News Agency

Analysis and Design Document

Student: Cordea Corina

**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 4

3. System Architectural Design 5

4. UML Sequence Diagrams 6

5. Class Design 6

6. Data Model 7

7. Bibliography 8

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)

Readers:

* + Do not need to login
  + Can view a list of articles
  + Can read an article

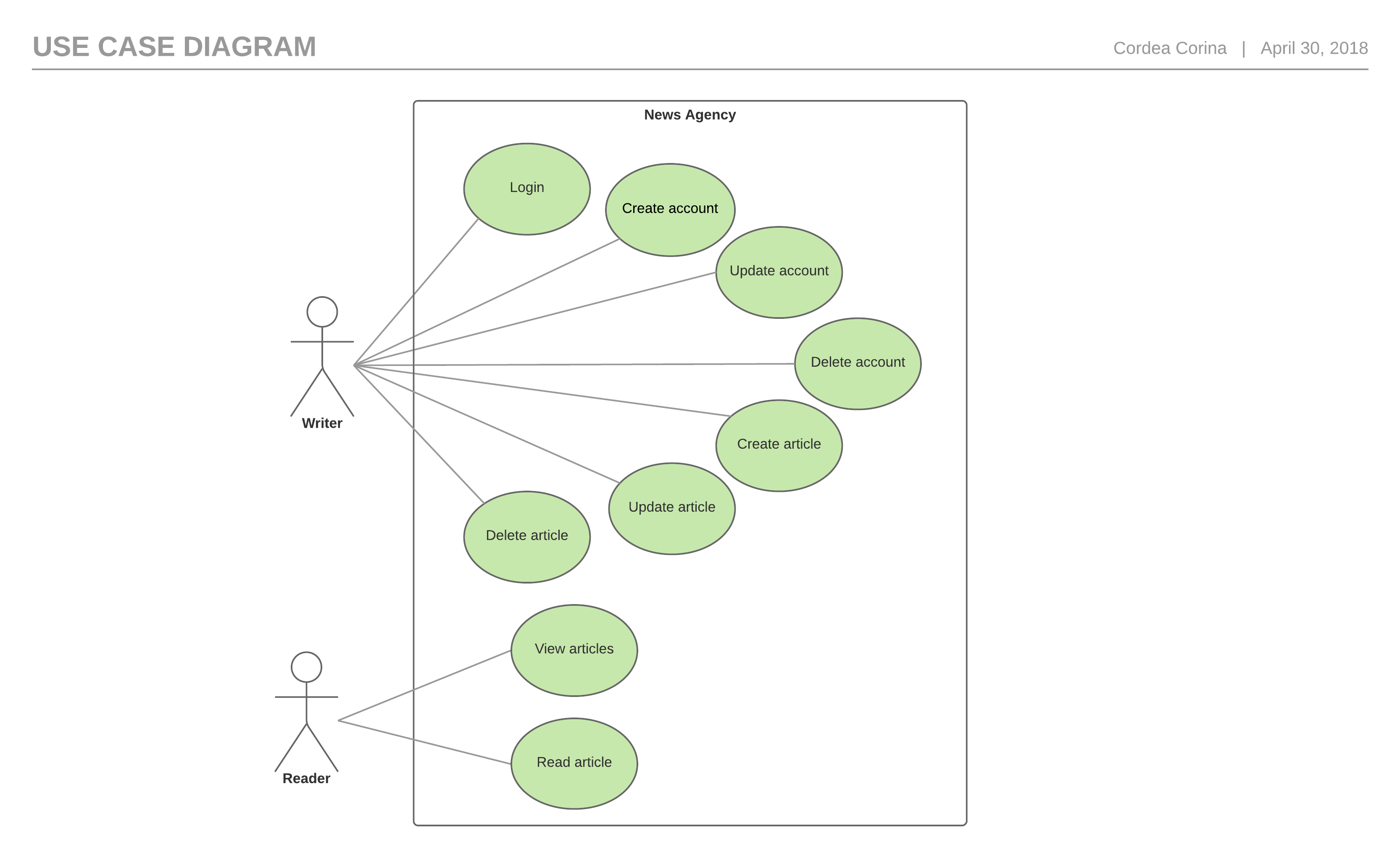
Writers can:

* Authenticate
* Create, update or delete article
* Create, update or delete account

# Non-functional Requirements

* Availability – the system should achieve 95% available time
* Security – only writers should be able to create or modify articles, readers can only view them
* The application must support multiple concurrent users
* Readers must view a new article in real time, without performing any refresh operation

2. Use-Case Model



Create article

Use case: generate reports for a particular period containing the activities performed by a student

Level: user-goal level

Primary actor: writer

Main success scenario: the writer opens the application, logs into the system, selects write article option, enters the title, abstract, author and body, presses a button then the article is created

Extensions: - the administrator opens the application, enters the login information wrong and receives an error message then enter it correctly and successfully logs in, selects write article option, enters the title, abstract, author and body, presses a button then the article is created

3. System Architectural Design

**3.1 Architectural Patterns 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.

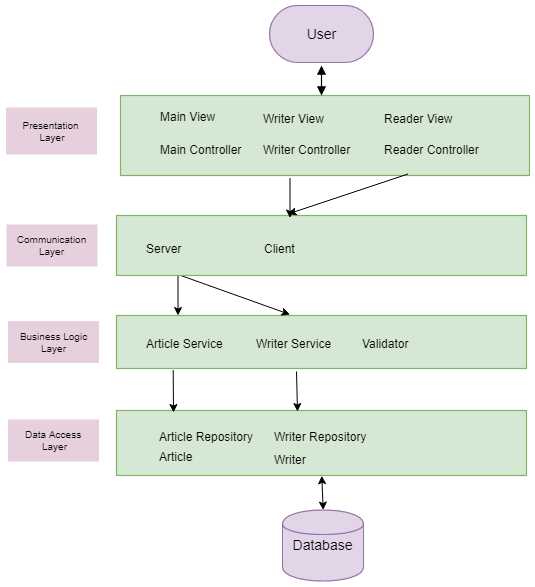
The Layers architectural pattern is used to structure the application by dividing it into groups of subtasks based on their functional responsibility. More specifically, the application is divided in 3 layers (bottom to top): data access, business logic and presentation layer. Each layer can only access the one beneath it. By using this pattern, the maintainability of the application and also the reusability of components are considerably increased.

Also, the MVC architectural pattern is used to separate application’s concerns: handle input, processing and output.

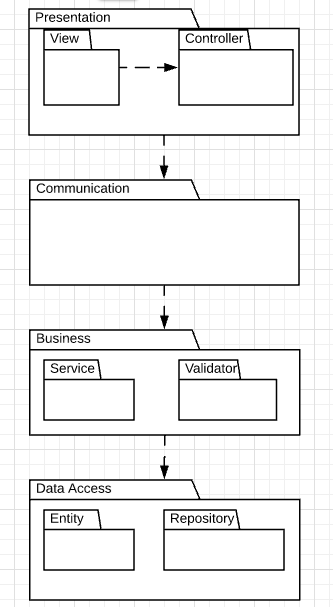
* + The model component encapsulates data and functionality (processing).
  + View components display information obtained from the model to the user (output).
  + Each view has an associated controller component that handles input.

**3.2 Diagrams**

* Conceptual architecture

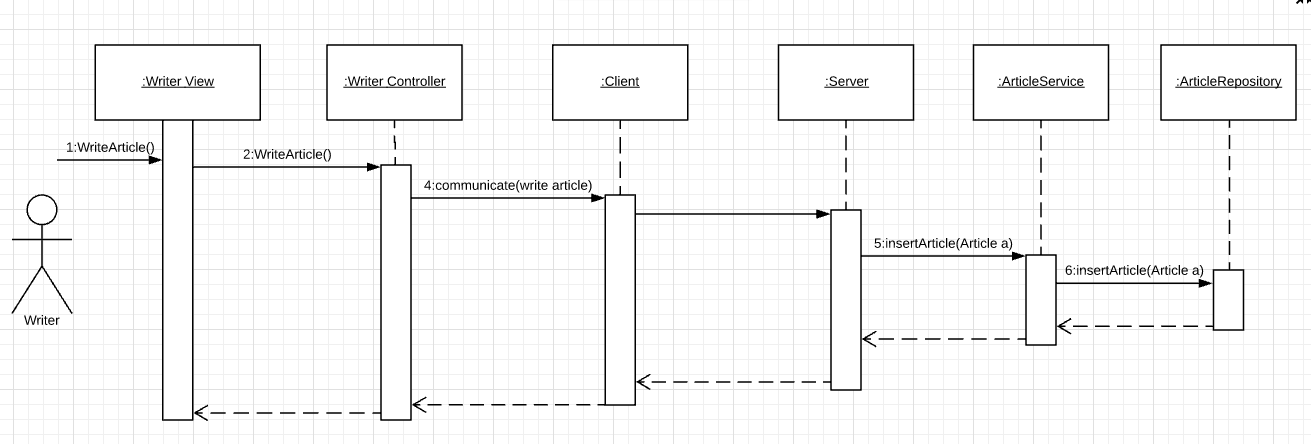
****

* Package

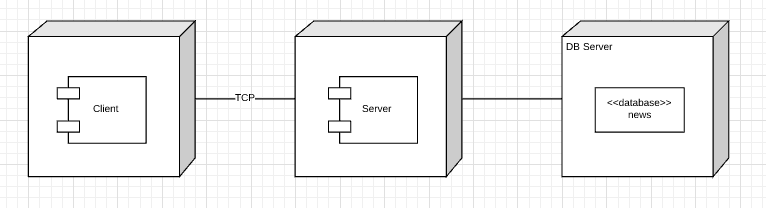


4. UML Sequence Diagrams

Sequence diagram – write an article



Deployment diagram



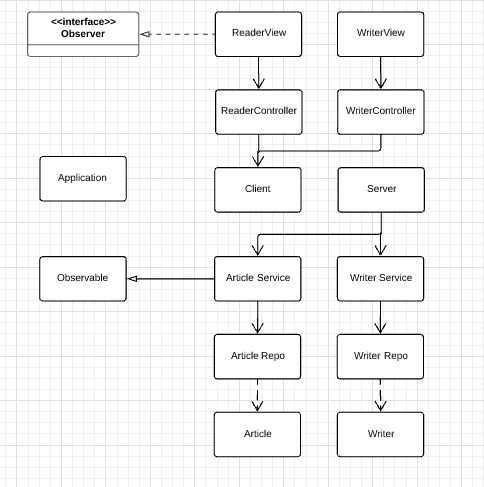
5. Class Design

**5.1 Design Patterns Description**

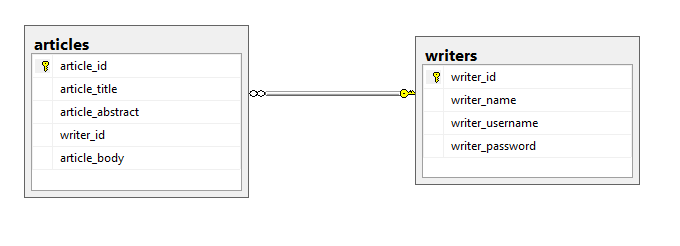
The **observer pattern** is a software design pattern in which an object, called the **subject**, maintains a list of its dependents, called **observers**, and notifies them automatically of any state changes, usually by calling one of their methods.

In this application, the readers are the observers, which will be notified when a new article appears. The subject is the list of articles of the news agency, which will be stored in the article service.

**5.2 UML Class Diagram**



6. Data Model



7. Bibliography

<http://users.utcluj.ro/~dinso/PS2018/Lectures/>

<https://msdn.microsoft.com/en-us/library/ff650706.aspx?f=255&MSPPError=-2147217396>

<https://en.wikipedia.org/wiki/Unified_Modeling_Language>

<http://www.waitingforcode.com/spring-framework/design-patterns-in-spring-framework-part-1/read>

<https://en.wikipedia.org/wiki/Observer_pattern>

<https://docs.oracle.com/javase/tutorial/networking/sockets/clientServer.html>