Software Requirements Specification

for

*SuperPrice*

Version 1.0

Prepared by P04-02

Software Engineering: Processes & Tools

06/08/2023

Table of Contents

Table of Contents ii

Version History ii

1. Introduction 1

1.1 Overview 1

1.2 Purpose 1

1.3 Stakeholders 1

2. Features & Requirements 1

2.1 Functional Requirements 1

2.2 Non-Functional requirements 3

3. Use Cases 3

4. System Architecture 7

5. Data Model 8

6. UI Design 9

7. Assumptions & Constraints 14

8. Dependencies 14

9. Testing & Acceptance Criteria 14

Appendix A: Glossary 16

Appendix B: Additional Documentation 16

Sprint 1 Planning Notes 16

Sprint 2 Planning Notes 16

Sprint 0 Retro 17

Version History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Description/Changes** | **Version** |
| Milestone 1 | 20.08.2023 | Initial Version | 1.0 |

# Introduction

## Overview

SuperPrice is a website focused on comparing prices on a product offered at several stores and finding the best price possible for a customer. It also allows users to manage deliveries of products they purchase through the website. Retailers will be able to affiliate themselves with the application by paying a commission fee.

## Purpose

The purpose of the product is to find the best deals for customers, and make managing the deliveries of those purchases easier for customers. Retailers will also have the opportunity to have their products promoted to customers, increasing business.

## Stakeholders

The stakeholders of this project include the product owner, who will benefit from the successful launch of the product. The customers, who will save money on their purchases. The developers who are responsible for the production of the application. And participating retailers, who will have the opportunity to promote their business through the application.

# Features & Requirements

## Functional Requirements

### Product Search & Categorisation

Users will be able to search the catalogue of products. They can either search for a specific product, or browse by category.

To search for a specific product, the user can type the name of the product into the search bar, click the search button, and any matching results from the database will be listed. To browse by category, the user will be presented with a selection of categories to choose from. After clicking on a category, they will be presented with a list of products under that category.

After being presented with the search results, they can also click on a specific product to see a focused view of the product which will include a more detailed overview about the product.

**REQ-1:** The website will display a search bar, in which a user can type the name of a product they are searching for into. There will also be a search button that triggers the app to load the results of the search.

**REQ-2:** The website will present a list of available product categories, which the user can select one of at a time. Once they have selected one, only products under that category will be presented to the user.

**REQ-3:** In the database, each product will contain an image, name, price, retailer, and category attribute.

**REQ-4:** The search bar queries will be validated to prevent SQL injection.

**REQ-5:** In the case of a search yielding no results. The user will be notified that the search produced no results.

**REQ-6:** When presented with the list view of products, the name, image, price, and associated retailer will be shown for each product.

**REQ-7:** The focused view of a product will contain the name, images, description, and all price listings and retailers for the product.

### Price Comparison

To help the user find the best deal, the user will always be presented with the best price offering first. When presented with search results, the best price of that product and the associated retailer will be shown. To view more prices, a user can click on the product search result, taking them to a focused view of the product. Here, all prices and associated retailers for that product will be shown in ascending order.

**REQ-1:** In the search results listings. The lowest price for each product listing will be shown along with associated retailer.

**REQ-2:** In the focused view of a product. All the available prices for that product will be displayed in ascending order. The retailers offering each price will also be shown alongside.

**REQ-3:** In the case that multiple prices are equal, the listings will be prioritized by rating, then alphabetically by retailer names.

### Delivery Organization

The user will have the ability to add products to their cart and checkout through a standard online shopping experience. In particular, during the delivery selection phase, the user will be presented with available delivery options offered, and the available time slots to be delivered in.

**REQ-1:** During checkout, the user will be presented with the available delivery options to choose from. The user can select either same day delivery, or pick any future date to deliver on.

**REQ-2:** After selecting a delivery day, the user can also choose a particular time-frame for their delivery. This includes morning, afternoon, and evening options.

### Notifications & Alerts

The application will send notifications and alerts to a user about price drops and special offers. To do this, users can opt-in to a mailing list by providing their email address. They will then be notified of any offers through an email to the provided address.

**REQ-1:** On the footer of the website, there will be an option to sign up to the mailing list. The user is prompted to input their email address, and clicking the submit button will add their email to the mailing list.

### User Reviews & Ratings

The user will have the ability to place reviews and ratings on products they have previously purchased through the application. Any reviews and ratings stored in the database will also be shown in the focused product view pages.

**REQ-1:** From their account page, the user will be presented with an option to review products. On clicking this, they will be presented with a list of previously purchased products. From here, they can navigate focused view page for that product.

**REQ-2:** The database will have a review and rating object that is associated with both a product and customer.

**REQ-3:** On the focused view of a product, there will also be a prompt to make a review for that product from that page. If they have previously purchased the product, they will be able to leave a review. Otherwise, they will be notified that they cannot leave a review if they have not purchased it before.

**REQ-4:** On the focused view of a product, the average rating and a list of all reviews, with their individual ratings will be shown.

## Non-Functional requirements

Usability

The product will have a user-friendly and intuitive interface. This will make it accessible to a wide range of users. The website will follow conventional web design patterns that will be familiar to the user. *See UI Designs for more details on the design of the application.*

Performance

The search function is expected to produce results in an acceptable time frame. Exact times TBD.

Security

The products search feature should prevent SQL injection.

# Use Cases

**User Story 1:**

**As a** frequent shopper, **I want** to be able to look up specific products and different categories, **so that** I can quickly find the items I need.

Tasks

UI Design

* Design the layout for homepage to include a prominent search bar
  + - Estimation: 2 hours

Front-End Implementation:

* Implement the homepage using HTML and CSS
  + - Estimation: 3 hours

Search Functionality - Front end

* Set up Javascript functions to handle user input and trigger search queries
  + - Estimation: 3 hours

Search Functionality - Back-end

* Write SQL queries for search function
  + - Estimation: 3 hours

Database Integration

* Design the database to store product information
  + - Estimation: 3 hours

**User Story 2:**

**As a** budget conscious customer, **I want** to search for specific products and compare prices across different supermarkets in my area, **so that** I can easily identify the stores offering the lowest prices for my desired item.

Tasks

UI Design

* Design the layout for displaying items with details such as price, store etc.
  + - Estimation: 2 hours

Front-End Implementation:

* Implement the item display layout using HTML and CSS
  + - Estimation: 3 hours

Sort Functionality - Front end

* Set up Javascript functions to sort products based on price, distance etc.
  + - Estimation: 3 hours

Sort Functionality - Back-end

* Write SQL queries to sort products.
  + - Estimation: 2 hours

**User Story 3:**

**As a** busy individual, **I want** to have multiple delivery options, including to be able to set delivery times, **so that** I can easily organise deliveries to my house and be efficient to save time.

Tasks

UI Design

* Design the layout for the ordering page with delivery options.
  + - Estimation: 2 hours

Front-End Implementation:

* Implement the ordering page and delivery option layout using HTML and CSS
  + - Estimation: 3 hours

Delivery Options Functionality - Front end

* Set up a form for users to select delivery option.
  + - Estimation: 1 hours

Delivery Options Functionality - Back-end

* Write SQL queries to get details for different delivery options such as price, time estimate etc.
  + - Estimation: 3 hours

**User Story 4:**

**As a** time, constrained shopper, I would want to have a quick and easy price comparison to save time and money when comparing different brand options.

Tasks

UI Design

* Design the layout for comparing prices for a product between different brands.
  + - Estimation: 2 hours

Front-End Implementation:

* Implement the comparison page layout using HTML and CSS
  + - Estimation: 3 hours

Price Comparison Functionality - Front end

* Set up Javascript functions to sort items that are being compared to.
  + - Estimation: 2 hours

Functionality - Back-end

* Write SQL queries to get items from different stores that are similar to a given item.
  + - Estimation: 2 hours

**User Story 5:**

As a mother of 4 children, I would want to buy reliable and best quality products for my children by looking at reviews for an item so that I can decide if the item will be safe to use on my children. I would like to place reviews on items I have purchased to let others know the quality of the product.

Tasks

UI Design

* Design a review section for the product page.
  + - Estimation :1 hour

Review Functionality - Front end

* Set up Javascript functions to display reviews.
  + - Estimation :2 hour

Review Functionality - Back-end

* Write SQL queries to retrieve reviews.
  + - Estimation :2 hour

**User Story 6:**

**As a** savvy shopper always on the lookout for special offers, **I want** the application to provide timely notifications and alerts about price drops and exclusive promotions, **so that** I never miss out on great deals and discounts, enabling me to make the most of my shopping budget.

Tasks

UI Design

* When logging in, user can opt to select to subscribe to newsletters and announcements
  + - Estimation :1 hour

Front end Functionality

* Add JS functions to save an input email to the database
  + - Estimation :3 hours

# A diagram of a company Description automatically generatedSystem Architecture

**Presentation Layer:**

* The presentation layer is responsible for displaying a user interface for the end user to interact with the website.
* The data presented by the presentation layer will be retrieved through the application layer.
* The technology chosen for this layer is NextJS and is used over ReactJS as NextJS is used to create web applications through server-side rendering, overall improving the performance of the website on the end-user side.

**Application Layer:**

* The application layer is responsible for providing logic and interactions between the presentation layer and the data layer.
* The layer consists of the gateway and the product microservice. This microservice will interact with the data layer, requesting and receiving product data. The gateway will then respond to the request from the presentation layer.
* The technology chosen for this layer is Spring Boot. The choice of Spring over other Java APIs, for example Javalin, is that Spring Boot makes it easy to create stand-alone, production-grade APIs that can be easily run.

**Data Layer:**

* The data layer is responsible for providing a gateway for the application layer to request and retrieve product data.
* Product requests are processed and will interact with an external database to retrieve the requested data. This data is then sent back to the application layer.
* The technology used for this layer is SQLite, as it allows for a lighter application without requiring an external service to run. SQLite directly stores data into a singular file, making it easy to copy and backup. Furthermore, considering that the data stored is very basic, it does not require a more complicated database like MySQL, and does not require any complex configurations.

# Data Model

In the database, we will have the following entities:

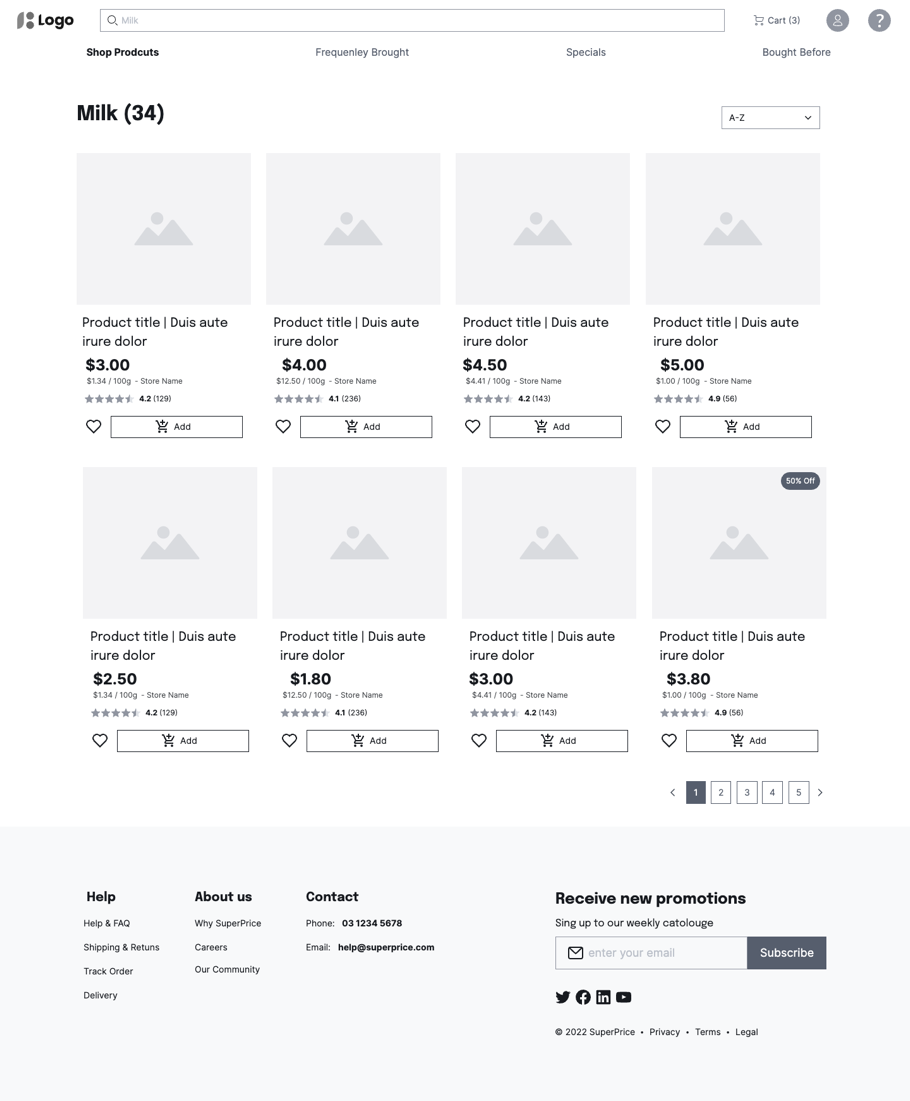
* Customer
* Product
* Review

# UI Design

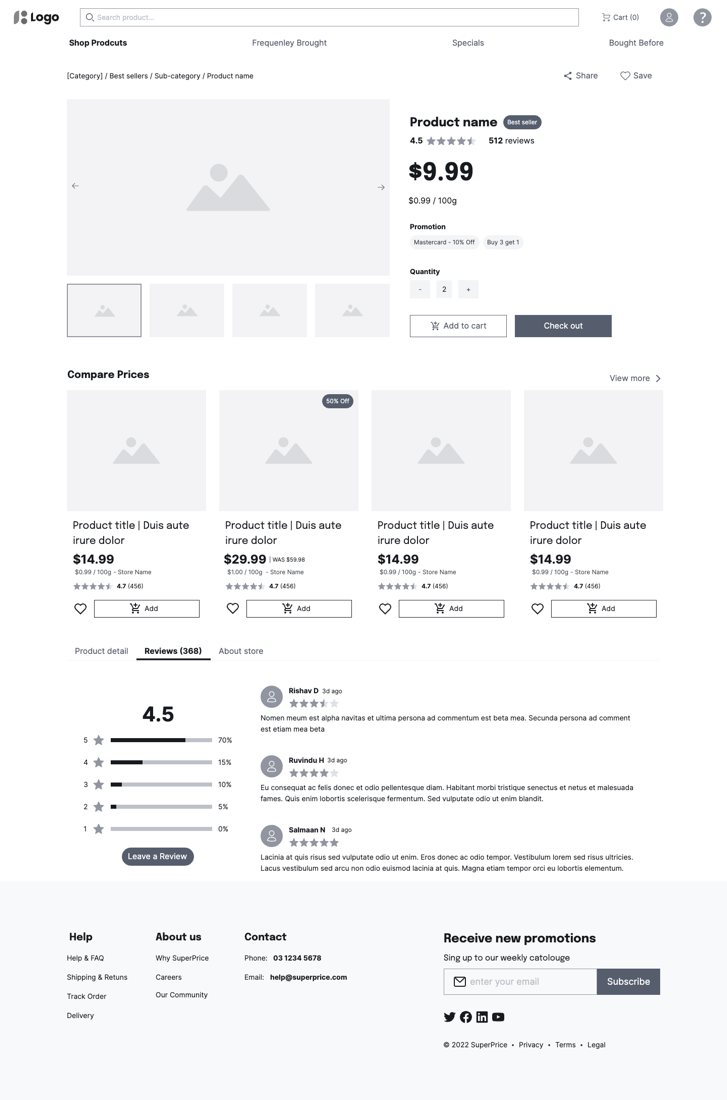
**A screenshot of a website

Description automatically generatedHome/Landing**

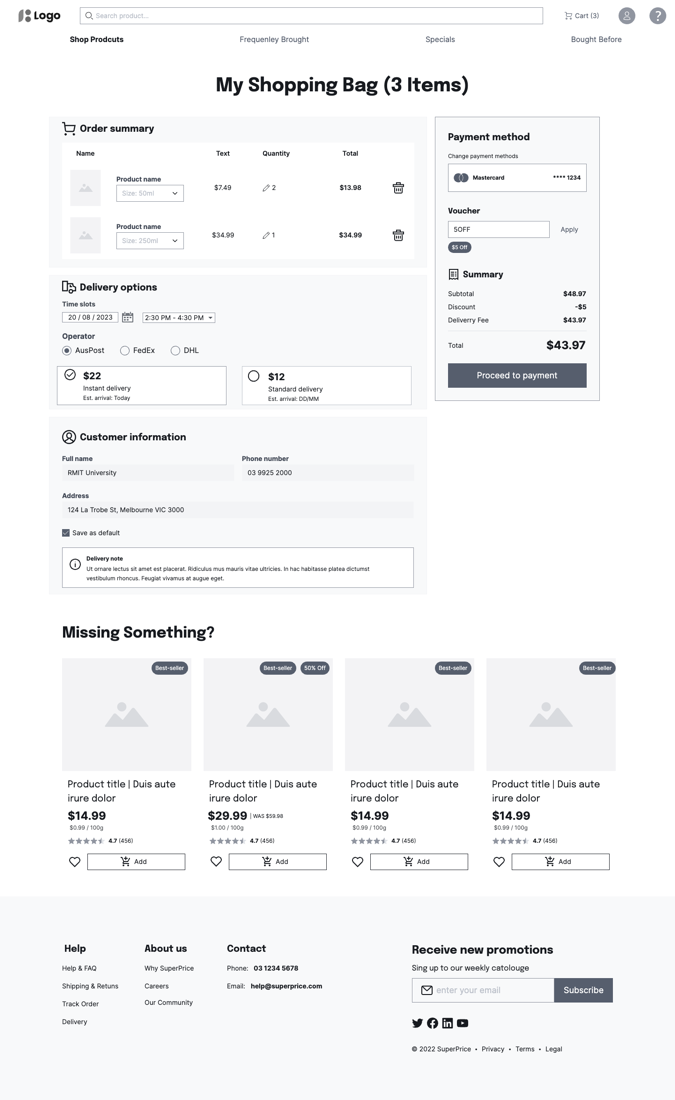
**Search Results**

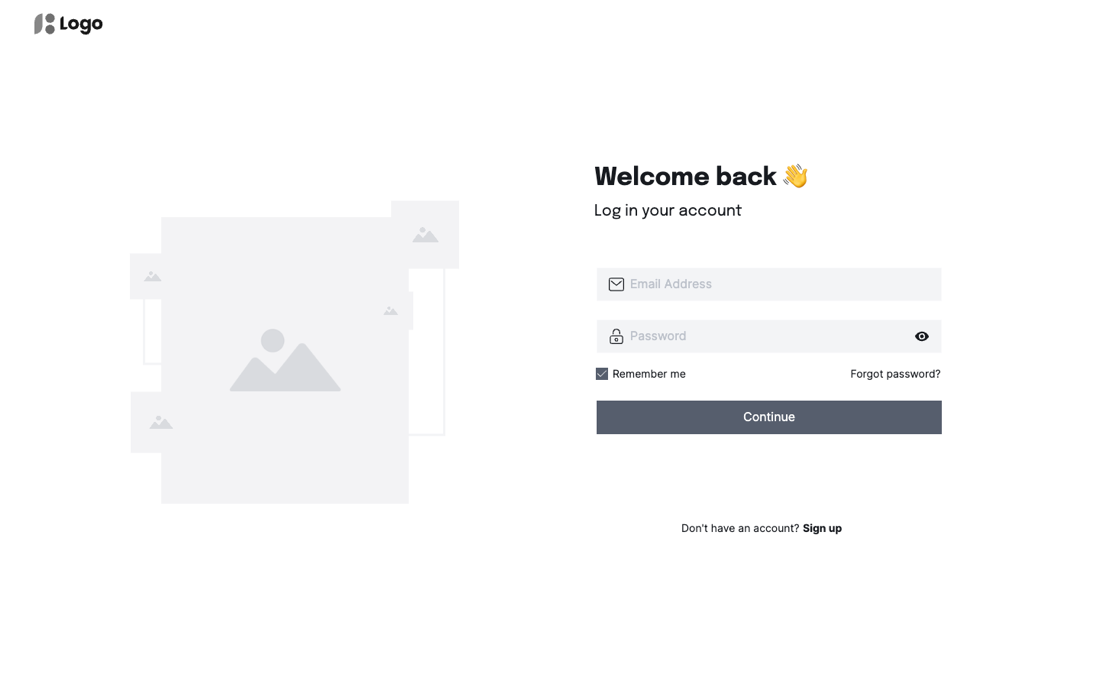


**Individual Product**

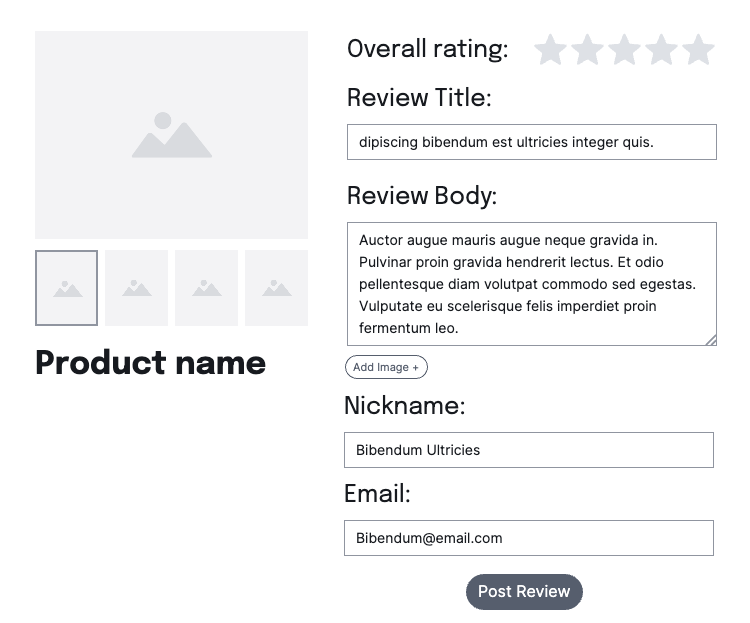


**Checkout**

**Sign-in**



**Review**



# Assumptions & Constraints

We are creating the website in English only.

We are creating our own products and prices for demonstration purposes, because we don’t have access to supermarket databases.

We are using Typescript, Javascript for the front-end programming, Java and SQL for back-end programming.

We are not using HTTPS as we do not own a server for hosting the website and thus cannot get a SSL certificate.

# Dependencies

The product depends on the following external systems, libraries, and/or API’s.

* SpringBoot
* Next.js
* SQLite

# Testing & Acceptance Criteria

User Story 1 Acceptance Criteria

**Given:** The user is in the applications main page.

**When:** The user enters a product name, brand, or keyword in the search bar and clicks the search button.

**Then:** The application should display relevant search results instantly, matching the user's input.

**And:** The user should be able to navigate through different product categories.

User Story 2 Acceptance Criteria

**Given:** The user is in the applications main page.

**When:** The user enters a product and sorts the prices of the searched product.

**Then:** The application should display the stores that are offering the lowest price for the product.

**And:** The user should be able to get products between a given price range.

User Story 3 Acceptance Criteria

**Given:** The user is in the ordering page of a product.

**When:** The user enters a specific date and time for the order to be delivered.

**Then:** The application should display an order placed confirmation

**And:** Give an option to set a reminder for the date the order is placed to.

User Story 4 Acceptance Criteria

**Given:** The user has selected a product and is on the page of this product.

**When:** The user clicks compare prices.

**Then:** The application should display the same or similar products from supermarkets with the prices.

User Story 5 Acceptance Criteria

**Given:** The user is on a page after searching a product.

**When:** The user clicks on an item.

**Then:** The application should display the reviews of the item along with the item details.

User Story 6 Acceptance Criteria

**Given**: The user has made an account for the application

**When**: There are available promotions in the shop

**Then**: An email is going to be sent to the user of the promotions that are available.

Appendix A: Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| MVP | Minimum Viable Product |
| ERD | Entity-Relationship Diagram |

Appendix B: Additional Documentation

## Sprint 1 Planning Notes

Duration: 3 Weeks

Goal: The aim of Sprint 1 is to create a minimum viable product.

What is the team’s vision for the sprint?

We have mainly committed the items from the product backlog that focus on producing a functional product by the end of the sprint. With an MVP, we will be able to get feedback on the functionality of the product. In its working form, the MVP will have all the main features listed in section 4.

Estimation

We estimate that the tasks for sprint 1, as we are only creating an MVP will require less effort than sprint 2.

## Sprint 2 Planning Notes

Duration: 3 Weeks

Goal: The aim of Sprint 2 is to turn the MVP into a finished product.

What is the team’s vision for the sprint?

We want to create a viable product that has the full functionality to successfully operate and satisfy the requirements of the product owner. If full functionality cannot be achieved, then we will try to complete the required requirements so that the program can be operational, possibly sacrificing the implementations of some non-functional requirements.

<Which items of the product backlog will be committed to the sprint backlog and why?

What will the potentially shippable product look like in the end?

What features will it have in its working form?>

Estimation

We estimate that sprint 2 will be where most of our effort goes into. Polishing the website to a final product will require more effort than the MVP. We may also run into problems in sprint 1 that will need to be addressed in sprint 2, which will require more effort.

## Sprint 0 Retro

Things That Went Well

* + The team collaboration, everyone did their tasks on time and was able to join weekly meetings to keep everyone updated.
  + Everyone was happy with what we achieved, and we had no arguments about who did more. Work was fairly distributed.

Things That Could Have Gone Better

* The team could improve on keeping up to date with requirements.

Things That Surprised Us

* + The amount of documentation that is needed.
  + How we jumped straight into the sprint. Had to hit the ground running.

Lessons Learned

* + Scrum Standups require every individual to speak individually rather than as groups.

Final Thoughts

Things to keep

* + - Collaboration and fair work distribution, as well as keeping each other up to date.

Things to change

* + - Keeping more up to date with the requirements
    - Making sure that everyone is individually contributing to each SCRUM meeting