Software Requirements Specification

for SuperPrice

COSC2299: Major Project - Milestone 1

Date: 17th September 2023

Group P8-01

Prepared by:

Anthony Imani - s3950363

Dylan Khan - s3916841

Ledy - s3409664

Ronald Ho - s3436258

Thomas Yao - s3832771

Victoria Needs - s3811305

Table of Contents

1. Introduction

1.1 Purpose

Online shopping has risen over the last decade and led to an influx of platforms which service their audience through the web. The intent of this project is to develop a price-matching web application service which can facilitate the purchasing and delivery of groceries across a range of retailers. Users of SuperPrice will be able to pick the best deals in their area then choose a time for delivery all in one place.

1.2 Product Scope

As mentioned in the purpose; the core function of this product is to facilitate the purchasing of groceries and delivery of goods to the user. To further enhance the user's experience several key features will be included such as: a *User-Friendly Interface*, *Product Reviews*, *Price Comparison* and *Notifications & Alerts*.

1.3 Product Functions

- Browse Products
- Search Products
- Add Products to Cart
- View Order History
- Create Account/Login
- Organise Delivery
- Checkout
- Review Product

1.4 Design and Implementation Constraints

1.4.1 Feedback & Iteration

The reliance on stakeholder input on the implementation of requirements and deliverables of this project can lead to a delay in meeting deadlines. Members must be thorough when gathering requirements and ask for concise instruction where there might be an ambiguity.

1.4.2 Design Conventions & Standards

As newer developers it is more likely that code that is developed does not follow industry standards. Issues can arise from the use of improper naming conventions or document formatting such as difficulties in troubleshooting and maintainability.

1.4.3 Data Security

Given the software handles sensitive information such as financial information secure coding practices are essential. The cost of implementing these controls may inhibit the progress of development, however, the use of industry-standard encryption protocols is a necessity.

1.5 Assumptions and Dependencies

1.5.1 Backend

1. Spring Boot framework for developing microservices
2. Spring Cloud for implementing microservices architecture
3. Spring Cloud Gateway for API routing
4. Spring data JPA for data access, manipulation and storage
5. Maven for testing, building and dependency management

1.5.2 React

1. React-router-dom: handling routing within the app
2. Axios: to make HTTP requests to backend
3. Chakra-ui: component library to make the styling more consistent and clean
4. Moment or some other date time library - important for delivery times
5. Graphql/websocket library to support real-time data updates
6. Formik: if advanced forms are required
7. Jsonwebtoken or auth0 to handle auth
8. react-payment

1.5.3 Assumptions and Constraints

- Users will need to login to access all features
- It is assumed that the database is updated in real time based on local supermarket data
- It is assumed that there is existing delivery infrastructure in place - e.g. through Uber or another app, this app itself will not be managing the actual delivery of items
- Constraints:
- Not all items may support delivery services
- The app may need top handle large traffic during sales or special events e.g. christmas
- This app will only be available on the web - there is no app-solution

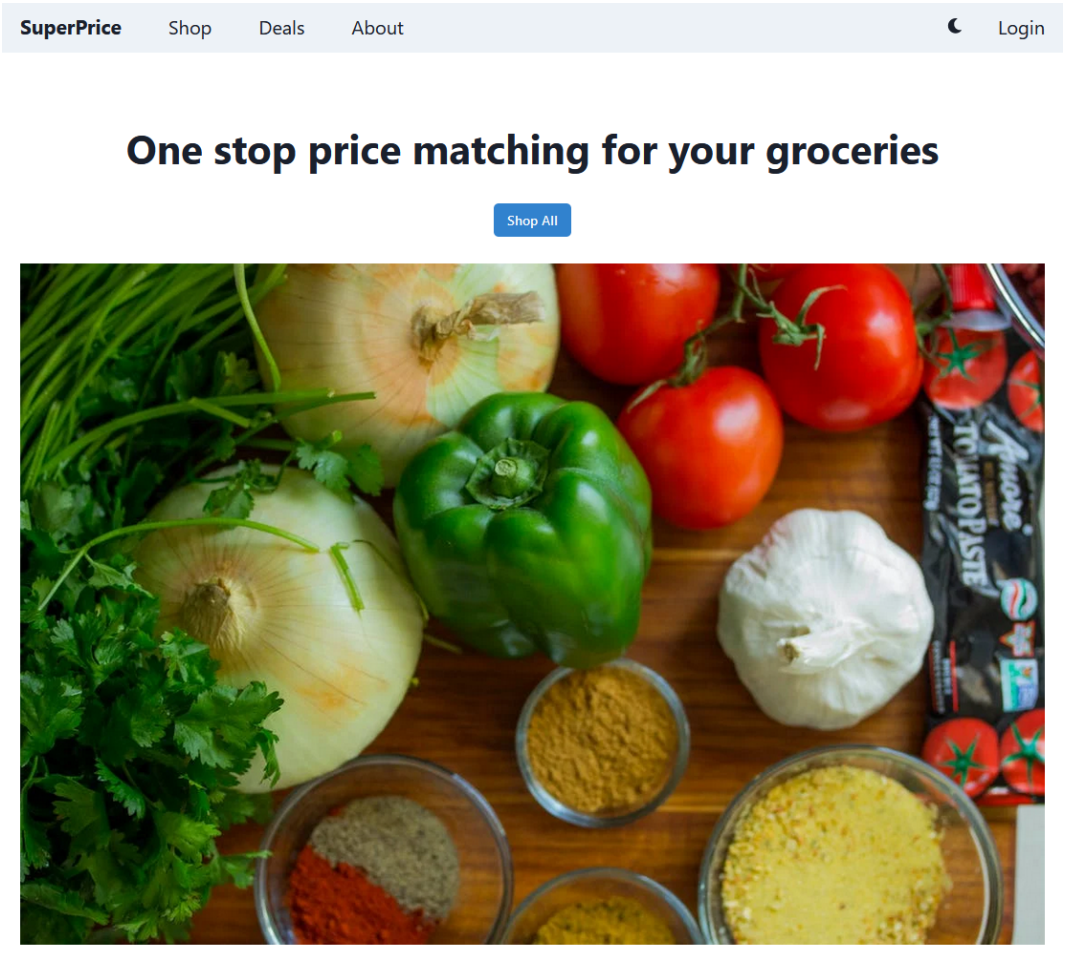
1.5.4 External APIs & Libraries

- Google maps for showing store locations and route calculation, display and real-time delivery tracking.

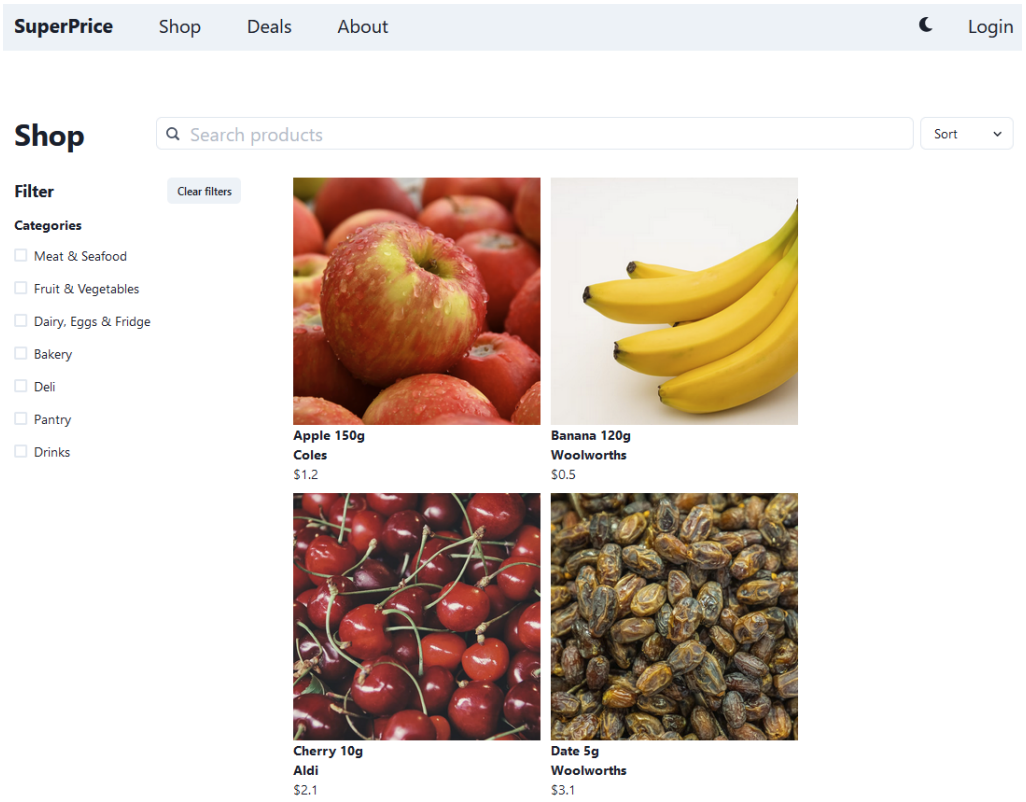
2. User Interfaces

The design for the SuperPrice is a fairly minimalistic web design using navigation bar as its main interface for users to navigate to different pages from Homepage, Product page to Cart page. In our design, we also make sure we have search functionality so that users can easily use the search bar to find the product that they are looking for.
In the Shop page (image 2.1.2) we also design a filter component where users can easily filter their search based on the product category that they are looking for.
In the Product page (image 2.1.3) users will be able to see the item that they have selected. In the same page, at the bottom section of the page, users can also see similar items that the user might be interested in to give users more options to consider before adding an item into their cart.
We have implemented a colour switcher (image 2.1.4) to change the colour theme from light to dark colours to provide users with a feeling of extra personalisation, and better usability and accessibility for the users of SuperPrice.
As an MVP, we will only be accommodating a desktop view. Below are several snapshots of the SuperPrice web app. More details of the design can be in this [figma design here](#).

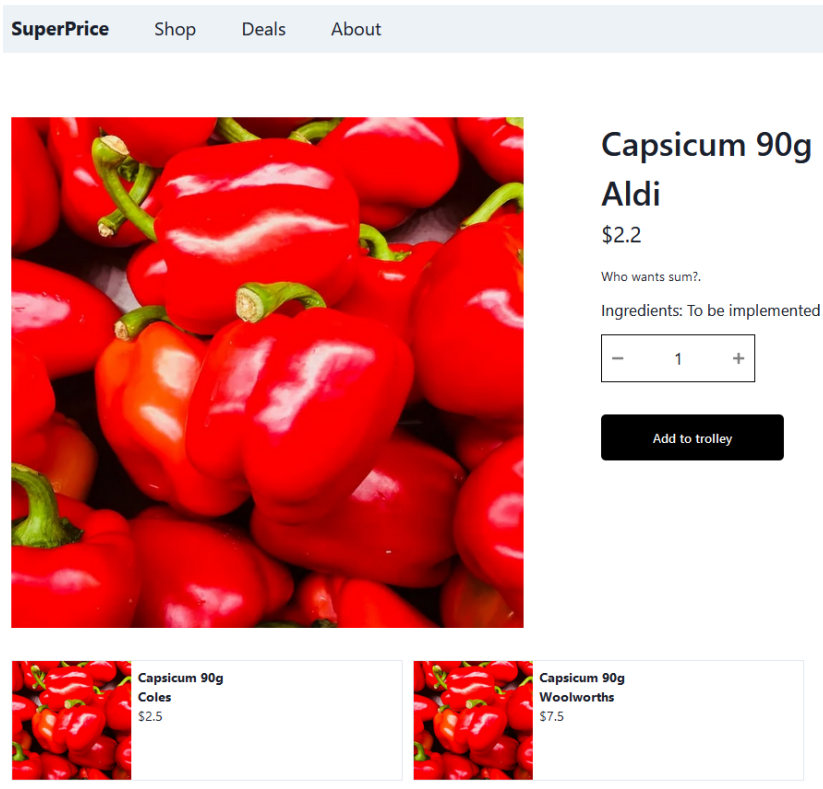
2.1 Homepage



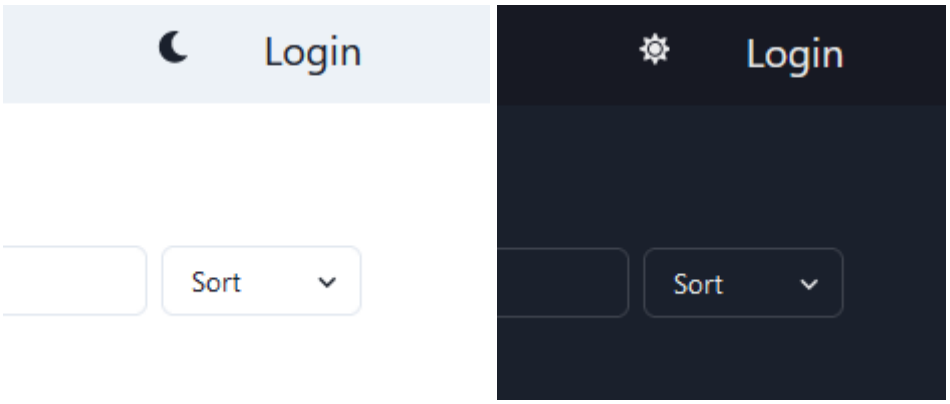
2.2 Shop page



2.3 Product page



2.4 Theme switcher



3. System Features

3.1 User Accounts

3.1.1 Create New Account

Description and Priority

Use Case ID: ACC01 **Priority:** Low

Key Feature: No

This feature allows users to register an account with the SuperPrice application.

Stimulus/Response Sequences

1. The user clicks the option to “Create Account”
2. The System displays the Create Account page
3. The user enters their details (name, email, password) and clicks “Confirm”
4. The system validates the user details
5. The system displays a confirmation of account creation
6. The system displays the home page with user name in top corner of page

Functional Requirements

REQ-1: *The user shall only be able to create an account with a valid email and password*

REQ-2: *The system shall validate the user credentials and save credentials via hashed password encryption mechanisms*

REQ-3: *The system shall send an email to the user’s email address to confirm successful account creation and request email validation*

3.1.2 User Login

Description and Priority

Use Case ID: ACC02 **Priority:** Low

Key Feature: No

This feature allows users to log in to their account

Stimulus/Response Sequences

1. The user clicks the option to “Login”
2. The system displays fields for email and password
3. The system authenticates the user
4. The system displays the home page with logged in user name in top corner of page

Functional Requirements

REQ-1: *The user shall only be able to log in with valid credentials*

REQ-2: *The system shall authenticate login using hashed password encryption mechanisms*

REQ-3: *The system shall send an email to the user’s email address to confirm successful account creation and request email validation*

3.1.3 Update Account Information

Description and Priority

Use Case ID: ACC03 **Priority:** Low

Key Feature: No

This feature allows users to edit/update their account information such as name, email, password, and location

Stimulus/Response Sequences

1. The user clicks the option to “Edit Account”
2. The system displays the user account current information
3. The user edits the current details of selected information
4. The user clicks the “Save” button
5. The system displays the relevant confirmation/error message
6. The system updates the account information

Functional Requirements

REQ-1: *The system shall allow for a logged in user to edit their account Information*

REQ-2: *The system shall validate information (email, password)*

REQ-3: *The system shall send a notification to the user of successful changes*

3.1.4 View Order History

Description and Priority

Use Case ID: ACC04 **Priority:** Low

Key Feature: No

This feature allows users to view their order history and the details of each order. This feature also gives users the option to reorder a past order.

Stimulus/Response Sequences

1. The user selects option to “View Order History”
2. The system displays a list of user orders (newest to oldest)
3. The user has the option to select order for more details
    a. The system displays itemised order information
    b. The user can select “Reorder”
    c. The system takes the user to “View Cart” page upon reorder

Functional Requirements

REQ-1: *The system shall provide option to view past orders only if user is logged in*
REQ-2: *The system shall provide chronological order history from newest to oldest*
REQ-3: *The system shall allow users to select an order from the list*

3.2 Products and Product Reviews

3.2.1 Search Product

Description and Priority

Use Case ID: PROD01 **Priority:** High

Key Feature: Yes

This feature allows users to search for products by name

Stimulus/Response Sequences

1. The user clicks on the search bar
2. The system allows the user to enter a product name
3. The system displays list of matching products
4. The user selects the product from list

Functional Requirements

REQ-1: *The system shall search the database for products that match the user criteria*
REQ-2: *The system shall display matching items with name, image, price to user*
REQ-3: *The system shall inform the user if no matches are found*

3.2.2 Browse Products

Description and Priority

Use Case ID: PROD02 **Priority:** High

Key Feature: Yes

This feature allows users to browse products by category or special offers

Stimulus/Response Sequences

1. The user clicks on the option to browse by category
2. The user selects their desired category
3. The system searches the database for products fitting the category criteria
4. The system displays the products to the user

Functional Requirements

REQ-1: *The system shall ensure all products in the system are categorised*
REQ-2: *The system shall allow a user to select a category from a list of categories*
REQ-3: *The system shall allow the user to select multiple categories*
REQ-4: *The system shall allow the user to select a product from category list*

3.2.3 View Product

Description and Priority

Use Case ID: PROD03 **Priority:** High

Key Feature: Yes

This feature allows users to view product information including description, price, rating/reviews, and compare product prices

Stimulus/Response Sequences

1. The user clicks on a product to view
2. The system displays product information page which includes product price, description, ratings, and price comparisons
3. The user will have the option to add the product to their cart

Functional Requirements

REQ-1: *The system shall store product information including price, description, rating, image in the database*
REQ-2: *The system shall retrieve item information from database to be displayed*
REQ-3: *The system shall allow the user to add the item to their shopping cart*

3.2.4 Review Product

Description and Priority

Use Case ID: PROD04 **Priority:** Low

Key Feature: Yes

This feature allows users to review a product they have purchased

Stimulus/Response Sequences

1. The user clicks on the option to “Review Product”
2. The system displays product review page along with review form
3. The user completes the review information including rating, recommendation, and description
4. The user submits the response
5. The system displays the relevant success/error message
6. The system displays the posted review

Functional Requirements

REQ-1: *The system shall only allow users to review items they have purchased*
REQ-2: *The system shall only validate reviews with a rating*
REQ-3: *The system shall allow users the option to edit/delete their review*

3.2.5 View Product Reviews

Description and Priority

Use Case ID: PROD05 **Priority:** Low

Key Feature: Yes

This feature provides users with the option to view all the reviews of a selected product

Stimulus/Response Sequences

1. The user selects the option to “View Reviews” on product page
2. The system displays all relevant product reviews for the user

Functional Requirements

REQ-1: *The system shall ensure product reviews are stored in the database*
REQ-2: *The system shall allow users the option to view reviews by rating/date*
REQ-3: *The system shall allow users the option to edit/delete their review if their review is on page*

3.3 Shopping Cart and Order Checkout

3.3.1 Add Product To Cart

Description and Priority

Use Case ID: CART01 **Priority:** High

Key Feature: Yes

This feature allows users to add selected products to their shopping cart

Stimulus/Response Sequences

1. The user clicks on the "Add to Cart" button on product page
2. The system displays a notification that the product has been successfully added to the cart
3. The shopping cart icon shall increment with each added item

Functional Requirements

REQ-1: *The system shall create a new shopping cart item upon first "Add to Cart" selection*

REQ-2: *The system shall allow the user an option to select quantity of items to add to cart*

REQ-3: *The system shall allow the user the option to remove item from the cart*

3.3.2 View Cart

Description and Priority

Use Case ID: CART02 **Priority:** High

Key Feature: Yes

This feature provides the user the ability to view their current shopping cart details

Stimulus/Response Sequences

1. The user selects "View Cart"
2. The system displays an itemised list of products that have been added to the cart along with their quantity and price.
3. The system displays the total number of items to user, as well as total price and option to checkout, continue shopping, or save cart for later

Functional Requirements

REQ-1: *The system shall allow users the option to edit their cart by changing item quantities or deleting items*

REQ-2: *The system shall ensure a user is logged in upon "save" option*

3.3.3 Checkout

Description and Priority

Use Case ID: CART03 **Priority:** High

Key Feature: Yes

This feature allows users to check their order details, confirm their order, and pay for their order.

Stimulus/Response Sequences

1. The user selects “Proceed to Checkout”
2. The system displays the checkout page with itemised order details
3. The user selects option for delivery/pickup
4. The user provides payment information
5. The user confirms order
6. The system validates payment information
7. The system confirms the order is placed

Functional Requirements

REQ-1: *The system shall display delivery options if “delivery” is selected*
REQ-2: *The system shall allow the user to edit the cart contents on checkout page*
REQ-3: *The system shall ensure payment is valid prior to order confirmation*
REQ-4: *The system shall allow users the option to login or checkout as guest*
REQ-5: *The system shall send the user an email when order has been confirmed*

3.3.4 Order Delivery

Description and Priority

Use Case ID: CART04 **Priority:** High

Key Feature: Yes

This feature allows users to select order delivery information

Stimulus/Response Sequences

1. The user selects the “delivery” option on checkout page
2. The system asks the user to enter an address for delivery
3. The system validates the user address
4. The system displays available delivery windows for the user
5. The user selects their preferred delivery options
6. The system adds the delivery cost to order total
7. The delivery information is displayed upon order confirmation

Functional Requirements

REQ-1: *The system shall ensure the user has entered a valid address*

REQ-2: *The system shall provide only available delivery windows and options relevant to the user address*

REQ-3: *The system shall increment the order total as per the delivery cost*

REQ-4: *The system shall allow the user to add additional delivery information/instructions*

REQ-5: *The system shall include the delivery information with order confirmation email*

4. Use Cases

Use Case Name	Create New Account
Use Case ID	ACC01
Description	This system creates a new account for the User
Primary Actor	User
Preconditions	None
Main Flow	1. The use case starts when the user selects “Create New Account” 2. The system asks the user to enter their details comprising: email address, name, password, and password again for confirmation 3. The system validates the user details 4. The system creates a new account for the user
Postconditions	A new account has been created for the User
Alternative Flows	Invalid Email Invalid Password Cancel

Use Case Name	User Login
Use Case ID	ACC02
Description	The user logs in to SuperPrice application
Primary Actor	User
Preconditions	The User has created an account
Main Flow	1. The user selects option to login 2. The system displays fields for user to enter account email and password 3. The system validates account credentials 4. The system logs the user in 5. The system displays information to the user that the login was successful
Postconditions	The user is logged in
Alternative Flows	Invalid email Incorrect Password Cancel

Use Case Name	Update Account Information
Use Case ID	ACC03
Description	Users can edit their account information
Primary Actor	User
Preconditions	Users have created an account Users are logged in
Main Flow	1. The user selects the option to “Edit Account” 2. The system displays an edit option next to account information such as 2.1 Name 2.2 Email 2.3 Password 2.4 Location 2.5 Notification Preferences 3. The system validates user information 4. The system saves the new user information 5. The System confirms for the user that the changes have been made
Postconditions	User account changes
Alternative Flows	Invalid Email Cancel

Alternative Flow	Invalid Email Address
ID	ACC01.1, ACC02.1, ACC03.1
Description	This system informs the user that their entered email is invalid
Primary Actor	User
Preconditions	The user has entered an invalid email address
Alternative Flow	1. The alternative flow begins after **Step 3** of the main flow 2. This system informs the user that they have entered an invalid email address 3. The system displays the correct format of an email address to the user 4. The system asks the user to try again
Postconditions	None

Alternative Flow	Invalid Password
ID	ACC01.2, ACC02.2, ACC03.2
Description	This system informs the user they have entered an invalid password
Primary Actors	User
Preconditions	The user has entered an invalid password
Alternative Flow	1. The alternative flow begins after **Step 3** of the main flow 2. This system informs the user that they have entered an invalid password 3. The system displays if password confirmation is not matching to the user 4. The system displays the requirements of a password to the user 5. The system asks the user to try again
Postconditions	None

Use Case Name	View Order History
Use Case ID	ACC04
Description	Users can view their order history
Primary Actor	User
Preconditions	Users have created an account Users are logged in Users have made an order
Main Flow	1. The user has selected the option to “View Order History” under their account 2. The system displays the user’s order history 3. For every order the system will display 3.1 The order number 3.2 The date of the order 3.3 The quantity of items in order 3.4 The total price 3.5 An option to view the order details 3.6 An option for user to reorder
Postconditions	None
Alternative Flows	None

Use Case Name	Search Product
Use Case ID	PROD01
Description	This system finds products based on user search criteria and displays them to the user
Primary Actor	User
Preconditions	None
Main Flow	1. The use case starts when the user selects “search” or types in the search bar 2. The system searches for products that match the user criteria 3. If the system finds matching products then: 3.1 For each product found 3.1.1 The system displays the product name 3.1.2 The system displays a thumbnail of the product 3.1.3 The system displays the product price 4. Else 4.1 The system displays message that no matching products could be found 4.2 The system displays close product matches if any found
Postconditions	None
Alternative Flows	None

Use Case Name	Browse Products
Use Case ID	PROD02
Description	The system finds products based on user selected category criteria and displays them to the user
Primary Actor	User
Preconditions	None
Main Flow	1. The user selects the requested category 2. The system searched for products that match the user category 3. If the system finds matching products then: 4.1 For each product found 4.1.1 The system displays the product name 4.1.2 The system displays a thumbnail of the product 4.1.3 The system displays the product price 4. Else 5.1 The system displays message that no matching products could be found
Postconditions	None
Alternative Flows	None

Use Case Name	View Product
Use Case ID	PROD03
Description	The system displays product information to the user
Primary Actor	User
Preconditions	The user has selected a product to view
Main Flow	1. The use case starts when a user has selected a product to view 2. The user selects a particular product 3. For the product selected the system displays 3.1 An image of the product 3.2 The product name 3.3 The product description 3.4 The product's average rating 3.5 The product price 3.6 The store name of the best price 3.7 The product's category 3.8 An option to add item to cart 3.9 An option to view alternative store product prices
Postconditions	None
Alternative Flows	None

Use Case Name	Review Product
Use Case ID	PROD04
Description	Users are able to review products
Primary Actor	User
Preconditions	Users have selected a product to review
Main Flow	1. The use case starts when users have selected the option to "Review Product" 2. The user selects the option to leave a product review 3. The system displays the product information to user 4. The system provides the user a text box to type their review 5. The system asks users to enter a product rating out of 5 6. The system asks the user to select Yes or No if they would recommend the product 7. The system validates the product review information 8. The system displays the review once the user has confirmed their review
Postconditions	A review has been added to a product The average product rating is calculated
Alternative Flows	No rating selected Cancel

Alternative Flow	No Rating Selected
ID	PROD04.1
Description	The system informs the user that they have not entered a product rating
Primary Actors	User
Preconditions	The user has tried to post a review without entering product rating
Alternative Flow	1. The alternative flow beings after **Step 7** of the main flow 2. The system informs the customer that their review has not been posted as they have not selected a product rating 3. The system asks the customer to try again
Postconditions	None

Use Case Name	View Product Reviews
Use Case ID	PROD05
Description	Users are able to view product reviews
Primary Actor	User
Preconditions	Users have posted reviews for a product The user have selected a product for which they would like to see the reviews
Main Flow	1. Users select the option to “View Product Reviews” 2. The system displays the product details 3. The system displays the product reviews for the selected product 4. For each review posted for the product, the system displays 3.1 The rating 3.2 The review 3.3 If the product is recommended 3.4 Any replies to the review 5. The system provides the user with the option to filter reviews by rating 6. If the user has left a review, the system provides the options for the user to edit or delete their review
Postconditions	None
Alternative Flows	None

Use Case Name	Add Product To Cart
Use Case ID	CART01
Description	The system adds a product to the user's shopping cart
Primary Actor	User
Preconditions	A user finds a product they want to add to their shopping cart
Main Flow	1. The use case starts when the user selects "Add to Cart" 2. The user selects the option "Add to Cart" 3. The system asks the user to select the quantity of the item 4. The system adds the product to the user's shopping cart
Postconditions	The product is added to the user shopping cart
Alternative Flows	None

Use Case Name	View Shopping Cart
Use Case ID	CART02
Description	The system displays the details of the user's shopping cart
Primary Actor	User
Preconditions	The user selects the option to view their shopping cart
Main Flow	1. The use case starts when the user selects "View Cart" 2. The user selects the "View Cart" option 3. The system displays a list of products currently in the shopping cart 4. For each product in the product cart the system displays 4.1 The product name 4.2 A thumbnail of the product 4.3 The quantity of the items in the cart 4.4 The price of the item/s 5. The system displays the total number of items in the cart 6. The system displays the total price of the shopping cart 7. The system displays the options to: 7.1 Continue shopping 7.2 Save cart for later 7.3 Checkout
Postconditions	None
Alternative Flows	None

Use Case Name	Checkout
Use Case ID	CART03
Description	The user confirms order details and places order
Primary Actor	User
Preconditions	The user has added items to their shopping cart
Main Flow	1. The use case starts when a user selects “Checkout” 2. The System displays a summary of the shopping cart which includes 2.1 All product names, quantities, and prices 2.2 Total products 2.3 Total price 3. The system displays space for user to add a voucher/discount code 4. The system provides an option for users to select from delivery or collect from store 5. If the user is not logged in, the system asks the user if they would like to login or to checkout as guest 5.1 The system validates user account information 6. The system asks the user to select their preferred payment option 7. The system asks the user to enter their payment information 8. The system validates the user payment information 9. The system places the order for the user 10. The system confirms the order has been made
Postconditions	An order has been placed for the user
Alternative Flows	Invalid payment information Invalid login details Cancel

Alternative Flow	Invalid Payment Information
ID	CART03.1
Description	This system informs the user their payment information is invalid
Primary Actors	User
Preconditions	The user has entered incorrect payment information
Alternative Flow	1. The alternative flow begins after **Step 8** of the main flow 2. The system informs the user that they have entered invalid payment information 3. The system shows the user where the information is incorrect 4. The system asks the user to try again
Postconditions	None

Alternative Flow	Invalid Login Details
ID	CART03.2
Description	This system informs the user their login information is invalid
Primary Actors	User
Preconditions	The user has entered incorrect login information
Alternative Flow	1. The alternative flow beings after **Step 5** of the main flow 2. The system informs the user that they have entered invalid account information 3. The system displays the error to the user (invalid email/password) 4. The systems asks the user to try again
Postconditions	None

Use Case Name	Order Delivery
Use Case ID	CART04
Description	The user selects delivery options for their order
Primary Actor	User
Preconditions	The user has entered the checkout for their order
Main Flow	**1. Include (checkout)** 2. The user selects the option for “Delivery” in order checkout 3. The system asks the user to add/select their preferred delivery address 4. The system validates the delivery address 5. The system displays the available delivery windows to the user. For each available delivery window the system displays 2.1 The date of delivery 2.2 The estimated time of delivery 2.3 The cost of delivery 2.4 The name of the delivery service 6. The system allows users to enter additional delivery information 7. The system validates the delivery information 8. Upon the user confirming the order the system displays the delivery information for the user
Postconditions	Delivery information is saved for the order
Alternative Flows	Invalid address Delivery window no longer available Cancel

Alternative Flow	Invalid Address
ID	CART04.1
Description	This system informs the user they have entered an invalid address
Primary Actors	User
Preconditions	The user has entered an invalid address
Alternative Flow	5. The alternative flow beings after **Step 4** of the main flow 6. The system informs the user that they have entered an invalid address 7. The system displays the invalid address 8. The systems asks the user to try again
Postconditions	None

Alternative Flow	Delivery Window Unavailable
ID	CART04.2
Description	This system informs the user their selected delivery window is no longer available
Primary Actors	User
Preconditions	The user has selected a delivery window that is unavailable
Alternative Flow	9. The alternative flow beings after **Step 7** of the main flow 10. The system informs the user that the delivery window they have selected is no longer available 11. The system displays the unavailable delivery window 12. The system displays the available delivery window options to the user 13. The systems asks the user to select a new delivery window
Postconditions	None

5. Other Nonfunctional Requirements

5.1 Performance Requirements

5.1.1 Search Performance:
Search results for products should be returned within 2 seconds of the user initiating a search.

5.1.2 Price Comparison Load Time:
Price comparisons should load within 2 seconds after a user selects a specific product for comparison.

5.1.3 Application Load Time:
The application should fully load within 5 seconds upon initiation

5.1.4 Scalability:
The system should be able to handle growth of up to 50% in user base year-on-year without degradation of performance.

5.1.5 Response Time for Delivery Options:
Upon checkout, delivery options should be displayed within 3 seconds.

5.2 Security Requirements

5.2.1 Data Protection:
All user data, including personal details and purchase history, must be encrypted both in transit and at rest.

5.2.2 Authentication:
Users should be authenticated using strong authentication methods, such as two-factor authentication.

5.2.3 Authorization:
Different user roles and permissions should be defined to restrict access to certain parts of the application

5.3 Usability Requirements

5.3.1 Accessibility:
The application should be accessible following WCAG 2.1 AA guidelines to ensure a wide range of users, including those with disabilities, can access and use SuperPrice.

5.3.2 Mobile Responsiveness:
SuperPrice should be fully functional and visually appealing on a variety of devices, including smartphones, tablets, and desktops.

5.3.3 User Guide:
A simple and intuitive user guide or tooltips should be available for new users.

5.4 Reliability Requirements

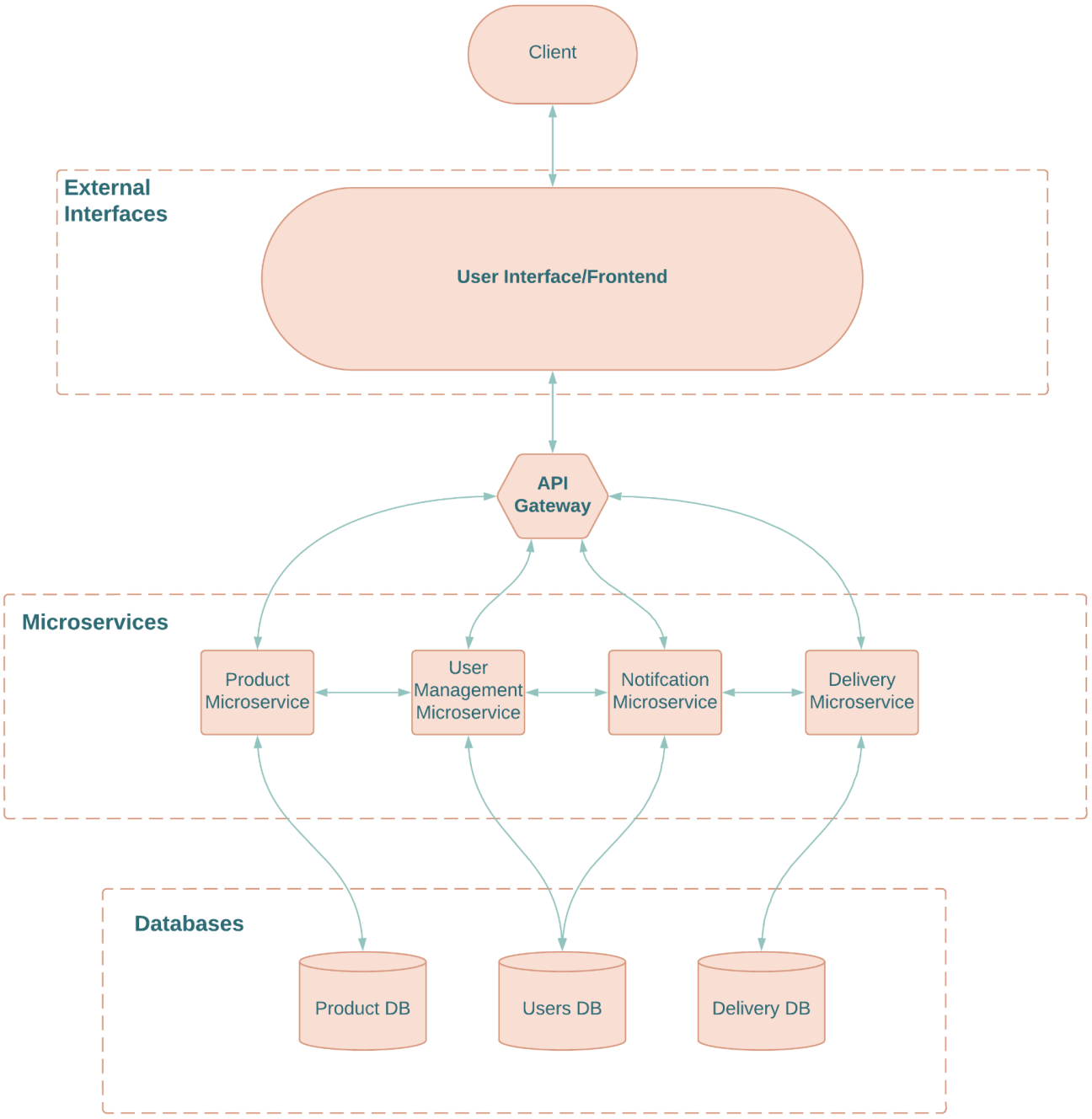
5.4.1 Uptime:
The SuperPrice application should aim for a 99.9% uptime, excluding scheduled maintenance.

5.4.2 Backup:
Regular backups of all user data and application configurations should be taken and stored in a geographically separate location.

5.4.3 Notification Reliability:
All price drop notifications and special offer alerts should reach the intended recipients with a reliability rate of at least 99%.

5.4.4 Error Handling:
The system should handle errors gracefully, providing users with clear messages and potential steps to resolve any issues they encounter.

6. System Architecture



System Architecture Diagram (Microservices)

6.1 Microservices Architecture:

Microservices Architecture is chosen for the **SuperPrice** web app for its advantages in Modularity, Flexibility, and Independent Deployment allowing for continuous deployment of the product to stakeholders. Modularity and flexibility in Microservices Architecture align with the **SuperPrice** application's dynamic and evolving nature. Breaking down the app into smaller specialised microservices enables agility in development and deployment. Microservices Architecture enhances maintainability deep into the product's lifecycle as modifications will not significantly affect surrounding modules. Independency in deployment ensures updates and changes can be rolled out to specific components without affecting the entire system, enhancing maintainability and reducing risk. Availability and reliability of the **SuperPrice** service will be ensured through the benefit of fault isolation: the failure of one microservice will not disrupt the overall operation. Given the foreseeable fluctuation in the number of users and varying workloads, individual services can be scaled based on their specific resource requirements, optimising resource usage and cost-effectiveness, streamlining scalability.

6.2 Spring Boot:

Spring Boot framework chosen for building microservices due to its ease of development, built-in features and support for ***RESTful APIs***. Individual services responsible for specific functionalities. Each microservice is designed to govern a specific domain or feature. Microservices communicate with each other through RESTful APIs. Spring Boot allows for rapid application setup, configuration and deployment with features such as embedded servers and starter packages.

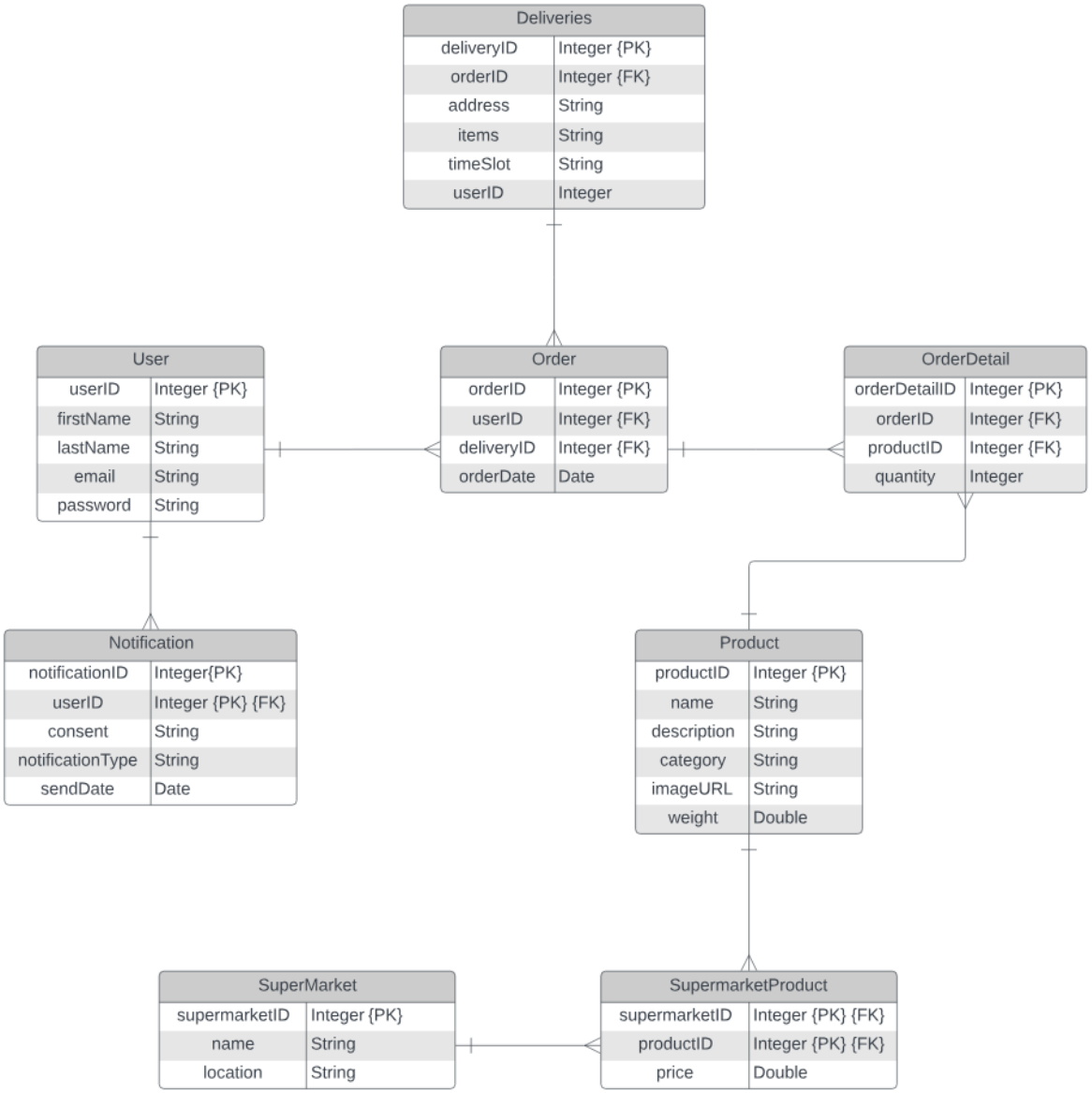
6.3 API Gateway:

Spring Cloud Gateway will be the technology for the API Gateway for its seamless integration with Spring Boot. API Gateway acts as a central entry point for all client requests. Spring Cloud Gateway will handle request routing to appropriate microservice(s). Simplify client access to multiple microservices providing a single point of contact.

6.4 Database:

Relational database chosen to align with the structured nature and well-defined relationships of product data, user reviews and other information in the **SuperPrice** application. Relational Database enables complex queries that require joining and aggregating data from multiple tables, allowing for features such as price comparison and product categorisation.

Data Model:



6.5 External Interface:

Users interact with the frontend with features including (but not limited to) search, price comparison, viewing and leaving reviews. Frontend interface communicates through RESTful APIs via the API Gateway to microservice(s), enabling seamless interactions and data exchange between different components of the **SuperPrice** app.

6.6 Front end component structure:

1. **App (Main Container Component)**
    a. State:
        i. currentUser: hold info about the logged-in user
        ii. products: array of all products - could hold this in context
        iii. notifications: array of user-specific notifications
    b. Children:
        i. Navbar
        ii. SearchBar
        iii. MainContent

2. **Navbar**
    a. Children:
        i. Logo
        ii. ProfileButton
        iii. NotificationIcon
        iv. ThemeSwitcher
        v. ShoppingCart

3. **SearchBar**
    a. State:
        i. searchQuery: current search text
        ii. searchResults: Results of the current search returned from BE?
    b. Children:
        i. InputField
        ii. SearchButton
        iii. SortingOptions
        iv. ClearInput

4. **MainContent**
    a. State:

i. openedContent (or store this in context?)

b. Children:

i. ProductListing (list or grid view of products)

ii. DeliveryOptions (shows available delivery options)

iii. NotificationList (List of notifications for the user)

5. **ProductListing**
   a. State:
      i. selectedCategory: the category currently being viewed
   b. Children:
      i. ProductItem (each individual product with price comparison)
      ii. CategoryFilter (allows filtering by product category - category selector)

6. **ProductItem**
   a. State:
      i. currentLowestPrice: the lowest price from all stores
   b. Children:
      i. ProductName
      ii. PriceComparison (shows prices from different stores)
      iii. AddToCart
      iv. ChangeQty

7. **PriceComparison**
   a. Children:
      i. StorePrice (price from an individual store - displays the store as well)

8. **DeliveryOptions**
   a. State:
      i. selectedTimeSlot: chosen delivery time slot.
   b. Children:
      i. TimeSlotPicker (user can choose a delivery time)

9. **NotificationList**
   a. Children:
      i. NotificationItem (individual notification about price drops or offers)

10. **UserContext: (holds user-related data globally)**
    a. Data:

i. userInfo: data about the user - may not be required
ii. Other required info if needed later

11. **ProductsContext: (to manage product-related data globally)**
    a. Data:
        i. allProducts: complete list of products - optional if not stored in app context
        ii. productUpdates: real-time updates for product prices and availability - maybe grab this from an open socket connection

12. **NotificationsContext: (to manage notifications globally)**
    a. Data:
        i. allNotifications: list of all user notifications

7. Testing Criteria

At the current stage only, Alpha testing conducted by the development team has been planned. Once we have assessed the product and addressed any bugs or inefficiencies we will conduct user-acceptance tests. This section may also be referred to when functional testing.

Delivery Organisation:

1. The most efficient delivery route is calculated based on the location of the customer(s) and selected stores.
2. Users are able to reserve a time slot for their delivery.
3. Users are notified when a delivery has been successfully scheduled and arrived.
4. Users are notified if their preferred delivery window is no longer available

***User-Friendly Interface*:**

1. A list of recently viewed items are saved and available from a menu link.
2. A list of suggested items is listed beneath the search bar based on search bar content.
3. The navigation menu is clear and menu items are appropriately labelled.
4. Users can clearly see if they are logged in by having their account name in top corner of all pages

***Reviews*:**

1. Users can read and submit reviews.
2. A sorting option is available by rating when search results are shown.
3. Users are able to edit and delete their reviews
4. Reviews shouldn't be posted without a rating

***Price Comparison*:**

1. Search results can be sorted by cost.

2. When viewing an item, similar items and their costs from all stores are displayed.

Product Search & Categorisation

1. The search functionality returns relevant results based on the provided query.
2. The user is able to browse the collection of items belonging to a given category.
3. The user is able to browse all products with special offers.

Notifications & Alerts

1. Users have the ability to set a level of notifications they are happy to receive and how.
2. Notifications are on time and provide relevant information based on user preferences.

Note: *See full Test & Acceptance criteria here.*

Appendix A: Glossary

Term	Definition
Database	Collection of all of the information monitored by a given system.
GitHub	A platform and cloud-based service for software development and version control using Git, allowing developers to store and manage their code.
Hashed Encryption	a one-way encryption process such that a hash value cannot be reverse-engineered to get to the original plain text.
JavaScript	The computer language commonly used to create interactive web browsers.
Spring Boot	An open-source framework for building stand-alone applications based on the Spring Framework.
API Gateway	A server or middleware architectural component acting as an entry point for client requests to access multiple APIs (Application Programming Interface).
Spring Cloud Gateway	An open-source framework that provides a way to build API Gateways in a microservice architecture. Part of the Spring Cloud Ecosystem.
RESTful API	Representational State Transfer API. An interface that two systems use to exchange information securely over a network.
Microservices Architecture	An architectural pattern that arranges an application as a collection of loosely coupled, fine-grained services, communicating through lightweight protocols.
Stakeholder	A person with an interest or concern in the product.

Appendix: Milestone 2

Summary of Project Changes from M1 to M2:

This Milestone we have implemented multiple elements on both the frontend and backend of our project, and are on track to complete implementation for the next sprint. The code in our github repo reflects all new functionality and elements that are new from Milestone 1. We have executed multiple tests, and collected results, some of which are reflected further down in this document.

Front end implementation of components include:

- Header
- Navbar
- Shop Page
- Products including both horizontal and vertical product carts
- Single product page
- Search bar
- Login
- Signup
- Theme switcher

Back end implementation of components include:

- Set up the microservices structure
- Implemented search function
- API endpoints
- Database schema for User, Products, Supermarket, Deliveries
- Price comparison logic
- Initialising structure for reviews
- Connected multiple components including products, login, and authorisation

SRS Changes:

- User Interfaces screenshots have been updated, added screen shots of the theme switcher.

- Data Model has been updated to correspond with implemented database schemas. Changes from original include:
    - Removed productCategory table
    - Changes to Deliveries table
    - Added foreign keys to all relevant tables

- Front end component structure has been updated to correspond with implemented functionalities and planned functionalities, more specifically:
    - Navbar
    - Search bar
    - Product items

- Added Milestone 2 Appendix

Contribution Statement:

Ronald Ho (s3436258)	Thomas Yao (s3832771)	Anthony Imani (s3950363)	Victoria Needs (s3811305)	Ledy (s3409664)	Dylan Khan (s3916841)
Set up microservices structure https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/13	Frontend - Authorisation: login and sign up UI https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/37	Databases for users and products https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/23	Commenced front end implementation of shopping cart https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/64	Frontend header implementation https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/41	Frontend - shop page https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/32
Implemented search function from backend	Frontend - setup FE repo and libraries https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/12	Microservices structure https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/13	Documentation - SRS updates	Frontend - Search bar https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/47	Frontend - product page https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/34
Multiple api endpoints https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/22	Filter component - https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/43	multiple api endpoints https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/22	contributed data to products db	Sort button component https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/70	completed search bar functionality https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/4
schema and data for product database https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/23	Homepage component - https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/31	user registration endpoint	Initialised order summary page https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/46	initialised search bar functionality https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/4	Vertical product cards https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/42

CI for backend	Login and sign up tests	User database schema	product cards test	unit tests	Product gallery
Price comparison logic	Documentation - minutes and sprint retro	login authentication and password encryption	broke the repo	Scrum master	connected products backend
searchbar test	documentation - frontend CI and tests descriptions	Deliveries Database initialisation https://github.com/cosc2299-sept-2023/team-project-group-p8-01/pull/74	contributed to testing and debugging	Sprint planning note	Horizontal product cards https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/45
Reviews structure initialised	connected login and authorisation https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/102			Created About Static Page https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/96	Front end unit tests
	CI workflow https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/56			Added dummy data for Meat & Seafood https://github.com/cosc2299-sept-2023/team-project-group-p8-01/issues/87	Fixed the whole repo from merge errors
16.67%	16.67%	16.67%	16.67%	16.67%	16.67%

Describing your CI with Multiple screenshots of failure and success:

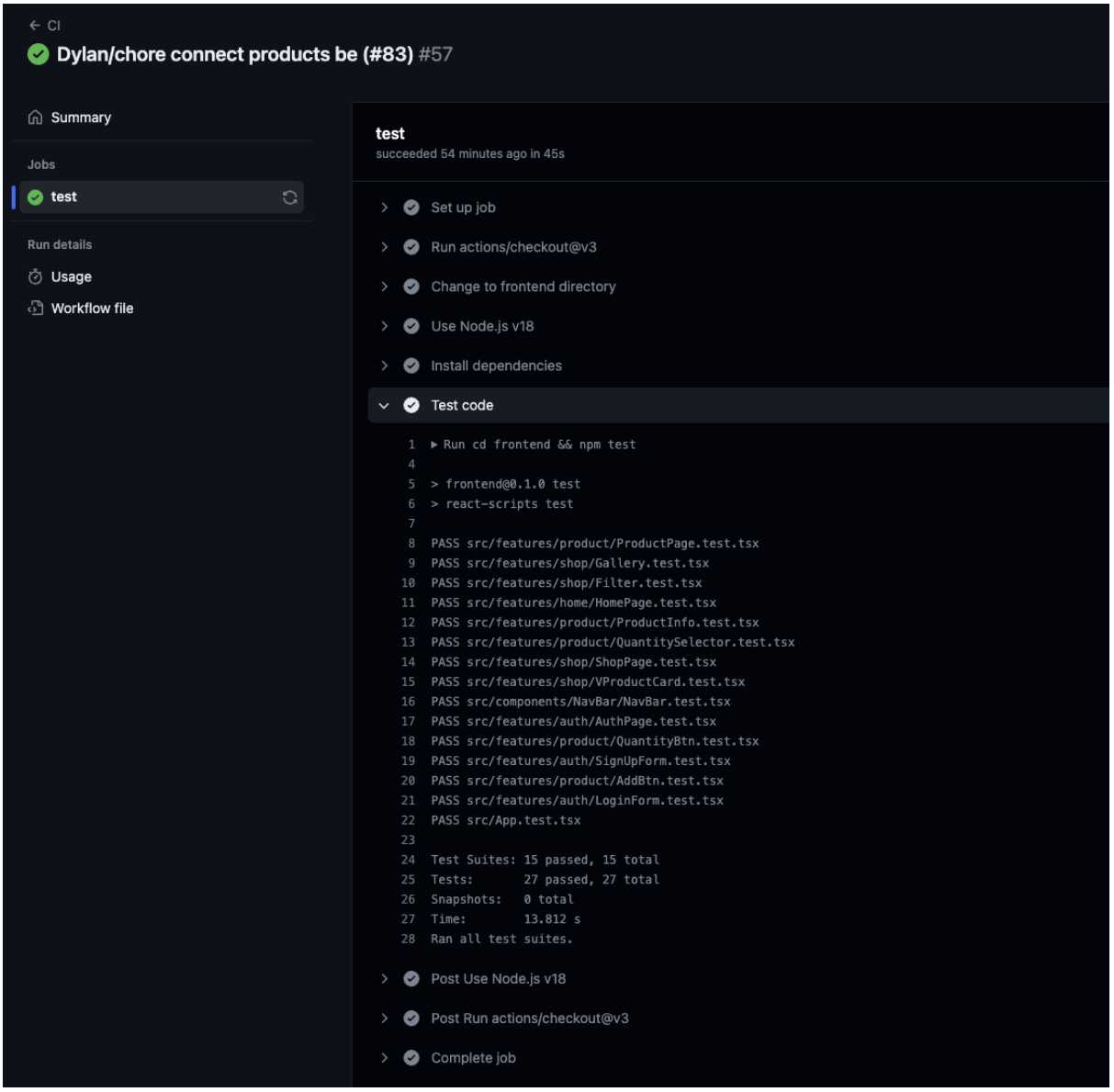
Continuous integration (CI) is an essential component of our development in sprint 1. After the core frontend architecture was set up, along with the CSS framework, a basic pipeline was created to make sure errors and bugs are detected early in the development process. This allowed us to confidently merge in features even if there were 6 of us developing concurrently.

For the frontend pipeline, we used github actions to trigger it whenever there was a push or PR created for the ‘main’ branch.

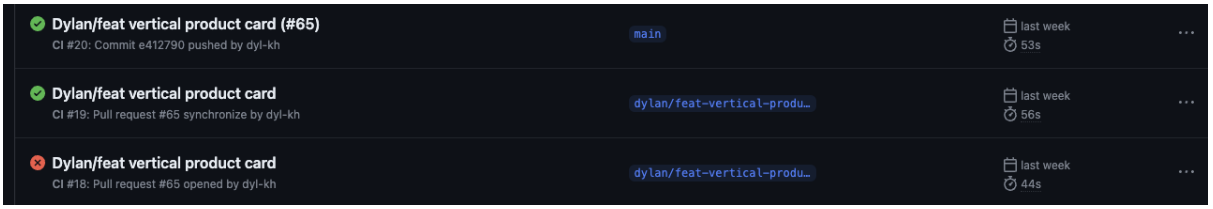
The pipeline sets up its environment on the latest version of Ubuntu OS, then checks out the latest code from the repository and ensures that the tests are running on the most recent version of the codebase.

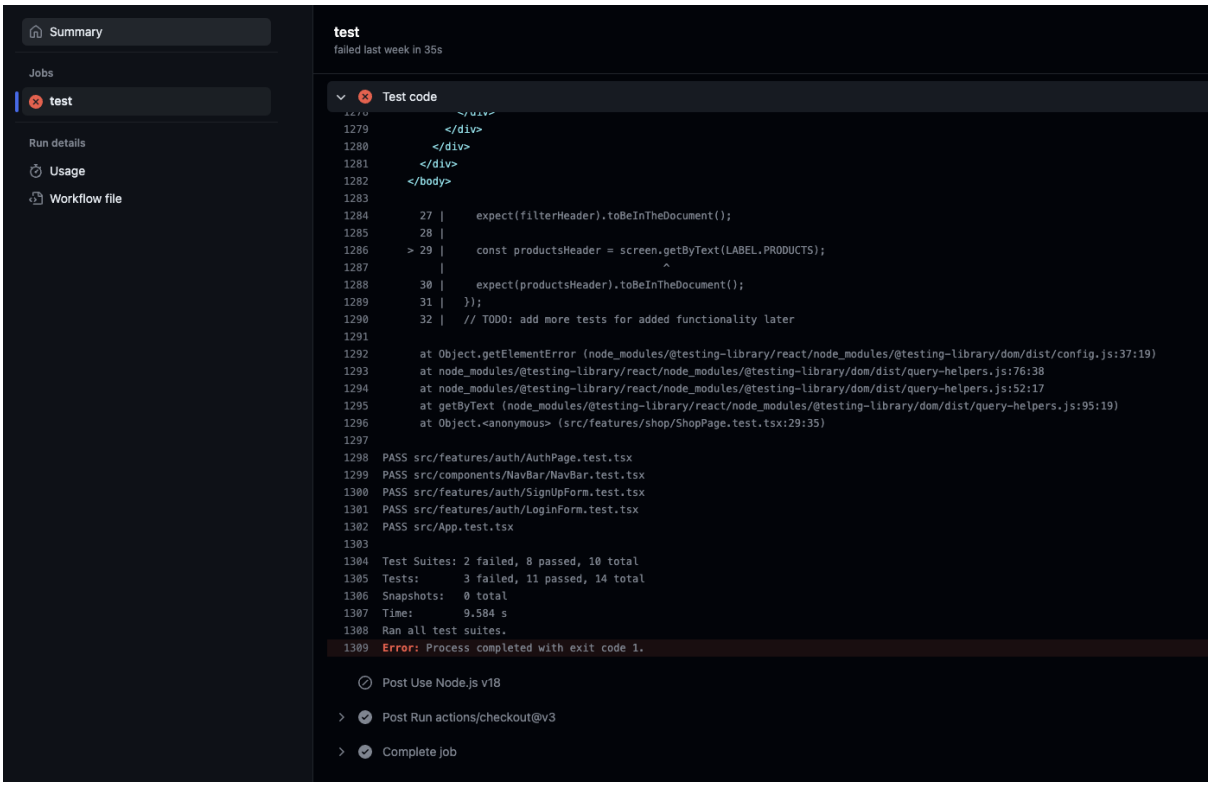
It then navigates to the 'frontend' directory, where it sets up the node environment, and installs the dependencies required for the frontend repo to run tests.

Lastly, the frontend pipeline runs the tests using `npm test`, arguably the most important step in the development process to protect the existing work of other developers. All of our team members respected this pipeline when merging changes into the main branch to make sure that we were not breaking any existing functionality.



- As evident in the second screenshot, one of the frontend unit tests failed which caused the entire pipeline to fail. This indicated to the developer that the branch was not ready to merge until it passed.





Documentation of test cases and evidence/documentation of test execution and results:

All of the frontend tests used the React Testing Library suite which allowed us to focus more on the user interaction instead of the underlying implementation because the suite promotes the idea of testing components in a way that users would actually use them. For example, instead of testing exactly what functions within components are returning, we mainly focused on testing the final components that were presented to the user.

This is evident in the ‘renders the home page correctly’ test, where final display elements are tested.

```
28 ▶  test( name: 'renders the home page correctly', fn: () : void  => {
29        renderWithRouter(<HomePage/>);
30
31        // check that all labels, buttons and images are rendered
32        const headlineElement : HTMLElement  = screen.getByText(LABEL.HEAD_LINE);
33        expect(headlineElement).toBeInTheDocument();
34        const shopAllButton : HTMLElement  = screen.getByRole('button', {name: LABEL.SHOP_ALL})
35        expect(shopAllButton).toBeInTheDocument();
36        const image : HTMLElement  = screen.getByRole('img');
37        expect(image).toBeInTheDocument();
38    });
39
```

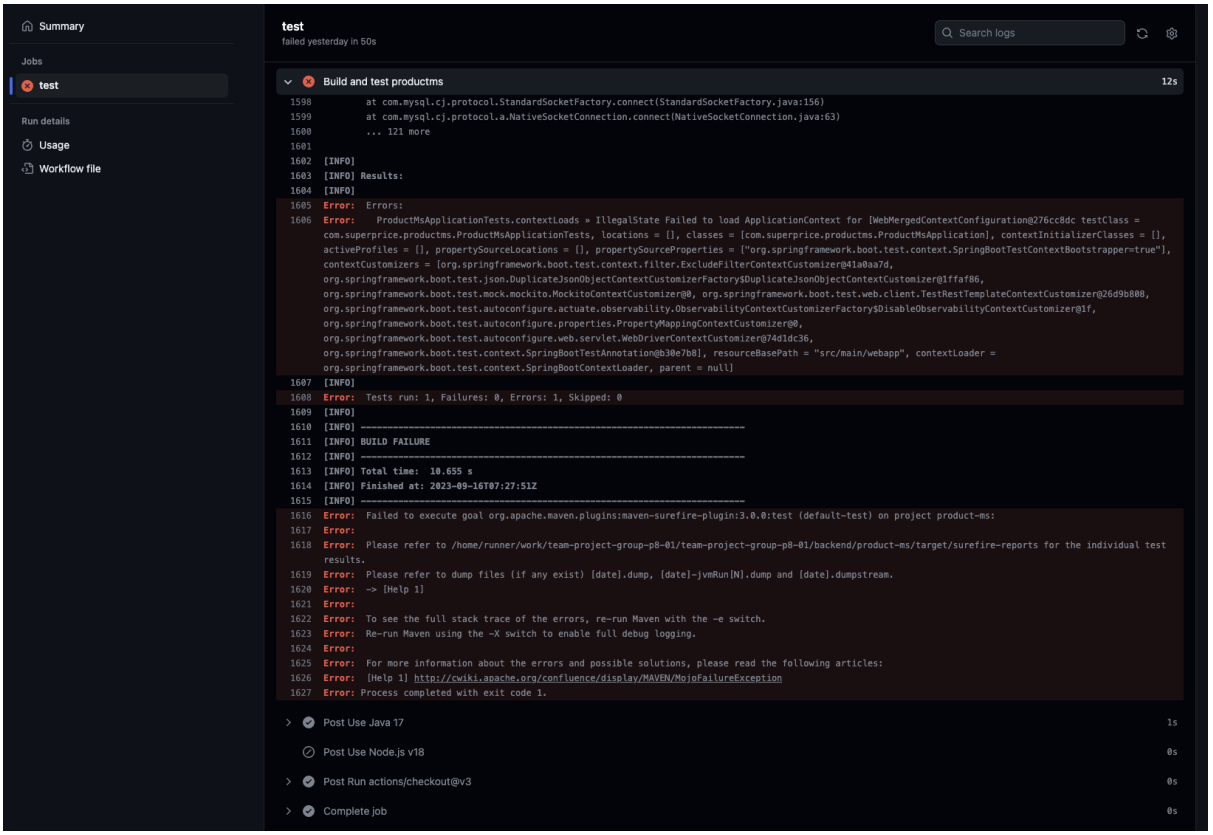
The tests are all documented with comments and in most cases, the test name sufficiently describes exactly what it is testing for.

```
PASS  src/features/product/ProductInfo.test.tsx
PASS  src/features/shop/Filter.test.tsx
PASS  src/features/shop/Gallery.test.tsx
PASS  src/features/product/ProductPage.test.tsx
PASS  src/features/home/HomePage.test.tsx
PASS  src/features/product/QuantitySelector.test.tsx
PASS  src/features/shop/VProductCard.test.tsx
PASS  src/components/NavBar/NavBar.test.tsx
PASS  src/features/shop/ShopPage.test.tsx
PASS  src/features/auth/AuthPage.test.tsx
PASS  src/features/product/QuantityBtn.test.tsx
PASS  src/features/auth/SignUpForm.test.tsx
PASS  src/features/product/AddBtn.test.tsx
PASS  src/features/auth/LoginForm.test.tsx
PASS  src/App.test.tsx

Test Suites: 15 passed, 15 total
Tests:       27 passed, 27 total
Snapshots:   0 total
Time:        2.831 s
Ran all test suites.

Watch Usage: Press w to show more.
```

Tests and CI pipeline for backend

As discussed in the sprint planning note:

- We were having issues at the very end of sprint 1 getting the backend CI to work on github actions.
- It was working fine locally, but backend tests were not passing on the GH actions machines.
- It was unanimously decided that we would move this ticket to sprint 2.
- The backend CI pipeline and tests are still included in an open PR, they are just not merged into main. https://github.com/cosc2299-sept-2023/team-project-group-p8-01/pull/79
- These will be documented in sprint 2.



Backend unit tests:

We faced a technical issue with the workflow for backend CI and investigations are being made to resolve this discrepancy. There are a total of 8 passing test cases for two microservices product and user. It should be noted that boundary tests are yet to be designed and implemented. The results of the user microservice tests:

```
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 4.072 s - in com.superprice.userms.UserMsApplicationTests
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  9.641 s
[INFO] Finished at: 2023-09-17T16:41:42+10:00
[INFO] ------------------------------------------------------------------------
```

These tests involve mocking the service and controller classes to test the registration and authentication of users. Our tests were created by first inverting the assertion so that the tests fail.

```
@Test
void testUserRegistrationSuccess() {
    UserDto userDto = new UserDto();
    userDto.setUserID(1);
    userDto.setFirstName("John");
    userDto.setLastName("Doe");
    userDto.setEmail("johndoe@example.com");
    userDto.setPassword("securepassword");

    when(service.userRegistration(userDto)).thenReturn(userDto);

    ResponseEntity<Object> response = controller.userRegistration(userDto);

    assertEquals(HttpStatus.BAD_REQUEST, response.getStatusCode());
}
```

```
UserControllerTest (com.superprice.userms.contro
    testUserRegistrationSuccess()

org.opentest4j.AssertionFailedError:
Expected :400 BAD_REQUEST
Actual   :201 CREATED
```

Document Links:

Sprint Retro:
https://github.com/cosc2299-sept-2023/team-project-group-p8-01/blob/main/docs/Milestone2/SEPT%20M2%20Retro.pdf

Sprint 2 Planning Notes:
https://github.com/cosc2299-sept-2023/team-project-group-p8-01/blob/main/docs/Milestone2/SuperPrice%20Sprint%202%20Planning%20Notes%20(Updated).pdf

Product and Sprint Backlogs:
https://github.com/cosc2299-sept-2023/team-project-group-p8-01/blob/main/docs/Milestone2/Updated%20P08-01%20-%20Product%20and%20Sprint%20Backlogs%20-%20Product%20Backlog.pdf

Minutes:
https://github.com/cosc2299-sept-2023/team-project-group-p8-01/blob/main/docs/Milestone2/SEPT%20-%20minutes%20m2.pdf