Word Processor

Analysis and Design Document

Student: Iulian-Laurențiu Borșa

**Group: 30238**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[I. Project Specification 4](#_Toc5299550)

[II. Elaboration – Iteration 1.1 4](#_Toc5299551)

[1. Domain Model 4](#_Toc5299552)

[2. Architectural Design 4](#_Toc5299553)

[2.1 Conceptual Architecture 4](#_Toc5299554)

[2.2 Package Design 6](#_Toc5299555)

[2.3 Component and Deployment Diagrams 6](#_Toc5299556)

[III. Elaboration – Iteration 1.2 7](#_Toc5299557)

[1. Design Model 7](#_Toc5299558)

[1.1 Dynamic Behavior 7](#_Toc5299559)

[1.2 Class Design 7](#_Toc5299560)

[2. Data Model 7](#_Toc5299561)

[3. Unit Testing 7](#_Toc5299562)

[IV. Elaboration – Iteration 2 7](#_Toc5299563)

[1. Architectural Design Refinement 7](#_Toc5299564)

[2. Design Model Refinement 7](#_Toc5299565)

[*[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]* 7](#_Toc5299566)

[V. Construction and Transition 8](#_Toc5299567)

[1. System Testing 8](#_Toc5299568)

[2. Future improvements 8](#_Toc5299569)

[VI. Bibliography 8](#_Toc5299570)

# Project Specification

A word processor is software application which allows users to create, edit, and save documents. It enables you to write text, store it electronically on a cloud, display it on an interface and modify entering or deleting characters from the keyboard.

Word processing is the most common application for computers. Today, most word processors are delivered either as a cloud service (e.g. Google Docs) or as software that users can install on a PC or other device (e.g. Microsoft Office Word).

# Elaboration – Iteration 1.1

# Domain Model

A domain model is a system of abstractions that describes selected aspects of a sphere of knowledge, influence or activity. The model can then be used to solve problems related to that domain. The domain model is a representation of meaningful real-world concepts pertinent to the domain that need to be modeled in software. The concepts include the data involved in the business and rules the business uses in relation to that data.

A domain model generally uses the vocabulary of the domain, thus allowing a representation of the model to be communicated to non-technical stakeholders. It should not refer to any technical implementations such as databases or software components that are being designed.

* an **user** is able to connect to the **platform** (application)
* the **platform** is linked to a **cloud**
* on the **cloud**, all **users** which are connected to the **platform** can access the **documents**, but not all of them
* there are two types of **documents**: **public documents** and **private documents**
* everybody can view **public documents**
* **private documents** can be accessed just by an **user** with permissions

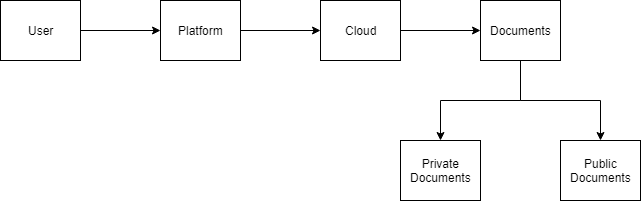


Figure 2.1 - Domain Model Diagram

# Architectural Design

## Conceptual Architecture

The intent of conceptual architecture is to direct attention at an appropriate decomposition of the system without delving into the details of interface specification. I chose a layered architecture pattern because it is easy to testing components, it’s a common pattern used for general desktop applications.

There are 3 big layers:

* **Presentation Layer** doesn’t need to know or worry about how to get customer data

it is used just to display information on a Graphical User Interface

* **Business Logic Layer** doesn’t need to be concerned about how to format customer data for display on a screen or even where the customer data is coming from

it is used just to get the data from the persistence layer, perform business logic against the data and pass that information up to the presentation layer

* **Persistence Layer** communicates with Database

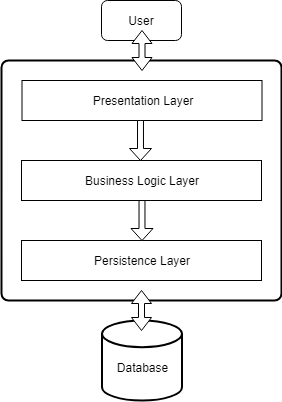


Figure 2.2.1 - Architectural Pattern

## Package Design

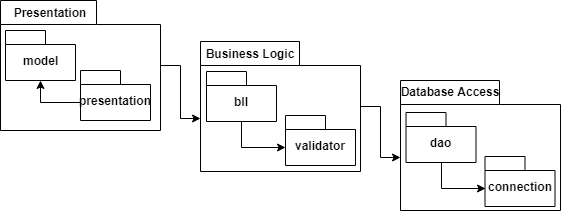


Figure 2.2.2 - Package Diagram

## Component and Deployment Diagrams

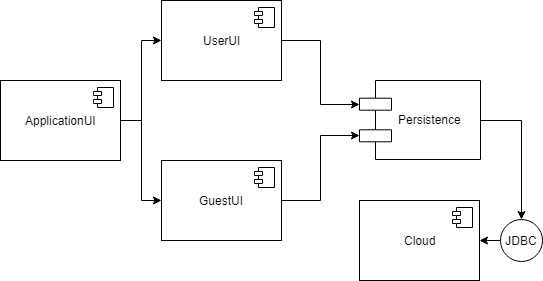


Figure 2.2.3 - Component and Deployment Diagram

# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

*[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]*

## Class Design

*[Create the UML class diagram; apply GoF patterns and motivate your choice]*

# Data Model

*[Create the data model for the system.]*

# Unit Testing

*[Present the used testing methods and the associated test case scenarios.]*

# Elaboration – Iteration 2

# Architectural Design Refinement

*[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]*

# Design Model Refinement

## *[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]*

# Construction and Transition

# System Testing

I’m using **Graphical User Interface Testing**, who is the process of testing a product graphical interface. This is done through the use of a set of test case scenarios. To generate a set of test cases, I will try to cover all the functionality of the system and verify entire GUI system, first as normal user, and than as guest.

I can use **Unit Tests** to verify individual methods and classes. This way of testing is good for Business Logic Level and Data Access Objects.

# Future improvements

The application could be modified for online users. So, they could access documents and modify them in real time.

# Bibliography

<http://www.bredemeyer.com/ArchitectingProcess/ConceptualArchitecture.htm>

<https://simplicable.com/new/conceptual-architecture-vs-physical-architecture>

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

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

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