Assignment 3

Analysis and Design Document

Student:Popovici Marius

**Group:30238**

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 for this application is to build a deal search engine for furniture products using an Object-Oriented Language. An user should be able to create an account ang login to search for various provided deals. Deals are managed by staff and can be filtered by price, name and type. If a deal is available, users can add the associated product to their cart and proceed to checkout.

Payments can be done via a cash only policy and need to be validated by staff. Tihs creates an order in the system that can be tracked by the user from the Order History section. The state of an order is updated by staff. Once an order is delivered, the user can provide feedback in a form, in the specific Order History entry details.

# Functional Requirements

*[Present the functional requirements]*

The functinal requirements of this application are:

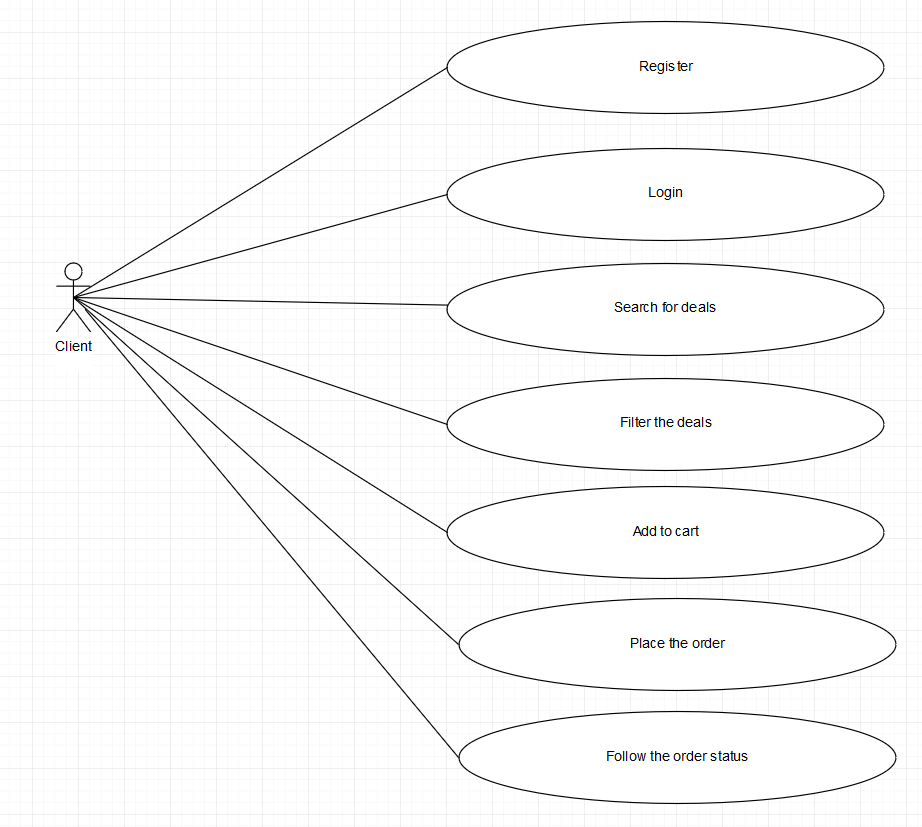
* Register – users can create an account that will be used later
* Login – User can access the application data using the chosen username and password registered before
* Search for deals and filter them– They are able to filter the product list by the type of the sale.
* Order – after products were added in cart, they can proceed to checkout.
* Order history – clients can view the status and details of their orders

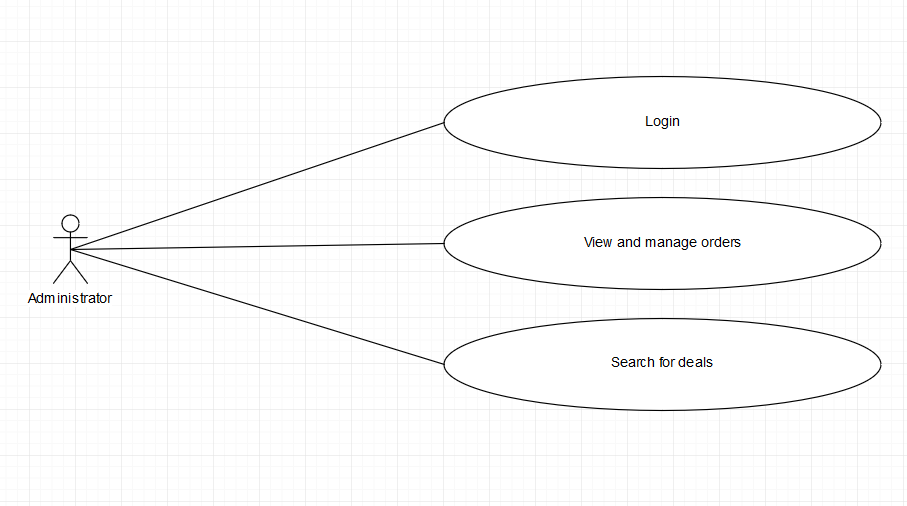
# Non-functional Requirements

*[Discuss the non-functional requirements for the system]*

* User interface – system should provide a graphical user interface for offering to clients the possibility of interracting with the app.
* Documentation – clients should be taught how to use the app
* Reliability – the system should keep all the information confidentially and the access to it should be done just through authentification
* Flexibility – the system should be flexible to changes
* Security – the clients should login before having the possibility of using the application
* Data validation – all entered data should be verified
* Error handling – exceptions should be caught and the client should be notified when one happens.

2. Use-Case Model





**Use case description:**

**Login:**

Use case goal: description of how a client can log into the system of Furniture Deals management.

Main success scenario:

1. When accessing the first page, a client is requested to login or sign up.
2. The client should press the “login” button
3. A new page will be loaded, where the client should fill in his credentials.
4. If the username and password are found in the database, the client will be logged into the system and redirected to the main page where are listed the disponible offers.

Alternate scenarios:

If a client enters incorrect data, he/she is notified and asked to enter valid data.

3. System Architectural Design

**3.1 Architectural Pattern Description**

This project is implemented using the Django framework (written for Python), so the used architecture is client-server.

Along that, the project was developed using CQRS architecture pattern. The responsability for accessing the database are split to two types of queries: for reading from database and for writing in database.

**3.2 Diagrams**

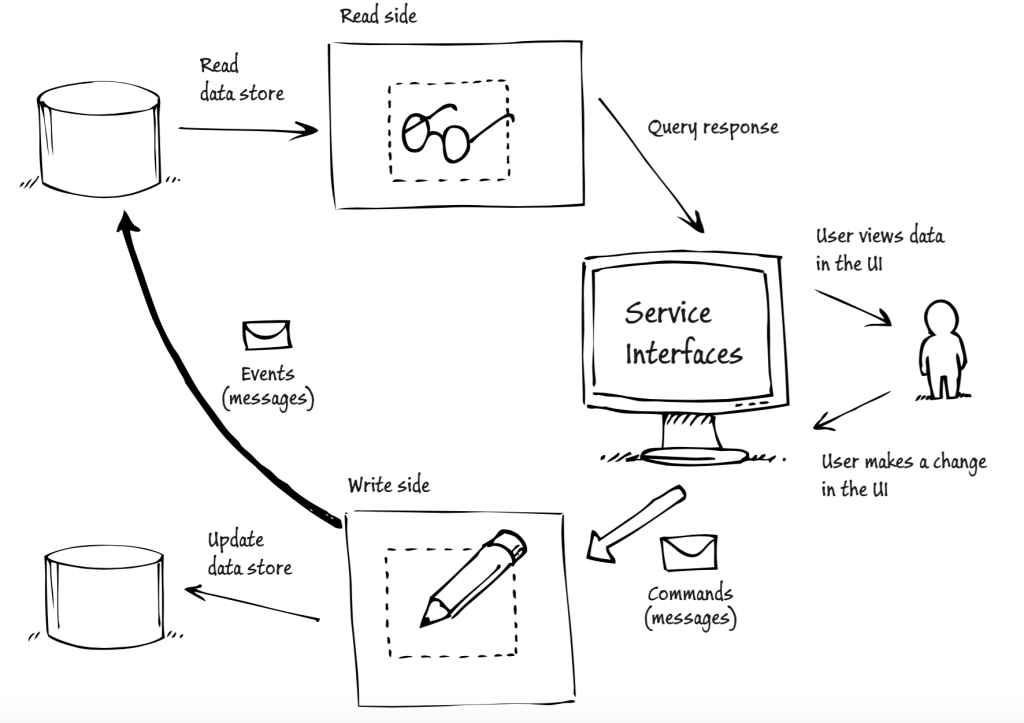


CQRS, which means Command Query Responsibility Segregation, comes from CQS (Command Query Separation) introduced by Bertrand Meyer in Object Oriented Software Construction. Meyer states that every method should be either a query or a command.

The difference between CQS and CQRS is that every CQRS object is divided in two objects: one for the query and one for the command.

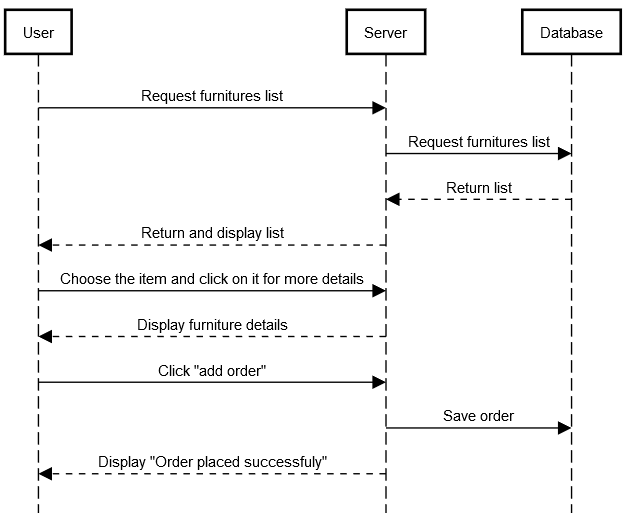
A command is defined as a method that changes state. On the contrary, a query only returns a value.

The following schema shows a basic implementation of the CQRS pattern inside an application. All messages are sent through commands and events.



1. UML Sequence Diagrams

Order a furniture item.



5. Class Design

The structure of the code is as follows: an “app” is created for every structure in the database. Example: accounts, furnitures, orders.

The main application is called “assign2”. All the other apps should be registered in this main app. It is the only class with a settings file.

An application contains some components as: urls, views, models, templates (for rendering html templates).

The urls file contains the reference to the implementation of the function which will be triggered when accesing a link. The implementation itself is done in views.

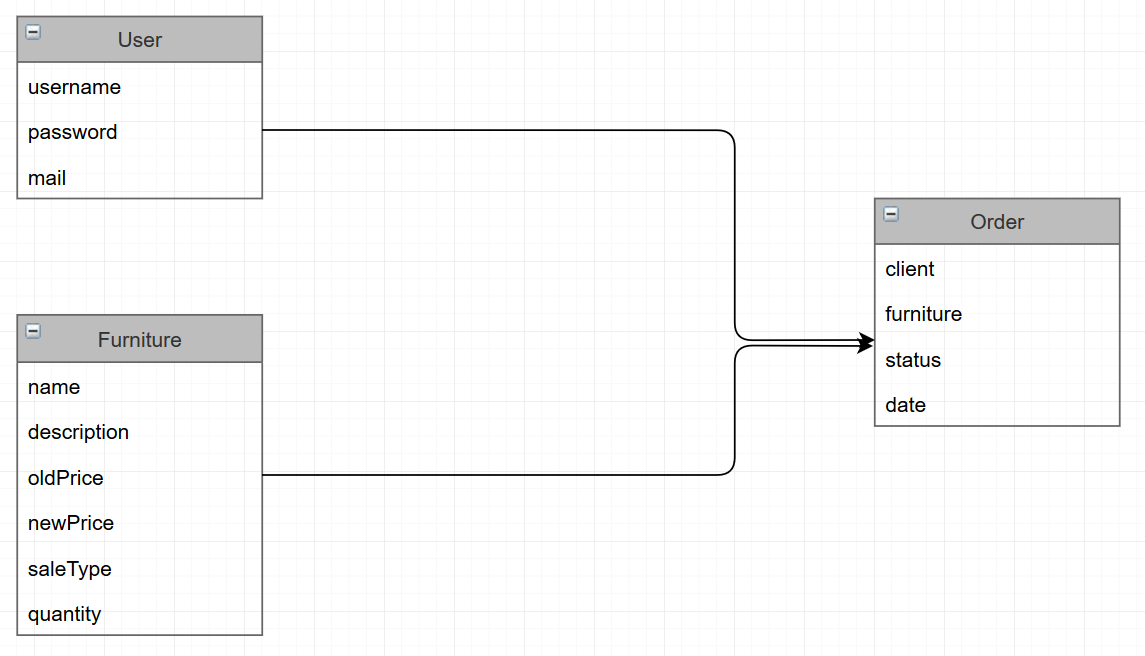
The model component is the “reflection” in code of the corresponding database table (because Django uses Active-Record Pattern).

For notifying the clients, I used observer design pattern. By using this pattern, clients will be messaged, by mail, that their command status has been modified.

For giving the client an opportunity of choosing a bonus (present) for ordering from my shop, I used the decorator design pattern.

6. Data Model

Data is contained into three tables: one for users, one for furnitures and one for orders.

**

7. System Testing

Being a web application, the functionality is tested using the application itself.

8. Bibliography

<https://www.djangoproject.com/>

<https://www.youtube.com/playlist?list=PL4cUxeGkcC9ib4HsrXEYpQnTOTZE1x0uc>