Software Design Document for Smart Fridge Kitchen Assistant

Prepared by Sai Sivva, Ahmed Humayun, and Joshua Wilson on behalf of the Smart Fridge Team

JOSHUA WILSON
SAI SIVVA
AHMED HUMAYUN
SHU YANG
ZHEN LI

CSC 532: Software Engineering

Contents

Revision History

Rev.#	Date	Nature of Revision	Version
1	10.30.2016	ORIGINAL VERSION	1.0

1 Introduction

The main purpose of the Software Requirement Specifications (SRS) is to provide information about what is to be expected of the Smart Fridge Kitchen Assistant upon completion. This document specifies all the functional and non-functional requirements of the software being developed, and explains all the entities involved in the development of the software.

1.1 Purpose

1.1.1 Vision Statement At the core, the system is a user interface for kitchen inventory management. However, it will help the user to maintain their personalized nutrition plans, and provide feedback to reduce energy and food waste. While there do exist other fully contained smart fridges with integrated touch screen panels, the cost of these are prohibitively expensive for most people. This project aims to create an inexpensive product which one may use with ones current “dumb” refrigerator. The project may be used as a standalone system or may be integrated with other smart house applications.

1.1.2 Scope The scope of this project is to refine the existing Smart Fridge Project. This includes creating a user interface for using for a touch screen platform, implementing a customizable nutrition plan for the user, broadening the project to include several inventories, and allowing for a bulk fridge update using information from receipts. Additionally, the product will allow for integration with the Pillar smart house project.

1.2 Document Conventions

This document follows MLA Format which suggests to use bold-faced text to emphasize section and subsection headings. Underlining is to point out words in the glossary and italicized text is used to label and recognize diagrams.

1.3 Intended Audience and Reading Suggestions

This document is intended to be read by the developers, testers, project manager, system architect, and analysts of the Smart Fridge Team.

1.4 References

Unless otherwise noted, all figures may be found at the end of the document.

2 Overall Description

The Smart Fridge Kitchen Assistant is a refinement of an existing Smart Fridge product. The software is a Java web application to be accessed on an apache server run by a Raspberry Pi 3 with attached touchscreen an camera modules.

2.1 Product Perspective

The smart fridge inventory management system helps to keep track food items and can generate shopping list based on personalized nutrition preferences, save energy by suggesting temperature based on inventory, and manage the trash generated due to expired products.

2.2 Product Functions

The main function of this project is to improve the customer experience by improving the user interface and adding functionality. The key features are as follows:

1. Uses information available on grocery receipts to determine each of the UPCs from the groceries that the user has purchased. The system will then communicate with a product database through the Nutritionix^{TM}API to determine the nutritional information about the product. The product, along with the nutritional information is the entered into the database automatically and an approximate expiration date is set for the item.
2. Allows for touch querying and updating through a newly designed interface. This interface will also display the contents of the refrigerator via icons representing each food item.
3. Generates personalized shopping list based on nutritional needs and expiration of current food items.
4. Integrates a waste and energy model. Waste from spoiled is approximated and shown to the user. This feedback will allow them to reduce spoilage in the future. Additionally, a temperature is suggested from the contents of the refrigerator. This may either be integrated with an existing fridge temperature adjustment system, or the user can manually adjust the temperature of their fridge.
5. Integration with the Pillar smart home system to allow Pillar to access the Smart Fridge database and make adjustments as needed.

2.3 User Classes and Characteristics

The target users of this system are middle to low income users who wish to keep track of their food consumption and waste. After creating an account, users specify their nutritional needs (e.g. weight loss, high protein diets, vegan) based on which a nutrition plan is constructed. Additionally, the user may, with the aid of the software, create weekly meal plans from which shopping lists are created.

2.4 Operating Environment

The software is developed to run as a web application on a LAMP server hosted by a Raspberry Pi 3, which uses Raspbian (a Debian-based operating system). In principle, any computer or mobile device on the same network as the Raspberry Pi will be able to access the server through an Internet browser.

2.5 Design and Implementation Constraints

The Raspberry Pi 3 (RPi3) has Model B Quad-Core 1.2 GHz 1 GB RAM, On-board WiFi and Bluetooth Connectivity.

2.6 User Documentation

Dr. Box and the previous team have already prepared the user document for the previous version. We intend to add details of the new specifications to that document and we will present it to the user.

2.7 Assumptions and Dependencies

It is assumed that the receipts contain QR codes which allow for the system to obtain the UPC of all products purchased by the user upon scanning. Each user must create a user ID, and all food additions and consumptions from the fridge are properly logged.

3 External Interface Requirements

3.1 User Interfaces

The main aim is to provide a bulk update feature along with personalized nutrition plans, shopping lists, and an overhauled user interface. Reference Figure 3.1 and Figure 3.2 for the current user interface.

3.2 Hardware Interfaces

The product uses the a Raspberry Pi Camera module to scan a receipt for a QR code. The Raspberry Pi Touchscreen module is used for the touchscreen GUI.

3.3 Software Interfaces

The product is a web application to be included in an Apache server which is developed to run on a Raspberry Pi running the Raspbian Jessie operating system. In principle, however, this may be run from any computer with the appropriate server, camera, and touch screen configuration.

3.4 Communications Interfaces

The product is designed to interface with use the Nutritionix API to interface with their database. Additionally, the product will communicate with the Pillar through yet-to-be-developed interface.

4 System Features

4.1 Registration

4.1.1 Goal:
To register the user into the database and to gather health and dietary information.

4.1.2 Input:
user ID, password, health and dietary information

4.1.3 Output:
User registration successful, added user to database

4.1.4 Main Scenario:
To register the user into the database and to gather health and dietary information.

4.1.5 Pre-Condition:
Registration screen displayed

4.1.6 Steps:

1. User selects “register” from screen
2. User specifies user ID, password
3. and health and dietary information on screen.

4.1.7 Post-Condition:
User information stored into database

4.1.8 Representation:
See Figure 4.1

4.2 Login

4.2.1 Goal:
To allow the user to login to the web application

4.2.2 Input:
user ID, password

4.2.3 Output:
user credentials verified

4.2.4 Main Scenario:
User is authenticated and brought to home page

4.2.5 Pre-Condition:
Login screen is displayed

4.2.6 Steps:

1. User inputs credentials into login screen
2. Credentials are checked against user database
3. User is verified

4.2.7 Post-Condition: Homepage is displayed

4.2.8 Representation: See Figure 4.2

4.3 Bulk Update

4.3.1 Goal: Allows user to update fridge with information from receipt

4.3.2 Input: Photograph of receipt

4.3.3 Output: Item and nutritional information input into database

4.3.4 Main Scenario: User needs to update large quantify of items which were recently purchased

4.3.5 Pre-Condition: User is logged in an camera module is activated

4.3.6 Steps:

1. User holds receipt up to camera
2. QR code from receipt is scanned
3. From QR code, UPC information is gathered
4. UPC information is used with Nutritionix API to determine nutritional information
5. Item nutritional information and nutritional information entered into inventory database

4.3.7 Post-Condition: Inventory screen is displayed

4.3.8 Representation: See Figure 4.3

4.4 Display Fridge Contents and Single Item Update

4.4.1 Goal: To allow the user to know the contents of the fridge without opening the fridge

4.4.2 Input: User request to see contents

4.4.3 Output: Display contents on screen, updated database

4.4.4 Main Scenario: User wishes to see the smart fridge inventory and/or update the contents

4.4.5 Pre-Condition: User is logged in

4.4.6 Steps:

1. User uses touch screen to request contents of fridge
2. Query Inventory Database for contents
3. Display Inventory Icons on screen
4. User touches icon
5. Display number of items and nutritional information
6. User edits item information

4.4.7 Post-Condition: Display Inventory Page

4.4.8 Representation: See Figure 4.4

4.5 Manual Update

4.5.1 Goal: Allows the user to manually enter and edit items in database as spreadsheet

4.5.2 Input: Item and Nutritional information

4.5.3 Output: Database updated

4.5.4 Main Scenario: User needs to enter items without receipt which are not already in the fridge.

4.5.5 Pre-Condition: User is logged in and on main page

4.5.6 Steps:

1. User request inventory spreadsheet screen
2. User adds or changes database entries
3. User confirms changes
4. Home screen displayed

4.5.7 Post-Condition: Database updated

4.5.8 Representation: See Figure 4.5

4.6 Recommend Temperature

4.6.1 Goal: Recommends optimal fridge temperature based on current items in fridge

4.6.2 Input: Fridge inventory

4.6.3 Output: Optimal Temperature

4.6.4 Main Scenario: User wishes to reduce energy usage by increasing fridge temperature

4.6.5 Pre-Condition: Application is running on server, there is at least one item in the fridge inventory, user updates contents

4.6.6 Steps:

1. Determine items in fridge
2. Calculate optimal temperature for inventory
3. From QR code, UPC information is gathered
4. Display optimal temperature

4.6.7 Representation: See Figure 4.6

4.7 Estimate Waste

4.7.1 Goal: Estimates food waste due to spoilage

4.7.2 Input: Expired food item

4.7.3 Output: Amount of food wasted (normalized)

4.7.4 Main Scenario: User wishes to know the amount of food wasted due to spoilage

4.7.5 Pre-Condition: User is logged, food item has expired

4.7.6 Steps:

1. Query inventory for expired food
2. Log Expired Food items
3. Assign value to expired food items
4. Display value on homepage

4.7.7 Representation: See Figure 4.7

4.8 Create Shopping List

4.8.1 Goal: Creates shopping list for user based on recipe requirements and current food in inventory

4.8.2 Input: Recipes, inventory

4.8.3 Output: Shopping list

4.8.4 Main Scenario: User wants to have a shopping list with the items needed for certain recipes, with items already in the inventory not included.

4.8.5 Pre-Condition: User logged in, recipe page displayed

4.8.6 Steps:

1. User adds recipe to meal plan
2. Determine ingredients for recipe
3. Determine if ingredients are already in inventory
4. Update Required ingredients list
5. Add items to shopping list
6. User opens Shopping List Page
7. Display shopping list
8. Print Shopping List

4.8.7 Post-Condition: Confirm Print

4.8.8 Representation: See Figure 4.8

5 Nonfunctional Requirements

5.1 Performance Requirements

To ensure that no errors, bugs pop up when user tries to access the software product. Interface between Pillar and Smart fridge must be good so that user does not notify any issues.

5.2 Safety Requirements

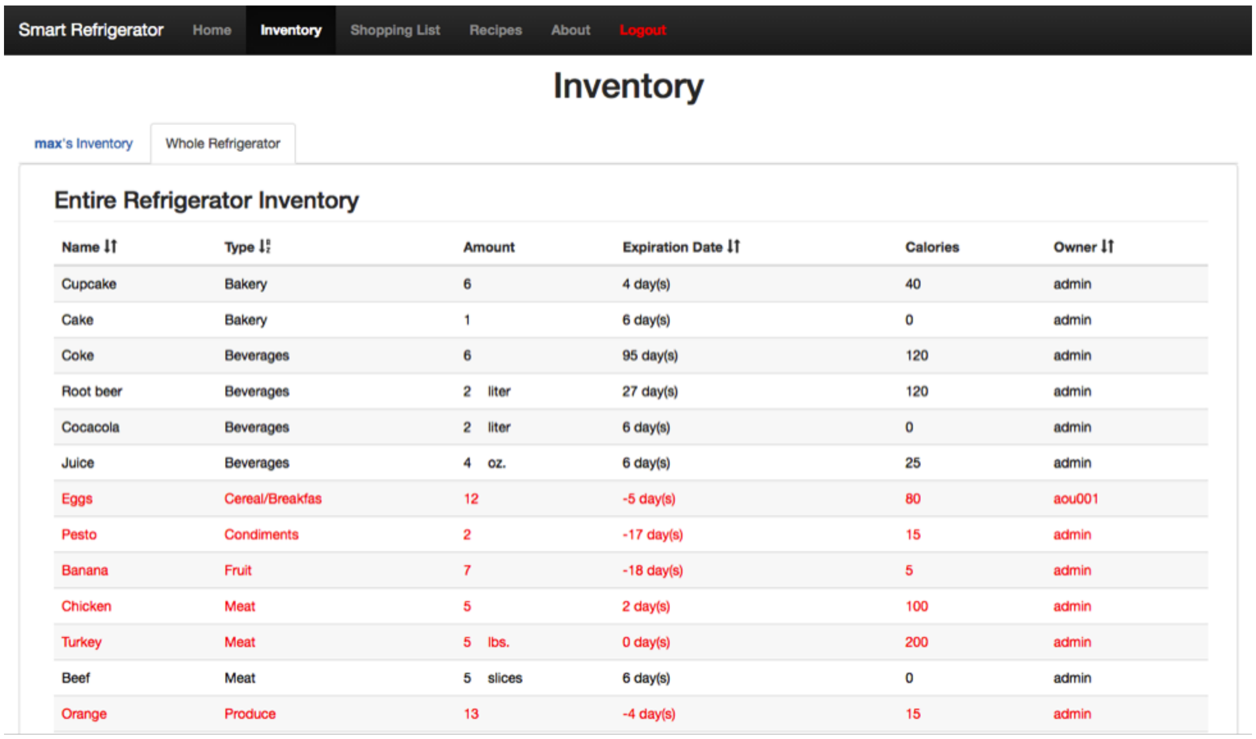
Safety is a crucial requirement for this system because different users have different nutritional requirements and needs and due importance has to be given to allergies to various food products while generating a personalized food plan.

5.3 Security Requirements

Security is also an important aspect as a user's nutritional plans are confidential in nature. Therefore, measures must be taken to protect the user from nonconsensual leaks of information to 3rd parties (e.g. creators of geared ads).

5.4 Software Quality Attributes

The designed software will be adaptable to any environment, reliable to all users, flexible for different inventories, portable to several devices, and accessible regardless of platform.



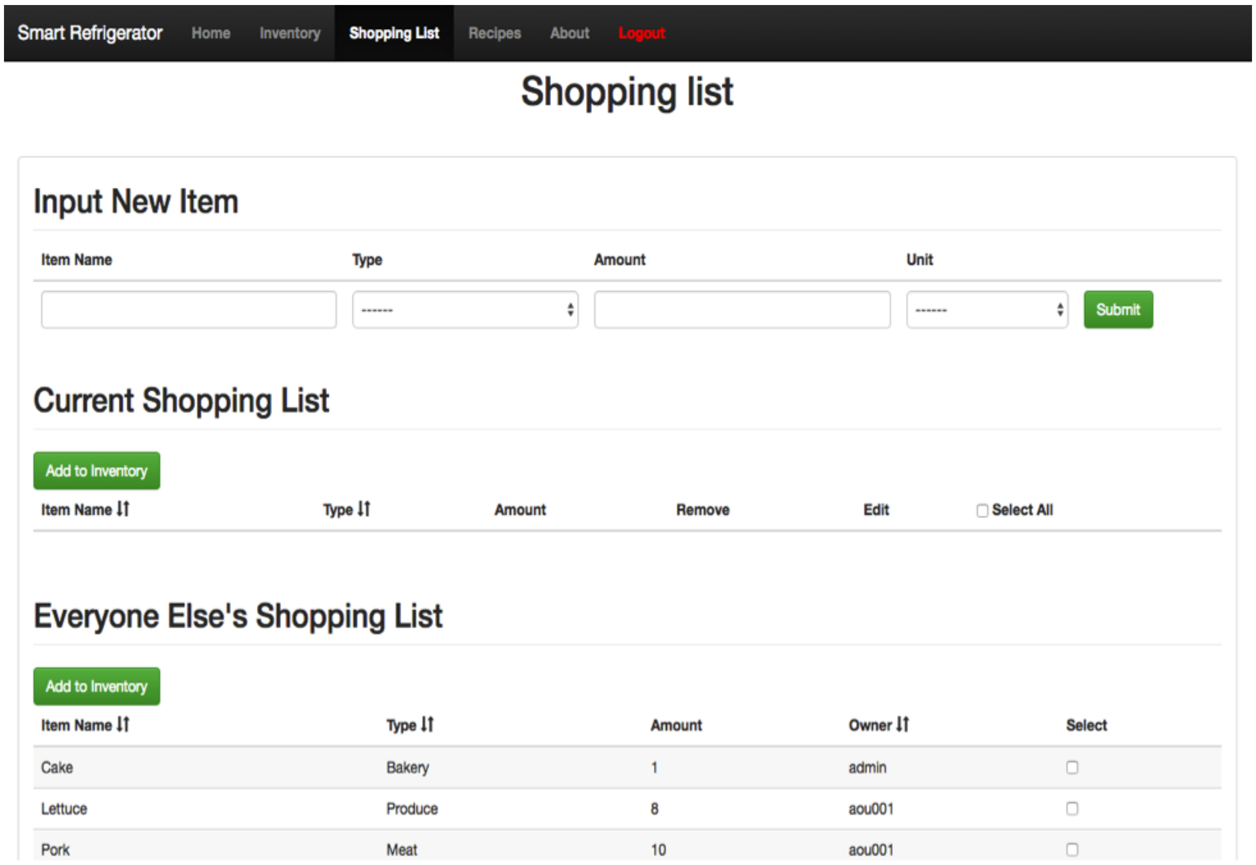
Inventory

max's Inventory | Whole Refrigerator

Entire Refrigerator Inventory

Name	Type	Amount	Expiration Date	Calories	Owner
Cupcake	Bakery	6	4 day(s)	40	admin
Cake	Bakery	1	6 day(s)	0	admin
Coke	Beverages	6	95 day(s)	120	admin
Root beer	Beverages	2 liter	27 day(s)	120	admin
Cocacola	Beverages	2 liter	6 day(s)	0	admin
Juice	Beverages	4 oz.	6 day(s)	25	admin
Eggs	Cereal/Breakfas	12	-5 day(s)	80	aou001
Pesto	Condiments	2	-17 day(s)	15	admin
Banana	Fruit	7	-18 day(s)	5	admin
Chicken	Meat	5	2 day(s)	100	admin
Turkey	Meat	5 lbs.	0 day(s)	200	admin
Beef	Meat	5 slices	6 day(s)	0	admin
Orange	Produce	13	-4 day(s)	15	admin

Figure 3.1: Inventory interface from previous project showing the inventory



Shopping list

Input New Item

Item Name	Type	Amount	Unit
	------		------

Current Shopping List

Item Name	Type	Amount	Remove	Edit	Select All

Everyone Else's Shopping List

Item Name	Type	Amount	Owner	Select
Cake	Bakery	1	admin	☐
Lettuce	Produce	8	aou001	☐
Pork	Meat	10	aou001	☐

Figure 3.2: Shopping list interface from previous project showing the inventory

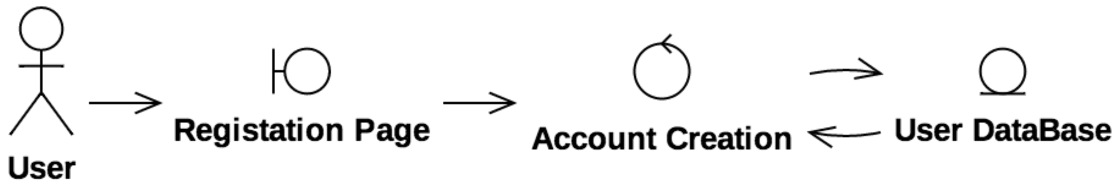


Figure 4.1: Registration

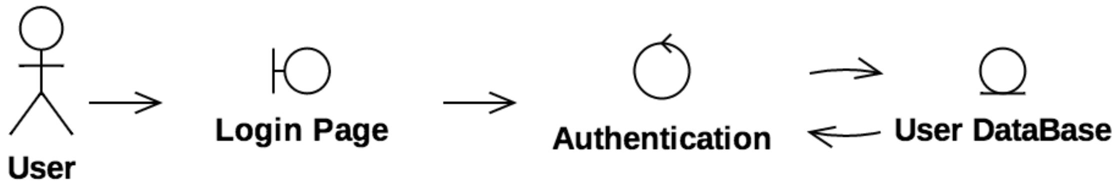


Figure 4.2: Login

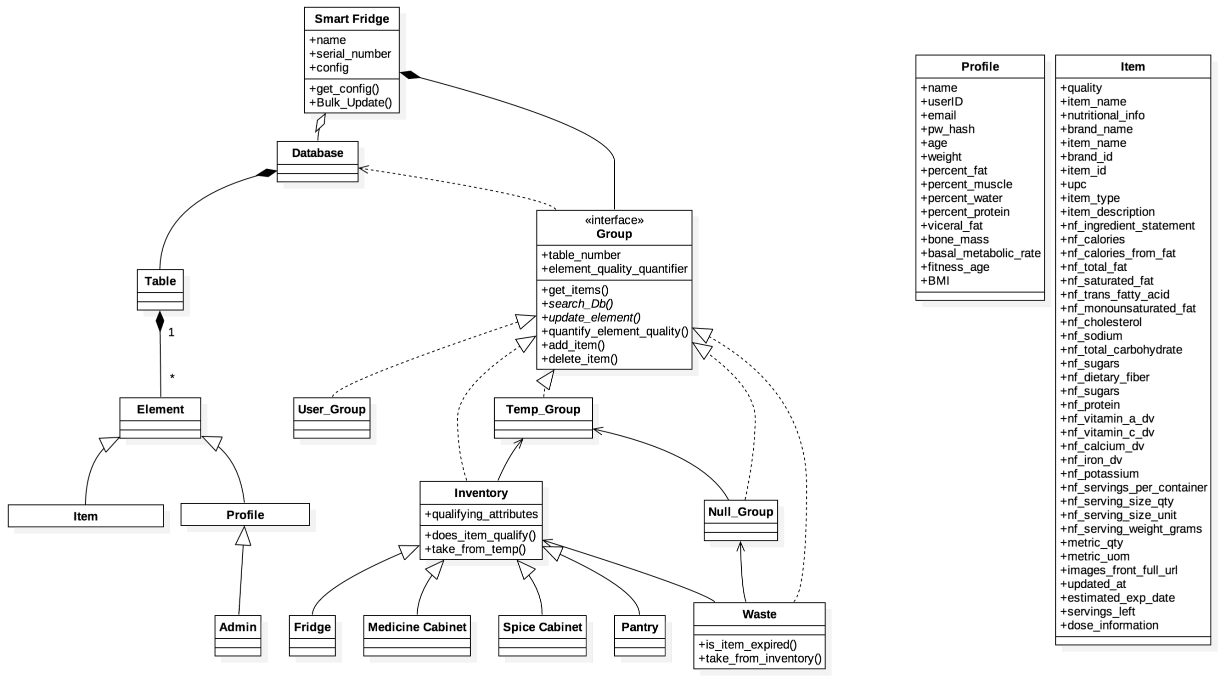


Figure 4.3: Bulk Update

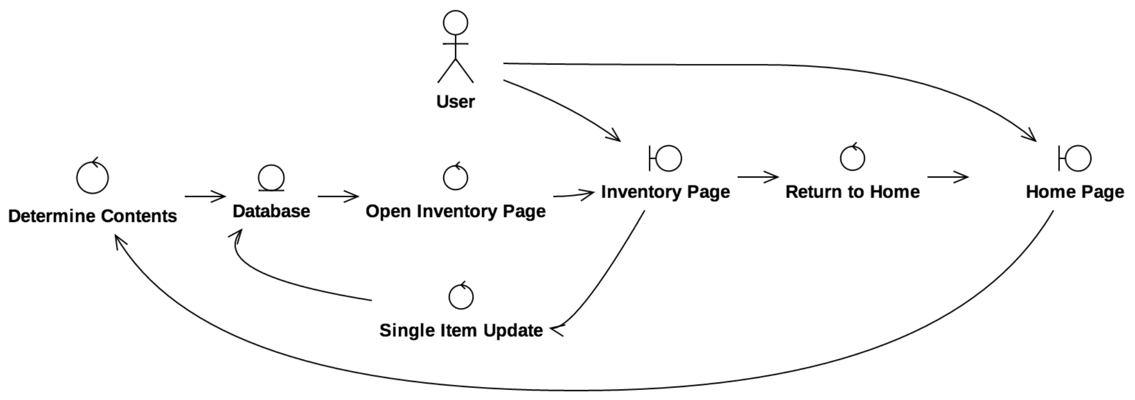


Figure 4.4: Display Fridge Contents and Single Item Update

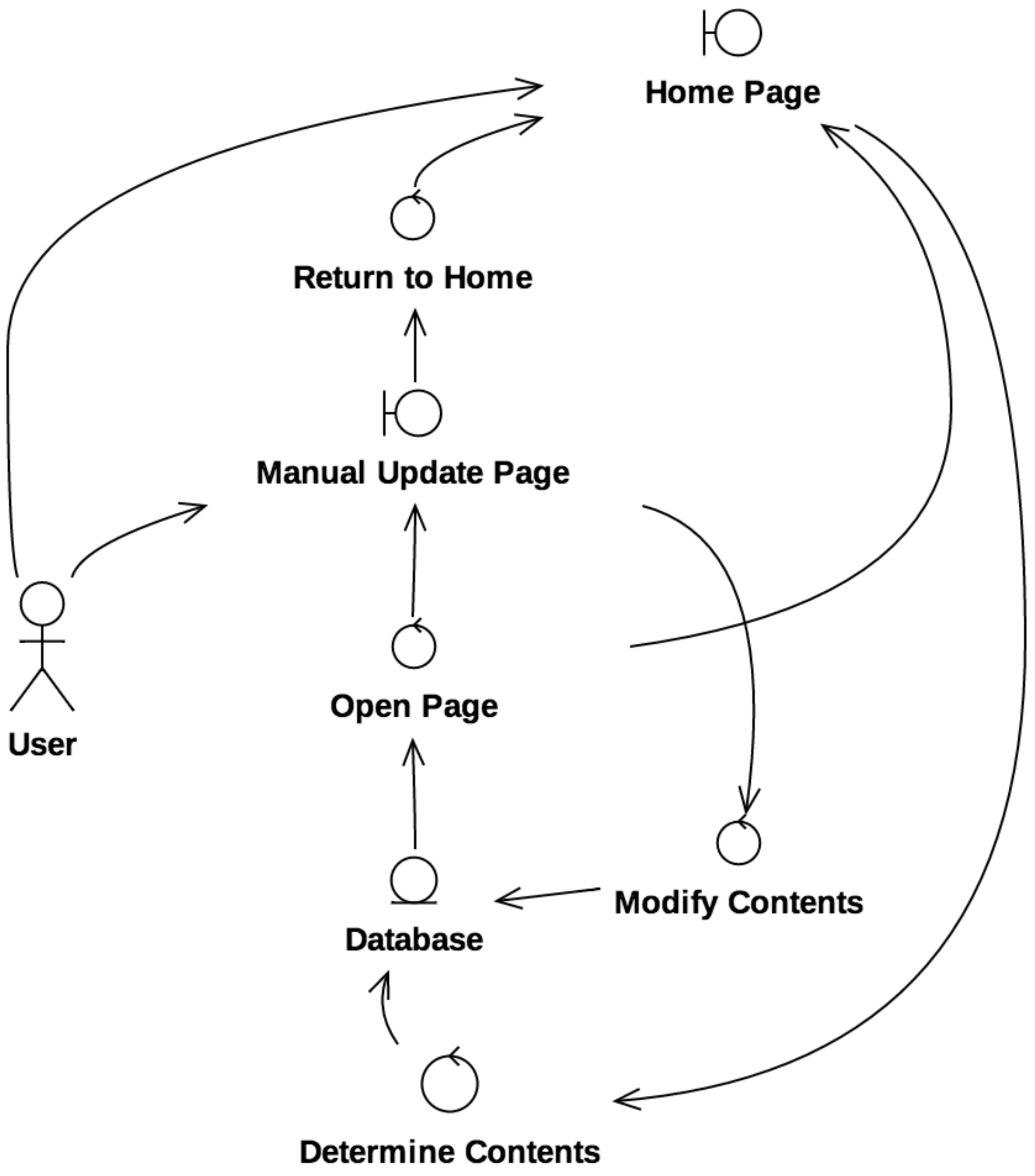


Figure 4.5: Update

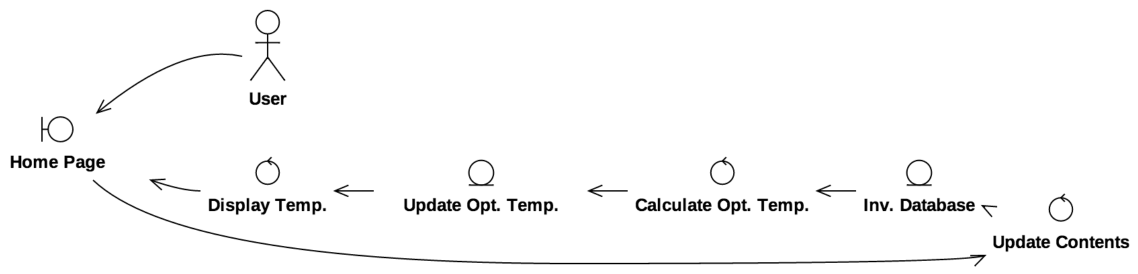


Figure 4.6: Recommend Temperature

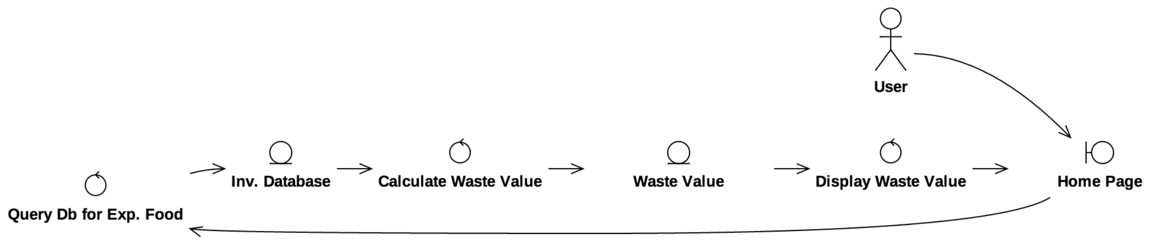


Figure 4.7: Estimate Waste

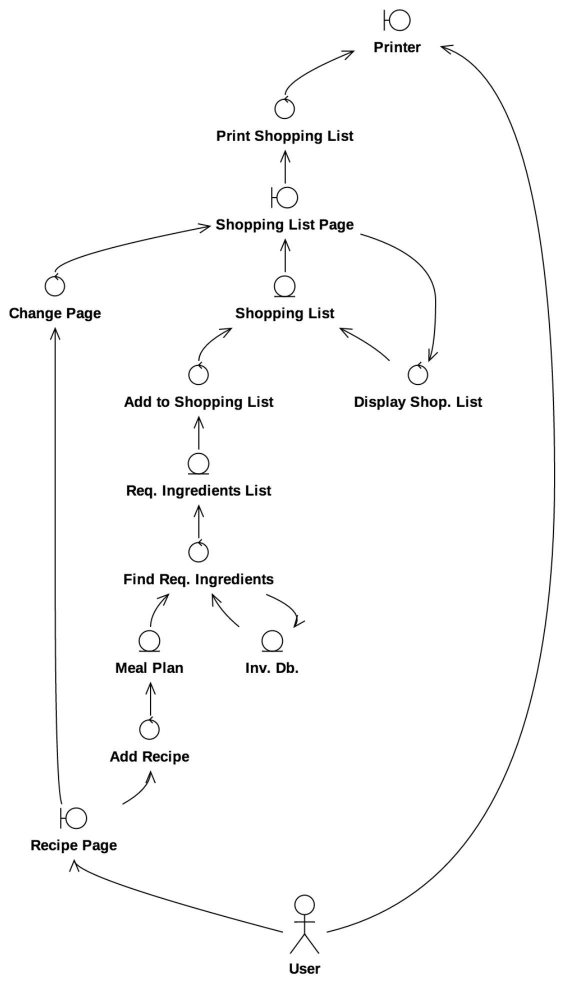


Figure 4.8: Create Shopping List