Software Requirements Specification

for

SuperPrice

Version 2.0 approved

Prepared by Darby, Laura, Lance, Lawrence, Saksham, Krisanahari

Group-P8-07

17/09/2023

Contents

1. Introduction

1.1 Product Scope

The product that will be covered in this document is SuperPrice; an application that is intended to enable an enhanced shopping experience for grocery shoppers. This document will cover the application's functional and non-functional requirements, the system architecture, the data model, the user interface design, assumptions and constraints, dependencies, as well as testing and acceptance criteria.

1.2 Purpose and Intended Audience

SuperPrice is intended to allow an easier and more efficient shopping experience for supermarket customers. Using this application, customers can find the best price option across multiple supermarkets and stores near them for the products that they intend to purchase. There is also an option for customers to have their items delivered to their doorstep. The project aims to develop a user-friendly platform to make it available for a wide range of users. This is done by prioritising a smooth and easy experience for everyone through an intuitive and straightforward user-interface. Furthermore, this document is drafted for the product's stakeholders; specifically - the software developers, the project manager, and the client.

2. Overall Description

2.1 Product Perspective

The SuperPrice - Price Matching and Delivery Application represents an innovative, standalone solution tailored to elevate the consumer shopping journey. Its primary objective is to refine the grocery shopping paradigm by offering a sophisticated platform where consumers can effortlessly compare prices from a myriad of retailers and coordinate deliveries. This application seamlessly integrates with a diverse range of local supermarkets and retail outlets, granting users access to a comprehensive repository of products and their respective prices. Such integration guarantees that consumers are furnished with precise and contemporaneous data regarding product prices and their availability. Architecturally, the SuperPrice application is web-centric, bifurcated into backend and frontend modules, and is designed to function in real-time, ensuring users receive the most recent data.

Central to the application's ethos is its commitment to economizing both time and financial resources for its users through an all-encompassing price-matching mechanism and streamlined delivery coordination. With an intuitive user interface at its core, SuperPrice is poised to serve a diverse demographic, from shopping novices to seasoned consumers. In addition, it offers a robust platform for users to impart their feedback via ratings for individual products, thereby augmenting its credibility and user trust.

2.2 **Product Functions:**

- Product Search and Categorization: Allows users to find specific products or browse through different categories.
- Price Comparison: Enables users to compare prices across different supermarkets.
- Delivery Organization: Facilitates the organization of deliveries with multiple options.
- User-Friendly Interface: Prioritizes simplicity and ease of use.
- User Ratings: Allows users to leave ratings for products.

2.3 **User Classes and Characteristics:**

- Novice Shoppers: Those new to online shopping who prioritize a user-friendly interface.
- Experienced Shoppers: Users familiar with online shopping platforms and looking for advanced features.
- Bargain Hunters: Users specifically looking for the best deals and price drops.
- Reviewers: Users who rely on or provide ratings for products.

2.4 Operating Environment

- Web-based application with backend and frontend components.
- Specifications:
    - **Code Repository:** Github
    - **Task Management:** Github Project
    - **Group Communication:** Microsoft Teams
    - **Documentation:** Office 365 online
    - **CI/CD tool:** Github Action
    - **Java version:** 17
    - **Back-end framework:** Spring Boot
    - **Front-end framework:** Node.js
    - **Build tool:** Maven
    - **Data-base:** H2-console with Flyway Data migration
    - **Unit testing framework:** Junit4 for Java, Front-end manual testing framework .
    - Migrated from H2 to mySQL
    - Docker for configuration management
    - AWS ECR for elastic container registries
    - AWS Beanstalk for elastic to live application

2.5 User Documentation

- Sign In
  - To sign in as a current user, type in your username and password into the following fields on the sign in page, and press the big green button Sign In to be signed into the application.
- Sign Up
  - If you do not already have an account, you can press "Don't have an account? Sign up" on the sign in page, which will take you to the sign up page. Whilst on the signup page, you need to enter in all the following details to become a new user: first name, last name, user name, email, password, address, notifications, card name, card number, card expiration date, card cvv. Once you have successfully put in these details, you can press the big green button Sign Up which will take you back to the sign in page, where you can sign in as a new user.
- Browse Products
  - Once you have signed into an existing account, you will be taken to the home page which is the product list page. This page will give you the ability to search all products. If you want to search by category, there is a bar on the left hand side of the page which you can scroll through and choose your desired category. Otherwise, you can search through the whole list of products (sorted in alphabetical order), or you can use the search bar on the right hand side of the product page which will show you a selection of results to your search.
- Product Details
  - To see a product's details, you can click on your desired product, which will bring you to a page specifically for this product. On this page, you can see details such as the product name, price, weight and rating. You can select the quantity of that product that you would like, and you can also give a rating of the product by using the 5 stars on the page. You are automatically given the cheapest price found from all stores, however, if you would like to compare the price against other stores, there is a drop-down on the price that shows you other stores that have this item, and their price for it. To add your item to your cart, you can click the green button "add to cart" which will add your item, and its quantity, straight to the cart.
- User details
  - As a user, to see your details, you can click the little person icon in the top right corner of the navigation bar. Once clicked, you can view all of your personal information. For privacy reasons, your card information will be hidden, but if you would like to see your information, all you have to do is click the box to reveal your information.
- Cart
  - To view your cart, you can click the little shopping cart icon in the top right corner of the navigation bar. Once clicked, you can view all the products that you have added to your cart. To delete an item from your cart, you can press the red bin button on the right corner of each item. To check out this cart, you can press the checkout button.
- Checkout
  - On the cart page, once you have clicked the checkout button, you will be taken to the checkout page. Here you can add your information for delivery and your card information. Once you have finalised your information, you can press the confirm order button.

2.6 Assumptions and Dependencies

2.6.1 Assumptions:

- The application assumes that the majority of users have access to a stable internet connection.
- The application assumes that all integrated supermarkets and stores provide accurate and timely data.
- It is assumed that users will primarily use the application for grocery shopping, rather than for other types of products.
- The application assumes that users will trust and rely on the reviews and ratings provided by other users.
- The application assumes that the majority of users will be using modern browsers and devices that support the latest web standards.

2.6.2 Dependencies:

- The application's functionality is dependent on the real-time data integration with various supermarkets and stores.
- Dependency on third-party payment gateways for processing transactions.
- The application might rely on third-party APIs for additional features like maps, notifications, etc.
- The success of the application is contingent on regular updates and maintenance to ensure accurate pricing and product availability.
- The application's performance might be influenced by the server's uptime and the efficiency of the hosting service.

---

3. User Stories

3.1 Product Search and Categorization – Saksham & Krisanahari

User Story 1: As a user, I want to be able to search for a specific grocery item.

Acceptance Criteria:

Scenario 1: Successfully search for a specific grocery item

Given: I am on the homepage of the online grocery website.

When: I enter the name of a grocery item into the search bar,

Then: The application should display a list of results that match the item name I entered. The results should include a photo of the product, the product name, and its price.

User Story 2: As a user, I want to be able to browse products by category.

Acceptance Criteria:

Scenario 1: Successfully browse products by category

Given: I am on the homepage of the online grocery website.

When: I click on a specific category, like ‘Fruits’ or ‘Vegetables’,

Then: The application should display a list of products that fall under the selected category. Each product listing should include a photo of the product, the product name, and its price.

User Story 3: As a user, I want to be able to checkout.

Acceptance Criteria:

Scenario 1: Successfully checkout

Given: I am on the cart of the online grocery website.

When: I click on checkout,

Then: The application should display the payment and delivery information screen.

3.2 Delivery Organization - Lance

User Story 1: As a user, I want to be able to select the date and time of delivery of my items.

Acceptance Criteria:

Scenario 1: Successfully select the date and time of delivery

Given: I am on the delivery setup page.

When: The application displays the calendar,

Then: I can select a date that I prefer for my delivery

Then: It will display available timeslots on that day which I can choose from. After selecting these, a summary of the details will be displayed. That is, the date and time of the delivery.

User Story 2: As a user, I want to be able to select the delivery option that I prefer.

Acceptance Criteria:

Scenario 1: Successfully choose the preferred delivery option

Given: I am on the delivery setup page.

When: the application displays the available delivery options,

Then: it will proceed accordingly to ask for further details about my chosen method of delivery.

3.3 User Sign In Request

User Story 1

As a user, I want to sign-in to my account.

Acceptance Criteria:

Given: I am on the login page.

When: I enter my username and password,

Then: the application will verify my details and sign me in if it is correct.

3.4 User Sign Up Request - Laura

User Story 1:

As a user, I want to create an account.

Acceptance Criteria:

Given: I am on the login page.

When: I click 'Don't have an account? Sign Up',

Then: the application should take me to a page where I can put my details in, and create a new account. Once the account has been created, I should be redirected to the login page to log in with my new account.

3.5 Price Comparison - Lawrence

Agile Epic: Price Comparison

User Story 1:

As a user, I want to compare prices across different supermarkets, so I can make informed decisions and find the best deals.

Acceptance Criteria:

Given that I am on the product details page,

When I select a product,

Then I should see a list of supermarkets in the local area offering that product and their corresponding prices.

And the list of supermarkets should be sorted in ascending order based on the product prices, with the lowest price displayed first.

User Story 2:

As a user, I want to compare prices of products in different categories across multiple supermarkets, so I can make informed purchasing decisions for my groceries.

Acceptance Criteria:

Given I am a registered user of the SuperPrice application.

When I browse through different product categories.

Then I should be able to see a list of products in each category.

And the list should display the prices of the products from various supermarkets.

And the products in each category should be sorted based on the lowest price available.

3.6 Database Implementation - Lawrence

Agile Epic: Design and Apply Database Schema for SuperPrice

User Story 1.

As a developer, I want to design and apply a database schema for SuperPrice so that the application can store, manage, and retrieve data efficiently.

Acceptance Criteria:

Given I have a clear understanding of the data requirements for SuperPrice,

When I design the database schema,

Then it should effectively represent all entities, relationships, and constraints necessary for the Super Price application.

Agile Epic: Flyway Used To Manage Database Migrations for H2 database

User Story 2.

As a database administrator, I want to use Flyway to manage database migrations so that I can ensure consistent and version-controlled changes to our H2 databases.

Acceptance Criteria:

Given I have an H2 database and a Flyway migration script,

When I execute Flyway commands,

Then the migration should be correctly applied to the H2 database

And I should be able to view the status of migrations using the Flyway info command.

Agile Epic: Implement H2-Console for Database Environment

User Story 3.

As a developer, I want to implement the H2-Console in the database environment so that I can easily interact with and monitor the H2 database via a web interface.

Acceptance Criteria:

Given I navigate to the H2-Console URL,

When I enter valid database credentials and attempt to login,

Then I should be able to access the H2 database and execute SQL statements.

Given I navigate to the H2-Console URL,

When I enter incorrect database credentials and attempt to login,

Then I should see an error message indicating a failed connection attempt.

Agile Epic: Input Sample Data into Database

User Story 4.
As a customer, I want to select products and place an order, So that I can purchase the items I need.
Acceptance Criteria:

Given a list of available products and my customer information,

When I select products with IDs 101 and 202, and place an order,

Then the database should record the order with the selected products, associate it with my customer information, and update the stock quantities of the purchased products.

User Story 5:

As a system administrator, I want to migrate our application's data from the H2 database to MySQL so that we can take advantage of MySQL's scalability, manageability, and other enterprise-level features.

Acceptance Criteria:

Given: that I have a backup of the H2 database,

When: I start the migration process,

Then: the system should create a MySQL database schema that mirrors the H2 schema.

Acceptance Criteria:

Given: that I have started the migration process,

When: data transfer begins,

Then: all data from the H2 database should be replicated accurately in the MySQL database.

Acceptance Criteria:

Given: that the migration process is complete,

When: I check the MySQL database,

Then: all tables, records, indexes, and constraints should be consistent with the H2 database.

Acceptance Criteria:

Given: that the migration has been successful,

When: I launch the application connected to the MySQL database,

Then: the application should operate normally with no loss of functionality or data.

Acceptance Criteria:

Given: that the application is running on the MySQL database,

When: I execute database-dependent operations,

Then: the response time and performance should be at least as good as, if not better than, when the application was using the H2 database.

3.7 Continuous Integration (CI/CD) - Lawrence

Agile Epic: Implement Continuous Integration

User Story 1.

As a developer, I want to have Continuous Integration (CI) implemented for our project so that my code is automatically tested and built whenever I push changes, ensuring that it integrates seamlessly with the existing codebase.

Acceptance Criteria:

Given that I have made code changes and pushed them to our version control repository,

When the changes are pushed,

Then the CI system should automatically trigger a build and test sequence for the latest changes.

3.8 User-Friendly Interface

User Story 1: As an inexperienced user I want to be able to use SuperPrice without any training or prior experience.

Acceptance Criteria -
Scenario 1: Successfully navigate the website for the first time with no experience
Given: I have no prior experience with the website, or similar websites
When: I interact with the website, to execute core functions
Then: The interface must be simple and clean enough to not overwhelm a new user, or someone who has not used a competitor's website before. Subheadings must be clearly sectioned and provide a concise description of the content that it holds.

4. System Architecture

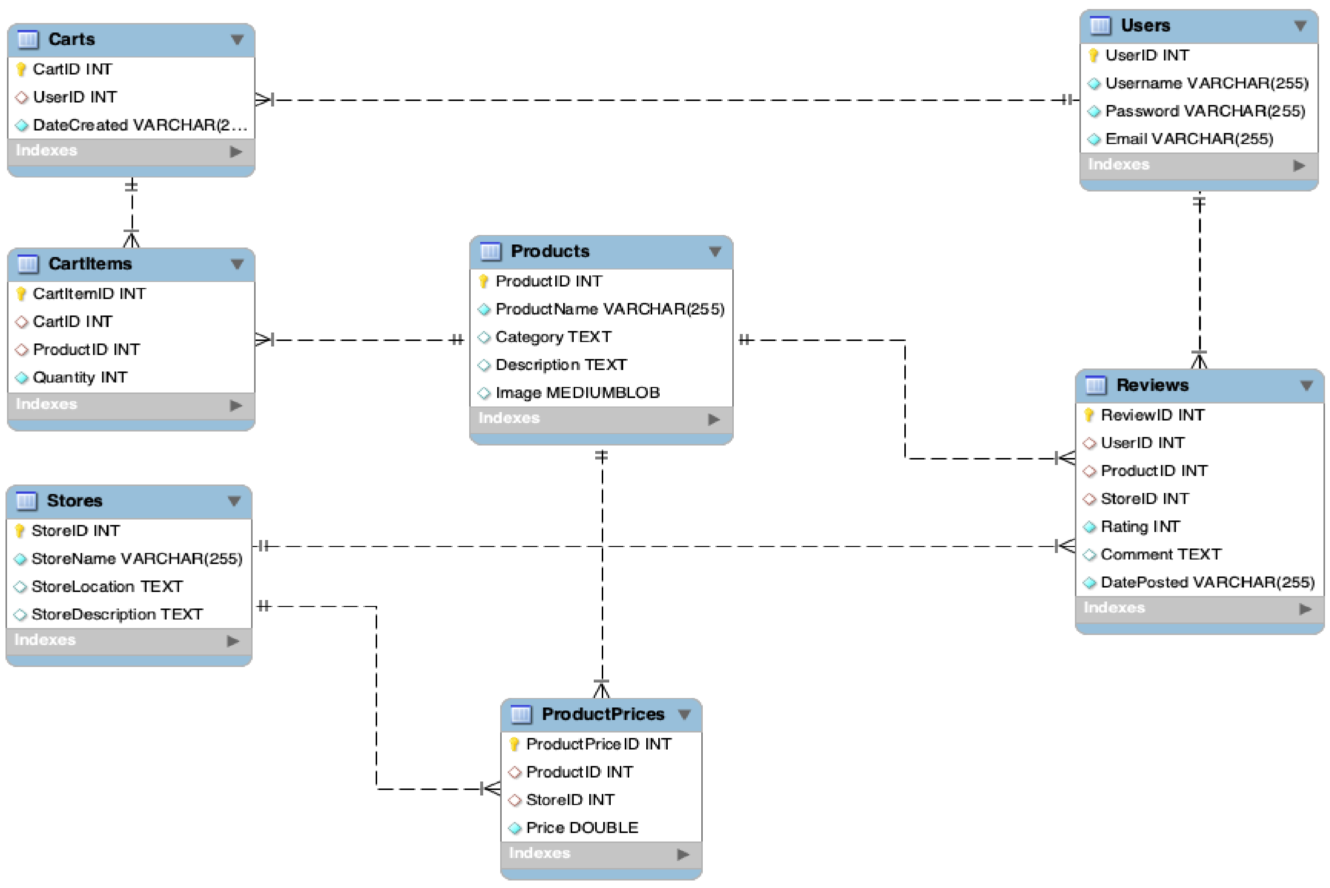
The system components include the Client, API Gateway (Rest API) and the services of the application.

The API Gateway being followed is REST API. This component serves as the gateway between the client and the server when the application's resources are required or requested.

Next, are the services of the application, namely, Product Information, User Information, and Delivery Organization.

- The Product Information component involves any product-related functionality within the application. For example, the product search and the product prices.
- The User Information component is composed of user-related, such as their name, purchases, order delivery details, and notification preferences.
- The Delivery Organization component involves the application's delivery feature.

The interaction of these components starts with the client-side of the system. A request is sent to the server by the client which the API Gateway handles. As the request is made, the server validates this and proceeds to communicate with the system's services/resources and database. Finally, the system captures the result that will be conveyed to the client as a response to the request.



6. Other Nonfunctional Requirements

6.1 Performance Requirements

***Response Time**: The application should display search results within 2 seconds of the user initiating a search.*

***Load Time:** The web pages of the application should load within 3 seconds under normal network conditions.*

***Scalability:** The system should support up to 100,000 concurrent users without any degradation in performance.*

***Real-time Data:** Price updates and availability checks should be reflected in real-time to the end-users.*

6.2 Safety Requirements

***Availability:** The system should achieve a minimum uptime of 99.9%.*

***Backup:** All data should be backed up daily, and there should be a mechanism in place for quick data restoration in case of any failure.*

***Disaster Recovery:** A disaster recovery plan should be in place to ensure the application's availability in case of any catastrophic events.*

***Error Handling:** The application should provide clear error messages and should be able to handle unexpected failures gracefully, ensuring that users do not experience abrupt crashes.*

6.3 Security Requirements

***Data Encryption:** All data transferred between the client and the server should be encrypted using industry-standard encryption protocols.*

***User Authentication:** The application should implement multi-factor authentication for user accounts.*

***Data Privacy:** User personal and financial information should be stored in encrypted formats and in compliance with GDPR or other relevant data protection regulations.*

***Session Management:** User sessions should automatically expire after 15 minutes of inactivity.*

***Audit Logs:** All user activities and system transactions should be logged and stored for at least 6 months.*

6.4 Software Quality Attributes

User Interface: *The UI should be intuitive and user-friendly, with a focus on ease of navigation.*

Mobile Responsiveness: *The application should be fully functional and responsive on various devices including desktops, tablets, and mobile phones.*

Accessibility: *The application should be accessible for users with disabilities, adhering to the Web Content Accessibility Guidelines (WCAG) 2.1 Level AA.*

Multilingual Support: *The application should support at least English and one other local language, with capabilities to easily add more languages in the future.*

API Integration: *The system should support seamless integration with various supermarket and store databases through well-defined APIs.*

Notifications: *The notification system should be capable of sending instant notifications to users without any delay.*

6.5 Business Rules

Automated Testing: *The system should support automated unit and integration tests to ensure the robustness of the application.*

Code Quality: *The source code should adhere to industry-standard coding practices and should be well-commented for easier maintainability.*

Updates & Patches: *The system should support seamless updates and patches without causing significant downtime.*

7. Use Cases

Use Case 1	Search for an item/product	
Actors	User/Customer	
Preconditions	The user must be on the homepage.	
Postconditions	A list of items that match the name of the product entered and the corresponding price.	
Flow of activities	**Actor**	**System**
	1. The user selects the search bar 2. The user enters the product that they want to purchase.	1. The search bar allows the user to enter a text. *2.* The application redirects the user to another page to show the result of their search.
Alternative Flows/ Exceptions	1. The product entered cannot be found (I.e., it is not in the system) - The application will then display a prompt to the user that this is the case.	

Use Case 2	Select the date and time of delivery	
Actors	User/Customer	
Preconditions	The user must be on the delivery setup page.	
Postconditions	The available timeslots on that selected day will be displayed and a summary of the delivery's details will be shown.	
Flow of activities	**Actor**	**System**
	1. The user first selects the date of delivery. 2. The user selects their preferred timeslot for their delivery. 3. The user selects proceed.	1. The application gets the available timeslots for the selected day and displays this. *2.* --- 3. The application displays a summary of the delivery details (date and time).
Alternative Flows/ Exceptions	1. The selected date does not have the user's preferred timeslot - The user selects the "back" button to go back to the calendar and choose another preferred date. - Then proceeds to number 2 of flow of activities.	

Use Case 3	Compare prices across different supermarkets	
Actors	User/Customer	
Preconditions	The user must be on the product details page.	
Postconditions	A list of supermarkets in the local area offering the selected product will be displayed.	
Flow of activities	**Actor**	**System**
	1. The user selects a product.	*1.* The application fetches all the supermarkets that sell the product and displays a list.
Alternative Flows/ Exceptions	1. The product selected cannot be found in the local area. - The application will then display a prompt to the user that this is the case.	

Use Case 4	Browse products by category	
Actors	User/Customer	
Preconditions	The user must be on the homepage.	
Postconditions	A list of products that fall under the selected category will be displayed. Each product will include its corresponding price.	
Flow of activities	**Actor**	**System**
	1. The user selects a specific category (for example, 'Fruits and Vegetables")	*1.* The application redirects the user to another page to show the result of their selection.
Alternative Flows/ Exceptions	1. The user cannot find the product that they want in their selected category. - Refer to use case 1 instead.	

Use Case 5	Design and Apply Database Schema for SuperPrice	
Actors	Developer	
Preconditions	Developer has access to the system where the application SuperPrice is being developed. Developer understands the requirements and data needs of the SuperPrice application.	
Postconditions	A database schema has been designed and applied. The application SuperPrice can efficiently store, manage, and retrieve data based on the schema.	
Flow of activities	**Actor**	**System**
	1. Analyzes the data requirements for SuperPrice. 3. Designs the database schema based on the analyzed requirements. 5. Applies the schema to the SuperPrice system.	2. Provides necessary data and interfaces for analysis. 4. Accepts the schema design. 6. Updates the database with the new schema. Confirms successful schema application.
Alternative Flows/ Exceptions	▪ The designed schema is found to be inefficient or not meeting the requirements. o The developer revises the schema and reapplies it to the system. ▪ Errors occur while applying the schema. o The developer debugs and corrects the schema before reapplying.	

Use Case 6	Implement Continuous Integration for the Project	
Actors	Developer	
Preconditions	Developer has access to the codebase of the project. The project has a version control system in place. Developer has knowledge of Continuous Integration (CI) tools and practices.	
Postconditions	Continuous Integration (CI) is set up and operational for the project. Code is automatically tested and built whenever changes are pushed, ensuring seamless integration.	
Flow of activities	**Actor**	**System**
	1. Sets up a CI server or uses a CI service. 3. Configures the CI server or service to watch the project's repository for changes. 5. Pushes changes to the codebase.	2. Registers and configures the CI setup. 4. Observes the repository for changes. 7. Triggers the CI process. Automatically tests and builds the project. - Generates and provides CI reports.
Alternative Flows/ Exceptions	▪ The CI process detects errors or failures in the code. o The developer reviews the error reports, fixes the issues, and pushes the corrected code. ▪ The CI process experiences issues or becomes non-operational.	

	○ The developer troubleshoots the CI setup and makes necessary adjustments.

Use Case 7	Sign in	
Actors	User/Customer	
Preconditions	The user must be on the login page	
Postconditions	The user signs into their profile.	
Flow of activities	**Actor**	**System**
	1. The user first enters their username and password. 2. The user selects login.	1. The application handles the form changes. *2.* The application handles the username and password input and verifies if the user is in the database. *3.* The application finds the user in the database and proceeds to log them in.
Alternative Flows/ Exceptions	*1.* The user's credentials cannot be found in the database (he/she is not registered). • Proceed to sign up/register page.	

Use Case 8	Sign up	
Actors	User/Customer	
Preconditions	The user must be on the login page	
Postconditions	The user creates their account.	
Flow of activities	**Actor**	**System**
	1. The user first enters their personal details: first name, last name, username, email, password, address, notifications, card name, card number, card expiration, card cvv 5. The user has now created their account, and is redirected to the sign in page	2. The system checks all required fields have been filled 3. The system checks if this user is already in the database 4. If the user isn't in the system, the user is added to the database
Alternative Flows/ Exceptions	*1. The user has not filled in all the required fields* *2. The user has already got an account* *3. The user can click "have an account? Sign in" to sign in instead of signing up*	

8. User Interface Design:

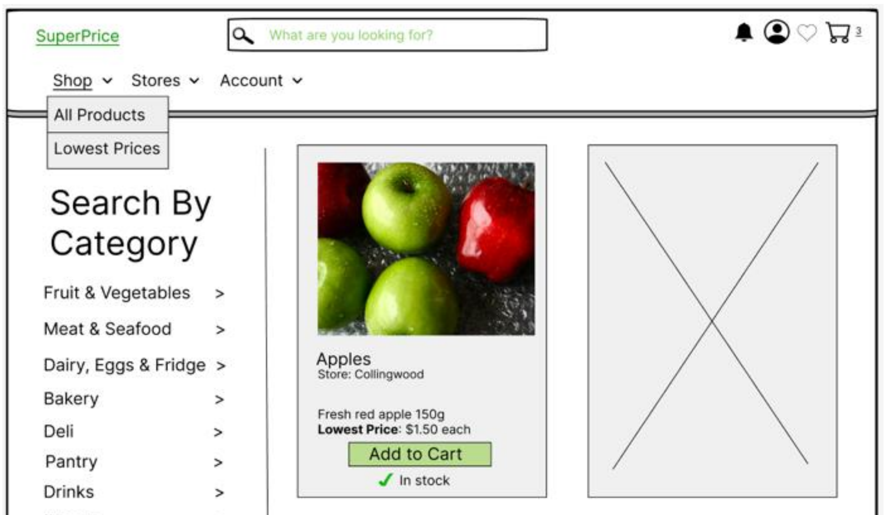
https://www.figma.com/file/iTvWwwcLNZOSkjWoyK4Ptc/SEPT?type=design&node-id=0%3A1&mode=design&t=LdbC65w11mw5y7L1-1

8.1 Product details page

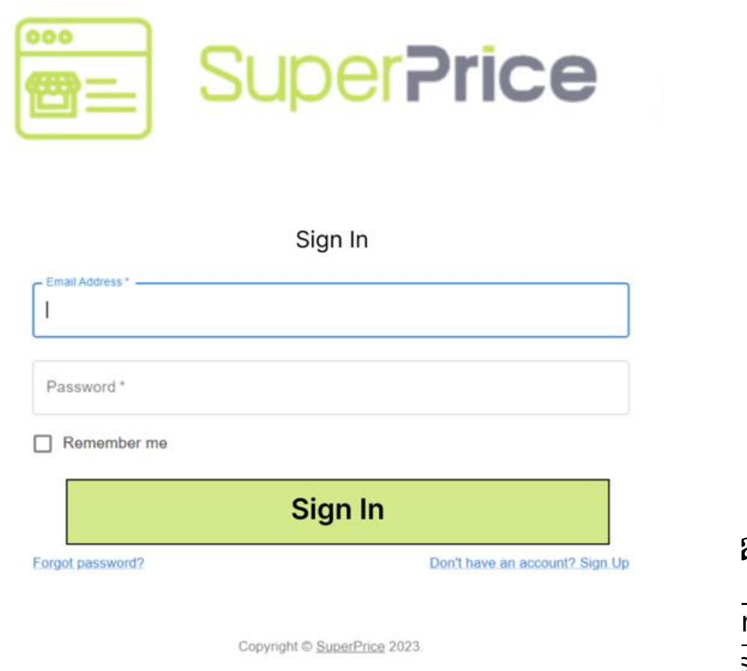
8.2 Cart page

8.3 Checkout page

8.4 Product List



8.5 Sign in Page


Sign In

Email Address *

Password *

Remember me

Sign In

Forgot password?

Don't have an account? Sign Up

Copyright © SuperPrice 2023.

ated)

		riteria	Testing
Categorization	able to search for a specific grocery item	search for a specific grocery item	There is a search bar that is on the Product list page which upon use, the user is able to search for the item they want, and whilst typing, the products will be displayed according to the user's search.

Product Search and Categorization	As a user, I want to be able to browse products by category	Successfully browse products by category	The default home page shows all of the products available. Alternatively, users can click on the tab "shop" from the navigation bar which will also display a hyperlink named "All products" which will take the user to this page. On this page, the user will be able to browse products under their desired category.
Product Search and Categorization	As a user, I want to be able to checkout	Successfully checked out	In the top corner of the navigation bar, there is a cart icon that users can click to view their cart. Once clicked and on the cart page, users can review their items and finalise their transaction by clicking on the "checkout" button which will take the user to another page where they can safely and successfully pay for their items.
Delivery Organization	As a user, I want to be able to select the date and time of delivery of my items	Successfully select the date and time of delivery	Once the user has clicked on the cart page/ once the user has reviewed their cart, they can continue to the delivery page. This will show options for delivery and also gives the user the ability to choose which date and time they would like the order to be delivered. The delivery summary

			will be updated and displayed to the user once they have clicked on their desired options, as well as any additional fees.
Price Comparison	As a user, I want to compare prices across different supermarkets, so I can make informed decisions and find the best deals	Successfully compare the prices across different supermarkets	When the user has selected on a product, the page will come up with product information. On the product page, the cheapest price will be the default price, but there is also the ability to select the drop-down box which shows the user the price of this product at other stores.
Price Comparison	As a user, I want to compare prices of products in different categories across multiple supermarkets, so I can make informed purchasing decisions for my groceries	Successfully compare prices of products in different categories across multiple supermarkets	The user is able to select different categories on the product list page. Once the category has been selected, the user can find a product of their choice. Once the product has been selected, the user can see the price differences across different stores on the product page.
Database Implementation	As a developer, I want to design and apply a database schema for SuperPrice so that the application can store, manage, and retrieve data efficiently, Furthermore, implement a database system that is production line graded and able to be	1. A comprehensive database schema is designed which covers all the necessary entities and relationships. 2. The schema is successfully applied to the database. 3. The application can efficiently store,	1. Test the database for any normalization errors. 2. Run CRUD (Create, Read, Update, Delete) operations on the database to ensure data integrity. 3. Benchmark the database for performance during high load scenarios.

	efficiently scaled, through the utilisation and transition from h2 to MySQL	manage, and retrieve data without errors.	
Implement Continuous Integration	As a developer, I want to have Continuous Integration (CI) implemented for our project so that my code is automatically tested and built whenever I push changes, ensuring that it integrates seamlessly with the existing codebase.	1. A CI pipeline is set up that automatically triggers on code pushes. 2. Automated tests run successfully on every code push without any manual intervention. 3. Developers receive feedback on the results of the build and test phases promptly.	1. Push a sample code change to trigger the CI pipeline. 2. Verify that the code is automatically built and tests are executed. 3. Ensure that the feedback mechanism (e.g., notifications) works as intended and informs the developer of the CI process outcomes. Is there anything else you'd like to add or modify?
User-Friendly interface	As an inexperienced user I want to be able to use SuperPrice without any training or prior experience	Successfully navigate the website for the first time with no experience	The website is very self-explanatory, with large fonts for important headings and subheadings. There are big blue buttons on the website to help users be drawn to where to click, for example "checkout" is in a big blue box to help users see where to click.

10. Product Backlog (Updated)

ID	ITEM	OWNER	PRIORITY	EFFORT	STATUS	DEFINITION OF DONE	NOTES	EPIC
1	Frontend-Backend Integration	Backend	High	13	Completed	Frontend and backend communicate smoothly	Requires thorough testing and debugging	Application CI/CD
2	Delivery Management System	Backend	High	13	Completed	Users can schedule and track deliveries	Requires integration with third-party	Product Delivery
3	Price Comparison Algorithm	Backend	High	13	Completed	Application compares prices across stores	Requires complex algorithm development	Product Search & Comparison
4	Product Search Functionality	Backend	High	8	Completed	Users can search for products	Implement search algorithms and filters	Product Search & Comparison
5	Product Database Setup	Backend	High	8	Completed	Database stores product information	Integrate with chosen database (e.g., MySQL)	Product Search & Comparison
6	User Registration and Login	Backend	High	5	Completed	User can register and log in securely	Requires encryption and validation	User Management
7	User Interface Design	Frontend	High	8	Completed	Application has a user-friendly interface	Collaborate with UI/UX designers	UX/UI Design
8	Category Browsing	Frontend	High	5	Completed	Users can browse products by category	Integrate with backend for category data	Product Search & Comparison
9	User Ratings	Frontend	Medium	8	Completed	Users can leave reviews and ratings	Integrate with backend for review storage	User Management
10	User Profile Management	Backend	Medium	5	Completed	Users can edit profile settings	Allow users to update personal information	User Management
11	Notifications	Backend	Medium	8	Completed	Users receive price drop notifications	Implement notification system	Alerting & Notifications
12	CI/CD Implementation	DevOps	Medium	8	Completed	Automatic testing and deployment set up	Integrate with GitHub Actions	Application CI/CD
13	Performance Optimization	DevOps	Medium	8	Completed	Application is optimized for performance	Fine-tuning of backend and frontend	Application

								Enhancement
14	Delivery Options Setup	Backend	Medium	5	Completed	Users can choose delivery options	Implement flexible time slot selection	Product Delivery
15	Deployment and Hosting	DevOps	Medium	5	Completed	Application deployed to recommended env	Utilize recommended hosting environment	Application CI/CD
16	User Documentation	Documentation	Low	5	Completed	Comprehensive user guide and documentation	Essential for user onboarding	User Management

11. Sprint 2 Burndown Chart

12. Sprint 2 Backlog

PBI ID	PBI NAME	TASK ID	TASK	ASSIGNED TO	NOT STARTED	IN PROGRESS	COMPLETED	NOTES
PBI-1	User Interface Enhancements	#28	Sign-In Page UI	Saksham			✓	
PBI -1	User Interface Enhancements	#30	Sign-up Page UI	Laura			✓	
PBI-1	User Interface Enhancements	#29	Product Details Webpage	Saksham			✓	
PBI-1	User Interface Enhancements	#36	Show the entire list of products webpage	Saksham			✓	
PBI-1	User Interface Enhancements	#37	Search for a specific Item	Saksham			✓	
PBI-2	Frontend and Backend Integration	#32	Integrating Frontend with the Backend	Hari/Lance			✓	
PBI-3	User Navigation	#33	Breadcrumbs Menu for easy navigation	Saksham			✓	
PBI-3	User Navigation	#34	FilterBar to search through list of products	Saksham			✓	
PBI-3	User Navigation	#48	Create links on nav bar to different pages	Laura			✓	
PBI-4	User Reviews & Ratings	#49	Frontend design for user reviews and ratings	Saksham			✓	
PBI-4	User Reviews & Ratings	#50	Backend API for user reviews and ratings	Lawrence			✓	
PBI-5	Price Comparison	#53	Optimization of price comparison	Lawrence			✓	

			for real-time data					
PBI-6	Improved User Experience	#56	Enhanced sorting feature for product list	Saksham			✓	
PBI-6	Improved User Experience	#57	Enhanced filtering feature for product categories	Saksham			✓	
PBI-6	Improved User Experience	#58	Improve overall site layout for intuitive navigation	Saksham/Laura			✓	
PBI-7	Backend Database	#59	Database schema refinement	Lawrence			✓	
PBI-7	Backend Database	#60	Database connection pooling	Lawrence			✓	
PBI-7	Backend Database	#61	Data backup automation	Lawrence			✓	
PBI-8	Backend Optimization	#62	API response caching	Hari			✓	
PBI-8	Backend Optimization	#63	Backend load balancing	Hari			✓	
PBI-9	Backend Security	#64	Implement JWT token authentication	Lance			✓	
	Backend	#65	Backend Notifications	Lance			✓	

13. Sprint Retro

Project Sprint Planning Notes
Team: 7
Sprint: 2
Attended: Laura, Lance, Saksham, Lawrence, Krisanahari
Scrum Master: Laura Gatt
Product Owner: Prakash
Development team: Laura, Lance, Saksham, Lawrence, Krisanahari

1. Things That Went Well

What went well? What the team is happy about?

- We had good communication throughout the sprint.
- The team had clear distinction of the tasks allocated to each member.
- Everyone was concerned about each other's progress and were helping each other when needed.
- The allocation of tasks for frontend, backend, and database was clearly agreed upon.
- The end goal for the sprint was clear and straightforward.

2. Things That Could Have Gone Better

What could have gone better? What the team could improve?

- Everyone having a good and clear understanding of how the application is built (I.e., how everything is connected) could have brought out a better outcome for this sprint.
- The team could have been a little better at communicating new ideas and developments
- The team could have organised a couple more meetings to touch base

3. Things That Surprised Us

What wasn't expected?

- It was surprising how much effort it would take for the docker and aws section of the sprint
- We were also surprised by the amount of trial and errors before actually having everything running smoothly and properly.
- It was interesting how many people it took to get things up and running.

4. Lessons Learned

What have you learned from the above points?

- Next time, it is a good idea to plan and set clear tasks before tackling a big task
- Next time, make sure that you make it clear to all team members what you are working on
- Next time, expect that it might take a lot of people to complete one task

5. Final Thoughts

Things to Keep

- Meetings and constant communication are necessary to keep for the team to achieve the goal.
- Constant use of the project board and communication tools to update everyone on their progress.

Things to Change

- Inconsistent routine or schedule for meetings and communication between team members.
- Better planning of the tasks to be done and allocation of participation in the project.

Summary

Overall, the sprint went well as the goal was achieved. Good communication was practised throughout but could be improved by having a consistent routine or schedule and response time between team members to avoid blockage of tasks. We are all very proud of each others contribution to this project and are happy with the ultimate outcome of this website.

14. Appendix