SmartBin: A Windows Application for Waste Management

Version 1.0

By: Svetya Koppisetty, Curtis Hayes, Diego Chiok, and Edgar Legaspi

Mentor: Yuan Li

California State University San Marcos: CS 490

February 10, 2025

Table of Contents

Cover Page i
Table of Contents ii

1. **Introduction** 1

2. **Software Requirements** 2
User Interface Requirements 2
Functional Requirements 2
Quality of Service Requirement #1 2
Quality of Service Requirement #2 2

3. Architectural Design 3
Administrative Client Design 3
User Client Design 3
Server Design 3

4. Implementation Plan 4
Implementation Platform and Frameworks 4
Implementation Tasks 4
Implementation Schedule 4

References iii

1. Introduction

SmartBin is a windows application using the detect-waste ResNet50 ai model to detect any waste within a scene. The application is designed to run on Snapdragon X Elite laptop. Our application addresses the problem of littering by using an object detection model to identify and categorize waste in images, promoting proper disposal in a society where a significant portion of waste is mismanaged. Our goal is to leverage modern AI-driven computer vision models to develop a system capable of detecting waste efficiently. We will design our application to function as a standalone product that enables users to detect waste. However, we envision expanding its scope beyond individual use, involving it into use for autonomous systems that are capable of classifying and properly disposing of waste.

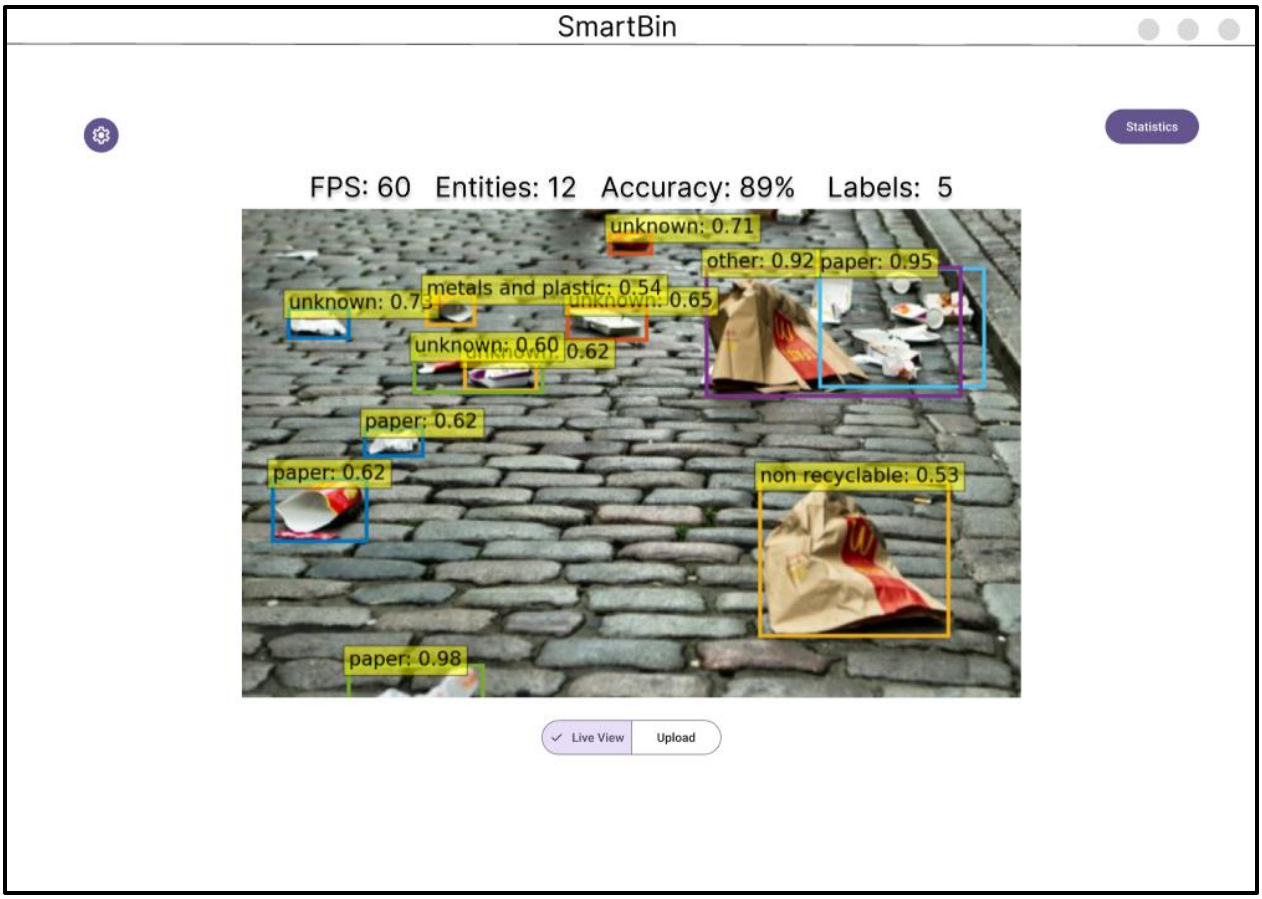
2. Software Requirements

2.1 User Interface Requirements

2.1.1 Toggling between Live Feed and Uploading an Image

The first UI Mockup features a toggle button located in the bottom right side of the screen. It will allow users to switch between Live View Mode and the Upload Image Screen. The input consists of the user interacting with toggle button, while the output is either the Live View Mode or the Upload Image Screen. To ensure a seamless user experience, the toggling feature is made to be responsive and easy to use. Since the toggle button performs a straightforward action, potential errors are low.

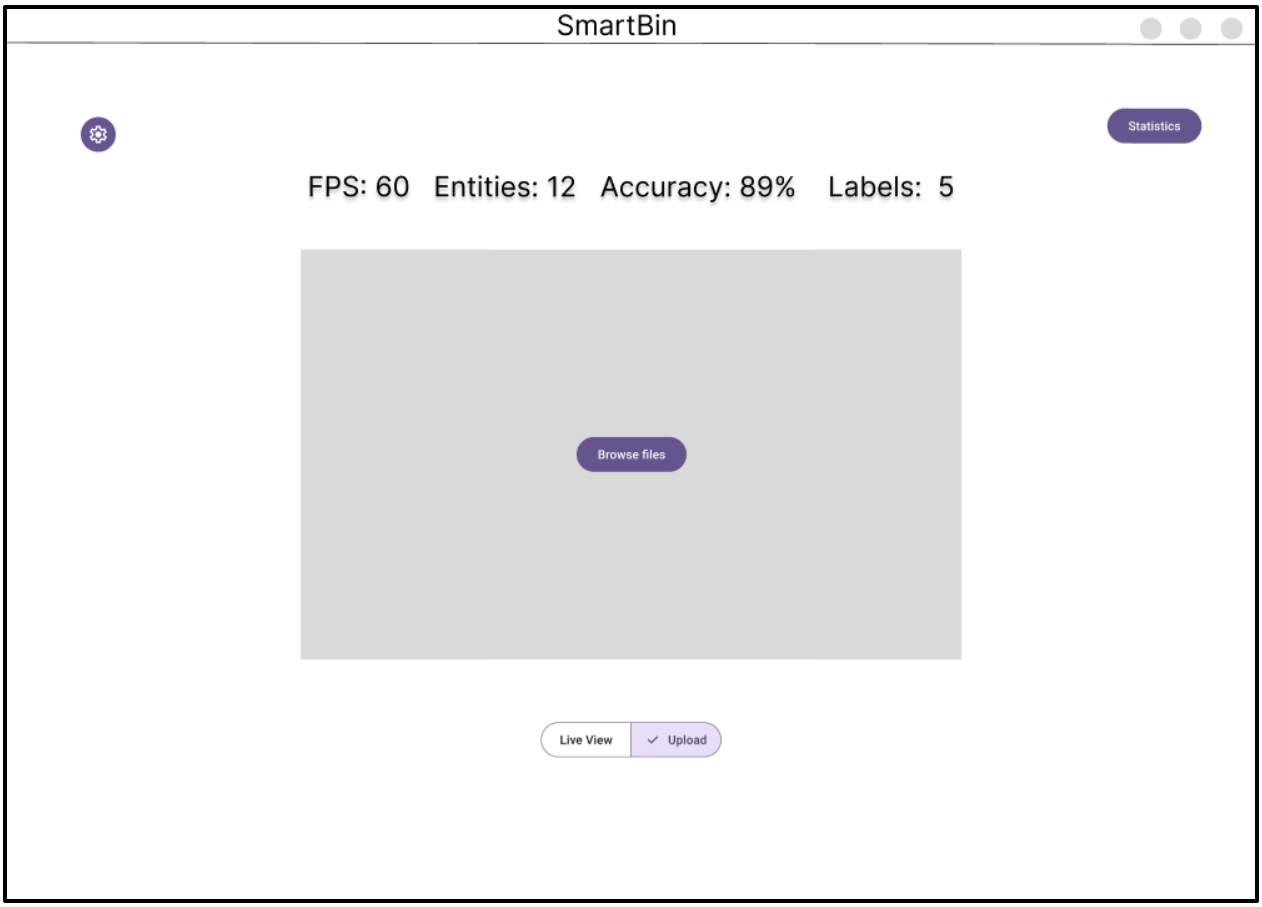
Figure 1: Live Feed and Uploading an Image



2.1.2 Uploading an image to detect waste

The second UI Mockup is an upload page. This will be opened when the user toggles the switch to Upload Image in the first UI Mockup. This screen will allow users to upload an image, which then will be analyzed to detect and categorize the waste. Potential errors include: no file uploaded, incompatible file type, file upload failure. The input would be the user's selected image, and the output would be the same image with bounding boxes around the detected waste items that are labeled accordingly. The layout will take up about one-third of the screen when toggled. Once the user clicks the upload button, the window will close, returning to the main interface.

Figure 2: Upload Image Page



2.1.3 Performance / Stats

The Third UI mockup will focus on the performance chart. It is designed to be user-friendly, informative, and visually engaging. Most of the user input is automatic, using data gathered from the AI model during waste detection. Detection speed, processing latency, accuracy, power efficiency, user feedback loop, and session summary are among the real-time, interactive performance indicators that are shown to the user in graphical formats. The layout of the interface is divided with key performance indicators at the top, interactive charts in the middle, and session summary and user feedback at the bottom. The report layout allows users to download detailed performance data in a structured format, such as detection history, estimated environmental impact data, commonly detected logs, and overall session stats. Options such as Live Detection, Performance, and Settings are easily accessible through menu structures.

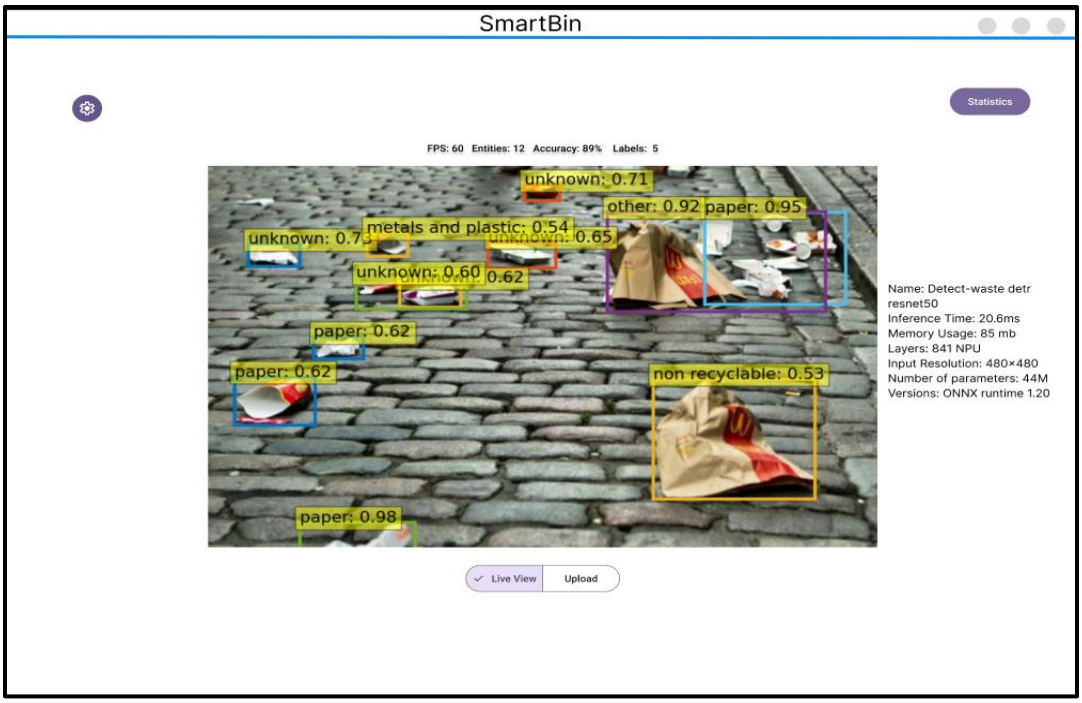
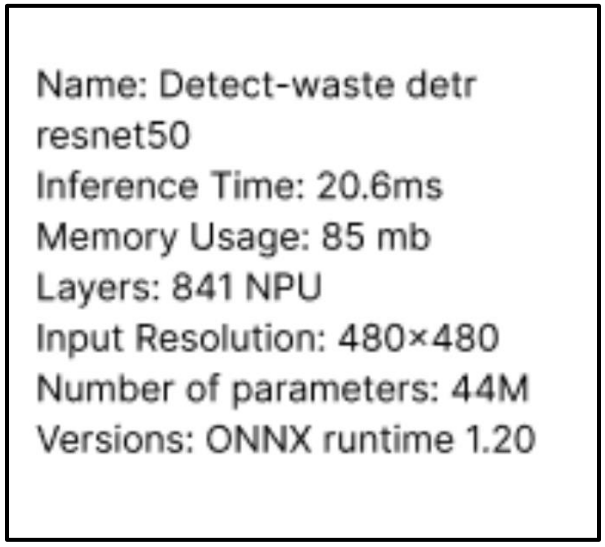


Figure 3: Toggle for Information

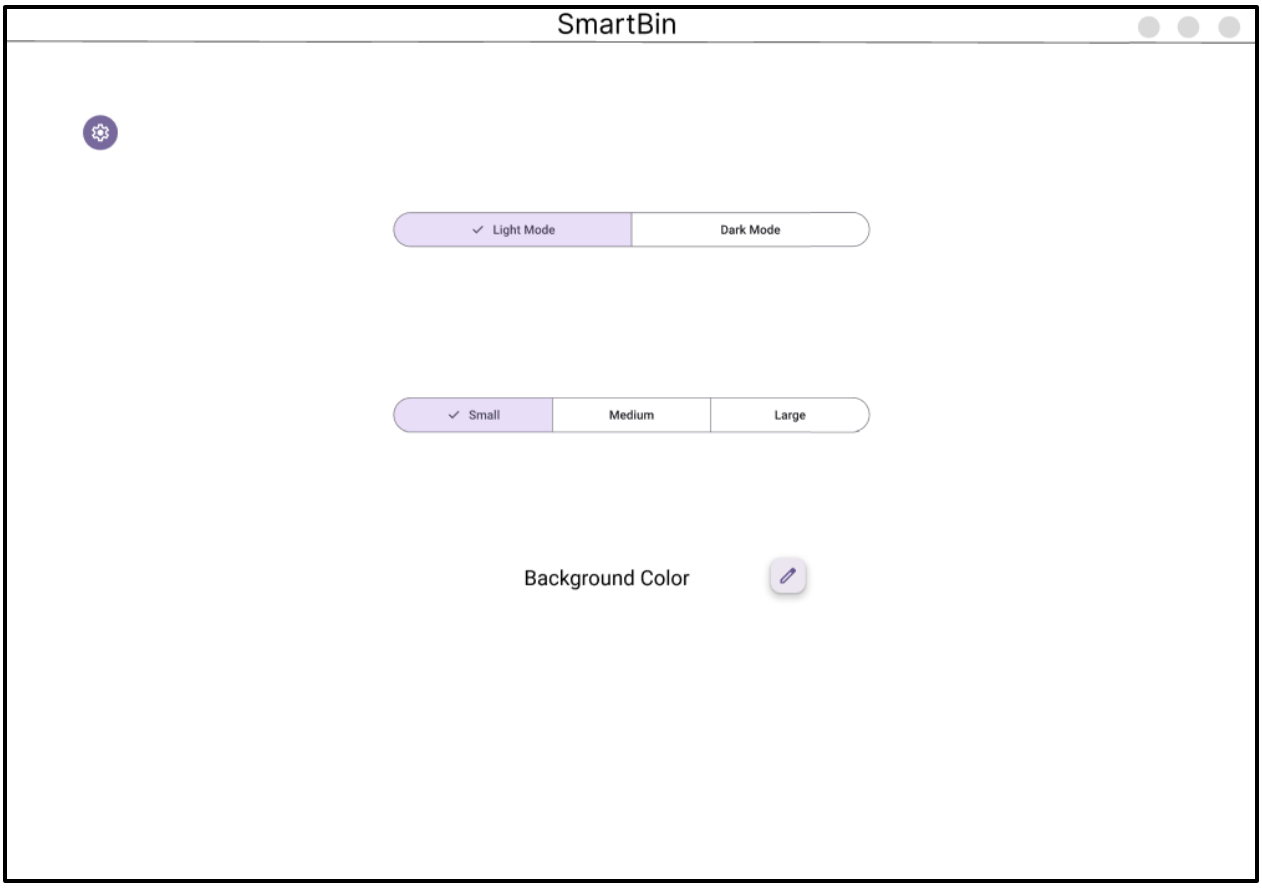
Information Window



2.1.4 Settings

The final mockup will feature a settings page that allows users to customize the app according to their preferences. When the settings button is toggled, the interface will seamlessly switch to the settings page within the same window. This page will include options such as a light/dark mode toggle, adjustable text sizes, and a text box where users can enter a custom background color using hex codes for a personalized experience.

Figure 4



2.2 Functional Requirements

Priority Definitions

Req#	Requirement	Priority	Date Rvwd	Reviewed / Approved
FR_PD_1	The system shall allow the user to have a scanning feature where the system uses a camera to scan.	1	2/10/25	Diego Chiok, Edgar Legaspi, Curtis Hayes, Svetya Koppisetty
FR_PD_2	The system shall detect all instances of litter that are within the scope of the scan.	1	2/10/25	Diego Chiok, Edgar Legaspi, Curtis Hayes, Svetya Koppisetty
FR_PD_3	The system shall detect all instances of recyclables (i.e. bottles, etc) that are within the scope and label these items on the screen.	1	2/10/25	Diego Chiok, Edgar Legaspi, Curtis Hayes, Svetya Koppisetty
FR_PD_4	The system shall have a hazard detection with a warning system that detects and warns users of potential danger.	3	2/10/25	Diego Chiok, Edgar Legaspi, Curtis Hayes, Svetya Koppisetty
FR_PD_5	The system shall allow users to toggle any features in the application they don't desire, i.e. toggling bounding boxes, waste count, accuracy, etc.	2	2/10/25	Edgar Legaspi, Diego Chiok, Curtis Hayes, Svetya Koppisetty

Req#	Requirement	Priority	Date Rvwd	Reviewed / Approved
FR_PD_6	The system shall work for Qualcomm devices and uploaded to AI Hub with no complications or caveats	1	2/10/25	Edgar Legaspi, Diego Chiok, Curtis Hayes, Svetya Koppisetty
FR_PD_7	The system shall have a toggle to switch between a file upload and a live feed.	1	2/19/25	Diego Chiok, Edgar Legaspi, Curtis Hayes, Svetya Koppisetty
FR_PD_8	The system shall have a button that allows users to upload files	1	2/19/25	Diego Chiok, Edgar Legaspi, Curtis Hayes, Svetya Koppisetty
FR_PD_9	The system shall have a button that allows live feed to stop	1	2/19/25	Diego Chiok, Edgar Legaspi, Curtis Hayes, Svetya Koppisetty
FR_PD_10	The system shall have a button to begin live feed	1	2/19/25	Diego Chiok, Edgar Legaspi, Curtis Hayes, Svetya Koppisetty
FR_PD_11	The system shall have a button to bring down performance statistics of software.	1	2/19/25	Diego Chiok, Edgar Legaspi, Curtis Hayes, Svetya Koppisetty
FR_PD_12	The system shall account for multiple file uploads.	2	2/19/25	Diego Chiok, Edgar Legaspi, Curtis Hayes, Svetya Koppisetty
FR_PD_13	The system should allow for a history feature with the file uploads.	3	2/19/25	Diego Chiok, Edgar Legaspi, Curtis Hayes, Svetya Koppisetty

2.3 Performance

Performance is an essential non-functional requirement for our system as it directly impacts the accuracy and functionality of waste detection. Performance is a key focus for our team, as we aim to ensure the system operates efficiently and effectively. The performance of the object detection model must be able accurately detect and identify most forms of recyclables while decreasing potential errors. The main objective of our application is to achieve a high detection rate, ensuring that recyclables and non-recyclables are not mislabeled. To ensure the prioritization of the system's performance, several functional requirements have been established: The system shall allow the user to have a scanning feature, the system shall detect all instances of litter that are within the scope of the scan, and the system shall detect all instances of recyclables.

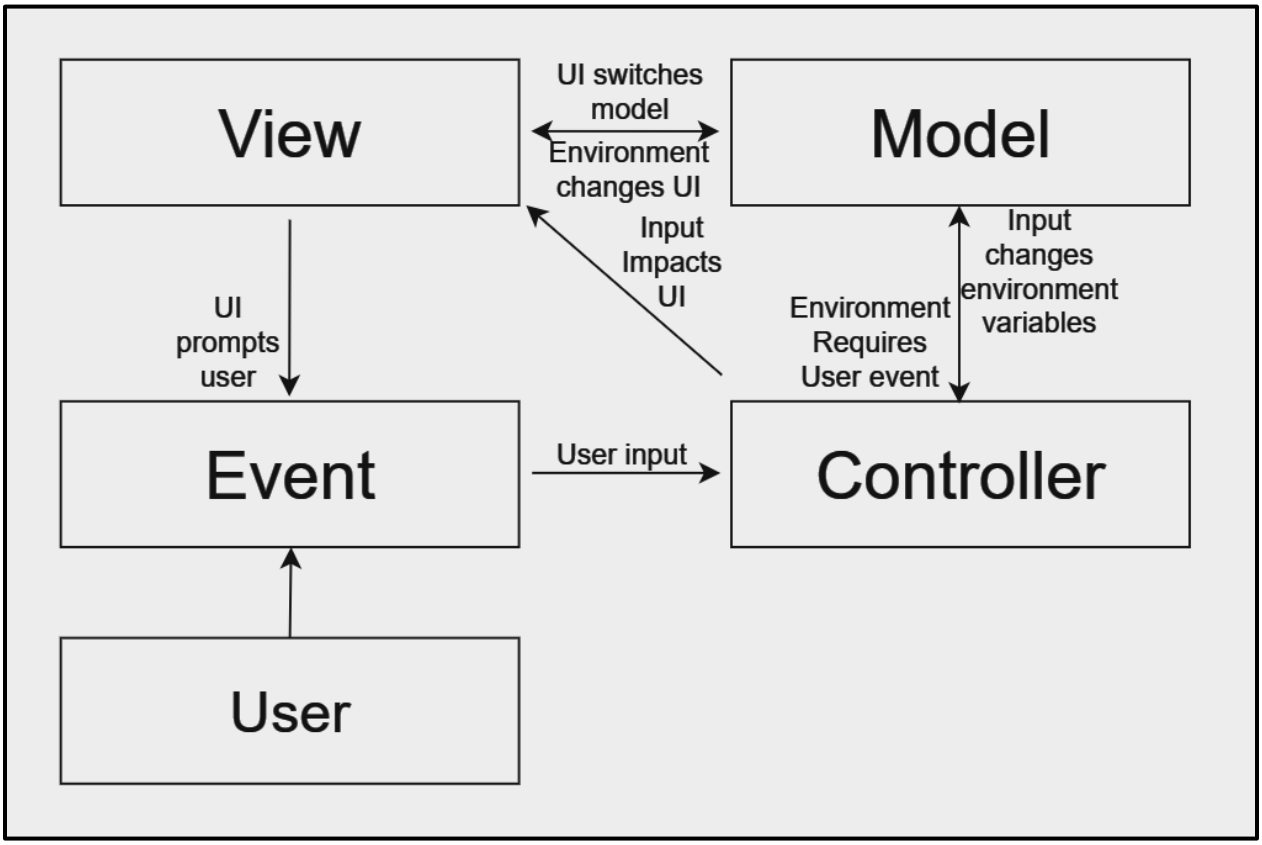
2.4 User Accessibility

User-friendliness is another necessary non-functional requirement for our system. Since the application is designed for diverse age groups and demographics, it must be straightforward and accessible. To achieve this, we aim to make the UI have universal qualities (i.e. green means start, red means stop) to ensure easy clear and seamless accessibility of all features the app has to

offer. This is evident in functional requirements: the system shall allow users to toggle any features, and the system shall work for Qualcomm devices.

3. Architectural Design

Figure 5:



3.1 Model Design

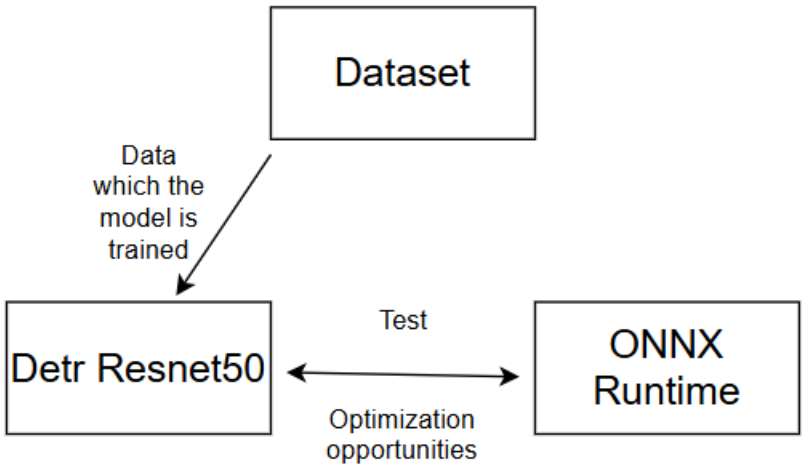
3.1.1 Purpose

The purpose of this module is where our deep learning model will run, along with any dependencies, environments variables or data required to run our model.

3.1.2 High-level Design

According to figure 6, the model module is broken down into lower-level modules, as in the DeTR ResNet50 model that will serve as the backbone of our application. The ONNX runtime would serve as a adapter layer, to allow python machine learning libraries to work with C++. The datasets are the data required to train and validate our model. This model is responsible for managing all the data driven components of our system.

Figure 6:



-

3.1.3 Required Interface

Interfaces that will be required for this module would be functions that'll allow users to upload their own data for the model to make inferences on. Another interface would allow users to toggle on or off the bounding boxes, accuracies.

3.1.4 Provided Interface.

This module has no provided interface.

3.2 View Design

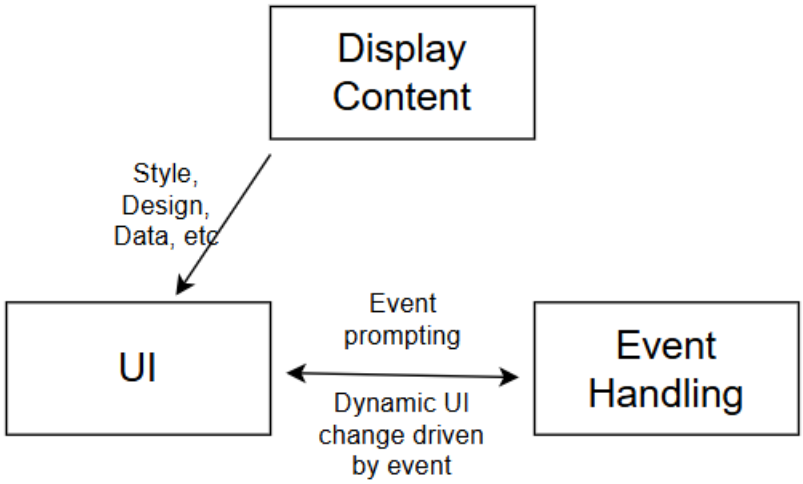
3.2.1 Purpose

This module provides real-time visualization of the model in action using OpenCV as the primary library for rendering computer vision output on the user interface. Since it relies on the model module, any modifications to the model will directly affect the displayed scene.

3.2.2 High-level Design

This module would use a computer vision library to display scenes using the camera in the computer. All rendering processes are managed within this module, while the processed data from the model module is displayed in the view module.

Figure 7:



3.2.3 Required Interface

The module provides a view-only canvas displaying the current scene. Any modifications to the view will be dictated by changes in other modules, ensuring seamless integration with the rest of the system.

3.2.4 Provided Interface

This module has features for adjusting the window's dimensions to improve the user experience and provide a clearer view of the canvas. It additionally includes a feature that tracks and shows the quantity of frames processed in a second, giving information about the system's performance in real time.

3.3 Controller Design

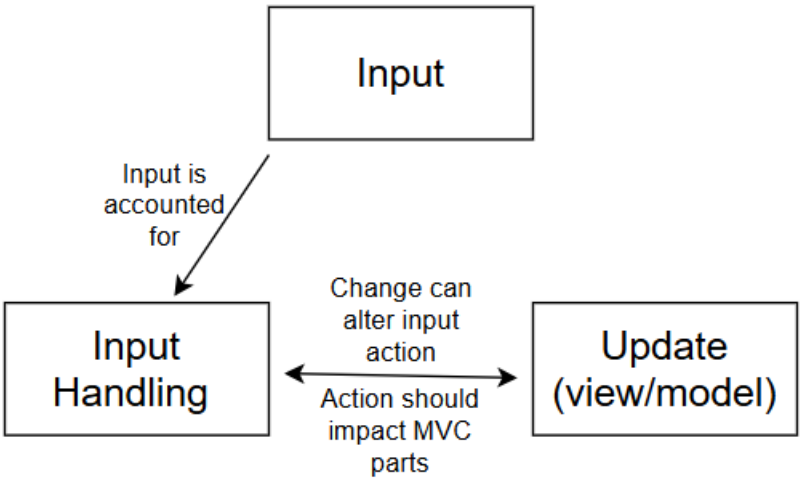
3.3.1 Purpose

This module will allow for the user to toggle between a live camera scan and a file upload. This module is dependent on the model module and the view module, since the controller must be able to adjust to changes and the controller must be accessible to the user.

3.3.2 High-level Design

This module will use an interactive framework to account for user events that may occur in the application. This module should allow for the user to toggle back and forth between different scanning features. In turn, these user events dynamically impact various aspects of the model and view modules.

Figure 8:



3.3.3 Required Interface

To operate the live scan feature and enable users to initiate and halt the scanning process as necessary, the controller module needs an interface. Additionally, it has a file upload function that lets users edit previously taken photos. This module offers an interface for tracking the scanning model's performance and allows input tweaks for personalized scanning preferences to improve user control.

3.3.4 Provided Interface

The module has a toggle that lets users alternate between live camera scanning and file uploads to improve usability. It also has a performance display toggle that allows users to turn on or off the scanning model's real-time performance metrics.

4. Implementation Plan

4.1 Implementation Platform and Frameworks

- Visual Studio
- Docker
- Pytorch
- ONNX runtime
- OpenCV

4.2 Implementation Tasks

- Import pre-trained model
- Make the Windows application
- Implement our model with OpenCV
- Implement functions to enable file uploads for model inference.
- Develop controller user interfaces that’ll connect with model and view modules

4.3 Implementation Schedule

The following is an example of the schedule table.

Category	Development Activities	Outcomes	Timeline	Personnel
Model	Run, compile and preform inferences on our deep learning model (DeTr ResNet50)	To confirm our model performs as expected.	3/3 – 3/6	Diego Chiok, Edgar Legaspi, Curtis Hayes, Svetya Koppisetty
UI	Develop the UI and make the functionality	To make sure user can interact with our program	3/10 - 3/12	Diego Chiok, Edgar Legaspi, Curtis Hayes, Svetya Koppisetty
Window App	Develop the main window application our model will run using DWM Api for the window framework	Have a basic windows application	3/13-3/27	Diego Chiok, Edgar Legaspi, Curtis Hayes, Svetya Koppisetty
Database	Set up a database for our system to hold files.	Make sure database can store files	4/28 - 4/30	Diego Chiok, Edgar Legaspi, Curtis Hayes, Svetya Koppisetty

References

1. Model (DeTr ResNet50): https://github.com/wimlds-trojmiasto/detect-waste