News Agency

Analysis and Design Document

Student: George-Cristian Cimpoies

**Group: 30432**

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

Design and implement a client-server application for a news agency.

# Functional Requirements

The application should have two types of users (readers and writers). The reader can use the application without the need to login, while the writer needs a username and a password in order to authenticate.

The regular reader can perform the following operations:

-View a list of articles

-Can read an article

The writer user can perform the following operations:

- Create, update or delete article

- Create, update or delete account

# Non-functional Requirements

* Availability: the system should be available all the time, except the planned maintenance periods
* Performance: the system should have a time response no longer than 5 seconds
* Security: clear distinction between readers and writers. Readers can not alter the database.
* Reusability: the system should be design as having components that could be reused by other systems if necessary
* Usability: the system will have a user-friendly interface and will be easy to use

2. Use-Case Model

*Use case:* Create article

*Level:* User-goal level

*Primary actor:* Writer

*Main success scenario:*

-the writer logs in her/his account using a username and a password

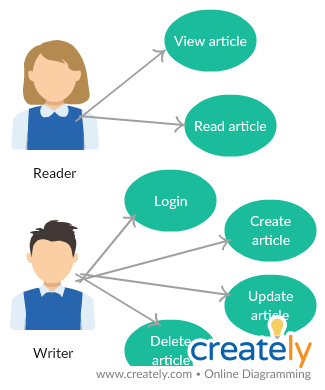
-the writer introduces the fields required to insert a new article

-the writer presses the insert button

*Extensions:*

-problems at the log in, such as student forgetting the username or the password

-error encountered when accessing the database



3. System Architectural Design

**3.1 Architectural Pattern Description**

Client-Server architectural pattern - The client/server architectural style describes distributed systems that involve a separate client and server system, and a connecting network. Servers are the providers of services and clients are the ones that request a service. In our application, the data will be transferred in JSON format using sockets. JavaScript Object Notation (aka JSON) is a very popular alternative to XML for transmitting data to the web browser.

The architectural pattern used is Layers. Components within this pattern are organized into horizontal layers, each layer performing a specific role within the application. Although it does not specify the number and types of layers that must exist in the pattern, most layered architectures consist of three standard layers: presentation, business and database.

1. Presentation layer: responsible for handling all user interface and browser communication logic
2. Business layer: responsible for executing specific business rules associated with the request
3. Database layer: responsible for executing SQL statements to retrieve the corresponding data and pass it back up in the business layer.

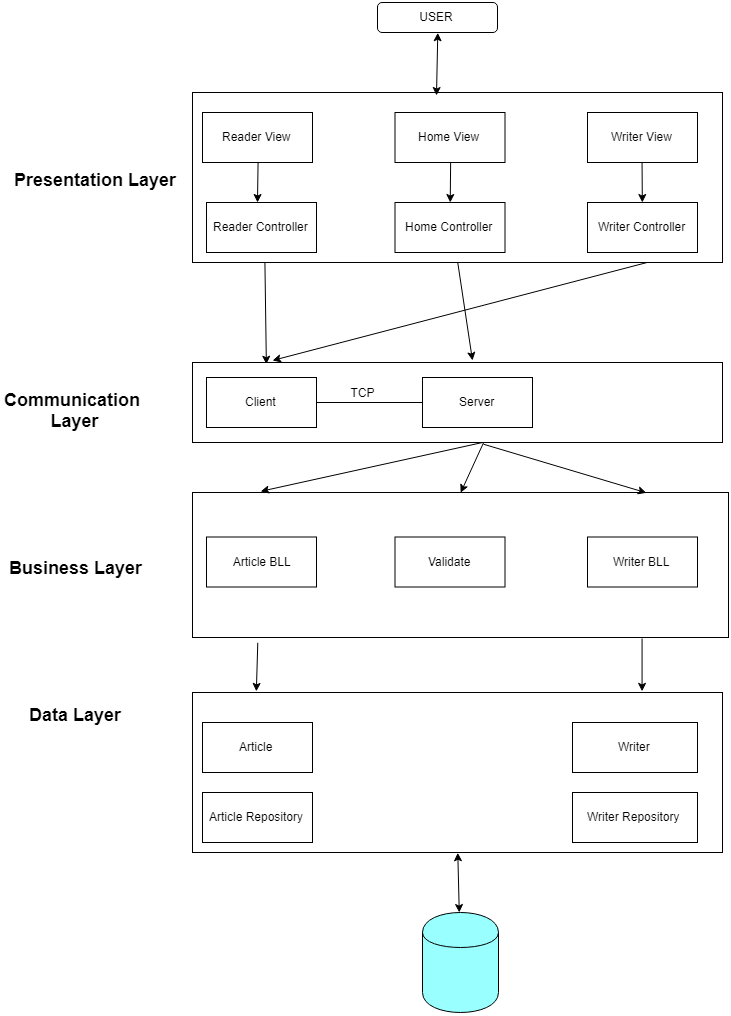
MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate

application's concerns as follows:

1. Model - This part of the framework is to store the data of the application, such as databases, text data, files and/or other web resources.
2. View - This is the graphical user interface of the application. That would contain different buttons, text boxes and other controls to let the user interact with the application to complete his projects depending on the sort of the software he is using.
3. Controller - The actual back-end code constitutes the controller of the framework. A controller controls the data coming from the users or going to the user from a model.

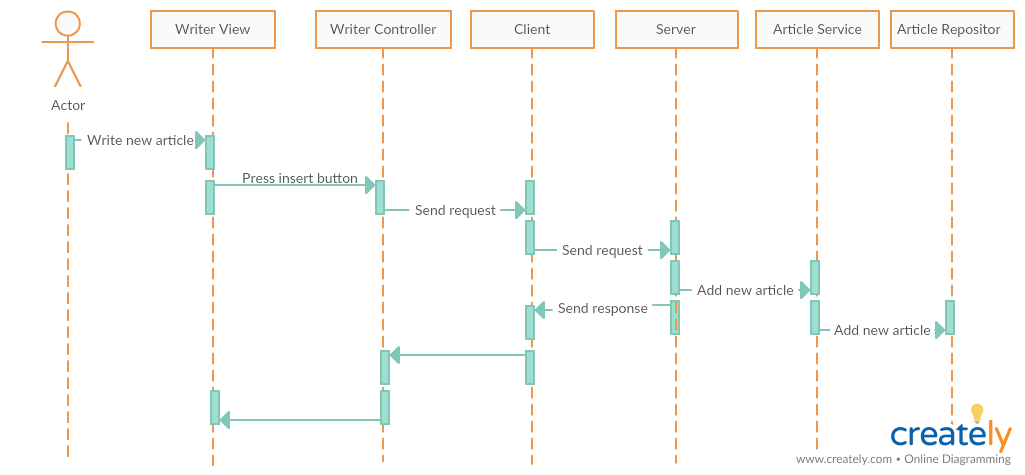
**3.2 Diagrams**

**System architecture diagram**



1. UML Sequence Diagrams

The sequence diagram for writing an article:

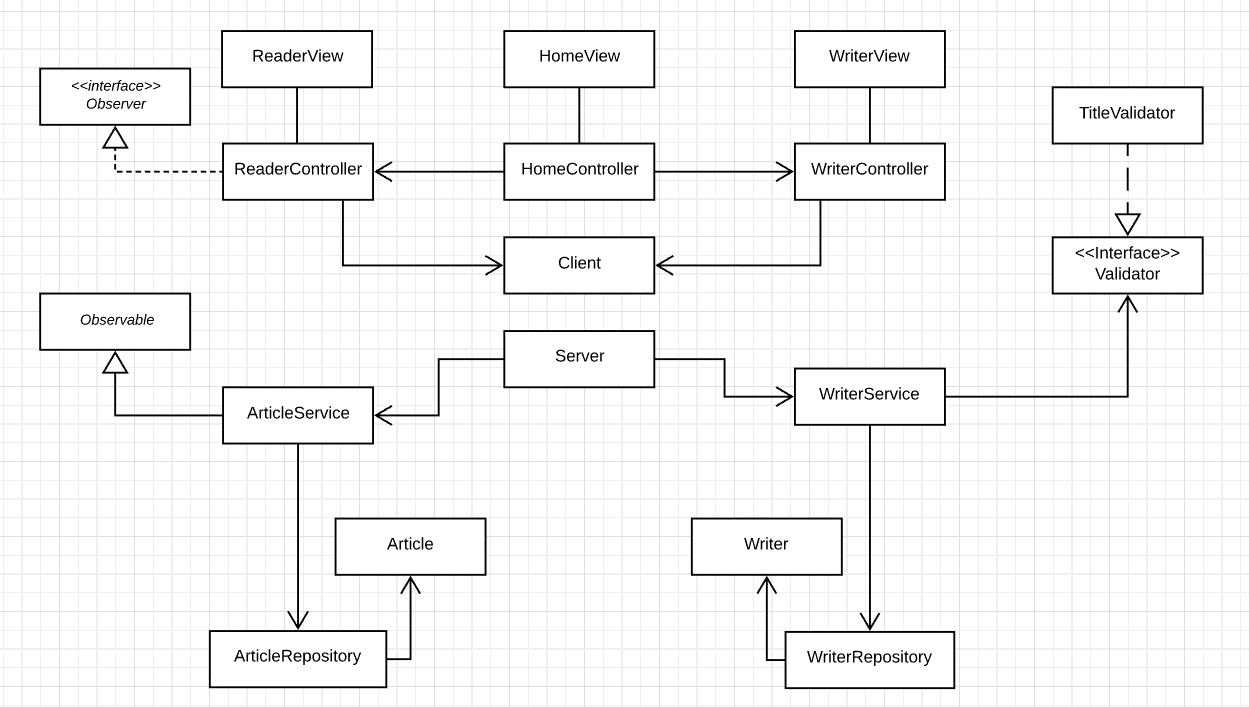


5. Class Design

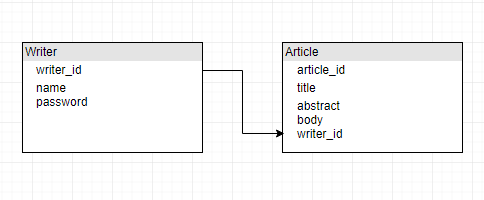
**5.1 Design Patterns Description**

Observer pattern is used when there is one-to-many relationship between objects such as if one object is modified, its dependent objects are to be notified automatically. Observer pattern falls under behavioral pattern category. Observer pattern uses three actor classes. Subject, Observer and Client. Subject is an object having methods to attach and detach observers to a client object.

**5.2 UML Class Diagram**



6. Data Model



1. System Testing

Mockito is a popular mock framework which can be used in conjunction with JUnit. Mockito allows us to create and configure mock objects, by simplifying the development of tests for classes with external dependencies significantly.

A mock object is a dummy implementation for an interface or a class in which we define the output of certain method calls.

1. Bibliography

<https://www.safaribooksonline.com/library/view/software-architecture-patterns/9781491971437/ch01.html>

<https://msdn.microsoft.com/en-us/library/ee658109.aspx>