Bin Fun Game

Brandon Chan, Calvin Chan, Chris Yoon, Sebastian Lee, Stephanie Lam

University of British Columbia

EECE 409/429/419/439/400/469

April 15, 2016

PL40 - BIN FUN GAME

Final Project Report

Brandon Chan
Calvin Chan
Chris Yoon
Sebastian Lee
Stephanie Lam

Executive Summary:

A system was designed to keep track of the number of items thrown into one of the recycling bins in the Nest at the Vancouver campus of the University of British Columbia. Currently, many items found in these bins have been recycled incorrectly. The ultimate goal of the system described in this report is to engage bin users and motivate them to recycle properly. The system was constrained to be easy-to-use, inexpensive, water-resistant, low maintenance, and secure enough to withstand movement and theft.

The purpose of this document is to review the initial phase of the Bin Fun Game project. Three separate documents regarding the initial phase of the project are included in the final report that are labeled; requirements, design and validation. Each document can be found in the listed order. The requirements document outlines the requirements, constraints and goals of the project. The design document outlines all of the decisions considered during the creation of the project. Finally, the validation document outlines the testing strategies and how each requirement is met. The project owners encourage readers to gain knowledge from this report and help to contribute to the project's initiative.

PL40 - BIN FUN GAME

Requirements Document

TABLE OF CONTENTS

1 INTRODUCTION

UBC, as part of the Zero Waste Action Plan[1], aims to encourage people on campus to reduce and recycle waste. As described by our clients, the current problem is that there are still a significant number of waste items that are consistently placed incorrectly into the recycling stations. The client's goal for this project is to design and test a Fun Theory product, that will engage campus users and motivate them to recycle properly. The Fun Theory, an initiative of Volkswagen, states that "something as simple as fun is the easiest way to change people's behaviour for the better."[2]

Success of the project will be justified by the satisfaction of the user requirements and constraints by the development team. A successful project would perform well by being able to detect when any item is placed in the recycling station. The project should also be easily maintainable, robust and allow for change. A final component of success is that the entire project can be completed within the allotted budget for a capstone project.

[1] https://sustain.ubc.ca/campus-initiatives/recycling-waste/what-ubc-doing/waste-action-plan
[2] http://www.thefuntheory.com/

2 FUNCTIONAL ASPECTS

Requirement	Actors	Goal
R1	Client (Researchers), System (Used data to display), End Users (Students or any other individual walking by)	An automated method of counting the number of items dropped into each individual waste bin. This counting method will be referred to as a frequency counter.
R2	Client (Researchers), System	A database that is able to store and modify the following frequency counter data: ● Time and date. ● Specific bin information such as unique identification numbers and location.
R3	Client (Researchers)	A user-interface that our clients can use to make changes to the database from R2. These changes must allow our client to: ● Download signal data from R1 from the database from specific date ranges. ● Input accuracy data into the database for a particular date.
R4	End Users (Students or any other individual walking by)	The final product must have the system users engage physically with the recycling stations. Our client with the background in psychology, believes that a physical aspect of the game will help to increase user engagement.
R5	Clients and future systems crew	The project must be maintainable in the sense that the project client should be able to preserve the longevity of our project and continue developing the product after we have stopped working on it. ● R5.1 The hardware must be accessible; in the event that a broken module needs to be replaced, a system administrator should be able to replace the broken modules. ● R5.2 The software must be susceptible to change; in the event that a feature needs to be added, modified or removed, a system administrator should be able to do so.

3 NON-FUNCTIONAL ASPECTS

Non-functional aspects: what quality attributes the product needs to exhibit; the “ilities”: reliability, security, portability, interoperability, etc.

The qualities that our product needs to exhibit are:

- Reliability: The product needs to be reliable in terms of:
    - Accuracy of the signal data. The frequency counter [R1] needs to be able to record the accurate time of when an object is dropped into a bin. The frequency counter also needs to be able to have a high rate of detecting items.
    - Database. The transfer of data must be reliable as to maintain the correctness of all the data.
- Security: The product will need to be secure in terms of:
    - Physical security of hardware. The hardware must be securely fastened to the bins as to avoid theft of the system components.
    - Security of the software is not a priority for this product.
- Portability: The product will need to be portable in terms of:
    - Physically able to switch the entire system to another bin without much work.
- Interoperability: The product will need to be interoperable in terms of:
    - This quality ties in with R5, in which the system must be susceptible to change and easily integratable.

4 CONSTRAINTS

C1. The design cannot exceed current floor plan of the recycling bins. The project design cannot obstruct the current floor plan of the NEST. The project must be designed to fit any bin, regardless of location.

C2. The project design must not interfere with or burden a janitor's process of cleaning. If the system is to be mounted inside the recycling bin, there must be enough clearance so that the janitor's process is not disturbed while removing the bins. There is currently 2-3" of clearance inside the bin (see Appendix A.2).

C3. The system must be detachable or able to move with the bin.

C3.1 All four recycling bins are not connected together but are instead in pairs. The system must be usable for all 4 bins but still able to move the pairs apart. (Figure 2)

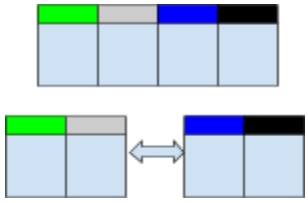


Figure 2:Recycling Bin Pairs

C4. The system should be robust as well as the following:

C4.1 Should be water-resistant so that any liquid spillage has a low chance of damaging the hardware.

C4.2 Should be securely fastened so that parts are not easily stolen through force.

C4.3 Should be securely fastened so that no parts fall off when the bin is moved.

C5. A power source is needed for all hardware components.

C6. No custom hardware should be used in the design. Any broken parts should be easily obtainable and replaceable.

C7. The system cannot be annoying for users. The system will be placed in high-traffic areas and should not disturb anyone in the area.

C7.1. The system cannot make too much noise.

C7.2. The system cannot produce too much light.

C8. The interface should be simple and easily understandable for all users.

C9. Budget: $650

4 Future Recommendations

This section of the document is a list of tasks that we recommend for the client to consider if chosen to extend the project in the future. The following tasks were not discussed with the client and so they have not been included in the section above.

1. A script that would automatically start the system as well as other scripts necessary for the system to function well. The startup script would take in a bin location as a parameter.
2. A new type of sensor to detect when items are thrown into the bins. Please see the *Design Document* for more detail on the issue.
3. Fix issue a user-interface issue on the web application game; there is an extra button on the top left the screen.
4. Create a script or modify the system to automatically delete old files from the SD card as well as the database. Both methods or storage will become full over time without intervention.

PL40 - BIN FUN GAME

Design Document

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

List of Figures

1 INTRODUCTION

1.1 High-Level Overview

The major subcomponents of this project was a microcontroller, sensor, display and database. These components are assembled as depicted in Figure 1. A sensor is attached underneath the lid of the recycling stations, and streams data into a microcontroller. The microcontroller will continuously and simultaneously update a database and a display.

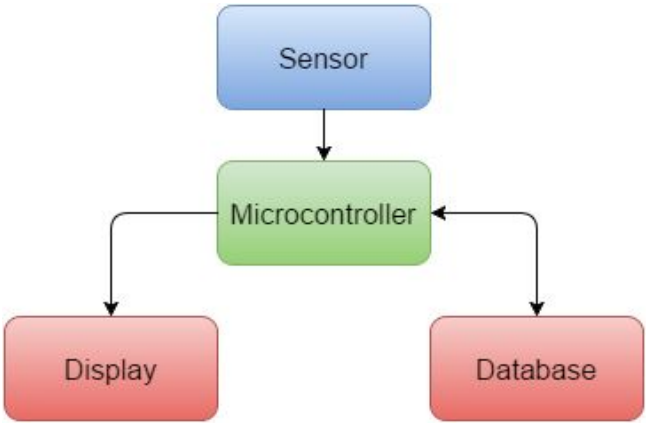


Figure 1. Flowchart of Major Subcomponents

1.2 High-Level Software Architecture

The image below depicts the software architecture that is used for the sensor, microcontroller and the database. There are 5 software classes that are used in the project; sensor, queue, buffer, main and SD. The role of each class is as follows:

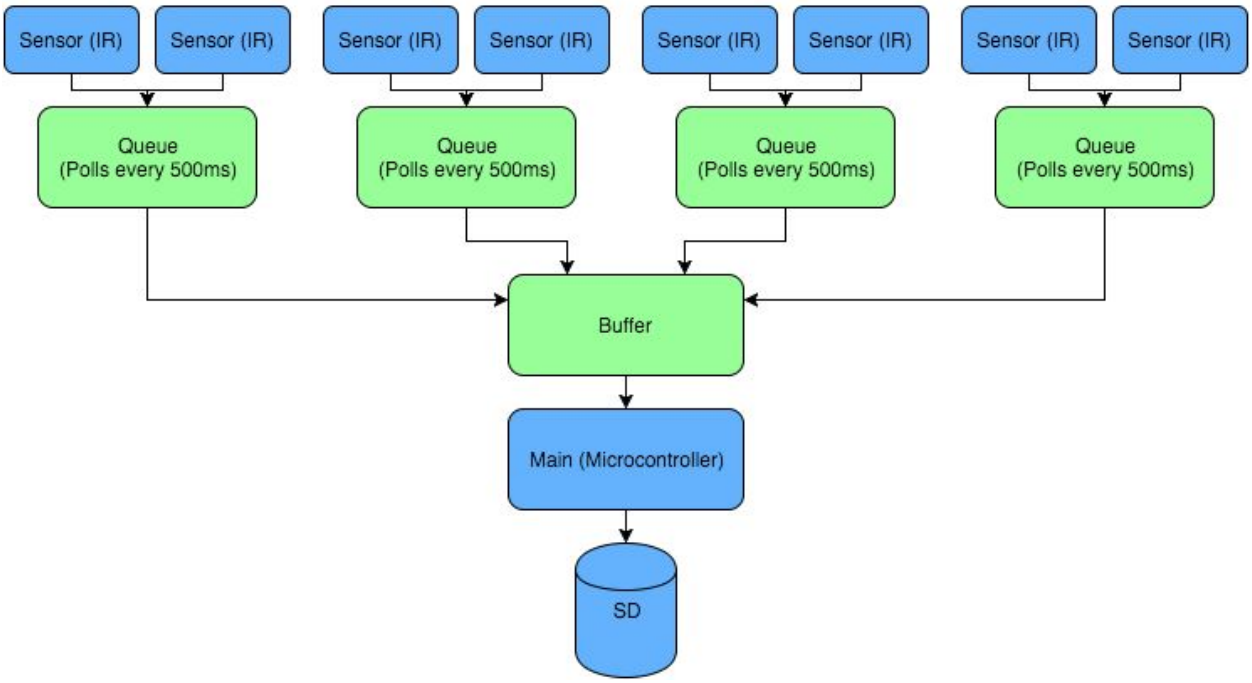


Figure 2. High-Level Software Architecture of System

- Sensor: Specifies which GPIO pins on the Raspberry Pi the sensor must be attached to, as well as which colour bin each sensor should be placed in. The sensor class will utilize two individual sensors to detect an item that is dropped in the bin, and store a timestamp when done so. When an item is detected, the sensor class records the date, time, color of bin, and location the item is placed, in a string variable. The stored string variable is expected to be copied into a queue, and the variable is then overwritten by another signal.
- Queue: The queue class uses a deque as the internal data structure. Upon initialization, the queue class creates a sensor object. The role of the queue class is to check, every 500 milliseconds, if the sensor class has detected any items placed into the bin. The 500 millisecond wait is created so that if the two sensors in the same bin detect an item, the item would not be counted twice. The queue is designed to be able to quickly retrieve multiple signals from the sensors before the buffer reads from the queue.
- Buffer: Upon initialization, the buffer class creates four queue objects, one for each bin colour. The role of the buffer class is to gather all of the signal data from each queue using a round robin scheduling algorithm (repeatedly check from each one in order). The

buffer also maintains a count of the total number of items placed in each bin.

- Main: The main class is used to initialize the system and all of its classes. Upon initialization, the main class creates a buffer and SD object. The main class is also responsible for initiating the round robin check from the buffer class indefinitely. The main class periodically removes the data from the buffer, and stores it in both the SD file and the remote database.
- SD: The SD class is used to write all the signal data to CSV files on the SD card of the Raspberry Pi.

1.3 Client Server Architecture

The client–server architecture is a distributed application structure that separates the workload between the providers of service and service requesters. The providers of service are called servers and the service requesters are referred to as clients. The client and server communicate with each other over the internet. A representation of the separation between clients and servers can be seen in Figure 3 below. Clients initiate communication sessions with servers and the servers will serve the content of an application back to the client. The client server architecture is particularly useful for web applications that do not have a need for high scalability or large amounts of processing power. All of the web applications of our final product utilize the client-server architecture to meet the needs of the requirement.

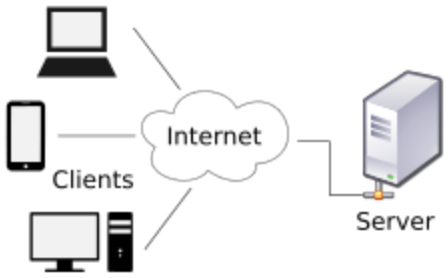


Figure 3. Diagram of Client-Server Architecture

1.4 Programming Language Decision

The software language that is used to implement the code to facilitate detection of the sensors is Python. We have specifically chosen Python because it was found[1] to perform reasonably well. In addition, there are many tutorials that can be found online that have implemented similar projects to ours using the same sensor and microcontroller. Finally, a third reason is that Python is known to be easy to learn and so there is a short learning period.

[1] http://blog.dhananjaynene.com/2008/07/performance-comparison-c-java-python-ruby-jython-jruby-groovy/

2 REQUIREMENT 1 DECISIONS

2.1 Introduction

Requirement 1 is to have an automated method of counting items placed in the bins. To have items automatically detected when placed in a bin, there is a need for hardware components. Our choice is to have a microcontroller attached to a sensor to keep track of when items were placed in the bin. Therefore, for this requirement we considered several types of microcontrollers and sensors to use.

2.2 Final Microcontroller Decision

We had compared both the Raspberry Pi and the Arduino. and found they both satisfy all of the project's constraints. Therefore, our reasoning for choosing the Pi is that our group has more experience working with a Raspberry Pi. The Pi is a popular product, meaning it is well-tested, and replacements or maintenance documentation can be easily obtained if required. We have also found that the Pi has more built-in features that would not require additional installations, in contrast to the Arduino. For the Arduino to match all of the needed functionality, namely a SD card reader, HDMI output and USB ports, the resulting size of the add-ons to the Arduino will be larger than a stock Raspberry Pi. In addition, the Arduino with add-ons would leave much more debugging in the hands of the client.

2.3 Final Sensor Decision

An ultrasonic sensor adheres to the constraints and requirements, but ultrasonic sensors are more expensive than IR sensors. Laser sensors have a faster response time compared to IR sensors. However, laser sensors are not optimal for our design because the laser sensor has a smaller detection radius compared to an IR sensor. A small detection radius may not be able to detect small objects such as gum wrappers. It would be possible to install multiple laser sensors to broaden the detection radius, but the cost of purchasing multiple laser sensors conflicts with constraint C9. The IR sensor adheres to all constraints and requirements, and is also a low budget solution for this project. Therefore, after examining all of the options, we concluded that the best

sensor to use for this project was the IR sensor.

The final product made use of two passive infrared sensors (module HC-SR501) in each bin at the recycling station. The use of two IR sensors proved to have a more stable average detection rate of 70% with a 10% deviation. We had used a single IR sensor in each bin which resulted in a 70% average detection rate with a 20% deviation.

2.4 Tested Alternate Solution 1

When we had found that the IR sensors yielded a 70% detection rate, we had attempted to solve the problem using ultrasonic sensors (HC-SR04). After prototyping the ultrasonic sensor, we had tested the detection rate to be an average of 30%.

The ultrasonic sensor is an active sensor, meaning that the sensor needed to be activated in order for it to detect something. The activation time for the sensor was 10 milliseconds and then the sensor would spend 10 milliseconds detecting objects within its detection range. However, the sensors required a 10 millisecond wait time before re-activation. This wait time was meant to allow for an analog input signal to settle. The result was that the sensor would actually only detect objects once every 20 milliseconds, which proved to be problematic after testing.

2.5 Tested Alternate Solution 2

We had also attempted to use the ultrasonic and IR sensors together in hopes that the detection radius would increase. We connected the two types of sensors to the microcontroller and conducted some tests by throwing various recyclable items in front of the sensors. The result was that the detection rate was only as good as the IR sensor and there were only two cases (out of 50) where the ultrasonic sensor detected an object when the IR sensor did not. However, there were more cases when the IR sensor detected an item while the ultrasonic did not.

2.6 Proposed Alternative to Final Decision

Due to the imperfect detection rate of the IR sensors, we devised an alternative solution

that would offer a much higher detection rate. The solution involves the use of laser sensors. Whenever an object collides with one of the lasers, a signal will be sent to the Raspberry Pi to indicate that an object has been thrown into the bin. However, since laser sensors only have a small detection radius, there is a possibility that small objects may not hit any of the lasers. To minimize this possibility, multiple laser sensors can be attached to each bin, where each laser covers a different area of the bin. The more laser sensors, the higher the detection accuracy.

As an example, we created a rough schematic, seen in Figure 4 below, of three laser sensors connected to one bin. Since there are four bins, there will be a total of twelve laser sensors. We will also require a separate 5V power source since the Raspberry Pi will not be able to generate enough current to power a large amount of lasers. Thus, we included a 5V voltage converter in the schematic.

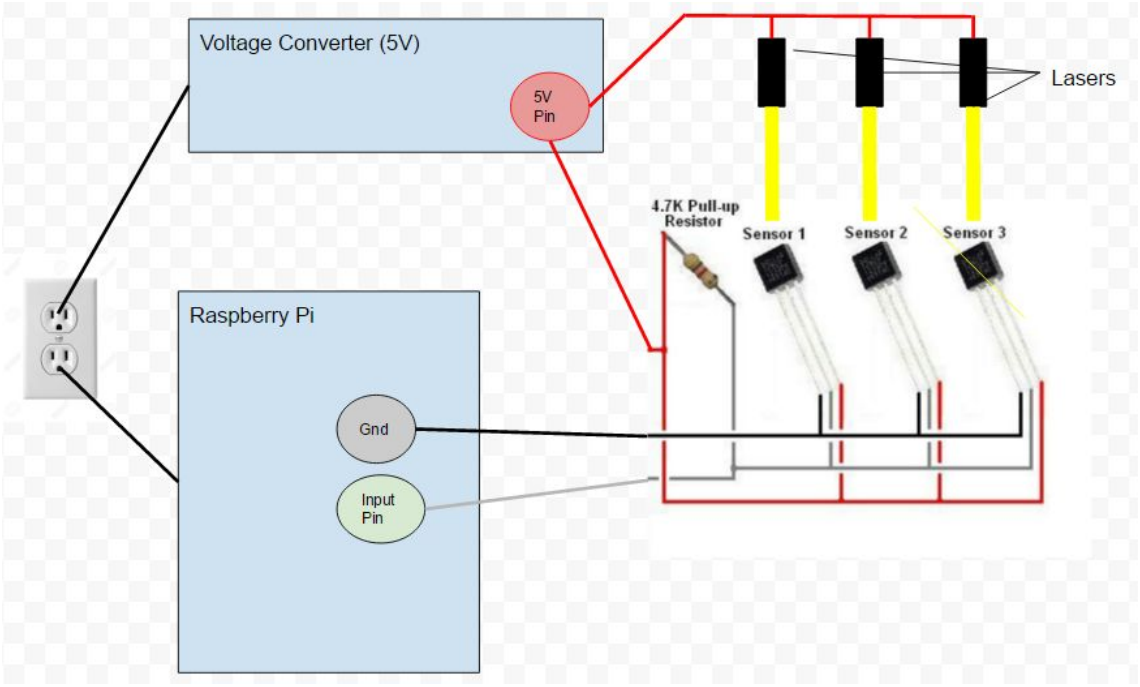


Figure 4. Schematic of Laser Solution

3 REQUIREMENT 2 DECISIONS

3.1 Introduction

Requirement 2 is to have a database that the microcontroller can connect to and store data that is related to the recycling station. The data stored for each item placed in the bin is the date, time, location and bin colour. We decided to have two locations to store data; one location was the SD card on the microcontroller and the second location was a remote database hosted on Amazon web services.

3.2 Storage on SD card

The decision to store data on the SD card was made so that, in the case that the microcontroller lost connection to an internet connection, data would not be lost over the internet. A second consideration was that the SD card was free storage in comparison to a remote database which would require maintenance and would need to be increased in size when the database was full, which would cost the client money.

The data is stored on the SD card in a comma separated value (csv) file. The choice of a csv file was that the data is easy to read, and the files would be easily compatible with a spreadsheet, thus able to be consolidated into a graph for visualization.

3.3 Remote Database

We decided between two types of databases to install on our Amazon web server: Oracle and MySQL. Although Oracle offered more features and functionality, it was designed for commercial use and the extra features were not required for this project. Thus, we decided to install a MySQL database on the server since it was more user-friendly, and our team had previous experience using this tool. Furthermore, the MySQL framework allowed our team to connect to the database using Python, which kept our code's programming language consistent with other components of the project.

4 REQUIREMENT 3 DECISIONS

4.1 Introduction

Requirement 3 is to have users of the recycling station be physically engaged with our project when used. To engage the users with the bin, the client decided to have a monitor display the number of items that were placed in each bin every day, as well as display the percentage of items that were correctly placed in each bin from the previous day. A second component was to have a web application game [2]that would teach users where fifteen of the most problematic items belong.

4.2 Monitor Display

A web browser that would be displayed on a monitor attached to the Pi was a request by the client. The major decisions for the design of the display were the colours, animations, accuracy score and the bin counter. All of the colours were chosen to be pleasing to the eyes, not be too confusing and to match the colours of the bins as closely as possible. The images at the bottom of the screen were created to resemble each individual bin and the colours of the images must not confuse the user. The accuracy scores had to be automatically updated according to values that were input into our database by our clients. Finally, the display needed to show the number of items placed in each bin. Videos[34] of the monitor display in action can be found in the footnote.

[2] https://www.youtube.com/watch?v=b5GJckuh9NM
[3] https://www.youtube.com/watch?time_continue=1&v=UKnCsrDNccI
[4] https://www.youtube.com/watch?v=9QDQ9PxBHro

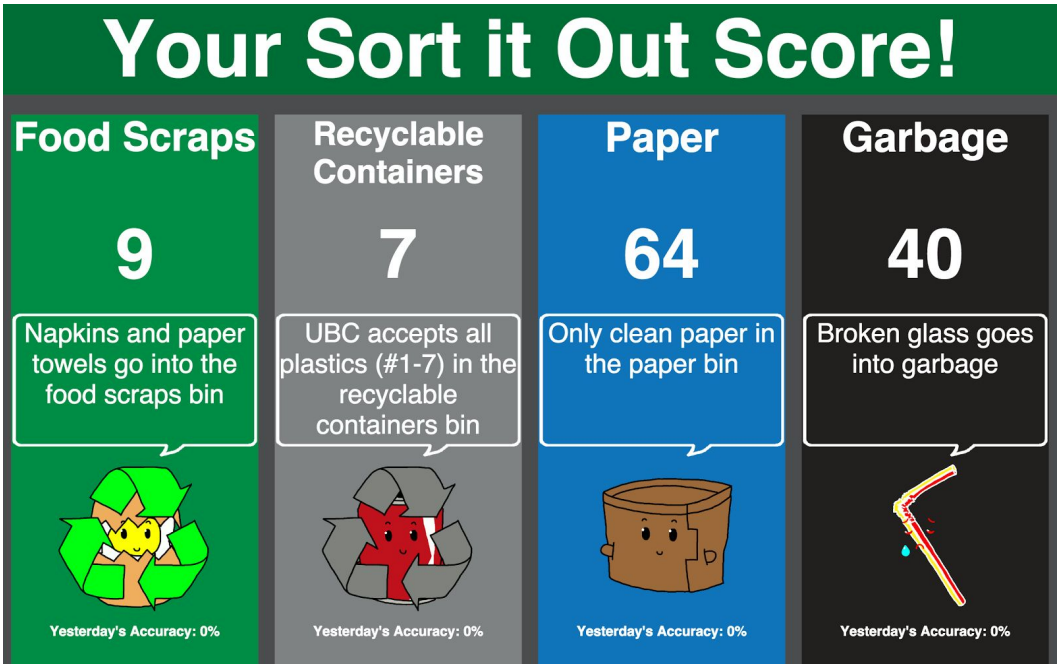
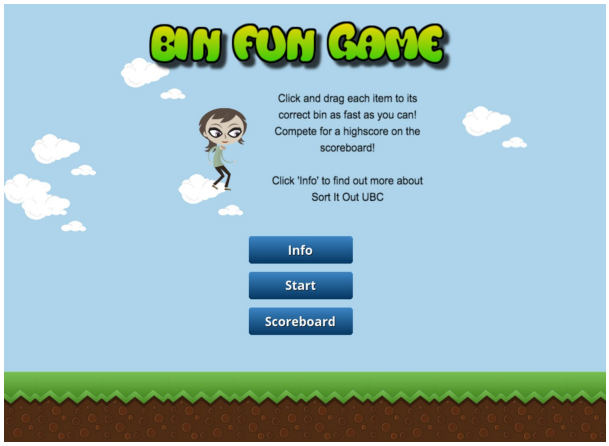
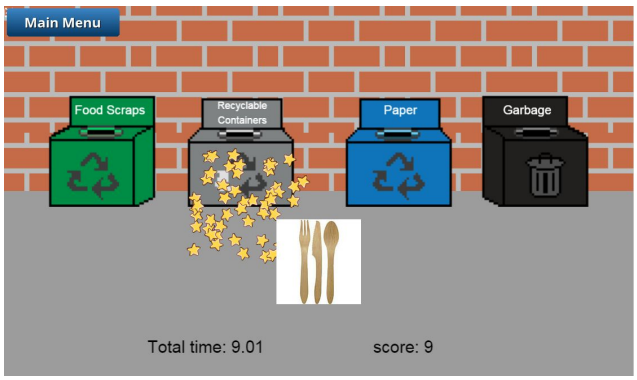


Figure 5. Screenshot of Monitor Display

4.3 Web Application Game

We are using an open source HTML5 game framework called Phaser. Our decision to use the Phaser framework is because we have past experience in using the framework. In addition, Phaser has an extensive amount of documentation, as well as online examples. The game is a remake of our client's existing game, and thus no gameplay [5]decisions are made by us.





[5] www.binfungame.com

Figure 6 & 7. Screenshot of Homepage (left) and Gameplay (right) of Web Application Game

5 REQUIREMENT 4 DECISIONS

5.1 Introduction

There are two components to this requirement. The first component is to implement a web based user interface [6]that allows the client to update the accuracy score on the web display of the recycling station. The second component is to implement a web based user interface [7]that allows the client to download the CSV files directly from the database, so that they may view the data on their local machine. Both of the web based interfaces are implemented using the client-server architecture because there would be low traffic, and there is no need for scalability. The technologies we use to implement the web applications are as follows:

- JavaScript
- HTML
- Bootstrap
- CSS
- JQuery
- Python (Webpy)

5.2 Client Interface to Access Database

The client user interface allows the clients to select data from the database based on the specified inputs. The client enters a start date, end date, bin location, and bin color, which are used to query data from the database. Figure 8 below shows a screenshot of the interface. The start and end date is the time period in which the client wants the data of. A CSV file is then created to store the data received from the database, and is also saved onto the Amazon web server. When the client clicks "Download", a download window pops up, allowing them to save the newly created CSV file onto their own computer. The input fields are chosen to give the client flexibility in their data search, but also keep the user interface very simple.

[6] http://ec2-54-218-32-132.us-west-2.compute.amazonaws.com:8082/
[7] http://ec2-54-218-32-132.us-west-2.compute.amazonaws.com:8083/

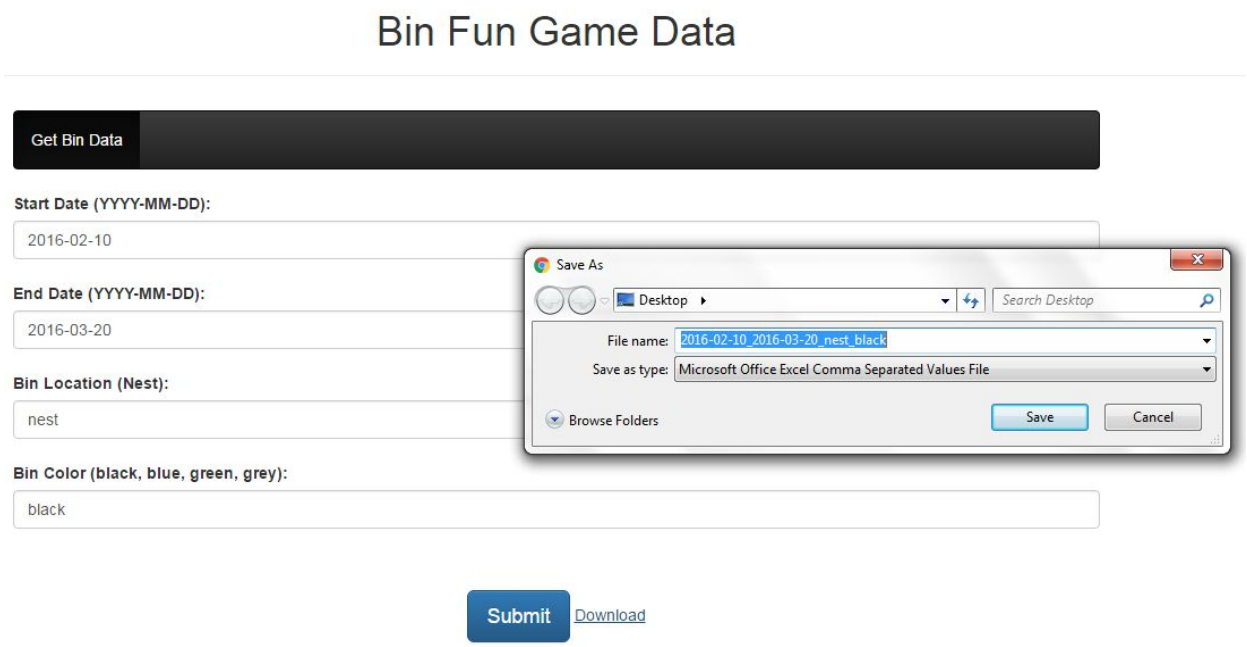


Figure 8. Screenshot of Client Interface to Access Database

5.3 Client Interface to Update Accuracy Score

The interface for the accuracy scores includes four text boxes and a calendar. The text boxes allow the client to input the accuracy scores for each bin, while the calendar allows the client to select which date the accuracy scores were calculated. When the client clicks “Submit”, the accuracy scores and the selected date are uploaded into the database. The input fields are chosen to maintain a simple user interface. Figure 9 below is a screenshot of the interface.

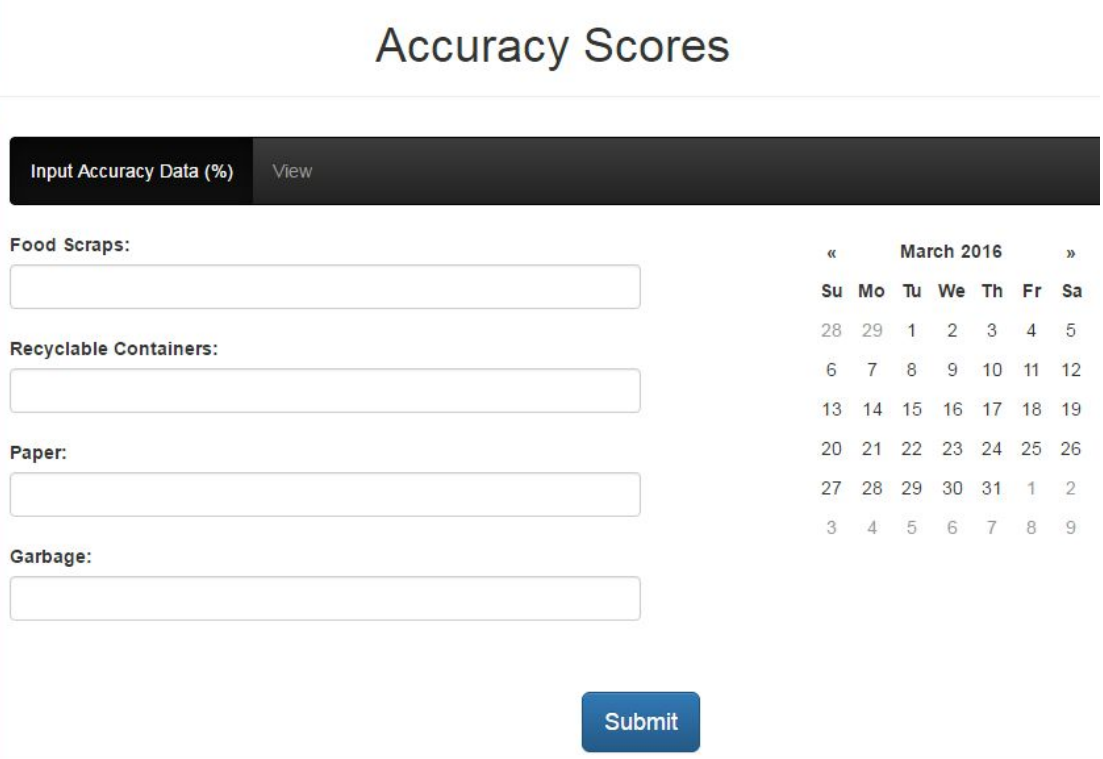


Figure 9. Screenshot of Client Interface to Update Accuracy Score

5.4 Programming Language Decision

The backend framework of both user interfaces are implemented using Python and the WebPy library. Python is specifically chosen to stay consistent with the use of Python for the aforementioned requirements.

Bootstrap is chosen as a frontend framework because it allows for faster and easier web development. Bootstrap is commonly used in industry to develop web applications. In addition, HTML, CSS and JavaScript are used because these are standard web application development technologies and because Bootstrap uses these technologies.

5.5 Implementation of Client-Server Architecture

The web server is hosted on Amazon web services. Amazon web services is a free and easy to use web hosting service. We are using WebPy for the web server, instead of the default Apache engine, to allow for a client to connect to the server, and to communicate with the

database that is also installed on the web server.

Figure 10 below depicts the workflow of a client-server web application. When a client opens a browser on their computer and requests a web page from a server, the server responds to the client by sending the data necessary to load the corresponding web page. The web page is displayed in a client browser by utilizing JavaScript, HTML and Bootstrap. The client can then use the interface from their web browser.

When a client submits a request via the user interface, the client's browser sends a request to the server using JQuery. The database processes the request on the web server using the WebPy framework. The server has access to the database and all of its content. When the server completes processing the request, the server sends a JQuery response back to the client.

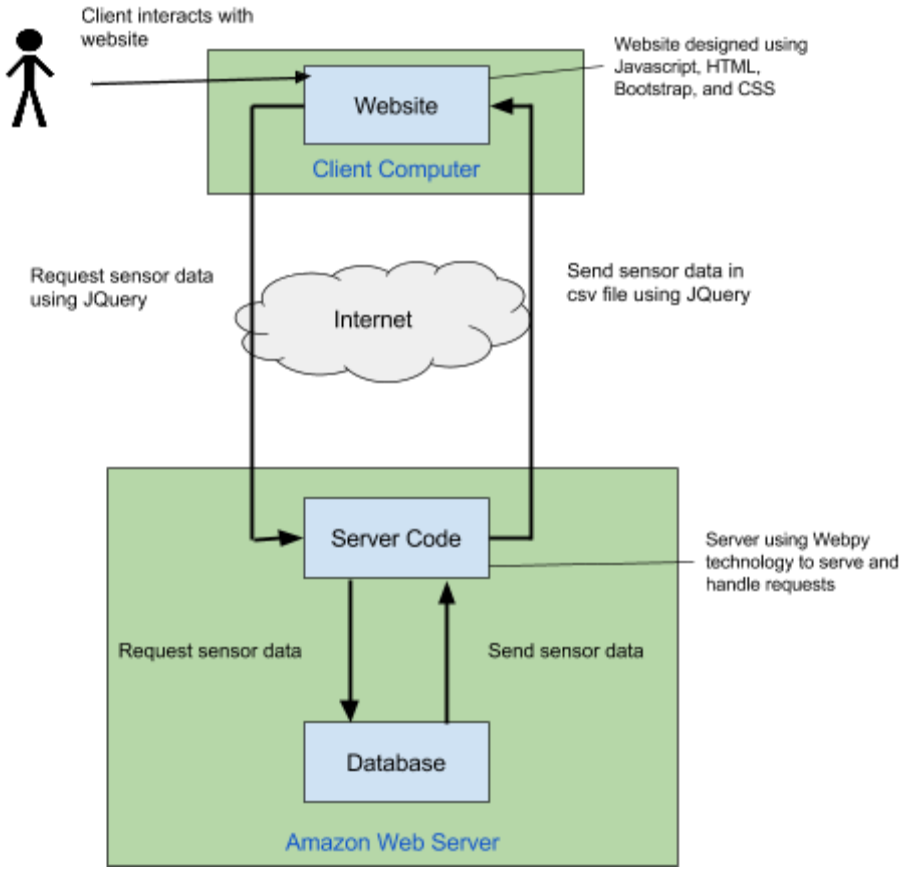


Figure 10. In-depth Diagram of Client-Server Architecture

6 REQUIREMENT 5 DECISIONS

6.1 Introduction

The client's request is that the entire project be easily maintainable even after the period of which the capstone group has completed the project. The hardware components must be robust and be able to sustain reasonable force. The software architecture is designed to allow for change in any module such as sensors. In addition, adequate documentation is provided to the client so that they understand how to use the system, as well as allow for another developer to change the system.

6.2 Hardware Components

All hardware components are specially selected to be very inexpensive and easily purchased online from common suppliers such as ModMyPi, Amazon and Ebay. In the case that any hardware component fails, the client would be able to search components online and buy them from anywhere in the world.

6.3 Pi Case

We are using a 3D printed case for the microcontroller. The case is specially designed to cover all of the inputs of the Raspberry Pi that are not being used for the initial phase of the BinFunGame project. AutoDesk is used to create the images seen below. Figure 11 on the left shows the part of the case that holds the Pi and is able to be screwed onto the bin station. Figure 12 on the right is the cover of the case that is also designed to be screwed to keep both pieces of the case together.

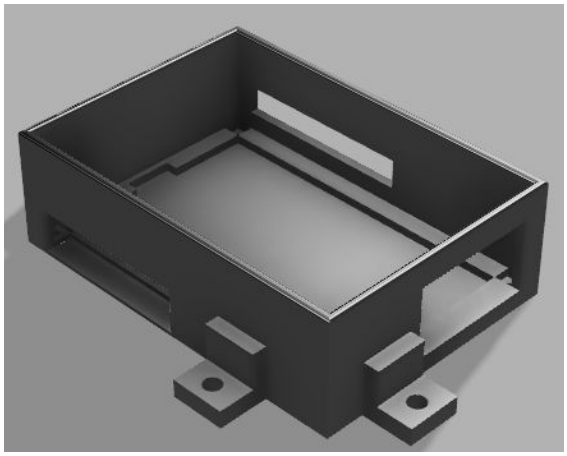
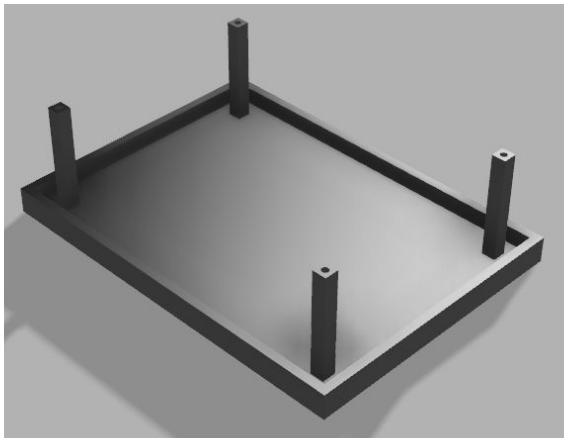



Figure 11 & 12. 3D model of Raspberry Pi Case

6.4 Documentation

The development team has left copious amounts of documentation for the client to resume the project in the future. The documentation files/practices are as follows:

- Industry standard code comments are written above all software methods explaining what each method does.
- High-level software diagram to outline the software architecture. The diagram can be referred to in section 1.2 of this document.
- Step-by-Step instructions explaining how to use each of the features that we developed for the client and require manual labour.
- A requirements document outlining the requirements, constraints and goals that we met and those of which we did not satisfy.
- A design document that outlines all of our design decisions and provides alternative solutions to designs that failed the requirements.
- A validation document that outlines the results of each of the implemented features as well as the test cases.

7 CONCLUSION

7.1 System Overview

The final product that was completed during the initial phase of the BinFunGame project is fully depicted in the Figure 13 below. There are passive infrared sensors that are connected to a Raspberry Pi which stores signal data on its SD card. The Pi also uses Wifi to send data to a database server that is hosted on Amazon web services. The server is implemented using the client-server architecture to update its own display which the Pi is able to display from a web browser and a computer monitor. The web game and client user interfaces are also hosted from Amazon web services and use the client-server architecture.

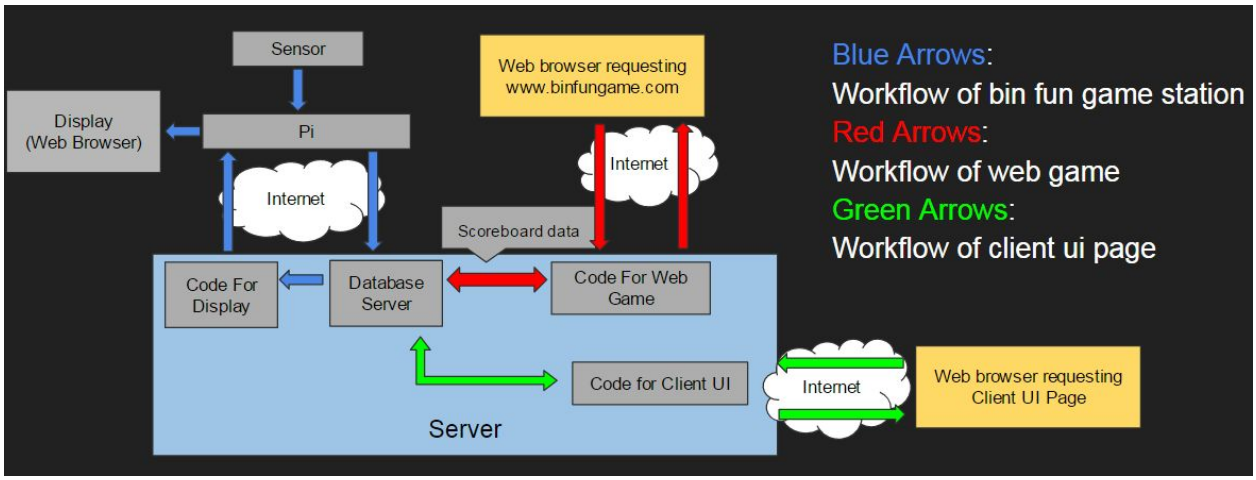


Figure 13. System Overview

7.2 Conclusion

All of the client's requirements are satisfied by the product made during the initial phase of the BinFunGame project. Most of the implementation is working and meets expectations, however the sensors are operating at optimal performance. Although the initial phase of the project is complete, the development team suggests that the project be refined in the future.

PL40 - BIN FUN GAME

Validation Document

TABLE OF CONTENTS

1 REQUIREMENT 1

1.1 Conclusion

The infrared (IR) sensors detect when an item has passed in its range of sight. We used two IR sensors in each bin to count the number of items that pass through. The setup of the double IRs had been manually tested, and the tested rate of detection was~70%.

The sensor was initially tested by having the program log print statements in a command prompt whenever movement was detected from the sensor. After implementation of the monitor display (refer to requirement 3), tests were conducted by confirming that the total number of items were correctly incremented on the monitor whenever a signal was detected.

1.2 Test Cases

Sensor Test Cases		
Test cases	Steps	Desired Result
Green Sensor	Insert an object into the green bin.	The display count for Green should increase by one. Animation should play, which should display a tip about recycling and a small gif.
Blue Sensor	Insert an object into the Blue bin.	The display count for Blue should increase by one. Animation should play, which should display a tip about recycling and a small gif.
Grey Sensor	Insert an object into the Grey bin.	The display count for Grey should increase by one. Animation should play, which should display a tip about recycling and a small gif.
Black Sensor	Insert an object into the black bin.	The display count for Black should increase by one.

		Animation should play, which should display a tip about recycling and a small gif.

2 REQUIREMENT 2

2.1 Conclusion

A MySQL database is set-up to store signal data, which sends data from the Raspberry Pi to the database over the internet. The database was tested through semi-manual test cases which were contained in a python script. By running the script, we created tables, inserted entries, and pulled data back from the database for verification. Each entry that was sent to the database included: the time and date at which an item is placed into the bin, and the color and location of that bin. We also tested inserting and retrieving data that contained the count values for each bin. The correctness of the database was evaluated by manually reading the contents of the database after insertion. Our test cases proved that the implementation was correct by sending and storing data in the expected formats.

2.2 Test Case

The desired results were verified by using a tool called MySQL Workbench which allowed us to view all entries in the database. We first ran our test cases to make changes to the database, and then we used the tool to ensure that the changes to the database were correctly made. We also used print statements, which printed information to the screen, for certain test cases to verify correctness.

Database Test Cases		
Test cases	Steps	Desired Result
Insert Sensor Data	Inserts an entry of sensor data into the database. The entry contains a bin color, a bin location, the time, and the date.	The database should contain the entry

Insert Multiple Entries of Sensor Data	Inserts two entries of sensor data simultaneously into the database.	The database should contain both of the entries that were inserted into the database.
Create a Count Table and Insert Count Values	Creates a table designed to hold the count values for each bin, and then inserts several entries into the table.	The database should contain contain a new table that is populated by all the entries that were inserted into it.
Update and Pull a Database Entry	Update a database entry, then retrieve the same database entry.	The database entry that was retrieved should have a different value to what it previously was.

3 REQUIREMENT 3

3.1 Conclusion

Both of the components that were implemented for requirement 3 have been manually tested. The monitor display has been manually tested by the development team by throwing items into each bin and verifying that each count on the display incremented. In addition, campus users have "interacted" with the station by throwing items into the bins, and looking at the monitor for feedback. The web game was manually tested by the development team and by end users. The end users were not able to find any problems with the web game[1].

3.2 Test Cases

The test cases used for the monitor display can be found in section 1 of this document. Below are the test cases used to test the web application game.

Web Game Test Cases		
Test cases	Steps	Desired Result
Loading State	Go to the URL www.binfungame.com.	Before Menu State, see Loading State which displays an image of 'Emily' and a loading bar.
Menu State: Initial	After Loading State, game should enter Menu State.	In Menu State, should see the Title, 'Emily',Instructions, background, and 3 buttons: Info, Start, Scoreboard
Menu State: Info	In Menu State, click on Info button	Clicking on Info button should open up a new link that should display more information about Sort-It-Out. https://sustain.ubc.ca/campus-initiatives/recycling-waste/sor

[1] https://www.youtube.com/watch?v=5YZQ-Cj28Cw

		t-it-out
Menu State: Start	In Menu State, click on Start button	Clicking Start button should bring you to Main Game State.
Menu State: Scoreboard	In Menu State, click on Scoreboard button	Clicking Scoreboard button should bring you to Scoreboard State.
Main Game State: Initial	After Loading Main Game state by clicking Start Button	Should display background, 4 bins, and brief instructions
Main Game State: Main gameplay	Click anywhere on screen, click and hold on item, then drag to a correct bin.	After initial click, should show countdown of 3 seconds before starting. After an item should show up with a timer increasing in time along with a score beside the timer. After dragging item into correct bin, an animation should play and a new item should appear. The score should increment by one.
Main Game State: Main gameplay 1	Click anywhere on screen, click and hold on item, then drag to an incorrect bin.	After initial click, should show countdown of 3 seconds before starting. After an item should show up with a timer increasing in time along with a score beside the timer. After dragging item into incorrect bin, an animation should play and the same item should return to the starting location. The score should not increment.
Main Game State: Main gameplay finish	Repeat steps in Main Game State: Main gameplay 1. Reach max score (15).	Should display Game Over screen. Items should stop spawning. Timer should stop.
Scoreboard State: Before game	Enter Scoreboard State after clicking scoreboard button,	The Scoreboard State should display the top 10 scores with the user's submitted name.

		There also should be a submit and start button. The personal score displayed should be 9999.99.
Scoreboard State: After game	Enter Scoreboard State after clicking scoreboard button,	The Scoreboard State should display the top 10 scores with the user's submitted name. There also should be a submit and start button. The personal score displayed should be the time gotten after the finishing one game.
Scoreboard State: Submit	In Scoreboard State, click the Submit button. Enter a name and click OK.	After clicking the Submit button, a window should pop up and ask you to enter your name. After entering name and click OK. The top 10 Scoreboard List should update with the updated highscores.

4 REQUIREMENT 4

4.1 Conclusion

The accuracy and database access user interface (UI) were manually tested. The accuracy data has been entered in the accuracy UI, and updated on the monitor as “yesterday’s accuracy” for each bin. The database access UI was also manually tested by selecting a range of dates and bin colour to download data. The created CSV file, containing the data pulled from the database, was opened and found to be in the correct format.

4.2 Test Cases

Accuracy UI Test Cases		
Test cases	Steps	Desired Result
Insert Accuracy Data	User picks a day, and enters a percentage for the accuracy data on the UI. The submission is inserted into the database. The entry contains the date and accuracy percentage.	The database should contain the entry
Update Accuracy on the Monitor	Pulls accuracy data of previous day, and updates “Yesterday’s Accuracy” on the monitor.	The database should contain the entry, and the percentage should be shown on the monitor.

Client UI Test Cases		
Test cases	Steps	Desired Result
Pull Requested Data from the Database	User enters a start date, end date, bin location, and bin color into the corresponding entry forms. Then, user presses “Submit”. The entered data is used, in a query, to pull data from the	The pulled data should be entries starting from the start date, until the end date. It should be from the specified location and bin color. Each entry should contain a date, time, bin location, and bin

	database.	color.
Create CSV File	Uses the pulled data from the previous test case. First, a new CSV file is created, and named with the data the user has entered. Then all the data is entered into a CSV file.	A new CSV file that should contain all the entries from the start to end date, and from the specified location and bin color.
Send File to User	User presses “Download”, and a download window pops up, allowing them to save the new CSV file onto their computer.	The CSV file downloaded should be the newly created CSV file from the previous step.

5 REQUIREMENT 5

5.1 Conclusion

The project is maintainable. The code was designed to be modular, so any module can easily be added, removed, or modified. All hardware components are inexpensive, and easy to obtain and replace. There was no custom hardware used in the project. The documentation of the test cases, software methods, and prototyped solutions are clear, and easy to understand. There are also step-by-step instructions on how to start the system, how to use Amazon web services, how to use both of the user interfaces and how to connect to the remote database. The instructions are clear enough so that another technological individual can resume the project with ease.