Test Report

For

Automated Retail Checkout System using YOLOv5 in Computer Vision

(Project Id: PCS24-30)

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE

Session 2023-24

Mr. Anmol Jain
(Project Guide)

Mr. Abhishek Goyal
(Testing Lab Faculty)

Submitted by

Kshiteesh Kumar (2000290120088) - 7B
Kumari Bhavya Chaubey (2000290120089) – 7B
Nandita Yadav (2000290120100) – 7B

Submitted To:

Prof. Neha Shukla
Project Coordinator
Dept. of Computer Science

TEST REPORT FOR AUTOMATED RETAIL CHECKOUT SYSTEM USING YOLOV5 IN COMPUTER VISION

ChangeLog

Version	Change Date	By	Description
001	30.10.2023	Nandita Yadav	Initial Draft

1 Introduction

Our implementation focuses on enhancing customer experience by creating a system that can scan the bought items, detect these items with the help of pre-trained model and generate a bill containing the total cost that needs to be paid.
When input image/video containing the items is passed to the system, the system detects whether the data exists in the dataset or not. If yes, then the pre-trained model detects and recognizes the object. Then the detect method, that makes use of Tkinter, has a list that contains the per unit cost of each item. The bill is then created in the dataframe and the price of each item is calculated. These price values are then added to the dataframe and output is displayed.

1.1 Scope

1.1.1 In Scope

Functional Requirements:

1. The image/video must be loaded to the GUI.
2. The image/video format should be .jpg or .gif.
3. The output of the predicted item must be correct.
4. The bill must contain sum of all items.
5. The system should provide a user-friendly interface for users to upload image/video.

Non-Functional Requirements:

1. The confidence threshold must be 0.25.
2. The speed to perform object detection should be high.
3. It should be able to handle a scalable large number of items in image.
4. The accuracy of object detection should be high.

1.1.2 Out of Scope

1. The images must be limited to only the grocery items.
2. The image dataset consists of only 5 test items and therefore, the uploaded images/video must be of 5 items.

1.2 Quality Objective

- Ensure the Application Under Test conforms to functional and non-functional requirements.
- Ensure that the dataset on which the model is trained, contatins all the necessary images.
- Ensure that the price of each quantity is written.
- Ensure the AUT meets the quality specifications defined by the client.
- Bugs/issues are identified and fixed before go live.
- Prioritize a user-friendly and intuitive interface to enhance user experience, making it easy for users to upload images

1.3 Roles and Responsibilities

- **QA Analyst:** Nandita Yadav
  Role: To conduct testing on software, websites, and other technical products to identify and resolve bugs, defects, and other potential issues.

- **Test Manager:** Mr. Abhishek Goyal
  Role: To manage all test processes, including test plans, resources, costs, timescales, test deliverables and traceability.

- **Configuration Manager**: Prof Akankasha
  Role: To ensure daily end-to-end delivery of CM services in accordance with the configuration management plan

- **Developers:** Kshiteesh Kumar, Bhavya Chaubey, Nandita Yadav
  Role: To Develop the model and trained it.

- **Installation Team**: Kshiteesh Kumar, Bhavya Chaubey, Nandita Yadav
  Role: Responsible for smooth execution of the program

2 Test Methodology

2.1 Overview

The test methodology selected for the project is Agile. An Agile methodology is the most suitable for this project. It allows for flexibility, ongoing testing, and adaptation, which are essential for projects that involve machine learning, image processing, and AI. Agile enables you to respond to changing requirements and refine the style transfer algorithm as you gain insights from testing and user feedback.

It is sequential development process that flows like a waterfall through all phases of a project (analysis, design, development, and testing, for example), with each phase completely wrapping up before the next phase begins.

2.2 Test Levels

The testing to be performed is white box testing.
The testing is performed by the developers team along with QA and Configuration Manager.

1. **Unit Testing:**
   - Test individual components of the system, such as YOLOv5 model layers and data preprocessing functions.
   - Ensure that each unit operates as expected and handles edge cases correctly.

2. **Integration Testing:**
    - Test the integration of various modules and components within the system.
    - Verify that data flows correctly between components and that the YOLOv5 model integrates seamlessly with the rest of the system.

3. **Functional Testing:**
    - Verify that the system can correctly identify and classify retail items.
    - Test different types of products, including various sizes, shapes, and labels.
    - Ensure that object detection and recognition work accurately.

4. **Performance Testing:**
    - Evaluate the system's speed and responsiveness, ensuring that it can handle real-time checkout scenarios.
    - Assess how the system performs under various workloads and peak usage conditions.

5. **Regression Testing:**
    - Continuously run regression tests to identify and fix any issues that may arise with system updates or changes.

2.3 Test Completeness

- 100% test coverage
- All Manual & Automated Test cases executed
- All open bugs are fixed or will be fixed in next release
- Images are correctly predicted.
- Images are loaded efficiently.
- The model generates output correctly.

3 Test Deliverables

.

- **Test Cases**

Test Case No.	Test Case Name	Description
1	Detect Orange	The model must be able to detect and recognize orange.
2	Detect Banana	The model must be able to detect and recognize banana.
3	Detect Apples and bill	The model must be able to detect and recognize multiple apples
4	Detect multiple objects	Multiple objects are passed and detected
5	Display Total Price	The total price of the items must be displayed.
6	Display Total Price	The total price of the items must be displayed.
7	Image Format	The image should be of .jpg format
8	Image Format	The image should be of .jpg format

- **Requirement Traceability Matrix**

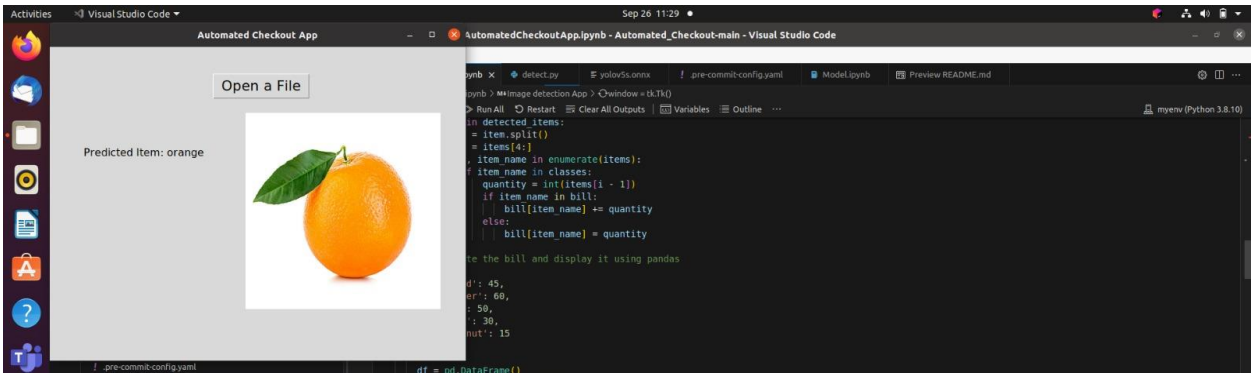
Input	Expected Output	Actual Output
Orange image	Orange is predicted	Orange is predicted.
Banana Image	Banana is predicted	Banana is predicted
3 Apples Image	Apple predicted	Apple prediced
Apple, Banana, Orange	Apple, Banana, Orange is Predicted	Apple, Banana, Orange is Predicted
3 apple, 2 banana, 3 oranges	290	290
2 apple, 2 banana, 2 orange	200	200
orange .jpg image	orange is predicted	Orange is predicted.
apple.png image	no output	no output

- **Bug Report**

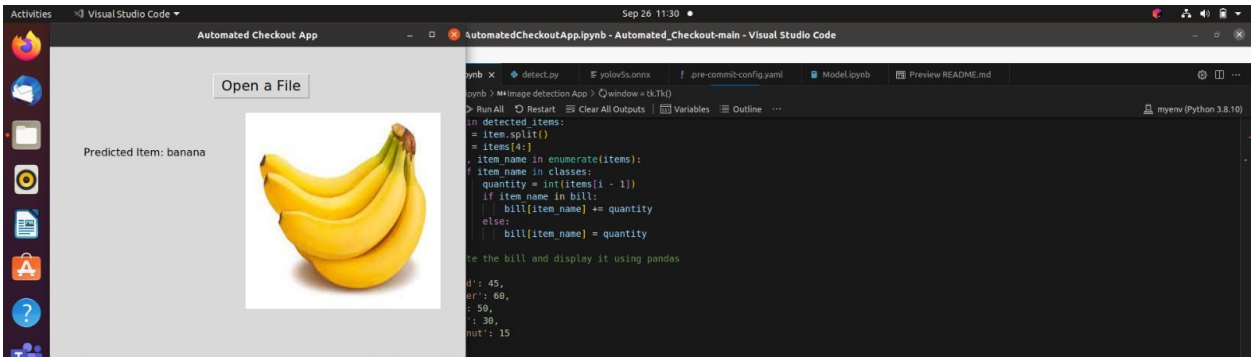
Test Case No.	Test Case Name	Input	Expected Output	Actual Output	Test Case Pass/Fail
1	Detect Orange	Orange image	Orange is predicted	Orange is predicted.	Pass
2	Detect Banana	Banana Image	Banana is predicted	Banana is predicted	Pass
3	Detect Apples and bill	3 Apples Image	Apple predicted	Apple prediced	Pass
4	Detect multiple objects	Apple, Banana, Orange	Apple, Banana, Orange is Predicted	Apple, Banana, Orange is Predicted	Pass
5	Display Total Price	3 apple, 2 banana, 3 oranges	290	290	Pass
6	Display Total Price	2 apple, 2 banana, 2 orange	200	200	Pass
7	Image Format	orange .jpg image	orange is predicted	Orange is predicted.	Pass
8	Image Format	apple.png image	no output	no output	Pass

- **Test case Output Images**

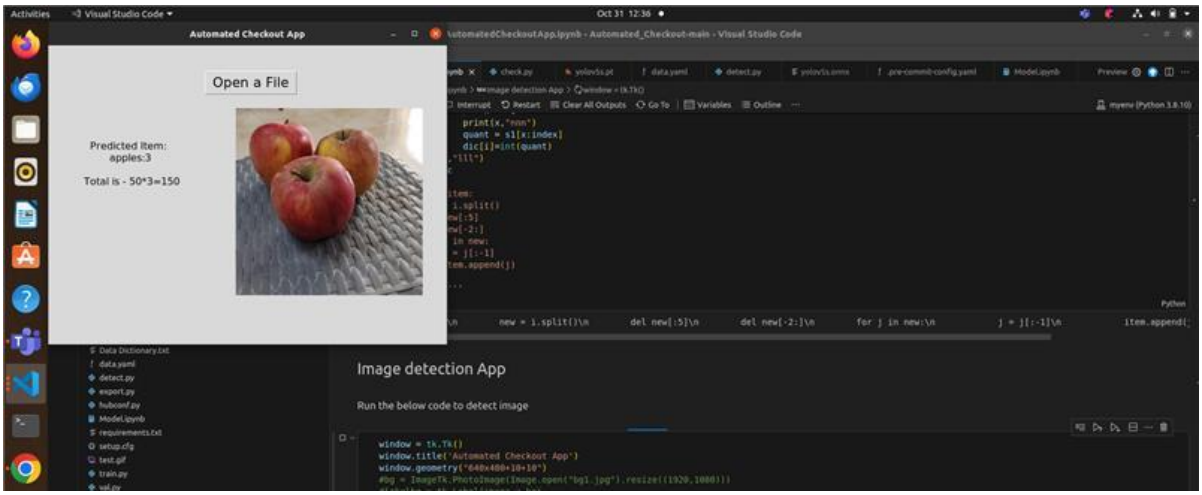
Test case 1:



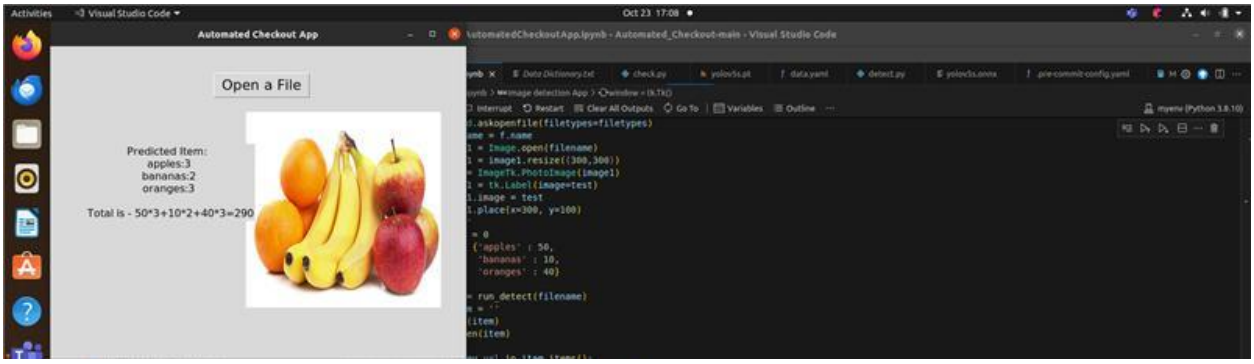
Test case 2:



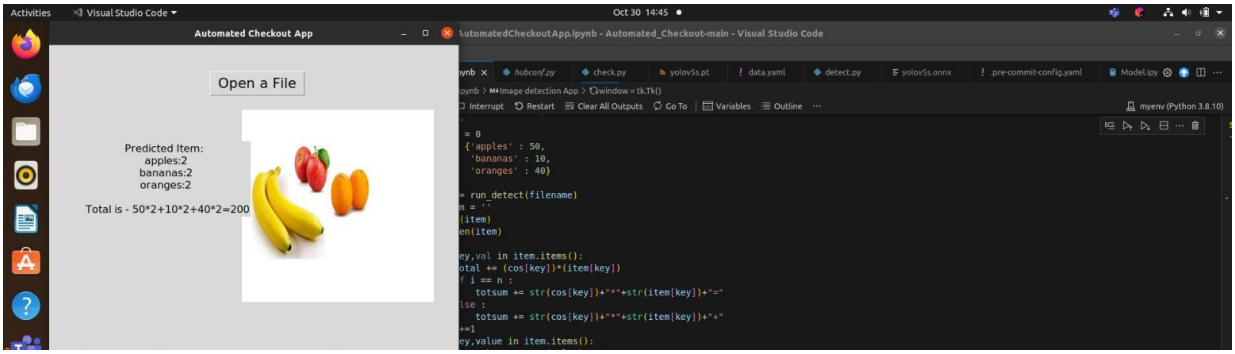
Test case 3:



Test case 5:



Test case 6:



4 Resource & Environment Needs

4.1 Testing Tools

No testing tool is required. Manual Testing is done.

4.2 Test Environment

Following **software's** are required in addition to client-specific software.

- VS code
- Operating System
- Camera

5 Terms/Acronyms

TERM/ACRONYM	DEFINITION
API	Application Program Interface
AUT	Application Under Test