A logo with text on it

Description automatically generated

(CIS128-6) - SOFTWARE DESIGN PATTERNS AND DATA ARCHITECTURES

Assignment – Design and implement a desktop/console or web application using C# and .NET Core which demonstrates the use of design patterns and design and implement a data architecture.

Submitted By

IMRAN AL MUNYEEM (2149039)

Guided By

DR. MAHMOUD ARTEMI

**Abstract**

*The Online Bookstore is a feature-rich web application crafted with C#, ASP.NET Core MVC, .NET Entity Framework Core, HTML5, CSS, and Bootstrap, offering a seamless experience for book enthusiasts and administrators. It empowers administrators with tools for effortless book management, user registration, and login functionality, along with a virtual shopping cart. Rigorous testing ensures reliability, making it a dynamic hub where technology converges with literature for immersive reading* experiences.

Table of Contents

[1. **Description** 5](#_Toc145036291)

[2. **Requirement Analysis** 5](#_Toc145036292)

[2.1 Technology Stack Analysis 5](#_Toc145036293)

[2.2 Functional Requirements 5](#_Toc145036294)

[2.3 Non-Functional Requirements 6](#_Toc145036295)

[2.4 Project Planning 6](#_Toc145036296)

[3. **Software Architectural Design** 6](#_Toc145036297)

[3.1 MVC Pattern 6](#_Toc145036298)

[3.2 Why choose MVC Pattern 7](#_Toc145036299)

[4. **Data Architectural Design** 9](#_Toc145036300)

[4.1 Three-Tier Data Architecture 9](#_Toc145036301)

[4.2 Why choose Three-Tier 9](#_Toc145036302)

[5. **Object-Oriented Programming** 10](#_Toc145036303)

[6. **Agile Methodology** 10](#_Toc145036304)

[7. **Design and Development** 11](#_Toc145036305)

[7.1 Design Making Concept 11](#_Toc145036306)

[7.2 Frontend Development 12](#_Toc145036307)

[7.3 Logic Development 12](#_Toc145036308)

[7.4 Database Development 12](#_Toc145036309)

[7.5 Authorization and Authentication 13](#_Toc145036310)

[8. **Data Modelling** 13](#_Toc145036311)

[8.1 Data Model Classes 13](#_Toc145036312)

[8.2 Data Normalization 14](#_Toc145036313)

[8.3 ERD Diagram 14](#_Toc145036314)

[9. **Testing** 15](#_Toc145036315)

[9.1 Unit Testing 15](#_Toc145036316)

[9.2 User Acceptance Testing 15](#_Toc145036317)

[9.3 API Testing 17](#_Toc145036318)

[9.4 Responsiveness Testing 17](#_Toc145036319)

[10. **Version Controlling & Documentation** 20](#_Toc145036320)

[10.1 Version Controlling with Git 20](#_Toc145036321)

[10.2 Documentation Management with Readme.md 20](#_Toc145036322)

[11. **Issue Management** 21](#_Toc145036323)

[11.1 Technical Challenges 21](#_Toc145036324)

[11.2 Database Challenges 22](#_Toc145036325)

[11.3 Impact 22](#_Toc145036326)

[11.4 Solution 22](#_Toc145036327)

[12. **CI/CD Implementation** 23](#_Toc145036328)

[13. **Future Scalability** 23](#_Toc145036329)

[13.1 Enhanced Features 23](#_Toc145036330)

[13.2 More Content 23](#_Toc145036331)

[13.3 Mobile App Integration 24](#_Toc145036332)

[14. **Possible Alternatives** 24](#_Toc145036333)

[14.1 Alternative Backend Technologies 24](#_Toc145036334)

[14.2 Database Alternatives 24](#_Toc145036335)

[15. **Feelings throughout the project** 24](#_Toc145036336)

[15.1 Excitement and Enthusiasm 24](#_Toc145036337)

[15.2 Learning and Exploration 25](#_Toc145036338)

[15.3 Frustration and Challenges 25](#_Toc145036339)

[16. **Acknowledgement** 25](#_Toc145036340)

[17. **Conclusion** 25](#_Toc145036341)

[18. **Reference** 26](#_Toc145036342)

[19. **Appendix** 26](#_Toc145036343)

[19.1 Use Case Diagram 26](#_Toc145036344)

[19.2 Unit Test Screenshots 28](#_Toc145036345)

[19.3 User Interfaces 34](#_Toc145036346)

[19.4 Git Commit History 37](#_Toc145036347)

[19.5 How to Run this project 37](#_Toc145036348)

# 1. Description

In a world increasingly driven by technology, the pursuit of knowledge, adventure, and escapism finds its new home in my dynamic and feature-rich web application – online Bookstore. I have developed a smooth and user-friendly platform for readers and administrators that is powered by C#, ASP.NET Core MVC, .NET Entity Framework Core, HTML5, CSS, and Bootstrap. My toolkit for admins enables you to easily manage books, including the ability to create, read, edit, and delete items as well as manage user roles. Registration and login are made simple by the user management system, and you can even add books to your virtual shopping cart. Users can even perform the search operation based on their budget range, title or author name. Dynamic toaster notification will give the amazing experience when anyone would interact with the create, read, edit, delete operation. Overall, I have ensured top-notch quality with rigorous testing, including unit testing (C#), integration testing (JavaScript/Cypress), API testing (Postman), and responsiveness testing to provide the user their best experience.

# 2. Requirement Analysis

### 2.1 Technology Stack Analysis

Before starting the project, I had to evaluate whether the chosen tech tools would be suitable to build this project, possible issue, and solution, had to look for free resources and community support also whenever necessary.

### 2.2 Functional Requirements

A project is successful only when its user experience is good enough. I had to analyse and make sure that the project has to be user-friendly so that users can easily interact with it, feel good, interact with it on any available device they have. I also had to consider the features and functionalities so that these things work properly. Create, read, delete, edit operation was one of the important parts in data which I kept in mind before starting the project.

### 2.3 Non-Functional Requirements

Performance, readability, scalability, security, maintainability was in the requirements also before starting this project. That’s why I planned to go with OOP principals, MVC, three-tier architecture so that I can manage all of them.

### 2.4 Project Planning

Project planning was one of the crucial parts of this project. I created the project plan with milestones and deadlines for the smoothness of the project workflow and success. I allocated required resources and necessary tools before proceeding.

# 3. Software Architectural Design

### 3.1 MVC Pattern

MVC (Model, View, Controller) is one of the most popular software architectural design patterns where model contains the model classes, views contain the view classes, and controller contains controller classes. Here model classes create the relationship with database, controller control the model classes and acts as intermediary between model and views, and views classes contains html files to display the user interfaces of the application.

A diagram of a model

Description automatically generated

Figure 1: MVC (Model, View, Controller) software architectural pattern.

### 3.2 Why choose MVC Pattern

The use of the Model-View-Controller (MVC) pattern in my project was highly advantageous for several reasons:

**Separation of Concerns:** MVC enforced a clear separation of concerns, which is particularly beneficial in a complex project like an online bookstore. Each component (Model, View, and Controller) has a distinct role, making it easier to manage and maintain different aspects of the application.

**Modularity and Reusability:** MVC promotes modularity by isolating different parts of my application. This allows for easier code reusability and testing. For example, I can reuse the Controller logic when adding new features or expand the View templates without impacting the underlying data handling in the Model.

**Scalability:** As my online bookstore grows and potentially expands its features, MVC allows for scalable development. I can add new Views for different types of users or devices (e.g., mobile, desktop) while keeping the same underlying Model and Controller logic.

**Maintainability:** With MVC, changes to one component are less likely to affect others. This facilitates maintenance and updates, as I can modify specific parts of the application without extensive rework. For instance, I can update the View's appearance without altering the business logic in the Controller or the data handling in the Model.

**Maintainable User Management:** My robust user management system, including registration, login, and shopping cart features, is well-suited for MVC. These functionalities can be neatly organized within the Controller and View components, ensuring that user interactions and data handling remain distinct.

Here is the MVC pattern of my project:

A screenshot of a computer

Description automatically generated

Figure 2: MVC pattern of bookstore.

# 4. Data Architectural Design

### 4.1 Three-Tier Data Architecture

Apart from 1-Tier and 2-Tier, I choose 3-Tier Data Architecture design. A 3-tier data architecture design is a way to structure a software application into three distinct layers:

**Presentation Layer:** This is the user interface layer, responsible for displaying information to users and handling their interactions. Here, in this layer I stored html, CSS, JavaScript, bootstrap, and other static pages.

**Application Logic Layer**: This layer contains the business logic and processing of the application. It acts as an intermediary between the presentation layer and the data storage layer, handling tasks like user authentication, data manipulation, and application workflows. C# controller logics are the part of this layer in my application.

**Data Storage Layer:** The project database is the part of this layer in my application. This layer works for storing and retrieving the data. Models and database are connected to work on this layer.

### 4.2 Why choose Three-Tier

**Scalability:** This kind of tier is highly scalable as it provides separate UI, logic, and database layer. For instance, if I get huge traffic in my website in future I can very easily scale the logic and database layer separately without impacting the other parts of the application.

**Modularity and Maintainability:** 3-tier architecture is also maintainable because of its clean separation. Every layer is separated and has its own responsibilities which makes it easier to understand, modify, and develop.

**Security:** Security is one of the core benefits of using three tier architecture. Because of the separation, I would be easily protecting my application from SQL injection, XSS injection since the logic and database layer is separated.

**Performance:** I can optimize the performance of the website by separating the database, logic, and UI layer

**Maintainable Codebase:** For the clean separation, it’s very easy to debug the code and fixes to bugs whenever I needed.

# 5. Object-Oriented Programming

I built this project by strictly following object-oriented principles. I encapsulated the different functionalities within classes and methods where I made sure that only authorized users can access and manipulate these functionalities.

Besides, I followed abstraction by stay away from the complexity of interacting with my database by using Entity Framework. Entity Framework abstracted the database operations into a set of C# classes, allowed me to work with objects and entities rather than dealing directly with SQL queries.

# 6. Agile Methodology

Throughout the development process of my project, I followed agile methodology by emphasizing iterative development, where I continuously worked on small increments of the project, emphasizing on customer feedback and satisfaction by developing features for both customers and administrators, conducting continuous testing, receiving continuous feedback and reviews from my course instructor **Dr. Mahmoud Artemi**. These characteristics proved that I strictly followed agile methodology throughout the development of my project.

# 7. Design and Development

### 7.1 Design Making Concept

Before starting the project, I have conducted decision analysis based on functionalities, user interactions where I found out what actions to take after completion of the single step.   
Here’s the complete flowchart of the decision analysis:

A screenshot of a computer screen

Description automatically generated

Figure 3: Decision making flowchart.

### 7.2 Frontend Development

I used a combination of web development technologies and tools to develop the frontend of my bookstore web application. Here's a breakdown of the tools and technologies I used:   
 **Views:** My frontend development stack includes HTML5 and CSS3 for basic structure and styling, Bootstrap for UI components and responsiveness, JavaScript for interactivity, and Cypress for testing.

### 7.3 Logic Development

Backend logics were developed by C# classes. I created the controller classes those are responsible for create, read, edit, delete operation. I also used entity framework for relation making with database. It helped me to interact with a relational database using C# classes and LINQ queries.

### 7.4 Database Development

I used a MSSQL database for my Online Bookshop web application. Here's how I used the database:

I used SQL Server and Microsoft SQL server management studio.

**Usage of the Database:**

**Data Storage:** I utilized the database for storing books, Cart, Cart Item, Order, Order Items, Contact, Users data and their interactions.

**Entity Framework Core:** Using the help of entity framework core I interacted with the relational database using the C# controller classes and LINQ queries. I used this framework develop and perform to create, read, update, and delete records in the database.

**User Management:** The registration and login functionalities contain different input fields names, usernames, passwords, email, address, and profiles were default feature of the .net core scaffolding item. I implemented it and stored in database.

**Book Management:** For create, edit, read, delete operations in book I used the database to keep the book information like tittle, description, author name, price, image.

**Shopping Cart:** For the cart operation, I used temporary storage of items in the database.

**User Roles:** Admin can manage different roles which operation was connected to database and users.

### 7.5 Authorization and Authentication

I used scaffolded items from the .net core which is a default package of the .net core team, and it helped me to build the user authorization and authentication process by providing default registration and login. The top characteristics of this feature was it came up with built in validation. After implementing this feature, I connected it to database where I stored all the information of the users.

# 8. Data Modelling

### 8.1 Data Model Classes

I developed book data model that included properties like Title, Author, Description, Language, Image, Creation Date and Price. User data model to keep information such as Username, Email, Password Hash, and other relevant attributes. Similarly, I created order, order items, cart model classes. Entity Framework Core mapped this model classes with database to store this information.

### 8.2 Data Normalization

In my project I utilized data normalization by eliminating data redundancy where I minimized the duplication of data, improving data integrity by providing primary and foreign key constraints. This normalization process helped me to avoid update anomalies, retrieve and query data efficiently, make the application scalable.

### 8.3 ERD Diagram



Figure 4: ERD Diagram of iBookStore Database.

# 9. Testing

Both white-box and black-box testing were conducted on this project. Including smoke, sanity to end to end testing to test the overall features of the whole website.

### 9.1 Unit Testing

I used the XUnit testing framework and Moq library to create mock objects for the different controller classes. Mocking allowed me to separate the controller's behaviour from its dependencies, making it easier to test. These unit tests helped ensure that the controller behaves as expected under different scenarios and with different input data.

A screenshot of a computer program

Description automatically generated

Figure 5: List of the unit tests I have conducted.

### 9.2 User Acceptance Testing

I used JavaScript and Cypress to conduct integration test. I started from smoke testing to check the stability of the website. Later, I conducted end to end testing for the whole website. All the tests passed successfully. Check out the mochawesome html report:

A screenshot of a computer

Description automatically generated

Figure 6: Login Page Test.

A screenshot of a computer

Description automatically generated

Figure 7: Contact and Registration Page Test.

A screenshot of a computer

Description automatically generated

Figure 8: Books and Roles Page Test.

### 9.3 API Testing

During the development of the project, I conducted API (Application Programming Interface) testing to check the routing connection to make sure the seamless communication between website to database. Ended with “connection okay” message all the time.

### 9.4 Responsiveness Testing

I also tested the responsiveness of the webpages to know whether it displays properly on different devices. I made sure that user get the best user experience.  
Here have a look at the different view for the different devices:

**Tested on the device with 75% width and it looks completely fine. So, it passed the responsiveness test here:**

A computer with a website on the screen

Description automatically generated

Figure 9: Responsive test for laptop with 75% width.

**Tested on the device with 50% width and it looks completely fine. So, it passed the responsiveness test here:**

A screenshot of a computer

Description automatically generated

Figure 10: Responsive test for tablet with 50% width.

**Tested on the iPhone devices and it looks completely fine. So, it passed the responsiveness test here:**

A cell phone with a screen showing a photo

Description automatically generated

Figure 11: Responsiveness test for iPhone 14Pro.

# 10. Version Controlling & Documentation

### 10.1 Version Controlling with Git

Throughout the development of the whole project, I used both GitHub application and Git CLI which helped me to manage and control the versions of my code.

### 10.2 Documentation Management with Readme.md

This is one of the useful features of the GitHub I used. I used readme.md file from GitHub to the documentation and user guide for the project. Anyone can go through it for end-to-end instruction including how to build this project from the scratch and run it.

# 11. Issue Management

During the development of the Online Bookshop web application, several challenges and issues were encountered, necessitating effective issue management and troubleshooting strategies to ensure the project's success:

### 11.1 Technical Challenges

**Version Mismatches:** One of the primary issues faced was dealing with version mismatches between various libraries and packages used in the project, particularly in the .NET Core ecosystem. Different packages often had dependencies on specific versions of other packages, leading to conflicts.

**Deprecated Libraries**: Some libraries used in the project had deprecated features or were no longer actively maintained, causing potential compatibility and security issues.

A screenshot of a computer error

Description automatically generated

Figure 12: Scaffolded issue.

**Compatibility with Latest .NET Core:** Certain packages and libraries were not fully compatible with the latest version of .NET Core, posing challenges when attempting to integrate them seamlessly.

### 11.2 Database Challenges

**Database Configuration:** Setting up and configuring the database to work seamlessly with Entity Framework Core was a critical challenge. Ensuring the correct connection string and database provider configuration was essential for proper data management.

**Data Integrity:** Maintaining data integrity and consistency while performing CRUD (Create, Read, Update, Delete) operations on the database required careful handling, especially when dealing with complex relationships between entities.

### 11.3 Impact

Version mismatches resulted in conflicts, making it challenging to integrate and maintain packages consistently, deprecated libraries posed a risk in terms of compatibility, security, and stability, potentially leading to issues in the application, compatibility issues hindered the seamless integration of certain components, requiring additional effort to adapt the project to the latest .NET Core version, incorrect database configurations caused database-related issues, leading to errors in data retrieval and storage.

### 11.4 Solution

Taking the help of community forums like Stack Overflow and platforms like Quora was conducted to identify potential solutions to version mismatch issues. Besides, leveraging online resources, including YouTube tutorials and official documentation, helped in resolving many of the compatibility problems.

Detailed documentation and guidance from the project instructor **Dr. Mahmoud Artemi** helped me a lot and were referred to for configuring the database and resolving data integrity issues.

# 12. CI/CD Implementation

I used Jenkins to for CI/CD implementation. I created freestyle job and pulled the code directly from the git and setup automatic pipeline to build, run, test it whenever I needed. It was one of the crucial parts of this successful project development.

# 13. Future Scalability

This application is highly scalable, and we can easily add more features to in in the near future and I’m going to discuss few of them:

### 13.1 Enhanced Features

**AI Recommendation Systems:** We can use AI recommendation system which can take and analyse data from the user based on their browsing history, interest, and simple questions and finally recommend them their favourite books.

### 13.2 More Content

**Expand Book Category:** We can add new books, new categories, genre, author continuously to grow this book shop like a real business.

**Multilingual feature:** We can easily introduce multilingual feature so that everyone from the world can access the website to read and buy the books.

### 13.3 Mobile App Integration

Develop We can develop and integrate android and IOS applications for this book platform. As a result, we’ll get more users.

# 14. Possible Alternatives

### 14.1 Alternative Backend Technologies

**Node.js and Express:** Instead of using C# and ASP.NET Core MVC, I could use JavaScript, Node.js with Express the backend development since node.js is very well known for its non-blocking and event-driven architecture and it provides excellent performance for web applications.

### 14.2 Database Alternatives

**NoSQL:** Although Entity Framework Core is a powerful for relational databases, but I could use MongoDB as an alternative since it’s more popular and suitable for the complex projects.

# 15. Feelings throughout the project

Journey of developing the Online Bookstore web application has been a rewarding yet challenging experience.

### 15.1 Excitement and Enthusiasm

Before starting this project, I was very excited to build this dynamic book shop platform using .net core and C# as I got the opportunity to learn a new thing.

### 15.2 Learning and Exploration

During the development of this project, although the learning part was quiet challenging since learning and developing at the same time is really difficult, but the end result was really good.

### 15.3 Frustration and Challenges

Got frustrated several times because of the technical issues like package mismatched, deprecated, .net core errors, lack of sufficient time. These things were challenging but ended up with a successful project with time management skill.

# 16. Acknowledgement

I extend my sincere appreciation to **Dr. Mahmoud Artemi**, my course instructor, for his valuable guidance, resources, and lectures that significantly contributed to the successful development of the Online Bookstore web application. His live lectures, video footages, particularly lectures on database normalization, data modelling, and ERD model design enhanced my skill and helped me a lot throughout the development of this project. **Dr. Mahmoud Artemi**'s mentorship played a pivotal role in enhancing my technical skills and knowledge in this project, indeed.

# 17. Conclusion

The Online Bookstore is a seamless combination of cutting-edge technology. This dynamic web application, which was expertly created with C#, ASP.NET Core MVC, .NET Entity Framework Core, HTML5, CSS, and Bootstrap, delivers an engaging experience for book lovers and administrators. It provides access to a wide literary universe by providing strong facilities for book management, user registration, and login. It is a long-lasting platform where technology highlights the charm of books because rigorous testing ensured its stability. The Online Bookstore is more than just a website; it serves as a gateway to endless literary exploration, where each book is a potential adventure.

# 18. Reference

OpenAI. (n.d.). ChatGPT. [https://chat.openai.com]

# 19. Appendix

### 19.1 Use Case Diagram

A diagram of a person

Description automatically generated

Figure 13: Use case diagram.

### 19.2 Unit Test Screenshots

A screen shot of a computer program

Description automatically generated

Figure 14: Unit test for admin controller class

A screen shot of a computer program

Description automatically generated

Figure 15: Unit test for book controller class

A screen shot of a computer program

Description automatically generated

Figure 16: Unit test for cart controller class

A computer screen shot of a program code

Description automatically generated

Figure 17: Unit test for contact controller class

A computer screen shot of a program code

Description automatically generated

Figure 18: Unit test for order controller class

A screen shot of a computer program

Description automatically generated

Figure 19: Unit test for store controller class

### 19.3 User Interfaces

A screenshot of a computer

Description automatically generated

Figure 20: Book list interface.

A screenshot of a computer

Description automatically generated

Figure 21: Contact form interface.

A screenshot of a book

Description automatically generated

Figure 22: Home page interface.

A screenshot of a library

Description automatically generated

Figure 23: Login page interface.

A screenshot of a computer

Description automatically generated

Figure 24: Register page interface.

### 19.4 Git Commit History

A screenshot of a computer

Description automatically generated

Figure 25: Git commit history.

### 19.5 How to Run this project

**Step 1:** Install visual studio 2022, SQL Server, SQL Server Management Studio

**Step 2:** Unzip the project folder in your local.

**Step 3:** Open the “iBookStore.Web.csproj” or “Solution” file in visual studio.   
**Step 4:** Open the “appsettings.json” and modify the config file. Change the server’s name to your own server’s name.  
**Step 5:** Run Below Command

**Add-migration initialization**   
// It will push the data from model classes to database.   
 **Update-Database   
//** It will update the tables in database.