

**Software Requirements Specification**

*COSC2299 Software Engineering: Process and Tools (2350)  
Group P7\_7*

#### 

Team:

Ashley Mullin - s3937907

Clarence O'toole - s3943755

Jaspal Singh-Walia - s3899925

Manik Thapliyal - s3953664

Sri Kalyan Rohan - s3958905

Ramon Aguila - s3840883

Table of Contents:

[Revision History 2](#_n1xap4hwifge)

[Introduction 3](#_w2nnulougf00)

[Purpose 3](#_xhl6x1v552co)

[Functional Requirements 3](#_3iii5kulpj2x)

[Non-functional Requirements 4](#_3rlu4i845nv8)

[Wireframes 4](#_m5itcvewb4gr)

[1. Product Page 4](#_f0oezldw1ob5)

[2. Landing Page 5](#_u4jud8mqy22i)

[3. Profile (My Orders) 6](#_nkgs5aluharw)

[4. Profile (Personal Information) 7](#_gc8dyieehp6s)

[5. Profile (Account Settings) 8](#_ofhoyxm0u1rb)

[6. Profile (Wishlist) 9](#_ycnglus7cgg2)

[7. Item Page 10](#_kcmkklxqth1t)

[8. Shopping Cart 11](#_gtwvmbz4umfa)

[9. Checkout 12](#_zkvwo35468j)

[10. Registration Page 13](#_33gipm1sbjfw)

[11. Login Page 14](#_2zv9xwjhxlpc)

[Architecture 15](#_92vn7tvvxab8)

[Tools and Technologies Used 16](#_e2d4mmgohik9)

[User Stories 16](#_58ffcc1paryz)

[Unit Tests 26](#_26rav77k854f)

[UML 26](#_hzjx6b5bpa4i)

[Refactoring report](#_1bs1u6d94gg0) 27

# Revision History

| **Name** | **Date** | **Reason for changes** |
| --- | --- | --- |
| Ramon | 19/08/23 | Removed product backlog,  sprint 1 and 2 backlogs, and  Sprint 0 retro. These are uploaded on GitHub as separate documents |
| Ramon | 11/09/23 | Updated SRS document according to feedback from Milestone 1   * Added numbers to wireframes * Added to architecture “tools and technologies used” * Added sections for functional and non-functional requirements * Fixed grammar mistakes |
| Ramon | 17/09/23 | Updated functional requirements based on changes discussed with the Product Owner. |
| Clarence | 17/09/23 | Added UML Diagrams |
| Ramon | 5/10/2023 | Edit SRS document for Milestone 3:   * Added introduction * Added purpose * Edited functional requirements * Updated tools and technologies * Updated tests * Added refactoring report |

# Introduction

Welcome to the Software Requirements Specification (SRS) document for Software Engineering: Process & Tools. This document comprehensively outlines the requirements, functionalities, and design aspects of Superprice. The SRS serves as a fundamental guide for all stakeholders involved in the development and testing of the software.

# Purpose

The purpose of this SRS document is to provide a clear and unambiguous understanding of the software system, its features, behaviour, and constraints. It serves as a foundation for communication between the development team and stakeholders, ensuring a shared vision of the project and facilitating the development process.

# Functional Requirements

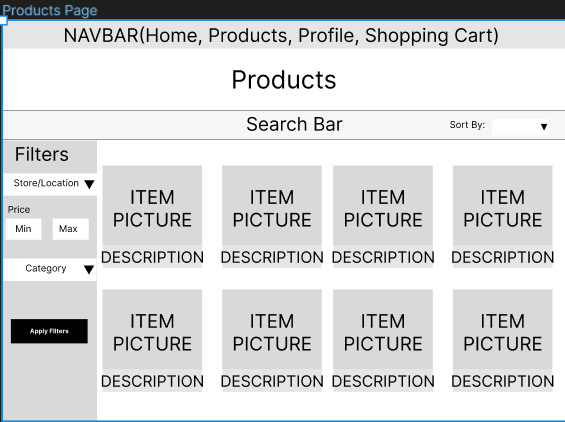
1. **Search**: Users can search for grocery products using keywords.
2. **Item Viewing**: Users can view information about a product, including its name, description, store, price, stock status, and available discounts.
3. **Cart Management**: Users can add items to their cart and remove unwanted items.
4. **Delivery Options**: Users can choose and order their selected items for delivery, with various delivery options available based on the store.
5. **Checkout Process**: Users can proceed to checkout and complete the purchase of items in their cart. They will be provided with a summary of their order.
6. **[Enhancement feature] Notifications Feature**: Users can access a dedicated products page where they can view products for which they've activated notifications, as well as products currently on promotion. They can also check the promotion dates.
   1. **Previously:** This feature was meant to involve sending pop-up notifications for price changes and new promotions. We changed it to the current version as it’s easier to implement and provides a more intuitive and seamless experience.

# Non-functional Requirements

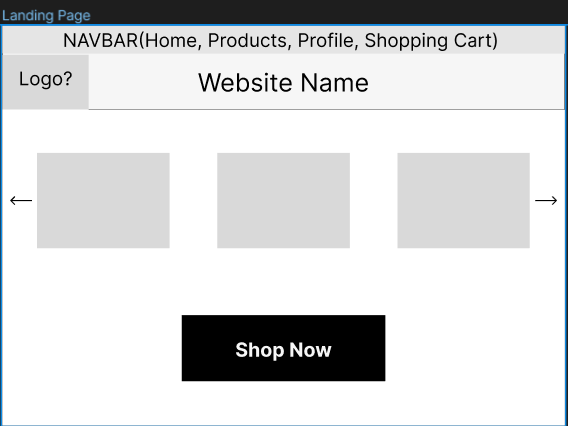
* **Usability**: The user interface will be intuitive and easy to use, with clear navigation and responsive design for various devices.
* **Maintainability**: Code should be well-documented and modular to allow for easier maintenance and updates.
* **Performance**: The application should have a response time of less than 2 seconds for common user actions (e.g., searching for products, adding/removing items from the cart) to ensure a smooth user experience.

# Wireframes

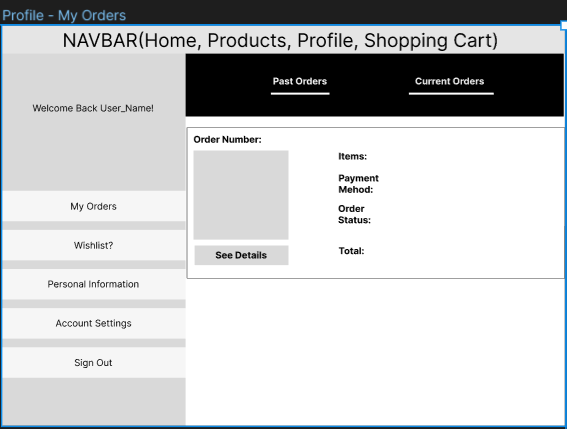
### Product Page



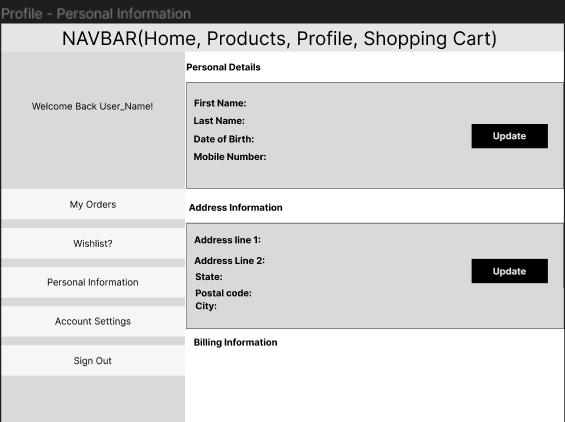
### Landing Page



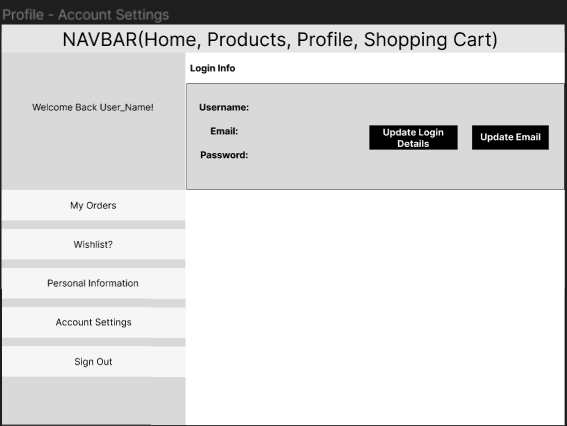
### Profile (My Orders)



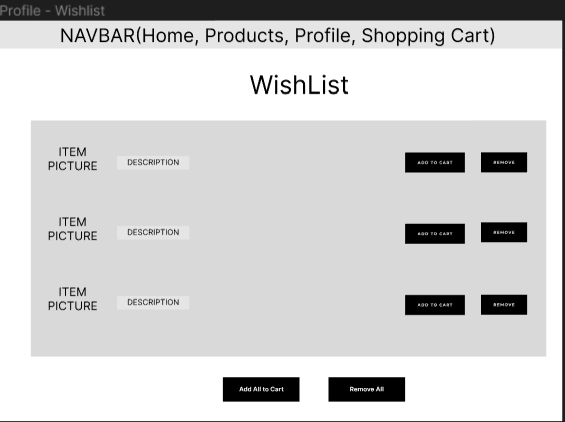
### Profile (Personal Information)



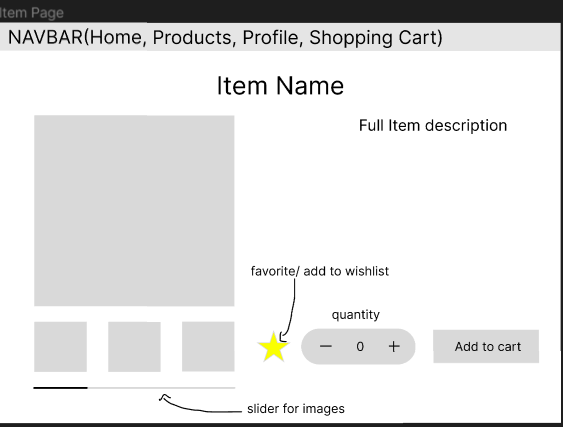
### Profile (Account Settings)



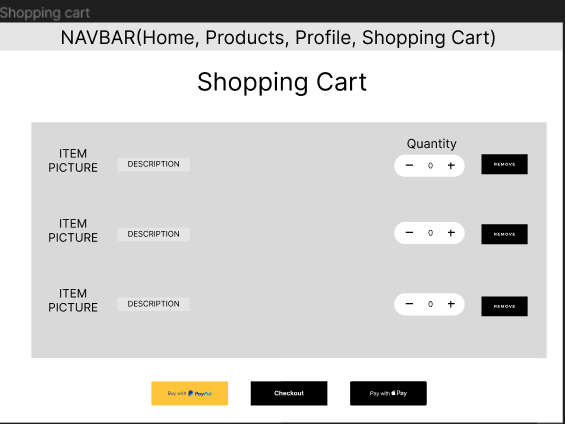
### Profile (Wishlist)



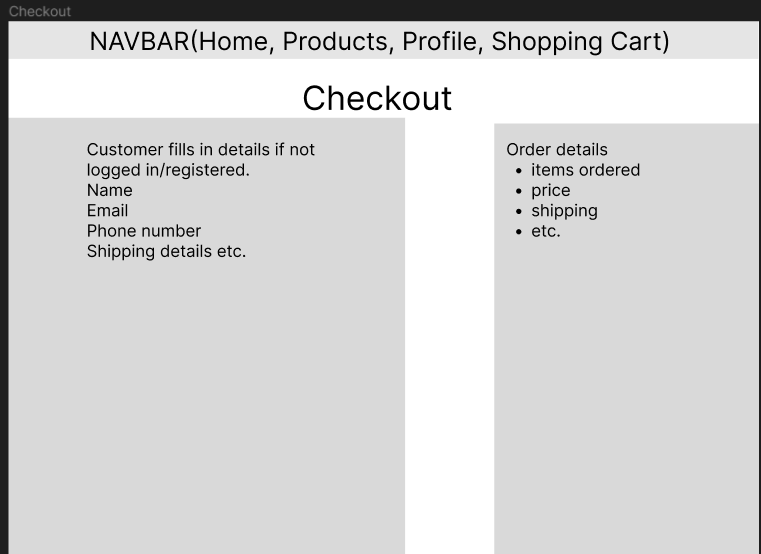
### Item Page



### Shopping Cart



### Checkout



### Registration Page

# 

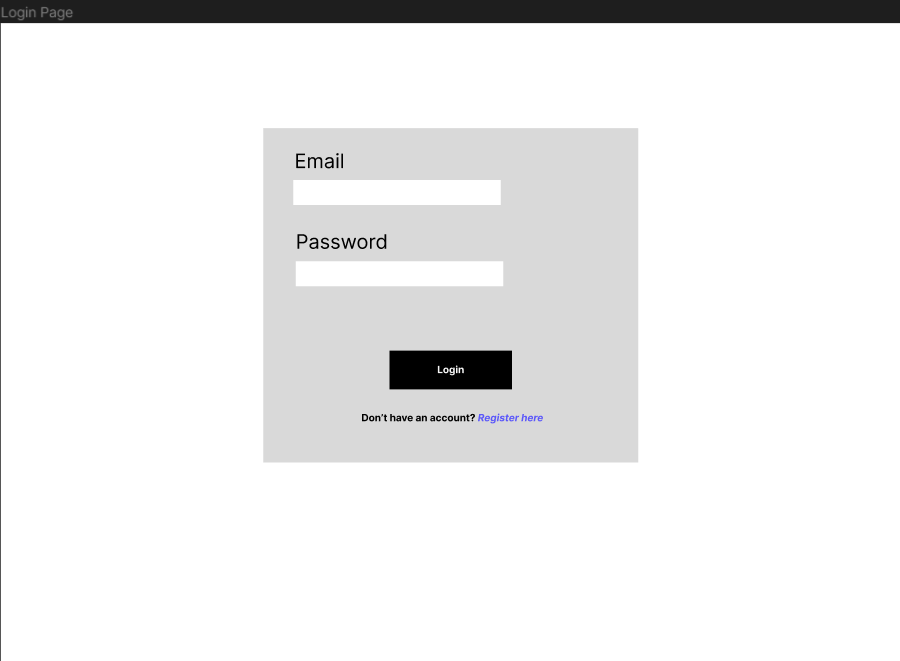
# 

# 

# 

# 

### Login Page



# Architecture

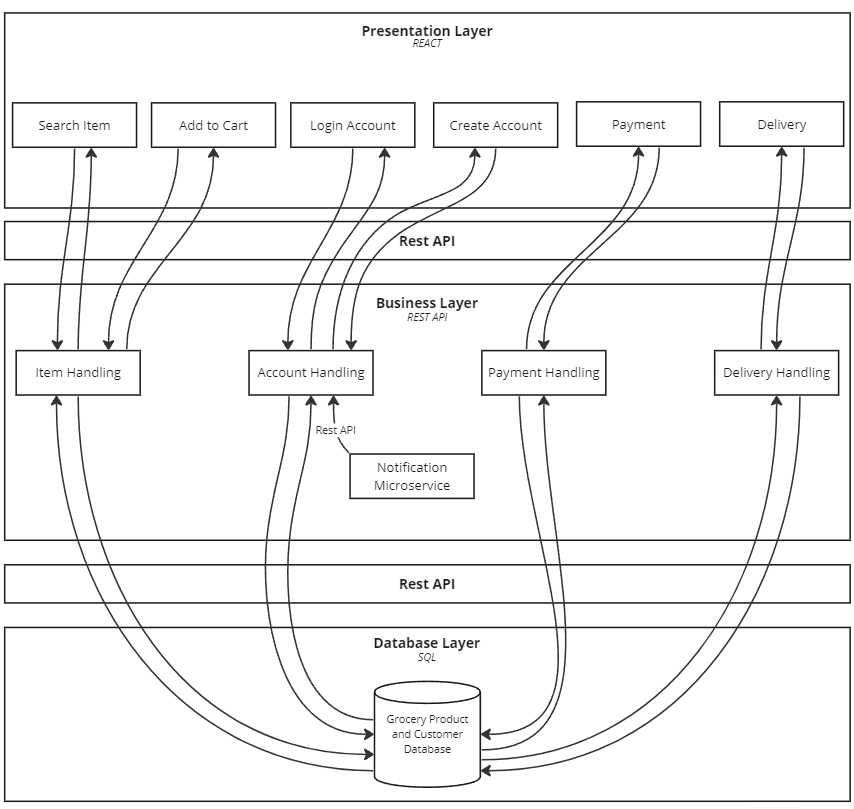


Figure 1.0 - System architecture diagram

Included in the final version:

* Search Item: This component enables users to query the product database and retrieve a list of relevant products.
* Add to Cart: Users can add an item to their cart.
* Payment: The system prompts users to pay for their order and choose a preferred payment service.
* Delivery: Users can choose different delivery options.
* Grocery Product and Customer Database: Utilizing MySQL, this database stores product-related data such as cost and stock.
* Notification: Users can access a dedicated products page where they can view products for which they've activated notifications, as well as products currently on promotion. They can also check the promotion dates.

Not included:

* Login Account: Users can log in to access their account.
* Create an Account: New users can create a new account.
* Account Handling: It manages login information and relevant user details.
* Payment Handling: Responsible for processing user payments and ensuring the correct amount is charged.
* Delivery Handling: Calculates delivery details, including time, location, and status.

### Tools and Technologies Used

* **Rest API**: Utilized for communication between system layers, promoting a modular design that enhances security. This approach eliminates direct connections between the presentation layer and the database, simplifying development by allowing teams to work independently without dependencies on each other's features.
* **H2 Database**: This was used as our initial database solution due to its user-friendly interface, saving valuable development time and streamlining processes.
* **Maven Dependencies**: Integrated to optimize software development by providing a pre-built, well-organized, and thoroughly tested framework. This streamlines the development process, ensuring efficiency and reliability in our project.
* **React**: Used for front-end development, React facilitates the rapid creation of user-friendly websites. Its structured approach and automatic updates simplify code management, enabling seamless additions, modifications, and error debugging to enhance the overall development experience.

# User Stories

**Definition of done**

* Code is written by multiple developers and/or peer-reviewed
* Unit tests cover 80% of situations, and all tests pass
* Functional testing based on acceptance criteria pass
* Implemented feature is intuitive to use, and passes usability tests
* Code tested and verified to work on website browsers (Chrome and Firefox), using Windows
* The code contains no known bugs
* Code is well-documented
* Code approved by the product owner

| **Story #1** | Search - Keywords | **Priority** | High |
| --- | --- | --- | --- |
| **Estimate** | 3 |
| As a | customer | | |
| I want to | Search for a product using keywords, | | |
| So that | can quickly find the item I'm looking | | |
|  | | | |
| Acceptance criteria | **Scenario: Search with results**  Given I am on the products page and have entered keywords of the item in the search bar, when the I click "search",  then the system will display products that include the keywords in the name, description or category.  **Scenario: Search with NO results**  Given I am on the products page and have entered the name of the item in the search bar and the item does not exist,  when I click "search",  then the system will display to my screen an error message, saying that the product does not exist. | | |

| **Story #2** | Search - Category | **Priority** | High |
| --- | --- | --- | --- |
| **Estimate** | 3 |
| As a | customer | | |
| I want to | browse different categories | | |
| So that | can explore a range of offerings from stores close to me | | |
|  | | | |
| Acceptance criteria | **Scenario: Viewing the categories**  Given I am on the home page,  when I click 'Categories',  then the system will display a list of categories that the user can select from.  **Scenario: Selecting a category and displaying results**  Given I have the categories displayed,  when I click a category,  then the system will display all the items from the selected category. | | |

| **Story #3** | Search - Viewing an item | **Priority** | High |
| --- | --- | --- | --- |
| **Estimate** | 2 |
| As a | customer | | |
| I want to | view an item's information | | |
| So that | I can make an informed decision about whether to purchase it | | |
|  | | | |
| Acceptance criteria | **Scenario: Displaying the item's information**  Given I entered a search and the system is displaying at least one item,  when I click on the item,  then the system will display the key information about the item.  *key information includes Name, description, stock availability, store and available delivery options.*  **Scenario: Displaying updated information**  Given I entered a search and the system is displaying at least one item,  when I click on the item,  then the system will display the updated information about the item.  examples of information changes can include price, stock availability and delivery options. | | |

| **Story #4** | Sorting results by price | **Priority** | High |
| --- | --- | --- | --- |
| **Estimate** | 3 |
| As a | customer | | |
| I want to | sort the search results from cheapest to most expensive | | |
| So that | I can choose from the cheaper options first | | |
|  | | | |
| Acceptance criteria | **Scenario: Sorting the search results**  Given I entered a search that displays at least one item,  when I click 'Sort by Price',  then the system will sort the items in ascending order of price. (Lowest price first, highest price last) | | |

| **Story #5** | Cart | **Priority** | High |
| --- | --- | --- | --- |
| **Estimate** | 5 |
| As a | customer | | |
| I want to | view my cart, | | |
| So that | I can check what items I’m missing or need to remove before checking out. | | |
|  | | | |
| Acceptance criteria | **Scenario: Viewing an empty cart**  Given I am a registered user,  when I click 'Cart' on the navigation bar,  then the system will direct me to the 'Cart' page and display a message saying, "The cart is currently empty".  **Scenario: Viewing cart with at least one item in it**  Given I am a registered user and I am on the homepage and I added at least one item in the cart,  when I click 'Cart',  then the system will direct the user to the 'Cart' page and display the items added to the cart.  **Scenario: Removing an item from the cart**  Given I am a registered user and I am on the 'Cart' page and there is at least one item in the cart,  when I click 'Remove item',  then the system will remove the selected item from the cart.  **Scenario: Editing quantity**  Given I am a registered user and I am on the 'Cart' page and there is at least one item in the cart,  when I click 'Edit' and enters a new number,  then the system will update the item quantity and total price accordingly. | | |

| **Story #6** | Adding an item to the cart | **Priority** | Medium |
| --- | --- | --- | --- |
| **Estimate** | 2 |
| As a | customer, | | |
| I want to | add an item to my cart | | |
| So that | purchase it together with other items | | |
|  | | | |
| Acceptance criteria | **Scenario: Adding a new item to the cart**  Given I am logged in to the website and I have an item page opened,  when I click 'Add to Cart',  then the system will add 1x item to the user's cart.  **Scenario: Adding an existing item to the cart**  Given I am logged in to the website and I have an item page opened and the item is already in the cart,  when I user click 'Add to Cart',  then the system will increase the item's quantity by one. | | |

| **Story #7.1** | Selecting delivery options | **Priority** | Medium |
| --- | --- | --- | --- |
| **Estimate** | 3 |
| As a | registered customer | | |
| I want to | be able to pick from a wide range of delivery options such as standard, express, and same-day | | |
| So that | I can select the best option that suits my needs | | |
|  | | | |
| Acceptance criteria | **Scenario: Delivery option drop-down**  Given I am on the 'Cart' page and I added at least one item in the cart,  when I click on 'Select delivery type',  then the system will display a drop-down bar, showing the available delivery options based on the items in the cart.  **Scenario: Updating the delivery price**  Given I am on the 'Cart' page and I have the delivery options open,  when I click on a delivery option,  then the system will display the new delivery price and add it to the grand total. | | |

| **Story #7.2** | Selecting delivery options | **Priority** | Medium |
| --- | --- | --- | --- |
| **Estimate** | 3 |
| As a | registered customer | | |
| I want to | designate a preferred window of time for my delivery to arrive | | |
| So that | my delivery does not arrive when I am away from the house. | | |
|  | | | |
| Acceptance criteria | **Scenario: Displaying delivery time frames**  Given I am on the 'Cart' page and I selected a delivery option,  when I click 'Select delivery time frame',  then the system will display a list of available delivery time frames.  **Scenario: Selecting a delivery time frame**  Given I am on the 'Cart' page and the list of available delivery time frames are displayed,  when I select a time frame,  then the system sets the order's delivery time to the selected time frame. | | |

| **Story #8** | Checkout | **Priority** | Medium |
| --- | --- | --- | --- |
| **Estimate** | 3 |
| As a | customer | | |
| I want to | check out my order | | |
| So that | I can pay and receive the items I ordered | | |
|  | | | |
| Acceptance criteria | **Scenario: Check out - Success**  Given I am on the 'Cart' page and I added at least one item in the cart and I have selected a delivery option  when I click 'Check out',  then the system will direct me to the payment page.  **Scenario: Check out - No items in the cart**  Given I am on the 'Cart' page and the cart is empty,  when I click 'Check out',  then the system will display an error message, saying that my cart is empty.  **Scenario: Check out - delivery option not selected**  Given I am on the 'Cart' page and the cart is empty,  when I click 'Check out',  then the system will display an error message, asking to select a delivery option.  **Scenario: Payment - Success**  Given I am on the 'Payment' page,  when I enter valid credentials,  then the system will process the order and display a message saying that my order has been placed.  **Scenario: Payment - Fail**  Given I am on the 'Payment' page,  when I enter invalid credentials,  then the system will display an error message, saying that the credentials entered are invalid. | | |

| **Story #9** | Notification preferences | **Priority** | Medium |
| --- | --- | --- | --- |
| **Estimate** | 2 |
| As a | customer | | |
| I want to | customise my notification preferences | | |
| So that | I can select what I receive notifications for | | |
|  | | | |
| Acceptance criteria | **Scenario: Accessing the notification preferences**  Given I am logged in and I am on the profile page,  when I click 'Notifications',  then the system will direct the user to the 'Notifications' page, displaying a list of notifications that can be toggled.  Notification items include promos, and price drops on favorited items.  By default, all notifications will be turned off.  **Scenario: Turning on notifications**  Given I am logged in and I am on the 'Notifications' page,  when I click 'ON' for a notification,  then the system will update their preferences and send timely notifications accordingly.  **Scenario: Turning off notifications**  Given I am logged in and I am on the 'Notifications' page and a notification is turned on,  when the user clicks 'OFF' for notification,  then the system will update their preferences and stop sending the user notifications. | | |

| **Story #10** | Notifications - Special offers | **Priority** | Medium |
| --- | --- | --- | --- |
| **Estimate** | 2 |
| As a | customer | | |
| I want to | promptly receive notifications about special offers | | |
| So that | I can take advantage of the promos before they expire | | |
|  | | | |
| Acceptance criteria | **Scenario: Receiving notifications**  Given I am logged in and I have notifications for special offers on,  when a store releases a promo,  then the system will send a timely notification to the user.  the notification will be displayed as a browser notification. | | |

| **Story #11** | Favoriting an item | **Priority** | Medium |
| --- | --- | --- | --- |
| **Estimate** | 2 |
| As a | customer | | |
| I want to | ‘favourite’ an item | | |
| So that | I can be notified when there is a promo or discount for the selected item | | |
|  | | | |
| Acceptance criteria | **Scenario: Favoriting an item**  Given I am logged in and I have the item page opened,  when I click the star button,  then the system will add the item into favourites and notify me when there is a promo or discount.  **Scenario: Un-favoriting an item**  Given I am logged in and the favourited item page opened,  when I click the star button,  then the system will remove the items from favourites and stop notifying me when there is a promo or discount. | | |

| **Story #12** | Registration | **Priority** | 3 |
| --- | --- | --- | --- |
| **Estimate** | Low |
| As a | new customer | | |
| I want to | create an account | | |
| So that | I can save my personal details for future purchases | | |
|  | | | |
| Acceptance criteria | **Scenario: Registering with a new email**  Given I am not already registered on the website,  when I register with a new email,  then the system will create a new account with the inputted information.  **Scenario: New customer profile**  Given I am registering with a new email,  when I fill in the form for creating a profile,  then the system will create the profile with the inputs from the form.  **Scenario: Existing customer**  Given I have already registered on the website,  when I register with my email,  then the system will display an error message saying that they already have an account, and display a hyperlink to the 'Login' page. | | |

| **Story #13** | Logging in | **Priority** | Low |
| --- | --- | --- | --- |
| **Estimate** | 1 |
| As a | registered customer | | |
| I want to | log in to my account | | |
| So that | I can access my account and make purchases | | |
|  | | | |
| Acceptance criteria | **Scenario: Logging in with valid credentials**  Given I am on the login page and I have an existing account in the system,  when I enter a valid email address and password,  then the system will redirect me to the homepage, displaying my username on the top right corner.  **Scenario: Wrong password**  Given I am on the login page and I have an existing account in the system,  when I enter an incorrect password,  then the system will display an error message, displaying that the user had inputted an incorrect credential.  **Scenario: Email is not registered**  Given I am on the login page and I do not have an existing account in the system,  when I enter an unregistered email and password,  then the system will display an error message, displaying that the user had inputted an incorrect credential. | | |

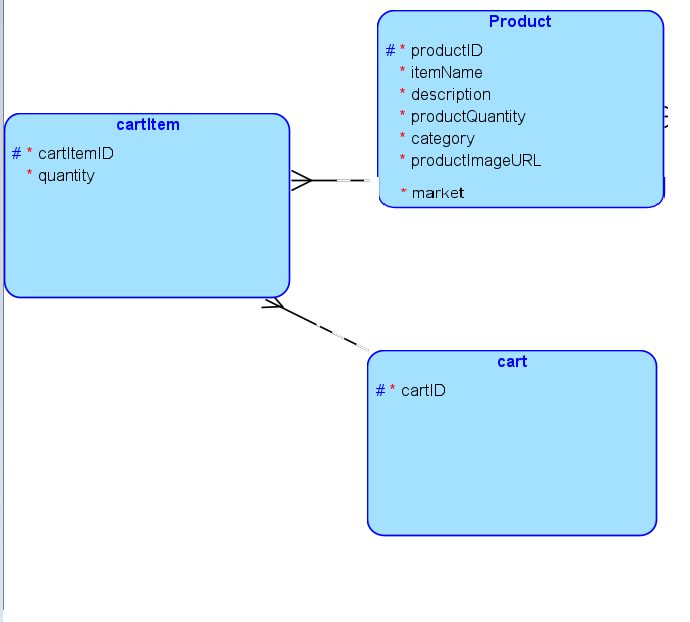
| **Story #14** | Personal details (Profile) | **Priority** | Low |
| --- | --- | --- | --- |
| **Estimate** | 1 |
| As a | registered customer | | |
| I want to | edit my personal details (name, address etc.) | | |
| So that | I can quickly check out my future purchases | | |
|  | | | |
| Acceptance criteria | **Scenario: Accessing profile**  Given I am logged in and I am on the homepage,  when I click 'Profile' on the navigation bar,  then the system will direct the user to the 'Profile' page.  *The profile includes name and address.*  **Scenario: Editing profile**  Given I am logged in and I am on my profile page,  when I click 'Edit profile',  then the system will display a form for changing the profile details.  **Scenario: Updating the profile**  Given I am logged in and I have the 'Edit Profile' form opened,  when I enter new information and clicks 'Update Profile',  then the system will update the profile information accordingly. | | |

# Unit Tests

| **Feature** | **Test** |
| --- | --- |
| Search by keyword | One result: Search for an item using the exact name.  Synonym: Search for an item using a keyword.  e.g., Searching “Basketball” should output “Molten Basketball”.  Multiple results: Search for items with a common name should output multiple results.  e.g., Searching for “Coke” should output multiple coke items. |
| Item viewing (Product info) | Displaying all products: Check if the function returns a complete list of product objects. |
| Cart Management | Displaying cart products:   * Check if the cart returns a complete list of cart products. * Verify the contents by checking item names.   Add to cart:   * Check if adding an existing item to the cart increases the total total cart quantity by one. * Ensure that non-existing items cannot be added to the cart.   Remove from the cart:   * Check if removing an item from the cart decreases the total total cart quantity by one. * Ensure that items that are NOT in the cart cannot be removed. |
| Notifications | Displaying all products: Check if the function returns a list of complete product objects that have a promotion and notifications on.  Find by keyword success: Searching with a valid keyword should output object/s.  Find by keyword fail: Searching with an invalid keyword should output an empty list. |

# UML

To create the database, we made a UML diagram to depict how different classes will associate with each other.



# Refactoring report

| **Area** | **Explanation** |
| --- | --- |
| REST API  [5/10/23] | **Scenario**  In the initial state of our API, we had one Repository class, 'SuperpriceRepository,' responsible for various functions, including product management, search, and cart operations. However, this approach presented several issues:   * Excessive Responsibilities   + The 'SuperpriceRepository' class had multiple responsibilities, violating the principle of single responsibility.   + This made the class complex and challenging to maintain. * Code Length and Readability:   + Each method within 'SuperpriceRepository' was quite lengthy, making the code difficult to read and organize. * Unit Test Challenges:   + The 'SuperpriceRepositoryTest' class, which contained unit tests for 'SuperpriceRepository,' also grew in size and complexity.   + Unit tests were failing due to connection clashes, complicating the testing process.   **Solution**  To address these challenges, we implemented the following solution:   * Refactoring Repository Classes:   + We divided the monolithic 'SuperpriceRepository' into three distinct classes: 'ProductRepository,' 'NotiRepository,' and 'CartRepository.'   + Each of these new classes was designed to have a single responsibility, which improved code organization and readability. * ResultSetUtil   + We also created ‘ResultSetUtil’ class containing the extractProduct(Resultset rs) method to reduce code duplication and to make changes easier to implement. * Isolating Unit Tests:   + By breaking down the Repository classes, we could isolate unit tests for each class, making it easier to maintain and run tests without interference. * Maintaining Monolithic Structure:   + While we split the Repository classes, we maintained a monolithic structure at the Service and Controller levels.   + This approach minimized dependencies and ensured a cohesive system design.   By following this approach, we successfully improved code organization, readability, and testability in our REST API, addressing the initial challenges we encountered. |
| REST API  Tests  [8/10/23] | **Scenario:**  As we expanded our feature set, the number of tests increased, impacting the backend testing process. When running the entire suite with commands like "mvn test”, the RepositoryTest classes executed smoothly individually, but collectively they caused database timeouts.  **Solution:**  To address this issue, we introduced mocking in tests that involved database manipulation (such as adding and removing products). This approach ensured that the tests remained independent of each other and helped mitigate the database timeout problem during overall test execution. Moreover, utilising mock also enables the tests to focus more on the behaviour, rather than the actual impact on the database. |