Wasteless A3

Analysis and Design Document

Student:Sas Cosmin Andrei

**Group:30431**

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

Design an application that helps users manage their food and minimize food waste. The user should be able to see all his food and each item will be colored red if it’s expired or green if it’s good.

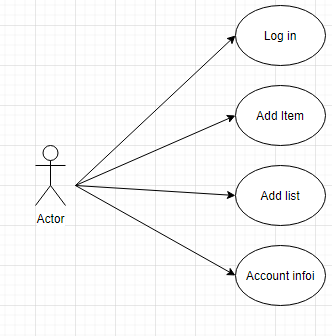
# Functional Requirements

*Once a user is authenticated he can input grocery lists and items. He can list all his items and request information about his account.*

# Non-functional Requirements

* *Implement and test the application*
* *Commit the work you do on your Git repository. Do it iteratively as you progress, not all at once (this will incur a penalty on your final mark)*
* *Use any OOP language you like. Non-exhaustive: Python, C#, Java, Ruby, C/C++, JS+Typescript*
* *Use a CQRS architecture, use a mediator pattern to handle requests*
* *Use a decorator pattern for changing the color of the report (green for above the ideal*
* *rate and red for under)*
* *The data will be stored in a database*
* *All the inputs of the application will be validated against invalid data before submitting the data and saving it in the database.*

2. Use-Case Model



Use case: Show food

Level: user-goal level

Primary actor: User

Main success scenario:

*1.Check if user is logged in*

*2.Request server for food list*

*3.Receive food list*

*4.Display food list for the user*

Extensions:

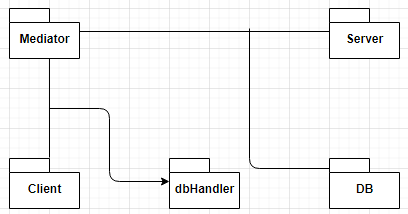
3. System Architectural Design

**3.1 Architectural Pattern Description**

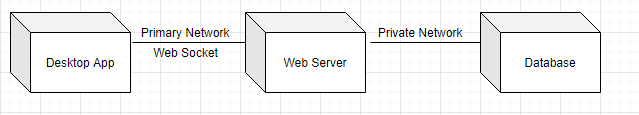
*For this project I was required to use the Client Server Architecture. Client–server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate 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 it requests content or service from a server. Clients, therefore, initiate communication sessions with servers, which await incoming requests.*

**3.2 Diagrams**

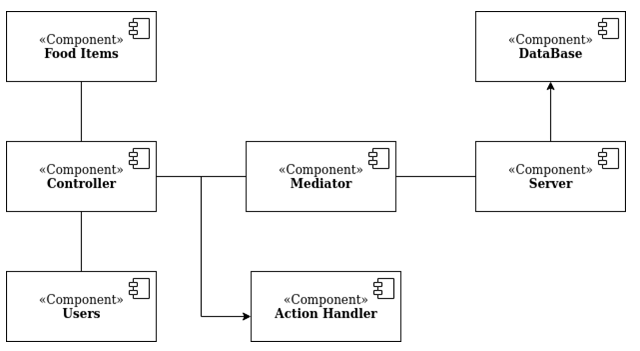
*Package Diagram*



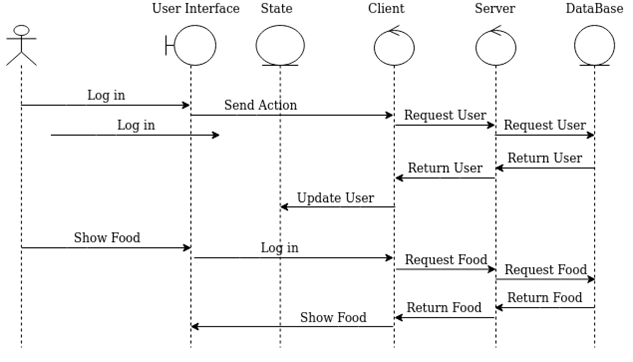
*Deployment Diagram*



*Component Diagram*

**

4. UML Sequence Diagrams

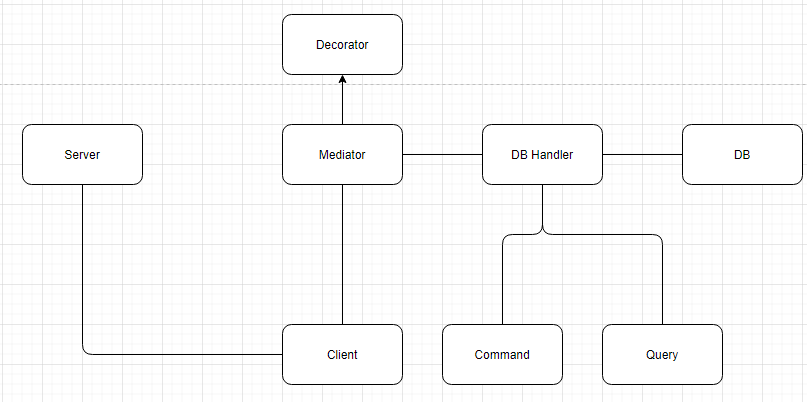
**

5. Class Design

**5.1 Design Patterns Description**

*The mediator pattern defines an object that encapsulates how a set of objects interact. Objects will no longer communicate directly. They will communicate only with the mediator. This reduces the dependencies between communicating objects, thereby reducing coupling.*

**5.2 UML Class Diagram**

**

6. Data Model

*Profile: name(char), password(char), calories(int)*

*List: user, name(char), date(datetime)*

*FoodItem: list, name(char), expiration date(datetime), calories(int)*

7. System Testing

*Each component has been tested individually and after assuring functionality they have been put together. After that, data-flow testing was done to ensure that all components function as expected.*

8. Bibliography

Youtube – sentdex – for sockets