URS - LunchList

Group 1: CTRL ALT ELITE

Jules Hoeberechts, Tom, Joosten, Tim Ketels, Qouc Boa Le, Liam Sangers, Dustin v.d. Veerdonk

Contents

[Functional & Non-Functional Requirements 4](#_Toc194305581)

[Functional Requirements 4](#_Toc194305582)

[Non-functional Requirements 4](#_Toc194305583)

[MoSCoW 5](#_Toc194305584)

[Must-Have 5](#_Toc194305585)

[Should-Have 5](#_Toc194305586)

[Could-Have 5](#_Toc194305587)

[Won't-Have 5](#_Toc194305588)

[EER 6](#_Toc194305589)

[Use Cases 7](#_Toc194305590)

[Use Case 01 7](#_Toc194305591)

[Use Case 02 8](#_Toc194305592)

[Use Case 03 9](#_Toc194305593)

[Use Case 04 10](#_Toc194305594)

[Use Case 05 11](#_Toc194305595)

Project description

In this project we will help the company "Code For Good(CFG)" with making a shopping list app. They want this app so they can easily make a shopping list with the entire team with no struggles and conflicts in buying items double.

The app will require a database so that the shopping lists can be saved, this way previous shopping lists can be stored and viewed to know what has been ordered in previous days. The app will be made as an asp.net application so that everyone who has permission can easily access it.

# Functional & Non-Functional Requirements

## Functional Requirements

FR-01: The user can see and edit the grocery list.

C-01.1 Contains image, with price and product name.

C-01.2 Multiple users can be active at once on the same list.

C-01.3 Users can view, add, edit and remove all items on the list.

FR-02: The user can clear the entire list with one button.

FR-03: The user gets to see every product only once, instead of one for every retailer.

FR-04: The system will make the shopping list using the selected items and best prices while trying to use just one retailer.

FR-05: The user can check off items when shopping.

## Non-functional Requirements

* The application should load within 2 seconds under normal office network conditions.
* The interface should be intuitive and easy to use, even for users with minimal tech experience.

# MoSCoW

## Must-Have

* Users can add groceries to a shared list
* Multiple users can edit the list in real-time
* Products can be added, edited, and removed
* Simple and accessible interface for office members

## Should-Have

* Connect the app to a retailer's API (Albert Heijn, Jumbo, etc.)
* Retrieve products directly from the retailer's catalog
* Support multiple retailers for price comparison
* Optimize the shopping list based on cost

## Could-Have

* Analyze purchasing behavior and suggest items based on past orders
* Categorize products (e.g., snacks, office supplies, drinks)
* Automatic reminders for frequently bought items
* Budget tracking per week/month

## Won't-Have

* Advanced AI-driven optimizations (e.g., predicting best ordering times based on discounts)
* Extensive data processing beyond basic tracking and suggestions

# EER

We made this EER so our team can have an easy overview over what the project includes like entities with attributes and relations between them. We can later use this as a reference sheet to help me make class diagrams and database designs. This is also a way to easily explain the relations to the stakeholder and check if we are missing something they want to see in the project.

A diagram of a diagram

AI-generated content may be incorrect.

# Use Cases

## Use Case 01

**Name:**  edit grocery list

**Functional Requirement:** FR-01

**Actors:** User (Office Member)

**Preconditions:**

* The user has access to the application.
* A grocery list exists.

**Flow of Events:**

1. The user opens the grocery application.
2. The system displays the grocery list.
3. The user selects an item from the list.
4. The system allows the user to edit the item (e.g., change quantity, update name, upload an image).
5. The user confirms the edit.
6. The system updates the grocery list and displays the updated item.

**Alternative Flow:**

* If the item is removed instead of edited, the system removes it from the list.
* If another user edits the list simultaneously, the system updates the list in real-time.

**Postconditions:**

* The grocery list reflects the changes made by the user.
* Multiple users can view the updated list.

## Use Case 02

**Name:** clear grocery list

**Functional Requirement:** FR-02

**Actors:** User (Office Member)

**Preconditions:**

* The user has access to the application.
* The grocery list contains at least one item.

**Flow of Events:**

1. The user navigates to the grocery list.
2. The user clicks the "Clear List" button.
3. The system prompts the user for confirmation.
4. The user confirms the action.
5. The system removes all items from the grocery list.

**Alternative Flow:**

* If the user cancels the action at step 4, the list remains unchanged.

**Postconditions:**

* The grocery list is empty.

## Use Case 03

**Name:** Single Product

**Functional Requirement:** FR-03

**Actors:** User (Office Member)

**Preconditions:**

* The user has access to the application.
* The grocery list exists and contains items.

**Flow of Events:**

1. The user looks at the list of products
2. All the products that are the same but from different retailers get hidden.
3. Only one product will be displayed of each item.

**Alternative Flow:**

* If the system allows duplicates due to a different brand or variation, it appends a distinction to the product name.

**Postconditions:**

* Each product appears only once on the list.

## Use Case 04

**Name:** Finalize grocery list

**Functional Requirement:** FR-04

**Actors:** User (Office Member)

**Preconditions:**

* The system is connected to at least one retailer.
* The user has added items to the grocery list.

**Flow of Events:**

1. The user finalizes their grocery list.
2. The system retrieves prices from available retailers.
3. The system determines the best prices for the selected items while prioritizing a single retailer.
4. The system compiles the shopping list with the chosen retailer’s products.
5. The user reviews the optimized shopping list.

**Alternative Flow:**

* If no single retailer offers all selected items, the system selects the best alternative retailer(s) while keeping the list as consolidated as possible.

**Postconditions:**

* The grocery list is optimized for the lowest cost while trying to use just one retailer.

Use Case 05

**Name:** check of items

**Functional Requirement:** FR-05

**Actors:** User (Office Member)

**Preconditions:**

* The user has access to the grocery list while shopping.
* There is a finalized grocery list.
* The grocery list contains at least one item.

**Flow of Events:**

1. The user opens the grocery application while shopping.
2. The user selects an item they have placed in their cart.
3. The system marks the item as checked off (e.g., strikethrough, grayed out, or moved to a "completed" section).
4. The user repeats steps 2-3 for each purchased item.
5. The system saves the updated list.

**Alternative Flow:**

* If the user mistakenly checks off an item, they can uncheck it.

**Postconditions:**

* Checked-off items are visually distinguished from unpurchased items.
* The system retains the checked-off status until the list is cleared or reset.

## Use case diagram

A diagram of a grocery store with Ice hockey rink in the background

AI-generated content may be incorrect.