**COSC2299 Software Engineering: Processes & Tools**

Software Requirement Specifications

BrokieHub

**Group P3 Number 8**

**GitHub Repository:** [**https://github.com/cosc2299-sept-2023/team-project-group-p03-08**](https://github.com/cosc2299-sept-2023/team-project-group-p03-08)

|  |  |  |
| --- | --- | --- |
| **Name** | **Student ID** | **Contribution** |
| Kiran Kulkarni | s3943716 | 16.67% |
| Jamie Truong | s3947728 | 16.67% |
| Peter Fulton | s3896790 | 16.67% |
| Andy Chen | s3935474 | 16.67% |
| Benjamin Nippard | s3945124 | 16.67% |
| Tyler Humbert | s3947682 | 16.67% |

[Introduction 2](#_Toc154589424)

[Purpose 2](#_Toc2131485450)

[Stakeholders 2](#_Toc701279822)

[Functional Requirements 2](#_Toc1107134467)

[Non-Functional Requirements 3](#_Toc1430323715)

[Key Features 3](#_Toc1964533548)

[User Stories 3](#_Toc369218369)

[System Architecture 8](#_Toc1881364709)

[Structure Overview 9](#_Toc746007876)

[User Interface 9](#_Toc297026850)

[UI Design 9](#_Toc1883465235)

[Wireframes 10](#_Toc928611932)

[10](#_Toc1018805655)

[Design Choices | Sprint 1 12](#_Toc1444177108)

[Design Choices | Sprint 2 12](#_Toc2024203069)

[Testing – Testing Plan 13](#_Toc1360114005)

[Unit Testing 14](#_Toc1456807433)

[Sprint 1 – Tests 14](#_Toc582312955)

[Products 14](#_Toc1566669348)

[Deliveries 14](#_Toc1988355006)

[Notifications 14](#_Toc1670512856)

[Carts 14](#_Toc195088110)

[Carts 15](#_Toc259402617)

[Integration Testing 16](#_Toc73030208)

[Sprint 2 – Integration Testing 16](#_Toc496759643)

[System Testing - Acceptance Criteria 17](#_Toc66991911)

[Sprint 1 – Acceptance Testing 17](#_Toc572574028)

[Sprint 2 – Acceptance Testing 17](#_Toc2121278560)

[Product Backlog 19](#_Toc6286541)

[Sprint 1 and 2 20](#_Toc1775530105)

[Estimated Backlog – Sprint 1 20](#_Toc514458162)

[Estimated Backlog – Sprint 2 20](#_Toc211959385)

[Actual Backlog – Sprint 1 20](#_Toc1810508715)

[Actual Backlog – Sprint 2 21](#_Toc1756082352)

[Sprint 0 Retrospective 21](#_Toc1745799278)

[Sprint 1 Retrospective 22](#_Toc1535591792)

[Sprint 1 Burndown Chart 23](#_Toc1032101236)

[Sprint 2 Retrospective 23](#_Toc1550801223)

[Sprint 2 Burndown Chart 24](#_Toc801113212)

[SRS Appendix 24](#_Toc510273724)

[Delta 24](#_Toc240237344)

Introduction

## Purpose

SuperPrice is a web application for comparing grocery items between different stores. The website aims to streamline the shopping experience to make it more pleasant for shoppers. By removing the need for them to navigate several websites to search for the lowest price on the same product. The website will also be beneficial for storeowners (assuming they offer the lowest price) as it will bring customers to purchase their products.

It would achieve this by interpreting data from an assortment of databases and presenting the item the user desires at a given location for a price, along with any discounts or deals. The data received from the databases for any given item would be presented nicely in a web page. This web page would also include a suggested display for identical products with pricing from other store locations, along with a graph to display the historical pricing of a given item.

## Stakeholders

The client is a key stakeholder, being the person who commissioned us for this work and thus is the most invested in the success of the project. He will be the primary representative of the customer base we are developing the project for, but not necessarily the only customer we design for. As such, the client will be working closely alongside us and integrated with the development process as part of the Agile methodology.

Another stakeholder is the developers working on the product, who are directly responsible for the outcome of the project. The project will be developed under the Agile methodology and as such the team must keep up with the workload to see a successful project.

An external stakeholder that can affect the success of the project is the supermarkets from whom the developers will be sourcing the data from. The reliability and accessibility of any database APIs or other methods of sourcing data will directly influence the reliability of our project and as such, any major hinderances is liable to also cause issues in the project’s development cycle.

# Functional Requirements

* **Compare Different Prices**: Enable users to simultaneously view and contrast the prices of multiple products.
* **Order a Delivery**: Allow users to schedule the delivery of all items in their cart seamlessly.
* **Notifications**: Implement a system that sends various notifications to users as needed.
* **Product Search**: Facilitate the search for specific items through an efficient and effective search function.
* **Categorizing Products**: Provide the ability for users to filter items by category, enhancing discoverability.
* **Login System**: Implement a secure user account creation process for purchasing items.
* **Search Filters**: Include search filters to aid users in narrowing down product selections.
* **Pricing History**: Enable users to view the historical pricing data of products.

# Non-Functional Requirements

* **Legal (Security)**: Ensure compliance with relevant legal regulations to maintain security and privacy.
* **Alcohol Age Limit (Security)**: Implement age verification for purchasing age-restricted products.
* **Font Size (Usability)**: Allow users to adjust the font size on the webpage for readability.
* **Website Navigation (Usability)**: Design the website to enable easy navigation and price comparison.
* **24/7 Availability (Reliability)**: The website should be accessible for use 24 hours a day.
* **Themes (Usability)**: Incorporate customizable themes to enhance the user experience.

Key Features

o Product Search and Categorization: The application will provide a comprehensive search feature, allowing users to find specific products or browse through various categories. This functionality will ensure a smooth and hassle-free shopping experience.

o Price Comparison: The SuperPrice application will enable users to search for specific products and instantly compare prices across different supermarkets in the area. Users can easily identify the store offering the lowest price for their desired items.

o Delivery Organization: SuperPrice will facilitate the organization of deliveries for users who wish to have their groceries brought to their doorstep. The application will offer multiple delivery options, including time slots to ensure flexibility and convenience.

o Notifications and Alerts: The application will provide timely notifications and alerts to keep users informed about price drops or exclusive offers. This feature will ensure that users never miss great deals.

o User-Friendly Interface: SuperPrice will boast a user-friendly and intuitive interface, making it accessible to a wide range of users. The application will prioritize simplicity and ease of use, ensuring a seamless experience for both novice and experienced shoppers.

# User Stories

|  |  |  |  |
| --- | --- | --- | --- |
| **Story #1:** | **Price Comparison** | **Priority** | High |
| **Effort** | 10/10 |
| As a | Customer | | |
| I want | To compare prices between multiple stores | | |
| So that | I can choose the lowest price in a store near me | | |
| Acceptance criteria | Scenario: customer comparing prices for a product with comparisons   * Given I am a customer * When I click to view price comparisons of a product * Then the system will return a results page of stores with similar or lower prices   Scenario: customer comparing prices for a product without comparisons   * Given I am a customer * When I click to view price comparisons of a product * Then the system will return a page without price comparisons for this product | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Story #2:** | **Delivery** | **Priority** | High |
| **Effort** | 9/10 |
| As a | Full-time worker or student | | |
| I want | To be able to order my groceries online for delivery | | |
| So that I can | Save the time and effort from having to physically go to the store. | | |
| Acceptance criteria | Scenario: full-time worker or student ordering groceries for delivery   * Given I am a full-time worker or student, * When I click on the ‘Delivery’ option after adding items to my cart, * Then the system will send a notification confirming delivery   Scenario: Customer orders groceries for delivery and enters invalid address   * Given I am a full-time worker or student, * When I click on the ‘Delivery’ option after adding items to my cart, * Then the system will prompt me for a correct delivery address | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Story #3:** | **Notifications** | **Priority** | High |
| **Effort** | 9/10 |
| As a | Mother | | |
| I want | To be notified of the best deals and price changes | | |
| So that I can | Be informed of making the best decisions that cut costs for my household | | |
| Acceptance Criteria | Scenario: Mother wants to be notified of the best deals   * Given I have chosen for deal notifications * When businesses add new deals to their websites * Then the system will send a notification alerting of the new deals available.   Scenario: Mother wants to be notified of the price of a specific product   * Given I have clicked the notifications button on a product page * When the price changes for this product * Then the system will send a notification alerting of the price change | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Story #4:** | **Simple Search Bar** | **Priority** | High |
| **Effort** | 8/10 |
| As a | New user | | |
| I want | To be able to search for products in a quick and easy manner | | |
| So that I can | Have a seamless shopping experience despite not having used the system before. | | |
| Acceptance Criteria | Scenario: User searches for a product by key terms   * Given I have entered the key terms for my search * When I click the search option * Then all items matching key terms will appear   Scenario: User searchers for all products   * Given I have entered no terms in the search bar * When I click the search option * Then all items will appear   Scenario: User search returns no products   * Given I have entered the key terms for my search * When I click the search option * and no products match those terms * Then the website displays a message about no matching items | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Story #5:** | **Navigation** | **Priority** | High |
| **Effort** | 8/10 |
| As a | New user | | |
| I want | To be able to navigate through the website easily | | |
| So that I can | Have a seamless shopping experience despite not having used the system before. | | |
| Acceptance Criteria | Scenario: A new user is navigating our website for the first time   * Given I am a new user on the website * When I look around the website * Then I want to able to figure out what each button/link will do | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Story #6:** | **Categorisation** | **Priority** | High |
| **Effort** | 8/10 |
| As a | Browsing shopper | | |
| I want | I want products to exist under a category | | |
| So that I can | Find the exact products I am looking for by category if I am unsure of what the product is | | |
| Acceptance Criteria | Scenario: Customer who is unsure of what exactly they want to search for products   * Given I have selected/searched for the category of products I want * When I hit the search button * Then I want to find products under that category * And find the product I am looking for * And find products like what I may be looking for | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Story #7:** | **Search Filter** | **Priority** | High |
| **Effort** | 7/10 |
| As a | Customer | | |
| I want | To filter for certain stores in my search | | |
| So that | I can only see the prices relevant to me | | |
| Acceptance criteria | Scenario: Customer wants to filter out groceries too far away   * Given the customer is on the map feature to search for grocers offering prices * When the customer filters for only grocers within 5km of location * Then the map should only display grocers within a 5km radius   Scenario: Customer wants to filter groceries by a specific feature (e.g. Vegetarian or Gluten Free)   * Given the customer has selected their desired feature * When the customer selects the filter button * Then all items are filtered to match their filters | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Story #8:** | **Historical Pricing** | **Priority** | Medium |
| **Effort** | 6/10 |
| As a | Customer | | |
| I want | To preview historical pricing | | |
| So that | I can purchase items when they are at a lower price | | |
| Acceptance criteria | Scenario: Customer opens item with historical pricing   * Given I am a customer * When I click onto an item * Then I see the current price * and I see a list of previous prices and specials   Scenario: Customer opens item without historical pricing   * Given I am a customer * When I click onto an item * Then I see the current price * And a message that historical pricing is unavailable | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Story #9:** | **Login** | **Priority** | Low |
| **Effort** | 2/10 |
| As a | Returning user | | |
| I want | Be able to login to my account | | |
| So that | I can track access my account | | |
| Acceptance criteria | Scenario: customer logging on correctly   * Given I have an account * When I go to login * And enter my correct details * Then I can sign into my account   Scenario: customer logging on incorrectly   * Given I have an account * When I go to login * And enter incorrect details * Then I am prompted that entry was incorrect and offered to reset password   Scenario: customer logging on without account   * Given I do not have an account * When I go to login * Then I am presented with the option to create an account | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Story #10:** | **Different lighting themes** | **Priority** | Medium |
| **Effort** | 5/10 |
| As a | Customer | | |
| I want | Be able to change the lighting of the website | | |
| So that | I do not strain my eyesight whilst using the website | | |
| Acceptance criteria | Scenario: customer changing light themes   * Given I am logged in to the website * When I go to the settings * Then I can change the lighting theme of the webpage | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Story #11:** | **Track Deliveries** | **Priority** | Low |
| **Effort** | 4/10 |
| As a | Logged In User | | |
| I want | To be able to track my current deliveries | | |
| So that | I can view when my purchases will arrive. | | |
| Acceptance criteria | Scenario: Customer track delivers   * Given I have logged into my account * When I click on track delivers * Then I can see my current deliveries * And I can see how long they take till delivery | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Story #12:** | **Create account** | **Priority** | Low |
| **Effort** | 3/10 |
| As a | new user | | |
| I want | To be able to create an account | | |
| So that | I can track and save my shopping history | | |
| Acceptance criteria | Scenario: customer opens create account page   * Given I am a new user * When I press the create account button * Then I am given a page to input my details   Scenario: customer inputs details and creates an account   * Given I am on the create account page * and I have input my details * When I press the create account button * Then an account is generated for me | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Story #13:** | **Recover password** | **Priority** | Low |
| **Effort** | 3/10 |
| As a | Account holder | | |
| I want | To be able to recover my password | | |
| So that | I can access my account if I forget my password | | |
| Acceptance criteria | Scenario: A user has forgotten their password   * Given I have an account and I have forgotten the password * When I press the forgot password button * Then I am sent a recovery email   Scenario: User opens their recovery email   * Given I have clicked on the link in the recovery email * When I enter my new password * Then my password is updated | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Story #14:** | **Add item to Shopping cart** | **Priority** | Medium |
| **Effort** | 6/10 |
| As a | Account holder | | |
| I want | To be able to store items I want to purchase in a cart | | |
| So that | I can order multiple items simultaneously | | |
| Acceptance criteria | Scenario: User has found a product and wishes to add it to their cart   * Given I have chosen an item * When I add an item to my cart * Then the item I added will appear within it * and cart price updates   Scenario: User has found a product and wishes to add multiple to their cart   * Given I have chosen an item * Given I have selected my quantity * When I add the item to my cart * Then selected quantity will be added to my cart * and cart price updates | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Story #15:** | **Remove item from Shopping cart** | **Priority** | Medium |
| **Effort** | 6/10 |
| As a | Account holder | | |
| I want | To be able to remove items I have in my shopping cart | | |
| So that | I can remove accidental additions to it | | |
| Acceptance criteria | Scenario: A user has an item they wish to remove from their cart   * Given I have an item in my shopping cart * When I select to remove an item from my shopping cart * Then the item I selected will be removed * and the shopping cart price is updated | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Story #16:** | **Browsing** | **Priority** | Medium |
| **Effort** | 4/10 |
| As a | user | | |
| I want | To be able to view item prices | | |
| So that | I can decide whether I want to use this website | | |
| Acceptance criteria | Scenario: A user with an account wishes to view an item's price   * Given I am logged into an account * When I navigate the site * and find an item I like * Click on said item * Then I can see the price of said item   Scenario: A user without an account wishes to view an item's price   * Given I am not logged into an account * When I navigate the site * and find an item I like * Click on said item * Then I can see the price of said item | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Story #17:** | **View Item** | **Priority** | High |
| **Effort** | 6/10 |
| As a | User | | |
| I want | To view the details of an item | | |
| So that | I can decide if I want to purchase it | | |
| Acceptance criteria | Scenario: A user has found an item they what to learn about   * Given I have found an item * When I click on an item * Then I see the details of the item * And have the option to add to cart | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Story #18:** | **View Cart** | **Priority** | High |
| **Effort** | 8/10 |
| As a | User who is shopping | | |
| I want | View items in my cart | | |
| So that | Know what I am planning to buy | | |
| Acceptance criteria | Scenario: User opens their shopping cart with items   * Given I have items in the shopping cart * When I click on the shopping cart * Then I can see the items in the cart   Scenario: User opens their shopping cart without items   * Given I have no items in the shopping cart * When I click on the shopping cart * Then I am prompted with a message that shopping cart is empty | | |

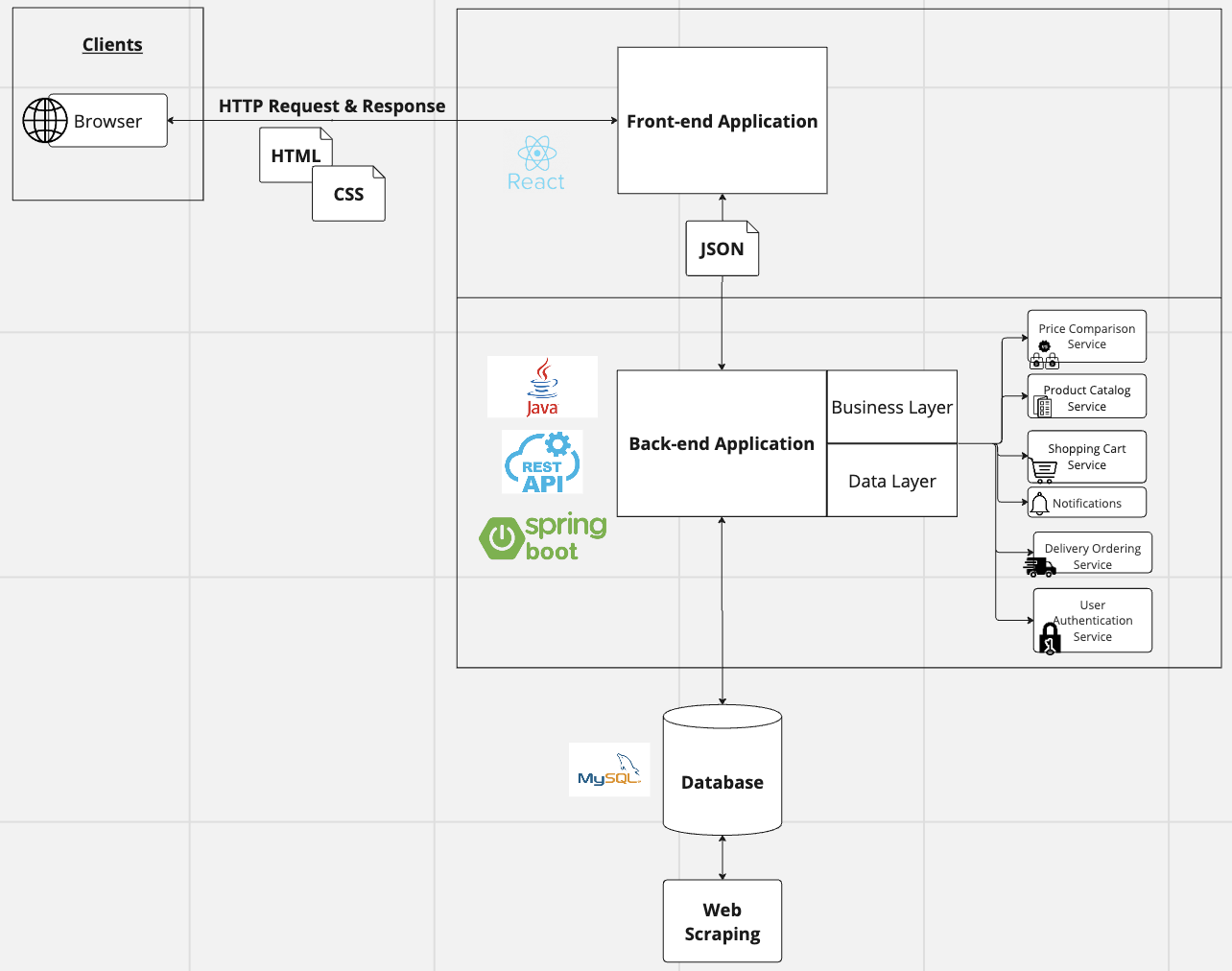
|  |  |  |  |
| --- | --- | --- | --- |
| **Story #19:** | **Change Cart Quantity** | **Priority** | Med |
| **Effort** | 6/10 |
| As a | customer | | |
| I want | To adjust the quantity of items in my cart | | |
| So that | I can order the correct amount of items | | |
| Acceptance criteria | Scenario: User has items in their cart and wishes to reduce the quantity   * Given I have an item in my cart * When I reduce the quantity * and I hit save * Then the quantity is decreased * And the cart price is updated   Scenario: User has items in their cart and wishes to increase the quantity   * Given I have an item in my cart * When I increase the quantity * and I hit save * Then the quantity is increased * And the cart price is updated | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Story #21:** | **Checkout Cart** | **Priority** | High |
| **Effort** | x/10 |
| As a | Customer | | |
| I want | To checkout my cart | | |
| So that | I can purchase items I have selected | | |
| Acceptance criteria | Scenario: Customer has a cart with items   * Given I have opened my cart * When I hit the checkout button * Then I am taken to the checkout screen   Scenario: Customer has a cart without items   * Given I have opened my empty cart * When I hit the checkout button * Then I am notified my cart is empty | | |

# System Architecture

## Structure Overview

The system architecture that we have adopted for our project is the modern web application architecture that consists of a front-end application with a presentation layer and a back-end application consisting of the business and data layer.



The front-end application consists of the presentation layer that enables users to interact with the back-end application through a browser that sends HTTP requests where our front-end application will return a response.

The back-end application consists of the business and data layer. The business layer consists of our models, controllers and services. The modelscontain the business concepts and manages the data of our application. The controllershandle HTTP requests and queries based on the business logic and models. The controller receives user inputs from the presentation layer, validates it and then passes that information to the models. The services contain the overarching business logic of our project. The other layer for our back-end application is the data layer consisting of our repositories which contains the persisting data of the database for our project. We have also included a connection from the database to web scraping in order to fill the product database with products from the web.

The technologies employed for the components of our software architecture include using React for the front-end framework, Spring Boot for the back-end framework and MySQL for data storage and querying.

# User Interface

## UI Design

The overall design of the web application follows the philosophy of unobtrusiveness and to display the data with as much data and transparency as possible. As such, the layout of the application will simply consist of a header, and content to display depending on the webpage. This is to ensure that there can be little confusion as to where to navigate, as all our features will be primarily displayed in the header. To follow along with this simplicity, our UI will not involve any animation or fancy UI components as they could cause issues with screen readers and reduce the accessibility of the application.

In terms of style guides, our primary colours will be cornflower blue and white as they are colours that have a calming effect and fits with the goal of the application, which is to make a wise decision on purchasing products. Additionally, to be as transparent about the display of data, items will be displayed in a grid format with as text to indicate prices as it is easier to navigate different stores and items.

Lastly, we have primarily designed the UI for a desktop application, as we are restricted by the development skills and time constraints to develop the project. This means that the UI design assumes users are utilising a mouse and keyboard to navigate the website and thus relies heavily on button navigation.

## Wireframes

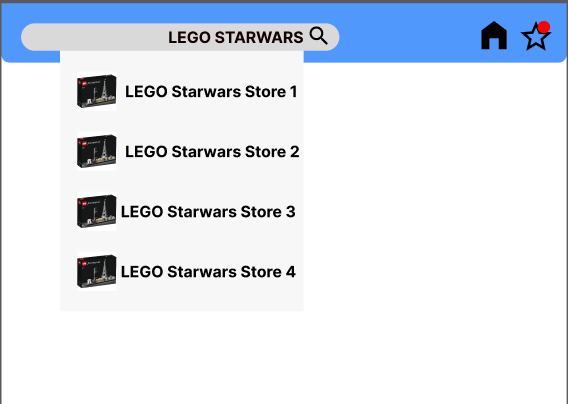
Homepage designs:

Two wireframes were created to prototype the landing page of the website. On the left is a mock-up of the landing page for a user who is not signed in. It will thus display the most relevant deals daily, so those who aren’t looking for any items can thus simply look at what is the determined best deals of the day.

## 

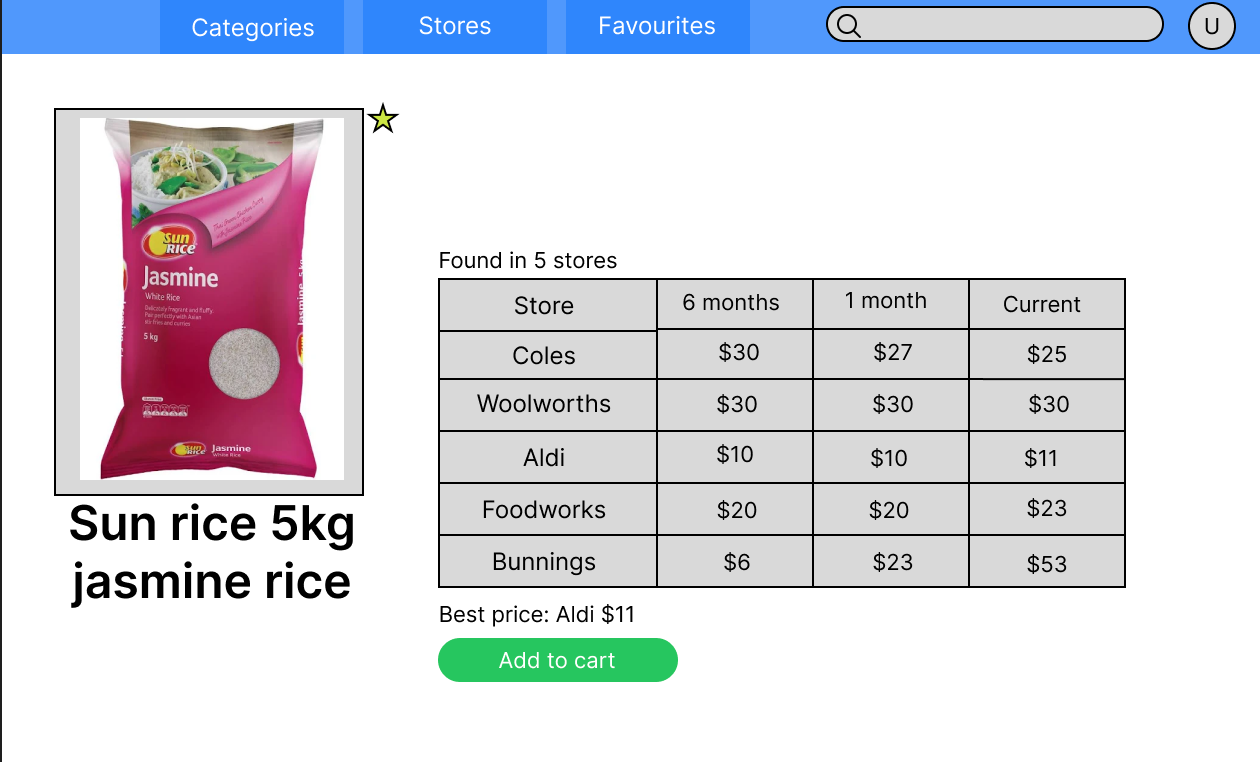
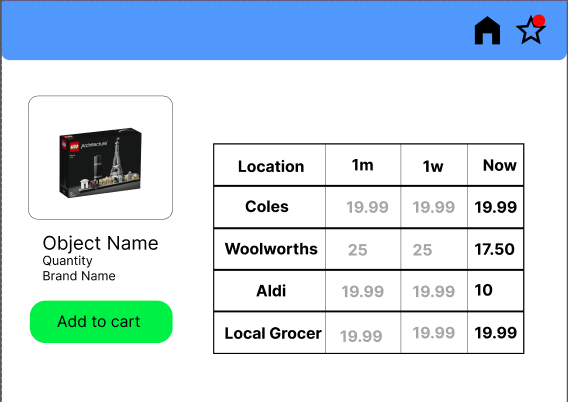
Menu Designs:

The two wireframes represent two extreme possibilities for how the menu could be designed. The left wireframe represents the simplest representation of the menu, with only a search bar, home button and favourites button that would lead to profile. The second wireframe represents a highly detailed menu version that lists several places to look as well as more information from the search.



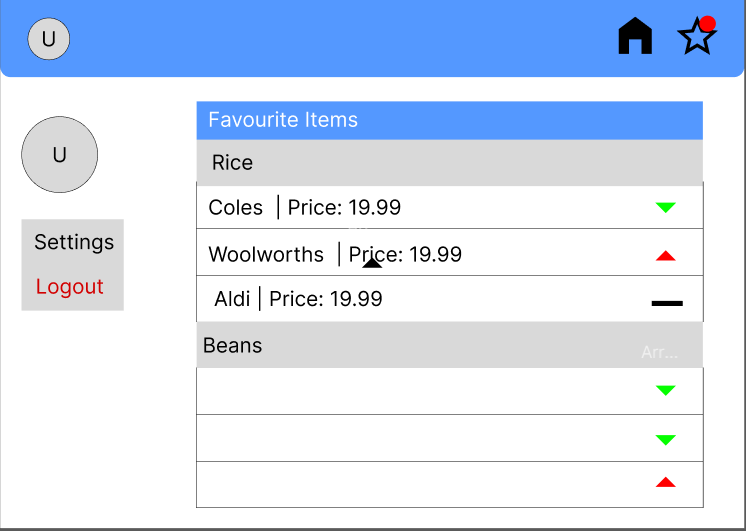
Product page designs

These two wireframes represent what the look of the product page will be displayed and is designed to show as much information as possible in a clear manner. The different prices are displayed in a grid and is made the emphasis of the page relative to the rest of the page content to emphasise the data when viewing the page.



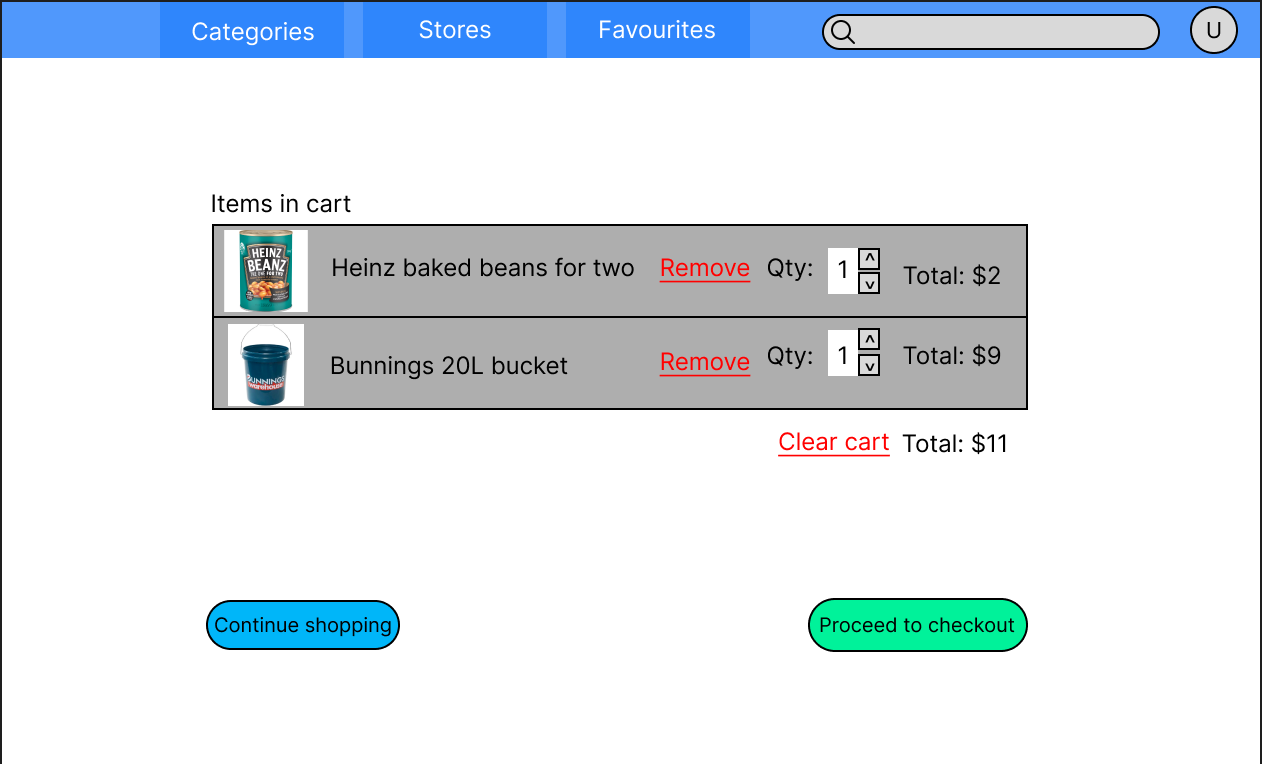
User profile design:

The profile page is intended to be used as the quick access to all the items the user has been favourited with a quick indicator of whether the price is up or down from their previously measured price. Additionally, this is where they can access settings and the logout ability.



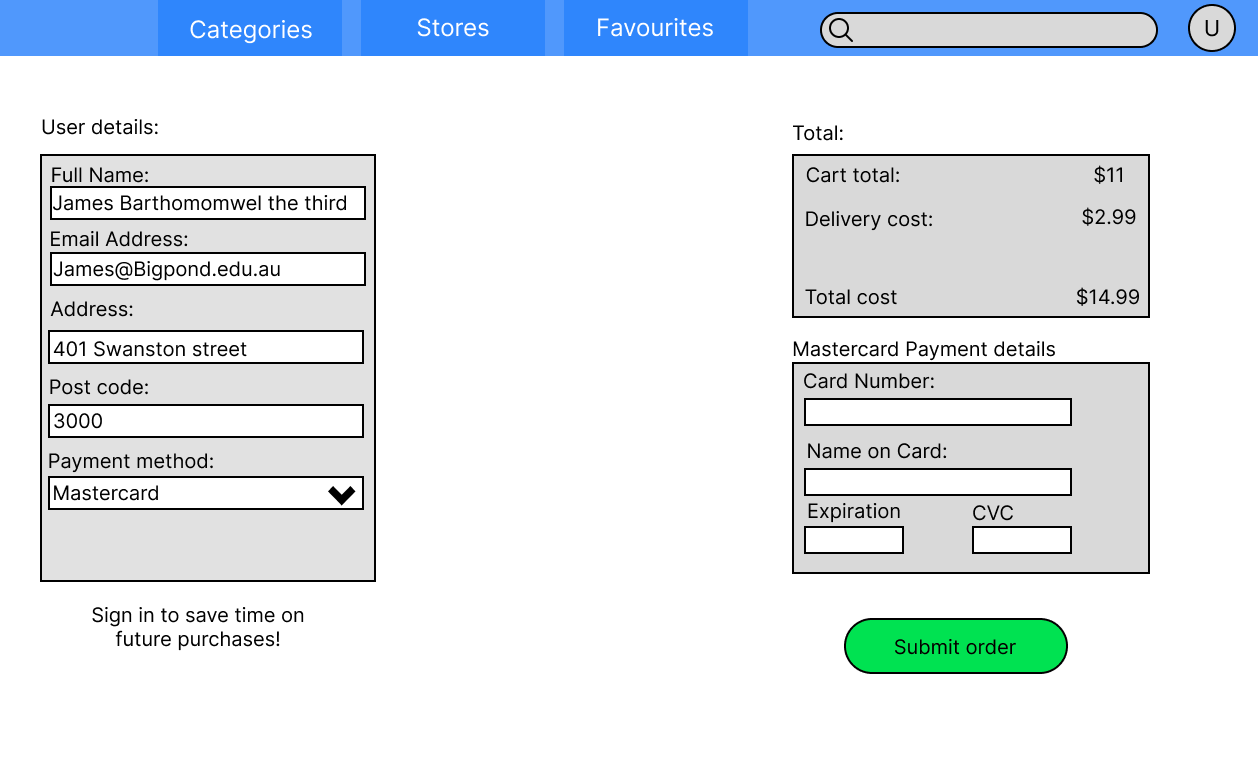
Cart menu design:

The wireframe of the cart menu design allows users to adjust the quantity of the products they’ve put in the cart. The removal components that would remove products or clear the entire cart has been highlighted in bright red to ensure that users don’t accidentally click on those without intending to.



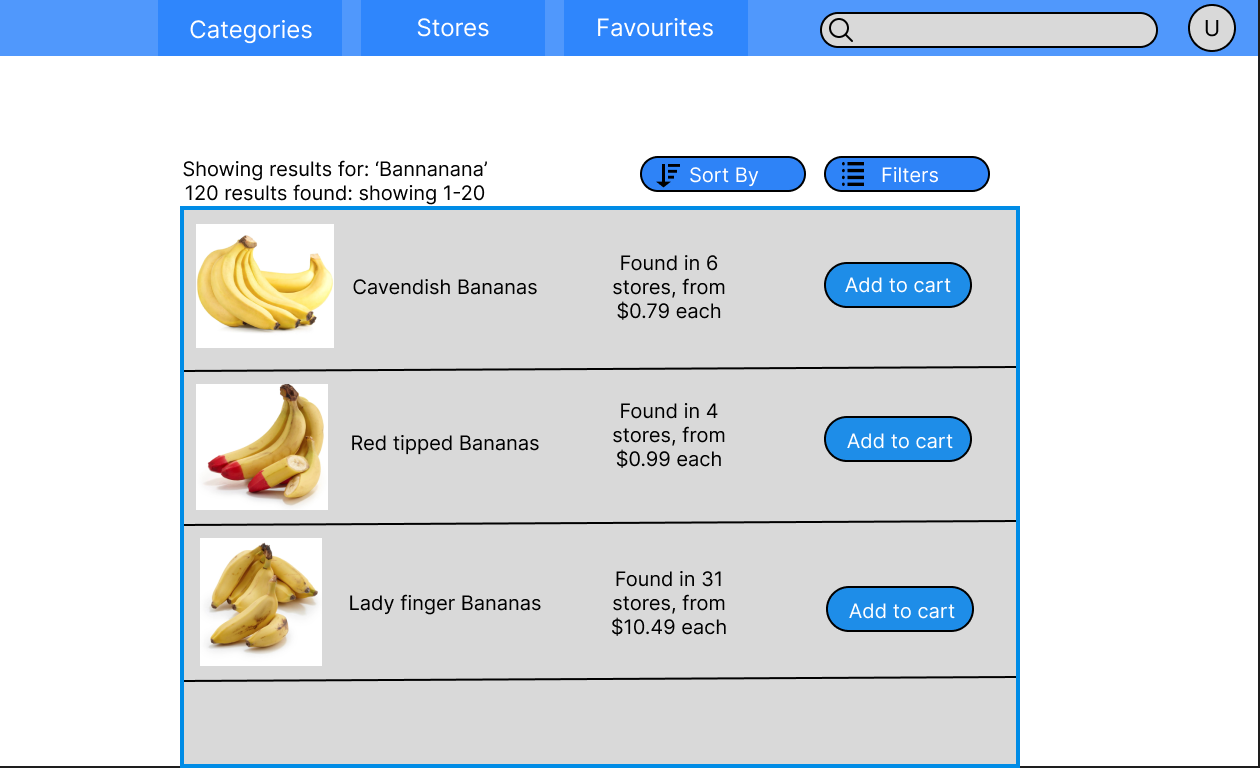
Checkout Page Design

The checkout page is orientated in a way that when reading from left to right, top to bottom, the user has a chance to double check the pricing before submitting their payment details and ordering.



Search Results Page Design:

This wireframe clearly displays all products from the query to the customer. It also allows the user to sort and filter the results, enhancing their searching experience.



## Design Choices | Sprint 1

Regarding development of our frontend, the decision to use the front-end framework React for development was made based on the experience levels of the development team with the front-end framework, as well as the inherent benefits of React. In comparison to other front-end frameworks, React’s usage of a virtual DOM and optimisation ensures that the loading of the website is faster.

Additionally, as the project has a deadline restriction for its development, the development team cannot afford to waste any time to learn any new frameworks and so picking a framework that is familiar helps speed up the development of the project.

One major UI design choice was abandoning the icon-based navigation in favour of words in the navigation bar as any accessibility tools such as screen-readers may struggle to read what the icon is and that words makes it easier to discern what the toolbar is over an image which can be interpreted differently based on the user.



Figure 1 Original Icon Chosen

Figure 2 The Icon Chosen

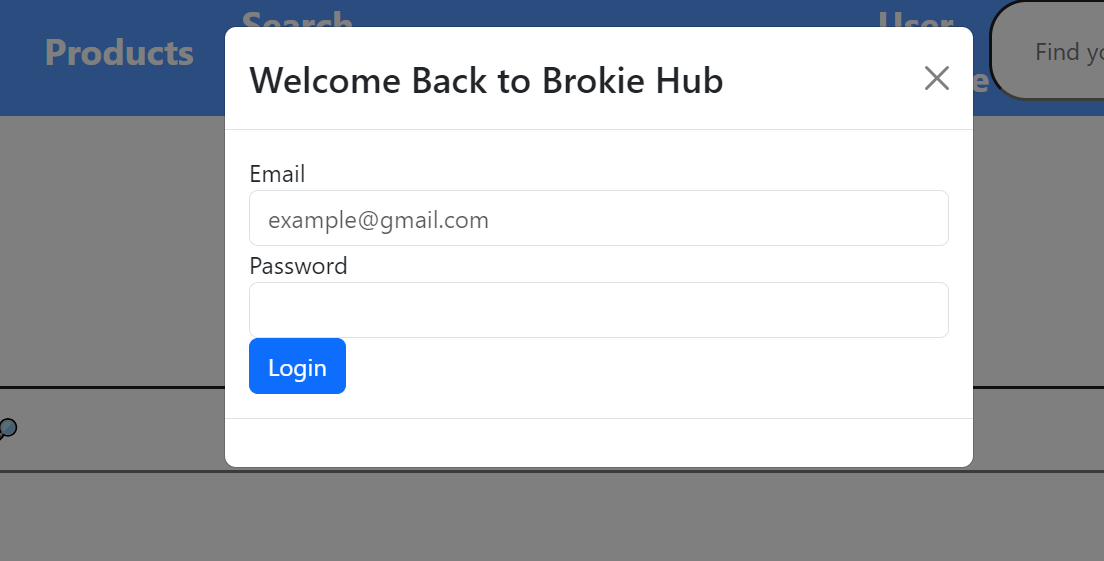
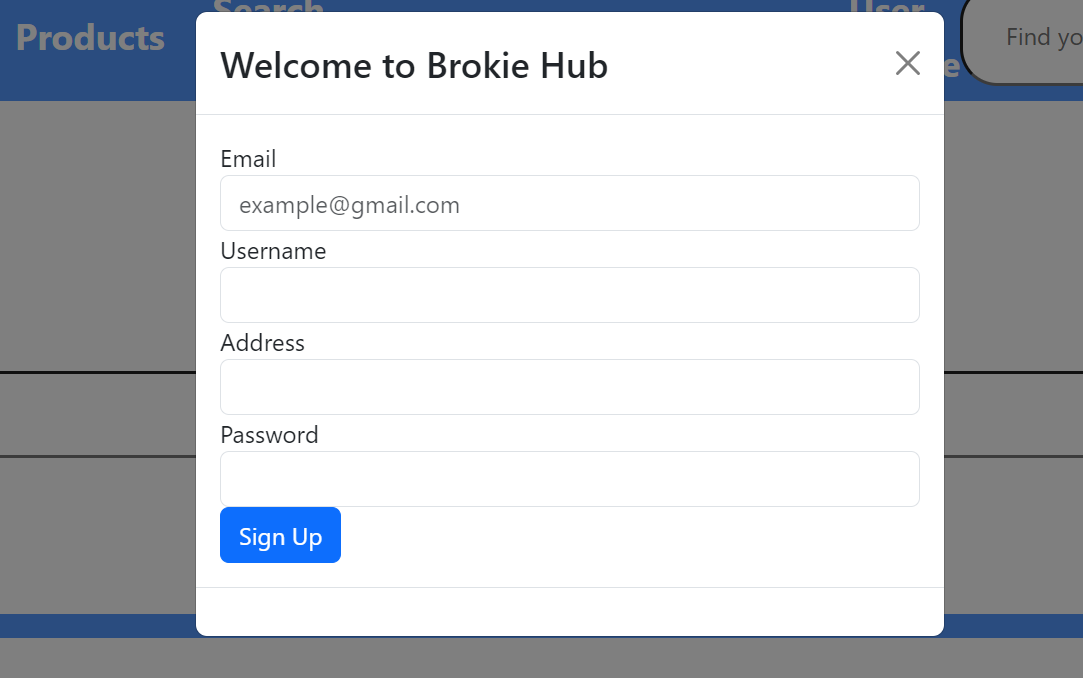
Another major UI design choice that was absent from the wireframes is the business logo and website title that is prominently displayed on the landing page. The logo was sourced from flaticon.com and chosen for its simplicity in design and contrasting with the shade of blue we chose to use as our main colour. Initially we intended to go with an alternative neon colour version of the icon, but it proved very difficult to display along with the shade of blue as well as not fitting with the impression we intended to give with the website. The title itself was inspired by websites such as YouTube that used contrasting font colours and background text to catch the users’ eye and remain memorable with their associated colour.

Most of the UI design was based on the wireframe diagrams, constructed during sprint 0. Minor changes were made to button styling and icons, but functionality and position remained the same. A greater change that was made, was splitting the table into more columns to reduce the functionality of each column to one function.

## Design Choices | Sprint 2

The front-end design in this sprint was primarily focused around implementing the front-end logic to connect the front-end to the back-end servers and databases. One of the more significant features to be implemented, the login system, was designed to utilise the useContext hook from React to hold the login user as a state. UseContext was specifically chosen due to the ability to declare a variable that can then be accessed by any component declared inside its provider, thus allowing for a global state that can be accessed and modified without needing to pass it down as a prop to child components. For example, if a component that dealt with shipping needed to access the user’s address, instead of needing to have that data passed as a prop through 20 layers of components to reach the shipping component, the shipping component can thus access it using the hook and bypassing all those layers. This comes with the benefit of keeping the code much more maintainable in the future, allowing for future components to be written without interfering with pre-established code.

When it came to designing the login and sign-up pages, I elected to choose a modal design to represent them as it would help focus the user’s attention to those areas compared to an entirely separate page for logging in or signing up. Since all other elements such as the navigation bar or search bar are unnecessary for the functionality of the sign up, I elected to choose a modal instead to hide away unnecessary elements and focus user’s attention on the form itself. Additionally, the only validation that was implemented in the front-end was using required html tags to prevent empty strings and using regex for email validation. As there was no explicit requirement for further validation, the database queries and subsequent HTTP responses were relied on for error handling and validation instead. However, this section can easily change in future sprints if the product owner requires further validation.



A smaller design choice was made for the search results pages to no longer display how many searched results came up and how many page there were. Another slight change was to cart page, where it now tells the user to login, if they want to use the cart.

# Testing – Testing Plan

## Unit Testing

Regarding unit testing, we will take a testing first approach by developing unit tests prior to development of front and backend functionality. Our testing will primarily focus on functional testing since for our first 2 sprints these will be our main priority and non-functional testing will primarily be left until a later sprint.

### Sprint 1 – Tests

#### Products

|  |  |
| --- | --- |
| Repository | |
| Test Name | Testing For |
| ReturnsWhenAvailable | Tests Database returns the correct amount and values of test data |
| insertProduct | Tests database inserts product correctly and checks number of records increases, and the return matches the expected |
| deleteProduct | Tests database decreases upon delete and correct Boolean is returned |
| failDeleteProduct | Tests database doesn’t change upon failed deleted and correct Boolean is returned |
| updateProduct | Tests update product correctly returns updated product and adds correctly to datasource |
| priceUpdate | Checks price update works correctly returning the expected product that matches the database |
| search\_name | Checks searching works for name |
| search\_category | Checks searching works for category |
| search\_supermarket | Checks searching works for supermarket |
| search\_priceUpper | Checks searching works for price upper |
| search\_priceLower | Checks searching works for price lower |
| search\_priceLower\_priceUpper | Checks searching works for both price upper and lower range |
| search\_all\_params | Checks searching works for all parameters at once |
| Controller | |
| Test Name | Testing For |
| should\_returnEmpty\_When\_noRecords | An empty return if no records |
| should\_returnEmpty\_When\_Records | An return when there is records |
| return\_on\_insert | A response entity with created status returned on function call |
| return\_true\_delete | An accepted status return if delete works |
| return\_false\_delete | A bad\_request status return if delete fails |
| return\_on\_edit | An accepted status return if edit works |
| return\_price\_update | An accepted status return if update works |
| search\_correct\_params | Tests correct search parameters are passed to the service |

#### Deliveries

|  |  |
| --- | --- |
| Repository | |
| Test Name | Testing For |
| ReturnsWhenAvailable | Tests return all returns the deliveries |
| insertDelivery | Tests insert deliveries returns the correct delivery and updates size of db |
| deleteDelivery | Tests if delete delivery is found that db size decreases and True is returned |
| deleteDeliveryNotFound | Tests if delete delivery is not found that db size doesn’t decreases and False is returned |
| updateDeliveryCorrectId | Tests if a correct ID is given and then that the delivery is updated and database changes |
| updateDeliveryIncorrectId | Tests that if an incorrect ID is given that null is returned and delivery doesn’t update |
| returnsContents | Tests that the correct delivery contents is returned |
| Controller | |
| Test Name | Testing For |
| should\_returnEmpty\_When\_noRecords | Tests returns no records if empty |
| should\_returnDeliveries\_When\_availableInService | Tests it Returns deliveries in services |
| return\_on\_insert | Tests returns on insert |
| return\_on\_edit | Tests return on edit |
| delete\_called\_once | Tests that delete is called once |
| orderContents | Tests order contents returns correctly |

#### Notifications

|  |  |
| --- | --- |
| Repository | |
| Test Name | Testing For |
| ReturnsWhenAvailable | Returning correct amount and correct notifications |
| insertNotification | Testing insert notification updates db and returns correct record |
| deleteNotification | Testing delete notification deletes the correct notification and returns True |
| failDeleteNotification | Testing Delete notification doesn’t delete and returns false |
| updateNotification | Testing update notification changes the database and returns correct notification |
| Controller | |
| Test Name | Testing For |
| should\_returnEmpty\_When\_noRecords | Returning empty if no records in db |
| should\_returnNotifications\_When\_availableInService | Testing return if records in Db |
| return\_on\_edit | Testing return notificaiton and Accepted status on edit |
| return\_true\_delete | Testing that true delete will return an ACCEPTED status |
| return\_false\_delete | Testing that a false delete will return a BAD\_REQUEST Status |
| return\_on\_insert | Testing that an insert will return a notification and will return a CREATED status |

#### Carts

|  |  |
| --- | --- |
| Repository | |
| Test Name | Testing For |
| ReturnsWhenAvailable | Checks that cart returns >0 results when returning all available carts |
| DeletesWhenAvailable | Checks remove item from cart returns true |
| AddsWhenAvailable | Checks add item to cart returns true |
| SetQuantityWhenAvailable | Checks set quantity for cart returns true |
| Controller | |
| Test Name | Testing For |
| returnEmptyWhenNoResults | Checks that cart controller returns empty for no results |
| returnResultsWhenAvailable | Checks that cart controller returns results |
| returnSpecificCartWhenSearched | Checks that controller can return a specific cart |
| addProductTest | Checks that controller can add a product to cart |
| removeProductTest | Checks that controller can remove a product from a cart |
| incrementUpTest | Checks the controller can increment an items quantity in a cart |
| incrementDownTest | Checks the controller can decrement an items quantity in a cart |
| SetQuantityTest | Checks the controller can set an items quantity in a cart |
| Service |  |
| Test Name | Testing For |
| returnNothingWhenEmpty | Service returns empty list if cart db is empty |
| returnResultsWhenAvailable | Service returns list if cart is not empty |
| returnCartWhenSearched | Service correctly searches all carts to find and return a matching cart id |
| returnNullWhenNotFound | Service will correctly search all carts to not find and return null |
| validAddItemTest | Service will correctly add the item to the correct cart and return True |
| failWhenItemExistsInCart | Service is unable add the item to the correct cart and return false |
| validRemoveItemTest | Service will correctly remove the item from the correct cart and return True |
| failWhenItemThatDoesntExist | Service is unable remove the item from the correct cart and return False |
| validIncrementUpTest | Tests increment logic correctly increments |
| validIncrementDownTest | Tests decrement logic correctly decrements |
| validSetQuantityTest | Tests set quantity logic correctly sets quantity |
| callRemoveItemWhenQuantityIsDecrementedToZero | Tests that remove item is called when the quantity is decremented down to 0 |
| failWhenIncrementingAMissingProduct | Tests you cannot increment a missing object |
| failIncrementWhenCartDoesntExist | Tests you cannot increment in a cart that doesn’t exist |

Sprint 2 – Tests

#### Carts

|  |  |
| --- | --- |
| Controller | |
| CheckoutTest | Tests that the checkout function returns an int deliveryID |
| Service | |
| validCheckout | Tests the checkout returns a deliveryID when valid |
| invalidCheckoutEmptyCart | Assert that the service returns 0 when the cart is empty |
| invalidCheckoutNoCart | Assert that when checkout is called for a non existant cart 0 is returned |
| Repository | |
| CheckoutWhenAvailable | Assert that when the database is available, execute the relevant queries and return the newly generated deliveryID |

Users

|  |  |
| --- | --- |
| Controller | |
| returnEmptyWhenNoResult | Controller should return an empty arraylist with 0 as its size |
| returnResultsWhenAvailable | Given a user is available, return a populated arraylist |
| returnSpecificUserWhenSearched | Given a user with ID 1, controller should return the user with the ID of 1 |
| deleteUserWhenAvailable | Controller should return a HTTP response 200 when deleting an existing user |
| InvalidDeleteUser | Controller should return a HTTP response 400 when attempting to delete a nonexistent user |
| createUser | Controller should return a HTTP response 201 when creating a new user |
| InvalidcreateUser | Controller should return a HTTP response 400 when attempting to create a new user with invalid details |
| attemptLogin | Controller should return a HTTP response 200 when validating a login attempt |
| invalidattemptLogin | Controller should return a HTTP response 400 when detecting an invalid login attempt |
| Service | |
| returnNothingWhenEmpty | Service should return an empty arraylist with 0 as its size |
| returnResultsWhenAvailable | Given a user exists in database, service should return a populated array list |
| returnCartWhenSearched | Given a user with ID 1, service should return the user corresponding with ID 1 |
| returnNullWhenNotFound | Service should return null given a user that doesn’t exist |
| ValidAddUser | Service should return true when a new user is created |
| InvalidAddUser | Service should return false when a new user cannot be created |
| RemoveUserWhenAvailable | Service should return true when deleting an exisitng user |
| ValidAttemptLogin | Service should return false when attempting to delete a nonexistent user |
| InvalidLoginWrongEmail | Service should return true when validating a valid login attempt |
| InvalidLoginWrongPassword | Service should return false when it detects an invalid login attempt |
| Repository | |

Front-end Unit Tests

|  |  |
| --- | --- |
| Test Name | Testing For |
| Login Modal | |
| Renders Login Form Properly | Checks if the HTML document contains the actual element that’s being rendered |
| Detects Empty Response in Email Input via Required Tag | Checks for if the Email Input element will consider an empty string a valid response or not |
| Detects Empty Response in Password Input via Required Tag | Checks for the Password Input element to see if an empty string will register as invalid or not |
| Sign Up Modal | |
| Renders Sign Up Form Properly | Checks if the HTML document contains the actual element that’s being rendered |
| Detects Empty Response in Email Input via Required Tag | Checks for if the Email Input element will consider an empty string a valid response or not |
| Detects Empty Response in Name Input via Required Tag | Checks for the Name Input element to see if an empty string will register as invalid or not |
| Detects Empty Response in Address Input via Required Tag | Checks for the Address Input element to see if an empty string will register as invalid or not |
| Detects Empty Response in Password Input via Required Tag | Checks for the Password Input element to see if an empty string will register as invalid or not |

## Integration Testing

For integration testing since we are using a model view controller infrastructure, we will use an incremental method of testing as it is well suited for the integration testing. This will require us to run unit test on each of our components before gradually implementing our components one by one. This method works best with the process of unit testing

### Sprint 2 – Integration Testing

*Products*

|  |  |
| --- | --- |
| allProducts | Ensures that products present in the database are returned when allProducts is called |
| findProductById | Ensures that a specific product is returned when searching for it with its id |
| searchProduct | Ensures that a list of products that meet the search criteria are returned when called |
| insertProduct | Ensures that products are able to be added to database and returns correct Https status when created |
| deleteProduct | Ensure products can be deleted and returns the correct https status when removed |
| updatePrice | Ensures products can be updated, and the correct https status when updated |
| priceUpdate | Ensure the price of products can be updated and the correct https status when updated |
| priceInsert | Ensure new prices can be added to the database, and the correct https status when added |

*Carts*

|  |  |
| --- | --- |
| allCarts | Verifies that all carts are returned when called |
| getCartByID | Ensures that a specific cart is returned when using the cart search function |
| getCartByCustomerID | Same as above but searches by customer id instead of cart id |
| removeProduct | Verifies that products can be removed from cart, and verifies the http response code |
| addProduct | Verifies that products can be added to cart, and verifies the http response code |
| IncrementProduct | Checks that products in cart can be incremented 1 quantity up, and verifies the correct http response code is returned |
| decrementProduct | Checks that products in cart can be decremented 1 quantity down, and verifies the correct http response code is returned |
| setQuantity | Checks that the products in cart can have their quantity set to a specific number, and the correct response code is returned |
| Checkout | Verifies that the checkout functionality works, clearing the cart and turning the products into an order, verifies the correct orderID is returned |

User

|  |  |
| --- | --- |
| allUsers | Checks to ensure users are returned when this function is called |
| getUserByID | Ensures a specific user is returned when called using their id |
| getUserByEmail | Same as above but with email |
| deleteUser | Verify users can be deleted, and the method returns the correct http response when called |
| createUser | Verify users can be created, and the method returns the correct http response when called |
| loginUser | Verify that when given the correct credentials the correct http response is returned |

Delivery

|  |  |
| --- | --- |
| allDeliveries | Verifies that deliveries are returned when called |
| getContentsById | Verifies that the contents of a deliveries are returned when called |
| insertDelivery | Ensures a new deliveries can be made, and verifies the correct http response |
| deleteDelivery | Verifies that deliveries can be removed from system, and the correct http response is returned |
| updateDelivery | Verifies that delivery information can be updated, and returns the correct http repsonse |

Notifications

|  |  |
| --- | --- |
| allNotifications | Verifies that all notifications present in the database are returned when called |
| insertNotification | Verifies that notifications can be added to the database, and verifies that the correct http response is returned |
| deleteNotification | Verifies that notifications can be deleted, and that the correct http response is returned |
| updateNotification | Verifies that notifications can be updated once in the database, and that the correct http response is returned |

## System Testing - Acceptance Criteria

For the system testing we will use the acceptance criteria from our user stories which are listed in the user stories section of this document

### Sprint 1 – Acceptance Testing

For sprint 1 due to a planned lack of connectivity with the frontend and backend no user stories are near enough to completion to require acceptance testing. As mentioned above for sprint 2 acceptance testing will use the acceptance criteria from user stories to dictate the acceptance tests being run.

### Sprint 2 – Acceptance Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Number | Scenario | User Story | Context | Result (Y/N/Partial) |
| 1 | Customer comparing prices for a product with comparisons | #1 Price Comparison | Given I am a customer  When I click to view price comparisons of a product  Then the system will return a results page of stores with similar or lower prices | Y |
| 2 | Customer comparing prices for a product without comparisons | #1 Price Comparison | Given I am a customer  When I click to view price comparisons of a product  Then the system will return a page without price comparisons for this product | Y |
| 3 | Worker ordering groceries for delivery | #2 Delivery | Given I am a full-time worker or student,  When I click on the ‘Delivery’ option after adding items to my cart,  Then the system will send a notification confirming delivery | N |
| 4 | Customer orders groceries for delivery and enters invalid address | #2 Delivery | Given I am a full-time worker or student,  When I click on the ‘Delivery’ option after adding items to my cart,  Then the system will prompt me for a correct delivery address | N |
| 5 | A user with an account wishes to view an item's price | #3 Browsing | Given I am logged into an account  When I navigate the site  and find an item I like  Click on said item  Then I can see the price of said item | Y |
| 6 | A user without an account wishes to view an item's price | #3 Browsing | Given I am not logged into an account  When I navigate the site  and find an item I like  Click on said item  Then I can see the price of said item | Y |
| 7 | Mother wants to be notified of the best deals | #4 Notifications | Given I have chosen for deal notifications  When businesses add new deals to their websites  Then the system will send a notification alerting of the new deals available. | N |
| 8 | Mother wants to be notified of the price of a specific product | #4 Notifications | Given I have clicked the notifications button on a product page  When the price changes for this product  Then the system will send a notification alerting of the price change | N |
| 9 | User searches for a product by key terms | #5 Search Bar | Given I have entered the key terms for my search  When I click the search option  Then all items matching key terms will appear | Y |
| 10 | User searchers for all products | #5 Search Bar | Given I have entered no terms in the search bar  When I click the search option  Then all items will appear | Y |
| 11 | User search returns no products | #5 Search Bar | Given I have entered the key terms for my search  When I click the search option  and no products match those terms  Then the website displays an message about no matching items | Y |
| 12 | A new user is navigating our website for the first time | #7 Navigation | Given I am a new user on the website  When I look around the website  Then I want to able to figure out what each button/link will do | Y |
| 13 | Customer who is unsure of what exactly they want to search for products | #8 Categorisation | Given I have selected/searched for the category of products I want  When I hit the search button  Then I want to find products under that category  And find the product I am looking for  And find products like what I may be looking for | Y |
| 14 | Customer wants to filter out groceries too far away | #11 Search filters | Given the customer is on the map feature to search for grocers offering prices  When the customer filters for only grocers within 5km of location  Then the map should only display grocers within a 5km radius | N |
| 15 | Customer want to filter groceries by a feature (e.g Vegetqrian/Gluten Free) | #11 Search Filters | Given the customer has selected their desired feature  When the customer selects the filter button  Then all items are filtered to match their filters | N |
| 16 | Customer opens item with historical pricing | #15 Historical Pricing | Given I am a customer  When I click onto an item  Then I see the current price  and I see a list of previous prices and specials | N |
| 17 | Customer opens item without historical pricing | #15 Historical Pricing | Given I am a customer  When I click onto an item  Then I see the current price  And a message that historical pricing is unavailable | N |
| 18 | customer logging on correctly | #16 Login | Given I have an account  When I go to login  And enter my correct details  Then I can sign into my account | Y |
| 19 | Customer logging on incorrectly | #16 Login | Given I have an account  When I go to login  And enter incorrect details  Then I am prompted that entry was incorrect and offered to reset password | P – no reset link but error |
| 20 | customer logging on without account | #16 Login | Given I don't have an account  When I go to login  Then I am presented with the option to create an account | Y |
| 21 | customer changing light themes | #17 Different Lighting Themes | Given I am logged in to the website  When I go to the settings  Then I can change the lighting theme of the webpage | N |
| 22 | Customer track delivers | #13 Track Deliveries | Given I have logged into my account  When I click on track delivers  Then I can see my current deliveries  And I can see how long they take till delivery | N |
| 23 | customer opens create account page | #12 Create account | Given I am a new user  When I press the create account button  Then I am given a page to input my details | Y |
| 24 | customer inputs details and creates an account | #12 Create account | Given I am on the create account page  and I have input my details  When I press the create account button  Then an account is generated for me | Y |
| 25 | A user has forgotten their password | #10 Recover Password | Given I have an account and I have forgotten the password  When I press the forgot password button  Then I am sent a recovery email | N |
| 26 | User opens their recovery email | #10 Recover Password | Given I have clicked on the link in the recovery email  When I enter my new password  Then my password is updated | N |
| 27 | User has found a product and wishes to add it to their cart | #9 Add item to Shopping cart | Given I have chosen an item  When I add an item to my cart  Then the item I added will appear within it  and cart price updates | Y |
| 28 | User has found a product and wishes to add multiple to their cart | #9 Add item to Shopping cart | Given I have chosen an item  Given I have selected my quantity  When I add the item to my cart  Then selected quantity will be added to my cart  and cart price updates | P – can add multiple only through the cart option |
| 29 | A user has an item they wish to remove from their cart | #6 Remove item from Shopping cart | Given I have an item in my shopping cart  When I select to remove an item from my shopping cart  Then the item I selected will be removed  and the shopping cart price is updated | Y |
| 30 | A user has found an item they what to learn about | #18 View Item | Given I have found an item  When I click on an item  Then I see the details of the item  And have the option to add to cart | Y |
| 31 | User opens their shopping cart with items | #19 View Cart | Given I have items in the shopping cart  When I click on the shopping cart  Then I can see the items in the cart | Y |
| 32 | User opens their shopping cart without items | #19 View Cart | Given I have no items in the shopping cart  When I click on the shopping cart  Then I am prompted with a message that shopping cart is empty | P – no message for empty cart |
| 33 | User has items in their cart and wishes to reduce the quantity | #20 Changes Quantity of Item From Shopping Cart | Given I have an item in my cart  When I reduce the quantity  and I hit save  Then the quantity is decreased  And the cart price is updated | Y |
| 34 | User has items in their cart and wishes to increase the quantity | #20 Changes Quantity of Item From Shopping Cart | Given I have an item in my cart  When I increase the quantity  and I hit save  Then the quantity is increased  And the cart price is updated | Y |
| 35 | Customer has a cart with items | #21 Checkout Cart | Given I have opened my cart  When I hit the checkout button  Then I am taken to the checkout screen | N |
| 36 | Customer has a cart without items | #21 Checkout Cart | Given I have opened my empty cart  When I hit the checkout button  Then I am notified my cart is empty | N |

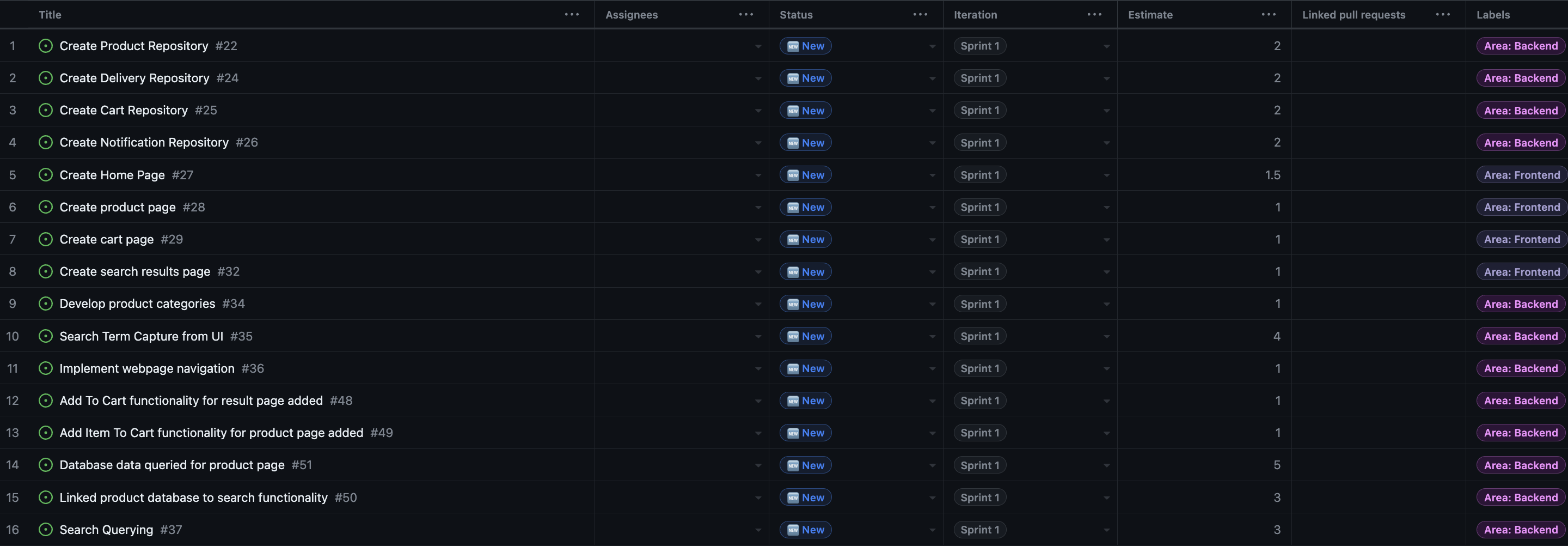
# Product Backlog

# Sprint 1 and 2

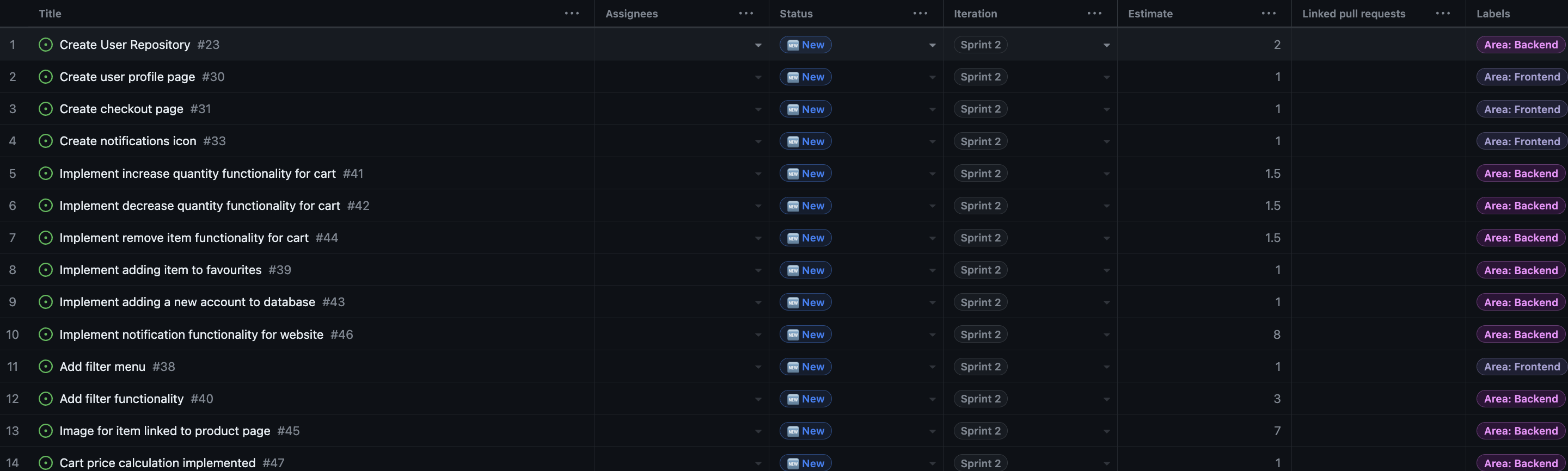
The sprint 1 and 2 backlogs are recorded inside the GitHub Project Board screenshots are provided here for reference

GitHub Project Board Link: <https://github.com/orgs/cosc2299-sept-2023/projects/163>

## Estimated Backlog – Sprint 1



## Estimated Backlog – Sprint 2



## Actual Backlog – Sprint 1



Added Tasks

* Create Flyway migration for database schema #57
* Return list of items in deliveries#56
* Create DB Schema#52
* Develop Practice Products#53

Moved Tasks Sprint 2 to Sprint 1

* Implement remove item functionality for cart#44
* Implement decrease quantity functionality for cart#42
* Implement increase quantity functionality for cart#41
* Create user profile page#30
* Create checkout page#31

Tasks Removed Due to Lack of Relevancy

* Implement Webpage Navigation#36

Tasks Moved to Sprint 2 From Sprint 1

* Search Term Capture from UI#35

## Actual Backlog – Sprint 2



Added Tasks

* Connecting backend to search page #76
* Port test data into database #81
* User login #75
* Connecting backend to product page #77
* Connecting backend to notifications page #79
* Connect Backend to user profile page #78
* Connect Backend to cart page #80

Tasks Added from Sprint 1 to Sprint 2

* Search Term Capture from UI#35

Tasks Removed from Sprint 2 and added Sprint 1

* Implement remove item functionality for cart#44
* Implement decrease quantity functionality for cart#42
* Implement increase quantity functionality for cart#41
* Create user profile page#30
* Create checkout page#31

Tasks Moved to Sprint 3 From Sprint 2 due to time constraints

* Create notifications icon #33
* Implement notification functionality for website #46
* Implement adding item to favourite #39

# Sprint 0 Retrospective

1. Things That Went Well

The team rapidly built rapport, aligning closely on project goals, with each member demonstrating commitment. The distribution of the workload was fair and even, fostering a healthy work environment. Communication was constructive, positive, and frequent, a factor that has contributed to the team being on track to achieve project milestones. Our proactive approach has been a highlight of our success.

2. Things That Could Have Gone Better

Our client interactions were lacking some depth and detail, regarding information. In particular, the functional and non-functional requirements were not sufficiently explored. To improve this, a well structured and planned approach to client meetings is needed, including prepared questions to obtain a clearer understanding of their needs and expectations.

3. Things That Surprised Us

We were confronted with unexpected challenges in task distribution, resulting in a need for collaborative effort on certain sections. The number of meetings required each week was also a surprise, with a minimum of four being necessary. A difficulty in understanding specific tasks due to conflicting specifications was an unforeseen challenge. After careful examination and discussion, we overcame this, but it served as a learning opportunity.

4. Lessons Learned

Our experience has taught us the importance of a flexible approach to teamwork, recognizing that a mix of individual and small group work can be most effective. More careful planning and a structured approach to client interactions are vital, as well as the need to host regular meetings to ensure we are all on the same page and have a clear understanding. These lessons will be essential for the upcoming sprints, contributing to ongoing refinement and success.

5. Final Thoughts

So far, the project and teamwork has gone smoothly, with the team looking forward to the upcoming sprints and working collaboratively.

# Sprint 1 Retrospective

1. Things That Went Well

The focus of this sprint was the development of both the frontend and the backend. Our strategy was to divide the team into two groups. Jamie, Kiran, and Peter were assigned to the frontend, while Andy, Ben, and Tyler focused on the backend. By equally distributing the tasks and maintaining team collaboration, we managed to complete our set milestones. This sprint, we accomplished a template website, which is poised to be integrated with the data from our almost fully developed backend.

2. Things That Could Have Gone Better

Pivotal feedback from our client suggested that the integration of the frontend with the backend is a more intricate task than anticipated and should be approached over multiple sprints. However, this feedback came in later than desired due to the absence of the client during our scheduled meeting. Consequently, our roadmap for the next sprint is more intensive. A technical challenge arose when Jamie used the Pulsar editor, leading to a direct commit into the main branch on GitHub rather than creating a separate branch. However, these commit compliactions were not know prior to him using the Pulsar editor.

3. Things That Surprised Us

For this sprint a key surprise was the difference between the actual tasks completed and the task descriptions. Many tasks had descriptions that were too simplistic or only covered one aspect of a specific task although the timing for given tasks was still roughly equivalent. Ultimately, we believe that this can be attributed to a lack of understanding about what the exact tasks entailed which prevented accurate descriptions to be written but our understanding was good enough to map out the general timing and effort need for these tasks.

Additionally, some tasks had to be added like task #57 flyway migration and adjusted like #24 which was changed to ‘basic delivery functionality’ from ‘implement delivery repository’ this was done to account for oversights in our planning. We believe this is similarly attributed to lack of understanding about the finer details of implementing this project.

4. Lessons Learned

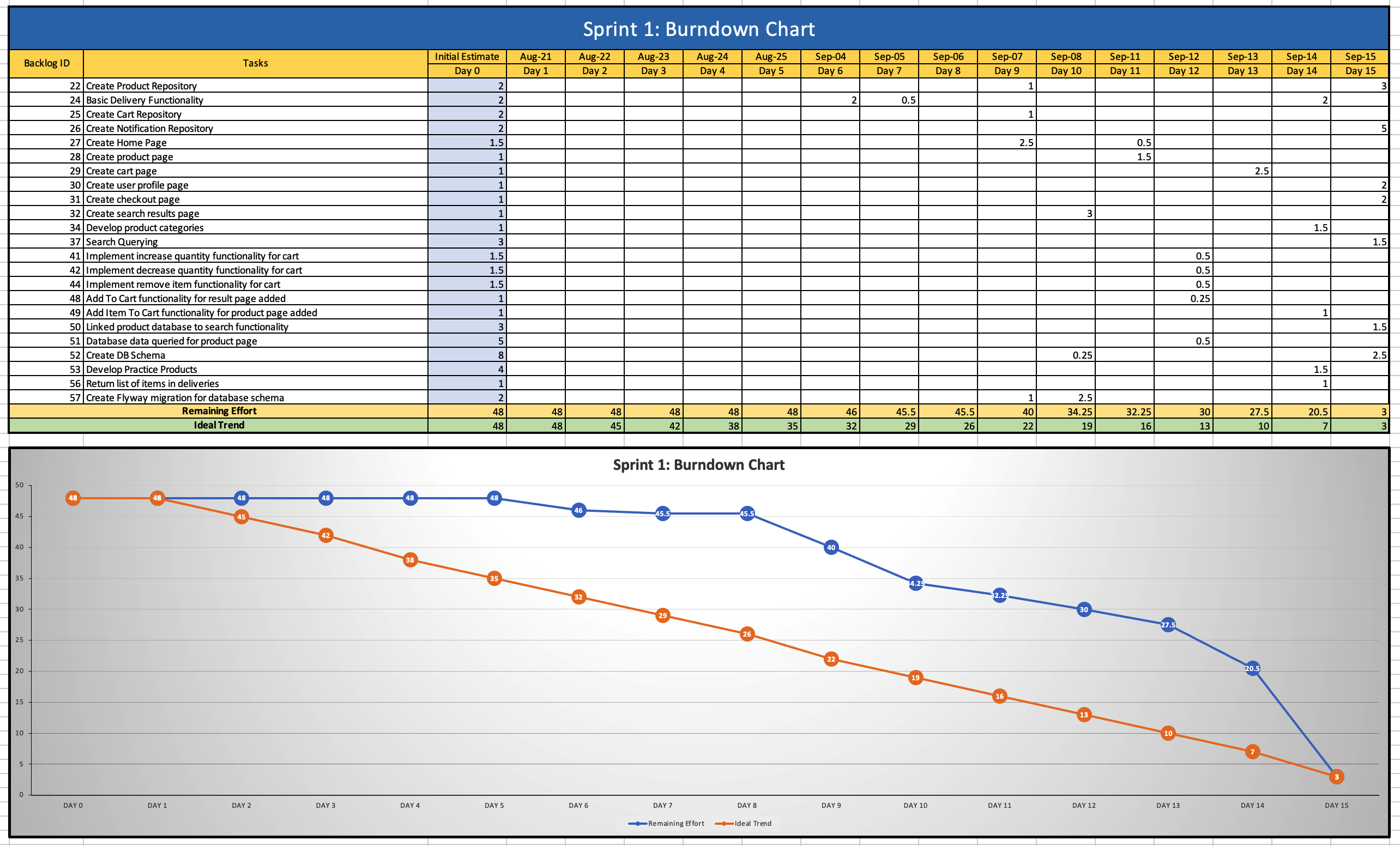
This sprint highlighted the significance of adhering to the best branching practices. Jamie's direct commit from a different editor served as an important reminder of this, even though it did not result in any immediate problems. Emphasizing the importance of understanding the nuances and potential pitfalls of various tools.

5. Final Thoughts

Overall, the sprint was a success. Our teamwork shone through, enabling us to meet our objectives. As we move forward, we will incorporate these insights gained, to ensure an improved next sprint, with smoother workflows and better outcomes.

# Sprint 1 Burndown Chart

In the Milestone 2 folder in the git repository is the actual file because the information in this image below, is quite small.



# Sprint 2 Retrospective

1. Things That Went Well

The focus of this sprint was to connect the frontend and backend, adding search and filtering functionality, implementing users and docker functionality. These tasks were distributed evenly between our group. By equally distributing the tasks and maintaining team collaboration, we managed to complete our set milestones. This sprint, we accomplished a functional website with docker integration.

2. Things That Could Have Gone Better

During task division, the scale of each tasks time estimations, were severely underestimated. This resulted in many tasks not being completed when expected. An example of this was connect the frontend to the backend, being assigned to one person. This was the result of our group believing that once one connection is made between both ends, the rest would come much easier. Unfortunately, this assumption was untrue, resulting in a lot of help being needed from other team members to complete the task. This caused abandonment some functionality, leaving the features notifications and favourites behind.

3. Things That Surprised Us

As touched on earlier, the main surprise was the scale of certain tasks and our under estimations of it. This really impacted the flow of the project’s projected millstones completion times, as we were behind target most of the time. The time some tasks took were well over three times what we expected, which made our task allocation not ideal. Furthermore, we had to reduce some features to compensate for this.

4. Lessons Learned

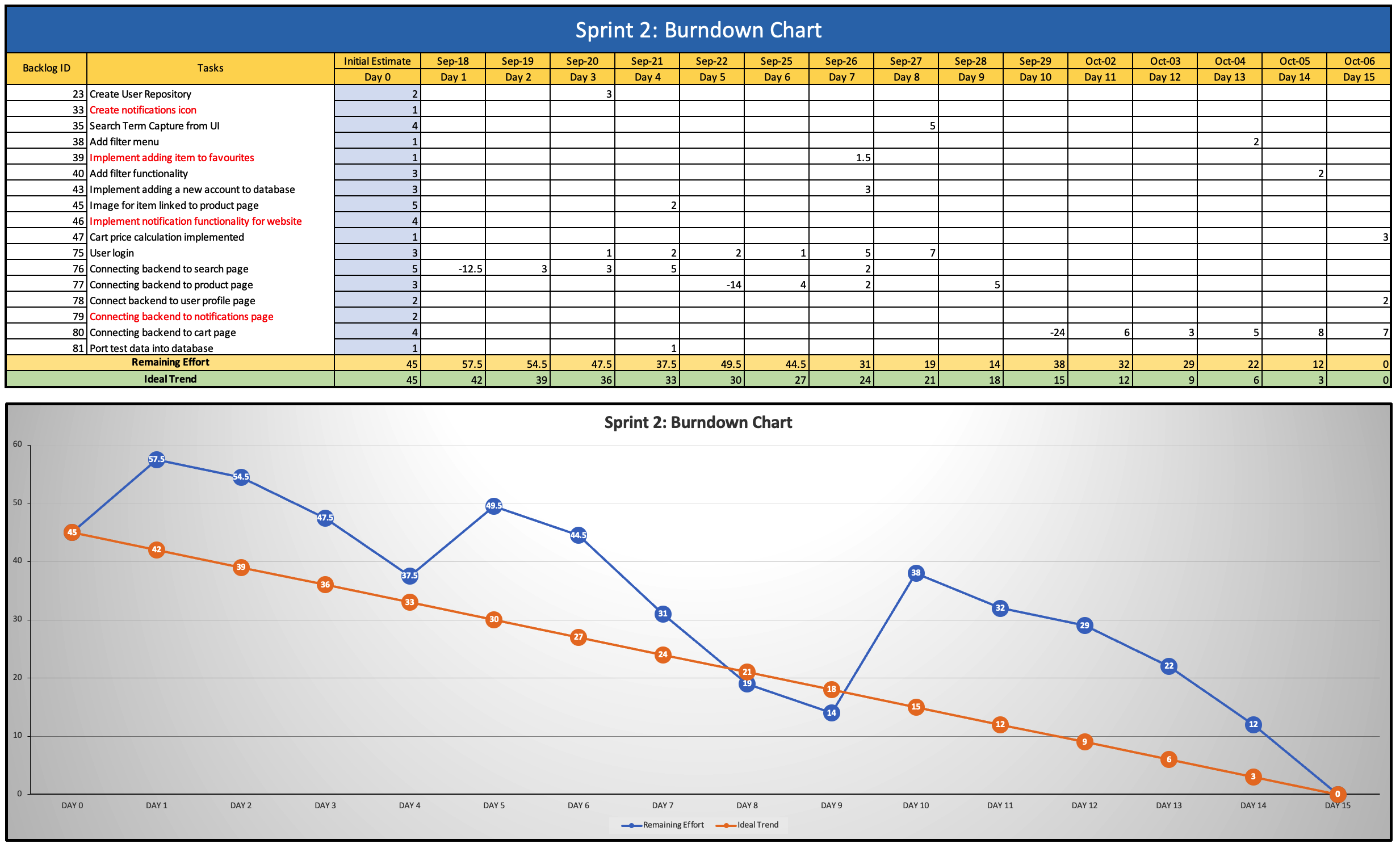
This sprint highlighted the significance of extensively researching tasks time for a better estimation of how long each task takes. This coupled with a slight overestimation of time, to account for issues, would result in a more accurate expected workload. This would also allow for tasks to be more evenly distributed and a better projection of what is considered achievable, well before the sprint ends. In our case, we would have been able to evaluate the importance of keeping the features notifications and favourites, well before the sprint was ending.

5. Final Thoughts

Overall, the sprint was a success. Though we were unable to complete all the functionality we aimed to do, we still have a functional website with docker implementation. We have learned from our mistakes from our previous sprints, which shows growth throughout our teams, as we had not encountered the troubles we had during the sprint before. We all worked in tandem, helping each other when needed, and are happy with our final product.

# Sprint 2 Burndown Chart

In the Milestone 3 folder in the git repository is the actual file because the information in this image below, is quite small.



# SRS Appendix

## Delta

For our delta between Milestone 1 & 2, no features were dropped.

For our delta between Milestone 2 & 3, no features were dropped.

For our delta beyond Milestone 3, features that were dropped included price history, notifications and favourites. Price history was a feature that would show the timeline of price changes for a particular product. This feature was not implemented due to time constraints. Favourites was a feature where users could save particular products into their favourites that could be viewed on the user profile page. This feature was not implemented due to time constraints. Notifications was a feature that would notify the user of any special discounts on their favourite items. Similarly, this was also not implemented due to time constraints. These features would be aimed to be completed in future sprints.