Bilkent University

Department of Computer Science

Senior Design Project

Project Name: PriceWise
Final 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

May 13, 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

In recent years, the common problem of all citizens living in Turkey is high inflation. The biggest blow that citizens receive from high inflation is shopping for basic needs. People now try not to miss affordable price opportunities for any product while shopping. Daily grocery shopping is among people's biggest expenses. Therefore, it is important to find cheap products in grocery shopping.

People generally buy certain products as a routine, even though the prices are high. However, online shopping has also clarified that distinct convenience stores sell identical items (same product and brand) at varying prices on their websites. A chocolate bar from a particular brand may cost 50 TL in Store A. On the other hand, Store B may sell the same chocolate bar for 30 TL [1]. Since the prices of the products are constantly changing, it is not easy to find where the suitable product is at the moment.

PriceWise is a mobile app that assists people in shopping. It collects the latest data from various stores and shows all the different prices of the same product. Thanks to this, people will easily compare and find the cheapest price. However, the most important feature that distinguishes PriceWise from its counterparts is that it shows the market with the most optimized price in the shopping lists. People will enter the products in their lists, and the app will compute the total list prices from different stores. Then, PriceWise will display the cheapest shopping list price to the user. Our goal with this application is to improve people's shopping experiences and enable them to save money without reducing their quality of life.

2. Requirements Details

2.1 Functional Requirements

- Secure login and authentication process.
- Tracking of user selections and preferences to offer more personalized shopping suggestions.
- Ability for users to create, edit, and manage their accounts.
- A feature to obtain explicit user consent for tracking their selections and preferences, in compliance with data privacy and protection regulations.
- Clear communication to users about how their data will be used to enhance their app experience and the measures taken to ensure data privacy and security.

- Users can create, edit, and save multiple shopping lists.
- Functionality to add, remove, or modify items in the shopping list.
- Capability to search for individual items or entire shopping lists.
- Retrieval of best available prices for each item from a range of websites and stores.
- Display of price comparison for similar products from different brands (e.g., suggesting cheaper milk from brand X over pricier brand Y).
- Option to view and select alternative products based on price and brand.
- Functionality to suggest optimized split shopping lists, indicating which items to buy from which sellers or websites for overall cost minimization.
- Real-time updating of prices and availability of items from various online sources.
- Push notifications for price drops, special offers, or other relevant alerts based on user preferences and shopping lists.
- Feature for users to provide feedback or report issues.
- Access to customer support for assistance with app-related queries.
- Limited functionality (viewing saved lists) available in offline mode.
- Using the probability system that takes into account users' preferences and habits.
- Providing users with a suggested list based on their shopping habits.
- Continuous adaptation of the model to refine suggestions based on user interactions.

2.3 Non-Functional Requirements

2.3.1 Usability

- The application should have an user friendly and intuitive interface to provide a positive user experience.
- The layout should be similar to other competitors as this will ease the transition process of the users to our application.
- In order to add many items to a shopping list more easily users will have a current list that they have chosen and they can add as many items as they wish.

2.3.2 Reliability

- The application should be accurate and up to date since the prices of many products in Turkey change frequently. It should inform the users about incoming sales for the products.
- In order to be accurate with the prices, the algorithm will work every 24 hours and admins can run the algorithm individually to update the prices if they think it is necessary.
- The application should be available for almost all times except for maintenance times.

2.3.3 Performance

- The application should have minimal response times under 2 seconds to ensure a positive user experience.
- The application should be able to handle high request traffic during special days such as holidays and Black Friday as these days are expected to be peak times for the usage of this application.

2.3.4 Cost

- There is no fee to download, use and become a member of the application.

2.3.5 Security

- The users' information such as username, email, and password should be secured from 3rd party softwares.

3. Final Architecture and Design Details

3.1 Overview

In this section, the general architecture of the software and the subsystem decomposition are discussed. 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 most important algorithms like list and product suggestion.

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.

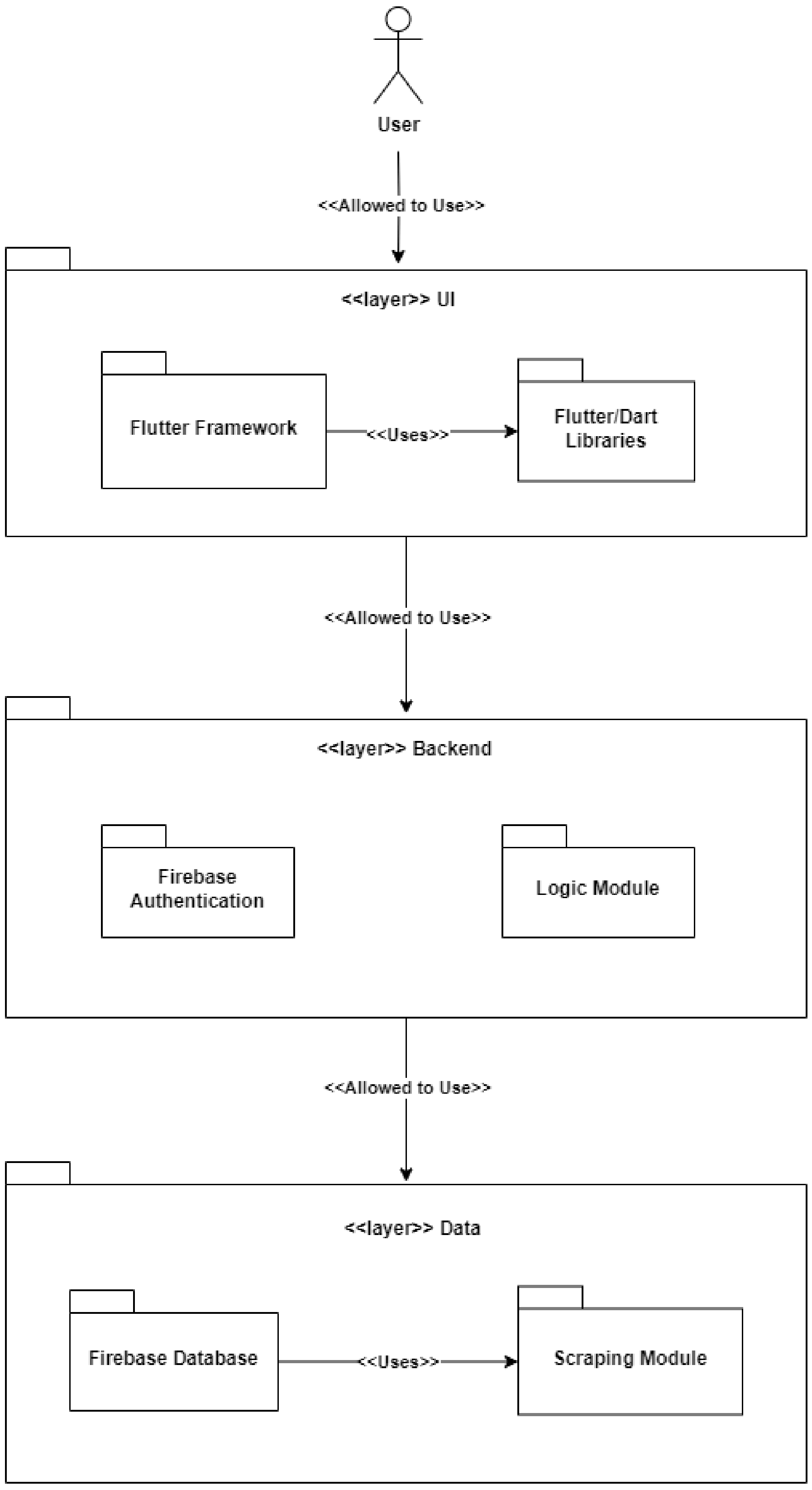


Figure 1. Subsystem Architecture Schema

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 [2]. The information of the products 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. Development/Implementation Details

In this section, the processes of front-end design, implementation of core back-end functionalities, and data fetching-storing operations will be discussed.

4.1 User Experience and Interface Design

Creating an easy-to-use front end was important because solving the time-consuming and tiring nature of grocery shopping was among our biggest promises. Even if we satisfy all the required backend functionalities, our app would lose its purpose and become pointless if users get exhausted and confused by a complicated design. We aimed to make the shopping process as smooth as possible. During our current market analysis, we also analyzed the designs of the websites of the biggest convenience stores in Türkiye such as Migros [3]. They not only have proven success in user engagement but also if we use similar design features, people would get used to our app very quickly.

To design a user-friendly app interface, we used Figma and Canva which are advanced design tools with lots of capabilities. We could create the optimal front-end design in our minds in a relatively short time.

4.2 Fetching and Storing Data

The most important part of the application is to have sufficient data. There are 12,000 products in the final version of the system. The products were scraped from the cimri.com website. These products include product names, cheapest prices, similar products, market prices, and product features. Data is pulled from Python with the BeautifulSoup library.

4.3 Product and List Suggestions

4.3.1 Product Suggestions

The product suggestion algorithm is designed to propose alternative products based on a combination of name similarity and price competitiveness. The algorithm operates by comparing the name of a product currently in the user's shopping list to other products within the same category, using a similarity metric to ensure relevance. The similarity metric is based on the intersection of terms in the product names, which helps in identifying products that are likely substitutes or variations.

In addition to name similarity, the algorithm considers the price of products. It suggests products that are not only similar but also cheaper than the item currently considered by the user. The price filter is set to suggest products that are at least 50% of the price of the original product to avoid recommending items that are not comparable in quality or size. The combined approach of using name similarity and price evaluation ensures that the suggestions are both relevant and economically advantageous to the user.

4.3.2 List Suggestions

List suggestions in the application are generated through two distinct approaches: market-specific lists and mixed-market lists. These suggestions aim to provide users with cost-effective alternatives for their entire shopping list, potentially sourced from specific markets or a combination of various suppliers.

1. Market-Specific Lists: This approach focuses on creating suggested lists where all items are sourced from a single market. This is particularly useful for users who prefer shopping from one vendor due to loyalty programs, convenience, or other factors. The algorithm evaluates each product in the original list, searches for the cheapest available

option for that product from the specified market, and compiles these into a new list. This method uses price comparisons and name similarity metrics to ensure that the suggested products match the user's original intentions as closely as possible.

2. Mixed-Market Lists: For users who prioritize cost over vendor loyalty, mixed-market lists provide an alternative. This method generates suggestions by selecting the cheapest available product across all markets, regardless of the vendor. It also employs name similarity and price comparison to ensure relevance and cost-effectiveness. The result is a shopping list that potentially minimizes the total cost by leveraging price disparities across different vendors.

Both approaches utilize a combination of name similarity for product matching and price comparisons to ensure that the suggestions are practical and offer real savings.

4.4 Product Search Algorithm

The product search feature in the application is designed to help users find products quickly and efficiently, using a flexible search algorithm similar to "flex search." This algorithm allows for partial and complete matches against product names, enhancing user experience by accommodating various search behaviors.

The search functionality is implemented using Firestore's query capabilities, specifically leveraging the `orderBy`, `startAt`, and `endAt` methods. These methods enable the algorithm to perform substring matching, which is essential for implementing a flexible search that can handle inputs starting from any part of the product names. This feature is particularly useful when users are unsure of the exact names or only remember part of the names.

When a user types a query into the search box, the search algorithm converts the input into a case-insensitive pattern. This pattern is then used to query the database for matching product names, allowing for both full-name matches and partial matches. The search results are limited to the top 100 matches to ensure performance efficiency and user manageability of results.

The product search algorithm enhances the shopping experience by making it easy for users to find exactly what they need quickly, even with only partial information, thereby saving time and improving the overall usability of the application.

5. Test Cases and Results

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

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

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.	Pass

2. Sign in.	User is signed in successfully.	Pass
3. Click on the Search page button.	Search page is displayed.	Pass
4. Enter any item in the search box.	Relevant search results for the item are displayed.	Pass
5. Select an item from the search results.	Item details page is displayed.	Pass
6. Click the back button.	Navigated back to the search results page.	Pass

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.	Pass
2. Sign in.	User is signed in successfully.	Pass
3. Click to the desired list.	List’s contained items and their total price is displayed.	Pass
4. Click on a suggested list displayed.	Suggested shopping list details are displayed.	Pass

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

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

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

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.	Pass
2. Sign in.	User is signed in successfully.	Pass
3. Click on Profile button.	Profile page with user credentials is displayed.	Pass
4. Click on View Past	Page displaying past	Pass

Shopping Lists button.	shopping lists is displayed.	
5. Select and click on any past list displayed.	Details of the selected past shopping list are displayed.	Pass
6. Click on another page button.	Navigated away from the past shopping lists page.	Pass

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.	Pass
2. Sign in.	User is signed in successfully.	Pass
3. Go to the Lists page.	Lists page is displayed.	Pass
4. Select the Suggested List.	Suggested list based on user preferences is displayed.	Pass
5. Change the products according to desires (delete/ add product).	Products and the total price of the list are successfully changed in the list.	Pass

6. Click the Back button.	Navigated back to the Lists page.	Pass

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.	Pass
2. Go to Duty page.	Duty page for admin tasks is displayed.	Pass
3. Click the Update Database button.	Update Database interface is displayed.	Pass
4. Perform the necessary updates.	Updates are successfully applied to the database.	Pass
5. Click the Back button.	Navigated back to the Duty page.	Pass

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.	Pass

2. Go to Duty page.	Duty page for admin tasks is displayed.	Pass
3. Click the Send Notification button.	Notification sending interface is displayed.	Pass
4. Enter the message and recipient details.	Message and recipient details are correctly entered.	Pass
5. Click the Send button.	Notification is sent successfully.	Pass
6. Click the Back button.	Navigated back to the Duty page.	Pass

All test cases have been completed successfully. In tests, it takes 1 second to move from one page to another, the reason for this is that the products are fetched from the database. It takes 2 seconds to open the product recommendation page, because the similarity level of all products is measured. It takes 5 seconds to open the list suggestion page, this is because more than one product is checked. Refreshing the database takes approximately 2 hours as it requires scraping all data from the website.

6. Maintenance Plan and Details

The products in the database will be updated regularly every day at 6 am. It takes approximately 2 hours to scrape the products from the website. Generally, the opening time of most markets is 10:00, and updating the database is likely to reach this time in the worst case scenario. Other than the database, there are no features that need to be updated regularly. It is normal and natural for errors to exist in the program and in such cases we plan to fix the error in an organized manner by error.

7. Other Project Elements

7.1. Consideration of Various Factors in Engineering Design

7.1.1 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 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.

7.1.2 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.

7.1.3 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.

7.2. Ethics and Professional Responsibilities

Our primary professional responsibility is to display the most recent product prices to inform the users accurately. We frequently need to get the updated price data from the markets, preferably once every day. If we neglect this, all price comparisons, list optimization algorithms, and item suggestions of PriceWise would give wrong results. The app would not be able to find the optimum shopping list. Therefore, PriceWise would be unsuccessful in general. Promising to help people save their budgets and then giving false directions that cause them to lose money would be highly unethical. It would also kill the usability of PriceWise and the credibility of our team.

Furthermore, we must retrieve user data only if the user allows us to do so. Then, we must use this data to only suggest alternative items for their lists and nothing else. We should not share user data with third parties.

Lastly, we should not produce any biased result in favor of a grocery store that offers a bribe to us to show the users their products in the optimum solutions even when there are better alternatives.

7.3. Teamwork Details

7.3.1. Contributing and functioning effectively on the team

In order to ensure the success of our project each team member plays a significant role, to achieve this creating an emphasis on effective collaboration is a key element. It is vital that all team members are aligned regarding individual workloads and associated deadlines, with the goal of increasing each member's strengths to ensure balanced participation. We held most of our meetings online. The reason for this was to increase continuity and 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. Thus, we held meetings every week via Discord. At these meetings, everyone was telling us what they did that week. Then we were evaluating the situation, what was missing and what extra we could add. Finally, we were deciding what we needed to do until the next meeting. 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.

7.3.2. 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.

7.3.3. 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.

7.3.4. Meeting objectives

When the project is completed, we think that we have completed all the features we wrote in the first report. But, for example, instead of using machine learning in the recommendation system, we used a probability-based system. We considered the products of users' previous purchases as a single product, and then compared the products of that category one by one. This method was called Jaccard Index, but we thought of this way ourselves. The reason why we did not use Machine Learning was that we had to create a very serious shopping database to create a model. We have approximately 12,000 products, but the feature dictionaries of each product are not within the scope we want. We thought that in order to have a good recommendation system based on machine learning, there should be a shopping list almost as many as the number of products and a complete feature dictionary of all products. We thought that with Jaccard Index we would have a better recommendation system with less shopping database. Apart from this, we added recipes to our database. These recipes show what products are used to make that recipe. If a user is considering using this recipe, the user can automatically create a list with a click of a button.

7.4 New Knowledge Acquired and Applied

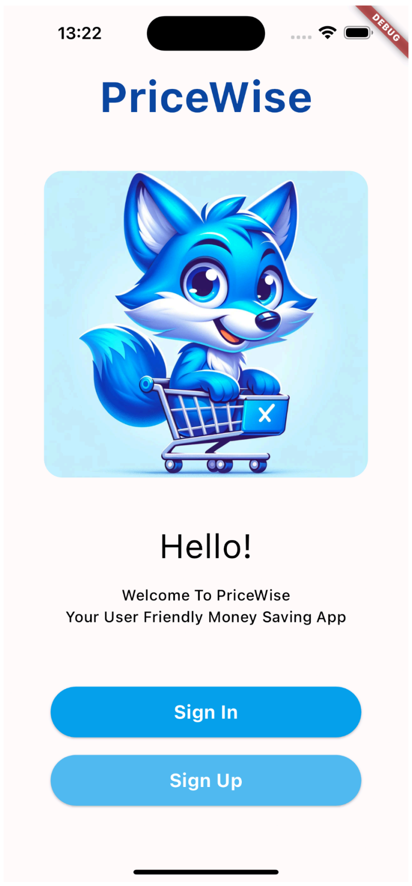
In this project, we use these following technologies:

- Flutter
- Google Firebase Database
- Mobile App Development
- Web Scraping
- Database Management

In order to progress effectively in the project, we need to have a certain level of knowledge on these issues. We can learn this information from courses (Database Management) as well as from sources available on the internet. We also need to know how to proceed and what to do when developing the project. We all have experience in this regard with the CS 319 course.

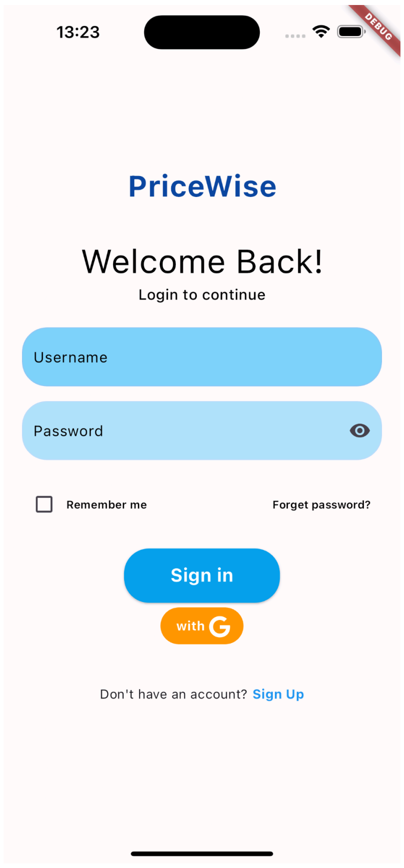
8. User’s Manual

8.1 Opening Page



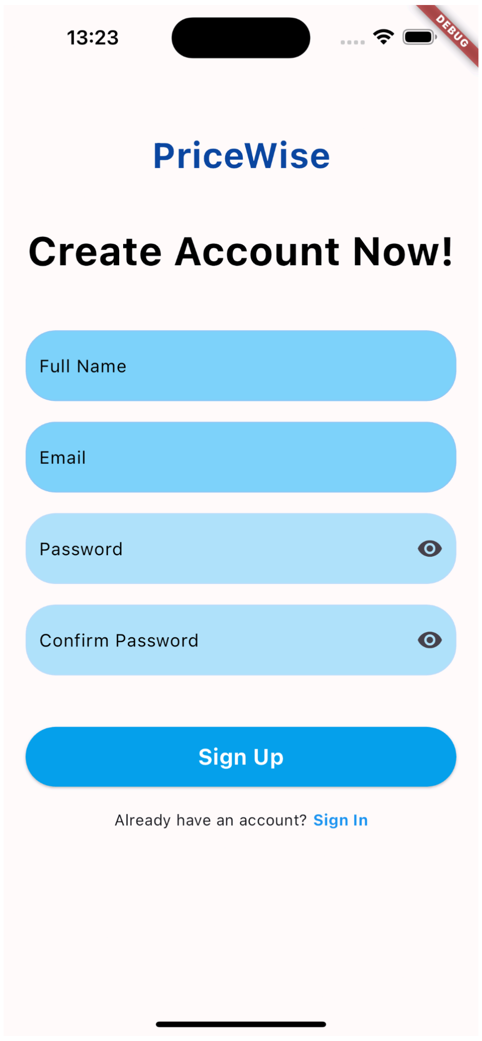
On the Opening page, users are presented with two options. Member users can log in to the application with the Sign In button, while non-members can use the Sign Up button.

8.2 Sign In



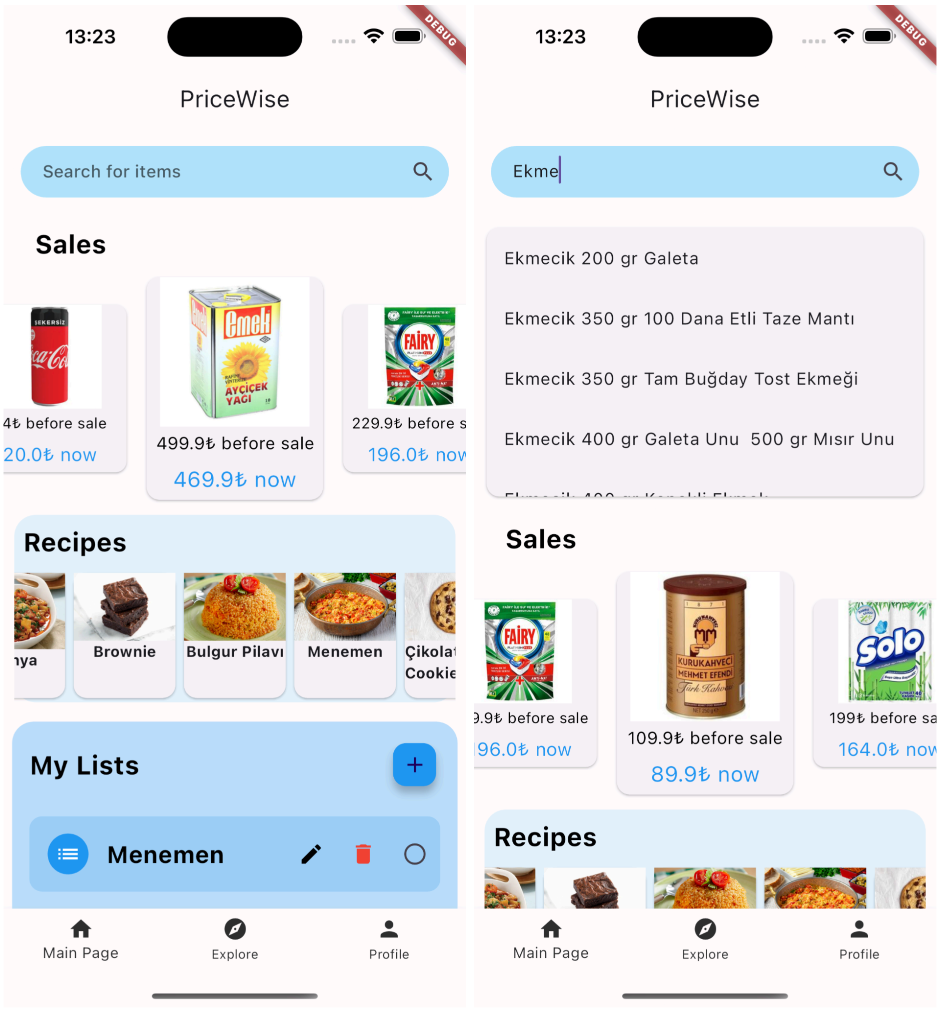
On the Sign In page, there is a form for the username and password information required for registered users to log in. Users who wish can log in to the application as a member without constantly entering their user information, with the Remember Me feature. As an additional option, there is also the option to sign in with Google.

8.3 Sign Up



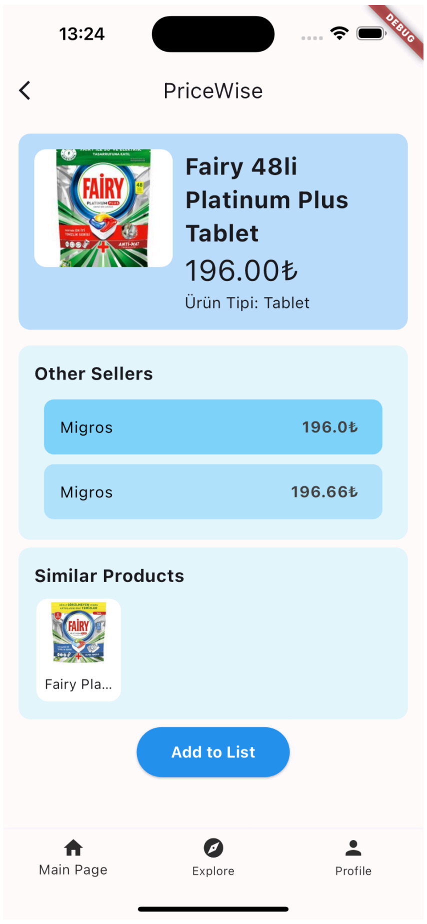
On this page, users who are not yet members can become members by entering their full names, e-mail addresses and passwords. There is also a button to go directly to the Sign In page from this page.

8.4 Main Page



On the Main page, there are four main features offered to users. In the first section, which is the Sales section, the products that are on sale appear. In the Recipes section, there is a ready-made recipe option under this section. You can choose any of the ready-made recipes in the database and automatically create a list. The last section contains My Lists. This section contains the user's lists. Users can create lists as they wish. At the top there is the Search form. Here, users can search and find the product they want.

8.5 Product Page


13:24

PriceWise

Fairy 48li Platinum Plus Tablet

196.00₺

Ürün Tipi: Tablet

Other Sellers

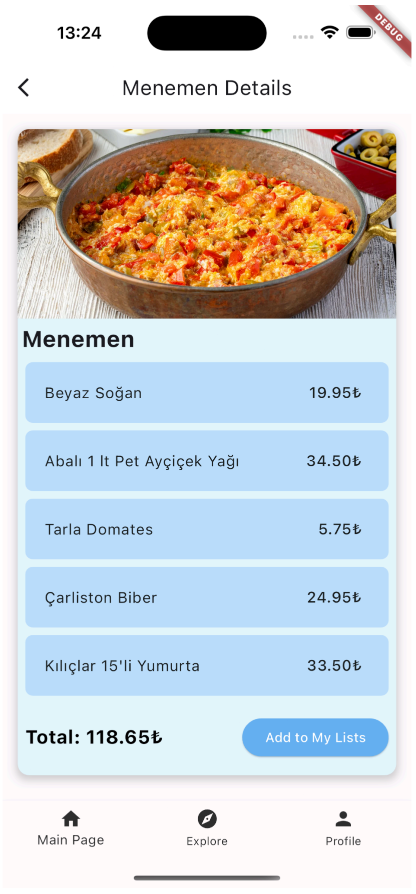
Migros	196.0₺
Migros	196.66₺

Similar Products

Fairy Pla...

The Product Page contains information about the product. The cheapest price of the product and all prices in other markets are visible. Products similar to this product appear in the Similar Products section. Users can add the selected product to a list if they wish.

8.6 Recipe Page



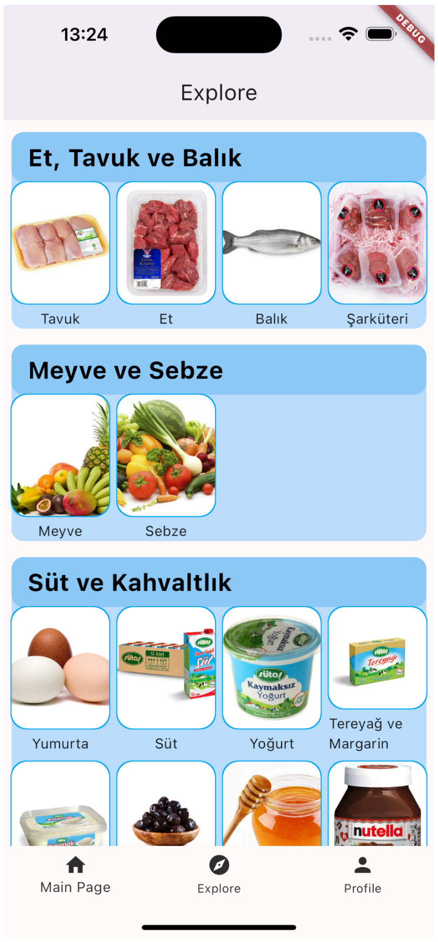
Menemen

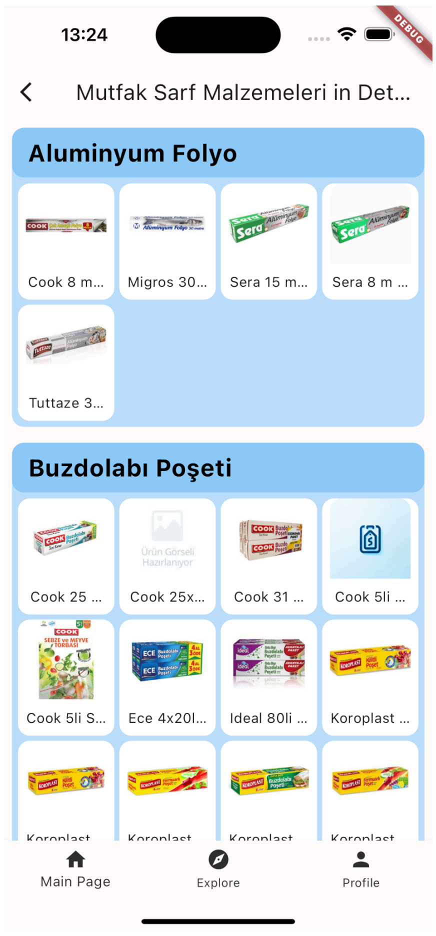
Beyaz Soğan	19.95₺
Abalı 1 lt Pet Ayçiçek Yağı	34.50₺
Tarla Domates	5.75₺
Çarliston Biber	24.95₺
Kılıçlar 15'li Yumurta	33.50₺

Total: 118.65₺

On the Recipe page, the products required for the selected recipe are written. The list with the cheapest price for the selected recipe is displayed. If preferred, a new list with the name of the recipe will be created.

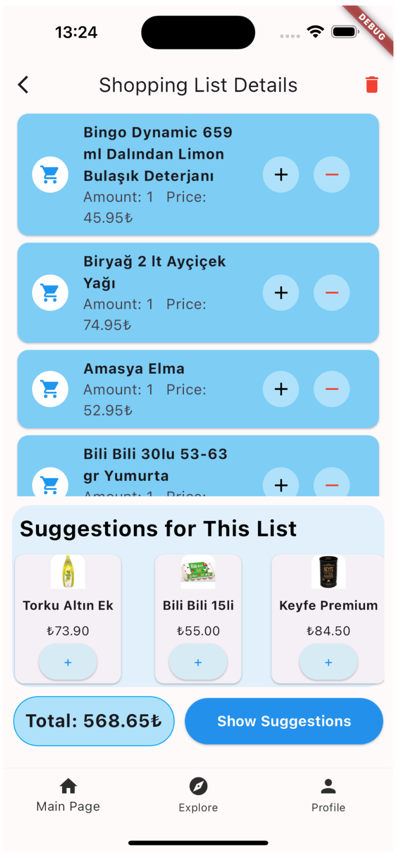
8.7 Explore Page





On the Explore page, there are 12,000 products in the database, distributed according to categories.

8.8 List Page


13:24

Shopping List Details

Bingo Dynamic 659 ml Dalından Limon Bulaşık Deterjanı
Amount: 1 Price: 45.95₺

Biryağ 2 lt Ayçiçek Yağı
Amount: 1 Price: 74.95₺

Amasya Elma
Amount: 1 Price: 52.95₺

Bili Bili 30lu 53-63 gr Yumurta

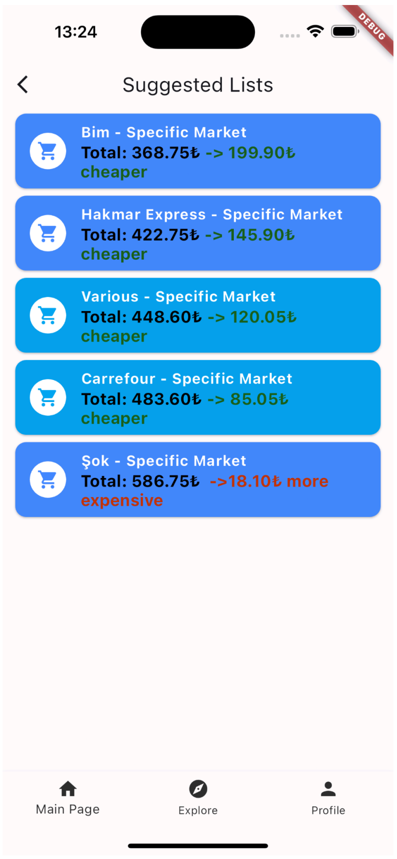
Suggestions for This List

Torku Altın Ek ₺73.90

Bili Bili 15li ₺55.00

Keyfe Premium ₺84.50

Total: 568.65₺ Show Suggestions


13:24

Suggested Lists

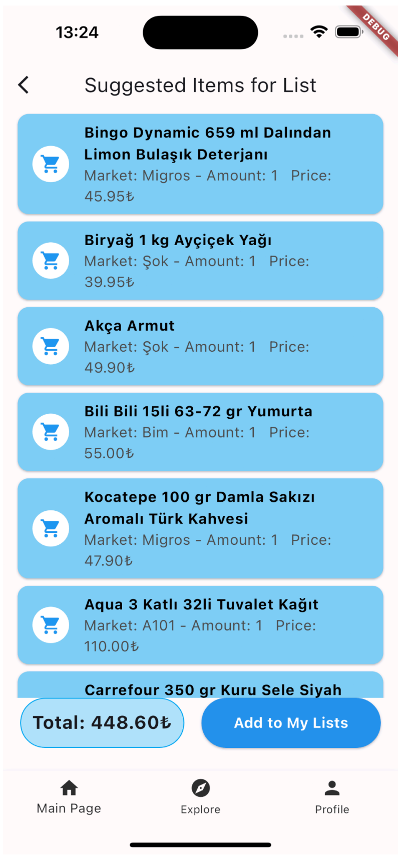
Bim - Specific Market
Total: 368.75₺ -> 199.90₺ cheaper

Hakmar Express - Specific Market
Total: 422.75₺ -> 145.90₺ cheaper

Various - Specific Market
Total: 448.60₺ -> 120.05₺ cheaper

Carrefour - Specific Market
Total: 483.60₺ -> 85.05₺ cheaper

Şok - Specific Market
Total: 586.75₺ ->18.10₺ more expensive


13:24

Suggested Items for List

Bingo Dynamic 659 ml Dalından Limon Bulaşık Deterjanı
Market: Migros - Amount: 1 Price: 45.95₺

Biryağ 1 kg Ayçiçek Yağı
Market: Şok - Amount: 1 Price: 39.95₺

Akça Armut
Market: Şok - Amount: 1 Price: 49.90₺

Bili Bili 15li 63-72 gr Yumurta
Market: Bim - Amount: 1 Price: 55.00₺

Kocatepe 100 gr Damla Sakızı Aromalı Türk Kahvesi
Market: Migros - Amount: 1 Price: 47.90₺

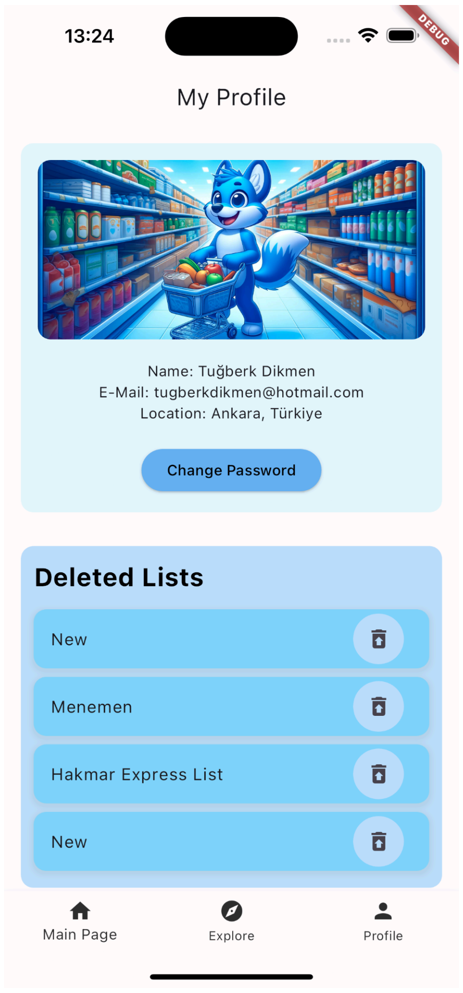
Aqua 3 Katlı 32li Tuvalet Kağıt
Market: A101 - Amount: 1 Price: 110.00₺

Carrefour 350 gr Kuru Sele Siyah

Total: 448.60₺ Add to My Lists

On the List page, the products in the selected list are listed. Users can increase or decrease the number as they wish. At the bottom there is the Show Suggestions button. From here users can see suggested lists for the specified list.

8.9 My Profile Page



Name: Tuğberk Dikmen
E-Mail: tugberkdikmen@hotmail.com
Location: Ankara, Türkiye

Deleted Lists

- New
- Menemen
- Hakmar Express List
- New

On the My Profile page, users' information and the last 5 deleted lists are shown. If preferred, users can recover the deleted list in its final form.

9. Conclusion and Future Work

To conclude, we worked devotedly as a group to finish the project from September 2023 to May 2024. The features we planned to develop while designing the application at the beginning of the year were included in the final version of the application. This was the first big project where everyone in our group was at the center. The experiences in this process will have great contributions in your future business life.

Our first plan for the future is to see the reactions to the application at CS Fair. Then, we plan to make the necessary updates based on the feedback and upload the application to the Google Play Store and App Store.

10. Glossary

Agile Methodology: This methodology refers to a set of principles for software development that emphasizes flexibility, collaboration and iterative development.

API: An API is a set of rules and protocols that allows different software applications to communicate with each other.

Back-end: The server side of the components of a software application.

BeautifulSoup: Python library for parsing HTML and XML documents. It provides different tools for extracting data from web pages.

Cloud Firestore - Firebase: NoSQL database provided by Google. It is a flexible, scalable and real time database.

CRUD operations: These operations refer to the basic actions that can be performed on data in a database.

Dart: Dart is a programming language, primarily used for building web, mobile applications.

Database: A set of structured collection of data.

Discord: Communication platform used for online meetings.

Flutter: Open Source UI software development kit, primarily used for building applications for web and mobile.

Front-end: User facing components of a software application.

Jaccard Index: Also known as Jaccard similarity coefficient. It is a measure used to find the similarity of two different subsets.

Machine Learning: Machine Learning is a subset of Artificial Intelligence that focuses on the development of algorithms and models which makes computers learn and make predictions based on other data.

NoSQL: Database Management System that provides a mechanism for storage and retrieval of data.

Python: High level programming language known for its simplicity, readability. It can be used for various purposes.

UI: User Interface refers to the visual elements of a software application

Web Scraping: A concept for extracting data from websites.

11. References

[1] “CIMRI - Fiyat Karşılaştırma,” Google, https://www.cimri.com/market (accessed Nov. 16, 2023).

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

[3] “Migros-Hemen”, Google, migros.com.tr , (accessed Nov. 12, 2023).