**El Patron Project** **Design Document**

**Introduction**

**about the project**

The El Patron Project is a web application that is designed to provide a platform for managing orders deliveries and pickups for a successful drinks bar named El Patron held in Tira City.

**Scope**

The web application will provide the end user the experience to order drinks and other products supplied by the business simply and fast, and the web application will help the managers to keep track of the orders they need to supply.

It will include the user interface and client-side functionality using a JSON Server REST API to mock all server side related.

In addition, as a second phase of the project’s development, the web application will include the option for the customer to add/remove ingredients as he desires.

and will include interference with a third party in order to complete the payment process and approved order will be sent to the client and business with the estimated delivery time.

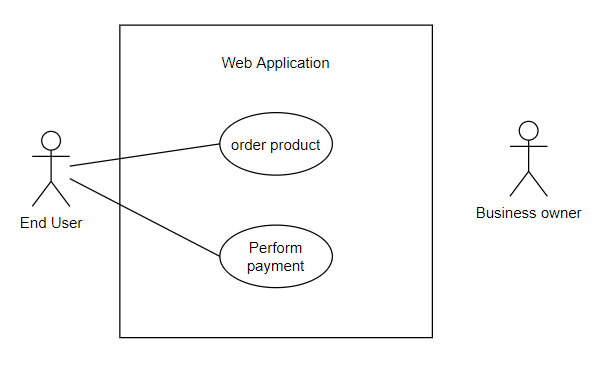
**Overview**

The purpose of this document is to describe the design specifications for the web application.

This document includes introduction, overall description of the software, architecture and detailed design, data handling and tests.

**Design**

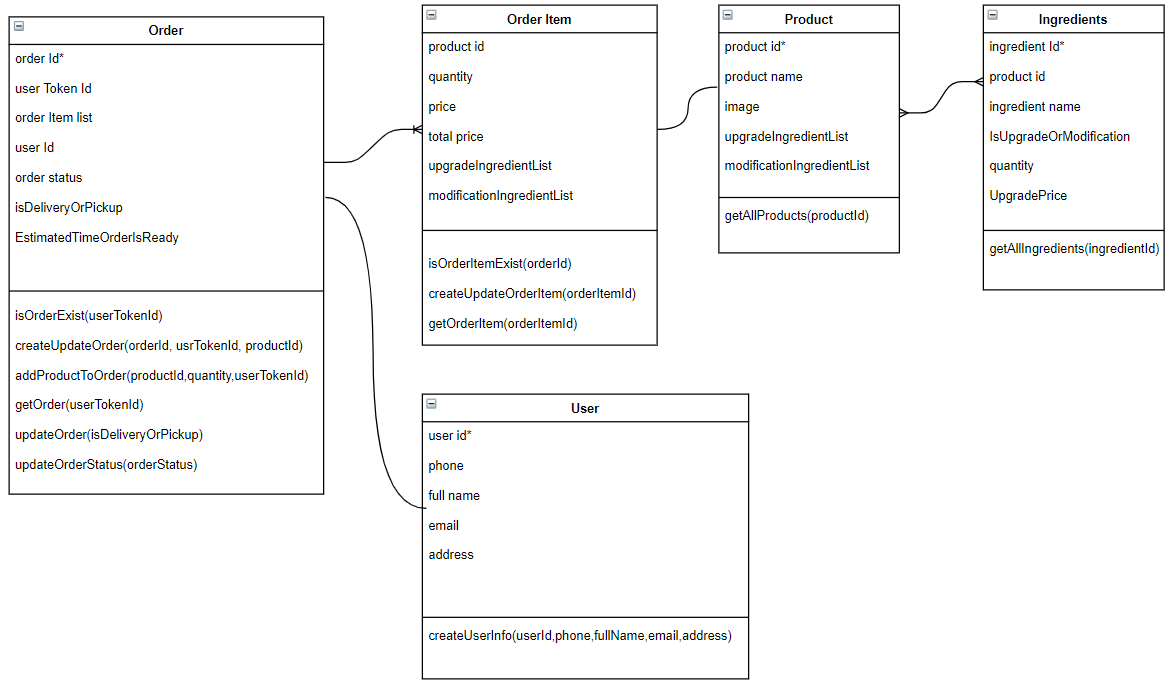
1. **Architecture**
   1. **Use Cases**

****

**Note:** Edit/Delete product is out of scope for now

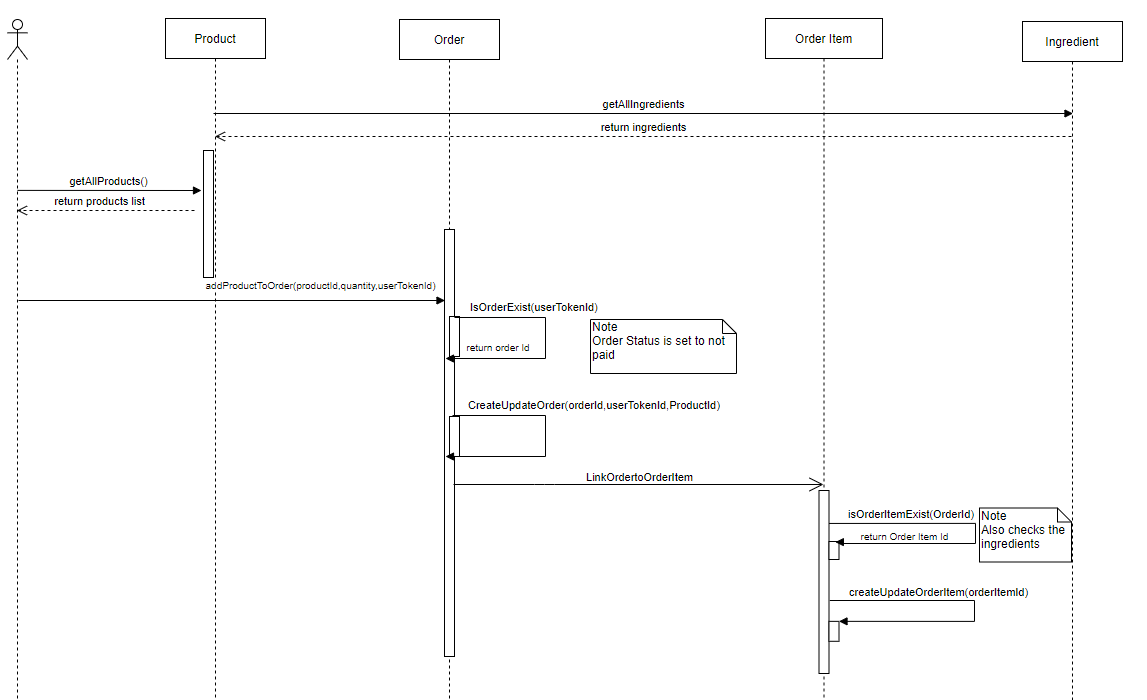
**Note:** Business owner receives a confirmation email after payment is conducted(for second phase)

* 1. **Components and Functions**

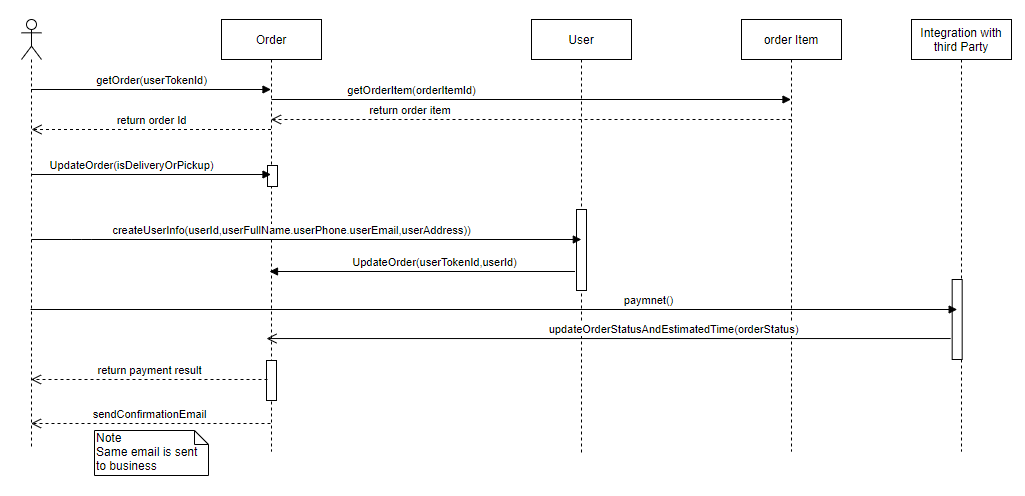


* 1. **Sequences**

Order Product (Use Case)



Perform Payment (Use Case)

****

1. **Data Handling**

Data handling and all operations required in this project will be implemented using Mock Server instead of implementing server-side and database management.

The mock server has a Json file that is used as the database. all data is forwarded from and to the Json file using REST API.

The system components mentioned above are used for describing the data in the Json file Respectively:

* Order object - contains the relevant data of the ordered products by a specific user, the order can be in paid/not paid/approved statuses
  + Order Id, User Id, products list, total price and order status
* User object - contains the information of the user to be used to send the order confirmation after the order is signed.
  + User Id, User first and last name, email and address
* Products Object - contains each product information that will be added to the Order object after the product is selected by the user
  + Product Id, product name price and image.

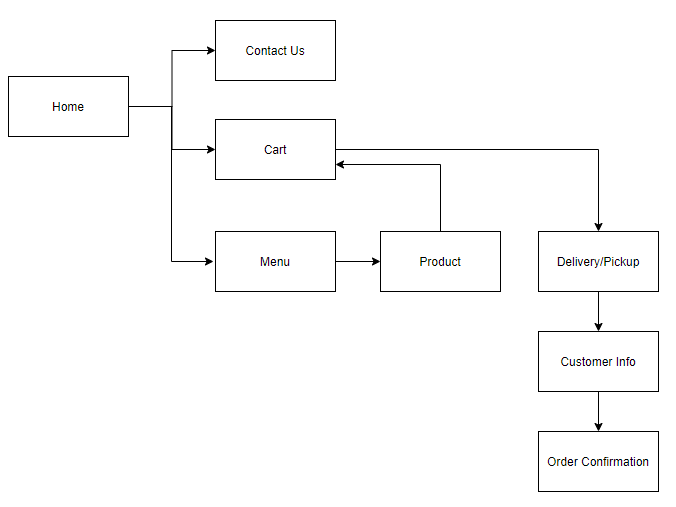
1. **Integrations**

For Payment conducting using the web application, an integration with a third party such as PayPal is needed.

As of this moment the process of payment will be out of the project’s scope as its implementation is depending on the work progress and time frames

1. **User Interface**

Tree of the web pages:

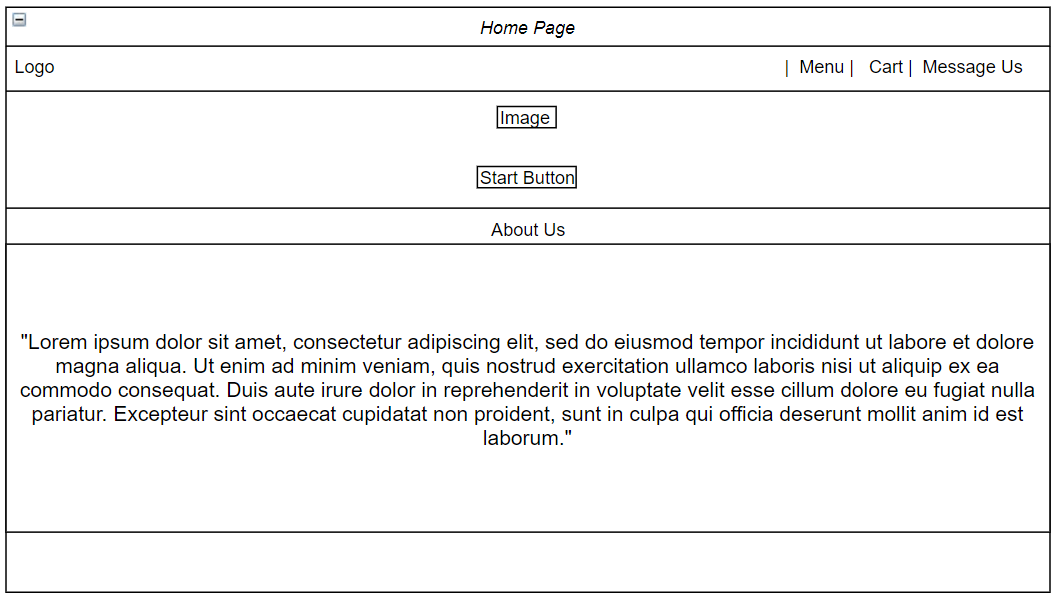


* 1. **Home**

This is the main page of the web, it has the information and logo of the business.

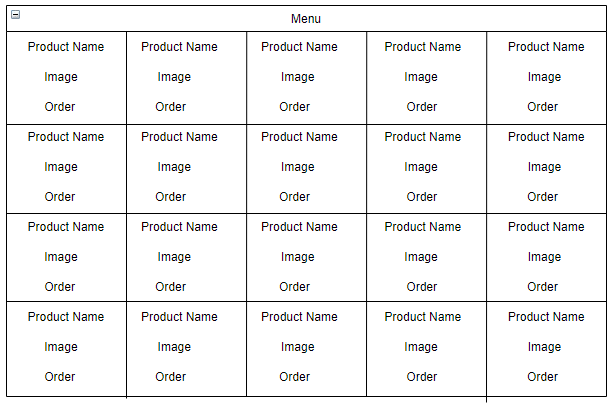
It navigates the user through the different screens after the required button is clicked.

In the home page the user has access to Menu, Cart and Message Us.

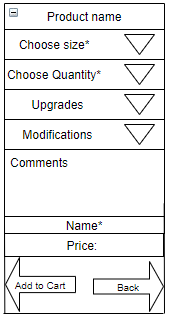
****

* 1. **Menu**

The Menu page lists all the products. The user can scroll up and down the page to see all the products. Each box contains one product with the product name and image to be presented to the user and an order button.

****

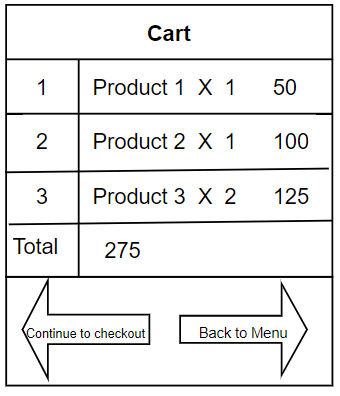
* 1. **Product**

****

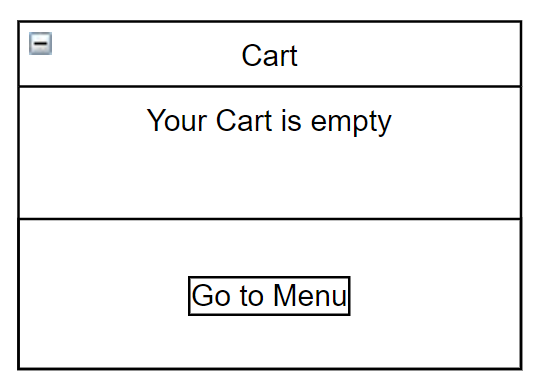
This page is filled by the user by choosing from drop down lists the size and quantity and personally customize the product as desired.

* 1. **Cart**

This page is the summary of the order, the cart presents the products list, quantity and prices. The user either approves the order details and continues to checkout, or decides to go back to the menu.

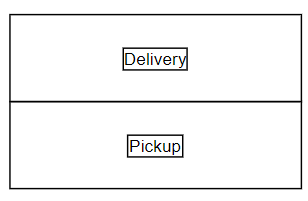
****

If no products were added to the cart, the cart page will notify the following:

****

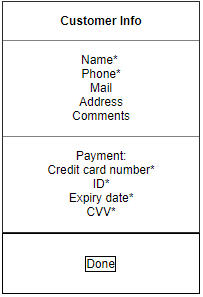
* 1. **Delivery/Pickup**

This page is for the user to choose one one of the options before continuing to payment

****

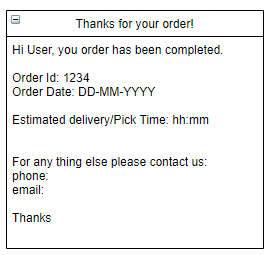
* 1. **Customer Info**

This page is filled by the user with his personal information and payment information in order to finalize the order.

****

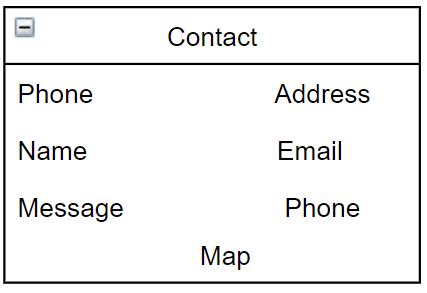
* 1. **Order Confirmation**

The order confirmation page contains the order details and is presented to the user after the payment is done.

****

* 1. **Contact Us**

This page is filled by the user with messages and requests that are sent to the business mail.

****

1. **Tests and Monitor**

UI Tests - check all components throughout the entire project and verify the interactions are working correctly, UI test includes:

Links functionality, Buttons functionality, alerts, submission functionality and data is saved to the Json file correctly, etc..

Sanity Tests - check if the application flow is working properly, check the application functionality, verify the use cases described above are implemented correctly from start to finish, including:

* Enter the home page and view the menu, the cart and the message us pages
* From the menu add a new product to the cart (new line in the cart)
* Add the same product to the cart (quantity change)
* Add the same product in a different size (new line in the cart)
* Verify cart is updated accordingly
* Verify total price is updated
* Verify Delivery/Pickup selection
* Checkout and customer info filled
* Update database with the new order and user

1. **Logs**

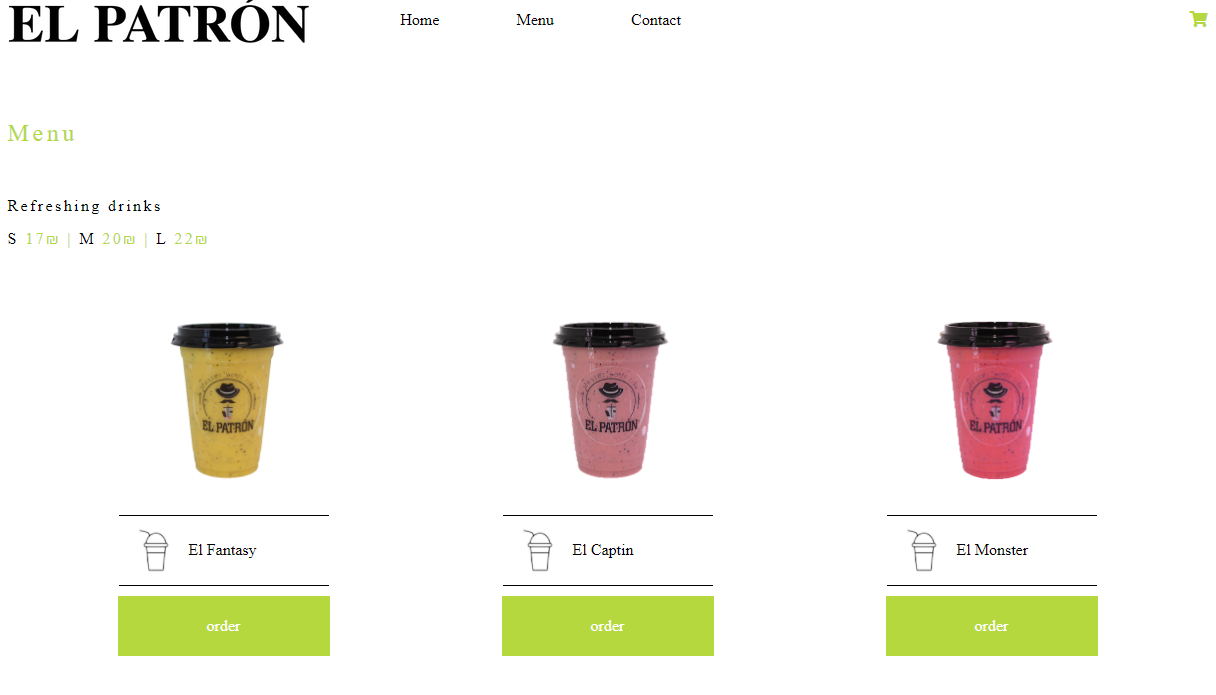
Logs will be added in order to document the success or failure of all the methods throughout all the processes in the project.

There are two options that were taken into consideration for logs implementation, either to write the data into a file or to use an external directory. The implementation will depend on the project progress and time constraints.

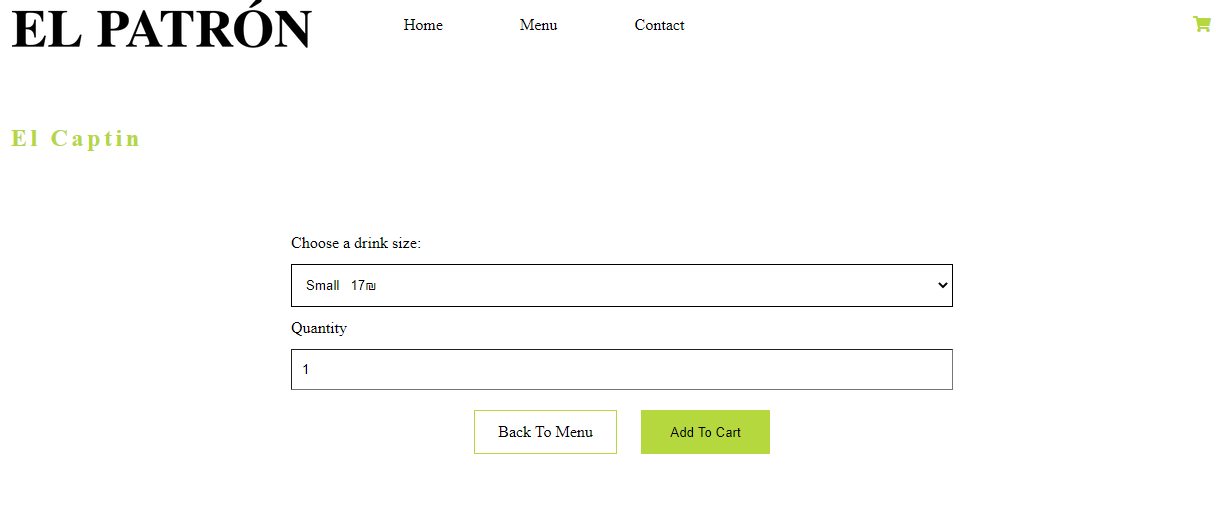
Appendix A

**Sanity tests execution and results**

Enter the home page and then enter the menu page, and then Press the ‘order’ button

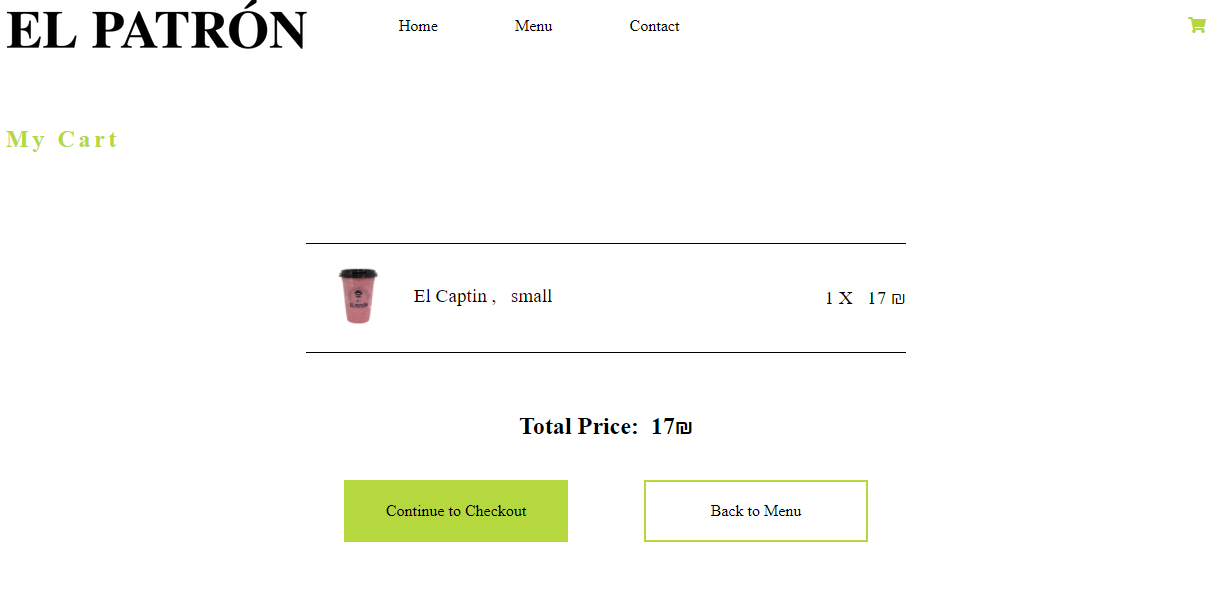


The product page is opened to select the size and quantity

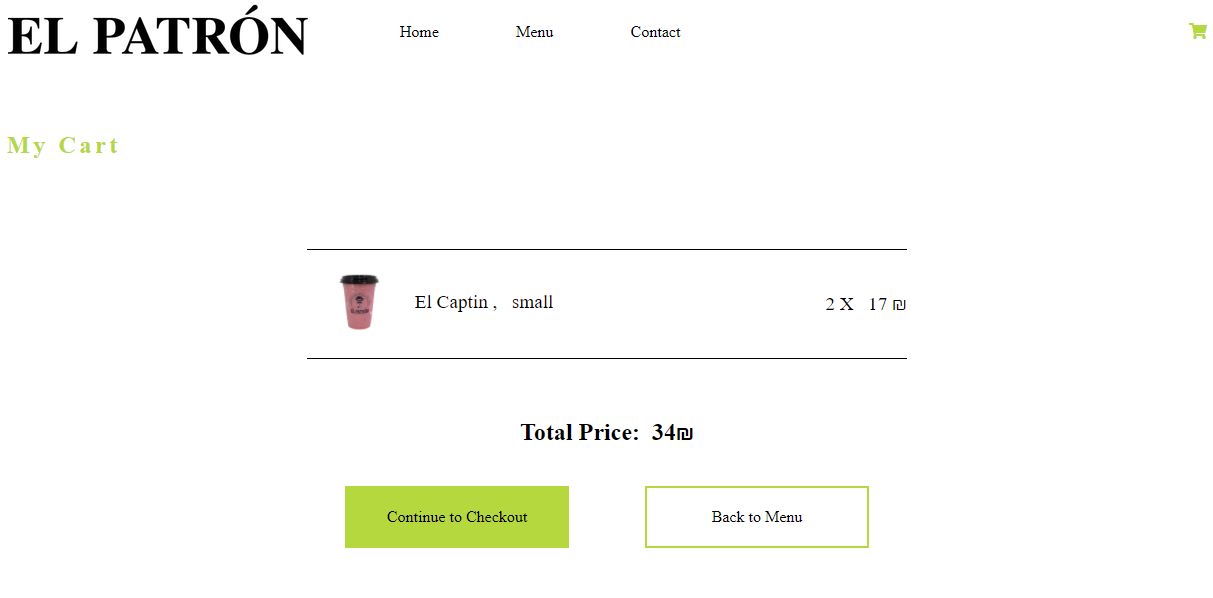


Press to add the product to the cart, the cart is updated with the product and the total price is updated.

A new userTokenId is issued for the customer and an order is added to the database with the orderId, userTokenId, orderItem and orderStatus is set to not paid.

****

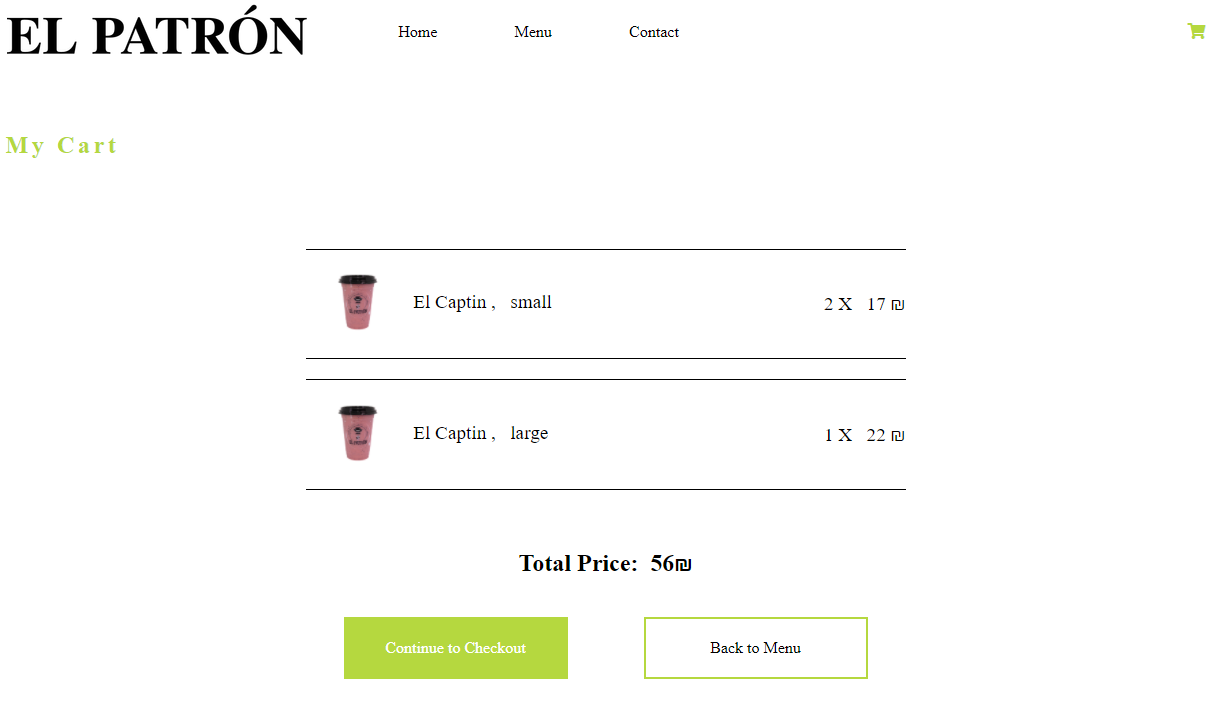
Pressing ‘Back to Menu’ to add the same product

****

The quantity is updated to 2 and the total price updated accordingly,

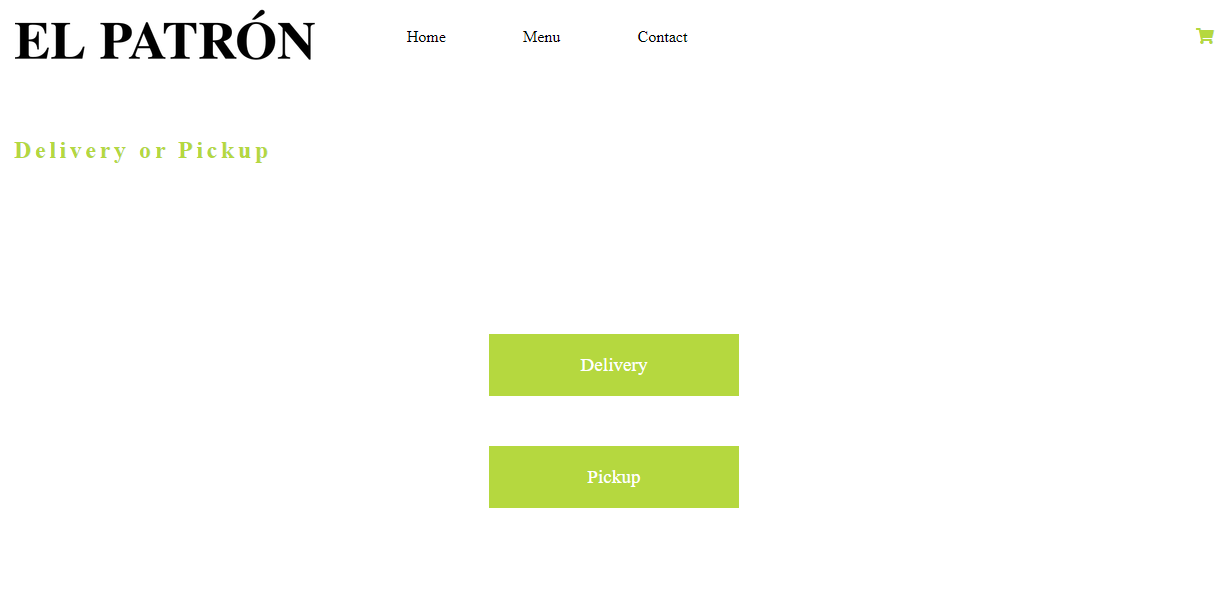
Press Back to Menu to add the same product in different size.

In the database orderItem in the order is updated with the quantity and prices.

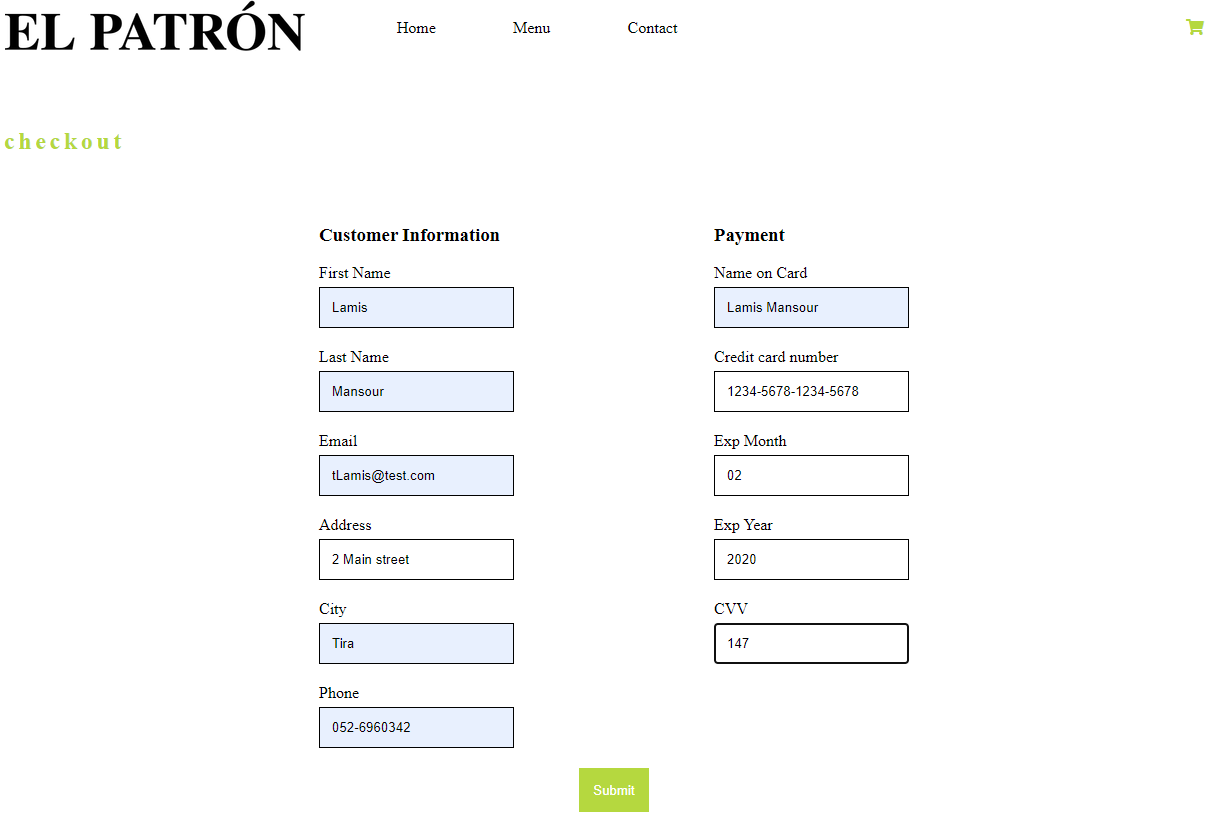
****

Continue to Checkout will open the Delivery or Pickup.

Then the Checkout page is opened

****

All fields are required to be filled by the customer prior to submitting.



After submitting a valid form, the confirmation page is opened.



A new user is added to the database with the customer information and a unique userId is issued.

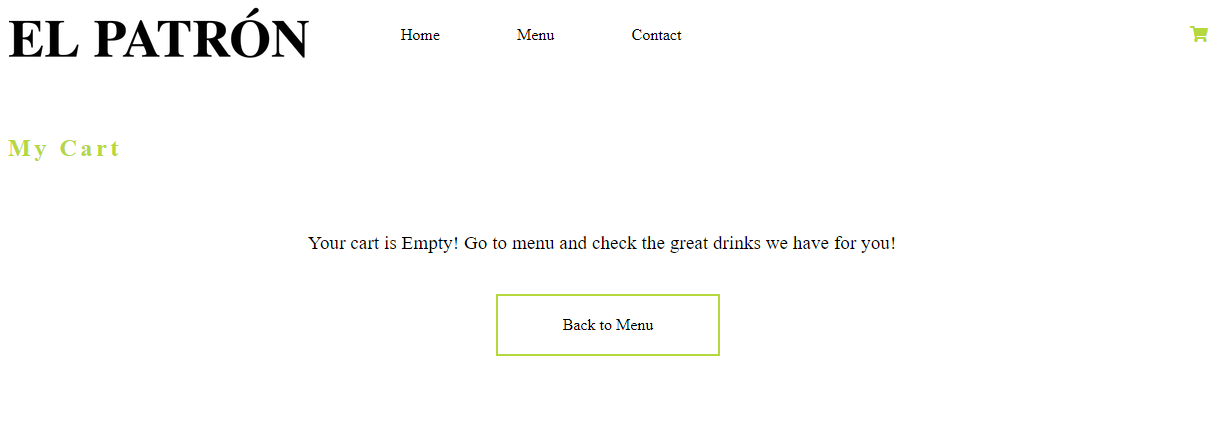
\*If the customer is already found in the database (check if the phone is already in the database) the same userId is used

The database is updated with the userId and status is changed to approved.

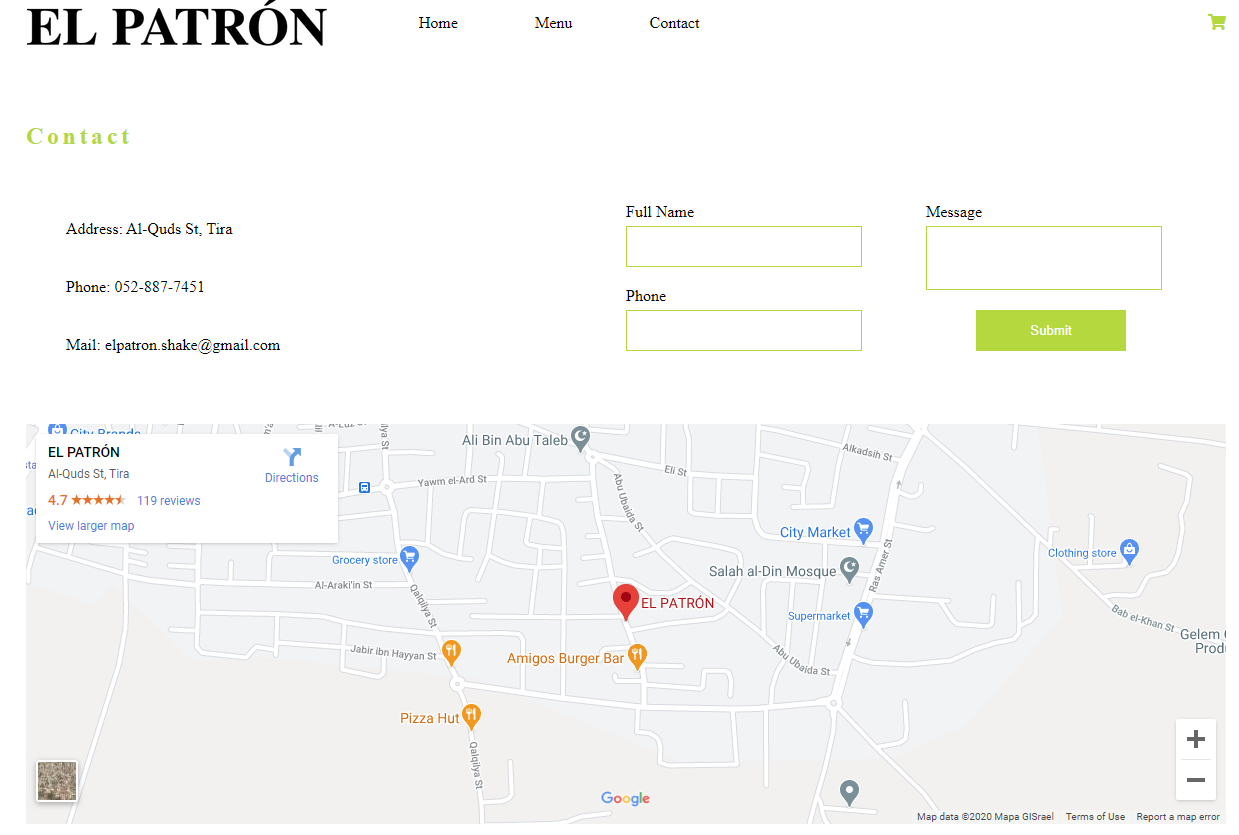
Order:



1. **Open cart page before the customer added any product, the empty cart page is opened**

****

1. **Open the contact page**

****

**Test result**

Tests passed successfully