Software Formalization

Year: 2021 **Semester:** Spring **Team:** 4 **Project:** Virtual Queue
Creation Date: March 3rd **Last Modified:** March 4, 2021
Author: Devansh Rathi **Email:** drathi@purdue.edu

Assignment Evaluation:

Item	Score (0-5)	Weight	Points	Notes
Assignment-Specific Items				
Third Party Software		x2		
Description of Components		X3		
Testing Plan		x3		
Software Component Diagram		x4		
Writing-Specific Items				
Spelling and Grammar		x2		
Formatting and Citations		x1		
Figures and Graphs		x2		
Technical Writing Style		x3		
Total Score				

5: Excellent 4: Good 3: Acceptable 2: Poor 1: Very Poor 0: Not attempted

General Comments:

1.0 Utilization of Third Party Software

For this project most of the software on the MCU will be developed by our team for this project. However, a few node packages were used for creating the web server.

These are the following node packages that were used:

Name	License	Description	Use

Http	MIT	It is a very simple package that helps with managing incoming and outgoing HTTP requests.	It is used by the package Express to manage requests to the webserver and by Socket.io to do communication using sockets.
Express	MIT	This framework provides an abstraction over the http module and provides helpful and easy to use API to manage the web server.	It is used to manage the requests coming from the client's device and the MCU.
Socket.io	MIT	Socket.io provides bi-direction communication between the client and the server.	It is used to communicate from the server to the client's device to update the queue information and indicate when the barcode was read.
Crypto	MIT	It is a collection of standard cryptography algorithms.	The sha1 algorithm provided by this module is used to generate the ID of the customer
Web push	MIT	Web-push can send notifications to the client's device anytime they have the browser open.	This is used to notify the customer when it's their turn to enter the store.
QRCode	MIT	Created by davidshimjs, it can generate a QR-code for a given input string.	This is used to generate a QR-code using the customer's ID to be scanned by the QR-Code scanner at the MCU.

Table1: Third party software used in the webserver

2.0 Description of Software Components

The project is divided into two main software components: MCU and the webserver. The webserver does queue management and communicates with the customer device, whereas the MCU processes the information coming from the hardware components and sends this information to the web server. The sub-components of these two are as follows:

Web Server:

- **Queue management:**
  The web server keeps a track of the queue. When customers join, enter the store, or leave it, it is updated and the function to notify the members of the queue as needed is executed.

- **Potential queue management:**
  Intermediary queue where the customer data and the ID is stored between the customer having generated the QR-code with their ID to scan till the customer scans the QR-code and is moved to the main queue. The ID before being scanned has a time to live of 10 minutes after which the ID and customer details are deleted from the server. They can generate a new ID.

- **Store management:**
  The web server keeps the track of the number of people in the store. When there is a request to enter the store by scanning a QR-code at the MCU, it is checked if the number of people in the party in the customer's party can enter the store. If they can, the server will notify the customer and return the number of people expected to enter the store to the MCU.

- **Customer endpoints**:
  These are the endpoints in the web API that the customer's device will call. These can be accessed by anyone so they do not have any verification. These endpoints are used to serve web pages to the customer and for the customer to send their data to the web server. The webpages serve as a way for the customer to enter their information, view the information provided by the server for the customer, and generate the QR-code that is meant to be scanned at the MCU. The data sent to the server is used by the server to create the customer ID, remove the customer from the queue if the customer indicates to do so, and subscribe to the notifications that the server can push on the customer device.

- **MCU endpoints:**
  These are the endpoints in the web API that the MCU will call. Since only the MCU is supposed to be able to call these endpoints, each request has a security layer in the form of a 'storeSecret' that only the server and MCU should know. Every request contains this as a URL parameter, which the MCU checks to confirm the identity. These endpoints are used for the MCU to communicate information to the web server when a QR-code is scanned and people physically enter or leave the store.

- **Sockets:**
  This is used for all the messages that have to be sent from the server to the customer's device. This allows us to avoid continuous polling from the customer's device to the server to get the updated information about their spot in the queue and to get information whether the barcode on the device was successfully scanned by the MCU or not.

MCU:

- **Communicating to the web server:**
  The MCU will connect to the internet through the ESP-01 WiFi module, which utilizes an ESP8266 chip. The module communicates to the MCU via UART, which the MCU will handle using DMA and the related interrupts for message reception and transmission. The messages used to communicate with the module are formatted as AT commands, which will be used to send the HTTP GET requests to the web server. The MCU will send out different requests based on certain events. If the QR scanner triggers an interrupt, the MCU will transmit the QR data to the web server. If the thermopile's logic code detects an invalid temperature or the PIR sensor logic determines there has been an invalid entry, it will send a notification to the server that there is a need for employee intervention. Finally, if an exit is detected by the PIR sensor logic, a notification that a customer has left will also be sent to the web server. All these send requests can be handled by the same function sendMessageToServer, which will take in a character array representing the data to be sent, convert it to the proper AT command format and send that message via UART to the module. Once the web server sends a response, an interrupt will be triggered and the message will be handled by parseServerResponse, which will call the appropriate function, such as updating the display, initiating a temperature scan, or allowing a certain number of customers to enter validly.

- **Reading the QR-code:**
  The QR scanner communicates back to the MCU much like a keyboard, sending the text content of the QR code when it is scanned. A UART channel will be linked to the QR scanner, and when an interrupt is received indicating a QR code has been scanned, the QR contents will be sent to the sendMessageToServer function formatted into an AT command containing the proper JSON.

- **Detecting enter and exit:**
  - The PIR sensors used to detect entering and exiting will be using I2C interface
  - Temperature data will be read using I2C1 from the MCU by calling readRegister16(byte location)
    - IR Sensor 1 is at location 0x06,
    - IR Sensor 2 is at location 0x08,
    - IR Sensor 3 is at location 0x0A,
    - IR Sensor 4 is at location 0x0C,
  - Taking the differences from the sensor for example Temperature IR4 - Temperature IR2 if the value is > 0 then it means the customer is passing from IR4 to IR2, conversely if value is < 0 then it means the customer is passing from IR2 to IR4

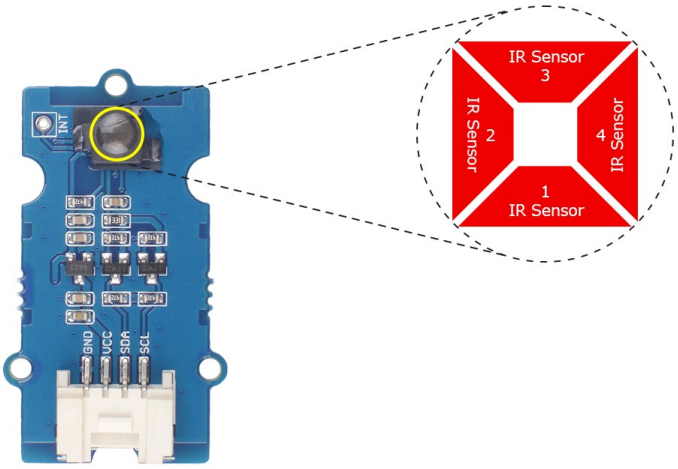


Figure 1: PIR sensor

- **Reading from thermometer:**
  The code to take a customer's temperature will be handled by a single function, takeTemperature(). This function will be called by parseServerResponse if the customer who scanned their barcode was the next customer in the queue, so it is now their turn to enter. This function will read values from the ADC and perform the appropriate calibration to convert it to fahrenheit. This value will then be compared to the predetermined valid range. If the customer's temperature is valid, the PIR logic will be notified to accept the next entry. If it is invalid, sendMessageToServer will be called with a message notifying the server that employee assistance is needed.

- **Writing to the display:**
  An LCD screen will be utilized to display information such as current capacity levels, customers who will soon be eligible to enter the place of business, and directions for entering the building. Four-line SPI protocol will be used to communicate with a display driver (RA8875 Driver Board) that will clock and write to the LCD screen.

3.0 Testing Plan

The testing for the web server is relatively easier as a lot of sophisticated tools that come with using NodeJS and javascript can be used without a lot of setup. However, the MCU is more likely to have bugs since there is only access to basic tools in C. A way for the MCU to communicate over UART to print statements on the computer's terminal was set up to help debug.

The testing plan for the web server is as follows:

- **Queue and store management:**

A test bench is created that simulated people joining the queue, leaving the queue, entering the store, and leaving the store. Many different types of situations are created virtually, and desired output is given for every test case. This can be run using ‘npm test’.

- **Potential queue management:**
  Since the potential queue is just a temporary place to store the customer data and the ID until they have scanned the barcode, it is relatively straightforward. The time to live of the ID in the potential queue is tested by varying the time to live and logging out the ID just before it is destroyed. Adding and removing customers from this is tested out by simulating customers joining and leaving the potential queue and logging out the queue after every change.

- **Customer endpoints:**
  The GET endpoints are fairly straightforward since it only used to serve web pages. Only the webpage being loaded needs to be checked when a request is made.
  The POST endpoints are checked by simulating the whole workflow by running this as a local server and using it as it is intended to be used by the customers.

- **MCU endpoints:**
  The MCU requests can be simulated by just opening another tab and sending the get requests the MCU is supposed to send with the corresponding ‘storeSecret’. Using this method, the workflow will be simulated as it is intended to be used.

- **Sockets:**
  Most of the MCU endpoints send a socket message to a particular customer or customers through sockets. So a subset of the simulation used for MCU endpoints can be used and the socket communication can be observed on the client-side console by logging.

The testing plan for the MCU is as follows:

- **Communicating to the web server:**
  A simple web server at ‘ptsv2.com’ that sends back simple responses to the MCU’s requests was used. The request made by the MCU can be logged out by the web server and observed there. After the ESP8266 chip is connected to the WiFi (the code for which is readily available) the function is fairly straightforward. The get request is a string will be sent to the ESP8266. The ESP8266 uses ‘AT’ commands so the output string will be formatted accordingly. The response of the web server will be parsed and printed out on the computer’s terminal to check if it matches.

- **Writing to the display:**

An array of different strings will be hard coded in the MCU and every time a button is pressed the MCU should send the next string. The string displayed can be compared with the string that was sent to test writing to the display.

- **Reading the QR-code:**
  Since the web server is already set up to be able to generate QR-codes. The page where the QR-codes are generated will be opened on a device and the QR-code will be read by the MCU and printed into the computer's terminal. This string should match the string on the console of the page.

- **Detecting enter and exit:**
  The PIR sensor will be placed in a doorway. We will set up the MCU such that a LED is lit when motion in one direction takes place and another LED is lit in the opposite direction. Then it will be checked if the correct LED lights up when people walk through the doorway in various ways.

- **Reading the thermometer:**
  The thermometer will be pointed at something and the temperature will be printed onto the computer's terminal. The output temperature will be compared to an actual thermometer. The outputs should match if it is working correctly.

4.0 Sources Cited:

[1] Waveshare. *Barcode Scanner Module User Manual* [Online]. Available: https://www.waveshare.com/w/upload/d/dd/Barcode_Scanner_Module_Setting_Manual_EN.pdf

[2] Espressif. *ESP8266 Datasheet* [Online]. Available: 0a-esp8266ex_datasheet_en (espressif.com)

[3] Adafruit. *7.0" 40-pin TFT Display* [Online]. Available: 7.0 40-pin TFT Display - 800x480 without Touchscreen ID: 2353 - $37.50 : Adafruit Industries, Unique & fun DIY electronics and kits

[4] STMicroelectronics. *STM32L4: PWR* [Online]. Available: https://www.st.com/content/ccc/resource/training/technical/product_training/ce/57/a3/86/7a/3d/4d/87/STM32L4_System_Power.pdf/files/STM32L4_System_Power.pdf/jcr:content/translations/en.STM32L4_System_Power.pdf

Appendix 1: Software Component Diagram

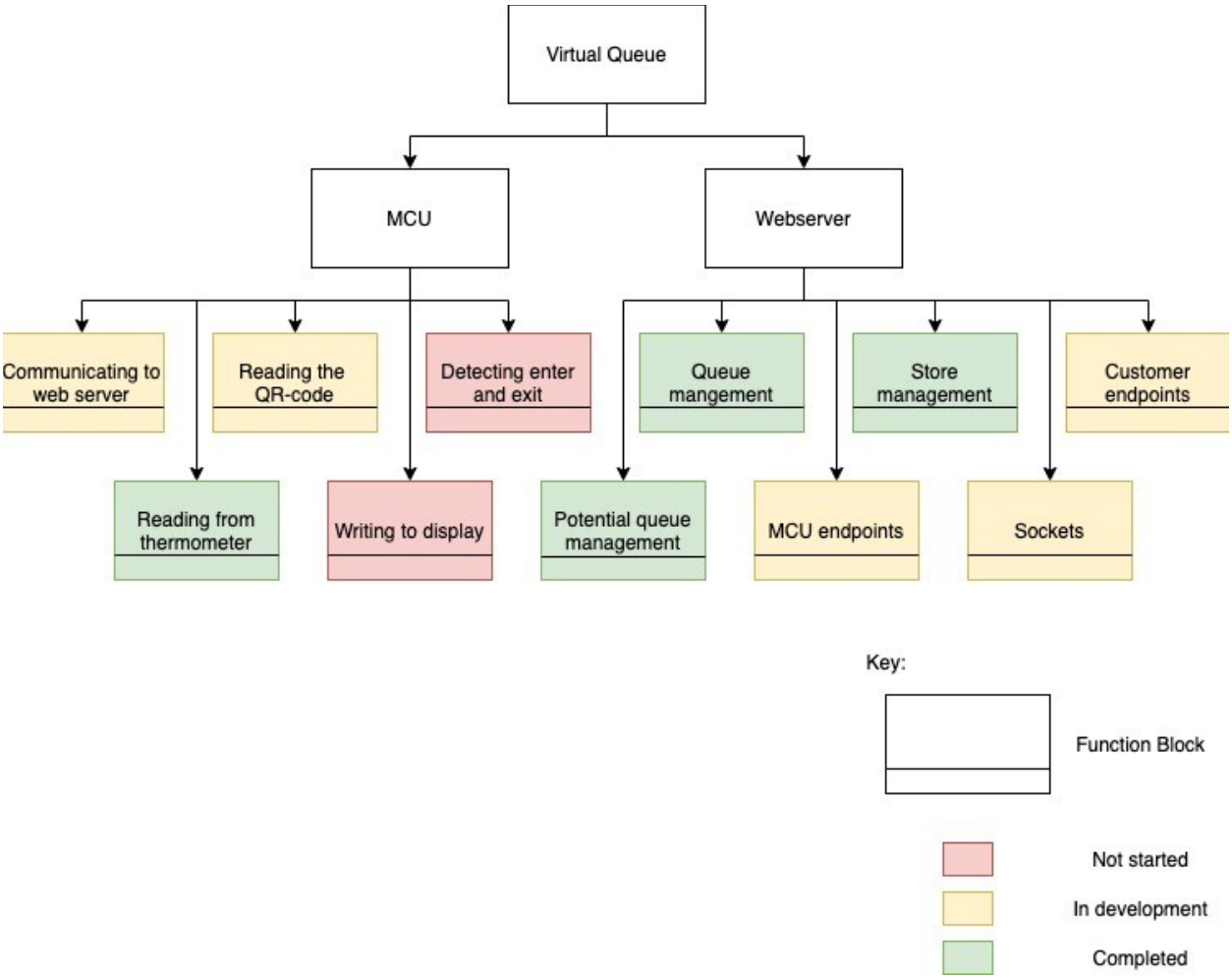
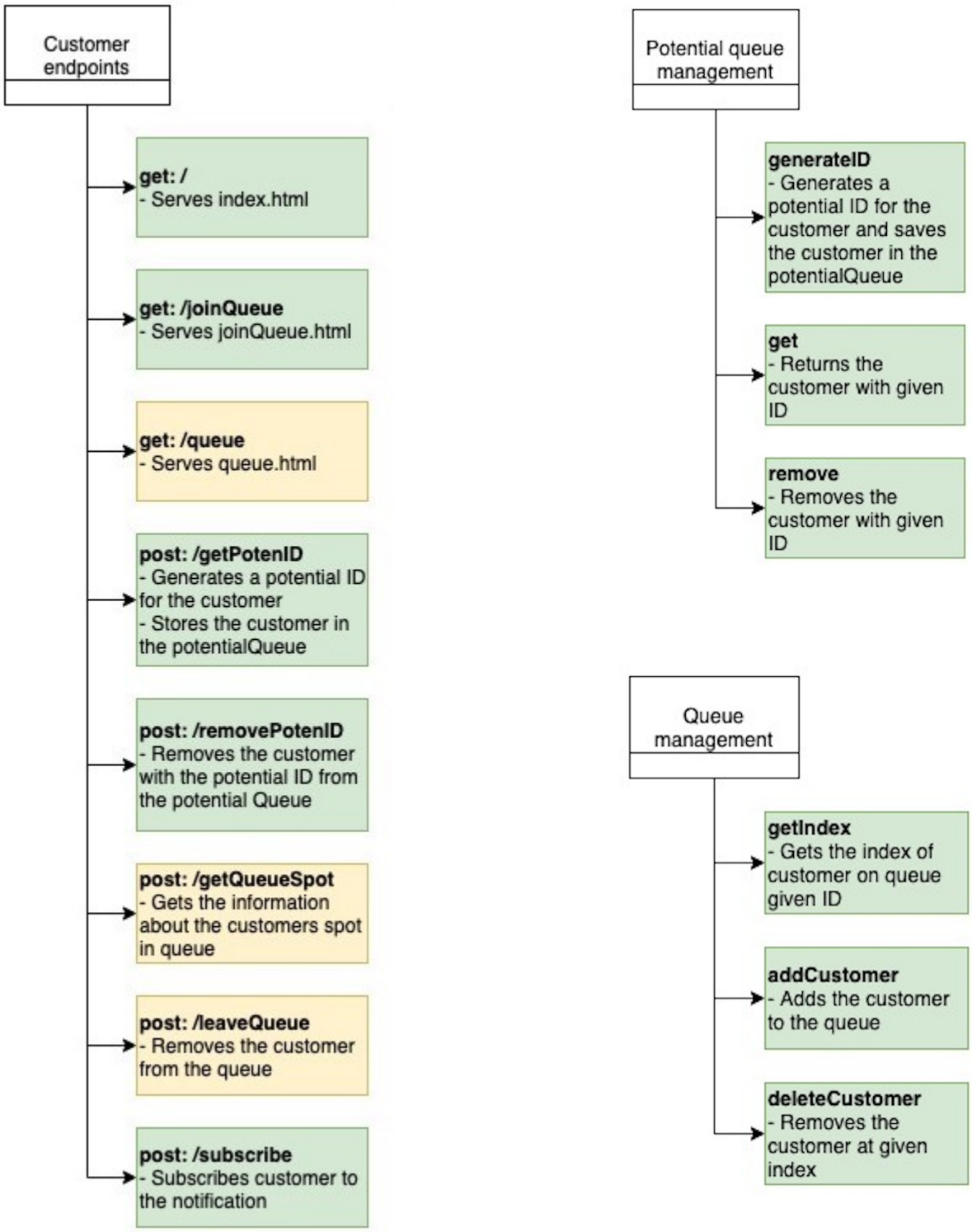
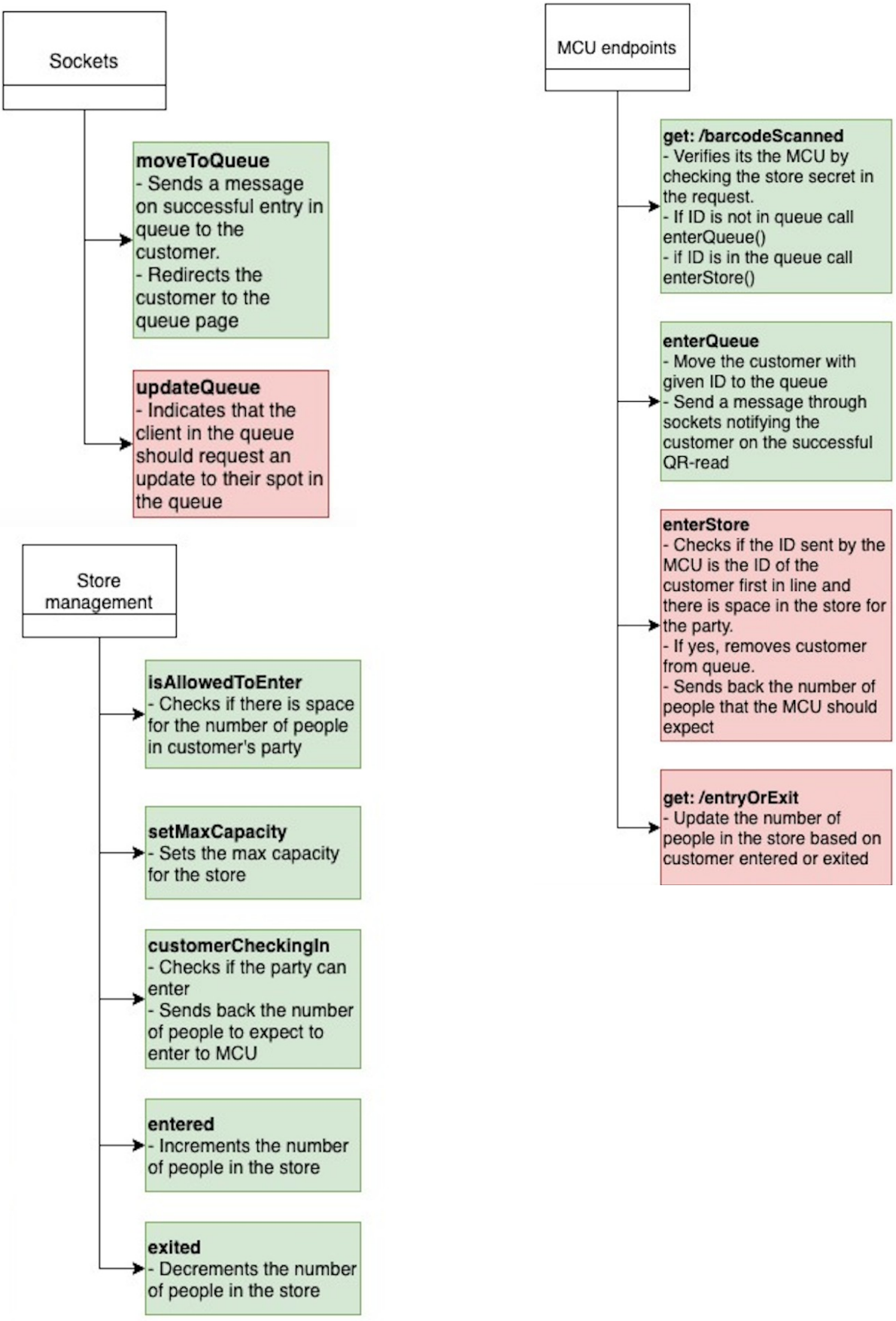
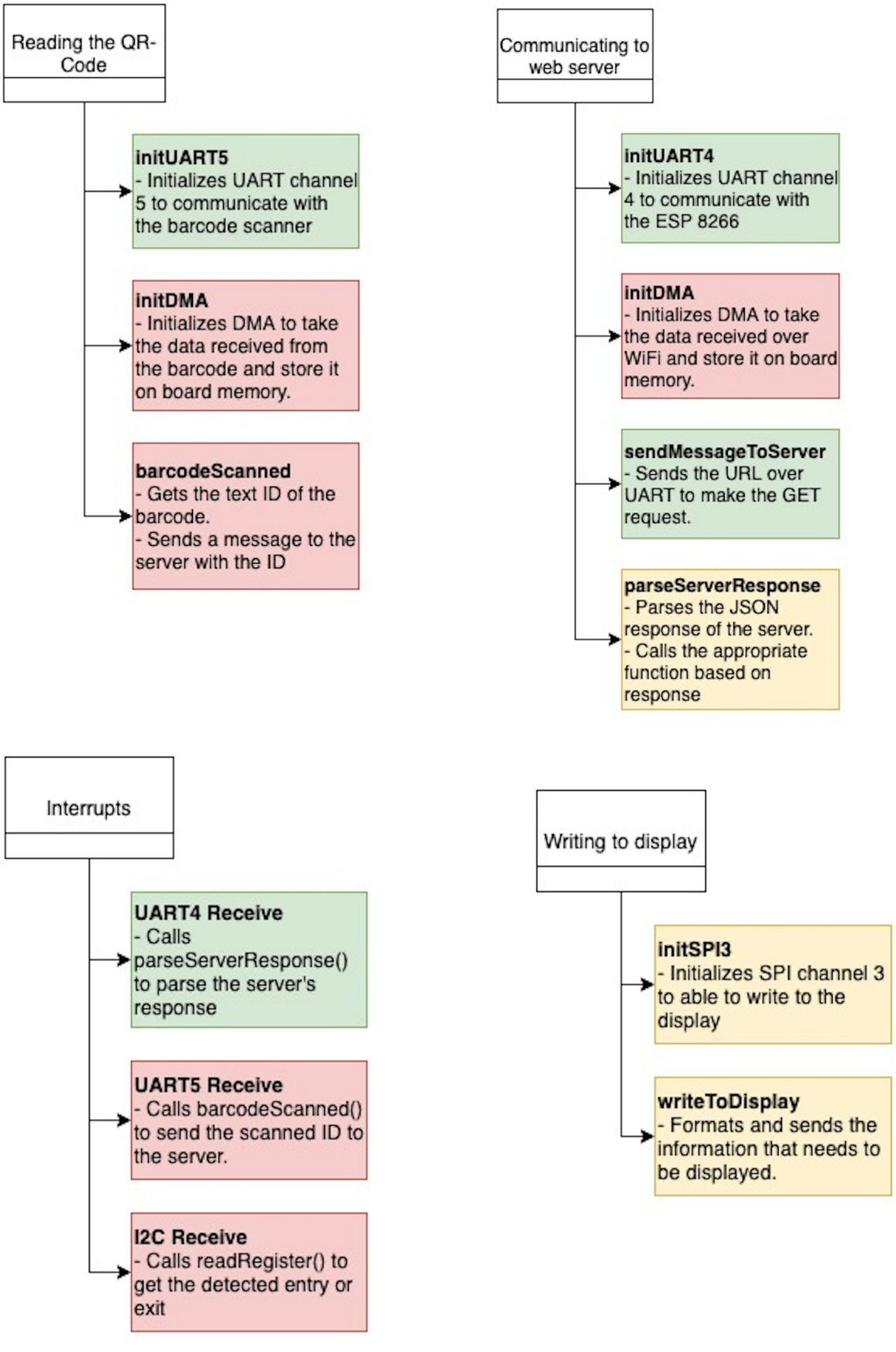
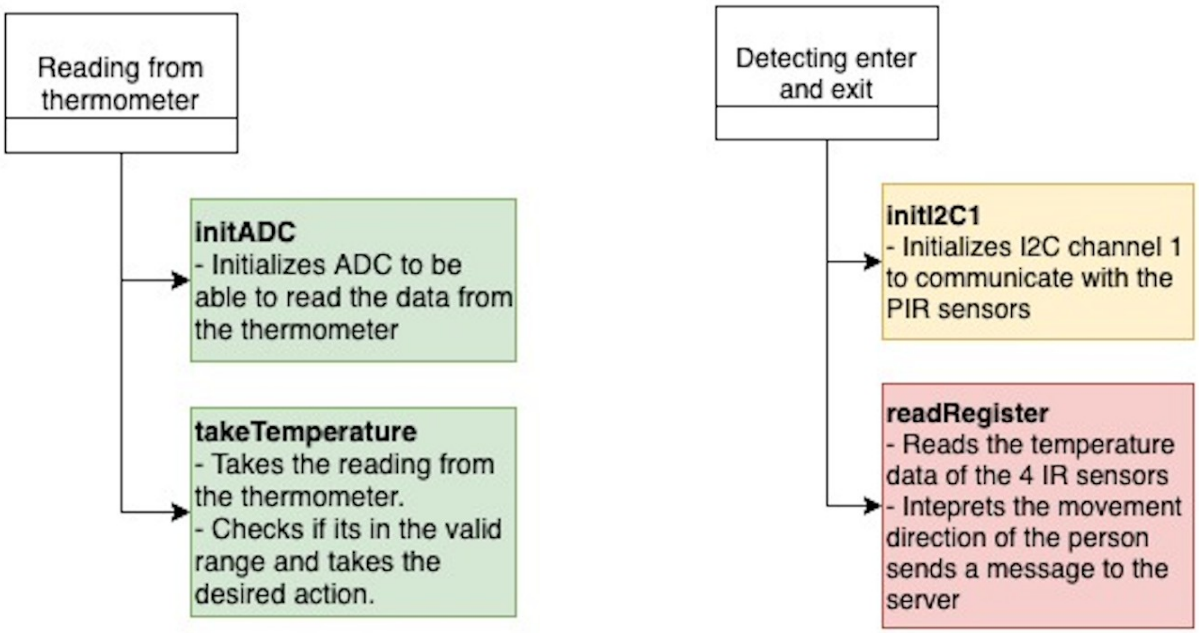


Figure 2: The overall code structure


Customer endpoints
get: /
- Serves index.html
get: /joinQueue
- Serves joinQueue.html
get: /queue
- Serves queue.html
post: /getPotenID
- Generates a potential ID for the customer
- Stores the customer in the potentialQueue
post: /removePotenID
- Removes the customer with the potential ID from the potential Queue
post: /getQueueSpot
- Gets the information about the customers spot in queue
post: /leaveQueue
- Removes the customer from the queue
post: /subscribe
- Subscribes customer to the notification
Potential queue management
generateID
- Generates a potential ID for the customer and saves the customer in the potentialQueue
get
- Returns the customer with given ID
remove
- Removes the customer with given ID
Queue management
getIndex
- Gets the index of customer on queue given ID
addCustomer
- Adds the customer to the queue
deleteCustomer
- Removes the customer at given index


Sockets
moveToQueue
- Sends a message on successful entry in queue to the customer.
- Redirects the customer to the queue page
updateQueue
- Indicates that the client in the queue should request an update to their spot in the queue
Store management
isAllowedToEnter
- Checks if there is space for the number of people in customer's party
setMaxCapacity
- Sets the max capacity for the store
customerCheckingIn
- Checks if the party can enter
- Sends back the number of people to expect to enter to MCU
entered
- Increments the number of people in the store
exited
- Decrements the number of people in the store
MCU endpoints
get: /barcodeScanned
- Verifies its the MCU by checking the store secret in the request.
- If ID is not in queue call enterQueue()
- if ID is in the queue call enterStore()
enterQueue
- Move the customer with given ID to the queue
- Send a message through sockets notifying the customer on the successful QR-read
enterStore
- Checks if the ID sent by the MCU is the ID of the customer first in line and there is space in the store for the party.
- If yes, removes customer from queue.
- Sends back the number of people that the MCU should expect
get: /entryOrExit
- Update the number of people in the store based on customer entered or exited


Reading the QR-Code
initUART5
- Initializes UART channel 5 to communicate with the barcode scanner
initDMA
- Initializes DMA to take the data received from the barcode and store it on board memory.
barcodeScanned
- Gets the text ID of the barcode.
- Sends a message to the server with the ID
Communicating to web server
initUART4
- Initializes UART channel 4 to communicate with the ESP 8266
initDMA
- Initializes DMA to take the data received over WiFi and store it on board memory.
sendMessageToServer
- Sends the URL over UART to make the GET request.
parseServerResponse
- Parses the JSON response of the server.
- Calls the appropriate function based on response
Interrupts
UART4 Receive
- Calls parseServerResponse() to parse the server's response
UART5 Receive
- Calls barcodeScanned() to send the scanned ID to the server.
I2C Receive
- Calls readRegister() to get the detected entry or exit
Writing to display
initSPI3
- Initializes SPI channel 3 to able to write to the display
writeToDisplay
- Formats and sends the information that needs to be displayed.


Reading from thermometer
initADC
- Initializes ADC to be able to read the data from the thermometer
takeTemperature
- Takes the reading from the thermometer.
- Checks if its in the valid range and takes the desired action.
Detecting enter and exit
initI2C1
- Initializes I2C channel 1 to communicate with the PIR sensors
readRegister
- Reads the temperature data of the 4 IR sensors
- Inteprets the movement direction of the person sends a message to the server