“Phil”

By *DrinkTank*

Aaron Riggs, Andre Manz, Chris Baierski, Wesley Scott

**1 - Project Overview**

**1.1 - Definition and Motivation**

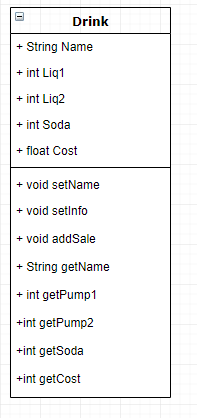
During the 2017 fiscal year the liquor industry did over $700 Billion in sales with roughly 20% in loss due to over pouring, theft and free drinks. Shrinkage accounts for $10 billion in losses in 2017 for bars in the US. When it comes to any kind of beverage, the amount being poured needs to be exact. Most cases of over pouring are unintentional. Eyeballing a glass of wine or counting out shots for a cocktail are just two examples. Even when performed by highly skilled, honest servers, free pouring of alcohol is questionable. With cocktails the amount of liquor in each glass is typically so small that over pouring even a little will double the cost of that serving. A 1oz overpour will lose you 50% of the potential revenue from a bottle of liquor. Cocktails are unique because the liquor is often poured over ice and mixed with other ingredients, making it even more difficult to accurately freepour. You can’t see exactly how much liquor is going in. Liquor has a higher profit margin than beer or wine. The cost of a bottle of vodka can range from $27 for 33.8 oz. on the lower end to $20 for 24.5 oz. for more premium options. The goal of this project is to make these numbers shrink for the overpuring of cocktails and mixed drinks, this would save bars all across the United States a lot of money in the long run. If only a 20% improvement were made then that would be another $2Billion dollars that would go into the revenue of these businesses. This device “Phil” could make the current way that bartenders pour the simple drinks a thing of the past, “Phil” would save bar allot of time as well in pouring simple drinks so that the bartenders can focus on the more complicated pours. If you wanted to take “Phil” out to a wedding or party of some kind it would be easy to transport “Phil” with only a little bit of breakdown and setup.

**2 - Design Details**

**2.1 - Class Diagrams**

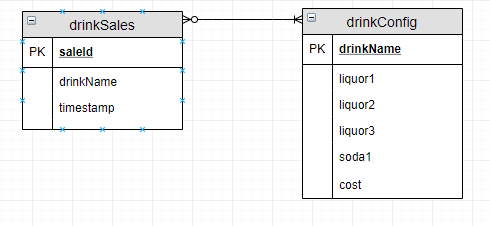
2.1.1 Drink

Due to the nature of our hardware/software setup, we don’t have much in the way of traditional “classes”. The one that we do have, “Drink” is what is being poured by PHIL, and consists of the Drink that is poured when a button is clicked on the GUI, as well as its configuration of pumps, and costs that are stored in the logs file.



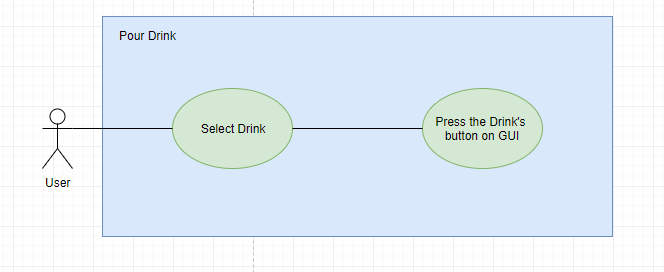
2.1.2 ER Database Diagram

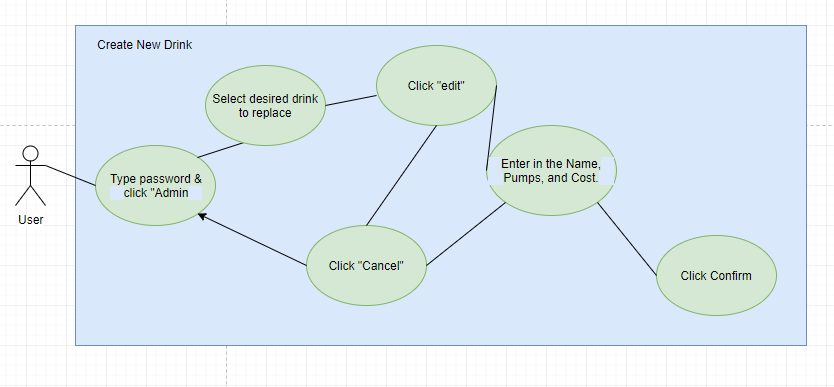
We will also be including our ER diagram for our database in this section. The below image will show the relations of our database.

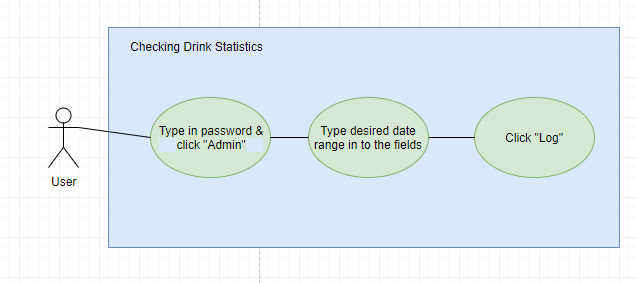


**2.2 - Use Case Diagrams**

The following image shows our 3 primary Use Case Diagrams, updated from our original plans. The only real major changes are that each drink button has it’s very own “edit” button, allowing for the modification of the drink configuration dynamically.

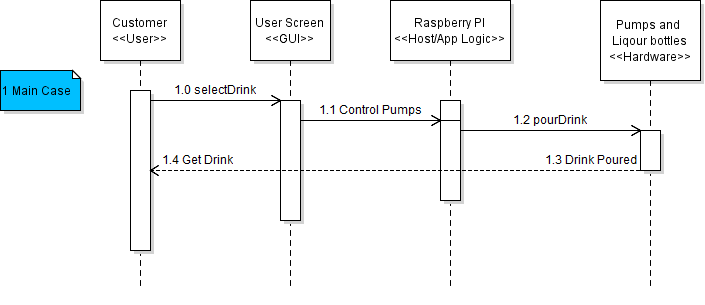


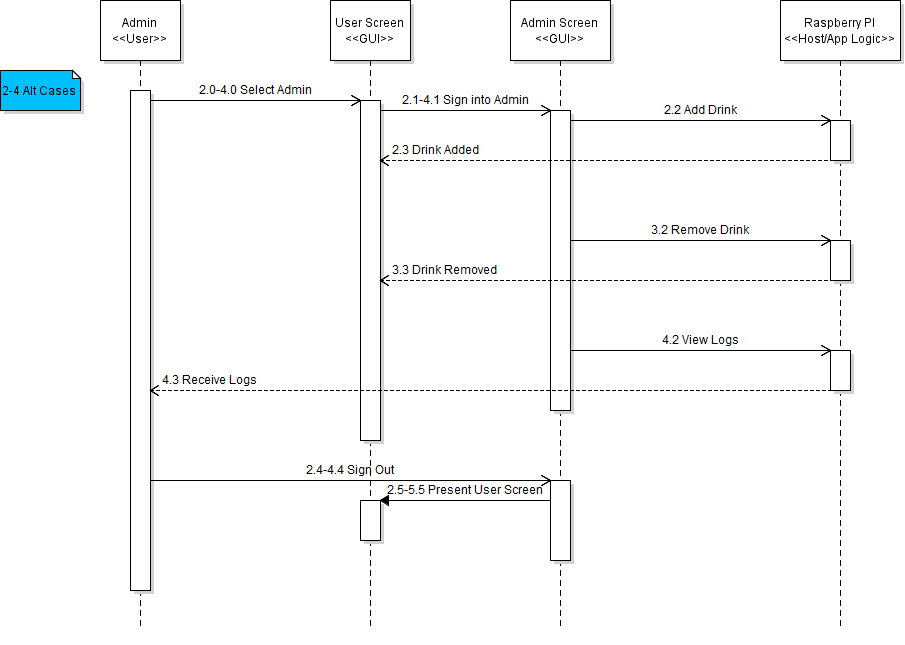




**2.3 - Sequence Diagrams**

Our Sequence Diagrams actually did not change from our original planning phase. The following diagrams are split into the “Main Case” and “Alt Cases” for readability.





**3 - Implementation Details**

**3.1 - Implementation Overview**

Our implementation of PHIL came along very well this semester. We successfully built a functional system that satisfied all of our Functional and Non-Functional Requirements. PHIL can pour drinks accurately, and is user-friendly, allowing it to complete and solve our problem definition and motivation.

**3.2 - Implementation Process**

Our main philosophy while building PHIL was to make him modular. This allows us to overhaul and greatly improve one aspect of PHIL, without interrupting any of the core functionalities. In order to keep development on track, we met regularly to make sure each of the core components were able to integrate and work properly together.

3.2.1 Hardware Implementation

The hardware we didn’t already have was purchased very quickly in the semester. The first thing we did during this project was to assemble and test the hardware. We set up the Raspberry Pi, Arduino, Relays, and Power Supply, and pumps right away to ensure that our project didn’t have any integral flaws. Multiple iterations of the GUI were constantly being tested with the hardware.

3.2.2 GUI Implementation

The GUI was developed in Python version 3.7.1, using the native Tkinter graphical user interface package. The first iteration of the of the GUI was simply 3 buttons corresponding to the 3 different pumps to ensure that we could correctly manage the hardware components from our program. Version 1.0 of the GUI was completed about halfway through the semester, and allowed for the pouring of 3 static drinks that could be renamed. Version 2.0 of the GUI was completed a week or two later, and brought more dynamic drink creation/modification functionality. It also was tested and developed with database stat logging in mind.

3.2.3 Database/Logging Implementation

The database was designed, and the tables were created fairly early on. However, stat logging and integration with the rest of the project didn’t begin until version 2.0 of the GUI began development. During each iteration of both the logs and the GUI, compatibility between the two were tested to ensure successful integration.

3.2.4 MyCroft AI Implementation

This was the only part of the project that didn’t really fully rely on other components. We added MyCroft implementation about halfway through the project. While voice recognition was unsuccessful due to microphone driver issues, we were able to implement a proof of concept chatbot. Mycroft was installed and set up using the Linux Command line using the following commands:

cd ~/  
git clone https://github.com/MycroftAI/mycroft-core.git  
cd mycroft-core  
bash dev\_setup.sh

./start-mycroft.sh all

./start-mycroft.sh cli

Mycroft was then instructed to install the relevant skills using the text interpreter. This was used to install the Cocktails Skill. Due to the missing files in the Mycroft SDK, we were unable to create a custom skill to allow Mycroft to run the functions that activated the pumps.

**3.3 - Completed Functionalities**

Development was fairly smooth and went well. Because of this, we were able to complete just about all of our projected functionalities, and were even able to make progress on some of our stretch goals.

3.3.1 Completed Hardware

Full hardware setup and functionality was completed. Drinks can be poured in defined ratios with only a +- 5% margin of error.

3.3.2 Completed GUI

Full GUI functionality was completed. Users can pour drinks using buttons. Users can sign in with an admin password to edit drinks. Users can edit drinks, modifying the drink name, which pumps run (and for how long), and how much the drink costs. Finally, users can specify a date range to download a stat/log file.

3.3.3 Completed Database and Logging

Full database and logging functionality was completed. Once the file has been downloaded (see 3.3.2), the user can open it in Microsoft Excel, MacOS’ Numbers, or Google Sheets. The file will contain all the drinks poured in the specified date range with date and time poured, name of drink, and cost of drink.

3.3.4 Completed MyCroft AI (Chatbot Proof-of-Concept)

A proof of concept chatbot using MyCroft AI was completed. Users can utilize the AI as a substitute bartender, by asking it conversational questions. You can also ask how to make specific drinks, for which it will tell you the list of ingredients, as well as describe the process for making the drink.

**3.4 - Incomplete Functionalities**

As mentioned above, there wasn’t much we left incomplete during the course of this project. If we had more time and budget, we would have made more quality-of-life improvements. Anything not mentioned in the subsections below are fully functional.

3.4.1 Incomplete Hardware

While our hardware was fully functional, we were not able to get it in a portable case. While this didn’t affect our pouring functionality, it did result in a non-portable version of our project. Given more resources, this could have been completed, and was not the result of a design or implementation flaw.

3.4.2 Incomplete MyCroft AI (Voice Recognition)

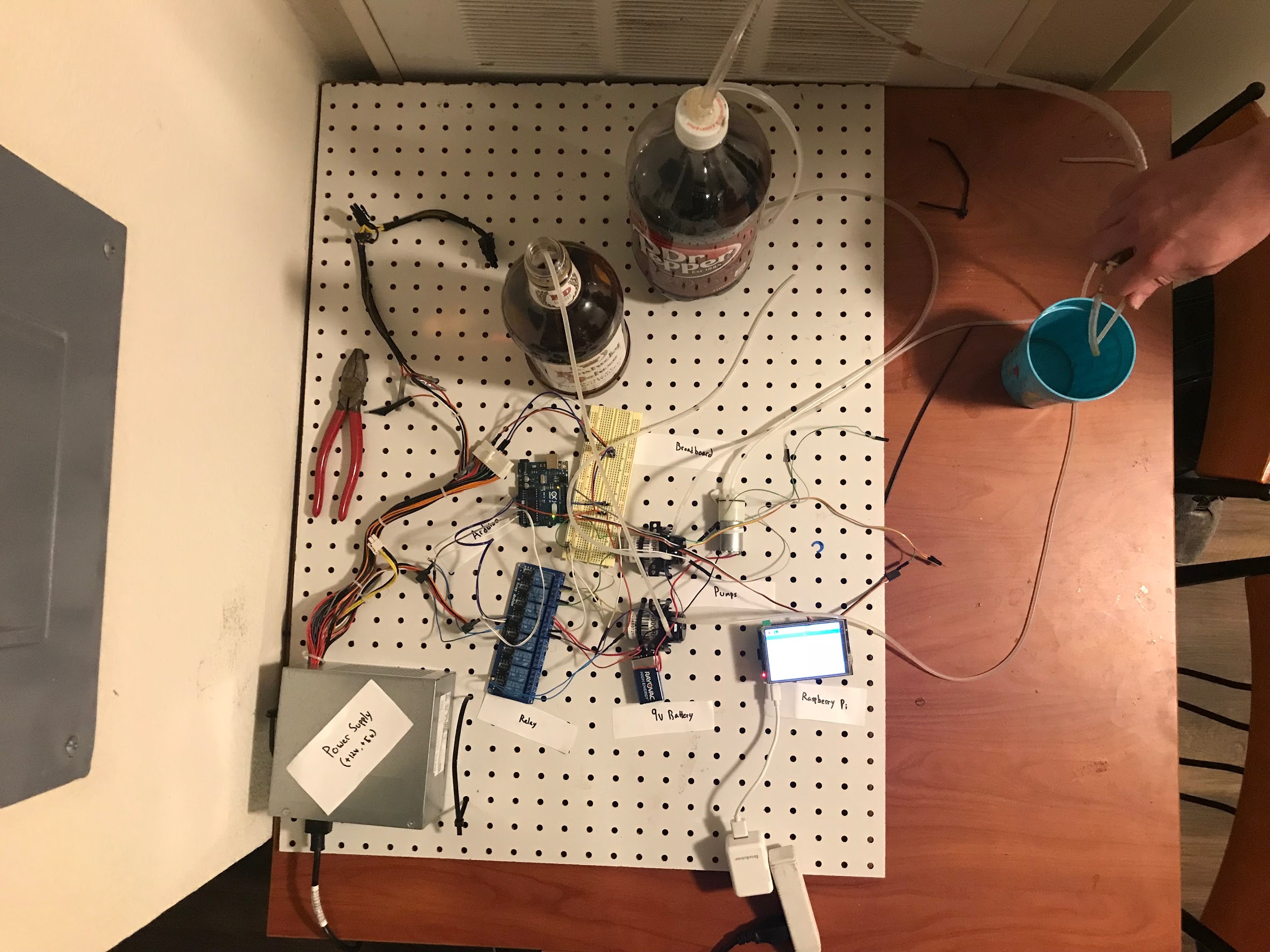
Originally intended to have voice recognition capabilities, issues with bluetooth and microphone drivers prevented us from this goal. Instead, we implemented a chatbot that you could ask conversational and drink-related questions as a proof of concept. \*Note that voice recognition was not an original functionality goal, but was included about halfway through the semester since we had most of our other functionality progressing well.

**4 - Full Demo**

**4.1 - Hardware Pictures**

The following images depict our hardware layout. In the pictures, you can see all the major hardware components with labels:

* Raspberry Pi
  + Used to compute business logic and house GUI (with touchscreen)
* Arduino
  + Used to translate GUI signals to hardware signals
* Relay
  + Used to normalize voltages traveling between hardware components
* Breadboard
  + Used for power distribution
* Power Supply
  + From an old computer, this supplies the power for our hardware components
* Pumps and Hoses
  + Mechanisms for actually pouring the drinks from their bottles





**4.2 - Video of GUI 2.0 working**

Shown during our meeting time, we presented a two-minute long video of us pouring two different drinks back-to-back using version 2.0 of the GUI. We demonstrated how using the admin and edit buttons, a manager could successfully change the type of drink poured by a single button. Upon request of Dr. Lim, a link to download this video will be sent along with this report.

**4.3 - Hands-on demo of GUI 2.0 working on computer**

After showing the video, we pulled up the GUI running on a Surface template. We then walked through all of our front-end functionality, while any actions that would have been handled by the pumps and hoses being printed to the live console. We showed:

* That the pouring of different drinks led to different pumps being activated
* That you can sign in to the admin section via password
* That you can successfully edit a drink
* That you can successfully pour a new drink
* That you can enter in a date range, and download a stat’s log file
* That the log downloaded successfully
* That the log showed the following information of drinks poured in the date range selected:
  + Date and Time poured
  + Drink Name
  + User inputted cost of the drink

**4.4 - Database, Middleware, Drink Logging**

**4.4.1 - Database**

The database is hosted with SQLite3. The RDMS is perfect for this scenario as it is light-weight and does not put a heavy load on the RPi resources. In order to initialize the database:

$ sqlite3 phil.db

At which point, the two tables (drinkConfig, drinkSales) are added with the usual CREATE TABLE command based off the following schema:

drinkConfig(drinkName: VARCHAR(255), liquor1: INTEGER, liquor2: INTEGER, soda1: INTEGER, cost: DECIMAL(2, 2))

drinkSales(saleId: INTEGER, drinkName: VARCHAR(255), timestamp: DATETIME)

**4.4.2 - Middleware**

The Python package, sqlite3, is very useful for interfacing between the GUI and database. In order to open a connection and create a cursor to execute queries, add the following:

conn = sqlite3.connect(‘phil.db’)

curs = conn.cursor()

The two main functions for querying the database are createDrink() and addSale(). createDrink() adds a new drink configuration to the drinkConfig table if the user inputted configuration does not already exist in the database, otherwise the selected on-screen drink button will be updated without hitting the database. addSale() simply adds the drink poured and a timestamp to the drinkSales table each time a drink is successfully poured. The two queries’ formats are depicted below:

INSERT INTO drinkConfig(drinkName, liquor1, liquor2, soda1, cost)

VALUES(‘drink’, int(0-9), int(0-9), int(0-9), decimal(00.00))

INSERT INTO drinkSales(drinkName)

VALUES(VARCHAR(255))

**4.4.3 - Logging**

Phil’s logging functionality is based around the Python os package. Phil creates a file called ‘log.csv’ and writes each drink, cost of that drink, time the drink was poured, and the total profit made off of those drinks in a specified date range. This is done simply by joining both the drinkConfig and drinkSales tables on the drinkName attribute and writing each row returned to ‘log.csv’. This is accomplished with the following\*:

|  |
| --- |
| with open('./log.csv', 'w+') as write\_file:  write\_file.write("Time of Sale, Drink Sold, Sale Price \n")  for row in curs.execute("SELECT ds.timestamp, dc.drinkName, dc.cost FROM drinkSales AS ds, drinkConfig AS dc WHERE ds.drinkName = dc.drinkName AND timestamp BETWEEN ? AND ?", (startDate, endDate)):  write\_file.write(str(row[0]) + "," + str(row[1]) + "," + str(row[2]) + "\n")  curs.execute("SELECT SUM(dc.cost) FROM drinkSales AS ds, drinkConfig AS dc WHERE ds.drinkName = dc.drinkName AND timestamp BETWEEN ? AND ?", (startDate, endDate))  total = str(curs.fetchone()[0])  write\_file.write("," + "TOTAL = ," + total) |

**\***note that the formatting is not quite pythonic, due to width of the queries.

**4.5 - Video of MyCroft AI Chatbot**

Before entering the Q&A section of the demo, we showed off what we were able to finish of the MyCroft AI. Meant to be a voice recognition AI akin to Siri, Google Home, or Alexa, we had issues with the microphone drivers. So as a proof of concept, we showed a video of us successfully typing conversations with the AI bot. We demonstrated that the chat bot could answer basic conversational questions such as the following:

* What is your name?
* How is the weather?
* What can you do?
* How are you?

We also then demonstrated it had functionality specific to our project, by asking it how to make certain cocktails, and it would answer with both the list of ingredients, as well as the instructions for making the drink.