Wasteless 3

Student:Cioban Dumitru-Darius

**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

The application provides the option of a user to manage their food waste, by tracking it using this application, by tracking how food is wasted, and providing notifications when food waste is imminent, giving the user the option to donate.

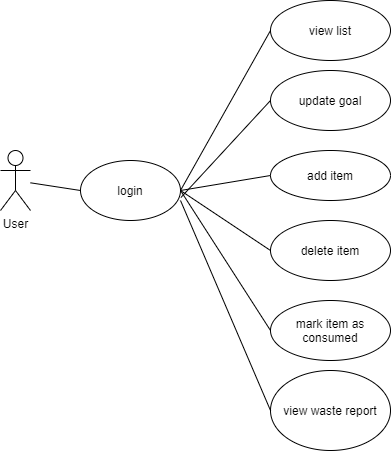
# Functional Requirements

*The functional requirements are adding and viewing the grocery list, viewing reports regarding the food waste, and notifications providing the donation option.*

# Non-functional Requirements

*The non-functional requirements are using a CQRS architecture, using a mediator and a decorator pattern and a database to store the data.*

2. Use-Case Model

**

*Use case: add item*

*Level: user-goal level*

*Primary actor: user*

*Main success scenario: The user inputs the data of the new grocery item they want to add. It passes through the validator, and if the data is valid, a new item is added to their list.*

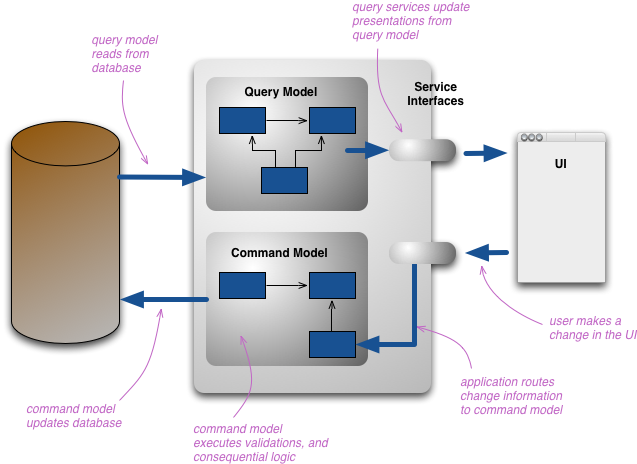
*Extensions: The validator returns an invalid value, signifying that the data is not according to the standard. An error message is displayed, giving info on how the data should look like. Another possibility is the case where the item wasn’t successfully added to the database, in which case the update won’t take place and nothing will be changed.*

3. System Architectural Design

**3.1 Architectural Pattern Description**

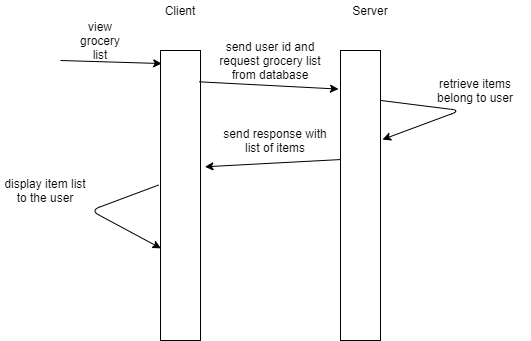
*The architectural pattern used is the CQRS architecture. It is implemented on base of the client-server architecture, and what it does is that it splits all requests into commands (which update the database) and queries (which query the database).*

**3.2 Diagrams**



*It can be seen from the above diagram (taken from the web) that the user, through the UI, uses the available query and command models to work on the database. This improves the security of this item, as for example, in the case of this system and the add item command, its model doesn’t have a consumption date (not needed) and also no id (it will be auto-instantiated by the database and shouldn’t be a choice), while the response only returns the id of the item (the rest of the information can be obtained by querying an item with that id).*

4. UML Sequence Diagrams

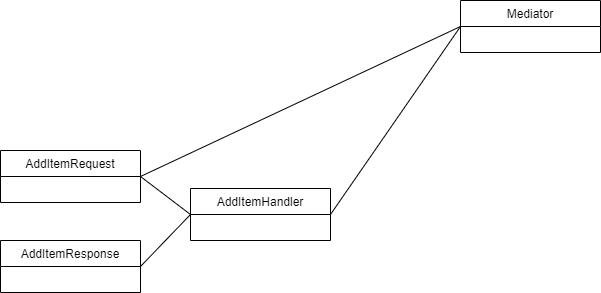
**

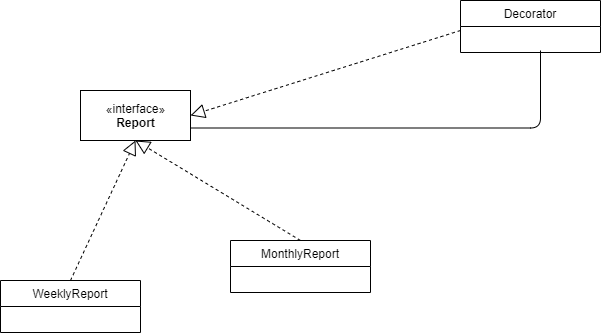
5. Class Design

**5.1 Design Patterns Description**

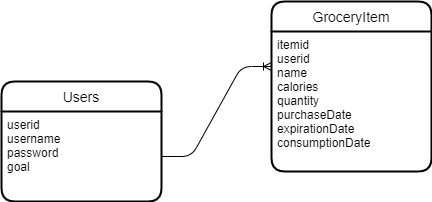
*The used designs patterns are the Mediator and the Decorator patterns. The mediator one is used to handle more easily the communication between the client and the server. The way it works it that it has a getHandler(request) method which, based on the request passed as parameter, it returns the handler which handles the requests and responses of that command/query. The decorator pattern wraps a class, in this case the report, and acts as one. That way, the decorator can then be used to access that object’s various methods.*

**5.2 UML Class Diagram**

*This is the diagram for the mediator pattern, in the case of the add item command. The handler receives a request and returns a response, and the mediator receivs a request and returns the handler.*

  
  
*This is the diagram for the decorator pattern. It can be noted that it wraps a Report object (either weekly or monthly) and also implements the Report interface, to be able to act like a Report.*

6. Data Model

**

*The data is represented by 2 tables. One for the users, where their id is the primary key. Username and password are used for authentication and goal for functionality. The grocery item table has an item id as primary key, an user id (establishes a one to many relationship between users and items), a name, calorie count, quantity and 3 dates representing the purchase date, the expiration date and the consumption date.*

7. System Testing

*The testing was done by feeding the system both valid and invalid data, seeing how it reacts. However, there are some scenarios which haven’t been tested or have been tested but haven’t been updated (for example the need of inputing only numbers in number fields).*

8. Bibliography