Software Requirements Specification

for

SuperPrice

Version 1.02 approved

20 August 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Charles Earnshaw	15/09/2023	Incorrect architecture diagram	1.01
Charles Earnshaw	17/09/23	Changed database model used	1.02

		note: as of now we still plan to implement the features in the SRS as they are written although the Milestone 2 may not currently reflect that	
Charles Earnshaw	25/09/23	Due to time constraints, removed most functionality relating to the delivery and shopping cart process as well as other features. Functionality removed includes ● 4.5 Alternative products shown on product page (PRO-2) ● 4.6 Choosing time slot for delivery (CHE-1) ● 4.7 Viewing Shopping Cart (SHO-1) ● 4.8 Notifications and alerts (HOM-3) ● 4.14 Placing an order (CHE-2) ● 4.15 Credit Card Payment (CHE-3) ● 4.15 Cancel Order (CHE-4) ● 4.16 Adding items to shopping cart (PRO-3)	1.03

1. Introduction

1.1 Purpose

The purpose of this document is to specify the software requirements of the SuperPrice application. The SRS will cover aspects such as the home page, searching, login, product page, shopping cart and checkout. This document will contain all parts of the application, frontend and backend, but does not include any off-the-shelf dependencies.

1.2 Document Conventions

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>

No conventions as of yet

1.3 Intended Audience and Reading Suggestions

Developers	The document will provide detailed functional and nonfunctional requirements as well as wireframes, class diagrams and architectural diagrams of which developers can use as a reference during development.
Clients	Clients are expected to use this document inorder to ensure that the system properly aligns with their business goal and requirements
Testers	Testers will use this document to design test cases and to ensure that the functional requirements have been met.
Project Managers	Project managers will use this document to monitor the development progress and ensure that development aligns with the client's goals
Marketing staff	Marketing staff can use the document to gain an end-user's understanding of system which will help them convey the product's capabilities
Document writers	Document writers will use the document to create user documentation

1.4 Product Scope

The SuperPrice application will be able to compare the prices of products from a range of different super markets. Additionally, the application will allow users to order from those super markets and have it delivered to their address. A shopping cart system, delivery scheduling and payments will be managed by this application.

The software is intended to help customers order their desired products from the cheapest available super markets in a convenient way. If implemented, this software should replace the need for shoppers to manually compare prices across a range of different supermarket websites, saving them time while also facilitating supermarket commerce. This system will allow customers to do all their grocery shopping at the cheapest price through one website.

1.5 References

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

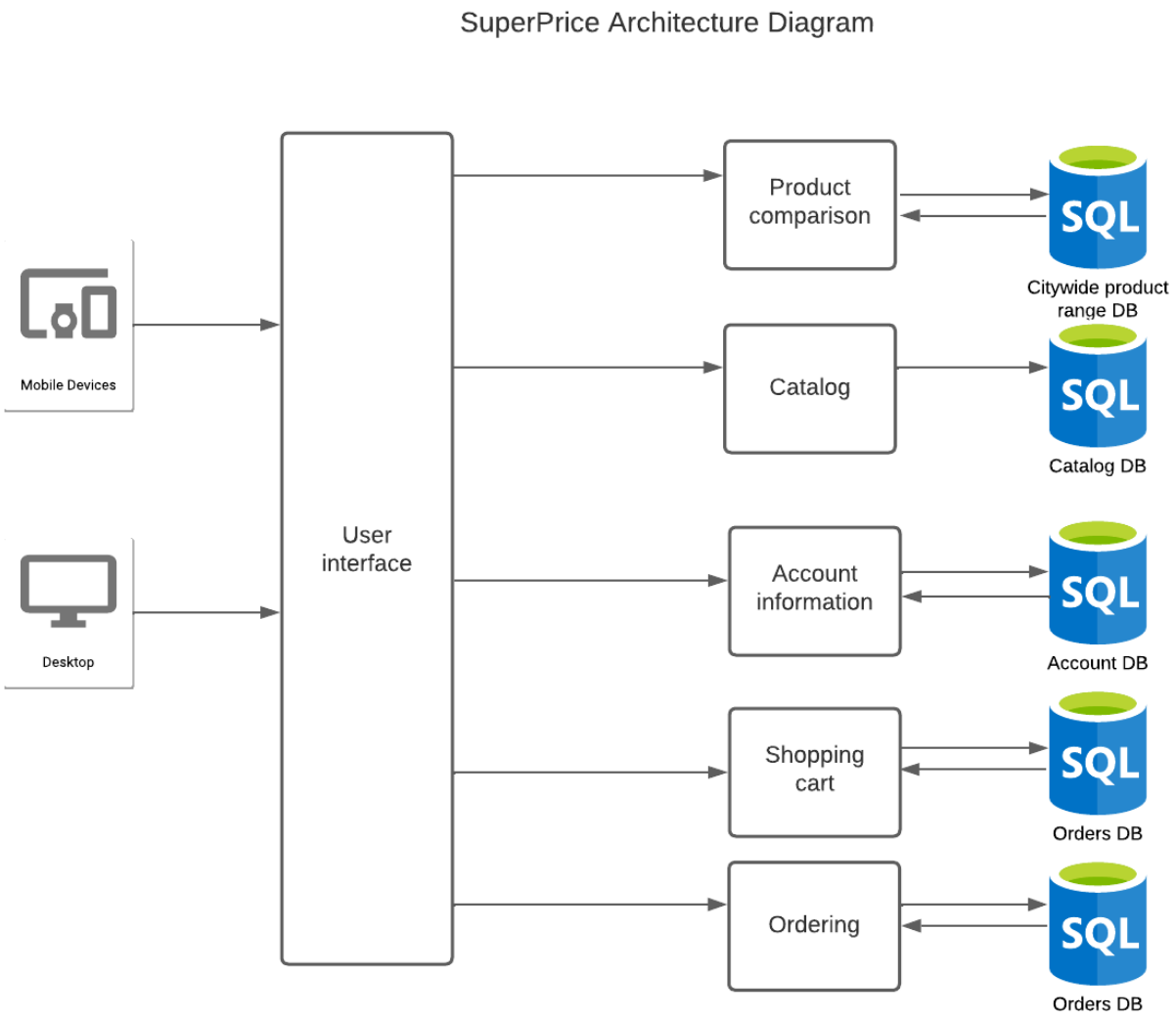
As a part of our payment procedure, we plan on using the Stripe financial infrastructure. More information on this service can be found at: https://stripe.com/au

2. Overall Description

2.1 Product Perspective

This is to be a new, self contained product.

The following diagram shows the architecture with which the product will be developed.



The architecture includes the following components- React for the frontend user interface, SpringBoot for the logic layer, MariaDB for the store layer. Each function is self-contained, which allows for scalability and also reduces the fallout from errors or failures.
Relevant services will communicate with each other via HHTP calls.

Frontend - React.js

React was chosen because

- Have to write less javascript
- Widely used framework
- Simple to use
- Easy to find engineers with skills in React

Bootstrap

Bootstrap was chosen because
- Ease of use compared to writing CSS by hand
- Faster development time because less code needs to be written
- Lower development costs
- Consistency across different platforms

Backend - Spring Boot

Spring Boot was chosen because
- It was recommended by the product owner
- Extremely widely used
- Easy to find engineers with knowledge of Spring Boot

Relational database - H2-database

H2 database was chosen as
- It was easy to use with spring boot
- light weight
- Effective and we expect it'll accelerate development

Payment processing - Stripe

Stripe was chosen because
- Industry standard payment product
- Provides consistent user interface
- Don't have to deal with regulatory requirements for handling credit card data
- Can be integrated with very little code/effort

2.2 Product Functions

The product will perform the following main functions.
- Searchable database of supermarket products
- Price comparison between different sellers of equivalent products
- Ability to choose between alternative products based on price
- Order and delivery

2.3 User Classes and Characteristics

Casual shopper

The casual shopper uses the SuperPrice application to check the price of availability of a product in one or more supermarkets. They might otherwise check directly on the website of a nearby supermarket, but it's more convenient to have one site that aggregates price information from multiple supermarkets.

The casual shopper is likely to use the application only occasionally, perhaps once per month. They are likely to use the product search feature to search for products they are interested in. They might have any level of technical expertise.

To best serve the casual shopper, the application should make specific products as easy to find as possible using the search feature. The search should accurately identify products relevant to the user's query and be as fast as possible, both in the UI and in the processing of the search.

Since the casual shopper is only an infrequent user, they are a less important user class to serve, but maximising convenience for the search function will create a much better experience for all users.

Bargain hunter

The bargain hunter uses the SuperPrice application to save as much money as possible on groceries. If they didn't use SuperPrice, they would make multiple search queries on the websites of multiple supermarkets to try and find the lowest prices. Using SuperPrice would instead allow them to make one search on one website to find the lowest price.

The bargain hunter is likely to use the application very frequently, likely multiple times per week, in order to find deals and specials that they can save money on. They are likely to use the product search and browse by category, price comparison and alternative products features. They are likely to have a higher level of technical expertise than other users, since they would be advanced users of other shopping websites.

To best serve the bargain hunter, the application needs to identify all possible cost saving options. The alternative products feature needs to identify all possible alternative products, not display unsuitable alternative products, and present the options in a clear way with price and other information prominently displayed.

The bargain hunter is the most important class of user to serve. They are likely to be the most frequent user of the application and earliest adopter. Therefore, serving this user class well is key to the success of the product.

Working Parent

The Working Parent uses SuperPrice to order their weekly groceries and book deliveries. They are busy with work or family responsibilities and they don't want to have to visit the supermarket in person. If they didn't use SuperPrice, they would book a delivery directly on a supermarket website, or use apps like UberEats and Doordash to have their groceries delivered. SuperPrice can offer a more convenient experience because it is tailored for grocery shopping unlike UberEats and Doordash, and it allows users to order from multiple different supermarkets on the same website, unlike supermarkets' own websites.

The Working Parent cares about saving time more than saving money. Convenience is key for this user class. The Working Parent includes not just parents, but also businesspeople, who might not have time to buy groceries, and people with disabilities who might not be able to go to supermarkets.

The Working Parent is likely to use the application moderately frequently, perhaps once per week, in order to purchase their regular groceries. They are likely to use the product search, browse by category and order and delivery features. They are likely to have a moderate level of technical expertise.

To best serve the Working Parent, the search function and browse by category features should be as fast as possible, allowing the user to place their grocery order quickly so they can get back to their busy schedules. The order and delivery should also be fast and convenient.

The Working Parent is the second most important user class to serve. They are likely to make up the bulk of users who are not bargain hunters. They are also much less price conscious than bargain hunters, which means that they could potentially be advertised or upsold to, allowing opportunities for monetisation.

2.4 Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

2.5 Design and Implementation Constraints

The design is constrained to be in a web application format due to the client specifications. It's also constrained by elicited functional requirements such as having a search feature, and categorisation features. Due to unfamiliarity working with each other's code, the development team together, may face inconsistency in implementation and design. There may be a reduction of complexity so as to better improve integration and understandability of code and design features amongst the team.

Accessibility of external data is a concerning possible implementation constraint. There appears to be APIs that interact with Coles and Woolworths data, however, there are possible issues with being able to access publicly. Undocumented private APIs may be required, or a workaround solution such as manually entering some data, or web scraping could be used.

Timing is a major implementation constraint. The client requirements specify the project must work end to end at the end by the end of the first sprint. The development team must prioritize work based on client specifications, implementing all main functional features within a three week period. In a following three week sprint there will be improvements made to the quality of the code.

2.6 User Documentation

As the project will be implemented as a website there will not be user documentation provided. The nature of web applications require that the user is able to easily understand and operate the provided functionality. This is facilitated by intuitive user interface and user experience design.

2.7 Assumptions and Dependencies

It is assumed that there will be some appropriate method to receive product data from major supermarkets, whether it be through public or private APIs, or through other workarounds such as web scraping or manual entry.

It is assumed that there is an email service that can be easily integrated into the system if necessary.

It is assumed that the functionality of the checkout will be created by the team and stripe will be used only for the processing of payments. It is assumed checkout behavior can be coded in the front end, and the database does not need to include entities such as ‘cart item’ or ‘session’.

The project is dependent on many external systems. MySQL will be used to implement the database. Apache server will be used to take the website live.

There are several libraries that will be leveraged to create the system for the application to run. SpringBoot will be used as a reliable java framework in order to create stand-alone, secure applications. React will be used to create a complex, interactive and modular user interface. Java Database Connectivity (JDBC), will be implemented to create connections to the database.

Other library dependencies include the java collections library that will be used to implement the search comparison algorithm, and, if required, JSoup HTML Webcrawler may be used for web scraping.

3. External Interface Requirements

3.1 User Interfaces (TBD)

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

3.2 Hardware Interfaces (TBD)

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

3.3 Software Interfaces (TBD)

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

3.4 Communications Interfaces (TBD)

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

4. System Features

ID Conventions

- HOM ← Home page
- PRO ← Product page
- SEA ← Search results
- CHE ← Checkout
- SHO ← Shopping cart
- LOG ← log in (sign in and sign up)

Priority conventions

- Should/would/could are three categories used to describe a feature's priority

Requirement conventions

Description and Priority

Brief description of the feature and the priority of the feature

Stimulus/Response Sequences

A use case for the feature and how an example of the feature responding to user input

Function Requirements

- The specific functional requirements for the feature and the acceptance criteria

4.1 Search for products using keyword (HOM-1)

Description and	Priority: Must

Priority	Users can search for products on the home page using a search box
Stimulus/Response Sequences	1. User enters name of a product into the search box from the home page 2. The system displays a list of matching products is displayed
Functional Requirements	**HOM-1.1:** A list of matching products is shown Given I've entered the name of a product into the search box When I hit enter Then a list of matching products is displayed

4.2 Browse by categories (HOM-2)

Description and Priority	Priority: Must Users can select the category to browse from the home page.
Stimulus/Response Sequences	1. From the home page user selects the browse categories button 2. The system displays a list of categories 3. User clicks the name of a category 4. The system shows a list of all products under that category
Functional Requirements	**HOM-2.1:** UI element on home page to open categories page or menu Given I'm on the home page When I click on the button or other UI element to browse categories Then A categories page or menu opens And I see a list of all categories **HOM-2.2**: Clicking on a category on the categories page opens a product list page for that category Given I have the categories page or menu open When I click on the name of a category Then A product list page opens for that category And I see a list of all products in that category

4.3 prices shown on product page (PRO -1)

Description and Priority	Priority: Must Prices from the different supermarkets are shown on the product page
Stimulus/Response Sequences	1. User Clicks on a product 2. System shows the product page 3. System shows the price of that product across multiple different super markets
Functional Requirements	**PRO-1.1:** List of all super markets containing that product is shows Given I'm looking for a particular product When I go to the product page for that given product Then I should see a list containing all the supermarkets that sell the given product And the list should be sorted by price, lowest to highest And I should see a price associated with each supermarket **PRO-1.2**: Products that don't exist display N/A instead of a price Given there exists a product that is not available from a supermarket When I'm on the product page for that product Then it should display "N/a" instead of a price

4.4 Prices shown in search results (SEA-1)

Description and Priority	Priority: Could The prices of a product is shown in the search results
Stimulus/Response Sequences	1. User searches for a product 2. System shows that product's price and the corresponding supermarket
Functional Requirements	**SEA-1.1:** Icons in search with price for stocked items Given I'm on the search results page When A product is available at a major supermarket Then The search result should feature in the UI an icon for that supermarket And The price at that supermarket should be shown next to it **SEA-1.2:** No price listed for unstocked items Given I'm on the search results page When A product is not available at a particular major supermarket Then The search result should not show an icon for that supermarket And The search result should not show a price for that item at that supermarket **SEA-1.3:** Lowest price bolded Given I'm on the search results page When A product is available at a major supermarket And: That supermarket offers the cheapest price available Then That price should be highlighted in the UI (such as with bold text)

4.9 Filter search results by category (SEA-4)

Description and Priority	Priority: Should Users can filter their search results by category

Stimulus/Response Sequences	1. User selects to filter search results by category 2. System only shows search results that correspond to the chosen category
Functional Requirements	**CAT 4.1** UI element on search results page to filter by category Given I'm viewing search results When I click a UI element, such as a checkbox menu, to filter by category Then the search results are updated And I see a products within given categories relevant to search query

4.10 Sign in (LOG-1)

Description and Priority	Priority: Could User are able to sign in to the website if they have a valid account
Stimulus/Response Sequences	1. User puts their login information 2. System grants them access to their account
Functional Requirements	**LOG-1.1:** Successful login upon entering correct details Given I'm on the login page And I have filled my username and password details correctly into the form When I click the sign in button Then I will be redirected to the home page And I will infer confirmation that I am signed in **LOG-1.2:** Rejected login upon entering false details Given I'm on the login page And I have filled my username and password details incorrectly into the form When I click the sign in button Then the login page will be refreshed And I will not be logged in And an error message will provide visual confirmation of failure

4.11 Sign up (LOG-2)

Description and Priority	Priority: Could Users are able to sign up to create an account
Stimulus/Response Sequences	1. User puts in their information 2. If valid, system creates an account using the user's information
Functional Requirements	**LOG-2.1:** Successful login upon entering correct details Given I'm on the sign up page And I have entered my username, password, password again details correctly into the form When I click the sign in button Then I will be redirected to the home page And I will infer confirmation that I now have an account **LOG-2.2:** Rejected login upon both passwords being different Given I'm on the sign up page And I have entered my username correctly, And I have entered my password and password again details incorrectly into the form When I click the sign up button Then the login page will be refreshed And I will not be logged in And an error message will provide visual confirmation of failure **LOG-2.3:** Rejected login upon entering existing username Given I'm on the sign up page And I have entered my username, And my username already exists for another user And I have entered my password and password again details correctly into the form When I click the sign up button Then the login page will be refreshed And I will not be logged in And an error message will provide visual confirmation of failure

4.12 Sort results by price (SEA-2)

Description and Priority	Priority: should Users are able to filter their search results by price
Stimulus/Response Sequences	1. User searches for a product 2. System shows them a list of corresponding products 3. User selects to sort prices from low to high 4. System organizes list of products from low to high
Functional Requirements	**SEA-2.1**: UI element on search page to sorting options Given I'm on the search results page And I click the sort options drop down box When I click to sort by price low to high Then the search results page will refresh And I see items in order of the price property, low to high

4.13 Filter search results using minimum and maximum price filters (SEA-3)

Description and Priority	Priority: Could Users are able to filter their search results by price by adding minimum and maximum prices.
Stimulus/Response Sequence	1. User searches for a product 2. System shows them a list of corresponding products 3. User inputs a minimum price of $5 and maximum price of $10 4. System adjusts the list of products shown to be only products that correspond to that list
Functional Requirements	**SEA-3.1:** Minimum and maximum inputs on search page Given I'm on the search results page When I input a price minimum And when I input a price maximum Then the search results page will refresh And I'll see items within this price range

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- *Each page must load within 2 seconds.*
- *A search result will be available within 2 seconds of the user's request.*
- *The database will be able to store all products required to fulfill the requirements of the program without affecting performance.*

5.2 Security Requirements

- *The system must be secured from unauthorized access.*
- *A user will be required to create a "strong" password when creating their login.*
- *Passwords will be hashed before being stored in the database.*
- *User information will never be shared to third-parties.*

5.3 Software Quality Attributes

Usability:
- *The user will be able to navigate to their desired page of the program with an 80% success rate.*
- *If a user makes an error, they will be notified of the error and how to proceed.*
- *The overall layout of the program will be intuitive and attractive.*

Localisation:
- *The time zone of the program will be suited to the time zone of the logged in user.*
- *The currency of the program will be suited to the local currency of the logged in user.*
- *Measurements of products will be suited to the local system used by the logged in user. (Imperial vs Metric)*
- *The language of the program will be suited to the language specified by the logged in user.*

Maintainability:
- *Products will be able to be added and removed from the database whenever needed.*
- *Individual sections of the program will be testable and modifiable without affecting other sections of the program.*

Compatibility:
- *The program will operate on all web browsers.*

6. Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

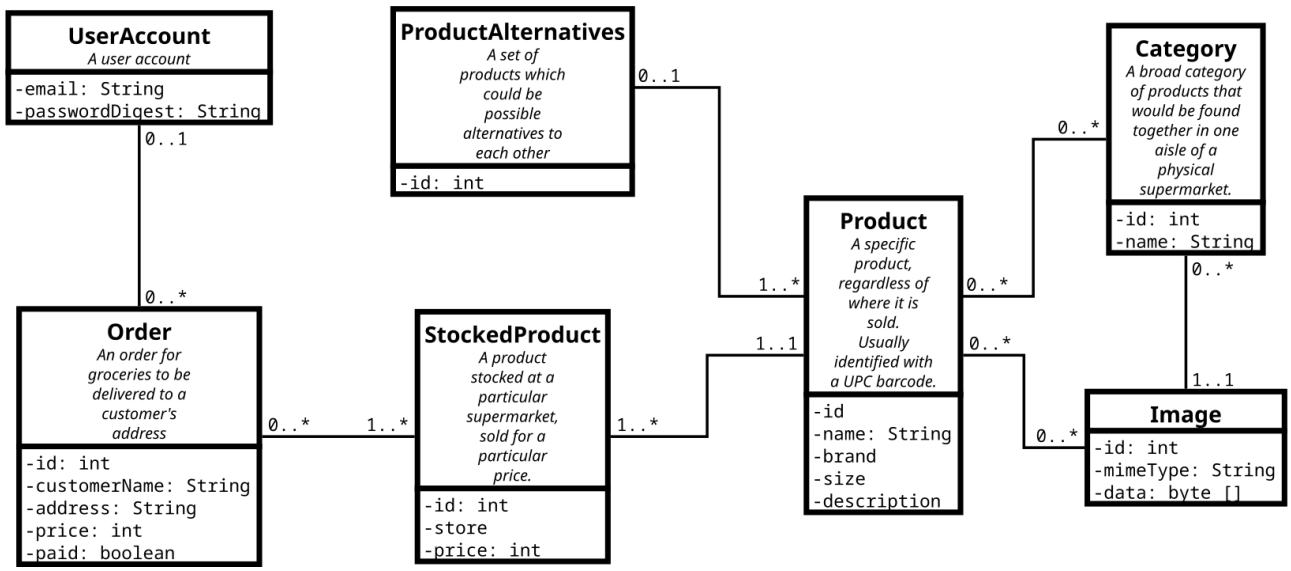
<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

SRS: Software Requirements Specification
UI: User Interface

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

B.1 - Data Model Diagram



B.2 - UI Wireframes

Please see attached file
https://github.com/cosc2299-sept-2023/team-project-group-p01-08/blob/main/docs/Appendix%20B.2%20-%20Wireframes.pdf

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>