

FINAL PROJECT

Section 00313

****

SUBMITTED BY

Rashmeet Kaur (1896566)

Amritpal Singh (1896564)

Anshul Kapoor (1896225)

Khushkirat Singh Virk (1896533)

Raminderjit Singh(1894513)

Deepak Maan(1894830)

TABLE OF CONTENTS

1. ***Project Overview****……...……………………………………..3*
   1. *Purpose of the project…………………………………..3*
   2. *Goal of the project………………………………………3*
   3. *Scope…………………………………………………….3*
2. ***Requirements****…………………………………………….…4*
   1. *Functional requirements……………………………….4*
   2. *Non-functional requirements…………………………..5*
3. ***Use Case Diagram…****……………………………………..6-8*
4. ***Use case scenario****………………………………………9-21*
5. ***Class Diagram****………………………………………….22-23*
6. ***Entity-Relationship Diagram****………………………...24-25*
7. **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.

1. **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 | Special notifications should be sent and received for orders placed or delivered . | SHOULD |
| 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 order history. | COULD |
| FR 11 | User can perform CRUD operations on their end by adding, Deleting Updating the items in the cart. | COULD |

**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 heshould 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**

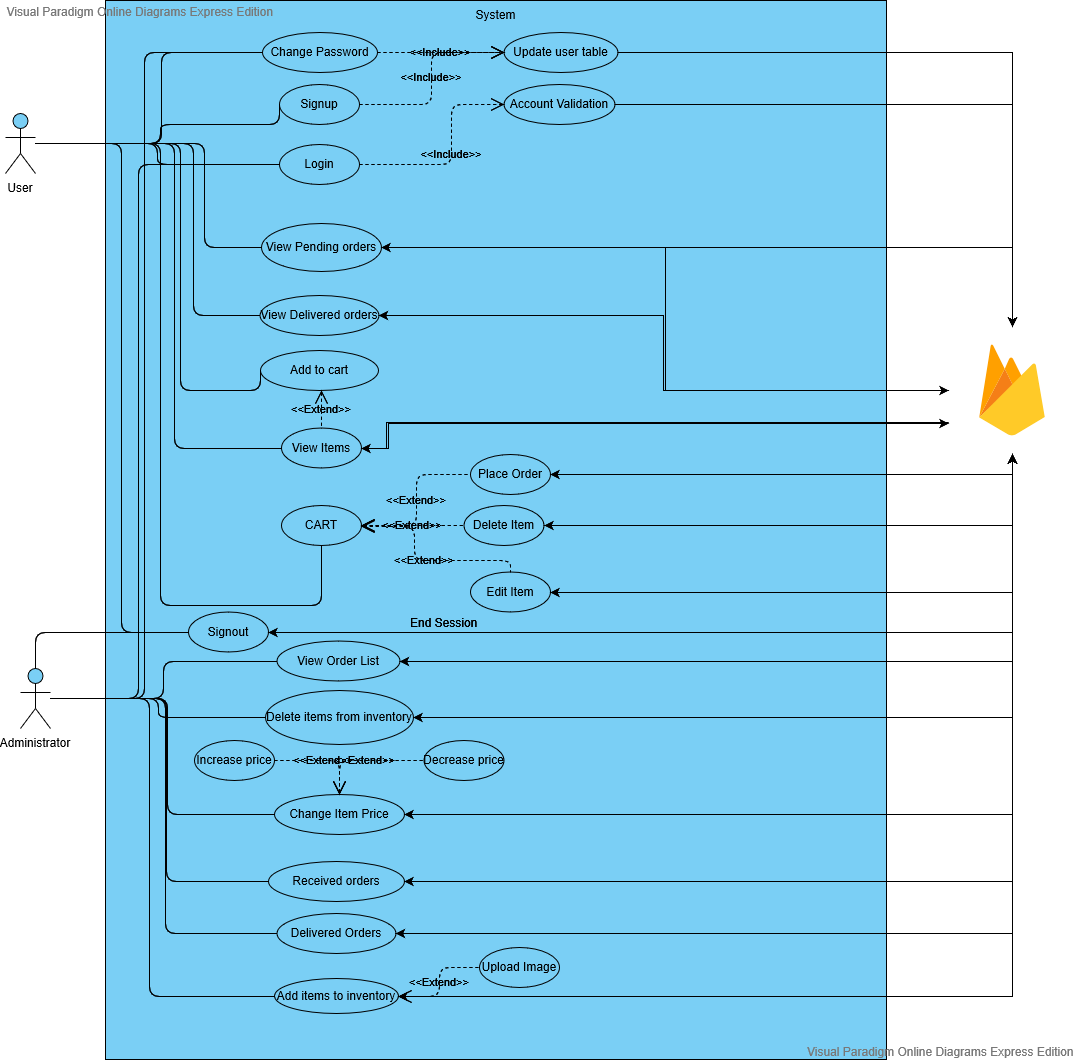
The application should be reliable for making payments for the customer account safety because lack of this could lead to negative feedback from the customers.

1. **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.



1. **USE CASE SCENARIOS**
2. *UC1- SIGN UP*

|  |  |  |
| --- | --- | --- |
| System: | SAPORI ITALIANO |  |
|  |  |  |
| Identifier: | UC-1 |  |
|  |  |  |
| Author(s): | Team 7 |  |
|  |  |  |
| Version: | None |  |
|  |  |  |
| Name: | Sign Up |  |
|  |  |  |
| Pre- | The user doesnot have account on Sapori Italiano. |  |
| Condition(s): |  |  |
|  |  |  |
| Post- | The user is directed to the Login Page of the application. |  |
| Condition(s): |  |  |
|  |  |  |
| Trigger: | The User has clicked on the Sign-up button. |  |
|  |  |  |
| Normal Flow: | 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 login Page. |  |
|  |  |  |
| Alternate Flow: | From there, user clicks on Login. |  |
|  |  |  |
| Exceptional | **Exception:** There is an occurrence of internet connection failure. |  |
| Flow(s): | 1. The system will prompt the message “No Internet Connection, try Again”. |  |
|  |  |
|  |  |  |
|  |  |  |
| Related | **Primary**-New User |  |
| Actor(s): | **Secondary**-None |  |
|  |  |
|  |  |  |
| Related Use | Login |  |
| Case(s): |  |  |
|  |  |  |
|  | UC-1: Sign Up New User |  |

1. *UC2- LOG IN*

|  |  |  |
| --- | --- | --- |
| System: | SAPORI ITALIANO |  |
|  |  |  |
| Identifier: | UC-2 |  |
|  |  |  |
| Author(s): | Team 7 |  |
|  |  |  |
| Version: | None |  |
|  |  |  |
| Name: | Login |  |
|  |  |  |
| Pre- | The user is logged In to account on Sapori Italiano. |  |
| Condition(s): |  |  |
|  |  |  |
| Post- | The user is directed to the homepage specific to the user. |  |
| Condition(s): |  |  |
|  |  |  |
| Trigger: | The user has clicked on the Login button. |  |
|  |  |  |
| Normal Flow: | 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 details are matched. |  |
|  | 6. User is directed to the homepage. |  |
| Alternate Flow: | User clicks on signup button. |  |
|  |  |  |
| Exceptional | **Exception:** If user enters wrong password or username. |  |
| Flow(s): | 2. The system will prompt the message “Wrong User name or password, Try again”. |  |
|  |  |
|  |  |  |
|  |  |  |
| Related | **Primary**-Registered User |  |
| Actor(s): | **Secondary**-None |  |
|  |  |
|  |  |  |
| Related Use | Logout |  |
| Case(s): |  |  |
|  |  |  |
|  | UC-2: Login |  |

1. *UC3- MY CART*

|  |  |
| --- | --- |
| System: | SAPORI ITALIANO |
|  |  |
| Identifier: | UC-3 |
|  |  |
| Author(s): | Team 7 |
|  |  |
| Version: | None |
|  |  |
| Name: | My Cart |
|  |  |
| Pre- | 1. The user is registered and has an account on Sapori Italiano |
| Condition(s): | 2. User Logs In with correct username and password  3. User Choose the Options of my cart from side menu. |
|  |  |
| Post- | The user is directed to my cart page. |
| Condition(s): |  |
|  |  |
| 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. |
|  |  |
|  |  |
| Alternate Flow: | None |
|  |  |
| Exceptional | None |
| Flow(s): |  |
|  |  |
| Related | **Primary**-Logged in User |
| Actor(s): |  |
|  |  |
| Related Use | Pending Orders |
| Case(s): |  |
|  |  |
|  | UC-3: My Cart |

1. *UC4- PENDING ORDERS*

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

1. *UC5- DELIEVERED ORDERS*

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

1. *UC6- NEW ORDER*

|  |  |
| --- | --- |
| System: | SAPORI ITALIANO |
|  |  |
| Identifier: | UC-6 |
|  |  |
| Author(s): | Team 7 |
|  |  |
| Version: | None |
|  |  |
| Name: | New Order |
|  |  |
| Pre- | 1. The user is registered and has an account on Sapori Italiano |
| Condition(s): | 2. User Logs In with correct username and password |
|  |  |
| Post- | The user is directed to View items page. |
| Condition(s): |  |
|  |  |
| Trigger: | The user has clicked on Plus Button. |
|  |  |
| Normal Flow: | 1. User is on the home page |
|  | 2. User clicks on the plus button.  3. System will fetch all items in inventory from firebase.  4. User clicked on add to cart.  5. Details of new order will be saved in firebase. |
| Alternate Flow: | None |
|  |  |
| Exceptional | None |
| Flow(s): |  |
|  |  |
| Related | **Primary**-Logged In User |
| Actor(s): |  |
|  |  |
| Related Use |  |
| Case(s): | None |
|  |  |
|  | UC-6: New Order |

1. *UC7- PROFILE*

|  |  |
| --- | --- |
| System: | SAPORI ITALIANO |
|  |  |
| Identifier: | UC-7 |
|  |  |
| Author(s): | Team 7 |
|  |  |
| Version: | None |
|  |  |
| Name: | Profile |
|  |  |
| Pre- | 1. The user is registered and has an account on Sapori Italiano |
| Condition(s): | 2. User Logs In with correct username and password. |
|  |  |
| Post- | The user is directed to the enter information page. |
| Condition(s): |  |
|  |  |
| Trigger: | The user has clicked on the update profile. |
|  |  |
| Normal Flow: | 1. User clicks on the user profile button. |
|  | 2. User enters the personal information in the form. |
|  | 3. User clicks on submit. |
|  | 4. User information is saved on server and don’t have to fill information again. |
| Alternate Flow: | None |
|  |  |
| Exceptional | None |
| Flow(s): |  |
|  |  |
| Related | **Primary**-Registered User |
| Actor(s): |  |
|  |  |
| Related Use | Login |
| Case(s): |  |
|  |  |
|  | UC-7: Profile |

1. *UC8- CHANGE PASSWORD*

|  |  |
| --- | --- |
| System: | SAPORI ITALIANO |
|  |  |
| Identifier: | UC-8 |
|  |  |
| Author(s): | Team 7 |
|  |  |
| Version: | None |
|  |  |
| Name: | Change Password |
|  |  |
| Pre- | 1. The user is registered and has an account on Sapori Italiano |
| Condition(s): | 2. User is on setting page. |
|  |  |
| Post- | The user is directed to Login page. |
| Condition(s): |  |
|  |  |
| Trigger: | User clicks on change password options. |
|  |  |
| Normal Flow: | 1. User is on the setting page |
|  | 2. User clicks on change password options  3. User is directed to change password page.  4. 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 | None |
| Flow(s): |  |
|  |  |
| Related | **Primary**-Logged In User |
| Actor(s): |  |
|  |  |
| Related Use |  |
| Case(s): | None |
|  |  |
|  | UC-8: Change Password |

1. *UC9- LOGOUT*

|  |  |
| --- | --- |
| System: | SAPORI ITALIANO |
|  |  |
| Identifier: | UC-9 |
|  |  |
| Author(s): | Team 7 |
|  |  |
| Version: | None |
|  |  |
| Name: | Logout |
|  |  |
| Pre- | 1.The user is registered user and has an account on Sapori Italiano.  2. User chooses profile option. |
| Condition(s): |  |
|  |  |
| Post- | The user is directed to Login page. |
| Condition(s): |  |
|  |  |
| Trigger: | User clicks on logout options. |
|  |  |
| Normal Flow: | 1. User is on the home page. |
|  | 2. User chooses profile option.  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 | None |
| Flow(s): |  |
|  |  |
| Related | **Primary**-Logged In User |
| Actor(s): |  |
|  |  |
| Related Use |  |
| Case(s): | Login |
|  |  |
|  | UC-9: Logout |

1. *UC10- VIEW ORDER LIST*

|  |  |  |
| --- | --- | --- |
| System: | SAPORI ITALIANO |  |
|  |  |  |
| Identifier: | UC-10 |  |
|  |  |  |
| Author(s): | Team 7 |  |
|  |  |  |
| Version: | None |  |
|  |  |  |
| Name: | View Order List |  |
|  |  |  |
| Pre- | 1. The user has an account on Sapori Italiano and logged in.  2. The user is on home page. |  |
| Condition(s): |  |  |
|  |  |  |
| Post- | The user is directed to View Order Page of the application. |  |
| Condition(s): |  |  |
|  |  |  |
| Trigger: | The user has clicked on the View Order button. |  |
|  |  |  |
| Normal Flow: | 1. User clicks on View Orders button. |  |
|  | 2. System will fetch customer orders from firebase.. |  |
|  | 3. System will display the list of orders to user. |  |
| Alternate Flow: | None |  |
|  |  |  |
| Flow(s): |  |  |
|  |  |
|  |  |  |
| Related | **Primary**-Administrator |  |
| Actor(s): | **Secondary**-None |  |
|  |  |
|  |  |  |
| Related Use | Change Item Price |  |
| Case(s): |  |  |
|  |  |  |
|  | UC-10: View Order List |  |

1. *UC11- CHANGE ITEM PRICE*

|  |  |  |
| --- | --- | --- |
| System: | SAPORI ITALIANO |  |
|  |  |  |
| Identifier: | UC-11 |  |
|  |  |  |
| Author(s): | Team 7 |  |
|  |  |  |
| Version: | None |  |
|  |  |  |
| Name: | Change Item Price |  |
|  |  |  |
| Pre- | 1. The user has an account on Sapori Italiano and logged in.  2. The user is on view items page. |  |
| Condition(s): |  |  |
|  |  |  |
| Post- | The user is directed to view items page of the application. |  |
| Condition(s): |  |  |
|  |  |  |
| Trigger: | The user has clicked on the Change Price button. |  |
|  |  |  |
| Normal Flow: | 1. User click on Change Price. |  |
|  | 2. User can increase or decrease price of item. |  |
|  | 3. Changes will be reflected in firebase. |  |
|  | 4. User will be redirected on View items page. |  |
|  | 5. Admin is redirected to the Change Price Page and can see the changes. |  |
| Alternate Flow: | None |  |
|  |  |  |
| Related | **Primary**-Administrator |  |
| Actor(s): | **Secondary**-None |  |
|  |  |
|  |  |  |
| Related Use | View Remaining Stock |  |
| Case(s): |  |  |
|  |  |  |
|  | UC-11 Change item price |  |

1. *UC12- VIEW REMANING STOCK*

|  |  |  |
| --- | --- | --- |
| System: | SAPORI ITALIANO |  |
|  |  |  |
| Identifier: | UC-12 |  |
|  |  |  |
| Author(s): | Team 7 |  |
|  |  |  |
| Version: | None |  |
|  |  |  |
| Name: | View Remaining Stock |  |
|  |  |  |
| Pre- | 1. The user has an account on Sapori Italiano and logged in.  2. The user has opened side menu from home page. |  |
| Condition(s): |  |  |
|  |  |  |
| Post- | The user is directed to View Remaining Stock Page of the application. |  |
| Condition(s): |  |  |
|  |  |  |
| Trigger: | The user has clicked on the View Remaining Stock button. |  |
|  |  |  |
| Normal Flow: | 1. User has clicked on remaining stock button from side menu. |  |
|  | 2. System will fetch items that are still available from firebase. |  |
|  | 3. User will be redirected to View remaining stock page and data will be displayed. |  |
| Alternate Flow: | None |  |
|  |  |  |
| Flow(s): |  |  |
|  |  |
|  |  |  |
| Related | **Primary**-Administrator |  |
| Actor(s): | **Secondary**-None |  |
|  |  |
|  |  |  |
| Related Use | View Delivered Orders |  |
| Case(s): |  |  |
|  |  |  |
|  | UC-12: View Remaining Stock |  |

1. *UC13 - RECEIVED ORDERS*

|  |  |  |
| --- | --- | --- |
| System: | SAPORI ITALIANO |  |
|  |  |  |
| Identifier: | UC-13 |  |
|  |  |  |
| Author(s): | Team 7 |  |
|  |  |  |
| Version: | None |  |
|  |  |  |
| Name: | Received Orders |  |
|  |  |  |
| Pre- | 1. The user has an account on Sapori Italiano and logged in.  2. The user has opened side menu from home page. |  |
| Condition(s): |  |  |
|  |  |  |
| Post- | The user is directed to Delivered Orders Page of the application. |  |
| Condition(s): |  |  |
|  |  |  |
| Trigger: | The user has clicked on the Received Orders button. |  |
|  |  |  |
| Normal Flow: | 1. User clicked on Received orders button from side menu. |  |
|  | 1. System will fetch orders that are requested by clients from firebase. |  |
|  | 3.User will be redirected to Recieved Orders page where the fetched data will be displayed. |  |
| Alternate Flow: | None |  |
|  |  |  |
| Flow(s): |  |  |
|  |  |
|  |  |  |
| Related | **Primary**-Administrator |  |
| Actor(s): | **Secondary**-None |  |
|  |  |
|  |  |  |
| Related Use | None |  |
| Case(s): |  |  |
|  |  |  |
|  | UC-13: Received Orders |  |

1. **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.

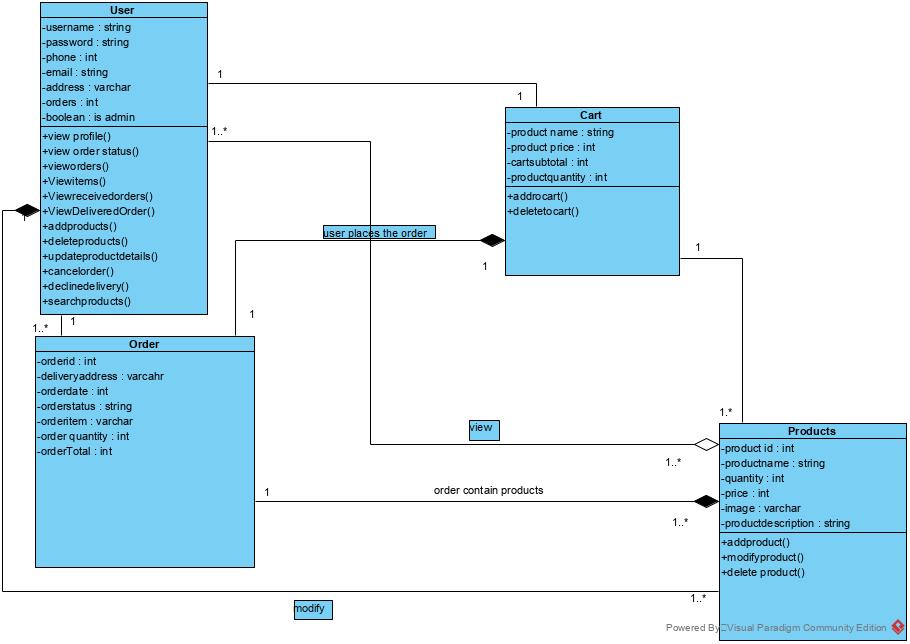
Some different form of association are discussed as below**:**

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

**For example:**

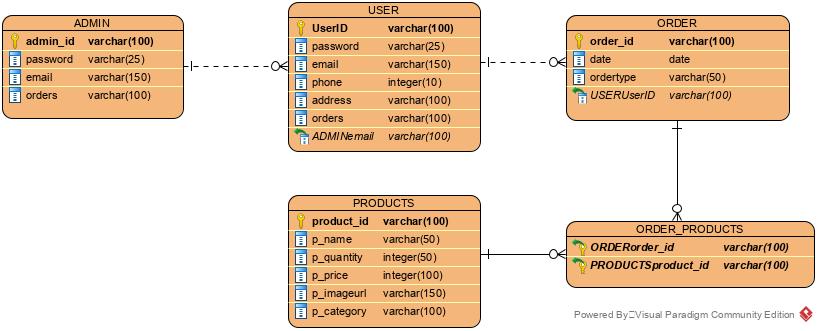
A person may own none, one or many cars, and a car must have one or many owners. And they are at the same level, none of them is a part of the other.

1. **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.
2. **Composition** : Illustrated with a filled diamond. The class on the diamond end owns the class on the other end.



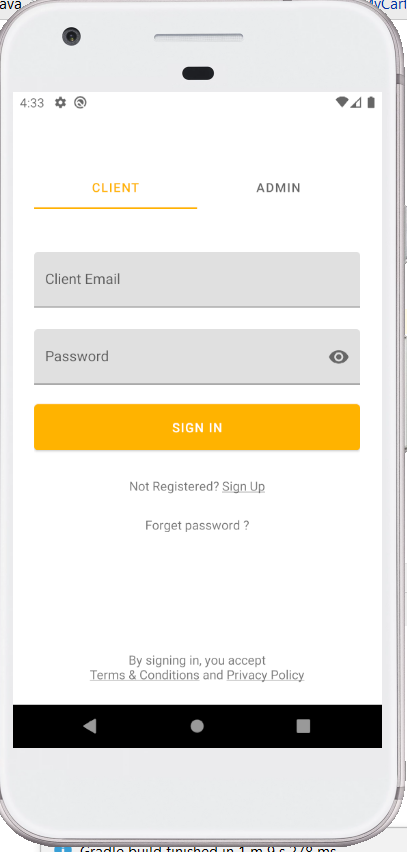
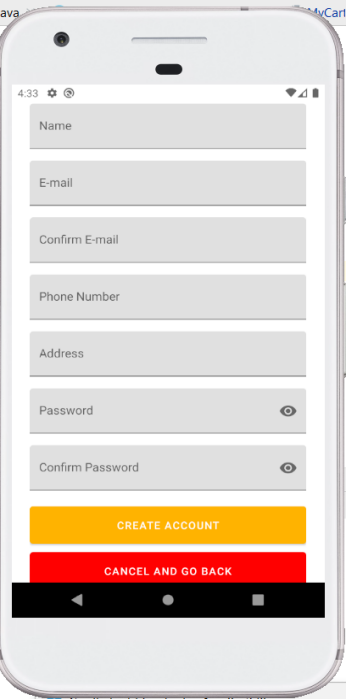
1. **ENTITY-RELATIONSHIP DIAGRAM(ERD)**

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****.

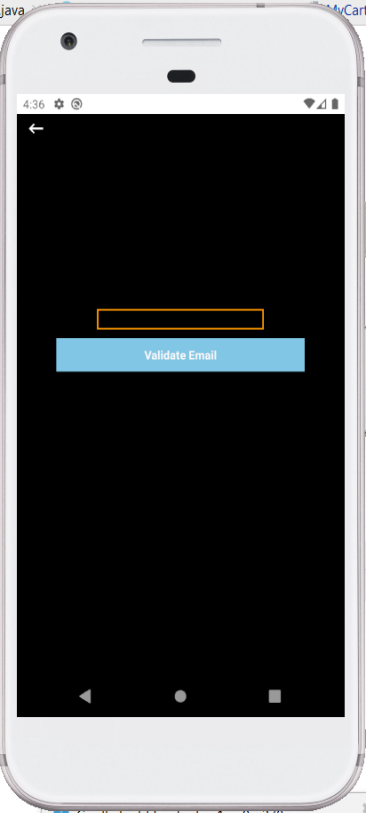
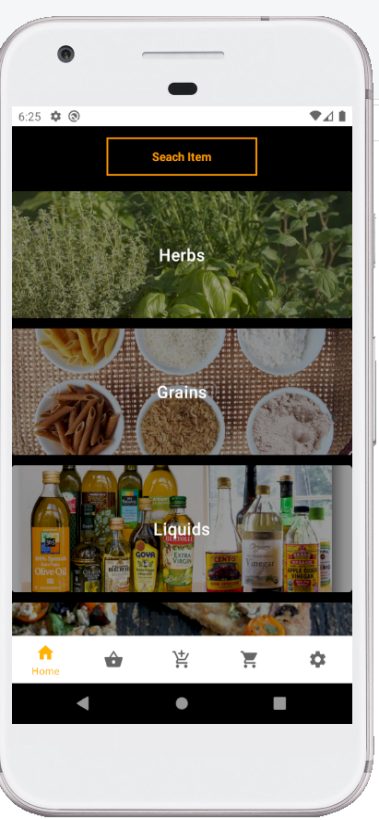
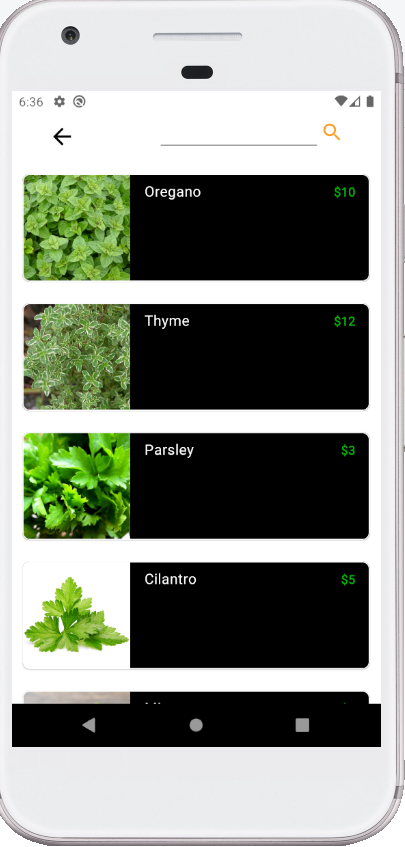
****

1. **SCREEN FLOW**

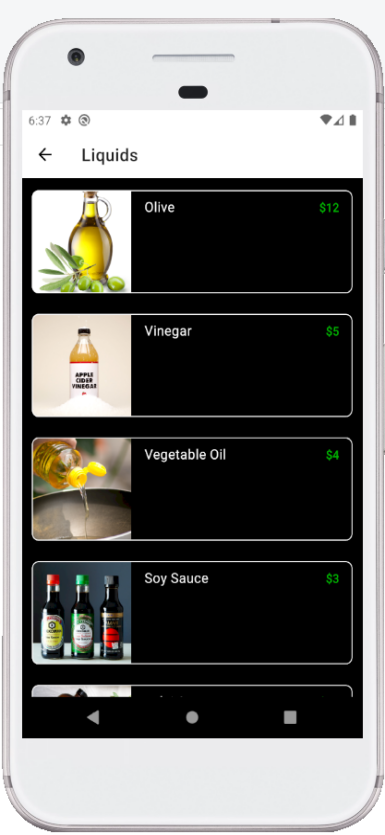
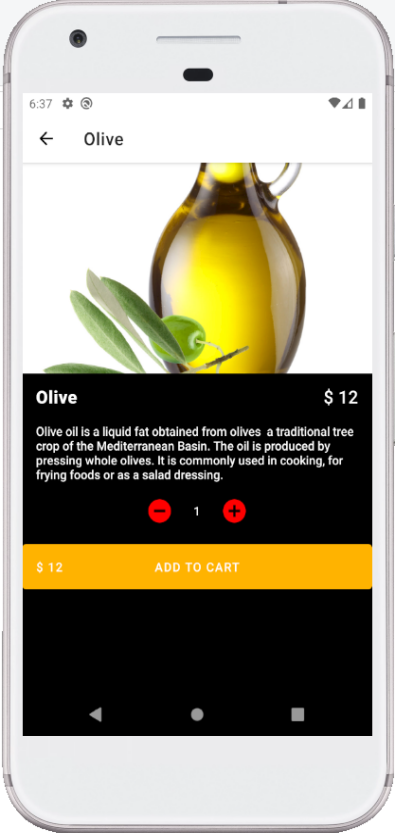
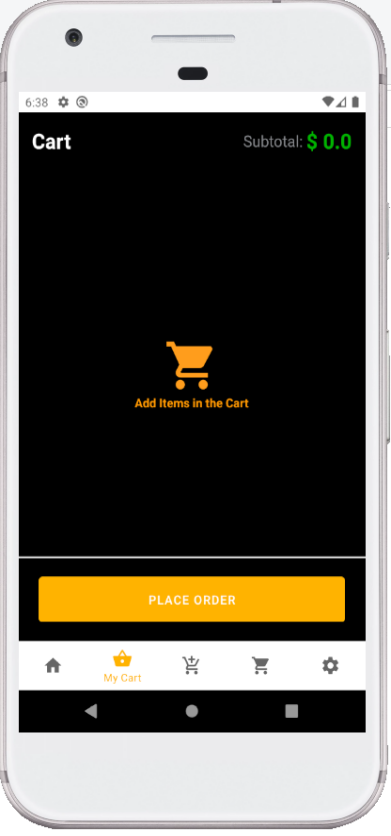
* *Client Side*

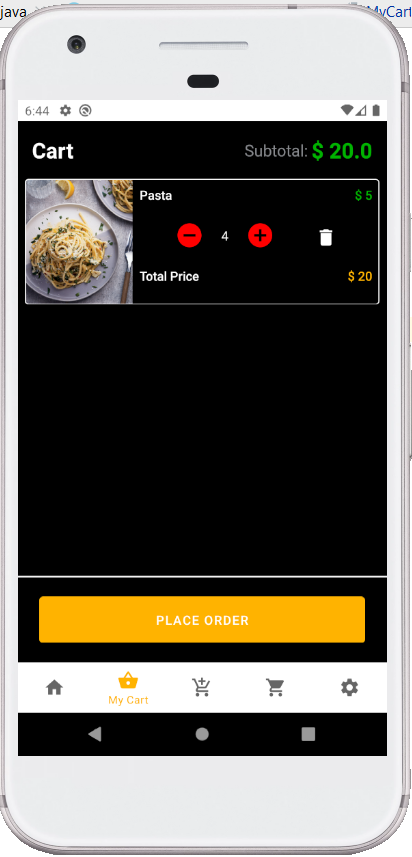
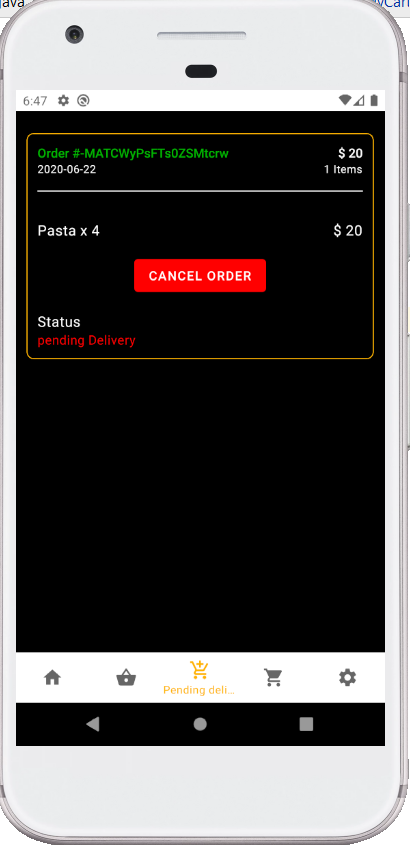
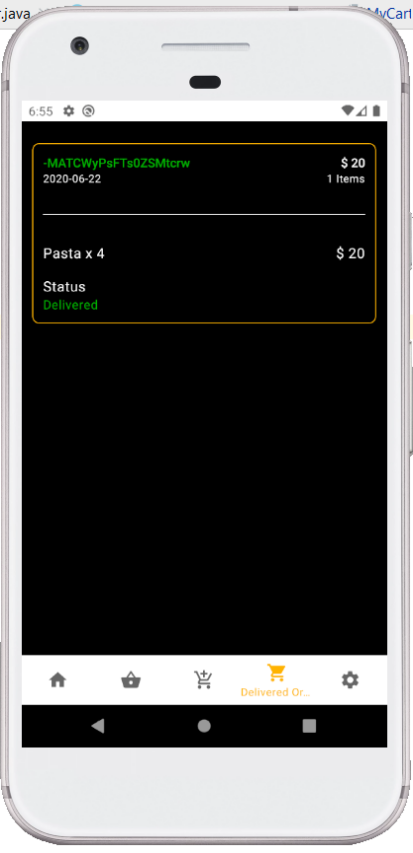
Splash Screen Client Login Sign Up

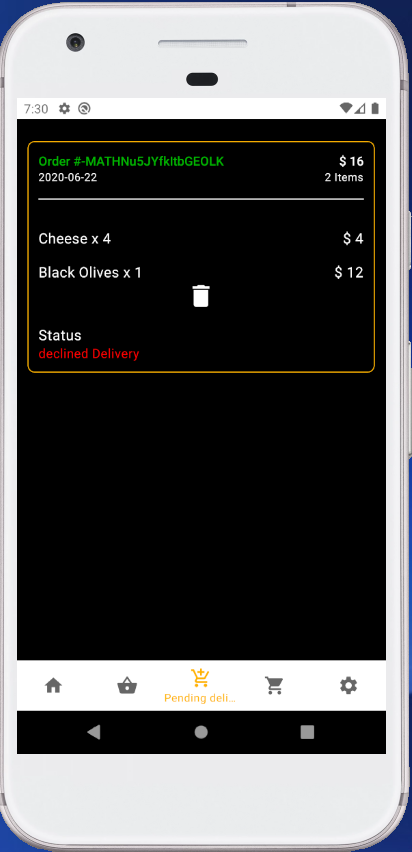
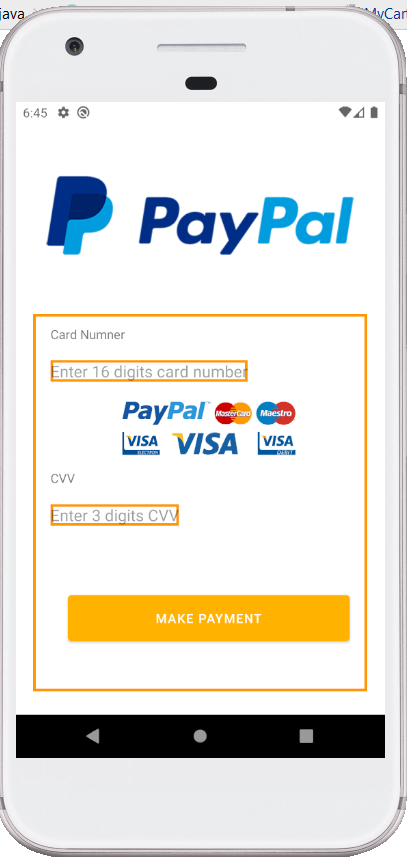
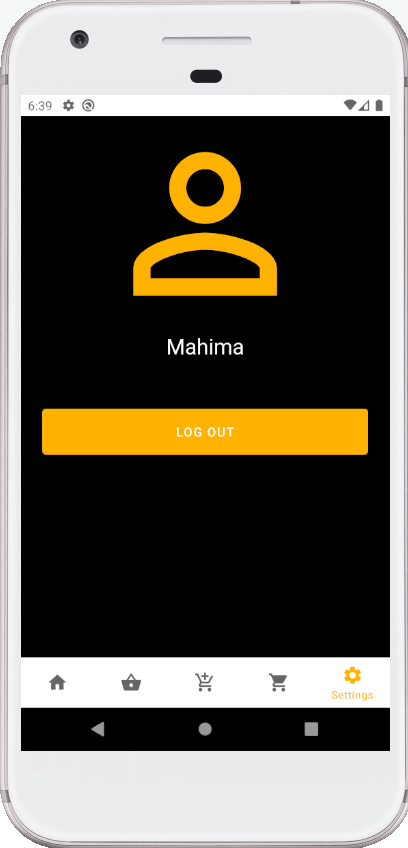
Forgot Password Home Login Search Button

Categories Display Item Empty Cart

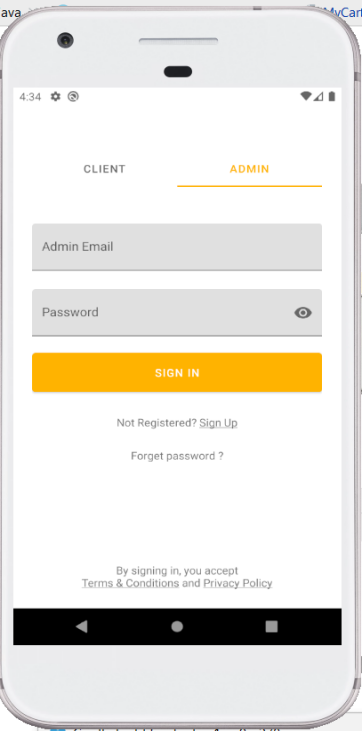
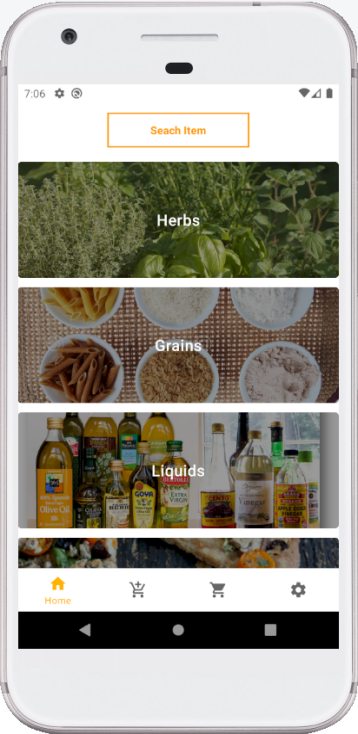
  

Item added to cart Pending Order Accepted from admin

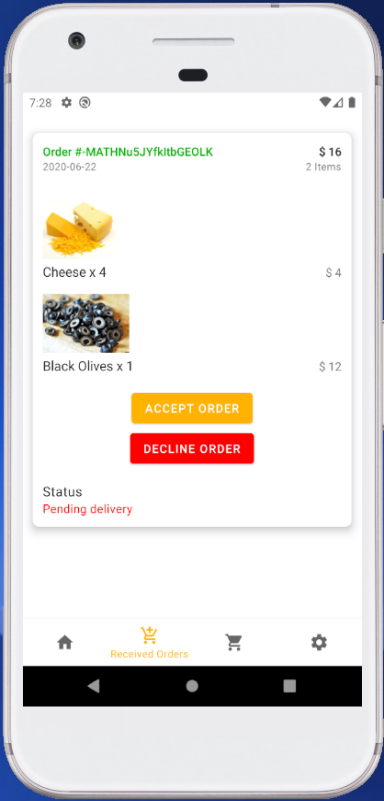
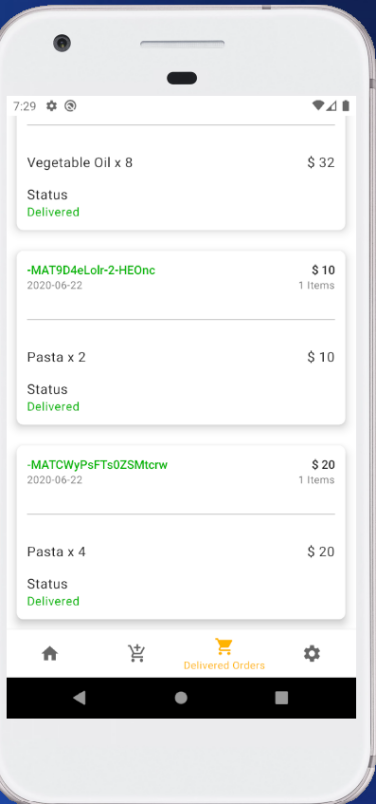
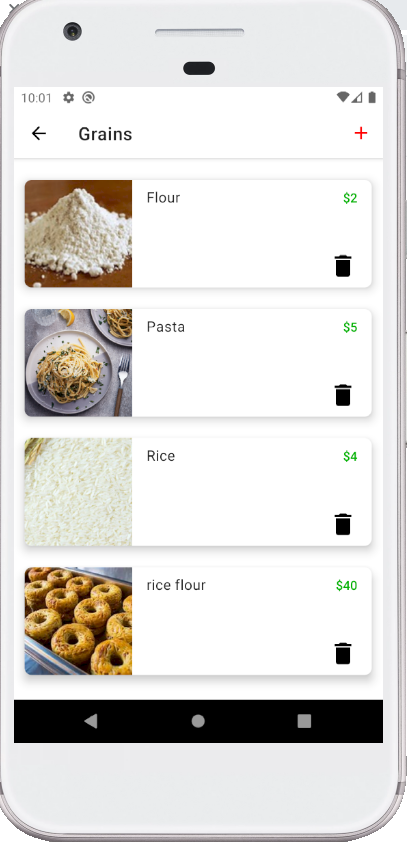
  

Declined from admin Payment Client Logout

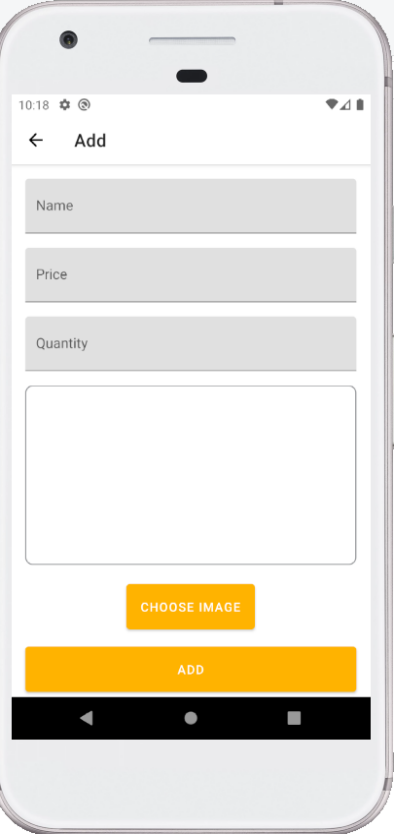
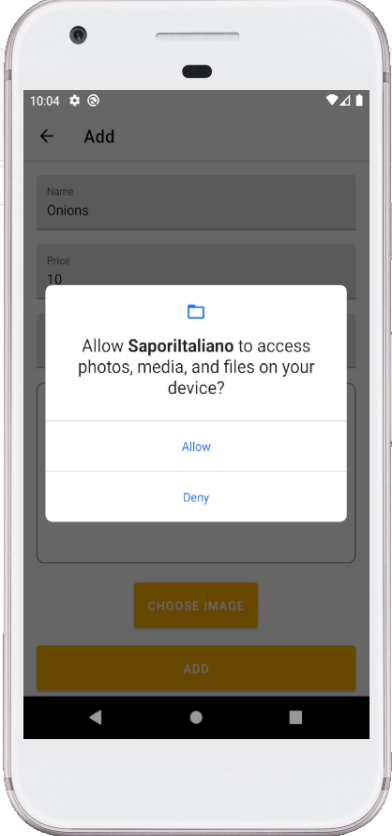
* *Admin Side*

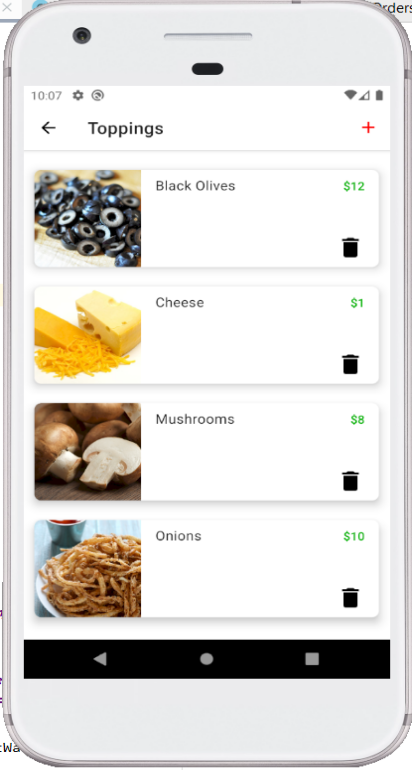
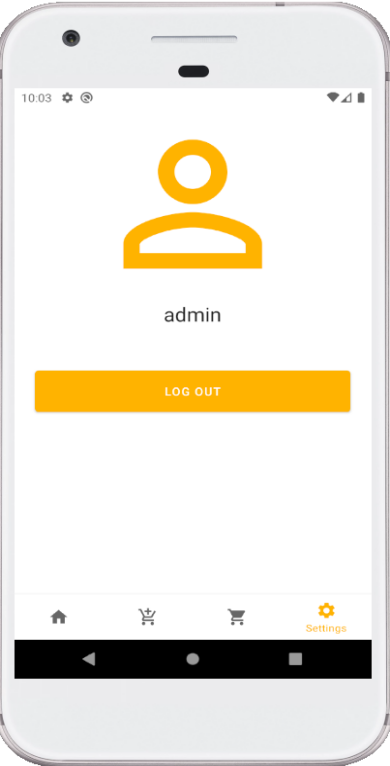
Splash Screen Admin Login Admin Home

Received Orders Delivered Orders Admin Category

Adding new Item Camera Permissions Selecting Image

New item added Admin Log out