Software Requirements Specification

for

<Sahulat>

Version 1.0 approved

Prepared by <Group No. 2>

<Section No. 5218, Class No. 4>

<Oct, 14, 2021>

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the application “Grocery List Maker”. It will provide the overall description of the application, its interfaces, system features and nonfunctional requirements under which it must operate. This document is intended for users and potential developers of the application.

1.2 Document Conventions

This Document was created based on the IEEE template for System Requirement specification Documents.

1.3 Intended Audience and Reading Suggestions

This document is intended for developers, project owner, project managers and users.

- Users can read section 2.1, 2.2, 2.3 as it includes the perspective of the product, the functions users will be able to perform and the knowledge/technology they will need in order to use the application. For more understanding of the application, they can see the use-case diagram.
- Project Managers, Developers, Team Lead should read from Section 2 to Section 6 as, Section 2 contains the functions, the user class, environments application will be working in as well as constraints. Section 3 talks about the interfaces on which the application will function on. Section 4 contains the system features that application needs to have and the use cases in detail. Section 5 contains the non-functional requirements and section 6 includes the additional requirements.

1.4 Product Scope

Based on your buying habits, this app will predict the items that you will need in the days to come and create a grocery list for you. The user will feed in data through daily purchases which this app will keep track of. Based on these purchases, this app will notify what to buy, when to buy and from where to buy. The user should be able to choose a grocery list that saves money or a list that helps him buy stuff conveniently. It also marks the best route to make the purchases.

1.5 References

IEEE Template for System Requirement Specification Documents:

https://goo.gl/nsUFwy

Use Case Diagram was designed by using:

http://www.umletino.com/umletino.html

Some features and ideas by taken by following link:

https://www.bestproducts.com/eats/food/g1505/grocery-shopping-list-apps/

2. Overall Description

2.1 Product Perspective

This product is aimed towards a person who wants to save his time when planning to visit a grocery store. As grocery lists are usually made manually by a person which requires a lot of time and effort. Sometimes a person can forget to add certain items in the list which results in multiple trips to the grocery store. This product will help the user in generating a grocery list based on his past purchases and help the user to maintain a record of his past purchases and expenditure.

2.2 Product Functions

User will be able to:

Manage Accounts:

• Create an account or signup through Gmail or Facebook.

• Login using his registered email and password or gmail or facebook email.

• Request for new password using forgot password option.

• Log out of the account.

• Delete the account.

Manage Purchases:

• Add receipt of his past purchases to the database through QR Code, Image Processing/ Barcode or manually.

• Maintain a track record of his previous purchases and edit previous purchases.

Generate Grocery List:

• Generate a grocery list based on his past purchases

• Manually add or delete items from the generated list

Manage Inventory:

• Create inventory

• Update inventory

• View and delete inventory

Manage Budget:

• set a monthly or yearly budget for groceries

• maintain a track record of his previous expenditure

Get Route:

• Get a store recommendation for the list and find the best route.

• Filter store recommendations based on price and location.

Share List :

• Share the list with multiple users such as family members using a link or email.

Manage App Settings:

• Change the color theme of app to Dark Mode or Light Mode

• Change the appearance of lists using grid view or tiles view

• Change notifications settings

• Change user profile

2.3 User Classes and Characteristics

It is a very generic product and can be used by anyone who wants to make a grocery list. It can also be used by restaurant owners as they deal with large amounts of groceries as well. The users must have a smartphone and access to internet connection. They should know the basics of English and how to scan/ take pictures as well as how to sign-up/ log in.

2.4 Operating Environment

The application will be functional on
• Android OS 11
• iOS 14
The application will require services from these applications:
• Camera
• Maps
• WhatsApp
• Facebook

2.5 Design and Implementation Constraints

- The grocery list application is developed in the programming language Dart using Flutter as the software development kit. The application uses a modular design where each component represents a widget. These widgets are wrapped up into a module and all the modules depend on each other through well-written APIs. There are several APIs available to make the development process easy.

- Firebase is used as our database that enables us to develop our application in flutter. It provides backend database with secure access to build rich, collaborative applications directly from the client side.

- If the user does not allow access to location, the app will not be able to determine the shortest route of a store.

- If the user does not allow access to the camera, image processing/ scanning will not be able to function properly.

2.6 User Documentation

The application will have a user manual and tutorials for users to easily access every component of the application.

2.7 Assumptions and Dependencies

• We are assuming that user will share at least 3 months of grocery data for the list generating algorithm to work properly

• The prices of products for price comparison will be dynamically added to the database when the user adds his list of past purchases.

3. External Interface Requirements

3.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

3.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

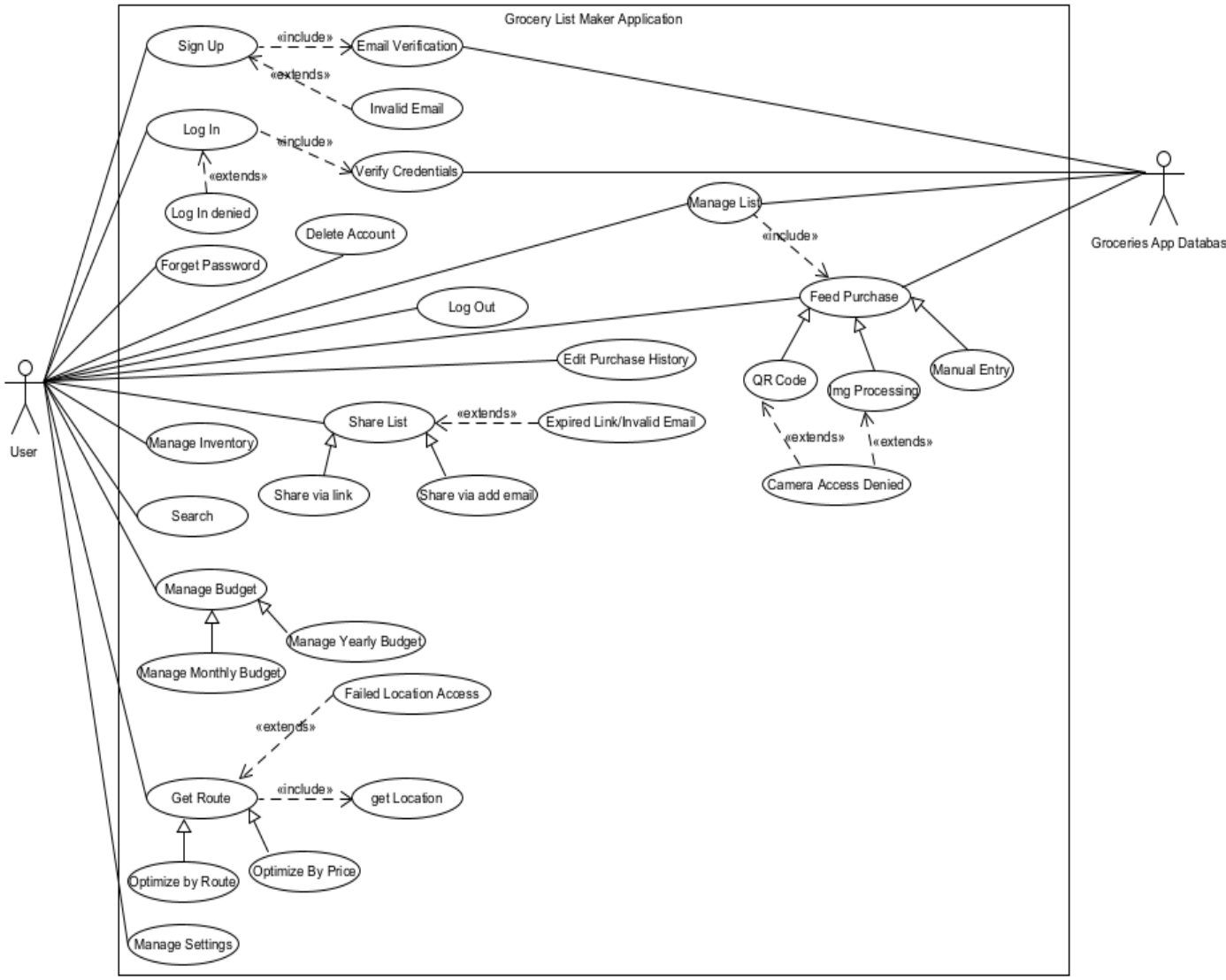
3.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

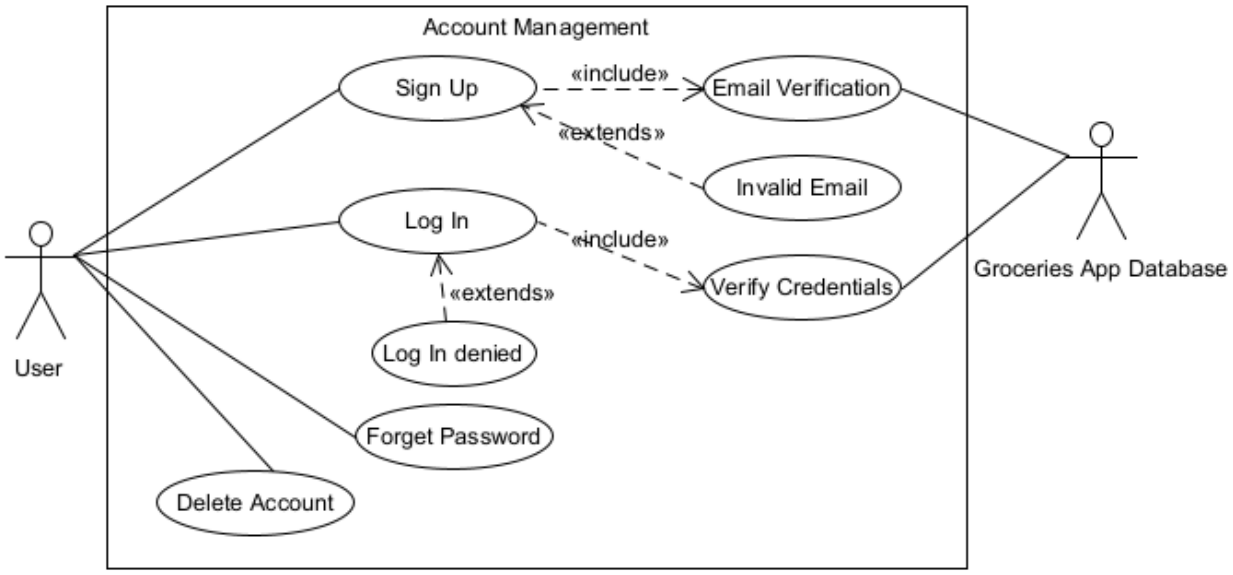
3.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

4. System Features



4.1 Account Management



4.1.1 Description and Priority

Account management is the feature which enables the system to provide and manage accounts of multiple users. It let's the user create, delete, request a new password, log in and log out of the account. It is a high priority feature as the application can only be accessed if the account exists.

4.1.2 Stimulus/ Response Sequence

Signup: When a new user opens the application, the first form which is displayed will show two buttons, one for log-in and one for sign-up. User taps on the sign-up button, the system will be prompted to show the sign-up form which will have two fields: Email address and password. The

user details are then stored in the database and user is given access to the application and main page is shown, if the email provided is verified and password meets the criteria. Otherwise, an error message is displayed providing the reason of sign-up failure.

Login: When user opens the application, a form with log-in and sign-up buttons is displayed. When user taps on log in button, the system returns a log in form with email address and password fields which need to be filled by an existing user. After the user has entered the details, the system verifies the data with the existing user account information from the database. System will return the main page of the application upon successful completion of the log-in attempt.

Forgot Password: When user enters invalid information, a forgot password button appears. This provides an option for requesting a new password by the system. Forgot password form only has one field to be filled which is email address. After user enters email address, it is cross verified from the database and a new password is sent to the user through an email by the system. Using that password, user can successfully log in to the application.

Logout: A logged in user goes in the settings menu and taps on the logout button. System saves any recent information in the database and approves the log out request of the user. A user successfully logs out and a login/ signup form is returned.

Delete Account: A logged in user goes in the settings menu and taps on the delete account button. System asks for the confirmation for this function by the user. After getting confirmation, the system deletes the account of the user and all the data related to that user successfully and login form is returned.

4.1.3 Functional Requirements

FR 1.1 System must show a sign-up/log in button.

FR 1.2 System shall provide an option to sign up for a new account.

FR 1.3 System shall display a form in which user can fill the details like email address and password.

FR 1.4 System shall give the option to sign up using a Gmail account (only Gmail address will be required).

FR 1.5 System shall give option to sign up using Facebook account.

FR 1.6 System shall verify the email address provided at the time of sign up and after that grant access to the user otherwise it will show an error.

FR 1.7 System shall give option to option to log in.

FR 1.8 System shall give option to log in using Gmail account.

FR 1.9 System shall give the option to log in using Facebook account.

FR 1.10 System shall give option to log in using email address and password.

FR 1.11 System shall display a login form with details to be filled by the user. System will cross check the data provided at the time of login from the database, if correct, the system will give access to the user .

FR 1.12 System shall show error if data provided is invalid.

FR 1.13 System shall give option to log out.

FR 1.14 System shall save any information left and log the user out.

FR 1.15 System shall display forgot password button if login attempt fails.

FR 1.16 System shall give option to request for new password.

FR 1.17 System shall send a new generated password to the user through the given email.

FR 1.18 System shall give option to delete account.

FR 1.19 System shall ask the user for confirmation of account deletion.

FR 1.20 System shall delete the account and any related information regarding the user.

Use case name	**Sign up**
Related requirements	FR 1.1 FR 1.2 FR 1.3 FR 1.4 FR 1.5 FR 1.6
Goal in context	New account will be created.

Preconditions		1. Email provided exists and must be correct. 2. Password meets the system's criteria. 3. Email given must be a unique one. 4. There should be an internet connection.
Successful end condition		The user account is created after the valid information is entered and the homepage is returned as a logged-in user.
Failed end condition		New user account is not created for different reasons and an error message is shown.
Primary actors		User
Secondary actors		Groceries App Database
Trigger		A new user taps on sign up.
Included cases		Email Verification
Main flow	Step	Action
	1	New User taps on the sign-up button and the system returns a sign-up form to be filled with details like email and password.
	2	User enters the details.

	3 Include: Email Verification	System verifies the information provided and saves the information in the database.
	4	A new user account is created.
	5	Main page is returned with the new user logged in.

Use case name	**Log-in**
Related requirements	FR 1.1 FR 1.7 FR 1.8 FR 1.9 FR 1.10 FR 1.11 FR 1.12
Goal in context	Existing users can log in and access the application.

Preconditions		User must exist and must be connected to the internet.
Successful end condition		User logs in the account and the main page is returned.
Failed end condition		The login attempt failed due to several reasons and an error message is returned.
Primary actors		User.
Secondary actors		Groceries App Database
trigger		An existing user taps on the login button.
Included cases		Verify credentials.
Main flow	Step	Action
	1	An existing user taps on the log-in button and system returns a log-in form to be filled with user credentials.
	2	User enters the credentials.
	3 Include: Verify Credentials	Credentials are verified from the database.
	4	User is logged in to his account and the main page is returned giving the user access to the application.

Use case name	**Log out**
Related requirements	FR 1.13 FR 1.14
Goal in context	A logged-in user is able to log out from the system.
Preconditions	User must be logged in and connected to the internet.
Successful end condition	User is logged out of his account.

Failed end condition		User is unable to log out.
Primary actors		User.
Secondary actors		None
Trigger		Logged in user taps on logout button.
Included cases		None
Main flow	Step	Action
	1	A logged in user taps on the logout button.
	2	Any changes or details are stored in the application database.
	3	The user is logged out, and login form is returned.

Use case name	**Forgot Password**
Related requirements	FR 1.15 FR 1.16 FR 1.17
Goal in context	Password is reset after the user's request.

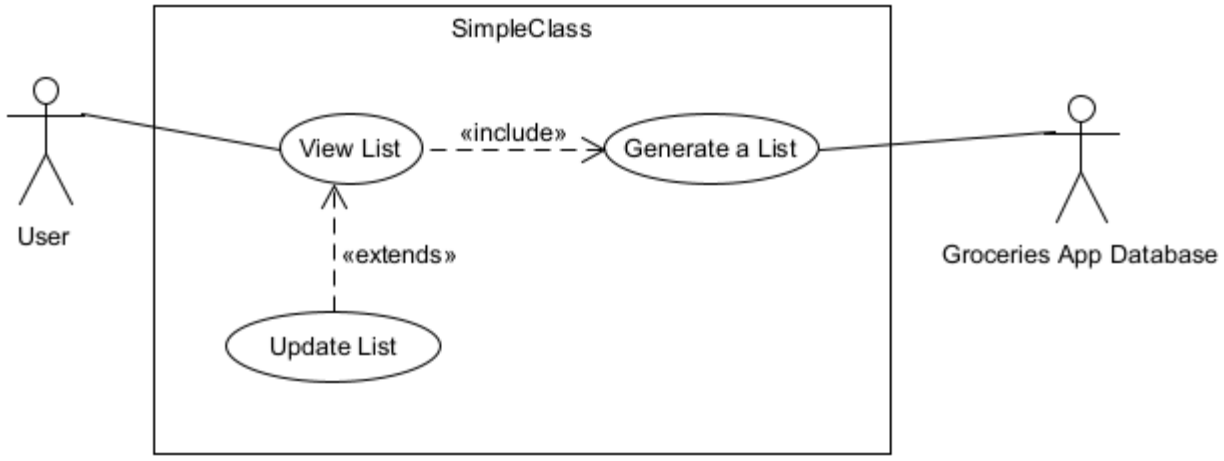
Preconditions		User account must exist and there must be an internet connection.
Successful end condition		An alternate password is provided through email, and the user is able to log in.
Failed end condition		The attempt to provide an alternate password fails.
Primary actors		User
Secondary actors		None
trigger		User taps on the forgot password button.
Included cases		Verify Credentials
Main flow	Step	Action
	1	User taps on the forgot password button.
	2	The system prompts the user to enter the email address used for their existing account.
	3	User enters the email address.
	4 Include: Verify Credentials	Email address is verified by the database.

	5	If an account exists, an email with a new password is sent. Else, an error message is shown.
	6	User enters the new credentials and is returned to the main page of the application with all the access.

Use case name	**Delete Account**
Related requirements	FR 1.18 FR 1.19 FR 1.20
Goal in context	A logged-in user is able to delete their account from the system.

Preconditions		User must be logged in and there must be an internet connection.
Successful end condition		User account is deleted, and all the data is cleared from the system's database.
Failed end condition		User is unable to delete the account.
Primary actors		User.
Secondary actors		None
Trigger		Logged in user taps on the delete account button.
Included cases		None
Main flow	Step	Action
	1	A logged in user goes to the settings page and taps on the delete account button.
	2	System prompts the user for confirmation of the delete action.
	3	If the user taps yes, the account and all the related information of the user is deleted and a login form is returned. Else, the delete function halts.

4.2 List Management



4.2.1 Description and Priority

List management enables the system to create a list based on the pattern of previous purchases by using a prediction algorithm. It also enables the user to update the list by giving an option to add or delete the items making it more customizable for the user. It is also a high priority feature as the application is based on the list generation.

4.2.2 Stimulus/ Response Sequence

Generate list: User taps on the view grocery list which prompts the system to generate and display the list. System implements the prediction algorithm and generates the list based on the data of previous purchases stored in the database. There should at least be data of three or more previous purchases for the prediction algorithm to generate a list. List is then displayed to the user.

Update List: User taps on update list. System will display the list with two options: add items or delete items. User can simply select the item from the list and delete it or add items they wish to in the list by selecting the add option. The changes are then saved into the database and the list is updated and displayed.

4.2.3 Functional Requirements

FR 2.1 System shall give an option to view the list.

FR 2.2 System shall check purchase entries from the database.

FR 2.3 System shall implement a prediction algorithm.

FR 2.4 System shall generate a list based on previous purchases.

FR 2.5 System shall display the list.

FR 2.6 System shall show an error message if the purchase list entries are less than three.

FR 2.7 System shall give an option to update the list.

FR 2.8 System shall give option to add or delete items from the list.

FR 2.9 Items shall be selected directly from the list for deletion.

FR 2.10 System shall delete the items from the database and update the list.

FR 2.11 System shall provide fields for addition of the items.

FR 2.12 System shall check the format and data types of the data being filled.

FR 2.13 System shall add the data in the database and update the list.

FR. 2.14 System shall show an error message if data entered is invalid or incorrect.

Use case name	**Generate List**
Related requirements	FR 2.1 FR 2.2 FR 2.3 FR 2.4 FR 2.5

		FR 2.6
Goal in context		By using a prediction algorithm, a grocery list is generated based on the past purchases.
Preconditions		User must be logged in. There must be internet connection. User must have entered data of three or more purchases.
Successful end condition		System creates a list after finding a pattern in past purchases and using a prediction algorithm.
Failed end condition		List is not generated because there is not enough data provided to do prediction on.
Primary actors		Groceries App Database
Secondary actors		User
Trigger		Logged in user taps on view grocery list button.
Included cases		None
Main flow	Step	Action
	1	A logged in user taps on the view grocery list button.

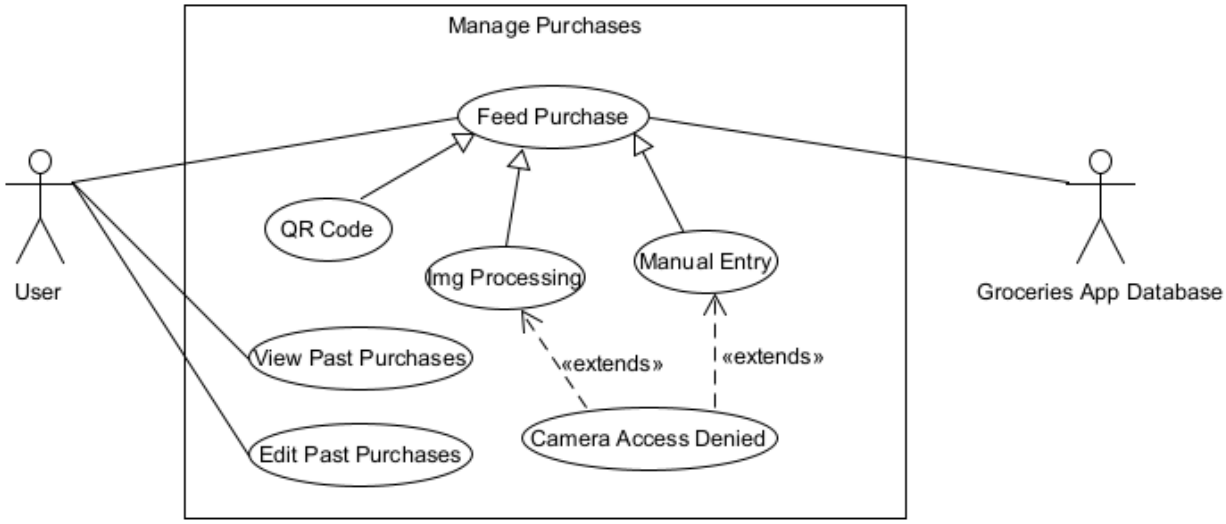
	2	System carries out the implementation of the prediction algorithm.
	3	If purchases are more than 3, the system displays a list. Else, an error message is shown.

Use case name	**Update List**
Related requirements	FR 2.7 FR 2.8 FR 2.9 FR 2.10 FR 2.12 FR 2.13 FR 2.14
Goal in context	A logged-in user is able to update the list by adding and deleting items.
Preconditions	User must be logged in and connected to the internet.

		There must be a list which can be updated.
Successful end condition		User is able to make changes in the list by adding and deleting items.
Failed end condition		User was unable to update the list because data entered was invalid.
Primary actors		User.
Secondary actors		None
Trigger		User taps on the update button.
Included cases		None
Main flow	Step	Action
	1	A logged in user taps on the update button.
	2	System displays two options, one for adding items and one for deleting items.
	3	User selects either of the two and adds or deletes items.
	4	The changes are then saved into the database.

	5	System updates the list and the user can view the changes.

4.3 Manage purchases



4.3.1 Description and Priority

Manage purchases lets the user enter their previous grocery purchases receipts/lists easily in the system's database which the system can then use to generate grocery lists. It also lets the user edit their purchase history for a better optimized list.

4.3.2 Stimulus/ Response Sequence

Feed Purchase: User taps on the feed purchase button which prompts the system to provide three options of entering the purchase, manually, through a QR code and by taking a picture of the receipt. User selects one of these options and enters the data. System asks for confirmation before making any changes to the database. Upon the confirmation, data is saved into the database.

Edit purchase history: User taps on edit purchase history button. System then asks the user to select the date of the purchase list that the user wants to edit. Upon selection, the system displays the list to be edited. User deletes or make changes to the data. User taps on the update button and the system updates that data in the database.

4.3.3 Functional Requirements

FR 3.1 System shall give an option of entering purchase.

FR 3.2 System shall give an option to enter the data manually.

FR 3.3 System shall give an option to enter data by scanning a QR code.

FR 3.4 System shall give an option to enter data by taking a picture.

FR 3.5 System shall ask for camera access for scanning purposes.

FR 3.6 System shall ask for confirmation from the user for adding the data to the database.

FR 3.7 System shall display an error message if confirmation or access is denied.

FR 3.8 System shall add data of purchases in the database.

FR 3.9 System shall give an option to edit purchase history.

FR 3.10 System shall ask the user for selecting a purchase list date.

FR 3.11 System shall display the purchase list added on the date selected by the user.

FR 3.12 System shall let user delete the entries user wishes to.

FR 3.13 System shall let user alter the entries.

FR 3.14 System shall have an update purchase history option.

FR 3.15 System shall update the changes/ data in the database.

Use case name	**Feed Purchase**

Related requirements	FR 3.1 FR 3.2 FR 3.3 FR 3.4 FR 3.5 FR 3.6 FR 3.7
Goal in context	Previous grocery purchases are added to the database.
Preconditions	User must be logged in.
Successful end condition	List of purchased items is added to the database.
Failed end condition	Adding of items failed due to one or more reasons i.e., wrong data, scanning failed, camera access was denied etc.
Primary actors	User.
Secondary actors	Groceries App Database
Trigger	Logged in user taps on feed purchase button.
Included cases	None

Main flow	Step	Action
	1	User taps on the feed purchase button.
	2	System prompts the user to add the details through either of the three options : 1. Manually. 2. By scanning QR code. 3. By taking a picture of the receipt.
	3	User feeds the past purchased items by selecting one of the options.
	4	Manually, the user adds items one by one, along with the store name and once everything is added, taps on the “add items” button. System asks the user for confirmation of this action.
	5	After confirmation, a list of purchased items is added to the database. Else, an error message is displayed stating the reason.
	6	If it is through scanning QR code or by taking a picture, the system prompts the user for confirmation of camera access.
	7	If access is given, the user can scan / take the picture and data from the receipt is added to the database. Else, an error message is shown.

Use case name	**Edit Purchase history**
Related requirements	FR 3.8 FR 3.9 FR 3.10 FR 3.11 FR 3.12 FR 3.13 FR 3.14 FR 3.15
Goal in context	User can edit the purchase history in the database.
Preconditions	User must have an internet connection and must be logged in.
Successful end condition	Purchase history is updated in the database.
Failed end condition	Data was not updated due to one or more reasons i.e., Incorrect data etc.
Primary actors	User.
Secondary actors	None

Trigger		Logged in user taps on edit purchase history.
Included cases		None
Main flow	Step	Action
	1	User taps on the edit purchase history button.
	2	System prompts the user to select a purchase date.
	3	Once the user selects the date, the system shows the data of the selected receipt to be edited.
	4	User then edits the list by deleting an item or by changing the quantity or brand.
	5	User taps on the update button and the system updates the data in the database.

4.4 Inventory Management

4.4.1 Description and Priority

Inventory Management is a feature of medium priority keeping in mind the application's purpose. This feature creates and updates the inventory for the user automatically. It allows the user to view the inventory when it is created or when any changes are made to it. It also allows the user to delete the inventory manually and receive updates regarding the inventory on a daily basis.

4.4.2 Stimulus/ Response Sequence

Create Inventory: Once the user has made their purchase, they will feed it to the app either via QR code, Image processing or manual entry. The app will have to make a request to the mobile phone to access the camera in order for the user to feed purchases via QR code/Image processing. The user will then scan the QR code on the receipt, or they will take a photo of the entire receipt and feed it into the app. Upon receiving this, the app will prompt the user to either create an inventory or just feed in the purchase. Upon clicking create an inventory the app will extract all the necessary information, convert it to a list and automatically create an inventory out of it. If the user is to feed in purchases manually, they can opt for that option and one by one list down all the items in the receipt, along with the price. They will then get an option to either create an inventory or just feed the purchase. If they click on the create inventory button, a new inventory will be created, else only the purchase will be fed. All the purchases will be stored in the Groceries App Database. Which will then be backed up in the cloud.

Update Inventory: Once the user has created their inventory, they can update it either through feeding in purchases via QR code/Image processing or manually. If they feed in purchases via QR code/Image processing, they receive an option to add to inventory out of this purchase. If they click on this option, the inventory is updated based on the current purchase. Otherwise, no inventory will be created, and a record of the purchase will just be stored. If they feed in a purchase manually, they will have an option to add to inventory based on the current purchase, otherwise, only a record of the purchase will be stored. The decrease in the items in the inventory will be based on the buying patterns of the user, to give an effect that the inventory is updating as the user consumes items.

View Inventory: Once the inventory has been created and the user wants to view it, they will launch the app and must be signed into their account. They will then have to navigate to the home page and click on the view inventory button to view the items in their inventory. The quantity won't be listed, however, there will be a health bar next to each item which will increase, and decrease based on the user's buying habits and consumption.

Delete Inventory: If the user wishes to delete the inventory, they must first view the inventory. Once they are viewing the inventory, they will click on the bin icon in the inventory section for the delete option to show. Upon clicking the delete option the app will ask for a confirmation for the deletion, saying that this action won't be undone. The inventory will then be deleted permanently.

Send Inventory Update: For the user to receive prompt updates regarding their inventory, the notification settings for the app must be turned on first through the phone's settings. Then the inventory notification within the *change notification settings* of the app must be turned on. The buying patterns of the user and a certain time will trigger the application to send inventory updates to the user.

4.4.3 Functional Requirements

FR 4.1 The application will show a feed purchase button on the home screen.

FR 4.2 The user can feed a purchase via QR code/Image processing.

FR 4.3 The user can feed a purchase manually.

FR 4.4 Create inventory button will appear after a purchase is fed via QR code/Image processing or manually.

FR 4.5 Clicking this button will cause the app to extract all necessary information from the receipt and generate an inventory from it, in the case of QR code/Image Processing.

FR 4.6 For manual entry, the app will provide an interface where the user can enter the items and total cost from the receipt manually.

FR 4.7 Create inventory button will appear to allow the user to make an inventory from their manual list.

FR 4.8 A feed purchase only option will also appear with the create inventory button in both cases, if the user wishes to feed an old purchase receipt.

FR 4.9 The user can view the inventory after clicking on the view inventory button on the home page.

FR 4.10 The user will be allowed to add a purchase to an already existing inventory through the add to inventory button.

FR 4.11 The user will be allowed to only feed a purchase to the database if they want to feed an old purchase, through the feed purchase only option.

FR 4.12 The inventory will update whenever a purchase is added.

FR 4.13 The inventory will be updated based on the buying patterns of the user.

FR 4.14 The user will be allowed to delete an inventory by pressing on the bin button in the inventory section.

FR 4.15 The user will then be prompted to confirm the deletion of the inventory.

FR 4.16 Following the deletion, the deleted inventory cannot be restored.

FR 4.17 The past purchases of the users, however, will not be affected by any inventory deletion.

FR 4.18 The app will send inventory updates to the user based on their buying patterns.

FR 4.19 These updates will be most frequent when some item in the inventory is running out.

FR 4.20 These updates will be less frequent when the inventory is full.

FR 4.21 The user can change the inventory notification settings in the change notification section of the settings page.

Use case name	Manage Inventory
Related requirements	FR 4.1 FR 4.2 FR 4.3 FR 4.4 FR 4.5 FR 4.6 FR 4.8 FR 4.9 FR 4.10 FR 4.11 FR 4.12 FR 4.13 FR 4.14 FR 4.15 FR 4.16 FR 4.17 FR 4.18 FR 4.19 FR 4.20 FR 4.21
Goal in context	The user will be able to manage their inventory. They will be able to create, view, delete, update, and receive inventory updates.

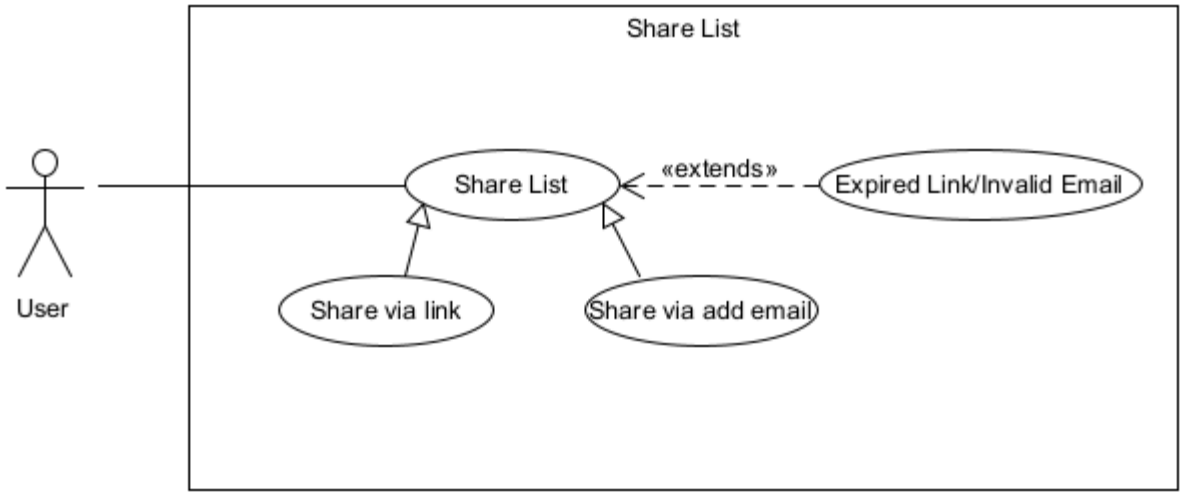
preconditions	1. For creating an inventory, the user must first feed in a purchase. 2. For viewing an inventory, an inventory must exist. 3. For deleting an inventory, an inventory must exist beforehand, and the user must be viewing it for the delete option to show. 4. For updating an inventory, the user must feed in a purchase. Or they must consume their items. 5. The notifications must be turned on for the app, for the user to receive updates.
Successful end condition	The user can create, update, delete and view the inventory. They can also receive inventory updates.
Failed end condition	The user is not able to create, update, delete and view the inventory. They do not receive inventory updates.
Primary actors	User
Secondary actors	None

trigger		1. The user clicks on the create inventory button to generate an inventory. 2. For deleting an inventory, the user selects and holds on the inventory for the delete option to show. They then click on it. 3. For viewing an inventory, the user clicks on the inventory button on the home page. 4. For updating an inventory, the user clicks on the add to inventory button. Or, as the user consumes the items, the inventory updates itself. 5. A certain time of the day triggers the app to send a notification to the user.
Included cases		None
Main flow	step	Action
Create inventory	1	The user feeds in a purchase via QR code/Image processing or manually.
	2	If the user feeds it via QR code/Image processing, they receive an option to create an inventory out of this purchase. If they click on this option, an inventory is made for them based on the current purchase.
	3	If they click on the “Just feed it in” option, then no inventory will be created, and a record of the purchase will just be stored.

	4	If the user feeds in the receipt manually, they receive an option to create an inventory out of this purchase. If they click on this option, an inventory is made for them based on the current purchase. If they click on the “Just feed it in” option, then no inventory will be created, and a record of the purchase will just be stored.
View inventory	1	The user opens the application and logs in
	2	The user clicks on my inventory banner on the home page.
	3	The user views their inventory.
Update inventory	1	The user feeds in a new purchase via QR code/Image processing.
	2	They receive an option to add to inventory out of this purchase. If they click on this option, the inventory is updated based on the current purchase.
	3	If they click on the “Just feed it in” option, then no inventory will be created, and a record of the purchase will just be stored.
	4	Or the user consumes the items in their inventory and the inventory updates itself based on the user’s buying patterns.
	5	If the user enters the items manually, then they must click the add to inventory button for the inventory to update. Otherwise, a record of the purchase will just be stored.

Delete inventory	1	The user clicks on the bin icon in the inventory section for the delete option to show.
	2	The user clicks on the delete inventory option to delete the inventory.
Send Inventory Update	1	The user will turn on the notification settings.
	2	The buying patterns of the user and a certain time will then trigger the app to send inventory updates to the user.

4.5 Share List



4.5.1 Description and Priority

Share list is a feature of medium priority keeping in mind the application's purpose. This feature allows the user to share their grocery and inventory lists with their family members so each

member can keep track of the lists, make changes and update others regarding which item needs buying and which doesn't.

4.5.2 Stimulus/ Response Sequence

Share via Link: Once the user is signed into the app and has created both grocery and inventory lists, they will be able to share both the lists through the share link feature. The home page of the app will consist of a share lists button, which when clicked will open up a page consisting of a send invite via WhatsApp and send invite via SMS option. When the first option is clicked, the app will redirect the user to WhatsApp where they can forward the link to anyone. When the second option is clicked, the app will redirect the user to messages in the phone, where they can enter the phone numbers of users and forward the link to them. The link in both cases will only be accessible to those users to which the link has been originally forwarded to. After a certain period of time, or when the link has been accessed, the link will expire.

Share via Add Email: Once the user is signed into the app and has created both grocery and inventory lists, they will be able to share both the lists through the share link feature. The home page of the app will consist of a share lists button, which when clicked will open up a page consisting of a send invite via adding emails. The user will, one by one, enter the email addresses of all the people they want to share the list with and click on the send invite button to forward the link to them. The link in both cases will only be accessible to those users to which the link has been originally forwarded to. After a certain period of time, or when the link has been accessed, the link will expire

4.5.3 Functional Requirements

FR 5.1 There will be a share link button on the home page.

FR 5.2 There will be a send invite via WhatsApp option.

FR 5.3 There will be a send invite via SMS option.

FR 5.4 There will be a send invite via adding emails option

FR 5.5 The app redirects the user to WhatsApp so they can forward the link to anyone.

FR 5.6 The app redirects the user to messages so they can send the link to any phone number.

FR 5.7 The user will enter email addresses one by one and send the link to each individually.

FR 5.8 The email address must be of an already existing user.

FR 5.9The Phone number must be a valid one.

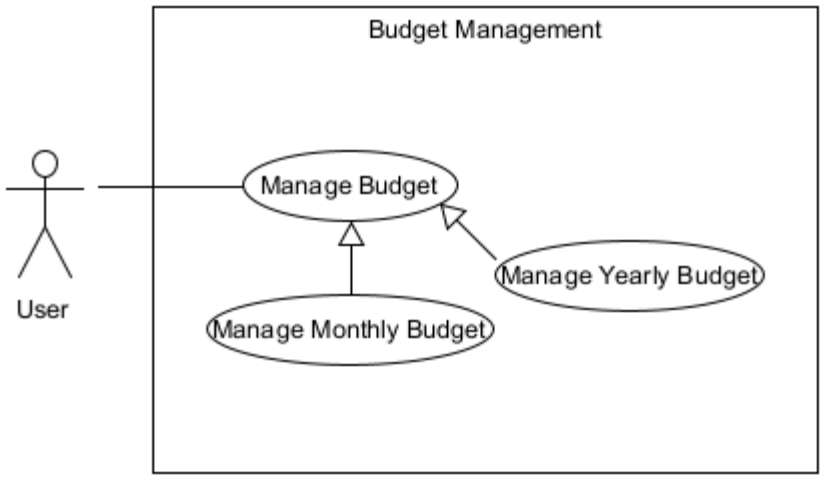
FR 5.10 The link will expire within a fixed period of time.

FR 5.11 The link will expire once it has been accessed.

Use case name	Share List
Related requirements	FR 5.1 FR 5.2 FR 5.3 FR 5.4 FR 5.5 FR 5.6 FR 5.7 FR 5.8 FR 5.9 FR 5.10
Goal in context	The inventory and predicted lists will be shared among already existing users, which can be accessed and to which changes can be made by all parties.
Preconditions	1. Accounts of users to which the lists will be shared must exist beforehand. 2. The phone numbers to which the link will be shared should be valid.
Successful end condition	Users who have been added via email, or to which a link has been sent, can view, and make changes to the lists.
Failed end condition	Expired link / Invalid email. Hence, no access to the lists.
Primary actors	User

Secondary actors		None
trigger		The send invites button is clicked if users are to be added via their email addresses or if a link is to be sent to the users via WhatsApp or SMS.
Included cases		None
Main flow	step	Action
	1	The home page will have a share list option, which can be clicked.
	2	Share via add email and share via link option will appear.
	3	The user can add the email addresses of already existing users and click on send invites to grant them access to the lists.
	4	The user can also send invites to existing users through WhatsApp and SMS.

4.6 Budget Management



4.6.1 Description and Priority

Budget management is a feature of low priority in context with the application's purpose. It allows the user to manage their budgets (Monthly or Yearly). The user can set a monthly or a yearly budget and also maintain a track of their previous expenditures. This feature also allows the user to receive prompt updates regarding their budgets and helps them in making efficient purchases and saving money.

4.6.2 Stimulus/ Response Sequence

Manage Monthly Budget: Once the user is signed into the app, they can view the manage budget button on the home page. Upon clicking this button they will receive two options, *Manage Monthly Budget* and *Manage Yearly Budget.* If they click on the manage monthly budget button, they will be able to enter and set their monthly budget which will update as the user keeps on feeding purchases.

Manage yearly Budget: Once the user is signed into the app, they can view the manage budget button on the home page. Upon clicking this button they will receive two options, *Manage Monthly Budget* and *Manage Yearly Budget.* If they click on the manage yearly budget button, they will be able to enter and set their yearly budget which will update as the user keeps on feeding purchases.

Send Budget Updates: Once the user has set their budgets, they can turn on the send budget notification option, provided the apps notifications are turned on. The user will then receive prompt budget notifications that will help them with managing their budget.

4.6.3 Functional Requirements

FR 6.1 The application has a manage budget button on its home page.

FR 6.2 There is a manage monthly budget button for setting monthly budgets.

FR 6.3 There is a manage yearly budget button for setting yearly budgets.

FR 6.4 The user can always view their budgets by navigating to the manage budgets section.

FR 6.5 Whenever a new purchase is fed by the user, the budget will change accordingly

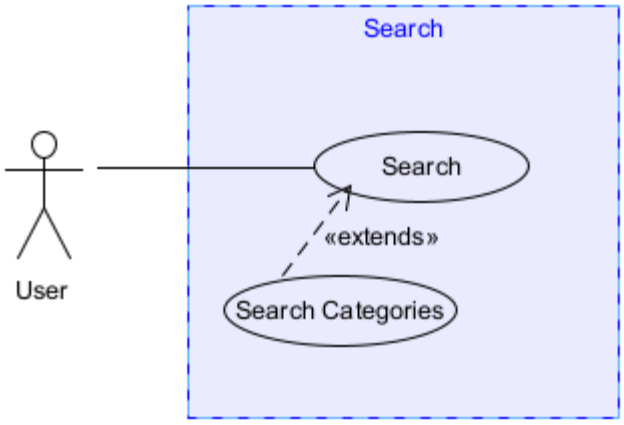
FR 6.6 The user will receive prompt budget notifications.

Use case name	Manage Budget
Related requirements	FR 6.1 FR 6.2 FR 6.3 FR 6.4 FR 6.5 FR 6.6
Goal in context	The user will be able to manage their monthly and yearly budget. The user will be able to set their monthly and yearly budgets; Whenever a purchase is fed to the app, the monthly and yearly budgets will change accordingly. This helps the user in making efficient purchases and saving money. The user will also receive notifications to update them regarding their budget.

preconditions	1. For entering a budget, the user must only be signed in to the app. 2. For setting monthly and yearly budgets, the user must enter the respective budgets. 3. For receiving notifications, the app notifications must be turned on.
Successful end condition	The user is able to set and manage their monthly and yearly budgets. They will receive prompt notifications updating them regarding their budget.
Failed end condition	1. The user is not able to enter their budgets. 2. The monthly and yearly budgets do not decrease after feeding in a purchase. 3. The user does not receive notifications even though the app notifications are turned on.
Primary actors	User
Secondary actors	None
trigger	1. For setting monthly and yearly budgets, the user enters the budgets and clicks on the set budget button. 2. For receiving updates, a certain time of day triggers the app to send a notification to the user.

Included cases		None
Main flow	step	Action
Set Monthly/Yearly Budget	1	The user clicks on the manage budgets button on the home page.
	2	The user chooses between the monthly and yearly budgets to set and enters a budget for either or both of them.
	3	Whenever a purchase is fed by the user, the monthly and yearly budget changes accordingly.
Budget update	1	The user turns on the app notifications.
	2	After setting the budgets, a certain time of day will trigger the app to send notifications to the user.

4.7 Search



4.7.1 Description and Priority

Search is a low priority feature based on functionality of the application. Search option will allow the user to search the inventory for a specific item; also, there will be categorization of items i.e., foods, stationary, electronics, dairy etc. for user to browse through, having a search option allows for easy and helpful user experience and is almost a must in any application using lists.

4.7.2 Stimulus and response

Search button: when user presses the search button; the application will shift to search page from current page where a input block has been displayed, the user will then input the thing he want to search for i.e., if a user wants to search for pencils; they will write pencils in the input bar after pressing the search button algorithms will select the pencils in the recorded inventory list database and display them as a form of list. This is a typical search and retrieve method.

Categories button: when the user presses the categories button; the application will open the categories page in which multiple categories name + pictures will be displayed for better visual effect; the user will then select the wanted categories; the algorithm will then use the database to search and retrieve same categorized item and will then display those items as a form of list to the user.

4.7.3 Functional requirements:

FR 7.1 all items must be categorized at the time of input.

FR 7.2 the application must have a search button and page.

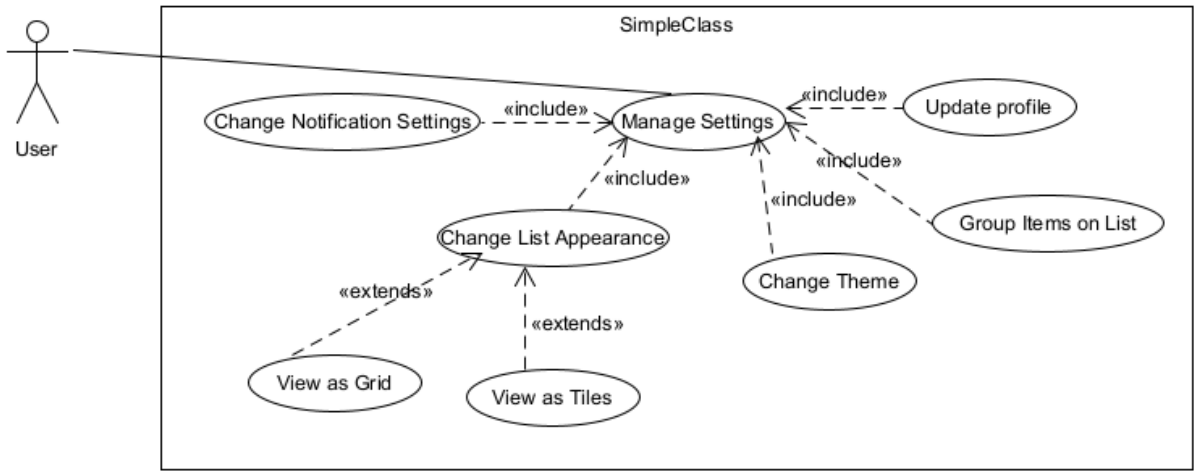
FR 7.3 The application must have a categories button and page.

FR 7.4 application should be connected to the database for items search transversal and retrieval.

Use case name	Search
Related requirements	FR 7.1 FR 7.2 FR 7.3 FR 7.4
Goal in context	The user is able to search for whatever they are looking for. Be it a specific category of items or items within the inventory itself etc.
preconditions	1. The user must be signed in to the app.
Successful end condition	The user is able to search for whatever they are looking for.
Failed end condition	A wrong search result is shown/ No search result is shown.
Primary actors	User
Secondary actors	None

trigger		Clicking on the search text area and entering the search query.
Included cases		None
Main flow	step	Action
	1	The user signs in to the app.
	2	The user clicks on the search text area and enters their search query.
	3	Search results related to the query are displayed or no results are displayed.

4.8 Manage Settings



4.8.1 Description and Priority

The manage settings feature is of medium priority in context with the application's purpose. This feature allows the user to personalize their end of the application, update their privacy information, change their list viewings, and update the application's notification settings.

4.8.2 Stimulus/ Response Sequence

Change Theme: The change theme sub-feature allows the user to toggle between the light and dark mode of the application. To access this sub-feature, the user will have to be signed in to the application. They will then navigate to the settings/profile page of the application. There they will locate the change theme option. Upon clicking it, the user will be shown a toggle button. Clicking on this button will change the theme of the app from light to dark and vice versa.

Change List view: The change list view sub-feature allows the user to change the list views from grid to tiles and vice versa. To access this sub-feature, the user will have to be signed in to the application. They will then navigate to the settings/profile page of the application. There they will locate the change list view option. Upon clicking it, the user will be shown two options, change list to grid view, and change list to tiles view. The user will then click on the view button they want their list to look like. The predicted and inventory lists will change accordingly.

Group Items on List: The group items on list sub-feature allows the user to group items in the lists that are similar to each other. To access this sub-feature, the user will have to be signed in to the application. They will then navigate to the settings/profile page of the application. There they will locate the group items on the list option. Upon clicking it, the user will be shown a toggle button. The user will then click on the button and the items in the lists will be grouped together based on their similarity.

Change Notification Settings: The user can change notification settings for inventory updates, budget updates and general app notifications through the change notification sub-feature. To access this sub-feature, the user will have to be signed in to the application. They will then navigate to the settings/profile page of the application. There they will locate the change notification settings option. Upon clicking it, the user will be shown toggle buttons for inventory, budget, and other general app notification options. Clicking on these buttons will update the respective notification settings.

Update Privacy Settings: This sub-feature allows the user to update their privacy settings, including but not limited to changing their name, their profile picture, their email address, password etc. To access this sub-feature, the user will have to be signed into the application. They will then navigate to the settings/profile page of the application. They will then locate the change privacy settings option. Upon clicking it, the user will be directed to a page where they can change their name, their profile picture, their email address, password and more. Any changes made will have to be verified by the user through their password. Changes will be saved, and the data will be updated in the database.

4.8.3 Functional Requirements

FR 8.1 The entire application can be viewed in light mode.

FR 8.2 The entire application can be viewed in dark mode.

FR 8.3 The lists can be viewed as a grid.

FR 8.4 The lists can be viewed as tiles.

FR 8.5 The items in the list can be grouped together based on their similarity.

FR 8.6 The inventory updates can be turned on and off.

FR 8.7 The budget updates can be turned on and off.

FR 8.8 General app updates can be turned on and off.

FR 8.9 The user can update their profile picture

FR 8.10 The user can update their name.

FR 8.11 The user can change their password.

FR 8.12 The user can change their email address.

FR 8.13 All changes will be verified by the user's credentials.

FR 8.14 All changes will be saved and updated in the database.

Use case name	Manage Settings
Related requirements	FR 8.1 FR 8.2 FR 8.3 FR 8.4 FR 8.5 FR 8.6 FR 8.7 FR 8.8 FR 8.9 FR 8.10 FR 8.11 FR 8.12 FR 8.13 FR 8.14
Goal in context	The goal is to allow the user to manage their account settings. These include personalization settings, notification settings and privacy settings.

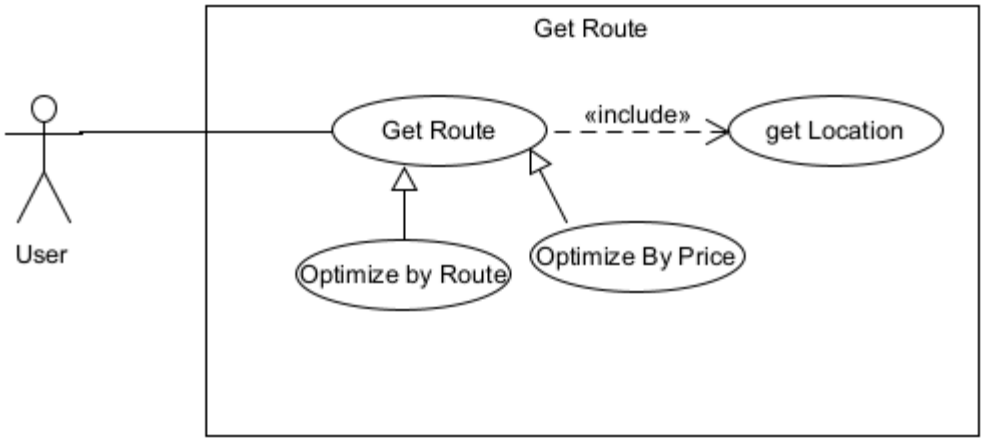
Preconditions	1. For personalization settings, such as: Light/Dark mode, viewing list in a grid/tiles form and grouping items on a list, the user should only be signed into the app. 2. For notification settings, the user must be signed into the app. 3. For privacy settings, the user must be signed into the app, and they must verify their credentials to save changes.
Successful end condition	The user is able to personalize their account, they are able to modify notification settings and they are able to update their privacy settings.
Failed end condition	Any changes made are not reflected by the app.
Primary actors	User
Secondary actors	None

trigger		1. For switching between light/dark modes, the user will click on the toggle button. 2. For switching the list view between grid and tiles, the user will click on the respective buttons. 3. For grouping items on a list, the user will click on the respective button. 4. For changing notification settings, the user will click on the toggle buttons to change settings for inventory, budget, and general app updates. 5. For updating privacy info, the user will click on the save changes button after they have verified their credentials.
Included cases		None
Main flow	step	Action
Change Theme	1	The user navigates to the settings/profile page
	2	The user locates the change theme option and clicks on it
	3	The user clicks on the toggle button to change themes.
Change List View	1	The user navigates to the settings/profile page

	2	The user locates the change list view option and clicks on it
	3	The user then clicks on the view button they want their lists to look like.
Group Items on List	1	The user navigates to the settings/profile page
	2	The user locates the Group items on list option and clicks on it
	3	The user then clicks on the toggle button to group the items in the list that are similar to each other.
Change Notification Settings	1	The user navigates to the settings/profile page
	2	The user locates the change notification settings option and clicks on it
	3	The user then clicks on the toggle buttons for inventory, budget, and general app notifications to change their notification settings.
Update Privacy Settings	1	The user navigates to the settings/profile page
	2	The user locates the Privacy option and clicks on it

	3	The user may change their name, their profile picture, their email address, and password.
	4	Any Changes made will be verified by the user through their password. Changes will be saved after clicking the save changes button. The data will be updated in the database.

4.9 Get Route



4.8.1 Description and Priority

get route is a high priority feature basing on functionality of application. Get route is an optimization feature on an attained inventory list, which helps/allows the user to filter their list by options (shortest distance or lowest cost).in Get route, the user can filter their list to find the closest recorded stores based on route or based on lowest cost comparison between recorded stores in the application. Get route feature output works in accordance with google locations/maps to allow user more accessibility but when only conditional requirements are met.

4.9.2 Stimulus and response

After the inventory list has been generated. There will be a filter button; by pressing that button; The user can filter the inventory list through following two options:

Filter By Route: once the inventory list has been generated, the user presses the filter button; then select the route option; the user can filter the list by route comparison to get the shortest distance to reach between the recorded stores in application where purchases have been made already; once that store has been selected by the algorithm, the parameter(stores' name) is passed to google maps and the option to shift page, to view route in google map-google map route selection button- will be shown in info box on the app page, the user then selects that button which will shift from our app page to google map; in which route by shortest distance, time taken to reach that store, traffic on that route and other functionality of google maps can into play to help the user make the decisions better; that ultimately leads to better application performance.

Filter by cost: once the inventory list has been generated, the user presses the filter button; then select the cost option; the user can filter the list to find the store and route to that store from where the item the user purchased costed less as compared to other recorded stores; by selecting the cost filter option; algorithm performs lowest price comparison between items purchased at recorded stores in the past-based on lowest price- the store is selected; the parameter(stores' name) is passed to google maps and the option to shift page, to view route in google map-google map route selection button- will be shown in info box on the app page along with stores name and maybe the total price difference compared to other stores, the user then selects that google maps button which will shift from our app page to google map; in which route by shortest distance, time taken to reach that store, traffic on that route and other functionality of google maps can into play to help the user make the decisions better; that also ultimately leads to better application performance.

4.9.3 Functional Requirements

FR 9.1 The application will show a filter button on the page.

FR 9.2 The filter button will show the buttons of following

A. Filter by Cost B. Filter By Route

FR 9.3 The application must be connected to a workable internet connection.

FR 9.4 The store name will be displayed on the app\ page after algorithm optimization based on route or cost option.

FR 9.5 The store name will be passed Google Maps to show the button for redirecting the page

FR 9.6 The user can select Google Maps button to view the route to the outputted store

FR 9.7 The Google Maps will display all its functionality for the route to reach the store.

FR 9.8 The location option on the user phone should also be turned on for better application experience.

Use case name	Get Route
Related requirements	FR 9.1 FR 9.2 FR 9.3 FR 9.4 FR 9.5 FR 9.6 FR 9.7 FR 9.8
Goal in context	The user gets the route of the supermarket that is either optimized based on the price, or that is optimized based on the distance from their home.
Preconditions	1. The app must request location access from the mobile phone.

Successful end condition		The name of the supermarket is displayed on a button next to the total cost of the items in the predicted list. Clicking on the button redirects the user to google maps where they can see the route from their home to the supermarket.
Failed end condition		1. The name of the supermarket is not displayed on the button. 2. Clicking on the button does not redirect the app to google maps.
Primary actors		User
Secondary actors		None
Trigger		1. Clicking on the optimized by cost button 2. Clicking on the optimized by destination button.
Included cases		Get Location
Main flow	Step	Action
Optimized by cost	1	The app is granted location access by the mobile phone.
	2	The user clicks on the predict a list button to generate a list based on the user's buying habits.

	3	The user then clicks on the optimized by cost button to generate a destination from where the user can get the items the cheapest.
Optimized by Destination	1	The app is granted location access by the mobile phone.
	2	The user clicks on the predict a list button to generate a list based on the user's buying habits.
	3	The user then clicks on the optimized by destination button to generate a destination which is closest to the user's home.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The grocery list app requires a system with at least 2GB RAM.
- The minimum storage requirements for this application are a 16GB ROM.
- The mobile phone should be at least working on Android 7 for Android users.
- The mobile phone should be at least working on IOS 11 for IOS users.

- The routing between pages must be seamless.
- Redirection of the app to WhatsApp, Google Maps and messages should not take longer than 2 to 3 seconds.
- For features like connecting to Google maps, the user should have an active internet connection.
-
- The email verification following SignUp must be verified within a minute or the code will be expired and must be sent again.
- If the user is not logged in within a minute, they must re-enter their credentials and try again.
- The forget password link must be accessed within a minute otherwise it will have to be sent again.
- The app will function in a series of sessions. Any inactivity of more than 30 minutes will expire the session and the user must Login again.
- The share link for the lists must be accessed within 30 minutes of it being sent, otherwise the link will expire.
- Notifications must be acknowledged by the user within a certain period of time, otherwise they will be sent again.
- Notifications must open up within a maximum of 2 seconds.
- The application must generate a list for the user upon request within a maximum of 10 seconds.
- The Cost and destination should be dynamically updated as the user optimizes by cost or distance and when the user updates the list.
- Purchases fed in by QR code or Image Processing must be converted to a list within a maximum of 10 seconds.
- Inventory must be created within 5 seconds of requesting about it.
- The user must be logged out of the application within 10 seconds of requesting it.
- Any changes in theme, list or notification settings must be in effect instantaneously.

5.2 Safety requirement

- If the user accesses the app from another device, a verification email must be sent to them to authorize their use.
- The application must always ask the user for access to their location.
- The application must always ask the user for access to their mobile phone’s camera.
- Due to a virus or an OS failure, the database could be destroyed at any time. Therefore, our database will be regularly backed up on the cloud to prevent any data loss for the user.

- There will be a bug report feature in our app, which the user can use to report any difficulties or bugs to improve the overall performance for our app.

5.3 Security requirement

- All the user's personal information including name, IP address, email address, password etc. will be stored in a secured database and needs user permission for access.
- If a user shares their list to another user they will have an option to make that list read-only or editable.
- SignUp and any privacy changes of the user will be verified by their credentials for changes to be saved.
- After 30 minutes of the application being idle, the user will be logged out and they must sign in again.
- Anonymization & aggregation, so that route information may be shared safely without disclosing personal information.
- In order to create a viable offering for the user we will build a simple, transparent system that can be understood and trusted by the people that are using it.
- Encryption, for all data that is privacy sensitive.
- Open source / disclose security policies & practices.
- For account deletion, the user will be asked to verify their credentials. Following this all the data of the user will be deleted from the database and cloud.

5.4 Software quality attributes

Adaptability: The application is intended to work specifically for IOS and Android environments. There will be no performance differences regarding the features of the app in both the environments provided the user has the latest versions of the OS installed. If this is not the case, there may be performance differences and some features such as list prediction and route prediction might not function properly. The application might also not work for versions lower than that specified in this document.

Availability: The application is guaranteed to work whenever the user clicks on the icon on their phone's screen. It is made to be reliable and will perform all of the functionalities listed in this document. Be it predicting, inventory creation, route optimization etc. The user will always be able to use the app the way it is intended to unless the app requires the user to install an update.

Correctness: The user will always be able to use the app the way it is intended to, with all the components/features of the app working in the way defined in this document. If for some reason the features don't work in the intended way, the user can report the abnormalities through the bug report tool.

Flexibility: Even though the application is designed to be run for IOS and Android, it is designed in such a flexible way that with minor changes it can work on a different platform such as HarmonyOS for Huawei.

Interoperability: The application allows users to share their lists amongst each other. Android users will be able to share their lists between IOS users and vice versa. Users in both environments will be able to view each other's lists in the same format.

Reusability: The application is designed in such a way that it will reduce re-coding and re-working on individual components. These components can then be reused with ease if a similar application to this is created in the future.

Usability: Our software will consist of a user manual defining all the necessary guides on how to use the app the way it is intended to be used.

Testability: This software consists of various testable components. Separate test units/functions are defined within the code that can be run individually to test their functionality and to see their cohesiveness with the overall application.

5.5 Business rules

- For the "where to buy" business requirement, our business rule will be that our app will provide routes to only those stores where the user has shopped in the past.
- The 'optimize by price' use case will be based upon the history of past purchases therefore the latest price might be different due to inflation rate.
- It is suggested to always check before buying. (guest or outstation constraint)
- The owners of the product are not responsible for unavailability of products and their quality.

6. Other Requirements

There are no other requirements.

Appendix A: Glossary

- Algorithm

A process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.

- Backend

Backend is a generic term whose meaning changes according to what part of an application or software infrastructure you are working on.

- Cloud

Cloud storage is a service model in which data is transmitted and stored on remote storage systems, where it is maintained, managed, backed up and made available to users over a network -- typically, the internet.

- Dark Mode

Dark Mode is an extension that helps you quickly turn the screen to dark at night time.

- Database

Structured set of data held in a computer, especially one that is accessible in various ways.

- Developers

An IT professional who uses programming languages to create computer software.

- Generate

Produce or create.

- Image processing

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it.

- Inventory

A complete list of items such as property, goods in stock, or the contents of a building.

- Light Mode

Dark-font text on light background

- Navigate

Plan and direct the form of transport, especially by using instruments or maps.

- Notification

The act or an instance of giving notice or information

- Preconditions

A condition or predicate that must always be true just prior to the execution of some section of code.

- Primary actors

The stakeholder that calls on the system to deliver one of its services.

- QR code

QR codes are frequently used to track information about products in a supply chain

- Secondary actors

A person, business processes, or applications that provides a specific result or information to a use case in order for the end goal of the use case to be achieved.

- Trigger

to cause something to start

- User Data

All data, information and other content of any type and in any format, medium or form

- User Privacy

Privacy is the ability of an individual or group to seclude themselves or information about themselves

- WhatsApp

WhatsApp Messenger is a cross-platform instant messaging application

Appendix B: Analysis Models

NOT APPLICABLE

Appendix C: Future Aspects

- *As modernization comes into play, and market competitors will definitely spring up, we constantly need to improve our application in the technology race.*

- *We will utilize the full use of Artificial intelligence to incorporate smart assistance i.e, voice control, smart suggestions, control assistance and much will be used, to keep up with future trends.*

- *We will constantly upgrade/improve the application and include new and amazing features such as direct buying features for items from the comfort of application etc.*

- *In the future,the application will not remain specific to grocery but will include much more; the application will be able to differentiate between list and output the kind of store we need to go for our items instead of a supermarket i.e, hardware store, stationary store, dairy shop etc*