



**Cégep de la Gaspésie
et des Îles**

FINAL PROJECT

Section 00313



SUBMITTED BY

Rashmeet Kaur (1896566)

Amritpal Singh Sidhu(1896564)

Anshul Kapoor (1896225)

Khushkirat Singh Virk(1896533)

Raminderjit Singh(1894513)

Deepak Maan(1894830)

TABLE OF CONTENTS

1. Project Overview	3
1.1 Purpose of the project	3
1.2 Goal of the project	3
1.3 Scope	3
2. Requirements	4
2.1 Functional requirements	4
2.2 Non-functional requirements	5
3. Use Case Diagram	7-8
4. Use case scenario	9-24
5. Class Diagram	25-26
6. Entity-Relationship Diagram	27
7. Screen Flow (client side)	28-35
8. Screen Flow (Admin Side)	36-40
9. Data Base Schema	41-45
10. References	46

1. PROJECT OVERVIEW

1.1 The Purpose of the project

An Italian ingredients distributor cited the issues of maintaining different lists for delivered and received products manually. Not only this but it was difficult to view and manage orders, whether they have been received or delivered. He had to analyze the inventory items manually which is very time consuming and is prone to human error as well.

It takes a lot of time and effort to keep a record of your sales and generate manual invoice and send it to the customers. An application which could manage all the above-mentioned problems could prove to be very fruitful to the distributor as well as the consumer because the customer also can manage their account. This could help him generate automatic invoices and send to the generated mails by not wasting useful labour in trivial works.

1.2 Goal of the project

The goal of the project to meet requirements at every level for both the Admin and the user. This application is going to help the users (restaurants) maintain the orders, the list of goods which have been ordered, delivered or have been put into a pending list.

The application aims to resolve issues of maintaining manual invoice by replacing it with automatic invoice generation which would then be send to the registered mails automatically. Special notifications would be sent to the customers regarding unauthorized login attempts or delivery updates etc.

1.3 Scope

The scope of an inventory system can cover many needs as follows :

- To ensure that the supply of raw material and finished goods will remain continuous to that production process is not halted and demands of customers are duly met.
- To keep inventory at sufficiently high level to perform production and sales activities smoothly.

- To minimize investment in inventory to maximize profitability.
- To reduce the losses of theft, obsolescence etc.
- To minimize loss through deterioration, pilferage, wastages, and damages.
- To optimize various costs indulged with inventories like purchase cost, carrying a cost, storage cost, etc.
- To avoid both overstocking and under-stocking of inventory.
- To eliminate duplication in ordering stocks.

2. REQUIREMENTS (Functional and Non-functional)

2.1 FUNCTIONAL REQUIREMENTS

REQUIREMENT ID	REQUIREMENT STATEMENT	MOSCOW MUST,SHOULD, COULD,WOULD
FR 1	User must sign up and then login using his/her credentials for using the application.	MUST
FR 2	User can order goods, see the list of pending, received or delivered goods.	COULD
FR 3	User should add the goods to the cart.	SHOULD
FR 4	User could reset the password with the credentials used for signing up.	COULD
FR 5	User Could Cancel the order before accepted by the admin.	COULD
FR 6	Admin should be able to manage CRUD operations.	MUST
FR 7	User gets notified when a new item is added to cart	MUST
FR 8	User can view a list of purchased items in the history.	COULD
FR 9	Maintain right number of products in stock.	SHOULD

FR 10	User can edit their profile information.	COULD
FR 11	User can perform CRUD operations on their end by adding, Deleting Updating the items in the cart.	COULD
F12	User Could search the items from the different Categories	COULD
F13	User must need to accept the terms and conditions.	MUST

2.2 NON-FUNCTIONAL REQUIREMENTS

A non-functional is essential to ensure the usability and effectiveness of the entire software system. Failing these requirements can result into failure of customer satisfaction.

NFR 1 – USABILITY REQUIREMENTS

The application should be easy to use for users working with restaurant industry, people who have knowledge of online shopping and someone who is fully capable of maintaining and managing the inventory operations. It should be designed in a way that users do not face any problems working with the application or understanding it.

NFR 2 – PERFORMANCE REQUIREMENTS

The application should reboot when not working properly without effecting the data of the user and saving it automatically. Bad execution results in negative feedback from the user. It should respond quickly to the user request.

NFR 3 – IMPLEMENTATION REQUIREMENTS

The functionalities of the application should be implemented using Android Studio 3.6.3, Fire-base and Just in mind 8.7.7.

NFR 4 – PORTABILITY REQUIREMENTS

The application should be able to work at all the versions of android (Oreo, nougat, marshmallow etc.).

NFR 5 – PRIVACY REQUIREMENTS

The application does not allow access to personal information to anyone else than the user or the administrator. Nobody can order, change or edit any information on the application without a successful login.

NFR 6 – MAINTAINABILITY REQUIREMENTS

The application should be easy to maintain for the administrator which means he should be able to make any kind of changes (in the prices of the products, he should be able to add or delete products) and these changes made should be made in real time which means for example as soon as there is a change in price of an item the if there is a user using the application, the new price will be reflected.

NFR 7 – RELIABILITY REQUIREMENTS

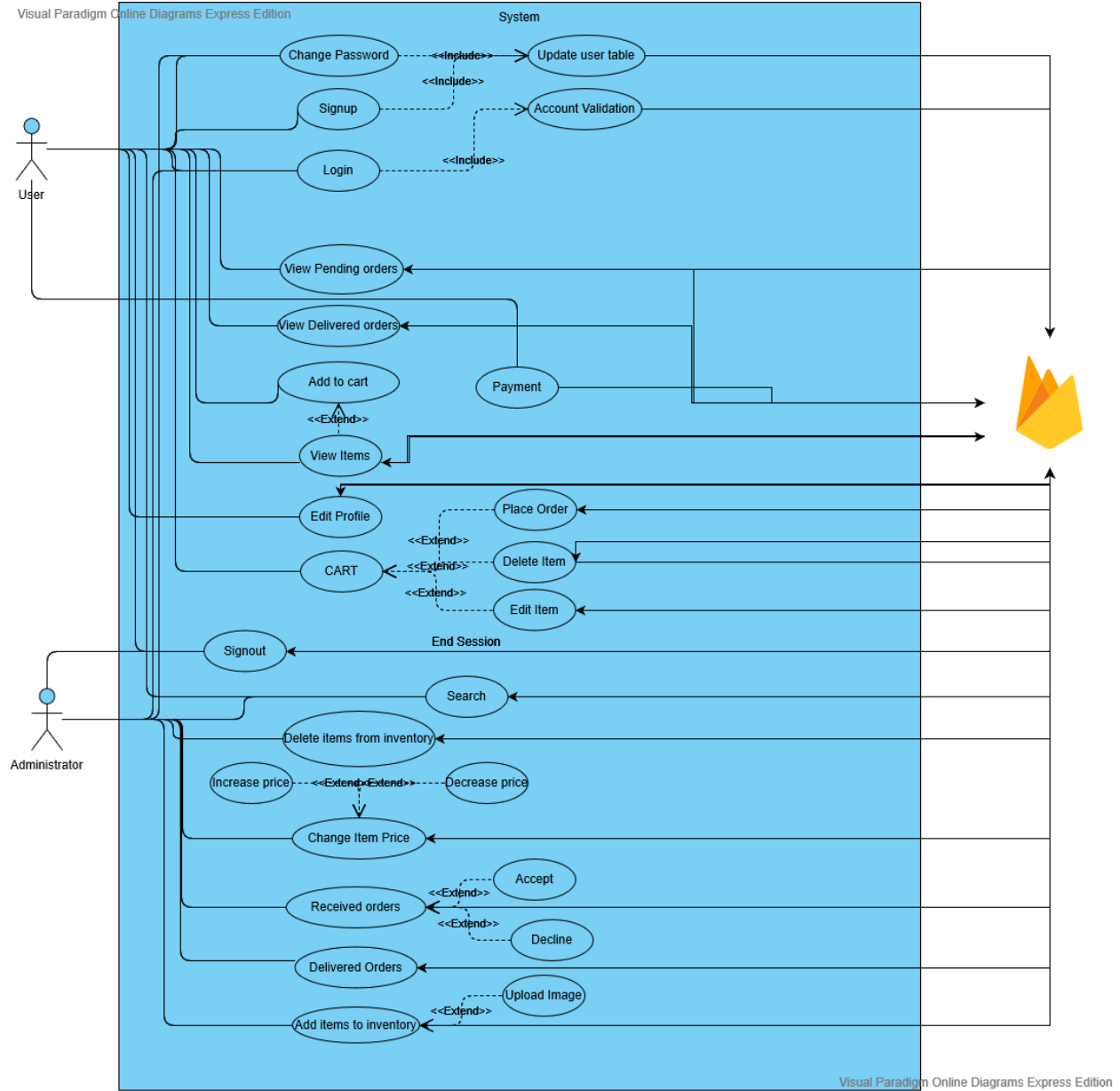
The application should be reliable for making payments for the customer account safety because lack of this could lead to negative feedback from the customers.

3. USE CASE DIAGRAM

A UML (Unified modelling language) use case diagram is a primary form to represent system requirements which is still underdeveloped. The use cases just represent the expected working using visual representation and not the exact method of how it is going to happen.

A use case diagram is created using some symbols and notations. For making a use case diagram we used **VISUAL PARADIGM 16.1**. The symbols & notations are explained below:

- ✧ **USE CASES:** Horizontally shaved ovals that represent the different uses that a user might have.
- ✧ **ACTORS:** Stick figures that represent the people actually employing the use cases.
- ✧ **RELATIONSHIPS:** A link between use cases and actors. There are different kind of relationships in a use case diagram which are used in the diagram below.
- ✧ **SYSTEM BOUNDARY BOXES:** The rectangle around the use cases is called system boundary box and is the scope of the system. The use cases inside the represent the functionality that you intend to implement.



4. USE CASE SCENARIOS

UC1- SIGN UP

System:	SAPORI ITALIANO	
Identifier:	UC-1	
Author(s):	Team 7	
Version:	None	
Name:	Sign Up	
Pre-Condition(s):	The user does not have account on Sapori Italiano.	
Post-Condition(s):	The user is directed to the Login Page of the application.	
Trigger:	The User has clicked on the Sign-up button.	
Normal Flow:	<ol style="list-style-type: none"> 1. User clicks on Sign Up button to create the user profile. 2. User enters personal information such as Username, Email, Phone and password to create an account. 3. User information will be updated in table at Firebase server. 4. User is redirected to the Home Page. 	
Alternate Flow:	From there, user clicks on Login.	
Exceptional Flow(s):	Exception: There is an occurrence of internet connection failure. <ol style="list-style-type: none"> 1. The system will prompt the message “No Internet Connection, try Again”. 	
Related Actor(s):	Primary -New User Secondary -None	
Related Use Case	Login	
	UC-1: Sign Up New User	

UC2- LOG IN

System:	SAPORI ITALIANO	
Identifier:	UC-2	
Author(s):	Team 7	
Version:	None	
Name:	Login	
Pre-Condition(s):	The user is logged In to account on Saponi Italiano.	
Post-Condition(s):	The user is directed to the homepage specific to the user.	
Trigger:	The user has clicked on the Login button.	
Normal Flow:	<ol style="list-style-type: none"> 1. User clicks on the Login button. 2. User enters the user name and the password to get logged In. 3. User clicks on login button. 4. System will match the details with server. 5. The dentails are matched. 6. User is directed to the homepage. 	
Alternate Flow:	User clicks on signup button.	
Exceptional Flow(s):	Exception: If user enters wrong password or username. The system will prompt the message “Wrong User name or password, Try again”.	
Related Actor(s):	Primary -Registered User Secondary -None	
Related Use Case(s):	Logout	
	UC-2: Login	

UC-3 –Home

System:	SAPORI ITALIANO
Identifier:	UC-3
Author(s):	Team 7
Version:	None
Name:	Home
Pre-Condition(s):	1. The user is registered and has an account on Saponi Italiano 2. User Logs In with correct username and password
Post-Condition(s):	The user is directed to the Home page.
Trigger:	The user has clicked on the login button.
Normal Flow:	1. User Logs In with correct username and password 2. User is directed to home page. 3. System will fetch details from firebase. 4. Entries will be displayed to the user.
Alternate Flow:	None
Exceptional Flow(s):	There is connection failure.
Related Actor(s):	Primary -Registered User
Related Use Case(s):	None
	UC-3: Home

UC-4- Cart

System:	SAPORI ITALIANO
Identifier:	UC-4
Author(s):	Team 7
Version:	None
Name:	My Cart
Pre-Condition(s):	1. The user is registered and has an account on Saponi Italiano 2. User Logs In with correct username and password 3. User Choose the Options of my cart from bottom.
Post-Condition(s):	The user is directed to my cart page.
Trigger:	The user has clicked on my cart option.
Normal Flow:	1. User is on the home page 2. User clicks on my cart option. 3. User is directed to my cart page. 4. System will fetch cart items from Server. 5. User can edit items from cart.
Alternate Flow:	None
Exceptional Flow(s):	None
Related Actor(s):	Primary -Logged in User
Related Use Case(s):	Pending Orders
	UC-4: Cart

UC-5– Pending Orders

System:	SAPORI ITALIANO
Identifier:	UC-5
Author(s):	Team 7
Version:	None
Name:	Pending Orders
Pre-Condition(s):	1 The user is registered and has an account on Saponi Italiano 2. User Logs In with correct username and password 3. User Choose the Options of pending orders from side menu.
Post-Condition(s):	The user is directed to pending orders page.
Trigger:	The user has clicked on pending orders option.
Normal Flow:	1. User is on the home page 2. User clicks on the Pending orders button. 3. User is directed to pending orders page. 4. System will fetch orders that has Pending status and display those to user.
Alternate Flow:	None
Exceptional Flow(s):	None
Related Actor(s):	Primary -Logged In User
Related Use Case(s):	Delivered Orders
UC-5: Pending Orders	

UC-6– Delivered Orders

System:	SAPORI ITALIANO
Identifier:	UC-6
Author(s):	Team 7
Version:	None
Name:	Delivered Orders
Pre-Condition(s):	1 The user is registered and has an account on Sabori Italiano 2. User Logs In with correct username and password.
Post-Condition(s):	The user is directed to delivered orders page.
Trigger:	The user has clicked on delivered orders option.
Normal Flow:	1. User is on the home page 2. User clicks on delivered orders button. 3. System will fetch items from firebase that has status delivered. 4. System will display delivered orders to the user.
Alternate Flow:	None
Exceptional Flow(s):	None
Related Actor(s):	Primary -Logged In User
Related Use Case(s):	New Orders
	UC-6: Delivered Orders

UC-7 – Profile

System:	SAPORI ITALIANO
Identifier:	UC-7

Author(s):	Team 7
Version:	None
Name:	Profile
Pre-Condition(s):	1. The user is registered and has an account on Saponi Italiano 2. User Logs In with correct username and password. 3. User is on Settings page of the application.
Post-Condition(s):	The user is directed to the Profile page.
Trigger:	The user has clicked on the Edit profile button.
Normal Flow:	1. User clicks on the Edit profile button. 2. User edits the personal information in the form. 3. User clicks on submit. 4. User information will be updated on server.
Alternate Flow:	None
Exceptional Flow(s):	None
Related Actor(s):	Primary -Registered User
Related Use Case(s):	Login
	UC-7: Profile

UC-8–Change Password

System:	SAPORI ITALIANO
Identifier:	UC-8
Author(s):	Team 7
Version:	None
Name:	Change Password
Pre-Condition(s):	1. The user is registered and has an account on Sapori Italiano 2. User is on LOGIN page.
Post-Condition(s):	The user is directed to Login page.
Trigger:	User clicks on Forgot password options.
Normal Flow:	1. User is on the LOGIN page 2. User clicks on Forgot password button 3. User is directed to change password page. 4. User enters email address and clicked on validate. 4. If validated, user enters new password and confirm password. 5. Changes will be updated in user table on firebase. 6. User will be directed to login page.
Alternate Flow:	None
Exceptional Flow(s):	None
Related Actor(s):	Primary -Logged In User
Related Use Case(s):	None
	UC-8: Change Password

UC-9-Logout

System:	SAPORI ITALIANO
Identifier:	UC-9
Author(s):	Team 7
Version:	None
Name:	Logout
Pre-Condition(s):	1.The user is registered user and has an account on Sapori Italiano. 2. User is on Setting page.
Post-Condition(s):	The user is directed to Login page.
Trigger:	User clicks on logout button.
Normal Flow:	1. User is on the home page. 2. User chooses Setting button. 3. User clicks on logout button. 4. System will end the current session. 5. User will be redirected to Login page.
Alternate Flow:	None
Exceptional Flow(s):	None
Related Actor(s):	Primary -Logged In User
Related Use Case(s):	Login
UC-9: Logout	

UC-10: Cart

System:	SAPORI ITALIANO	
Identifier:	UC-10	
Author(s):	Team 7	
Version:	None	
Name:	Cart	
Pre-Condition(s):	1. The user has an account on Saponi Italiano and logged in. 2. The user is on Cart page.	
Post-Condition(s):	Changes will be reflected in the firebase.	
Trigger:	The user has clicked on the editing buttons (increase, decrease, delete) available on the cart page.	
Normal Flow:	1. User click on increase, decrease or delete button. 2. The changes will be reflected in the cart as well as on the firebase. 3. The user can place the order whenever he want to.	
Alternate Flow:	None	
Related Actor(s):	Primary -Registered User	
	Secondary -None	
Related Use Case(s):	Pending Orders	
	UC-10 cart	

UC-11: Add item

System:	SAPORI ITALIANO	
Identifier:	UC-11	
Author(s):	Team 7	
Version:	None	
Name:	Add item	
Pre-Condition(s):	1. The user has an account on Saponi Italiano and logged in as administrator. 2. The user has clicked on plus sign available in a particular category which add items.	
Post-Condition(s):	The user is directed to add item page of the application.	
Trigger:	The user has clicked on the plus button available in a particular category.	
Normal Flow:	1. User clicks on plus button on top right corner. 2. User is redirected to the add item Page. 3. User has to give name, description, quantity and price of the item. 4. User can add image of the item if needed. 5. User clicks on Add item. 6. Item will be added in that particular category.	
Alternate Flow:	None	
Exceptional Flow(s):		
Related Actor(s):	Primary-Administrator	
	Secondary-None	
Related Use Case(s):	Add item	
	UC-11: Add item	

UC-12: Delivered Orders

System:	SAPORI ITALIANO	
Identifier:	UC-12	
Author(s):	Team 7	
Version:	None	
Name:	Delivered Orders	
Pre-Condition(s):	1. The user has an account on Saponi Italiano and logged in. 2. The user has clicked on Delivered Orders button from bottom navigation bar.	
Post-Condition(s):	The user is directed to Delivered Orders Page of the application.	
Trigger:	The user has clicked on the Delivered Orders button.	
Normal Flow:	1. User clicks on Delivered Orders button to see list of orders that are accepted by admin. 2. User is redirected to the Delivered Orders Page. 4. System will fetch the list from firebase and display it to user.	
Alternate Flow:	None	
Related Actor(s):	Primary -Registered User	
	Secondary -None	
Related Use Case(s):	Pending Orders	
	UC-12: Delivered Orders	

UC-13: Pending Orders

System:	SAPORI ITALIANO	
Identifier:	UC-13	
Author(s):	Team 7	
Version:	None	
Name:	Pending Orders	
Pre-Condition(s):	1. The user has an account on Saponi Italiano and logged in. 2. The user has clicked on Pending Orders button from bottom navigation bar.	
Post-Condition(s):	The user is directed to Pending Orders Page of the application.	
Trigger:	The user has clicked on the Pending Orders button.	
Normal Flow:	1. User clicks on Pending Orders button to see list of orders that he has placed. 2. User is redirected to the Pending Orders Page. 3. System will fetch the list from firebase and display it to user. 4. User can edit or delete items from order.	
Alternate Flow:	None	
Related Actor(s):	Primary -Registered User	
	Secondary -None	
Related Use Case(s):	Received Orders	
	UC-13: Pending Orders	

UC-14: Search Item

System:	SAPORI ITALIANO	
Identifier:	UC-14	
Author(s):	Team 7	
Version:	None	
Name:	Search Item	
Pre-Condition(s):	1. The user has an account on Saponi Italiano and logged in. 2. The user has clicked on Search item at home page.	
Post-Condition(s):	The user is directed to particular page where the item is present	
Trigger:	The user has clicked on the search Item.	
Normal Flow:	1. User clicks on search item for searching a particular product. 2. System will match the string with related entries in the firebase. 3. Once matched, system will display those items to user.	
Alternate Flow:	If not matched, system will display message “No result found”.	
Flow(s):		
Related Actor(s):	Primary -Administrator, Registered User	
	Secondary -None	
Related Use Case(s):	None	
Case(s):		
	UC-14: Search Item	

UC-15: Received Orders

System:	SAPORI ITALIANO	
Identifier:	UC-15	
Author(s):	Team 7	
Version:	None	
Name:	Received Orders	
Pre-Condition(s):	1. The user has an account on Saponi Italiano and logged in. 2. The user has clicked on Received Orders button.	
Post-Condition(s):	The user is directed to Received Orders Page of the application.	
Trigger:	The user has clicked on the Received Orders button.	
Normal Flow:	1. User clicks on Received Orders button to see list of orders. 2. User is redirected to the Received Orders Page. 3. System will fetch the orders that are placed by users. 4. System will display the result.	
Alternate Flow:	None	
Flow(s):		
Related Actor(s):	Primary -Administrator	
	Secondary -None	
Related Use Case(s):	None	
	Uc-15 Received Orders	

UC-16: Payment

System:	SAPORI ITALIANO	
Identifier:	UC-17	
Author(s):	Team 7	
Version:	None	
Name:	Payment	
Pre-Condition(s):	1. The user has an account on Saponi Italiano and logged in. 2. The user has clicked on Place order Button in his cart. 3. The user is directed to make payment page.	
Post-Condition(s):	The user is directed to make payment page of the application.	
Trigger:	The user has clicked on the place order button.	
Normal Flow:	1. User clicks on place order button in his cart. 2. User is redirected to the make payment page. 3. User has to fill in the card details. 4. User clicks on make payment button to pay the invoice.	
Alternate Flow:	None	
Exceptional Flow(s):	1. The details entered are not correct. 2. There is connection failure.	
Related Actor(s):	Primary -Administrator Secondary -None	
Related Use	None	
Case(s):		
	UC-16:Payment	

5. CLASS DIAGRAM

A class diagram is a type of static structure diagram that describes the structure of the system by showing the system's:

1. **Class:** A class is blueprint of an object. These two go hand in hand and we can't talk about one of them without mentioning the other. Every class describes a type of object.
2. **Attributes:** An attribute represents a characteristic of a class that is of interest for the user of the IT system.
3. **Operations** (or methods): An operation is a method or a function that can be performed by an instance of a class or interface.

In a class diagram class is represented with boxes that contain three compartments:

- The top compartment contains the name of the class. First letter for the class name is capital.
- The middle compartment contains the attributes of the class. .
- The bottom compartment contains the operations the class can execute.

Association in classes:

Association is basically a relationship between the two different classes , how two different classes and their attributes interact with each other.

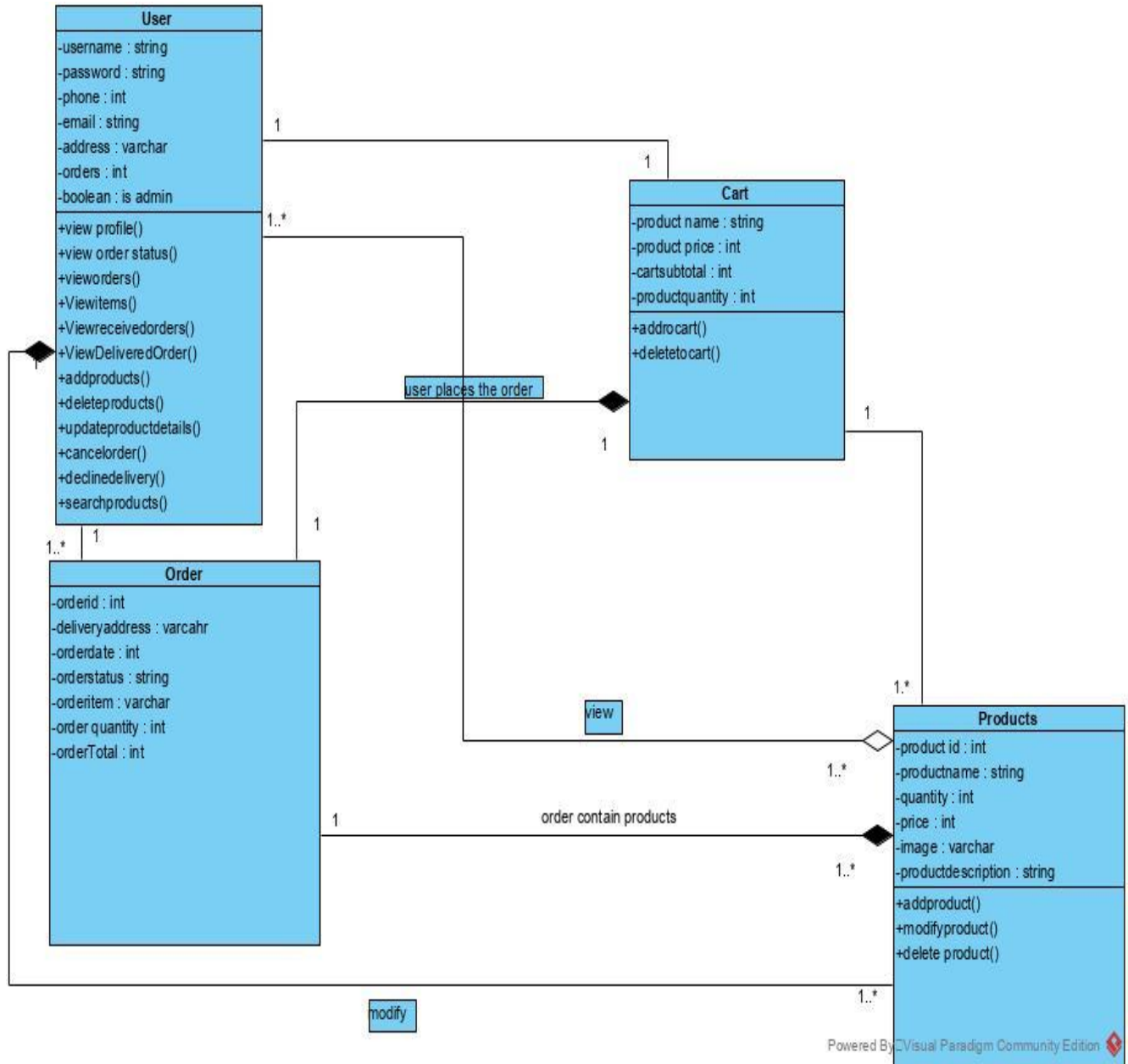
Some different form of association is discussed as below:

1. **Association:** Both classes are equal.

For example:

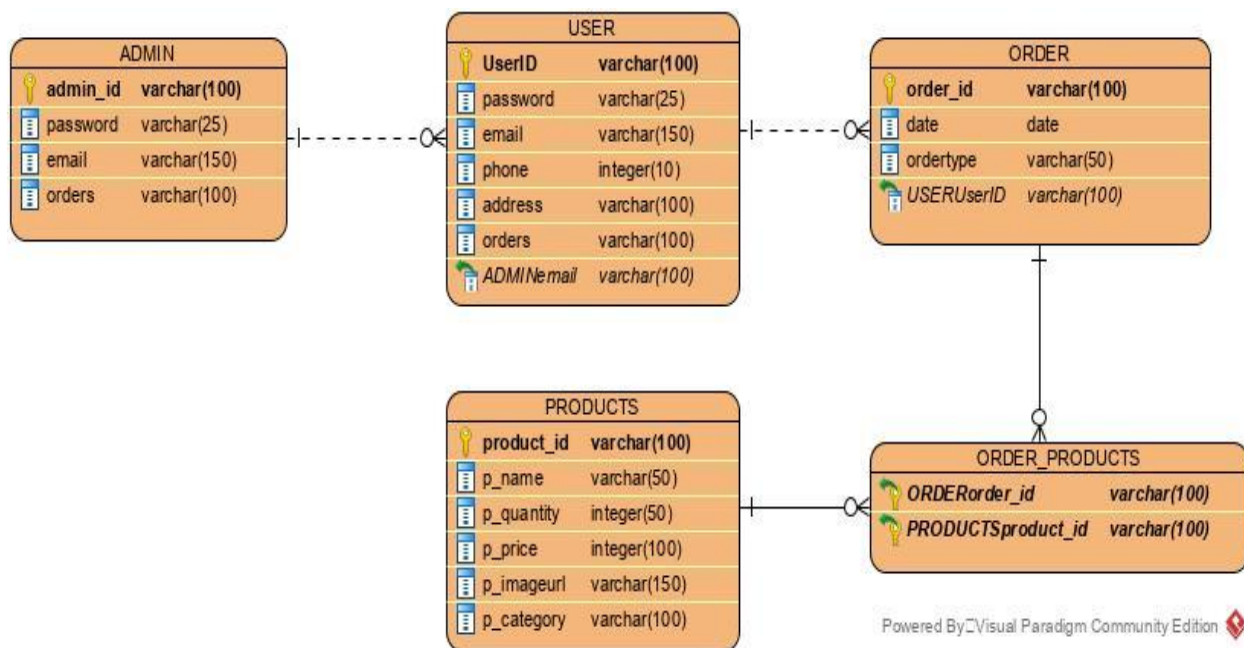
A person may own none, one or many cars, and a car must have one or many owners. And they are at the same level, none of them is a part of the other.

2. **Aggregation:** Illustrated with an open diamond. We have a hierarchy where the class on the diamond end is the whole, and the class on the other end is a part.
3. **Composition:** Illustrated with a filled diamond. The class on the diamond end owns the class on the other end.



6. ENTITY-RELATIONSHIP DIAGRAM(ERD)

Entity Relationship Diagram, also known as ERD, ER Diagram or ER model, is a type of structural diagram for use in database design. An ERD contains different symbols and connectors that visualize two important information: **The major entities within the system scope**, and the **inter-relationships among these entities**.



7. SCREEN FLOW

❖ Client Side

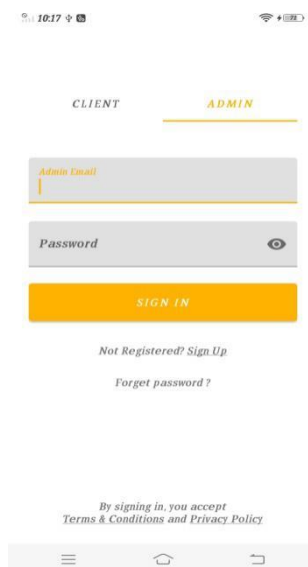
- Splash Screen

This is the first screen which will appear when user runs the application.



- Client and Admin Login

This screen appears after splash screen from which user can login, sign up and change its password. It contains toggle buttons for user and admin.



- Sign up

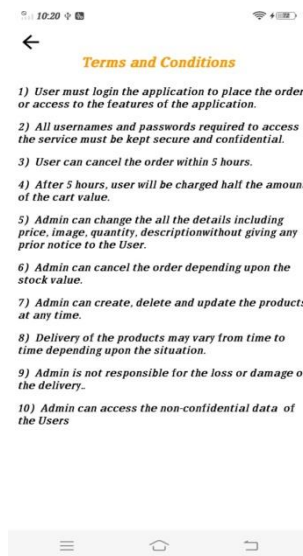
From this screen user can fill its credentials to sign up for this application.



A screenshot of a mobile application's sign-up form. The form consists of several input fields stacked vertically: Name, E-mail, Confirm E-mail, Phone Number, Street, Apartment, City, State, Zip, Country, Password, and Confirm Password. Each field has a placeholder text. Below the fields, there is a checkbox labeled 'Terms and Conditions' with a link 'Read All' next to it. At the bottom, there are two buttons: a yellow 'CREATE ACCOUNT' button and a red 'CANCEL AND GO BACK' button. The status bar at the top shows the time as 10:16 and various icons.

- Terms and Conditions

This is terms and condition page which appears when user clicks on the check box to accept the terms and conditions to use this application.



A screenshot of the 'Terms and Conditions' page in a mobile application. The page has a white background with a black back arrow icon at the top left. The title 'Terms and Conditions' is centered at the top in a bold, orange font. Below the title, there is a list of 10 terms and conditions, numbered 1 through 10. The text is in a small, black font. At the bottom of the page, there is a navigation bar with three icons: a hamburger menu, a home icon, and a back arrow.

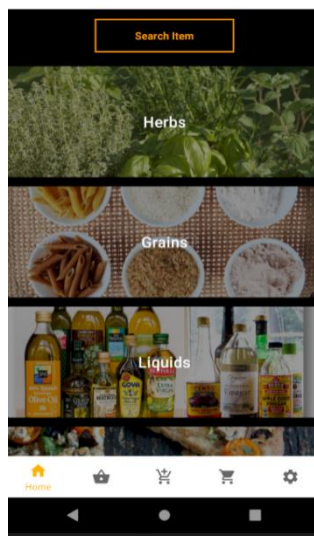
- **Forgot Password**

This screen opens when user clicks on forgot password on Admin and client login screen.



- **Home Screen**

This screen appears when client is successfully logged in the application. This screen has search option, categories available and bottom navigation bar with black background.



- Items in Specific Category

This screen appears when user clicks on particular category which shows all the items available with image, name and price.



- Product Details

This screen opens when the user choose a specific item from the list.

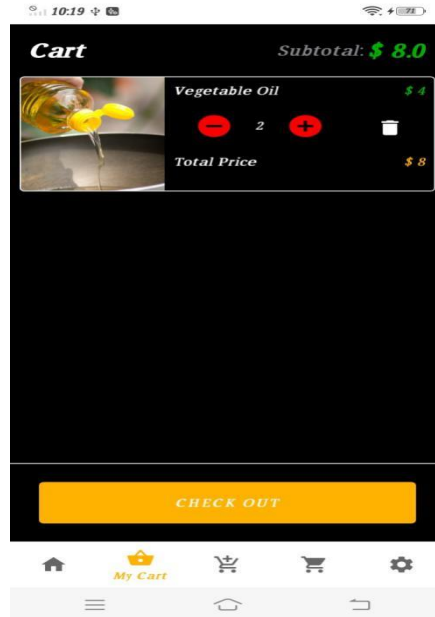
This page shows item name, item details, weight, price per item, items in stock and gives users the option to choose item the want to buy (max 10).



- User's cart

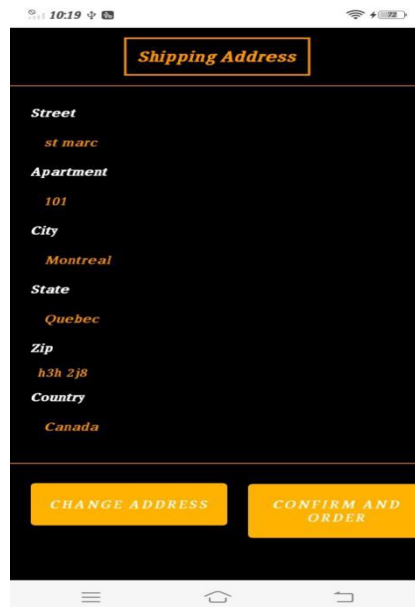
The is user's cart page it opens when user add product to cart.

It gives option to add or subtract the number of items and option to delete whole item from the cart .It Shows Subtotal of items in cart.



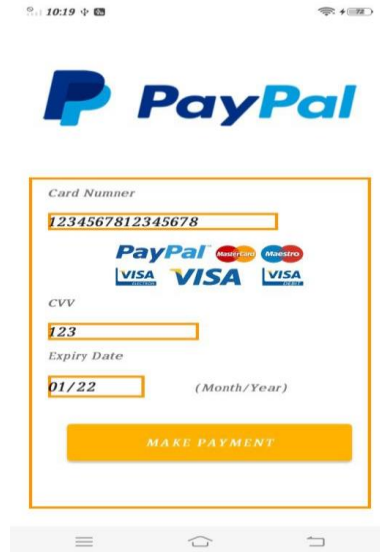
- Shipping Address

When user clicks on checkout button it prompt this screen on which user can fill their desired address and can make changes to it.



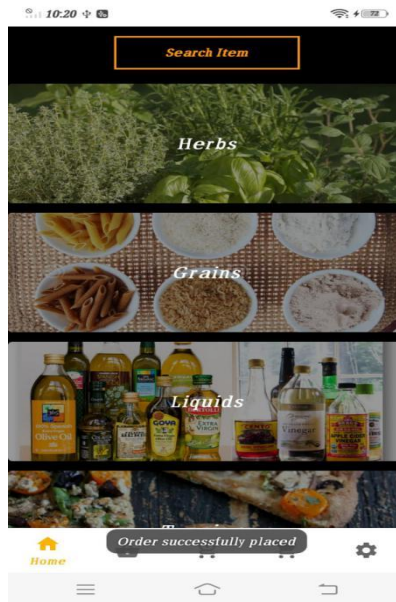
- Make Payment

When user clicks on confirm and order button on shipping address page it opens payment page where client enters it valid credentials to make payment via PayPal.



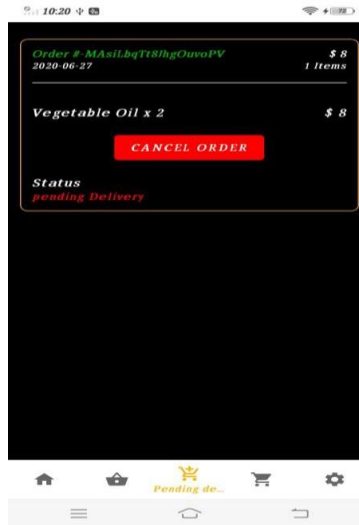
- Order Paced{Toast}

When the order is successfully placed it bring back user to home page and show the toast at the bottom.



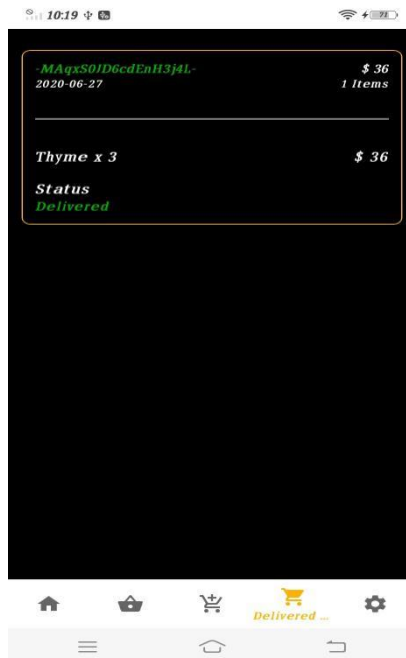
- Pending Orders Fragment

User has chosen pending orders from bottom navigation bar. It gave user option for canceling the order and also shows the status of the order.



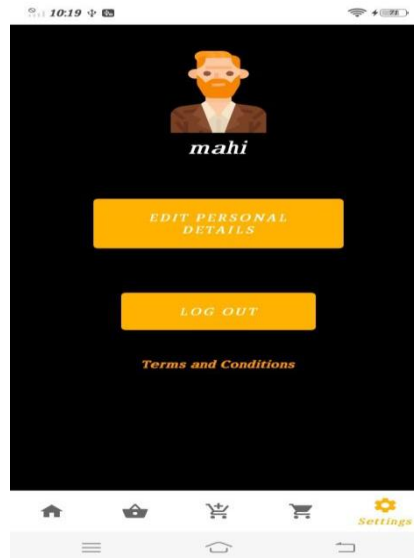
- Delivered Orders Fragment

User has chosen delivered orders from bottom navigation bar. It shows orders which are being successfully delivered to user.



- Settings

User has chosen settings from bottom navigation bar which provides options for logging out the user account and change user details.



- Edit Details by Client

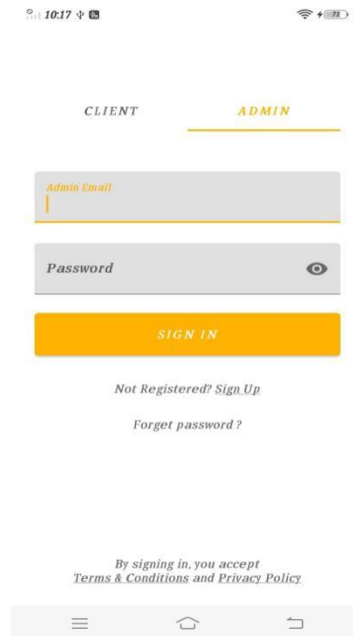
This page helps user to edit personal profile.

A screenshot of the 'Edit Personal Details' form in the app. The title 'Edit Personal Details' is at the top in orange. Below it, there are several input fields with labels: 'Name' (with 'mahi' entered), 'Street' (with 'st marc' entered), 'Apartment' (with '101' entered), 'City' (with 'Montreal' entered), 'State' (with 'Quebec' entered), 'Zip' (with 'h3h 2j8' entered), 'Country' (with 'Canada' entered), and 'Phone' (with '2345670986' entered). At the bottom of the form is a yellow button labeled 'UPDATE USER'. The status bar at the top shows the time as 10:20 and battery level at 92%.

❖ Admin Side

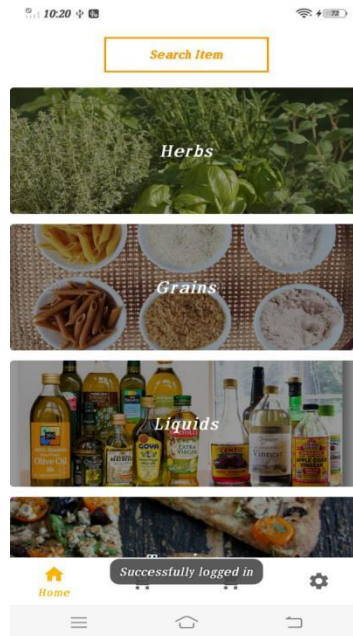
- Admin Login Side

This screen appears after splash screen from which user can login, sign up and change its password. It contains toggle buttons for user and admin.



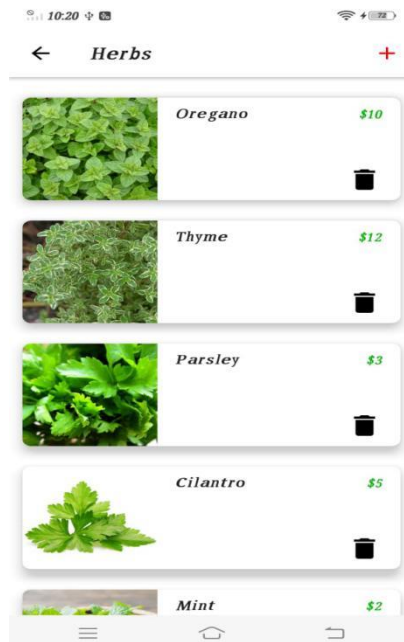
- Home Page (Admin Side)

This screen appears when admin is successfully logged in the application. This screen has search option, categories available and bottom navigation bar with white background.



- Items at Admin End

This page opens when admin enters particular category of herbs. Admin Can add new items into the category and can even delete any item.



- Edit Items

Items being edited by admin.

10:20

Edit Item

Name

Oregano

Description

Oregano is a herb from the mint, or Lamiaceae family. People have used it for thousands of years to add flavour to dishes and to treat health conditions. It features in the Mediterranean diet.

Price

10

Weight

500 grams

Quantity

37

UPDATE ITEM

- Add New Items

Admin Clicks on Plus sign on top right corner to add new products.

10:21

← Add

Name

Price

Quantity

Weight

Description

CHOOSE IMAGE

ADD

- Delivered orders

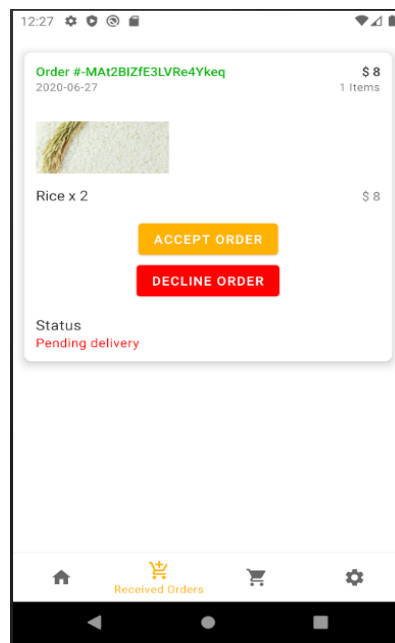
The admin has selected delivered orders from bottom navigation bar.



- Received orders

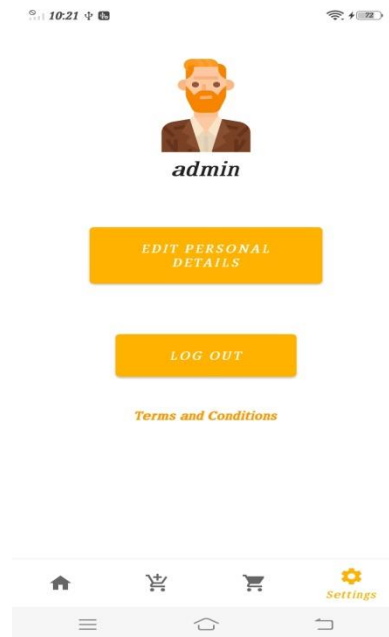
The admin has selected received orders from bottom navigation bar.

Admin has privilege to accept or decline the order.



- Settings

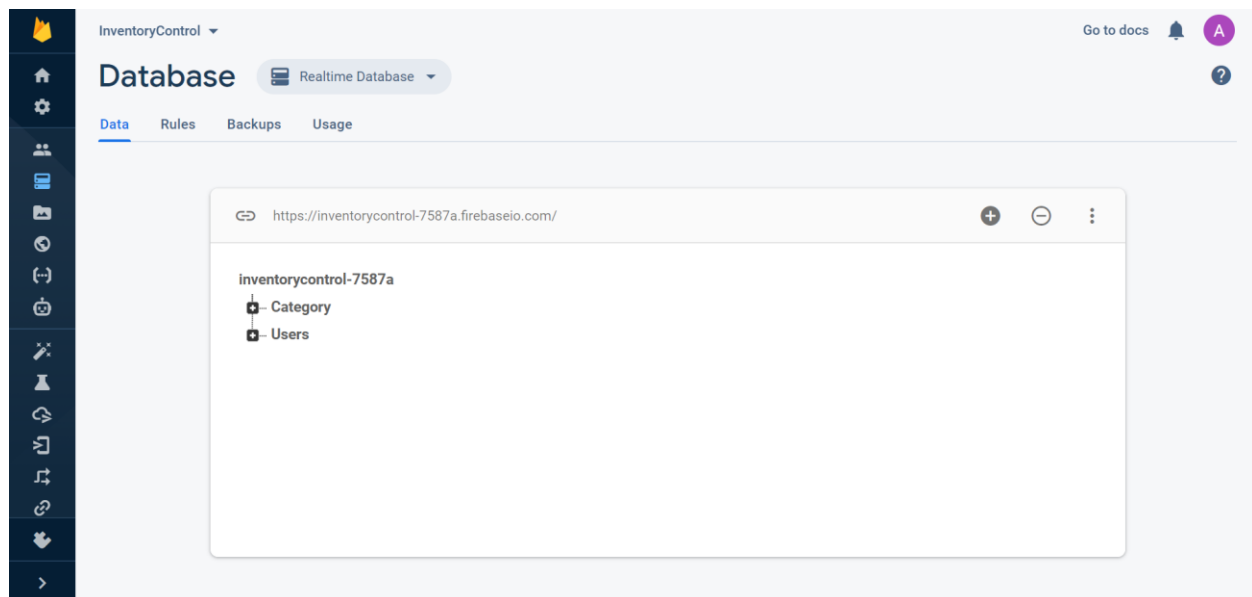
Admin has chosen settings from bottom navigation bar which provides options for logging out the admin account and change admin details.



Data Base Schema:

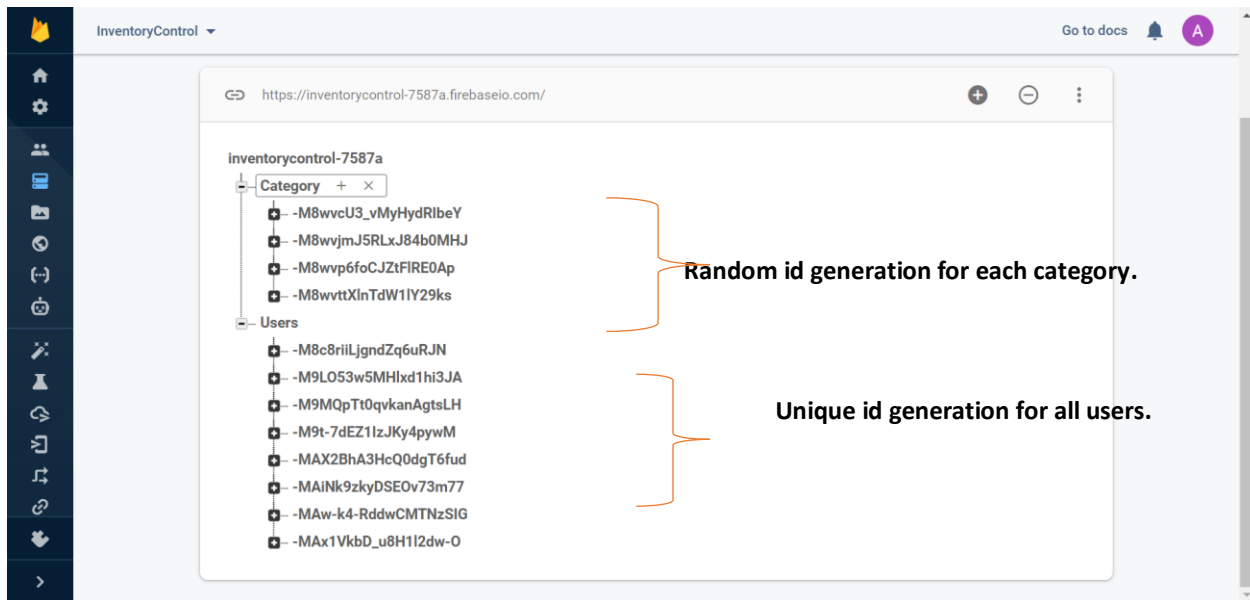
Data base is the core feature of the application .It makes the application fast, flexible, easy to use at anywhere. In this Inventory ordering and management application we are using the fire base database which is very efficient and fast to use.

Category for the products and **User** are basic building tables of the application

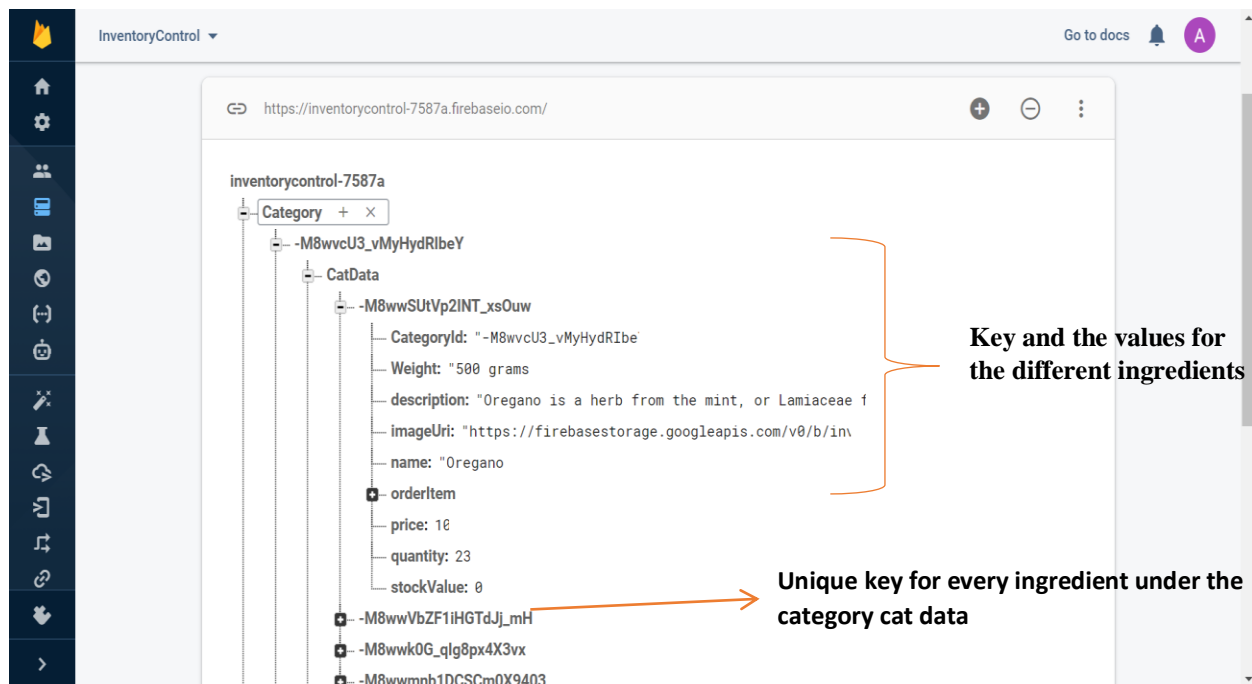


Here are the expanding the main nodes Category and users respectively.

We have 4 main Categories of ingredients and many users



Expanding further the category node, we can see the child nodes and every value has a unique key generated by the Firebase when we push the data to the database. **Expanding the data for the herbs section.**



In this section we are expanding the main node of the user and from the child node we further expand the details of admin with all the key and values like the details of the user and the orders of the user.



The screenshot displays a web application interface for 'InventoryControl'. The main content area shows a JSON tree structure representing user and order data. The tree is as follows:

```

{
  password: "1",
  phone: "",
  "-M9L053w5MH1xd1hi3JA": {
    PushToken: "cV1kROKD0Ro:APA91bHv0E3uypnyaNw7tZ08t0eCN4tqpfl",
    address: "anshul",
    email: "anshul",
    id: "-M9L053w5MH1xd1hi3J",
    isAdmin: false,
    name: "anshul",
    orders: [
      "-M9LP61p5OgjMlg9Sxfl": {
        ClientId: "-M9L053w5MH1xd1hi3J",
        ClientName: "anshul",
        orderDate: "2020-06-08",
        orderItems: {
          orderStatus: "delivered"
        }
      },
      "-M9LPYieuyS_ryMwTdCO",
      "-M9OUtoWUOJ0KZPOFPhq",
      "-M9OzNPWbau9loQl8v4c",
      "-M9U4eCHQvQJ3fKzaUbc"
    ]
  }
}

```

Annotations on the image:

- User details with ordered items:** Points to the user object (starting with `-M9L053w5MH1xd1hi3JA`).
- Expanding the child node under the user order section:** Points to the `orderItems` object within the first order.

References

<https://developer.android.com/>

<https://stackoverflow.com/>

<https://www.udemy.com/>

<https://www.geeksforgeeks.org/>

<https://www.javatpoint.com/>