<WasteLess>
Analysis and Design Document

Student: Manea Luca
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 6

5. Class Design 6

6. Data Model 7

7. System Testing 7

8. Bibliography 7

1. Requirements Analysis

1.1 Assignment Specification

Nowadays, earth is facing a more and more dangerous situation regarding its resources, especially food. A main factor for the lack of food is its poor managing when buyed. Design and implement an application that helps users manage food waste.

1.2 Functional Requirements

Once a user is authenticated he can input grocery lists and see reports of how much food is wasted weekly and monthly. A grocery list item has a name and a quantity as well as a calorie value, purchase date, expiration date and consumption date.

The system also allows users to track goals and minimize waste by sending reminders if waste levels are too high based on ideal burndown rates.

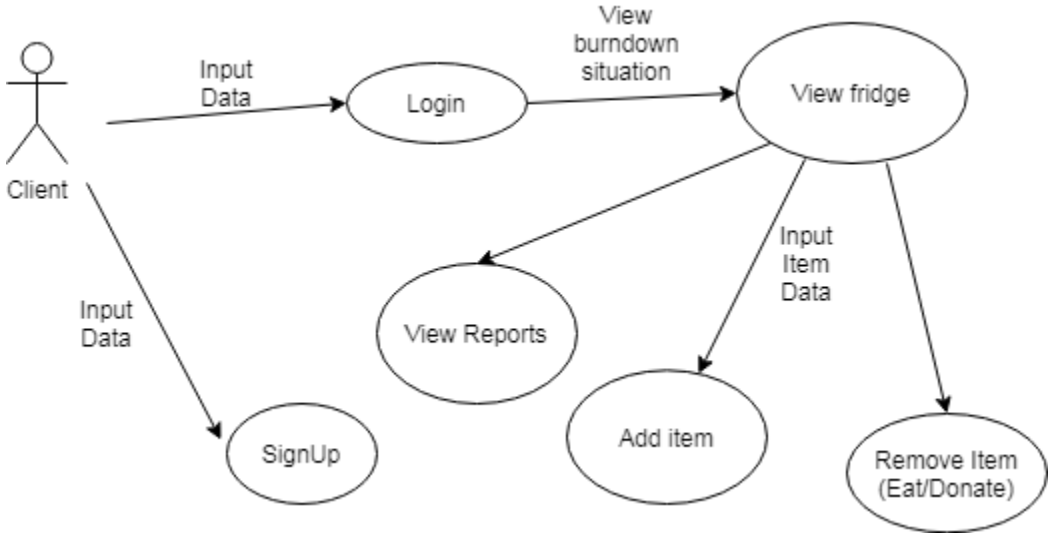
The ideal burndown rate for 100 calories worth of groceries due to expire in 5 days is 20 calories worth of groceries per day.

The system should provide you with options to donate excess food to various local food charities and soup kitchens and notify you of them prior to item expiration.

1.3 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: Sign-up
Level: user-goal
Primary actor: client
Main success scenario:

- The user inputs a username and a password
- The username is checked against validation rules
- The username is sent to the database through the command line where it is checked to be unique
- The user is now registered

Extensions:

- If the password is shorter than 8 characters/ username than 5 characters the operation fails
- A number is sent back to the client which encodes the possible outcomes(registered, not unique, database fail)

Use case: Login
Level: sub-function
Primary actor: client
Main success scenario:

- The user inputs a username and a password
- They are sent through the query bus
- The password is checked with the database entry corresponding to the username
- The user logs in
- The login query return the user's id
- Another query retrieves the user's items based on the id
- The user views his groceries

Extensions:

- The user views today's optimal burndown rate

Use case: Add item
Level: user-goal
Primary actor: client
Main success scenario:

- The user inputs data related to a product
- The data is validated
- Data is sent through the command query
- The object is added
- The fridge items are refreshed

Extensions:

- The user's view of the fridge is updated with the addition of the new object

Use case: View Reports
Level: sub-function
Primary actor: client
Main success scenario:

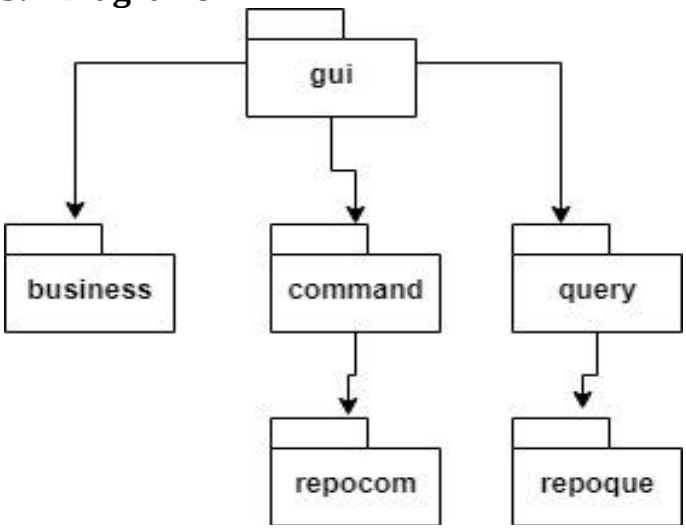
- The user is displayed a new window with expired items colored in red and edible ones in green

3. System Architectural Design

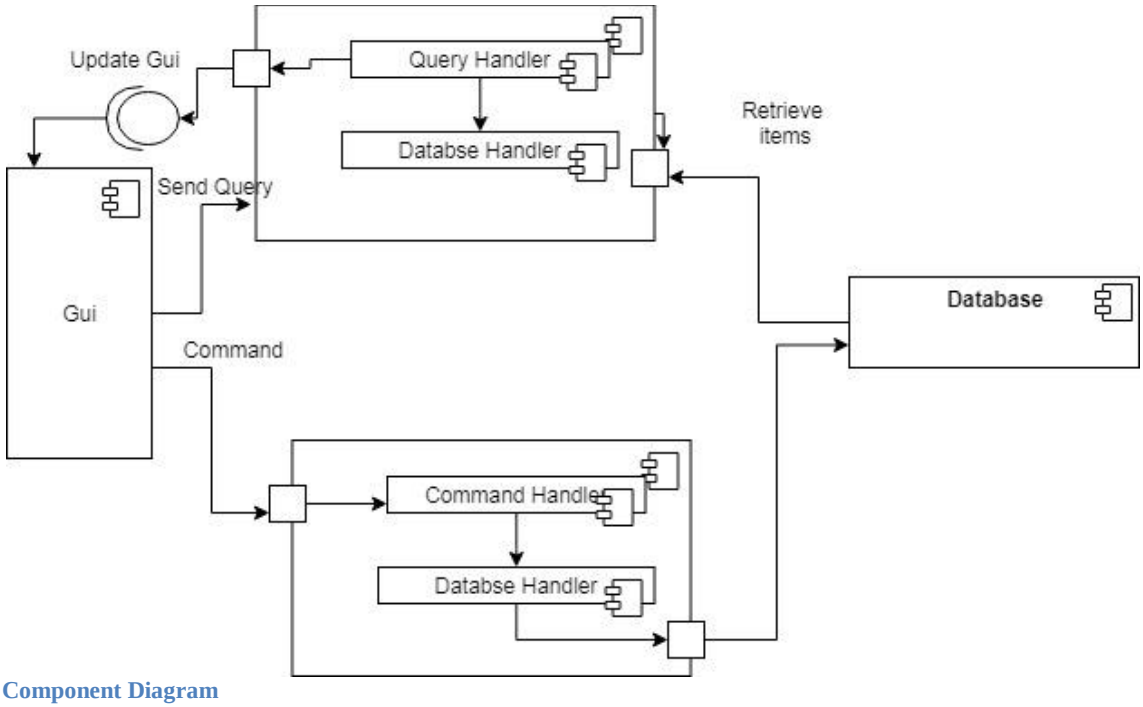
3.1 Architectural Pattern Description

The used architectural pattern is the layered one. Due to the simplicity of the application I used only three layers, which are reflected in the packages: Presentation layer, the business layer and the repository layer.

3.2 Diagrams

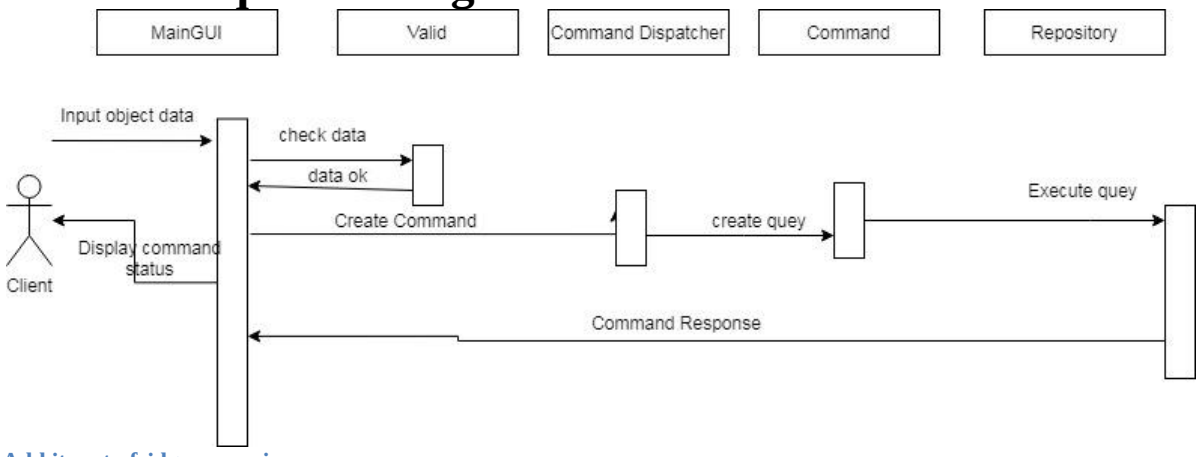


Package Diagram



Component Diagram

4. UML Sequence Diagrams



Add item to fridge scenario

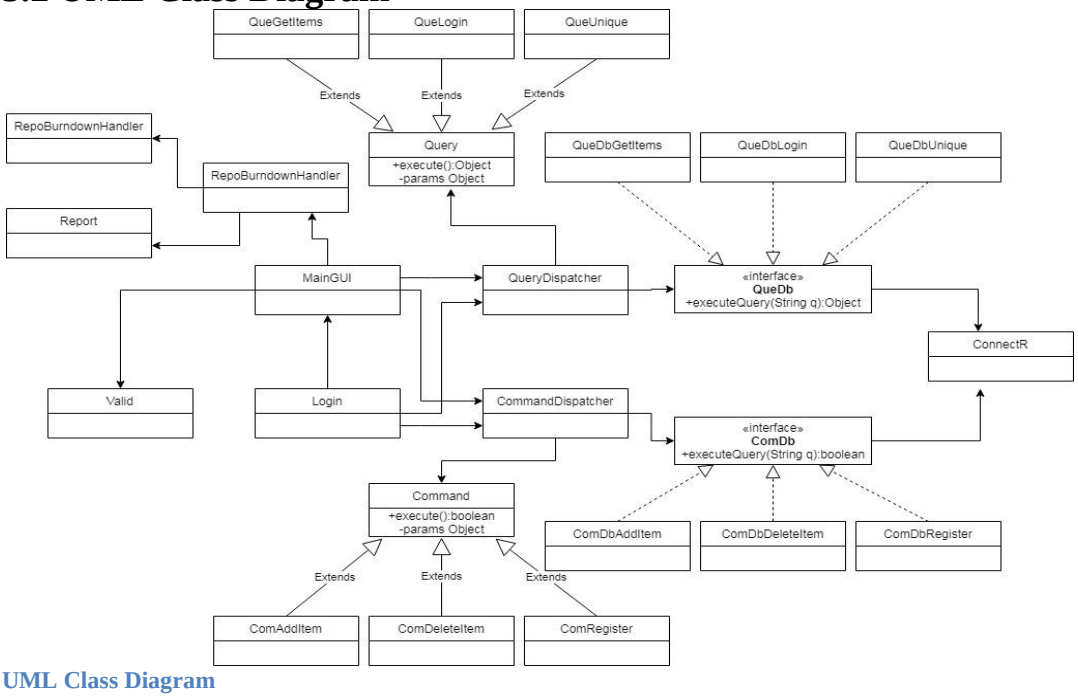
5. Class Design

5.1 Design Patterns Description

I used the Decorator pattern wrapped around a JTable class to create a colored table indicating expired and non-expired products. This was surprisingly hard and a trick was needed to solve the problem, more details in the code, Report class.

I used the mediator concept to implement a dispatcher pattern. The command and query dispatchers match the query/commands they receive with the appropriate "handlers", the database handlers. This improved the readability of the code and made it easier to maintain the command and query busses.

5.2 UML Class Diagram



UML Class Diagram

The general design of the application is based on the CQRS architecture . This implies the separation of the command and query busses, which can be observed in the UML diagram. The business of the application is mainly

directed from the GUI and resolved along the busses. There is an auxiliary class ReportBurndown which creates the Reports.

The Server connects to the database using the Mysql connector library.

6. Data Model

The data models used for this project were:

- The client model:
  - ID
  - Username
  - Password
- The Grocery Item model:
  - ID
  - Id_client
  - name
  - quantity
  - kcal
  - buy date
  - expiration date

7. System Testing

The main developing tool used in testing was printing to the system console. Classes and functions with specific tasks underwent unit testing . The testing also followed the flow of the user experience, as the application was developed following the same flow.

As the classes and methods regarding user login/register were completed I applied validation testing to check their correctness. The same was done with testing the second step of the application, the Fridge operations. Basically, two integration epochs took place: Login and Main one. After both proved successful, one validation step was done to check the whole app.

As the data sample used was not very large and it is not the scope of this application to achieve large data samples, partitioning and boundary analysis were not used.

8. Bibliography

1. https://stackoverflow.com/questions/57963474/what-are-the-differences-between-the-command-dispatcher-and-mediator-design-patt/58074565#58074565?newreg=5e72055bea14425cb3ba077c656bc613
2. https://en.wikipedia.org/wiki/Mediator_pattern
3. https://kariera.future-processing.pl/blog/cqrs-simple-architecture/
4. https://www.geeksforgeeks.org/the-decorator-pattern-set-2-introduction-and-design/
5. https://stackoverflow.com/questions/43433318/cqrs-command-return-values
6. https://stackoverflow.com/questions/3875607/change-the-background-color-of-a-row-in-a-jtable