**AUTOBILL INSTANT CHECKOUT SYSTEM**

**A Project Report**

Submitted in partial fulfilment of the requirements for the award of the Degree of

**BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)**

**By**

# Spruha Sharad Pawaskar

Seat Number: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Under the esteemed guidance of**

# Mr. Laxmikant Manchekar

**Assistant Professor, Department of Information Technology**

A black sign with red and blue text

AI-generated content may be incorrect.

## DEPARTMENT OF INFORMATION TECHNOLOGY

**VIDYALANKAR SCHOOL OF INFORMATION TECHNOLOGY**

**(Affiliated to University of Mumbai)**

**MUMBAI, 400 037**

**MAHARASHTRA**

**2024 - 2025**

## VIDYALANKAR SCHOOL OF INFORMATION TECHNOLOGY

**(Affiliated to University of Mumbai)**

**MUMBAI-MAHARASHTRA-400037**

## DEPARTMENT OF INFORMATION TECHNOLOGY

A black sign with red and blue text

AI-generated content may be incorrect.

**CERTIFICATE**

This is to certify that the project entitled, **"Autobill Instant Checkout System"**, is bonafied work of

**SPRUHA SHARAD PAWASKAR** bearing Seat No: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ submitted in partial

fulfilment of the requirements for the award of degree of BACHELOR OF SCIENCE in

INFORMATION TECHNOLOGY from University of Mumbai.

**Internal Guide**  **Coordinator**

**Internal Examiner** **External Examiner**

**Date:**       **College Seal**     **Principal**

**CERTIFICATES**

****

# ABSTRACT

The Auto Bill Instant Checkout System is a cutting-edge, IoT-based electronic billing solution designed to streamline the checkout process for businesses and consumers.

Key Features which are mainly used:

* Real-time Billing and Payment Processing
* Secure Payment Gateway Integration
* Instant Transaction Confirmation

This innovative system seamlessly integrates with existing payment infrastructure, enabling fast, secure, and accurate transactions.

Auto Bill provides a faster checkout shopping experience to minimize human interactions in the store to keep shoppers and employees safer during the pandemic. Auto Bill uses computer vision and machine learning to visually detect and instantly identify the items placed and the weight sensor measure the weights of the things placed on the top of the counter.

# ACKNOWLEDGEMENT

We would like to express our special thanks and gratitude to our project guide **Mr. Laxmikant Manchekar** for guiding us to do the project work on time and giving us all support and guidance, which made us complete our project duly. We are extremely thankful to her for providing such nice support and guidance.

We are also thankful for and fortunate enough to get constant encouragement, support and guidance from the teachers, the Vice Principal, the College Administrative Officer and the Principal of Vidyalankar School of Information Technology who helped us in successfully completing our project work.

# DECLARATION

I hereby declare that the project entitled, “**AUTOBILL INSTANT CHECKOUT SYSTEM**” done at Vidyalankar School of Information Technology, has not been in any case duplicated to submit to any other universities for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfilment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum

**Spruha S Pawaskar**

Name and Signature of the Student

## Table of Contents

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.**  **No** |  |  | **Content** | **Pg.**  **No** |
| **1** | 1.1 |  | BACKGROUND | 11 |
|  | 1.2 |  | OBJECTIVES | 11 |
|  | 1.3 |  | PURPOSE, SCOPE AND APPLICABILITY | 12 |
|  |  | 1.3.1 | PURPOSE |  |
|  |  | 1.3.2 | SCOPE |  |
|  |  | 1.3.3 | APPLICABILITY |  |
|  |  |  | TECHNICAL FEASIBILITY STUDY |  |
| **2** |  |  | SURVEY OF TECHNOLOGIES | 13 |
|  | 2.1 |  | WHY IS PYTHON BETTER |  |
|  | 2.2 |  | TECHNOLOGIES USED | 14 |
| **3** |  |  | REQUIREMENTS AND ANALYSIS | 16 |
|  | 3.1 |  | PROBLEM DEFINITION |  |
|  | 3.2 |  | REQUIREMENS SPECIFICATION |  |
|  | 3.3 |  | PLANNING AND SCHEDULING |  |
|  | 3.4 |  | SOFTWARE AND HARDWARE REQUIREMENTS | 19 |
| **4** |  |  | SYSTEM DESGIN | 20 |
|  | 4.1 |  | BASIC MODULES |  |
|  | 4.2 |  | OBJECT DETECTION | 21 |
|  |  | 4.2.1 | SCHEMA DESGIN |  |
|  |  | 4.2.2 | DATA INTEGRITY AND CONSTRAINTS |  |
|  | 4.3 |  | DIAGRAMS |  |
|  |  | 4.3.1 | E-R DIAGRAM/BLOCK DIAGRAM | 27 |
|  |  | 4.3.2 | CLASS DIAGRAM/DATAFLOW DIAGRAM | 28 |
|  |  | 4.3.3 | USE CASE DIAGRAM | 29 |
|  |  | 4.3.4 | SEQUENCE DIAGRAM | 30 |
|  |  | 4.3.5 | ACTIVITY DIAGRAM | 31 |
|  |  | 4.3.6 | USER INTERFACE DESIGN | 32 |
| **5** | 5.1 |  | IMPLEMENTATION AND TESTING | 33 |
|  | 5.2 |  | CODING DETAILS & CODE EFFICIENCY | 33 |
|  |  | 5.2.1 | CODE EFFFICIENCY |  |
|  |  | 5.2.2 | CODE SCREENSHOT |  |
|  |  | 5.2.3 | EXPLANATION OF CODE |  |
|  |  | 5.2.4 | PERFORMANCE CONSIDERATIONS |  |
| **6** | 6.1 |  | RESULTS AND DISCUSSIONS | 38 |
|  |  | 6.6.1 | TEST REPORTS |  |
|  | 6.2 |  | SYSTEM FUNCTIONING / USER DOCUMENTATION | 38 |
|  |  | 6.2.1 | SYSTEM COMPONENTS |  |
|  |  | 6.2.2 | SYSTEM FEATURES |  |
|  |  | 6.2.3 | HOW THE SYSTEM WORKS |  |
|  |  | 6.2.4 | USER INSTRUCTIONS |  |
| **7** | 7.1 |  | CONCLUSION AND FUTURE SCOPE | 42 |
|  |  | 7.1.1 | CONCLUSION |  |
|  |  | 7.1.2 | FUTURE SCOPE OF THE PROJECT |  |
| **8** |  |  | SECURITY ISSUES | 43 |
| **9** |  |  | REFERENCES | 44 |
| **10** |  |  | SUMMARY | 45 |
| **11** |  |  | PLAGIARISM REPORT | 46 |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Sr.**  **No** | **Contents** | **Pg.** **No** |
| **1** | Python | 13 |
| **2** | HTML | 14 |
| **3** | JavaScript | 14 |
| **4** | NodeJS | 15 |
| **5** | Arduino | 15 |
| **6** | Edge Impulse | 15 |
| **7** | Gantt Chart | 18 |
| **8** | Waterfall Model | 18 |
| **9** | Planning & Scheduling | 18 |
| **10** | E-R Diagram | 27 |
| **11** | Class Diagram | 28 |
| **12** | Use Case Diagram | 29 |
| **13** | Sequence Diagram | 30 |
| **14** | Activity Diagram | 31 |
| **15** | User Interface Design | 32 |

**Chapter 1: Introduction**

Auto Bill is an AI-powered autonomous checkout system for retail stores, that combines the power of computer vision and machine learning to provide an amazing shopping experience. Auto Bill provides a faster checkout shopping experience to minimize human interactions in the store to keep shoppers and employees safer during the pandemic.

Auto Bill uses computer vision and machine learning to visually detect and instantly identify the items placed and the weight sensor measure the weights of the things placed on the countertop. Once the items are identified, things are automatically added to the cart and the bill is generated instantaneously. QR code for payment is generated and users can pay the bill by scanning the QR code



### 1.1 Background

The Auto Bill Instant Checkout System was developed to address the growing need for efficient, secure, and accurate transaction processing in various industries. Auto Bill Instant Checkout System leverages cutting-edge technologies, including Internet of Things (IoT) devices, cloud-based infrastructure, advanced payment gateway integration, and mobile and web applications.

By automating billing and payment processing, the system provides real-time transaction tracking and analytics, secure payment processing, and support for multiple payment methods. The system's technical requirements include IoT devices, cloud-based infrastructure, payment gateway integration, mobile and web applications, and accounting and inventory management system integration. By leveraging these technologies, the Auto Bill Instant Checkout System provides a robust and reliable solution for businesses seeking to streamline their checkout processes and enhance customer satisfaction.

### 1.2 Objectives

The primary objective of the Auto Bill Instant Checkout System is to provide a fast, secure, and accurate transaction processing solution for businesses and customers. The system aims to streamline the checkout process, reducing wait times and labour costs while increasing productivity and customer satisfaction. The objectives of the Auto Bill Instant Checkout System are- To automate billing and payment processing, eliminating manual entry errors and discrepancies.

To provide real-time transaction tracking and analytics, enabling businesses to make informed decisions. It Improve customer experience through faster and more convenient checkout processes. It also Increase business efficiency and productivity through automated transaction processing.

### 1.3 Purpose, Scope, Applicability (Feasibility Study)

#### 1.3.1 Purpose

The Auto Bill Instant Checkout System aims to provide a fast, secure, and automated transaction processing solution for businesses. Its purpose is to- Facilitate efficient checkout processes, Automate billing and invoicing, Provide real-time transaction tracking and analytics, Integrate with existing accounting and inventory systems, Enhance customer satisfaction and operational efficiency

#### 1.3.2 Scope

The scope of this documentation encompasses the design, implementation, configuration, and operation of the Auto Bill Instant Checkout System. This documentation covers the following aspects-

* **System Overview:** Architecture, components, and functionality, Installation.
* **Configuration:** Hardware and software requirements, setup, and deployment,
* **UserInterface:** Navigation, features, and functionality for merchants.
* **TransactionProcessing:** Payment gateway integration, processing.
* **Security and Compliance:** Data protection, access control, industry regulations.

**1.3.3 Applicability**

#### Feasibility Study

* Compatible
* Secure , Scalable , Flexible , Versatile
* Efficient , Streamlined Checkout Processes

# Chapter 2 Survey of Technologies

## 2.1 Why is Python Better

### 

### *Figure 1 Django*

### Why we used Python?

* Rapid development capabilities
* Easy integration with various payment gateways and APIs
* Robust security features and libraries
* Cross-platform compatibility

## 2.2 Technologies Used

### HTML Code

HTML (Hypertext Markup Language) was used to design the user interface and layout of the Auto Bill Instant Checkout System's web-based application. The Auto Bill Instant Checkout System provides a seamless, user-friendly experience for merchants and customers, while ensuring a solid foundation for further development and customization.

#### *Figure 2 HTML Code*

**Pros and Cons:**

* HTML works on multiple devices and browsers.
* HTML code is simple to update and modify.
* HTML structure improves search engine rankings.
* HTML alone cannot handle complex logic.

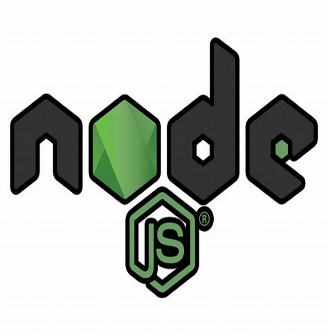
### JAVA SCRIPT



JavaScript was used to enhance the Auto Bill Instant Checkout System's functionality, usability, and security. By incorporating JavaScript, the Auto Bill Instant Checkout System delivers a seamless, secure, and interactive experience for merchants and customers, ensuring efficient and reliable payment processing. JavaScript Libraries and Frameworks are jQuery for DOM manipulation and event handling that React or Angular for building reusable UI components

***Figure 3 JavaScript***

### NODE JS

Node.js enables the Auto Bill Instant Checkout System to efficiently handle high volumes of transactions, ensuring a seamless and secure experience for merchants and customers. Node.js is used for Building the checkout system's backend API and Handling payment gateway integrations. By leveraging Node.js, the Auto Bill Instant Checkout System

delivers a Real-time analytics and reporting

***Figure 4 NodeJS***

### ARDUINO

**A blue logo with white text

Description automatically generated**

Arduino is an open-source platform that facilitates the creation of electronic projects. It consists of both hardware and software components. The hardware is a programmable circuit board, often referred to as a microcontroller, while the software is an Integrated Development Environment (IDE) used to write and upload code to the microcontroller

***Figure 5 Arduino***

### A colorful stripes on a black background Description automatically generatedEDGE IMPULSE

Edge Impulse is the leading edge AI platform for collecting data, training models, and deploying them to your edge computing devices. It provides an end-to-end framework that easily plugs into your edge MLOps workflow.

***Figure 6 Edge Impulse***

## Chapter 3 Requirements and Analysis

### 3.1 Problem Definition

The checkout process for online transactions is often cumbersome, time-consuming, and prone to errors, resulting in abandoned carts, dissatisfied customers, and significant revenue losses for merchants. Existing checkout systems frequently require multiple steps, lengthy forms, and inadequate payment options, leading to friction and frustration for customers.

Problem statement outlines the key challenges and objectives that the Auto Bill Instant Checkout System aims to address, providing a clear direction for the project's development and implementation. The system will integrate with various payment gateways, accounting and inventory systems, and provide real-time analytics and reporting.

### 3.2 Requirement Specification

Meeting these requirements, the Auto Bill Instant Checkout System will provide a fast, secure, and seamless checkout experience for customers, while streamlining payment processing and reducing errors for merchants.

Main roles of System are :-

* Integration with multiple payment gateways (e.g., Gpay, Paytm)
* Mobile optimization for seamless checkout experience

### 3.3 Planning and Scheduling

The following table gives the project plan for Phase 1 & Phase 2 of our project:

|  |  |  |  |
| --- | --- | --- | --- |
| **ACTIVITY** | **DESCRIPTION** | **EFFORT IN PERSON WEEKS** | **DELIVERABLE** |
| P1 - 01 | Requirement Analysis | 2 Weeks | Requirement Gathering |
| P1 – 02 | Existing System Study | 4 Weeks | Existing System Study |
| P1 – 03 | Technology Selection | 3 Weeks | Arduino |
| P1 - 04 | Modular Specifications | 2 Weeks | Module Description |
| P1 - 04 | Design & Modelling | 4 Weeks | Analysis Report |
|  | **TOTAL** | **15 WEEKS** |  |

*Table 1 – Phase 1*

|  |  |  |  |
| --- | --- | --- | --- |
| **ACTIVITY** | **DESCRIPTION** | **EFFORT IN PERSON WEEKS** | **DELIVERABLE** |
| P2 – 01 | Detailed Design | 3 Weeks | Document |
| P2 – 02 | User Interface Design | Included Above | UI Document |
| P2 – 03 | Coding & Implementation | 14 Weeks | Code Release |
| P2 – 04 | Testing & Bug Fixing | 2 Weeks | Test Report |
| P2 – 05 | Performance Evaluation | 4 Weeks | Analysis Report |
| P2 - 06 | Release | Included Above | System Releases |
|  | **TOTAL** | **23 WEEKS** | **Deployment Efforts are Extra** |

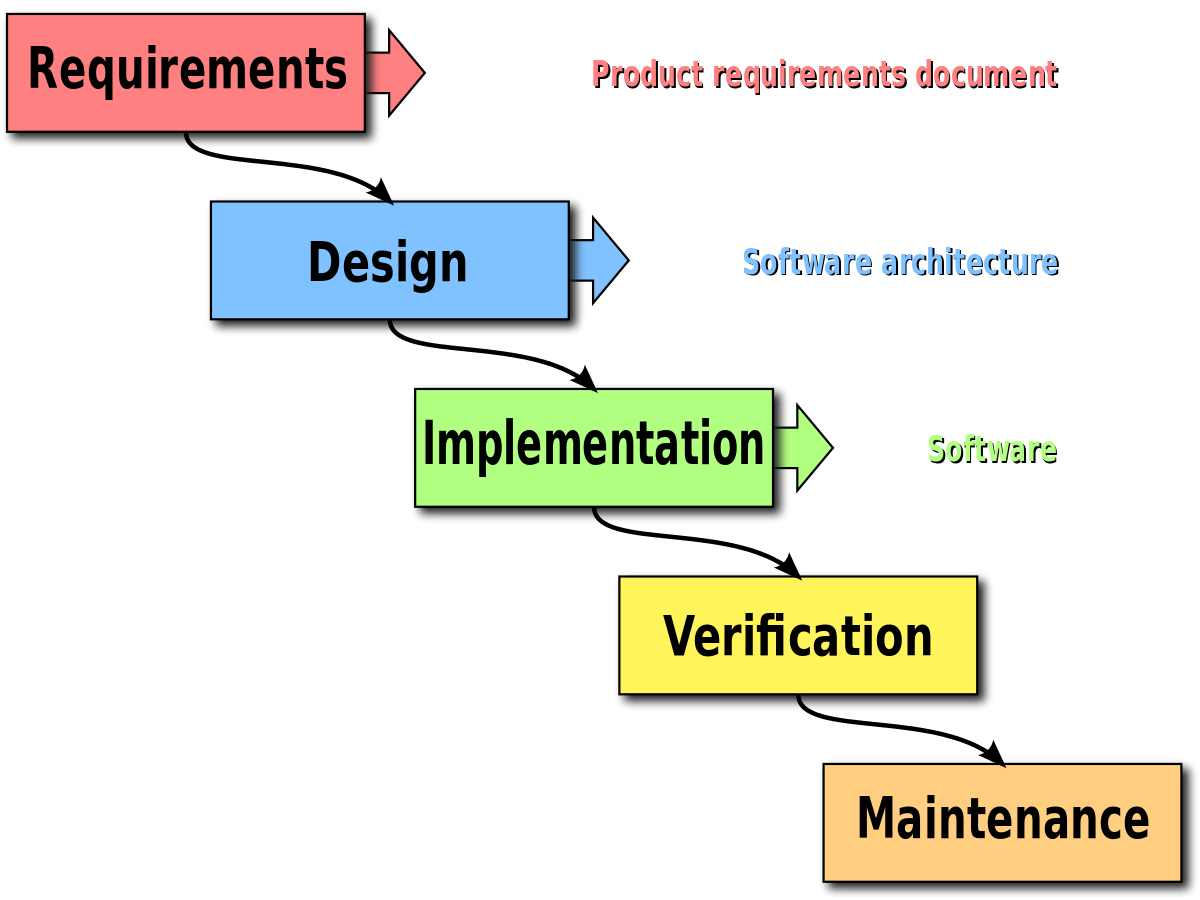
*Table 2 – Phase 2*

#### 3.3.1 Gantt Chart A chart with green squares AI-generated content may be incorrect.

***Figure 7 Gantt Chart***

#### 3.3.2 Waterfall Model

In this project we will be using Waterfall model because it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.



***Figure 8 Waterfall Model***

#### 3.4 Software and Hardware Requirement

**3.4.1 Platform**

The Auto Bill Instant Checkout System can be hosted using ESP32CAM.

##### 3.4.2 Languages Used

###### FRONT-END FOR SCRIPTING

* HTML - Page layout will be designed in HTML.
* JavaScript – All validation task and animation will be developed by JavaScript.

###### BACK-END

* C++ - For Programming the ESP32-CAM Module.
* NodeJS – For creating Web Server to interact with the ESP32-CAM.

## Chapter 4 System Design

#### 4.1 Basic Modules

The Basic Module serves as the foundation of the Auto Bill Instant Checkout System. It includes core functionalities such as- User registration and login, Payment method configuration, Transaction history and records, System configuration and settings.

##### The Admin Module

The Admin Module is designed for system administrators and merchants to manage and monitor the checkout process. Key features include- Dashboard analytics and reporting, Merchant management (add, edit, delete), Product and service management, Transaction monitoring and reconciliation, System security and access control, Integration with accounting and inventory systems.

##### The Customer Module

The Customer Module focuses on providing a seamless checkout experience for customers. Features include- User profile , management, Order history and tracking, Payment method management, Real-time transaction updates, Personalized offers and discounts, Mobile optimization for easy checkout.

##### The Security Module

The Security Module ensures the integrity and confidentiality of sensitive data. Features include- Encryption and tokenization, Access control and authentication, Regular security audits and testing, Compliance with industry standards, Incident response and management. This module safeguards customer and merchant data, protecting against potential security threats

#### 4.2 Data Design

The Auto Bill Instant Checkout System requires a robust data design to ensure seamless transaction processing, secure data storage, and efficient data analysis. The system's data design encompasses various components, including database schema, data models, data warehousing, and data security.

**4.2.1 Database Schema**

The database schema for the Auto Bill Instant Checkout System consists of

the following entities:

• **Customer**: customer\_id (PK), name, email, phone, address

• **Merchant**: merchant\_id (PK), name, email, phone, address

• **Transaction**: transaction\_id (PK), customer\_id (FK), merchant\_id (FK), date, amount, payment\_method

• **Payment Method**: payment\_method\_id (PK), payment\_type (e.g., credit/debit card, bank transfer)

• **Product/Service**: product\_id (PK), name, description, price

##### 4.2.2 Object Detection

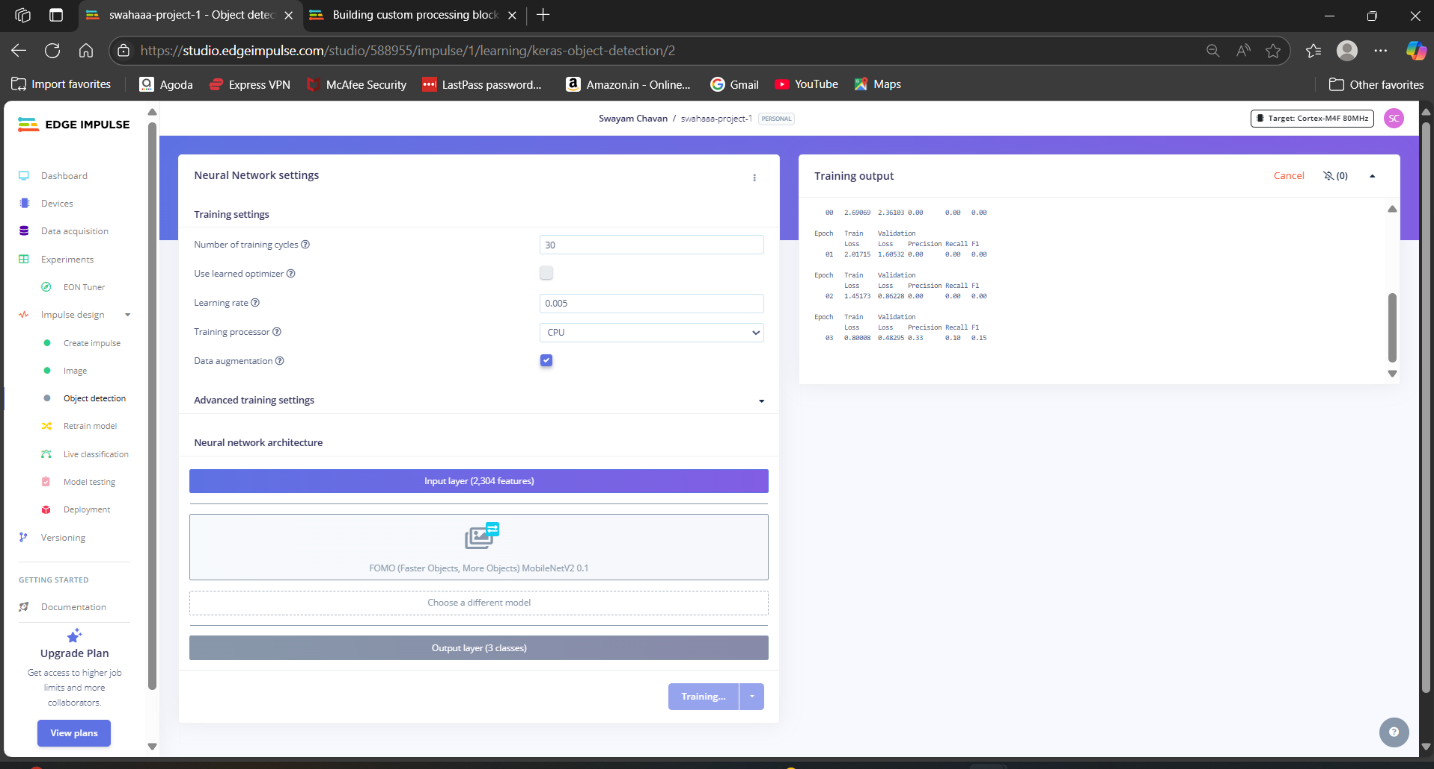
A logo with text overlay

Description automatically generated

Edge Impulse is one of the leading development platforms for machine learning on edge devices, free for developers and trusted by enterprises. Here we are using machine learning to build a system that can recognize the products available in the shops. Then we deploy the system on the Raspberry Pi 3B.

**Data Acquisition**

To make the machine learning model it's important to have a lot of images of the products. When training the model, these product images are used to let the model distinguish between them. Make sure you have a wide variety of angles and zoom levels of the products which are available in the shops. For the data acquisition, you can capture data from any device or development board or upload your existing datasets. So here we are uploading our existing datasets.

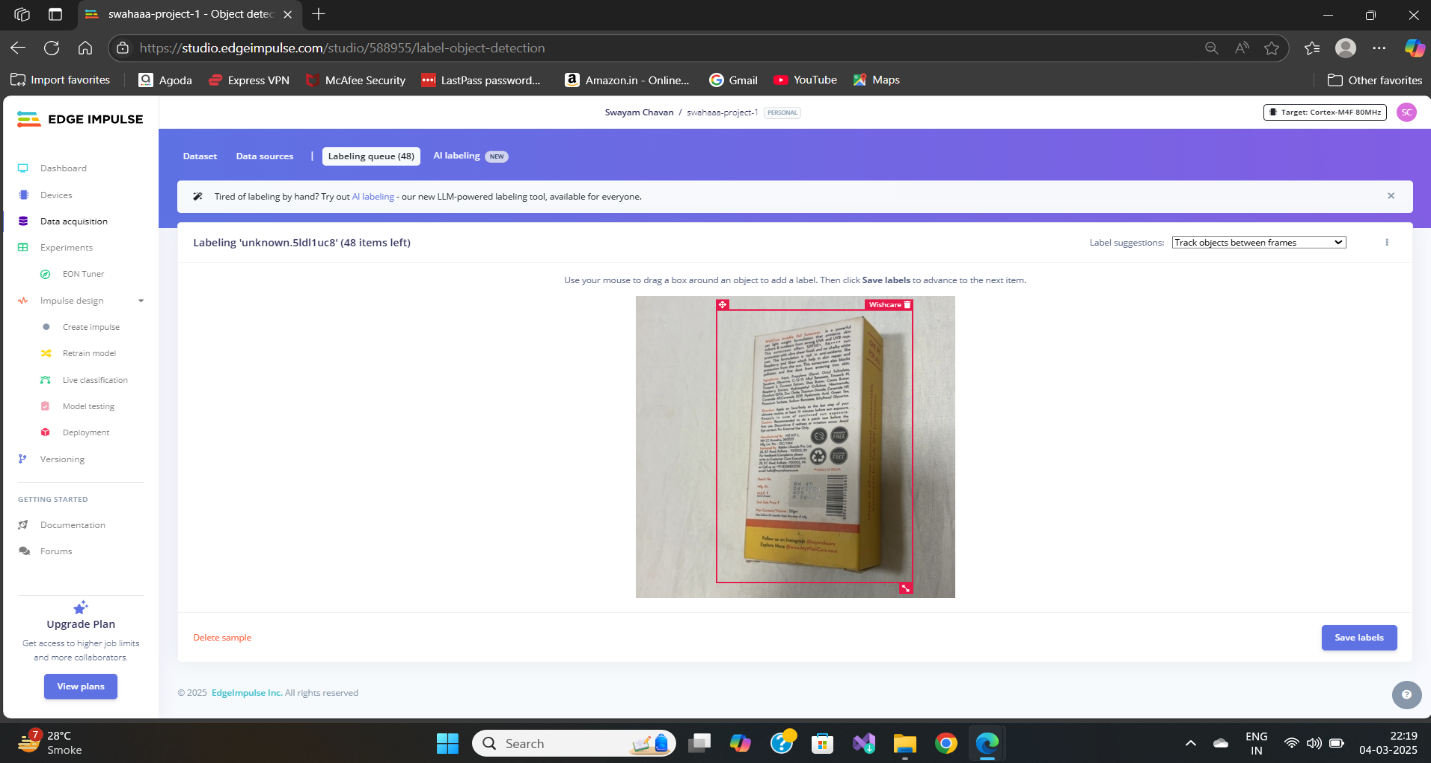


For uploading, just moves on to the *Data acquisition tab* and just choose a file. Then label it and upload it to the training section. The Edge Impulse will only accept either JPG or PNG image files. If you have any other format, just convert it to JPG or PNG format with the online converters.

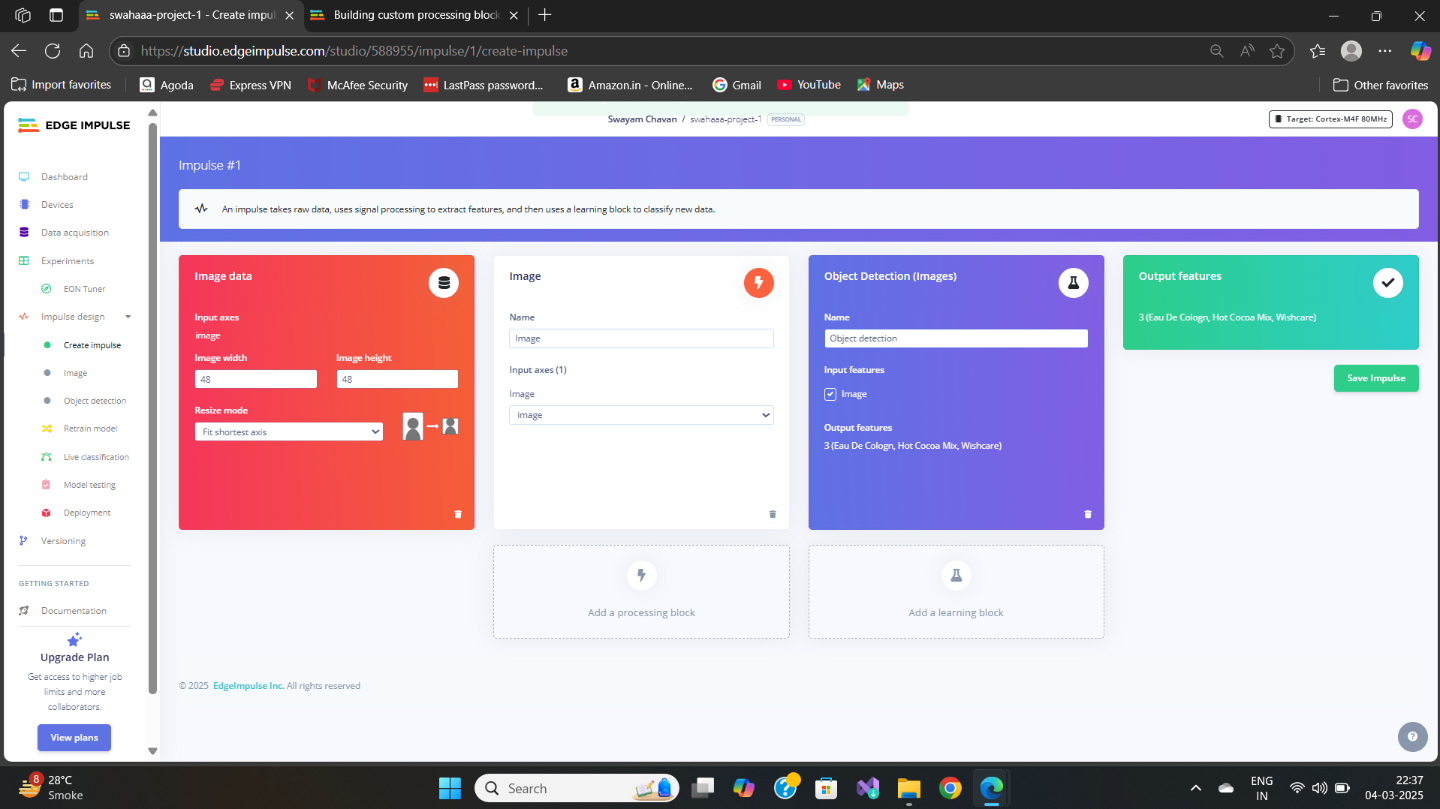
So we uploaded all the data with the four different labels such as Apple Lays, and Coke. So the system will only identify these objects when checking out. If you want to recognize any other objects other than these, you need to upload the dataset of them. Here we uploaded around 40 images for each object. Neural networks need to learn patterns in data sets, and the more data the better

**Labelling** **Data**

The labelling queue shows you all the unlabelled data in your dataset. Labelling objects is as easy as dragging a box around the object, and entering a label. To make the life a bit easier we try to automate this process by running an object tracking algorithm in the background. If you have the same object in multiple photos, we thus can move the boxes for you and you just need to confirm the new box. After dragging the boxes, click Savelabels and repeat this until your whole dataset is labelled.



**Designing** **An Impulse**



With the training set in place, you can design an impulse. An impulse takes the raw data, adjusts the image size, uses a preprocessing block to manipulate the image, and then uses a learning block to classify new data. Preprocessing blocks always return the same values for the same input (e.g. convert a colour image into a grayscale one), while learning blocks learn from past experiences.

For this system, we'll use the 'Images' preprocessing block. This block takes in the colour image, optionally makes the image grayscale, and then turns the data into a features array. Then we'll use a 'Transfer Learning' learning block, which takes all the images in and learns to distinguish between the two ('coffee', 'lamp') classes.

In the studio go to *Create impulse*, set the image width and image height to 96, the 'resize mode' to Fit shortest axis, and add the 'Images' and 'Object Detection (Images)' blocks. Then click Save impulse.

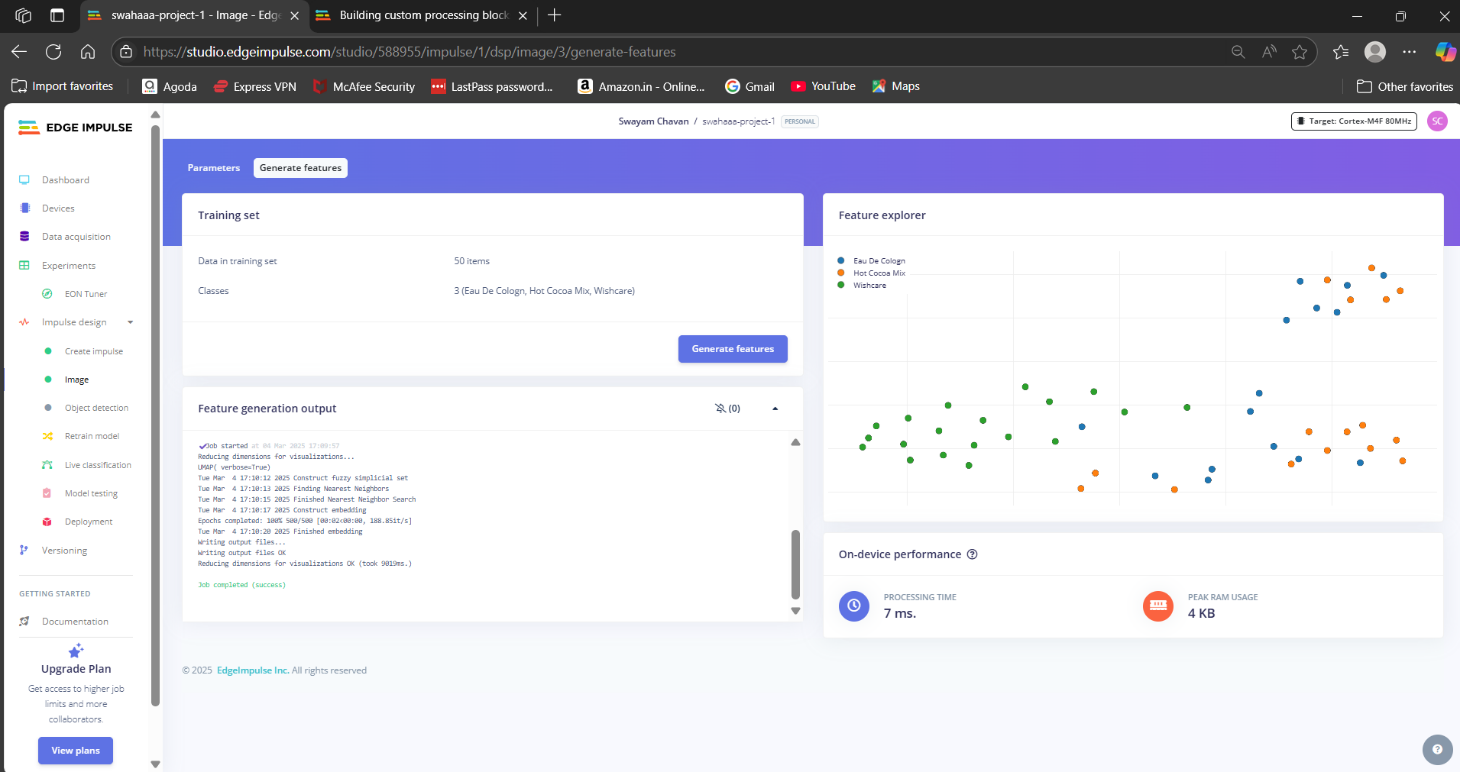
Then in the image tab, you can see the raw and processed features of every image. You can use the options to switch between 'RGB' and 'Grayscale' mode, but for now, leave the colour depth on 'RGB' and click *Save parameters*.

This will send you to the *Feature generation* screen. In here you'll:

* Resize all the data.
* Apply the processing block on all this data.
* Create a 3D visualization of your complete dataset.

Click **Generate features** to start the process.

Afterward the 'Feature explorer' will load. This is a plot of all the data in your dataset. Because images have a lot of dimensions (here: 96x96x3=27648 features) we run a process called 'dimensionality reduction' on the dataset before visualizing this. Here the 27648 features are compressed down to just 3, and then clustered based on similarity. Even though we have little data you can already see the clusters forming and can click on the dots to see which image belongs to which dot.

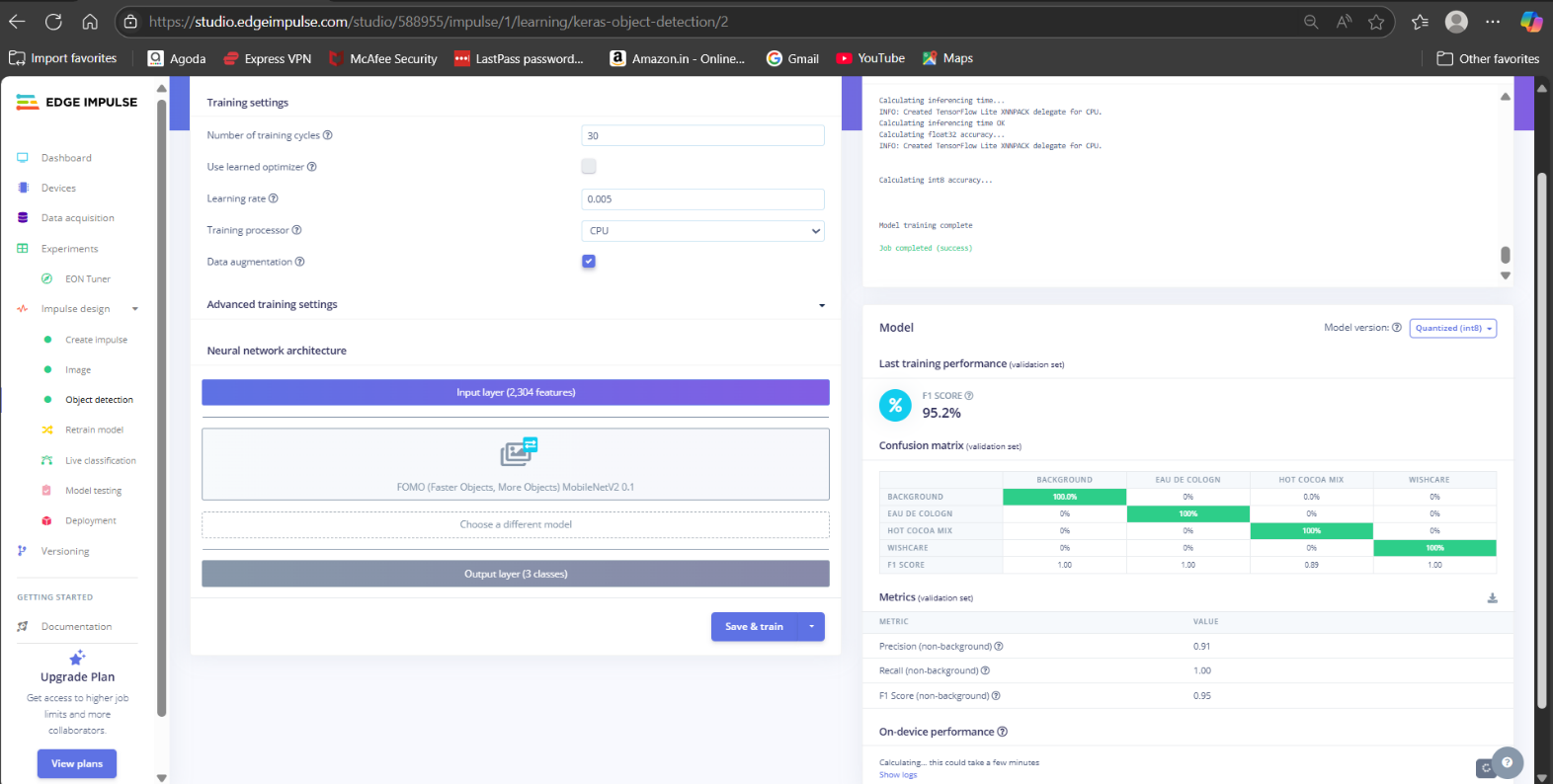


With all data processed it's time to start training a neural network. Neural networks are a set of algorithms, modelled loosely after the human brain, that are designed to recognize patterns. The network that we're training here will take the image data as an input, and try to map this to one of the three classes.

It's very hard to build a good working computer vision model from scratch, as you need a wide variety of input data to make the model generalize well, and training such models can take days on a GPU. To make this easier and faster we are using transfer learning. This lets you piggyback on a well-trained model, only retraining the upper layers of a neural network, leading to much more reliable models that train in a fraction of the time and work with substantially smaller datasets.

To configure the transfer learning model, click *Object detection* in the menu on the left. Here you can select the base model (the one selected by default will work, but you can change this based on your size requirements), and set the rate at which the network learns.

Leave all settings as-it is, and click *Start training*. After the model is done you'll see accuracy numbers below the training output. We have now trained our model.



With the model trained let's try it out on some test data. When collecting the data we split the data up between training and a testing dataset. The model was trained only on the training data, and thus we can use the data in the testing dataset to validate how well the model will work in the real world. This will help us ensure the model has not learned to overfit the training data, which is a common occurrence.

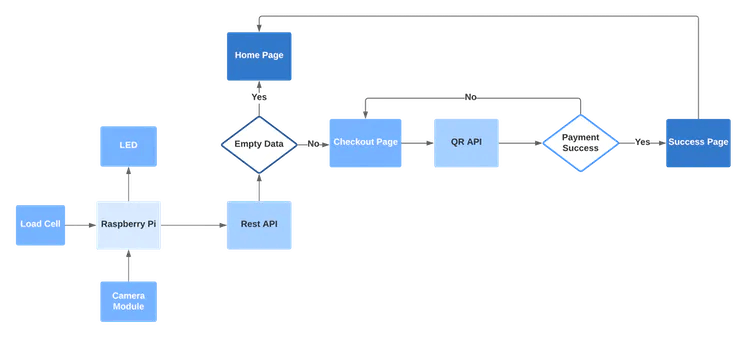
To validate your model, go to *Model testing* and select *Classify all*. Here we hit 93.75% precision, which is great for a model with so little data.

To see classification in detail, click the three dots next to an item, and select *Show classification*. This brings you to the Live classification screen with much more details on the file (you can also capture new data directly from your development board from here). This screen can help you determine why items were misclassified.

With the impulse designed, trained, and verified you can deploy this model back to your device. This makes the model run without an internet connection, minimizes latency, and runs with minimum power consumption. Edge Impulse can package up the complete impulse - including the preprocessing steps, neural network weights, and classification code - in a single C++ library or model file that you can include in your embedded software.

**4.3 Diagrams**

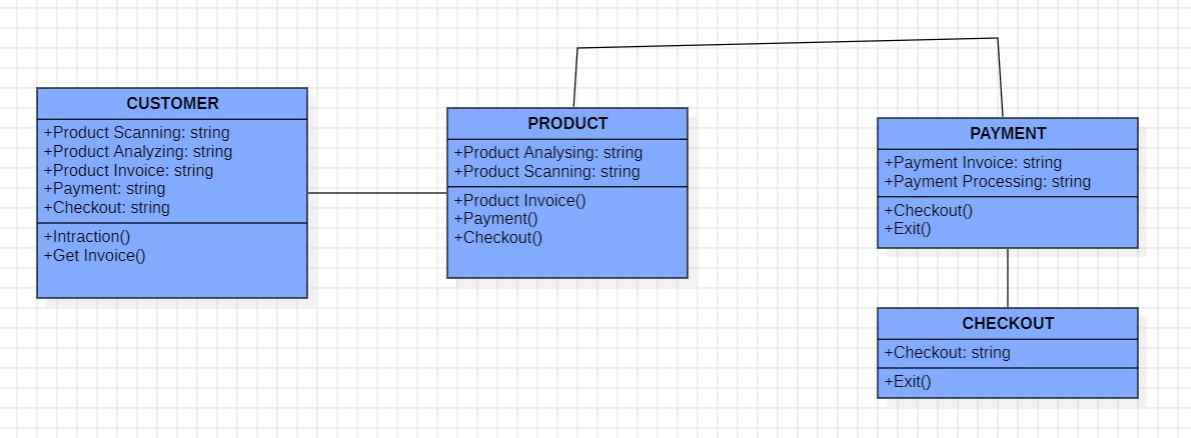
#### 4.3.1 E-R Diagram



##### Figure 10 E-R Diagram

* E-R (Entity-Relationship) Diagram is used to represents the relationship between entities in a table.
* ER diagrams represent the logical structure of databases.
* ER Diagram represent relationship between two database tables. If we assume entity is a database table, then all the columns of table are treating as attributes

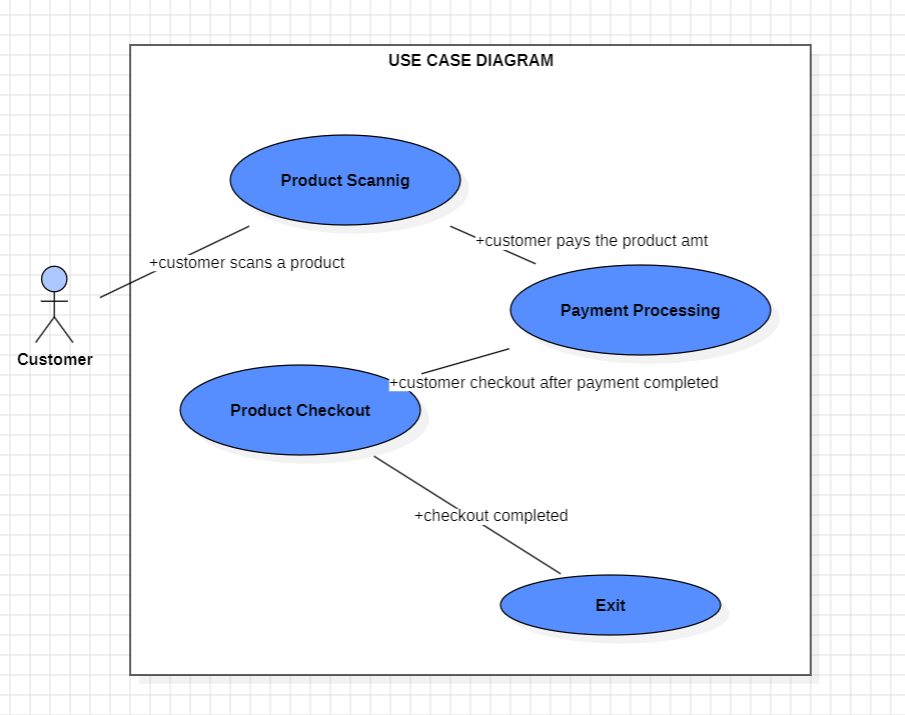
#### 4.3.2 Class Diagram



##### Figure 11 Class Diagram

* Class diagram is a static diagram. It represents the static view of an application.
* Class diagram describes the attributes and operations of a class and the constraints imposed on the system.
* The purpose of class diagram is to model the static view of an application

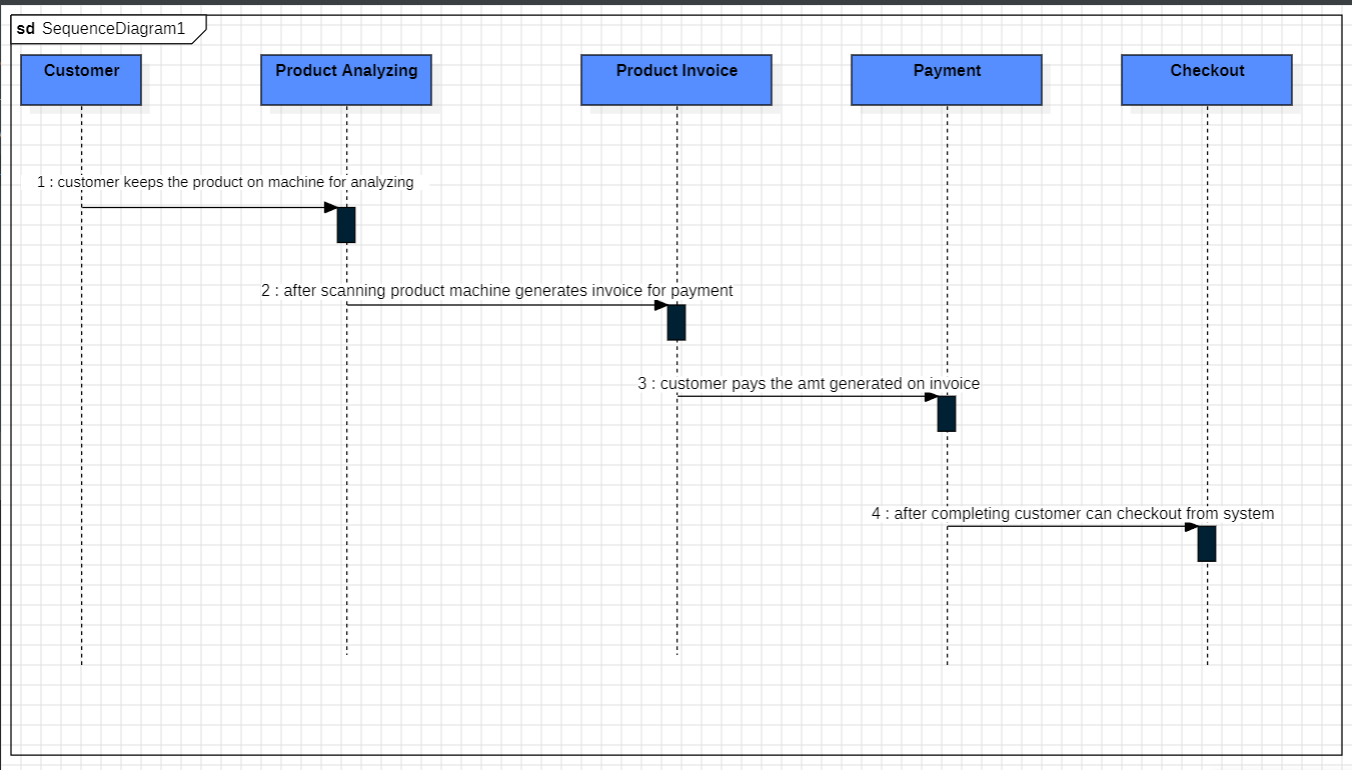
#### 4.3.3 Use Case Diagram



##### Figure 12 Use Case Diagram

* The use case diagram is usually referred to as behaviour diagram used to describe the actions of all customers in a system.
* All customers describe in use case are actors and the functionality as action of system.
* The Use case diagram is a collection of diagram and text together that make action on goal of a process.

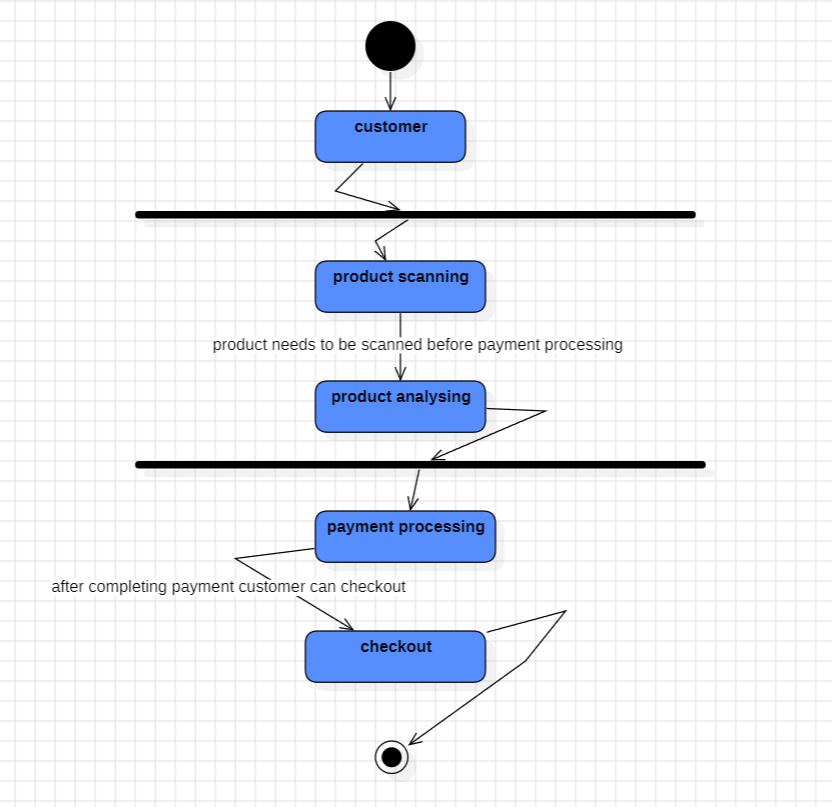
#### 4.3.4 Sequence Diagram



##### Figure 13 Sequence Diagram

* This is the Login Sequence Diagram of College Management System, where the admin will be able to login into his account using their own credentials.
* After login user can manage all the operations on Streams, Students, Courses, Attendance. All the pages such as Login, Attendance, Notification are secure, and user can access these pages after login.
* The diagram above helps us to demonstrate how to login page works in the system.
* The person will not be able to interact with these modules until and unless he provides his own identity

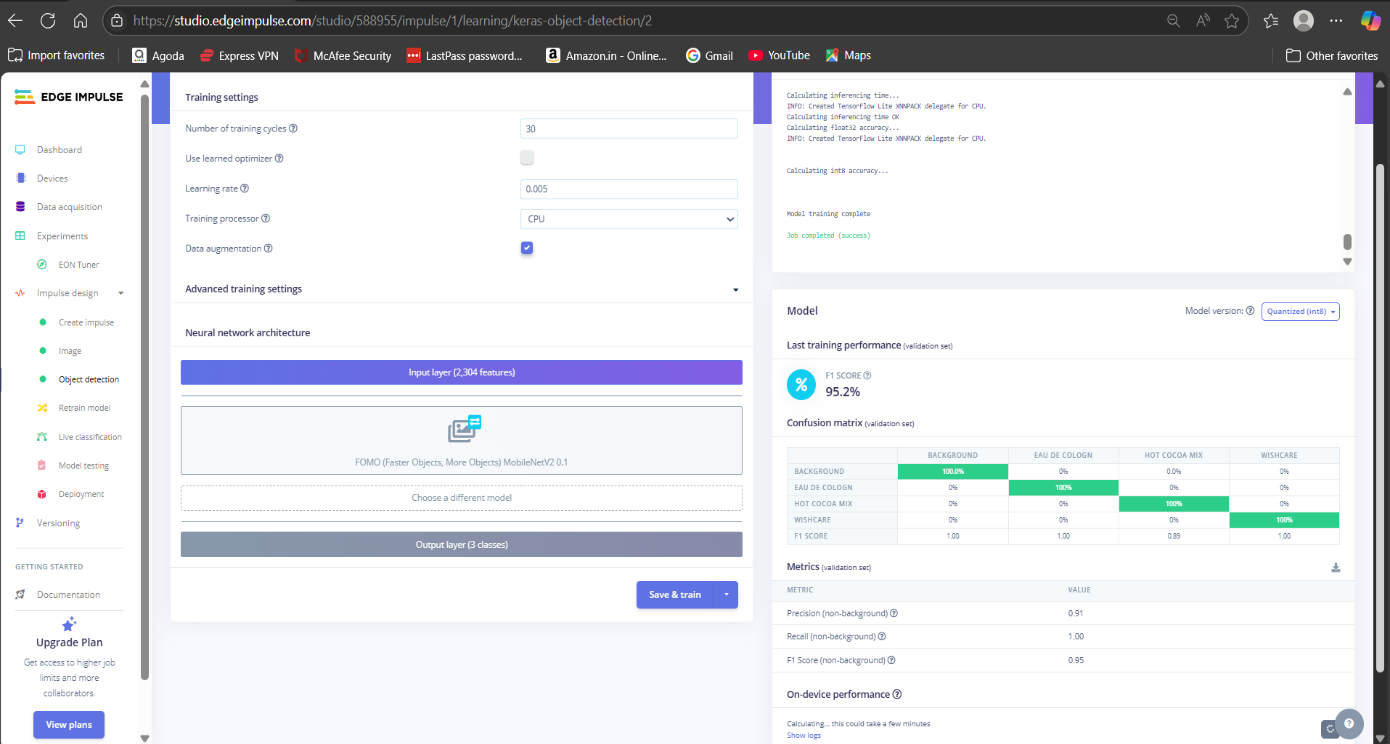
#### 4.3.5 Activity Diagram



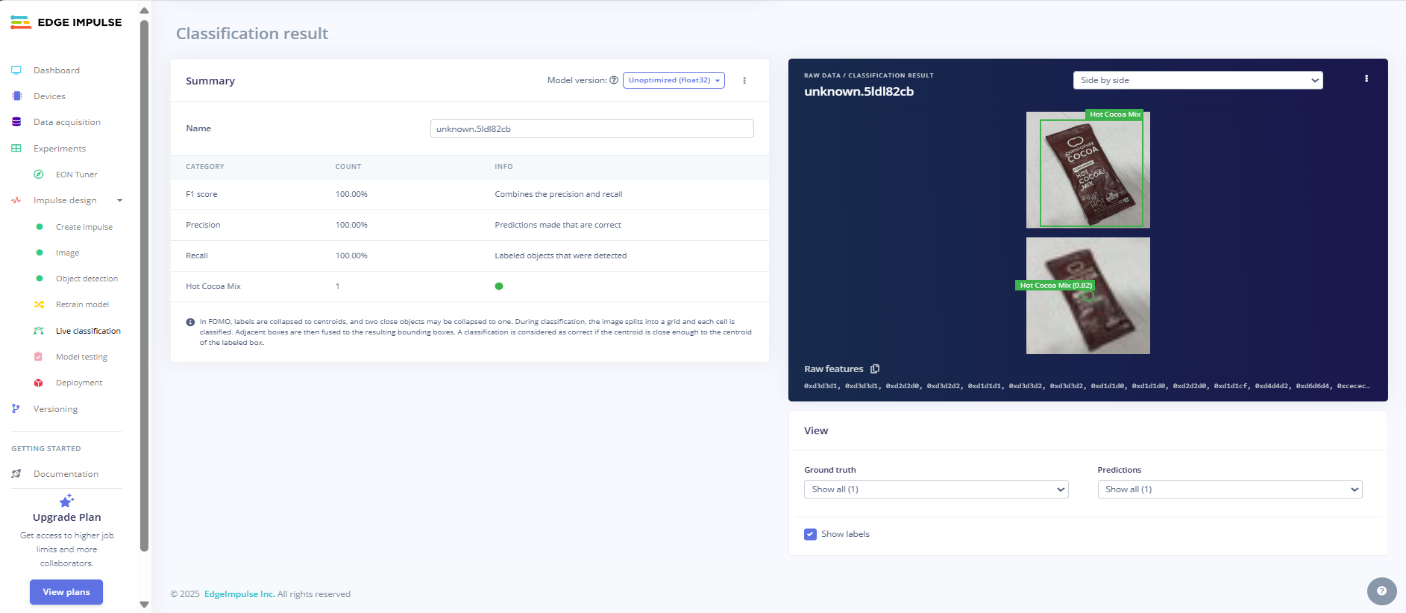
##### Figure 14 Activity Diagram

* The activity diagram used to describe flow of activity through a series of actions.
* An activity diagram shows the overall flow of control.
* Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system

#### 4.3.6 Interface Design



*Figure 15 Object Detection Interface*



*Figure 16 Live Classification*

**Chapter 5 Implementation and Testing**

### 5.1 Implementation Approaches

A screenshot of a project

Description automatically generated

### 5.2 Coding Details and Code Efficiency:

**5.2.1 Code Efficiency**

To improve performance and maintainability, the platform follows:

1. **ESP32-CAM**: Used for capturing barcode images and video streaming.
2. **Barcode Scanner**: Integrated with ESP32-CAM to scan product barcodes.
3. **Database**: Holds product information (e.g., price, product ID).
4. **Payment Gateway**: Used for processing the payment.

**5.2.2 Code Screenshot**

**A screenshot of a computer

AI-generated content may be incorrect.**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

*A screenshot of a computer

AI-generated content may be incorrect.*

**5.2.3 Explanation of Code Efficiency:**

1. **ESP32-CAM Initialization**:
   * The initCamera function configures the camera to start capturing images. Since the ESP32-CAM has limited resources, the camera initialization is optimized to handle only the necessary configurations.
2. **Wi-Fi Connectivity**:
   * The Wi-Fi connection process is done through the connectWiFi function. It ensures the system is connected to a Wi-Fi network for database access and payment processing.
3. **Barcode Scanning and Product Lookup**:
   * The barcode scanning process is handled by the scanBarcode function, where the ESP32-CAM captures an image, and the BarcodeReader library decodes the barcode.
   * The product price is fetched from Firebase using the getProductPrice function. This function retrieves the price associated with a barcode by querying a Firebase database.
4. **Database Integration (Firebase)**:
   * Firebase is used to store product information such as barcode, name, and price. This is an efficient method for managing product data because Firebase provides real-time database capabilities.
   * The database query returns a string representing the product price, which is converted into a float for price calculations.
5. **Checkout Flow**:
   * The checkout function scans a product, adds its price to the total, and prints the total to the Serial Monitor. It can be extended to interface with a payment gateway for payment processing.
   * The delay between scans (delay(5000)) ensures the system doesn’t overload with rapid requests.
6. **Code Efficiency**:
   * **Camera Performance**: The ESP32-CAM's camera is only activated when needed to capture an image, reducing resource consumption when not in use.
   * **Barcode Scanning**: The barcode decoding library is optimized for small, lightweight hardware. It efficiently decodes barcode images.
   * **Database Querying**: Using Firebase provides quick database access with minimal latency, and the data is cached automatically, improving retrieval times for frequently accessed information.

**5.2.4 Performance Considerations:**

1. **Memory Management**:
   * The ESP32-CAM has limited memory (520KB SRAM), and the code ensures that only essential operations are performed to reduce memory overhead. The camera is initialized only once during setup.
   * The Firebase queries are kept minimal, and only relevant fields (e.g., product price) are retrieved to reduce data load.
2. **Networking**:
   * Wi-Fi connection and Firebase communication are optimized to ensure smooth and fast operations, minimizing delays during the checkout process.
3. **Real-Time Product Lookup**:
   * By using Firebase for product information, the system ensures that updates to product prices or stock are reflected in real time.
4. **Payment Gateway**:
   * Integration with a payment gateway like PayPal, Stripe, or others can be done in the checkout function. This part can be optimized further by asynchronously processing payments to avoid blocking the main loop.

**Chapter 6: Results and Discussion**

**6.1.1 Test Reports**

The Autobill Instant Checkout System effectively managed and operated successfully in various scenarios.

| Feature | Tested for | Outcome |
| --- | --- | --- |
| Login | Secure User Login Achieved | ✅ Passed |
| Object Detection | Completed Object Detection Process | ✅ Passed |
| Product Display | Effective Product Display | ✅ Passed |
| Payment Screen | Correct Payment Barcode Displayed on Screen | ✅ Passed |
| Database | Secure Storage of Data | ✅ Passed |
| Payment Successful | Successful Payment Receipt Uploaded | ✅ Passed |

**6.2 System Functioning/ User Documentation**

How the Platform Works

**6.2.1 System Components**

1. **ESP32-CAM Module**: A small, low-cost microcontroller with integrated camera for capturing barcode images.
2. **Barcode Scanner**: Integrated into the ESP32-CAM, allowing it to decode barcodes from product packaging.
3. **Payment Gateway Integration**: Links to an external payment system (e.g., PayPal, Stripe) to facilitate secure and seamless payments once the total cost is calculated.

**6.2.2 System Features**

1. **Instant Barcode Scanning**: Users simply need to scan the barcode on each item they wish to purchase. The system automatically retrieves the product price from the database.
2. **Real-time Total Calculation**: The system continuously adds the prices of scanned products to calculate the total cost in real-time.
3. **Online Payment Processing**: Once the total is calculated, the system directs users to an online payment gateway for quick and secure payment processing.
4. **Automated Checkout**: After payment, users can receive a confirmation message or receipt. The process is designed for fast and efficient shopping without the need for manual checkout or interaction with a cashier.
   * 1. **How the System Works**
        1. **Wi-Fi and Camera Initialization**: The system connects to the Wi-Fi network for communication with the database and payment gateway, The camera is initialized to start scanning barcodes.
        2. **Barcode Scanning**: Once the user presents an item, the system captures an image of the barcode using the ESP32-CAM.
        3. **Database Lookup**: The system queries the database (Firebase) using the decoded barcode to retrieve product information like price and description.
        4. **Total Calculation**: Each item’s price is added to the total cost. The total is continuously updated in real-time as more items are scanned.
        5. **Payment Gateway Integration**: Once all items are scanned, the user is directed to an online payment gateway (like PayPal, Stripe, or any suitable API).
        6. **Completion**: After successful payment, the user receives a confirmation message. The process completes, and the user is free to leave the store with their purchases.
     2. **User Instructions**

* **Step 1: Start the System -** Ensure the ESP32-CAM is powered on and connected to a Wi-Fi network.The system should automatically initialize the camera and connect to the database.
* **Step 2: Scan Products -** Present each item’s barcode to the ESP32-CAM. The camera will automatically scan the barcode and send the image to the barcode decoder. The system will fetch the product’s information (like price) from the connected database.
* **Step 3: View Total -** As you continue scanning items, the system will display the total amount of the items scanned on the user interface or the serial monitor. You can see the running total in real-time as each product is added.
* **Step 4: Payment Processing -** Once all items are scanned, the system will direct you to an online payment gateway to make the payment. You will be presented with the total amount due and a payment option. You will be presented with the total amount due and a payment option.
* **Step 5: Confirmation -** After a successful payment, you will receive a confirmation message or receipt confirming the payment and completion of your transaction. You can proceed to leave the store with your purchased products.

**Chapter 7: Conclusion and Future Scope**

**7.1.1 Conclusion**

The Autobill Instant Checkout System using the ESP32-CAM is a transformative solution designed to simplify and expedite the checkout process in retail environments. By integrating advanced technologies such as barcode scanning, real-time database access, and payment gateway integration, the system offers an efficient, user-friendly, and automated checkout experience for customers.

With the ESP32-CAM acting as the central component for both scanning and data processing, the system ensures that items are quickly identified and added to the total cost without manual intervention. The system’s ability to seamlessly connect to online databases (such as Firebase) for real-time product information and process payments securely via an integrated payment gateway provides a comprehensive solution to modern retail challenges.

This system is beneficial for both retailers and customers, offering a faster, more convenient shopping experience that eliminates the need for long checkout lines, reduces human error, and accelerates the overall purchasing process.

**7.1.2 Future Scope of the Project**

While the current version of the Autobill Instant Checkout System is functional and efficient, there are several opportunities for future enhancement and expansion.

**Multi-Item Scanning and Detection**

* **Current Limitation**: The system scans one item at a time, which may limit the speed of checkout in high-traffic retail environments.
* **Future Scope**: Implementing multi-item scanning would allow the system to process several items simultaneously. This could involve advanced image processing techniques to detect multiple barcodes in a single frame or use additional sensors and cameras to speed up the scanning process.

**AI-based Object Detection for Non-Barcoded Items**

* **Current Limitation**: The system relies solely on barcodes for product identification, which limits its capability with non-barcoded or unique items.
* **Future Scope**: Integrating AI-based object detection (using TensorFlow or OpenCV libraries) can enable the system to identify products by image recognition. This would allow the system to process items without barcodes, making the checkout process more inclusive for various product types.

#### Security Issues

The Auto Bill Instant Checkout System is vulnerable to various security threats, which can compromise the integrity, confidentiality, and availability of sensitive data. By acknowledging and addressing these security issues, the Auto Bill Instant Checkout System can ensure protecting customers and merchants alike. Key security issues include:

* Cross-Site Scripting (XSS) Attacks: Malicious scripts can be injected into the

system, stealing customer data or taking control of user sessions.

* Authentication and Authorization Vulnerabilities: Weak passwords, inadequate access controls, or session hijacking can compromise system security.
* Encryption and Tokenization Issues: Inadequate or improperly implemented encryption can expose sensitive data.
* Encryption and Tokenization Issues: Inadequate or improperly implemented encryption can expose sensitive data.
* System Updates and Patching: Failure to regularly update and patch the system can leave it exposed to known vulnerabilities.

#### References

<https://youtu.be/MRi8rDEUH08?si=ZQRIOQep6D7W7IZ3> <https://www.w3schools.com/training/aws/introduction-to-aws-iot.php>

<https://github.com/CodersCafeTech/AutoBill>

<https://en.wikipedia.org/wiki/StarUML>

<https://en.wikipedia.org/wiki/Artificial_intelligence>

<https://en.wikipedia.org/wiki/Machine_learning>

<https://en.wikipedia.org/wiki/Computer_vision>

<https://en.wikipedia.org/wiki/Self-checkout>

<https://en.wikipedia.org/wiki/Retail_automation>

<https://en.wikipedia.org/wiki/Barcode_reader>

<https://en.wikipedia.org/wiki/Payment_gateway>

<https://en.wikipedia.org/wiki/Natural_language_processing>

<https://en.wikipedia.org/wiki/Internet_of_things>

<https://en.wikipedia.org/wiki/Edge_computing>

<https://en.wikipedia.org/wiki/Biometric_authentication>

<https://en.wikipedia.org/wiki/Payment_system>

#### Summary

The Auto Bill Instant Checkout System is an innovative IoT-based project designed to revolutionize the checkout experience for customers and merchants. Leveraging Internet of Things (IoT) technology, Artificial Intelligence (AI), and Machine Learning (ML), this system provides a fast, secure, and seamless checkout experience. The system consists of IoT-enabled point-of-sale (POS) devices, real-time payment processing gateways, AI-powered payment fraud detection, and ML-driven customer behaviour analysis. Integrated with accounting and inventory systems, AutoBill Instant Checkout System automates checkout processes, payment processing, and inventory management.

Key features include fast and secure checkout experiences, real-time payment processing, automated inventory management, personalized customer offers, and real-time analytics and reporting. Benefits include improved customer satisfaction, increased sales and revenue, reduced payment processing errors, enhanced security and compliance, and streamlined inventory management. Technically, the system operates on Windows, utilizing Python/JavaScript programming languages, NodeJS, and IoT protocols. Payment gateways include Gpay and Paytm. Target applications include retail and e-commerce, hospitality and food service, healthcare and medical billing, and financial institutions and banking.

Future scope includes integration with wearable devices, expansion to mobile payments, enhanced AI/ML capabilities, and blockchain-based security. By harnessing IoT technology, Auto Bill Instant Checkout System transforms the checkout experience, providing efficiency, security, and convenience for merchants and customers alike.

#### Plagiarism Report

[Website](http://plagiarisma.net/) link –<http://plagiarisma.net/>

A screenshot of a computer

Description automatically generated