Your Books EveryWhere

Student:Alexandru-Horea Anisorac

**Group:30238**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

The application that we have to implement is an application to search books and buy them into every client library.If another client has a book,than the others are being pushed to a queue until the client retrieves it.Clients can filter books after title,genre and author.These books have two more characteristics:release date and if it is available or note.

# Functional Requirements

The application will be designed and implemented in C# language and framework is ASP.NET.The Client should be able to log in,register,filter books and add them to their library.

The managers should manage the books and process the requests.

# Non-functional Requirements

First requirement is to implement the application with one architectural pattern and that is Client-Server.Another requirement is to use one OOP language.And the last one we have to use Observer design pattern.

2. Use-Case Model

*[Create the use-case diagrams and provide one use-case description (according to the format below).*

*Use-Case description format:*

*Use case: <use case goal>*

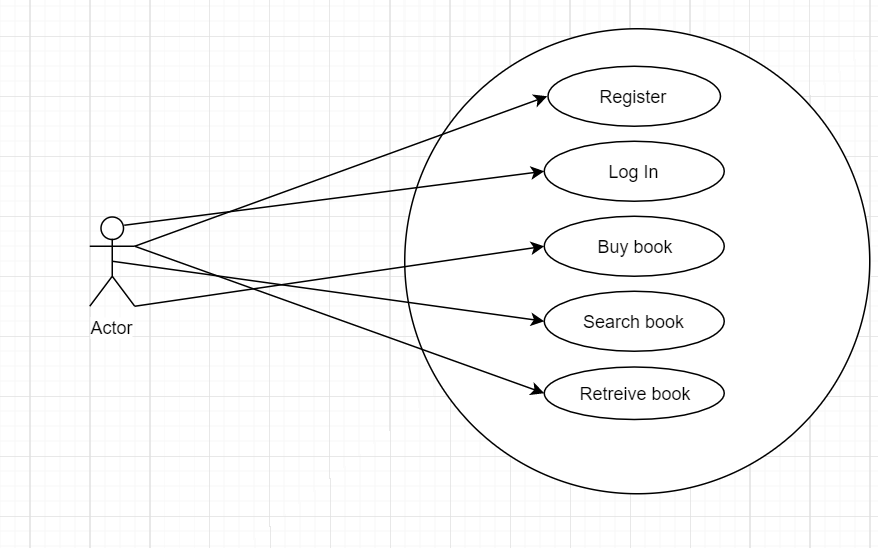
*Level: <one of: summary level, user-goal level, sub-function>*

*Primary actor: <a role name for the actor who initiates the use case>*

*Main success scenario: <the steps of the main success scenario from trigger to goal delivery>*

*Extensions: <alternate scenarios of success or failure>*

*]*



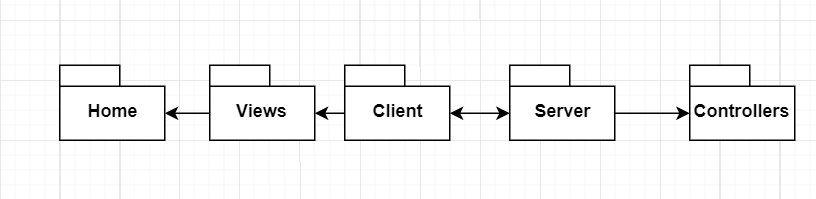
3. System Architectural Design

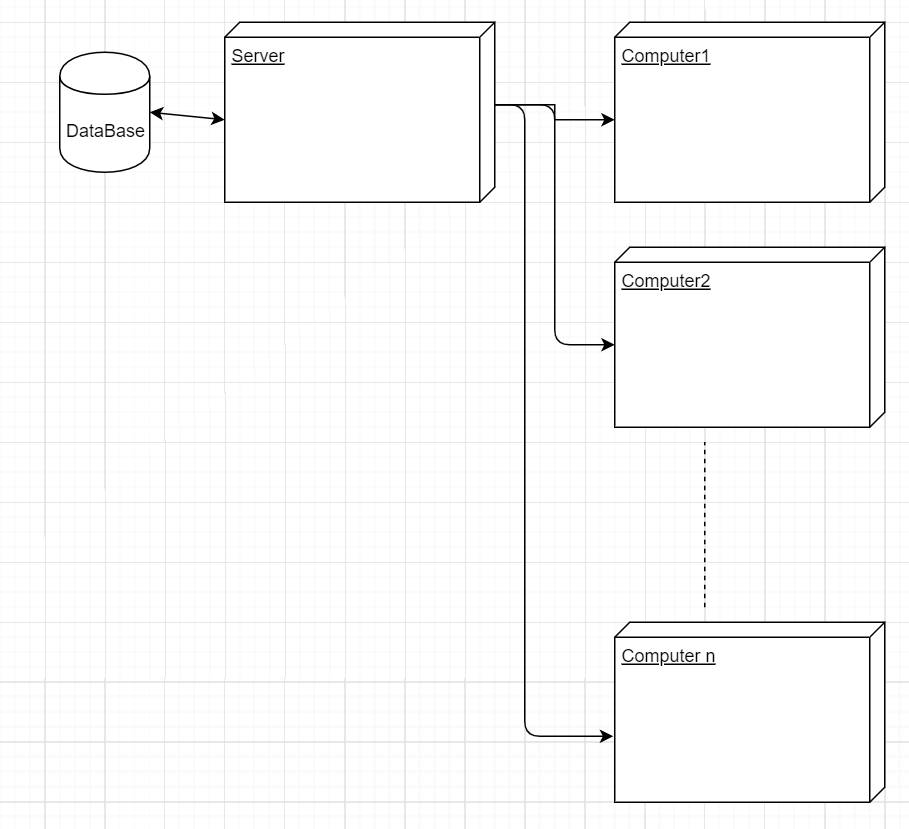
**3.1 Architectural Pattern Description**

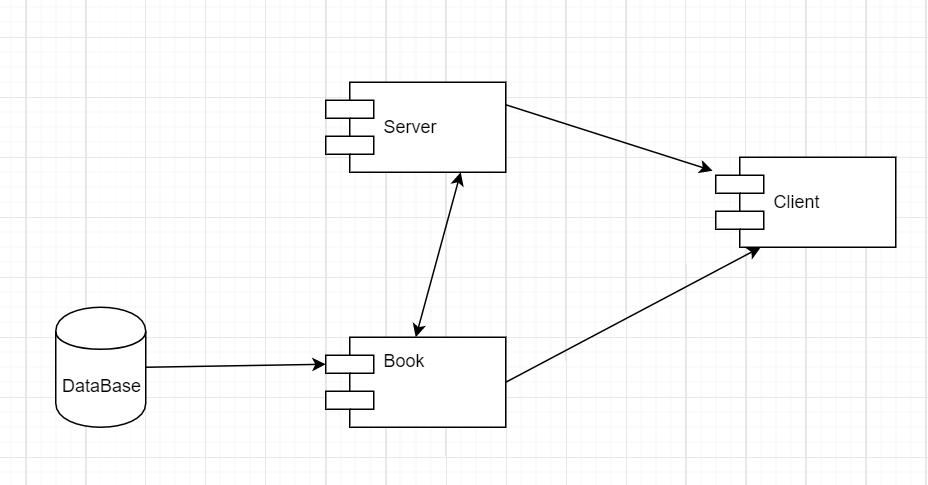
Client–server model is a [distributed application](https://en.wikipedia.org/wiki/Distributed_application) structure that partitions tasks or workloads between the providers of a resource or service, called [servers](https://en.wikipedia.org/wiki/Server_(computing)), and service requesters, called [clients](https://en.wikipedia.org/wiki/Client_(computing)). Often clients and servers communicate over a [computer network](https://en.wikipedia.org/wiki/Computer_network) on separate hardware, but both client and server may reside in the same system. A server [host](https://en.wikipedia.org/wiki/Host_(network)) runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests. Examples of computer applications that use the client–server model are [Email](https://en.wikipedia.org/wiki/Email), [network printing](https://en.wikipedia.org/wiki/Network_printing), and the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web).

**3.2 Diagrams**

*[Create the system’s conceptual architecture; use architectural patterns and describe how they are applied. Create package, component and deployment diagrams]*

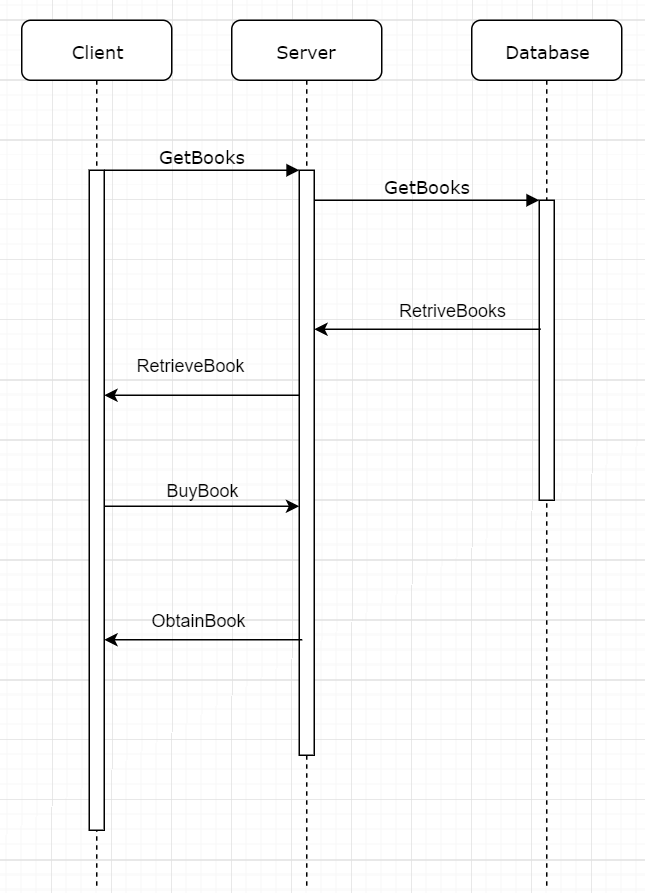






4. UML Sequence Diagrams

*[Create a sequence diagram for a relevant scenario.]*



5. Class Design

**5.1 Design Patterns Description**

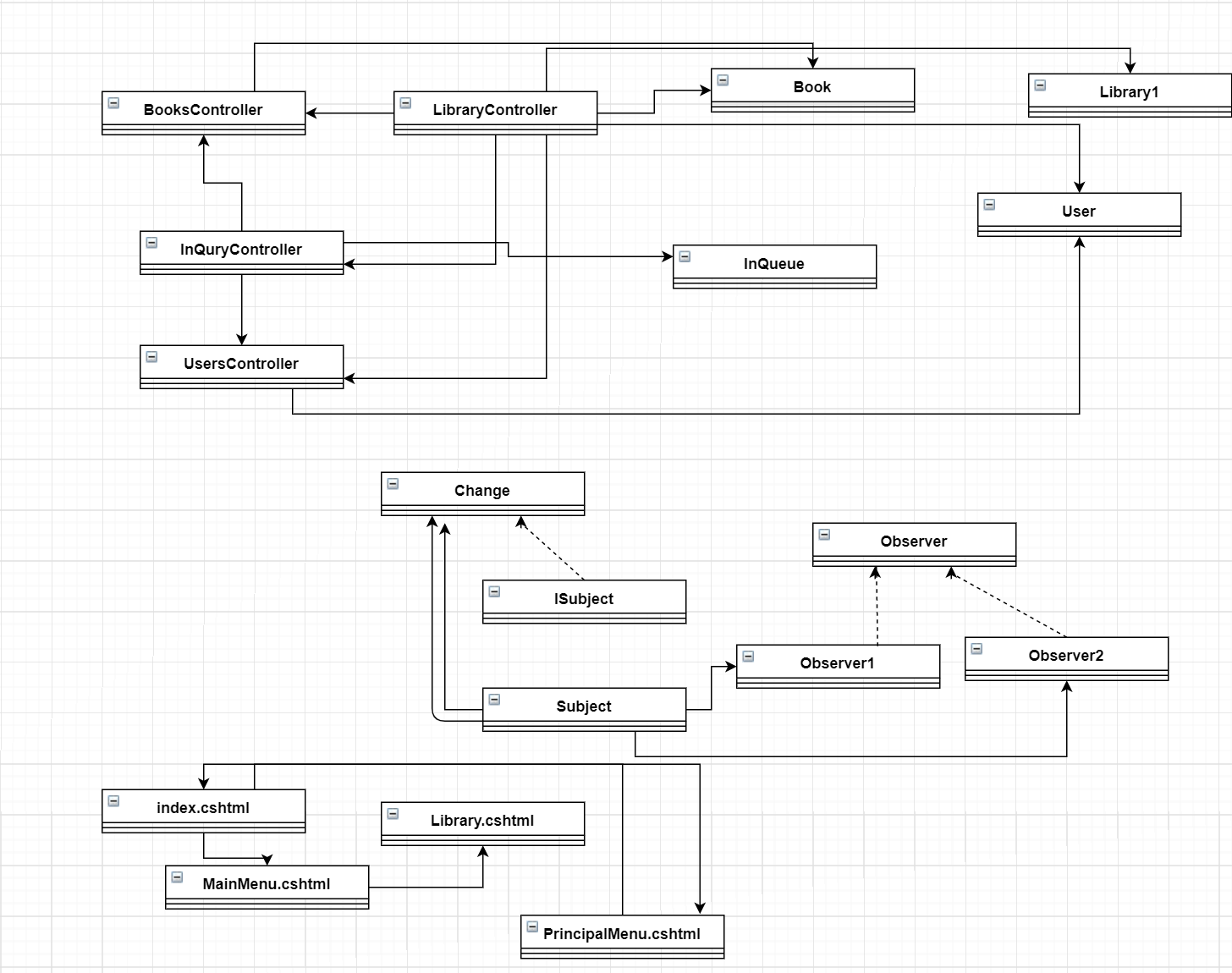
The observer pattern is a [software design pattern](https://en.wikipedia.org/wiki/Design_pattern_(computer_science)) in which an [object](https://en.wikipedia.org/wiki/Object_(computer_science)#Objects_in_object-oriented_programming), called the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their [methods](https://en.wikipedia.org/wiki/Method_(computer_science)).

It is mainly used to implement distributed [event handling](https://en.wikipedia.org/wiki/Event_handling) systems, in "event driven" software. Most modern languages such as C# have built-in "event" constructs which implement the observer pattern components.

The observer pattern is also a key part in the familiar [model–view–controller](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) (MVC) architectural pattern. The observer pattern is implemented in numerous [programming libraries](https://en.wikipedia.org/wiki/Programming_library) and systems, including almost all [GUI toolkits](https://en.wikipedia.org/wiki/GUI_toolkit).

**5.2 UML Class Diagram**

*[Create the UML Class Diagram and highlight and motivate how the design patterns are used.]*



6. Data Model

Data that we use in these application were :Users and Books.We create two independent classes which we control all the data.

7. System Testing

The system was tested with PostMan and passed all tests.

8. Bibliography