UNSW SYDNEY | Australia's Global University

REQUIREMENTS AND DESIGN WORKSHOP 20T1

SENG2021

DELIVERABLE 3

priceGauge

Written By:

Kwok, Henry	z5256258	z5256258@student.unsw.edu.au

Table of Content

Introduction

PriceGauge is a web application designed for everyone but primarily for students to access price information and statistics from australian local supermarkets in a rapid and efficient manner. Users can compare prices of different products from different supermarkets to see which supermarkets have cheaper products to save on their grocery expenses. This web-based application uses an extensive range of public APIs to access price information from a variety of supermarkets.

The purpose of this document is to track the detail and information gained during the design process. The information communicated by the client will be significant in guiding the development of the architecture and back-end system for the development team during the next phase of the project.

The intended target audience of PriceGauge are students in general, however it can be readily available for everyone who wants to compare prices of products from supermarkets. The first 2 supermarkets we have chosen to implement into our web application are Coles and Woolworths.We have also considered other platforms to expand for the future but we decided on these 2 initial supermarkets due to their popularity in Australian shopping centers and the availability of their public APIs.

Part 1: Software Architecture

Software Components - External APIs

PriceGauge will be using retrieving data from external APIs to display product information in a graphical format for users to compare their prices. These functions include searching for a product in a particular supermarket API and the API returning product information.

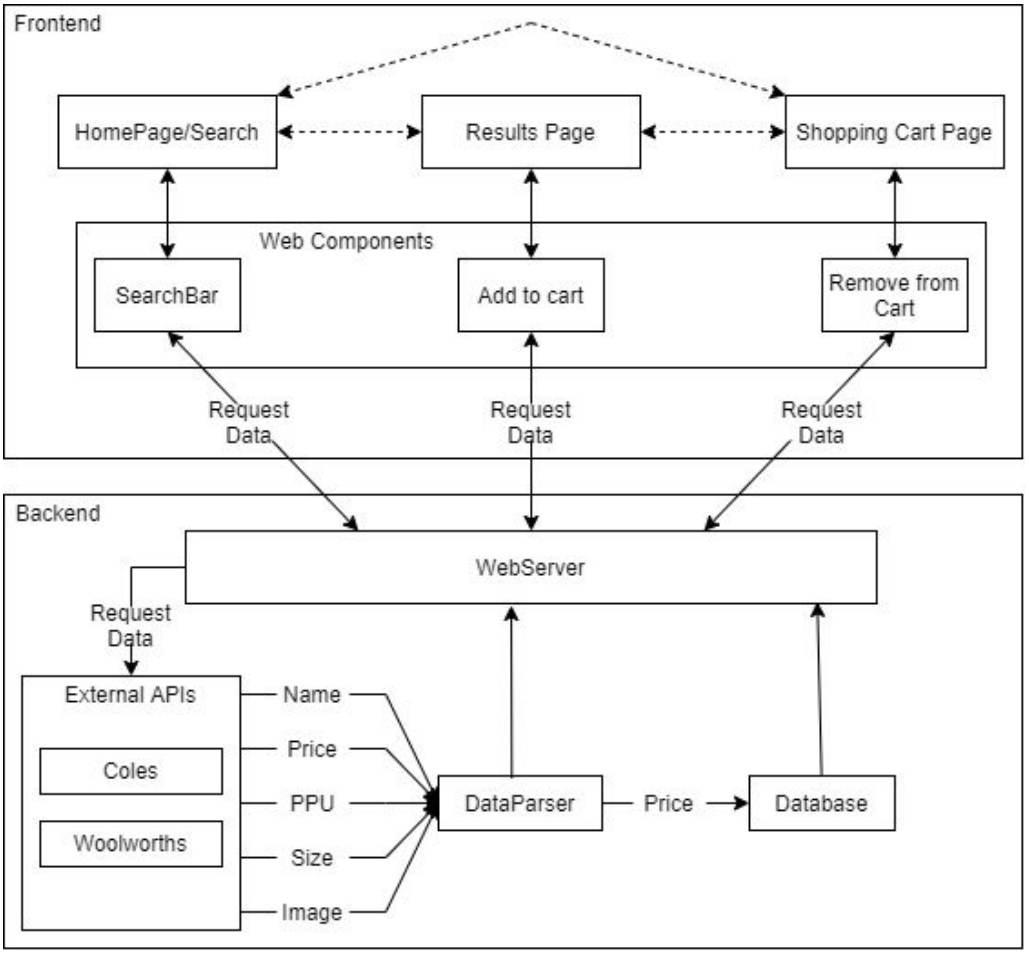
Coles and Woolworths API

Both APIs are free and they provide detailed product information. The only thing we need to do is to generate an API key for our web application and it is sufficient for our users to provide the product they want to inquire about, then the API returns product information for products found in their stock relating to the specific search terms. The APIs return a JSON list containing dictionaries for each product they have found. These keys and values are shown below.

API Data: Keys - Value

- Name - Product name
- Price - Price of the product in australian dollars
- Size - The size of the product
- Price_per_unit - Price of the product divided by the size
- Image_url - Url of the image of the product
- Link - Hyperlink back to the product's page from the original supermarket's website

Software Components - Webstack



In-house developed

- Frontend
    - Web components
        - Navigation bar
        - Product Search bar
        - Results List
            - Coles
            - Woolworths
        - Add to cart
        - Remove from cart
        - Total price
    - Templates (HTML)
        - Home
        - About
        - Results
        - Shopping Cart
    - Styling (CSS)
- Backend
    - Database (JSON)
    - Webserver (Python - Flask)

Third-party

- APIs
    - Coles
    - Woolworths
- Web Browser
- Cloud services for hosting
    - Virtual Desktop
    - DNS
- Website Domain

Software Components - Languages

Below is the language and frameworks we have decided to use on this project.

- Python - Backend
    - Flask - Webserver (backend)
    - JSON - Database (backend)
- CSS - Frontend styling
- HTML - Frontend templating
- Javascript - Frontend interactions

Choice of browsers and operating systems

Since I have a windows PC, I initially tested the website on windows 10 (operating system) and Chrome but every operating system and all the web browsers supporting js should work fine.

The backend and frontend should run on windows PC as well and since the backend is mainly written in python, it should work on linux and macOS as well as long as the same dependencies and libraries are installed.

Key benefits of architectural choices

- I am proficient in Python, HTML and CSS
- The advantages of python and its frameworks
    - Python: An interpreted high level coding language
        - Plenty of third party libraries
        - Easy to use
        - Clean and precise, makes debugging easier
    - Flask: Lightweight python web application framework
        - Quick and easy to set up
        - Minimalist and simple to use
    - Jsonify: Framework use to convert python objects to JSON format
        - Easy conversion of python objects (lists, dictionaries) into JSON format
        - Stores information in BSON (binary JSON), therefore can store structures such as arrays and dictionaries
        - No need to setup database schemas
- Whole backend uses python, therefore there is no need to use additional languages.
- Strong support of frameworks listed above online.
- CSS is beneficial for the website due to its adaptable nature, fast loading time and compatibility with other frontend languages.
- HTML is supported by all browsers and simple to edit.

According to the list from above, a lot of the architectural choices were dependent on ease of use and simplicity. This was because of the time constraints on this project, hence I needed to use more productive and simpler coding languages and frameworks to ensure my project is completed on time.

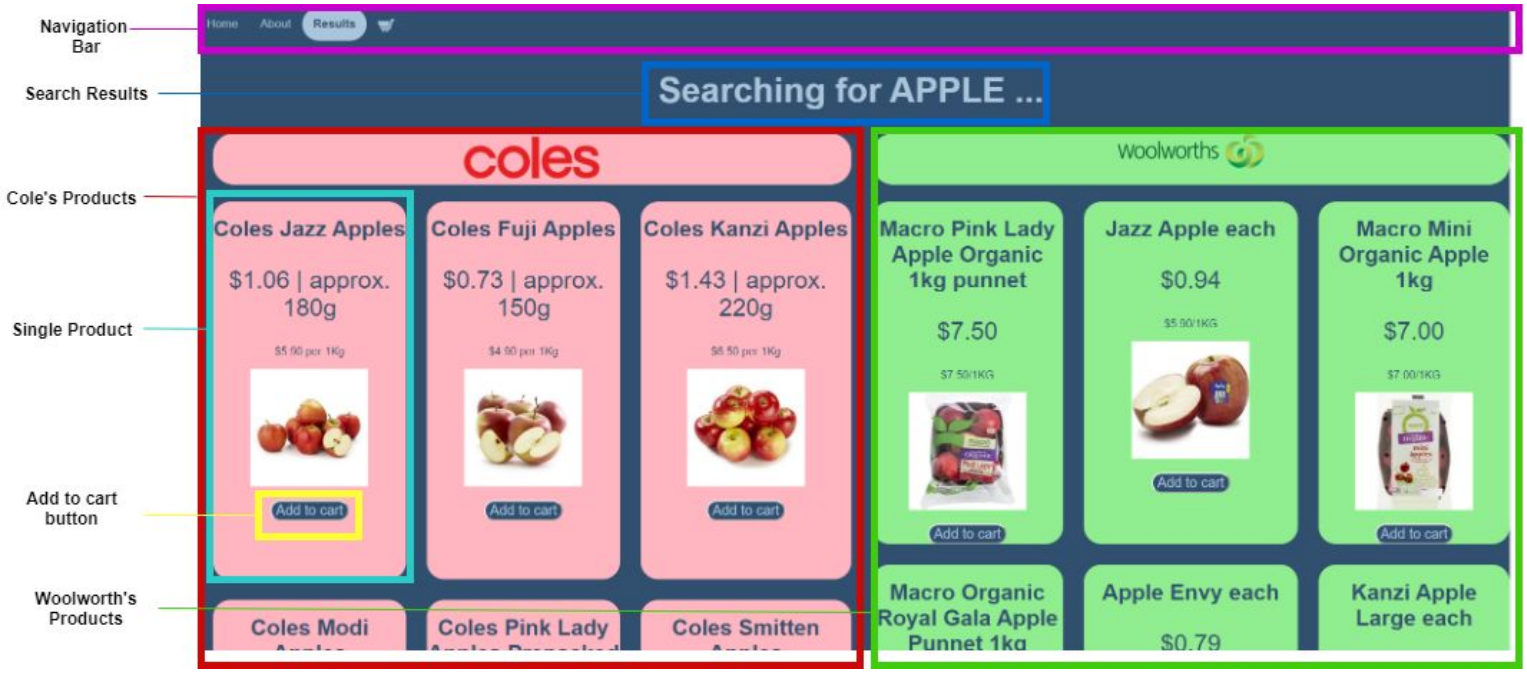
Part 2: System Features

Home page

The homepage consists of the 3 main elements

- Navigation bar - Used to navigate around the different pages of the website. Each button has an animation when hovered on and the current page is shaded light blue to indicate to the user
- Web application logo - Logo of the website and tells user what the website does
- Search Bar - Used to search for products that the user wants to inquire about

Results Page



The results page displays the results from the search from the homepage, it consists of 4 main elements.

- Search Results - Showed what the user searched for
- Coles/Woolworths products - Split of the page in half and color coded to show which products are from which supermarket
- Single product - Each product is shown in a box which enlarges when hovered on. The box contains product information such as the product's name, price, size, price per unit and image. The box also is hyperlinked to the products exact page from the supermarket's website.
- Add to cart - Button which allows users to store this product onto the shopping cart for future record.

Shopping Cart Page

The shopping cart page shows the items added from the results page, it consists of 4 elements

- Coles/Woolworths products - Split of the page in half and color coded to show which products are from which supermarket
- Single product - Each product is shown in a box which enlarges when hovered on. The box contains product information such as the product's name, price, size and image.
- Remove from cart - Button which allows users to remove this product from the shopping cart.

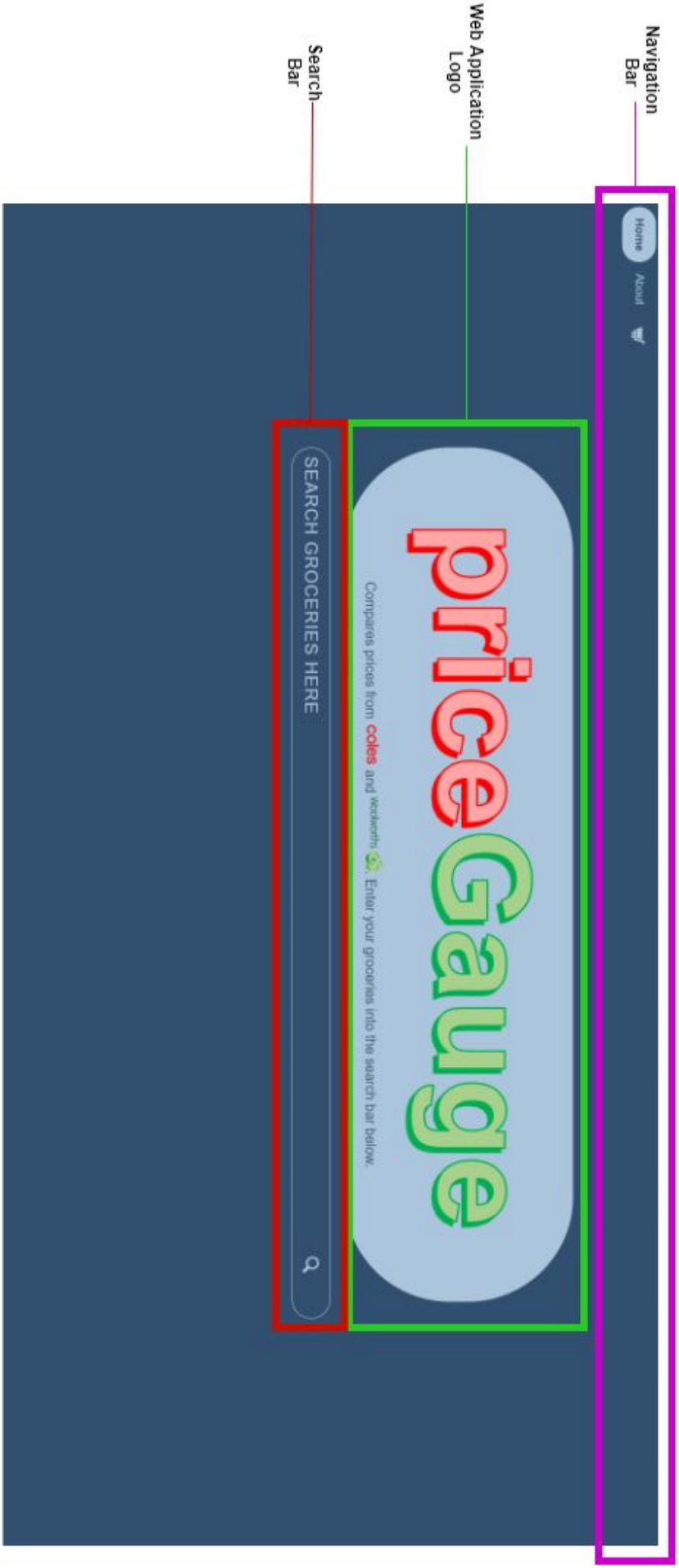
The diagrams above will be attached in the appendix below for better resolution.

Part 3: Problems addressed

One problem was caused by the external APIs. Since the APIs requested a lot of data, the response from them took a long time (around a minute for a single search request). Therefore I have decided to implement a caching system on the backend database. Everytime a product is first searched, it will be using the external APIs. Their product data will be stored locally on the database. The next time a user accesses the same product information, the external API's will not be requested, instead the product data will be requested from the database, hence making the response faster. However, every week, the local data will be deleted to ensure the results are correct, hence new results will be needed to be obtained from the external APIs.

Another problem was the shopping cart buttons (add/remove). Since wtForms (python library for scripting interactivity on the website) only responds to data from form html tags (e.g. the home-page product search bar), the buttons for the shopping cart will not work with this library. Therefore, I needed to use javascript instead to code the buttons to send the json dictionary of the specific product information back to the backend to save to the shopping cart list. As I had not used javascript before, I had some difficulty in implementing this function.

Appendix


Navigation Bar
Web Application Logo
Search Bar
Home
About
priceGauge
Compares prices from coles and Woolworths. Enter your groceries into the search bar below.
SEARCH GROCERIES HERE


Navigation Bar
Search Results
Cole's Products
Single Product
Add to cart button
Woolworth's Products
Home
About
Results
Searching for APPLE ...
coles
Coles Jazz Apples
$1.06 | approx. 180g
$5.90 per 1Kg
Add to cart
Coles Fuji Apples
$0.73 | approx. 150g
$4.90 per 1Kg
Add to cart
Coles Kanzi Apples
$1.43 | approx. 220g
$6.50 per 1Kg
Add to cart
Coles Modi
Coles Pink Lady
Coles Smitten
Woolworths
Macro Pink Lady Apple Organic 1kg punnet
$7.50
$7.50/1KG
Add to cart
Jazz Apple each
$0.94
$5.90/1KG
Add to cart
Macro Mini Organic Apple 1kg
$7.00
$7.00/1KG
Add to cart
Macro Organic Royal Gala Apple Punnet 1kg
Apple Envy each
$0.79
Kanzi Apple Large each


Navigation Bar
Home
About
Current shopping cart:
Cole's Products
coles
Single Product
Coles Kanzi Apples
$1.43 | approx. 220g |
Woolworth's Products
Remove from cart button
Remove from cart
woolworths
Jazz Apple each
$0.94
Remove from cart
Total: $2.37