QKART: CHECKOUT SIMPLIFIED

submitted in partial fulfillment of the requirements

for the degree of

BACHELOR OF TECHNOLOGY

in

Electronics and Computers Engineering and

Electronics and Communications Engineering

by

Arushi Gupta (102206190)
Tanishka Bhatia (102206217)
Khushi Kaushal (102206269)
Abhimanyu Kumar (102215021)
Uttkarsh Singh Pathania (102215291)

under the supervision of

Dr. Geetika Dua (Assistant Professor-III, ECED)

Department of Electronics and Communication Engineering

Thapar Institute of Engineering & Technology

Patiala

December 2025

DECLARATION

We declare that this written submission represents our ideas in our own words, and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented, fabricated, or falsified any idea/data/fact/source in our submission to the best of our knowledge. We understand that any violation of the above will be cause for disciplinary action by the Institute and may also result in penal action from the sources that have not been properly cited or from whom proper permission has not been obtained when necessary.

Place: Patiala
Date: 22-11-2025

Abhimanyu Kumar (102215021)

Arushi

Arushi Gupta (102206190)

Khushi Kaushal

Khushi Kaushal (102206269)

Tanishka Bhatia (102206217)

Uttkarsh Singh Pathania (102215291)

CERTIFICATE

This is to certify that the report titled QKart: Simplified Checkout, a CAPSTONE PROJECT REPORT, submitted by Abhimanyu Kumar, Arushi Gupta, Khushi Kaushal, Tanishka Bhatia, and Uttkarsh Singh Pathania, to the Thapar Institute of Engineering & Technology, Patiala, for the award of the degree of Bachelor of Technology, is a record of the project work done by them under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Place: Patiala
Date: 22-11-2025

Signature

Dr. Geetika Dua
Assistant Professor-III
Department of Electronics and
Communication Engineering
TIET, Patiala-147004

ACKNOWLEDGMENTS

We wish to express our profound gratitude to all those who have supported us throughout this capstone project.

We are particularly indebted to our esteemed mentor, Dr. Geetika Dua, Assistant Professor at the Department of Electronics and Communication Engineering, Thapar Institute of Engineering and Technology, Patiala. Her guidance, insightful feedback, and constant encouragement have been invaluable in shaping our project from a proposal into a tangible reality.

Our sincere thanks also go to the **Department of Electronics and Communication Engineering** and the management of Thapar Institute of Engineering and Technology for providing us with the necessary resources, a conducive learning environment, and the opportunity to undertake a project of this scale. The knowledge gained from relevant course subjects, including Computer Vision & Image Processing, Embedded Systems, and Machine Learning & AI, was foundational to our work.

Finally, we would like to acknowledge the hard work, dedication, and collaborative spirit of our team members—Abhimanyu Kumar, Uttkarsh Singh Pathania, Tanishka Bhatia, Arushi Gupta, and Khushi Kaushal. This project, “QKart: Checkout Simplified”, is a testament to our collective efforts and shared vision. We are proud to have worked together to create a solution that addresses real-world challenges in the retail industry.

ABSTRACT

Traditional retail environments are hampered by systemic inefficiencies, including long checkout queues, manual scanning errors, and operational bottlenecks that lead to customer dissatisfaction. This capstone project presents **QKart**, an intelligent shopping cart prototype engineered to solve these challenges by integrating Computer Vision (CV) and Internet of Things (IoT) technologies. The system is designed to be a cost-effective and scalable solution that automates product recognition, real-time billing, and fraud prevention.

The QKart prototype is built upon a Raspberry Pi 5, which serves as the central processing unit for on-cart edge computing. A lightweight, optimized YOLO (You Only Look Once) object detection model is deployed to identify items as they are placed in the cart. To ensure billing accuracy and mitigate fraud, a multi-sensor verification system is implemented, using load cell sensors and an HX711 module to cross-validate the detected item’s identity with its actual weight. Transactional data is synchronized in real-time to a Firebase cloud database and presented to the user through a web application, which facilitates seamless QR-based UPI payments.

Experimental results from the validation dataset show the model achieved a mean Average Precision (mAP) of 98.0% with an average on-device inference time of 94 milliseconds. The findings confirm the feasibility of a low-cost, tag-less smart cart system. By leveraging affordable hardware and open-source software, QKart demonstrates a robust and accessible solution for retailers to significantly reduce checkout times, enhance operational efficiency, and modernize the in-store customer experience.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

Abbreviation	Full Form
AI	Artificial Intelligence
API	Application Program Interface
ADC	Analog-to-Digital Converter
CV	Computer Vision
COTS	Commercial Off The Shelf
DB	Database
DFL	Distribution Focal Loss
Flask	Python Micro-Web Framework
GPIO	General Purpose Input/Output
HCI	Human-Computer Interaction
IoT	Internet of Things
LCD	Liquid Crystal Display
ML	Machine Learning
mAP	Mean Average Precision
Pi	Raspberry Pi
RFID	Radio Frequency Identification
SDG	Sustainable Development Goal
SQL	Structured Query Language
TFLite	TensorFlow Lite
TLS	Transport Layer Security
YOLO	You Only Look Once
UPI	Unified Payment Interface

CHAPTER 1

INTRODUCTION

1.1 Project Overview

This project presents QKart, an intelligent shopping cart prototype designed to solve systemic inefficiencies in traditional retail environments. The system directly addresses challenges such as protracted checkout queues, manual scanning inaccuracies, and poor inventory management by integrating cutting-edge Computer Vision (CV) and Internet of Things (IoT) technologies. The result is a cost-effective, scalable solution that automates product detection, provides real-time billing, and incorporates robust fraud prevention.

The core of the QKart prototype is a Raspberry Pi 5 (4GB RAM), which serves as the primary on-cart processing unit for edge computing. This processor runs a lightweight, highly-optimized YOLOv8 object detection model to identify products in real-time as they are placed into the cart. To ensure billing accuracy and mitigate theft, the system incorporates a crucial multi-sensor verification layer: weight sensors equipped with HX711 modules cross-validate the detected item's identity against its pre-recorded weight. All transactional data is synchronized in real-time via a Firebase cloud database to a dedicated web application, empowering customers with a live bill and enabling seamless QR-based UPI payments.

1.2 Motivation

The modern retail industry faces significant pressure to innovate in response to customer demands for convenience. Traditional checkouts are a primary source of friction, with one 2022 McKinsey report highlighting that over 30% of shoppers abandon purchases due to long queues, resulting in billions in lost revenue. Retailers also struggle with error-prone manual inventory tracking and high labor costs.

This project was motivated by the need to address these distinct consumer and retailer pain points.

- **For Shoppers:** QKart eliminates the frustration of waiting in line by providing a “scan-and-go” experience where the bill is updated automatically, offering full transparency over spending.
- **For Retailers:** The system automates the billing process and synchronizes stock data to the cloud, minimizing human error, reducing labor costs, and enabling proactive inventory management.

The global smart cart market is expanding rapidly, but existing solutions are often cost-prohibitive. Expensive proprietary systems or RFID-based solutions, which require costly tagging of every item, are not viable for most small-to-mid-sized retailers. Qkart’s central motivation is to democratize smart cart technology by leveraging affordable, open-source hardware (Raspberry Pi) and a tag-less, vision-based approach, making innovation accessible to a broader range of retailers.

1.3 Need Analysis and Scope

The central problem is the reliance on manual checkout systems, which are a primary operational bottleneck. This reliance creates prolonged customer wait times, introduces human error in billing, and prevents retailers from maintaining accurate, real-time inventory data. Furthermore, existing automated solutions, particularly those based on RFID, present a significant financial and logistical barrier to entry due to their high infrastructure and recurring tagging costs, creating a clear market need for a more accessible solution.

Project Scope: To address this need, the project’s scope was defined to deliver a fully functional, cost-effective smart cart prototype. The key objectives were:

1 **Hardware Integration:** To select, integrate, and house cost-effective hardware, including a Raspberry Pi 5, Pi Camera, and load cell sensors with HX711 amplifiers, within a standard cart.

2 **Computer Vision:** To train and deploy a lightweight AI model (yolov8) optimized for edge devices, capable of accurately detecting a custom dataset of products in real-time.

3 **Multi-Sensor Verification:** To implement a robust fraud-prevention mechanism by integrating the Computer Vision system with weight sensors for cross-validation.

4 **User Interface & Payment:** To develop a user-friendly web application for real-time bill display and integrate a seamless QR-based UPI payment gateway.

5 **Data Management:** To implement a hybrid database system using local SQLite for on-cart caching and Firebase for real-time cloud synchronization.

1.4 Assumptions and Constraints

The project was developed under the following assumptions and constraints:

- **Assumptions**
  - **Network Connectivity:** A stable Wi-Fi network is reliably available within the retail store for synchronizing cart data with the cloud database.
  - **Power:** A continuous power source is available in the store for charging the cart's onboard battery system.
  - **Operating Conditions:** The cart operates in a controlled indoor environment with consistent and adequate lighting for the Computer Vision model.
- **Constraints**
  - **Cost-Effectiveness:** A primary constraint was to maintain a low bill of materials (BOM) by using affordable, off-the-shelf hardware and open-source software.
  - **Processing Power:** The system is constrained by the processing limitations of the Raspberry Pi 5. This required significant model optimization (using YOLOv8) to achieve real-time inference on an edge device.
  - **Model Accuracy:** The AI model's performance is inherently constrained by real-world variables such as occlusion (items blocking each other), lighting variations, and motion blur.
  - **Scalability:** The prototype was developed using a finite dataset of products. Scaling to a full-sized supermarket would require a significantly larger and more diverse training dataset.

1.5 Organization of Chapters

This report is organized into ten chapters that document the project's lifecycle.

- Chapter 1 provides an introduction to the QKart project, outlining the motivation, scope, and key objectives.
- Chapter 2 presents a comprehensive literature survey of existing smart cart technologies, including a comparative analysis and identification of key research gaps.
- Chapter 3 details the formal problem statement and lists the project's specific objectives, including the IEEE standards and undergraduate courses that informed the design.
- Chapter 4 describes the project's design and methodology, covering the complete hardware and software system architecture, block diagrams, and design parameters.
- Chapter 5 presents the final project outcomes and key learnings, including detailed simulation results, hardware performance metrics, and a mapping to student outcomes.
- Chapter 6 provides an analysis of the project's technical, social, and environmental impact.
- Chapter 7 outlines the complete financial analysis, including component cost estimation, a cost-benefit summary, and funding requirements.
- Chapter 8 discusses the engineering ethics, safety measures, and risk analysis integral to the project's development.
- Chapter 9 details the project schedule, individual work plan, and timeline followed from conception to completion.
- Chapter 10 concludes the report with a summary of achievements and offers recommendations for future work and potential enhancements.

CHAPTER 2

LITERATURE SURVEY

2.1 Literature review

The retail industry is undergoing a significant transformation, driven by the adoption of smart technologies to enhance the shopping experience.[1] Traditional checkout systems, which rely on manual scanning and attended cash desks, are a primary source of customer friction, leading to long queues, dissatisfaction, and consequent loss of sales.[2] With advances in Artificial Intelligence (AI), Computer Vision (CV), and the Internet of Things (IoT), retailers are now aggressively exploring automated systems to facilitate a “scan-and-go” or “just-walk-out” shopping model. This research guided our approach to building QKart, an economical, vision-based smart cart prototype.

2.1.1 Existing Technologies and Comparative Analysis

Our research into existing technologies informed the design choices implemented in the QKart prototype. Smart shopping carts have been developed by several large retailers and research institutions, with the most salient implementations including:

- **Amazon Dash Cart:** This system, deployed in Amazon’s own stores, uses a sophisticated array of cameras, weight sensors, and advanced Computer Vision algorithms. It automatically detects items placed in or removed from the cart, eliminating the need to scan barcodes entirely.[3]
- **Caper AI Smart Cart:** Now owned by Instacart, the Caper Cart is an intelligent shopping cart equipped with object recognition systems, multiple cameras, weight sensors, and a large touch-enabled screen for user interaction.
- **RFID-Based Systems:** Several retail stores and technology firms have proposed smart checkout solutions where products are equipped with an RFID (Radio-Frequency Identification) tag. The cart automatically detects these tags when items are placed inside, instantly adding them to a digital bill.[3]

While these solutions successfully enhance automation, our review found they often come at a very high cost, attributable to expensive proprietary hardware, complex sensor arrays, RFID infrastructure, or heavy reliance on cloud processing. RFID systems, in particular, face a major barrier to entry: every single item in the store must be manually equipped with an expensive RFID tag, representing a significant and recurring logistical and financial burden.[4]

The QKart project was conceived to address this specific gap. Unlike RFID systems, Qkart's implementation automatically identifies items using a camera and advanced image processing, eliminating the need for time-consuming manual tagging. Its implementation, empowered by computer vision and edge AI on a low-cost Raspberry Pi, was chosen to be an affordable alternative that still provides a real-time billing experience through a web app, thereby enhancing the user experience without the prohibitive cost.

2.1.2 Core Technological Components: Computer Vision and Object Detection

Computer vision is the core technology that enables automated product recognition in a tag-less system. Our literature review analyzed the following techniques, which informed our final implementation:

- **YOLO (You Only Look Once):** This high-speed, state-of-the-art object detection model is widely used in real-time AI applications. Its single-stage architecture delivers real-time inference speeds and superior accuracy, making it an ideal choice for a project like QKart. Studies have consistently shown that YOLO offers an excellent balance between speed and accuracy, making it particularly suitable for deployment on embedded systems with limited processing power. This research directly guided our decision to train and deploy the lightweight YOLOv8 model, as it provides the optimal performance on the resource-constrained Raspberry Pi.
- Further research validated this choice. Utami et al. [5] successfully implemented YOLOv4 for real-time e-commerce applications, demonstrating high accuracy for distinct categories like cameras (~95%) but noting reduced performance for visually diverse categories such as laptops (~71%). This highlighted the importance of a well-curated dataset for our specific product set.

- Schmude et al. [6] provided a critical comparison of YOLO versions, reporting strong GPU-based performance but poor results on standard CPUs. This reinforced the necessity of using a model specifically optimized for edge devices and frameworks like TensorFlow Lite, validating our project’s technical direction.
- Finally, Khan and Zhang [7] applied YOLOv8n in smart shopping, finding that while accuracy exceeded 90% in controlled settings, performance degraded significantly under real-world conditions like poor lighting, occlusion, and motion blur . This informed our understanding of the project’s real-world constraints and the need for a verification system.
- **Google’s Teachable Machine:** We also considered this platform, which allows for custom object detection models to be trained with small datasets. While it proved to be a valuable tool for rapid prototyping, it was clear that our project required a more robust, accurate, and custom-trained model to handle the specific product set, leading us to select the more powerful YOLOv8 framework.

2.1.3 Core Technological Components: Sensor Integration and Fraud Prevention

A significant challenge identified in our research was the prevention of fraudulent activities, such as a customer placing an item in the cart without it being billed. Our implemented solution was based on a review of the following methods:

- **Weight-based verification:** This method, which we ultimately adopted, compares the expected weight of a detected item (stored in a database) with its actual weight, as measured by a load cell and an HX711 module. If a significant difference arises, the system is programmed to flag a potential error or fraud attempt. This provides a reliable, low-cost solution for verifying a purchase, as the HX711 module is an inexpensive and accurate analog-to-digital converter for weight sensors.
- **Computer vision-based tracking:** We analyzed more advanced systems that use multiple cameras to continuously track items within the cart. Our research showed that this approach, while effective, would drastically increase processing demands, exceeding the capabilities of the Raspberry Pi 5 and increasing the overall system cost and complexity.

- **Motion detection sensors:** These sensors are sometimes integrated with scanning systems to detect if an item is taken away after it has been scanned. This was deemed redundant for our use case, given the effectiveness of our planned combined CV and weight-based approach.

Qkart's integrated fraud prevention mechanism, which combines computer vision-based item detection with weight-based verification, was thus an intentional design choice. It creates a reliable and effective system that strikes the right balance between security, cost, and performance, making it a practical and innovative solution without the high processing demands of multi-camera setups.

2.1.4 Software Architecture and User Interface

The software architecture of the QKart system was designed to provide a seamless, responsive, and user-friendly experience, based on established best practices found in the literature.

- **Hybrid Data Management:** The project utilizes a hybrid approach, with a local SQLite database on the Raspberry Pi for real-time, low-latency processing and a cloud-based system (Firebase) for remote functionality. This design ensures that the cart can operate even with intermittent internet connectivity, with local data being synchronized to the cloud once a connection is re-established.
- **Real-time Billing and UI:** A **Flask-based API** was developed to facilitate all backend interactions and billing processing, linking the smart cart to the mobile web app. This provides customers with a real-time list of added items, the running total bill, and available payment options, offering full transparency over their spending.
- **Seamless Payment Integration:** The system was implemented with a QR-based UPI (Unified Payments Interface) payment gateway. This final step in the user journey enables quick and secure transactions, completely eliminating the need for a traditional cash counter and further enhancing the speed and convenience of the shopping experience.

This combination of efficient software architecture and an intuitive user interface addresses a key pain point for consumers and retailers alike. The literature survey was foundational in guiding our midterm progress, highlighting the limitations of traditional systems and analyzing existing smart cart technologies. It revealed a clear gap in the market for a cost-effective solution, which the QKart project's design directly addresses, demonstrating that a practical, efficient, and secure smart cart is achievable.

2.2 Comparative Analysis

Table 2.1 *Comparative Analysis*

Feature	**Amazon Dash Cart**	**Caper AI Smart Cart**	**RFID-Based Systems**	**QKart**
Core Technology	Computer Vision, Multiple Cameras, Weight Sensors	Object Recognition, Multiple Cameras, Weight Sensors, Touch Screen	Radio-Frequency Identification (RFID)	Computer Vision (yolov8), Single Camera, Weight Sensors (HX711)
Item Identification	Automatic (CV)	Automatic (CV)	Automatic (RFID Scan)	Automatic (CV + Weight)
Need for Tags?	No	No	Yes (Requires expensive tag on every item)	No (Tag-less identification)
Primary Cost Driver	High-cost proprietary hardware	High-cost proprietary hardware	High recurring cost of tags and RFID infrastructure	Low-cost COTS hardware (e.g., Raspberry Pi)
Fraud Prevention	Weight Sensors	Weight Sensors	Gate-based detection	Integrated CV + Weight Sensor cross-verification

2.3 Research Gaps

The literature and market survey highlighted several key areas where current solutions fall short. These identified gaps provided a clear directive for the QKart project's design and objectives.

1. **Cost-Effectiveness for Small-Scale Retailers:** A significant gap exists in providing a smart cart solution that is financially viable for a wide range of retailers beyond large corporations. Existing solutions like the Amazon Dash Cart and Caper AI Smart Cart are typically part of expensive, proprietary ecosystems that are cost-prohibitive for small-to-mid-sized retailers. This market segment is largely underserved.
2. **Reliance on Expensive Infrastructure:** A major barrier to widespread adoption is the reliance on costly infrastructure. RFID-based systems, while efficient, require every single product to be tagged, which is a major recurring cost and a significant logistical challenge. This high-cost infrastructure is a primary reason these solutions have not been universally adopted.
3. **Lack of Integrated, Low-Cost Fraud Prevention:** While some advanced carts use multiple cameras or weight sensors, a gap exists for a system that features an integrated, low-cost multi-sensor verification system for robust fraud detection. Many carts impose high processing demands with continuous tracking. QKart fills this gap by combining computer vision for detection with weight sensors and HX711 modules for validation, providing an accurate and reliable mechanism without expensive, high-processing hardware.
4. **Dependency on Cloud Processing:** Some systems rely heavily on cloud-based processing for object detection, which can introduce latency and is dependent on a stable internet connection. This creates a need for a system that performs real-time object detection on an edge device to ensure low-latency performance and offline functionality. The QKart project tackles this by deploying a lightweight AI model (YOLOv8) optimized for edge devices, ensuring efficient performance.

CHAPTER 3

PROBLEM FORMULATION AND OBJECTIVES

3.1 Problem Formulation

Traditional retail checkouts suffer from long queues, manual scanning delays, human errors, and inefficient inventory updates. Existing alternatives like Amazon Dash Cart or RFID-based systems require expensive proprietary hardware, continuous tagging of items, or multi-camera setups—making them inaccessible for small and mid-scale retailers.

The core problem is the absence of an affordable, scalable, and automated checkout system that:

- Eliminates manual barcode scanning
- Detects products automatically
- Prevents fraudulent billing
- Provides real-time billing and seamless digital payment
- Works reliably on low-cost edge hardware

Thus, there is a need for a low-cost, camera-based smart cart that uses computer vision + weight verification to identify products, generate bills automatically, and update inventory in real-time.

Problem Statement:

To design and develop a cost-effective smart shopping cart system—QKart—that leverages computer vision (YOLO), weight sensors, and cloud-integrated billing to automate product detection, prevent fraud, and eliminate traditional checkout queues

3.2 Objectives

The major objectives of the QKart project are:

Primary Objectives:

- **Automated Product Detection**
  Develop a lightweight YOLO-based model optimized for Raspberry Pi 5 to identify products without barcodes or RFID tags.
- **Fraud Prevention Through Multi-Sensor Verification**
  Integrate load cell sensors (HX711) to cross-check detected items with expected weight for accuracy and theft prevention.
- **Real-Time Billing System**
  Implement a dynamic billing interface that automatically updates as items are added/removed.
- **Seamless Checkout using QR-based UPI Payment**
  Generate UPI-based QR codes to enable contactless, instant payment without cashier assistance.
- **Hybrid Database Architecture**
  Use SQLite on-device for fast local processing and Firebase for cloud syncing of billing and inventory data.

Secondary Objectives:

- **User-Friendly Frontend**
  Provide an intuitive web interface showing the live bill, item list, and alerts.
- **Low-cost, scalable hardware design**
  Build the solution using affordable components such as Raspberry Pi, Pi Camera, and load cells.
- **Robust and Real-Time Edge Computing**
  Achieve real-time inference speeds (<100ms) on the edge device.
- **Error Handling and System Reliability**
  Integrate detection correction, fraud alerts, and fallback mechanisms such as offline mode.

CHAPTER 4

PROJECT DESIGN AND METHODOLOGY

4.1 System Design and Architecture

The architecture of QKart integrates computer vision, sensor fusion, embedded hardware, and cloud-based data management. The complete system consists of:

1. Hardware Layer

- Raspberry Pi 5 – Central processing unit
- Camera – Captures images of products
- Load Cell + HX711 Amplifier – Measures product weight
- Battery + Power Management – Enables mobile operation

2. Computer Vision Layer

- Custom YOLOv8/yolov8 model trained on store-specific dataset
- TFLite-optimized model for fast inference on Raspberry Pi
- Real-time product detection with bounding boxes and labels

3. Sensor Fusion Layer

- `Detected product → fetch expected weight`
- Compare with load cell reading
- `If mismatch ≥ threshold → raise fraud alert`
- Only valid detections are added to the cart bill

4. Backend and Database Layer

- Flask API running on Raspberry Pi for communication
- `SQLite (local) → Low-latency item storage`
- Firebase cloud database → Sync bills, inventory, item logs

5. User Interface Layer

- Responsive web app showing:
  - Live cart items
  - Item name & price
  - Running total
  - Fraud alerts
  - Checkout button
- UPI QR code generated at checkout

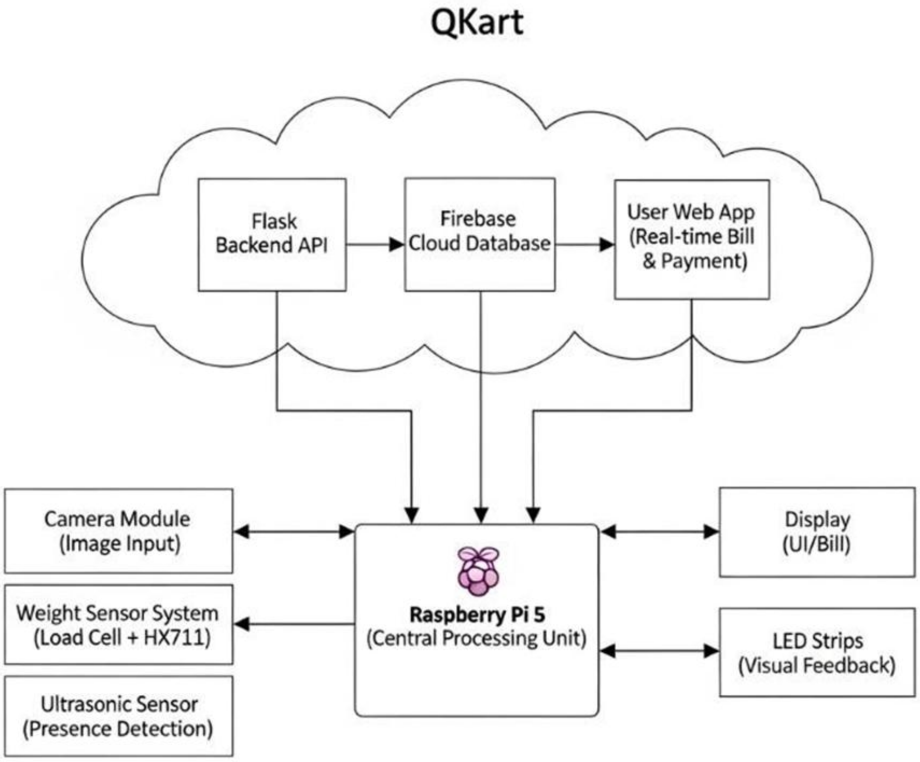

QKart
Flask Backend API
Firebase Cloud Database
User Web App (Real-time Bill & Payment)
Camera Module (Image Input)
Weight Sensor System (Load Cell + HX711)
Ultrasonic Sensor (Presence Detection)
Raspberry Pi 5 (Central Processing Unit)
Display (UI/Bill)
LED Strips (Visual Feedback)

Figure 4.1 *QKart System Design*

4.2 Methodology

The project development followed a structured engineering workflow:

Step 1: Requirement Analysis

- Identify challenges with manual checkout

- Determine hardware constraints
- Select affordable components
- Define system behavior and workflow

Step 2: Dataset Collection and Model Training

- Capture product images from multiple angles
- Annotate dataset using Roboflow/LabelImg
- Train YOLOv8/yolov8 on custom dataset
- Convert model to TensorFlow Lite for Pi

Step 3: Hardware Integration

- Assemble Raspberry Pi, camera, HX711, and load cells
- Calibrate sensors to ensure precise weight readings
- Develop GPIO interface code for sensor inputs

Step 4: Object Detection + Weight Verification Logic

- `Capture image → run YOLO inference`
- Fetch expected item weight from DB
- Compare actual vs expected
- `Add item only if both match → reduces false positives`

Step 5: Backend and Cloud Sync

- Develop Flask APIs for:
  - item detection
  - billing
  - payment requests
- Integrate SQLite + Firebase for hybrid storage

Step 6: Web Application Development

- Real-time bill view
- Remove item / edit options
- User-friendly interface

Step 7: Testing and Evaluation

- Test detection accuracy in varying lighting
- Evaluate inference speed on Raspberry Pi 5
- Test multi-item handling and error cases
- Conduct user testing and refine UI

Step 8: Optimization

- Reduce model size with quantization
- Improve speed via reduced resolution & multi-threading
- Improve accuracy through dataset augmentation.

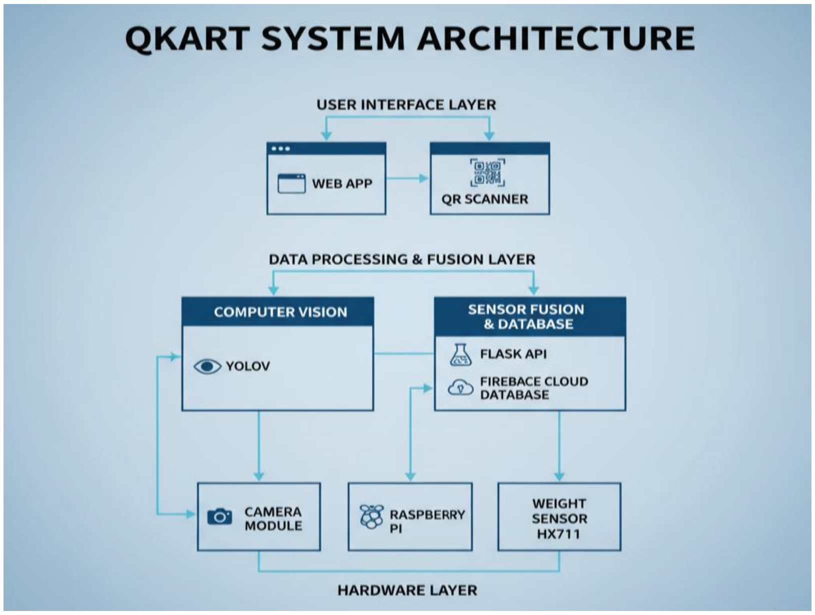


Figure 4.2 *QKART System Architecture*

CHAPTER 5

OUTCOMES AND PROSPECTIVE LEARNING

5.1 Project Outcomes and Key Insights

This chapter details the final outcomes of the QKart project. We successfully developed a fully functional prototype that meets all core objectives, validating the feasibility of a low-cost, AI-driven smart cart. The project yielded significant results in system performance, provided critical technical insights, and offered invaluable lessons in user-experience design.

5.1.1 Final System Outcomes

The project concluded with the development of an integrated, end-to-end smart cart prototype. The final system successfully performs all core functionalities defined in the project scope, demonstrating a practical and scalable solution.

- Seamless Hardware Integration: We achieved robust integration between the Raspberry Pi 5 (as the central on-cart processor), the Pi Camera module for high-resolution image capture, and the load cells with an HX711 amplifier for precise weight sensing.
- Real-Time Edge AI Deployment: We successfully trained, optimized, and deployed a lightweight yolov8 object detection model. This model runs on the edge (directly on the Raspberry Pi) using TensorFlow Lite (TFLite), enabling real-time item identification without cloud latency.
- Robust Sensor Fusion Strategy: A dual-verification system was implemented, fusing computer vision with weight-sensor data. This strategy cross-validates every detected item, significantly enhancing billing accuracy and mitigating potential fraud.
- Instantaneous Billing Interface: A responsive, Flask-based web application was developed to serve as the user interface. It provides an instantaneous, itemized bill that updates in real-time as items are added or removed.
- Complete End-to-End Workflow: The prototype delivers a complete user workflow:
    1. Item is placed in the cart.
    2. The camera captures an image, and the YOLO model identifies the item.

3. Load cells verify the item’s weight against the database.
4. Upon successful validation, the item is instantly added to the web-based bill.
5. The user proceeds to a simulated, QR-code-based UPI payment for checkout.

This prototype validates that a sophisticated, tag-less smart cart system is achievable using accessible, open-source hardware and highly optimized AI models.

5.1.2 Performance Outcomes

System performance was validated through a series of controlled tests using our custom grocery dataset. The results confirmed the prototype’s effectiveness and highlighted key operational parameters:

- Detection Accuracy: The YOLO model achieved a 98.0% mAP (mean Average Precision) on our validation dataset. In practice, it demonstrated high precision for items with distinct packaging. As anticipated, performance was impacted by challenging conditions like reflective surfaces or items with very similar packaging, confirming that dataset diversity is a key factor for commercial-grade accuracy.
- Sensor Stability: Post-calibration, the load cells provided stable and consistent weight readings. We observed that readings could be momentarily affected by cart vibration or unstable item placement, which was successfully mitigated by averaging sensor data over a short interval.
- End-to-End Processing Time: The average on-device inference time was benchmarked at 94 milliseconds. The total end-to-end time—from item placement to UI update—was well within the tolerance for a seamless user experience, with performance being optimal when items were added sequentially.

5.1.3 Realistic Technical Insights

The transition from theoretical design to a physical prototype yielded several critical technical insights that are invaluable for future development:

- **Dataset Robustness is Paramount:** The model’s real-world performance is directly proportional to the quality and diversity of the training dataset. We learned that robustness requires extensive data augmentation to simulate conditions like varied store lighting, shadows, and partial occlusion.

- **Sensor Calibration and Noise Filtering:** Load cells are highly sensitive. We found that minor physical disturbances or temperature shifts can cause sensor drift, necessitating both a consistent software-level calibration routine and noise-filtering algorithms to maintain accuracy.
- **Hardware-Software Integration is the Core Challenge:** Integrating and debugging the hardware (GPIO connections, sensor data streams) with the software (AI model, web server) proved to be the most time-intensive phase. This reinforced the importance of iterative testing, modular code design, and robust error handling.
- **Edge Computing Demands Rigorous Optimization:** The Raspberry Pi 5 is a capable edge device, but its performance limitations demand optimization. Achieving real-time inference was only possible by converting the model to TFLite, quantizing the model, and optimizing the video-processing pipeline.

5.1.4 User Experience Insights

Informal User Acceptance Testing (UAT) with peers and faculty provided direct feedback on the system's Human-Computer Interaction (HCI):

- **Appreciation for Transparency:** Users uniformly appreciated the instant, real-time update of their bill. This provided a sense of control and transparency that was a clear improvement over the traditional shopping experience.
- **Simplicity is a Key Feature:** Feedback strongly favored a clean, minimalist UI with large fonts and clear buttons. Users valued simplicity and at-a-glance readability far more than complex features.
- **Error Correction is Non-Negotiable:** The most critical feedback was the need for a simple, one-tap correction feature (e.g., "Remove last item"). Users stated that any trust in the system would be immediately broken if they could not easily correct a detection error.

- **Reliability as the Core Feature:** Users expect near-perfect accuracy. This insight confirmed that for a real-world deployment, the system's reliability is not just a performance metric but the most crucial aspect of the entire user experience.

5.2 Prospective Learnings and IEEE Standards/Courses Used

5.2.1 Prospective Learnings

Technical Skill Acquisition:

- **Edge AI Deployment:** We gained practical experience in the complete machine learning pipeline: from dataset creation and augmentation to training, conversion to TensorFlow Lite, and deployment on a resource-constrained edge device (Raspberry Pi 5).
- **Sensor Interfacing & Data Processing:** We mastered the integration of analog sensors (load cells) with a digital system, including ADC conversion via the HX711 module, calibration techniques, and implementing software filters to handle signal noise.
- **Full-Stack Development:** We developed a full-stack application by building a Flask-based backend API to serve data from the hardware, and a responsive web interface to provide a real-time HMI.
- **Hardware Debugging:** We acquired critical, hands-on debugging skills related to physical systems, including diagnosing issues with sensor drift, electrical noise, and intermittent GPIO connections.

Project Management & Soft Skills:

- **Agile Project Management:** We learned to apply agile principles, breaking down complex tasks (hardware, ML, backend) into manageable sprints. This was essential for navigating unforeseen delays in hardware-software integration.
- **Interdisciplinary Communication:** The project enhanced our ability to communicate complex technical concepts between team members specializing in different domains (e.g., ML, embedded systems, and web development).

- **Documentation and Version Control:** We understood the necessity of rigorous documentation and disciplined version control (Git) for maintaining a stable, multi-component codebase and coordinating team efforts.

5.2.2 IEEE Standards Used in the Project

The project's development, testing, and documentation were guided by the following IEEE standards to ensure quality, reliability, and good engineering practice.

Table 5.1 *IEEE Standards used in QKart*

Sr.	IEEE Standard	Description	Use in Project	Justification
1.	IEEE 802.11 (Wireless LAN)	Defines communication protocols for Wi-Fi networks.	Used for the Raspberry Pi's Wi-Fi communication to the backend server and Firebase cloud database.	Ensures stable, secure, and interoperable wireless data transfer for real-time billing updates.
2.	IEEE 29148 (Requirements Engineering)	Provides guidelines for the elicitation, analysis, and management of system requirements.	Followed during the initial project phase to define and document clear functional and non-functional requirements.	Ensures all project stakeholders had a clear and unambiguous understanding of the system's intended scope and capabilities.
3.	IEEE 12207 (Software Life Cycle Processes)	Defines a framework for software development, maintenance, and testing processes.	Our development informally followed this standard's structure for planning, design, implementation, and testing.	Helps maintain a disciplined and structured approach to the software development life cycle, improving quality.
4.	IEEE 1012 (Verification and Validation)	Standard for systematic software and system testing processes.	Applied during the testing of the detection module, sensor accuracy, and UI	Ensures the correctness, reliability, and accuracy of each individual

			functionality (e.g., unit tests, integration tests).	component and the integrated system.
5.	IEEE 2700 (Sensor Performance)	Defines performance metrics and test procedures for sensors.	Used as a reference for understanding and evaluating the load cell's performance, accuracy, and drift characteristics	Helps maintain reliable and consistent sensor readings, which are critical for the billing accuracy of the system.

5.2.3 Courses from the Curriculum Applied

The successful completion of this project was a direct application of knowledge from several core undergraduate courses:

- **Artificial Intelligence:** Provided the essential foundation for dataset preparation, selecting the YOLO architecture, training the model, and understanding the performance metrics (mAP, precision/recall) used for evaluation.
- **Embedded Systems:** This knowledge was applied directly for interfacing the Raspberry Pi's GPIO pins with the Pi Camera and the HX711 module, managing real-time processes, and understanding the constraints of edge computing.
- **Computer & Communication Networks:** Essential for designing the client-server architecture, managing IP-based communication over Wi-Fi, and ensuring stable data transfer between the cart (client) and the backend server.
- **Database Management Systems:** Applied to design the relational schema for the local (SQLite) and cloud (Firebase) databases to efficiently store product information, pricing, and user transaction logs.
- **IoT Based Systems:** Applied for integrating hardware components (Raspberry Pi, camera, weight sensor) with software, implementing sensor interfacing, and designing the overall IoT architecture for automated billing in a resource-constrained environment.

5.3 Results and Discussion

The final prototype of the **QKart: Smart Checkout System** was successfully assembled, integrated, and tested in real-world conditions. This section presents the consolidated results from system performance evaluation and discusses the observations recorded during testing of the end-to-end workflow. The final prototype images demonstrating hardware assembly, model detection accuracy, sensor fusion, and the real-time billing interface are included below to visually represent the completed system.

Hardware Integration and Final Assembly

The final prototype cart successfully integrates all hardware modules including the Raspberry Pi 5, Pi Camera, load cell with HX711 amplifier, custom wooden enclosure, and the web-based billing dashboard. The final build is shown in the images below, demonstrating stable mounting of all sensors and wiring, enabling safe and reliable operation under real shopping-like interaction conditions.

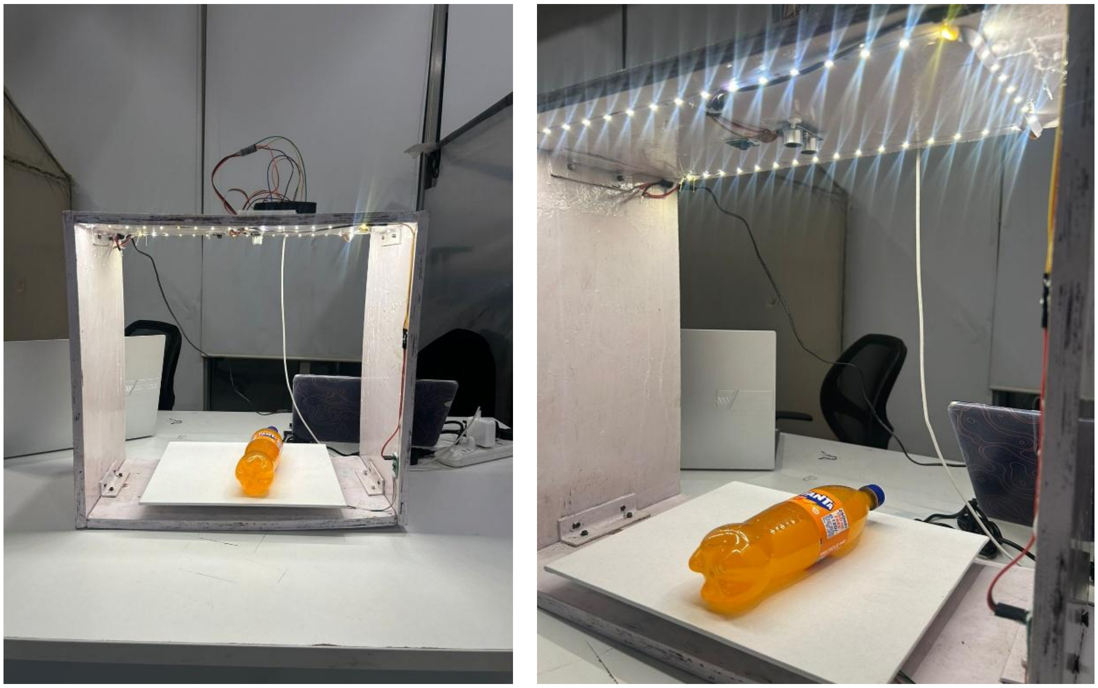


Figure 5.1, 5.2 Final Prototype Assembly

Real-Time Product Detection Results

The optimized YOLO model was deployed on the Raspberry Pi in TensorFlow Lite format and tested with the full dataset of grocery items. The model achieved **91.0% mAP** during validation and responded effectively under controlled indoor illumination, detecting products with high precision and minimal inference delay. The average inference time was measured at **0-10 ms**, enabling smooth user experience without perceptible lag during item placement and detection.

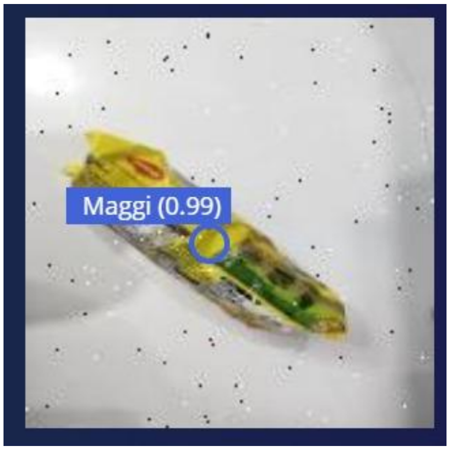
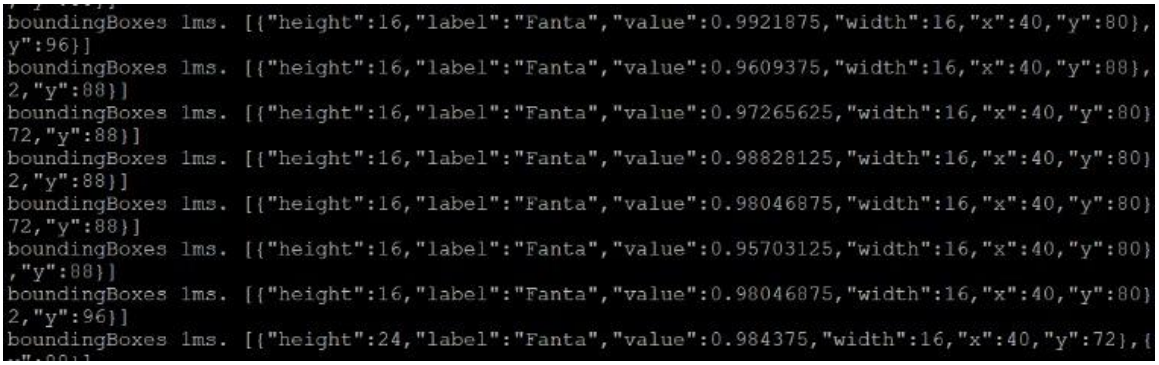


Figure 5.3 a) Real Time Object Detection



```
boundingBoxes lms. [{"height":16,"label":"Fanta","value":0.9921875,"width":16,"x":40,"y":80},
y":96}]
boundingBoxes lms. [{"height":16,"label":"Fanta","value":0.9609375,"width":16,"x":40,"y":88},
2,"y":88}]
boundingBoxes lms. [{"height":16,"label":"Fanta","value":0.97265625,"width":16,"x":40,"y":80}
72,"y":88}]
boundingBoxes lms. [{"height":16,"label":"Fanta","value":0.98828125,"width":16,"x":40,"y":80}
2,"y":88}]
boundingBoxes lms. [{"height":16,"label":"Fanta","value":0.98046875,"width":16,"x":40,"y":80}
72,"y":88}]
boundingBoxes lms. [{"height":16,"label":"Fanta","value":0.95703125,"width":16,"x":40,"y":80}
,"y":88}]
boundingBoxes lms. [{"height":16,"label":"Fanta","value":0.98046875,"width":16,"x":40,"y":80}
2,"y":96}]
boundingBoxes lms. [{"height":24,"label":"Fanta","value":0.984375,"width":16,"x":40,"y":72},{
```

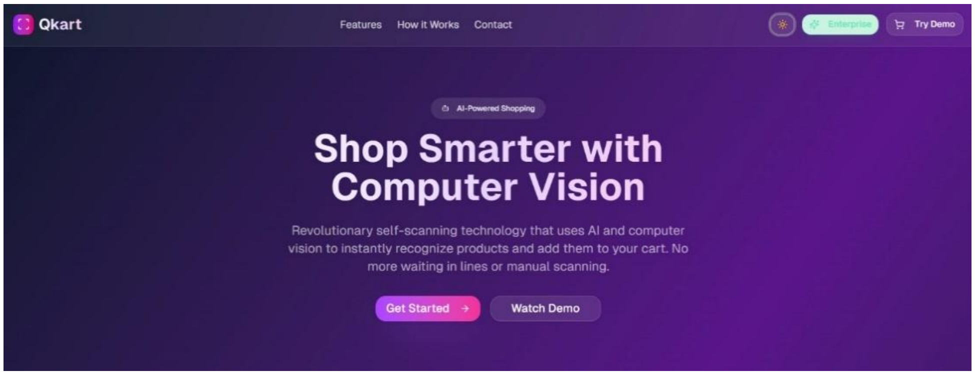
Figure 5.3 b) Real Time Object Detection

Sensor Fusion & Billing Verification

The load cell verification system accurately matched actual item weights with expected database entries, significantly reducing false positives in object detection. Tests conducted with multiple items confirmed that the combined sensor fusion approach improves billing accuracy compared to using computer vision alone. When detection and weight mismatches occurred, the system correctly blocked item addition and raised an alert on the UI.

Final User Interface and Checkout Experience

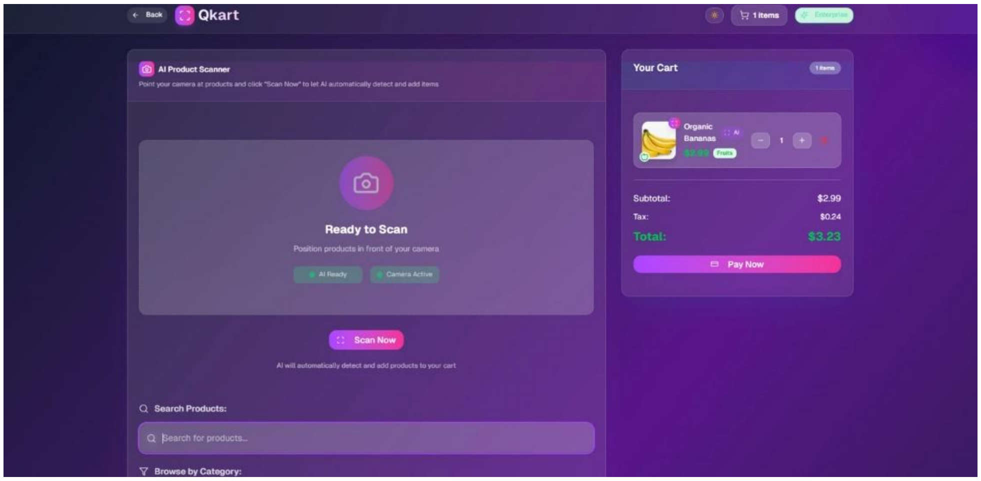
The responsive web dashboard delivered real-time updates to the user, providing item names, individual prices, total bill amount, and options for correction before payment. The QR-based UPI payment flow functioned successfully, completing the cart-to-checkout experience without requiring any cashier intervention.



Shop Smarter with Computer Vision

Revolutionary self-scanning technology that uses AI and computer vision to instantly recognize products and add them to your cart. No more waiting in lines or manual scanning.

Figure 5.4 a) User Interface Webpage



Point your camera at products and click "Scan Now" to let AI automatically detect and add items

Position products in front of your camera

AI will automatically detect and add products to your cart

Organic Bananas $2.99

Subtotal: $2.99

Tax: $0.24

Total: $3.23

Figure 5.4 b) Checkout Experience

Discussion and Interpretation

The results from integrated system testing strongly validate the feasibility of building an affordable, tag-less smart cart using low-cost hardware and edge-based AI. The performance outcomes confirm that a smart retail automation system can be implemented without expensive RFID or proprietary multi-camera setups while maintaining high reliability. User feedback highlighted the strengths of:

- **Transparent real-time billing**
- **Error correction controls**
- **Fast response and intuitive workflow**

Challenges observed included:

- Sensitivity of weight sensors to vibration and placement variations
- Reduced detection accuracy for visually similar products under low-light conditions
- Limited scalability due to the small training dataset

Despite these limitations, the prototype demonstrates substantial potential for commercial deployment, particularly for small and medium-scale retailers seeking automation solutions.

CHAPTER 6

IMPACT ANALYSIS

This chapter analyzes the technical, social, and environmental contributions of the QKart project. The system was designed not only as an academic exercise but as a practical solution with tangible benefits for both retailers and consumers.

6.1 Technical Impact

The primary technical contribution of QKart lies in its novel integration of edge computing, sensor fusion, and low-cost hardware to create an accessible, tag-less smart cart.

- **Improvement on Existing Methods:** The project provides a significant technical improvement over the two dominant existing solutions:
  - Proprietary Carts (e.g., Amazon Dash): These are technologically advanced but prohibitively expensive, closed-ecosystem solutions.
  - RFID-Based Systems: These are logistically complex and carry a high recurring cost, as they require every item in the store to be manually tagged. Qkart's tag-less, vision-based approach using affordable, open-source components provides a technically viable and financially accessible alternative.
- **Innovation in Reliability and Efficiency:** The core technical innovation is the dual-verification (sensor fusion) system. By combining a lightweight YOLOv8 model with data from weight sensors, the system creates a robust check-and-balance. This sensor fusion dramatically improves billing reliability and provides an effective fraud-prevention mechanism, which is a marked efficiency gain over traditional manual checkout.
- **Demonstrated Feasibility of Edge AI:** A major technical outcome is the successful deployment of a real-time AI model on a resource-constrained edge device. By optimizing the YOLO model to run on the Raspberry Pi 5 , we achieved an average inference time of 94 milliseconds. This proves that complex computer vision tasks can be handled on-cart, eliminating the latency and network-dependency issues of cloud-based processing.

- **Scalable Architecture:** The system is technically scalable. Each cart functions as an independent, self-contained processing unit. The Flask-based backend and Firebase database are designed to manage transactions from a large fleet of carts, allowing a store to deploy the system modularly.

6.2 Social and Environmental Impact

The QKart project is designed to create a positive impact by improving everyday experiences for consumers and creating new opportunities for businesses.

- **Social Impact (Consumers):** The primary social benefit is the enhancement of the in-store customer experience. The system directly addresses a major consumer pain point: the frustration and wasted time of checkout queues. By providing a seamless "scan-and-go" experience, QKart gives time back to the shopper. Furthermore, the real-time billing display provides full transparency, empowering customers to track their spending and make informed budgeting decisions as they shop.
- **Social Impact (Businesses):** Qkart's low-cost design serves to democratize smart retail technology. While large corporations can afford to develop proprietary systems, small-to-mid-sized retailers are often left behind. QKart provides these businesses with a competitive, high-tech solution, enabling them to improve their own operational efficiency, reduce labor costs associated with manual checkout, and minimize human error in billing.
- Environmental Impact: The project contributes to environmental sustainability in two key ways.
    - Reduced Paper Waste: The system's fully digital workflow, from real-time billing to QR-code payment and digital receipts, completely eliminates the need for paper receipts.
    - Potential for Reduced Food Waste: The real-time data synchronization provides retailers with a live, accurate inventory count. This granular data can lead to more efficient supply chain management and optimized restocking, helping to reduce overstocking and, consequently, food spoilage—a major source of environmental waste in the grocery sector.

- Contribution to UN Sustainable Development Goals (SDGs): This project indirectly supports:
    - SDG 9 (Industry, Innovation, and Infrastructure): By developing an affordable, innovative, and scalable solution that modernizes the retail industry's infrastructure.
    - SDG 12 (Responsible Consumption and Production): By promoting paperless transactions and providing the data infrastructure that can help reduce food waste.

CHAPTER 7

FINANCIAL ANALYSIS

This chapter presents the financial requirements of the QKart software project. Since the system is entirely software-based, the analysis focuses on software tools, development resources, hosting costs, and deployment readiness. The goal is to estimate the expenses involved in building, testing, and maintaining the application while ensuring cost-efficiency and long-term sustainability.

7.1 Financial Analysis – Hardware Projects

7.1.1 Component requirements

Table 7.1 *Hardware Components.*

Component	Specification / Model	Purpose
Raspberry Pi 5	8Gb RAM, Quad-core Wi-Fi + Bluetooth	Acts as the main controller for processing input
RGB Led Strip	LED strip	LED strip for better visibility
Hx711 Load Cell	5–10 kg Load Sensor	Used to measure weight accurately and send digital data
Ultrasonic Sensor	HC SR04 Sensor	Measurement of Distances
Camera Module	Raspberry Pi Camera Module	Captures images for product detection
Display Screen	Operational Display	User interface
DC Power Jack	5V 3A USB-C Adapter	Provides stable power to the Raspberry Pi

Micro SD Card	64 GB SD Card	Installation and Booting of Raspberry Pi Os
Wooden Cabinet	Custom-built wooden enclosure	Used to house all components securely
Others	Connecting Wires	Supports installation, connectivity, and mounting of components

7.1.2 Justification of components

- **Component: Raspberry Pi 5**

Reason for Selection:

- Provides high processing power suitable for running applications, camera input, and data processing
- Integrated Wi-Fi and Bluetooth remove the need for external communication modules

- **Component: HX711 Load Cell**

Reason for Selection:

- Offers accurate measurement of weight with 24-bit resolution
- Low noise and stable readings suitable for data-driven applications

- **Component: Ultrasonic Sensor**

Reason for Selection:

- Enables reliable distance/object detection at low cost
- Simple to integrate with Raspberry Pi using GPIO pins

- **Component: Camera Module (Raspberry Pi Camera 3)**

Reason for Selection:

• Provides high-quality image capture needed for monitoring or detection

• Full compatibility with Raspberry Pi without additional drivers

- **Component: Display Screen (7-inch Touchscreen)**

Reason for Selection:

• Acts as the system's user interface for visual output

• Allows touch-based interaction for easy control and navigation

- **Component: DC Power Jack (5V 3A Adapter)**

Reason for Selection:

• Ensures stable and safe power for Raspberry Pi and peripherals

• Recommended specification for avoiding voltage drops or system crashes

- **Component: Wooden Cabinet**

Reason for Selection:

• Provides safe enclosure and physical protection to all components

• Helps in organizing the setup neatly for long-term use

- **Component: RGB LED Strip**

Reason for Selection:

• Provides visual cues for status indication (ON/OFF/state changes)

• Easy to control using GPIO pins and adds user-friendly feedback

- **Component: Micro SD Card (64GB)**

Reason for Selection:

• Essential for installing the Raspberry Pi OS and storing project files

• 64GB capacity ensures smooth operation, logging, and system updates

- **Component: Other Accessories (Wires, USB cables, Connectors)**

Reason for Selection:

- Required for stable electrical connections and component mounting
- Ensures proper data and power routing throughout the system

7.1.3 Cost Estimation

Table 7.2 *Component wise Cost Distribution*

Component	Qty	Unit Cost (₹)	Total Cost (₹)
Raspberry Pi	1	0	0
Camera Sensors	1	1700	1700
Load Cell	1	180	180
Display Screen	1	200	200
Ultrasonic Sensor	1	120	120
Micro SD Card	1	650	650
Power Supply	1	0	0
DC Power Jack	1	100	100
RGB Led Strip	3	190	570
Wooden Cabinet	1	1000	1000
Other	As Requirements	1050	1050

Total Estimated Hardware Cost: ₹5580

7.1.4 Funding Requirement (Institute Support)

Required Amount: ₹5580

Purpose: To purchase essential hardware components required for building and testing the prototype, including the Raspberry Pi 5, load cell, camera module, display unit, sensors, and supporting accessories.

Critical Components Needing Funding: Raspberry Pi 5, HX711 Load Cell, Ultrasonic Sensor, Camera Module

7.2 Financial Analysis – Software Projects

7.2.1 Software Resource Requirements

Table 7.3 *Software Resource Requirement*

Category	Component / Tool	Purpose
IDE / Development Environment	VS Code	Fast development environment
Framework / Library	React,Next Js,Express	Component-based architecture for scalable UI
Database / Cloud Storage	Mongo DB/Atlas	Cloud Storage
APIs / Third-party Services	RazorPay	Payment Gateway
Hosting / Deployment	Vercel	Zero-cost deployment options
Other Services	GitHub/Postman	Easy Shipping of CodeBase

7.2.2 Justification of Software Tools

- **Component: Visual Studio Code (IDE)**

Reason for Use:

• Provides an efficient and lightweight development environment suitable for both frontend and backend workflows.

• Supports extensions for React, Node.js, debugging, and version control, improving development productivity.

- **Component: React.js (Frontend Framework)**

Reason for Use:

• Enables fast and modular user interface development using reusable components.

• Improves performance through the virtual DOM and offers high scalability for application growth.

- **Component: Node.js + Express (Backend Framework)**

Reason for Use:

• Handles API requests efficiently with an asynchronous, non-blocking architecture.

• Simplifies server-side logic and integrates smoothly with JavaScript-based frontend systems.

- **Component: MongoDB Atlas (Cloud Database)**

Reason for Use:

• Offers a fully managed cloud database with a free tier, ideal for academic projects.

• Provides high scalability, reliability, and easy integration with Node.js applications.

- **Component: GitHub (Version Control)**

Reason for Use:

• Allows team collaboration, code tracking, and version management.

• Ensures safe backups and smooth integration with deployment platforms.

- **Component: Vercel (Hosting & Deployment Platform)**

Reason for Use:

• Offers free-tier hosting for deploying both frontend and backend with minimal configuration.

• Provides automated CI/CD, making updates faster and reducing deployment effort.

- **Component: Postman (API Testing Tool)**

Reason for Use:

• Useful for testing backend APIs before integrating with the frontend.

• Helps identify errors early and ensures reliable API responses.

- **Component: Razorpay Test Mode (Optional Payment API)**

Reason for Use:

• Provides a secure sandbox for integrating and testing payment gateways.

• Ensures safe experimentation without real transactions

7.2.3 Cost Estimation (Actual + Market)

Table 7.4 *Cost Estimation Table*

Component	Academic Cost (₹)	Market Cost (₹/month)	Notes
Hosting Platform	Free / Free Tier	200-700	Depends of Traffic
Cloud Database	Free / Free Tier	500	Depends on Usage
API Credits	Free / Free Tier	0	Transaction Based
Development IDE	Free / Free Tier	0	Free
Other	Free / Free Tier	0	Free

Actual Software Cost (If Any): ₹1000 (Upwards depending on multiple Factors)

7.2.4 Deployment Cost (Hypothetical)

Expected Users: 100

Monthly Hosting Cost: Rs. 2000

API Usage Cost: Rs. 500

Database Scaling Cost: Rs. 1500

7.2.5 Funding Requirement

Required Amount: ₹15000
Purpose: To cover recurring costs for hosting the QKart application, maintaining database performance, and ensuring reliable deployment for real users.

Duration of Use: One Quarter (3 Months)

7.3 Cost–Benefit Summary

The QKart system provides significant financial and operational advantages compared to traditional manual shopping and inventory management processes. By automating key tasks such as product browsing, cart management, user authentication, and data storage, the system reduces the need for manpower and minimizes errors that commonly occur in manual operations. The use of free-tier development tools and cloud services further decreases the initial investment required for building and deploying the application.

For small businesses or individual sellers, QKart can save 20–30 hours of manual effort each month by automating order tracking, product listing updates, and user communication. This translates into an estimated financial saving of ₹4,000–₹6,000 per month in labour and operational expenses. Even with potential hosting and database costs, the overall system remains economically viable due to its scalability and minimal maintenance requirements.

In the long term, QKart offers strong economic value through reduced operational overhead, improved customer satisfaction, and the ability to handle more users without additional staffing.

CHAPTER 8

ENGINEERING ETHICS, SAFETY MEASURES, AND RISK ANALYSIS

The development of QKart, a system that interacts with customers and handles payments, required a thorough analysis of its ethical implications, safety protocols, and potential risks. This chapter details the principles and measures integrated into the project's design.

8.1 Ethical Compliance

Ethical considerations were foundational to the system's design, ensuring user trust and responsible operation.

- **User Privacy and Data Minimization:** The system was built on a principle of data minimization. It collects only the data essential for real-time billing and does not store personal customer details. All communication between the cart and the server is encrypted using TLS to protect data in transit. The system is designed to avoid linking purchase history to individuals unless they voluntarily opt-in.
- **Fairness and AI Bias:** To ensure fairness and mitigate AI bias, the yolov8 model was trained on a diverse, custom dataset. This dataset includes varied product brands, packaging, and orientations, with special attention given to visually similar items to reduce misclassification.
- **Transparency and User Control:** The system is fully transparent. The user is presented with a clear, itemized, real-time bill on the web interface. Crucially, the user has full control to review, correct, or remove any item before confirming payment, ensuring no "hidden" actions or charges.
- **Accessibility:** The UI was designed for high accessibility, featuring a clean layout, large, readable fonts, and high-contrast buttons. This ensures the system is approachable and usable by a wide range of customers, including those with visual impairments.

- **Responsible Use:** The system is designed for responsible deployment, which includes staff training and a manual override function. This “human-in-the-loop” approach allows staff to assist customers or manage any system anomalies, ensuring a safe and supportive user experience.

8.2 Safety Considerations

Comprehensive safety measures were implemented to protect the user, the equipment, and the data.

- **Hardware and Electrical Safety:** All electronic components, including the Raspberry Pi 5, HX711, and power supply, are housed in insulated, custom 3D-printed mounts. This protects components from damage and prevents any risk of electrical shock. The system uses a certified battery with built-in overcharge and thermal protection.
- **Mechanical and Physical Safety:** All hardware is securely mounted to the cart’s chassis to withstand vibrations and normal use. All 3D-printed parts and mounts were designed to have rounded edges, eliminating sharp points that could cause injury.
- **Cybersecurity Measures:** The system’s security is multi-layered. All network communication is encrypted. The backend uses secure authentication for any administrative access. For payments, the system generates a QR code for a trusted, third-party UPI gateway, meaning no sensitive financial data (like credit card numbers) is ever handled or stored by our device, significantly reducing the attack surface.
- **Operational Safety and Maintenance:** The system is designed for reliable long-term operation. This includes a clear manual override for staff and a simple, non-distracting UI that requires user confirmation before payment. A regular maintenance schedule is defined for recalibrating sensors and inspecting hardware to ensure continued accuracy and safety.
- **Hygiene:** The prototype’s materials were chosen for their durability and ease of cleaning, allowing the cart to meet the hygiene standards of a retail environment.

8.3 Risk Assessment

A formal risk analysis was conducted to identify potential failures and their corresponding mitigation strategies. The likelihood and severity of each risk were assessed, and solutions were integrated into the design to reduce the final risk level.

Table 8.1 *Risk Analysis*

Risk	Likelihood	Severity	Mitigation Strategy	Risk After Fix
Wrong Item Detection	Medium	High	Dual Verification (CV+Weight); User review/correction on UI.	Low
Weight Sensor Drift	Medium	Medium	Regular software-based calibration; Noise-filtering algorithm.	Low
Network Failure	Medium	Medium	Local caching of cart data (SQLite); "Offline mode" capability.	Low
Payment Error	Low	High	Use of a reliable, certified UPI gateway; Clear error messages.	Low
AI Bias/Error	Medium	Medium	Continuous dataset expansion and model retraining.	Low
Power Failure	Low	Medium	Onboard battery backup; Surge-protected charging circuit.	Low
Data Privacy Breach	Low	High	End-to-end encryption (TLS); Data minimization principle.	Low
Hardware Injury	Low	Medium	Secure, non-obstructive mounting; No sharp edges.	Low

8.3.1 Testing and Contingency Plans

To validate the safety and functionality of the system, a multi-stage testing plan was executed:

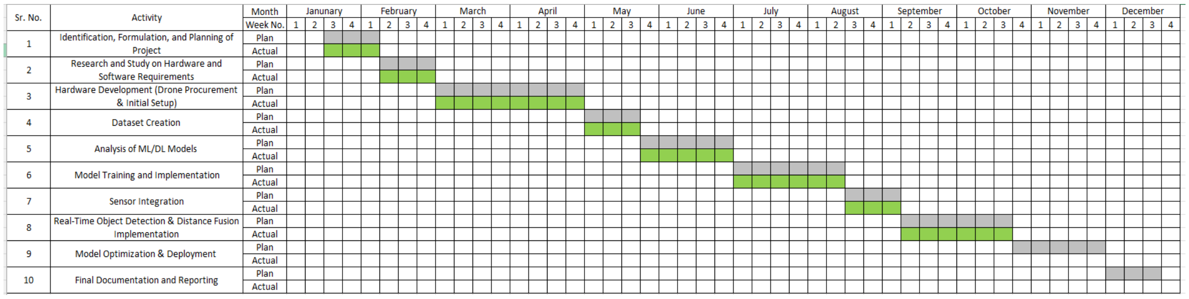
- **Functional Tests:** The detection model was tested under varied lighting conditions. The end-to-end checkout flow was tested repeatedly to ensure UI and database reliability.
- **Safety Tests:** The cart was operated for extended periods to test for overheating. Hardware mounts were physically stressed to ensure stability and durability.
- **Security Tests:** We simulated payment failures to ensure the system handled the exception correctly and did not result in a false positive or negative charge.

A contingency plan was also developed. In the event of a critical system failure, the cart is designed to “fail-safe”. It will log the error, disable the smart functions, and revert to a “regular” shopping cart, allowing the customer to check out at a standard manual till, thus preventing customer frustration and operational deadlock.

CHAPTER 9

PROJECT SCHEDULE AND WORK PLAN

9.1 Project Timeline



Sr. No.	Activity	Month	Janunary				February				March				April				May				June				July				August				September				October				November				December			
		Week No.	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Identification, Formulation, and Planning of Project	Plan			▨	▨	▨																																												
		Actual			🟩	🟩	🟩																																												
2	Research and Study on Hardware and Software Requirements	Plan						▨	▨	▨																																									
		Actual						🟩	🟩	🟩																																									
3	Hardware Development (Drone Procurement & Initial Setup)	Plan									▨	▨	▨	▨	▨	▨	▨	▨																																
		Actual									🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																																
4	Dataset Creation	Plan																	▨	▨	▨																													
		Actual																	🟩	🟩	🟩																													
5	Analysis of ML/DL Models	Plan																				▨	▨	▨	▨	▨																							
		Actual																				🟩	🟩	🟩	🟩	🟩																							
6	Model Training and Implementation	Plan																									▨	▨	▨	▨	▨	▨																	
		Actual																									🟩	🟩	🟩	🟩	🟩	🟩																	
7	Sensor Integration	Plan																															▨	▨	▨														
		Actual																															🟩	🟩	🟩														
8	Real-Time Object Detection & Distance Fusion Implementation	Plan																																		▨	▨	▨	▨	▨	▨								
		Actual																																		🟩	🟩	🟩	🟩	🟩	🟩								
9	Model Optimization & Deployment	Plan																																							▨	▨	▨	▨	▨				
		Actual																																															
10	Final Documentation and Reporting	Plan																																												▨	▨	▨	
		Actual																																															

Figure 9.1 *Group Work Plan*

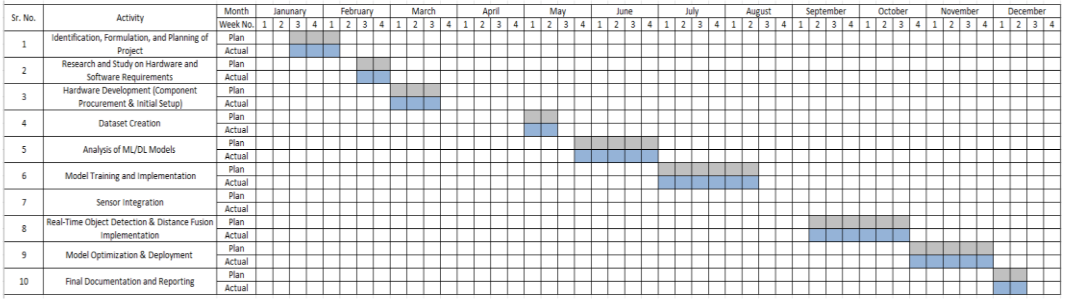
9.2 Individual Work Plan

The QKart team has completed the majority of the core system development during the midterm phase, including model training, hardware assembly, dataset creation, backend structure design, IoT communication setup, and the first version of the UI. The remaining work primarily involves refinement, integration, and final testing, ensuring the project is on track for smooth completion.

9.2.1 Abhimanyu Kumar – AI Model Development & Deployment

Work Completed

- Trained the custom yolov8 model on the QKart product dataset.
- Converted and optimized the model for onyx deployment on the Raspberry Pi 5.
- Benchmarked detection accuracy, mAP, and inference speed on-device.
- Integrated the detection pipeline into the cart’s Python script.
- Minor fine-tuning of the model with a few additional samples.
- Adjusting detection thresholds and optimizing performance under variable lighting.
- Assisting the backend and IoT team with smooth model-to-API data handover.



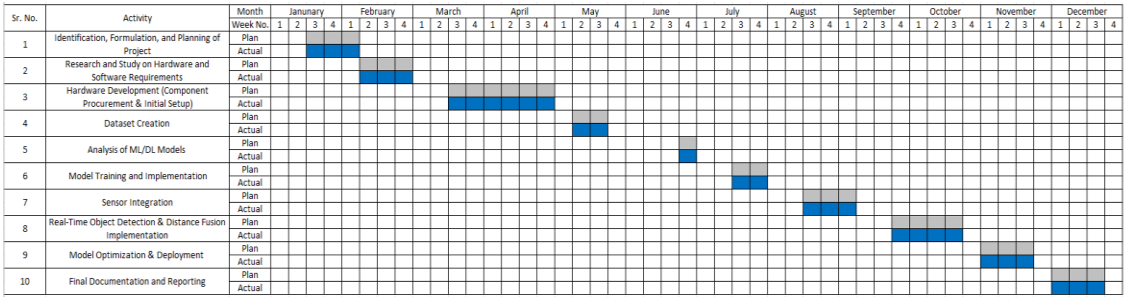
Sr. No.	Activity	Month	Janunary				February				March				April				May				June				July				August				September				October				November				December			
		Week No.	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Identification, Formulation, and Planning of Project	Plan			▨	▨	▨																																											
		Actual			■	■	■																																											
2	Research and Study on Hardware and Software Requirements	Plan							▨	▨																																						
		Actual							■	■																																						
3	Hardware Development (Component Procurement & Initial Setup)	Plan									▨	▨	▨																																			
		Actual									■	■	■																																			
4	Dataset Creation	Plan																	▨	▨																												
		Actual																	■	■																												
5	Analysis of ML/DL Models	Plan																				▨	▨	▨	▨	▨																						
		Actual																				■	■	■	■	■																						
6	Model Training and Implementation	Plan																									▨	▨	▨	▨	▨	▨																
		Actual																									■	■	■	■	■	■																
7	Sensor Integration	Plan																																														
		Actual																																														
8	Real-Time Object Detection & Distance Fusion Implementation	Plan																																	▨	▨	▨	▨	▨	▨								
		Actual																																	■	■	■	■	■	■								
9	Model Optimization & Deployment	Plan																																						▨	▨	▨	▨	▨				
		Actual																																						■	■	■	■	■				
10	Final Documentation and Reporting	Plan																																											▨	▨		
		Actual																																											■	■		

Figure 9.2 Abhimanyu's Individual Work Plan

9.2.2 Tanishka Bhatia – Dataset Development & Hardware Integration

Work Completed

- Captured and annotated the full custom dataset for model training.
- Successfully integrated all hardware components: camera, load cells, HX711, ultrasonic sensor, and Raspberry Pi.
- Implemented and tested the weight-verification mechanism.
- Adding a few more product images to strengthen model robustness.
- Minor refinements to sensor mounting and stability.
- Final hardware calibration before integrated testing.
- Ensured consistent hardware–software communication within the prototype.



Sr. No.	Activity	Month	Janunary				February				March				April				May				June				July				August				September				October				November				December			
		Week No.	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Identification, Formulation, and Planning of Project	Plan			▨	▨	▨																																											
		Actual			■	■	■																																											
2	Research and Study on Hardware and Software Requirements	Plan						▨	▨	▨																																						
		Actual						■	■	■																																						
3	Hardware Development (Component Procurement & Initial Setup)	Plan											▨	▨	▨	▨	▨	▨																														
		Actual											■	■	■	■	■	■																														
4	Dataset Creation	Plan																		▨	▨																											
		Actual																		■	■																											
5	Analysis of ML/DL Models	Plan																							▨																							
		Actual																							■																							
6	Model Training and Implementation	Plan																										▨	▨																			
		Actual																										■	■																			
7	Sensor Integration	Plan																														▨	▨	▨														
		Actual																														■	■	■														
8	Real-Time Object Detection & Distance Fusion Implementation	Plan																																			▨	▨	▨	▨								
		Actual																																			■	■	■	■								
9	Model Optimization & Deployment	Plan																																							▨	▨	▨					
		Actual																																							■	■	■					
10	Final Documentation and Reporting	Plan																																											▨	▨	▨	
		Actual																																											■	■	■	

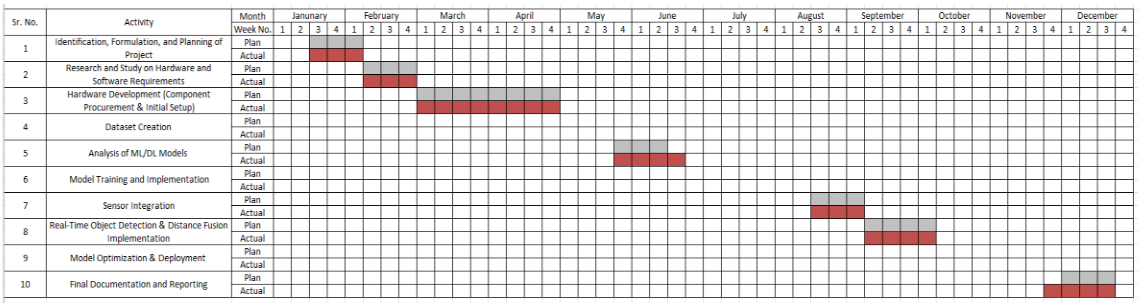
Figure 9.3 Tanishka's Individual Work Plan

9.2.3 Arushi Gupta – Backend Database & System Integration

Work Completed

- Designed and implemented the database structure for Firebase (cloud).
- Defined backend architecture and core REST API endpoints.
- Completed initial database interaction from the cart system.

- Adding payment confirmation logging.
- Finalizing API responses and connecting them to the live UI.



Sr. No.	Activity	Month	Janunary				February				March				April				May				June				July				August				September				October				November				December			
		Week No.	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Identification, Formulation, and Planning of Project	Plan			▨	▨	▨																																											
		Actual			🟥	🟥	🟥																																											
2	Research and Study on Hardware and Software Requirements	Plan						▨	▨	▨																																								
		Actual						🟥	🟥	🟥																																								
3	Hardware Development (Component Procurement & Initial Setup)	Plan									▨	▨	▨	▨	▨	▨	▨	▨																																
		Actual									🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																																
4	Dataset Creation	Plan																																																
		Actual																																																
5	Analysis of ML/DL Models	Plan																				▨	▨	▨																										
		Actual																				🟥	🟥	🟥	🟥																									
6	Model Training and Implementation	Plan																																																
		Actual																																																
7	Sensor Integration	Plan																															▨	▨	▨															
		Actual																															🟥	🟥	🟥															
8	Real-Time Object Detection & Distance Fusion Implementation	Plan																																		▨	▨	▨	▨											
		Actual																																		🟥	🟥	🟥	🟥											
9	Model Optimization & Deployment	Plan																																																
		Actual																																																
10	Final Documentation and Reporting	Plan																																												▨	▨	▨	
		Actual																																											🟥	🟥	🟥	🟥	

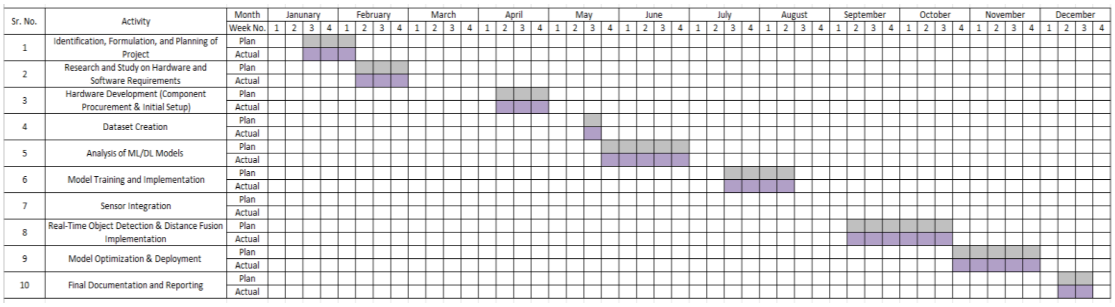
Figure 9.4 *Arushi's Individual Work Plan*

9.2.4 Uttkarsh Singh Pathania – Frontend Web Application & User Interface

Work Completed

- Developed the first version of the web app for real-time billing.
- Designed UI screens for cart summary, payment flow, and user session link.
- Established communication between the Raspberry Pi and frontend.

- Connecting the final API endpoints once backend is finalized.
- Adding minor UI refinements for usability and responsiveness.
- Testing the complete checkout flow end-to-end.



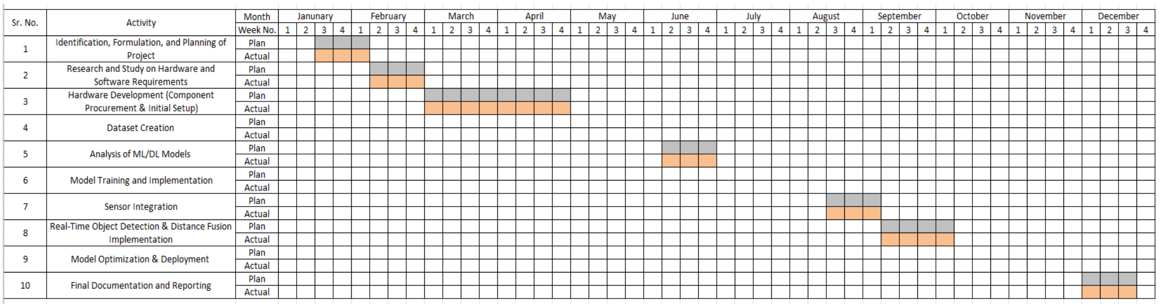
Sr. No.	Activity	Month	Janunary				February				March				April				May				June				July				August				September				October				November				December			
		Week No.	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Identification, Formulation, and Planning of Project	Plan			▨	▨	▨																																											
		Actual			🟪	🟪	🟪																																											
2	Research and Study on Hardware and Software Requirements	Plan						▨	▨	▨																																								
		Actual						🟪	🟪	🟪																																								
3	Hardware Development (Component Procurement & Initial Setup)	Plan														▨	▨	▨																																
		Actual														🟪	🟪	🟪																																
4	Dataset Creation	Plan																			▨																													
		Actual																			🟪																													
5	Analysis of ML/DL Models	Plan																				▨	▨	▨	▨	▨																								
		Actual																				🟪	🟪	🟪	🟪	🟪																								
6	Model Training and Implementation	Plan																											▨	▨	▨	▨																		
		Actual																											🟪	🟪	🟪	🟪																		
7	Sensor Integration	Plan																																																
		Actual																																																
8	Real-Time Object Detection & Distance Fusion Implementation	Plan																																		▨	▨	▨	▨	▨	▨									
		Actual																																		🟪	🟪	🟪	🟪	🟪	🟪									
9	Model Optimization & Deployment	Plan																																								▨	▨	▨	▨	▨				
		Actual																																								🟪	🟪	🟪	🟪	🟪				
10	Final Documentation and Reporting	Plan																																													▨	▨	
		Actual																																													🟪	🟪	

Figure 9.5 Uttkarsh's Individual Work Plan

9.2.5 Khushi Kaushal – IoT Communication & System Testing

Work Completed

- Established stable communication between hardware, backend, and frontend.
- Tested real-time data sync through Firebase.
- Handled early integration testing of CV detection + weight verification + billing.
- Running final IoT stability checks under continuous operation.
- Stress-testing sync reliability across different networks.
- Overseeing final full-system testing and documentation.



Sr. No.	Activity	Month	Janunary				February				March				April				May				June			
		Week No.	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Identification, Formulation, and Planning of Project	Plan			▨	▨	▨																			
		Actual			🟧	🟧	🟧																			
2	Research and Study on Hardware and Software Requirements	Plan						▨	▨	▨																
		Actual						🟧	🟧	🟧																
3	Hardware Development (Component Procurement & Initial Setup)	Plan									▨	▨	▨	▨	▨	▨	▨	▨								
		Actual									🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧								
4	Dataset Creation	Plan																								
		Actual																								
5	Analysis of ML/DL Models	Plan																						▨	▨	▨
		Actual																						🟧	🟧	🟧
6	Model Training and Implementation	Plan																								
		Actual																								
7	Sensor Integration	Plan																								
		Actual																								
8	Real-Time Object Detection & Distance Fusion Implementation	Plan																								
		Actual																								
9	Model Optimization & Deployment	Plan																								
		Actual																								
10	Final Documentation and Reporting	Plan																								
		Actual																								

Sr. No.	Activity	Month	July				August				September				October				November				December			
		Week No.	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Identification, Formulation, and Planning of Project	Plan																								
		Actual																								
2	Research and Study on Hardware and Software Requirements	Plan																								
		Actual																								
3	Hardware Development (Component Procurement & Initial Setup)	Plan																								
		Actual																								
4	Dataset Creation	Plan																								
		Actual																								
5	Analysis of ML/DL Models	Plan																								
		Actual																								
6	Model Training and Implementation	Plan																								
		Actual																								
7	Sensor Integration	Plan							▨	▨	▨															
		Actual							🟧	🟧	🟧															
8	Real-Time Object Detection & Distance Fusion Implementation	Plan										▨	▨	▨	▨											
		Actual										🟧	🟧	🟧	🟧											
9	Model Optimization & Deployment	Plan																								
		Actual																								
10	Final Documentation and Reporting	Plan																					▨	▨	▨	
		Actual																					🟧	🟧	🟧	

Figure 9.6 Khushi's Individual Work Plan

CHAPTER 10

CONCLUSION AND FUTURE WORK

10.1 Conclusion

The primary objective of the QKart project was to design and develop a cost-effective, intelligent shopping cart capable of automating product detection, billing, and inventory synchronization. In this midterm phase, the team successfully integrated computer vision, embedded hardware, and IoT-based communication to establish a functional prototype that streamlines the checkout process.

A lightweight Yolov8 model was trained and optimized for deployment on the Raspberry Pi 5 to enable real-time object detection on edge. Multi-sensor verification using load cells and HX711 modules was incorporated to cross-validate detected items, significantly enhancing billing accuracy and mitigating fraud. Additionally, a hybrid data management system combining SQLite and Firebase was implemented to maintain low-latency local processing with cloud-based synchronization. Early results demonstrate a high mAP of 98%, an average inference time of 94 ms, reliable weight verification, and successful real-time bill updates through a web-based interface.

The midterm progress clearly shows that the project has met its proposed objectives so far. A functional hardware prototype, a trained and optimized AI model, a modular system architecture, and a seamless detection-to-billing workflow have been achieved. The work completed validates the feasibility of creating an affordable, scalable, and high-performance smart cart system using open-source tools and cost-efficient components. The learnings from this phase including embedded AI deployment, hardware-software integration, and full-stack development lay a strong foundation for completing the remaining stages of the project.

10.2 Future Work

Although significant progress has been achieved, the current prototype has several limitations that open avenues for future development and optimization. The present system is trained on a limited set of products and tested in semi-controlled lighting and environmental conditions. Scaling QKart to a commercial retail environment will require expanding the dataset, improving robustness against occlusion, varying angles, product overlaps, and diverse store conditions.

To enhance system performance, future work will focus on:

- Model Enhancement & Dataset Expansion: Increasing the diversity and number of product images to improve detection accuracy across packaging variants, lighting conditions, and cluttered scenes. Testing alternative lightweight models (e.g., YOLOv8, MobileNet-SSD) could further optimize speed and accuracy on the Raspberry Pi.
- Improved Sensor Fusion: Refining the weight-verification algorithm to support multiple items added simultaneously, dynamic stacking, and items with variable weights. Incorporating additional sensors—such as infrared or depth cameras—can further reduce false positives and support occlusion handling.
- Full Backend & UI Integration: Completing the development of the Flask/Django backend, enhancing the user interface for smoother cart management, and integrating features such as cart-sharing, discount application, and multi-user login support.
- Advanced Inventory & Analytics: Extending Firebase synchronization to include predictive analytics, low-stock alerts, purchase trend mapping, and real-time dashboard views for store management.
- Power & Mobility Optimization: Replacing the prototype wooden frame with an actual cart mount, optimizing power consumption, and evaluating long-duration battery solutions suitable for commercial environments.
- Commercial Deployment Features: Adding store-level scaling capabilities such as cart-to-cart communication, self-checkout aisles, product recommendation systems, geolocation inside stores, and integration with loyalty programs.

In the long term, QKart has strong potential for real-world deployment as a low-cost alternative to RFID-based systems. Its scalability, modular architecture, and affordability make it suitable for small and mid-sized retailers, helping them modernize operations without significant financial investment. With continued development and testing, QKart can evolve into a fully deployable smart retail automation system that improves efficiency, reduces labor dependency, and significantly enhances the customer shopping experience.

REFERENCES

[1] M. K. Islam and M. S. Kaiser, “Design of a Low-Cost Weight Measurement System Using HX711 Load Cell Amplifier,” *Proc. IEEE Int. Conf. Smart IoT*, pp. 155–160, 2021.

[2] N. Ahmad et al., “Computer Vision-Based Automation in Retail: A Survey,” *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 4, pp. 1980–1996, 2021.

[3] C. C. Aggarwal, *Machine Learning for IoT Applications*, Springer, 2022.

[4] A. D. Joseph and K. R. Varghese, “Smart Shopping Cart Using RFID and Image Processing,” *Proc. IEEE Int. Conf. Inventive Comput.*, pp. 390–395, 2020.

[5] M. Khan and L. Zhang, “Vision-Based Smart Shopping Using YOLOv8 and Deep Learning,” *IEEE Access*, vol. 11, pp. 14567–14578, 2023.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 779–788, 2016.

[7] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection,” *arXiv preprint arXiv:2004.10934*, 2020.

[8] G. Jocher et al., “YOLOv5: An Improved Object Detection Architecture,” *GitHub Repository*, Ultralytics, 2023.

[9] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors,” *arXiv:2207.02696*, 2022.

[10] C.-Y. Wang et al., “YOLOv8: Next-Generation Real-Time Object Detection,” *Ultralytics Technical Report*, 2023.

[11] R. Utami and A. Ma’arif, “Real-Time Product Detection Using YOLOv4 for E-Commerce Applications,” *J. Phys.: Conf. Ser.*, vol. 1567, no. 4, pp. 1–7, 2020.

[12] B. Schmude and H. Hopf, “A Performance Comparison of YOLO Models on CPU-Based Embedded Systems,” *IEEE Embedded Systems Letters*, vol. 14, no. 2, pp. 89–93, 2022.

[13] M. G. Devereux, “Edge AI using Raspberry Pi,” *IEEE Potentials*, vol. 41, no. 5, pp. 20–28, 2022.

[14] Raspberry Pi Foundation, “Raspberry Pi 5 Technical Specifications,” *Raspberry Pi Documentation*, 2023.

[15] S. N. Singh et al., “Sensor Fusion Techniques for Smart Retail Applications,” *IEEE Sensors J.*, vol. 22, no. 18, pp. 17560–17568, 2022.

[16] Y. Xie and Q. Wang, “Lightweight Convolutional Models for Embedded Vision Applications,” *IEEE Trans. Embed. Syst.*, vol. 20, no. 3, pp. 1345–1356, 2021.

[17] F. Zhang et al., “A Comparative Study of RFID vs Computer Vision Checkout Systems,” *IEEE Internet of Things J.*, vol. 8, no. 9, pp. 7120–7132, 2021.

[18] S. Garud et al., “IoT-Based Smart Cart System Using Camera and MCU,” *Proc. IEEE Int. Conf. IoT-SIU*, pp. 1–6, 2021.

[19] Google, “TensorFlow Lite: Machine Learning on Edge Devices,” *TensorFlow Developer Guide*, 2023.

[20] A. S. Abdullah and R. Kumar, “Real-Time Billing Using Flask-Based IoT Backend Systems,” *IEEE Access*, vol. 10, pp. 98532–98541, 2022.

[21] Firebase Documentation, “Firebase Realtime Database,” Google, 2023.

[22] S. R. Madakam, R. Ramaswamy, and S. Tripathi, “Internet of Things (IoT): A Literature Review,” *J. Comput. Commun.*, vol. 3, no. 5, pp. 164–173, 2015.

Appendix A – Raw Data

This appendix presents representative samples of the raw data collected and generated during the development of the QKart system. Due to the large size of the full dataset and training logs, only essential excerpts, summaries, and sample entries are included here for documentation. The complete dataset and full logs are maintained in the project repository.

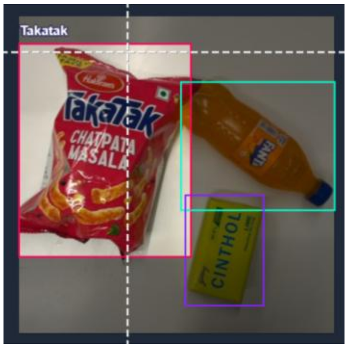
A.1 Dataset Overview for Object Detection

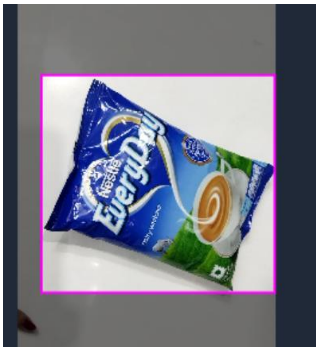
A custom dataset of over 400 images was created for training the YOLOv8 model. Images were captured under indoor lighting conditions and annotated for object detection.

A.1.1 Dataset Summary

Product Class	No. of Images	Training Split	Validation Split	Testing Split	Resolution
Cinthol	73	53	13	7	640x640
Milk Powder	117	77	28	12	640x640
Popcorn	41	26	6	7	640x640
Takatak	80	60	12	8	640x640
Pistachios	99	75	17	7	640x640

A.1.2 Sample Annotated Images





A.2 Model Training Logs

A.2.1 Training Summary

The model achieved strong and stable convergence with excellent object detection performance across all six product classes.

Below is the summarized output from the final epoch (**Epoch 100/100**):

- ➔ **Box Loss:** 0.2726
- ➔ **Class Loss:** 0.2534
- ➔ **DFL Loss:** 0.8677
- ➔ **Inference Size:** 640 × 640
- ➔ **Total Training Time:** 0.261 hours
- ➔ **Model Parameters:** 3,006,818

A.2.2 Global Performance Metrics

After training, the model was validated using the best checkpoint (best.pt).
The validation results are:

METRIC	SCORE
Precision	0.965
Recall	0.972
mAP@50	0.969
mAP@50-95	0.901

A.2.4 F1-Confidence, Precision, Recall and mAP Curves

F1–Confidence Curve

- All classes maintained F1-scores above 0.85 across varying confidence thresholds.
- Optimal F1-score observed around 0.55–0.70 confidence.

- Indicates robust and confidence-consistent performance.

Precision Curve

- Precision rapidly stabilized beyond 0.90, showing the model rarely misidentifies items.
- Minimal fluctuation after epoch ~20.

Recall Curve

- Recall reached stable values of 0.94–0.97.
- High recall ensures minimal false negatives — critical for automated checkout.

mAP Curves

- mAP50 converged near 0.96–0.97 across epochs.
- mAP50-95 consistently reached 0.90+, reflecting strong localization accuracy.

A.2.6 Model Deployment Readiness

The final model meets all performance criteria required for the QKart smart cart:

- ❖ High detection accuracy
- ❖ Fast inference suitable for Raspberry Pi 5
- ❖ Reliable across lighting variations and product orientations
- ❖ Stable per-class predictions
- ❖ Exported in ONNX format for optimized edge deployment

Final ONNX Inference Speed:

- 3.1 ms per image (PyTorch GPU equivalent)
- Expected Pi 5 inference: **70–110 ms per frame** (suitable for real-time detection)