Live Changes Are Visible By Others - Design Doc

**Status:** Under review

**Created:** Nov 03, 2022

**Last Updated:** Nov 08, 2022

**Authors:** Lior Mathan

**Approvers:**

**Additional Links:** [[Shared Doc req.pdf](https://drive.google.com/file/d/1fKQkn8He63isOvf0L3mgw0ppgdt_lz3o/view?usp=sharing), [Shared Documents - Design Doc](https://docs.google.com/document/d/1jd7uOGuQKWG18D8QNcbV64q2N91EqW6Y15Lc7c1MT7U/edit?resourcekey=0-41sj9qgW3AKXqmAXce7MQA#)]

# Background

As described in the application design document, this application allows several collaborators to update a single document simultaneously in an online editor.

For this requirement to work smoothly, the editor should always display the most updated content for all active collaborators.

In this document we will present the real-time document updates feature design.

More documentation:

* [Requirements - Live Changes Are Visible By Others](https://docs.google.com/spreadsheets/u/0/d/1HsdyZPh2FRAUv9G0ttrkR36XY15eAM2bUzGeNOxAXXY/edit)
* [Real-time document updates Feature - Use Cases](https://docs.google.com/document/u/0/d/1ADu0xRMM3eHu8I-54_z3ELSWYyAaH81NH4llW9QIT_E/edit)
* [User Story - live changes are visible by other users](https://docs.google.com/document/u/0/d/1P3eZV1JURN5HGEGjjo7s1T4tRJ4uyNfvPvJ2WqHSU4w/edit)
* [User Journey - User views changes made by collaborators](https://docs.google.com/document/d/1K5WNtfuuR-wHx0ZlAQke4PNJAtPWlC8F5gbmRnSwvUY/edit#)

# Objective

This feature should support the following key requirements:

1. Users with editing permissions can edit the document (typing in text / deleting text).

2. Everytime a user edits a document, it updates immediately.

3. The updated content of a document appears in all active editors in real-time, so that all authorized users of the document (viewers / editors / admin) can see the changes made immediately.

# 

# Overview

* Basic classes:
  + Document class - holds the document's metadata, authorized and active users, and the document’s content.
  + User class - holds user data and all documents linked to the user.
  + Editor class - allows updating and saving documents.
  + Content class - stores the document content.
* All document editing operations initiate in the editor class.
* Editor class uses the DocumentController, which uses the DocumentService for updating and saving content.
* Document class holds the list of authorized users and their permission.
* All document DB operations are carried out in DocumentRepository class.
* Permission is an enum which consists of the values: ADMIN, EDITOR, VIEWER.

# Alternative Solutions

The document can be updated using different design patterns.

The two alternatives that we will focus on:

1. Observer pattern - Observer is a behavioral design pattern that allows some objects to notify other objects about changes in their state.  
   It provides a way to react to events happening in other objects without coupling to their classes. Java offers a simple Observer pattern implementation
2. Add functionality inside setters / getters - the idea is to hide variables behind setter / getter pairs. Then, in the setter you can trigger a notification for all other collaborators that a change was made.

|  | **Observer pattern** | **Setters / getters additional functionality** |
| --- | --- | --- |
| **Performing actions** | Other editors with access to the updated document will be notified about the changes made in the file immediately. | Other editors with access to the updated document will be notified about the changes made in the file immediately. |
| **Efficiency during development** | The Observer design pattern can accelerate the development process. It provides a proven development paradigm, which helps save time without having to reinvent a solution. | The development is in the hands of the developer.  The developer must put thought into the design to ensure that the solution is maintainable and extensible. This is time consuming. |
| **Readability** | Standardization related to the design pattern is very useful to facilitate code readability. | Our invented code is less readable than a known design pattern solution. |

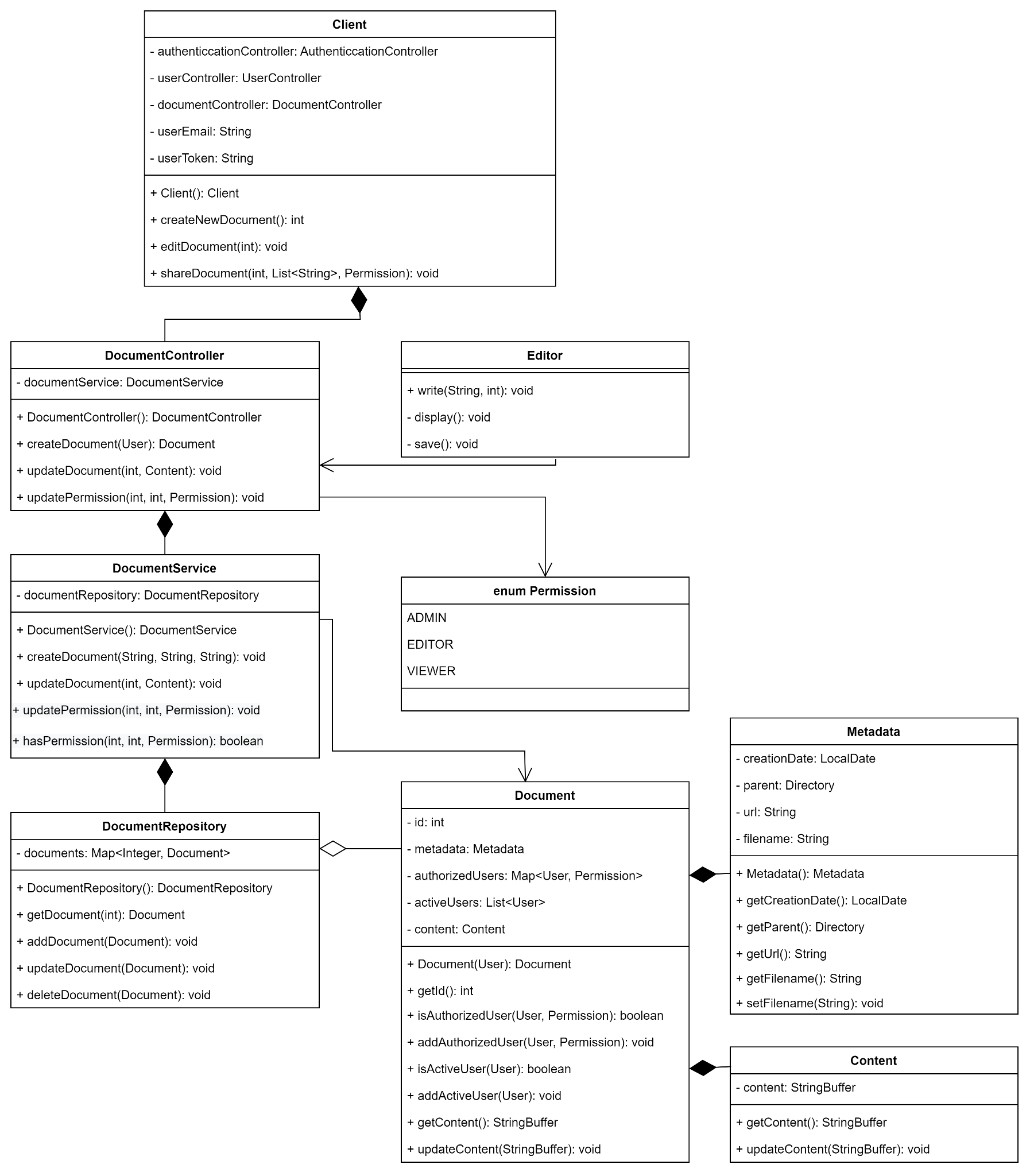
# The Bottom Line

In both solutions, the editors can be notified immediately. The Observer design pattern saves time and provides more readability while in the setters / getters solution we need to put thought into the solution which is time consuming, and it is also less readable.

Therefore, we chose to use the Observer design pattern.

# Detailed Design

## UML Diagram (class diagram) [Class diagram](https://app.diagrams.net/#G1gSshei52HFCJusNMyk2Ch_hkii2_84X3)

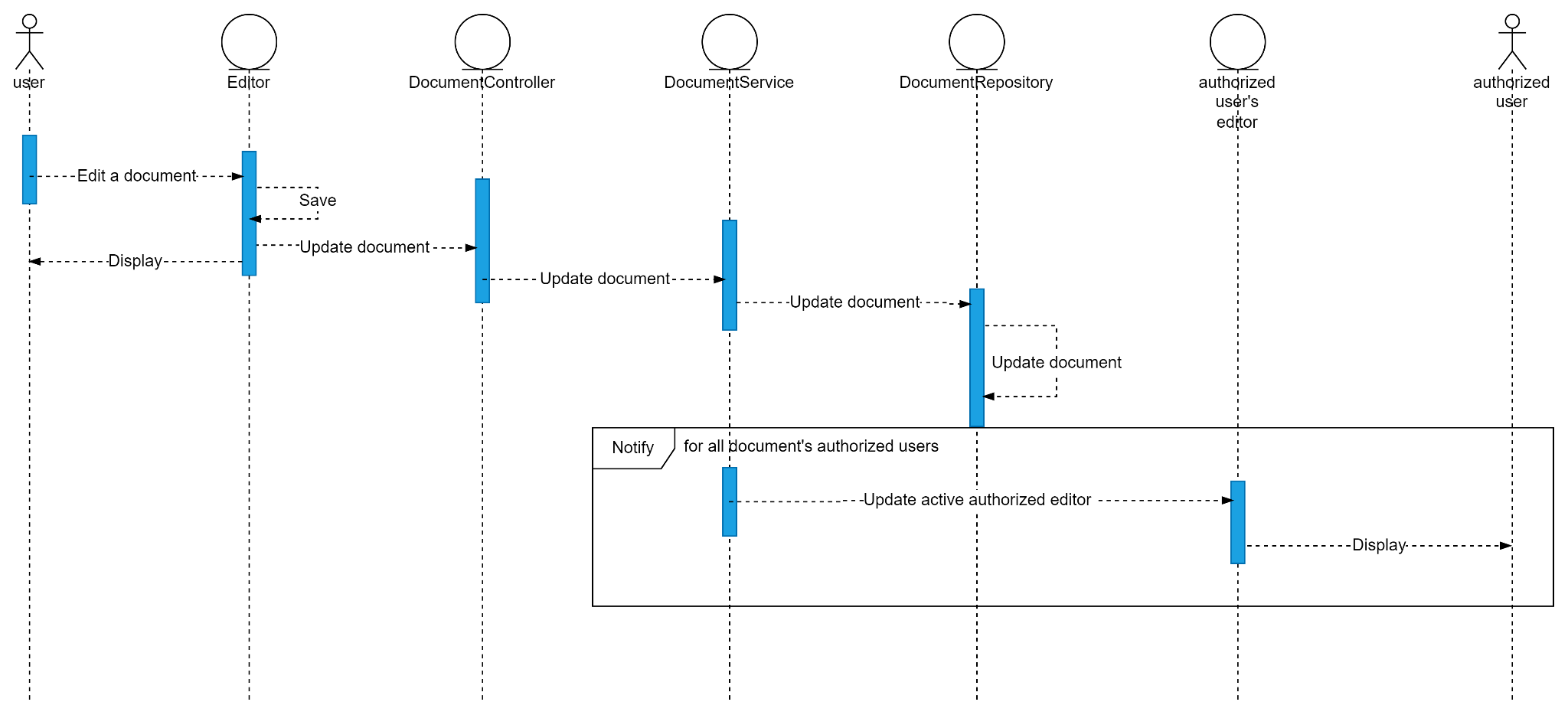


## 

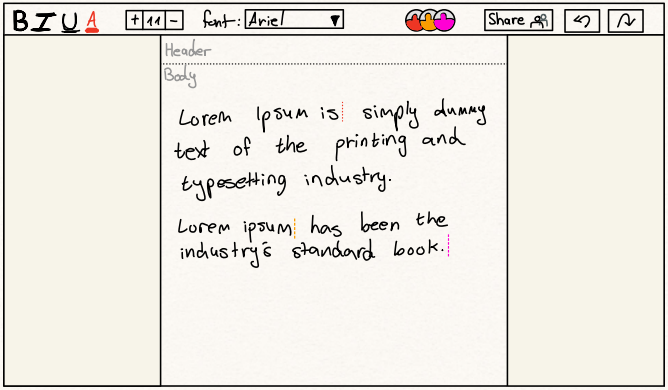
## 

## Sequence diagram

[link](https://app.diagrams.net/#G1oJJnM7VtfUCzwVON54YnyRrJd0ECNp8A)

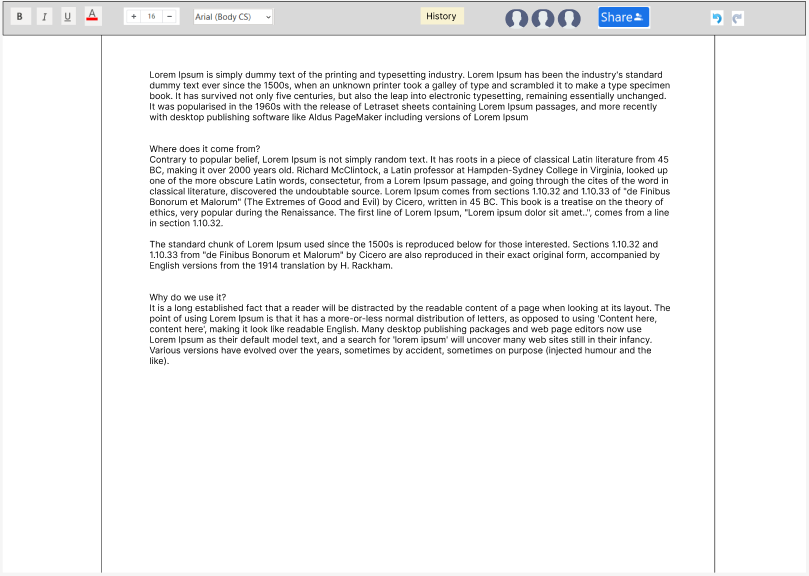


## Sketch



## 

## Mock-up



# Testing

[Testing - Live Changes Are Visible By Others](https://docs.google.com/spreadsheets/u/0/d/19XX6oIQXpvmBMvNP9Iq12x0lblAtC9mcv0DVHffz70E/edit)