Bilkent University

Department of Computer Science

Senior Design Project

Project Name: PriceWise
Detailed Design Report

Group: T2323

Tuğberk Dikmen, 21802480
Deniz Hayri Özay, 21803632
Mehmet Ali Öztürk, 21703425
Mete Arıkan, 21902316
Furkan Yıldırım, 21902514

Supervisor: Shervin Rahimzadeh Arashloo
Jury Members: Atakan Erdem, Mert Bıçakçı

March 15, 2024
This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

1. Introduction

The Pricewise system serves as a solution to address the challenges caused by fluctuating prices of products in the Turkish markets. With the increasing inflation and frequent price changes, consumers face difficulties in determining the most cost-effective options for their purchases. The purpose of the Pricewise system is to empower users with the ability to compare prices across different markets, facilitating informed purchasing decisions. By utilizing advanced features such as machine learning-based recommendations and real-time price tracking, Pricewise aims to provide users with shopping suggestions and opportunities to save money.

1.1 Purpose of the system

The primary goal of the Pricewise system is to offer users a comprehensive platform for price comparison and optimization of their shopping lists. With the data from various market chains and developing numerous algorithms, the system aims to deliver accurate and timely information on product prices and availability. Additionally, the system helps users to enhance convenience and efficiency by enabling the creation and management of shopping lists, personalized recommendations and notifications for discounts and special offers.

1.2 Design goals

The design goals of the PriceWise system can be considered both functional and non-functional requirements which aim to offer optimized user experience.

1.2.1 Functional Goals:

- Facilitate price comparison among multiple markets and different brands.
- Provide users with personalized shopping recommendations based on their preferences and purchasing history.
- Ensure data privacy and security through authentication and encryption.
- Offer a user-friendly interface and efficient list management functionalities.

- Enable real-time updates on product prices, availability and promotions to enhance user decision-making.
- Support scalability and adaptability to accommodate future growth and evolving user needs.

1.2.2 Non-Functional Goals:

Usability:

- Provide an user-friendly interface to increase the quality of user experience.
- Implement efficient methods for adding and managing items in shopping lists and offer various optimized shopping lists.

Reliability:

- Maintain accuracy and accurate information on product prices and sales.
- Regularly update prices through automated algorithms while allowing manual updates by admins.
- Ensure high availability of the application except during scheduled maintenance periods.

Performance:

- Achieve minimal response times under a specified time to ensure responsiveness.
- Handle high request traffic during peak periods, such as holidays and sales events.

Supportability:

- Support different versions of Android and iOS operating systems to maximize accessibility.

Scalability:

- Design the system to handle increasing user traffic and data volume over time.

Security:

- Implement robust measures to secure user information, including usernames, emails, and passwords, against unauthorized access.

1.3 Definitions, acronyms, and abbreviations

UI: User Interface. This is the point of human-computer interaction and communication in a device. In the PriceWise application it includes the screen of a smartphone.

API: Application Programming Interface. This is a way for two or more computer programs or components to communicate with each other.

ML: Machine Learning. This can be defined as the capability of the machine to imitate intelligent human behavior.

DB: Database. A structured set of data held in a computer, especially one that is accessible in various ways.

Android: Android Operating System. Android OS is a linux-based mobile operating system that primarily runs on smartphones.

iOS: iPhone Operating System. An operating system used for mobile devices manufactured by Apple Inc.

1.4 Overview

The Pricewise system will be designed as a mobile application designed to reform the way users approach shopping comparisons and decision making. With the power of technology, Pricewise aims to empower consumers with valuable information and practical tools to enhance their shopping experience and maximize their savings. From real-time price tracking to personalized recommendations, the system offers a variety of features to increase user satisfaction and their money.

2. Current software architecture

In this section, the functionalities of the existing systems and the areas where they lack according to our observations will be discussed briefly. At the end of this section, what we want to improve in the marketplace will be explained by describing the unique attributes of PriceWise and its differences between the current systems.

2.1. Similar Websites

2.1.1. Epey.com

Epey.com functions as a platform where users can search for products and compare their prices across various stores [1]. However, Epey.com offers only technological gadgets, kitchenware, and other durable items like shampoo, toothpaste, and olive oil. Notably, it lacks a selection of everyday grocery items such as chocolate, milk, or soda. It also does not provide a feature enabling users to create shopping lists nor have any built-in product recommendation system.

2.1.2. Cimri.com

Just like Epey.com, Cimri.com also offers price comparisons for identical products [2]. Moreover, it features a section named "cimri markette," enabling users to browse grocery items and construct shopping lists. However, if a product is unavailable at a store, Cimri.com does not suggest any alternative product. The website also has no recommendation system to split the grocery list among multiple stores. Therefore, the website restricts users to shopping from a single store at once, which limits its ability to deliver the most budget-friendly solution.

2.2. PriceWise

PriceWise aims to gather the functionalities of Epey.com and Cimri.com that are mentioned in the previous subsection and build upon them with its additional abilities that have no counterparts in the current marketplace. Unlike other platforms such as Cimri, PriceWise’s shopping list features will be dynamic as it will provide extra suggestions to

users to customize their shopping lists further. Firstly, when the users want to compute the total price of their shopping lists, it won't just calculate single-store list alternatives but will show the users all the possible shopping list alterations that also include the ones with multiple stores. For example, let's say there are 2 stores called Store A and Store B in the system. The current architecture creates only separate lists for Store A and Store B. PriceWise will provide both singular store lists and also a split list alternative which consists of both Store A and Store B. The prices vary a lot among different stores even if they sell identical products. Thanks to this, the users will have the opportunity to obtain the cheapest possible list as the algorithm will compare the prices from both stores for each item and choose the minimum one. Additionally, if the users can not find a certain product they want, the system will recommend similar available products based on their preferences with the help of machine learning algorithms.

3. Proposed software architecture

3.1 Overview

In this section, the general architecture of the software and the subsystem decomposition are discussed. In order to explain PriceWise's architecture, firstly, the system is decomposed into different layers and modules. We used Uses Style, Modular Style and Decomposition Style in our UML diagram which are generally accepted software architecture schema styles. Later on, in the Persistent Data Management Style, how the system's database operates with the scraped data is explained in detail. Finally, in the Access Control and Security subsection, the security measurements and access-permission boundaries of the users are discussed. In general, we aim to describe how the different parts of the system function with each other in this section.

3.2 Subsystem decomposition

The Pricewise system is organized into several subsystems, each responsible for distinct functionalities and components. This decomposition enhances modularity, scalability, and maintainability throughout the system.

UI Layer:

The UI layer is built using the Flutter framework which uses its extensive set of widgets and tools for creating visually appealing and interactive user interfaces.

Utilizes Flutter/Dart libraries to enhance UI functionality and improves development processes.

Backend Layer:

Handles backend operations, including user authentication, data storage, and machine learning integration.

Incorporates Firebase Authentication to manage user authentication securely and it will integrate with the ML part for predictive analytics and recommendation functionalities.

Data Layer:

Manages the storage and retrieval of data used by the system.

Leverages Firebase Database as the primary data storage solution, offering real-time data synchronization and scalable data storage capabilities.

Scraping Module:

Performs web scraping operations to gather product information from external sources.

Retrieves relevant data from online marketplaces and stores it in the Firebase DB for further processing and testing.

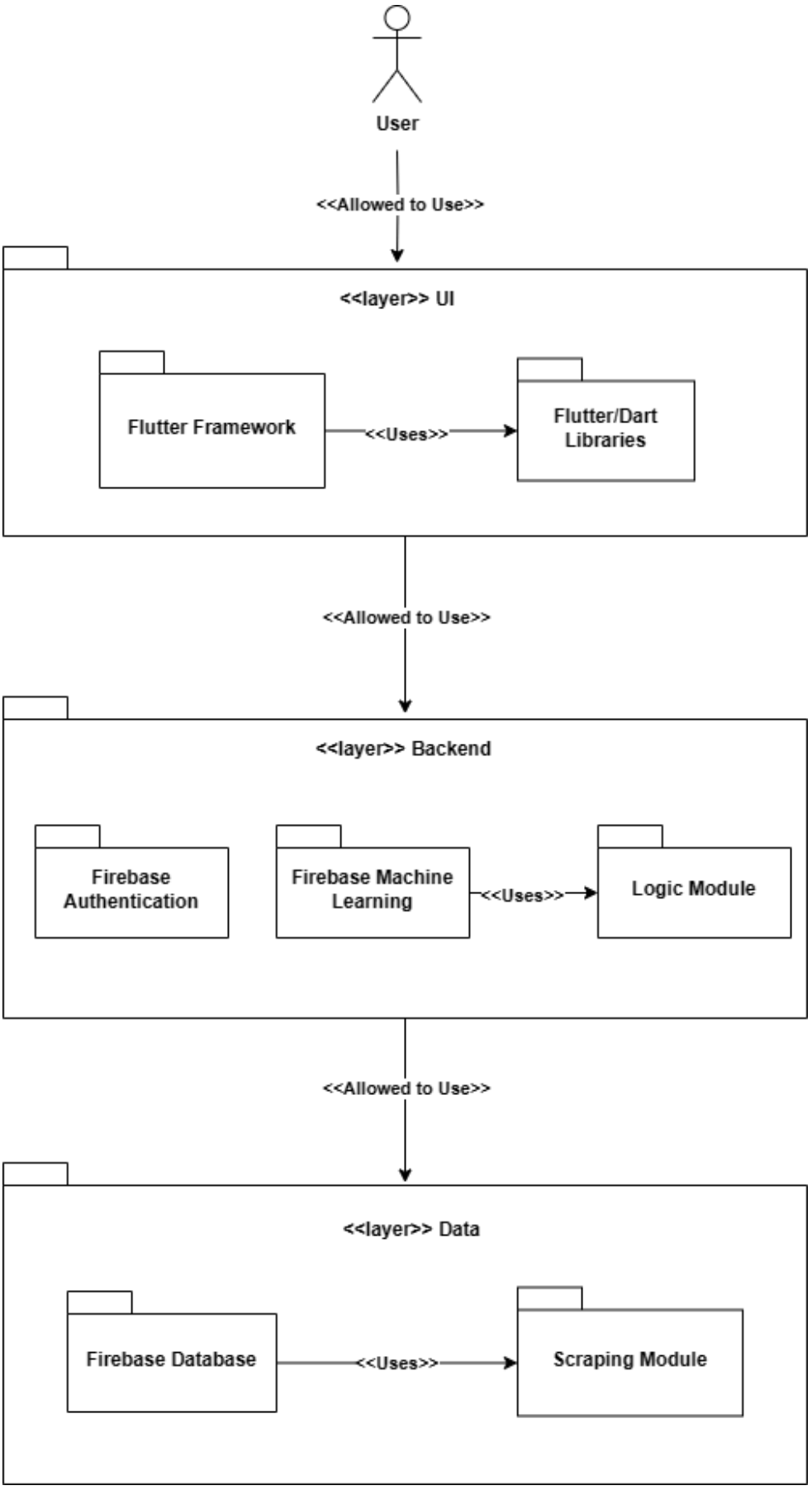

Figure 1. Subsystem Architecture Schema

3.3 Persistent data management

In the persistent data management section of our detailed design report, we highlight the utilization of Firebase Firestore and its authentication services as central components of our data management strategy. Firebase Firestore is a highly scalable NoSQL cloud database that facilitates the storing, syncing, and querying of data for mobile, web, and server development. Its NoSQL format is particularly advantageous for our application as it allows for flexible data structures and the efficient storage of complex, hierarchical data, such as Products, User Shopping Lists, and additional user-specific information. This flexibility is crucial in accommodating the dynamic nature of user data and product inventories, ensuring swift and efficient data retrieval. Moreover, Firestore's real-time data syncing capabilities enhance the user experience by providing immediate feedback and updates (real-time), which is essential for features like real-time shopping lists and inventory management.

Additionally, the integration of Firebase's authentication services streamlines the login and signup processes, securely managing user credentials and personal information. By storing user information within the authentication services, we leverage Firebase's robust security protocols to protect sensitive data while simplifying the authentication flow for users. This separation of concerns not only enhances security but also optimizes the performance by using Firestore for application-specific data like Products and User Shopping Lists. Communication with Firestore and the Authentication services is conducted through Firebase queries, mirroring a REST API's request and response model. This approach enables a structured and efficient way of managing data transactions, allowing for clear and concise data handling operations, including creating, reading, updating, and deleting records (CRUD operations). The combination of Firestore's NoSQL database with Firebase's authentication services presents a cohesive, secure, and scalable solution for managing persistent data, essential for delivering a seamless and responsive user experience.

3.4 Access control and security

Apart from the administrators, there is only one type of user which is a regular user. All of the users will benefit from the same functionalities, and have the same accesses and permissions. The users will be able to search for a product, see the prices, create grocery lists, and get alternative products and list recommendations from the system. Users will not be able

to interact with other users in any way. They won't be able to view or modify other users' shopping lists, nor view other users at all. By doing this, we aim to prevent any incidents that can harm user experience.

The users will authenticate from Firebase which is certified under major privacy and security standards [3]. The information of the products and the machine learning algorithms will also be hosted in the Firebase system.

Before getting any recommendations, users will be asked to allow the system to collect data from their shopping list actions. In order to show the most precise recommendations to users, the system firstly has to be trained with their previous list actions to learn their preferences. The collected data will only be used to improve the recommendation system. Other than this exception, no user data will be used or shared in any way.

4. Subsystem services

The subsystem services of our application are crucial in ensuring a seamless, secure, and personalized user experience. Below, we divided subsystem services into 5 subsystems and their components or services.

4.1. User Interface

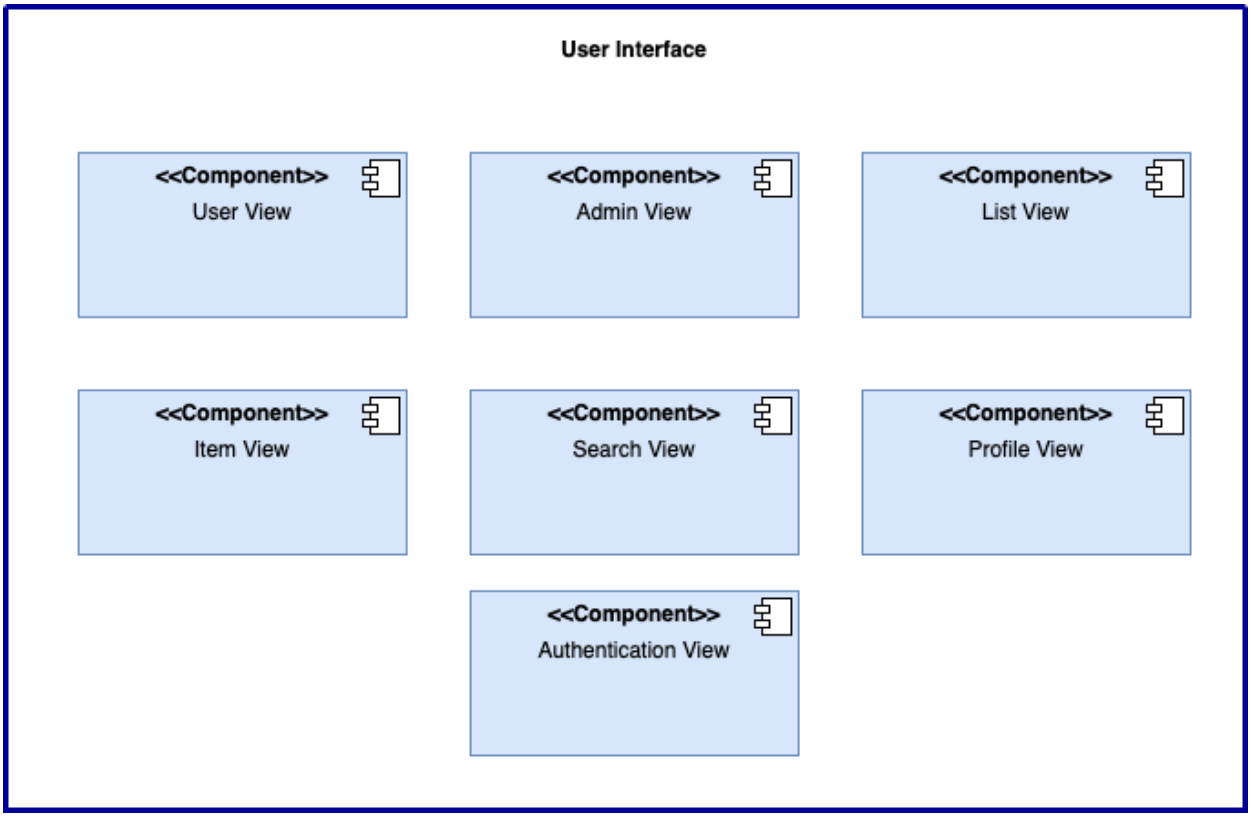

Figure 2. User Interface Subsystem Diagram

The User Interface (UI) subsystem is designed to work for both general users and administrators, ensuring a simple and efficient interaction with the application. It comprises seven key components:

- **User View:** The primary interface for end-users, offering a personalized dashboard that provides access to various features like shopping lists, product searches, and profile management.
- **Admin View:** A specialized interface for administrators to manage application settings, user accounts, product listings, and view analytics.
- **List View:** Allows users to create, modify, view and calculate total price of their shopping lists, enhancing the shopping experience by keeping track of items to purchase.
- **Profile View:** Enables users to manage their personal information, preferences, and view their deleted list history within the application.
- **Authentication View:** A secure portal for user login and registration, supporting both traditional and Google sign-in methods to accommodate user preferences.

- **Search View:** Offers a robust search functionality, allowing users to quickly find products by keywords, categories, or tags.
- **Item View:** Displays detailed information about products, including descriptions, prices, and related items, facilitating informed purchasing decisions.

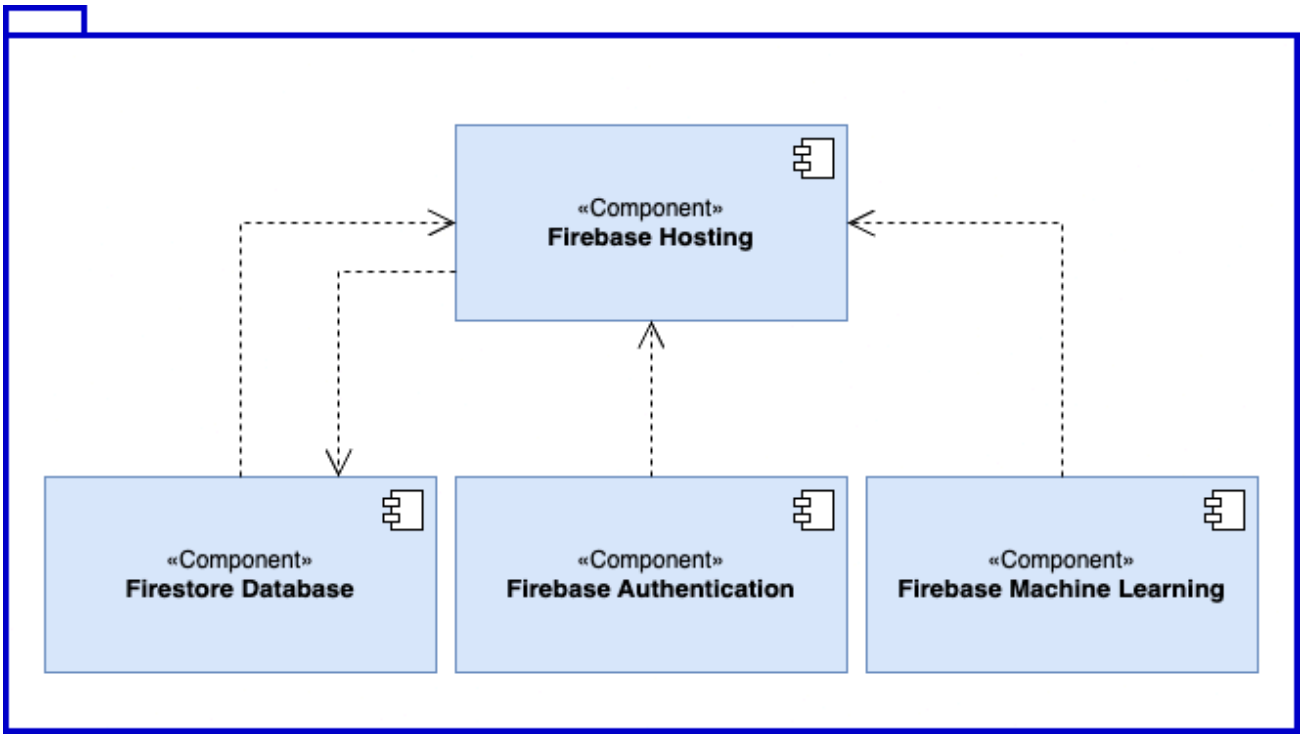

Figure 3. FireBase Components Subsystem Relation Diagram

4.2. Firebase Authentication

Firebase Authentication provides a comprehensive identity solution, handling secure user signup and sign-in processes. It supports a variety of authentication methods, including email/password and Google sign-in, ensuring flexibility and accessibility for users. This subsystem is integral to maintaining the security and integrity of user accounts, safeguarding personal information and preferences.

4.3. Firestore Database Persistent Data Management

Our application's persistent data management is the Firestore Database. This NoSQL database is for handling real-time data storage and synchronization across user interactions. It is organized into three primary collections:

- **Products:** Stores detailed information about the inventory, including descriptions, pricing, categories, product images and similar alternatives of the product.
- **Users:** Contains user profiles, including authentication details, preferences, and interaction histories, to personalize the user experience.
- **Lists:** Manages shopping lists created by users, tracking items they intend to purchase or explore further. In addition to that stores deleted shopping lists for further use.

4.4. Firebase Hosting

Firebase Hosting provides a fast, secure, and reliable way to host our web application globally. It ensures that our application is accessible from anywhere, on any device, with minimal latency. This hosting solution seamlessly integrates with other Firebase services, enabling automatic scaling to handle varying levels of traffic and ensuring a consistent, high-performance user experience.

4.5. Firebase Machine Learning

Integrating Firebase Machine Learning into our application brings a layer of intelligence and personalization. We utilize a Recurrent Neural Network (RNN) model, developed with TensorFlow, to analyze user behavior and preferences. This model is hosted online (Firebase Hosting), allowing it to continuously learn and improve its recommendations for user-based item suggestions. It dynamically adapts to user interactions, enhancing the relevance of product recommendations and personalized content, ultimately increasing the shopping experience.

5. Test Cases

Test ID: T-001 Sign In

Requirements: Ensure that registered users can sign in successfully.

Entry Conditions: Application should be opened, and the current user must not be logged in.

Exit Conditions: Users either successfully sign in or choose to continue as a non-registered user.

Test Steps	Outcome	Pass/Fail
1. Open the application.	Application launches successfully.	

2. Click the sign-in button.	Sign-in page is displayed.	
3. Fill in the required e-mail, password fields.	Required fields are filled with user credentials.	
4. Click the sign-in button.	Sign-in is successful, navigated to the main page.	
5. Attempt to navigate to the search item page without signing in.	Redirected to the sign-in or registration page.	

Test ID: T-002 Sign Up

Requirements: Ensure that non-registered users can register successfully.

Entry Conditions: Application should be opened, and the current user must not be logged in.

Exit Conditions: Users either successfully register or choose to cancel the registration.

Test Steps	Outcome	Pass/Fail
1. Open the application.	Application launches successfully.	
2. Click the Signup button.	Signup page is displayed.	
3. Fill in the required e-mail, full name, password, confirm password fields.	Required fields are filled with new user information.	
4. Click the register button.	Registration is successful, navigated to the main page.	
5. Attempt to register with existing credentials.	Information message displayed, prompting for Sign Up success status.	

Test ID: T-003 Search an Item

Requirements: Ensure that users (registered or non-registered) can search for items successfully.

Entry Conditions: Open the application, be a registered user or non-registered user, and navigate to the search item page.

Exit Conditions: Users can navigate to another page after searching.

Test Steps	Outcome	Pass/Fail
1. Open the application.	Application launches successfully.	
2. Sign in.	User is signed in successfully.	
3. Click on the Search page button.	Search page is displayed.	
4. Enter any item in the search box.	Relevant search results for the item are displayed.	
5. Select an item from the search results.	Item details page is displayed.	
6. Click the back button.	Navigated back to the search results page.	

Test ID: T-004 View Suggested Shopping List

Requirements: Ensure that users can view suggested shopping lists successfully.

Entry Conditions: Open the application.

Exit Conditions: Users click the back button after viewing.

Test Steps	Outcome	Pass/Fail
1. Open the application.	Application launches successfully.	
2. Sign in.	User is signed in successfully.	
3. Click to the desired list.	List’s contained items and their total price is displayed.	
4. Click on a suggested list displayed.	Suggested shopping list details are displayed.	

5. Click the back button.	Navigated back to the previous page.	

Test ID: T-005 Modifying Shopping List

Requirements: Ensure that users can modify a shopping list successfully.

Entry Conditions: Open the application and have at least one shopping list created.

Exit Conditions: Users click the back button after modifying a shopping list.

Test Steps	Outcome	Pass/Fail
1. Open the application.	Application launches successfully.	
2. Sign in.	User is signed in successfully.	
3. Click to the desired list.	List's contained items and their total price is displayed.	
4. Decrease or increase the count of the item desired.	Selected shopping list item count and total price updated.	
5. Click on the Modify Shopping List button.	List modification options are displayed.	
6. Make changes to the shopping list (change the name, icon or color of list).	Changes are saved, and an updated list is displayed.	
7. Click the back button.	Navigated back to the previous page.	

Test ID: T-006 View Sales

Requirements: Ensure that users can view current sales successfully.

Entry Conditions: Open the application.

Exit Conditions: Users click the back button after viewing sales.

Test Steps	Outcome	Pass/Fail
1. Open the application.	Application launches successfully.	

2. Sign in.	User is signed in successfully.	
3. Click on the Sales page button.	Sales page is displayed with current sales listed.	
4. Select any sale item listed.	Selected sale item details are displayed.	
5. Click the back button.	Navigated back to the previous page.	

Test ID: T-007 View Past Shopping Lists

Requirements: Ensure that registered users can view their past shopping lists successfully.

Entry Conditions: Open the application and the user must be logged in.

Exit Conditions: User clicks on another page button after viewing.

Test Steps	Outcome	Pass/Fail
1. Open the application.	Application launches successfully.	
2. Sign in.	User is signed in successfully.	
3. Click on Profile button.	Profile page with user credentials is displayed.	
4. Click on View Past Shopping Lists button.	Page displaying past shopping lists is displayed.	
5. Select and click on any past list displayed.	Details of the selected past shopping list are displayed.	
6. Click on another page button.	Navigated away from the past shopping lists page.	

Test ID: T-008 View Preferences Personalized Shopping Lists

Requirements: Ensure that registered users can personalize and view suggested shopping lists based on preferences.

Entry Conditions: Open the application, should have created a list and the user must be logged in.

Exit Conditions: User is able to modify and view the personalized list.

Test Steps	Outcome	Pass/Fail
1. Open the application.	Application launches successfully.	
2. Sign in.	User is signed in successfully.	
3. Go to the Lists page.	Lists page is displayed.	
4. Select the Suggested List.	Suggested list based on user preferences is displayed.	
5. Change the products according to desires (delete/ add product).	Products and the total price of the list are successfully changed in the list.	
6. Click the Back button.	Navigated back to the Lists page.	

Test ID: T-009 Update Database

Requirements: Ensure that admins can update the database successfully.

Entry Conditions: Admin must open the application.

Exit Conditions: Admin clicks the Back button after updating.

Test Steps	Outcome	Pass/Fail
1. Open the application.	Application launches successfully.	
2. Go to Duty page.	Duty page for admin tasks is displayed.	
3. Click the Update Database button.	Update Database interface is displayed.	
4. Perform the necessary updates.	Updates are successfully applied to the database.	

5. Click the Back button.	Navigated back to the Duty page.	

Test ID: T-010 Send Notification

Requirements: Ensure that admins can send notifications to users successfully.

Entry Conditions: Admin must open the application.

Exit Conditions: Admin clicks the Back button after sending a notification.

Test Steps	Outcome	Pass/Fail
1. Open the application.	Application launches successfully.	
2. Go to Duty page.	Duty page for admin tasks is displayed.	
3. Click the Send Notification button.	Notification sending interface is displayed.	
4. Enter the message and recipient details.	Message and recipient details are correctly entered.	
5. Click the Send button.	Notification is sent successfully.	
6. Click the Back button.	Navigated back to the Duty page.	

6. Consideration of Various Factors in Engineering Design

6.1 Constraints

Technical Constraints:

Data Availability: The availability and quality of data from various sources, including web scraping causes constraints on the design and functionality of the system.

Performance (Real-Time Updates): Updates in the products and pricing information are crucial for maintaining the accuracy and reliability of the platform. To ensure quick updates, optimization techniques will be explored to minimize processing delays and

maximize application speed. This is why application speed directly impacts user experience. With slower response times leading to reduced user satisfaction and engagement.

Platform Compatibility: Compatibility with different platforms and devices such as; mobile devices, must be considered to ensure a seamless user experience across various environments.

Integration Challenges: Integration with external systems such as; Firebase for database management and machine learning frameworks for predictive analysis may present technical challenges that need to be addressed during the design phase.

Time Constraints:

Project Schedule: The project timeline and deadlines for delivering the final product may impose time constraints on the design, development, and testing phases. Adopting an iterative development approach like Agile methodology, can help manage time constraints.

6.2 Standards

Data Standards:

Data Integrity: Adherence to data integrity standards ensures the accuracy and reliability of information processed and stored by the system.

Privacy and Security: Compliance with data privacy regulations, implementation of security practices preserve user data against unauthorized access and misuse.

Software Development Standards:

Coding Practices: Following coding standards, such as code readability, modularity, and reliability enhances the maintainability and extensibility of the application.

Testing Standards: Adoption of testing standards validates the functionality and performance of the system more properly and detects the potential problems in advance in the application.

User Experience Standards:

Accessibility: User experience standards ensure that the application is usable by all individuals and provides a smooth user experience and it helps to optimize for ease of use and efficiency.

6.3 Other Considerations

Public Health and Safety

The PriceWise system ensures that all products adhere to public health and safety regulations. In addition, PriceWise utilizes data from market chains that are officially recognized and regulated by relevant ministers. This ensures that the products listed on the platform meet certain standards of quality, safety, and legality as determined by the authorities. By sourcing data from these established market chains, Pricewise aims to provide users with reliable and trustworthy information for their shopping needs.

Public Welfare

The system aims to enhance public welfare by providing users with tools and information to make informed purchasing decisions, thereby promoting affordability, accessibility, and fairness in the marketplace.

Economic Factors

Economic considerations influence pricing strategies and product recommendations within the system. The platform aims to provide cost-effective solutions for consumers while supporting fair competition and sustainable business practices among markets.

Social Factors

Social factors such as consumer preferences and discounted products will be within the system algorithms. The platform adapts to changing user preferences to adopt relevant and impactful shopping experiences.

7. Teamwork Details

7.1 Contributing and functioning effectively on the team

As Algorteam, we have Discord meetings almost every week where we regularly report to each other. Being online may give the impression that productivity is low, but productive meetings take place thanks to features such as being able to show our work on the computer or meeting at any time and anywhere. Our meetings are always full staff and everyone actively talks about their ideas. Finally, we decide what we need to do until the next meeting date. Even though there were 5 people in the group, we divided our project into three main headings. Everyone has contributed greatly to each other's knowledge in these areas and we are in a much more advanced position than we were at the beginning of the year.

7.2 Helping creating a collaborative and inclusive environment

We divided our project into front-end, back-end and scraping. With the previous experience, our group's total knowledge in front-end and back-end sections was more than scraping. Since the most important part of our project was with data, we had to improve ourselves about scraping. Many of us have done research on scraping and improved ourselves. Sometimes, when problems arose, we tried to overcome these problems together.

7.3 Taking lead role and sharing leadership on the team

There are people who basically coordinate the three parts of the project. We try to solve problems together and in this case it increases efficiency. In these meetings, everyone puts forward their ideas about a problem and we try to choose the best outcome from these ideas together. It makes our job easier when everyone takes ownership of the project as a leader.

References

[1] “Her şeyi Karşılaştır,” Epey, https://www.epey.com/ (accessed March 12, 2024).

[2] “CIMRI - Fiyat Karşılaştırma,” Google, https://www.cimri.com/market (accessed March 12, 2024).

[3] “Privacy and security in Firebase,” Firebase, https://firebase.google.com/support/privacy (accessed Mar. 12, 2024).