LocalHost

Services Inc.

Design specification

LocalHost

Partners:
Angus Kan
Irene Leung
Kevin Cao
Patrick Cong
Rico Chao
Yoel Yonata

Contact:
Kevin Cao
hca119@sfu.ca

Submitted To:
Dr. Mike Hegedus, ENSC 405W
School of Engineering Science
Simon Fraser University

Issue Date:
March 13, 2022

List of Figures

List of Tables

Table of Content

1. Introduction

1.1 Background

Restaurants require a large number of staff to provide a quality dining experience for customers. However, it is hard for them to maintain this great level of quality and efficiency during peak dining hours. With an influx of customers during those hours, waiters and waitresses may have their attention split between many tables, resulting in missed details and unfulfilled requests. This puts extra stress on the waiters and waitresses, as well as an extra long wait for the customers.

1.2 Solution

The purpose of this project is to provide a solution to efficiently manage each table by using both hardware and software components to track the table statuses and respond to customers' inquiries. For the hardware component, we will use a camera module to capture live video of each table and transfer data to the software system. The software system will process the data to identify the table statuses, then notify the waiters and waitresses. More details about the design of both hardware and software components will be discussed in this document.

1.3 Challenges

There are few challenges we encountered or may encounter during the development of our project. These challenges are shown in Table 1.3 below.

Challenges	**Solutions**
Image processing result is not accurate.	Try to use different threshold numbers to see different results or use different models to do image detection.
Image quality from the camera is not clear.	Try to test it in different light conditions and find the optimal brightness for image quality. Try to test the camera at different distances to find an optimal distance for image detection
Camera WIFI connection is not stable for data transmission.	Use wired or bluetooth connection.

Table 1.3: Development challenges

1.4 Design Classification

This document will encode different stages of the development using the labelling in **Table 1.4** below.

Encoding	Development Stages
C	Conceptual Stage of Proof of Concept
P	Prototype

Table 1.4: Design Classification Encoding

2. System Overview

LocalHost Services is a restaurant management system that provides restaurant waiting staff with real-time information of the current table statuses in the restaurant. LocalHost has two major components: a hockey-puck shaped device that will be mounted by each table and a restaurant management software to help monitor the table statuses in the restaurant.

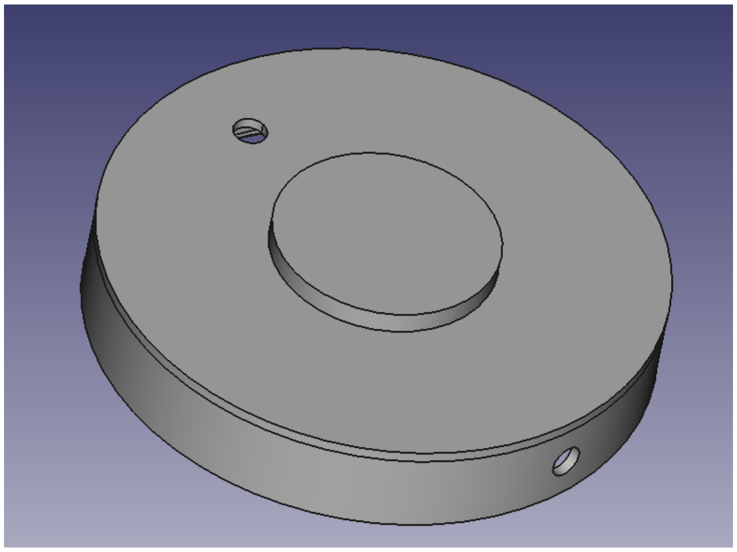

Figure 2.1 3D model of hockey puck shaped device.

The hockey-puck shaped device will contain a camera module attached to a microcontroller that will capture a video stream of each table. This video will then be processed to determine each table's status (whether the table is available, occupied, needs to be cleaned, etc.). The device will also have a button in the middle in case the customers need to buzz and notify the waiter. Inside the device there will also be a microcontroller connected to the camera, for the microcontroller, we will be using the ESP-32 CAM module. There will also be a ring of LED lights inside the device to show the table status on the device. This LED ring can also provide visual feedback to the customer by turning into flashing red colour if the buzzer is pressed in order to indicate that the table needs attention from the waiter. For power, the device will contain a barrel jack. This serves as an easy way to connect a power adapter that will supply the 5 volts needed to power this device. There will also be a switch showing on the outside and connected to the plug in case the device needs to be switched off, but still plugged in.

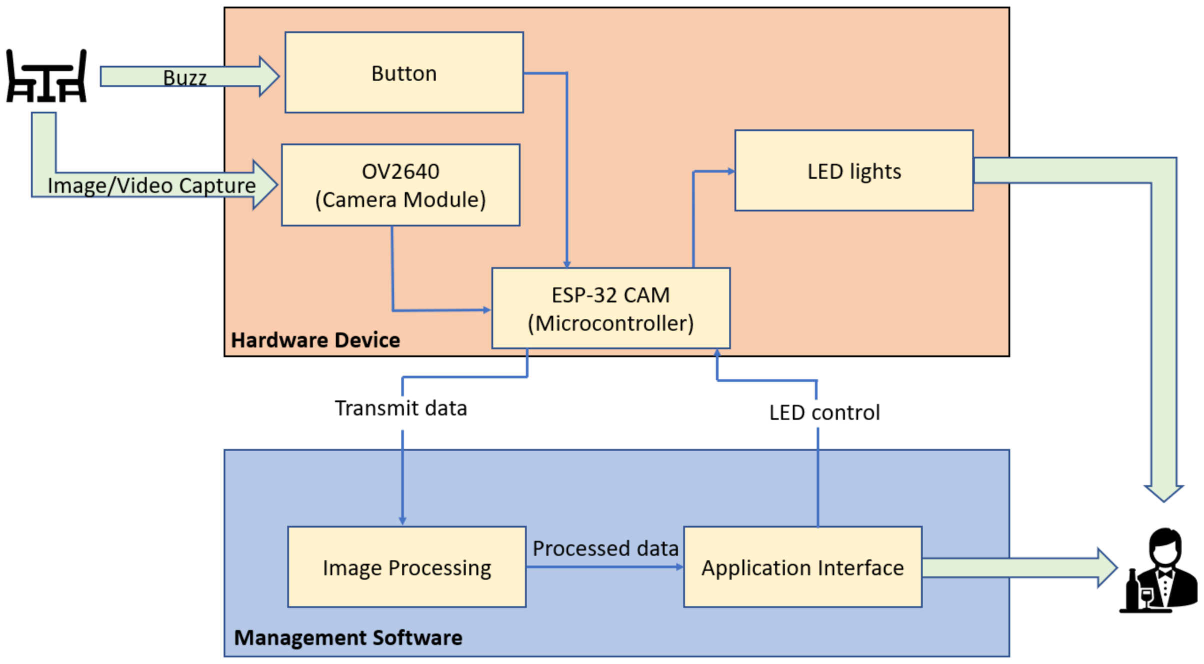

Figure 2.2 Flowchart of LocalHost's system

The other main component to our system is the restaurant management software. Our software will be a desktop application. The software will receive a video data stream coming from the mounted device and perform image and video processing. The software will take the data extracted from the video stream and display table statuses on the interface to provide information for the waiters. The software will also contain a layout editor feature that allows restaurant staff to create and modify the layout of the tables at the restaurant.

Design ID	Design Specification Requirement
[Des 2.1-C]	The image processing software must be able to differentiate different table states.
[Des 2.2-C]	The device must be able to communicate with the software management system and vice versa via Wi-Fi.
[Des 2.3-P]	The device must have an ON/OFF switch for power.
[Des 2.4-P]	The device will contain LEDs to provide visual feedback based on table statuses.

[Des 2.5-P]	The device will contain a button to buzz the waiter.
[Des 2.6-P]	The management software will be able to replicate the restaurant layout.
[Des 2.7-P]	Each device will cost under $100.

Table 2.1: Design Specification Requirement

3. Electrical Design

3.1 Microcontroller

At the center of LocalHost is the ESP32 Cam module by AI-Thinker with 520KB of SRAM and external 4M PSRAM. This microcontroller utilizes the Espressif ESP32 controller chip, which features bluetooth, Wi-Fi, and BLE (Bluetooth Low Energy) capabilities. The ESP32 chip is commonly used for IoT applications, ranging from smart devices for households to industrial wireless control. For wireless applications via Wi-Fi or Bluetooth, the ESP32 is the cheap and ideal solution.

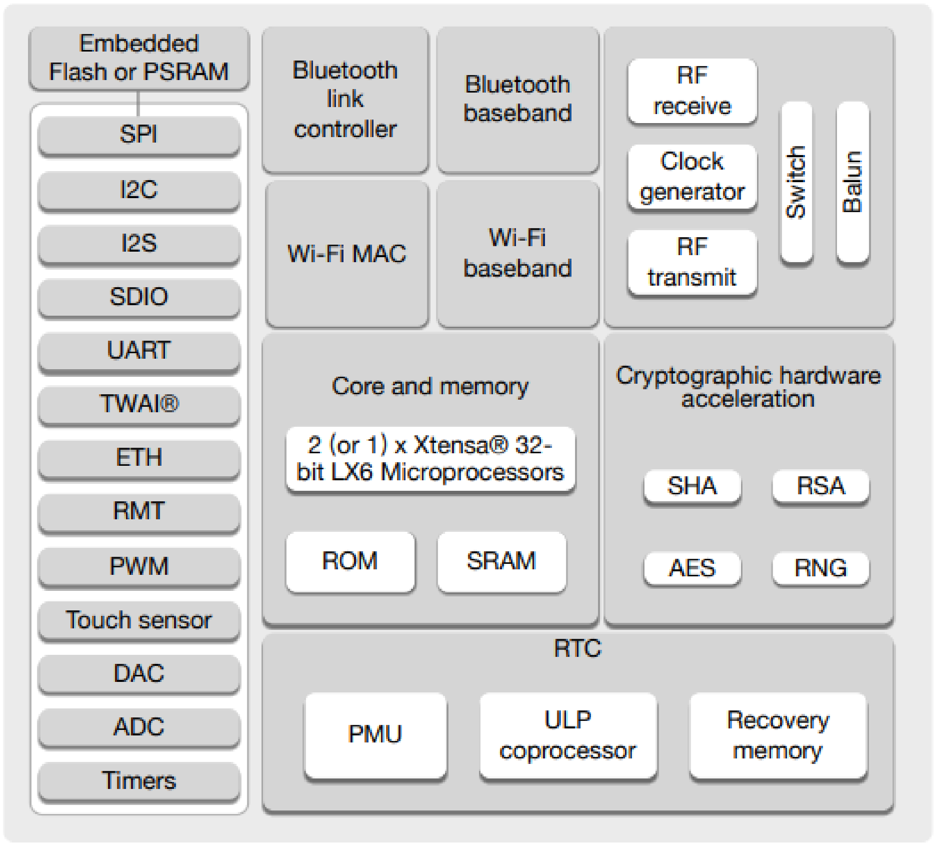

Figure 3.1.1 Diagram of the ESP32 Chip

The ESP32 Cam module is one of the smallest ESP32 boards on the market, with only 10 GPIO pins, as well as a built-in slot on the top of the board for the camera to connect. With the Wi-Fi capabilities of this board, LocalHost will stream video from the camera to the IP address of the local Wi-Fi. We will utilize our GPIO pins to connect our tactile button, as well as using 4 GPIO pins to output the desired colours to our LED strips[1].

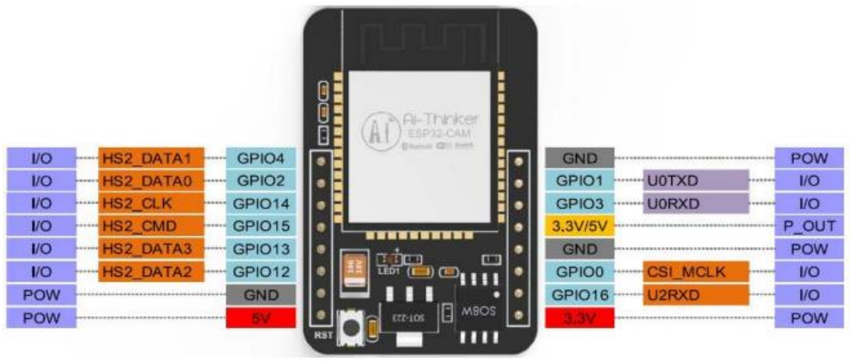

Figure 3.1.2 ESP32 Cam Module Pinout[1]

Design ID	Design Specification Requirement
[Des 3.1.1-C]	Includes a Camera Module interface
[Des 3.1.2-C]	Processor includes a Wi-Fi Module
[Des 3.1.3-C]	Contains enough GPIO pins for buttons and a LED indicator
[Des 3.1.4-P]	Must fit inside the LocalHost case

Table 3.1: Microcontroller Design Requirements

3.2 Camera Module

The camera module that is included with the ESP32 Cam module is the OV2640. Although this camera does not provide the clearest image or have the largest megapixels, the OV2640 will be sufficient as a camera sensor for image processing. Combining its cheap price point with its tiny form factor, this is the ideal camera module for LocalHost.

Design ID	Design Specification Requirement
[Des 3.2.1-C]	Interface with ESP32

Table 3.2: Camera Module Design Requirements

Figure 3.2.1 OV2640 Camera Module

3.3 FTDI USB To Serial

Due to the small form factor of the ESP32 Cam module, there is no built-in micro USB jack, so an FTDI rl232 is required in order to program the microcontroller. The FTDI is used to convert USB to serial UART so the Arduino IDE is able to upload the program to the ESP32.

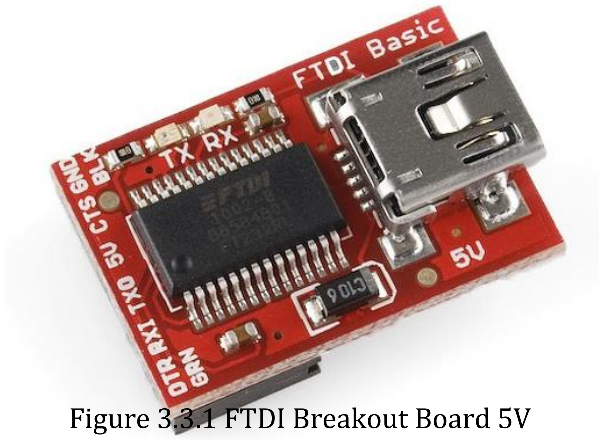

Figure 3.3.1 FTDI Breakout Board 5V

Design ID	Design Specification Requirement
[Des 3.3.1-P]	Converts USB to serial UART
[Des 3.3.2-P]	Outputs to 6 pins
[Des 3.3.3-P]	Outputs 5V

Table 3.3: FTDI Design Requirements

3.4 Power Supply

The power supply chosen for LocalHost will be a 5V power adapter, with amount of amperes based on how many camera modules the user would like to power at a time. The power consumption of just the camera with no flash is 180mA at 5V, while the power consumption of the ESP32 module with Wi-Fi turned on is 90mA at 5V. The current draw is around 270mA-300mA, so having our power adapter supply 500mA per module will be sufficient. With multiple modules, we will provide a larger power adapter.

Design ID	Design Specification Requirement
[Des 3.4.1-P]	Regulated DC voltage output
[Des 3.4.2-P]	Daisy Chained to multiple modules
[Des 3.4.3-P]	Can disconnect from module

Table 3.4.1 Power Supply Design Requirements

Figure 3.4.1 5V Power Adapter

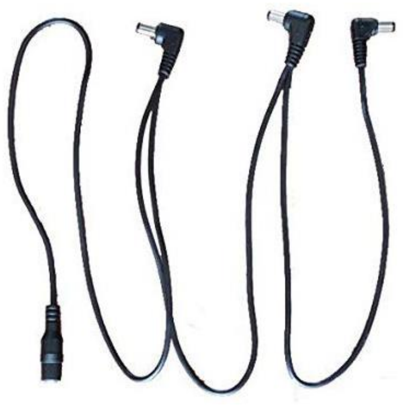

Figure 3.4.2 Cable for Daisy Chain

4. Software Design

The components of LocalHost's management software includes:

1. Controlling the hardware devices

2. Performing image comparison by comparing the captured photos to the photos of clean tables using openCV
3. Performing image processing to measure the current water levels in the cups using openCV

4.1.1 Hardware Control

Design ID	Design specification requirements
[Des 4.1.1-C]	The software will be able to connect to the hardwares using Wi-Fi.
[Des 4.1.2-C]	The software will be able to calibrate the device.
[Des 4.1.3-C]	The software will be able to wake up, turn off and reset hardware devices.
[Des 4.1.4-C]	The software will be able to change the colors of the LEDs on the hardware devices.
[Des 4.1.5-C]	The software will be able to display the live stream from a certain device upon request.
[Des 4.1.6-C]	The software will be able to change the mode of hardware devices. (performance/ normal/ energy saving)

Table 4.1.1 - Design Requirements for Hardware Control

In order to connect the hardware devices to the management software, the product will use Wi-Fi connection as default setting; if Wi-Fi is unstable or unavailable, the product will use Bluetooth connection. When the hardware devices are turned on for the first time or after being reset, the management software will have a pop up window to recalibrate the device with the corresponding table number, as well as the camera settings. In case of incorrect table status detection, the software can overwrite the current status of the table manually and change the color of the LEDs.

4.2 Image Compression

Design ID	Design specification requirements

[Des 4.2.1-C]	The software shall use the Structural Similarity Index to determine how similar the two photos are
[Des 4.2.2-C]	The software will use grayscale to convert images to black and white, to reduce image size

Table 4.2.1 - Design requirements of Image comparison

The SSIM from scikit-image is is a method for predicting the perceived quality of digital television and cinematic pictures, it is a perception-based model that considers image degradation as perceived change in structural information, while also incorporating important perceptual phenomena, including both luminance masking and contrast masking terms[2]. can give us a feedback score based on the similarity which ranges from -1 to 1, with -1 meaning the table is completely dirty and 1 meaning the table is perfectly clean.

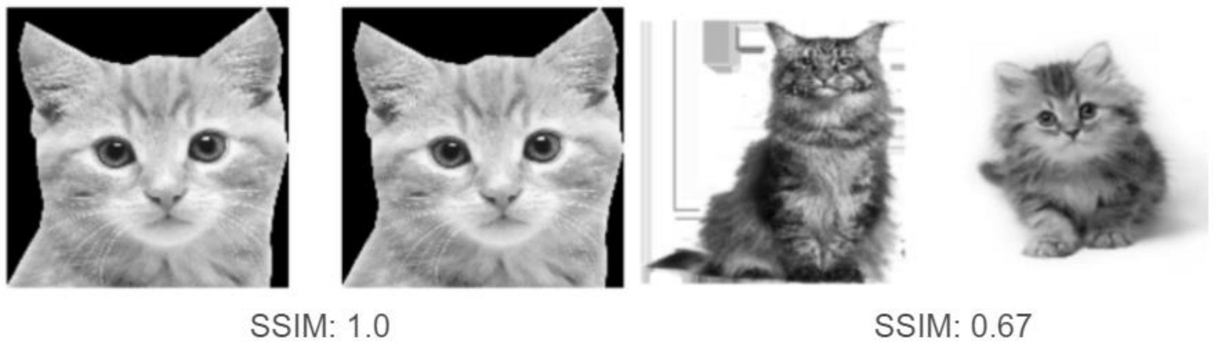

Figure 4.2.1: SSIM score of two same images and two different images[3]

The SSIM score is calculated with the formula below[2]:

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

Grayscale reduces the image sizes by assigning the same number to all three colors in a pixel instead of three different numbers,the value of each pixel only represents the intensity information of the light, thus theoretically reducing the image size by 66.66%, and leads to faster processing time.

4.3 Water Level Recognition

Design ID	Design specification requirements

[Des 4.3.1-C]	The software will use grayscale to convert images to black and white, to reduce image size
[Des 4.3.2-C]	The software shall be able to recognize all the cups in the photo using openCV
[Des 4.3.3-C]	The software shall be able to measure the percentage of the remaining water in the cup
[Des 4.3.4-C]	The software shall send notifications to the user if the water level of any cup on the table is below 20%

Table 4.3.1 Design requirement of water level recognition

The object detection will be done by using openCV and machine learning. The software shall first detect all the cups within its view, and then calculate the height difference between the bottom of the cup and the current water level.

For object detection, the software shall have a well trained AI model which will be fed with a large number of labeled photos of cups as positive images, and photos of objects with similar shapes as negative images. For measuring height difference, the software shall see the part of the cup which is filled with water as an object, and measure its height using openCV's imutiles library to measure the height of this object based on the pre-calibrated pixels per metric ratio.

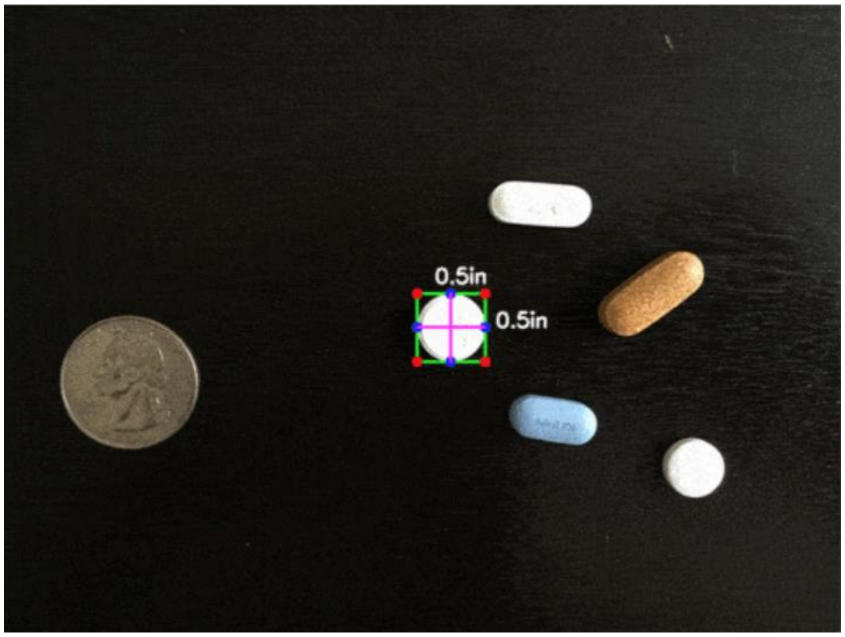

Figure 4.3.1: Measuring the size of the object using OpenCV[4]

4.4 Management Software

For our management software, it will be implemented as a Windows desktop application, with Electron.js. The management software will act as the application to receive feedback from the microcontroller and camera data, analyze it, and change the status of the tables, as well as update the LED lights on the device.

4.4.1 Front-End

The application will feature a login screen for users to log in, and once the user logs in, the user will be brought to the "Home" page, where an overview layout of the restaurant will be displayed, as shown in Figure 4.4.1, with the tables displaying different colours to indicate their corresponding statuses. There will be a sidebar, as shown in Figure 4.4.2, displaying all the different pages of the application such as "Home", "Reservation", "Layout Editor", and "Settings". The "Reservation" page gives the user an overview of the reservation details for the restaurant, and allows the user to make changes to reservations. The "Layout Editor" page lets the user change the overview layout of the restaurant such as adding another table or changing the position of a table. Updates from the system will be reflected on the application, either in the form of a notification pop up, as shown in Figure 4.4.3, or colour change of the table in the "Home" page.

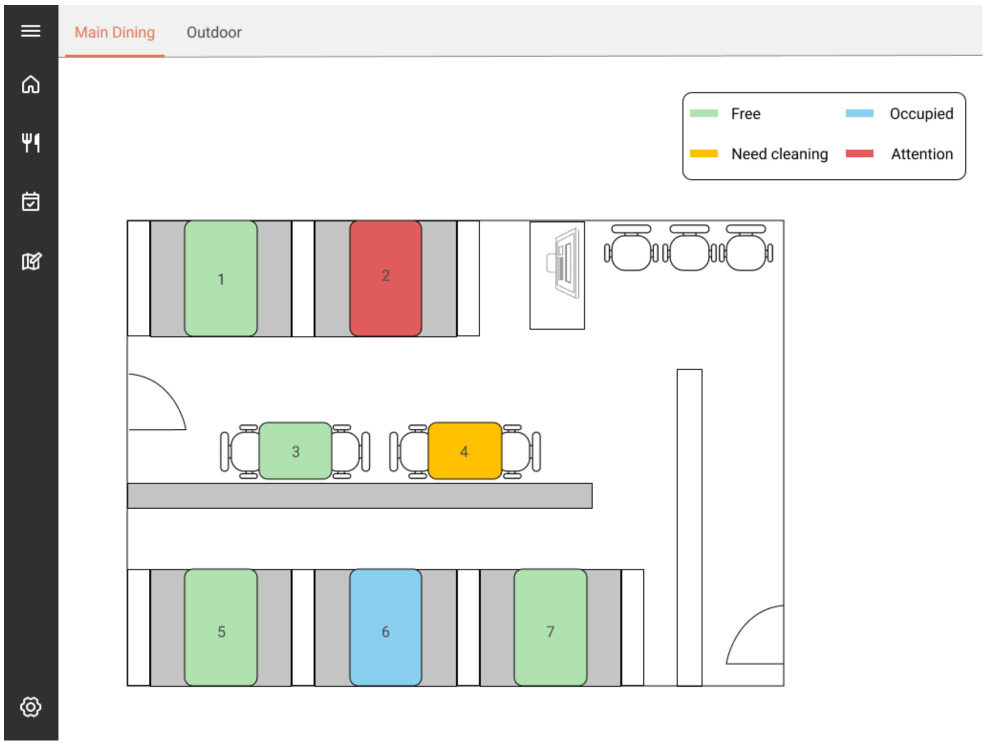

Figure 4.4.1: Screenshot of the “Home” page of the management application

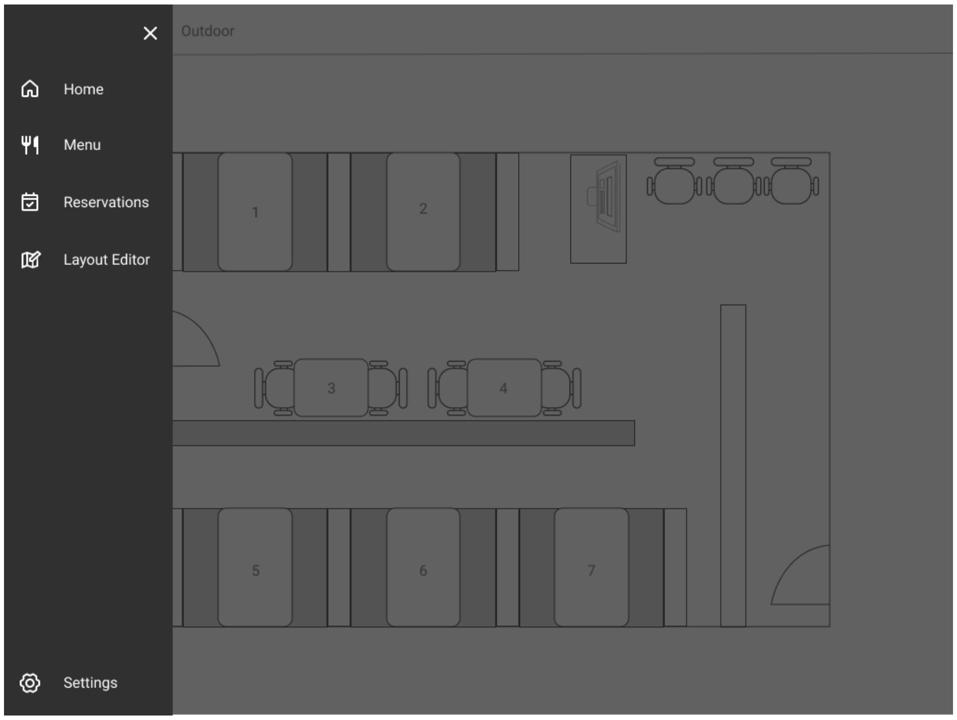

Figure 4.4.2: Screenshot of the sidebar of the management application

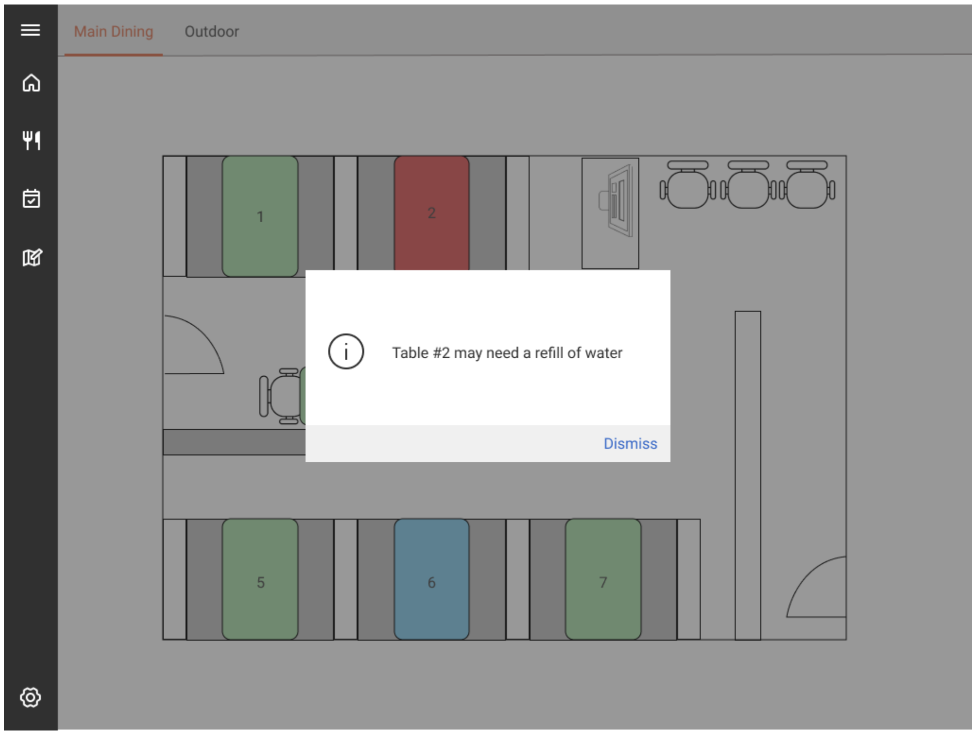

Table #2 may need a refill of water

Figure 4.4.3: Screenshot of the notification pop-up of the management application

4.4.2 Back-End

Back-end design of our application is split into two sections, with one section managing the general communication with the front-end of the application, as well as login system communication, and the other section responsible for data processing of the feedback collected by the microcontroller and the camera.

Data received from the microcontroller and camera will be sent to the back-end for analysis and processing. After processing the data, the back-end will provide the corresponding responses to the front-end of the application and the devices, updating the colour of the table in the layout view and the LED lights of the hockey-puck device respectively.

5. Conclusion

LocalHost is a user-friendly system which aims to help restaurants to improve their customer's dining experience, increase profit by improving the table turnover rate and reduce the stress of their workers. The device is both small and easy to install, all while providing real time table status using AI image processing from the software, which is based on the ESP32-CAM.

The Electronic Design

1. Microcontroller: An ESP32 will stream video via Wi-Fi as well as provide GPIO pins for buttons and LEDs
2. Camera Module: Captures video of the dining table
3. FTDI USB to Serial: Converts USB to UART so the user can upload code to the microcontroller
4. Power Supply: Provides power for the module

The Software system:

1. Image comparison: Reduce the size of the images, then use the Structural Similarity Index to determine the level of similarity
2. Object Recognition: Reduce the size of the images, and using openCV to identify the objects on the table, primarily targeting the cups

3. Water level recognition: Reduce the size of the images, using openCV to measure the percentage of the remaining water in the cups
4. Management Software: Application that will communicate with the microcontroller and process the data received by it

This Design Specification document will work as a guideline for LocalHost to develop the product. However, it may change as we encounter problems and try to find other solutions to fix them during development.

6. References

[1] Fedecastellaro, "Fedecastellaro/ESP32-S3-symbol-footprint: ESP32-S3 symbol and footprint for Altium designer. data taken from https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf," *GitHub*. [Online]. Available: https://github.com/fedecastellaro/ESP32-S3-SYMBOL-FOOTPRINT. [Accessed: 13-Mar-2022].

[2] "Structural similarity," *Wikipedia*, 24-Feb-2022. [Online]. Available: https://en.wikipedia.org/wiki/Structural_similarity. [Accessed: 13-Mar-2022].

[3]I. Mamun, "Image classification using SSIM," *Medium*, 12-Feb-2019. [Online]. Available: https://towardsdatascience.com/image-classification-using-ssim-34e549ec6e12. [Accessed: 13-Mar-2022].

[4] Linus, A. Rosebrock Javier, "Image difference with opencv and python," *PyImageSearch*, 07-Jul-2021. [Online]. Available: https://pyimagesearch.com/2017/06/19/image-difference-with-opencv-and-python/. [Accessed: 13-Mar-2022].

7. Appendix: Test Plan

7.1 Introduction

This section will outline the specific test plan for the whole system of LocalHost.

7.2 User Testing

Test Name	System turns on	Date	
Test purpose	Plug in the cord to the device		
Expected Behavior	Device should be automatically turned on; LEDs should light up to indicated device is online		
Actual Behavior			

Test Name	System pairing	Date	
Test Procedure	Turn on all devices and connect to the application on the user computer.		
Expected Behavior	All devices should automatically be available to connect with the application through WiFi.The following tests will be performed on LocalHost device and user interface for testing device’s functionality and user interface of the application		
Actual Behavior			

Test Name	Generate a virtual floor plan for the restaurant	Date	
Test Procedure	The first-time device is paired to the application, application should prompt user to create a virtual floor plan for the new device		
Expected Behavior	After devices connected to the application, user should be able to create a virtual floor plan for the restaurant		
Actual Behavior			

Test Name	Buzzer detection	Date	
Test Procedure	Press the button to buzz the waiters		

Expected Behavior	The device will communicate with the application to trigger notification to the waiters
Actual Behavior	

7.3 Software Testing

The following tests will be performed on the software of the LocalHost device. These tests will test all the device's software features and its subsystems.

Test Name	Buzzer LEDs detection	Date	
Test Procedure	Press the buzzer button on the device		
Expected Behavior	The device should light up the LEDs		
Actual Behavior			

Test Name	Water level detection	Date	
Test Procedure	Detect the changes of water level.		
Expected Behavior	The software should be able to detect the water level and send notification if the water level is too low.		
Actual Behavior			

Test Name	Notification test	Date	
Test Procedure	Press the button to buzz the waiters		
Expected Behavior	The application sends notifications with sound and pop up messages at the corresponding GUI.		
Actual Behavior			

Test Name	Clearing status	Date	
Test Procedure	Detecting the difference between plates is finished or not.		
Expected Behavior	The software should be able to detect the plate has been empty and send a notification for table cleaning.		
Actual Behavior			

7.4 Mechanical Testing

The following tests will be performed on the device as a whole, each test will verify a part of device for specific mechanical purpose

Test Name	Device Installation	Date	
Test Procedure	Install devices on the wall beside the table.		
Expected Behavior	Device should be easy to install and will secure on the wall mount		
Actual Behavior			

Test Name	Device drop test	Date	
Test Procedure	Drop the device from 1.5 meters above the ground		
Expected Behavior	Devices should not be damaged on its enclosure, electronic and camera should remain intact and functional.		
Actual Behavior			

Test Name	Device water resistant test	Date	
Test Procedure	Expose the device to humid environment, and spill small amount of water on it while it is mounted to the wall		

Expected Behavior	Device should not be damaged from the small exposure to water, its electronic should remain functional, and will not create short circuit and/or electrical leakage
Actual Behavior	