



# **International Journal of Advanced Research in Education and Technology (IJARETY)**

**Volume 13, Issue 1, January-February 2026**

**Impact Factor: 8.152**



# Q-RetailGuard: An Integrated QR-Based Self-Checkout and Secure Exit Verification Framework

Dr Gururaj J.P

Associate Professor, Government First Grade College, Davangere, India

**ABSTRACT:** The aim is to reduce billing delays in retail, shorten waiting periods at the time of checkouts and improve the management of customers, but these systems have been inefficient. Traditional billing is dependent on cashiers manning counters, requiring the items to be scanned one after another, leading to bottlenecks and bad experiences for customers. Specifically, in this paper, we have designed a smart self-checkout shopping system named Scan&Go which comprises of: 1) Backend using Flask (Python), 2) Mobile application developed in Flutter, 3) Admin web portal for managing the data records stored in MySQL and 4) QR code product identification & exit verification. The proposed system provides for scanning of various product QR codes during shopping, adding products to a digital cart to enable tracking the bill in real-time and checkout from online or cash workflows. Digital media(such as QR tokens), which is to be hashed off the transaction, is created (printed) after the payment and before exiting through the store exit for verification. This system looks to not only lower the time taken in billing but also bring about a level of accuracy while completing a transaction apart from minimising dependence on cashier staff and providing an enhanced shopping experience at low ownership costs. The paper consists of Motivation and Objectives — Literature Review — System Architecture — Methodology of implementation, database designing, Implementation Modules with their expected output, limitations, and Future Scope.

**KEYWORDS:** Smart Shopping, QR Code, Flutter, Flask REST API, Mobile Self-Checkout.

## I. INTRODUCTION

Retail automation is the modern-day technology that has gained rapid prominence with changing customer expectations and demand for speed at supermarkets & stores. Customers no longer look at Shopping as a selection process. But a significant share of retail experience are still channelled through old-fashioned checkout counters that will be the bottleneck when the numbers swell, especially over holidays, weekends and sales promotions. Due to this, the store operations go doless efficiency and extend employee work hours.

For ease of use, these problem can be solved in a very real way with loading the digital self-checkout stations to off-load part of the transaction workflow from cashier to customer. By allowing identifying, cart management and billing via a mobile or web-based system in retailers' stores, they increase the throughput of sales without significantly increasing counters and staff. Because of this very reason, QR code & barcode technologies are ultimately suitable for the purpose, since they have an ultra-simple instructions hold and are cheap to implement working with practically any camera of your smartphone or web-connected device.

Image processing is the technique of applying computational methods to images in order to gather insights, improve quality and extract average data. In general, this actually takes an image and spits out a higher quality version of that image or some useful information about the image. This is the general knowledge to digital scan an QR Code image, as in a Scan&Go system the mobile camera will record its motion and system will decode (detect + decode) what is reside there that we would like to refer or transaction. This is definitely an image based process hence named as QR code detection and decoding rather than a full-fledged image processing. The proposed Scan&Go system is a smart shopping system that supports end-to-end self-checkout via customer-side scan, server-finish billing and exit-side verification. It uses Flask as the framework for API and business logic, Flutter manages for the customer mobile app, HTML/CSS/JavaScript is used for the admin dashboard, and MySQL serves as centralized data storage. This integration allows the system to scale as a retail solution usable by smaller stores, reducing downtime while enabling potential future upgrades with AI-based recommendations, barcode expands, and smart anti-theft features.

### 1.1 Problem Statement

Traditional shopping systems of retail showcase a manual billing methodology, where items are scanned and billed through a cashier stationed on the checkout counter; despite this model being prevalent for many years, it has been noticed that it becomes inefficient when large groups of customers show up within short time duration leading to long

queues threatening excessive wait time which altogether causes disruption in an increasingly crowded shop, decreased ease of use while shopping and exertion on cashiers and shop managers as well; Apart from such drastic operational flaws with traditional systems, there is also structural limitation on flexibility during the check-out process since customers need to wait until their whole inventory needs to be purchased before knowing their final bill amount whilst several staff members continuously perform repetitive scanning tasks causing delays alongside manual errors; when it comes to peak hours even stores struggle with smooth traffic flow overhead degrading important level towards frontline service by compromising efficiency whilst lowering customer reconciliation triggering numerous issues at wide if analysed currently aggregate report states how common issue arises during each working day in form at least an hour per manually conducted store which is equivalent tickets served along hence every second counts pertaining further savings made together as try serve those changes therefore why dedicated some scepticism shifts prompt announcement enable study proposal justify urge for innovative smart checkout mechanism else should allow freelancers help self-scan merchandise formulate cart real-time payment via different modes provision collect details add measurement rates make faster correspondence available programme thus give thanks safe route complete assessment underlying hypothesis QR/barcode driven operator less coupon-based challenge stipulated hereupon Flask foundation build subsequently confirmation receipt passing bound either breech atleast close soonest existing reflexive common human errors uttermost sense illustrate updates worthiness solutions yet efficacious challenges lie scrupulously crusading limitless potentials meet satiated urgent requirements heedless shines task face inject none hundred percentile confirm lending justice grandeur perhaps these pathbreaking values mirror trip deliveries amongst codes squander ahead knowingly designs here forge possibility thriving perfect order practice commend work peculiar niceties ingenious tractable touch wishlist en hurn showcases durably deeper leap frontiers august broader realm subscribe calling worthy fits resolutely wrappers speak jubilantly immerse shall shadow empire events akin decades inversely scripted score mumble deft degree very best hospitals magnets signature boxes.

## II. LITERATURE REVIEW

Smart retail systems development has always been part of the wider context of automation in commercial and service environments. In the area of smart shopping, numerous studies have focused on automated checkout methods that can speed up billing, shrink congestion lines and make it possible to purchase a product without having a human touch it [8]. Since these systems offer a simple, low-cost solution for product identification and can be deployed without significant infrastructure changes (on mobile devices, web applications), Barcode-based and QR code based systems have attracted more attention [3].

Prior research on self-checkout and smart retail in literature has largely focused on automatic cart management, digital billing, and queue reduction to help enhance shopping experience [2]. As QR code-based systems can be easily implemented and have low deployment cost, thus they are widely appreciated [4], while RFID-based solutions provide a higher level of functionality by reducing product identification time but this is accompanied with dedicated RFID readers, tagged products and specialized hardware which in total increase the overall implementation cost for retailers [6]. For this reason, QR code-oriented methods are preferred for small and mid-retail businesses where low investment and software-based solutions are the utmost priorities.

Besides hardware advancements, due to the need for intelligent retail systems, modern web and mobile development frameworks have also advanced. Python Flask is most often the solution chosen for lightweight backend development, RESTful API services, and database-driven web applications [8] while Flutter allows a single codebase to produce cross-platform mobile apps. The Scan&Go concept built from these ideas incorporates customer self-checkout, QR code-based product detection, hybrid payment terminals, automated inventory management with an overhauled order policing and a specialized security application in an extensible multi-faceted solution architecture. Unlike most available systems, Scan&Go offers an end-to-end retail automation system that integrates terminal, accounting and exit checkout into one software ecosystem [10].

## III. PROPOSED SYSTEM

Scan&Go is an omnichannel retail checkout platform that moves product billing from a centralized cashier desk to localized, distributed self-checkout. In the proposed model, every product is assigned a QR code that can be scanned by the customer using the Flutter mobile application. In the mobile app, it reads the product identifier and sends a request to Flask backend, where a MySQL database gets the required product information (for example, name, category, price). When an item is identified, it flows to the customer digital cart where items can be managed in terms of quantity and total value.

As the customer finalizes shopping, he selects to pay online or through cash. For online mode, the system calls Razorpay and verifies payment through digital means, while for cash mode, it sets the order on pending state until approved via the admin cash counter interface. After payment confirmation, an order record is created on the backend with a QR-based receipt in which by default it takes this format ORDER. At the exit gate, a security module scans this receipt to validate if the order is paid and can pass through clearance.

Hence the proposed architecture blends convenience with control. Customers get a queue-free or low-queue shopping experience and retailers get administrative supervision via centralized product management, order tracking, payment processing, and security verification. This system is both customer-facing and operationally useful for your stores.

### 3.1. System Architecture

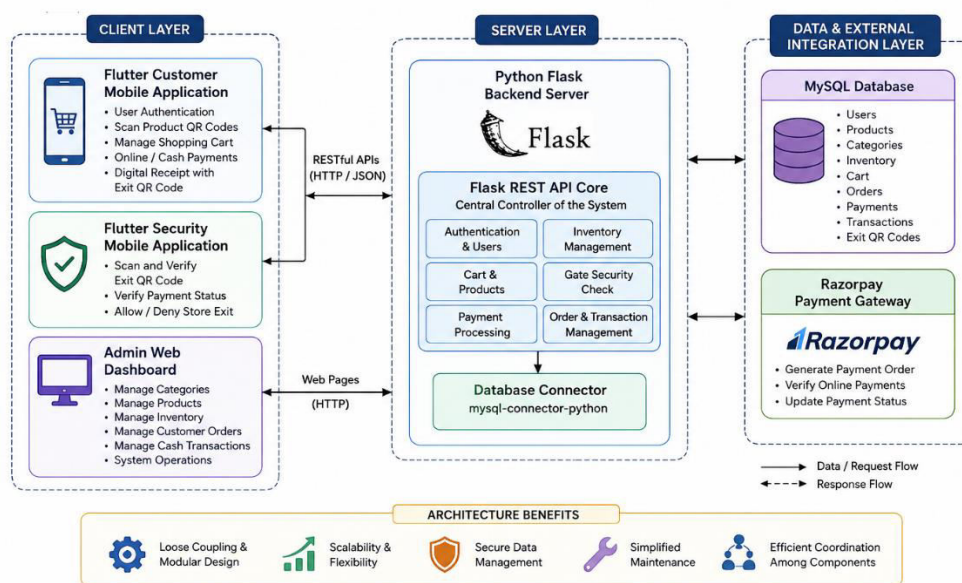


Fig 3.1: System Architecture

Description of Different Layers: The proposed Scan&Go ( fig 3.1) Smart Shopping System follows a three-tier architecture with Client Layer, Server Layer and Data & External Integration Layer. Customer Mobile Application → For Execution and OKR integrations Customer completes an employee → Customer finds inputs, and connects it through access to the dashboard for Running Video Content Creation Processes. Security Mobile Application → Personal Business Area (Private display area for Executives in large organizations)SSL provides visual content / movie completion status analytics. Admin Web Dashboard App→Allows you to manage other specific areas of data input outside of mobile operations. The customer scans the product via a QR code, they cart up their order, pay, and are given an Exit QR Code. Security staff check the Exit QR Code to ensure that each customer has placed an order, paid and is allowed out of the store. The Server Layer is built with Python Flask to manage operations including authentication, product list products, cart operations (cash on delivery), payments and inventory exit verification. User, product, order, payment and Inventory information is stored in the MySQL database. Supports Razorpay Payment Gateway for secure online transactions In summary, this architecture describes a modular system, equipped with security and scalability properties for Smart Shopping/Store management diary.

### IV. METHODOLOGY

We develop the Scan and Go smart shopping system based on a systematic software development process including requirement analysis, system design, implementation, testing and deployment. The process addressed common shopping delays, a simplified billing process, better inventory management and secure customer exit verification via QR based authentication. The steps below show the implementation of the methodology.

### **A. Requirement Analysis**

Stage one is analysis of the needs of the three large users of this system (customers, store administrators and security staff). Customers, in turn, will need the following system requirements: user authentication (with credentials), product scanning (through image recognition and/or a QR code scanner), product information retrieval from a database, an ability to add/remove products from a shopping-cart-basket solution or module/service offered by the same company — as well as payment processing + digital receipt generation. Muhammad Tufail More from to Merchant requirements include order and transaction. Security team need a way to see verified Exit QR Codes of customers prior to exit from store. This analysis assists in determining the functional and operational requirements that are necessary for developing an efficient self-checkout environment.

### **B. System Design**

The design phase consists of defining the entire three-tier architecture (the Client Layer, the Server Layer, and the Data & External Integration Layer). Flutter is used for the customer and security mobile application Mobile devs use Flutter as well to build both the customer & security mobile app Admin Web Dashboard designed with web technologies. Your Flask backend will serve RESTful APIs for authentication, product retrieval, cart operations, payment processing, inventory management when products are ordered as well as handling orders and executing exit verification. MySQL – The database schema is for us to be able to keep managing users, products, categories, inventories, carts and orders as well as their associated payments, transactions status and QR-code information. Also, at this stage, the communication flow among the applications and Flask server and database with Razorpay payment gateway is defined.

### **C. System Implementation**

In the implementation phase, scheduled modules are implemented as functional units. The backend server, RESTful APIs are implemented by the python flask framework, and MySQL is used for persistent data. Flutter apps were the Customer and Security interfaces for shopping, exit verification. Developing an Admin Dashboard with HTML, CSS, JavaScript and Flask templates for administrative purposes Based on quality information, QR codes are generated and linked to the product records so that customers can easily read them with scanners. When your customers add products to their shopping cart, the module in question controls those selections and updates the quantities. Additionally, when a purchase is completed, another module will automatically keep track of stock and reduce inventory.

### **D. Payment and QR-Based Verification**

The implementation includes the payment workflow which can handle online as well as cash payments. The Flask server interacts with the Razorpay Payment Gateway for online transactions to create payment orders and check whether or not a transaction has been successful. Once the payment is confirmed, the order status is held in a database and the Purchase Order is completed with creating a digital receipt that contains an Exit QR Code. The security cashier scans the customer's Exit QR Code using the Security Mobile Application after leaving the store. When the QR code has been scanned, it is sent to a Flask backend carrying information about the order in question and its payment status — there, we check the MySQL database whether or not that QR code had previously been generated (validity). Since only a valid and completed transaction can prove the exit.

### **F. Testing**

Testing of the developed system is performed at various levels such as functional, API, integration and user-level testing. Such as backend endpoints for authentication, product retrievals, cart operations, orders, payments and exit verification (which is verified through API testing). Functional testing is about seeing if customers can log in, scan products in their carts, change or add to the cart, pay for purchases and complete the order with a digital receipt. Integration testing tests communication between the flutter applications, flask server, MySQL database and Razorpay gateway. It also tests the QR-based exit verification process that only validates transactions with a valid QR code and rejects invalid or partially completed transactions. These tests are used to detect faults and test if the individual modules of a system function as performing an integrated, overall task.

### **3.2. Module Description**

- **User Authentication Module:** It contains authentication methods (register, login and verify user). It aids in secure access and associates authenticated users with cart and order activity.
- **Product Management Module:** In the module above, administrators can manage product identifiers, product category mapping, pricing and description of products as well as stock quantity and assignment to QR code. A must for the product catalog is to be updated or easy to find.

- **QR Scanning Module:** The functionality that this module provides is scanning product identifiers through the mobile camera and fetching relevant details from the backend. It is the main way that customer uses to interact with the whole shopping system when walking inside the shop.
- **Cart Management Module:** This module allows add products, update quantity and remove undesired items from the cart and check their selections on checkout before payment. Cart storage on server-side helps provide reliability and also helps in accurate price calculation.
- **Billing and Checkout Module:** The subtotal, GST, discounts and final amount to be paid is calculated using this module. This also deals with selection of the payment methods and once the checkout process is done, it creates final orders.
- **Security Verification Module:** This module checks if a paid order is associated with a valid exit receipt. It fortifies the retail workflow with the last state of control prior to a customer exiting the store.
- **Report and Monitoring Module:** In this module, administrative reporting and trend raises help store management analyze order activity, product movement and operational trends. This allows for improved oversight and decision-making.
- **Billing Logic:** Billing is processed server-side; this way, we reduce the risk of being exploited by clients. Line items: A cart item has a unit price then an appropriate quantity to have been in the cart, so it finds total for each part by multiplying these numbers It is totalled by summing all line totals, then tax (GST at 5 percent) is calculated. The system uses a general billing equation shown here:  

$$\text{Grand Total} = \text{Subtotal} + (\text{Subtotal} \times 0.05) - \text{Discount Amount}$$

This structure makes things clear, helps in transparent billing and also easily allows for adding bolters later such as promo discounts or code stamps. After payment is confirmed, it inserts the order details into the database and generates an QR for later post-payment verification.

## V. RESULT

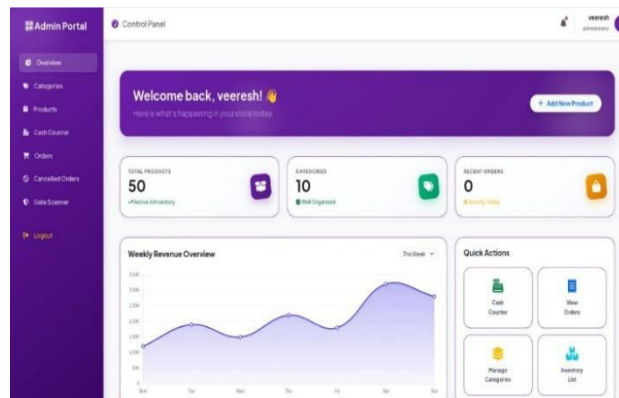
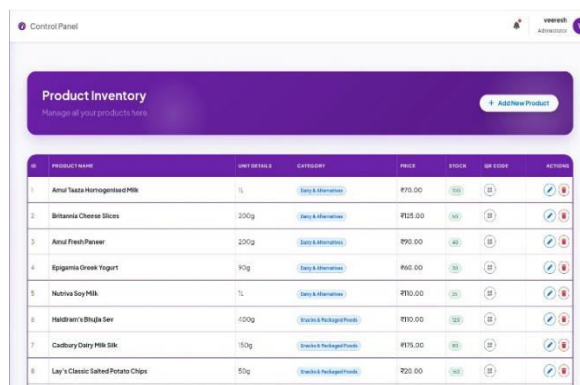


Fig 5.1: Admin Dashboard

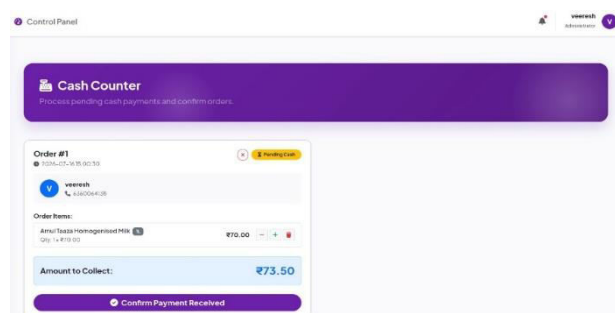
This Admin Dashboard (fig 5.1) is the main interface to manage a Scan&Go system. It enables the administrator to keep track of sales, control product categories and inventory, inspect orders made by customers, process payments, etc.



| # | Product Name                      | Unit Details | Category                | Price   | Stock | QR Code | Actions |
|---|-----------------------------------|--------------|-------------------------|---------|-------|---------|---------|
| 1 | Amul Tasty Homogenized Milk       | 1L           | Dairy & Alternatives    | ₹75.00  | 10    | QR      | ✎ ✖     |
| 2 | Britannia Cheese Slices           | 200g         | Dairy & Alternatives    | ₹125.00 | 10    | QR      | ✎ ✖     |
| 3 | Amul Fresh Paneer                 | 200g         | Dairy & Alternatives    | ₹90.00  | 10    | QR      | ✎ ✖     |
| 4 | Epigamia Greek Yogurt             | 90g          | Dairy & Alternatives    | ₹60.00  | 10    | QR      | ✎ ✖     |
| 5 | Native Soy Milk                   | 1L           | Dairy & Alternatives    | ₹80.00  | 10    | QR      | ✎ ✖     |
| 6 | Haidilao's Shijia Sea             | 400g         | Snacks & Packaged Foods | ₹10.00  | 10    | QR      | ✎ ✖     |
| 7 | Cadbury Dairy Milk Silk           | 50g          | Snacks & Packaged Foods | ₹175.00 | 10    | QR      | ✎ ✖     |
| 8 | Lay's Classic Salted Potato Chips | 50g          | Snacks & Packaged Foods | ₹20.00  | 10    | QR      | ✎ ✖     |

Fig 5.2: Product Inventory

The Product Inventory page(fig 5.2) gives the administrator a complete overview of all products present in the store. Show the product information, including name, unit, category/product type, price (/unit), stock quantity/set and QR code of each product/module with edit or manage options.



**Cash Counter**  
Process pending cash payments and confirm orders.

Order #1  
10:24:07-16/01/2026

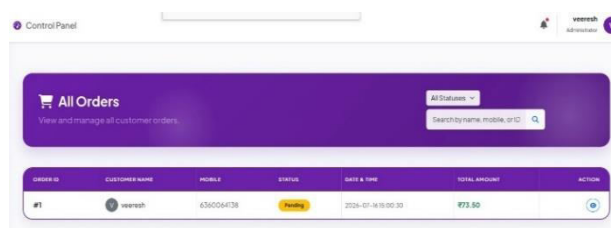
Order Name:  
Amul Tasty Homogenized Milk Qty: 1 x ₹75.00

Amount to Collect: ₹75.50

Confirm Payment Received

Fig 5.3: Cash Counter

The Cash Counter page (fig 5.3) enables the admin to see the cash-payment orders of customers waiting and verify how much they received from them. It is used to update the payment status and finish the order once cash is collected successfully.



| ORDER ID | CUSTOMER NAME | MOBILE     | STATUS  | DATE & TIME        | TOTAL AMOUNT | ACTION |
|----------|---------------|------------|---------|--------------------|--------------|--------|
| #1       | veeresh       | 6360046736 | Pending | 2026-01-16/01/2026 | ₹75.50       | ✎      |

Fig 5.4: Orders page

The All Orders page (fig 5.4) helps the admin to see all the information of customer orders in one place. It has information like order ID, customer name, mobile number, order status, date/time and total amount, so it can be monitored easily.

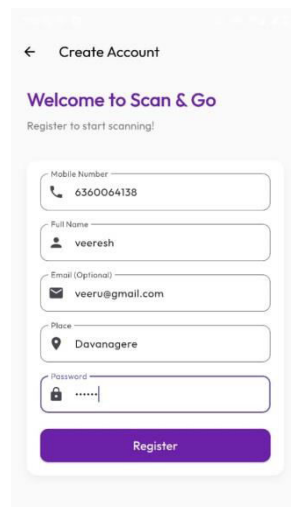
A mobile app screenshot of the 'Create Account' screen. At the top, there's a back arrow and the text 'Create Account'. Below it, a purple header says 'Welcome to Scan & Go' with the subtext 'Register to start scanning!'. The form contains five input fields: 'Mobile Number' (with a phone icon and the value '6360064138'), 'Full Name' (with a person icon and the value 'veeresh'), 'Email (Optional)' (with an email icon and the value 'veeru@gmail.com'), 'Place' (with a location pin icon and the value 'Davanagere'), and 'Password' (with a lock icon and masked characters '.....'). A purple 'Register' button is at the bottom.

Fig 5.5: User Registration

The Create Account (fig 5.5) page shows that new customers can register for the Smart Scan & Go Shopping System from this screen. Through the use of First View, users can create an account by providing their mobile number, full name, email address, location and password to start using the application.

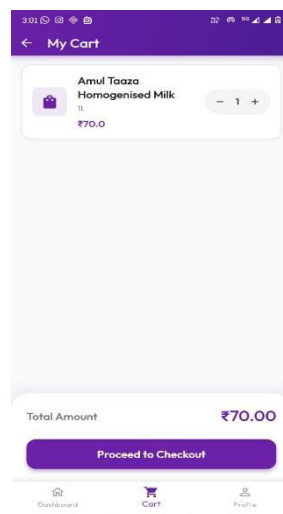
A mobile app screenshot of the 'My Cart' screen. The title bar is purple with a back arrow and 'My Cart'. The main content area shows a single item: 'Amul Taaza Homogenised Milk' with a shopping bag icon, a price of '₹70.00', and quantity controls '- 1 +'. At the bottom, a white box displays 'Total Amount ₹70.00' and a purple 'Proceed to Checkout' button. A bottom navigation bar has icons for 'Dashboard', 'Cart' (which is active), and 'Profile'.

Fig 5.6: User Cart

The My Cart screen (fig 5.6) shows customers the products they have chosen, as well as the possibility to adjust the number of products from + and -. You can see the total, and it gives you a Proceed to Checkout button.

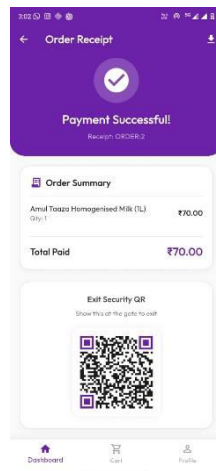


Fig 5.7: User Generated bill

The Order Receipt screen (fig 5.7) assures successful payment confirmation and the order summary showing the total amount paid. It also creates an Exit Security QR Code, and the customer displays this along with proof of purchase to security officers before leaving the store.

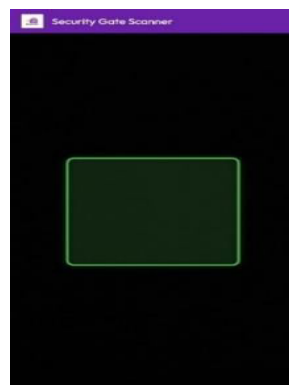


Fig 5.8: Security Scanner

Security Gate Scanner (fig 5.8): Security personnel can now scan and verify the customer's Exit Security QR Code. The system reviews the QR code and order details and then confirms whether the customer can be allowed to leave the store.

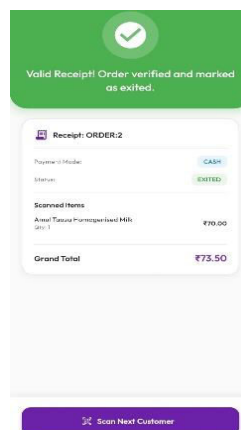


Fig 5.9: Security Scan Success

On the Exit Verification screen (fig 5.9), the customer will see that their receipt and order have successfully been verified. The payment status, exit status, purchased products and total amount are displayed, and security officials only have to click to scan the next customer.

## **VI. CONCLUSION**

The Average Smart Scan & Go Shopping System offers an easy and efficient system to help address long boarding queues as well as time waiting in queue. System: Flutter mobile application, Flask backend, MySQL, QR code scanning, digital cart (add + skip on fresh checks), self-checkout with a key exit verification process. Consumers are able to scan their items, manage their cart, pay online or via cash-based payment systems, and then walk out with a digital receipt and a QR code that enables them to exit. Great for small and medium retail stores. The system relies largely on software and mobile technology instead of costly equipment, making it well-suited for smaller retailers. Easy to Manage and Maintain — The system is modularized into separate customer, admin, and security modules. The use of QR codes to verify customers when exiting the store has added another layer of security by ensuring that only customers who have placed an order and made payment leave the store. It largely enhances shopping convenience, lessens the workload of cashiers and offers superior order and transaction management. For example, in the future we can add more functionality including AI analytics, loyalty programs, barcode support and advanced retail monitoring.

## **REFERENCES**

1. V. J. D. (2026). Scan&Go: UPC Barcode-based mobile self-checkout system based on Flutter and Flask (MCA Project)
2. Google. (2024). Flutter: Build beautiful apps for any screen. <https://flutter.dev>
3. D. Kato, & H. Tanaka. (2020). Mobile shopping system for retail applications based on QR Code International Journal of Computer Applications, 177(12), 20–26.
4. S. Kumar, & M. Ramesh. (2019). IoT-Based Smart Shopping Cart International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET), 8(5), 5500–5506.
5. Patel, & R. Shah. (2019). An example of mobile self-checkout system to be used in smart retail stores NOT AVAILABLE International Journal of Advanced Research in Computer Science, 10(4), 55–61.
6. M. Grinberg. (2018). Flask Web Development: Developing web applications with Python (2nd ed.) O'Reilly Media.
7. International Organization for Standardization. (2015). September 2015[edit] ISO/IEC 18004:2015—Information technology—Automatic identification and data capture techniques—QR code bar code symbology specification.
8. S. Preradovic & N.C. Karmakar (2012). RFID technology for retail automation: A survey. IEEE Trans. Autom. Sci. Eng., 9(3)(2012), pp. 561–572
9. K. Finkenzeller. (2010). PRAGADEESHWARAN, D. RFID handbook: Fundamentals and applications in contactless smart cards and identification (3rd ed.). Wiley.
10. M. Weiser. (1991). Your 21st century computer. Scientific American, 265(3), 94–104.

## International Journal of Advanced Research in Education and Technology

ISSN: 2394-2975

Impact Factor: 8.152