SMART GROCERY SHOPPING APPLICATION

(GROCAFAST)

CPIT - 499 Final Report

By

Elaf Aloufi	1911265
Layan Fakhurji	2110850
Manar Altaiary	1906775
Raneem Alshareef	1907696

Supervised By

Dr. Abeer Hakeem

Department of Information Technology
Faculty of Computing and Information Technology
King Abdulaziz University
Jeddah – Saudi Arabia

Spring 2023

DECLARATION by AUTHORS

"I/we certify that this work has not been accepted in substance for any degree and is not concurrently being submitted for any degree other than that of BS Information Technology being studied at King Abdulaziz University, Jeddah. I/we also declare that this work is the result of my/our own findings and investigations except where otherwise identified by references and that I/we have not plagiarized another's work".

[Signature]	[Signature]	[Signature]	[Signature]
			
[RANEEM]	[LAYAN]	[ELAF]	[MANAR]

DECLARATION by SUPERVISOR

I, the undersigned hereby certify that I have read this project report and finally approve it with recommendation that this report may be submitted by the authors above to the final year project evaluation committee for final evaluation and presentation, in partial fulfillment of the requirements for the degree of BS Information Technology at the Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah.

[Signature]

Dr. Abeer Hakeem

Acknowledgement

We would like to acknowledge and express our deepest thanks to our supervisor Dr. Abeer Hakeem. She has been very patient and giving throughout the project. Her insightful comments and suggestions have been invaluable in shaping our work.

Furthermore, all the talented group members have played major roles in maintaining the project up to our high standards, each in her own way. Thank you for your cooperation and the astonishing communication. This project would not have been possible without your dedicated efforts.

Last but not least, we would like to thank our families for their unwavering support, understanding, and patience. Their love and encouragement have been our source of strength and motivation.

Thank you all for making this project possible.

Abstract

Grocery shopping is one of the activities most of us find exhausting yet essential. This project aims to enhance the in-store grocery shopping experience by developing a Smart Grocery Shopping Application (GROCAFAST).

GROCAFAST provides shoppers with an in-store shopping route which helps save the shoppers' time and organizes the shoppers' flow inside the store using the shortest path algorithm (Dijkstra) to optimize the shopping route .

The application is developed for a specific grocery store, allowing shoppers to create a grocery list that is organized according to the store layout to guide the shopper from the entrance to the checkout point, passing through all the aisles that contain the items on the shopper grocery list. In addition to several features that aim to save the shoppers' time and improve grocery shopping experience. GROCAFAST is tested on a small scale to evaluate its effectiveness in enhancing the shopping experience. Results showed that the application can reduce the time required to complete a shopping trip.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Problem Definition

Grocery shopping is an essential activity, yet most of us find it exhausting. Everyone needs to go to grocery stores to buy food, drinks, laundry supplies, etc. However, shopping for groceries is not a simple activity as one may think. It demands preparation, clarity, and commitment.

Grocery shopping is one of the most common chores in our lives and can be one of the most stressful. Each trip to the grocery store takes an average of 41 minutes, which equates to approximately 53 hours annually. Navigating the store can make this routine task more overwhelming and challenging.

For instance, shoppers often (1) waste a great amount of time by crossing all the store aisles to collect their groceries; (2) shoppers struggle when they cannot find a particular item in-store where they expected to be. This happens because each store has its different layout. Even stores under the same brand have different layout designs and organizational structures. Time is precious, so shoppers want stores to be logically laid out to avoid wasting time while trying to find each item. (3) shoppers sometimes need help looking for a specific item. Not having an item that shoppers expect to find at the store, or available within the time frame they expect are major frustration points.

Although grocery stores have made significant improvements over the years, there are still inefficiencies in the shopping experience that can be addressed with technology. One such inefficiency is the use of paper lists, which can be easily lost or forgotten and do not allow for dynamic updates. To address this issue and enhance the in-store grocery shopping experience, we are developing a Smart Grocery Shopping Application (GROCAFAST). This application will enable shoppers to create dynamic shopping lists that can be updated in real-time and organized based on the layout of the grocery store.

The main objective of GROCAFAST is to enhance the in-store grocery shopping experience, save shoppers time, and ultimately increase their satisfaction.

1.2 Project Scope

As grocery shopping is an activity that is carried out regularly by a large segment of society, the impact of GROCAFAST is expected to be significant.

To develop and test GROCAFAST, we chose to collaborate with a local grocery store located in the female dormitory at King Abdulaziz University.

By conducting our tests in a real-world setting, we can gather valuable insights into the feasibility and effectiveness of GROCAFAST, identify areas for improvement, and ensure that GROCAFAST is designed to meet the needs and preferences of the target audience, which will ultimately contribute to its success.

1.3 Aims and Objectives

The primary objective of GROCAFAST is to enhance the in-store grocery shopping experience, which is a key factor in the success of any retail business. To achieve this objective, GROCAFAST offers a range of features designed to save shoppers' time -which will be detailed in the suggested solution section-

All the features provided by GROCAFAST aim to simplify the shopping process and save shoppers' time. By achieving these objectives, GROCAFAST can contribute to the success of retail businesses and increase shoppers' satisfaction and loyalty.

1.4 Suggested Solution

To help shoppers save time and enjoy an easy and smooth in-store grocery shopping experience, we propose a solution named GROCAFAST.

GROCAFAST offers various features, including (1) managing the shopper grocery lists (Figure 1). (2) generating a route map to guide the shopper inside the store, starting from the entrance to the checkout point, passing through all the aisles that contain the items on the shopper's grocery list, (Figure 2). (3) searching for a specific item and viewing its location inside the store (Figure 3, Figure 4), (4) providing personalized shopping suggestions based on the shopper's previous lists (Figure 5), and (5) creating an online chat room for shoppers to share reviews and exchange recommendations (Figure 6).

Furthermore, GROCAFAST benefits all stakeholders involved in the grocery shopping experience. Shoppers can enjoy a seamless shopping experience by following the in-store route determined based on their grocery list without needing assistance from the store staff. This would increase shoppers' satisfaction rates and lead to higher shoppers' loyalty. Additionally, since GROCAFAST stores shoppers' grocery lists in a database, manufacturers can analyze the data to identify the best-selling products and shoppers' preferences, enabling them to make informed decisions and identify future trends. This data analysis would provide manufacturers with valuable insights that can help them solve problems, improve their products, and make strategic decisions.

Overall, by providing a range of features that simplify the grocery shopping process and enhance the shopper experience, GROCAFAST can benefit all stakeholders involved, leading to increased shoppers' satisfaction and loyalty, improved sales for retailers, and valuable data insights for manufacturers.

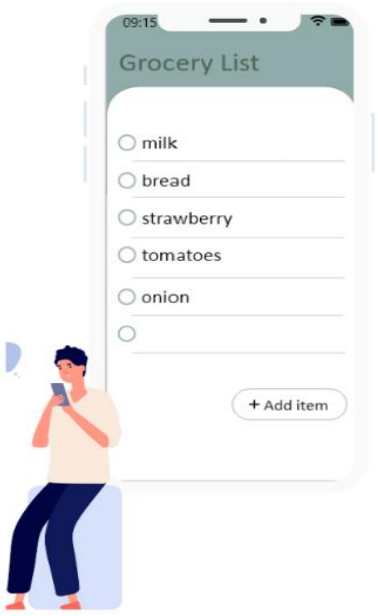


Figure 1: Create a grocery list.

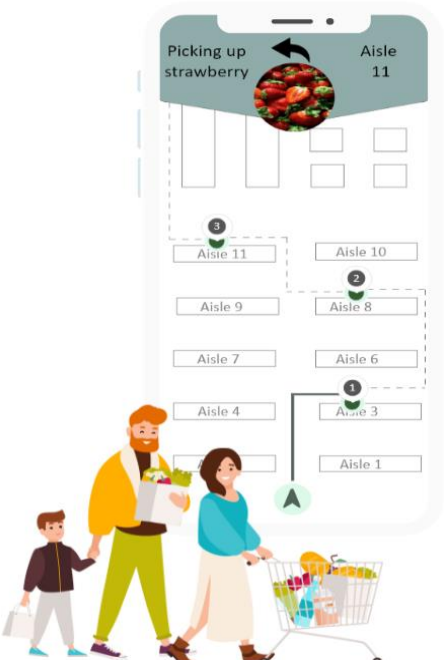


Figure 2: Generate a route map based on the item's location.

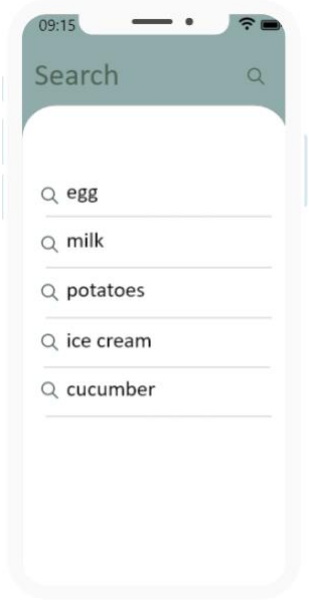


Figure 3: Search for a specific item.

Figure 4: Navigate to the item location.

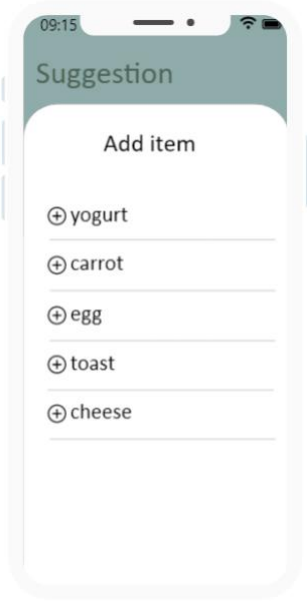


Figure 5: Shopping suggestions based on previous lists.

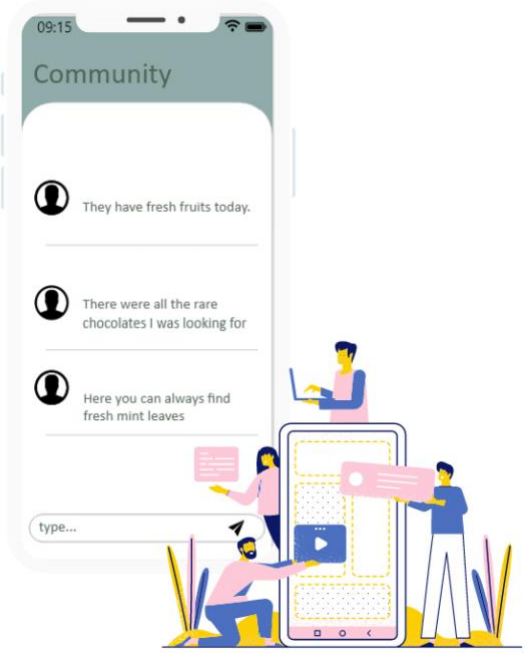


Figure 6: Share reviews

1.5 GROCAFAST Architecture

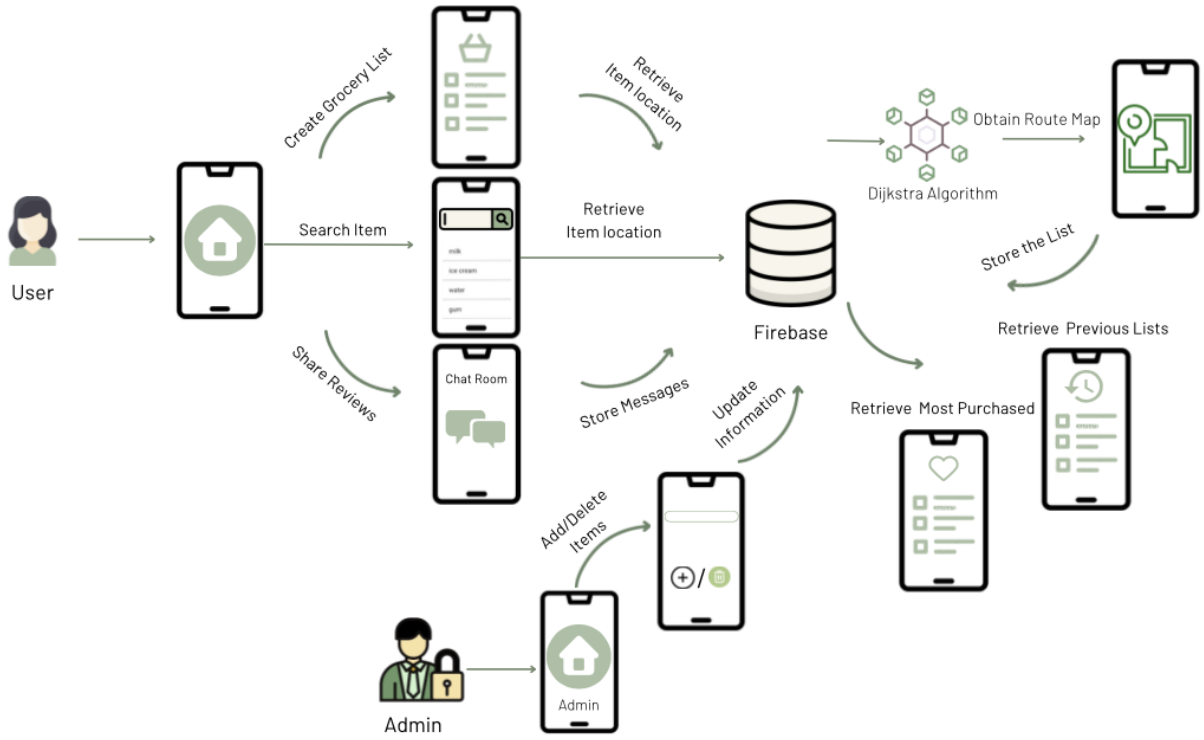


Figure 7: GROCAFAST Architecture

1.6 Methodology

A software development methodology is a framework used to structure, schedule, and manage the process of developing an information system. By choosing the appropriate methodology, a project can produce more accurate estimates, deliver more stable systems, create clear understanding of the work that lies ahead, and spots pitfalls earlier.

Since our project has a well-defined end scope and requirements, the Waterfall model is employed. Each phase of the project has specific deliverables and clear procedures that need to be followed to complete each phase before proceeding to the next one.

The basic phases of the waterfall methodology:

1. Requirements Gathering & Analysis

   To evaluate the effectiveness of GROSAFAST services and explore more features, an online survey is shared between regular shoppers to collect their opinions and suggestions. All required information is collected and analyzed to determine GROCAFAST's software/hardware specifications.

2. System Design

   GROSAFAST's design document must be established, which includes the application interfaces prototype and the architecture diagrams of the application.

3. Implementation

   The development process to build GROSAFAST must be started as outlined in both the analysis and the design phases.

4. Testing

   In this phase, GROSAFAST functionality must be tested at the local grocery store in the female dormitory at King Abdulaziz University.

5. Deployment

   In this phase, GROSAFAST must be accessible to end users.

6. Maintenance

   Once GROSAFAST is deployed, feedback from end users must be fetched to solve problems and bugs as they arise.

1.7 Designing Tools

This section lists the utilized tools. More tools might be added as the project proceeds through the development cycle.

- Google Forms

  To build an online survey and analyzing the results using clean response data with automatic summaries.

- Google Drive – Google Documents

  To collaborate on drafting our documents with real-time updates.

- Google Meets

  It is integrated with Google Workspace to conduct online meetings with a secure connection.

- Microsoft Word

  To create and collaborate on documents with built-in intelligent features.

- Microsoft Teams

  To conduct online meetings and edit files in real time.

- Draw.io

  A free platform to create and share diagrams integrated with several libraries of ready-to-use charts.

- Lucid

  An online design tool for creating charts and diagrams. -A backup tool if we encountered any limitations with Draw.io.

- Figma

  A cloud-based design tool that works on any platform and provides real-time updates for its embedded files. It is a user-friendly tool that facilitates developer handoffs and communication.

- Adobe XD

  A vector-based design tool for designing user interfaces (UI). -A backup tool if we encountered any limitations with Figma-.

- Canva

  A graphic design tool that simplifies digital design. It is also an inspiration source for design ideas.

- MySQL

  An open-source SQL database management system is used to add, access, and process data stored in a computer database.

- Visual Studio Code

  Source code editor to launch running apps with integrated debugging features.

- Android Studio

  An Integrated Development Environment (IDE) for Android app development that incorporates code editing and developer tools.

- Firebase

  A set of backend cloud computing services provided by Google to host databases.

- Beacon

  A small, wireless, and high-tech device that works based on Bluetooth Low Energy. It can transmit signals that other devices can receive.

- IndoorAtlas[19]

  Indoor positioning and location-based service that provides accurate location information inside buildings. It offers a range of SDKs and APIs.

- CubiCasa[20]

  A tool specialized in creating floor plans and 3D models of real estate properties using mobile devices.

1.8 Gantt Chart

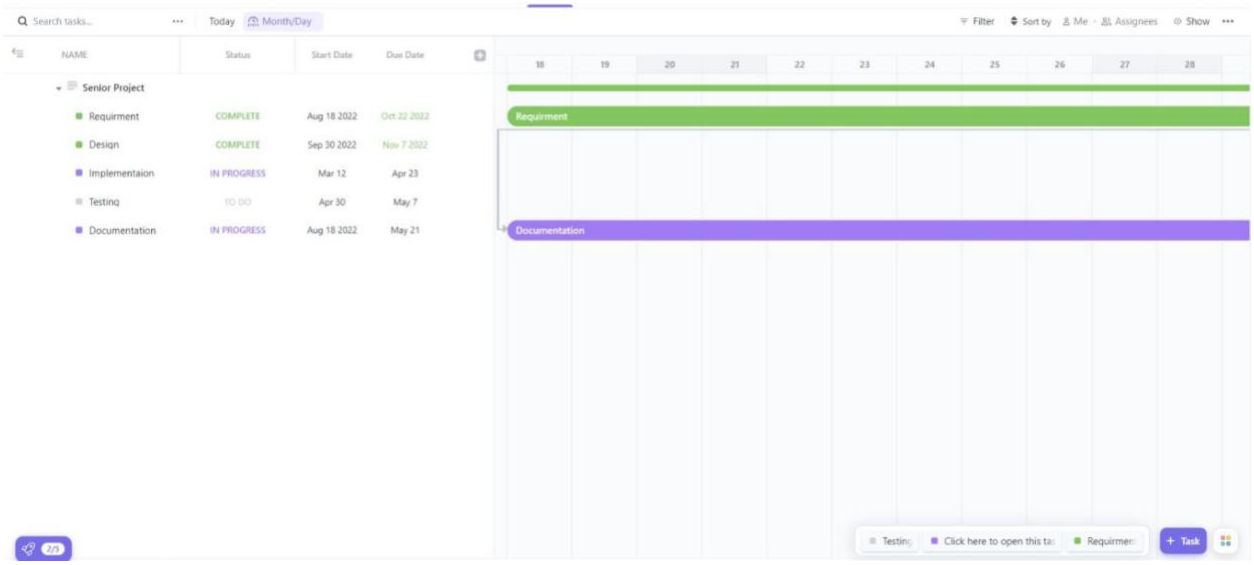


Figure 8: Gantt Chart

Chapter 2: Background and Similar Applications

2.1 Project Background

Grocery shopping is an essential activity in our lives. The act of buying groceries will never go away, but how we buy groceries has certainly evolved over the years. The old grocery system involved shoppers calling or bringing their grocery lists. Then, stores would bag items for pick-up or delivery. However, Clarence Saunders revolutionized the industry when he opened the first self-service grocery store, the Piggly Wiggly, in 1916 in Memphis, Tennessee. By the 60s, shoppers became used to strolling through aisles and hand-selecting items themselves.[9]

The following section overviews the history of **Grocery Lists** and how they evolved over the years.

2.1.1 Grocery Lists

Going to grocery stores without preparing a grocery list in advance is a waste of time. Shoppers will aimlessly shop without knowing what to buy or what is needed. Thankfully, the old grocery system proved that shoppers started writing grocery lists before the 50s, which means that they realized the importance of grocery lists from the day they started grocery shopping.[9]

Shoppers commonly shop for groceries with lists of items to purchase. The use of grocery lists represents a degree of pre-shopping planning, which reduces impulsive purchasing.[15]

The grocery list has transformed from a handwritten list to a digitized one. Studies showed that the paper shopping list is longer and includes more products than the digital shopping list. Furthermore, products on the paper shopping list are less hedonic than those on the digital shopping list.[7]

Modern shoppers thrive on time and convenience. They are searching for solutions to optimize time and energy. Luckily, many applications were developed to manage grocery lists to meet the consistently evolving shoppers ‘needs.

2.2 Summary of Similar Works

2.2.1 Applications

2.2.1.1 Application 1: *AnyList*[3]

AnyList is an application that creates grocery shopping lists, then intuitively organize them into categories and allows shoppers to share them. Additionally, it collects and organizes recipes.

Figure 9: AnyList Application

AnyList Application Features:

- Grocery shopping lists
  It allows the user to create multiple lists to organize items and separate them into categories.

- Siri
  It allows the user to add items by voice with Siri and Reminders Import feature.

- Favorites
  It allows the user to store items as favorites to use them later.

- Recipes
  It allows the user to enter recipes to organize them into collections by type or occasion and share them with a trusted partner.

- Notifications
  Providing optional push notifications for any shared list modifications.

- Design
  It allows the user to badge the app icon optionally with the number of items remaining on the lists and choose a color for each list to help distinguish lists.

Features are summarized as follows:

Features	*AnyList*	*GROCAFAST*
Create grocery shopping lists	✓	✓
Sort the list into categories	✓	
Siri	✓	
Favorites	✓	
Storing food recipes	✓	
Notifications	✓	
Custom list design		✓
Generates a route map to guide the shopper inside the store		✓
Search for an item and view its location in the store		✓
Provide personalized shopping suggestions based on the shopper's previous lists	✓	✓
Create an online chat room for shoppers to share reviews and exchange recommendations		✓

Table 1: Features of AnyList Application.

Relevance:

Both *AnyList* and *GROCAFAST* are based on the idea of creating and managing grocery lists.

Differences:

- *AnyList*
  The application sorts grocery lists automatically by categories and stores food recipe ingredients. *AnyList's* objective is to provide stress-free shopping, cooking, and meal planning.

- *GROCAFAST*
  Generates a route map to guide the shopper inside the store. GROCAFAST's objective is to improve the in-store grocery shopping experience.

2.2.1.2 Application 2: *Listonic*[10]

Listonic is an application that allows the shoppers to write and share a synchronized list of items across all platforms and automatically sorts them based on their category.

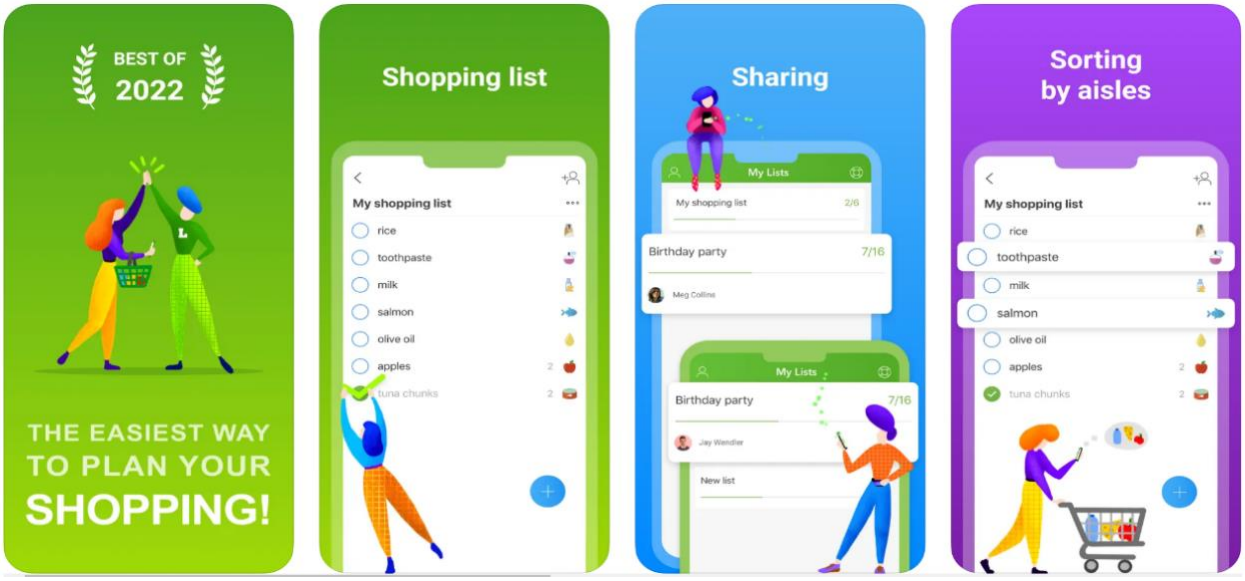


Figure 10: Listonic Application

Listonic Application Features:

- Smart and custom categories

  It allows the user to add list of items and automatically sort them by categories.

- Sync across all platforms

  It allows the user to share a synchronized lists with others across all platforms.

- Unlimited lists

  It allows the user to create and share various type of lists.

- Smart suggestions

  It provides an intelligent item prompter feature that suggests to the user the duplicated purchases based on the previous lists by understanding the users' shopping history.

- Control user’s budget

  It allows the user to control their spending by calculating the total price automatically once the item is added to the grocery list.

Features are summarized as follows:

Features	*Listonic*	*GROCAFAST*
Smart and custom categories	✓	✓
Sync across all platforms	✓	
Unlimited lists	✓	✓
Smart suggestions	✓	✓
Control user's budget	✓	
The app is linked to a store database		✓
Generates a route map to guide the shopper inside the store		✓
Search for an item and view its location in the store		✓
Provide personalized shopping suggestions based on the shopper's previous lists		✓
Create an online chat room for shoppers to share reviews and exchange recommendations		✓

Table 2: Features of Listonic Application.

Relevance:

Both *Listonic* and *GROCAFAST* are based on the idea of creating and managing grocery lists.

Differences:

- *Listonic*

  The application automatically sorts grocery lists based on categories resulting in saving the shoppers's time and money.

- *GROCAFAST*

  Generates a route map to guide the shopper inside the store resulting in saving the shopper's time and organizing the shopper flow inside the store.

2.2.1.3 Application 3: *Grocery* – Smart Shopping List[5]

Grocery is a grocery shopping application that retains the list sorted while shopping using simple sorting algorithms with syncing support for iCloud and Reminders. It combines intelligent sorting, shopping histories, and recipe management.

Additionally, *Grocery* has an intelligent sorting feature that allows the shoppers to minimize their steps and provides enjoyment every time the shopper crosses items off the list.

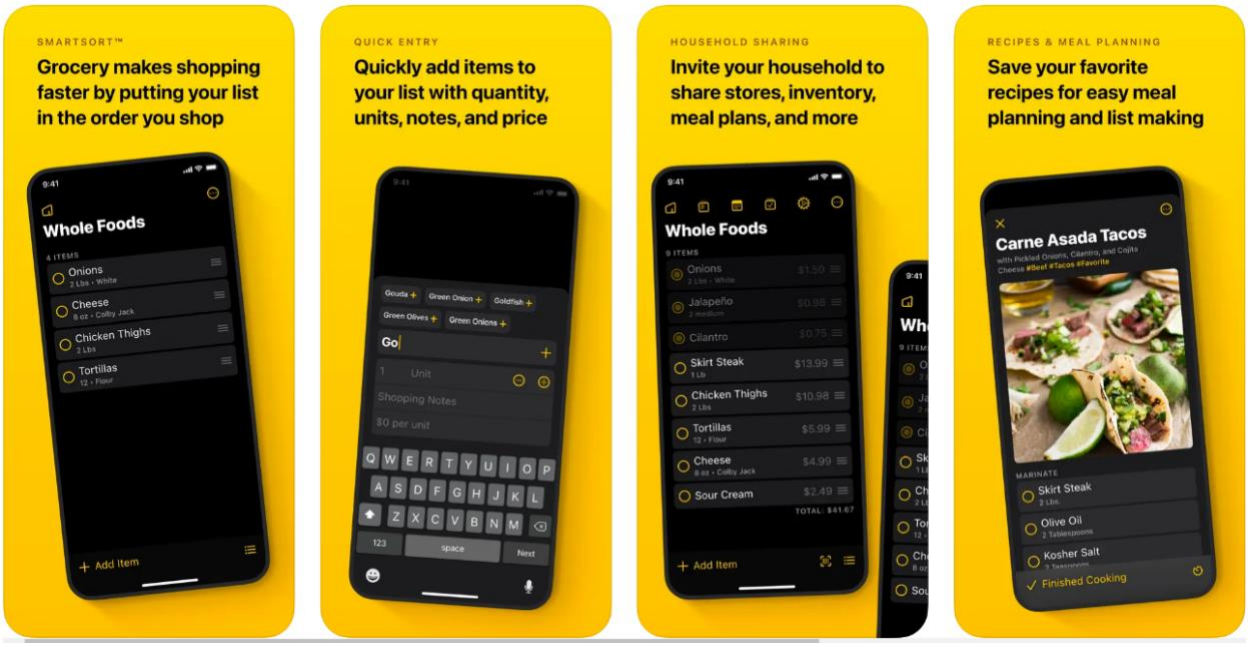


Figure 11: Grocery – Smart Shopping List Application

Grocery – Smart Shopping List Application Features:

- Smart Sorting

  The app automatically sorts the shopping list by two ways:

  - In the order in which the user crosses items off the list.
  - In the order specified by the user.

- Tracking the user's shopping list

  It analyses data from previous shopping trips to sort future lists.

- Pantry Tracking

  It allows the user to track items in the pantry and know when they expire.

- Reminder sharing

  It works with the reminder app to share lists with partners and stay synchronized once it is shared.

- Lightning Fast

  It provides a custom auto-complete with the user's list items and notes.

- Syncing

  It allows the user to share the shopping list with partners and it will be syncing in iCloud makes using Grocery across iOS devices seamless and easier.

- Siri

  It allows the user to add the shopping list items using Siri.

Features are summarized as follows:

Features	*Grocery*	*GROCAFAST*
Smart sorting	✓	✓
Sync across all platforms	✓	
Pantry Tracking	✓	
Smart suggestions	✓	✓
Lightning Fast	✓	
Reminder sharing	✓	
Siri	✓	
The app is linked to grocery store database		✓
Generates a route map to guide the shopper inside the store		✓
Search for an item and view its location in the store		✓
Provide personalized shopping suggestions based on the shopper's previous lists		✓
Create an online chat room for shoppers to share reviews and exchange recommendations		✓

Table 3: Features of Grocery Application.

Relevance:

Both *Grocery* and *GROCAFAST* are based on the idea of creating and managing grocery lists.

Differences:

- *Grocery* – Smart Shopping Application

  The application sorts the grocery list in the order in which the shopper crosses items off the list or in the order specified by the shopper to save the shoppers's time.

- *GROCAFAST*

  Generates a route map to guide the shopper inside the store resulting in saving the shopper's time and organizing the shopper flow inside the grocery store.

2.2.2 Comparative Study

2.2.2.1 Design and Development of SMART LIST: A Mobile App for Creating and Managing Grocery Lists.[11]

This paper described the design and development of a mobile app for creating and managing grocery lists. As a result, SMART LIST, a prototype of a mobile app for managing grocery lists, was developed, and evaluated.

The study contributes towards an understanding of system requirements for such apps and could be a reference model for developers and researchers to improve the process for creating and managing grocery lists.

Advantages:

- Easy to use.
- Simple to learn. No written instructions are needed.
- The feature of setting a reminder for an item running out of stock.

Disadvantages:

- Limited features.

Features are summarized as follows:

Features	**SMART LIST Application**	***GROCAFAST***
New user registration	✓	✓
Login to the system	✓	✓
Create a new shopping list	✓	✓
Modify a previous list	✓	✓
Add/ Remove items on the list	✓	✓
organize the list into categories		✓
Add a quantity of an item	✓	
The app is linked to a grocery store database		✓
Generates a route map to guide the shopper inside the store		✓
Search for an item and view its location in the store		✓
Set a reminder for grocery items that are almost running out of stock.	✓	
Provide personalized shopping suggestions based on the shopper's previous lists		✓
Create an online chat room for shoppers to share reviews and exchange recommendations		✓

Table 4: Features of SMART LIST Application.

Relevance:

Both *SMART LIST* and *GROCAFAST* apps are based on the idea of creating and managing grocery lists using modern smartphones.

Difference:

GROCAFAST is designed for a particular store. It considers the store aisles and the arrangement of products on each shelf to provide in-store navigation for shoppers. In contrast, the SMART LIST app is designed to digitize, organize, and manage grocery lists.

Evaluation:

The study provided an overview of how to understand the system requirements and user interface of a mobile app for managing grocery lists. It is an excellent reference model to develop similar apps and enhance the capabilities in creating and managing grocery lists.

2.2.2.2 The Smart Shopping List: An Effective Mobile Solution for Grocery List Creation Process.[17]

This paper describes how people continue to utilize outdated techniques for grocery shopping, such as writing a list of what must be bought on paper or memorizing the items, which is not reliable and ultimately leads to time and financial loss.

The paper introduces the *Smart Shopping List*, an Android application that improves the grocery shopping experience. The Apriori algorithm was used in the application. Therefore, this paper is a valuable resource that enables us to study similar algorithms.

Advantages:

- Shares a list via SMS.
- Notifies the user of the nearest grocery store.

Disadvantages:

- Users cannot modify or reuse previous lists.
- The application works only on Android platforms.

Features are summarized as follows:

Features	***Smart Shopping List***	***GROCAFAST***
Interactive shopping list that enables shoppers to add/remove/cross items		✓
Notifies the shoppers of the nearest grocery store in which they can find all their needs	✓	
Share the shopping list with others	✓	
Organize the list into categories		✓
Generates a route map to guide the shopper inside the store		✓
Search for an item and view its location in the store		✓
Provide personalized shopping suggestions based on the shopper's previous lists		✓
Create an online chat room for shoppers to share reviews and exchange recommendations		✓

Table 5: Features of Smart Shopping List Application

Relevance:

Both *GROCAFAST* and *Smart Shopping List* shoppers can create a grocery list and receive item suggestions.

Difference:

GROCAFAST is concerned with how much time shoppers spend shopping for groceries. It provides a route map to make it easier for shoppers to locate a specific item inside the store. On the other hand, the *Smart Shopping List* creates an interactive grocery list and locates the nearest store to the shopper.

Evaluation:

The paper encompasses a detailed explanation of how the application was implemented and the algorithms that were applied. By using a data mining algorithm, the paper declared that 82% of shoppers benefited from the *Smart Shopping List* application.

Chapter 3: Analysis

This chapter describes the data gathering techniques, the functional and non-functional requirements, and their initial analysis using Use Case diagrams, in addition to mapping these requirements using the mapping matrix.

To relieve any confusion, the store Shopper is now referred to as the User of the application.

3.1 Data Gathering

This section discusses the data-gathering techniques used and the findings from the collected data analysis.

3.1.1 In-Field Experiment

The main objective of the in-field experiment is to evaluate the effectiveness of the proposed solution. The in-field experiment was conducted in a specific Hyper Panda branch. We made sure to collect real data by choosing a grocery store branch that we do not normally visit, therefore, do not know its layout.

The experiment focused on calculating the time taken to collect a number of products using a pre-written grocery list containing 20 items.

On the first visit, the grocery list was written in disorder, and the items were unsorted by categories. It took an hour (60 minutes) to collect all the items on the list. On the second visit, the same grocery list was arranged and sorted by categories. This time, it took 35 minutes to collect all the items on the list.

As a result, we obtained solid proof that we are providing a solution for a real-world problem. This experiment helped us evaluate the effectiveness of GROCAFST and reinforce our argument.

3.1.2 Interview

<table>
<tr><td>Interviewee

Mr. Hazem Alahmadi</td><td>Interviewer

Elaf Aloufi

Layan Fakhurji

Manar Altaiary

Raneem Alshareef</td></tr>
<tr><td>Location/ medium:

Online via Google Meet</td><td>Appointment Date:

Start Time: 10:00 AM

End Time: 11:30 AM</td></tr>
<tr><td>Objectives:

Elaborate the idea of the project and understand how grocery store arrange products on aisles.</td><td>Reminders:

Mr. Hazem Alahmadi is a former IT employee at Hyper Panda</td></tr>
<tr><td colspan="2">Agenda:

• Introducing the project team.
• The project idea.
• Questions regarding the Hyper Panda aisles layout.
• Closing.</td></tr>
</table>

Table 6: Interview Table1

Interviewee: Hazem Alahmadi	**Date: 20 Sep 2022**
Question 1: After hearing the project idea, is it possible that Hyper Panda could grant us access to the database of the grocery store to implement the project?	**Answer:** Maybe yes, based on your official statement from your college, I just need the top management approval to proceed with your request.
Question 2: Do grocery stores change the aisles arrangement frequently?	**Answer:** No, all items are arranged based on category. The only change is when some companies do not provide their items weekly.
Question 3: Approximately, how many aisles are there in the grocery store?	**Answer:** There are 11 basic aisles in the grocery store. Even if you want to go to a new grocery store branch, you will likely find the same number of aisles.
Question 4: Do you think that GPS service could be used to provide in-store navigation for users inside the grocery store?	**Answer:** No, GPS service is used on a wider range like outdoors. It does not work inside buildings.

Table 7: Interview Table2

Conclusions

The interview was enriching. We gained valuable information that would help us in our project. Furthermore, we realized that GPS services cannot be used indoors, so we must figure out another way to develop in-store navigation.

After the interview, Mr. Hazem provided the layout of the aisles in one of the Hyper Panda branches, but he requested that the name of the branch stays confidential.

3.1.3 Survey

Finding relevant data for any project helps determine the project scope and the main problems to be solved.

A survey is an effective approach to collect information from the public. It helps shape the project requirements and evaluate its effectiveness.

An online survey written in both Arabic and English was created. There were ten straightforward questions in the survey including closed-ended, open-ended, and multiple-choice questions with some questions requiring answers to previous questions.

3.2 Functional Requirements

The method of Requirements Engineering defined in the Software Engineering book (chapter 4), 10th edition [14] is followed.

The functional requirements are divided into two parts, the user requirements (What the user can do) and the system requirements (How the system will make it happen).

User Requirements

R1: The user shall be able to create a new account.

R2: The user shall be able to login to the system.

R3: The user shall be able to create a grocery list.

R4: The user shall be able to modify a previous list (if any).

R5: The user shall be able to navigate inside the store using a route map.

R6: The user shall be able to search for an item and view its location inside the store.

R7: The user shall be able to receive personalized shopping suggestions.

R8: The user shall be able to create posts in the online chat room between users of the same store.

System Requirements

R1.1: The system shall store the user information (username) in the database.

R2.1: The system shall enable the user to login to the application after verifying the username.

R3.1: The system shall provide a space where the user can create a grocery list.

R4.1: The system shall store all the user's grocery lists.

R4.2: The system shall allow the user to modify previous grocery lists.

R5.1: The system shall organize the user's grocery list based on the store's aisles.

R5.2: The system shall generate a route map with the shortest path to collect all the items on the user's grocery list.

R6.1: The system shall allow the user to search for an item.

R6.2: The system shall locate the item searched by the user.

R6.3: The system shall display the location of the item to the user.

R7.1: The system shall store the user's previous grocery lists in the database.

R7.2: The system shall analyze the user's previous grocery lists.

R7.3: The system shall detect the user's most written items.

R7.4: The system shall display shopping suggestions to the user.

R8.1: The system shall create a space between users of the same store.

R8.2: The system shall allow the user to create posts.

3.3 Non-Functional Requirements

The non-functional requirements for GROCAFAST are:

1. **Usability**[13]

   Usability describes how well a user can use the system in a specific context to achieve a defined goal efficiently and satisfactorily.

   GROCAFAST interfaces must be simple and user-friendly, especially because it targets a large segment of society, some of which are novice users. The usability of GROCAFAST is measured by using the following factors:

   - Ease of learning

     The application must be easy to learn for both novices and users with experience in similar applications.

   - Ease of remembering

     The application must be easy to remember for casual users.

   - Understandability

     The user must understand the application functionality and services.

   - Subjective satisfaction

     The user must feel satisfied with the application.

2. **Maintainability**

"Maintainability states that software should be written in such a way that it can evolve to meet the changing needs of users." [14]

Since grocery stores are one of the most competitive and constantly growing sectors, the GROCAFAST must be in a continuous state of evolution. GROCAFAST measures the flexibility of how the application can be modified -for fixing issues or adding new functionality- with a degree of ease. It also focuses on the refactorability of the code by using component-based software engineering.

3. Reliability[14]

Reliability is the probability that a system failure will occur when the system is in use within a specified operating environment.

There are three possible techniques to achieve reliability, which are fault avoidance, fault detection, and fault tolerance. In our case, GROCAFAST will focus on fault tolerance, which is a runtime approach to dependability in which systems include mechanisms to continue in operation, even after a software fault has occurred and the system state is erroneous. Fault-tolerance mechanisms detect and correct this erroneous state so that the occurrence of a fault does not lead to a system failure.

By using dependable programming guidelines, the faults in the delivered application are reduced. Additionally, data about GROCAFAST operations is gathered to assess the reliability of the application. The data gathered include Rate Occurrence of Fault (ROCOF) or Time to Failure, which is the time or the number of transactions between system failures plus the total number of transactions. The data required include Rate Occurrence of Fault (ROCOF) or Time to Failure, which is the time or the number of transactions between system failures plus the total number of transactions.

Why security is excluded from the non-functional requirements:

GROCAFAST is a simple application. It only asks the user to enter a unique username to sign up. A unique username from the users is needed to:

- Store users' previous grocery lists to provide them with personalized shopping suggestions.
- Enable the users to participate in the online chat room created between users of the same store.

Therefore, security is not an important aspect to consider in GROCAFAST. Because the type of data the application collects about the users is public data, which has the lowest level of data classification. Since this type of data often gets shared and passed around, it is not considered an asset that requires a strong level of protection.

3.4 Stakeholders /Actors

Stakeholder:

- Project supervisor.
- Project team members.
- Users of the targeted store.
- Employees of the targeted store.
- Individuals who participated in the survey.

Actors:

- End users which are the store’s shoppers.

3.5 Initial Design / Analysis

3.5.1 Use Case Diagram

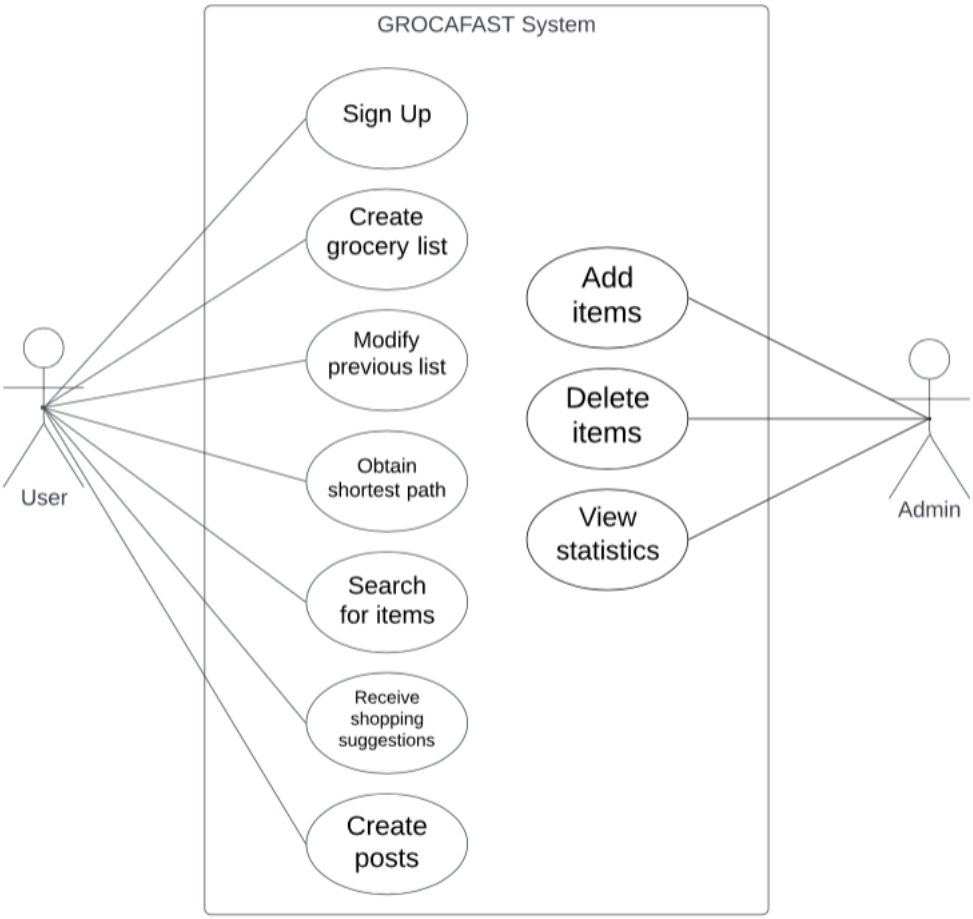


Figure 12: Use Case Diagram.

3.5.2 Elaborated Use Cases

Sign Up Use Case Description

Use Case Title	Sign Up
Actors	User
Description	The user enters a username to register in the application.
Data	A unique username.
Stimulus	The user chooses to create an account in the application.
Response	In a successful case, the user provides a unique username then he/she can access the application and use all its services. In case the user provides a username that already exists in the database, the application asks the user to try again with a different username.
Comments	The user must download the application to create an account.

Table 8: Sign Up Use Case Description.

Create Grocery List Use Case Description

Use Case Title	Create Grocery List
Actors	User
Description	The user can write a grocery list on the application.
Data	Grocery list items.
Stimulus	The user selects the Write a Grocery List option.
Response	In a successful case, the grocery list is stored in the database, and the user can proceed to display the sorted list. In case of a failure, if the user writes an exist item in the grocery list or does not write any item, error message displayed, and the user must try again.
Comments	The user must be logged into the application.

Table 9: Write Grocery List Use Case Description.

Modify Previous List Use Case Description

Use Case Title	Modify Previous List
Actors	User
Description	The user can modify previous grocery lists (if any).
Data	User’s previous grocery list.
Stimulus	The user selects the Modify a Previous Grocery List option instead of the Write a Grocery List option.
Response	In a successful case, the user can edit their grocery list, add more items, delete unwanted grocery lists, and then proceed to obtain the route map. In case of a failure, the database cannot retrieve the items, and the user must try again.
Comments	The user must be logged into the application. The user must have a previous grocery list to be able to modify it.

Table 10: Modify Previous List Use Case Description.

Obtain Shortest Path Use Case Description

Use Case Title	Obtain Shortest Path
Actors	User
Description	The user can obtain a route map generated by the application based on the user's grocery list to navigate the user inside the store.
Data	User's grocery list, products' categories, and the layout of the store aisles.
Stimulus	The user presses the arrow (Done button) after creating a grocery list.
Response	In a successful case, the route map is generated based on the user's grocery list to navigate the user inside the store. In case of a failure, the route map cannot be generated, and the user must try again.
Comments	The user must be logged into the application. The user must write a grocery list.

Table 11: Obtain Route Map Use Case Description.

Search for Items Use Case Description

Use Case Title	Search for Items
Actors	User
Description	When the user searches for an item, the application locates the item inside the store and view its location to the user.
Data	Item name, the layout of the store aisles.
Stimulus	The user selects the Search option and write the item name.
Response	In a successful case, the application locates the searched item inside the store and displays its location to the user. In case of a failure, the item cannot be located, and the user must try again.
Comments	The user must be logged into the application. The user must search for an item by its name.

Table 12: Search for Items Use Case Description.

Receive Shopping Suggestions Use Case Description

Use Case Title	Receive Shopping Suggestions
Actors	User
Description	Every time the user writes a grocery list on the application, the list is stored in the database and analyzed to detect the user's most written items in order to generate shopping suggestions for the user.
Data	User's grocery list and the products' categories.
Stimulus	Every time the user selects the Write or Modify a grocery list option.
Response	In a successful case, the application displays the shopping suggestions to the user. In case of failure, the database does not retrieve the items that written by the user, and the application asks the user to rewrite the items again.
Comments	The user must be logged into the application. The user must have previous grocery lists that have been analyzed to receive shopping suggestions.

Table 13: Receive Shopping Suggestions Use Case Description.

Create Posts Use Case Description

Use Case Title	Create Posts
Actors	User
Description	The user can create posts to share reviews in the space between the users of the same store created by the application.
Data	Username
Stimulus	The user selects the Create Post option.
Response	In a successful case, the user's post is created and displayed on the space. In case of a failure, the user's post cannot be displayed on the space, and the application asks the user to try again.
Comments	The user must be logged into the application.

Table 14: Create Posts Use Case Description.

3.6 Mapping Requirements

	R1	R2	R3	R4	R5	R6	R7	R8	R9
Sign Up	✓								
Write Grocery List			✓						
Modify Previous List				✓					
Obtain Shortest Path					✓				
Search for Items						✓			
Receive Shopping Suggestions							✓		
Create Posts								✓	

Table 15: Mapping Requirements.

The GROCAFAST is not a security-oriented application -as described in the non-functional requirements section-. Therefore, R2 and R10 (login and log-out) are not considered use cases.

Chapter 4: System Modeling

4.1 Modeling Approach

Our modeling approach is Object-Oriented Programming (OOP). OOP maximizes code reusability and makes programs more efficient and easier to understand.

Since grocery stores are one of the most competitive and constantly evolving sectors, GROCAFAST must be in a continuous state of evolution. Therefore, using OOP as a modeling approach helps in adding new functionalities in the future to fulfill the continuously prospering users' needs.

4.2 Diagrams for Design Approach

4.2.1 Class Diagram

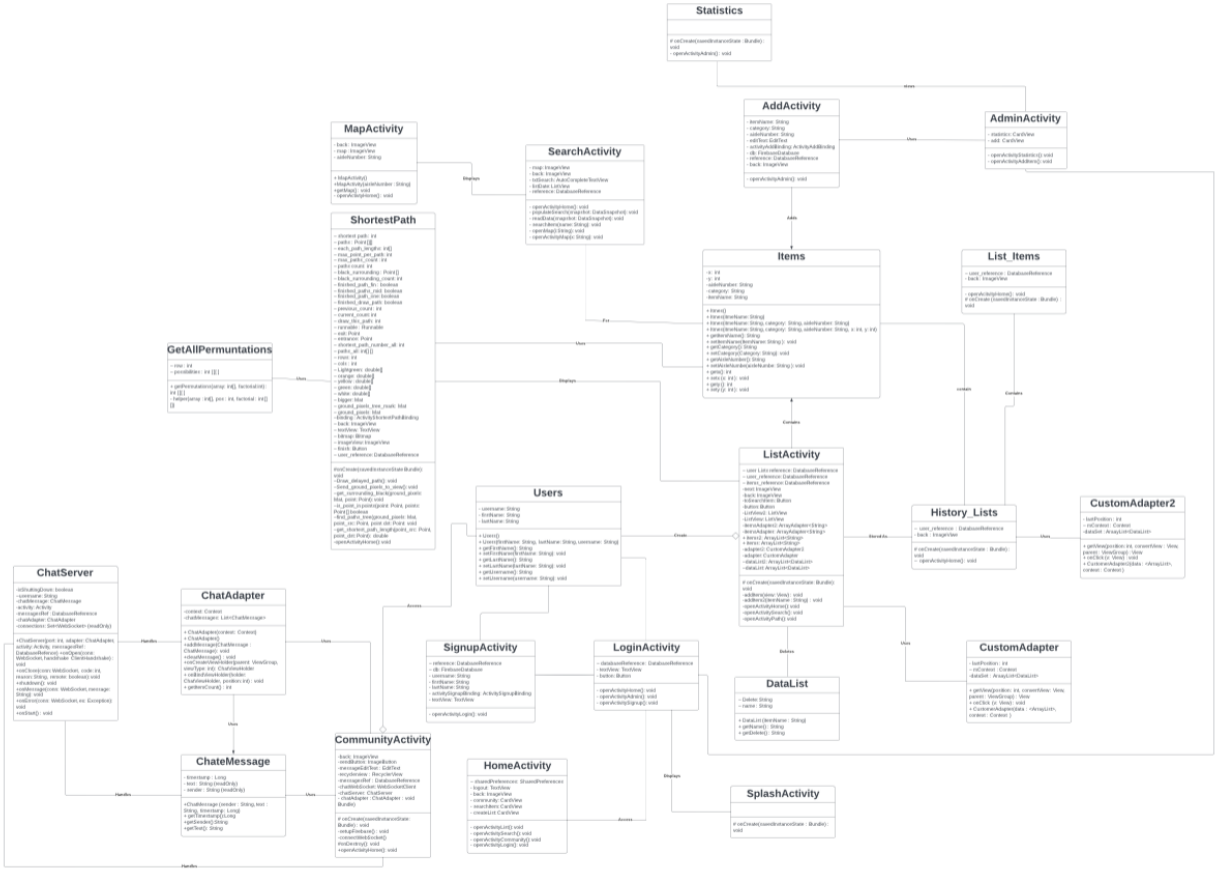


Figure 13: Class Diagram

4.2.2 Sequence Diagrams

Sequence diagrams are provided only for the most important use cases, which are Obtain Shortest Path, Search for Items, and Receive Shopping Suggestions.

Obtain Shortest Path Sequence Diagram

The Obtain Shortest Path sequence diagram describes how the system provides the route map with the shortest to the user.

The user creates a grocery list using the writeList() method, and the app stores it in the database using the storeList() method. Then, the sortList() method arranges the grocery list according to categories. The sorted list is displayed to the user then the user uses the getRouteMap() method to obtain the map.

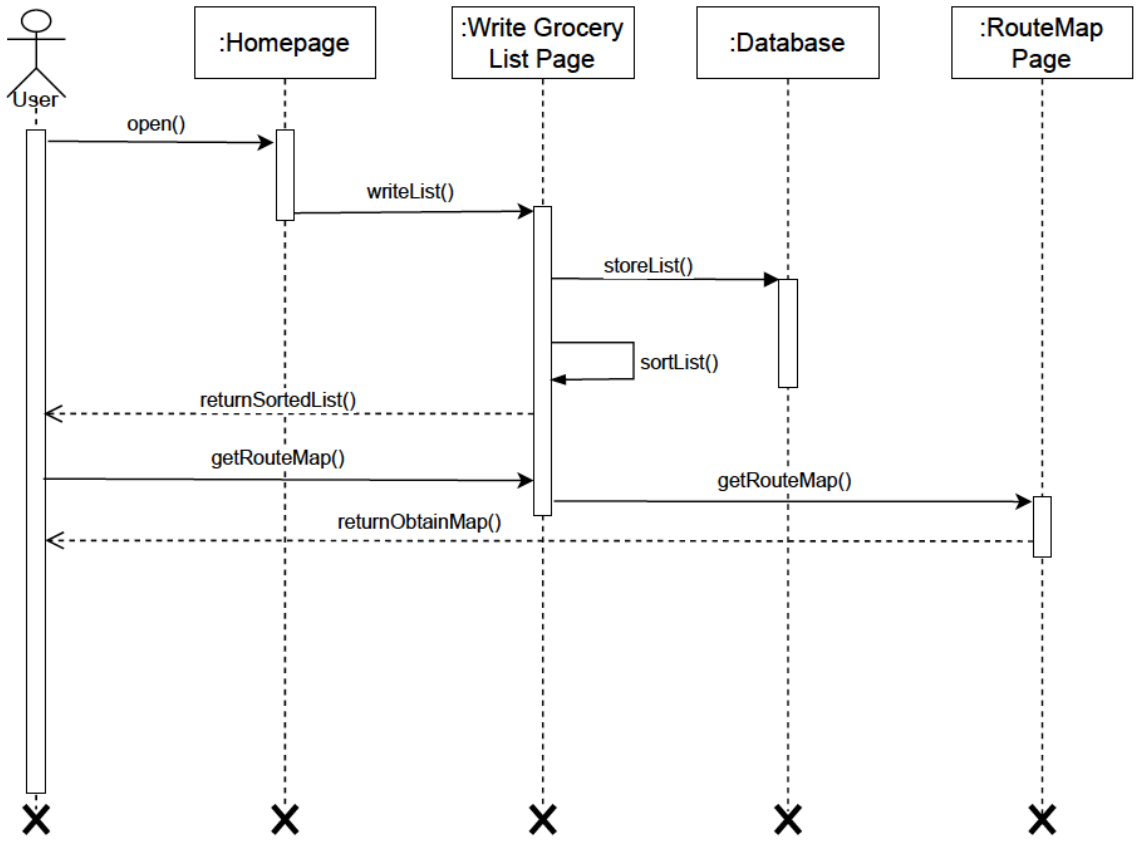


Figure 14: Obtain Shortest Path Sequence Diagram.

Search for Items Sequence Diagram

The Search for Items sequence diagram explains how a user can locate an item inside the grocery store and navigate to its location.

On the homepage, the user selects the search option and writes the item's name using the Search() method. Afterward, the system checks the item's location in the system database. If the item's location is found, the database returns the item's location and navigate the user to it. When the item's location is not found, a "Not Found Message" is returned to the user.

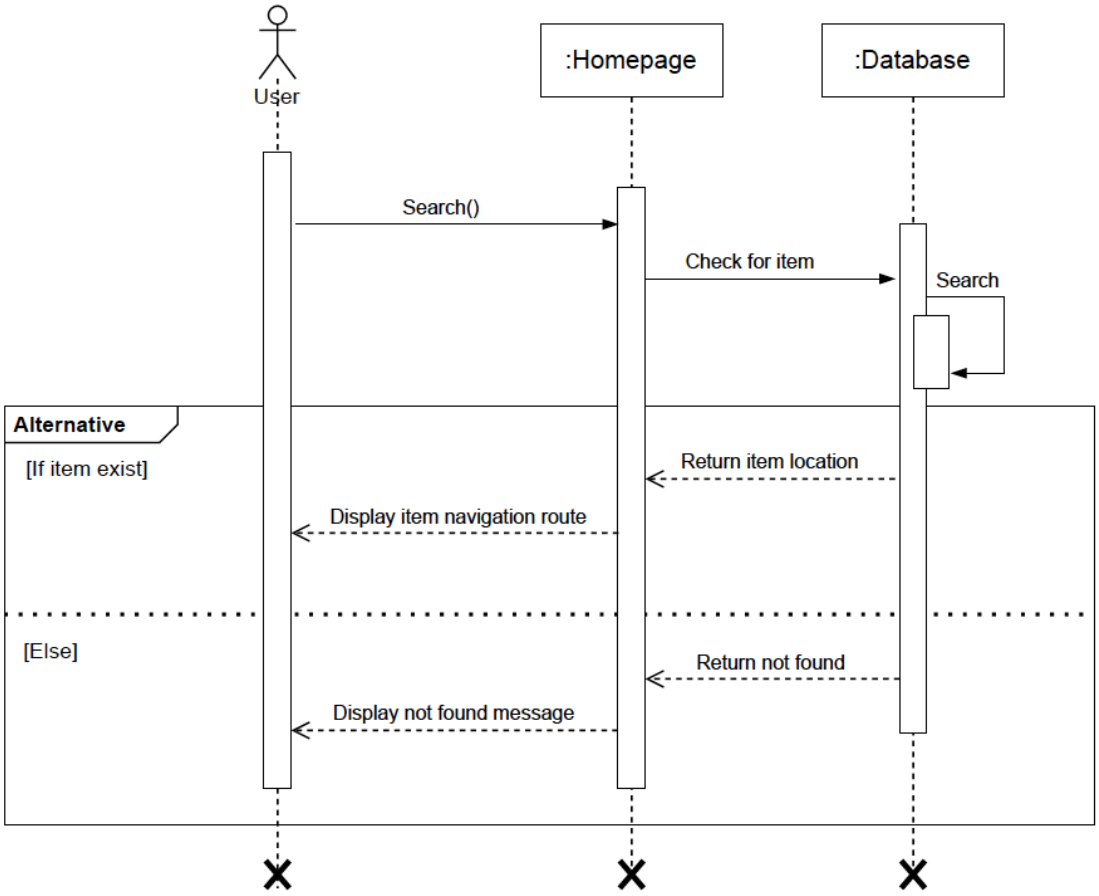


Figure 15: Search for Items Sequence Diagram.

Receive Shopping Suggestions Sequence Diagram

The Receive Shopping Suggestions sequence diagram describes how the user can receive shopping suggestions. The only way the system can offer the user a shopping suggestion is if it has stored the user's previous grocery lists in the database.

The user creates a grocery list using the writeList() method, and the app stores it in the database using the storeList() method. Then, the system analyzes the list to detect the more written items to provide the user with the names of these items as shopping suggestions.

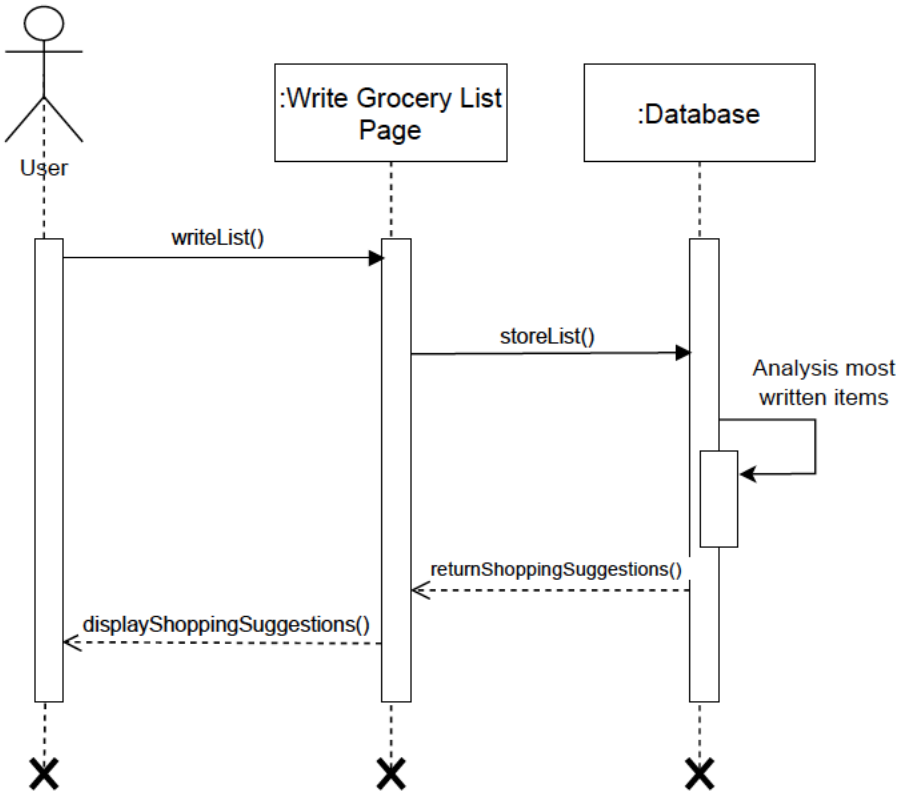


Figure 16: Receive Shopping Suggestions Sequence Diagram.

4.3 Data Model Design

4.2.1 ER Diagram

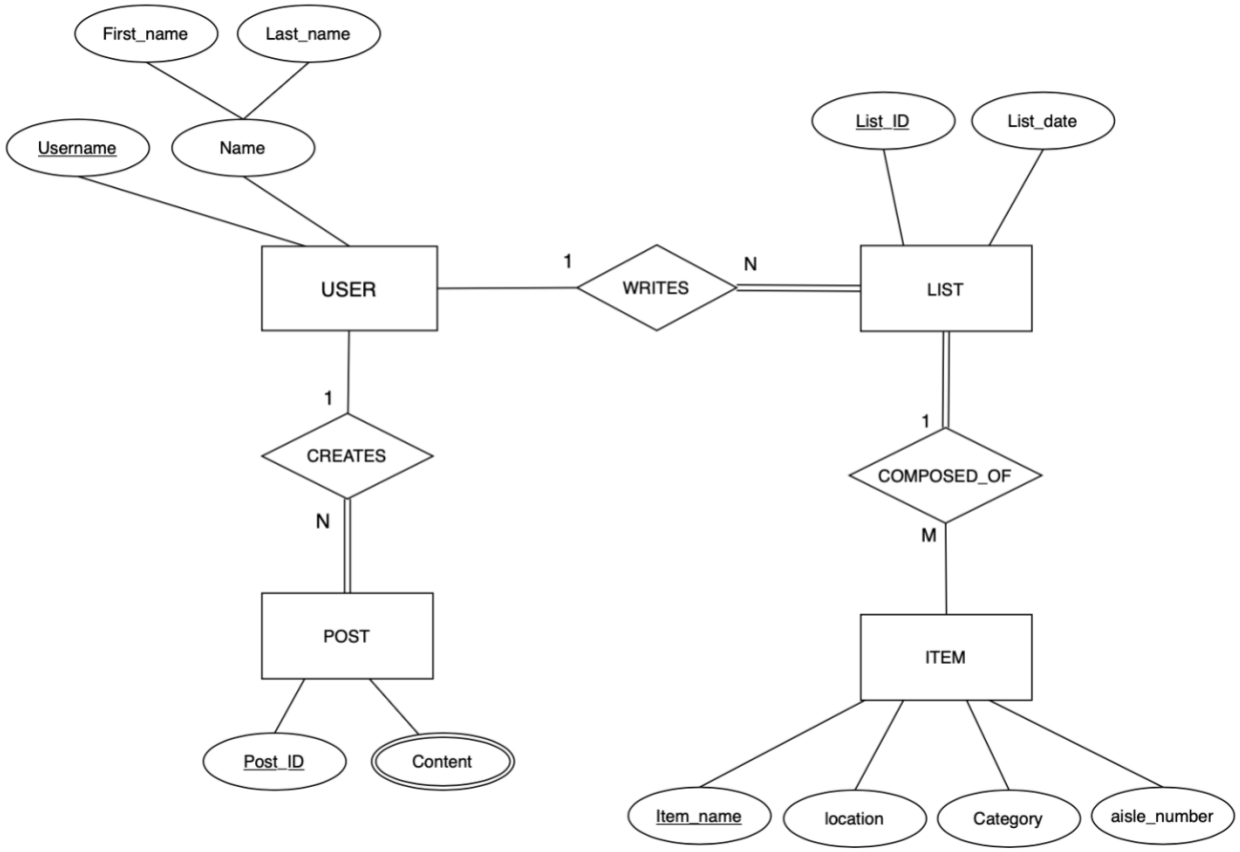


Figure 17: ER Diagram.

4.2.2 Relational Database Schema

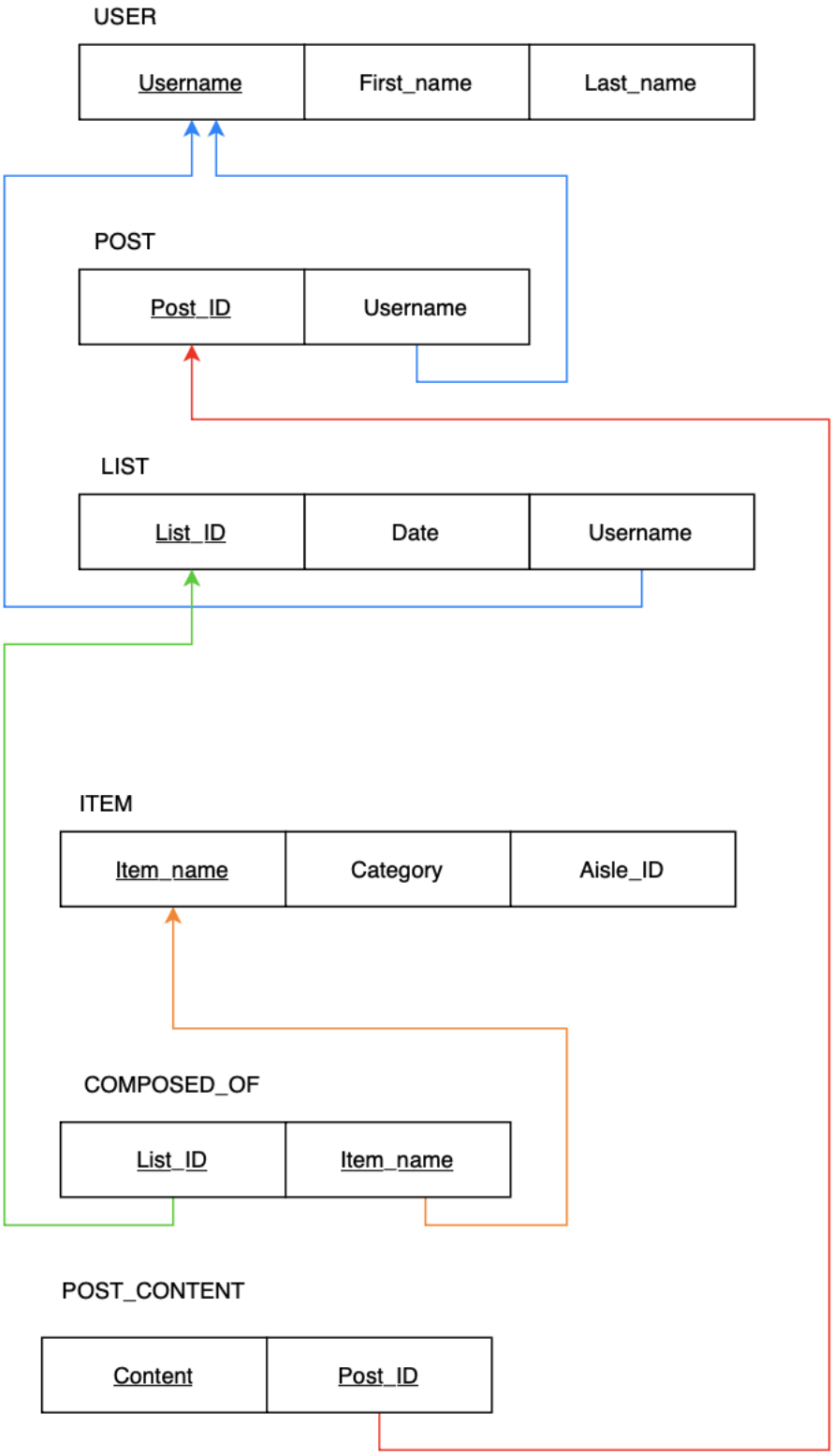


Figure 18: Relational Database Schema.

4.2.3 Normalized Database

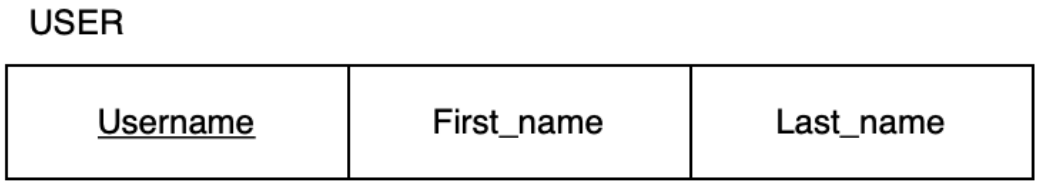


Figure 19: USER Relation.

1NF:

USER relation is in the first normal form because it does not have any multivalued or composite attributes.

USER [Username, First_name, Last_name]

2NF:

USER relation is in the second normal form because there are no partial dependencies.

USER [Username, First_name, Last_name]

3NF:

USER relation is in the third normal form because there are no transitive dependencies.

USER [Username, First_name, Last_name]

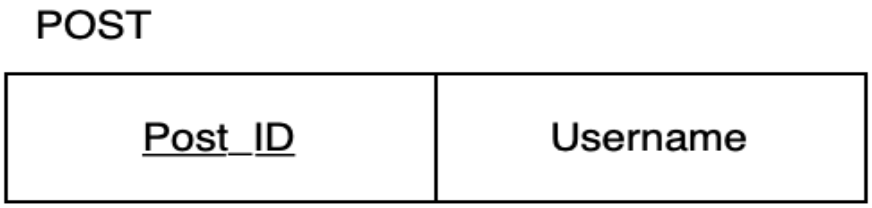


Figure 20: POST Relation.

1NF:

POST relation is in the first normal form because it does not have any multivalued or composite attributes.

POST [Post_ID, Username]

2NF:

POST relation is in the second normal form because there are no partial dependencies.

POST [Post_ID, Username]

3NF:

POST relation is in the third normal form because there are no transitive dependencies.

POST [Post_ID, Username]

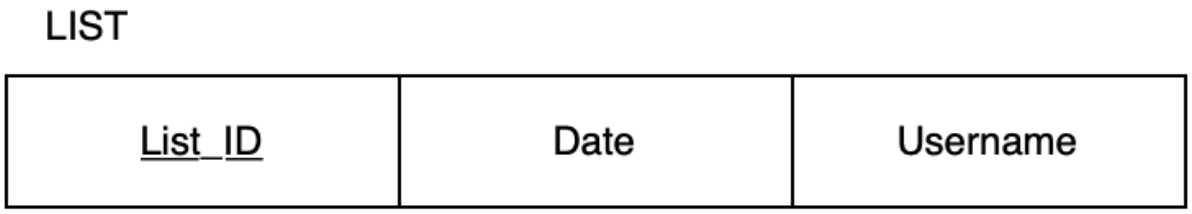


Figure 21: LIST Relation.

1NF:

LIST relation is in the first normal form because it does not have any multivalued or composite attributes.

LIST [List_ID, Date, Username]

2NF:

LIST relation is in the second normal form because there are no partial dependencies.

LIST [List_ID, Date, Username]

3NF:

LIST relation is in the third normal form because there are no transitive dependencies.

LIST [List_ID, Date, Username]

ITEM

Item_name	Category	Aisle_ID

Figure 22: ITEM Relation.

1NF:

ITEM relation is in the first normal form because it does not have any multivalued or composite attributes.

ITEM [Item_name, Category, Aisle_ID]

2NF:

ITEM relation is in the second normal form because there are no partial dependencies.

ITEM [Item_name, Category, Aisle_ID]

3NF:

ITEM relation is in the third normal form because there are no transitive dependencies.

ITEM [Item_name, Category, Aisle_ID]

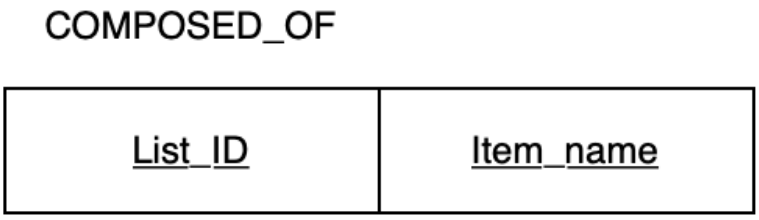


Figure 23: COMPOSED_OF Relation.

1NF:

COMPOSED_OF relation is in the first normal form because it does not have any multivalued or composite attributes.

COMPOSED_OF [List_ID, Item_name]

2NF:

COMPOSED_OF relation is in the first normal form because it does not have any multivalued or composite attributes.

COMPOSED_OF [List_ID, Item_name]

3NF:

COMPOSED_OF relation is in the third normal form because there are no transitive dependencies.

COMPOSED_OF [List_ID, Item_name]

POST_CONTENT

Content	Post_ID

Figure 24: POST_CONTENT Relation.

1NF:

POST_CONTENT relation is in the first normal form because it does not have any multivalued or composite attributes.

POST_CONTENT [Content, Post_ID]

2NF:

POST_CONTENT relation is in the first normal form because it does not have any multivalued or composite attributes.

POST_CONTENT [Content, Post_ID]

3NF:

POST_CONTENT relation is in the third normal form because there are no transitive dependencies.

POST_CONTENT [Content, Post_ID]

Chapter 5: Implementation

The implementation phase involved two approaches to developing GROCAFAST. The first approach utilized IndoorAtlas[19] technology to provide indoor positioning and location-based services. The second approach used the shortest path algorithm (Dijkstra) to optimize the shopping route within the store.

The first approach failed to achieve the desired results due to integration problems. As a result, the second approach was employed. Both approaches required a floor plan of the store, that was created using CubiCasa[20] , which is a tool used for creating high accuracy floor plans.

This section presents the details of the implementation process for both approaches, including the challenges encountered and the methods used to overcome them. The results of the implementation phase are also presented, providing insight into the success of the Dijkstra algorithm in achieving the project goals.

5.1 Creating Floor Plan

The local grocery store located in the female dormitory at King Abdulaziz University did not have a floor plan or any type of documentation that could demonstrate the store layout. Therefore, we used CubiCasa to create a floor plan.

CubiCasa allows developers to acquire floor plans to easily create indoor positioning applications. It provides highly accurate floor plans with precise measurements. (Figure 25) demonstrates the resulted floor plan (The blue rectangle shows where the grocery store is located).

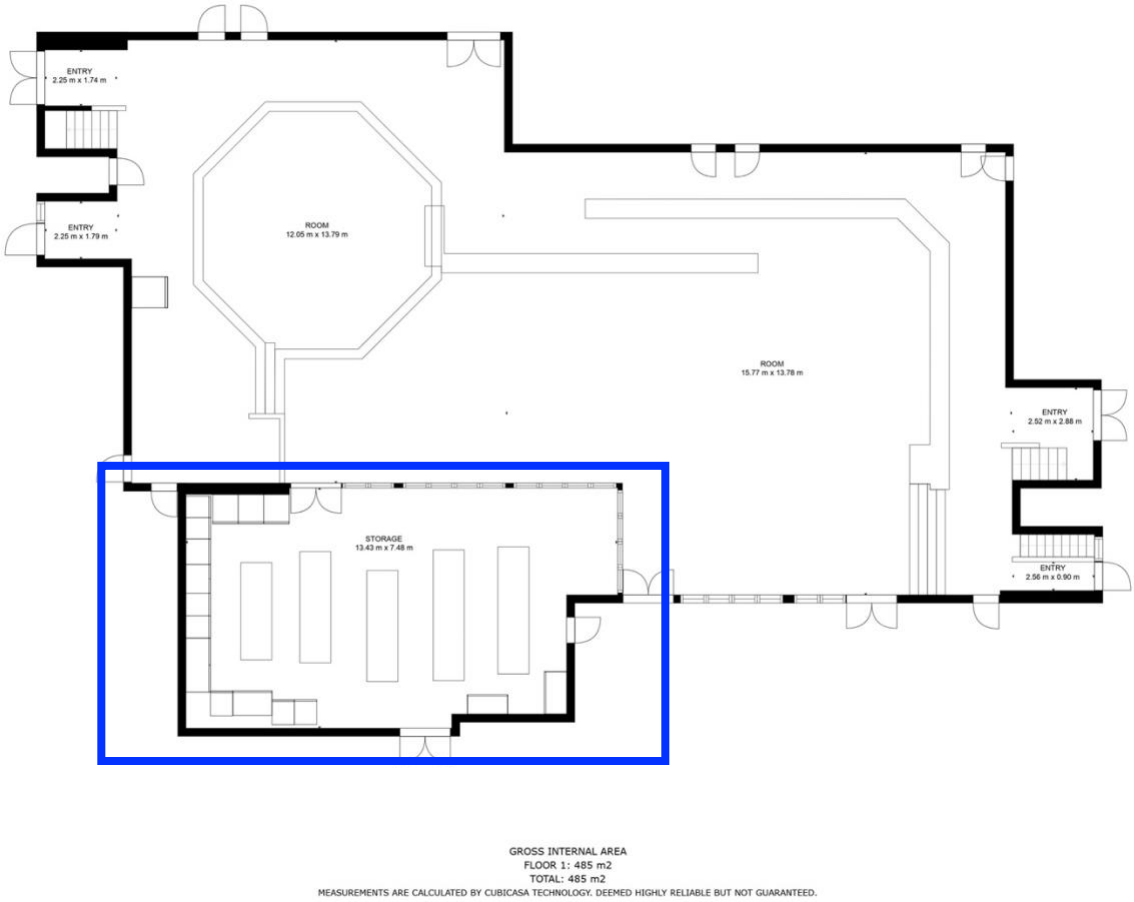


Figure 25: Floor Plan

Then, we improved the floor plan by including labels on each aisle of the store. (Figure 26).

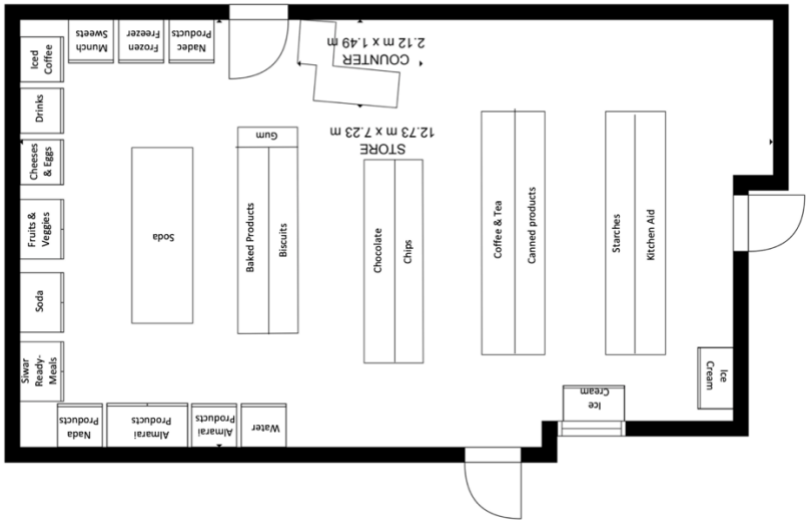


Figure 26: Labeled Floor Plan

5.2 First Approach (IndoorAtlas)[19]

IndoorAtlas is a technology company specialized in indoor positioning and location-based services. They use magnetic positioning technology to provide accurate location information inside buildings, where GPS signals do not exist. IndoorAtlas' technology can determine the position of a device using a unique algorithm that can interpret the magnetic signals in a building to create a map of the indoor space and determine the location of a device within that space.
The use of IndoorAtlas can help to achieve GROCAFAST goals by providing accurate indoor positioning and location-based services that can help customers conveniently navigate inside the store.

5.2.1 Tools

To use IndoorAtlas, users need to have access to a smartphone. Additionally, developers will need to integrate the IndoorAtlas SDK, which is provided by the company into their own applications.

5.2.2 IndoorAtlas Steps

5.2.2.1 Select location and add floor plans.

We selected the store's location, which is building 3 at the female dormitory and uploaded the floor plan of the floor where the store is located.

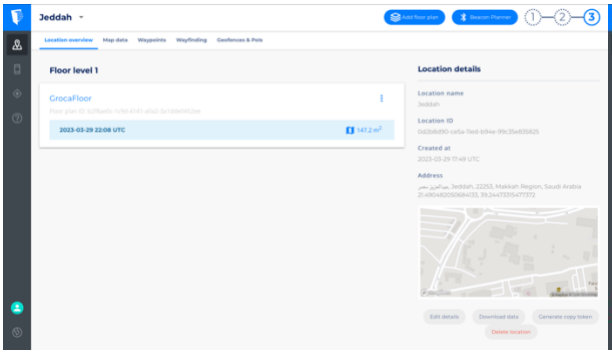


Figure 27: Select location (IndoorAtlas).

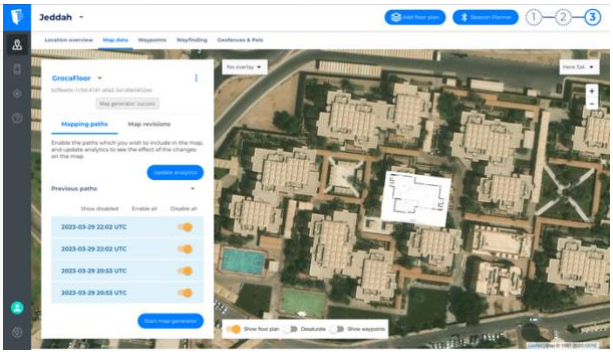


Figure 28: Add floor plans (IndoorAtlas).

5.2.2.2 Set up the indoor positioning technology.

We used six beacon sensors -each has a unique ID- as the positioning technology. First, we planned its locations in the floor plan using Beacon Planner BETA -a tool provided by IndoorAltas-. (Figure 29). Then, we installed it manually at the store in the specified spots.

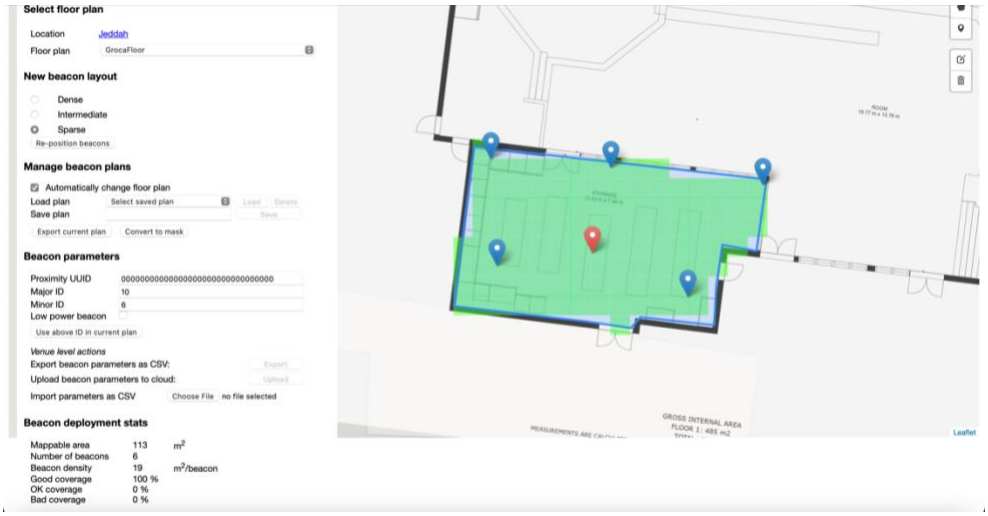


Figure 29: Setting Beacons (IndoorAtlas).

5.2.2.3 Add waypoint and fingerprinting.

Waypoints and fingerprints are added to improve the accuracy of indoor positioning. Waypoints are specific locations in the indoor space that have been manually labeled to provide reference points when determining the location of a device.

We have added the waypoint at the start and end of each aisle (Figure 30). Then, we fingerprinted the store using MapCreator -an application provided by IndoorAtlas- which is the process of recording the Beacon signal while walking from one waypoint to another (Figure 31). Finally, we uploaded the recorded path on the IndoorAtlas web tool.

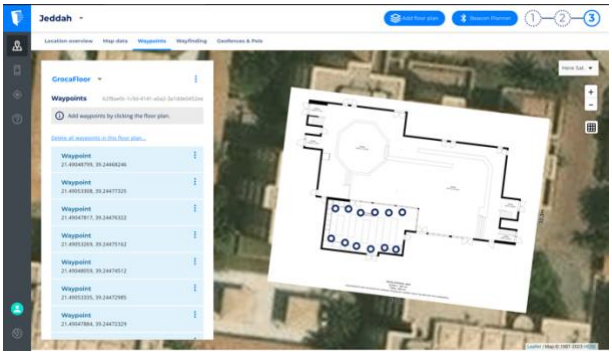


Figure 30: Adding waypoint (IndoorAtlas).

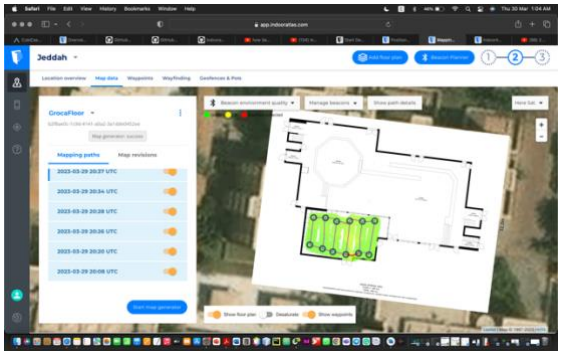


Figure 31: Fingerprints (IndoorAtlas).

5.2.2.4 Test positioning

After the whole store has been fingerprinted, we tested the positioning performance throughout the whole store using the MapCreator app's positioning mode to ensure the accuracy of the positioning. The testing involved recording ground truth paths and replaying the trace ids with a unique session to analyze the positioning performance, and whether all the installed beacons have been detected or not. Using the trace id of the session we can view a summary of the session including the quality of Beacon signals, in addition, we can detect any errors and replay the session for reviewing. (Figure 32, 33, 34).

Figure 32: Test positioning (IndoorAtlas).

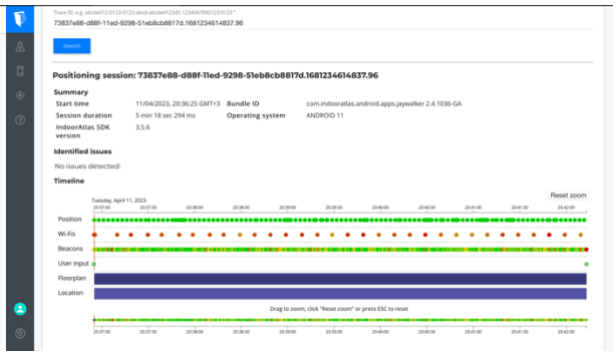


Figure 33: Test positioning (IndoorAtlas).

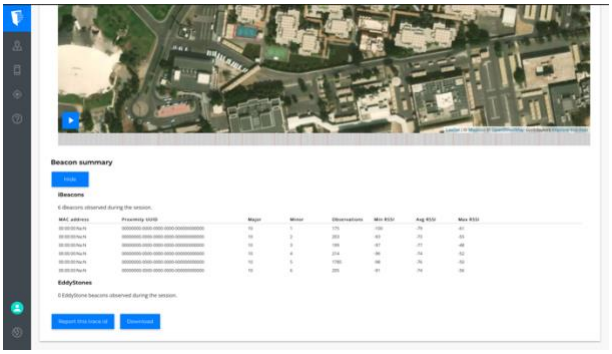


Figure 34: Test positioning (IndoorAtlas).

5.2.2.5 Integrate IndoorAtals API with GROCAFAST

After successfully completing all the necessary steps for using IndoorAtlas, the last step is the integration. However, integrating IndoorAtlas into GROCAFAST -which is developed using Android Studio- was more challenging than expected. The documentation provided by IndoorAtlas was limited, which made the integration process complex and time-consuming. Despite our best efforts, the integration was ultimately unsuccessful. We encountered several technical issues that we were unable to resolve without additional support from IndoorAtlas.

5.3 Second Approach (Dijkstra Shortest Path Algorithm)

After conducting a thorough evaluation of the first approach, we determined that the first approach was not successful in achieving our desired results. Therefore, we decided to explore alternative solutions and developed a second approach utilizing the shortest path algorithm (Dijkstra). The development of a second approach highlights the importance of being flexible and adaptable during the project and the willingness to pivot to alternative solutions when necessary to achieve the desired results.

Time is the most valuable thing in our lives, and GROCAFAST aims to save the customer's time and effort inside the grocery store. Shortest path algorithms have been used to solve the problem of finding the shortest path for customers inside the grocery store. The shortest path algorithms are a family of algorithms designed to solve the shortest path problem.

The most important problems that the shortest path algorithms solve are:

- Single-source shortest path (or SSSP) problem requires finding the shortest path from a source node to all other nodes with non-negative edge weight.
- Breadth-first search (or BFS) is finding the shortest path from a source node to all other nodes in an undirected and unweighted graph.
- All-pairs shortest path (or APSP) problem requires finding the shortest path between all pairs of nodes in a graph.

The most important algorithms for solving these problems are:

- Dijkstra's algorithm
- Bellman–Ford algorithm
- A* search algorithm
- Floyd–Warshall algorithm
- Johnson's algorithm
- Bidirectional search

Although all shortest path algorithms achieve their main goal which is finding the shortest path between point A and B or between point A and all other points in the graph, there are some variations that help the user distinguish which algorithm to use depending on the software developed.

In this project, Dijkstra's original algorithm is the appropriate shortest path algorithm due to the following reasons:

- It can be used with non-negative edge weights which means that we ignored any factors that obstruct the movements inside the store.
- It has a single source node which is the grocery store entrance point.
- It stores and updates the shortest path each time instead of storing all paths which results in reducing the time and effort.
- It focuses on finding the shortest path between two given nodes from the source node to all other nodes in the graph.
- It can be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined and which is our main target.

5.3.1 Implementation of Dijkstra's in GROCAFAST

This section demonstrates the implementation of Dijkstra in four classes (ShortestPath, ListAcitivity, GetAllPermulation, and Items) that are built to implement shortest path algorithm. The followings are descriptions of each class.

5.3.1.1 ShortestPath.java

This code defines an Android application that implements a pathfinding algorithm to optimize a shopping list. The main class, ShortestPath, contains the algorithm that calculates all possible paths between a set of shopping items and determines the shortest path that visits all items. The map is represented as a two-dimensional Mat object, which is initialized by setting certain pixels to specific colors using the OpenCV library. To calculate the shortest path between two points on the map, the code provides a method called get_shortest_path_length, which takes two Point objects as arguments and returns the length of the shortest path between the two points. The method resets some variables related to the pathfinding process and calls the find_paths_tree method, which generates all possible paths between the two points and stores them in the paths array. The find_paths_tree method uses a recursive algorithm to explore all possible paths and stores them in the paths array. After all possible paths have been generated, the method loops through the paths array to find the shortest path by checking the length of each path that ends at the destination point and selecting the one with the shortest length. The implementation of the get_shortest_path_length method depends on the find_paths_tree method, which in turn depends on the get_surrounding_black and is_point_in_points methods. These methods are used to explore the map and find all possible paths between the two points. The code also creates a 2D array called distances_matrix, which stores the distance between each pair of items in the shopping list, as well as the distance from the entrance to each item and from each item to the exit. The distances are calculated using the get_shortest_path_length method. The activity has a Button and an ImageView, which correspond to a camera button and an image view, respectively. The Bitmap object captured by the camera is displayed in the ImageView. The activity also has a TextView, which is not currently used.

Finally, the code draws the shortest path on a map, along with the items in the shopping list and the entrance and exit and scales the resulting image using OpenCV. The resulting image is then converted to a bitmap format.

Overall, this code segment combines pathfinding and image processing techniques to optimize a shopping list by finding the shortest path that visits all items. It appears to be a prototype or early version of an Android app for grocery shopping with functionality for displaying a map and calculating paths between items.

5.3.1.2 ListActivity.java

This Android app activity, called ListActivity, allows users to create a shopping list and view it as a list in the app. They can add new items to the list using an EditText and "Add" button and remove items by long-pressing on them. When an item is clicked, it is struck through to indicate completion. ListActivity uses Firebase Realtime Database to store and retrieve the shopping list items and their corresponding locations in the grocery store. The data is stored under the "Items" node in the Firebase database.The activity layout includes a ListView to display the shopping list items, an EditText to add new items, and several buttons and icons. The "Home" ImageView returns the user to the app's home page, the "Path" ImageView calculates the shortest path to retrieve the items from the store, and the "Search" Button searches for items.ListActivity extends the AppCompatActivity class and overrides the onCreate method to set up the user interface and event listeners. It includes several helper methods, such as setUpListViewList, which sets up the long click listener to remove an item from the list, and addItem, which adds a new item to the list view. Additionally, ListActivity includes openActivityHome, openActivitySearch, and openActivityPath methods to launch other activities when the user clicks the corresponding buttons or icons. The openActivityPath method calculates the shortest path to retrieve the items from the store and launches the ShortestPath activity.

5.3.1.2 Items.java

This class represents an item in a grocery store and contains information such as the item name, category, and aisle number.The class has several constructors, including a default constructor that sets the item name to an empty string and the x and y coordinates to 0, a constructor that takes only an item name as an argument, and a constructor that takes an item name, category, and aisle number as arguments.There is also a constructor that takes all of the above arguments as well as x and y coordinates. This constructor is used to create an instance of an item with all of its information, including its location on a map.The class also includes getters and setters for each of the instance variables, allowing the information to be accessed and modified outside of the class.

Overall, the Items class provides a way to represent and store information about items in a grocery store, including their location on a map.

5.3.1.3 getAllPermulation.java

provides a static method called getPermutations that generates all possible permutations of an input array of integers. The method takes two arguments: the input array and its factorial, which is used to determine the number of permutations that need to be generated.The method creates a two-dimensional array called possibilities to store all the permutations. It initializes the row variable to 0 and then calls the helper method, passing in the input array, the position of the current element, and the factorial.The helper method is a recursive function that generates all possible permutations of the input array. It swaps the current element with every subsequent element and recursively calls itself with the next position in the array. Once it reaches the end of the array, it adds the current permutation to the possibilities array and increments the row variable.

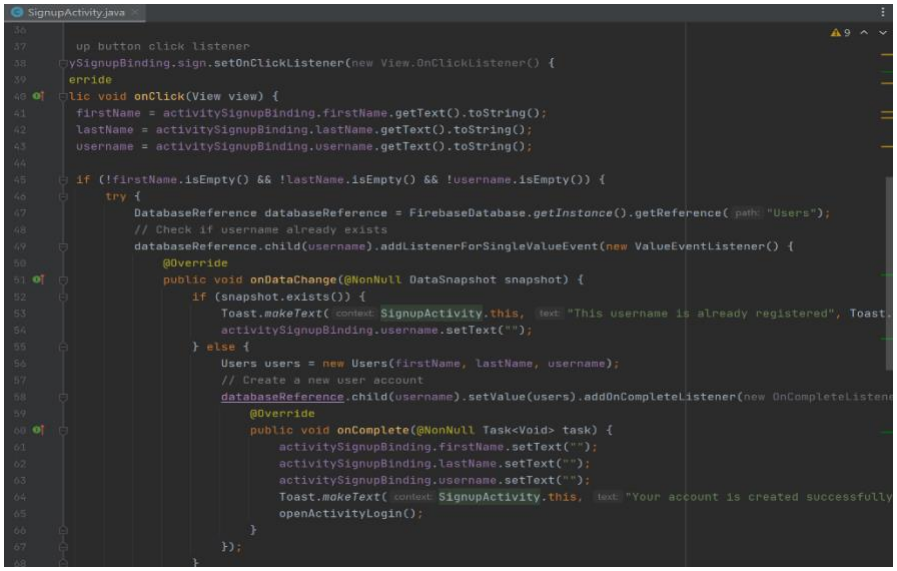
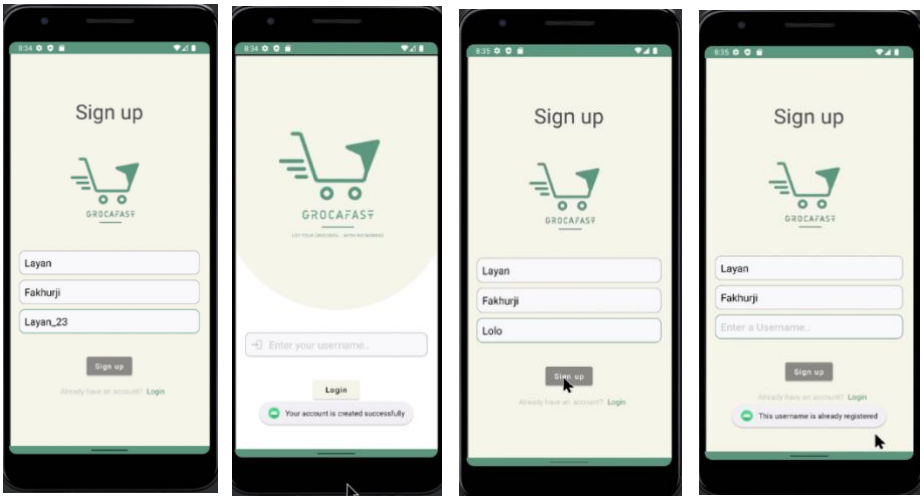
Finally, the getPermutations method returns the possibilities array containing all possible permutations of the input array.

Overall, the GetAllPermutations class provides a useful utility for generating all possible permutations of an input array, which can be helpful in a variety of applications such as pathfinding algorithms, sorting algorithms, and optimization problems.

5.3.2 Implementation of GROCAFAST functions

In the following section, we will describe the implementation of the functions of GROCAFAST. The functions are listed in the same flow as the functional requirements outlined in Chapter 3. Each function will be accompanied by a description, code snippet, and result to provide a comprehensive overview of how each function works and how it contributes to the overall functionality of GROCAFAST.

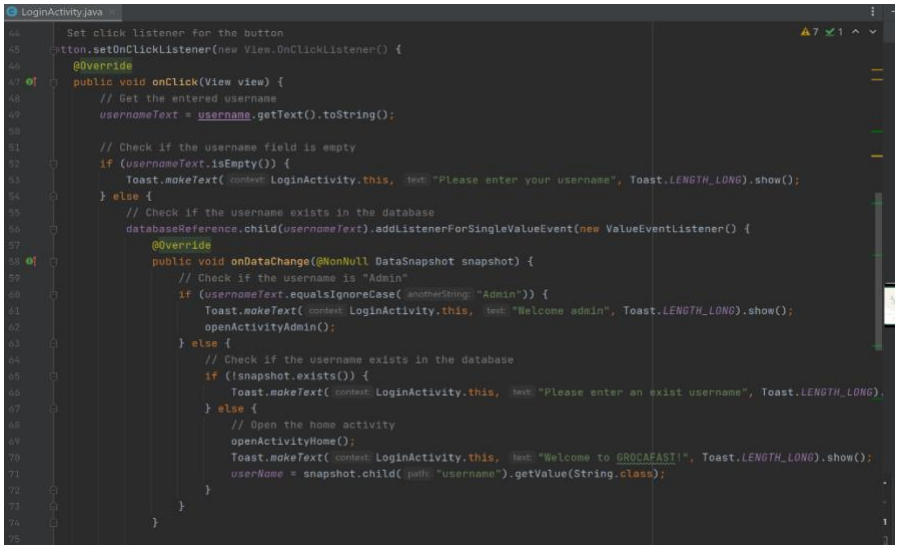
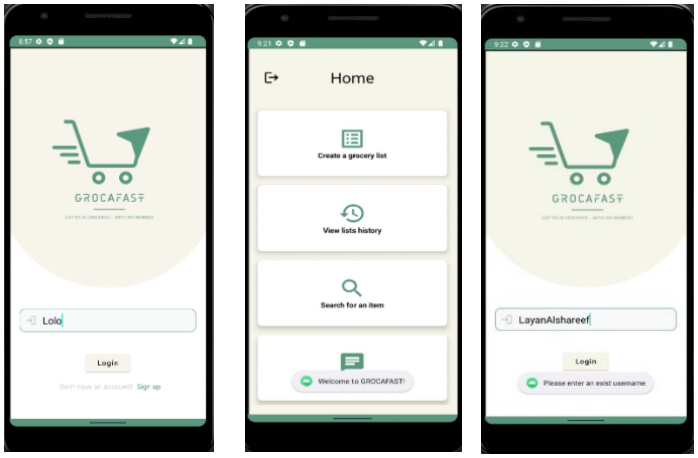
5.3.2.1 Create a new account.

Description	The user enters a username to register in the application. In a successful case, the user provides a unique username then he/she can access the application and use all its services. In case the user provides a username that already exists in the database, the application asks the user to try again with a different username.
Code Snippet	
Result	

```
SignupActivity.java
    up button click listener
ySignupBinding.sign.setOnClickListener(new View.OnClickListener() {
erride
lic void onClick(View view) {
 firstName = activitySignupBinding.firstName.getText().toString();
 lastName = activitySignupBinding.lastName.getText().toString();
 username = activitySignupBinding.username.getText().toString();

if (!firstName.isEmpty() && !lastName.isEmpty() && !username.isEmpty()) {
    try {
        DatabaseReference databaseReference = FirebaseDatabase.getInstance().getReference( path: "Users");
        // Check if username already exists
        databaseReference.child(username).addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if (snapshot.exists()) {
                    Toast.makeText( context: SignupActivity.this,  text: "This username is already registered", Toast.
                    activitySignupBinding.username.setText("");
                } else {
                    Users users = new Users(firstName, lastName, username);
                    // Create a new user account
                    databaseReference.child(username).setValue(users).addOnCompleteListener(new OnCompleteListene
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            activitySignupBinding.firstName.setText("");
                            activitySignupBinding.lastName.setText("");
                            activitySignupBinding.username.setText("");
                            Toast.makeText( context: SignupActivity.this,  text: "Your account is created successfully
                            openActivityLogin();
                        }
                    });
                }
```


Table 16: Create a new account implementation.

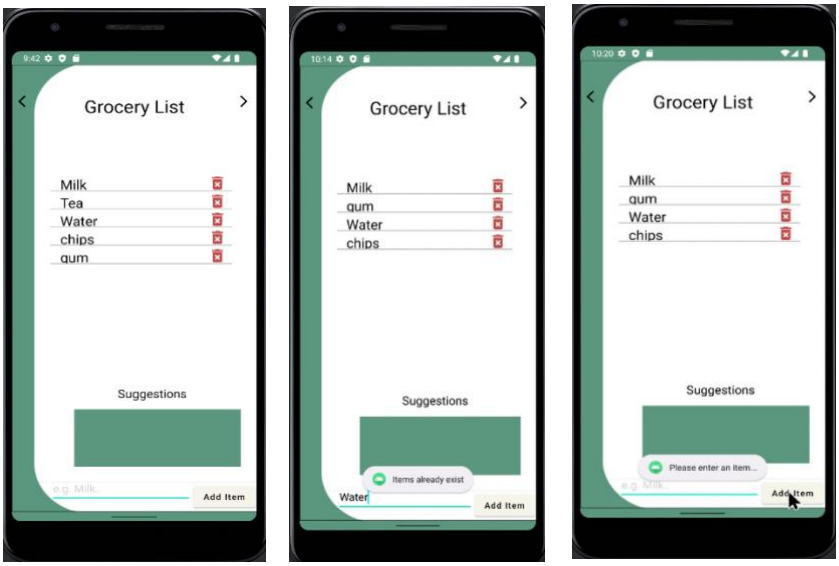
5.3.2.2 login to the system.

Description	The user logs into the system by providing the username. In a successful case, the user can access the application and use all its services. In case the user provides a wrong username that does not exist in the database, the application asks the user to try again with a different username.
Code Snippet	
Result	

```
LoginActivity.java
    Set click listener for the button
tton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // Get the entered username
        usernameText = username.getText().toString();

        // Check if the username field is empty
        if (usernameText.isEmpty()) {
            Toast.makeText( context: LoginActivity.this,  text: "Please enter your username", Toast.LENGTH_LONG).show();
        } else {
            // Check if the username exists in the database
            databaseReference.child(usernameText).addListenerForSingleValueEvent(new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot snapshot) {
                    // Check if the username is "Admin"
                    if (usernameText.equalsIgnoreCase( anotherString: "Admin")) {
                        Toast.makeText( context: LoginActivity.this,  text: "Welcome admin", Toast.LENGTH_LONG).show();
                        openActivityAdmin();
                    } else {
                        // Check if the username exists in the database
                        if (!snapshot.exists()) {
                            Toast.makeText( context: LoginActivity.this,  text: "Please enter an exist username", Toast.LENGTH_LONG).
                        } else {
                            // Open the home activity
                            openActivityHome();
                            Toast.makeText( context: LoginActivity.this,  text: "Welcome to GROCAFAST!", Toast.LENGTH_LONG).show();
                            userName = snapshot.child( path: "username").getValue(String.class);
                        }
                    }
                }
```


Table 17: Log in implementation

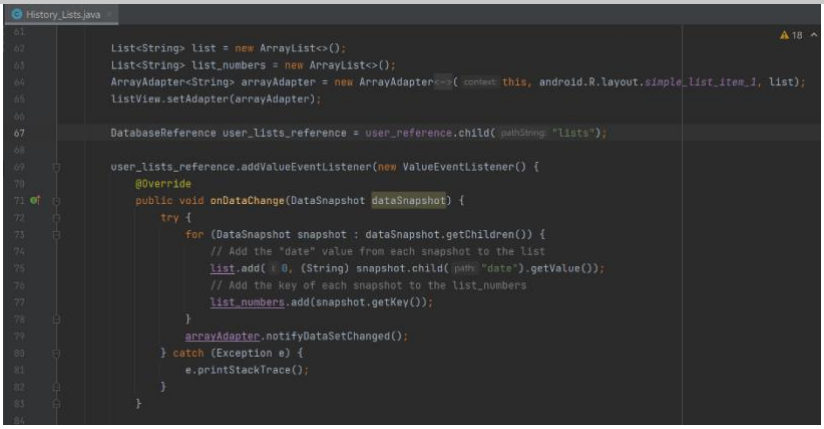
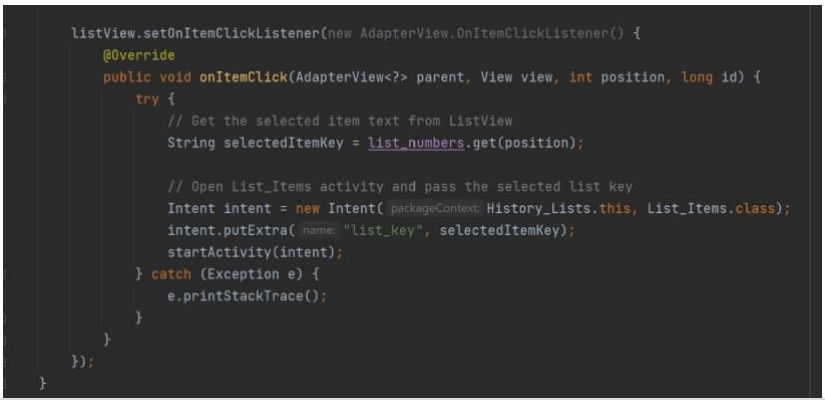
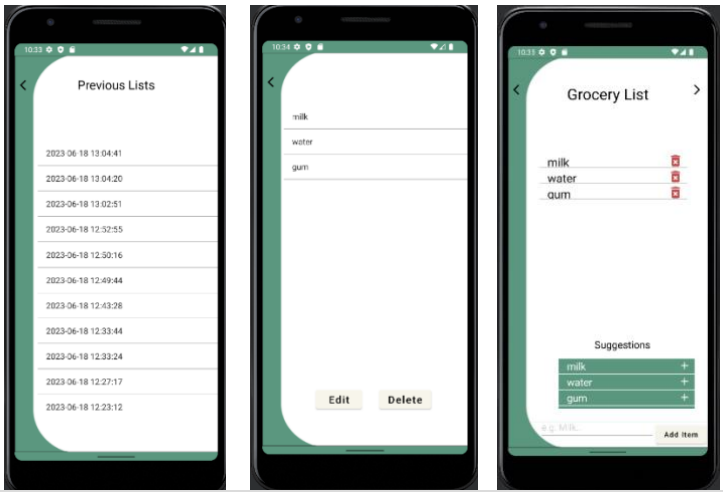
5.3.2.3 Create a grocery list.

Description	The user can create a grocery list on the application. In a successful case, the grocery list is stored in the database, and the user can proceed to display the sorted list. In case of a failure, if the user writes an exist item in the grocery list or does not write any item, error message displayed, and the user must try again.
Code Snippet	
Result	

```
//add item to the array list
1 usage
private void addItem(View view) {
    EditText input= findViewById(R.id.editText);
    String itemText= input.getText().toString();
    if(!(itemText.equals(""))){
        for (int c=0;c<items.size();c++) {
            if (items.get(c).equals(itemText)) {
                Toast.makeText(getApplicationContext(), text: "Items already exist", Toast.LENGTH_LONG).show()
                return;
            }
        }
        dataList.add(new DataList(itemText));
        adapter= new CustomAdapter(dataList,getApplicationContext());
        listView.setAdapter(adapter);
        items.add(itemText);
//          itemsAdapter.add(itemText);
        input.setText("");
    }
    else {
        Toast.makeText(getApplicationContext(), text: "Please enter an item...", Toast.LENGTH_LONG).show();
    }
}
```


Table 18: Create grocery list implementation.

5.3.2.4 Modify a previous list.

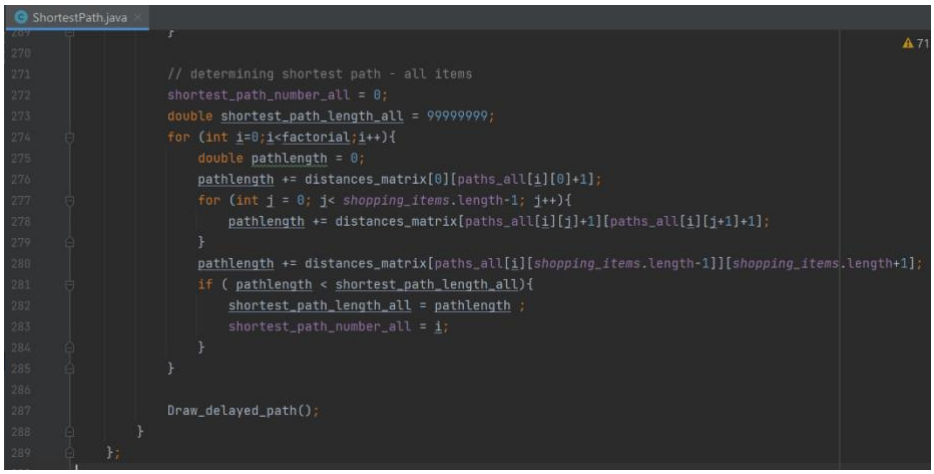
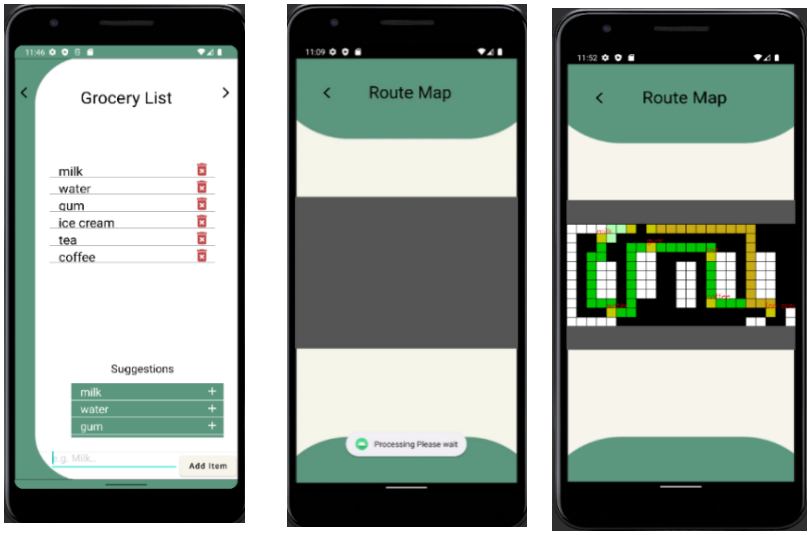
Description	The user can modify previous grocery lists (if any). In a successful case, the user can edit their grocery list, add more items, delete unwanted grocery lists, and then proceed to obtain the route map. In case of a failure, the database cannot retrieve the items, and the user must try again.
Code Snippet	 
Result	

```
List<String> list = new ArrayList<>();
List<String> list_numbers = new ArrayList<>();
ArrayAdapter<String> arrayAdapter = new ArrayAdapter<>( context: this, android.R.layout.simple_list_item_1, list);
listView.setAdapter(arrayAdapter);

DatabaseReference user_lists_reference = user_reference.child( pathString: "lists");

user_lists_reference.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        try {
            for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                // Add the "date" value from each snapshot to the list
                list.add( i: 0, (String) snapshot.child( path: "date").getValue());
                // Add the key of each snapshot to the list_numbers
                list_numbers.add(snapshot.getKey());
            }
            arrayAdapter.notifyDataSetChanged();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        try {
            // Get the selected item text from ListView
            String selectedItemKey = list_numbers.get(position);

            // Open List_Items activity and pass the selected list key
            Intent intent = new Intent( packageContext: History_Lists.this, List_Items.class);
            intent.putExtra( name: "list_key", selectedItemKey);
            startActivity(intent);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});
}
```


Table 19: Modify previous list implementation.

5.3.2.5 Generate a route map with the shortest path.

Description	The user can obtain a route map generated by the application based on the user's grocery list to navigate the user inside the grocery store. In a successful case, the route map is generated based on the user's grocery list to navigate the user inside the store. In case of a failure, the route map cannot be generated, and the user must try again.
Code Snippet	
Result	

```
// determining shortest path - all items
shortest_path_number_all = 0;
double shortest_path_length_all = 99999999;
for (int i=0;i<factorial;i++){
    double pathlength = 0;
    pathlength += distances_matrix[0][paths_all[i][0]+1];
    for (int j = 0; j< shopping_items.length-1; j++){
        pathlength += distances_matrix[paths_all[i][j]+1][paths_all[i][j+1]+1];
    }
    pathlength += distances_matrix[paths_all[i][shopping_items.length-1]][shopping_items.length+1];
    if ( pathlength < shortest_path_length_all){
        shortest_path_length_all = pathlength ;
        shortest_path_number_all = i;
    }
}

Draw_delayed_path();
}
};
```


Table 20: Generate a route map implementation.

5.3.2.6 Search for an item and view its location inside the store.

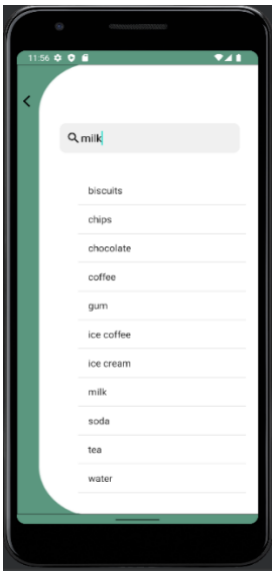
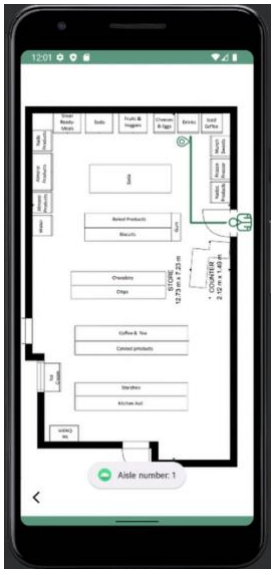
Description	When the user searches for an item, the application locates the item inside the store and displays its location to the user. In a successful case, the application locates the searched item inside the store and navigates the user to its location. In case of a failure, the item cannot be located, and the user must try again.
Code Snippet	SearchActivity.java ``` try { if (snapshot.exists()) { ArrayList<String> listitems = new ArrayList<>(); for (DataSnapshot ds : snapshot.getChildren()) { // Read item details from the snapshot and create an Items object Items item = new Items(ds.child(path: "itemName").getValue(String.class), ds.child(path: "c listitems.add(item.getItemName()); } ArrayAdapter arrayAdapter = new ArrayAdapter(getApplicationContext(), android.R.layout.simple listData.setAdapter(arrayAdapter); txtSearch.setOnItemClickListener(new AdapterView.OnItemClickListener() { @Override public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) { try { openMap(adapterView.getItemAtPosition(i).toString()); // Open the map for the sel } catch (Exception e) { e.printStackTrace(); } } }); } else { Toast.makeText(getApplicationContext(), text: "Item not found..!", Toast.LENGTH_LONG).show(); } ``` 21 2
Result	 

Table 21: Search for an item implementation.

5.3.2.7 Receive personalized shopping suggestions.

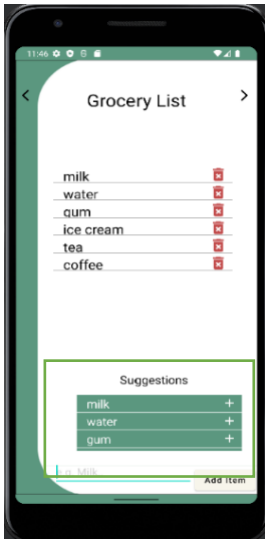
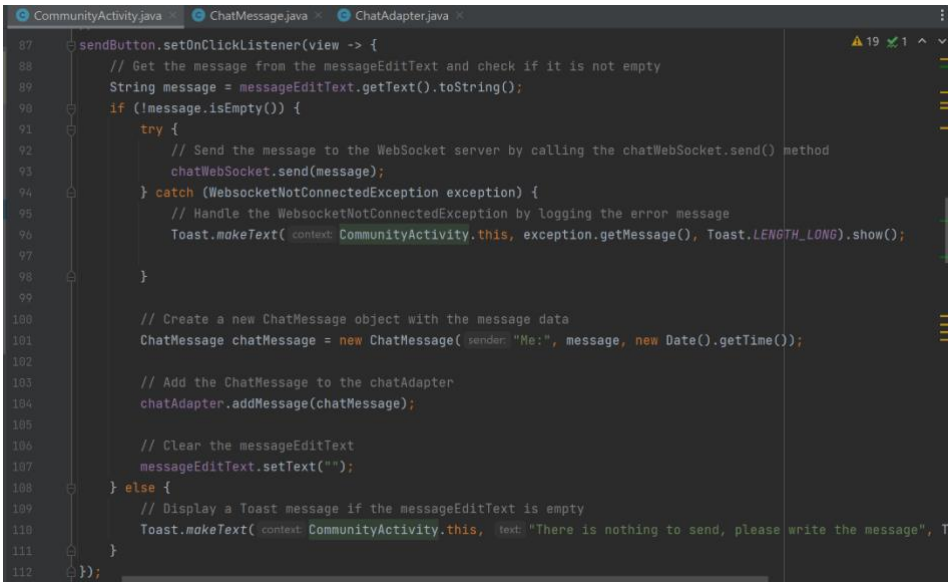
Description	Every time the user creates a grocery list on the application, the list is stored in the database and analyzed to detect the user's top three most purchased items in order to generate shopping suggestions for the user. In a successful case, the application displays the shopping suggestions to the user. In case of failure, the database does not retrieve the items that written by the user, and the application asks the user to rewrite the items again.
Code Snippet	ListActivity.java 223 @ public static String[] GetTOPThreeItems(int[] items_repeatation, String[] items_names){ 224 String[] top = new String[3]; 225 top[0]="";top[1]="";top[2]=""; 226 int[] topcount = new int[3]; 227 topcount[0]=0;topcount[1]=0;topcount[2]=0; 228 for (int j=0; j< items_repeatation.length ; j++) { 229 if (items_repeatation[j] > topcount[0]){ 230 top[0] = items_names[j]; 231 topcount[0] = items_repeatation[j]; 232 } 233 } 234 for (int j=0; j< items_repeatation.length ; j++) { 235 if (items_repeatation[j] > topcount[1] && !items_names[j].equals(top[0])){ 236 top[1] = items_names[j]; 237 topcount[1] = items_repeatation[j];}} 238 for (int j=0; j< items_repeatation.length ; j++) { 239 if (items_repeatation[j] > topcount[2] && !items_names[j].equals(top[0]) 240 && !items_names[j].equals(top[1])){ 241 top[2] = items_names[j]; 242 topcount[2] = items_repeatation[j]; 243 } 244 } 245 return top; 246 }
Result	

Table 22: Receive suggestions implementation.

5.3.2.8 Create posts in the online space between users of the same store.

Description	The user can create posts to share reviews in the space between the users of the same grocery store created by the application. In a successful case, the user's post is created and displayed in the space. In case of a failure, the user's post cannot be displayed on the space, and the application asks the user to try again.
Code Snippet	
Result	

```
sendButton.setOnClickListener(view -> {
    // Get the message from the messageEditText and check if it is not empty
    String message = messageEditText.getText().toString();
    if (!message.isEmpty()) {
        try {
            // Send the message to the WebSocket server by calling the chatWebSocket.send() method
            chatWebSocket.send(message);
        } catch (WebsocketNotConnectedException exception) {
            // Handle the WebsocketNotConnectedException by logging the error message
            Toast.makeText( context: CommunityActivity.this, exception.getMessage(), Toast.LENGTH_LONG).show();

        }

        // Create a new ChatMessage object with the message data
        ChatMessage chatMessage = new ChatMessage( sender: "Me:", message, new Date().getTime());

        // Add the ChatMessage to the chatAdapter
        chatAdapter.addMessage(chatMessage);

        // Clear the messageEditText
        messageEditText.setText("");
    } else {
        // Display a Toast message if the messageEditText is empty
        Toast.makeText( context: CommunityActivity.this, text: "There is nothing to send, please write the message",
    }
});
```


Table 23: Create posts implementation.

5.4. Code debugging and troubleshooting issues.

Collaborating on projects can be challenging. Here we mention some of the issues we faced throughout the project.

One issue we have encountered is an incompatibility between the Android Studio and SDK versions. This can cause problems when exchanging code, as the (build. gradle) file is the first file we run when we start a project. To overcome this issue and ensure that the project runs smoothly, we ensure that all team members are using the same version of Android Studio and the SDK.

Another issue we faced is with configuring Firebase, where the list of items was not being added to the database. To resolve this, we can double-check the configuration settings and make sure that we are passing the correct data to Firebase.

By addressing these issues, we improved the efficiency and productivity of our collaborative efforts.

Chapter 6: Testing

This chapter discusses the final phase of the traditional waterfall methodology, which is testing. In any project's life cycle, testing is a crucial component that cannot be overlooked. Software testing is particularly important in order to identify and eliminate any bugs that may arise and improve the overall user experience.

To ensure that GROCAFAST is working as intended, we performed six types of testing.

6.1. Unit Testing

Unit testing is the process of testing small components of the code. Unit tests typically run in an isolated environment, with mock data and services to test the unit's output. We have performed unit testing on two units in the code.

6.1.1. Test case 1: GetTOPThreeItems()

Functionality	Input	Expected result	Result
Check whether the method will return the user's top three most purchased items.	Two input arrays: one with integers for item repetitions and another with strings for the items. (tops[0],items_names[3]) (tops[1],items_names[4]) (tops[2],items_names[2])	The first item will be the most written item: (tops[0],items_names[3])	Pass

Table 24: Unit Testing, Test case 1.

Test case 1: Code

```
public class ListActivityTest {
    @Test
    public void Sorting_items_favorites() {
        int[] items_repeatation = new int[5];
        items_repeatation[0]=0;
        items_repeatation[1]=0;
        items_repeatation[2]=3;
        items_repeatation[3]=5;
        items_repeatation[4]=4;
        String[] items_names = new String[5];
        items_names[0] = "item0";
        items_names[1] = "item1";
        items_names[2] = "item2";
        items_names[3] = "item3";
        items_names[4] = "item4";
        String[] tops = ListActivity.GetTOPThreeItems(items_repeatation,items_name
        assertEquals(tops[0],items_names[3]);
        assertEquals(tops[1],items_names[4]);
        assertEquals(tops[2],items_names[2]);
    }
```

Figure 35: Unit Testing, Test case 1 code

Test case 1: Result

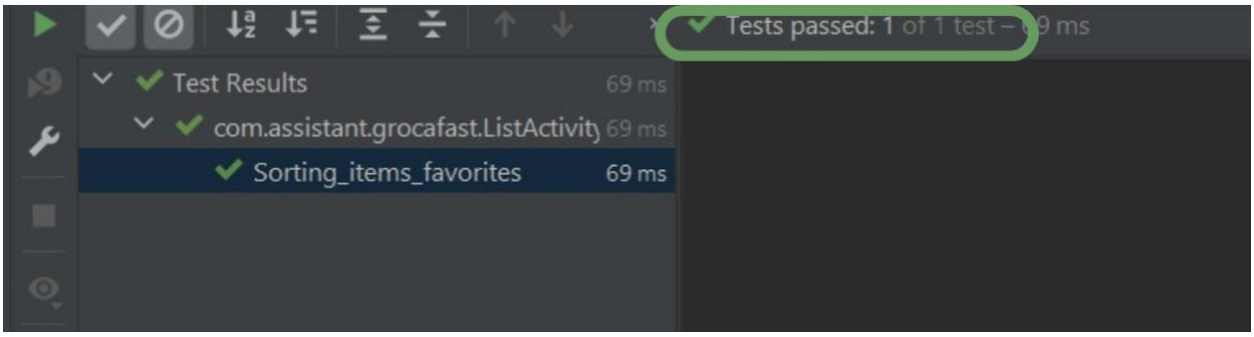


Figure 36: Unit Testing, Test case 1: Result.

6.1.2. Test case 2: getPermutations()

Functionality	Input	Expected result	Result
Check whether the method will return permutations of all paths for a given item.	Two inputs. The first array of integers to be tested and the second is the factorial according to a number of items.	The output from the getPermutations() method matches the expected value of both possible permutations of the input array.	Pass

Table 25: Unit Testing, Test case 2.

Test case 2: Code

```
@Test
public void testGetPer() {
    int[] inputArray = {1,2};
    int factorial = 2; // 2! = 4, since there are 2 elements in the input array

    int[][] expectedOutput = {
            {1,2},
            {2,1}
    };
    int[][] actualOutput = GetAllPermutations.getPermutations(inputArray, factorial
    assertArrayEquals(expectedOutput, actualOutput);
}
```

Figure 37: Unit Testing, Test case 2 code.

Test case 2: Result

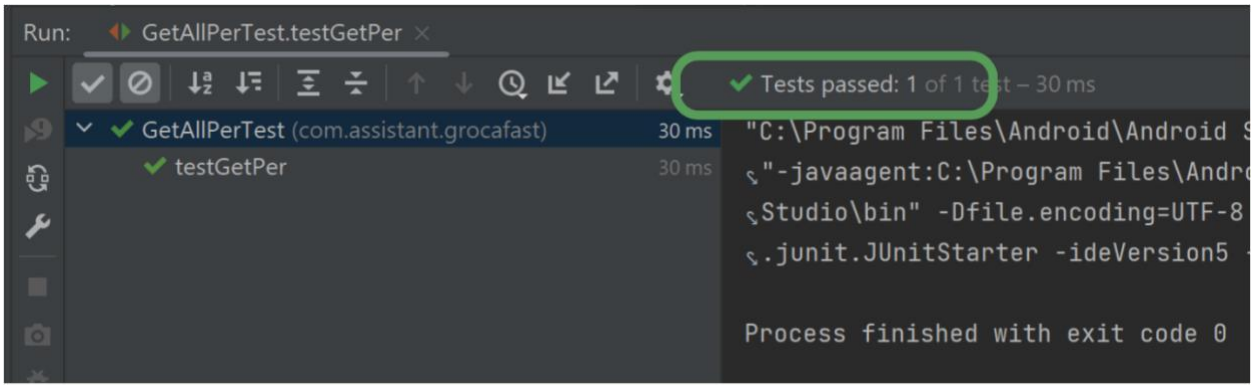


Figure 38: Unit Testing, Test case 2: Result.

6.2. Integration Testing

Integration testing combines and tests different components of the software to ensure that they interact together correctly. We performed two types of integration testing. The first one is the integration between two entities (Android studio and FireBase). The second one is the integration between the function and its related set of functions.

6.2.1. Integration between Android studio and FireBase

The first step is creating dependencies from the Firebase in Android Studio, then creating a database reference object to reference the user in the database. Finally, we apply the testing using any method that requires an update or retrieval from the database. We chose the sign-up method.

Figure 39: Integration Testing Dependencies.

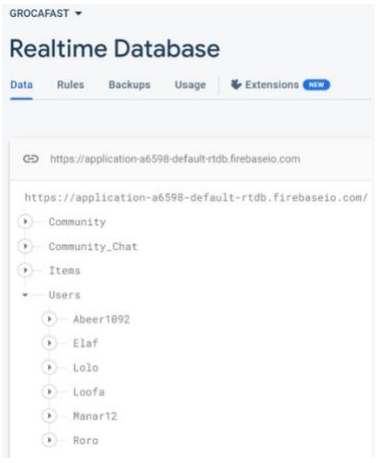
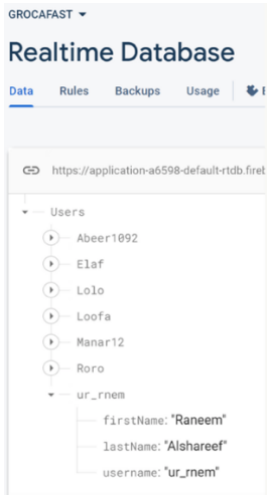
Before	User Sign-Up	After
		Database updated successfully. 

Table 26: Integration testing between Android studio and Firebase.

6.2.2. integration between the function and its related set of functions.

Test Case Scenario	Link between pages
Create a grocery list: Create a new account → login → create a grocery list.	Verified
View and modify a previous list: Login → View lists history → View and modify a previous list.	Verified
Obtain the shortest path: Login → Create a grocery list → Obtain the shortest path.	Verified
Search for an item: Login → Search for an item → View item location.	Verified
Receive personalized shopping suggestions: Login → (Have list history) → Create a grocery list → Receive personalized shopping suggestions.	Verified
Create posts: Login → Community space → Create posts.	Verified

Table 27: integration testing between the function and its related set of functions.

6.3. System Testing

System testing is a type of software testing that evaluates the complete system to ensure that it meets the specified requirements. We tested several scenarios that covered all the application features for the two types of users (Shopper and Admin). This type of testing did not involve coding.

Shopper

Test case scenarios	Result
View splash screen – signup valid username - login - view home page	**Pass**
View splash screen – login incorrect username - login correctly - view home page	**Pass**
View splash screen – login correct username - login correctly - view home page	**Pass**
View splash screen – signup invalid username - signup correctly - login correctly - view home page	**Pass**
View splash screen – login - view home page – create grocery list service – add valid items – get route map	**Pass**
View splash screen – login - view home page – create grocery list service – add invalid items	**Pass**
View splash screen – login - view home page – view lists history service – select previous list – edit list – get route map	**Pass**
View splash screen – login - view home page – view lists history service – select previous list – delete list	**Pass**
View splash screen – login - view home page – search service - search for an item – route map	**Pass**
View splash screen – login - view home page – search service - search for an undefined item – Search for valid item – get route map	**Pass**
View splash screen – login - view home page – community space service – write comment	**Pass**

Table 28: System testing for shopper.

Admin

Test case scenarios	Result
View splash screen – login incorrect username - login correctly - view home page	**Pass**
View splash screen – login correct username - login correctly - view home page	**Pass**
View splash screen – login - view home page – add items service – fill item fields correctly – add item	**Pass**
View splash screen – login - view home page – add items service – fill item fields invalid input – fill item fields correctly - add item	**Pass**
View splash screen – login - view home page – delete items service – fill item fields correctly – remove item	**Pass**
View splash screen – login - view home page – delete items service – fill item fields invalid input – fill item fields correctly - remove item	**Pass**
View splash screen – login - view home page – view statistics service	**Pass**

Table 29: System testing for admin.

6.4. Compatibility Testing

We tested GROCAFAST on three Android devices. The results indicated that the application is fully operational on all three devices. This type of testing did not involve coding.

Device	Android version	Memory space	Memory available	Pass/Fail
Google pixel 2 XL	11	4 GB	612 MB	Pass
Galaxy A10	11	2 GB	314 MB	Pass
ZTE Blade 10 Smart	9	4 GB	1 GB	Pass

Table 30: Compatibility testing.

6.5. Usability Testing

To evaluate the usability of GROCAFAST, we conducted usability testing using a predetermined set of steps. This type of testing did not involve coding.

6.5.1. Identify the project purpose.

The purpose of the project is to provide customers with the shortest path to collect all items on their grocery list.

6.5.2. Identify the intended users.

12 customers of the local grocery store. 9 students, 3 dormitory administration employees. The aggregate number of participants was aged 20 to 36 years old.

6.5.3. Test Environment

The usability testing is conducted in the building where the local grocery store is located.

6.5.4. Define tasks (features)

1. The user shall be able to create a new account.
2. The user shall be able to login to the system.
3. The user shall be able to create a grocery list.
4. The user shall be able to view and modify a previous list (if any).
5. The user shall obtain the shortest path to collect all items on the grocery list and navigate inside the store using a route map.
6. The user shall be able to search for an item and view its location inside the store.
7. The user shall be able to receive personalized shopping suggestions.
8. The user shall be able to create posts in the online space between users of the same store.

6.5.5. Criteria Of Measuring Usability

The Intended users have been asked to fill a form containing the criteria of measuring usability.

Measure	Strongly Agree	Agree	Disagree	Strongly Disagree
Learnability: How easy is it for users to accomplish basic tasks the first time they encounter the design?				
Efficiency: Once users have learned the design, how quickly can they perform tasks?				
Memorability: When users return to the design after a period of not using it, how easily can they reestablish proficiency?				
Errors: Can users easily recover from the errors?				
Satisfaction: How pleasant is it to use the design?				

Table 31: Criteria of measuring usability testing.

6.5.6. Analyze the acquired information.

Please note that: 1 →Strongly Agree, 2 → Agree, 3 → Disagree, and 4 → Strongly Agree.

Figure 40: Usability Testing, Task 1 Analysis.

Figure 41: Usability Testing, Task 2 Analysis.

Figure 42: Usability Testing, Task 3 Analysis.

Figure 43: Usability Testing, Task 4 Analysis.

Figure 44: Usability Testing, Task 5 Analysis.

Figure 45: Usability Testing, Task 6 Analysis.

Figure 46: Usability Testing, Task 7 Analysis.

Figure 47: Usability Testing, Task 8 Analysis.

6.6. Performance Testing

After completing the implementation, we conducted several experiments to test the performance of GROCAFAST and ensure that it met our objective, which is saving the shoppers' time by providing the shortest path for each shopper to collect the items on their grocery lists, which will ultimately enhance the grocery shopping experience.

The experiment focused on calculating the time taken to collect several items using a pre-written grocery list containing 7 Items. On the first visit, we used the note app on the iPhone to create the grocery list. It took 6 minutes and 31 seconds to collect all the items on the list. On the second visit, we used GROCAFAST to create the same grocery list and obtain the shortest path. This time, it took 4 minutes and 6 seconds to collect all the items on the list.

We repeated the experiment 3 times to ensure accuracy. The following table summarizes each experiment.

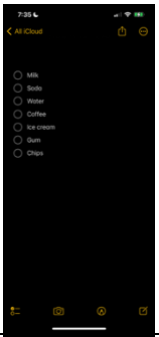
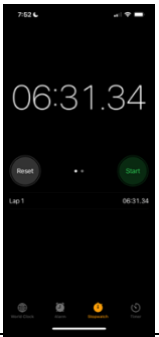
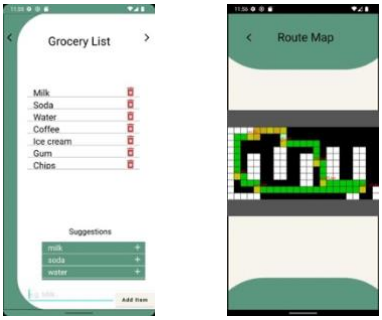
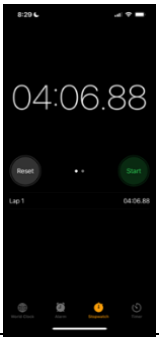
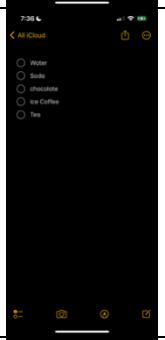
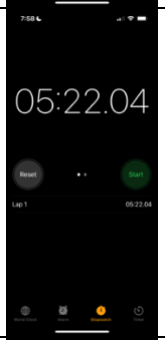
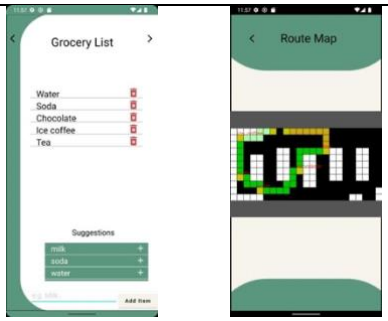
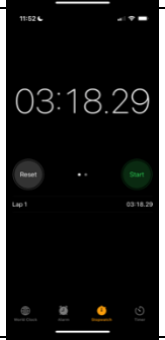
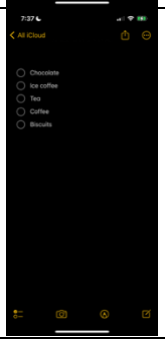
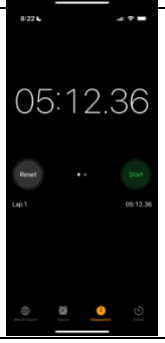
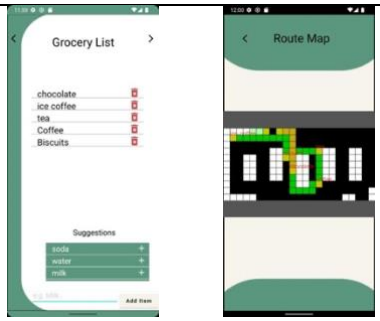
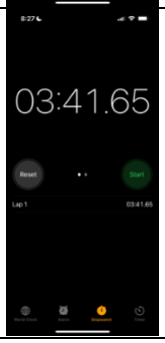
Experiment #	Items #	Normal List (iPhone)	Time Elapsed	GROCAFAST List	Time Elapsed	Difference in Time
1	7 items					2 min and 25 seconds approximately
2	5 items					2 min and 4 seconds approximately
3	5 items					2 min and 11 seconds approximately

Table 32: Performance testing.

The performance testing proved that GROCAFAST is an effective solution for a real-world problem and successfully attained the desired objective.

Chapter 7: Results and Discussion

This chapter demonstrates the results and the flow of steps to use GROCAFAST, the objectives achieved, and finally, the limitations.

7.1. Snapshots GROCAFAST for Shoppers

7.1.1. Landing Page

When GROCAFAST is launched, the landing page appears to display the application logo.

Figure 48: Results, Landing Page.

7.1.2. Log In

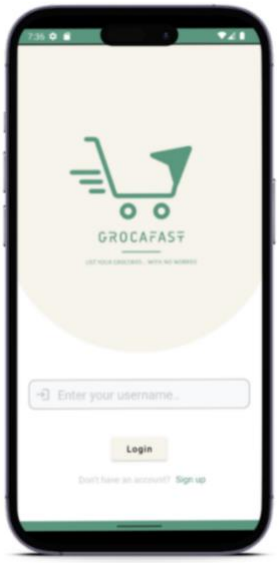


Figure 49: Results, Log In.

7.1.3. Create an Account

If the users did not have an account, they can create one by pressing on the “Sign Up” hypertext in the previous interface.

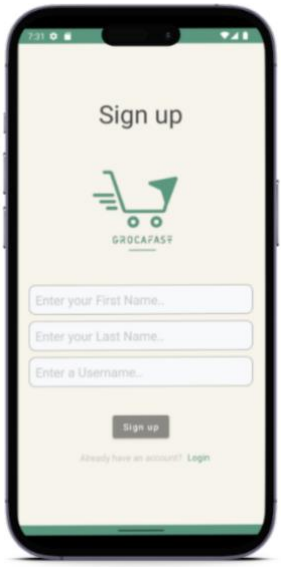


Figure 50: Results, Create an Account.

7.1.4. Home Page

Only users who have logged in can access the home page, which showcases all the services that users can enjoy.

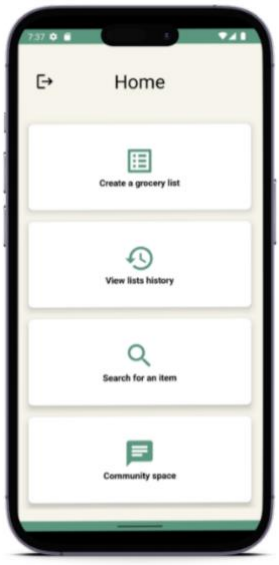


Figure 51: Results: Home Page.

7.1.5. Create grocery list and obtain the shortest path.

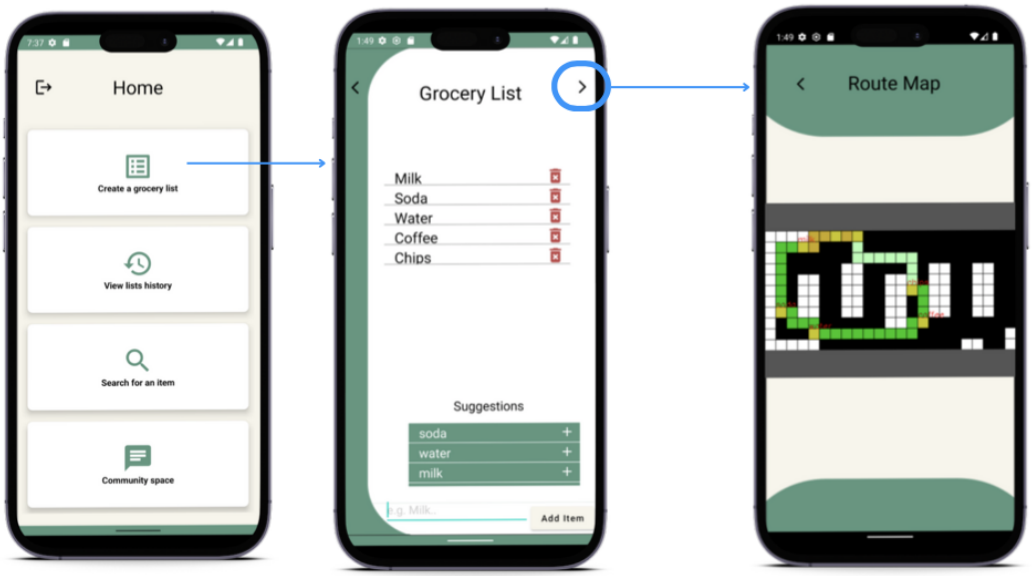


Figure 52: Results: Create grocery list and obtain the shortest path.

7.1.6. View List History

If the users have previous lists stored in the database, they can view it, edit it, or delete it.

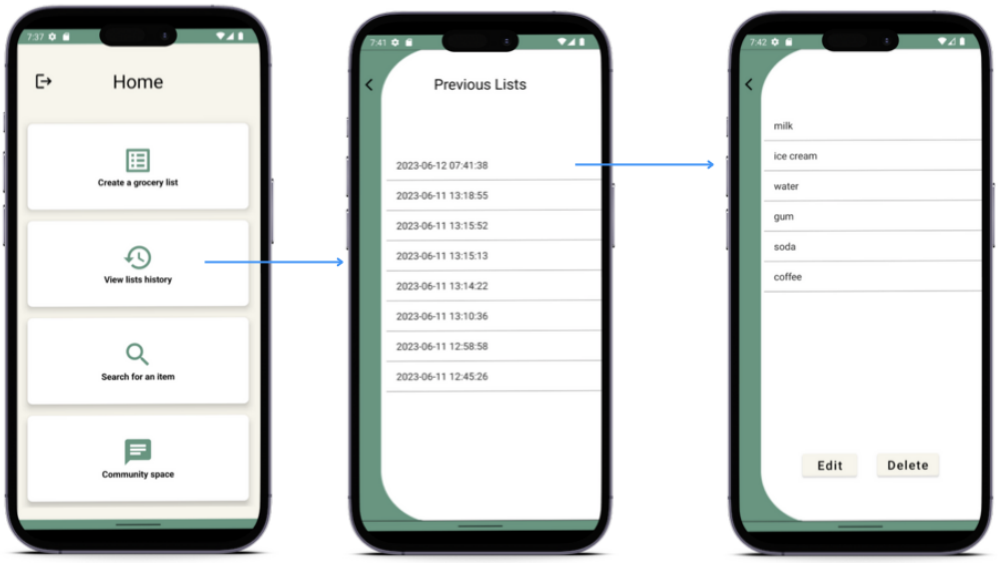


Figure 53: Results, View List History.

7.1.6.1 Edit a List

When the users press the "Edit" button on a previous grocery list, they will be directed to the "Create Grocery List" interface with the items on the previous list already written, so they can add or delete items as they prefer.

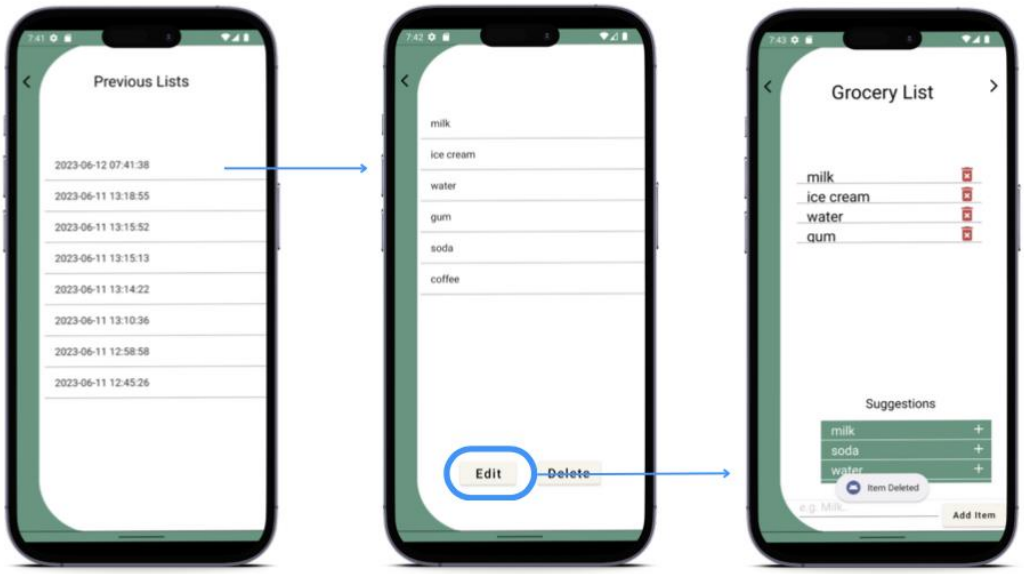


Figure 54: Results, Edit a List.

7.1.6.2 Delete a List

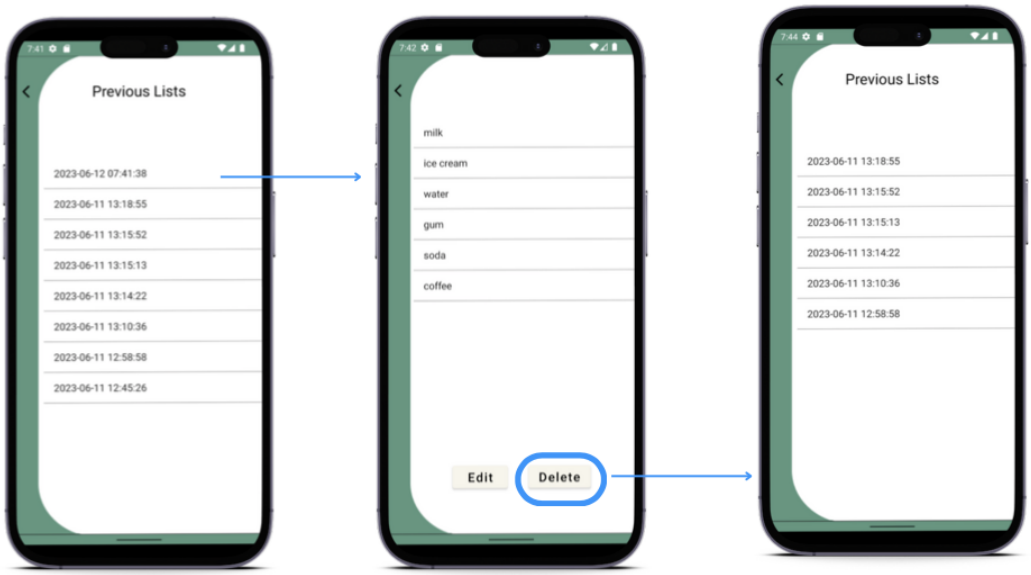


Figure 55: Results, Delete a List.

7.1.7. Search for an Item and View its Location.

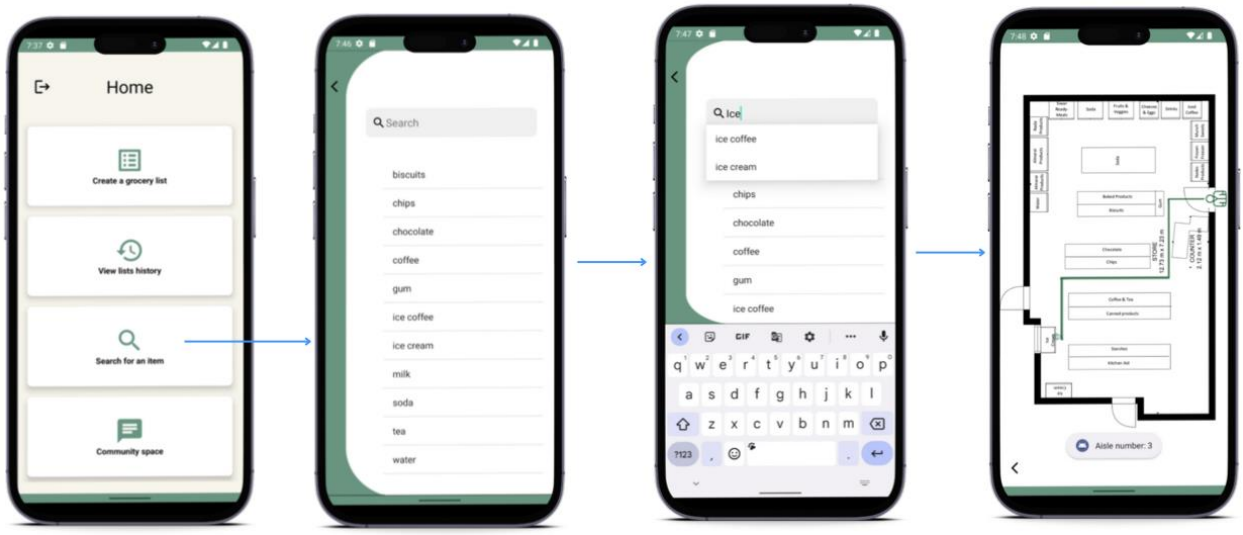


Figure 56: Search for an Item and View its Location.

7.1.8. Receive Personalized Shopping Suggestions.

Based on the stored lists in the database, GROCAFAST detects the top three most purchased items. These three items will then appear as shopping suggestions to the users the next time they create a grocery list.

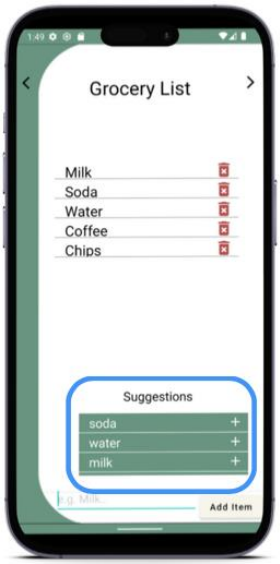


Figure 57: Results, Receive Personalized Shopping Suggestions.

7.1.9. Create Posts in an Online Chat Room

Users can share views and exchange recommendations in the online chat room between shoppers of the same store.

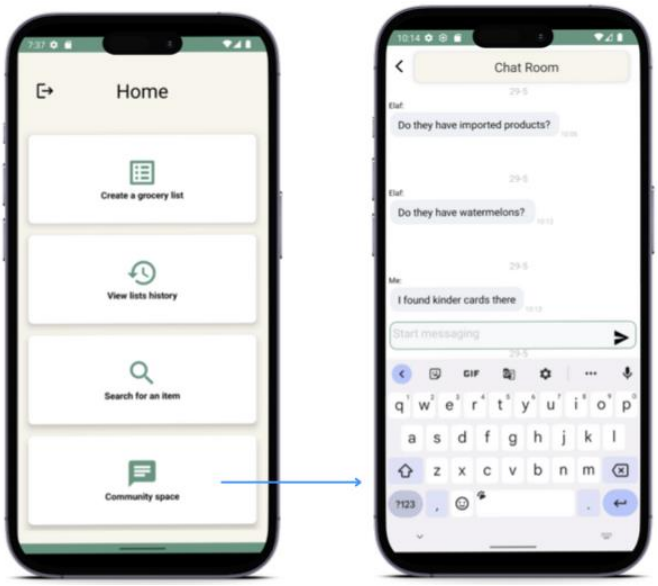


Figure 58: Results, Create Posts in an Online Chat Room.

7.2. Snapshots GROCAFAST for Admin

7.2.1. Log In

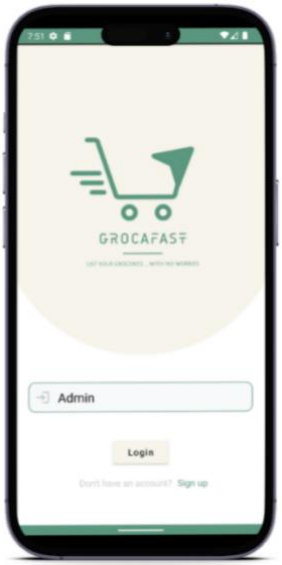


Figure 59: Results, Admin Log In.

7.2.2. Home Page

The admin functionalists are adding items to the database and viewing statistics (future work).

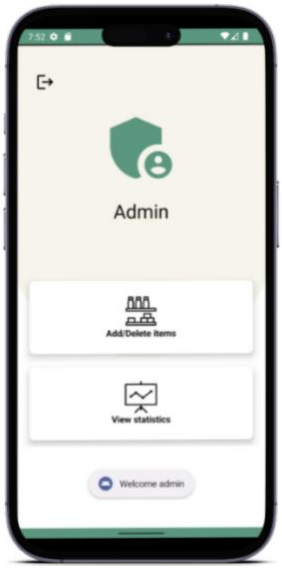


Figure 60: Results, Admin Home Page.

7.2.3. Add Items

The admin can add items to the database along with their category and aisle number.

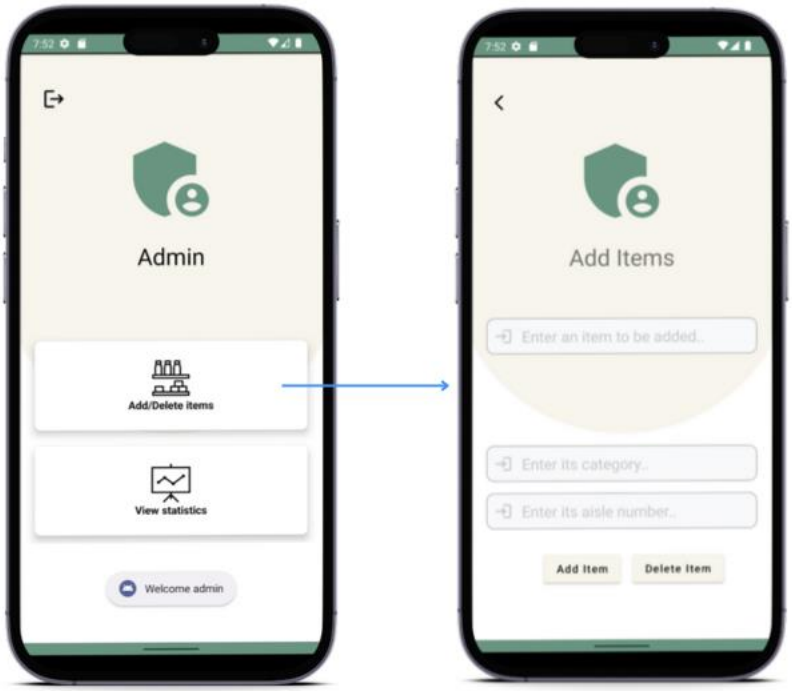


Figure 61: Results, Admin Add Items.

7.3. Achieved Objectives

The several types of testing we conducted confirm that GROCAFAST has effectively accomplished its primary goal of saving the shoppers' time. Additionally, the successful implementation of all the features outlined in the initial project scope has contributed to GROCAFAST's overall success in achieving its objectives.

7.4. Limitations

GROCAFAST encountered some limitations during its implementation. One of the main challenges was the unsuccessful integration with Indoor Atlas, which was due to the lack of documentation and technical support.

Additionally, due to the lack of cooperation from big grocery store, the project had to be implemented on a smaller scale, focusing solely on a local grocery store. This limitation prevented GROCAFAST from being tested in a larger and more diverse environment, but it did not limit its overall effectiveness.

Despite these challenges, the GROCAFAST team was able to adapt and overcome these obstacles by developing an alternative approach to successfully implement the application and achieve the desired results.

Chapter 8: Future Work and Conclusion

8.1. Future Work

There are several areas of future work that could be explored to improve the app's functionality and effectiveness.

One potential avenue of future work is to continue to explore different indoor positioning systems to include more location-based services. Additionally, more extensive user testing could be conducted to identify and address any usability issues and to gather feedback for further improvements.

Another area of focus could be to expand the scope of the project to include more large-scale grocery stores, potentially by partnering with industry stakeholders to gain access to the necessary resources.

Finally, continued development of the app's database and analytics capabilities could provide even more valuable insights for manufacturers and retailers, helping them to make data-driven decisions and respond to evolving shoppers' preferences.

Overall, there is significant potential for future work to build on the foundation established by GROCAFAST and to further enhance the in-store grocery shopping experience.

8.2. Conclusion

The GROCAFAST project has successfully developed a mobile application that enhances the in-store grocery shopping experience for shoppers.

By providing a range of features, such as generating a route map with the shortest path that guides the shopper from the entrance to the checkout point, passing through all the aisles that contain the items on the shoppers' grocery list, enabling shoppers to search for items and view their location inside the store, and suggesting personalized recommendations, GROCAFAST has streamlined the shopping process and simplified it for shoppers. Additionally, the app's database provides valuable insights to manufacturers, enabling them to make informed decisions and identify future trends.

While the project did face some limitations, such as the unsuccessful integration with Indoor Atlas and the lack of cooperation from larger grocery stores, the GROCAFAST team was able to overcome these obstacles and successfully achieve its primary objectives.

Overall, GROCAFAST has proven to be a valuable solution for enhancing the in-store grocery shopping experience and saving shoppers’ time.

REFERENCES

[1] Asana, “Everything you need to know about waterfall project management • asana,” *Asana* [Online]. Available: https://asana.com/resources/waterfall-project-management-methodology. [Accessed: 19-Oct-2022].

[2] “Apply Apriori algorithm in Supermarket Layout Research.” [Online]. Available: https://www.researchgate.net/publication/350667281_Apply_Apriori_Algorithm_in_Supermarket_Layout_Research. [Accessed: 21-Oct-2022].

[3] A. Store, "AnyList: Grocery Shopping List," 2012. [Online]. Available: https://apps.apple.com/us/app/anylist-grocery-shopping-list/id522167641. [Accessed 16 10 2022].

[4] A. Ross, “America's first supermarket at 100: How it changed the world,” *Time*, 09-Sep-2016. [Online]. Available: https://time.com/4480303/supermarkets-history/. [Accessed: 22-Oct-2022].

[5] C. Stoll, “Grocery - Smart Shopping List,” *App Store*, 10-May-2017. [Online]. Available: https://apps.apple.com/us/app/grocery-smart-shopping-list/id1195676848. [Accessed: 19-Oct-2022].

[6] E. Felsenthal, “Front line workers tell their own stories in the new issue of Time,” *Time*, 09-Apr-2020. [Online]. Available: https://time.com/4480303/supermarkets-history/). [Accessed: 22-Oct-2022].

[7] “Hub: Write or type? how a paper versus a digital shopping list influences the way consumers plan and shop: 10.1086/698877,” *Sci*. [Online]. Available: https://sci-hub.hkvisa.net/10.1086/698877. [Accessed: 22-Oct-2022].

[8] “Indoor navigation with mobile augmented reality and Beacon Technology for wheelchair users,” IEEE Xplore. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7897199. [Accessed: 21-Oct-2022].

[9] K. Farrell, “How we wound up with supermarkets: A history of the grocery store,” *10Best*, 21-Apr-2020. [Online]. Available: https://www.10best.com/interests/food-culture/10-points-grocery-store-history-supermarkets-changed-how-we-shop/. [Accessed: 22-Oct-2022].

[10] Listonic, “Shopping list - listonic,” *App Store*, 15-Oct-2009. [Online]. Available: https://apps.apple.com/gb/app/shopping-list-listonic/id331302745. [Accessed: 19-Oct-2022].

[11] N. Katuk, “Design and development of smart list: A mobile app for creating and managing grocery lists,” *Baghdad Science Journal*, 15-Nov-2019. [Online]. Available: https://www.academia.edu/40941578/Design_and_Development_of_Smart_List_A_Mobile_App_for_Creating_and_Managing_Grocery_Lists. [Accessed: 21-Oct-2022].

[12] Non Function requirements.docx, “Non Function requirements.docx,” *Google Docs*, 2020.https://docs.google.com/document/d/1LC15HrUPxUwq7gJdYsCBb1hSqAajUCW9/edit (accessed Nov. 10, 2022).

[13] Sharp, H. (2019). *Interaction design: Beyond human-computer interaction, fifth edition*. Wiley & Sons Canada, Limited, John.

[14] Sommerville, I. (2019). *Software engineering, Tenth Edition*. Pearson Education.

[15] S. Spiggle, “>grocery shopping lists: What do consumers write?: ACR,” *ACR North American Advances*, 01-Jan-1987. [Online]. Available: https://www.acrwebsite.org/volumes/6695/volumes/v14/NA-14/full. [Accessed: 22-Oct-2022].

[16] S. Davu, “The benefits of adhering to a software development methodology,” *Segue Technologies*, 14-Aug-2017. [Online]. Available: https://www.seguetech.com/benefits-adhering-software-development-methodology-concepts/. [Accessed: 19-Oct-2022].

[17] “The Smart Shopping List: An effective mobile solution for grocery list ...” [Online]. Available:https://www.researchgate.net/publication/323713789_The_smart_shopping_list_An_effective_mobile_solution_for_grocery_list-creation_process. [Accessed: 22-Oct-2022].

[18] V. Sharma, “Supermarkets as an example – ‘Apriori algorithm,’” LinkedIn, 13-Apr-2020. [Online]. Available: https://www.linkedin.com/pulse/supermarkets-example-apriori-algorithm-vaibhav-sharma/. [Accessed: 21-Oct-2022].

[19] *Your partner in creating Smart Indoor Spaces* (2022) *IndoorAtlas*. Available at: https://www.indooratlas.com/ (Accessed: 20 May 2023).

[20] *Cubicasa: Create a floor plan in 5 min for Real Estate & appraisals* (2023) *CubiCasa*. Available at: https://www.cubi.casa/ (Accessed: 20 May 2023).

[21] "Shortest Path Problem," *NVIDIA Developer*, Jul. 29, 2016. https://developer.nvidia.com/discover/shortest-path-problem#:~:text=The%20shortest%20path%20problem%20involves,to%20the%20shortest%20path%20problem. (accessed Mar. 21, 2023).

[22] "Shortest Path Algorithms | Brilliant Math & Science Wiki," *Brilliant.org*, 2023. https://brilliant.org/wiki/shortest-path-algorithms/ (accessed Mar. 21, 2023).

[23] Admin, "Floyd-Warshall Algorithm - GATE CSE Notes," *BYJUS*, Jul. 18, 2022. https://byjus.com/gate/floyd-warshall-algorithm-notes/ (accessed Mar. 21, 2023).

[24] "Floyd Warshall Algorithm DP 16," *GeeksforGeeks*, Jun. 07, 2012. https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/ (accessed Mar. 21, 2023).

[25] R. A. S, "A* Algorithm Concepts and Implementation," *Simplilearn.com*, Jul. 22, 2021. https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/a-star-algorithm#:~:text=A*%20Search%20Algorithm%20is%20a,path%20algorithm%20(Dijkstra's%20Algorithm). (accessed Mar. 21, 2023).

[26] "Bellman Ford Algorithm DP 23," *GeeksforGeeks*, Dec. 2012. https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/ (accessed Mar. 21, 2023).

Appendix A: Data Gathering

A.1 Blank Copy of the Survey

Figure 62: Blank Copy of the Survey Description

How much time do you spend collecting your monthly groceries at the supermarket? *

كم من الوقت تقضيه في جمع مشترياتك الشهرية من السوبرماركت؟

- 1-3 hours / ١-٣ ساعات
- 2-4 hours / ٢-٤ ساعات
- 3-6 hours / ٣-٦ ساعات
- Other: _____

Figure 63: Blank Copy of the Survey- Question 1

Do you think you spend much of that time searching for a product in the supermarket? *

هل تعتقد أنك تقضي الكثير من ذلك الوقت في البحث عن منتج معين في السوبرماركت؟

- Yes / نعم
- No / لا
- Maybe / ربما

Figure 64: Blank Copy of the Survey- Question 2

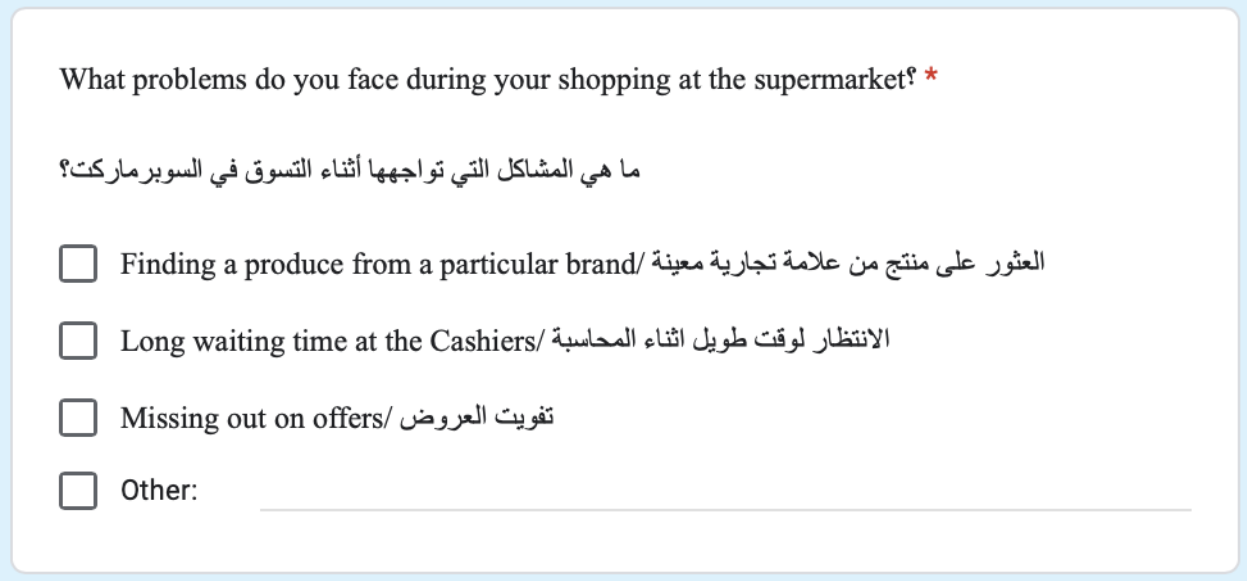


Figure 65: Blank Copy of the Survey- Question 3

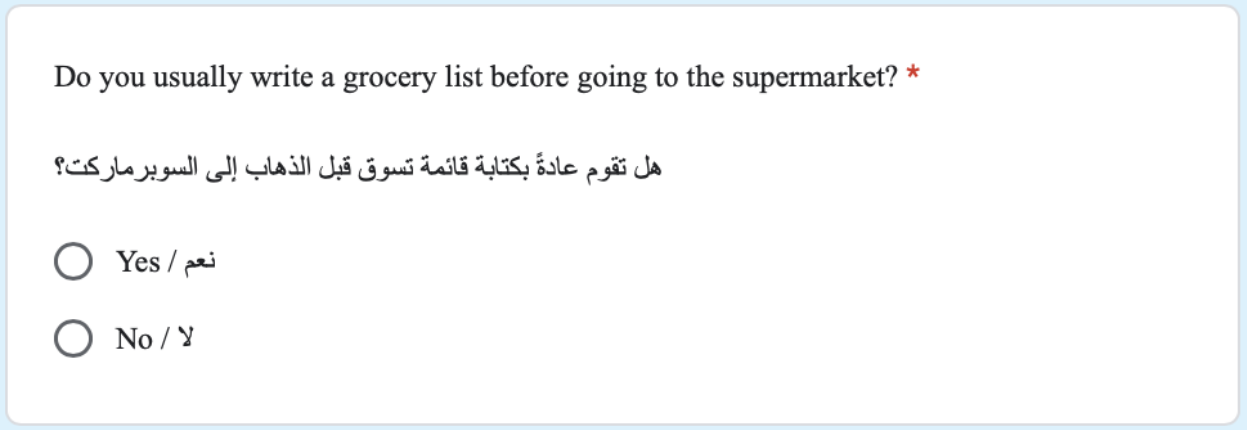


Figure 66: Blank Copy of the Survey- Question 4

If yes, what are the reasons?

إذا كانت الإجابة بنعم ، فما هي الأسباب؟

- [ ] Make shopping easier / لتسهيل عملية التسوق
- [ ] Help with meal planning / للمساعدة في التخطيط للوجبات
- [ ] Save money / لتوفير المال
- [ ] Save time / لتوفير الوقت
- [ ] Save energy / لتوفير الجهد
- [ ] Other: _____

Figure 67: Blank Copy of the Survey- Question 5

Which way is more effective, writing a Grocery Shopping List on paper or digitally? *

ما هي الطريقة الأكثر فعالية، كتابة قائمة التسوق على الورق أم رقميًا؟

- ( ) Paper / ورق
- ( ) Digitally / رقميًا

Figure 68: Blank Copy of the Survey- Question 6

If there is an application that arrange your grocery list items based on the aisles of your intended supermarket to provide you with the shortest path to collect all your groceries, would you like to use it? *

إذا كان هناك تطبيق يرتب عناصر قائمة تسوقك بناءً على ممرات السوبرماركت الذي ستذهب اليه لتزويدك بأقصر طريق لجمع جميع مشترياتك، هل سترغب في استخدامه؟

- ◯ Yes / نعم
- ◯ No / لا

Figure 69: Blank Copy of the Survey- Question 7

If yes, which of the following features would meet your requirements and make the Smart Grocery Shopping Application useful? (You can choose more than one option)

إذا كانت الإجابة بنعم ، أي الميزات التالية ستلبي متطلباتك وتجعل التطبيق مفيدًا؟ (يمكنك اختيار اكثر من خيار)

- ☐ A route map that guides you to a specific aisle that matches the items in your grocery list. / خريطة ترشدك إلى الممر الذي يحتوي على المنتجات الموجودة في قائمة تسوقك
- ☐ In-store navigation to help make finding items easier. / تسهيل العثورعلى منتج معين من خلال توجيهك الى موقعه داخل السوبرماركت
- ☐ Recommendation of items similar to your previous purchases. / اقتراح منتجات مماثلة لمشترياتك السابقة
- ☐ A space where customers can share reviews and exchange recommendations on a specific supermarket branch. / إنشاء مساحة تمكن العملاء من مشاركة وتبادل التوصيات لفرع سوبرماركت معين
- ☐ Other: _____

Figure 70: Blank Copy of the Survey- Question 8

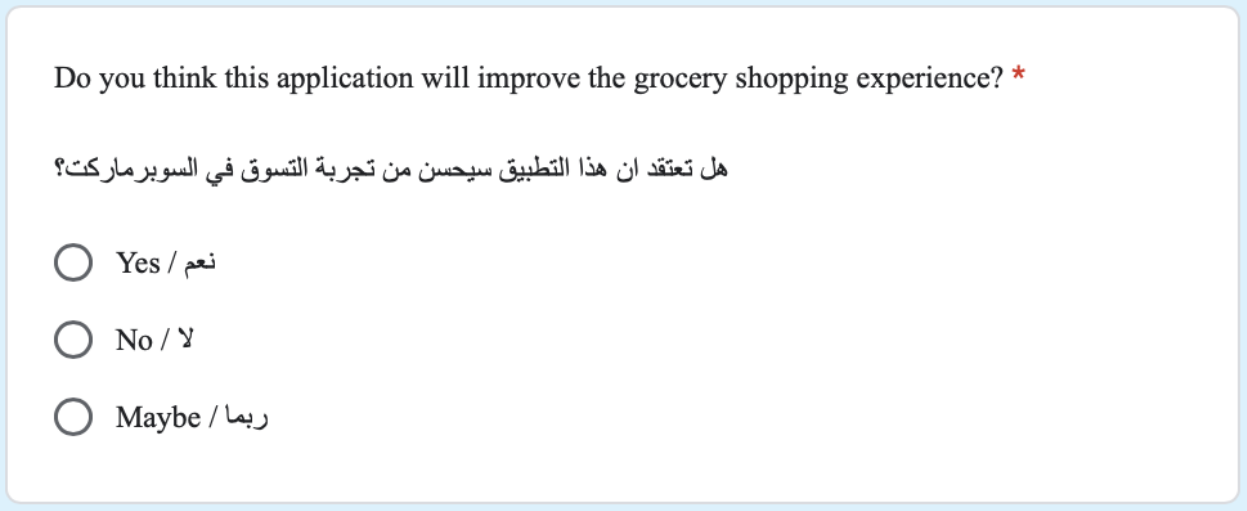


Figure 71: Blank Copy of the Survey- Question 9

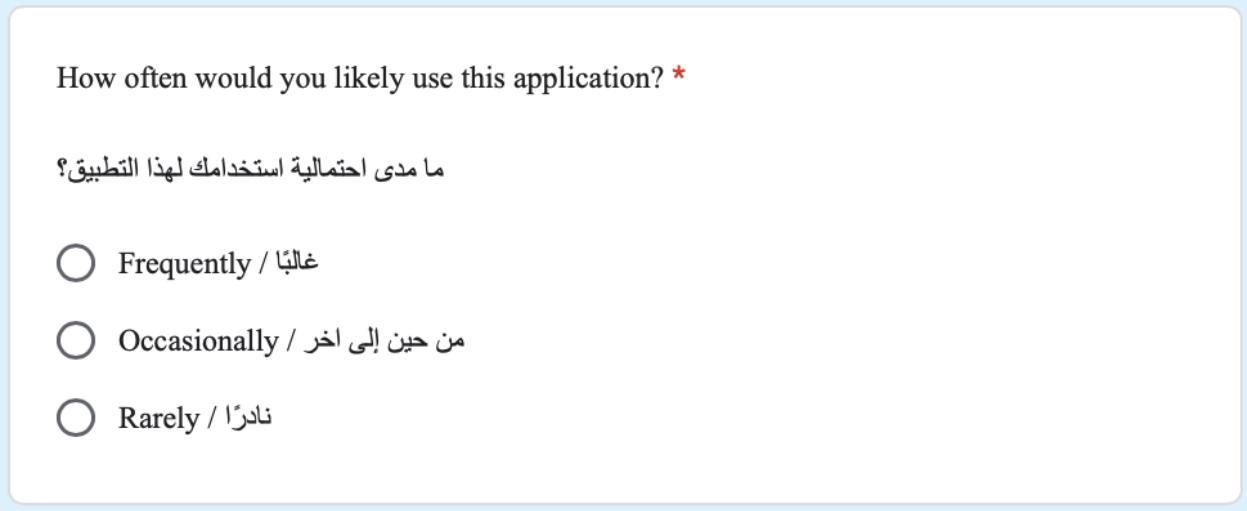


Figure 72: Blank Copy of the Survey- Question 10

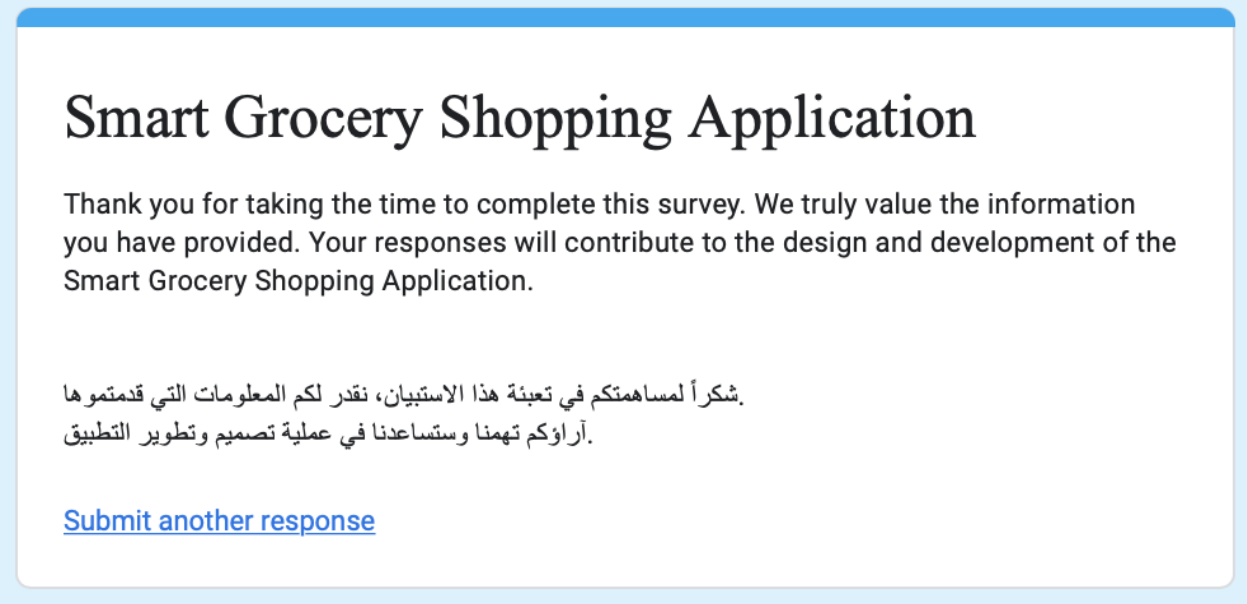


Figure 73: Blank Copy of the Survey Thank You Message

A.2. Survey Detailed Results

This section will present the detailed results of the 443 responses we received on the survey.

Question 1: In the following question, we noticed that 67.3% of the answers said they spent 1-3 hours collecting their usual groceries, which indicates a real problem. Collecting groceries should not be a time-consuming activity. The result assured us that GROCAFST is needed.

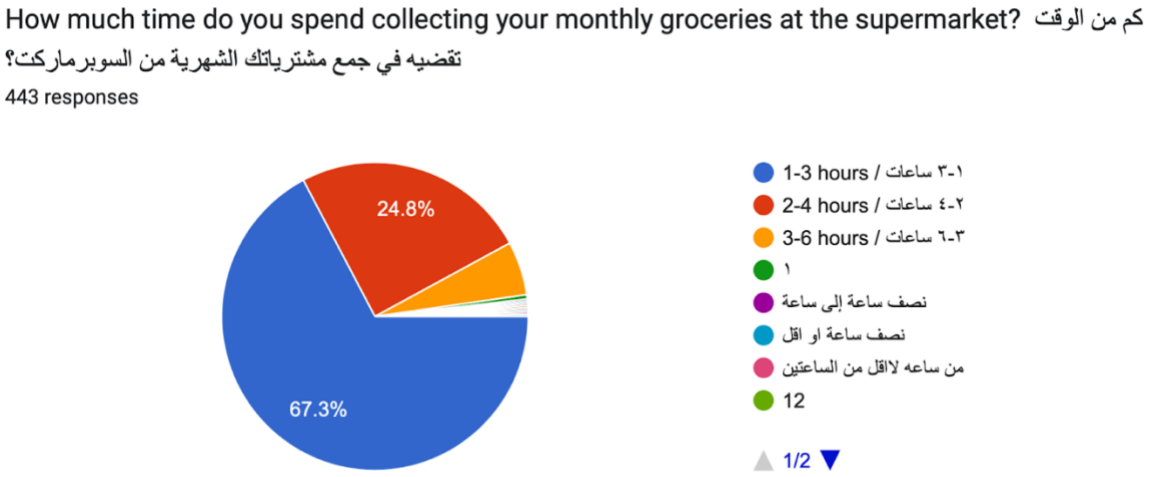


Figure 74: Survey Detailed Results- Question 1.1

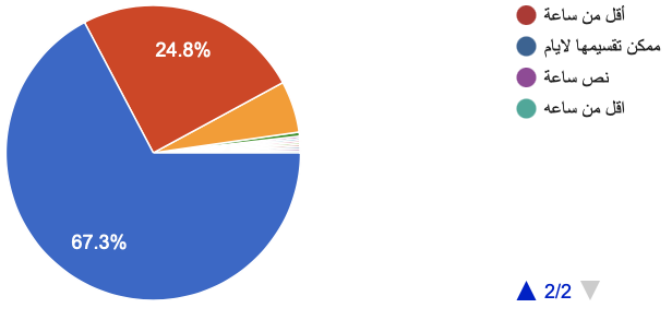


Figure 75: Survey Detailed Results- Question 1.2

Question 2: The following question shows that 52.8% of the answers said they spent much of their time in the supermarket searching for a product. As a result, the feature that enables the user to search for a product's location inside the supermarket appears to be useful in such cases.

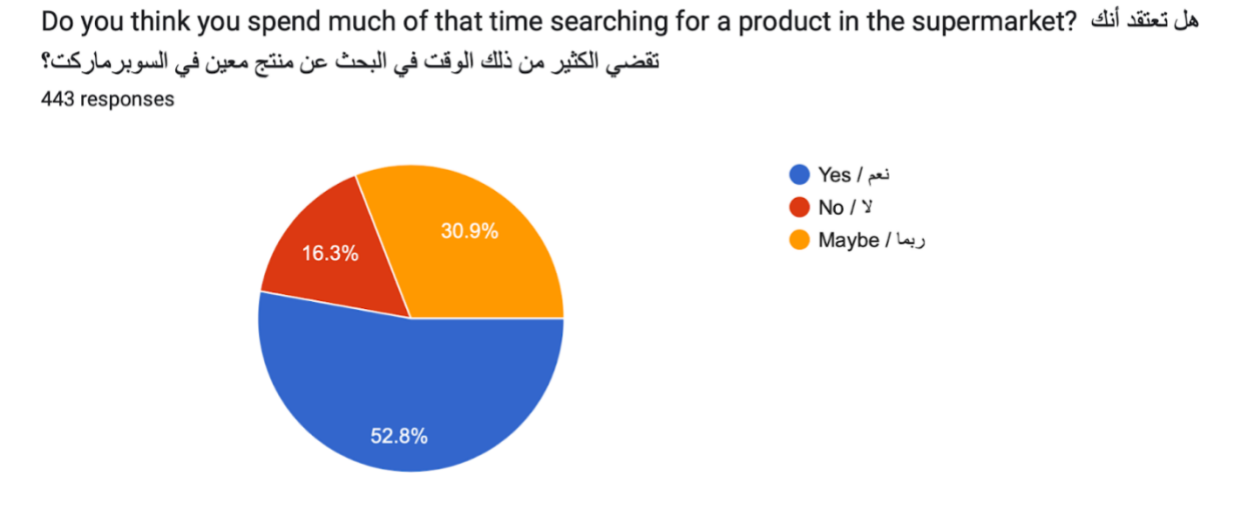


Figure 76: Survey Detailed Results- Question 2

Question 3: The following question shows that finding a product from a particular brand is one of the most encountered problems people face during their shopping at the supermarket.

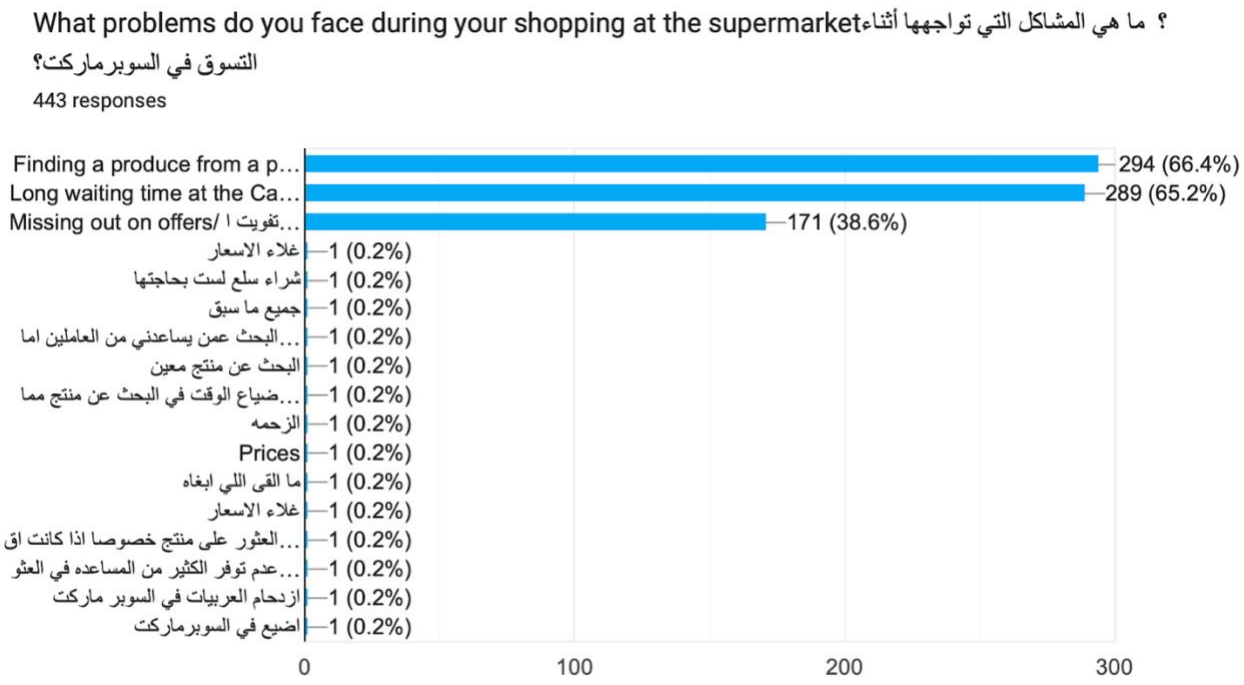


Figure 77: Survey Detailed Results- Question 3

Question 4: The following question shows that 77% of the answers said they write a grocery list before going to the supermarket, which ensures that new consumers strive on saving time and improving efficiency.

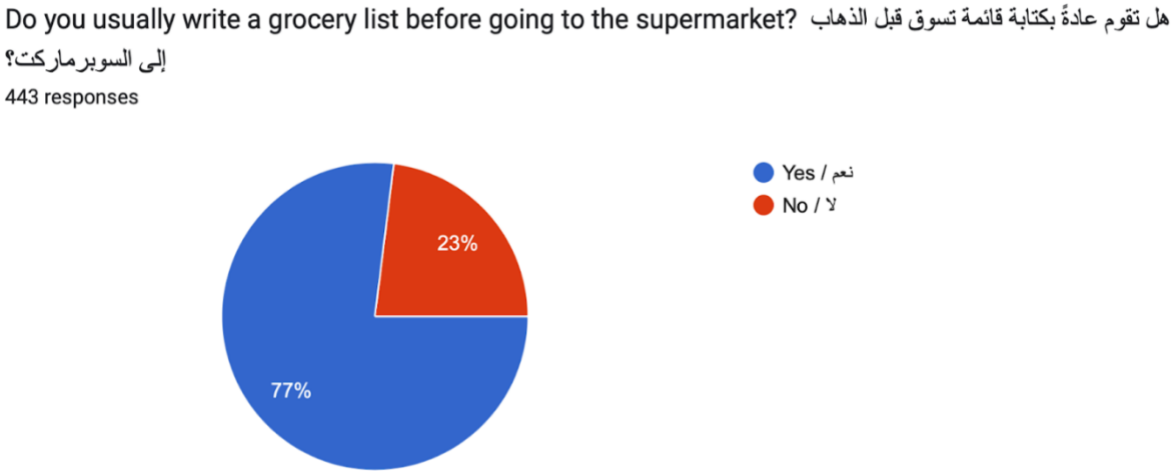


Figure 78: Survey Detailed Results- Question 4

Question 5: The following question reveals the reasons why people think writing a grocery list is important.

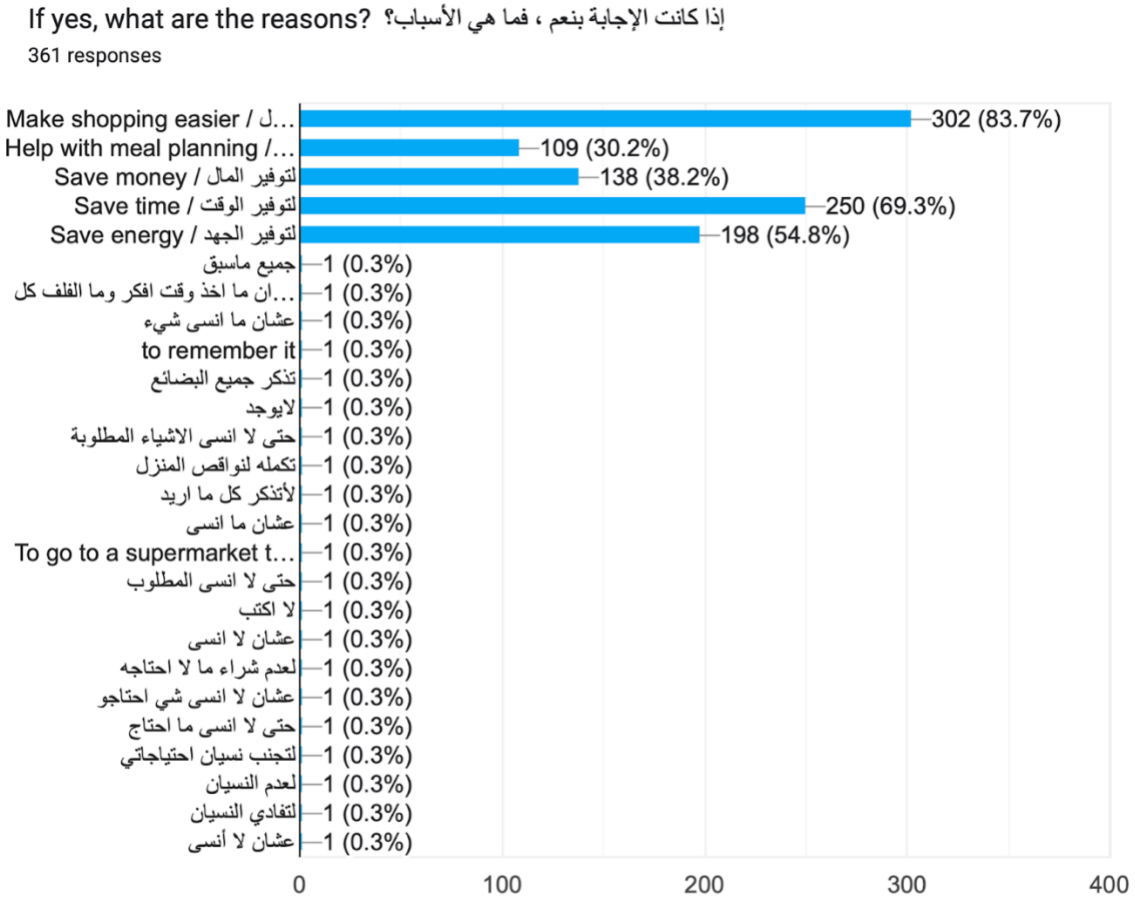


Figure 79: Survey Detailed Results- Question 5

Question 6: In the following question, we noticed that 72.9% of the answers said they prefer writing a grocery list digitally, which highly supports GROCAFAST because it is based on the idea of writing a digital grocery list.

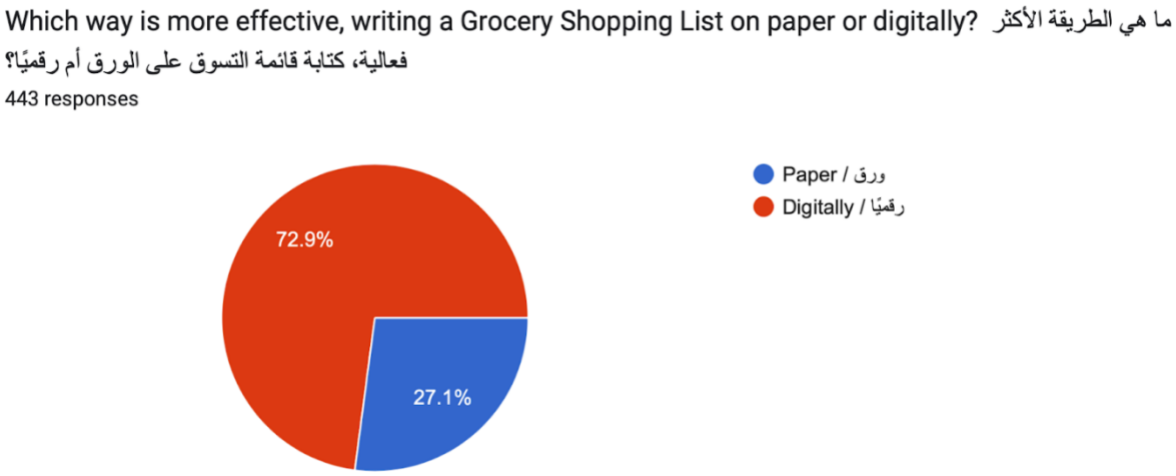


Figure 80: Survey Detailed Results- Question 6

Question 7: The following question shows that 92.8% of the answers said they would like to use GROCAFAST.

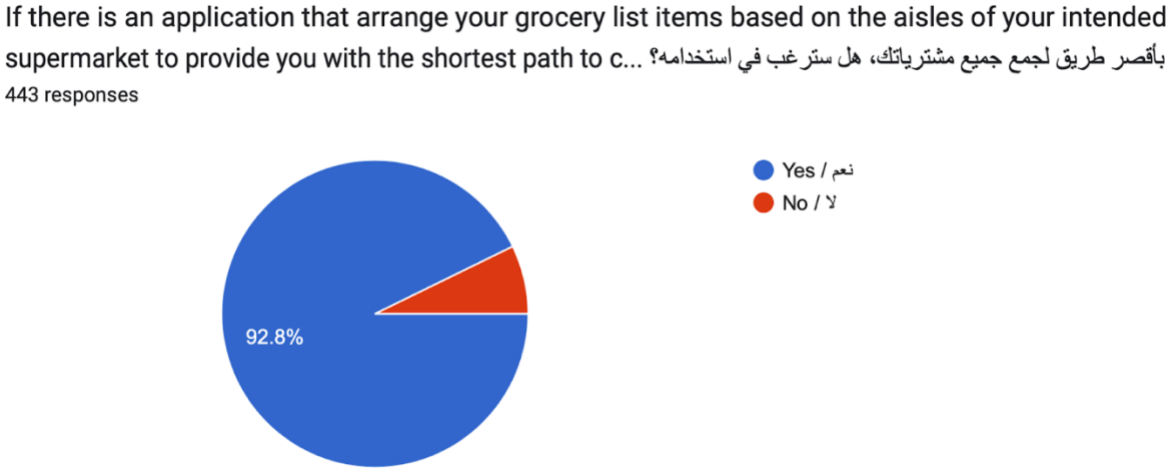


Figure 81: Survey Detailed Results- Question 7

Question 8: The following question shows the most liked features of GROCAFAST, we also had a number of useful suggestions that we plan on implementing in our future work.

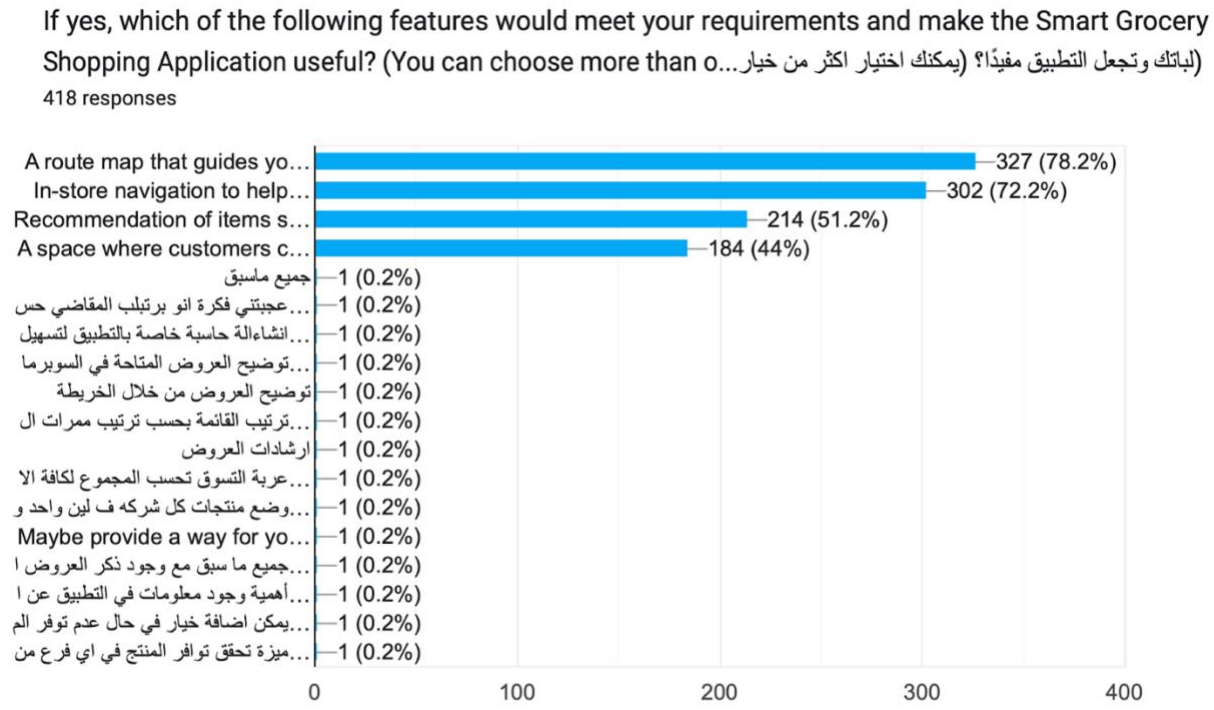


Figure 82: Survey Detailed Results- Question 8

Question 9: The following question shows that 84.4% of the answers said they think GROCAFAST will improve the grocery shopping experience.

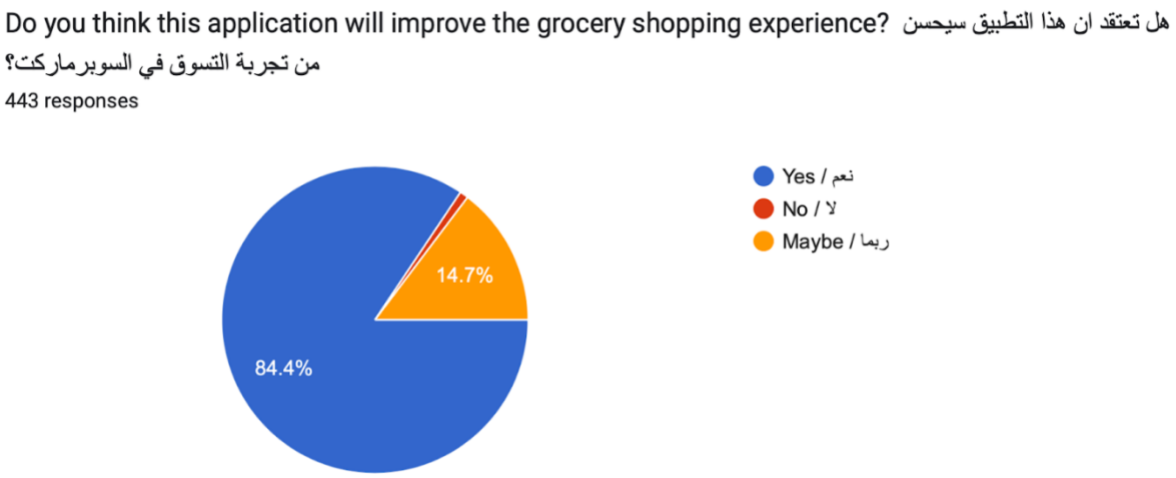


Figure 83: Survey Detailed Results- Question 9

Question 10: The following question shows that 64.8% would use GROCAFAST frequently, whereas 30.5% would use it occasionally.

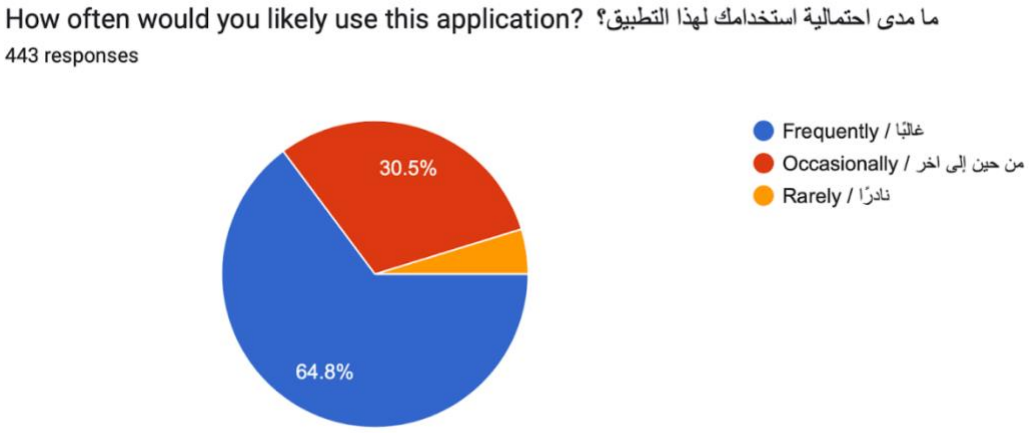


Figure 84: Survey Detailed Results- Question 10

Appendix A: User guide to use GROCAFAST

Welcome to GROCAFAST! This user guide will walk you through how to download the app, sign up or log in, and navigate the home page with its features.

Downloading the App (only for Android OS):

1. Visit Google Play Store on your mobile device.
2. Search for "GROCAFAST" in the search bar.
3. Click on the "Install" button to download the app onto your device.

Signing Up or Logging In:

1. After downloading the app, open it on your device.
2. If you are a new user, click on the "Sign Up" hyperlink and enter your username, first and last name. If the username already exists, the app will ask you to enter another username.
3. If you are an existing user, enter your username and click on the "Log In" button.
4. Once you have successfully signed up or logged in, you will be directed to the home page.

Figure 85: User Guide, Log In.

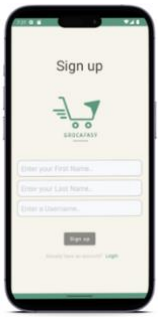


Figure 86: User Guide, Sign Up.

Home Page Features:

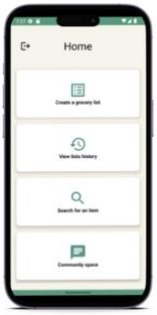


Figure 87: User Guide, Home Page.

1. **Create a Grocery List:**

   To create a new grocery list, click on the "Create Grocery List" tab on the home page. You can add items to your list by typing them or adding them from the suggestion section at the bottom.

   1.1 After you are done creating the grocery list and you are ready to start your shopping trip, simply click the arrow in the upper right-hand corner to generate the route map that will guide you through your shopping trip in the store.

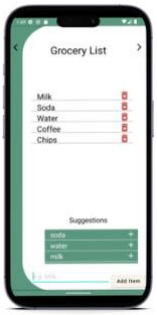


Figure 88: User Guide, Create Grocery List.

2. **View List History:**

   To view your past grocery lists, click on the "View List History" tab. You can view the details of each list. If you wish to make changes to a previous list, you can choose to edit it, which will take you back to the "Create Grocery List" page with the items already included. Alternatively, you may delete the entire list if desired.

Figure 89: User Guide, View List History1.

Figure 90: User Guide, View List History2.

3. **Search for an Item:**

   To search for a particular item and see where it is located in the store, click on the "Search For An Item" tab, then type the name of the item in the search bar. The app will display any matching items from its database and view their location in the store.

Figure 92: User Guide, Search for an Item1.

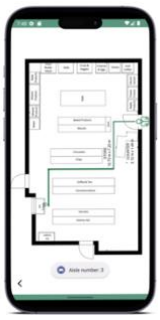


Figure 91: User Guide, Search for an Item2.

4. **Community Space:**

   To access the community space, click on the "Community Space" tab. Here, you can connect with other shoppers of the same store to share reviews, and exchange recommendations.

Figure 93: User Guide, Community Space.

We hope this user guide has helped you navigate GROCAFAST. Happy grocery shopping!