ASSIGNMENT A3

**Blog Application**

Student: Biris Alexandra

**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

The task is to 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

1. Availability: the system needs to be available 90% for the user with less activity during summer time
2. Accuracy: the system should accurately provide real time information
3. Performance: the system should respond to the user in less than several seconds from the time of the request submittal
4. Security: all system data must be backed up every 24 hours and this can be achieved by developing a second database
5. Usability: the system will have a GUI, it will be user friendly and it will not require a special training

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

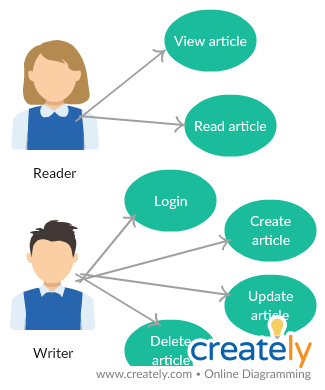
-the writer introduces the fields required to insert a new article

-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**

The Client–Server architecture is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. The communication that occurs between the client and the server must be reliable, so, no data can be dropped and it must arrive on the client side in the same order in which the server sent it. In this application, the clients communicate with the server through sockets, and the data is transmitted using JSON serialization.

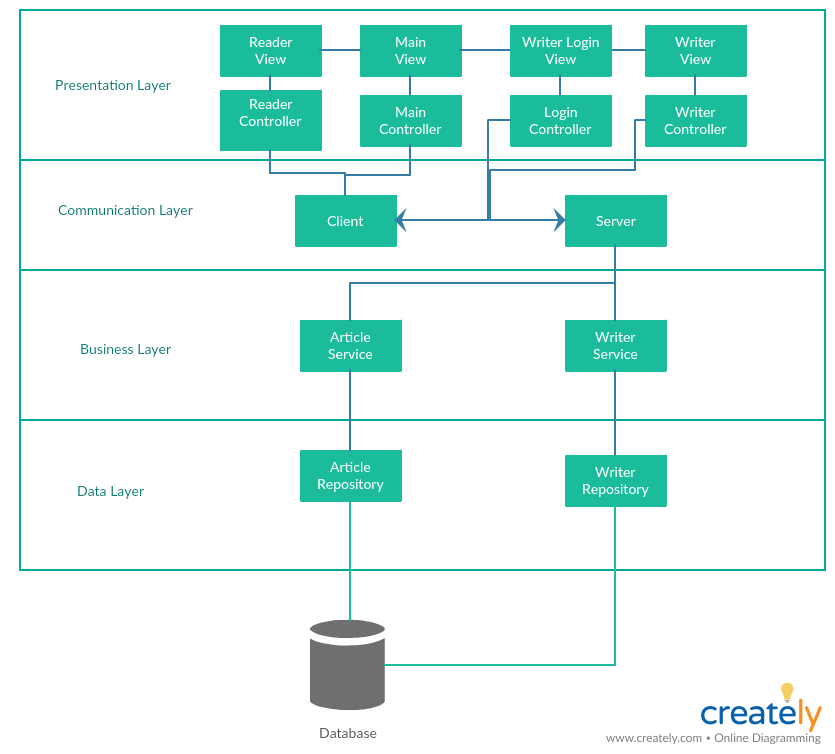
The architectural pattern used is Layers. Components within this pattern are organized into horizontal layers, each layer performing a specific role within the application. Although it does not specify the number and types of layers that must exist in the pattern, most layered architectures consist of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

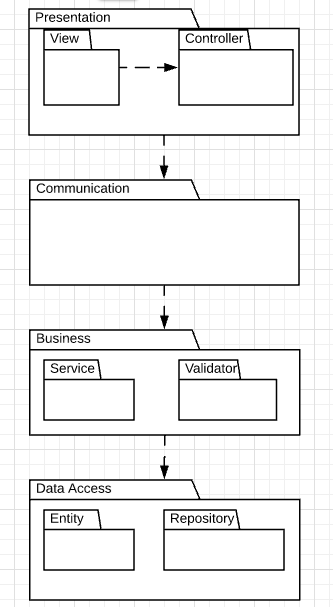
application's concerns as follows:

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.

****

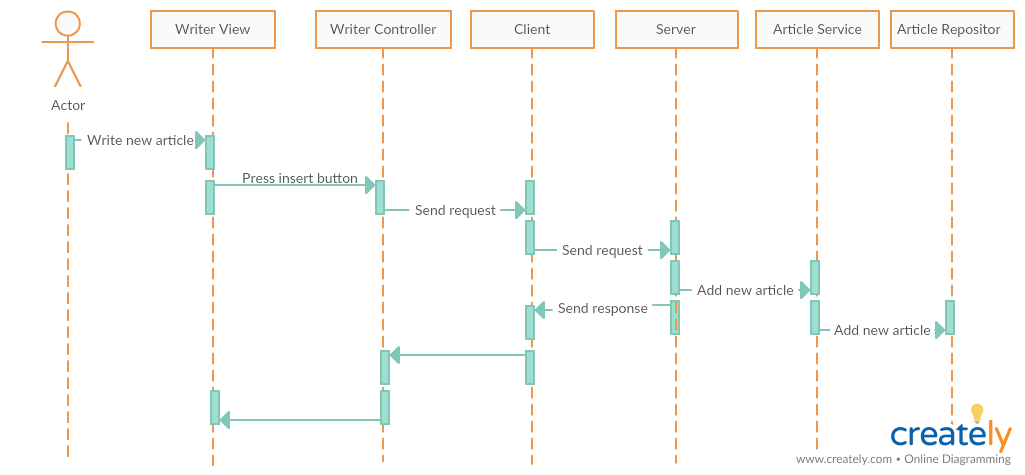
**3.2 Diagrams**

Package diagram:



4. UML Sequence Diagrams

The sequence diagram for writing an article:

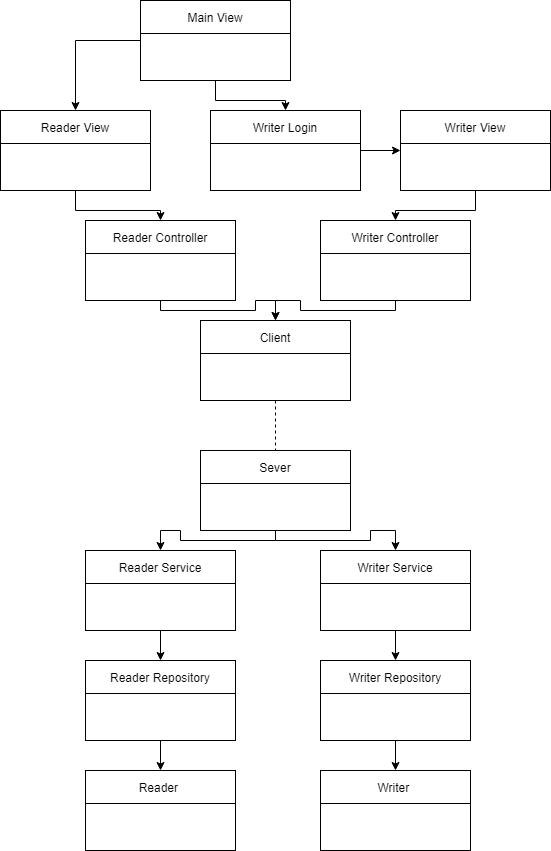


5. Class Design

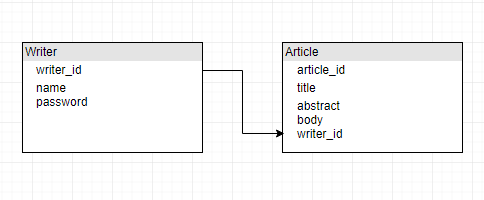
**5.1 Design Patterns Description**

Observer pattern is used when there is one-to-many relationship between objects such as if one object is modified, its depenedent 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



7. 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. Mock objects are configured to perform a certain behavior during a test.

8. Bibliography

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

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