Software Requirements Specification

for

Grocery List Maker

**Version 1.0 approved**

**Prepared by Group No. 2**

**Section No. 5216**

**9/10/2021**

**Table of Contents**

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

4. System Features 4

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

6. Other Requirements 5

Appendix A: Glossary 5

Appendix B: Analysis Models 5

Appendix C: To Be Determined List 6

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

Grocery shopping can feel way more stressful than it is. After going through long lines, screaming children, traffic and finally reaching home with the groceries only to realize that you have yet again forgotten one thing that you were supposed to get can be very frustrating. In the worst case, the next day of the trip to the market you discover the eggs are over!

Though more major brands are turning to online grocery deliveries, there is still a significant number of people opting for the old-fashioned way of the trip to the market. So, to help you to round up all your groceries on your next visit to the market and keep track of ending groceries, a Grocery List Marker is made.

## 1.2 Document Conventions

The document uses bullet points to draw attention to important information within the srs so that any reader can identify the key issues and facts quickly and easily. Similarly, certain keywords are made strong or bold to highlight the text and to also capture the readers' attention. It is used to draw extra attention to important concepts or to indicate actions. The bold tag is used to strongly emphasize points.

## 1.3 Intended Audience and Reading Suggestions

Developers, Project Managers, and Testers reading this document are advised to go through Section 2 as it talks about User Classes and Characteristics, Operating Environment, Design and Implementation Constraints, Section 3 as it talks about User Interfaces, Hardware Interfaces, Software Interfaces, and Communications Interfaces, Section 4 which describes various system features, Section 5 talks about Non-Functional requirements and Section 6 is about the requirements that do not fall under about categories.

The Users can go through Sections 2.1 and 2.2 first which talk about the product perspectives and functions. If they require a detailed version of the features of the product, then they are advised to go through Section 4.

## 1.4 Product Scope

Grocery List Marker is a multifunctional list marker that first learns the user’s grocery patterns and makes the list of groceries. It keeps a track of groceries purchased and spent but checking the frequency of items. Having a list of to-buys lets you prioritize your spending on what you need at the moment. You’re also likely to stick to the aisles and sections where your items are located instead of exploring other areas where plenty of “great products” are waiting to be discovered.

The application reminds the users of the finishing products at their house. Furthermore, it tells the best route of purchasing groceries based on the criteria set by the users. The best route can be determined through shops in users’ vicinity or their usual preferred shops or shops offering them more reasonable prices. Through this, the users would be able to save time or money, or both.

It will keep you focused. The items in the store(s) are organized according to their types (e.g. skincare, baby products, kitchenware, etc.), but the wide product choices can distract you from going directly to the item you’re supposed to buy. When you have a list of items to buy, you simply need to locate them on their designated shelves instead of wandering around the store aisles.

It will also reduce waste. When you buy items that you don’t have an immediate need for, there's a greater chance that they’ll only end up in the trash. This usually happens with fresh food. Since they don’t last long, you’ll have to consume them for a limited period. Once they rot or spoil, you’d need to discard them. Through Grocery List Marker, you can avoid wasting food because you’ll have a realistic idea of what and how many food items to buy.

## 1.5 References

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

<https://online.visual-paradigm.com/diagrams/features/dfd-maker/>

Used Google Docs and Microsoft Word.

# Overall Description

## Product Perspective

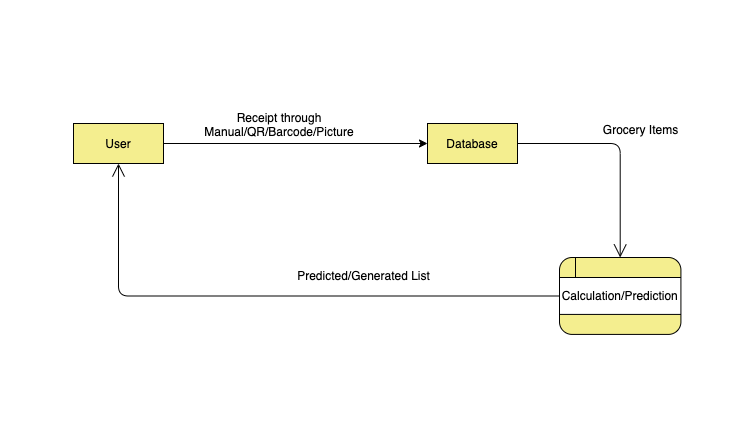
Family members, school staff, office janitorial staff, etc. have to purchase the same set of items every few weeks and months. Remembering and tracking when these items will run out and more will have to be purchased is a tiresome task that is otherwise either done on pen and paper, mentally or not at all.

To make this task easier, users enter their purchase details (including price and location) in an online database that can be accessed from anywhere. The system learns from the price data to make recommendations based on item cost. The system learns from the location data to make recommendations based on the distance from the user. This functionality is enabled by the use of the Google Maps API.

## Product Functions

Major Functions:

* Inputting purchase data through a QR code, OCR, or manual entry.
* Storing users’ purchase history online.
* Offering signup/login functionality to allow access to user data from anywhere.
* Editing or modifying purchase records.
* Learning from user purchase history
* Predicting imminent purchases based on either cost or distance.
* Sending mobile notifications for coupled predicted purchase items before the predicted date and time.
* Choosing the frequency at which the purchase notification is to be sent (i.e. weekly or monthly)
* Identifying and displaying the best purchase location based on the user’s set criteria (price or distance)



## User Classes and Characteristics

User classes:

* General public
* Administrator

The user should:

* Know English
* Have an existing functional Internet connection.
* Have an existing email address.
* Be able to operate a phone (this includes granting Location permissions to the app when asked)
* Be familiar with QR code operation.
* Be able to identify and enter appropriate street addresses.
* Be familiar with a signup/login system.

## Operating Environment

The application will run on mobile devices. The operating systems will be iOS (versions 13.0 and up) and Android (versions 6.0 and up). This application will require a GPS chip for acquiring the user’s location. The application will utilize the Google Maps API.

## Design and Implementation Constraints

* In case a large number of records have to be retrieved, the internet speed and access will affect the reliability and performance of the app. The user’s memory and processor speed will determine the speed at which learning takes place.
* The app depends on the Google Maps API which is a paid service. If the developer cannot afford to pay their fees in the future, the location functionality will not then work.
* The app expects a standardized input of receipt data from the QR code and OCR, otherwise, receipt parsing through that method will not work.
* GPS is required for detecting the user’s location. If this is not working accurately on the user’s device, they may select an incorrect location.
* If the location is not automatically ascertained but is entered manually by the user, the user may accidentally select the wrong location (due to a misspelling or similar sounding names).

## User Documentation

Specific screens that will run only on the first startup will show the user how to use the application.

There will be a dedicated and detailed “Help & FAQs” section of the app accessible from a menu.

A video will be available on YouTube that will showcase the app’s functionalities.

## Assumptions and Dependencies

Assumptions:

* Data stored in the QR Code by the user is of a standardized format that can be successfully recognized by the application.
* The user is entering the correct information.

Dependencies:

* The app is dependent on the Google Maps API. If in the future, this API functionality ceases to be offered or is altered in a breaking change, the location functionality of the app would be affected.
* The database which stores user data should be readily accessible over the Internet to the application.
* The code should be error-free.
* The end-user should understand how to use the product.

# External Interface Requirements

## 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.>

## 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.>

## 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.>

## 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.>

# System Features

**4.1 Account Management**

**4.1.1 Use Case 1: Sign-Up**

**4.1.1.1 Description and Priority**

The user can sign-up from the mobile application using his/her credentials. It is a **high priority** feature because once an account is created all the data stored in the database to predict the list is referred to that account which can be accessed using any device using the account. After signing up, the user will be able to make the grocery list and access the application features.

**4.1.1.2 Stimulus/Response Sequences**

The use case will start when the user presses the sign-up button.

They will be led to a screen where they will be given two options:-

* Either they sign-up using their Facebook or Google email IDs. The app will automatically pick up the name of the user from the selected option between these two.
* Or they will create a password and provide an email address.
* The user will be led to a screen where they will press “Verify email address” which will send the users an authentication email to the email addresses they used to make an account on this application.
* On that email, they will press the “Verified” button which will authenticate their email address.
* The user will then be asked to enter their full names(if they select the second option) and home address and password for this newly created account(twice, to verify it and avoid the mistake in the password if entered wrong).
* The password entered by the user will have to meet certain requirements.
* The password must contain at least one uppercase character, at least one special character, and some lowercase characters.
* The length of the password must be at least 8 characters.
* If these requirements for the password are not met, a red asterisk symbol will appear above the password bar which upon hovering onto will tell the user that the requirements have not been met and to enter the password again.
* The password has to be entered twice to confirm whether they match with each other and so that the user does not make a mistake.

**4.1.1.3 Functional Requirements**

**Req-1:** Sign up using Google(only the Gmail id is required). The users will provide their home address and password.

**Req-2:** Sign up using Facebook(only the Facebook email id is required). The users will provide their home address and password.

**Req-3:** Sign up using Email Id and Password. The users will provide their full names, home addresses, and password.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case name** | **Signup** | | |
| Related Requirements | **Req-1**  **Req-2**  **Req-3** | | |
| Goal in context | The system creates an account | | |
| Pre-conditions | The phone must be connected to the Internet | | |
| Successful End conditions | The user entered successful information and is returned to the homepage as logged in user | | |
| Failed end conditions | The user is unable to Signup for one or more reasons and is given an error message | | |
| Primary Actors | New User | | |
| Secondary Actors |  | | |
| Trigger | The user taps the Signup button on the sign-in page | | |
| Included cases | none | | |
| Main Flow | Steps | Actions |
|  | 1 | Application starts |
|  | 2 | The user enters credentials (Email Id and password twice to avoid a mistake) or signs up by tapping Google or Facebook icon |
|  | 3 | The system verifies information and sends an email to the user to verify the account. |
|  | 4 | The user receives an email for verification. |
|  | 5 | After verification, the account is created |
|  | 6 | The signup screen disappears and the user can access application features |
| Extensions | Steps | Branched Actions |
|  | 2.1 | The user does not enter all the information required. |
|  | 3.1 | The user already exists |
|  | 4.1 | The user enters the wrong email address |

**4.1.2 Use Case 2: Sign-In**

**4.1.2.1 Description and Priority**

The user can sign in from the mobile application using his/her credentials. It is a **high priority** feature because the user will get the predicted list based on data stored by him in the database which will be fetched once he/she logins correctly using his/her credentials into the application. After signing in, the user will be able to access the grocery list and application features.

**4.1.2.2 Stimulus/Response Sequences**

The use case will start when the user will ask to sign in to the application.

* This page will ask the user to sign in using Email and Password and click sign-in.
* If the email and password match then they will be signed in to their accounts otherwise an error message-invalid email/password will be displayed.
* If the user chooses to sign in either through Facebook or Google then they will have to click on the appropriate icon.
* If they’re already signed in to the device with their respective accounts then they will be signed in through the application otherwise they will be asked to enter their password for the account.
* If the password matches then they will be signed in to their accounts otherwise an error message-invalid password will be displayed.
* The Sign-In screen will allow users to enter their email and password.
* As the user enters the email and password the app will access the database and match the password corresponding with email-ID and state successful/unsuccessful Sign-In or will click on either Google or Facebook icon that will give them access to the application.
* If they type their email and password and click the Sign-In button then their credentials will be verified if they’re correct or not.
* If they enter the correct credentials the application will allow the user to access their account otherwise the application will display an error message-invalid email/password.
* If the user doesn't already have an account, they can create one by clicking the signup button, which will lead them to the signup page as explained above in 4.1.1.

**4.1.2.3 Functional Requirements**

**Req-4:** Sign In using Google(only the Gmail id is required)

**Req-5:** Sign in using Facebook(only the Facebook email id is required)

**Req-6:** Sign In using Email Id and Password

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case name** | **Sign In** | | |
| Related Requirements | **Req-4**  **Req-5**  **Req-6** | | |
| Goal in context | To log in application and access application features | | |
| Pre-conditions | * The phone must be connected to the Internet * The user must have already signed up | | |
| Successful End conditions | The user is authenticated and enters the main page as logged in user | | |
| Failed end conditions | * The user account is not present in the database * Cannot sign in due to internet connection failure | | |
| Primary Actors | Registered User | | |
| Secondary Actors |  | | |
| Trigger | The user taps the Sign-In button on the Sign-In page | | |
| Included cases | Verify Credentials | | |
| Main Flow | Steps | Actions |
|  | 1 | Application starts |
|  | 2 | The user enters credentials (Email id and password) or logs in by tapping Google or Facebook icon |
|  | Include :: Verify Credentials | Credentials are verified |
|  | 3 | The Sign-in screen disappears and the user can access application features |
| Extensions | Step | Branching Action |
|  | 2.1 | Invalid Email- error message is displayed |
|  | 2.2 | Invalid password- error message is displayed |
|  | 3.1 | The database does not verify the details or user does not already exist |

**4.1.3 Use Case 3: Verifying Credentials**

**4.1.3.1 Description and Priority**

The User ID and password entered by the user have to be first checked from the accounts stored in the database. If the User ID and Password match with that of the account in the database, then it will be verified that the user has an account. This will be a **high priority** feature because without signing in the user account can not be accessed, thus their stored grocery lists too.

**4.1.3.2 Stimulus/Response Sequences**

* The user will go to the sign-in screen of the application.
* They will have to enter the email used to create the account and the password they made to log into the application.
* There will be two boxes, a box with the title “Enter email” and a box with the title “Enter password”.
* When the user enters the required data into their respective boxes the application will check with the database whether the email and password match that which was initially stored when the user had created the account.
* If the data matches then the application will take the user to the home page of the application.
* If the data does not match then a red asterisk will appear above the “Enter email” title stating that “The email or password you provided is wrong, please enter the correct credentials.”
* If the user does not remember the password they can click on the “Forgot Password” button below the “Enter Password” box.

**4.1.3.3 Functional Requirements**

**Req-7:** Verify email and password from database

|  |  |  |
| --- | --- | --- |
| **Use case name** | **Verify Credentials** | |
| Related Requirements | **Req-7** | |
| Goal in context | To verify credentials and get data from the database | |
| Pre-conditions | The user must type Email and password. | |
| Successful End conditions | The user is successfully able to sign-in | |
| Failed end conditions | An error message is displayed- invalid email/password. | |
| Primary Actors | Database Administrator | |
| Secondary Actors | None | |
| Trigger | Users will tap the “Sign-In” button | |
| Included cases | none | |
| Main Flow | Steps | Actions |
|  | 1 | The user will enter his/her email and password. |
|  | 2 | The user taps the Sign-In button |
|  | 3 | The email and password are verified by checking if they match in the database. |
| Extensions | Steps | Branched Actions |
|  | 3.1 | The details are not verified. |

**4.1.4 Use Case 4: Forgot Password**

**4.1.4.1 Description and Priority**

This feature will allow the user to reset their password if they have forgotten the password they kept earlier for their account or they simply want to change their password for security concerns. This feature has a **high priority** as the user CANNOT access their account without the password.

**4.1.4.2 Stimulus/Response Sequences**

* If the user forgets their password, they will press the “Forgot Password” button which will send an email to their provided email address which will contain a code of six characters.
* The user will enter this code in a box under the “Provide Code” line correctly.
* If the code entered is wrong then the application will show a “Resend Code” button which the user will press and then the application sends another code to the email address and the user will have to correctly enter the code.
* When the code entered matches with the actual code the user will be led to a screen where they will have to enter the password for the application twice(as mentioned above in **4.1.1** and following the rules about the password mentioned in it) after which the application will take the user back to the sign-in screen to enter the necessary details to log in to their account.
* The user needs to remember their email sign-in credentials for this feature and should be able to open the email sent to their email address so they can view it to access the code required to verify the user and then they will enter the code they got from the email in the box under the “Provide Code” line correctly.
* The user has to make sure that they enter the code correctly or else they will not be able to reset the password.
* When the user does this step correctly they will be led to a screen where they will have to enter the password for the application twice which meets the following conditions: the password must contain at least one uppercase character, at least one special character, and some lowercase characters and must have at least 8 characters, after which the application will take the user back to the log-in screen to enter the email address and the newly created password to login to their account.

**4.1.4.3 Functional Requirements**

**Req-8**: Reset the password for the account.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case name** | **Forgot Password** | | |
| Related Requirements | **Req-8** | | |
| Goal in context | Account password is reset in the database | | |
| Pre-conditions | * The phone must be connected to the Internet * Must remember Email credentials (so that the code sent on email can be accessed) | | |
| Successful End conditions | The password is reset, and the user is returned to the Sign-In page | | |
| Failed end conditions | The password does not reset due to an incorrectly entered code or the user forgets their Email credentials | | |
| Primary Actors | Registered User | | |
| Secondary Actors | None | | |
| Trigger | The user taps the forgot password button on the Sign-In page | | |
| Included cases | none | | |
| Main Flow | Steps | Actions |
|  | 1 | Application starts |
|  | 2 | The user clicks the forgot password button where the user enters the email address |
|  | 3 | Then a six-digit code is sent to the user’s email ID |
|  | 4 | Once the correct code is entered user enters the new password twice according to password integrity rules |
|  | 5 | User clicks on confirm change password then he is led to the Sign-In page where he/she can enter email and new password to access the account |
| Extensions | Steps | Branched Actions |
|  | 4.1 | The user enters the incorrect code. |
|  | 4.2 | The user does not enter the code within 2 minutes due to which code is resent or the user can request for new code using resend code button |

**4.1.5 Use Case 5: Edit Profile**

**4.1.5.1 Description and Priority**

The user might want to edit some personal information in the app like their address or their account's password, so they will be able to do so. This feature has a **high priority** as if the user needs to change their home address the application will have to implement learning again for the grocery lists.

**4.1.5.2 Stimulus/Response Sequences**

* If the user wants to edit their account information they will first click on the “View Profile” button.
* This will lead them to a screen showing them their Full name, the last three characters of their password under the “Password” heading with some leading asterisks, and their Home Address.
* Below all this information there will be an “Edit Profile” button which upon clicking will lead the user to a screen asking them to provide the email address and password which they are using for the application.
* If the user enters all correct information they will be able to edit their home address.
* The user needs to remember their email address and password for this feature so the application can verify if it is indeed the user who wants to edit their profile information.
* The user needs to make sure that the address they provide should be that which is appearing on Google Maps or else the application will not be able to locate and search the shops around the given address correctly.
* There will be a "Reset Password" button which upon being clicked will take the user to a screen that will ask them to provide their previous password.
* After they enter their current password, they will be prompted to enter their new password which meets the following conditions: the password must contain at least one uppercase character, at least one special character, and some lowercase characters and must have at least 8 characters. They will be asked to enter it twice to make sure that there are no errors in the new password.

**4.1.5.3 Functional Requirements**

**Req-9**: The user will be able to change their home address that is currently stored in the database.

**Req-10**: The user will be able to reset their password.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case name** | **Edit Profile** | | |
| Related Requirements | **Req-9**  **Req-10** | | |
| Goal in context | To be able to edit profile. | | |
| Pre-conditions | * The phone must be connected to the Internet * The user must have logged in | | |
| Successful End conditions | The user successfully saves the edited details in the profile. | | |
| Failed end conditions | * The user did not enter a valid home address. * Cannot edit profile due to internet connection failure | | |
| Primary Actors | Registered User | | |
| Secondary Actors | None | | |
| Trigger | The user taps the Edit Profile button on the view profile page | | |
| Included cases | none | | |
| Main Flow | Steps | Actions |
|  | 1 | The user taps the “View Profile” button from where they can tap on the “Edit Profile” button |
|  | 2 | The user provides Email Address and Password |
|  | 3 | After this, the user will be able to edit the home address and password |
| Extensions | Steps | Branched Actions |
|  | 2.1 | The user enters incorrect information- an error message is displayed |
|  | 2.3 | User-provided the same password already stored in the database. |
|  | 2.4 | User-provided the same home address already stored in the database |

**4.1.6 Use Case 6: Sign-Out**

**4.1.6.1 Description and Priority**

This feature will allow the user to sign out of the application. The priority of this feature will be **high** because for example if the user sells their mobile phone without deleting the application, the other person who receives the user’s mobile phone will be able to view the home address of the old user. This will be a huge threat to the old user’s security.

**4.1.6.2 Stimulus/Response Sequences**

* The user will press the “Sign Out” button on the top left corner of the home page.
* The application will save all the current data in the database.
* The application will then take the user back to the sign-up/ sign-in screen.

**4.1.6.3 Functional Requirements**

**Req-11:** The application shall provide a mechanism for signed-in users to log out of the application.

**Req-12:** The current data in the application will be stored in the database, and the user will be signed out.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case name** | **Sign Out** | | |
| Related Requirements | **Req-11**  **Req-12** | | |
| Goal in context | To terminate the session in the application. | | |
| Pre-conditions | * The phone must be connected to the Internet * The user must have logged in. | | |
| Successful End conditions | The session is terminated. | | |
| Failed end conditions | Current application data is not stored in the database if the user signs out and does not have an internet connection | | |
| Primary Actors | Registered User | | |
| Secondary Actors | None | | |
| Trigger | The user taps the Sign-Out button | | |
| Included cases | none | | |
| Main Flow | Steps | Actions |
|  | 1 | The user taps the Sign-Out button |
|  | 2 | The application saves and terminates the session and logs out. |
| Extensions | Steps | Branched Actions |
|  | 2.1 | The application failed to save data due to Internet Connection failure. |

## 4.2 Storing Data into Database

## 4.2.1 Use Case 7: Adding Items through QR code

**4.2.1.1 Description and Priority**

The user will be able to add the list of items through the QR code which the user has bought which was included in the grocery list. This is a **high priority** feature because based on this stored item in the database our application will predict grocery list items, their respective shops from where they have to be bought, and when to buy them.

**4.2.1.2 Stimulus/Response Sequences**

* The user will first click on the “Add item to History” button in the app.
* This will give the user four options:-

(i) Scan Barcode

(ii) Enter Manually

(iii) Scan QR-Code

(iv) Take/Upload Photo

* If the user selects the QR code, the camera will open, and then the user will scan the QR Code.
* Then data from the QR code is scanned and shown to the user
* Once they click “Confirm” then the user will be asked to enter shop details manually.
* The user then clicks “ADD” this is added to the database and the user is led back to the “Add item to History” page.

**4.2.1.3 Functional Requirements**

**Req-13:** Scanning QR code of list (Need to enter shop address)

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case name** | **Adding through QR code** | | |
| Related Requirements | **Req-13** | | |
| Goal in context | The user adds grocery items to the database using a QR code | | |
| Pre-conditions | * The user must be logged in. * The user must have allowed the application to access the camera | | |
| Successful End conditions | Item or Items list is added to the database. | | |
| Failed end conditions | * An item was incorrectly detected by the application. * The user did not allow the application to access the camera | | |
| Primary Actors | Registered User | | |
| Secondary Actors | none | | |
| Trigger | The user clicks the QR code option | | |
| Included cases | none | | |
| Main Flow | Steps | Actions |
|  | 1 | The user clicks the “Add Items to History” button where there will be 3 options from which the user will choose QR code |
|  | 2 | The camera will open and the user will scan the QR code and confirm |
|  | 3 | The user enters the shop name manually and clicks add |
|  | 4 | The item or list of items is added. |
| Extensions | Steps | Branched Actions |
|  | 2.1 | The user did not give access to the camera to be used by the application |
|  | 2.2 | Unable to scan QR code due to blur or crooked QR code |

## 4.2.2 Use Case 8: Adding Items through Barcode

**4.2.2.1 Description and Priority**

The user will be able to add the list of items through Barcode which the user has bought which were included in the grocery list. This is a **high priority** feature because based on this stored item in the database our application will predict grocery list items, their respective shops from where they have to be bought, and when to buy them.

**4.2.2.2 Stimulus/Response Sequences**

* The user will first click on the “Add item to History” button in the app.
* This will give the user four options:-

(i) Scan Barcode

(ii) Enter Manually

(iii) Scan QR-Code

(iv) Take/Upload Photo

* The camera will open when the user selects the Barcode, and the user will scan the Barcode.
* The barcode data is then read and displayed to the user; after the user clicks "Confirm," the user is requested to manually enter store details; after the user hits "ADD," the item is added to the database, and the user is returned to the "Add item to History" page.

**4.2.2.3 Functional Requirements**

**Req-14:** Scanning Barcode of the list (Need to enter shop address)

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case name** | **Adding through Barcode** | | |
| Related Requirements | **Req-14** | | |
| Goal in context | The user add grocery items using a barcode | | |
| Pre-conditions | * The user must be logged in. * The user must have allowed the application to access the camera | | |
| Successful End conditions | Item or Items list is added to the database. | | |
| Failed end conditions | * An item was incorrectly detected by the application. * The user did not allow the application to access the camera | | |
| Primary Actors | Registered User | | |
| Secondary Actors | none | | |
| Trigger | The user clicks the Barcode option | | |
| Included cases | none | | |
| Main Flow | Steps | Actions |
|  | 1 | The user clicks the “Add Items to History” button where there will be 3 options from which the user will choose barcode |
|  | 2 | The camera will open and the user will scan the barcode and confirm |
|  | 3 | The user enters the shop name manually and clicks add |
|  | 4 | The item or list of items is added. |
| Extensions | Steps | Branched Actions |
|  | 2.1 | The user did not give access to the camera to be used by the application |
|  | 2.2 | Unable to scan bar code due to blur or crooked barcode or poor quality print. |

## 4.2.3 Use Case 9: Adding Items Manually

**4.2.3.1 Description and Priority**

The user will be able to add items manually which the user has bought which were included in the grocery list. This is a **high priority** feature because based on this stored item in the database our application will predict grocery list items, their respective shops from where they have to be bought, and when to buy them.

**4.2.3.2 Stimulus/Response Sequences**

* The user will first click on the “Add item to History” button in the app.
* This will give the user four options:-

(i) Scan Barcode

(ii) Enter Manually

(iii) Scan QR-Code

(iv) Take/Upload Photo

* When the user selects “Enter Manually”, A form opens up then the user enters item details and its shop details as well.
* Once the user clicks the ”ADD” button, these details are added to the database and the user is returned to the "Add item to History" page.

**4.2.3.3 Functional Requirements**

**Req-15:** Entering items to list manually and adding them to the database.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case name** | **Adding Manually** | | |
| Related Requirements | **Req-15** | | |
| Goal in context | The user adds grocery items to the database by manually entering the data. | | |
| Pre-conditions | The user must be logged in. | | |
| Successful End conditions | Item or list of items is added to the database. | | |
| Failed end conditions | An item was incorrectly detected by the application. | | |
| Primary Actors | Registered User | | |
| Secondary Actors | none | | |
| Trigger | The user clicks the Enter Manually option | | |
| Included cases | none | | |
| Main Flow | Steps | Actions |
|  | 1 | The user clicks the “Add Items to History” button where there will be 3 options from which the user will choose Enter Manually |
|  | 2 | A form will open and the user will enter the details. |
|  | 3 | The item or list of items is added. |
| Extensions | Steps | Branched Actions |
|  | 2.1 | The user entered incorrect details |
|  | 2.2 | The prediction algorithm will not work if the item details are stored incorrectly in the database |

## 4.2.4 Use Case 10: Adding Items List through Photo

**4.2.4.1 Description and Priority**

The user will be able to add the list of items by taking the photo of the receipt which the user has bought which was included in the grocery list. This is a **high priority** feature because based on this stored item in the database our application will predict grocery list items, their respective shops from where they have to be bought, and when to buy them.

**4.2.4.2 Stimulus/Response Sequences**

* The user will first click on the “Add item to History” button in the app.
* This will give the user four options:-

(i) Scan Barcode

(ii) Enter Manually

(iii) Scan QR-Code

(iv) Take/Upload Photo

* The camera opens when the user selects the “Take Photo” option.
* A picture of the receipt will be uploaded on which OCR will be used to detect the items and their quantities. the user is requested to manually enter store details; after the user hits "ADD," the item is added to the database, and the user is returned to the "Add item to History" page.

**4.2.4.3 Functional Requirements**

**Req-16:** Taking a photo of the receipt and scanning the items list and storing data into the database.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case name** | **Adding through Photo** | | |
| Related Requirements | **Req-16** | | |
| Goal in context | The user adds grocery items to the database by taking a photo. | | |
| Pre-conditions | * The user must be logged in. * The user must have allowed the application to access the camera/gallery | | |
| Successful End conditions | Item or list of items is added to the database. | | |
| Failed end conditions | The user did not allow the application to access the camera/gallery | | |
| Primary Actors | Registered User | | |
| Secondary Actors | none | | |
| Trigger | The user clicks the Take/Upload Photo option | | |
| Included cases | none | | |
| Main Flow | Steps | Actions |
|  | 1 | The user clicks the “Add Items to History” button where there will be 3 options from which the user will choose Take/Upload Photo |
|  | 2 | The camera will open and the user will take a photo |
|  | 3 | A picture of the receipt will be uploaded on which OCR will be used to detect the items and their quantities. |
|  | 4 | Users will enter shop names manually |
|  | 5 | User clicks add button and Item is added to the database |
| Extensions | Steps | Branched Actions |
|  | 2.1 | The user did not give access to the camera to be used by the application |
|  | 3.1 | Unable to detect items due to poor quality image |

## 4.3 Calculating Route Based On User’s Criteria

**4.3.1 Use Case 11: Calculating route based on Cost**

**4.3.1.1 Description and Priority**

The application will output and display a predicted list of groceries based on the grocery products that the user has supplied as well as the user's previous purchasing history based on the cheapest price. This is a **high priority** feature since the user will plan his or her shopping route based on this list, ensuring that no goods that are usually purchased are overlooked.

**4.3.1.2 Stimulus/Response Sequences**

* Based on prediction the items will be displayed according to cheapest cost price per item, for example, the topmost item in the list that is item 1 will be nearer to his current location and then item 2 shop will be nearer to item 1’s shop, etc.
* But the first factor here is cost price and the second is distance.
* The app can calculate the total cost of the items in the grocery list using a simple calculator and it will then be displayed.
* The shops where the users' desired items are the cheapest will be displayed. Keep in mind that these stores may be close or far apart.
* This feature will not guarantee time-saving.

**4.3.1.3 Functional Requirements**

**Req-17:** Total cost/price of all the items on the list will be displayed.

**Req-18**: Using the prediction algorithm, the predictions will be made based on cost.

**Req-19**: If the shop no longer exists or does not have the required item, it would not be shown in the route. (TBD)

|  |  |  |
| --- | --- | --- |
| **Use case name** | **Calculating Route Based on Cost** | |
| Related Requirements | **Req-17**  **Req-18**  **Req-19** | |
| Goal in context | To display Grocery List based on Cost | |
| Pre-conditions | The user must be logged in and should be connected to the internet. | |
| Successful End conditions | Users will be able to see Grocery List according to the cheapest price. | |
| Failed end conditions | If the application hasn’t been used twice for a similar grocery item then an empty list will be displayed | |
| Primary Actors | Registered User | |
| Secondary Actors | None | |
| Trigger | Users will tap the “Route Based On Cost” button | |
| Included cases | none | |
| Main Flow | Steps | Actions |
|  | 1 | Users will click on the grocery list button |
|  | 2 | The user has 3 options of the type of grocery to be displayed, user taps the button to display the list based on the cheapest cost. |
|  | 3. | The Grocery List will be displayed. |
| Extensions | Steps | Branched Actions |
|  | 3.1 | If two shops have the same price then the prediction algorithm may choose a shop that is further away from the user’s location. |
|  | 3.2 | An empty list will be displayed if the user hasn’t bought similar groceries at least twice. |

**4.3.2 Use Case 12: Calculating route based on Home Location**

**4.3.2.1 Description and Priority**

After learning from the grocery items that the user has inputted and the user's previous purchase history, the app will then output and calculate a grocery list based on distance from the user's home location. This is a **high priority** feature because based on this list the user will make his/her route to buy items and does not miss on any item which was bought regularly.

**4.3.2.2 Stimulus/Response Sequences**

* Based on prediction the items will be selected according to the nearest shops from the user’s home location where those items can be bought, for example, the topmost items that have a common shop will be nearer to his/her home’s location than other shops where those items can be cheaper but due to short in time, he will get the shop location where all items or majority can be bought.
* Then if an item is not in that shop the shop nearest to the current shop will be assigned for that item in the list.

**4.3.2.3 Functional Requirements**

**Req-20:**  Items on the list will be displayed according to the nearest from the user's home location.

|  |  |  |
| --- | --- | --- |
| **Use case name** | **Calculating Route Based on Home Distance** | |
| Related Requirements | **Req-17**  [**Req-19**](#_4.3__Calculating)  **Req-20** | |
| Goal in context | To display Grocery List based on Distance | |
| Pre-conditions | The user must be logged in and should be connected to the internet. | |
| Successful End conditions | Users will be able to see Grocery List according to the nearest shop from the user’s Home location. | |
| Failed end conditions | * If the application hasn’t been used twice for a similar grocery item then an empty list will be displayed | |
| Primary Actors | Registered User | |
| Secondary Actors | None | |
| Trigger | Users will tap the “Route Based On Home Location” button | |
| Included cases | none | |
| Main Flow | Steps | Actions |
|  | 1 | Users will click on the grocery list button |
|  | 2 | The user has 3 options of the type of grocery to be displayed, user tap the button to display the list based on the closest distance from home |
|  | 3. | The Grocery List will be displayed. |
| Extensions | Steps | Branched Actions |
|  | 3.1 | An empty list will be displayed if the user hasn’t bought similar groceries at least twice. |

**4.3.3 Use Case 13: Calculating route based on Current Location**

**4.3.3.1 Description and Priority**

The application will output and generate a grocery list depending on the distance from the user's current location after learning from the grocery items the user has entered and the user's past purchase history. This is a **high priority** feature since the user will plan his or her shopping route based on this list, ensuring that no goods that are usually purchased are forgotten.

**4.3.3.2 Stimulus/Response Sequences**

* The items will be selected based on the user's current location and the nearest shops where those items can be purchased.
* For example, the topmost items that have a common shop will be closer to his/her current location than other shops where those items can be purchased for a lower price, but due to time constraints, he will get the shop location where all items or the majority can be purchased.
* If an item isn't in that shop, the item in the list will be assigned to the store closest to the present shop.

**4.3.3.3 Functional Requirements**

**Req-21**: Using the prediction algorithm, the predictions will be made based on the shortest distance from the user’s current location.

|  |  |  |
| --- | --- | --- |
| **Use case name** | **Calculating Route Based on Current Location** | |
| Related Requirements | **Req-17**  [**Req-19**](#_4.3__Calculating)  **Req-21** | |
| Goal in context | To display Grocery List based on Location | |
| Pre-conditions | The user must be logged in and should be connected to the internet. | |
| Successful End conditions | Users will be able to see Grocery List according to the nearest shop from the user’s current location. | |
| Failed end conditions | Internet Connection Failure. | |
| Primary Actors | Registered User | |
| Secondary Actors | None | |
| Trigger | Users will tap the “Route Based On Current Location” button | |
| Included cases | none | |
| Main Flow | Steps | Actions |
|  | 1 | Users will click on the grocery list button |
|  | 2 | The user has 3 options of the type of grocery to be displayed, user tap the button to display the list based on the closest distance from the current location |
|  | 3. | The Grocery List will be displayed according to the nearest shop from the user’s current location. |
| Extensions | Steps | Branched Actions |
|  | 3.1 | An empty list will be displayed if the user hasn’t bought similar groceries at least twice. |
|  | 3.2 | Due to an internet connection failure, the application was not able to find the nearest shop. |

**4.4 Additional Features**

## 4.4.1 Use Case 14: Reminders/Notifications

**4.4.1.1 Description and Priority**

The application will be able to learn over time if the user is running out of grocery items so if the user allows, the application will be able to send a reminder through a push notification when the user is running out of grocery items. This is a **high-priority** feature. The user will be able to choose when they want to receive the notification. The application will give options to the user if they want to receive the notification every week or on monthly basis and at what time of the day.

**4.4.1.2 Stimulus/Response Sequences**

* The application will be using a prediction algorithm and data from the database to predict when the user will be running out of grocery items.
* If the user has initially selected to receive notifications every week then the prediction algorithm will work accordingly and send information via push notification.
* If the user has initially selected to receive notifications every month then the prediction algorithm will predict based on a month and send a push notification if the user is running out of grocery items.

**4.4.1.3 Functional Requirements**

**Req-22:** Based on days in a week, the user will be notified on the day that he decided

**Req-23:** Based every month, the user will be notified on the date of that month on which decided

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case name** | **Reminder** | | |
| Related Requirements | **Req-22**  **Req-23** | | |
| Goal in context | To remind user if they’re running out of grocery | | |
| Pre-conditions | * The user must be logged in * The user previously set a reminder to notify if the user is about to run out of a grocery item * The current time is when the grocery reminder is set to go off | | |
| Successful End conditions | If the user clicks the notification, the grocery list is displayed. | | |
| Failed end conditions | If the user clicks “Remind me later” and sets a time, the system closes the reminder and waits for the specified time. | | |
| Primary Actors | Registered User | | |
| Secondary Actors | none | | |
| Trigger | Time to send a reminder | | |
| Included cases | none | | |
| Main Flow | Steps | Actions |
|  | 1 | The push notification is displayed. It contains a message to check the grocery list items. |
|  | 2 | The user clicks the notification. |
|  | 3 | The grocery list is displayed. |
| Extensions | Steps | Branched Actions |
|  | 1.2 | The user didn’t set a time for a reminder. |

## 4.4.2 Use Case 15:List Sharing

**4.4.2.1 Description and Priority**

The users can share the grocery list that they have made. This feature has **low priority** as the application will simply copy the specific list to the clipboard feature of the keyboard which is a minute function.

**4.4.2.2 Stimulus/Response Sequences**

* The user will go to the “Current List” option in the application.
* There will be a “Share List” button below the list which the user will be able to see after scrolling down to the bottom of the list.
* When the user presses this button a popup menu will open, showing various social media apps like WhatsApp, Messenger, etc.
* The user will then click the specific social media app on which they wish to share their list.
* Upon clicking that app they will select with whom they want to share by clicking on the name of that specific person.
* Hence, the list will be shared.

**4.4.2.3 Functional Requirements**

**Req-24:** Users can share the list on different social media platforms

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case name** | **Grocery List sharing** | | |
| Related Requirements | **Req-24** | | |
| Goal in context | To be able to share the grocery list. | | |
| Pre-conditions | The user must be logged in and should be connected to the internet. | | |
| Successful End conditions | The user successfully shared the Grocery List. | | |
| Failed end conditions | The user failed to share the Grocery List and is returned to the screen where the List is displayed. | | |
| Primary Actors | Registered User | | |
| Secondary Actors | None | | |
| Trigger | User taps on the “Share” button | | |
| Included cases | none | | |
| Main Flow | Steps | Actions |
|  | 1 | The user after clicking the Share button will be displayed a list of social media platforms to choose from. |
|  | 2 | After tapping on the social media platform icon, the user will be able to share the list among others. |
| Extensions | Steps | Branched Actions |
|  | 1.1 | Due to Internet Connection failure, social media platforms weren’t displayed. |

## 4.4.3 Use Case 16: Customer Support

**4.4.3.1 Description and Priority**

This will serve as a platform for customers to direct their queries, concerns, and suggestions regarding the application. This is a **low priority** feature because it would not affect the main working for the application but serve as an external feature.

**4.4.3.2 Stimulus/Response Sequences**

* The “Customer Support” button will be at the bottom of the Home Page.
* When the user clicks on that button, three options will pop up:-

(i) “Demo Video” which upon pressing will demonstrate the user how to use the application.

(ii) “Frequently Asked Questions” which upon pressing will lead the user to a page where all FAQs are listed.

(iii) “Contact Us” which upon pressing will lead the user to a screen with the title of “Contact us through the email address provided below” under which an email address will be provided.

**4.4.3.3 Functional Requirements**

**Req-25:** The system shall provide FAQ for customer support.

**Req-26:** The system shall display the customer support email on the screen.

**Req-27:** The system will provide a Demo video.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case name** | **Customer Support** | | |
| Related Requirements | **Req-25**  **Req-26**  **Req-27** | | |
| Goal in context | To offer help to the user. | | |
| Pre-conditions | The user must be logged in. | | |
| Successful End conditions | The user will be able to get an answer for the problem that occurred. | | |
| Failed end conditions | None | | |
| Primary Actors | Registered User | | |
| Secondary Actors | None | | |
| Trigger | User taps on the “Customer Support” button on the home page | | |
| Included cases | none | | |
| Main Flow | Steps | Actions |
|  | 1 | The user after clicking the customer support icon has displayed a screen to choose an option. |
|  | 2 | The user can choose to either email for Customer Support Service, view Demo video, or view the FAQs. |
| Extensions | Steps | Branched Actions |
|  | 2.1 | Demo video could not be displayed due to internet connection |
|  | 2.2 | FAQS could not be displayed due to internet connection |

## 4.5 User-Made Changes

## 4.5.1 Use Case 17:Manage Grocery List

**4.5.1.1 Description and Priority**

The user will be able to edit the latest grocery list generated by the application using an algorithm. The feature has **high priority** and is kept because there can be instances when the user's grocery requirements will not be his usual ones as he/she might or might not be needing anymore or by the quantity will not be his/her usual purchasings’. This feature will allow the user to edit the previously generated grocery list.

**4.5.1.2 Stimulus/Response Sequence**

* The grocery list will be displayed to the user on the screen, if the item required is present in the list but the quantity has to be changed then the user can tap “+” to increase or “-” to decrease it.
* By clicking “Save” the new list will be saved.
* If the user does not want a particular item in the list, then they can select the item and tap “Remove” to remove it from the list. A new list will be generated without that item.
* If the user wants to add an item

**4.5.1.3 Functional Requirements**

**Req-28:** Delete item entry in the predicted grocery list.

**Req-29:** Edit item quantity in the predicted grocery list.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case name** | **Manage Grocery List** | | |
| Related Requirements | **Req-28**  **Req-29** | | |
| Goal in context | To Edit previously generated grocery list. | | |
| Pre-conditions | The user must be logged in and should be connected to the internet. | | |
| Successful End conditions | The grocery list is edited as per the user’s needs. | | |
| Failed end conditions | The item quantity will not be altered in the database if the mobile does not have an internet connection. | | |
| Primary Actors | Registered User | | |
| Secondary Actors | None | | |
| Trigger | User taps “Update” button in Grocery list page | | |
| Included cases | none | | |
| Main Flow | Steps | Actions |
|  | 1 | The user accesses the Grocery list page and clicks the “Edit List” button |
|  | 2 | The user either deletes an item or changes the item quantity. |
|  | 3 | The user taps the “Update” button and data is updated on the database and grocery list as well. |
| Extensions | Steps | Branched Actions |
|  | 2.1 | The user enters an invalid item quantity |
|  | 2.2 | A user mistakenly deletes an item that he/she did not intend |

**4.5.2 Use Case 18: Altar Records in Receipt History**

**4.5.2.1 Description and Priority**

The purpose of this feature is to allow the user to edit or delete an item from the list entered and to edit their account information. This feature has a **high priority** as the user might have bought more than the required amount of a certain item and might want to remove it from the list when they will be using the previous grocery list for the next month, or if they want to remove the item they click on the item in the list so they will be able to do so.

**4.5.2.2 Stimulus/Response Sequences**

* If the user wants to edit the quantity of an item or delete an item from the list they will select the “View Previous Lists” option in the app.
* This will lead them to a screen where all previous grocery lists are with the DATE as their titles.
* The user will click on the title to view the list. When the user clicks on the list they will be provided with two options:-

(i) “Edit Quantity of Item(s)” will allow them to edit the quantities of the items they might want more or less according to their needs.

(ii) “Delete Item(s) From List” will allow them to remove the item the user wants to be removed.

* After this, the user will press the “View new list” button below this which will lead them to a newly created list that will contain the edited list with all the changes made by the user.

**4.5.2.3 Functional Requirements**

**Req-30:** Delete item entry in receipt list in the history of receipts.

**Req-31:** Edit item quantity in receipt list in the history of receipts.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case name** | **Alter Receipt History** | | |
| Related Requirements | **Req-30**  **Req-31** | | |
| Goal in context | To edit receipt history in the database. | | |
| Pre-conditions | The user must be logged in and should be connected to the internet. | | |
| Successful End conditions | Data will be successfully updated in the database. | | |
| Failed end conditions | * Data was incorrectly altered. * Due to the internet connection, failure data was not updated. | | |
| Primary Actors | Registered User | | |
| Secondary Actors | None | | |
| Trigger | The user taps the “Update” button in Receipt History | | |
| Included cases | none | | |
| Main Flow | Steps | Actions |
|  | 1 | The user accesses the history receipt page and clicks the “Edit receipt” button |
|  | 2 | The user either deletes an item or changes the item quantity. |
|  | 3 | The user taps the “Update” button and data is updated on the database. |
| Extensions | Steps | Branched Actions |
|  | 2.1 | The user enters an invalid item quantity |
|  | 2.2 | A user mistakenly deletes an item that he/she did not intend |

Diagram

Description automatically generated

# Other Nonfunctional Requirements

## Performance Requirements

* Sign-up must require verification and verification must be done within a minute or the verification code will be expired.
* It should not take more than 3 seconds to load the initial screen.
* If logging into an account takes longer than 2 minutes, it must ask for the password again.
* If there is a notification and it is not acknowledged, the user must receive the notification again in 30 minutes.
* Cost and distance must be calculated within 2 minutes of the user asking about it.
* Each page must load within 2 seconds
* The app must cater to different mobile design screens approx. 5 inches to 12 inches.
* To have access to web utility (maps), the user must have an internet connection.
* For android users, the software version must be above 6.0.
* For IOS users the software must be above 13.
* The application must support 20,000+ users at a time without any trouble.

## Safety Requirements

User must receive an email if he/she tries to log in on another device for verification so that there is no misuse.

Designers must take account of the General Principles of Prevention when preparing designs.

If there is any sort of virus in the operating system which causes the failure of the application, the information of users must be backed up. This will make sure that the users never lose their data.

## Security Requirements

* All authentication information should be saved on authorize database for comparison and will require user permission to gain access
* Sign-up must require verification and verification must be done within a minute or the verification code will be expired
* All functions/features must-have assignable security levels.
* The system shall use secure sockets in all transactions that include any confidential customer information.
* The system shall not leave any cookies on the customer’s device containing the user’s password.
* The database should be secure from any malicious cyber-attacks.

## Software Quality Attributes

**FUNCTIONALITY: -**

* Accuracy: We want our application to be accurate and not miss any features that we have covered.
* Security: during the functionality of the Application, we must not lose any of the personal user’s information. We shall take all the security measures. Users must log in to have access to their account and login must be done using the user’s unique email and password that they set.
* Interoperability (the ability of computer systems or software to exchange and make use of information)
* Functionality Compliance (capability of the software application to adhere to standards, conventions, or regulations relating to functionality.)

**RELIABILITY: -**

* Fault Tolerance: if at any time this application faces some sort of malware or some problem, if there is some virus, so it can work with it and give out the warnings to the user. And can handle the situation without causing users much of an issue.
* Recoverability: if for supposing the software has crashed, so it should be able to recover, and the app can again be functional to the user.
* Reliability Compliance: this application must adhere to standards, conventions, or regulations relating to reliability.

**USABILITY: -**

* Understandability: the user or the operator must understand the instruction and structure of the application easily without any trouble.
* Learnability: if the application needs to learn any new information, it should be easy for it. And if there is any new feature for the user to understand, the application must cover all the aspects of learnability.
* Operability: the application must be easily operatable on the user's mobile.
* Attractiveness: the application must be nicely designed so it looks good to the user.
* Usability Compliance: this application must adhere to standards, conventions, or regulations relating to usability.

**EFFICIENCY: -**

* Time: every feature must function in an optimal time so that the user does not have to wait much for pages to load.
* Resource utilization: all the resources used must be someway utilized by the application or the user.
* Efficiency compliance: this application must adhere to standards, conventions, or regulations relating to Efficiency.

**MAINTAINABILITY: -**

* Stability: If the application is behaving the way it should then it is considered stable. App stability is commonly associated with crashes.
* Testability*:* Software application testability is the to which a software artifact supports testing in a given test context. If the testability of an application is high, finding the faults becomes a lot easier.
* Changeability: if there is a new software update, the application must adhere to its changes. Or if the changes are to be held on the application to provide the user with a better application so it must be able to act upon the changes.
* Maintainability Compliance: this application must adhere to standards, conventions, or regulations relating to Maintainability.

**PORTABILITY:-**

* Adaptability: the application should be adaptable to all IOS and Android devices. And, to all the software updates of a mobile phone.
* Coexistence: if there are other apps of such sort, it must be able to co-exist with them.
* Installability*:* Mobile application development is the process of creating software applications that run on a mobile device, and a typical mobile application utilizes a network connection to work with remote computing resources. Hence, the mobile development process involves creating installable software bundles implementing backend services such as data access with an API, and testing the application on target devices.
* Portability Compliance: this application must adhere to standards, conventions, or regulations relating to Portability.

## Business Rules

* All users need a valid email address or phone number.
* Each user can create only one account using a single email address.
* Users must provide all the necessary information for the app to function properly.
* Auto-calculation of user’s total.
* Confidentiality of user’s purchase.
* Confidentiality of user’s information.
* It is suggested that users must check before putting in any information or even before downloading the app.
* Any issues on the way to the shop will not be the developer's or applications' fault.
* Any issues with the product will also not be the developer's or applications' fault.

# Other Requirements

The items that are prohibited by the law in the individual country will not be added to the list. For instance, in Pakistan, the consumption of alcohol is banned.

**Appendix A: Glossary**

**Application program interface (API):** An interface, generally specified as a set of operations, that allows access to an application program’s functionality. This means that this functionality can be called on directly by other programs and not just accessed through the user interface.

**Constraints:** A constraint is anything that slows a system down or prevents it from achieving its goal. You could think of a constraint as a bottleneck in your processes that impedes your progress.

**Credential:** The [abilities](https://dictionary.cambridge.org/dictionary/english/ability) and [experience](https://dictionary.cambridge.org/dictionary/english/experience) that make someone [suitable](https://dictionary.cambridge.org/dictionary/english/suitable) for a [particular](https://dictionary.cambridge.org/dictionary/english/particular) [job](https://dictionary.cambridge.org/dictionary/english/job) or [activity](https://dictionary.cambridge.org/dictionary/english/activity), or [proof](https://dictionary.cambridge.org/dictionary/english/proof) of someone's [abilities](https://dictionary.cambridge.org/dictionary/english/ability) and [experience](https://dictionary.cambridge.org/dictionary/english/experience).

**Extension**: The extension use case consists of one or several behavior sequences (segments) that describe additional behavior that can incrementally augment the behavior of the base use case. Each segment can be inserted into the base use case at a different point, called an extension point.

**Functional requirements:** The basic system behavior. Essentially, they are what the system does or must not do, and can be thought of in terms of how the system responds to inputs. Functional requirements usually define if/then behaviors and include calculations, data input, and business processes.

**GPS:** The Global Positioning System (GPS) is a U.S.-owned utility that provides users with positioning, navigation, and timing (PNT) services.

**Hardware interface**: An architecture used to interconnect two devices together. It includes the design of the plug and socket, the type, number, and purpose of the wires, and the electrical signals that are passed across them.

**High priority requirements:** They are important because customers need the capability and urgent because they need it in the next release. Alternatively, there might be compelling business reasons to implement a requirement promptly, or contractual or compliance obligations might dictate early release.

**Low priority requirements:** They are neither important (customers can live without the capability if necessary) nor urgent (customers can wait, perhaps forever).

**OCR:** OCR stands for "Optical Character Recognition." It is a technology that recognizes text within a digital image.

**Operating Environment/ Integrated Application**: Operating environment or integrated applications environment is the [environment](https://en.wikipedia.org/wiki/Deployment_environment) in which users run [application software](https://en.wikipedia.org/wiki/Application_software).

**OR Codes:** While they may look simple, QR codes can store lots of data. But no matter how much they contain, when scanned, the QR code should allow the user to access information instantly.

**Primary actor:** It is of a use case is the stakeholder that calls on the system to deliver one of its services. It has a goal concerning the system – one that can be satisfied by its operation. The primary actor is often, but not always, the actor who triggers the use case.

**Product Scope:** The product scope is defined by the features and functions that characterize a product, service, or result. It is the result of the project or the solution to the problem.

**Product Perspective:** product perspective defines how the product contributes to fulfilling stakeholder needs and adjacent systems assumptions

**Secondary actor:** It is one from which the system requires assistance to complete the use case. A secondary actor never initiates the use case. It is invoked by the system's use cases to obtain information or a result.

**Trigger:** A trigger is the initiator of a use case. It is what causes the use case to start. There isn't a promise that this event happens – only an indication that this event triggers the start of a use case.

**Use Case:** A specification of one type of interaction with a system.

**User Class:** A user class is a set of developer-defined attributes (characteristics) and methods (behaviors) that you can use to refer to multiple data items as a single entity.

**User Interface:** The user interface (UI) is the point of human-computer interaction and communication in a device

**IOS:** an operating system used for mobile devices manufactured by Apple Inc.

**Android:** an open-source operating system used for smartphones and tablet computers.

**Graphical User Interface:** A graphical user interface is a form of user interface that allows users to interact with electronic devices through graphical icons and an audio indicator such as primary notation, instead of text-based user interfaces, typed command labels, or text navigation.

**Appendix B: Analysis Models**

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

**Appendix C: To Be Determined List**

**Section 4.3.1.1** [**Req-19**](#_4.3__Calculating).