International Journal of Technology and Applied Science (IJTAS)
E-ISSN: 2230-9004 ● Website: www.ijtas.com ● Email: editor@ijtas.com

SCANOVA: Intelligent Cashier-Less Retail Billing System

Darshan G[1], Dr. Abhilash C N[2], Tarun B G[3], H J Suhas[4], Vinesh Mullai G[5]

[1,2,3,4,5]Department of Artificial Intelligence and Machine Learning
SJB Institute of Technology, Bengaluru, India.

Abstract:
We built a cashier-less retail prototype that brings barcode-based product identification, a digital cart, automated GST billing, UPI payment, invoice generation and machine-learning-based transaction-risk analysis into a single web application. The frontend is written in Vite and TypeScript, and Supabase handles authentication, PostgreSQL storage and Row Level Security. Barcode tampering checks and anomaly scoring sit directly inside the scan-to-receipt flow. Our catalogue holds 40 products from 14 brands, and we tested normal checkout, an unusual basket and a tampered barcode. The results show that the main modules work together; the risk scores are observations from the prototype, not a formal fraud-accuracy benchmark.

Index Terms: cashier-less retail, barcode scanning, anomaly detection, Isolation Forest, Autoencoder, barcode tampering, UPI payment.

I. INTRODUCTION

Checkout is the slowest part of a shopping trip. In a conventional point-of-sale (POS) setup, staff scan every item, work out the bill, take the payment and print the receipt. Self-checkout hands the scanning to the customer, but it brings its own problems: missed scans, swapped barcodes and odd buying patterns. Fully automated stores avoid these with cameras, shelf sensors and computer vision, but that hardware is costly for small and medium-sized retailers [1], [5].

This project, SCANOVA, takes a cheaper route built on a camera-enabled web interface, a product catalogue and cloud services. The shopper picks a brand, scans a barcode (or types it in), reviews the product, fills a cart, pays digitally and gets a QR-verifiable invoice. Along the way, the system scores the transaction and stores the risk details for later review. The main contributions are as follows.

1) *An integrated scan-to-receipt workflow.* Product scan, catalogue match, cart, billing, payment, invoice and inventory update run as one connected sequence.
2) *Barcode tampering checks before the cart.* A suspicious barcode is blocked before an item can reach the cart.
3) *Automatic GST billing.* The subtotal and 18% GST are calculated by one billing rule that the cart, payment screen and invoice all share.
4) *UPI payment with digital invoices.* Each invoice carries a verification QR code, with PDF download

and print options.

5) *Transaction-level risk scoring.* Risk scores, bands and contributing factors are tied to persistent database records.

Existing work splits roughly into two camps: systems that recognise the physical object itself, and systems that recognise it through a code attached to it. Vision-based systems infer the product from image features, whereas barcode systems look up a catalogue record from a scanned identifier. We chose barcode lookup because it is affordable, and added tampering checks and anomaly analysis on top. The aim is therefore to secure the checkout workflow, not to replace every product-recognition method.

The rest of the paper covers related work (Section II), the system overview and architecture (Section III), the methodology (Section IV), the implementation (Section V), results (Section VI), and a discussion with conclusions (Sections VII and VIII).

II. RELATED WORK

Studies of cashierless retail cover camera-based, sensor-assisted and vision-driven checkout designs [1], [2]. Work on automatic checkout relies on object detection, tracking and image enhancement to cope with occlusion, motion blur and look-alike products [3]–[9]. Barcode-based smart trolleys pair product scanning with item verification and payment [10]. On the security side, recent machine-learning research looks at feature selection, supervised fraud classification and deep-learning methods

[11]–[14]. Image-manipulation studies analyse local noise and edge artifacts to spot altered regions [15]. That is useful background for detecting tampered barcodes, although it has not been tested on barcodes specifically.

Each of these lines of work covers only part of retail automation. Our project keeps the simplicity of barcode lookup and adds tampering checks and transaction anomaly analysis. Where fully vision-based checkout infers the product from images, our prototype treats the barcode as the primary product key and applies extra security checks before an item enters the cart and again during transaction analysis. Table I summarises how each line of work relates to this project.

TABLE I- SUMMARY OF RELATED APPROACHES

Approach	Useful capability	Relevance here
Cashierless retail [1], [2]	Vision/sensor-assisted checkout	Context for low-cost retail automation
YOLO / CNN [3]–[9]	Product detection and tracking	Related automated-checkout methods
Barcode trolley [10]	Barcode scan and item verification	Closest workflow comparison
Fraud ML [11]–[14]	Transaction anomaly / fraud scoring	Supports risk-analysis design
Image tampering [15]	Noise and edge-artifact analysis	Background for altered-barcode checks

III. SYSTEM OVERVIEW AND ARCHITECTURE

The system has a presentation layer, a managed backend and database, and three functional modules for barcode scanning, machine-learning analysis and payment (Fig. 1). The Vite and TypeScript frontend covers registration and login, brand selection, the scanner, product details, cart, checkout, and the receipt and profile screens. Supabase Authentication manages sessions, while PostgreSQL stores profiles, products, carts, orders, order items, payments, inventory and fraud records. Row Level Security (RLS) limits each user to the rows they are allowed to see.

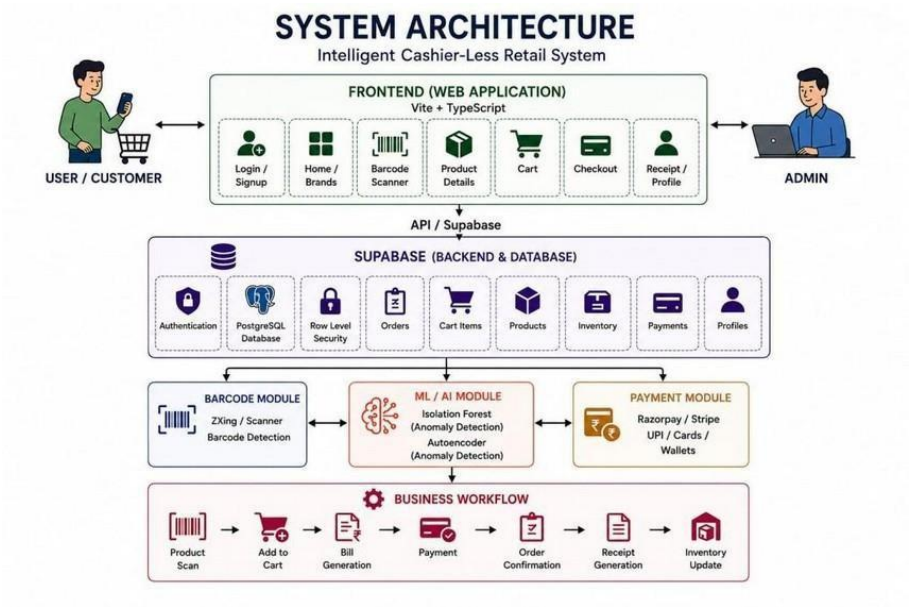


Fig. 1. Layered architecture of the cashier-less retail system.

A purchase moves through a fixed sequence: product scan → catalogue match → cart update → bill generation → payment → order confirmation → invoice generation → inventory update. The modules talk to the backend through REST and SDK calls from the frontend. Because order, payment and fraud records stay linked, any completed purchase can be traced back to its risk evaluation.

In the database, profiles connect to orders, and orders connect to order items and payments (Fig. 2). Order items use product identifiers to fetch product information. A fraud-detection record ties a transaction to its risk score, risk level, contributing factors and the action taken. Supporting tables hold scans, inventory and exit-verification data.

The schema follows the life of a purchase. Profiles store user-level details, and products map barcode identifiers to catalogue attributes. Cart items hold what a shopper has picked during a session; orders and order items hold the finished purchase and its line items. Payments record the transaction identifier, receipt number, method, amount, status and timestamp. Inventory tracks stock, and scan_events logs scanning activity. The fraud_detections and barcode_tampering_detections tables keep the risk analysis and the item-level tampering results, and the exit-verification tables support the verification workflow described in the project report.

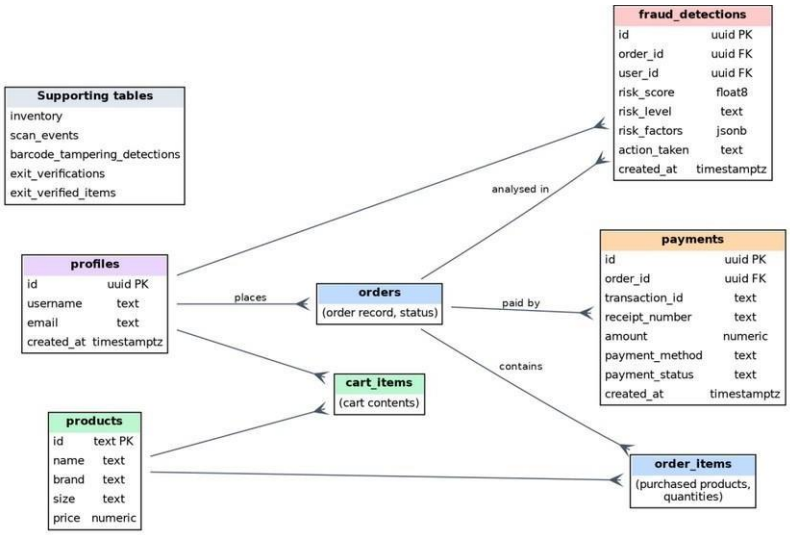


Fig. 2. Main database tables and relationships.

Row Level Security is central to how data access works. Authentication identifies the user, and RLS policies are meant to restrict reads and writes to that user's own records. This matters most for profiles, carts, orders and payment history, which should never be visible to other customers. Administrators get their own authorised access path, so store-level analytics are never mixed up with ordinary customer data.

A customer starts by registering or logging in and then chooses a product brand. The scanner reads a barcode through the camera, with manual entry as a fallback. Once the barcode is processed, the app either blocks the item and asks for a rescan, or retrieves the product details, which allows it to be added to the cart. The cart keeps quantities and a running total, and tax is calculated at checkout before the customer picks a payment method. After payment, the order and payment records are saved, the invoice is generated, the receipt is shown and inventory is updated.

Looking at the data flow, the customer provides credentials, scan inputs and cart actions, and the application returns product details, cart contents, billing totals, payment status and the receipt. The store administrator maintains the catalogue and stock and reviews sales, product and customer analytics. The payment provider or UPI app returns the payment result, which the application attaches to the order. For this to work, the user session, order, order items, payment record and fraud record all need to share consistent identifiers.

We also separate functional requirements from security requirements. A functional requirement says what a shopper can do, such as adding an item or downloading an invoice. A security requirement says who may access the resulting record. Supabase Authentication handles the session and RLS enforces row-level restrictions. Server-side validation and careful handling of payment status matter as well, since the user interface alone should never be the authority on whether a payment or an inventory change really happened.

IV. METHODOLOGY

A. Product Identification and Barcode Tampering Check

The customer selects a brand and scans a barcode with the device camera, or types it in when the camera is unavailable. Before any product lookup, the code is checked for signs of tampering. A

suspicious barcode is blocked and the interface asks for another scan. A code that passes is matched against the products table, and the system shows the product name, brand, size, stock and price. The catalogue has 40 products across 14 brands.

The workflow has to tell apart a code that is simply unrecognised from one that has been flagged as tampered. An unknown code should never quietly create a new product record. Instead, the interface offers a clear rescan or manual-entry option, and the catalogue stays the single source for name, brand, size, stock and price. Keeping this step apart from billing and risk analysis makes each stage easier to reason about (Fig. 3).

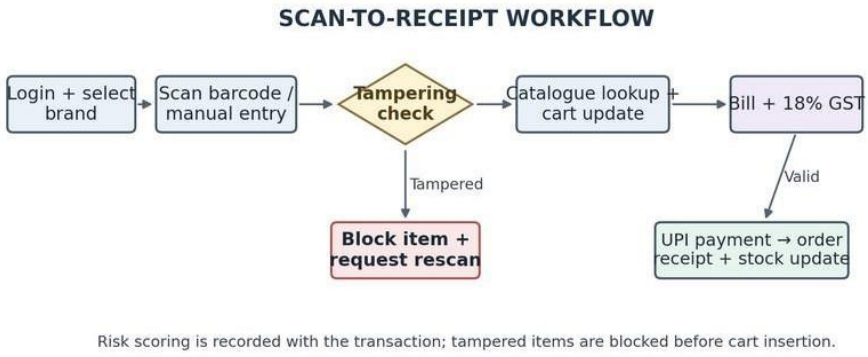


Fig. 3. Refined scan-to-receipt workflow.

B. Cart Management and Billing

Every addition, quantity change or removal updates the item count and the running total. At checkout, let S be the cart subtotal.

The application computes GST at 18% and the payable amount P as

$$GST = 0.18 \times S \quad (1)$$

$$P = S + GST \quad (2)$$

The payment screen and the final invoice show the same subtotal, tax and total, so the two never disagree.

As an example, the report lists an Air Max 90 at ₹12,999. At 18% GST the tax is ₹2,339.82, displayed as ₹2,340, and the payable amount is ₹15,339. The example shows why the cart, checkout, payment summary and invoice must all apply the same rounding and formatting rules. A transaction counts as complete only once the payment result has been handled and the order and payment records have been updated.

C. Payment and Digital Invoice

The payment screen shows the order summary, a QR code and UPI options such as PhonePe, Google Pay and Paytm. After a successful payment, the system creates an Order ID and a Transaction ID, updates the order status and saves the receipt number, amount, method, status and timestamp (Fig. 4). The digital invoice lists the purchased items, date, tax and total, and carries a verification QR code with PDF download and print options.

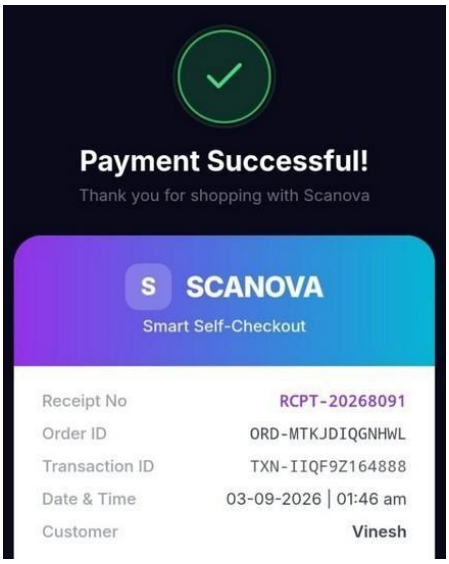


Fig. 4. Scanova payment receipt with order, transaction and receipt identifiers.

D. Fraud and Anomaly Analysis

The risk module evaluates eight transaction signals, each capturing a different kind of inconsistency. Price mismatch compares the billed price with what the catalogue expects. Quantity mismatch flags gaps between expected and recorded quantities. Repeated QR use points to a payment or verification code being reused, scanning speed picks up unusually fast scan sequences, and unusual item count catches atypical basket sizes. Transaction-pattern anomaly compares the current basket with learned behaviour, exit mismatch checks the purchase record against exit-verification data, and historical behaviour adds context from earlier activity where it exists.

Isolation Forest isolates observations that random partitions separate quickly, and it has recently been applied to online transaction-fraud screening [14]. The report also describes an Autoencoder signal based on reconstruction error, an approach covered in fraud-detection reviews [12], [13]. These signals are combined with the barcode-tampering result into a risk score between 0 and 1, which maps to a Low, Medium or High band (Fig. 5). The action taken and the contributing risk factors are stored against the order and user identifiers.

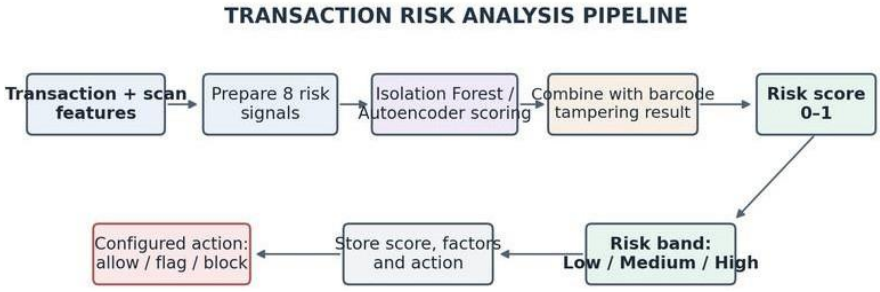


Fig. 5. Refined transaction-risk analysis pipeline.

By design, a tampered barcode is rejected before it can become a cart item (Fig. 6), while transaction-level anomalies are logged for further checking. A risk score only says how unusual something looks

to the configured model; it is not proof of fraud without further investigation.

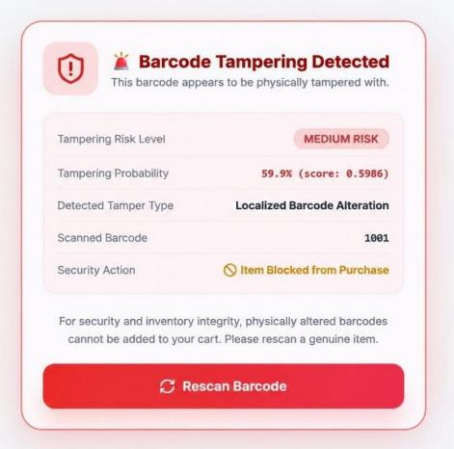


Fig. 6. Barcode tampering alert showing a medium-risk score and blocked item.

The score is a combined signal, not the probability that a customer cheated. Isolation Forest is good at finding records that differ from most others, and an Autoencoder can pick out patterns that a learned representation reconstructs poorly. We combine both with the tampering result and map the final score to a configured Low, Medium or High band. Because the risk factors and the action taken are stored, a reviewer can inspect the score later rather than face an unexplained alert.

At the product level, the barcode value links a scan to its catalogue entry. The catalogue supplies the attributes the customer sees and billing uses, so product names and prices are not copied into every scan event. Cart line items keep the product identity and quantity, and the order summary gives a stable snapshot of the purchase at checkout. The payment record then adds the transaction metadata, and the invoice shows the same order values the customer has already reviewed.

Responsibilities are divided by module. Authentication establishes the shopper session, product identification interprets barcode input and looks up the product, and the cart manages line items and totals. Checkout applies tax, payment records the result, and the receipt module presents a verifiable invoice. The fraud module evaluates each transaction separately, while inventory and analytics work from persisted records. This separation makes it easier to see at which stage something went wrong. The backend schema in the report has twelve related tables: profiles, products, cart_items, orders, order_items, payments,

inventory, scan_events, fraud_detections, barcode_tampering_detections, exit_verifications and exit_verified_items. Together they cover user identity, catalogue data, in-progress shopping, completed purchases, payment events and security checks. Keeping these entities apart avoids one oversized record for every kind of event and makes it simpler to query an order together with its items and risk outcome.

The receipt is more than a confirmation screen. It is the customer-facing summary of the stored transaction, with the receipt number, order and transaction identifiers, date, customer and item details, tax and total. Its verification QR code points back to the invoice record, and PDF download and print support let the customer keep a copy outside the app.

V. IMPLEMENTATION

The customer-facing flow is built with Vite and TypeScript. The scanner module reads barcodes through the camera and falls back to manual entry, and product lookup uses the barcode as the catalogue key. Supabase supplies authentication and PostgreSQL persistence, with RLS policies isolating each user's records. The payment and invoice components use order and transaction identifiers to connect the payment record to the bill and the receipt.

The implementation is split into cooperating modules: authentication and profiles; barcode scanning and product lookup; cart management; checkout and GST calculation; payment and order processing; invoice and receipt generation; fraud analysis; inventory administration; and the analytics dashboard. Each has a clear job, and order IDs, product IDs and transaction IDs tie the records together. As a result, an item can be traced from its scan event through the cart and order item to the final receipt.

The risk pipeline runs in five steps: feature preparation, model scoring, score combination, risk-band mapping and action recording. Feature preparation turns the current basket and session context into signals such as total value, item count, average item price, category diversity and scan timing. The model output is merged with the tampering result and mapped to a risk band, and the score, level, contributing factors and action are then stored with the relevant order identifiers.

The fraud module saves the score, risk level, risk factors and action taken. The wider report also describes an administrator dashboard with sales summaries, product performance and customer analytics, along with a seven-day sales forecast and an anomaly/risk evaluation log. It supports CSV exports and a PDF summary. These features draw on stored transaction and inventory records for operational review, sit apart from the customer-facing scan, payment and receipt steps, and do not replace the core scan-to-payment workflow.

Testing focuses on whether the workflow completes and stays consistent. A scan test checks that a valid code resolves to the right catalogue item and that an unknown or tampered code stays out of the cart. Cart tests cover additions, quantity changes and removals. Billing tests compare the subtotal, tax and payable total at checkout with the invoice, and payment tests confirm that order status and payment records share the correct identifiers. Receipt tests check that the invoice has the expected products, tax and verification QR code.

VI. RESULTS AND PERFORMANCE EVALUATION

The project report describes end-to-end testing of the shopping flow: barcode scan or manual entry, product lookup, cart changes, GST calculation, payment, receipt generation and fraud analysis, using the 40-product, 14-brand catalogue. The scores below come from test cases TC7–TC9 and are presented as development observations (Fig. 7).

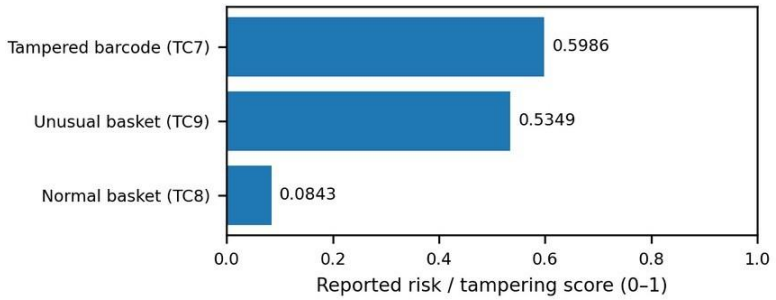


Fig. 7. Risk/tampering scores reported for test cases TC7–TC9.

The normal one-item basket (TC8) scored 0.0843 and was classified as Low risk. The unusual basket (TC9) scored 0.5349 and was classified as Medium risk. The tampered barcode (TC7) had a reported score of 0.5986 and was blocked instead of being added to the cart. Together, these cases show the intended difference between normal checkout, an unusual basket and item-level tampering.

The observations also confirm that the subtotal, GST and total matched between the payment screen and the invoice, and that manual barcode entry kept the flow usable when the camera could not scan. These results support the functional integration of the main modules. They do not give a labelled-dataset estimate of precision, recall, false-positive rate or overall fraud-detection accuracy, and the three cases show that different conditions are routed to different outcomes, not how accurate detection would be over a large population of transactions.

A web application on managed database services suits a prototype better than a store-wide sensor installation, and the 40-product catalogue gives a bounded set for exercising lookup and checkout. An operator can inspect stored orders and fraud records rather than rely on a passing on-screen alert. In a real deployment, the model thresholds, catalogue maintenance, payment confirmation and database policies would all need validating against the retailer's own procedures.

The three outcomes also illustrate the control logic. A normal basket proceeds with a low score, an unusual basket gets a medium score and is flagged for monitoring, and a tampered barcode is blocked before it reaches the cart. The responses differ because barcode tampering is an item-level condition, while a transaction anomaly is a basket-level signal. Keeping the two checks separate makes the outcome easier to explain to the customer and to review later.

VII. DISCUSSION

The design pairs low-cost barcode lookup with extra checks and avoids a full sensor-fusion installation. The database keeps user, order, payment and fraud records linked, and manual entry keeps checkout going when the camera is unavailable. There are trade-offs: barcode lookup is only as good as the catalogue, and an anomaly score can flag a legitimate but unusual purchase. This is why the system stores the risk and action information, so results can be reviewed in context.

One engineering point stands out: consistency across the application boundary. Price and stock should come from the catalogue, and subtotal and tax from one billing rule. Each payment should be tied to a single order, and the invoice should show that order's stored values. The links between order, order items, payment and fraud records then give a practical audit trail for investigating an unexpected amount, a failed payment or a risk alert.

VIII. CONCLUSION

The Intelligent Cashier-Less Retail System brings product scanning, tampering checks, cart management, GST billing, UPI payment, invoice generation and machine-learning-based risk scoring together in a Supabase-backed web application. Its architecture connects user and product data with order, payment and fraud records, and the workflow includes a manual barcode-entry fallback. The reported test cases show the intended handling of a normal basket, an unusual basket and a tampered barcode. Before the scores can be read as general fraud-detection performance, the risk model should be evaluated on labelled transaction data with clearly defined metrics.

REFERENCES:

1. E. Szabó-Szentgróti, S. Rámháp, and P. K. Kézai, "Systematic review of cashierless stores (just walk out stores) revolutionizing the retail," *Management & Marketing*, vol. 18, suppl. 1, pp. 427–448, 2023, doi: 10.2478/mmcks-2023-0023.
2. R. Y. Lee, T. Y. Chai, S. Y. Chua, Y. L. Lai, Y. W. Sim, and S. C. Haw, "Cashierless checkout vision system for smart retail using deep learning," *J. Syst. Manage. Sci.*, vol. 12, no. 4, pp. 232–250, 2022, doi: 10.33168/JSMS.2022.0415.
3. L. H. Pham *et al.*, "DeepACO: A robust deep learning-based automatic checkout system," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, 2022, pp. 3107–3114.
4. L. H. Pham, D. N. N. Tran, H. H. Nguyen, H. J. Jeon, T. H. P. Tran, H. M. Jeon, and J. W. Jeon, "Improving deep learning-based automatic checkout system using image enhancement techniques," in *Proc. IEEE/CVF CVPRW*, 2023, pp. 5333–5340.
5. A. Dhonde, P. Guntur, and V. Palani, "Adaptive RoI with pretrained models for automated retail checkout," in *Proc. IEEE/CVF CVPRW*, 2023, pp. 5507–5510.
6. A. Vats and D. C. Anastasiu, "Enhancing retail checkout through video inpainting, YOLOv8 detection, and DeepSort tracking," in *Proc. IEEE/CVF CVPRW*, 2023, pp. 5530–5537.
7. C. Wang *et al.*, "Smart retail SKUs checkout using improved residual network," *Sci. Rep.*, vol. 13, Art. no. 22512, 2023, doi: 10.1038/s41598-023-49543-x.
8. L. Tan, S. Liu, J. Gao, X. Liu, L. Chu, and H. Jiang, "Enhanced self-checkout system for retail based on improved YOLOv10," *J. Imaging*, vol. 10, no. 10, Art. no. 248, 2024, doi: 10.3390/jimaging10100248.
9. R. Skinderowicz, "Solving automatic check-out with fine-tuned YOLO models," *Procedia Comput. Sci.*, vol. 246, pp. 1649–1658, 2024, doi: 10.1016/j.procs.2024.09.644.
10. A. A. S. Gunawan, E. J. Andreal, W. Budiharto, H. Ngarianto, M. Attamimi, and H. Tolle, "Item verification on the smart trolley system using object recognition based on the structural similarity index," *Procedia Comput. Sci.*, vol. 227, pp. 147–158, 2023, doi: 10.1016/j.procs.2023.10.512.
11. E. Ileberi, Y. Sun, and Z. Wang, "A machine learning based credit card fraud detection using the GA algorithm for feature selection," *J. Big Data*, vol. 9, Art. no. 24, 2022, doi: 10.1186/s40537-022-00573-8.
12. A. Cherif *et al.*, "Credit card fraud detection in the era of disruptive technologies: A systematic review," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 35, no. 1, pp. 145–174, 2023, doi: 10.1016/j.jksuci.2022.11.008.

13. E. A. L. M. Btoush *et al.*, "A systematic review of literature on credit card cyber fraud detection using machine and deep learning," *PeerJ Comput. Sci.*, vol. 9, Art. no. e1278, 2023, doi: 10.7717/peerj-cs.1278.
14. H. Abbassi, S. E. L. Mendili, and Y. Gahi, "Digital banking fortification: A real-time isolation forest architecture for detecting online transaction fraud," *Eng. Res. Express*, vol. 6, Art. no. 025214, 2024, doi: 10.1088/2631-8695/ad4958.
15. X. Lin, S. Wang, J. Deng, Y. Fu, X. Bai, X. Chen, X. Qu, and W. Tang, "Image manipulation detection by multiple tampering traces and edge artifact enhancement," *Pattern Recognit.*, vol. 133, Art. no. 109026, 2023, doi: 10.1016/j.patcog.2022.109026.