Assignment 3 - Library

Student: Eric Dumea

**Group: 30433**

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.

# This projects consists of a Java desktop application for a news agency. The application has two main users, the readers who can only read articles and the writers who can write articles. Additionally there exists an administrator who can perform CRUD on writer accounts.

# Functional Requirements

Some requirements include :

* Writers log in
* Readers acceess to all articles for reading
* Writers article creation
* Client – Server Architecture
* Json serialization for data sent between client and server

# Non-functional Requirements

Some requirements include :

- System requires java and in order to run the application

- Maximum response time of 5 seconds

- Availability of 2 hours per day

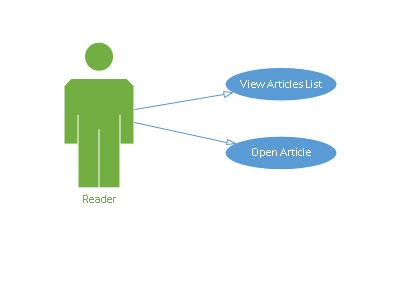
- Security provided by the email-password login

2. Use-Case Model

Use case: **Read Article**

Level: user-goal level

Primary actor: Reader

Main success scenario: open the application, frame with all articles pops up, select an article from the list, frame with whole article pops up

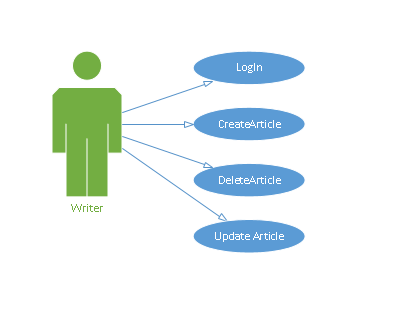
Use case: **Create Article**

Level: user-goal level

Primary actor: Writer

Main success scenario: open application, click the “Login” button, perform log in, provide article data, click “submit” button.

Extensions: wrong/missing information, and message pops up specifying the problem



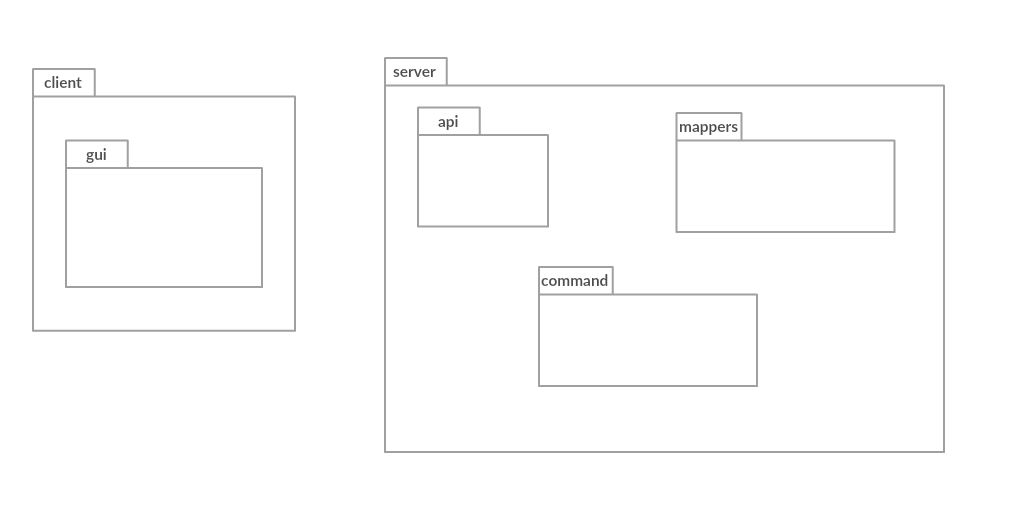
3. System Architectural Design

**3.1 Architectural Pattern Description**

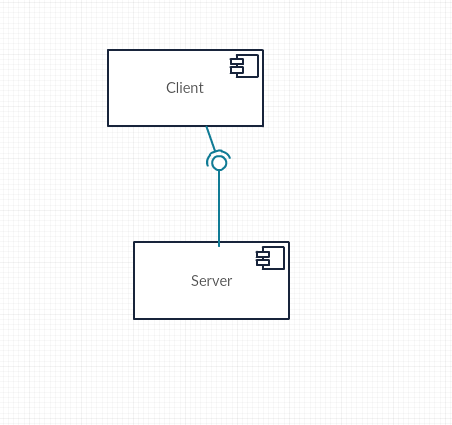
The **client–server model** is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host 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.[2]

**3.2 Diagrams**

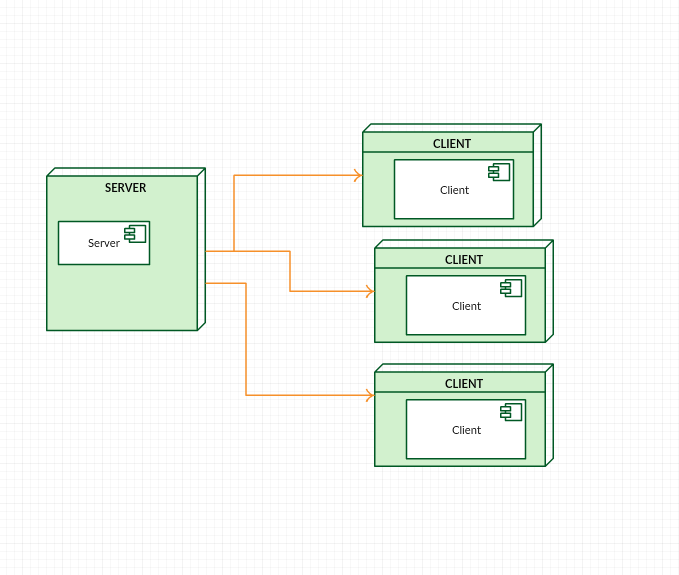
Package Diagram:



Component Diagram:

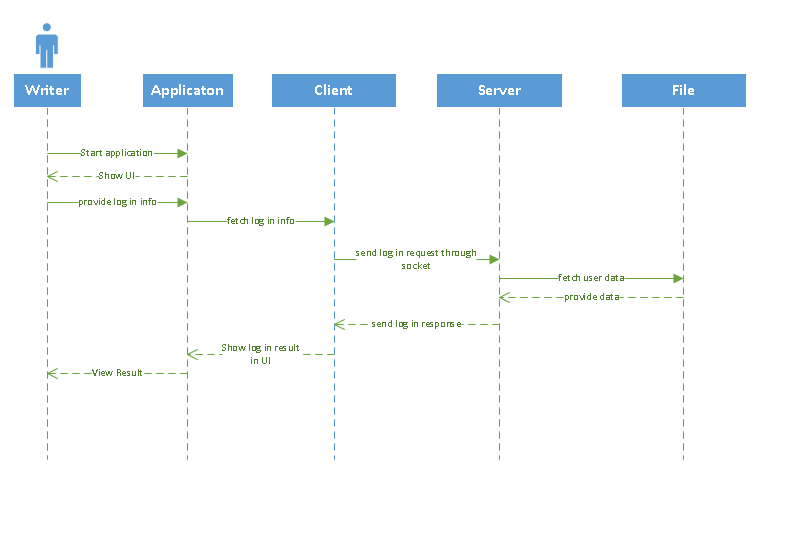


Deployment Diagram:



4. UML Sequence Diagrams

Sequence diagram for a writer log in



5. Class Design

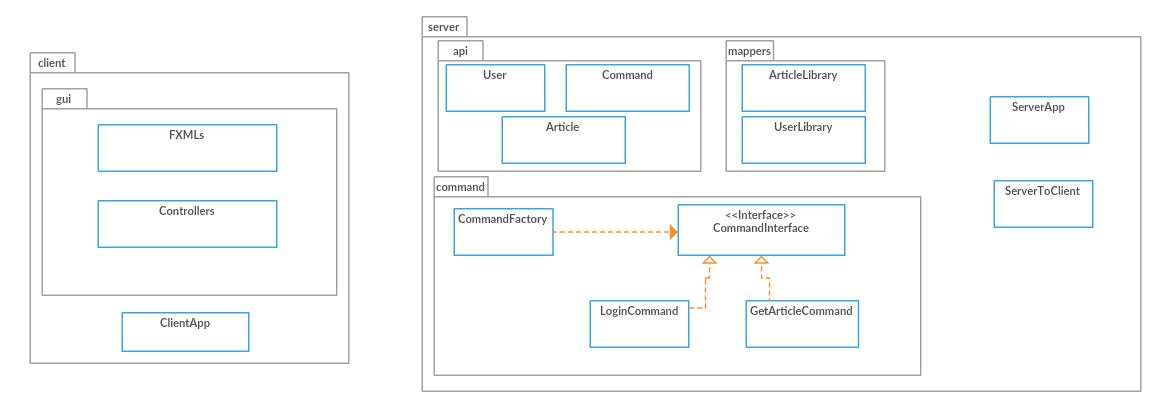
**5.1 Design Patterns Description**

**The Observer Pattern**

The observer design pattern enables a subscriber to register with and receive notifications from a provider. It is suitable for any scenario that requires push-based notification. The pattern defines a provider (also known as a subject or an observable) and zero, one, or more observers. Observers register with the provider, and whenever a predefined condition, event, or state change occurs, the provider automatically notifies all observers by calling one of their methods. In this method call, the provider can also provide current state information to observers.[1]

**5.2 UML Class Diagram**

The project has two modules. The data Models used for the application are contained in the ‚api’ package, which simulates an API. The Client module has the UI module, and the Server module has the Connection Module. A class is provided in the Server which serves the CRUD operations done by the application.



6. Data Model

The entity classes are constructed as follows:

User: email, password, name, admin right.

Article: title, author, abstract, body, list of related articles.

All the existing info is stored in a Library class which has a list of writers, articles and admins. Every time the application starts a json file containing the application data is deserialized so that the application can make use of the existing info.

7. System Testing

Testing was done using flags and console messages placed during development and constantly checking the Json file for updates. The System.out.println() method was used to check various functionalitites.

8. Bibliography

[1] - https://docs.microsoft.com/en-us/dotnet/standard/events/observer-design-pattern

[2] - https://en.wikipedia.org/wiki/Client–server\_model