2025 Pick ‘n Price

Client: COS221 Staff

Team: Primary Keys

Functional Requirement Specification

Member	Student Number
Adriano Jorge	u24634434
Dewald Colesky	u23536030
Inge Keyser	u23544563
Megan Lai	u23608821
Connor Bell	u24569608
Ferdinand Johannes Nel	u24594475
Zoë Joubert	u05084360

Table of Contents

Introduction

This document outlines the functional requirements for Pick ‘n Price, a web application to compare prices of similar products across the web. There are a few components, namely, the Web UI, API, and Database. The Web UI provides a smooth experience to all users and administrators. The API allows for database operations. The Database stores all required information.

Web UI

1. **Landing Page**
    1.1. The user should be able to register an account
    1.2. The user should be able to login if they already have an account
2. **Admin Panel**
    2.1. Admins can edit products
    2.2. Admins can add products
    2.3. Admins can delete products
3. **Product Listings**
    3.1. The user should be able to view all products
    3.2. The user should be able to filter products:
        3.2.1. Based on category
        3.2.2. Based on name
        3.2.3. Based on price
        3.2.4. Based on retailer
    3.3. The user should be able to expand product information:
        3.3.1. To see ratings
        3.3.2. To see average rating
        3.3.3. To see retailers
        3.3.4. To visit retailer website
        3.3.5. To watchlist a product
4. **Review Products**
    4.1. The user should be able to review a product
    4.2. The user should be able to update their rating of a product
5. **Watchlist**
    5.1. Add products to watchlist
    5.2. Remove products from watchlist
    5.3. Receive updates

API

6. User management

6.1. The user should be able to register an account using a valid reCAPTCHA token

6.2. The user should be able to log in using a valid reCAPTCHA token, and their user_type, API key and name should be returned by the API

7. Admin functionality

7.1. Admins should be able to view:

7.1.1. All users and their account details (excluding personal information like passwords)

7.1.2. All products and their related attributes

7.1.3. All retailers and their associated details

7.1.4. Recent ratings

7.2. Admins should be able to add:

7.2.1. Users (customers or staff)

7.2.2. Products

7.2.3. A price and retailer associated with a product

7.2.4. Retailers

7.3. Admins should be able to edit:

7.3.1. Users (including their user_type and other information)

7.3.2. Products

7.3.3. Prices and retailers associated with a product

7.3.4. Retailers

7.4. Admins should be able to delete:

7.4.1. A users and their associated information

7.4.2. A product and all information associated with the product

7.4.3. A retailer and all information associated with the retailer

7.4.4. A rating

8. Customer functionality

8.1. Customers should be able to view:

8.1.1. All unique categories of products

8.1.2. All products

8.1.2.1. Filtered by category, name and/or minimum average rating

8.1.2.2. Sorted by cheapest price, name or average rating

8.1.3. Detailed information about a product (including retailers, prices and ratings)

8.1.4. Products in their watchlist

8.1.5. Their product ratings (including product details and retailer details)

8.1.6. Their account details

8.1.7. General statistics about ratings

8.2. Customers should be able to add:

8.2.1. Products to their watchlist

8.2.2. One rating per product

8.3. Customers should be able to edit:

8.3.1. Their product ratings

8.3.2. Their account details

8.4. Customers should be able to delete:
    8.4.1. A product from their watchlist
    8.4.2. A product rating

9. Security functionality

9.1. All endpoints, except for login and register, should require the current user's API key
9.2. Register and login endpoints should require and authenticate a reCAPTCHA token
9.3. Rate limiting should be enforced per IP per endpoint to prevent API abuse
9.4. All inputs should be validated and protected from SQL injection

10. General functionality

10.1. All requests to the API should be made using POST with a JSON object as the payload
10.2. All responses from the API should be in JSON format
10.3. Successful responses should include a status, timestamp, message and data
10.4. Error responses should include a status, timestamp and error message
10.5. All responses should use correct and appropriate HTTP status codes

Database

11. Storage

11.1. The database should store all required information

11.1.1. User information:

11.1.1.1. Customer information

11.1.1.2. Admin information

11.1.1.3. Retailer information

11.1.2. Product information

11.1.3. Rating information

11.1.4. Customer watchlists

12. Access

12.1. Only authorized users should have access to the database