This will reduce the risk and keep Mr. PYM informed about the progress schedule instead of a last-minute surprise. There are daily checkpoints twice a day to keep up the progress.

HashedIn technical team has estimated the project to be doable by an experienced engineer in approximately 20 hrs. Given that HashedIn University team is newly introduced to .Net Framework, they are expected to take slightly longer. It's expected that they will heavily prioritize doing the important stuff to make the presentation on Thursday successful.

The milestones below show the must-have features and how they should be planned for delivery. The remaining time should be used to complete the functionality and prioritized based on the business context.

3. Functional Requirements

Using the application, the user should be able to signup and login. The user should be able to create, delete, update/review books based on the roles.

3.1 Books

Every book has a unique id(ISBN), description, author, title, genre, publication year, publisher.

Id – Unique alphanumeric identifier
Description – A summary of the content.
Author – Writer of the book.
Tittle – Name of the book
Genre – Predefined classification of the book.
Publisher – Company who published the book.
Publication year – Year when it was published.

Tasks:
We should be able to

a. Get a list of books (by author, by publisher).
b. Get details of the book (Including review, rating and description).
c. Create a book.
d. Update a book.
e. Delete a book.

3.2 Genre

Books can be tagged to a genre. One book can have more than one genre. CRUD on Genre can be created from Admin for now. There should be a way to tag and un-tag genre to and from the book.

Tasks:

We should be able to

a. Add a genre to the book.
b. Delete a genre from the book.

Book APIs should be able to accept/update the genre.

3.3 Review and Ratings

Users can add a review and rating to the book. A user can add review on any book but they can edit or delete only their review.

Tasks:

We should be able to

a. Add a review of a book.
b. Update a review as described above.
c. Delete a review as described above.
d. Add a rating (x out of 5; 1 < = x < = 5)

3.4 Roles and Permissions

There are 3 roles in the application and the users are permitted to perform above operations based on the roles they have.
Roles are.

1. Admin - Users with Admin Role have access to Create, Edit, and Delete Book + All the access of Author + All the access of standard users.
2. Author - Users with Author Role have access to Add, Edit, and Delete their books + All the access of standard users.
3. Standard User – User with this role can add a review or rate any book, also such users can find the added books, however they can't add a new book.

.

The Roles are explicit, and one User can have more than one Role. The Author is not bound to a single book. Allocating and de-allocating roles can be done through Admin.

Tasks:

Only users with Admin role should allowed to

- Create, Update and Delete Books by any Author.

Only users with Author should be allowed to
   a. Create, Delete, Edit their books.

3.5 Search Query Language for books.

Build a query language to search the issue. This should support AND and OR operators for now. There will not be any nesting.

The API should accept a query parameter as given below. dsql stands for DropShip Query Language

1. /books/?dsql= ***id=1 OR author=" Robert Peterson"***
2. /books /?dsql= ***id=1 AND author=" Robert Peterson"***
3. /books/?dsql= ***genre="fiction" AND watchers in ["Robert Peterson", "J.K Rowling"]***
4. /books/?dsql= ***genre!="fiction" AND watchers in ["Jordan Peterson"]***
5. /books/?dsql= ***genre="Art"***
6. /books/?dsql= ***publication_year>="2019"***

Explanation:
1. Returns the list of books which have given id(ISBN) or given author.
2. Returns the list of books which have has given id(ISBN) and given author.
3. Returns the list of books which have given genre and author name in **any** of the given names.
4. Returns the list of books which **does not** have given genre and authors name in **any** of the given names.
5. Returns the list of books which has given genre.
6. Returns the list of books which has publication_year attribute greater than or equal to the given value.

Summary:
Language should support AND and OR keywords.
Language should support **=**, **!=**, **<**, **<=**, **>**, **>=**, **in** notations.

All the APIs should be documented using postman so that the PYM Frontend team can easily integrate them.

4. Milestones

Based on the priority and deliverables need, the project is planned to have below the milestones. Every milestone is intended to have a deliverable.

Prerequisite: Document all the APIs using postman as soon as they are developed. Make sure to classify them according to the functionality.

Milestone - 1	1. Clone the project from {Shared link} Setup the boilerplate project and run migrations to create the initial schema in the database. 2. Add 5 books and authors manually APIs. Create a URL which returns all the issues from the database in JSON format. 3. Create URLs to get an book by its ID, update a book and delete a book, create all the roles mentioned. 4. Your class and database should have a proper relation between the tables. 5. Swagger integration
Milestone - 2	1. Build the APIs for books and users. Refer section 3.1 above. 2. Build Search and Filter Functionality. Refer to the respective sections in the functional requirements.
Milestone - 3	1. Build Pagination and Filtering on zero level fields (All the fields present in the entity) for **all** the list APIs. Eg: /books/?id=167785 OR author = “J.k Rowling” Filter Issues on Id, Author etc. 2. Implement search query language. Refer section 3.5. 3. All your APIs should be authenticated and should be accessed only by the proper role. 4. Logs implementation should be there.
Milestone - 4	1. Make sure that all the APIs which you create are authenticated. 2. Proper model validations and conditions are matching as per the document. 3. Make changes in models required for all the APIs and migrate them to update the schema. 4. Use of LINQ expression and exceptions handling. 5. Use of ENUMS, CONSTANTS, OOPS, Dependency Injects.
Milestone - 5	1. Custom Filters for common functionality. 2. IIS hosting 3. Use of Tuples 4. Extension Methods.

QUALITY FOCUS

Even though the client doesn’t mention quality, the quality focus has to be there. Whatever we deliver has to be delivered with excellent quality. This is important for getting continued business from PYM Industries and establishing the value HashedIn can deliver to them i.e. “High Quality delivered Quickly”

5. Non Functional Requirements

- Best Practices
    - Verify your codebase. You should follow all the coding standard and you should be able to write the unit test cases and should be able to cover 80% of you code.
    - Swagger integration, Logging and IIS hosting should be working.
    - Code comments, Enums, Model Validations should be implemented as the part of the code.

7. Links for Reading Material

1. TODO WebaPI with Entity framework
2. ORM Queries using Entity framework via LINQ
3. REST Endpoints
4. Authentication and Authorization
5. Linq Pagination, Sorting, Filtering
6. C# Basics