جامعة بيرزيت

BIRZEIT UNIVERSITY

Palestine IntelliCart

Prepared By:

Al-Ayham Maree

Huthayfa Mutan

Sara Ammar

Supervised By:

Dr. Ashraf Alrimawi

Graduation Project Report submitted to the Department of Electrical and Computer Engineering in partial fulfillment of the requirements for the degree of B.Sc. in Computer Engineering

Birzeit

July, 2024

Abstract

Retail centers witness large crowds of customers, and this crisis increases in large stores or those located in city centers, especially on occasions such as holidays, weekends, and discounts. These crises in stores eliminate the joy of shopping and exhaust customers due to long waiting lines at the bill counter, in addition to several issues, such as the fact that consumers should be able to examine the bill at any time, view product details upon purchase, and be notified about discounts. This project presents "Palestine IntelliCart ," a smart shopping cart, to redefine the traditional shopping experience and improve the field of retail technology by integrating modern technologies. The primary goals of "Palestine IntelliCart " are to simplify the payment process, reduce waiting times, enhance user convenience, and ensure accurate tracking of the products in the cart. The main components of this system are a Raspberry Pi that acts as a central control unit, along with a camera module, a high-performance barcode scanner for quick product recognition, a touch screen for customer interaction, and a weight sensor for extra contextual information. "Palestine IntelliCart " is anticipated to make a paradigm shift in the way of interactions between consumers and physical retailer stores.

المستخلص

تشهد مراكز البيع بالتجزئة إقبالاً كبيراً من الزبائن، ويزداد هذا الإقبال في المتاجر الكبيرة كتلك الموجودة في مراكز المدن وخاصة في أوقات مناسبات الأعياد وعطل نهاية الأسبوع وأوقات العروض و الخصومات. هذا الازدحام في المتاجر يقضي على متعة التسوق ويرهق الزبائن، بسبب طوابير الانتظار الطويلة عند نقاط الدفع، بالإضافة إلى عدة مسائل مثل ضرورة تمكين المستهلك من معرفة قيمة الفاتورة في أي وقت، والاطلاع على تفاصيل المنتج عند الشراء، وأن يكون على علم بالخصومات. و يقدم هذا المشروع "Palestine IntelliCart" عربة تسوق ذكية، لإعادة تعريف تجربة التسوق التقليدية، وتطوير التكنولوجيا في مجال البيع بالتجزئة من خلال دمج التقنيات الحديثة. تتمثل الأهداف الأساسية لـ "Palestine IntelliCart" في تيسير عملية الدفع، وتقليل أوقات الانتظار، وتعزيز راحة المستخدم، وضمان التتبع الدقيق للمنتجات في عربة التسوق. المكونات الرئيسية لهذا النظام هي "Raspberry Pi" الذي يعمل كوحدة تحكم مركزية، إلى جانب وحدة الكاميرا، وماسح الباركود عالي الأداء للتعرف السريع على المنتج، وشاشة تعمل باللمس للتفاعل مع العملاء، ومستشعر الوزن للحصول على معلومات إضافية عن محتويات العربة. ونتوقع أن تُحدث "Palestine IntelliCart" نقلة نوعية في طريقة التفاعل بين المستهلكين ومتاجر التجزئة الفعلية.

Table of Contents

List of Figures

List of Abbreviations

AI Artificial Intelligence

CV Computer Vision

IOT Internet Of Things

RFID Radio Frequency Identification

MOT Multi Object Tracking

RAM Random Access Memory

GPIO General Purpose Input/Output

CSI Camera Serial Interface

DSI Display Serial Interface

USB Universal Serial Bus

Wi-Fi Wireless Fidelity

FOV Field Of Veiw

QR Quick Response

YOLO You Only Look Once

1D One Dimension

2D Two Dimensions

3D Three Dimensions

RESTful Representational State Transfer

API Application Programming Interface

HTTP Hypertext Transfer Protocol

ID Identifier

ERD Entity-Relationship Diagram

SQL Structured Query Language

IOU Intersection over Union

JWT JSON Web Token

DOM Document Object Model

UI User Interface

Chapter 1

Introduction and Motivation

Contents

1.1 Motivation

Shopping is a delightful experience for many, as the joy of shopping is discovering new and unique goods. However, this experience may be spoiled by some drawbacks that turn a delightful experience into an unbearable, challenging task from which customers must regularly suffer. The motivation behind Palestine IntelliCart is to address these drawbacks by utilizing Artificial Intelligence (AI) such as Computer Vision (CV) technologies to achieve new and efficient shopping experiences without any negative drawbacks.

Many solutions have emerged to address the issue of long queues in retail establishments. 'Amazon Go' offers a cashier-less shopping experience with identity verification at the entrance, computer vision, and sensor-based item identification, eliminating the need for traditional payment procedures [1]. 'Tao Café' adopted a comparable approach, utilizing facial recognition technology and artificial intelligence for seamless identification and payment processing [2]. These solutions mark a significant shift in retail, prioritizing efficiency and convenience for customers. Other developments, such as the exploration of Radio Frequency Identification (RFID) technology, further contribute to the ongoing evolution of cashierless shopping experiences. Other projects propose shopping carts equipped with barcode readers or cameras trained to detect items using appearance features or barcodes since most retail items have barcodes as identifiers.

Most of the solutions presented are inapplicable and unreliable, alter the regular shopping experience, or cause serious security threats to the consumer and retail grocery. 'Amazon Go' and 'Tao Cafe' have several issues, such as the need for an enormous initial financial investment for implementation. In addition to the difficulty of applying it

to all products due to the various appearances of the items, there may be obstacles facing the cameras, causing a lack of visibility, and the use of digital fingerprints raises customer concerns about their privacy and security. Also, integrating RFID technology into the current infrastructure is complex and costly. It is difficult to place it on all the products, and it may be incompatible with the material of some products. Projects suggesting cameras face multiple issues, especially when there is a lack of light or training-biased data, causing biased and less efficient modules in real-world applications and some real-time performance problems. As for barcode-related projects, most of them cause security threats since they rely on trusting the customer or will require complex procedures, thus resulting in negative results instead of improving the shopping experience.

1.2 Problem Statement

Despite the current traditional shopping methods, several impediments impede their integration with development and modern expectations. Long and crowded waiting lines that exhaust customers and shortcomings in checking the bill or viewing product details in real-time while engaging in the shopping process constitute major obstacles that add a layer of complexity to the overall experience. It's necessary to develop a technological solution that addresses these issues without compromising the essence of traditional shopping, modifying the product form or nature, and ensuring easy integration with existing infrastructure environments.

1.3 Methodology

The 'Palestine IntelliCart ' project proposes a fully integrated system consisting of a Raspberry Pi as a control center for all other components and a processing brain for algorithms. A Raspberry Pi camera module is used for CV to track items, a barcode scanner for identifying items that enter the cart, a weight sensor to support the camera with additional information, and a Raspberry Pi touchscreen to provide an intuitive and user-friendly interface. 'Palestine IntelliCart 's main objectives are to create an innovative smart shopping cart that implements fast and secure payment operations to significantly reduce waiting time at purchase points. In addition to ensuring accurate tracking of products entering and leaving the cart in real-time by integrating barcode scanning techniques and computer vision in a way that gives priority to the customer's convenience, it aims to facilitate consumer interaction with product information and continuous follow-up of the purchase list and the total invoice value through the shopping cart interface. In addition to implementing technological solutions that do not require any modifications or additions to the nature of the products, the cart system can easily be integrated with the retail infrastructure, allowing store owners to adopt it without significant modifications.

Chapter 2

Background and Related Work

Contents

2.1 Full Market Renovations

Enhancing the shopping experience has been a goal for many researchers and big companies. Amazon created 'Amazon Go' in 2016, and Alibaba created 'Tao Cafe' in China in 2018. Both projects worked on creating a cashier-less experience for their customers. 'Amazon Go' employed multiple cameras and sensors utilizing CV and advanced algorithms. It verifies the identity of customers entering the store through their phones, then tracks the customer's process through the whole shopping process using CV and shelf sensing simultaneously to track activities, detect selected items from the shelf, and then calculate the customer bill. Upon leaving the store, the purchase process concludes seamlessly without traditional payment procedures, as the system automatically deducts the corresponding amount from the customer's account. Load cells can sense events such as removing or placing items and recognize the items involved. But mistakes may occur if customers put items in similar places or replace items of similar weights [1]. 'Tao Cafe uses a similar technique by detecting the customer's facial identity using cameras around the shop and then using CV to calculate your bill. Once the items are selected, the system processes the payment based on the recognized items. All the projects similar to the previous two contain unsurpassable issues, including an enormous initial investment to install cameras around the place and research for techniques and algorithms such as CV techniques appropriate for the market's needs. 'Amazon Go' required full CV learning for each product to allow tracking, thus requiring less adaptability with the changing products in the market [2].

2.2 Radio Frequency Identification RFID

Services combined with RFID tags for tracking and identifying products are expensive and require a significant initial investment because most products already have barcodes, so adding RFID tags to these products will need a new manufacturing line in each factory or a new labor force to add tags to each item. If RFID tags are found on products, the RFID systems may not be compatible with different vendors, hindering the ability to create a cohesive and integrated marketing system. RFID tags also have limited range capabilities, and their tracking signal may be affected by the nature of products such as metals and liquids; thus, it won't apply to most used items since a lot of items either contain liquids or have a high level of humidity [3], [4], [5].

2.3 Cameras on Carts

Some projects use cameras on carts to read the objects' movements or identify items added to the cart. [6] uses camera computer vision to detect if an empty hand or a hand is holding an item inside the cart and tracks these operations. [7] uses a camera to read barcodes and identify the items in the cart by the barcode. Other projects use cameras to identify the items by training them to detect previously known items. These techniques contain multiple problems since any lack of light or ambiguity will cause errors in identifying objects or their barcodes, and any biased training data will result in catastrophic results. For example, if the trained data for barcode techniques had 1d barcode type dominating on 2d, it would result in the module ignoring 2d or misreading it.

2.4 Barcode Scanners

Other projects enhanced shopping carts with a barcode scanner so that the customer can scan the code of his purchases himself, but they neglected to monitor shopping movements, such as returning the product, switching between one product and another, or placing products in a cart without scanning [8], [9].

Chapter 3

System Architecture

Contents

3.1 Hardware Overview

‘Palestine IntelliCart ’ proposes a fully integrated system Fig. 3.1, which consists of a Raspberry Pi 4 as a center of control for all other components and a processing brain for algorithms; a Raspberry Pi camera module 3 wide is used for computer vision to track items inside the camera frame; an EVAWGIB embedded mini barcode reader module for identifying items that enter the cart; a half-bridge strain gauge load cell weight sensor to support the camera with additional information; and a Raspberry Pi touch display 7” to provide an intuitive and user-friendly interface.

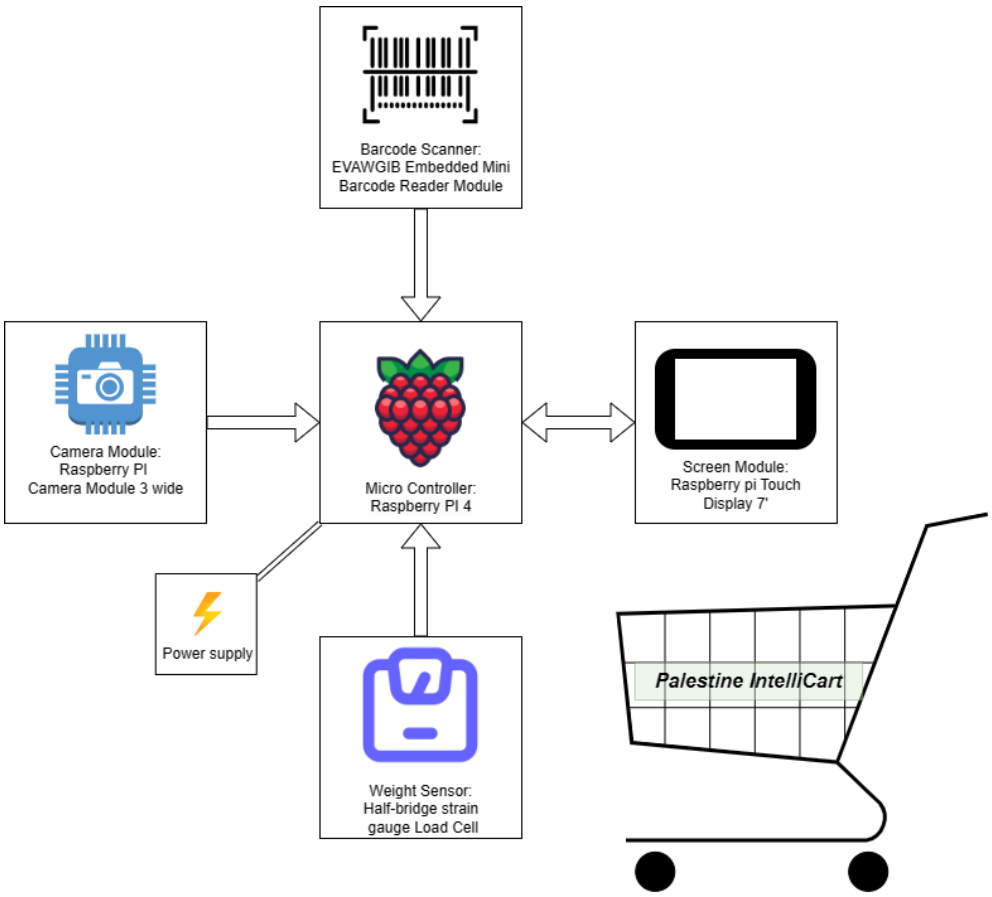


Figure 3.1: Block diagram

3.1.1 Raspberry pi 4

The Palestine IntelliCart project uses the Raspberry Pi 4 as the central control unit, integrating hardware components and facilitating intelligent decision-making. Its quad-core ARM Cortex-A72 processor and Random Access Memory (RAM) enable real-time data processing, computer vision algorithms, and user interface interactions. The General Purpose Input/Output (GPIO) pins enable connectivity with peripheral devices like weight sensors. The MIPI Camera Serial Interface (CSI) port enables connecting with a camera module. The MIPI Display Serial Interface (DSI) port enables connecting with a touch screen. The Universal Serial Bus (USB) port enables connecting with a barcode scanner. Built-in networking capabilities, including Wireless Fidelity (Wi-Fi)

and Ethernet, enable communication with external systems like cloud services or back-end servers, enhancing the functionality and scalability of the smart shopping cart system. This integration opens up possibilities for cloud-based data storage, remote monitoring, and analytics [10]. Figure 3.2 shows the Raspberry Pi 4:

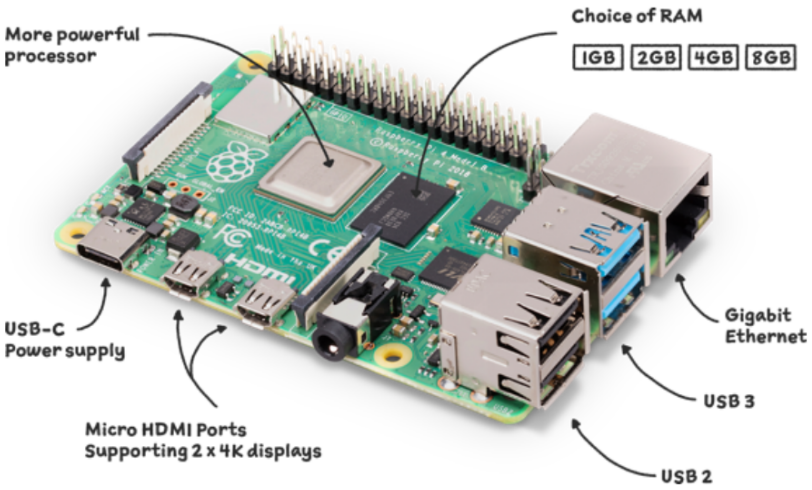


Figure 3.2: Raspberry Pi 4
[11]

3.1.2 Raspberry Pi camera module 3 wide

The Raspberry Pi Camera Module 3 Wide is one of the project's main components. The camera module's dimensions of 25 × 24 × 12.4 mm allow integration into the cart design without adding weight or hindering navigation. Its high resolution of 11.9 megapixels offers exquisite picture capture, allowing for reliable product tracking. The camera's Sony IMX708 sensor provides high picture quality and light sensitivity, which are required for consistent performance in a variety of lighting settings. The large horizontal Field Of Veiw (FOV) of 120 degrees and vertical FOV of 67 degrees give thorough coverage of the cart's interior, ensuring that no detail is overlooked [12]. Figure 3.3 shows the Raspberry Pi camera module 3 wide:

Figure 3.3: Raspberry pi camera module 3 wide
[12]

Mounted strategically on the cart, the camera captures a full view of its contents, allowing for real-time monitoring and verification of scanned objects. Furthermore, the camera's capacity to collect high-resolution pictures at a high frame rate allows for quick tracking of fast-moving items, which improves checkout efficiency and accuracy. The camera integrates the You Only Look Once (YOLO) v8 algorithm, which makes use of cutting-edge object detection capabilities to provide real-time analysis and identification of items in the cart. YOLOv8, developed by Ultralytics, is the most recent advancement in real-time object detection, expanding its capabilities to include instance segmentation, pose estimation, tracking, and classification [13].

3.1.3 EVAWGIB barcode scanner

The EVAWGIB Embedded Mini USB Fixed Mount Barcode Scanner Scan Engine is a critical component for accurate product identification. This tiny scanner has outstanding features, such as a resolution of 300,000 and the ability to quickly detect both One Dimension (1D) and Two Dimensions (2D) codes. Its mini-size module, which measures 7 cm in diameter, 5 cm in breadth, and 2.3 cm in height, assures minimal space consumption and seamless integration into the cart's design. The scanner's USB interface allows it to effortlessly interact with the Raspberry Pi 4, offering smooth interoperability and convenience of usage inside the cart system. Its automated sensing capability enables barcode induction reading, making it perfect for use with self-service equipment. EVAWGIB scanning code functionalities give quick read performance in milliseconds, thus improving efficiency and accuracy and giving seamless product identification for a better shopping experience. The barcode scanner is strategically mounted on the cart's upper edge, making it easy and straightforward to scan things as they are placed within the cart [14]. Figure 3.4 shows the EVAWGIB barcode scanner:

Figure 3.4: EVAWGIB barcode scanner
[14]

3.1.4 Weight sensor

The 4 pieces of 50 kg half-bridge strain gauge load cell weight sensor, when used with the HX711 AD Weight Module, reliably measure the weight of objects placed in the cart. This sensor was chosen for its excellent precision and dependability, which ensure accurate weight measurement for a wide range of items. Furthermore, the module used can read up to 200 kg, which corresponds well to the weight limit for a standard cart, offering adequate capacity for a variety of shopping items. The weight sensor is strategically placed at the bottom of the cart, allowing it to detect and quantify the weight of objects as they are added or withdrawn during the shopping process. This real-time weight data is critical to improving the functioning of our cart system,

allowing features like sensing the entry or removal of products from the shopping cart and fraud detection [15]. Figure 3.5 shows 4 pieces of load cells:

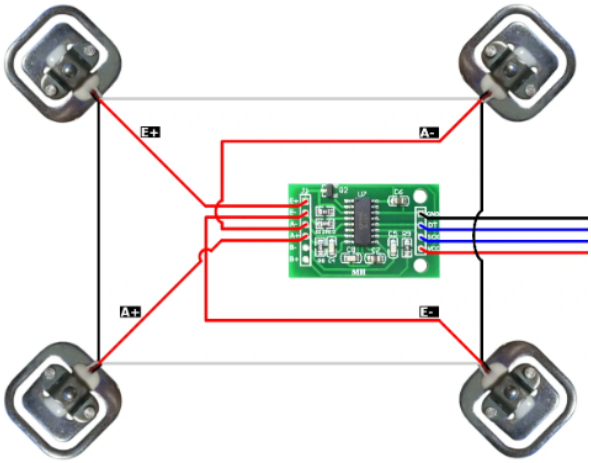


Figure 3.5: 4pcs 50kg Half-bridge strain gauge Load Cell
[15]

3.1.5 Raspberry Pi Touch Display

The cart utilizes a Raspberry Pi touchscreen (7-inch) to improve user engagement and experience. This display has an intuitive touch interface, allowing users to browse the Palestine IntelliCart application easily. The screen fits nicely inside the cart's design, measuring roughly 17.78 cm in width and 11.43 cm in height, and does not restrict the user's vision or mobility. Mounted neatly at eye level on the cart's handlebar, the touch screen allows customers to easily read their shopping list, receive real-time updates, and interact with the system's capabilities. Its intuitive touch features allow users to explore menus, examine items, and make selections and decisions [16]. Figure 3.6 shows the Raspberry Pi screen:

Figure 3.6: Raspberry Pi 7-inch Touch Display
[16]

3.2 Hardware 3D Renderings

The figures below (3.7-3.10) show Three Dimensions (3D) renders of the proposed system, highlighting the placement of the essential components within the shopping cart. The screen is prominently displayed at the front of the cart, above the handlebar, and serves as a visual interface for users. A Raspberry Pi device is placed behind the screen, and the scanner is easily situated in the front to allow for product scanning during placement. In addition, a camera is strategically positioned at the upper front corner to cover the cart area. These components are intended to be covered by a top frame that serves both protective and aesthetic functions. The figures also illustrate the location of weight cells, which are strategically distributed at the bottom of the cart and will be covered with a layer-to-load distribution.

Figure 3.7: System 3D Front Render

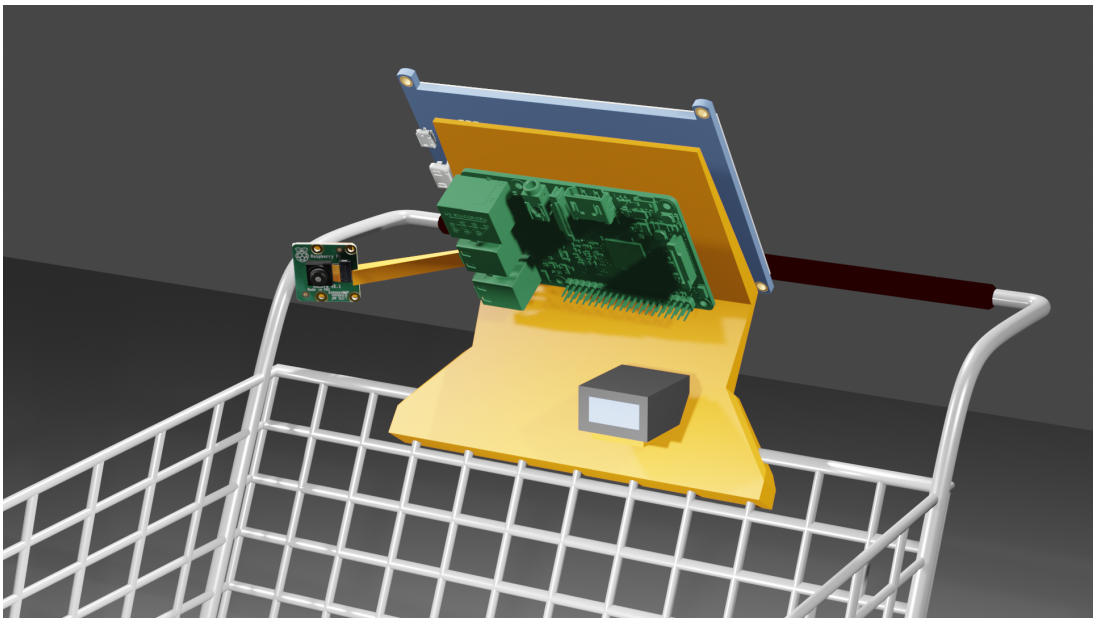


Figure 3.8: System 3D Back Render

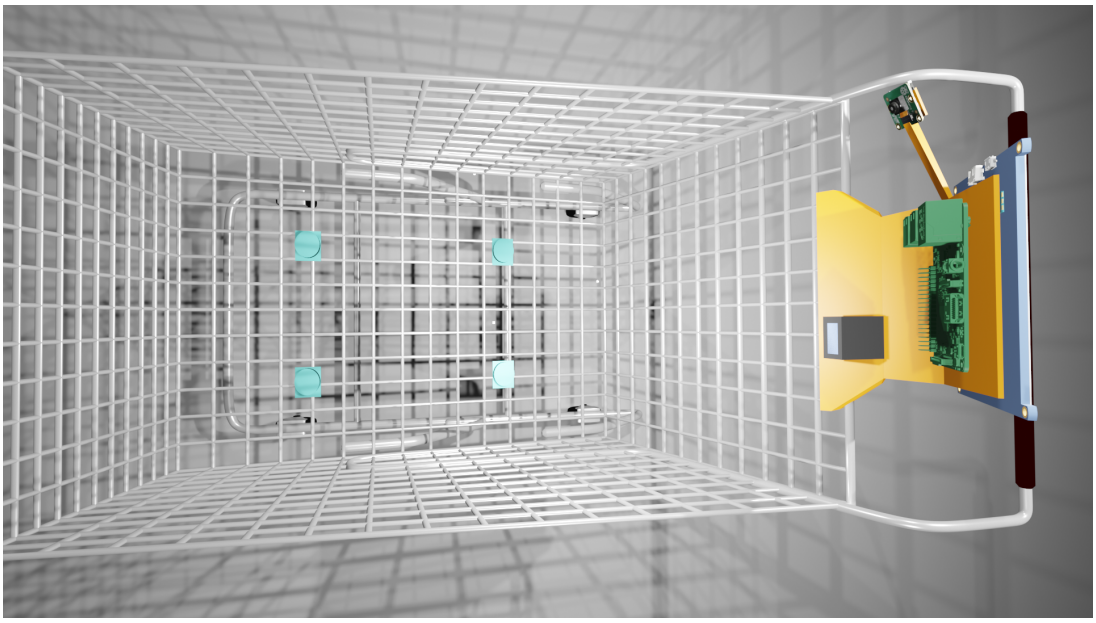


Figure 3.9: System 3D Top Render

Figure 3.10: System 3D Right Render

3.3 System Workflow

‘Palestine IntelliCart ’ workflow is divided into multiple steps. The workflow starts with preparing the system of the cart and initiating the shopping list, then the flow proceeds to waiting for one of the main actions, which consists of detecting an item entering the cart, detecting an item leaving the cart, moving an item inside the cart, searching for an item, showing selected item details, editing settings, requesting help, and checking out.

The cart actions are triggered when the camera or weight senses an item entering, leaving, or moving inside the cart. The computer vision algorithms, weight detection, and barcode scanners work simultaneously to update the information according to the detection movement.

Searching for an item and showing selected item details actions are triggered when the user clicks on the related icon, thus showing an info pop-up of the item info such as price, discount, etc. . Edit settings actions are triggered when the user clicks on related icons, thus allowing the customer to update their personal information. A help request action will show the most frequent questions about the shopping process.

The last action done in each shopping process is the check -out,which is where the customer pays for the items and finishes the shopping process.

Figure 3.11 shows a prototype for a welcome screen on our system:

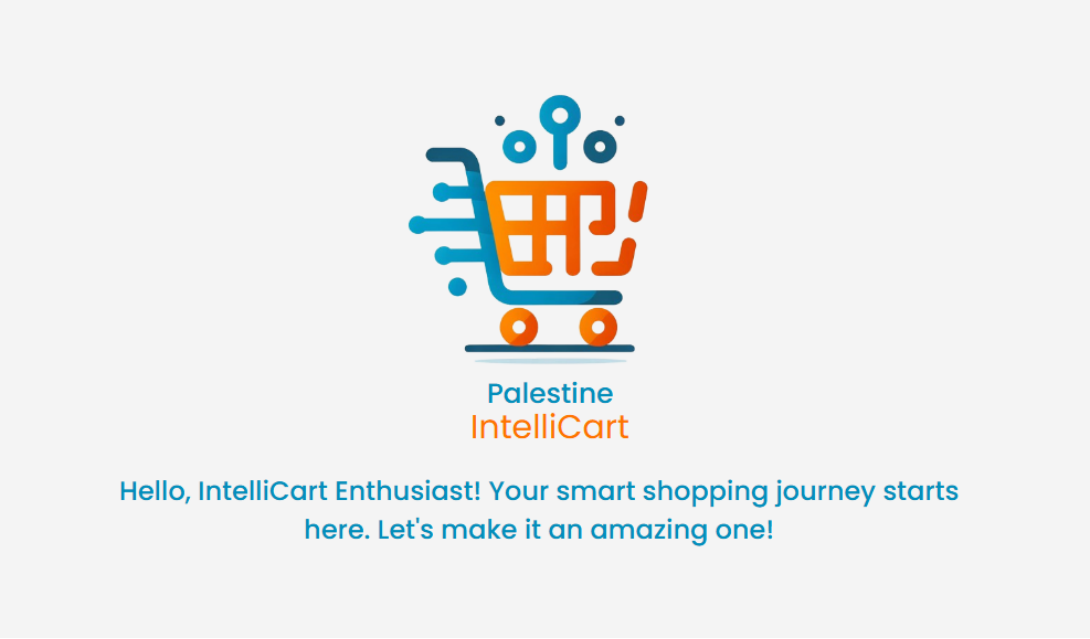


Figure 3.11: Welcome Screen Prototype

3.3.1 Shopping Start

‘Palestine IntelliCart ’ harnesses the power of advanced sensors, artificial intelligence, and IOT connectivity to create a modern and intuitive shopping companion. For a start operation, as shown in figure 3.12, the system checks whether the cart has a connection with the database of the supermarket; if it fails, the system will retry to establish the connection with the database of the supermarket until a connection is formed. Afterwards, the system uses the measured weight from the weight sensor and camera with computer vision algorithms to check if the cart is empty. If it is, the system will continue to the next step, and if it isn’t, it will pop up a warning to empty the cart. In the realm of ‘Palestine IntelliCart ’, the sign-in operation serves as the gateway to a personalized and secure shopping experience. Seamlessly integrating cutting-edge biometric authentication and user-friendly interfaces, the ‘Palestine IntelliCart ’ system ensures swift and secure entry, unlocking a world of tailored services for each customer. Therefore, the system will wait a piece of time for someone to sign in to the system, so the possible options for sign-in are: entering the system using account credentials such as username, email, phone number, and password for security reasons; scanning a QR code on a personal phone to link it with the system as a security precaution; or having the customer sign in as a guest to ease the operation of the sign-in, as shown on the login prototype in figure 3.13. Then the system will create the shopping list to start the shopping activities, and then it will wait for the actions taken by the customer to deal with them.

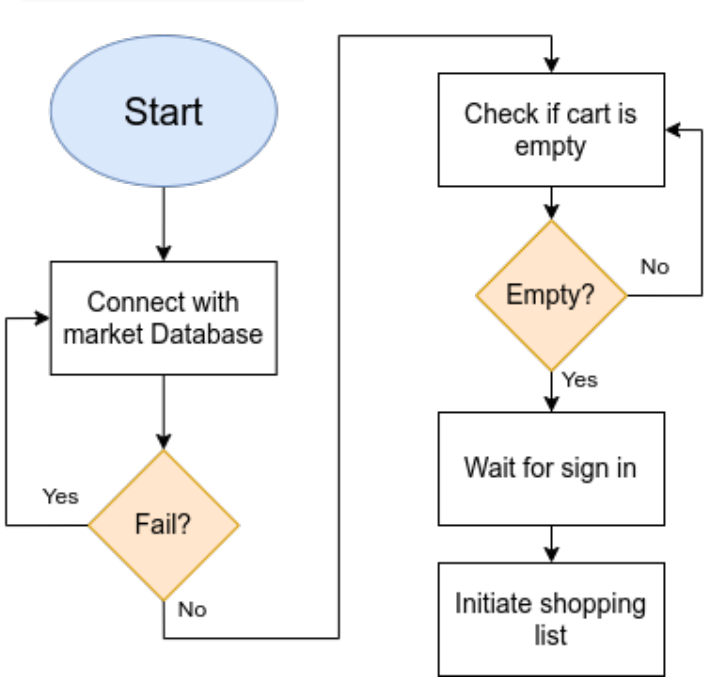


Figure 3.12: Shopping start flow chart

The following figure 3.13 shows a prototype for login method selection page on our system

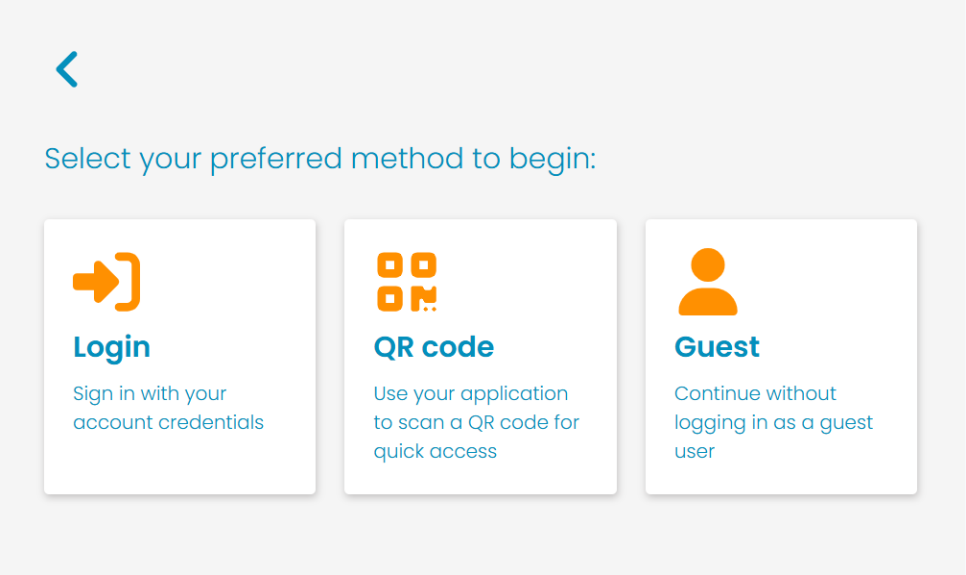


Figure 3.13: Login Method Selection Page Prototype

3.3.2 Shopping Cart Functions

After creating a new shopping list dedicated to the customer, the system begins to continuously monitor the actions that the customer can perform, which are eight operations as shown in figure 3.14, some of which the customer performs via the touch screen, such as searching for an item, displaying details of the chosen product, changing settings, requesting help, or making checkout at the exit point, which ends the shopping process. Other operations are detected by the shopping cart, such as detecting a product entering or exiting the cart or moving products inside the cart. The correctness of performing these operations is detected by integrating the information coming from the camera, barcode scanner, and weight sensor and reflecting it on the user's screen.

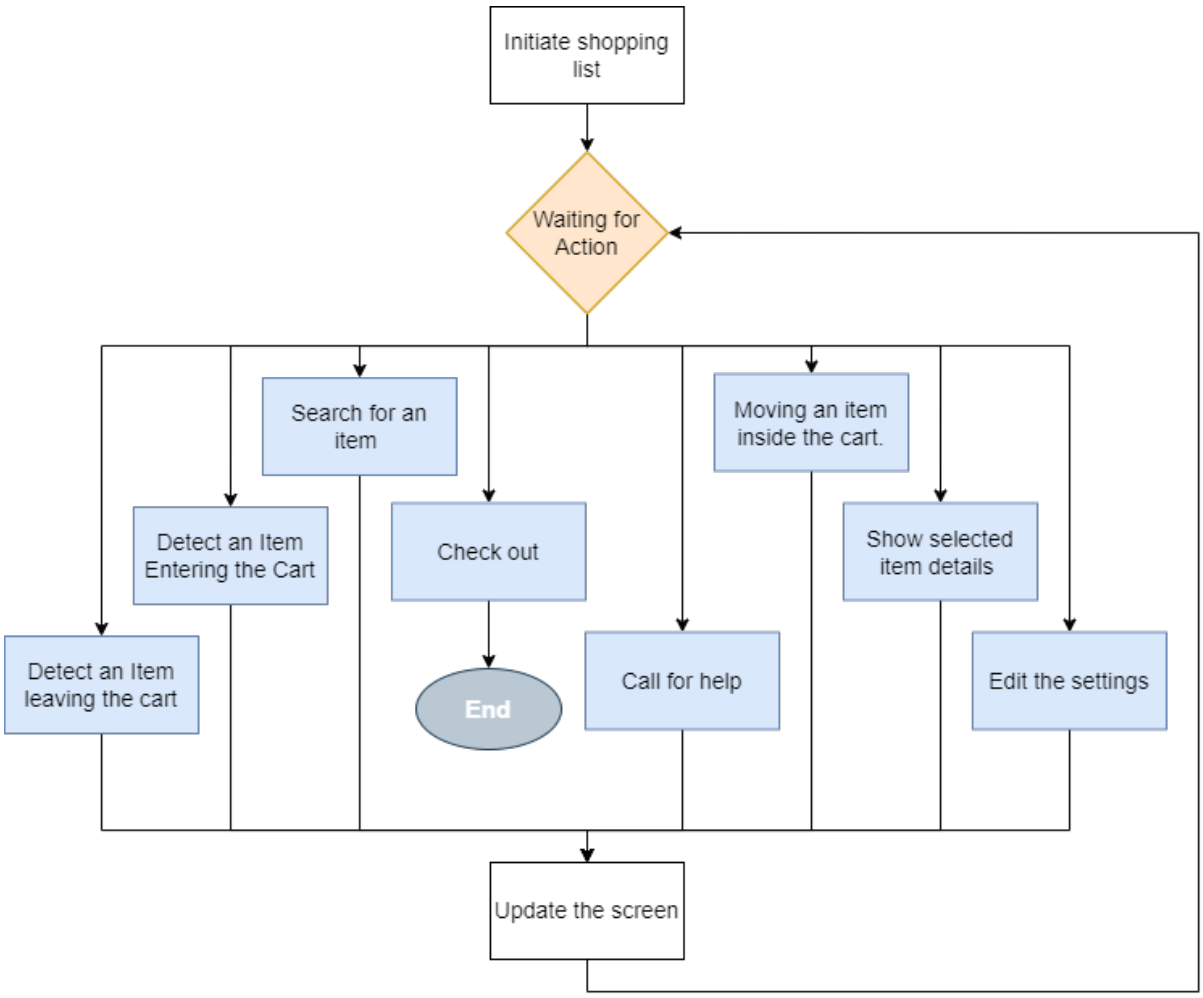


Figure 3.14: Shopping cart functions flow chart

The following figure 3.15 shows a prototype of the home page where the customer initiates the purchase process after the login process:

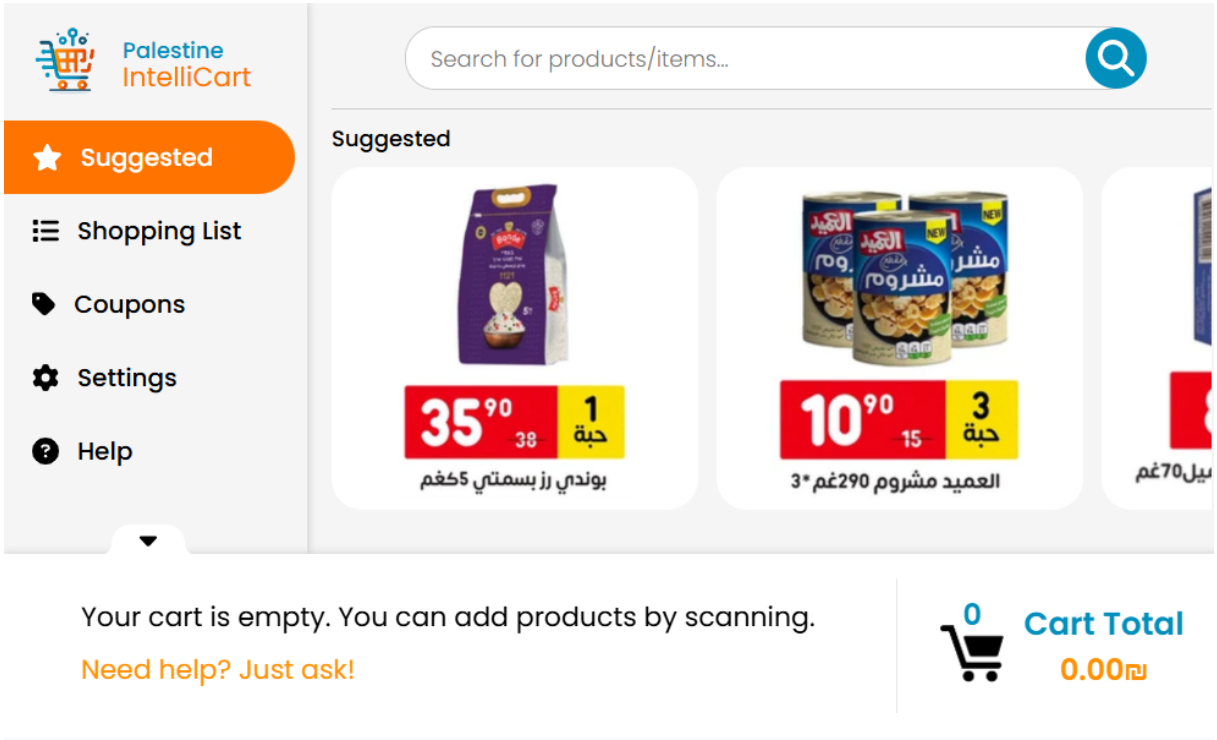


Figure 3.15: Home Page Prototype

3.3.3 Detection: An Item Entering the Cart

The 'Detect an item entering the cart' action is triggered when the camera detects an item entering its view. As shown on the flow chart in figure 3.16, once the item appears in the camera, it will begin tracking the object by feeding the video to the Multi Object Tracking (MOT) algorithm. It is expected of the customer to scan the barcode of the product while entering the cart, as the barcode scanner is located within the camera's field of view to ensure that no other product is inserted other than the currently active one. Once the item settles into the shopping cart, the weight sensor will detect its weight and assign this information to the item object. If the customer has scanned the item properly, the item will be added to the shopping list, and the action will be displayed on the screen mentioning the added item. But if the item wasn't scanned, a reminder will be issued to the customer to scan the item, and it will be added to the unscanned items list, which tracks all items that have not been scanned. As long as the list of unscanned items is not empty, a warning will be displayed informing the customer that the unscanned item should be scanned or removed from the cart. Therefore, the customer will be prevented from taking any action on the screen other than calling for help.

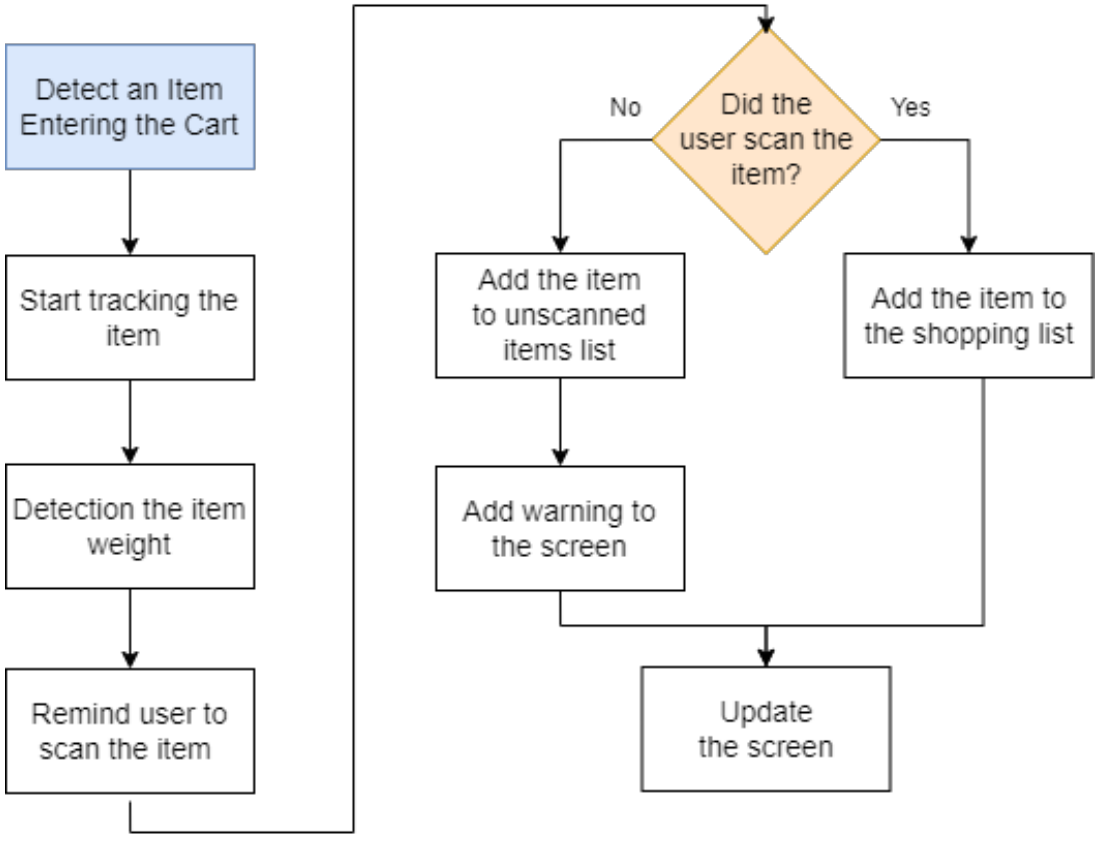


Figure 3.16: Detection an item entering the cart flow chart

The following figure 3.17 shows a hypothetical scene of the process of detecting the entry of a product into the shopping cart via a camera:

Figure 3.17: Detection of an item entering the cart by camera

Figure 3.18 shows a prototype for successfully inserting a product into the cart. It shows the interaction of the screen with the customer by showing the details of the product added to the cart and the updated bill:

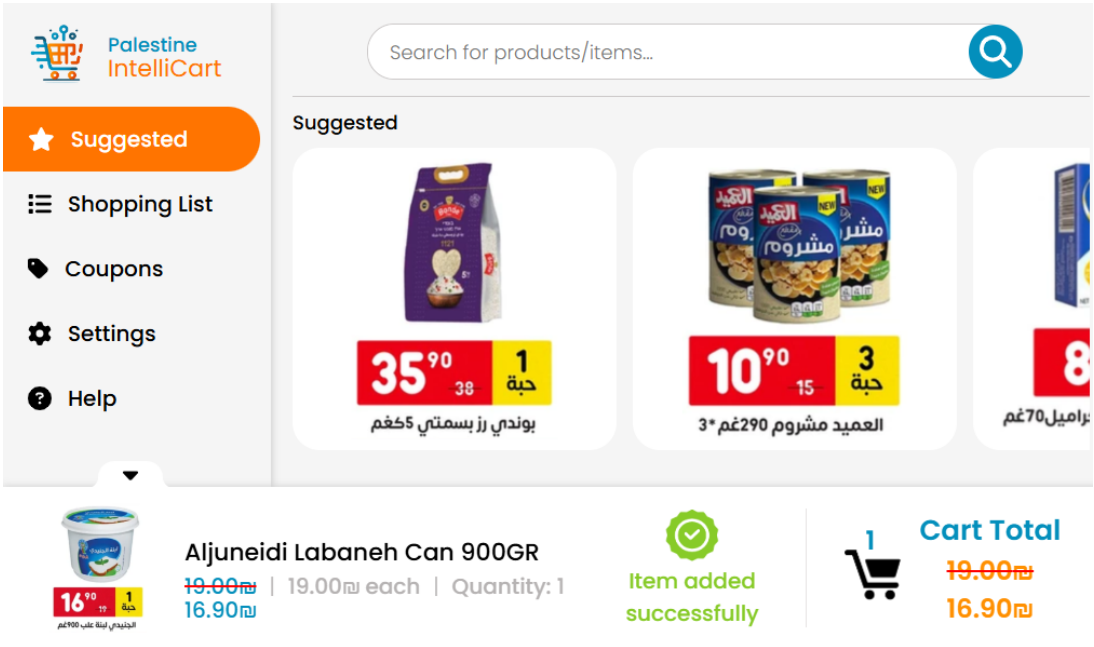


Figure 3.18: Notification: Product Successfully Added to Cart

The following figure 3.19 shows a prototype for a process in which a product was inserted into the cart without being scanned:

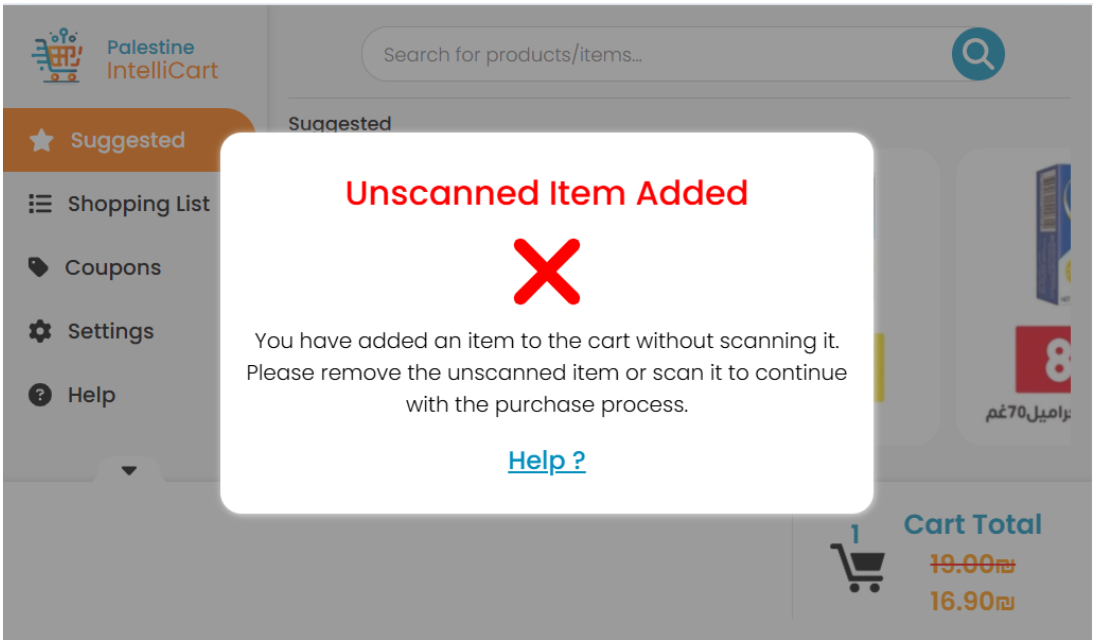


Figure 3.19: Warning: Non-Scanned Product Added

3.3.4 Detection: An Item Leaving the Cart

'Detect an item leaving the cart' action is triggered when the camera detects an item leaving its view. As shown in figure 3.20, as soon as the item starts moving, the item information will be retrieved. Using the information, if the item is known to be scanned, the item will be removed from the shopping list, and the user will be notified of the change. But if the item wasn't scanned, the item will be removed from the unscanned items list, and the warning will be removed only if the unscanned items list is empty after removing the item.

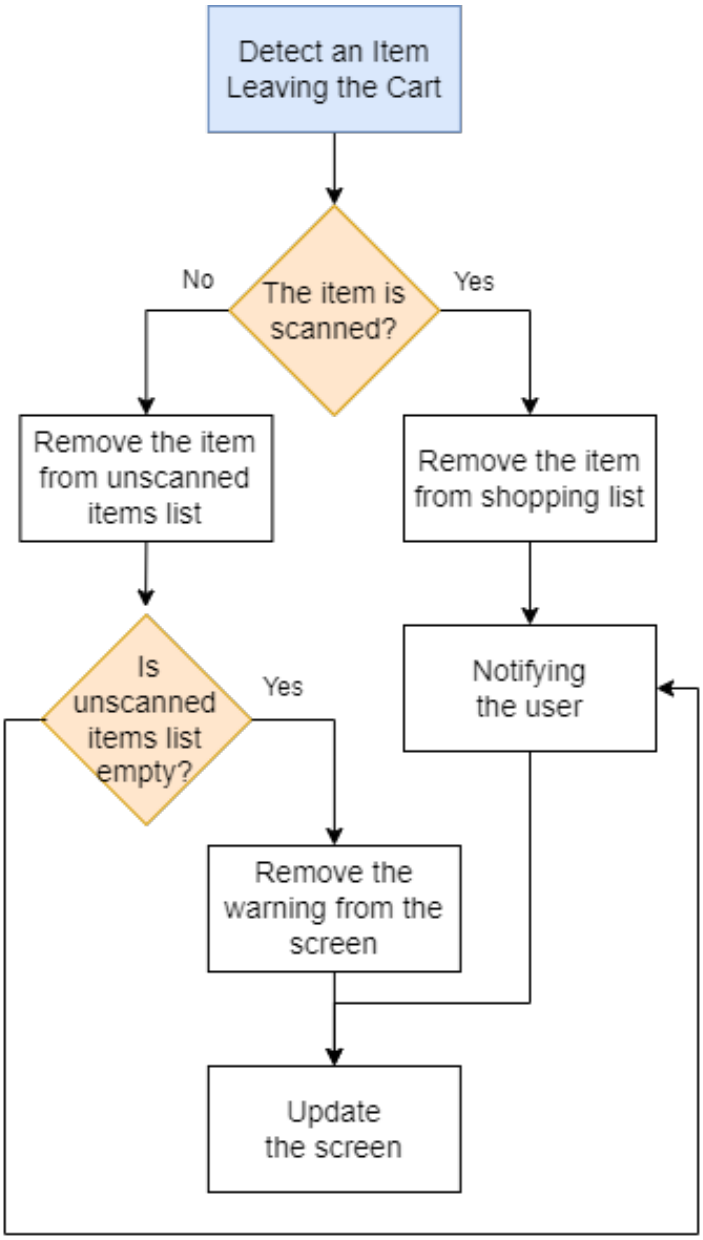


Figure 3.20: Detection an item leaving the cart flow chart

3.3.5 Detection: An Item Moving inside the Cart

Moving an item inside the cart action is triggered when the camera detects an item moving in its view. As shown in figure 3.21, as soon as the item starts moving, the item information will be retrieved and its information will be updated. Using the information, if the item wasn't scanned, the user will be asked to scan the item, and if the user scans the item, it will be removed from the unscanned items list and added to the shopping list. If the unscanned items list becomes empty, the warning will be removed.

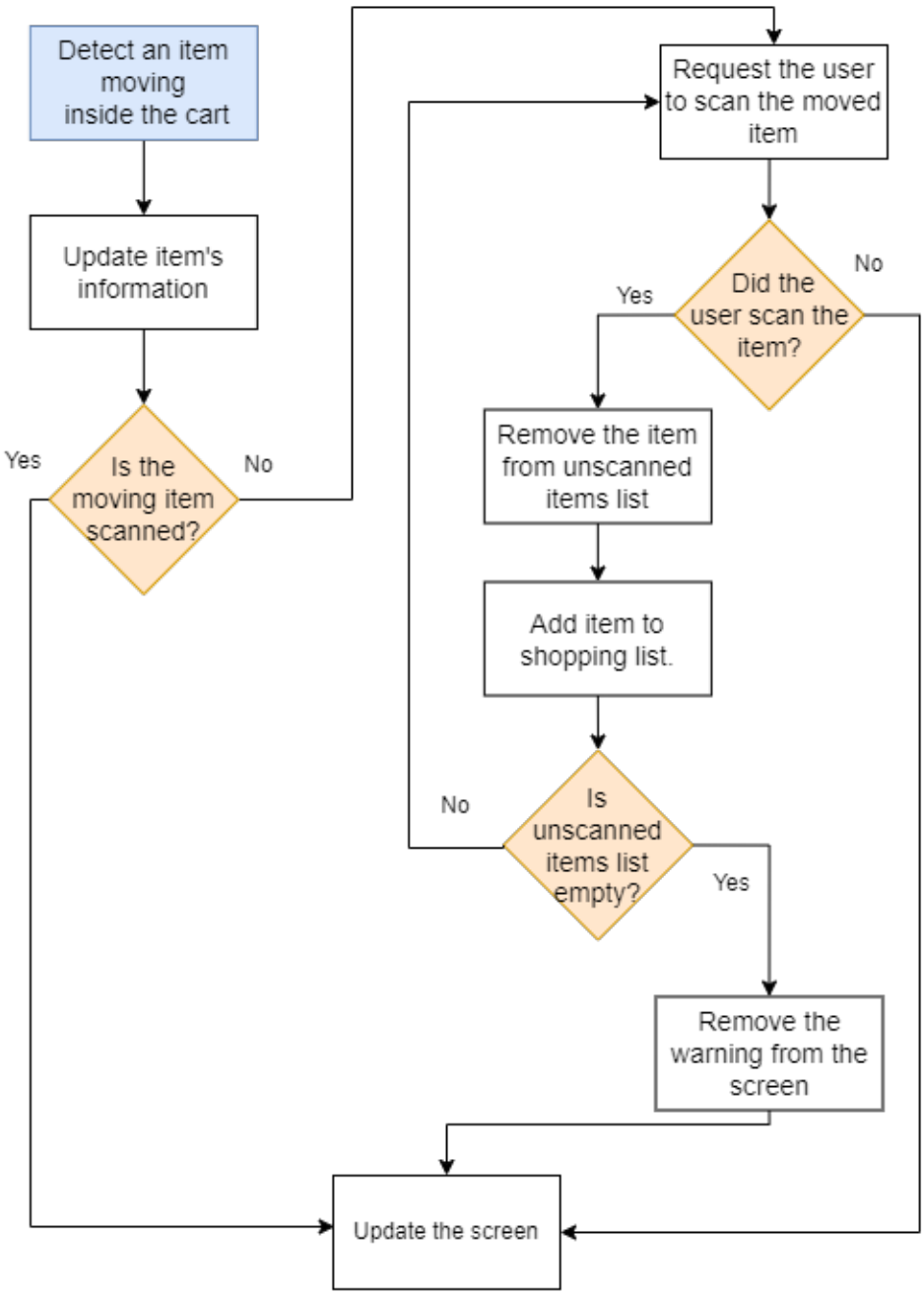


Figure 3.21: Detection an item moving inside the Cart flow chart

3.3.6 Search for an Item

There are variations of actions that can be done by customers provided by the system. One that might be a favorite for the customers is that the customer can search for any item, activate this service by clicking on the search icon, then enter a name, and the system will view a list of all possible items related to the name entered by the customer. Then the customer can click on an item from the list to view the information for it. Figure 3.22 shows a prototype example for searching for a cheese item in the system:

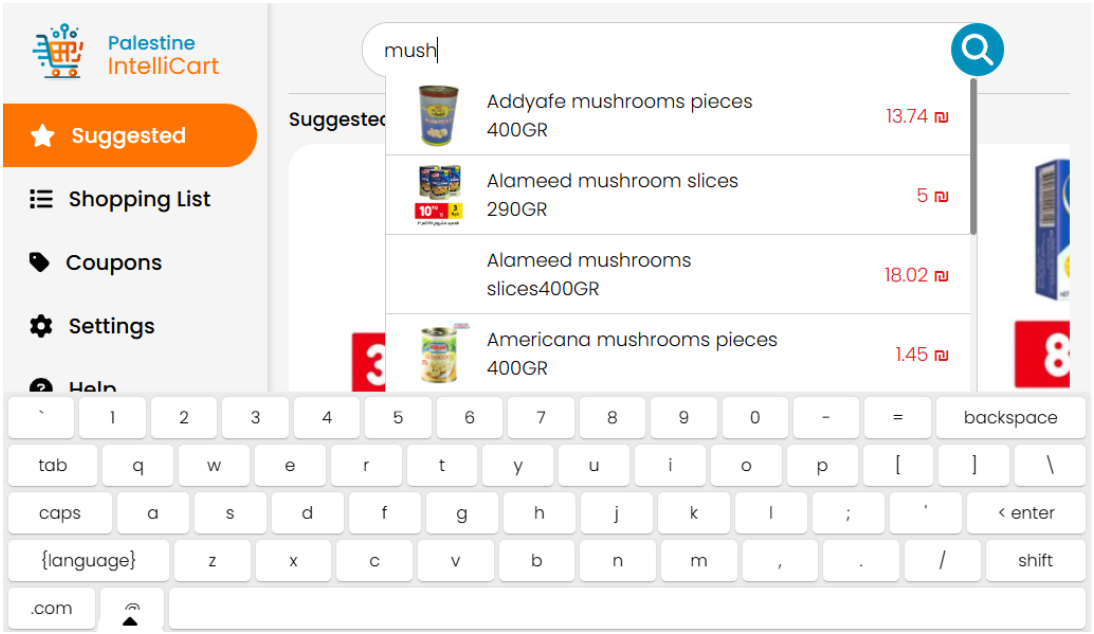


Figure 3.22: Search Item Page Prototype

3.3.7 Product Information Display

Based on the shopping list initiated by the system when the customer signed in, the system gives the customer the ability to see the details of the items, so the screen on the cart always shows the shopping list. Therefore, the customer can show the item details by clicking on the item icon, and then the system will update the screen and show the item details like price, production date, expiry date, and producing country and company. Figure 3.23 shows a prototype of the process for displaying details of a product:

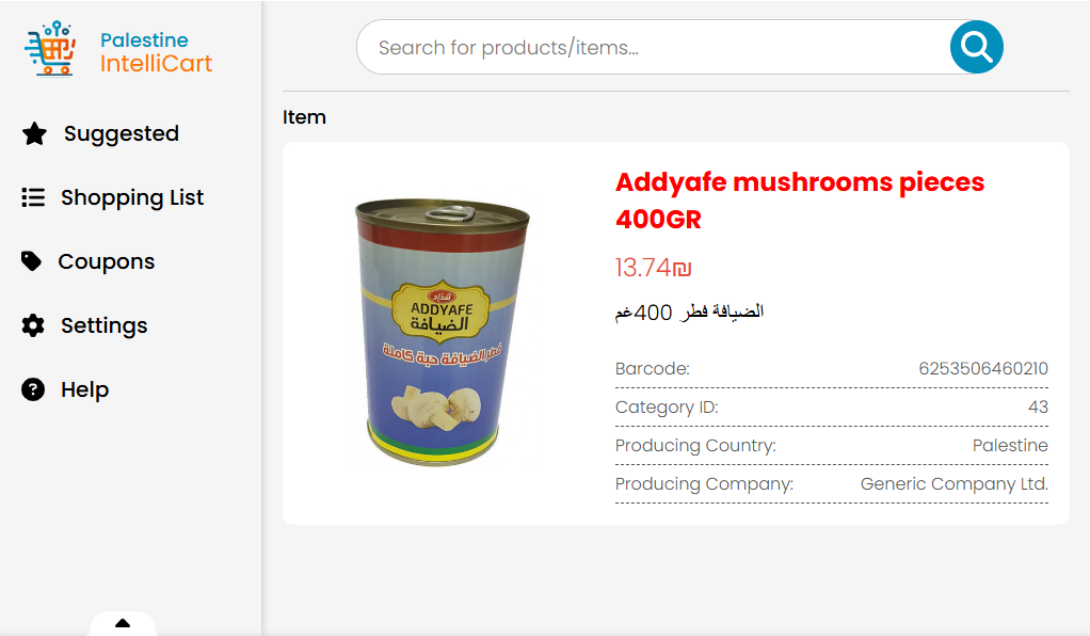


Figure 3.23: Product Information Display Page Prototype

3.3.8 Edit Settings

To provide ease of use, edit settings have been provided to the customer by clicking on the settings icon, and then the screen will show the settings list. The user is able to change the credentials related to the account, like username, email, phone number, password, and other properties with a high level of security, and edit the payment information that can be used for checkout and billing.

3.3.9 Request help

Request help can be triggered by clicking the icon named help. Then the most frequently asked questions will be suggested, such as “How can I add an item to the shopping list?” “Why is there a warning on my screen?” “I can’t scan an item; what is the issue?” and “How to remove an item from the shopping list.”.

3.3.10 Check Out

In the checkout process, the shopping process ends when the user selects the payment option. This option is only available if the list of unscanned items is empty. The user is then asked to choose a payment method, either in cash or via bank account. The system then displays the Quick Response (QR) code on the screen. At the checkout gate, the employee scans the code to get information about the purchase process, and after the user pays for the purchases, the employee sends a confirmation to the system that the payment process has been completed, which appears on the cart screen. After the customer leaves with the purchases, a message is sent to the customer containing information about the amount paid and a list of purchases if the customer has contact information on the account. Figure 3.24 shows the checkout process flow chart:

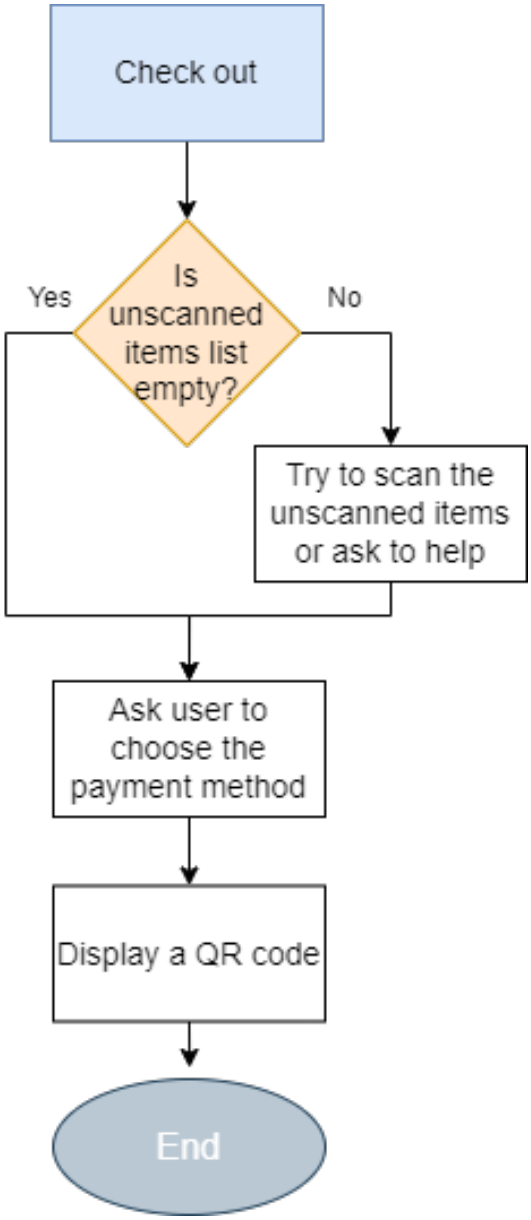


Figure 3.24: Check Out Flow Chart

The prototype in Figure 3.25 shows the process of displaying the details of the shop-

ping list and invoice to the customer and the presence of the checkout option to end the shopping process:

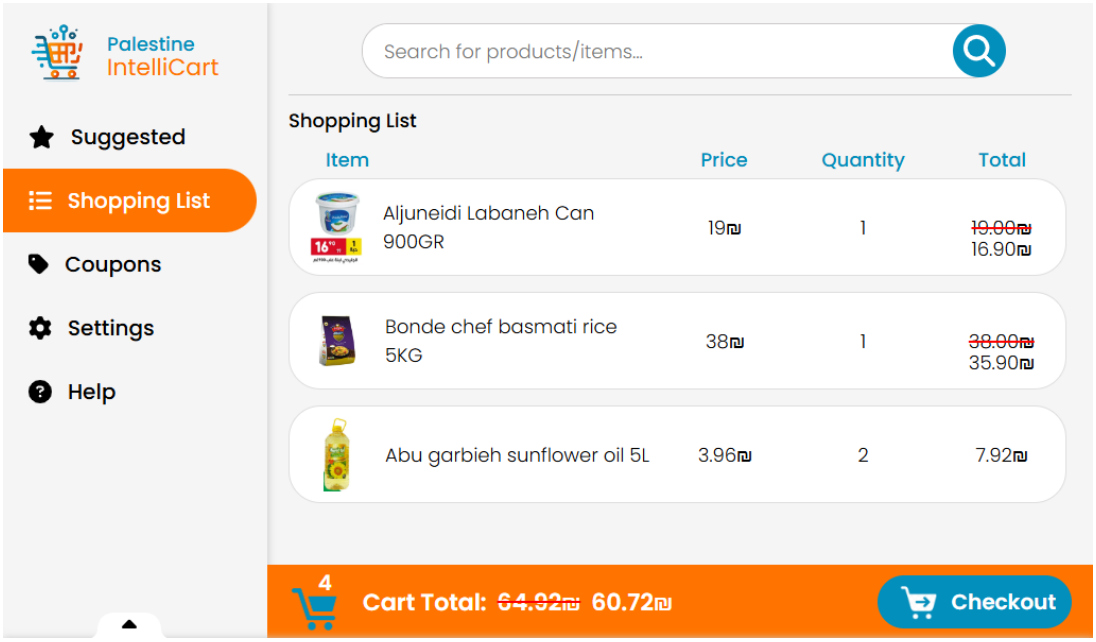


Figure 3.25: Shopping List Preview Prototype

Also, the prototype in Figure 3.26 illustrates the process of choosing the payment method by customer:

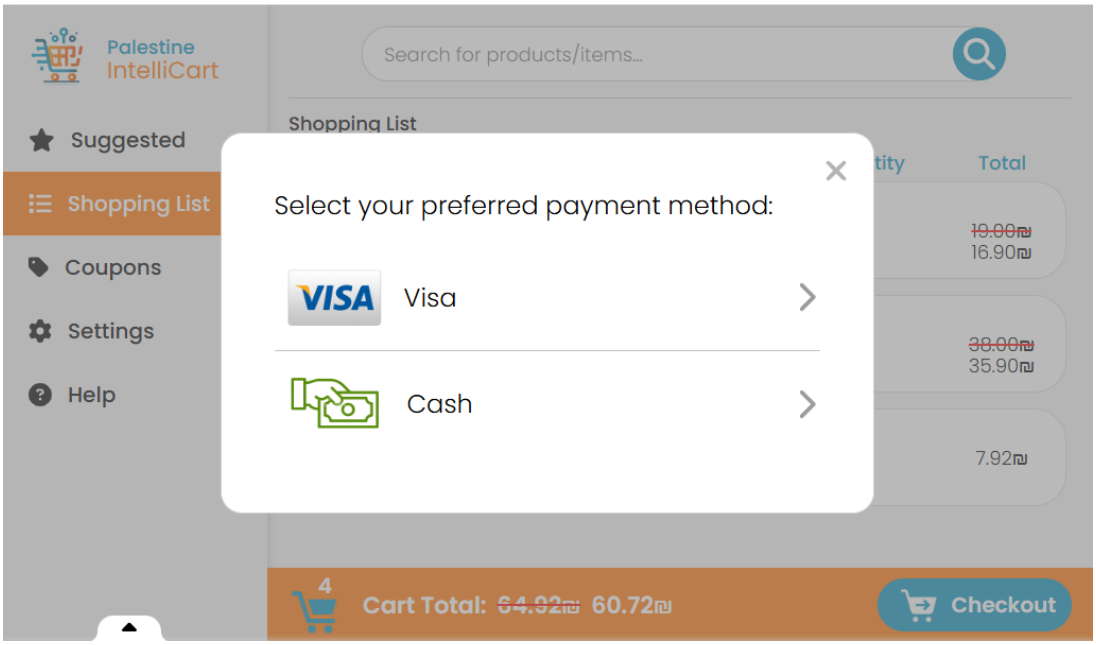


Figure 3.26: Payment Method Prototype

3.4 Hardware System Structure

The hardware system is the part that runs on the Raspberry Pi itself, starting with opening the web browser for the front end of the interactive display and consisting of all the parts that control Palestine IntelliCart functionality. Its main tasks are tracking the states of the objects in the cart and keeping the backend updated about their current state.

The system starts as soon as there is a new locally opened session, and it stores the session ID as an identifier for the operations done by this intellicart. The first step when initiating the current process is to identify the current state. If there are any objects in the cart at the start of the session, then the state is identified as corrupted and should be emptied before continuing the process. When the cart is empty, the system identifies the dimensions of the cart box, preparing it for the incoming load.

The process consists of three continuously running tasks, which are weight measurement, barcode recognition, and item tracking.

3.4.1 Weight Measurement

The weight sensors were distributed evenly on the base of the cart with an iron frame on top of the sensors to ensure accurate measurements, and then the measurement unit was calculated by using a predefined weight. The weight measurement task uses the GPIO of the Raspberry Pi to read the readings of the hx711 model, which gives a 24-bit number representing the current weight of the items in the cart. The system watches the changes in the weight value and reports any suspicious behavior that doesn't match the current state of recorded objects in the system.

Figure 3.27: Load Sensor Distribution

3.4.2 Barcode recognition

The EVAWGIB barcode scanner uses USB protocol to send the information about the scanned barcode as soon as it appears in front of the scanner. Since it uses the same method of outputting results as the regular keyboard, there is a need to filter the outputs of EVAWGIB from other resources, which is done by the hardware system.

When a barcode scan happens, the new barcode is associated with the closest track to the place of the barcode scanner. The system uses two measurements, which are the Euclidean distance and the IOU, to determine the closest object. After an item is scanned for the first time, the hardware system will inform the server that the added item has been identified, which will result in calls according to the current state of the track. There are three possible scenarios to happen regularly: the item gets scanned for the first time, meaning that the server will get informed of a new object inside the cart; the object was scanned previously, so the system will stay in the same state; and the object was scanned but then left the cart, resulting in informing the cart of this state.

Figure 3.28: Barcode placement in the cart.

3.4.3 Item tracking

The Raspberry Pi camera is used to track each item in the camera's view, whether it is still inside the cart or moving around it. Several computer vision models were tested to track items in the cart, YoloV8 was superior because of its community support, flexibility, speed, and accuracy.

YoloV8s model was trained and used to track objects in the camera view, with a deep sort tracker algorithm. YoloV8 tracks are initiated to track each item's movements in the cart, therefore detecting the current state for each item and sending a warning if an item wasn't scanned properly, or updating the status of the item in the shopping list according to the current place of the item.

The system was trained using a dataset of 50k training images, 6k validation images, and 20k testing images. Each image contains multiple instances of various products represented in 200 types. The model YoloV8s and YoloV8n were tested where YoloV8s

has 12 million parameters and YoloV8n has 3 million parameters. The following graphs show a general and specific description of the architecture of the yolov8 model.

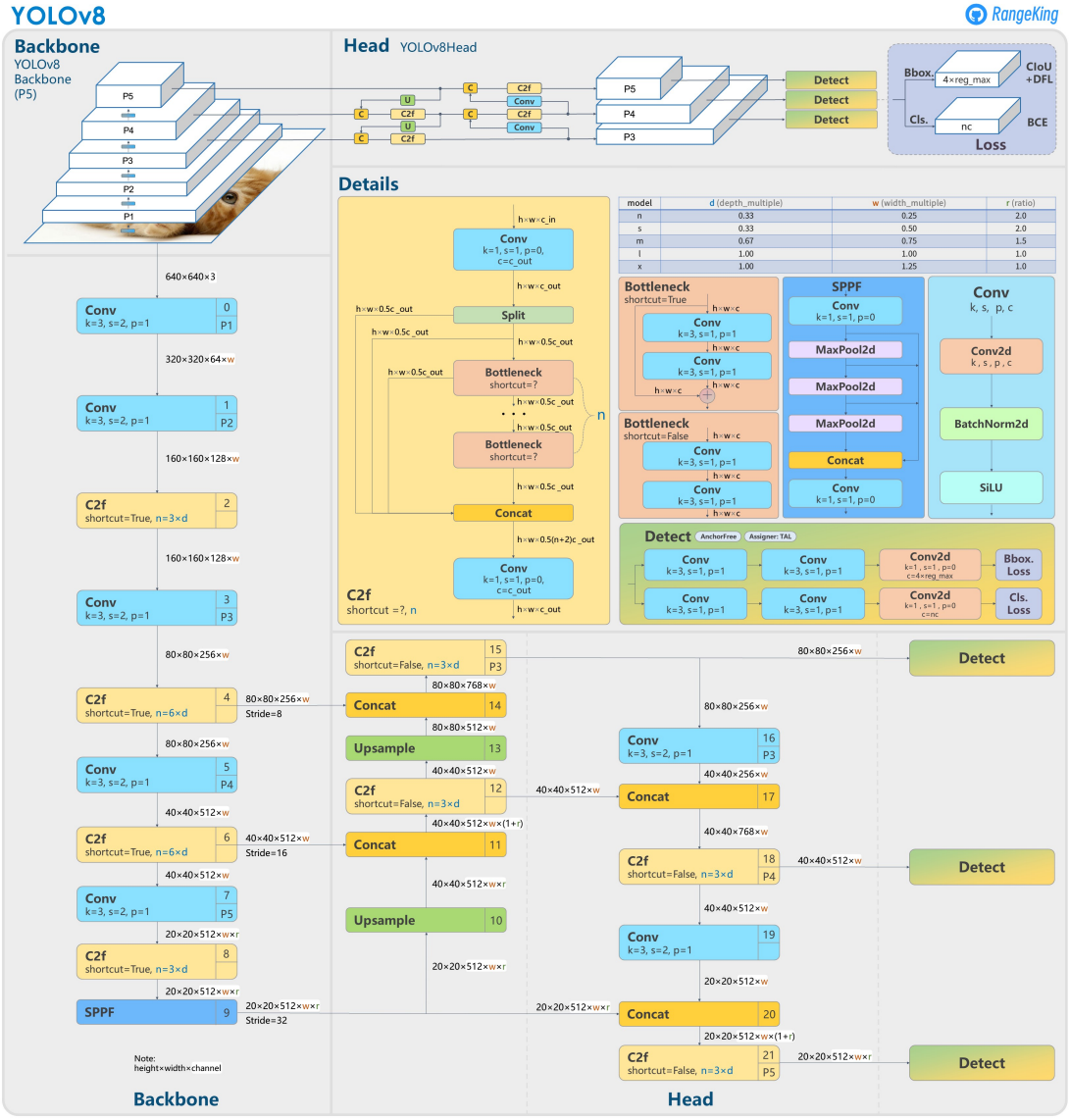
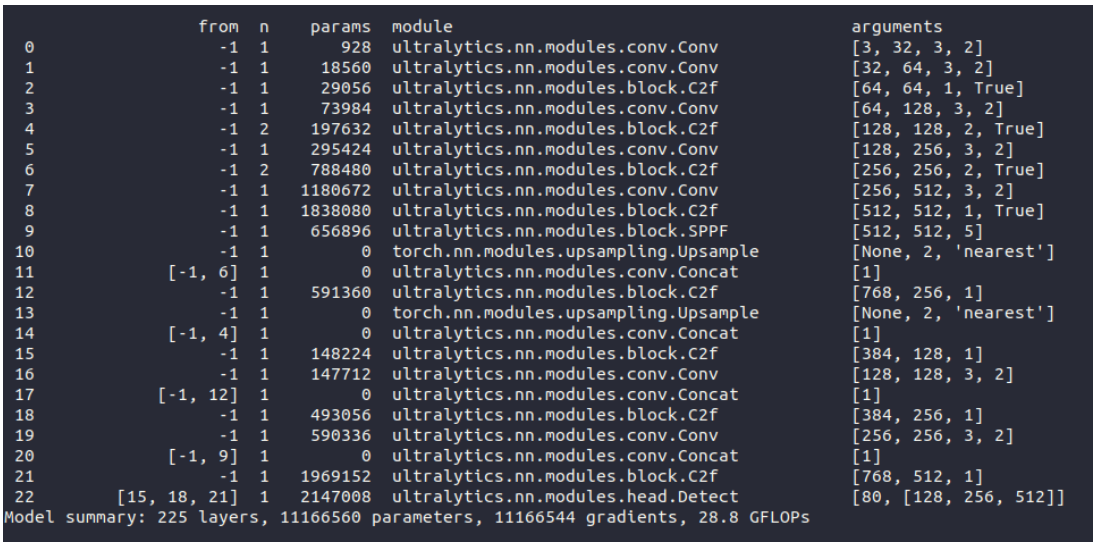


Figure 3.29: YoloV8 Overview



```
                 from  n    params  module                                       arguments
  0                -1  1       928  ultralytics.nn.modules.conv.Conv             [3, 32, 3, 2]
  1                -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  2                -1  1     29056  ultralytics.nn.modules.block.C2f             [64, 64, 1, True]
  3                -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  4                -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  5                -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  6                -1  2    788480  ultralytics.nn.modules.block.C2f             [256, 256, 2, True]
  7                -1  1   1180672  ultralytics.nn.modules.conv.Conv             [256, 512, 3, 2]
  8                -1  1   1838080  ultralytics.nn.modules.block.C2f             [512, 512, 1, True]
  9                -1  1    656896  ultralytics.nn.modules.block.SPPF            [512, 512, 5]
 10                -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11            [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                -1  1    591360  ultralytics.nn.modules.block.C2f             [768, 256, 1]
 13                -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14            [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 16                -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 17           [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
 19                -1  1    590336  ultralytics.nn.modules.conv.Conv             [256, 256, 3, 2]
 20            [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                -1  1   1969152  ultralytics.nn.modules.block.C2f             [768, 512, 1]
 22      [15, 18, 21]  1   2147008  ultralytics.nn.modules.head.Detect           [80, [128, 256, 512]]
Model summary: 225 layers, 11166560 parameters, 11166544 gradients, 28.8 GFLOPs
```

Figure 3.30: YoloV8s Architecture

3.5 Back-end Development

3.5.1 Backend Technologies and Tools

The backend for the Palestine IntelliCart project was developed using Python, leveraging its simplicity and extensive libraries. The Flask framework was utilized for building the server-side application, chosen for its modularity and flexibility. Data was managed using a MySQL database known for its reliability and performance. Testing of the system was conducted using Postman, allowing for thorough validation of API endpoints and ensuring robust functionality.

3.5.2 Database Design

3.5.2.1 ER Diagram

The database design for the Palestine IntelliCart project consists of several interrelated tables that manage user data, product information, purchases, payments, and offers. The following is the ER diagram representing these tables and their relationships:

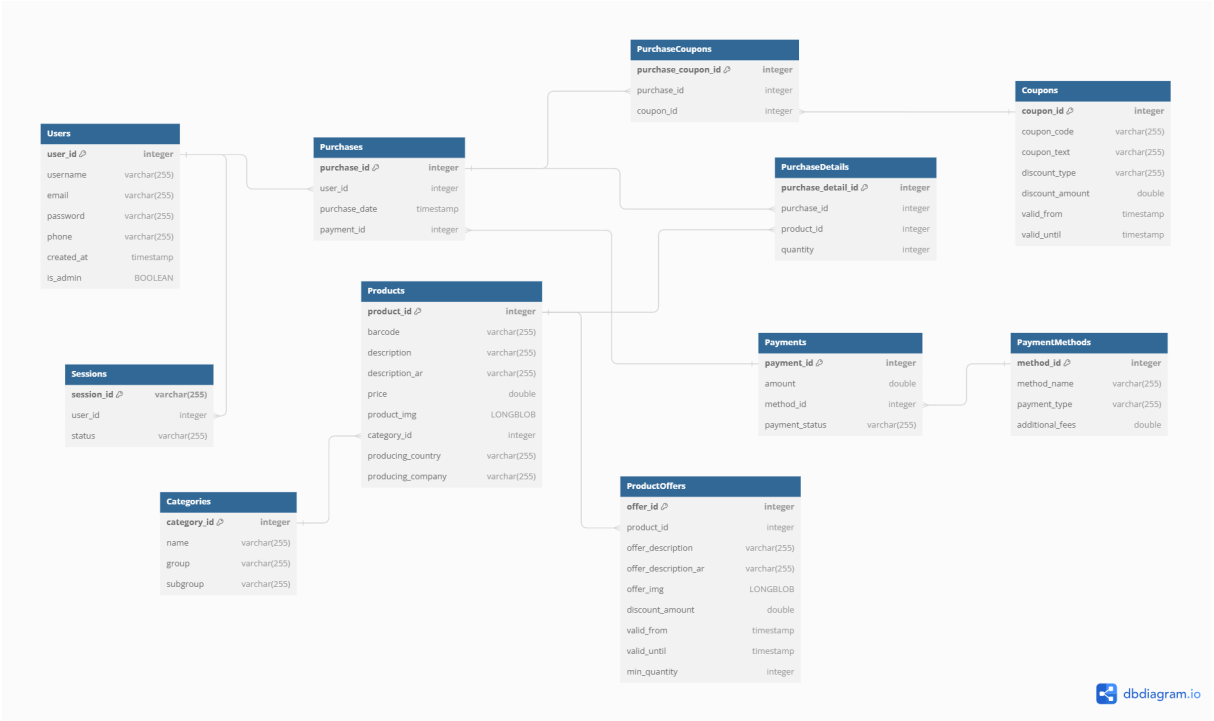


Figure 3.31: Database Schema

3.5.2.2 Data Models

Users Table: Stores user details, including admin status.

Products Table: Contains detailed product information.

Categories Table: Organizes products into hierarchical categories.

Purchases Table: Records transactions, linking to users and payments.

PurchaseDetails Table: Provides details on products within each purchase.

PaymentMethods Table: Lists available payment methods.

Payments Table: Manages payment transactions.

Coupons Table: Stores coupon details and validity.

PurchaseCoupons Table: Links purchases with applied coupons.

ProductOffers Table: Contains promotional offers on products.

Sessions Table: Manages user sessions with a foreign key linking to the Users table.

3.5.2.3 Database Normalization

The database was normalized to eliminate redundancy and ensure data integrity. Primary keys were utilized to uniquely identify records, while foreign keys were employed to enforce relationships between tables, maintaining referential integrity. Each table stores data relevant to a single entity, which reduces duplication and promotes efficient data management.

3.5.3 API Development

3.5.3.1 RESTful API Design Principles

The API for the Palestine IntelliCart project follows RESTful design principles to ensure it is stateless, scalable, and easy to maintain. Each endpoint is designed to perform a specific action related to the shopping cart, and HTTP methods (GET, POST, PUT, and DELETE) are used appropriately to handle requests.

3.5.3.2 Endpoints and Routes

User Management

- POST /users/add: Adds a new user to the database.
- PUT /users/updateuser: Updates the authenticated user's information.
- GET /users/getuser: Retrieves the authenticated user's information.

Authentication and QR code login

- POST /login: Authenticates a user using their email and password, and returns a JWT token.
- POST /guest/login: Creates a new guest user, generates a purchase ID, and returns a JWT token.
- POST /qr/login: Authenticates a user using a scanned QR code session ID, and returns a JWT token.

- GET /generate_login_qr: Generates a QR code for login and returns it as an image along with a session ID.
- POST /validate_qr: Validates a QR code by linking it to a user ID and updating its status to 'scanned'.

Category Management

- POST /categories/add: Adds a new category to the database.
- DELETE /categories/category_id: Deletes a category by its ID
- PUT /categories/category_id: Updates a category's information.

coupons Management

- GET /coupons: Retrieves a list of all coupons.
- POST /coupons/add: Adds a new coupon to the database.
- PUT /coupons/coupon_id: Updates a coupon's information.
- GET /coupons/valid: Retrieves only valid coupons based on current date/time.

Payment Method Management

- POST /add_payment_method: Adds a new payment method to the system.
- GET /payment_methods: Retrieves a list of all payment methods available.

Product Offers Management

- GET /product_offers: Retrieves a list of all product offers.
- POST /product_offers/add: Adds a new product offer to the system.
- PUT /product_offers/<offer_id>: Updates an existing product offer.
- GET /product_offers/product/<product_id>: Retrieves product offers for a specific product ID.
- GET /product_offers/active: Retrieves active product offers based on the current time.

Product Management

- GET /products: Retrieves a list of all products.
- POST /products/add: Adds a new product to the system.
- PUT /products/<product_id>: Updates an existing product.

- GET /products/<product_id>: Retrieves details of a product by its ID.

Purchase Coupons Management

- POST /purchase_coupons/add/<int:coupon_id>: Adds a coupon to a specific purchase identified by the coupon_id.
- DELETE /purchase_coupons/delete/<int:coupon_id>: Deletes a coupon from a specific purchase identified by the coupon_id.
- PUT /product_offers/<offer_id>: Updates an existing product offer.
- GET /product_offers/product/<product_id>: Retrieves product offers for a specific product ID.
- GET /product_offers/active: Retrieves active product offers based on the current time.

Purchase Details Operations

- POST /add_to_cart/<string:barcode>: Adds a product identified by barcode to the current purchase cart.
- POST /remove_from_cart/<string:barcode>: Removes a product identified by barcode from the current purchase cart.
- POST /log_scanning_error/<string:error_type>: Logs a scanning error of type error_type for the current purchase.
- POST /resolve_scanning_error: Resolves the scanning error for the current purchase.
- GET /purchase_details/products: Retrieves all products in the current purchase cart with their details.

Payment Management

- GET /generate_payment_qr: Generates a QR code for initiating a payment.
- POST /create_payment: Creates a payment for the current purchase.

3.5.3.3 Authentication and Authorization

The backend of the Palestine IntelliCart project employed robust authentication and authorization mechanisms to ensure secure access to the API endpoints. This is critical for maintaining the integrity and security of user data and ensuring that only authorized actions are performed on the system. The project has utilized JSON Web Tokens (JWT) for managing authentication and authorization.

Token-Based Authentication

JWTs were used to securely transmit information between the client and server. These tokens contain encoded data that can be verified and trusted. The tokens were issued upon successful login and must be included in the Authorization header of subsequent API requests. The token is passed as a Bearer token in the HTTP Authorization header:

Authorization: Bearer <JWT_TOKEN>

Tokens are generated using a secret key and include encoded user information such as the user id and purchase id.

Token Validation

Each endpoint that requires authentication validates the token by extracting it from the Authorization header. Then decode it using the secret key to retrieve the encoded information. The decoded information is verified to ensure it is valid and has not expired.

WebSocket Integration

In addition to standard HTTP methods, the backend also was integrated WebSocket for real-time communication. This is particularly useful for scenarios such as cart updates and scanning error notifications, offering immediate feedback to the user interface. The project uses SocketIO for WebSocket communication, allowing real-time, bidirectional communication between the client and server.

3.5.3.4 Error Handling

Complete error handling has been implemented into action to guarantee appropriate responses have been provided for a range of error situations. Typical error codes consist of:

- **400 Bad Request:** Invalid request format or parameters.
- **401 Unauthorized:** Missing or invalid authentication token.
- **404 Not Found:** Requested resource not found.
- **500 Internal Server Error:** General server error.

The API for Palestine IntelliCart project offers safe and reliable interactions between the client and server, making sure a seamless shopping experience, by following by specific instructions and standards.

3.6 Front-End Development

The front-end development of the Palestine IntelliCart project focused on creating a user-friendly and responsive interface using React, a popular JavaScript library for building dynamic web applications. The main components and functionalities of the front end are presented in this part, focusing on how will it work together with the back end to create a smooth shopping experience.

3.6.1 Framework and Tools

1. **React**

   React was chosen for its component-based architecture, which allows for efficient code reusability and easy management of the application's state. This framework is renowned for its ability to break down the user interface into discrete, reusable components, making it easier to develop and maintain complex applications. By structuring the front-end application using React components, each part of the interface can be independently developed and tested, ensuring a robust and error-free implementation. This modular approach not only enhances maintainability and scalability but also facilitates collaboration among developers, as different team members can work on individual components simultaneously. Moreover, React's virtual DOM mechanism improves application performance by minimizing direct manipulation of the actual DOM, leading to faster updates and rendering.

2. **Additional Tools and Libraries**

   Various tools and libraries were integrated with React to enhance the development process and the functionality of the application.

   (a) Redux: Used for state management, providing a single source of truth for the application's state.

   (b) Axios: A promise-based HTTP client used for making API requests to the backend.

   (c) React Router: Facilitates navigation between different views or pages within the application.

3.6.2 User Interface Components

1. **User Authentication**

   Initially, there is a page to choose one of the three login methods: Standard User Login, Guest Login, and QR Login. After selecting the appropriate method, each method is handled separately. The user authentication component manages the states for user login, registration, and authentication. It sends login requests to the backend and stores the received JWT token for subsequent authenticated requests. This ensures secure access to the application's features and personal user data.

2. **Home Page**

   The home page, accessible immediately after logging in, serves as the central hub for the IntelliCart application, offering users a comprehensive overview of available products and categories. It features a sidebar for easy navigation, a prominently placed search bar for quick product lookup, and a section showcasing featured products. The sidebar provides quick access to various sections of the application, such as "Suggested," "Shopping List," "Coupons," "Settings," and "Help," each represented by an intuitive icon and label. The search bar enables users to swiftly locate specific products, enhancing the user experience by reducing the need to browse through multiple categories. The featured products section highlights popular or recommended items, encouraging users to explore and discover new products. This integrated layout ensures users can efficiently navigate, search for products, and manage their shopping experience with ease.

3. **Sidebar Navigation**

   The sidebar navigation component provides easy access to various sections of the application by leveraging React Router's NavLink component, enhancing the user experience by offering a structured and intuitive navigation menu. The sidebar includes links to the following sections, each represented by an icon and a label for quick identification:

   - **Suggested:** Recommends products based on offers.

   - **Shopping List:** Allows users to view and manage their shopping list.

   - **Coupons:** Displays available coupons and discounts that users can apply to their purchases.

   - **Settings:** Provides access to user settings and preferences.

   - **Help:** Offers help and support options for users who need assistance.

4. **Update Component**

   The update component is a crucial part of the front-end interface, designed to always appear for the user whenever changes occur in the shopping cart. This component provides real-time feedback to the user, ensuring they are immediately informed of any additions, removals, or updates to the items in their cart.

3.6.3 Integration with Backend

1. **API Requests**

   The front-end application communicates with the backend through various API requests to perform operations such as fetching product data, adding items to the cart, and processing payments. Axios is used to make HTTP requests to the backend endpoints. The responses are then used to update the application state managed by Redux.

2. **WebSocket Integration**

   Real-time updates were achieved by integrating WebSocket events, which provide immediate feedback on cart operations and scanning errors. Each user

communicates via WebSocket based on their unique token. The front end listens for WebSocket events emitted by the backend and updates the UI accordingly. This enhances user engagement and ensures the shopping cart's accuracy, reducing the likelihood of discrepancies between the user's actions and the cart's displayed contents.

Chapter 4

System Results and Evaluation

Contents

4.1 Backend Results

The Palestine IntelliCart project's backend was developed to guarantee reliable and secure interaction among the database and user interface. The key results of the backend development are highlighted in this section, with a focus on the effectiveness, security, and real-time capabilities offered by the WebSocket integration and deployed API endpoints.

4.1.1 Secure User Authentication and Authorization

- **User Login**

  The system generates JWT tokens upon user login, which are used to authenticate subsequent requests. Users can log in using their email and password, or create a temporary guest session without providing an email or password. QR code authentication is also available for a secure and convenient method. The system ensures user authentication and security. In the following, the postman tests for the three login methods:

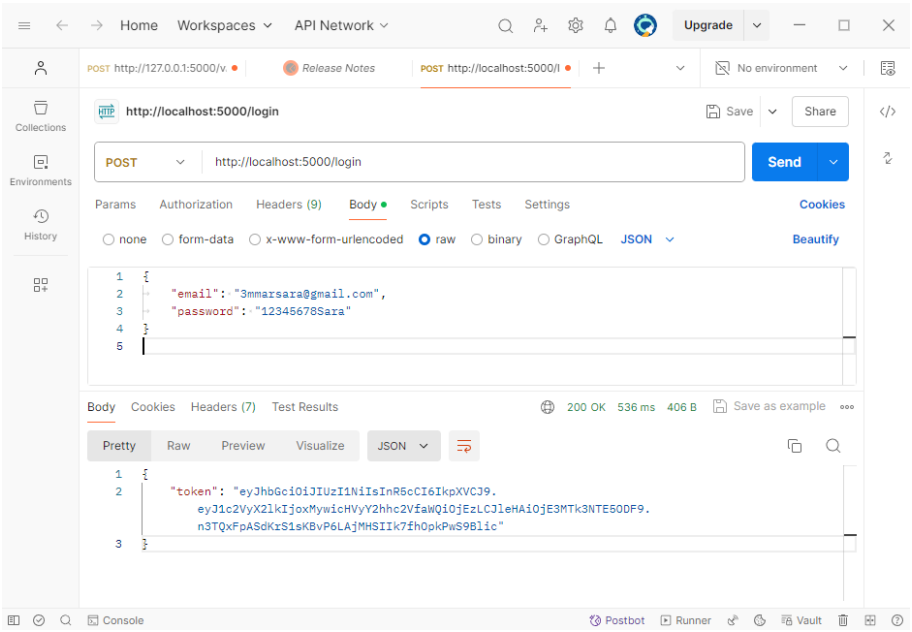


Figure 4.1: Standard User Login

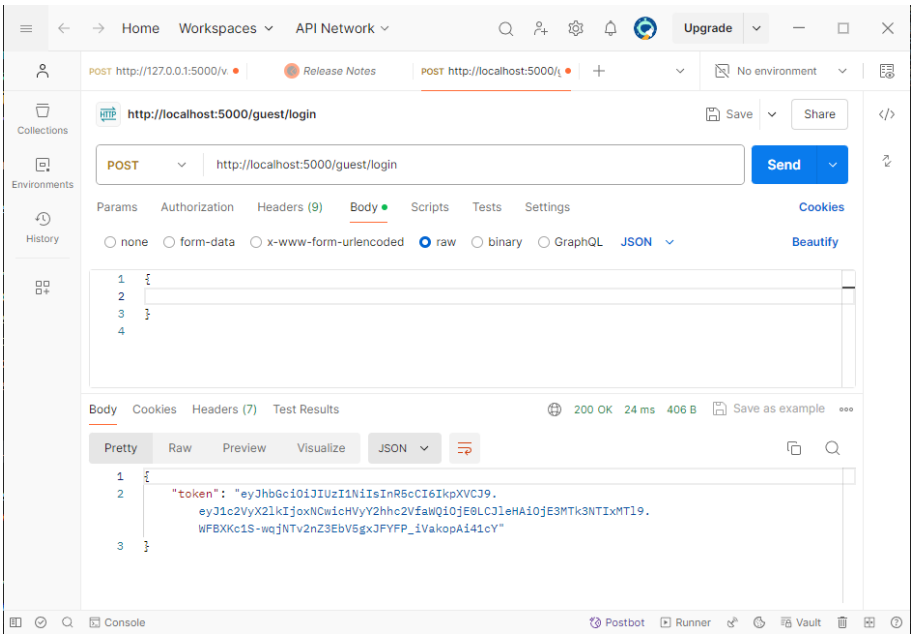


Figure 4.2: Guest Login

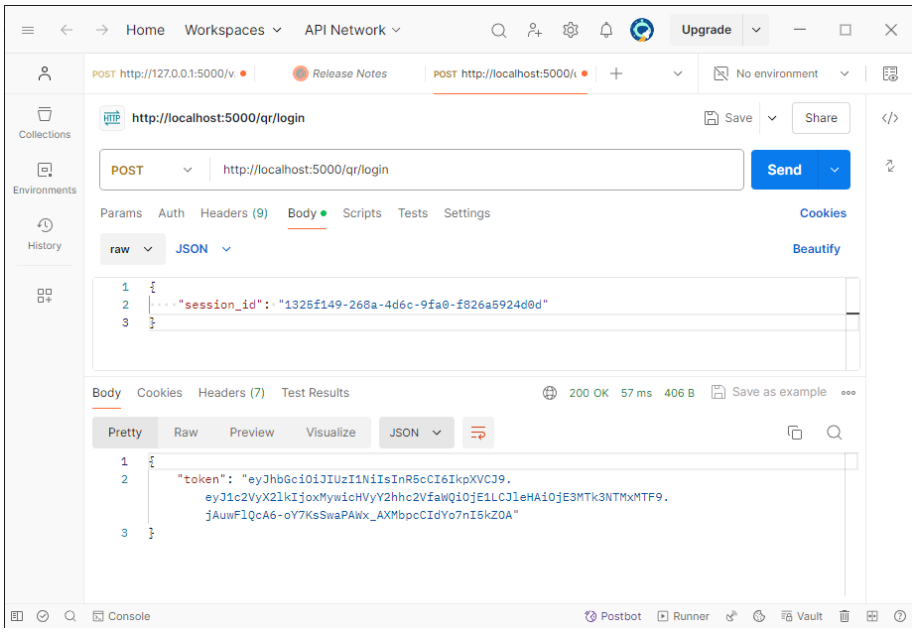


Figure 4.3: QR Login

- **Get User Information**

  The token was validated by each protected endpoint to ensure only authorized users could access or modify resources, and it was decoded using a secret key for verification of validity and expiration. The following is a Postman test to get user information:

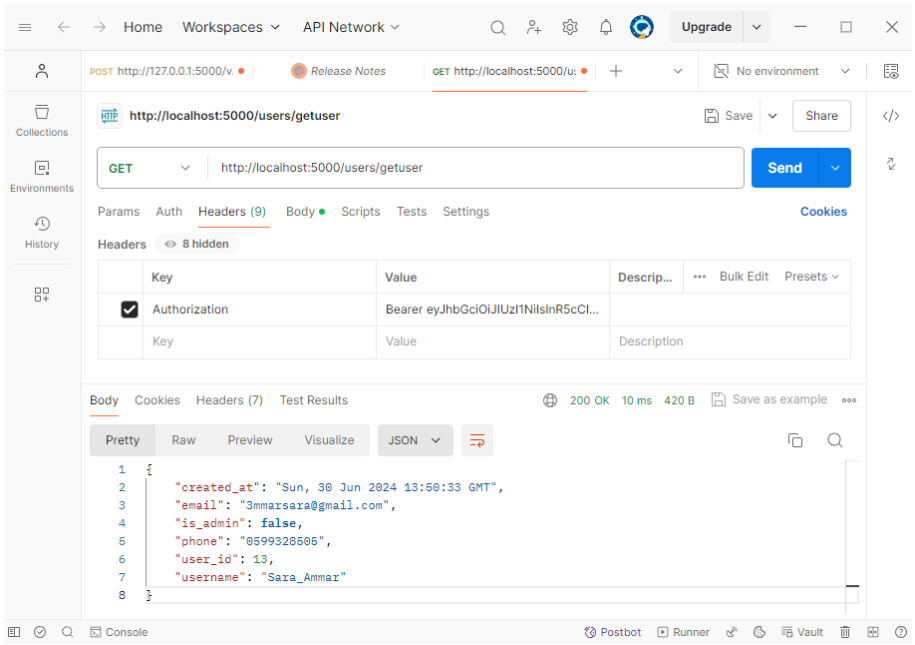


Figure 4.4: Get User Information

4.1.2 Efficient API Endpoints

- **Product Management**

  The API was provided with comprehensive product management capabilities, including adding, updating, retrieving, and deleting products. An example of using Postman to add a product is provided:

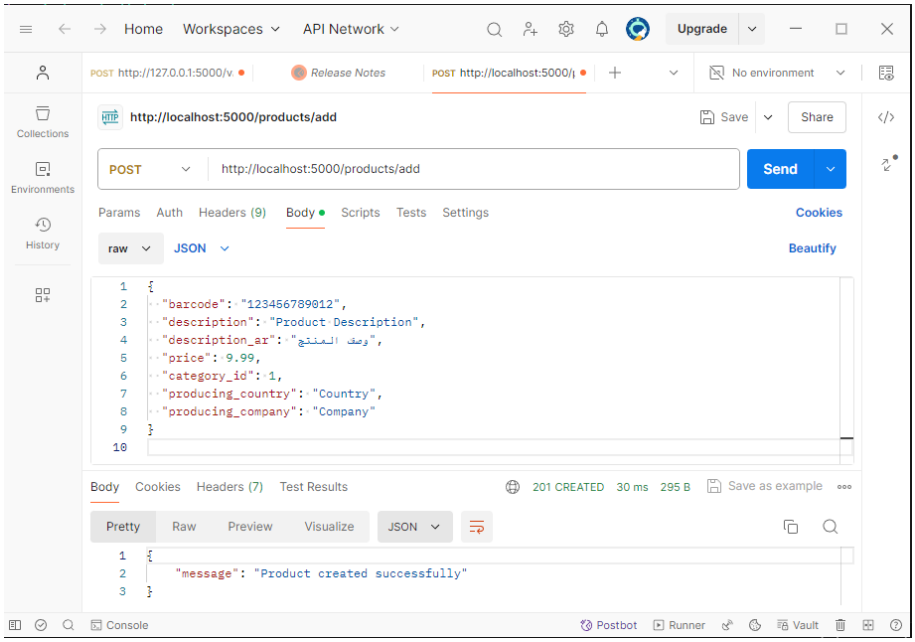


Figure 4.5: Add product

- **Cart Operations**

  Users were allowed to manage their shopping carts efficiently through the API, with endpoints for adding and removing products and viewing cart details. These endpoints interact with purchase details, ensuring accurate cart management, as demonstrated in the examples of using Postman:

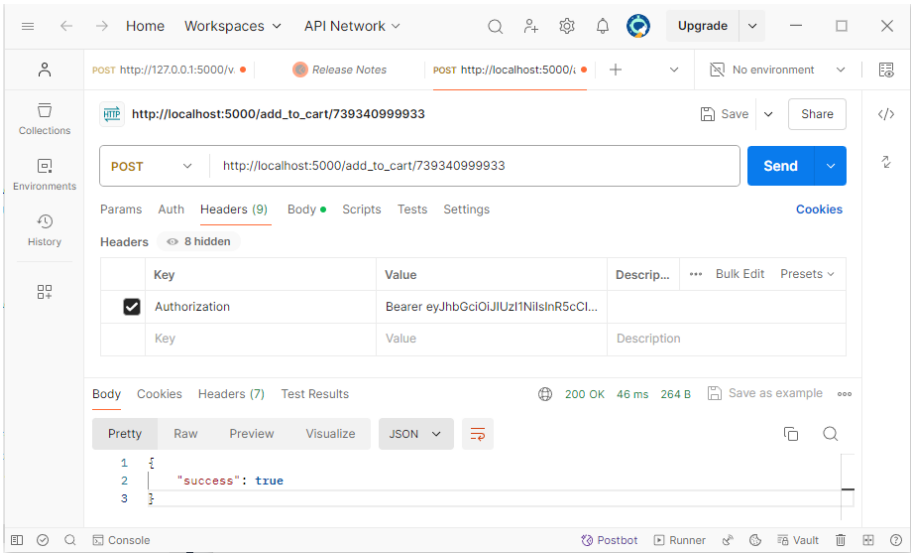


Figure 4.6: Add Product to Cart

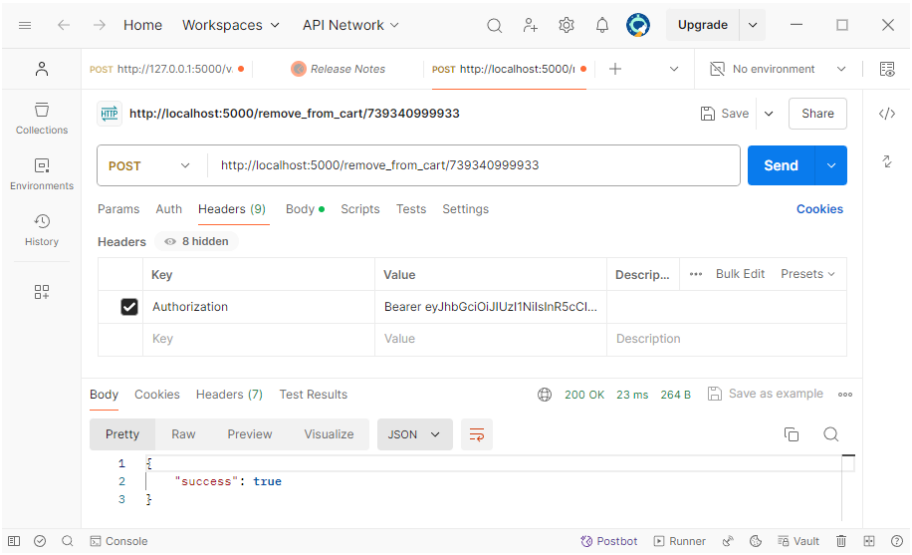


Figure 4.7: Remove Product From Cart

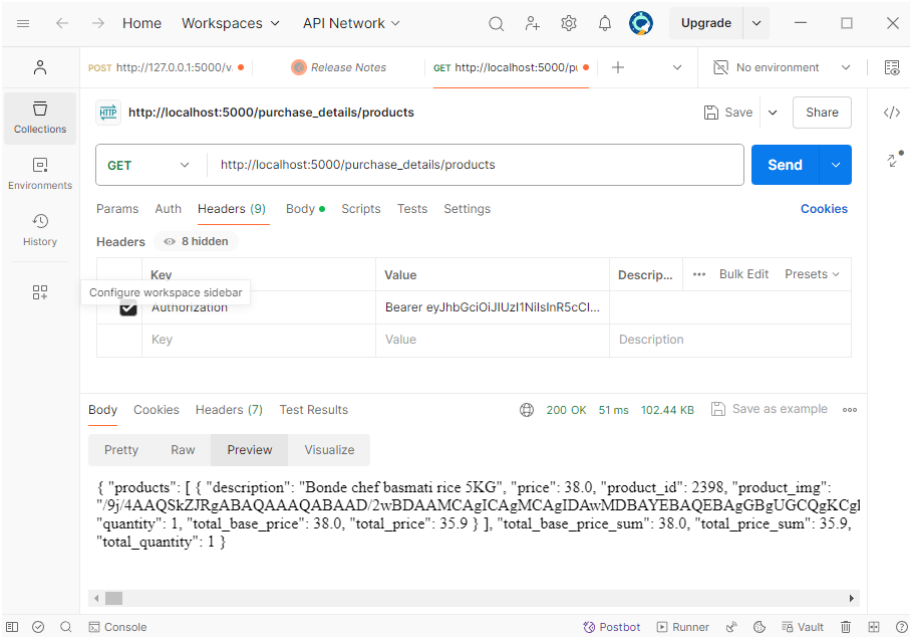


Figure 4.8: Get Cart Details

- **Coupon Application**

  In the system, users were allowed to add and remove coupons to/from their purchases, ensuring flexibility in applying discounts. Examples of using Postman to add and remove coupons are provided:

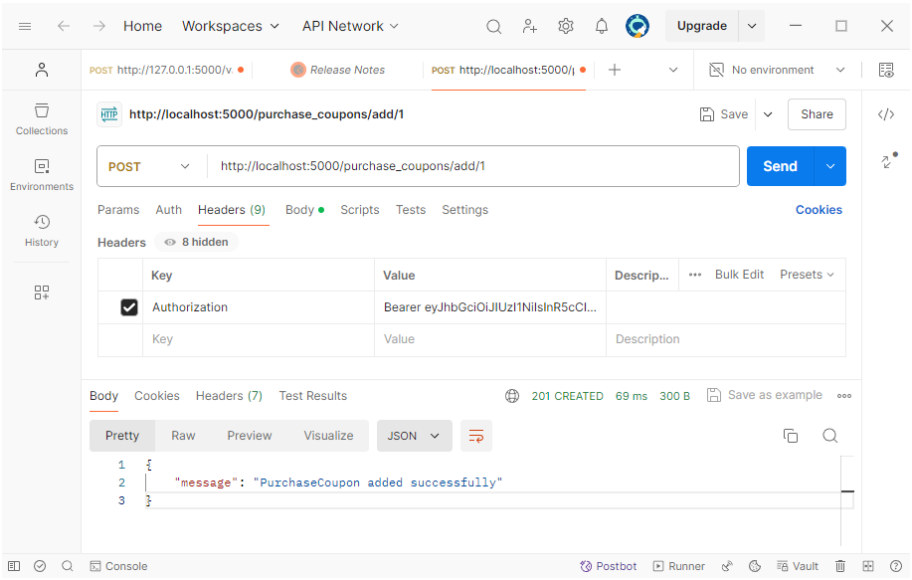


Figure 4.9: Add Coupon to the purchases

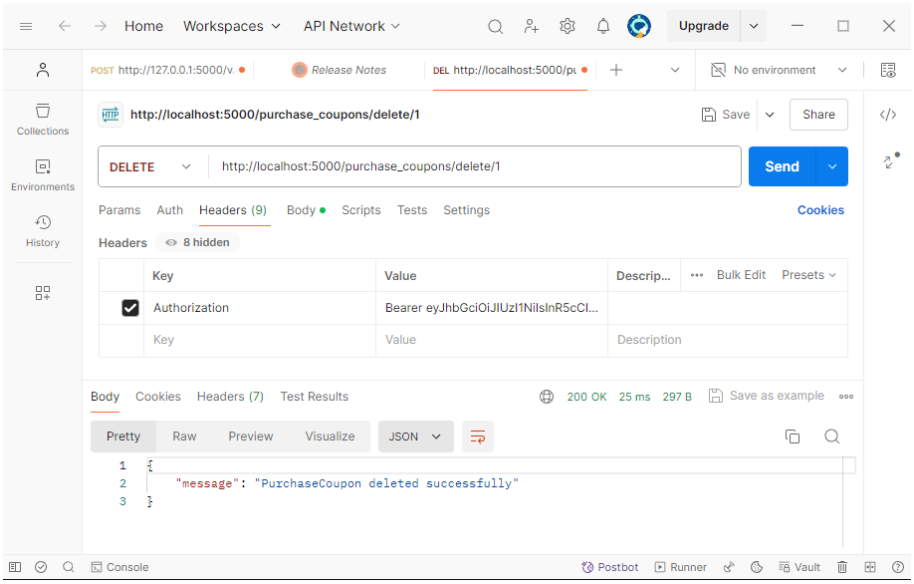


Figure 4.10: Remove Coupon from the purchases

- **Offer Management**

  The creation of special offers for products was allowed by the API, including discounts for specific periods. The endpoint allowed for the creation of offers with details like description, discount amount, validity period, and minimum quantity. Discounts are applied directly to the product if valid. An example of adding an offer is provided:

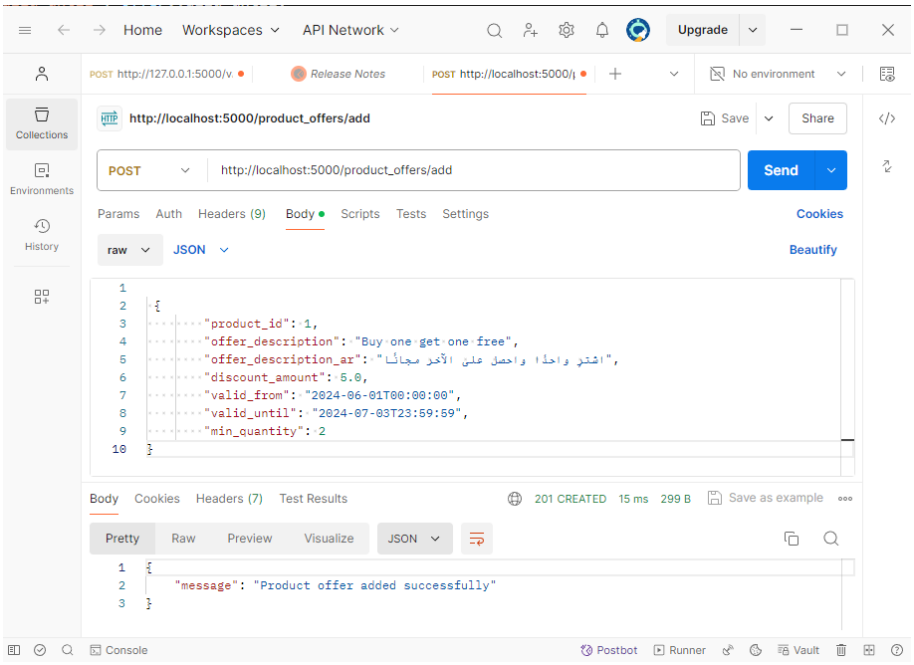


Figure 4.11: Add Offer on Product

4.1.3 Real-Time Cart Updates

WebSocket events were implemented to provide real-time updates for cart operations and scanning errors. SocketIO was used by the backend to emit events when cart changes occur or scanning errors are logged. This integration ensures that any changes made to the cart, such as adding or removing products, are instantly reflected on the user's interface. Additionally, any errors encountered during product scanning are logged and communicated instantly, allowing for prompt resolution. The following APIs utilize WebSocket events to enhance real-time interactivity and feedback for users:

- **Add to Cart:** When a product is added to the cart, the system emits a WebSocket event with the details of the added product, including its description, price, quantity, and image. This ensures that the user's cart is updated in real-time without requiring a page refresh.

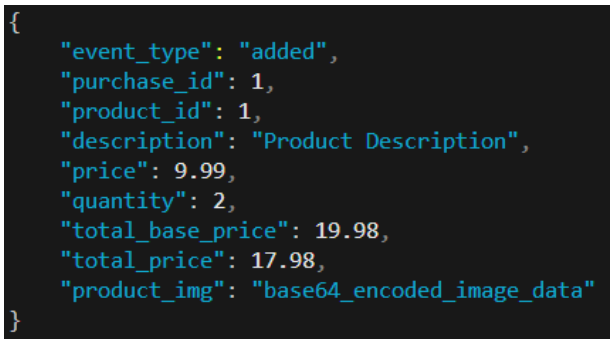


Figure 4.12: The Emitted Socket Event for the Add Event

- **Remove from Cart:** When a product is removed from the cart, a WebSocket event is emitted with the updated details of the cart. This includes the remaining quantity of the product, the updated total base price, and the total price. This real-time update helps users see the immediate impact of their actions on their cart.

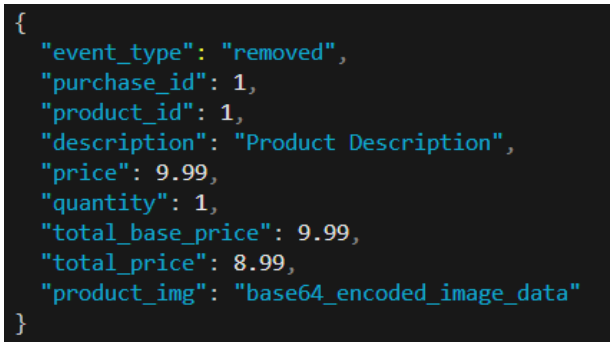


Figure 4.13: The Emitted Socket Event for the Remove Event

- **Log Scanning Error:** When a scanning error occurs, such as an unscanned item being added or removed, the system logs this error and emits a WebSocket event with the error details. This allows for immediate notification and resolution of scanning issues, improving the accuracy of the cart's contents.

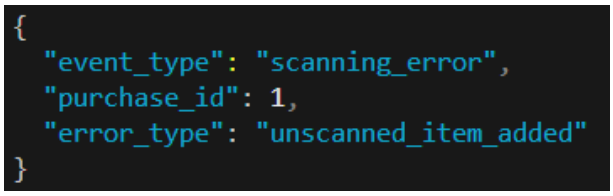


Figure 4.14: The Emitted Socket Event for the Log Scanning Error Event

- **Resolve Scanning Error:** When a scanning error is resolved, a WebSocket event is emitted to update the system and the user's interface about the resolution. This ensures that any previously reported errors are promptly addressed and cleared from the system.

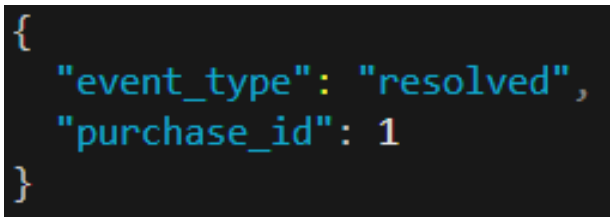


Figure 4.15: The Emitted Socket Event for the Resolve Scanning Error Event

- **Create Payment:** Upon creating a payment, a WebSocket event is emitted to confirm the payment status in real-time. This includes the details of the purchase, the amount paid, the payment method used, and the payment ID. This immediate feedback ensures users are promptly notified of successful transactions.

```
{
  "event_type": "payment_completed",
  "purchase_id": 1,
  "amount": 44.95,
  "method_id": 1,
  "payment_id": 123
}
```


Figure 4.16: The Emitted Socket Event for the Create Payment Event

4.2 Tracking Results and AI model Evaluation

YoloV8s model was chosen to track objects in the cart using a deep sort tracking algorithm. The first step was to train the model itself on the detecting task, then the next part was to test it on the tracking task and configure it according to the needs of Palestine IntelliCart .

4.2.1 Model Training

The model was trained and tested on several configurations and multiple stages to achieve the best possible results. The model was trained on a dataset of 80k images, each image contains instances of several types of grocery to train the model on various shapes and possibilities. During the training the YoloV8s achieved the best results thus it is used as the tracking mind. The following results show retraining our best model on new data for 10 batches.

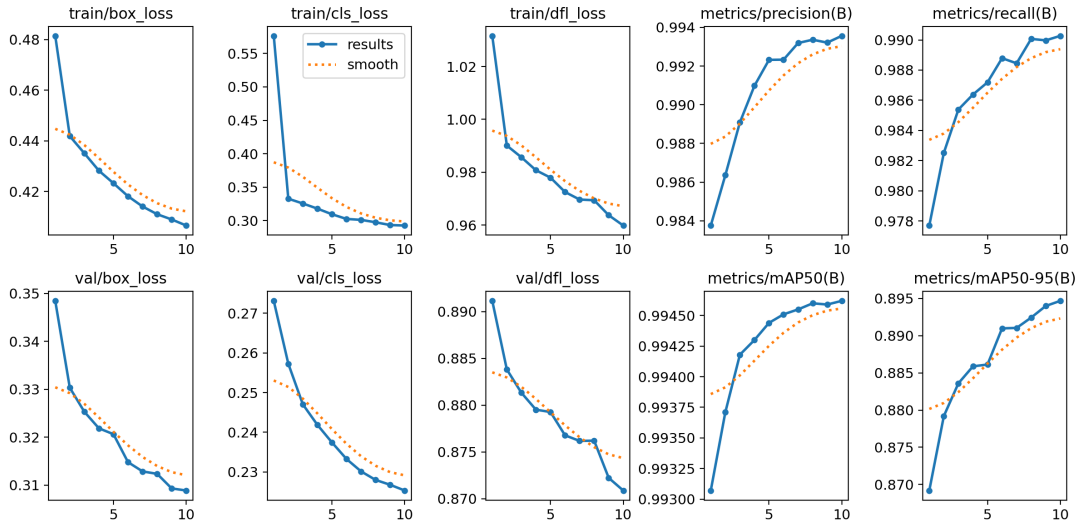


Figure 4.17: YoloV8s model last training.

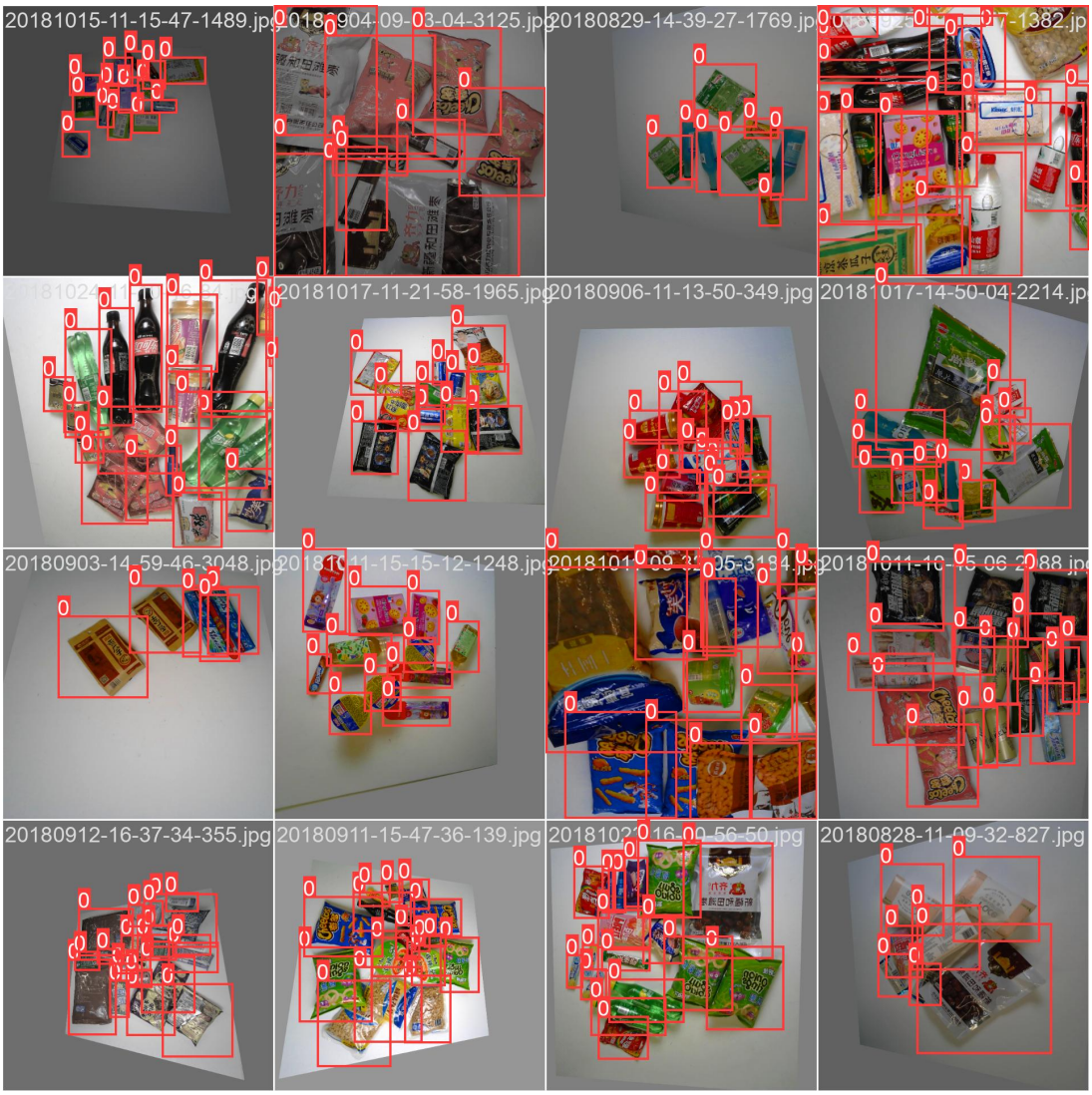


Figure 4.18: Training results.

After training the model on the general dataset to acquire knowledge about product shapes and possible configuration, the best model was retrained on a cart-related batch to eliminate background error and get more accurate results. The following shows the validation set of training the data.

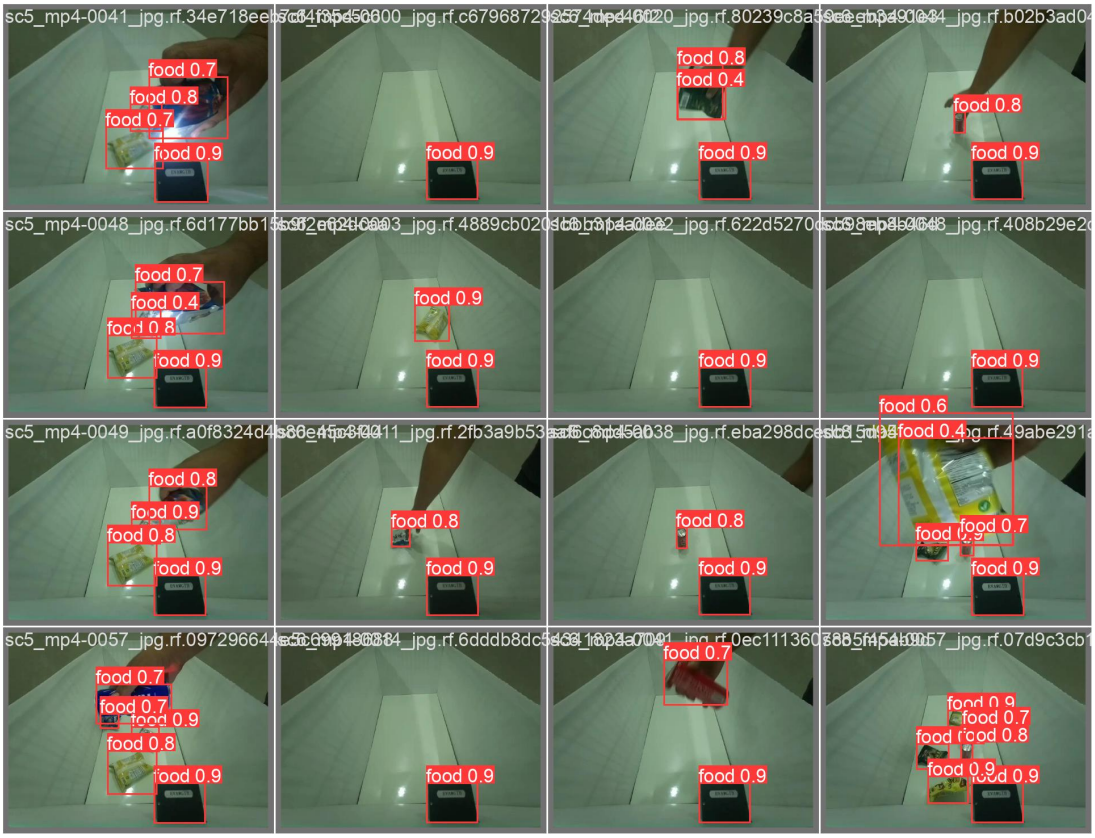


Figure 4.19: Cart Validation set.

The results show that the model was capable of accurately detecting products in the camera view which results in an accurate model.

4.2.2 Tracking Testing

Tracking items were coded to apply the model-detecting task on each frame and then deep sort tracking was applied to the results to update each track of detected objects, as a result, each object has a track associated with it providing information about the track including the barcode if scanned. The tracking controls which objects are added to the shopping list and which are considered unscanned items according to their movement and feature history. The following scenario shows a tracking path result.

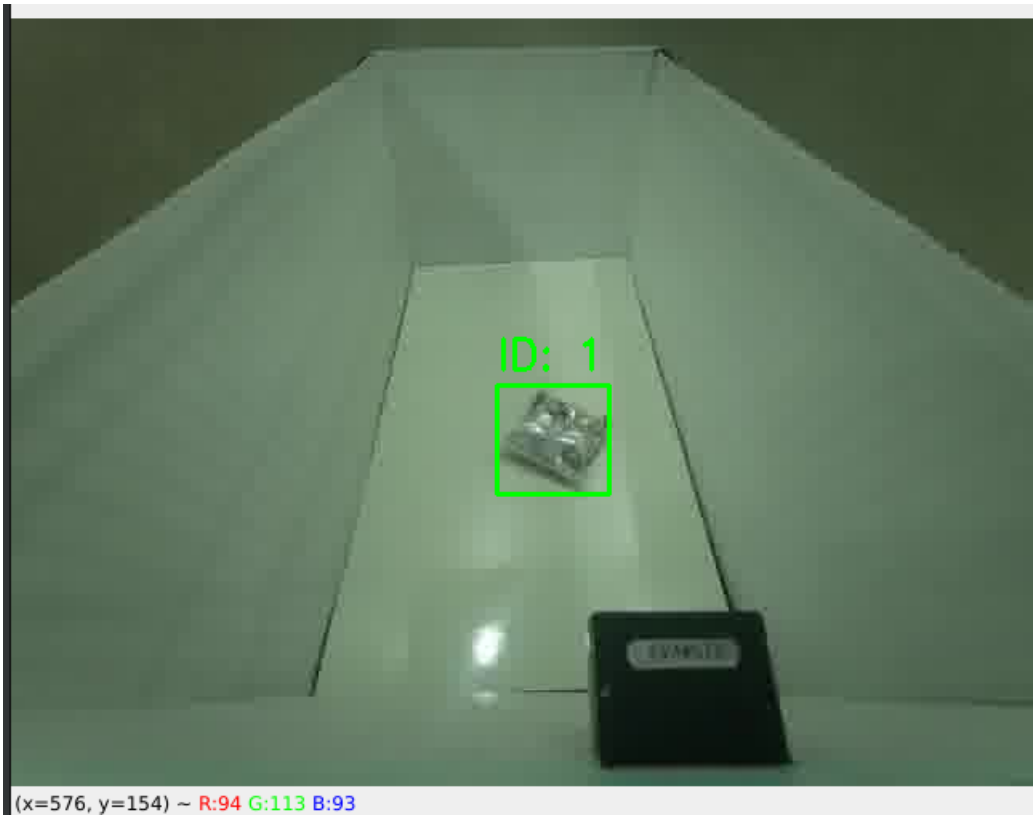


Figure 4.20: Tracking scenario showing a single item, frame 1.

Figure 4.21: Tracking scenario showing an item entering the cart, frame 2.

Figure 4.22: Tracking scenario showing an item getting scanned, frame 3.

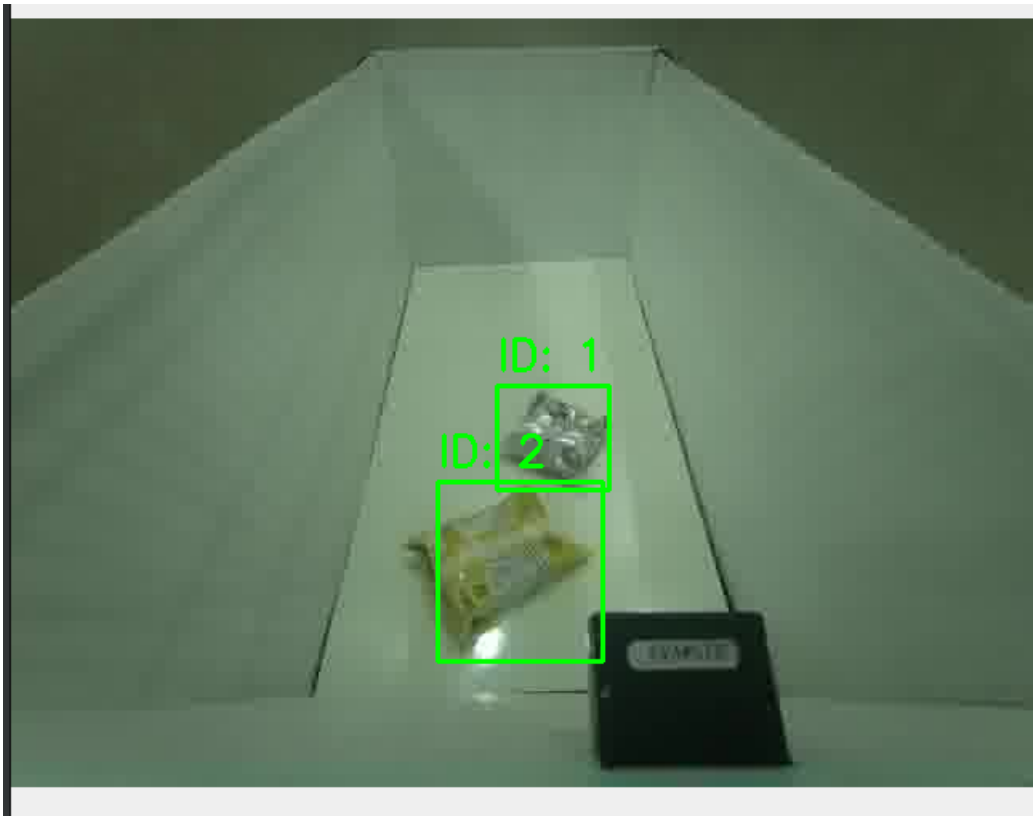


Figure 4.23: Tracking scenario showing two items in the cart, frame 4.

4.3 Overall Results

This section provides a comprehensive walk-through of the process of using the cart with a simple shopping process. The cart was polished and rehabilitated from an old, rusty cart to a new, smart, and fully integrated cart with a full back-end system working as the backbone of the intellicart project and a front end providing a fully interactive user experience.

Figure 4.24: Intellicart Final Result.

When a user interacts with the cart, the following screen is activated, showing a welcome screen welcoming the user and showing the Palestine IntelliCart logo.

Figure 4.25: Welcome screen

After an assigned time of about a few seconds, providing enough time for the system initiation protocol, the signing-in screen opens and views three options. For this walkthrough, the guest option is chosen for direct entry.

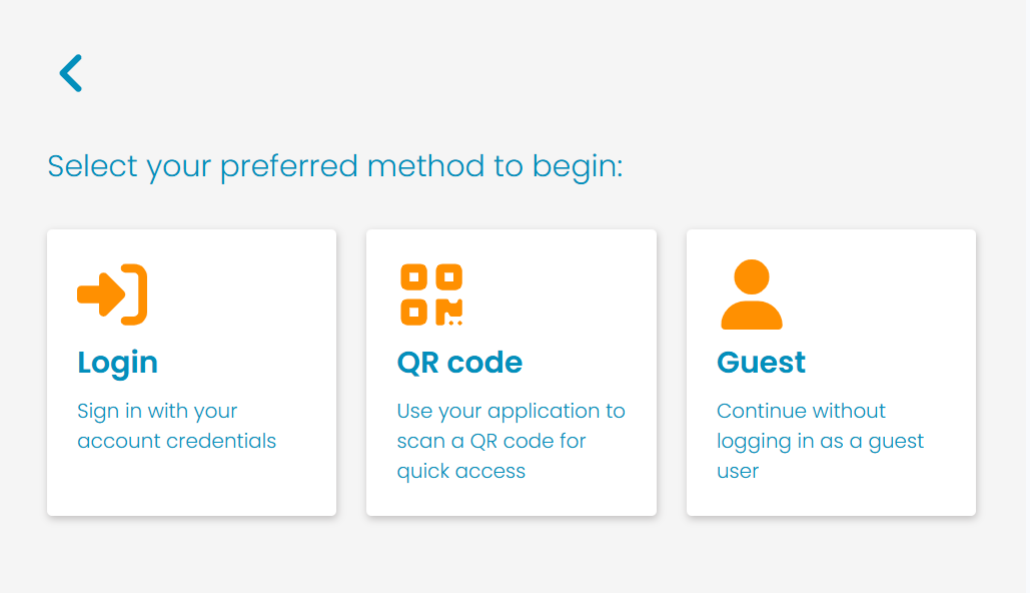


Figure 4.26: Login Page

The following figure shows the main page, providing various options for navigating through the shopping.

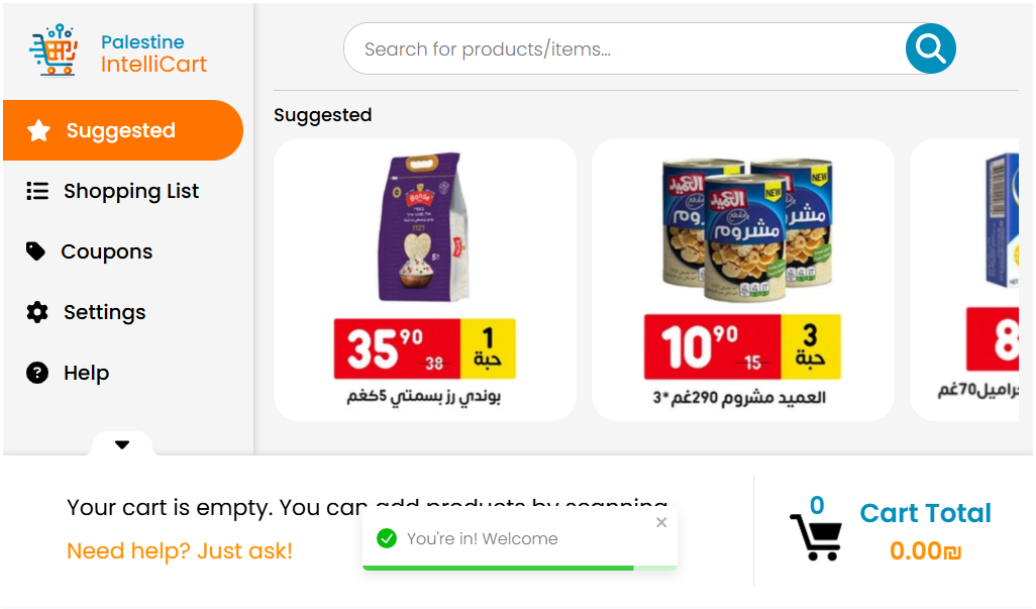


Figure 4.27: Main Page Result

The user decides to look for some snacks, so the user searches for tiger chip flavors.

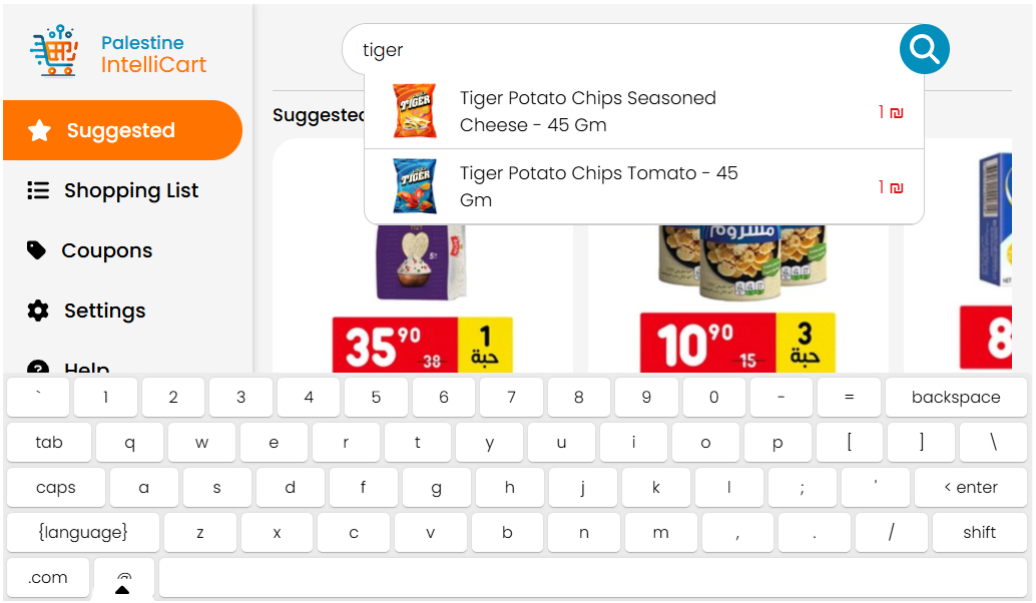


Figure 4.28: Search result.

The user chooses one of the flavors to examine the features before adding it to the cart.

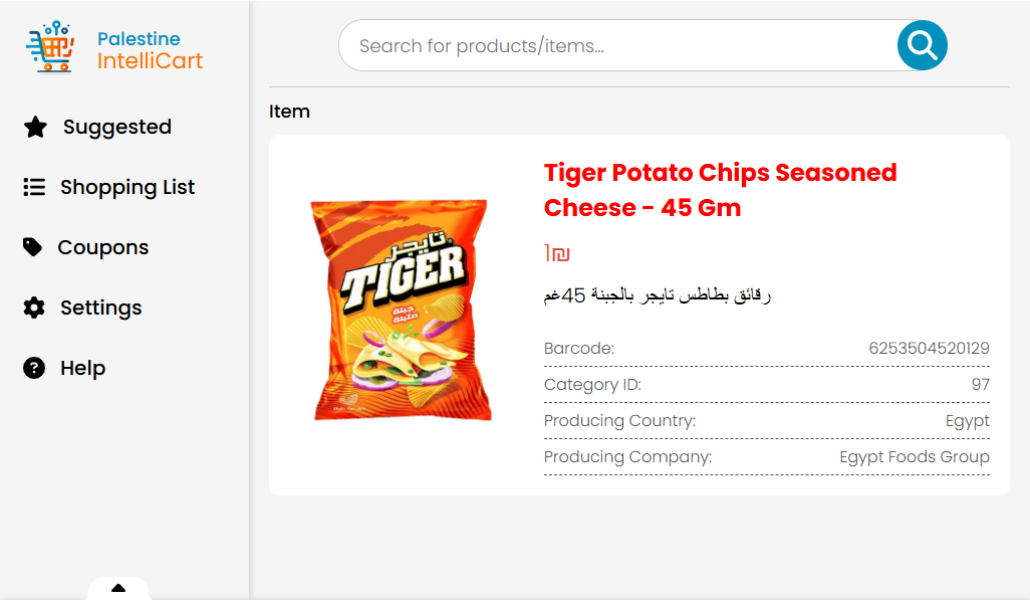


Figure 4.29: Enter Caption

Overall scenario frame 1 shows a single registered item inside the cart after scanning it, thus adding it to the list.

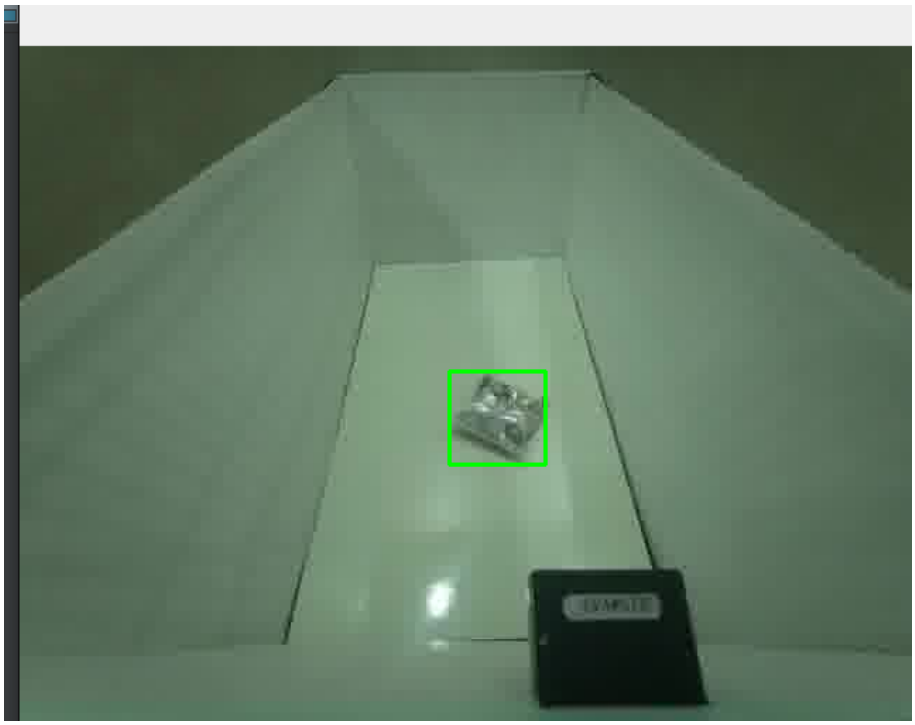


Figure 4.30: Overall scenario, frame 1.

The front end on the screen of the cart shows the response of adding the item to the shopping list and updating the values.

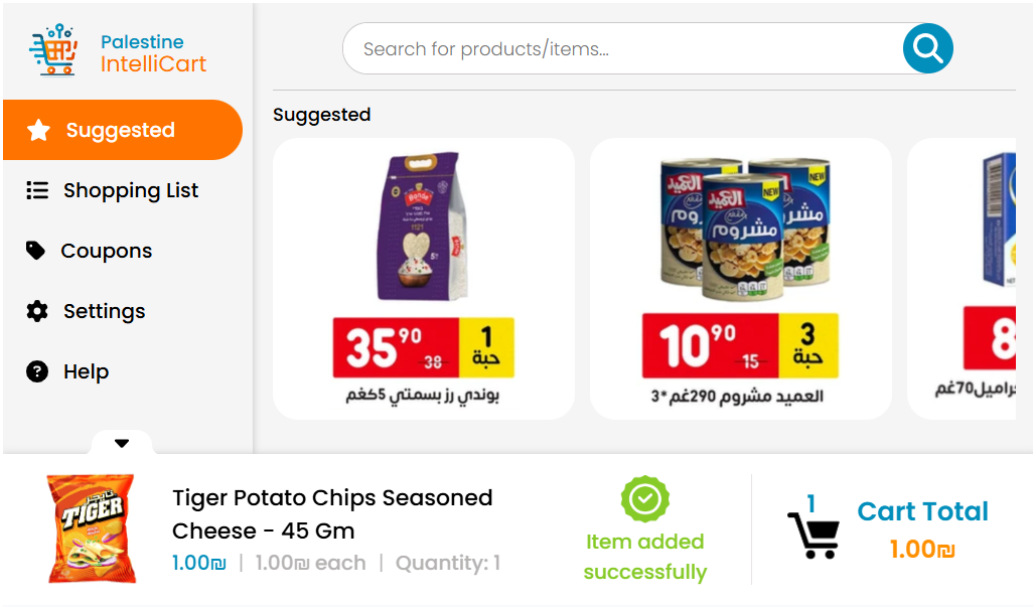


Figure 4.31: First item added to the list.

Overall scenario frame 2 shows a new item entering the cart without scanning the item, thus causing a warning screen that should be fixed.

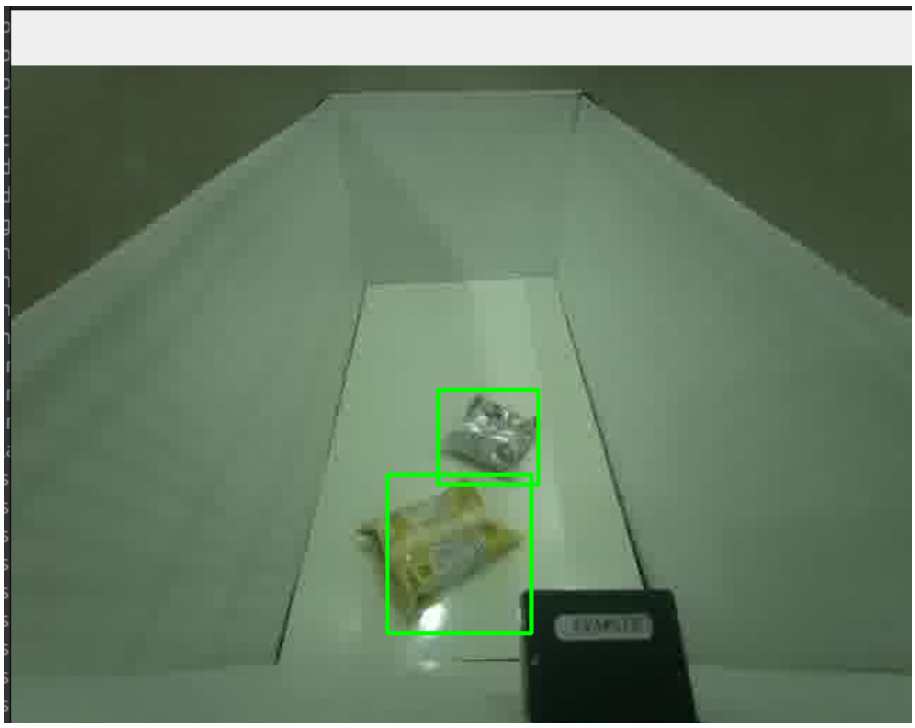


Figure 4.32: Overall scenario, frame 2.

The following figure shows a warning as a result of the wrong usage of adding an item to the cart without scanning its barcode. The warning screen stays on until the mistake is resolved.

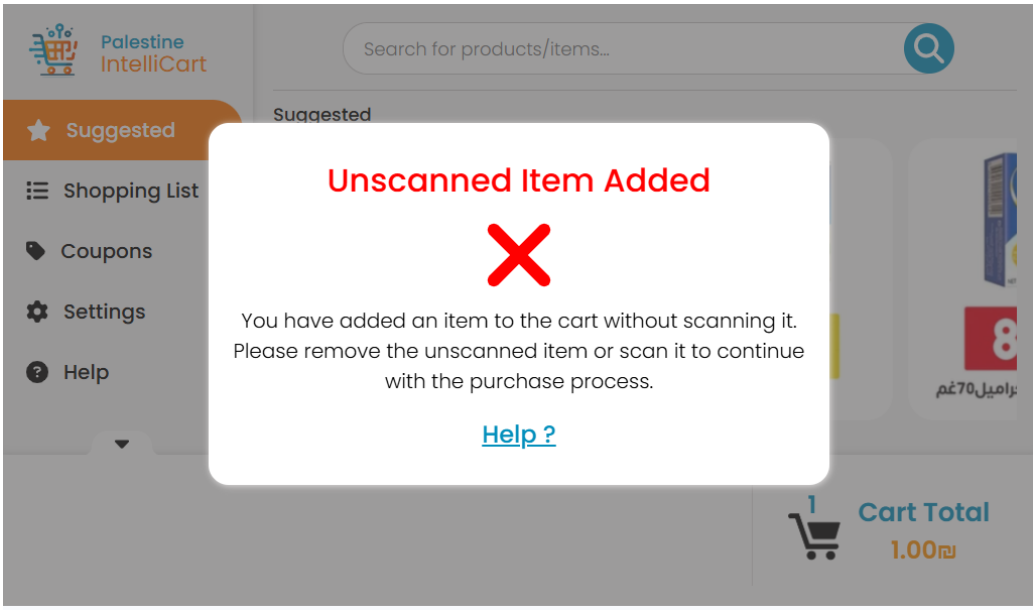


Figure 4.33: Unscanned item warning.

Overall scenario frame 3 shows the unscanned item being scanned, thus removing the warning screen and adding the item to the shopping list.

Figure 4.34: Overall scenario, frame 3.

The following figure shows the warning from the previous screen disappearing and the item being added to the shopping list.

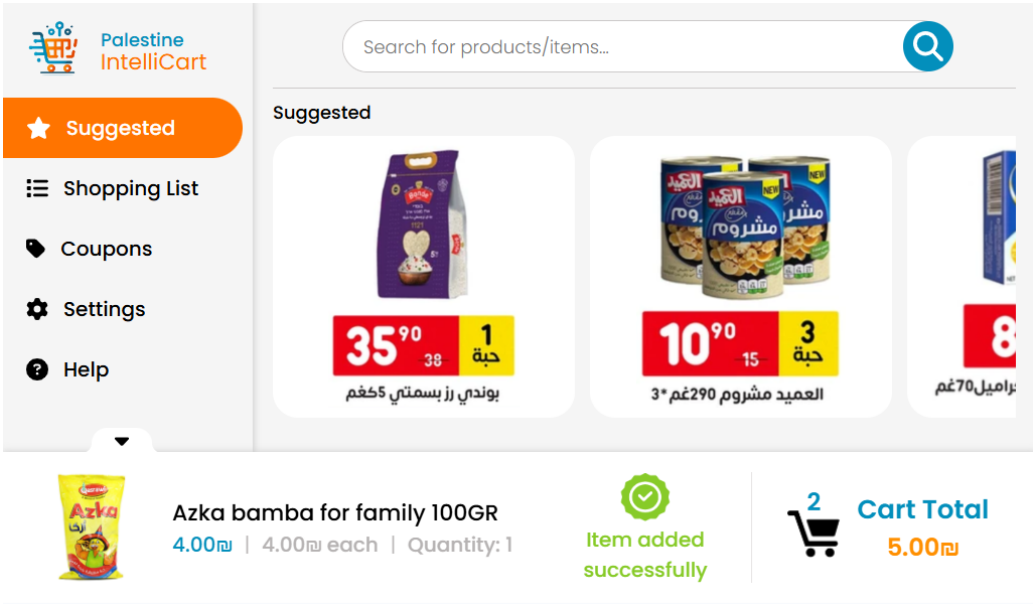


Figure 4.35: Adding second item to shopping list.

Overall scenario frame 4 with all previously scanned items.

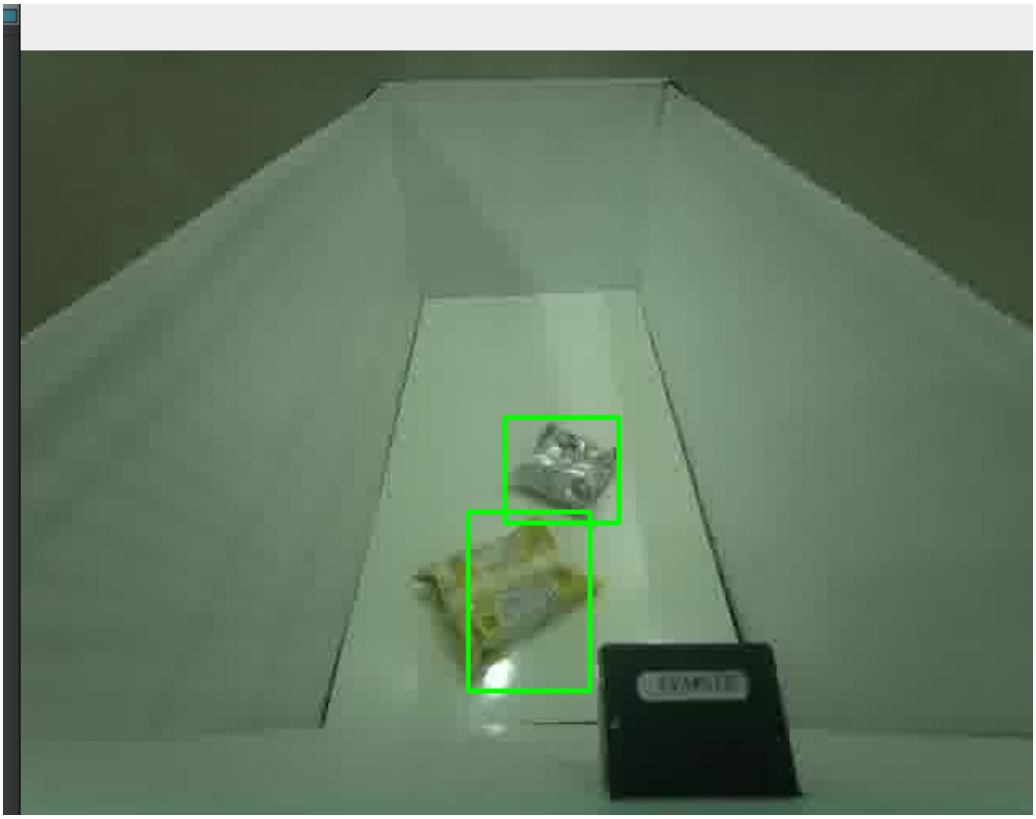


Figure 4.36: Overall scenario, frame 4.

Overall scenario frame 5 with a new item being scanned.

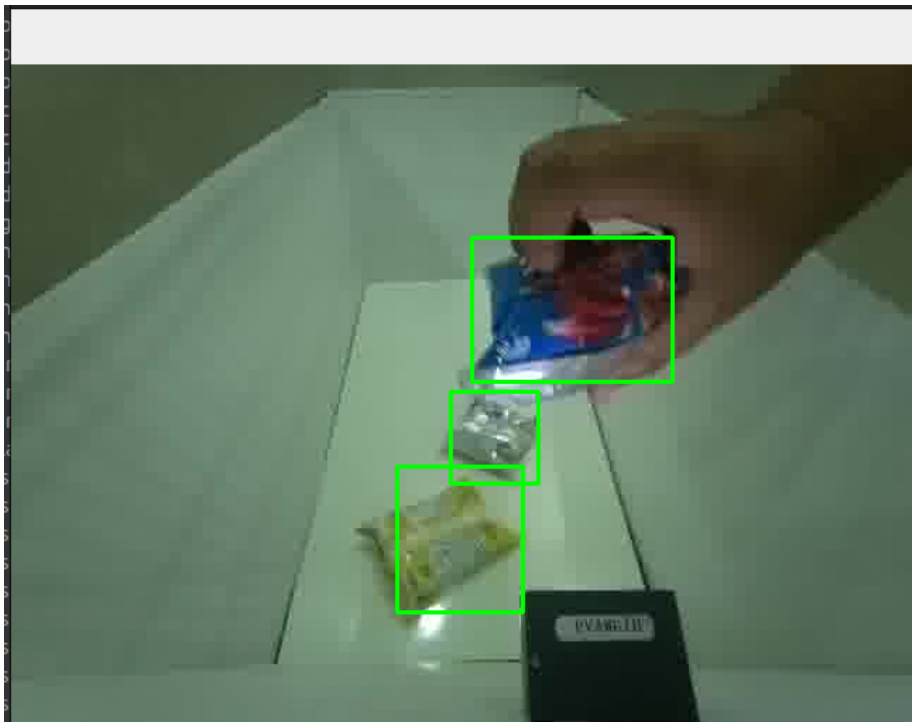


Figure 4.37: Overall scenario, frame 5.

The front end on the screen of the cart shows the response of adding the 3rd item to the shopping list and updating the values.

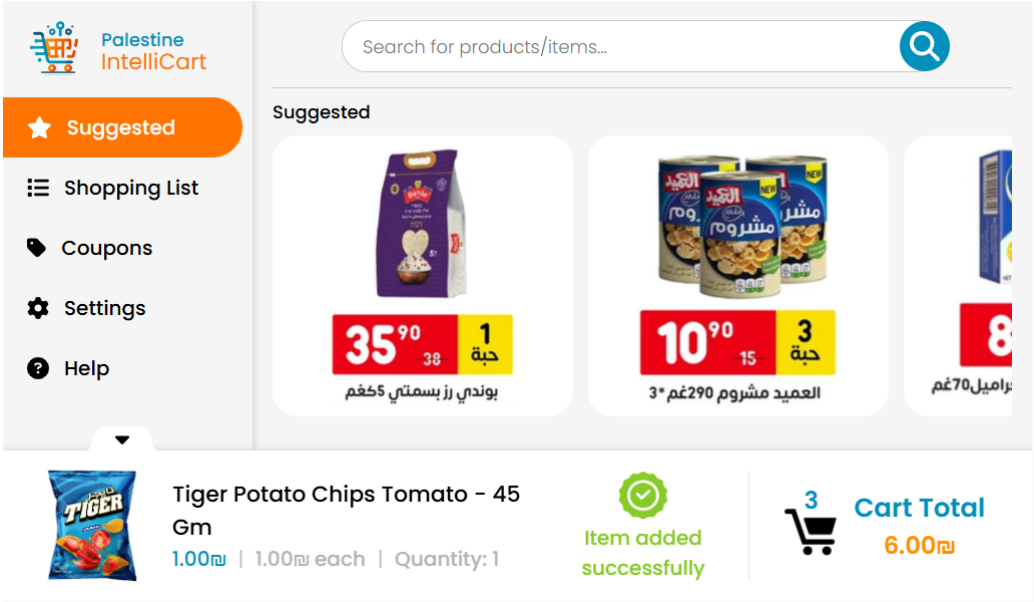


Figure 4.38: Adding 3rd item.

Overall scenario frame 6 with all previous items scanned.

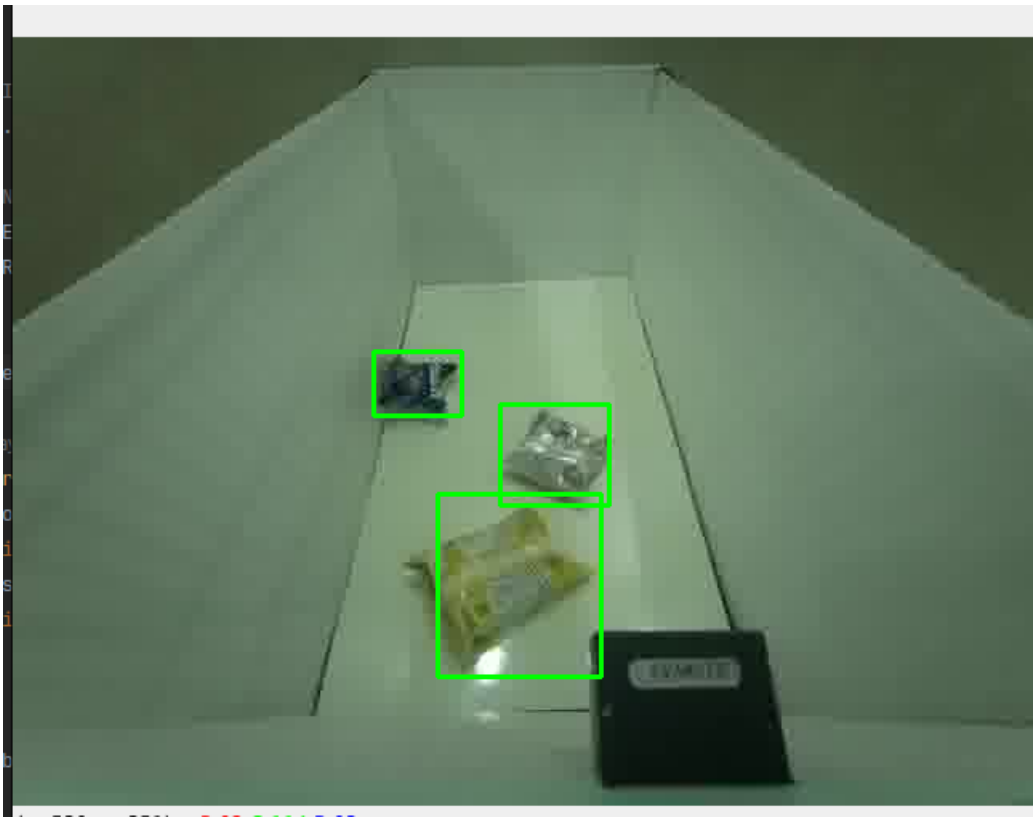


Figure 4.39: Overall scenario frame 6.

The following figure shows the final shopping list for this shopping journey and provides the user with the ability to check out.

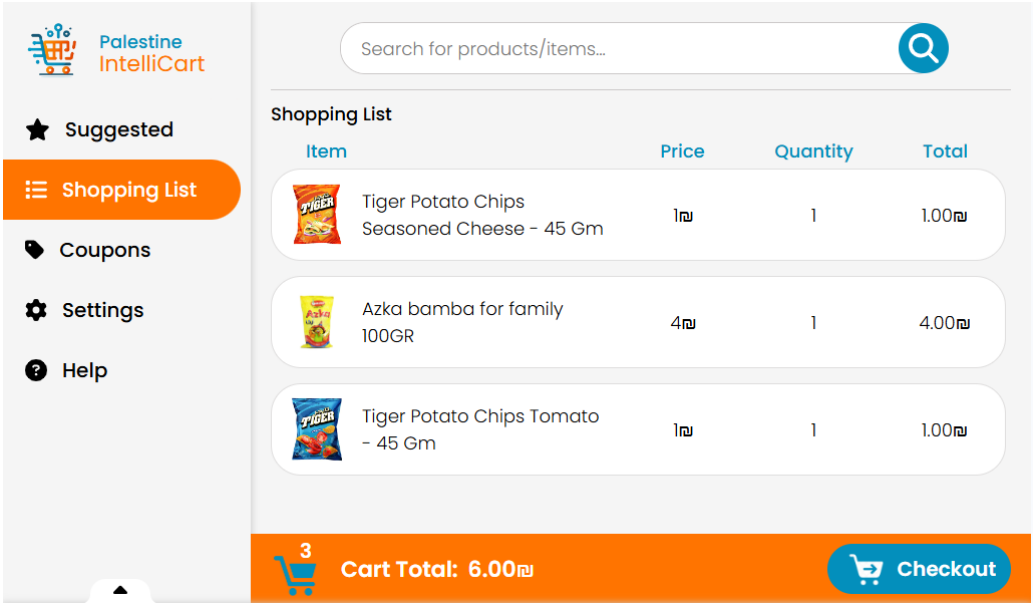


Figure 4.40: Shopping list result.

The following figure shows the result of applying one of the coupons available to the current shopping procedure and as a result, the checkout value is reduced.

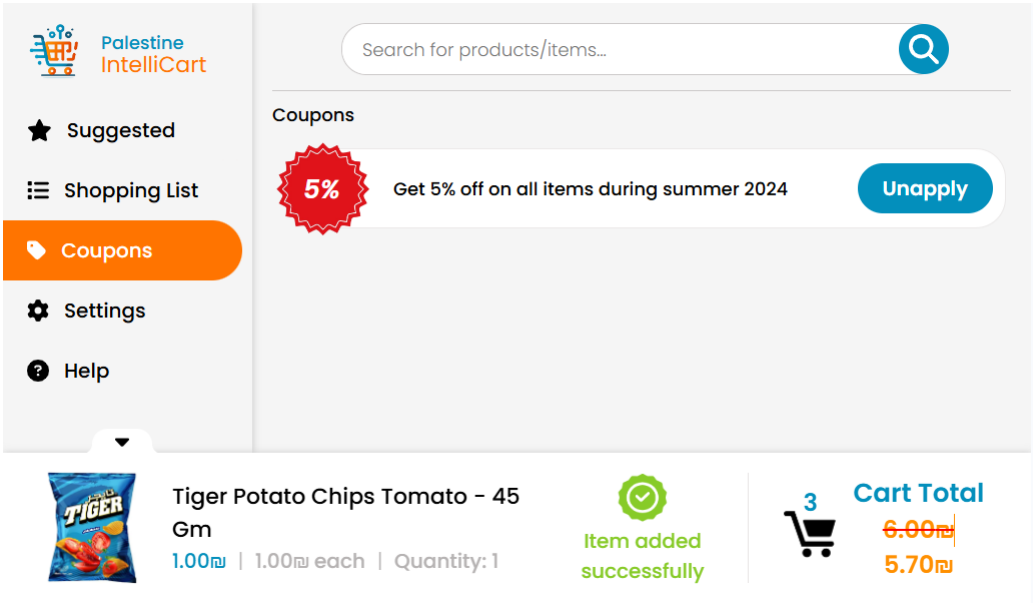


Figure 4.41: Coupon applied result

The user presses the checkout button, and thus the following screen appears, providing the user with the option of paying cash or via a visa.

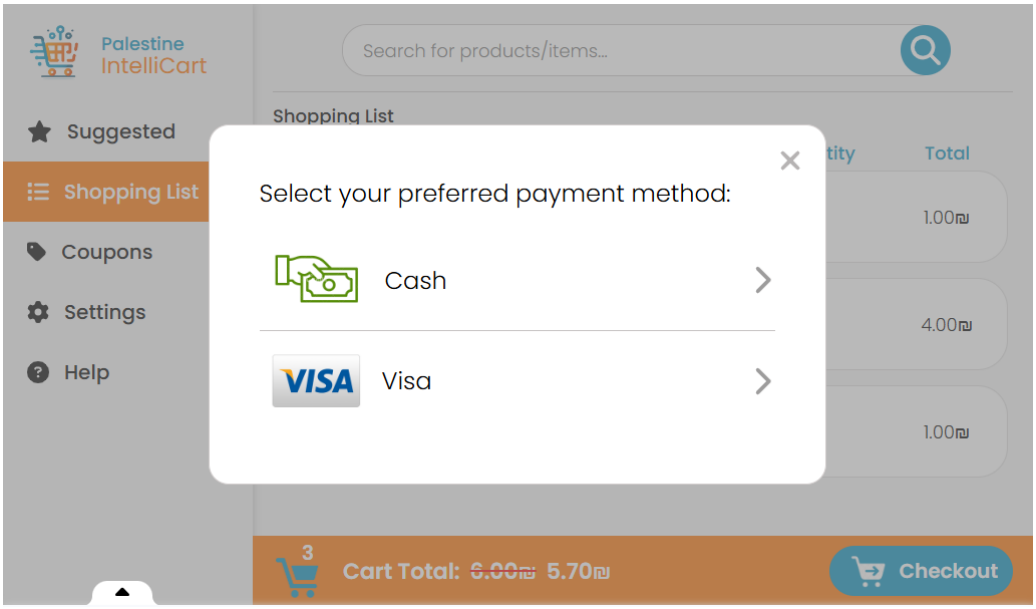


Figure 4.42: Checkout options

The user chooses to pay in cash, therefore the cashier scans the barcode that links the information of the shopping procedure to the cashier's register, and the user pays the cashier to complete the payment.

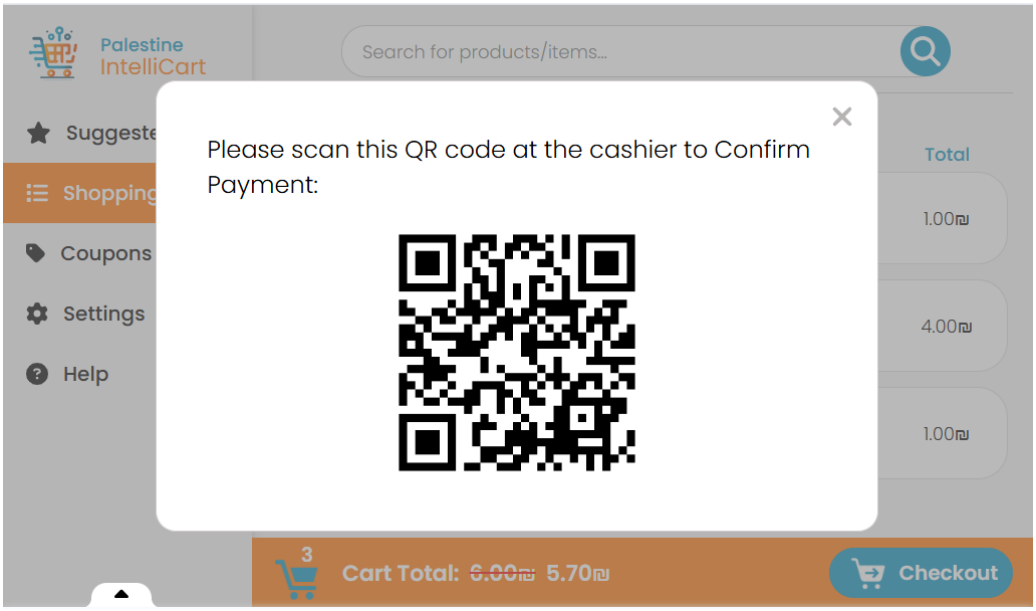


Figure 4.43: QR payment code.
this

After the payment is issued and accepted by the trusted authorities, the shopping journey concludes, and the user can take the groceries and leave the market.

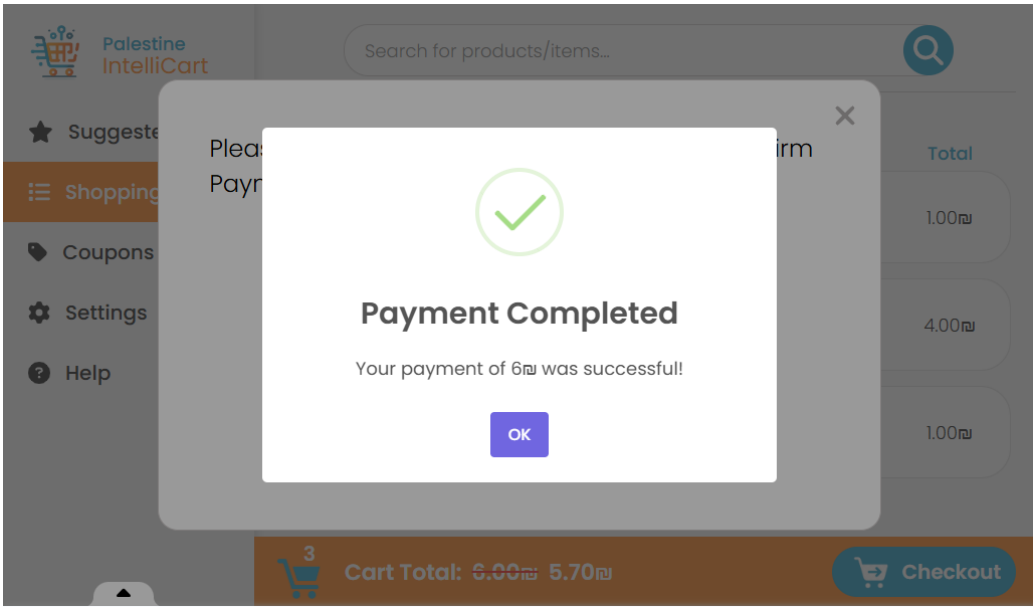


Figure 4.44: Completed Payment notification.

Chapter 5

Conclusion and Future Work

In conclusion, the Palestine IntelliCart was designed, implemented, and tested on a few simple and small scenarios. The AI model was implemented, trained, and tested on various data sets. The back end was implemented and tested on various complex and simple scenarios. The front end was designed and implemented to be user-friendly and easy to use.
As a result, the model performed satisfactorily, but it still requires further improvement. Our current model can be tricked by real users who may intend to cause harm. So, it still needs further improvement and covering of complex and tricky scenarios to be implemented in a real market.
Based on our ideas, 'Palestine IntelliCart ' has significant potential; thus, it would require extensive training and data collection to cover a broader range of complex and tricky scenarios.
The final step is to deploy the cart and conduct real-world testing after covering all tricky scenarios. We plan to coordinate with major stores like Al-Shini Extra malls to test Palestine IntelliCart within the mall as a promotional activity.

Bibliography

[1] A. Go, “Amazon.com: Amazon go: Stores,” 2023. Accessed: November 18, 2023.

[2] H. Vinny, “Alibaba opens cashier-free retail store in china.” Accessed: November 18, 2023.

[3] A. Jain, A. Bhola, S. Upadhyay, A. Singh, D. Kumar, and A. Jain, “Secure and Smart Trolley Shopping System based on IoT Module,” ., 12 2022. Accessed: November 20, 2023.

[4] P. Kalaivani, C. R. Rajan, and S. Sandhiya, “An automated billing system in shopping malls using Bascart,” *Bulletin of Scientific Research*, pp. 62–68, 5 2019. Accessed: November 19, 2023.

[5] C. Roopa and N. C. Reddy, “Research on smart shopping cart,” *International journal of scientific research in computer science, engineering and information technology*, pp. 359–363, 8 2020. Accessed: November 21, 2023.

[6] M. S. H. Chi, “Smart self-checkout carts based on deep learning for shopping activity recognition,” Accessed: November 24, 2023.

[7] N. Pradhan, R. K. Tyagi, and P. Nagpal, “BARCODE RECOGNITION TECHNIQUES: REVIEW APPLICATION,” *International journal of innovative research in computer science and technology*, vol. 9, 5 2021. Accessed: December 2, 2023.

[8] A. Kumar, A. Gupta, S. Balamurugan, S. Balaji, and R. Marimuthu, “Smart Shopping cart,” *2017 International conference on Microelectronic Devices, Circuits and Systems (ICMDCS)*, 8 2017. Accessed: November 26, 2023.

[9] S. Mekruksavanich, “Design and Implementation of the Smart Shopping Basket Based on IoT Technology,” *Design and Implementation of the Smart Shopping Basket Based on IoT Technology*, 10 2019. Accessed: December 2, 2023.

[10] Raspberry Pi Foundation, *Raspberry Pi 4 Datasheet*, 2023. Rev. 1.0.

[11] R. P. Ltd, *Buy A Raspberry Pi 4 Model B – Raspberry Pi*. Accessed: January 24, 2024.

[12] “Raspberry Pi Documentation - camera.” Accessed: January 26, 2024.

[13] Ultralytics, “Track,” 2 2024. Accessed: January 30, 2024.

[14] “EV-X821T-2D CMOS Barcode Scanner Module-Shenzhen Dalang Electronic and Technology Ltd. China barcode scanner qr reader module Manufacturers, Suppliers, Factory, Customization OEM ODM.” Accessed: January 22, 2024.

[15] “50kg Load Cells with HX711 and Arduino. 4x, 2x, 1x Diagrams. - Circuit Journal.” Accessed: January 22, 2024.

[16] “Raspberry pi 7-inch touch display datasheet.” Accessed: January 25, 2024.