

Project Documentation: AI-Powered Smart Billing System (Revision 1.1)

1. Executive Summary

1.1. Project Title

An AI-Powered Smart Billing System using Real-time Object Detection for Automated Retail Checkout

1.2. Project Goal

To revolutionize the Point of Sale (POS) process by developing a functional, low-latency computer vision system capable of identifying retail products from a live camera feed, aggregating their quantities, and automatically calculating the total bill.

1.3. Abstract

This project focuses on the development of an intelligent **Object Detection Billing System** designed for high-efficiency retail and inventory environments. It utilizes a state-of-the-art, real-time object detection model trained on specific product SKUs. The system ingests a live video stream, localizes and classifies products, interfaces with a relational product database (for pricing and metadata), and dynamically generates an itemized bill. The final deliverable is a self-contained prototype that demonstrates accelerated, error-reduced, and fully automated checkout functionality.

2. Scope and Objectives

2.1. Scope of Work (Inclusions)

- **Real-Time Detection:** Processing and annotation of a video stream at a target of ≥ 15 Frames Per Second (FPS).
- **Product Catalog:** Initial training and integration for a defined catalog of 10 to 15 unique product classes (SKUs).
- **Billing Engine:** Logic for calculating total cost, handling multiple instances of the same item, and displaying line-item summaries.
- **User Interface (UI):** A simple GUI/Web interface displaying the live video feed, bounding boxes, product names, and the running total bill.

2.2. Objectives and Success Metrics

Objective ID	Description	Success Metric (KPI)	Target
O1	Achieve high detection accuracy on the product dataset.	Mean Average Precision (mAP)	$\geq 85\%$
O2	Ensure acceptable operational speed for real-time use.	Inference Speed	$\geq 15\text{fps}$
O3	Validate the automated pricing calculation process.	Billing Accuracy	100%

3. Technical Design

3.1. Core Technology Stack

Layer	Technology	Rationale
Programming	Python 3.x	Industry standard for machine learning and computer vision.
ML Framework	YOLOv8 (or equivalent single-shot detector)	Chosen for its superior balance of speed (FPS) and accuracy (mAP), crucial for real-time performance.
Vision Library	OpenCV	Handling video stream input, frame processing, and graphical overlays.
Database	SQLite (Prototype)	Simple, lightweight embedded database for fast product data retrieval.
UI	Streamlit or Flask	Quick deployment of an interactive, real-time dashboard.

3.3. Database Schema (Product Catalog)

The database will store core product metadata and pricing:

Field Name	Data Type	Description	Constraint
product_id	Text/Integer	Unique SKU identifier.	Primary Key
class_name	Text	The model's predicted class label	Indexed
unit_price	Decimal (2 places)	Price of a single unit.	Not Null
description	Text	Full product name.	Optional

3.4. Billing and Tracking Logic

A critical component is the **Object Tracker**. Since the detection model provides bounding boxes for every frame, a robust tracking algorithm (Simple Online and Realtime Tracking - SORT) will be necessary to:

1. **Assign IDs:** Give a persistent ID to each detected object instance.
 2. **Count Uniques:** Prevent double-counting the same item across consecutive frames.
 3. **Removal Logic:** Detect when an object (item) is removed from the checkout area, decrementing the bill accordingly.
-

4. Implementation Phases

Phase	Tasks	Deliverables
P1: Setup & Data Prep	Define SKUs, collect raw images, manual bounding box annotation, set up cloud/local environment.	Annotated dataset (in YOLO format), Initial Database.
P2: Model Training	Configure YOLOv8, train the model, fine-tune hyperparameters, validate performance against O1 and O2.	Trained model weights , Performance Report.
P3: Integration	Implement Database lookup function, develop object tracking logic, build the Billing Engine (addition/subtraction).	Functional Billing Engine
P4: Prototype Finalization	Develop the UI, merge all components, comprehensive system testing (simulated purchases), documentation finalization.	Functional End-to-End Prototype, Final Project Report.