

**Senior Design Project**

**Design of An Automated Drink Dispensing System**

ENGE476 Senior Design Project I

Department of Engineering and Aviation Sciences

University of Maryland, Eastern Shore

Nathan Bane

Chris Blanks

Ryan Valente

Project Advisor, Dr. Lei Zhang

Submitted

Dec. 7, 2018

List of Contents

[1. Introduction 8](#_Toc6782498)

[1.1. Backgound/Motivation 8](#_Toc6782499)

[1.2. Objective 10](#_Toc6782500)

[1.3. Design Requirements 10](#_Toc6782501)

[1.4. Design Constraints 10](#_Toc6782502)

[1.5. Design Methods 10](#_Toc6782503)

[2. Project Description 11](#_Toc6782504)

[2.1. System Description 11](#_Toc6782505)

[2.2. System Diagram 12](#_Toc6782506)

[2.3. System Functions 13](#_Toc6782507)

[3. Implementation Plan 14](#_Toc6782508)

[3.1. Tasks 14](#_Toc6782509)

[3.2. Team Organization 15](#_Toc6782510)

[3.2.1. Responsibility of Nathan. 15](#_Toc6782511)

[3.2.2. Responsibility of Chris. 15](#_Toc6782512)

[3.3. Timeline/Milestones/Delivery Plan 15](#_Toc6782513)

[4. Implementation 17](#_Toc6782514)

[4.1. Implementation of Task 1. 17](#_Toc6782515)

[4.2. Implementation of Task 3 25](#_Toc6782516)

[4.3. Implementation of Task 4. 26](#_Toc6782517)

[4.4. Implementation of Task 5. 42](#_Toc6782518)

[4.5. Implementation of Task 6 48](#_Toc6782519)

[5. Conclusion 52](#_Toc6782520)

[Appendix 54](#_Toc6782521)

[A. Component Specs 54](#_Toc6782522)

[1. Specs of the Raspberry Pi 3 Model B 54](#_Toc6782523)

[2. Specs of the Embedded Board 55](#_Toc6782524)

[B. Industry Standards 55](#_Toc6782525)

[C. Source Code. 56](#_Toc6782526)

[1. Source Code of the Desktop Application 56](#_Toc6782527)

[2. Source Code of the Phone Application 98](#_Toc6782528)

[References 127](#_Toc6782529)

List of Figures

[Fig. 1. Embedded System Diagram: Main Computer, Hardware Controller 12](#_Toc6782441)

[Fig. 2. Full-System Finite State Machine 13](#_Toc6782442)

[Fig. 3. Fluid Control Microcontroller 17](#_Toc6782443)

[Fig. 4. High-Side Switch (ON Semiconductor, 2014) 18](#_Toc6782444)

[Fig. 5. Schematic for High-Side Switch 18](#_Toc6782445)

[Fig. 6. High-Side Switch Simulation 19](#_Toc6782446)

[Fig. 7. Full H Bridge Typography (Texas Instruments, 2016) 20](#_Toc6782447)

[Fig. 8. Motor Driver Logic Table (Texas Instruments, 2016) 20](#_Toc6782448)

[Fig. 9. Motor Driver Schematic 21](#_Toc6782449)

[Fig. 10. I2C GPIO Expander and 5V Regulator 22](#_Toc6782450)

[Fig. 11. I2C Thermometer 23](#_Toc6782451)

[Fig. 12. I2C Input Current Sensor 24](#_Toc6782452)

[Fig. 13. PCB Layer Stack-Up (Altium, 2017) 24](#_Toc6782453)

[Fig. 14. Grid Geometry Example 26](#_Toc6782454)

[Fig. 15. MVC Pattern 27](#_Toc6782455)

[Fig. 16. Class Diagram of Desktop Application 27](#_Toc6782456)

[Fig. 17. Root Window with Mode Selection Buttons 28](#_Toc6782457)

[Fig. 18. Main Window of Customer Window 28](#_Toc6782458)

[Fig. 19. Order Page for Customer 29](#_Toc6782459)

[Fig. 20. Order Confirmation box 29](#_Toc6782460)

[Fig. 21. Login Window 30](#_Toc6782461)

[Fig. 22. Admin Options Menu Bar 31](#_Toc6782462)

[Fig. 23. CRUD Principle 31](#_Toc6782463)

[Fig. 24. DrinkProfileManager Window 32](#_Toc6782464)

[Fig. 25. Drink Editor Window 32](#_Toc6782465)

[Fig. 26. Log Window with Clear Option 33](#_Toc6782466)

[Fig. 27. ApplicationUserEditor Window 33](#_Toc6782467)

[Fig. 28. “Add a User” Window 34](#_Toc6782468)

[Fig. 29. “Delete a User” Window 34](#_Toc6782469)

[Fig. 30. IP Address Window 35](#_Toc6782470)

[Fig. 31. “Help” Menu Bar Options 35](#_Toc6782471)

[Fig. 32. How to Operate Instructions Window for Customer 36](#_Toc6782472)

[Fig. 33. Contributors Window 36](#_Toc6782473)

[Fig. 34. Log Manager 37](#_Toc6782474)

[Fig. 35. Parent and Child Directories 37](#_Toc6782475)

[Fig. 36. Example of the Text File Format of Each Drink Profile 38](#_Toc6782476)

[Fig. 37. Shared Data File Format 39](#_Toc6782477)

[Fig. 38. PeripheralDevice Inheritance Diagram 40](#_Toc6782478)

[Fig. 39. MEI Bill Acceptor 40](#_Toc6782479)

[Fig. 40. MEI Bill Acceptor Interface Schematic 41](#_Toc6782480)

[Fig. 41. 3D Printed Picamera holder 41](#_Toc6782481)

[Fig. 42. Phone app Icon with Notification Bubble 42](#_Toc6782482)

[Fig. 43. Login Page of Phone App 43](#_Toc6782483)

[Fig. 44. Tabbed View of Main Menu 44](#_Toc6782484)

[Fig. 45. URL Settings Options 45](#_Toc6782485)

[Fig. 46. URL Editor Layout 45](#_Toc6782486)

[Fig. 47. Inventory Fragment View 46](#_Toc6782487)

[Fig. 48. ADDS Phone App Notification 47](#_Toc6782488)

[Fig. 49. Sales Fragement View 47](#_Toc6782489)

[Fig. 50. System Status Fragment View 48](#_Toc6782490)

[Fig. 51. Bunny Ears Filter 49](#_Toc6782491)

[Fig. 52. Bunny Ears Math 49](#_Toc6782492)

[Fig. 53. Rainbow Mask Filter 50](#_Toc6782493)

[Fig. 54. Rainbow Glasses Filter 50](#_Toc6782494)

[Fig. 55. Desktop Application Package on PYPI 51](#_Toc6782495)

[Fig. 56. Raspberry Pi 3 Model B Specifications 54](#_Toc6782496)

[Fig. 57. Raspberry Pi 3 Model B General Purpose Input & Output Pins 54](#_Toc6782497)

List of Tables

[Table 1. Project Timeline and Delivery Plan 14](#_Toc531700688)

[Table 2. Project Timeline and Delivery Plan (Continued) 15](#_Toc531700689)

Abstract

With the advent of touchscreen appliances, businesses have been integrating new technology into their establishments in order to increase the productivity of their workers, reduce costly long-term expenses, and stay competitive with other businesses who are adopting other modern technologies. Within the current drink making appliances market, there is a growing need for this kind of technology. This paper proposes a solution for addressing the issue of long wait times for service and delivery of beverages in establishments that serve alcohol. Herein the design of an embedded system at the hardware and software level is proposed in order to solve this issue in a cost effective and efficient manner.

**Keywords**— Automation, Embedded, Electronics, Software, Image Processing, I2C, Motors, Dispensing

1. Introduction

Over the course of the project the team will detail the development of an automated drink making machine that will alleviate wait times in bars, allow users to pick complicated drinks, and facilitate a good buying experience for the customers.

* 1. Backgound/Motivation

In most popular bars, there is always the issue of long lines of customers and overworked bartenders. Why does this usually occur? The ratio of bartenders to customers is usually very small, so there is more demand than supply. This issue can cause a few problems for the bartenders and customers. If the bartenders rush to meet the demand of the customers then that will liken the chance of errors being made in the drink making process, which will lower the quality of the drinks and cause customers to become unsatisfied with the service. If the bartenders are too slow to meet the demand, then some customers will become irritated due to the long wait time. When wait times get too long, customers tend to migrate to other bars that are not as busy. This is very disadvantageous for the bar owner and bartender because that means less revenue. Limiting the number of customers in one period or space is an option, but that does not maximize on the potential profit.

One solution that some bar owners take is to just hire more staff members, but this is not always feasible due to limited budgets and lack of space. Additionally, you cannot always predict when a large wave of people will come to the bar, so it is better to have a more flexible solution that takes up a very small amount of space, does not cost thousands of dollars, and can be reliable at any moment. The perfect solution to help bar owners and bartenders is an automated machine that meets the previously listed criteria. This machine would be able to work at any moment of the day and make bartenders more productive.

After doing some market research on the usefulness of an automated drink making machine, the machine must be easily customizable and intuitive to use. When the machine is easily customizable, a bar owner can make more use of the machine and cater to multiple groups of people. For example, if the bar owner can add or exchange the drinks in the machine easily then the owner can meet the needs of the customers a lot faster and enhance their experiences. An intuitive interface is a very important feature of any product. When it is simple to use, it will be used a lot more than if it was convoluted. In addition, if the bar employees have a hard time trying to figure out how to use the machine then that makes them less productive, which means less revenue. Ideally, any bar owner would want a machine that would augment the abilities of their workers.

A couple other considerations for an automated drinking machine are the payment process and the issue of people that have had too many drinks. Traditionally, most customers to bars will either pay upfront for their drinks or will ask the bartender to open a tab until they are done. Sometimes customers drink too much, so they are cut off from buying any more drinks. This is a task that is usually left to the bartender’s discretion. An automated machine can help with these two common tasks. The automated machine could handle cash, credit cards, and electronic payments (e.g. PayPal, Venmo, Bitcoin). This could potentially allow more money to be spent at the bar. With the automated machine handling drinks, the drink making process would be objective and customers would get exactly what they bought. The issue of preventing customers from drinking too much is also an achievable task for an automated machine. The machine could incorporate a system for detecting when a customer is past an acceptable level of drunkenness. This could be done with the use of a facial recognition camera that monitors the facial expressions of recurring customers over the night and decides whether they should have another drink within a certain time frame. Another option is using an infrared camera to do a similar job as the facial recognition camera. Regardless of the method, the maker of an automated drink making machine must take these two issues into consideration.

There a few solutions on the market already for automating the drink making process. The first solution used the layout of a 2-D plotter to design their automated drink making machine. The alcoholic beverages are fastened in their own slots and with their bottoms upward. The alcohol is dispensed by utilizing 25 ml Beaumont Taps. A servo actuates these taps and dispenses the alcohol into a cup that is attached to a linear actuator. The cup is then moved to the next drink dispensing location for the next ingredient for a mixture. Once the mixture has been finished, the patron picks up the cup from the machine. There are also lights that are integrated into the design that make their drink making process more entertaining. The user interface is just an array of buttons and a Liquid Crystal Display (LCD) screen.

The second solution utilizes two industrial robotic arms to mix a wide range of drinks. The robotic arms are located on a platform about the height of a counter, and the drinks are suspended above the robot platform. It uses the common utensils that a standard bartender uses. This allows the robot to make any drink that a bartender can make. The robot arms are also capable muddling, stirring, shaking, and straining actions, which are common for drink making. This is a very extendable design because the robots are dexterous enough to perform many actions, so this solution can make a lot of drinks. It can make two drinks per minute, which yields around 1,000 drinks per day.

Both solutions can create alcoholic drinks and allow patrons easy access to drinks. However, the first solution has a slow drink making process. Each pour the machine makes is a pre-calibrated pour that cannot be changed. As a result, it can only mix imprecise volumes. In addition, the machine’s unwieldy size is not suitable for most bar applications. The electronics and mechanism are open to the patrons allowing them to potentially damage the machine. Another con is that it could expose drinks to impurities before the customer gets the drink. The machine also doesn’t dispense ice or other ingredients, so only basic drinks can be made. The second solution would be very expensive and too large for most bar owners. It also requires a lot of overhead to operate. The industrial robot arms are heavy pieces of equipment which will require special structures to support. It will also have high power requirements necessitating the need for a specific electric service that is not always available. These specific needs require a huge investment to install. This puts it out of reach for most bar owners.

The designing and building of an efficient automated drink making machine is no easy task due to all the considerations to keep in mind. It requires keeping the bar owner, bartender, and customer happy with the product. It requires a flexible and extensible design, so that it can be appealing to large groups of people. It is a suitable engineering challenge to undertake.

* 1. Objective

Design an automated drink making machine that will offer a more economical solution to reduce wait times for drinks, create drinks, and reduce the human error in the drink making process.

* 1. Design Requirements

1. The machine will fit in a 3’ x 2’ x 3’ (L x W x H) space.
2. The dry weight, the machine without ice or fluids, of the machine will be less than 50 lbs.
3. The machine will be able to serve at least 10 different drinks due to the combinations of 4 liquors, 1 liqueur, and 7 flavoring additives.
4. The machine will feature an easy to use graphical user interface allowing the patrons to select various drinks for purchase.
5. The machine will be able to be powered from 120/220 V outlets.
6. The machine will be capable of operating without the main computer.
7. The phone application will allow store managers to remotely monitor the machine and view drink analytics.
8. Shared data will be updated every minute via a HTTP server.
9. This machine must securely store profits and information from users.
   1. Design Constraints
10. The machine must conform to machine food and safety requirements.
11. The machine must be on a raised, supporting platform in order to make the touchscreen close to the eye level of users.
    1. Design Methods

The first step is to design the board for the embedded system section. The second step is to design a computer application for the main computer. The third step is to write code to interface all the peripherals with the main computer and embedded board. The fourth step is to design a phone application that can pull system data from the main computer. The fifth step is to identify a power supply that will support the whole machine. The sixth step is to implement image recognition using the camera and main computer. The final step is to use a Field Programmable Gate Array to create extra features for the machine. After these steps are completed, our project will be done.

1. Project Description
   1. System Description

This system is composed of three main systems: the embedded electronics, the desktop application, and the phone application.

The embedded electronics can be broken down into two sections: the hardware controller board and the main computer. The hardware controller board will use a microcontroller to control fluid flow throughout the system via controlling valves, dc brushed motors, and peristaltic pumps. An additional motor controller will be used to dispense ice from the ice container. An onboard sensor module will be used to gather data regarding the operation of the machine. The main computer will host the touchscreen application that customers will use in order to select and purchase drinks from the system. The main computer will also interface with the payment collector, Field Programmable Gate Array (FPGA), and camera. The payment collector will send signals to the main computer when payment has been received. The FPGA will control extra features (e.g. LED light show, security). The camera can potentially be used to check driver licenses and apply amusing filters to the customer’s face (for entertainment).

The desktop application will control the states of the system (e.g. wait, dispensing, cleaning) through a series of commands sent to peripheral devices. It will be an application made with the Python 3 programming language and its supporting libraries. The app will have a graphical user interface (GUI) that will contain buttons, images, and dropdown menus. The customer will interact with the application through the touchscreen. Information regarding the inventory, purchases/sales, and customers will be stored in the local file system of the main computer.

The phone application will be used for remote monitoring and configuring of the system. The app will be created in Android Studio using the Java programming language and will be available for any android phone. The application will consist of a main menu, buttons, dropdowns, graphical displays, and images. For the app to acquire and display data about the whole system, it will have to use the HyperText Transfer Protocol (HTTP) to gain access to the file system of the main computer. In addition, the app will be able to change system configuration settings by editing a special file in the main computer.

* 1. System Diagram

The **Embedded System Overview** is divided into two sections the **Main Computer** and the **Hardware Controller**. The main computer provides and interface between the customer using a touchscreen and the hardware such as pumps, valves, and sensors. The main computer is a Raspberry Pi 3 a 1.2 GHz Linux capable computer. The customer’s order is placed on the tough screen and executed by sending I2C Commands to the hardware controller. It enables the two sections to accomplish different tasks at the same time.

The hardware controller completes the interface between the main computer using a STM32 microcontroller and the hardware. The onboard microcontroller receives its commands through the I2C Bus. Using clever programming, it will command its various subassemblies to actuate the motors, valves, and other hardware. The hardware controller has three main sections the **Sensor Module** which senses important data such as the security of the system through tamper alarms, the **Fluid Microcontroller** controls flow of fluid through the system, and the **Motor Control** section will oversee any additional motor control that is needed.

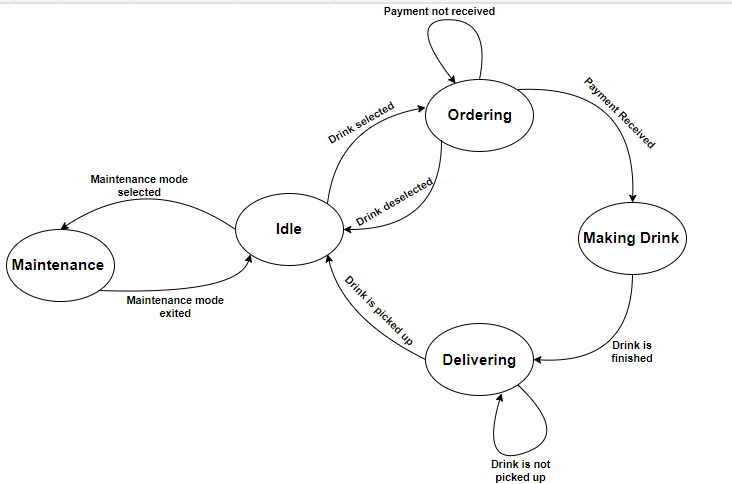


1. Embedded System Diagram: Main Computer, Hardware Controller
   1. System Functions

The full system will operate as a simple finite state machine. It will exist in the following states: idle, maintenance, ordering, drink making, and delivering. The following steps are how it will proceed through a successful drink making/delivery cycle:

1. The system will start in an idle state until a user selects a drink.
2. After a drink is selected, the system will migrate to the ordering state.
3. When the payment for the drink is received, the system will migrate to the drink making state.
4. When the drink finishes, the system will migrate to the delivery state.
5. After the drink is picked up by the user, the system will finish the cycle by migrating back to the idle state.

The system will also migrate to previous states for inverse actions performed by the user. For example, if the user deselects a drink while in the ordering state then the system will go back to the idle state. Also, the system will loop in the same state if certain actions are not performed. An example would be if the user does not pick up his or her drink while in the delivering state then the system will stay in the delivering state until the drink is picked up. Of course, these kinds of stalls will have to be handled in some fashion because stalls will be disadvantageous for the customer and owner of the machine. In addition, the system will offer a maintenance state that will allow the user to perform maintenance duties on the system and check the status of each device in the machine. Exiting the maintenance state will bring the system back to the idle mode.



1. Full-System Finite State Machine
2. Implementation Plan
   1. Tasks
3. **Design sub-assemblies for the embedded board**
   1. Design a High-Side Switch
   2. Design a Motor Driver
   3. Design a I2C GPIO Expander and Supporting Circuitry
   4. Design a I2C Thermometer and Supporting Circuitry
   5. Design an I2C Input Power Current Sensor
   6. Design a Printed Circuit Board
   7. Create a Bill of Materials, Order Components and Printed Circuit Board
   8. Assemble Components onto Printed Circuit Board
   9. Develop Test Code for the Printed Circuit Board to Test Assembly
4. **Select/Integrate Microcontroller**

Subtask 1. Select microcontroller and develop supporting circuitry for microcontroller.

Subtask 2. Select and design supporting circuitry for I2C EEPROM

Subtask 3. Develop circuitry to interface with a resistive touch screen

Subtask 4. Interface switches with microcontroller to manipulate modes of microcontroller

Subtask 5. Layout printed circuit board

Subtask 6. Assemble components onto printed circuit board

Subtask 7. Code Microcontroller to receive commands through the serial data line

1. **Develop a communication protocol between the main computer, and the embedded development board.**
2. **Create a main computer application**
   1. Design a GUI using Python’s Tkinter Library that the customers will use
   2. Develop an efficient way to organize monitoring/drink information in the file storage system
   3. Add in capabilities for serial communication between the main computer and peripheral devices.
   4. Interface the main computer with the embedded boards and peripherals.
3. **Create a phone application**
   1. Design the user interface of the phone app with Android Studio’s SDK
   2. Allow the user to update system information (e.g. drinks currently stored)
   3. Allow the phone app to access the system information stored on the main computer and acquire relevant data for use within the app
4. **Add extra capabilities and entertaining features**
   1. Incorporate facial recognition into the main application.
   2. Adapt desktop application for compatibility with PYPI.
   3. Team Organization
      1. Responsibility of Nathan.

Task 1, Task 2, Task 3

* + 1. Responsibility of Chris.

Task 3, Task 4, Task 5, Task 6

* 1. Timeline/Milestones/Delivery Plan

1. Project Timeline and Delivery Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Time | Task | | Comments | Responsible Personnel |
| Fall 2018 Semester | | | |  |
| Week 1-4 | N/A | | Project proposal and planning | Nathan Bane |
| Week 4-  Week 7 | Task 1 (subtask 1.1,1.2) | | Design and draw up a schematic for a high-side switch. Develop circuitry to control a brushed DC Motor . |
| Week 8-  Week 10 | Task 1(subtask 1.3-1.5) | | Design and layout circuitry to support an I2C Thermometer, Current Sense Circuitry, and I2C GPIO Expander |
| Week 11- Week 13 | Task 1(subtask 1.6-1.7) | | Layout the components on the Printed Circuit Board and order Printed Circuit Board with parts |
| Week 13 - End of Semester | Task 1(subtask 1.8-1.9) | | Solder components onto the printed circuit board and develop code to test the printed circuit board |
| Spring 2019 Semester | | | |  |
| Week 1-4 | Task 2(subtask 2.1-2.4) | Choose a suitable microcontroller and develop supporting circuitry. | | Nathan Bane |
| Week 5-  Week 8 | Task 2(subtask 2.5-2.6) | Layout printed circuit board, order components , and assemble components onto printed circuit board | | Nathan Bane |
| Week 9  -End of the semester | Task 2(subtask 2.4 ), Task 3 | Develop Code for Embedded. Develop communication scheme between master computer and fluid microcontroller | | Nathan Bane  & Christopher Blanks |

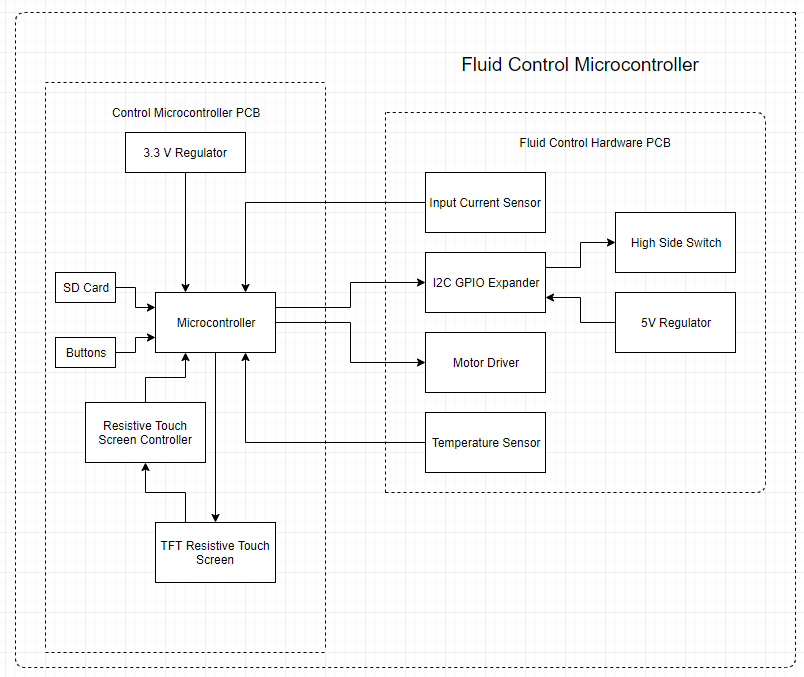
1. Project Timeline and Delivery Plan (Continued)

|  |  |  |  |
| --- | --- | --- | --- |
| Time | Task | Comments | Responsible Personnel |
| Fall 2018 Semester | | |  |
| Week 1-5 | N/A | Project proposal and planning | Chris Blanks |
| Week 6-  Week 11 | Task 4  (subtask 4.1,4.2, 4.3) | Design and create the main application. Test the inputs and outputs of the system, so that it will interface correctly with the embedded boards and other peripherals. |
| Week 12-  End of the semester | Task 5  (subtask 5.1, 5.2, 5.3) | Design and create the phone application. |
| Spring 2019 Semester | | |
| Week 1-6 | Task 6 (subtask 6.2) | Restructure desktop application software for compliance and compatibility with the Python Package Index |
| Week 3-  End of the semester | Task 4(subtask 4.4 ), Task 6 (subtask 6.1) | Finish interfacing and testing the main computer/application with the other systems. Incorporate facial recognition into the main application.  Documentation. |

1. Implementation
   1. Implementation of Task 1.

#### Design a Valve and Pump Control Board

The purpose of the valve and pump control board is to facilitate control of the hardware necessary to manipulate the flow of fluid through-out the system. It receives commands from the control microcontroller PCB through the I2C Bus and translates those commands into actions through various subassemblies. There are 5 main sub-assemblies: high-side switch, dc brushed motor driver, I2C GPIO expander, I2C thermometer, and input power current sensor. The high-side switch controls the valves directing the flow of fluid. Each subassembly plays an important role in the overall function of the system.



1. Fluid Control Microcontroller

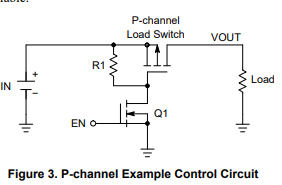
#### Subtask 1: High-Side Switch

The high-side switch will control the valves which directs the flow of liquid through the system. The high-side switch will enable the system to turn on and off specific valves. There will be three banks of 5 valves in the system. Four of the valves will be used to select which fluid will be pumped and the fifth valve will be used as a main on and off valve to cut off flow to that section of the system.

There are two main topographies of load switches that are used in modern circuitry: the high-side switch which utilizes a PMOS transistor to switch the load on and off and the low-side switch which utilizes the NMOS transistor to switch the load on and off. These switches control the flow of power through the system. They are often used in battery powered systems to turn off less used sections of the system to conserve power and then turn power on when that section is needed.

Utilizing the high-side switch has an advantage over the low-side switch. The high-side switch utilizing the PMOS power transistor facilitates the use of a simpler control block (ON Semiconductor, 2014). The control block involves shorting the gate of the PMOS power transistor to ground. It can be accomplished through the use of a logic level NMOS transistor whose gate is connected to the pin of a microcontroller.

The circuit topography of the high-side switch dictates that the switch is a normally open switch. A normal open switch requires power to close the switch connecting the power rail to the load. The control block of our circuit utilizes a microcontroller to output a logic level high to the gate of the NMOS transistor. The NMOS transistor conducts power shorting the gate of the PMOS power transistor to ground putting it a closed switch configuration connecting the power rail to the load.



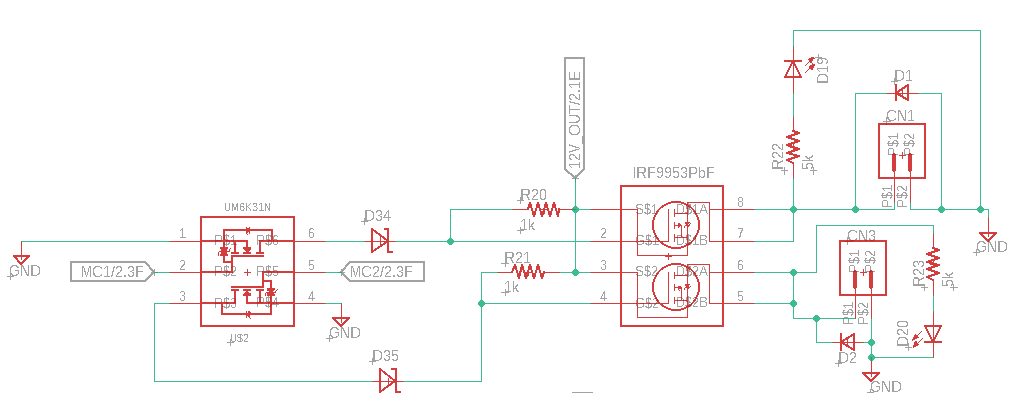
1. High-Side Switch (ON Semiconductor, 2014)

The high-side switch does have a disadvantage in comparison to the low-side switch. The low-side switch uses an NMOS power transistor to connect the power rail to the load. A determining factor in the power loss of a MOSFET is the internal resistance of the transistor when conducting power. Due to the construction of the NMOS transistor it has a naturally lower resistance. The lower resistance reduces the power lost due to high current flowing through the device. The high-side switch will have higher power loses in comparison to a similarly specified low-side switch (ON Semiconductor, 2014).

The requirements for our high-side switch was determined by the solenoid valve chosen to control the flow of liquid through the system. We chose values 150% of current requirements which means that the PMOS power transistor can handle current of up to .75 amps and 24V. The requirements allow the circuit to support any manufacturing imperfection in the resistance of the coil of the solenoid and reduce the chance that a random current spike could damage the components of the high-side switch.

After much research a load switch was chosen for this application. The International Rectifier IRF9953PBF is a dual packaged PMOS power transistor in a small outline integrated circuit (SOIC) package that can handle -30 VDSS, 1.8 A, -20 VGS, and -1 VTH (International Rectifier, 2004). The dual PMOS package was chosen in order to reduce the number of components needed to solder to the printed circuit board and save on space thus reducing the amount of board area needed for the components. It will reduce the number from 16 individual pass transistor into 8 dual package transistors.

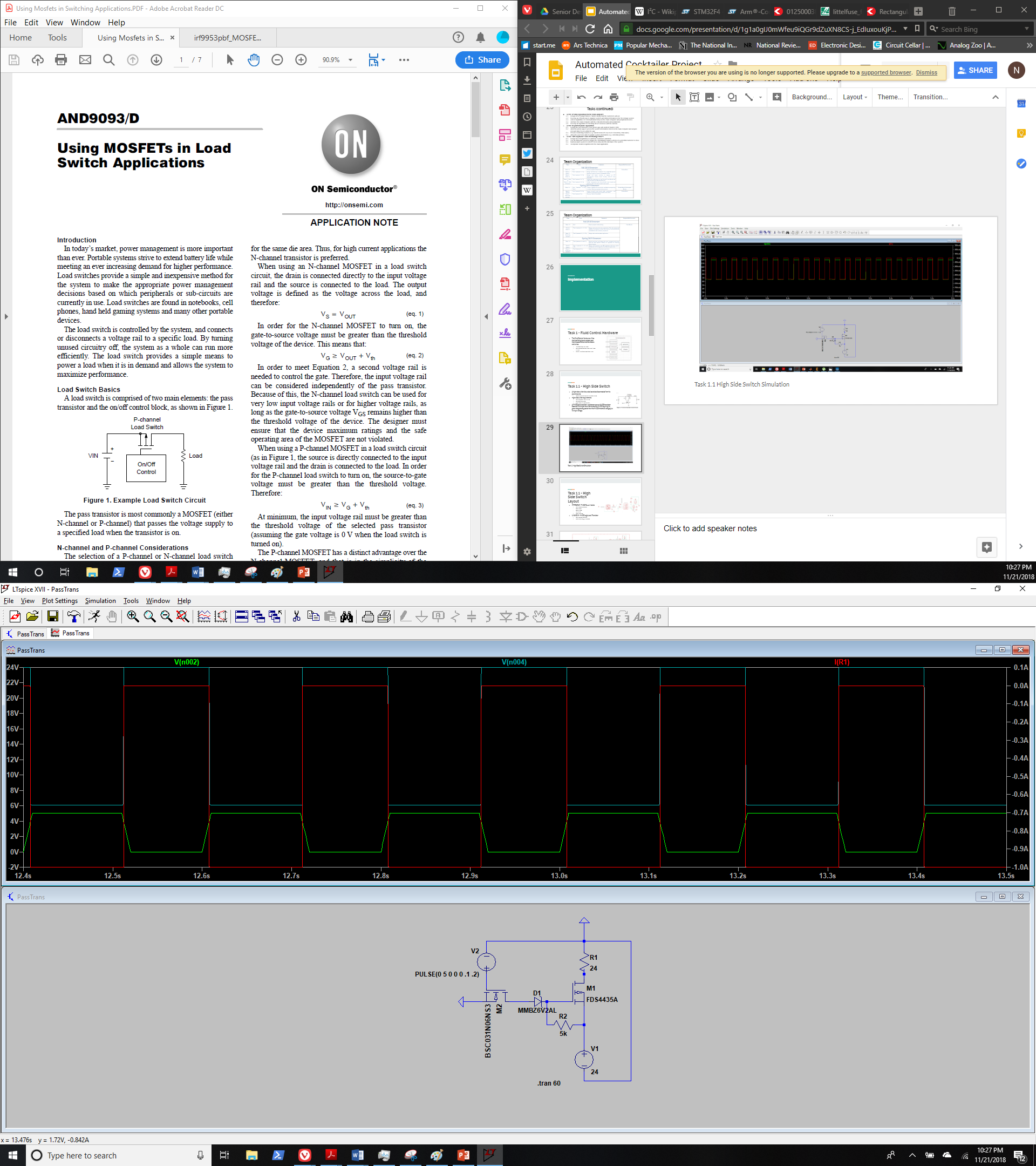
When the IRF9953PBF was originally chosen the voltage requirements of the system were only 12 V. The -20 volts maximum between the gate and the source (VGS) did not pose a significant problem. However, when it was determined that the voltage requirements would need to be raised to 24 V in order to reduce the current flowing through the board the maximum VGS became a problem. In order to prevent the voltage from exceeding the max VGS it was determined that a Zener diode would need to be placed in series with the control transistor. The Zener diode was reversed biased to raise the voltage from ground to the avalanche voltage of the Zener diode. The avalanche voltage chosen for this project was 6.2 V because it will protect VGS when it is 24 V and allow the pass transistor to operate when the power applied to the system is 12 V.



1. Schematic for High-Side Switch

Two methods were used to confirm the design of the system, hand calculations and computer simulations. In order to analyze the system by hand it was necessary to separate the high-side switch into its two main component the power transistor block and the control block. It enables us to simplify the analysis of the circuit because it eliminates unnecessary calculations. The power transistor has two modes open switch and closed switch. In open switch mode the gate of the power transistor is connected to the power rail and in closed switch mode the gate is connect to ground through the Zener diode. The control block is simply an NMOS transistor operating in switch mode.

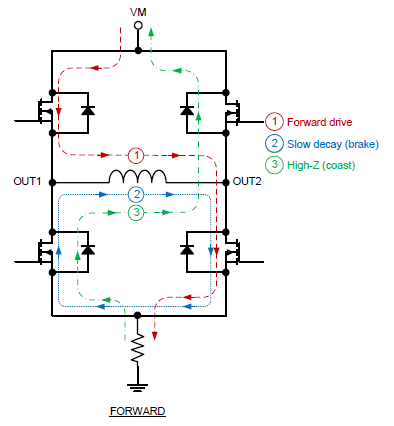
The computer simulation was conducted in LTSpice an open source circuit simulation program. LTSpice did not have the components that we needed to simulate so similar specified components were chosen to substitute for the missing components. The circuit was created in LTSpice and simulated to determine the overall function of circuit. The simulation node V(n004) is VGS of the MOSFET which can be seen does not exceed a difference of -20 V. The logic voltage V(n002) switches the pass transistor and current flows through I(R1) which is the load.



1. High-Side Switch Simulation

#### Subtask 2: Motor Driver

A motor driver is needed to control the DC brushed motor peristaltic pump. It will control the volume and flow rate of the fluid going to the dispensing location. The basic theory behind a motor driver is a full h bridge and the necessary control circuitry. A full h bridge motor driver consists of 2 NMOS transistors which are in series connecting VCC to GND in parallel with two additional NMOS transistors which themselves are is series connecting VCC to Gnd. One lead of the motor is connected in between the 2 NMOS transistor in series.



4

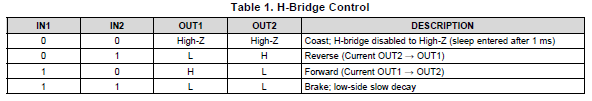
3

2

1

1. Full H Bridge Typography (Texas Instruments, 2016)

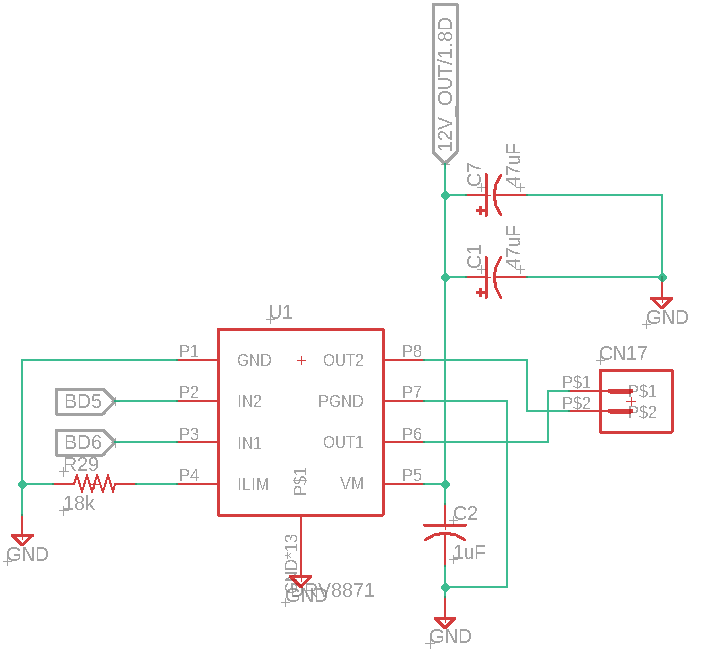
To select the direction of the motor the control circuitry chooses either NMOS transistors 1 and 4 or NMOS transistors 2 and 3. The pairs of NMOS transistors allow the control lines to be reduced from four pins to two pins. Sending a pulse width modulation (PWM) signal to the NMOS transistors will not only control the direction of rotation of the motor but also speed of rotation of the motor. The average voltage of the signal corresponds to the average voltage out to the motor thus setting the speed of rotation of the motor. The motor driver datasheet has logic table which corresponds with the expected outcomes of various inputs to the control pins on the motor driver.



1. Motor Driver Logic Table (Texas Instruments, 2016)

The requirements for motor driver were set by the requirements of the pump 24 V and 1.5 A. We set the requirements of the motor driver to 200% of the current requirements to give headroom for current draw of the motor. Therefore, the motor driver chosen was the Texas Instruments DRV8871 motor driver which supports voltages of up to 45 V and 3.6 A (Texas Instruments, 2016). The DRV8871 has several important features that protect the integrated circuit: overcurrent protection and thermal shutdown. The overcurrent protection is set by a resistor connecting the pin ILIM to ground. The resistor value is determined by the equation the desired trip current of 3 A. The calculated resistor value was 21 kilo-ohm (Texas Instruments, 2016).

In order to protect against rapid voltage changes due to the power supply needing to catch up to the power requirements of the circuit large capacitors were chosen to intervene in these instances. The values chosen were determined by the DRV8871 datasheet which was 47 uF.



1. Motor Driver Schematic

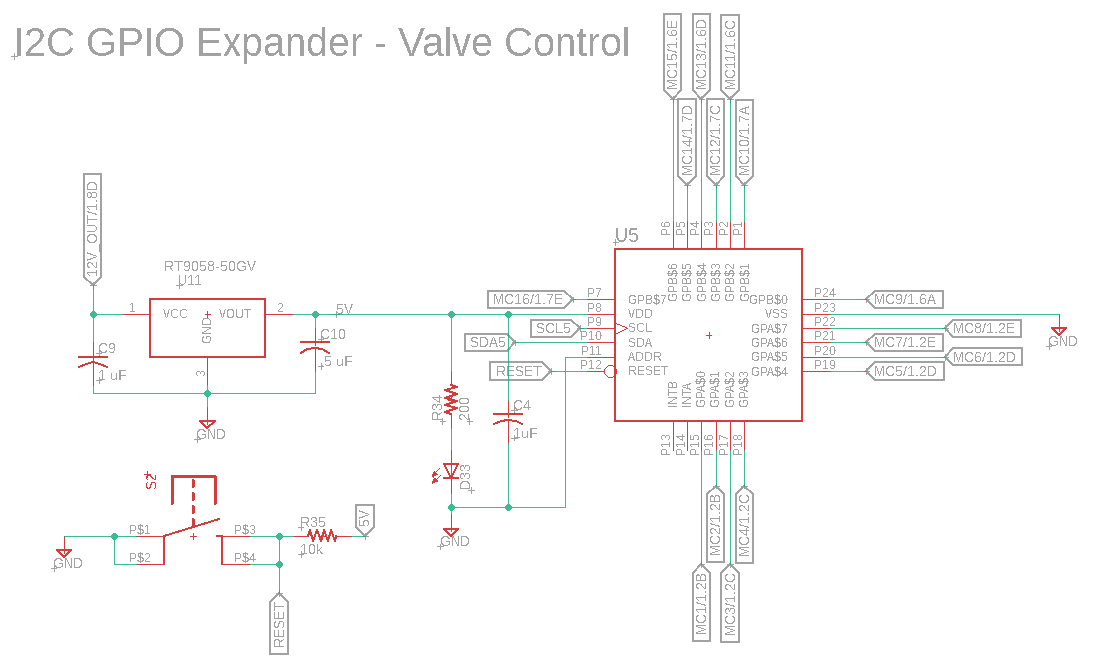
The design of the system requires three motor drivers to reduce the time spent delivering fluid to the cup. The purpose of the project is to reduce wait times as much as possible. The pumps selected for the system can deliver a maximum flow rate of 1000 ml/min. A simple 16-ounce cup will require 473.5 ml of fluid to fill the cup. Simple math demonstrates that it will take nearly 30 seconds to fill a cup. We wished to reduce that time to under 10 seconds. This requirement will require 3 pumps to deliver the necessary volume in the specified time frame.

#### Subtask 4: Design a I2C GPIO Expander and Supporting Circuitry

An I2C GPIO Expander is used to extend the capability of microcontroller through the use of the I2C bus. In our case the GPIO Expander reduces the pin count on the microcontroller needed to control the valves from 16 different pin to 2. It enables the designers to reduce the number of wires coming to and from the board significantly. It also allows the board to be used by a variety of devices such as Arduinos, Raspberry Pi, and various other embedded boards.

The I2C GPIO Expander was chosen using the various parameters given to the project namely the need to control 16 valves. The students conducted a parametric search on Digi-Key to find the desired part. The device chosen was the Microchip MCP23018 16 Bit I2C GPIO Expander. It is capable of delivering 25 mA per pin for a total of 400 mA (Microchip Technology Inc., 2008). Allowing for communications speeds of up 3.4 MHz enabling it for very high-speed communication. These specifications allow it to drive a variety of devices making it useful for our project.

Included during the design and selection of the GPIO Expander the student also selected a suitable 5V linear regulator to support all 5V devices on the PCB. The regulator needed to be able to convert 12-24V into a steady 5V output. After conducting a parametric search on Digi-Key the Richtek RT9058-50GV 5V 100 mA linear regulator was selected. The datasheet specified that the linear regulator required a 1 uF smoothing input capacitor and an additional .5 uF smoothing output capacitor (Richtek Technology Corporation, 2016).



1. I2C GPIO Expander and 5V Regulator

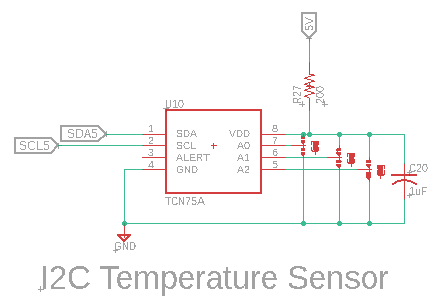
#### Subtask 5: Design a I2C Thermometer and Supporting Circuitry

It is important to be able to monitor board temperatures to make sure the devices on the printed circuit board do not exceeded the maximums designated by the designer. The most common reason for failure among electrical components is due to overheating. It is important to recognize issues before they occur as a result it was decided that it would be important to include a thermometer to monitor the overall health of the system.

A simple parametric was conducted on Digi-Key to look for a component that was capable of I2C communication. The I2C thermometer chosen for this project is the Texas Instruments TCN75A. The device is a little limited in communication speed to 400 kHz and has a resolution that is selectable between 0.5 - 0.0625 degrees Celsius (Microchip Technology Inc., 2010). These devices will get very precise information of the printed circuit board.

Each I2C device has a 7-bit address to be referenced by the microcontroller. In the case of the temperatures sensor the manufacturer has selected the devices first 4 bits of the address while the user has the ability to select the last 3-bits of the address through the use of jumpers and 0-ohm resistors. When the pin is connected to 5V the that bit of the address will be a 1 and it is just the opposite when the pin is connected to ground. It gives the user the ability to change an address if two devices share the same address (Microchip Technology Inc., 2010).

The resistor is placed in series with the 5V power rail to form a low pass filter with the capacitor. It will filter out any noise coming from the power supply. Filtering the noise from the power supply is important because it could create an error in the measurement thus giving the microcontroller an incorrect view of the overall status of the machine.



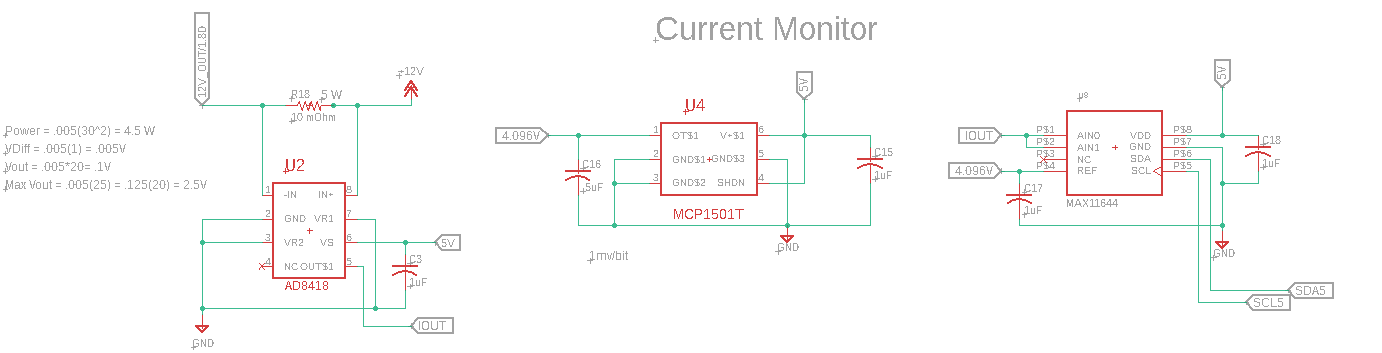
1. I2C Thermometer

#### Subtask 6: I2C Input Current Sensor

Monitoring the current draw of the system is an important feature of the design. It enables the system to determine whether a device has failed or determine whether a pump is actually pumping a fluid. The basic theory behind the I2C current sensor is measuring the voltage drop across a known resistor value and using Ohm’s Law to determine the current flowing through the resistor. The position of the resistor in the circuit allow for us to monitor the current flowing through the whole circuit. In order to measure the voltage drop across the current sense resistor the input voltage to the resistor and the output voltage from the resistor is feed to an operational amplifier. The operational amplifier amplifies the change in voltage between the two inputs allowing it to measured by an analog to digital converter.

In order to select the right current sense resistor some math was necessary to determine its value. The current sense operational amplifier’s gain is 20 and the max voltage of the operational amplifier is set by the input voltage to the operational amplifier which is 5V. Therefore, the divide the max voltage by the gain of the operation amplifier to determine the max voltage drop across the resistor which is . The max calculated current draw of the device is 12.5 A therefore the fuse selected for the device is 15 A. Using ohms law, the calculated resistor value is: . However, according to analog to digital converter the max voltage that can be measured by the device is set by the voltage reference which is 4.096 V. The resistor value chosen for the current sense resistor is 10 mΩ. It is also necessary to determine the power rating of the resistor needed using the equations . In order to give plenty of headroom a 7W .01 Ohm resistor was chosen.

The components chosen for the input current sensor: Operational Amplifier – Analog Devices AD8418A; Analog to Digital Converter – Maxim MAX11644; and 4.096 V Voltage Reference – Microchip MCP1501T. The components selected allow the system to see current draws as little as 10 mA and as much as 15 A.



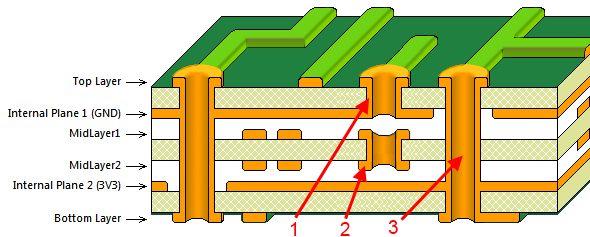
1. I2C Input Current Sensor

The analog to digital converter selected for this project is the MAX11644 which is a 12-bit I2C capable ADC. Although there is an ADC on the microcontroller selected for this project it was decided to separate the task away from the microcontroller to protect the microcontroller from voltage spikes, allow for higher ranges due to the ability to use 5 volts analog inputs instead of the microcontroller ADC’s 3.3 V analog input, and the possibility to use a voltage reference. The ADC will use the 4.096 V reference and the 12 bit resolution to enable the ADC to read as little as a millivolt plus or minus a millivolt.

#### Subtask 7: Design of Printed Circuit Board

The printed circuit board (PCB) is an important part of any modern electrical assembly. The PCB enable all of the components to be connected up without the use of wire or other methods such as perfboard. The PCB consists of two main materials the copper plain in which all electrical power is carried and the insulation material which is usually FR4. These layers can be stacked in multiples of 2 (i.e. 2,4,6, 8, etc.) as high as the designer specifies.

Through an etching process the copper can be removed leaving behind thin section of copper called signal wires. The signal wires are what connect the electrical components on the same sheet of copper. To connect different sheets of copper a via is used. A via is a copper plated hole that will allow electricity to flow through it connecting two different planes of copper. These vias are what enable the multilayer boards that are so common today.



1. PCB Layer Stack-Up (Altium, 2017)

We designed our PCB using a software called Autodesk Eagle. The design process for the printed circuit board starts with the components. The user must first draw up the component in accordance with its datasheet. Each datasheet has a pinout and footprint for the components. The pinout shows what each pin of the component does and the footprint shows the size and location of pads that the pins will be soldered to. In Eagle this is done with component editor. The component editor allows the user to draw up the pinout for the part and the foot print of the part.

The second step that must be completed to design a printed circuit board is laying out the schematic of the circuit. Each component and electrical connection between components is specified in the schematic. The electrical connection between pins of components is created by using the net command in Eagle. The net command also allows the user to specify trace width of the signal wires in the printed circuit board. It is important to specify the trace width to allow the desired current to flow. If the trace is too small for the specified current than the trace has the potential to destroy itself.

The third and final step occurs after the schematic has been completed. The layout of the PCB. The layout begins with the specifications from the desired printed circuit board manufacturer. Each manufacturer has limits on the what can be performed such as limits on the minimum trace width or how close each trace can be to each other. These specifications are entered into the software so that it knows what can be done.

During the third step it must also be specified how many layers the designer wishes to use for his design. We chose to use a four-layer board for a few reasons. The first reason is that it is important to keep the ground and power wires to be as large and clear of obstructions as possible. It helps improve the flow electricity through the board because it is able to take the shortest path to the load. To meet these requirements, we chose to have separate power and ground planes each using a layer of copper. The second reason is that it enables the signal wires to be isolated from high current areas of the board. Improving signal integrity. The four layers are: signal, power, ground, and signal.

The components are place in their desired location. It is important to keep the signal wires as short as is physically possible. As a result, components are placed close together. A general rule is that the components are generally oriented in the same direction. Now this rule doesn’t have to be strictly adhered to but it does help during the hand soldering process allowing the technician to keep a steady procession during the soldering process.

#### Subtask 8: Create a Bill of Materials, Order Components and Printed Circuit Board

Creating a bill of materials is relatively easy by using Digi-Key’s bill of material manager. Eagle allows the user to print out a bill of materials with component values and names of components. We took the generated list and enter each component into Digi-Key and selected the desired quantity of the components. If a component could not be found on Digi-Key we used a different distributer Mouser. Using these two distributers we were able to find all of the components and buy them.

The process of ordering a printed circuit board is a little more complicated. First in Eagle the user most generate a set of Gerber files. The Gerber files tell the manufactures were the copper needs to be removed and where drill holes need to be placed. A Gerber file is needed for each layer of the board and a separate drill and silkscreen layer. The Gerber files are generated using the CAM Function in Eagle. It allows the user to select which layers go with each layer of the printed circuit board.

The manufacturer we chose for this project is JLC-PCB. The offer ten 4-layer board for $91.00 per meter square. It is one of the cheapest manufacturers of printed circuit boards available. Once the Gerber files are generated it is simple to upload them to JLC’s website and order them.

* 1. Implementation of Task 3

In order to complete task 3, a custom communication scheme had to be developed between the Master Computer and the Fluid Microcontroller PCB. Due to software issues with the Fluid Microcontroller board, software-interrupt based communication could not be used, so a new way had to be developed to control the interactions between each board. General Purpose Input/Output (GPIO) pins were connected between each board in order to control whether the one board could send data to the other board. The Fluid Microcontroller would set their GPIO pin low (i.e. bringing the pin to the board’s GND voltage level) when communication could be carried out, and a high GPIO pin (3.3 Volts) would mean that the Fluid Microcontroller was busy with another task and could not receive data. When the GPIO pin was low, the I2C protocol was used to transfer data packets between each computer. The Master Computer acted as the master device on the I2C bus line, and the Fluid Microcontroller acted as a slave device on the same line. The “Polling” technique is being used for checking the GPIO pin and reading transferred data into a buffer.

The reason that communication was needed between each board is because the Master Computer and Fluid Microcontroller must have matching sets of data for making drinks. The Fluid Microcontroller does not use persistent storage for drink recipes or inventory placement, so this information must be transferred to the Fluid Microcontroller on startup of the Master Computer’s software. Until this information is transferred, the Fluid Microcontroller will be in an idle state. Because the Fluid Controller is in an idle state waiting for recipe data, the GPIO pin for controlling communication will be held low. When the Master Computer polls this GPIO pin, it should immediately begin the process of packing together the drink recipes into a drink menu data structure, which will be unpacked into an expected format before transferring to the Fluid Microcontroller. Once the microcontroller receives this data, it will store it in RAM for later use. While the microcontroller is processing and storing the recipe data, its GPIO pin will be brought high in order to prevent the Master Computer from sending any more data while it is busy. Eventually, the GPIO on the Fluid Controller will be brought low when it is done. This allows the Master Computer to send drink order commands to the Fluid Controller, which will also cause the GPIO state to change in order to control the communication.

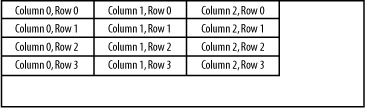
The storage of the recipes for the master computer will be in Master Computer’s main memory as a text file. On startup of the desktop application software, the application will read this file and display its contents in an organized fashion for viewing by the user of the system. When changes are made to the recipes, the text file containing the recipes will be edited in order to keep the information being shared with the Fluid Microcontroller accurate. In addition, the information about the inventory items for the full system was stored in the main memory as well, so that the desktop application could transfer inventory item positions (i.e. valve numbers) to the Fluid Controller board. If changes occur in the offered inventory items, then these changes would require that the recipes be updated too.

The recipes are composed of times related to the amount of time to keep a pump running, which will transfer a precise amount of liquid to a cup. Each inventory item will correspond to a valve in the system (these are controlled by an onboard register on the Fluid Controller), so the recipe pump times are ordered by the id of the inventory items. The order in which recipes are transferred over to the Fluid Controller determines the id number of the recipe. When order commands are sent to the Fluid Controller, the Fluid Controller will expect a recipe id number and quantity of drinks that need to be made.

* 1. Implementation of Task 4.

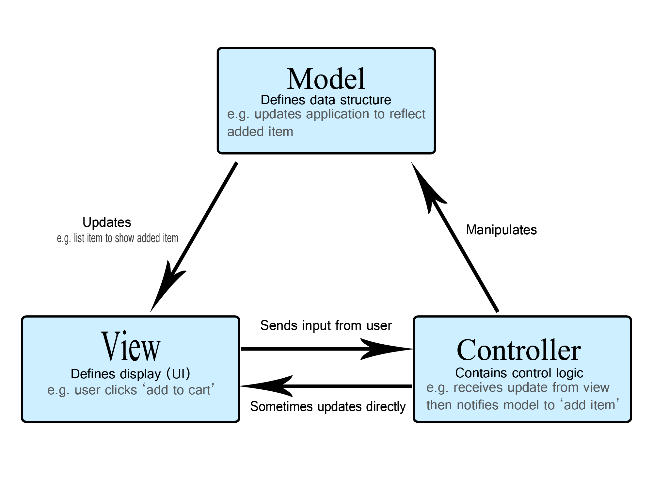
#### Subtask 1. Design A GUI using Python’s Tkinter Library

This subtask dealt with designing a Graphical User Interface (GUI) using the Python programming language and its Tkinter library. Python is a high-level scripting language that is used for general purpose programming. Its Tkinter library comes with the initial installation of the Python toolkit, and it offers common GUI components/widgets (e.g. buttons, labels, scrollbars). The Tkinter library directly interfaces with the Tk widget toolkit, which is open source, native to various operating systems, and cross-platform. By using Python and Tkinter for the main application, this allows for easy extendibility and portability. For organizing GUI elements/widgets, there are usually three options: Place geometry, Grid geometry, and Pack geometry. The Place geometry option places elements at certain pixel coordinates. The Grid geometry splits a window into a grid and requires placements of elements within ordered cells. The Pack geometry fills in specified zones of a window with the use of keywords (e.g. TOP, BOTTOM, FILL, STICKY). The geometry that was picked for this project with Grid geometry because it is the simplest to use and maintain. The following figure shows what Grid geometry looks like.



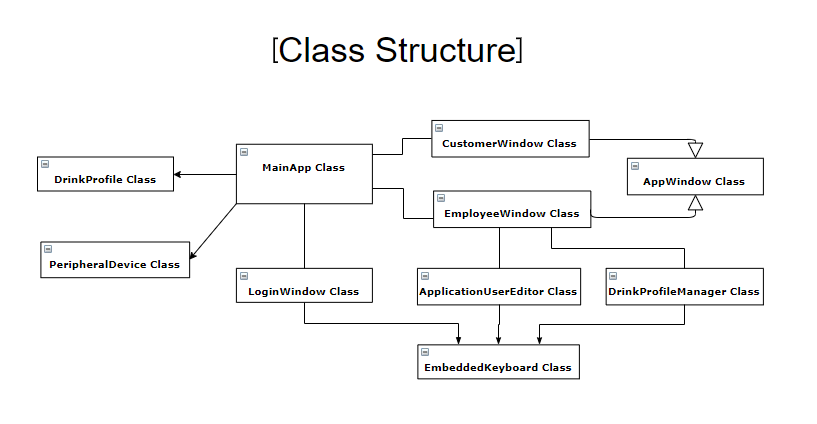
1. Grid Geometry Example

For most GUI applications, the structure of the GUI is separated into different parts. This is usually done for two reasons: the work can be split between different people and there can potentially be more code reuse if the complexity is reduced. These parts are usually the model, view, and control. The model directly manages the data, logic, and rules of the application. The view is any output representation of information. The control accepts input and converts it into commands. This project’s GUI follows this design pattern (also known as MVC).



1. MVC Pattern

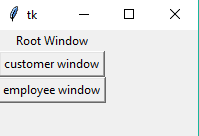
The model would be the MainApp class, which controls the file management of the whole system and directly adds/updates system information. The view is defined completely in the AppWindow, CustomerWindow, EmployeeWindow, and LoginWindow classes. The control is defined in a few different classes because every window needs to report information back to the MainApp object that acts as the central hub. The class structure is shown in a Unified Modeling Language (UML) diagram in the next figure.



1. Class Diagram of Desktop Application

The classes that were made for this project each focus only on a few features. By splitting up the offered features in multiple classes, there is higher readability and lower coupling. These two results encourage reusability of code. For example, the CustomerWindow and EmployeeWindow classes are two classes that could have been merged within one file. The benefit of splitting them up is that they can be customized for a specific user if they exist on their own. Inheritance was used to give the same base functionality to the CustomerWindow and EmployeeWindow classes. The two classes could inherit methods that were useful for both, but still specialize.

Before getting to the customer or employee window, the user of the application must select the mode to operate in for the session. This will be triggered by a physical switch in the future, but the current code makes the user enter a “1” in the shell in order to get into the employee mode or any other number to enter the customer mode. The user is also given the chance to select a button in the root window that will redirect them to one of the two windows. This root window is shown in the following figure.



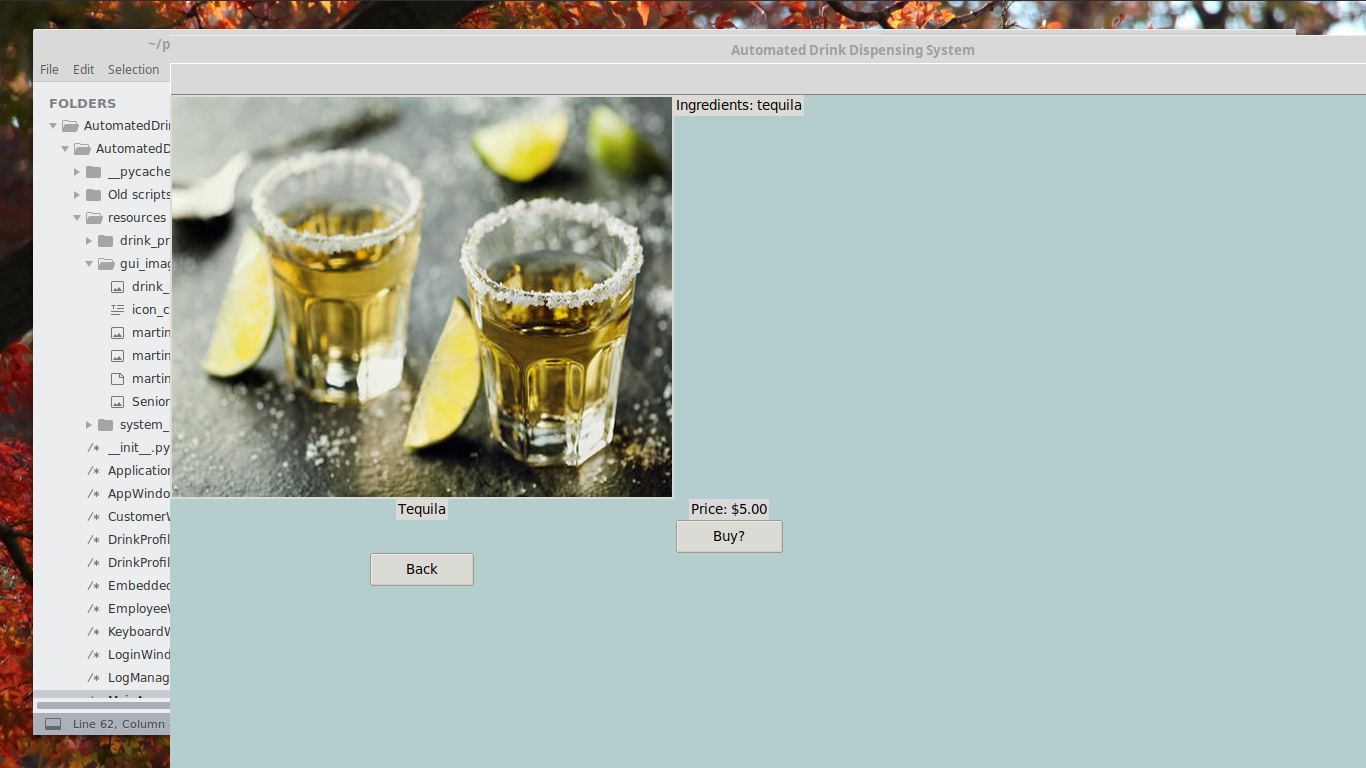
1. Root Window with Mode Selection Buttons

If the user selects either window, then the root window will be withdrawn and the selected window will be displayed. The employee window requires login credentials to use, but the customer window does not require them. The customer window will launch in full screen mode and prevent the user from minimizing, moving, or exiting the window. Ideally, the customer should only have privileges for viewing and ordering desired drinks, so that’s why the window for customers is limited. A “Help” menu was also added that gives users information about how to operate the machine and contact information for the creators of the machine. This window is shown in the following figure, and it is the main screen for the customer window.



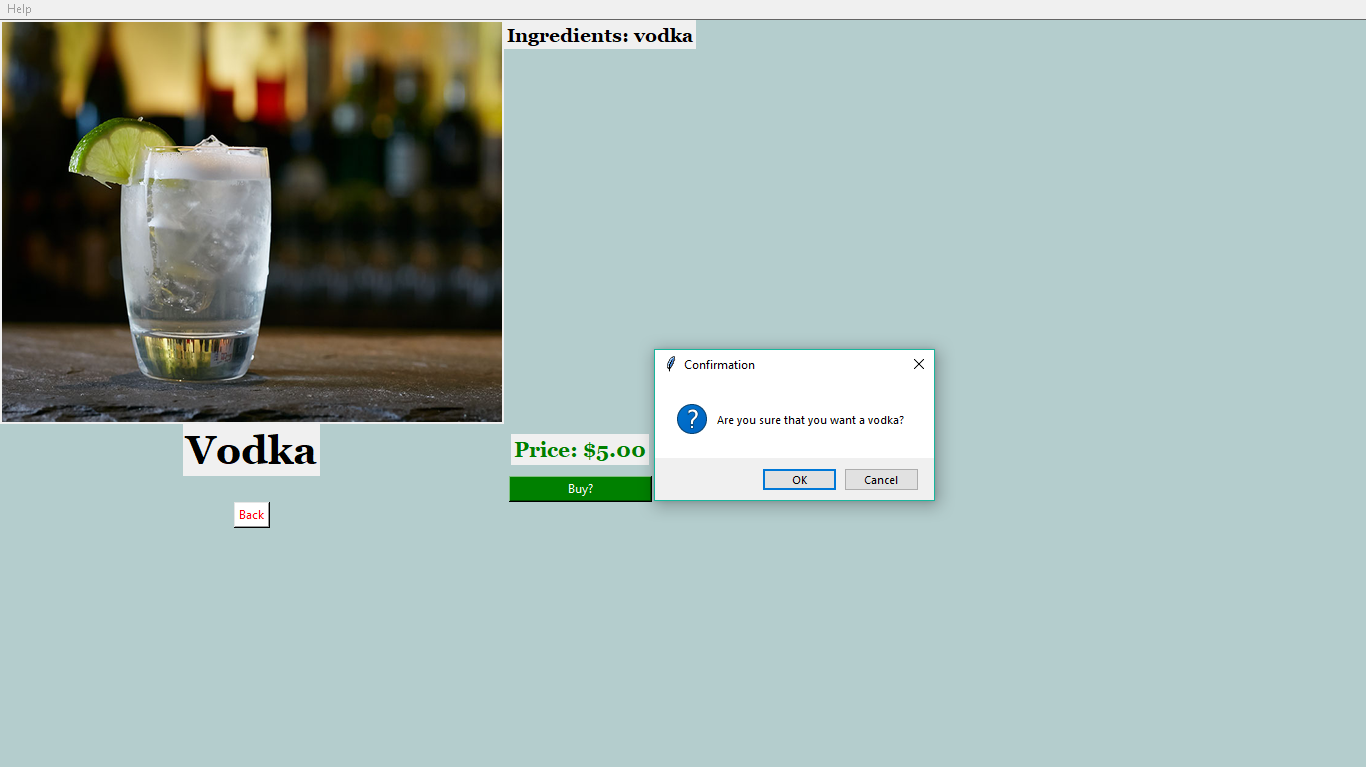
1. Main Window of Customer Window

The CustomerWindow class uses methods from the AppWindow class in order to display each active drink that is stored in the local file system. Active drinks are drinks that are currently capable of being served by the system, and the employee users are the ones that determine whether a drink is active. Within the main screen of the customer window, an active drink will be represented by three widgets: a label showing the drink’s name, a button for selecting the drink’s ordering page, and an image of the drink that is retrieved from the local storage system. Each set of widgets have vertical and horizontal padding in order to utilize all the space within the window. With the current code, only five widget sets can exist within each row. A variable number of active drinks can be loaded into the desktop application, so the main screen will eventually need to be vertically scrollable in order to display all the options. If the customer presses any one of the buttons, then the order page for the selected drink will be displayed. The following figure shows what the order page generally looks like for the customer mode.



1. Order Page for Customer

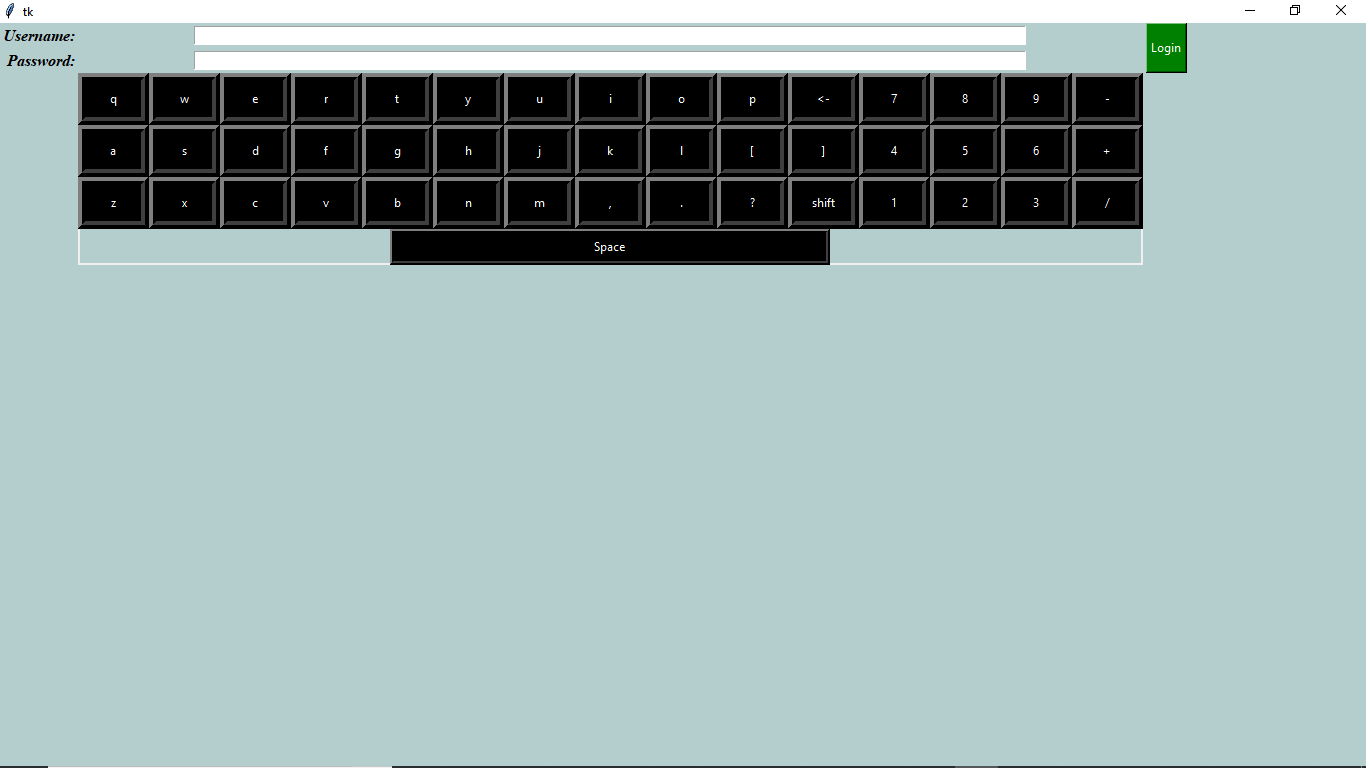
The order page for the customer displays some of the information stored in each drink profile from the local file system. An enlarged picture of the drink is inside a label widget, and it is shown in the top left. The ingredients are stored inside a label widget, and the label is placed in a grid adjacent to the picture. The name of the drink is stored in a label widget directly below the picture. A “back” button widget is placed in the row under the drink name label. The price is stored in a label widget, and it is shown in the column to the right of the first column. The “buy” button is shown below the price label widget, and it initiates the ordering process. If the user clicks on the button, then a confirmation box is displayed. This feature can be seen in the following figure. Eventually, a pop-up window will be displayed to show whether the customer has paid yet.



1. Order Confirmation box

Although the employee mode uses a similar format for the main menu and the ordering pages for each drink, there are a lot more features for that mode. If the user decides to enter employee mode, then the login window will appear and require the user to provide valid credentials before proceeding. The Login window is defined in the LoginWindow class. This class is responsible for the following features: providing an interface for the user to login, attempting to match the user’s login credentials with what’s stored in the local file system, denying entry if the user exceeds the max number of failed attempts, and decrypting/encrypting user login credentials. A login window instance will have the following widgets existing in adjacent grids: a username label, a password label, a username entry field, a password entry field, a canvas that will hold the embedded keyboard, and a login button. The login window instance will also determine whether a registered user is either a regular employee or an admin employee. Admin employees have more privileges than regular employees, so different options will be displayed when the employee window appears. The maximum number of failed attempts per session is three, and if that number is exceeded then the application will automatically bring up the customer mode window. All the registered users of the system will have their username and password stored in a single file. This file will be encrypted using Python’s cryptography library in order to keep entry into the employee mode secure.

Because this desktop application is meant to be a touchscreen application as well, an on-screen keyboard is embedded into the login window. The embedded keyboard is defined in the EmbeddedKeyboard class. It is a collection of button widgets that execute a certain command depending on the character that is displayed in the button. The special keys of the keyboard are the spacebar, backspace, and shift keys. These keys perform the same function as they do on a regular keyboard. The other keys on the embedded keyboard just directly output a letter to the current entry field. In addition, the embedded keyboard can pass input to at least 6 different entry fields, and each entry field that is associated with the embedded keyboard instance will touch bind, which basically makes the entry field active if the user touches the entry field. This embedded keyboard offers all lowercase and uppercase alphanumeric characters. The uppercase characters can be accessed by pressing the shift key. The login window and embedded keyboard can be seen in the following figure.



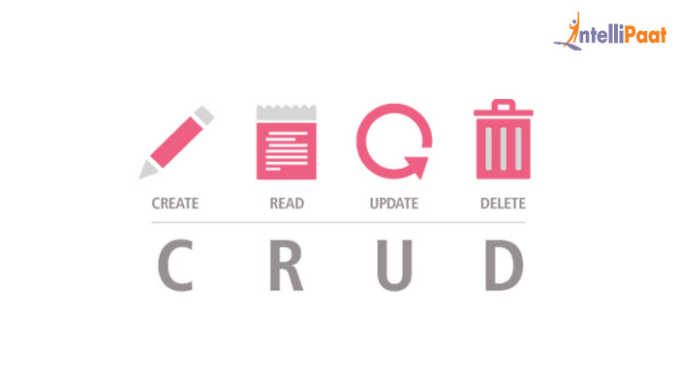
1. Login Window

When valid login credentials are provided, the login window is replaced by the employee window. The only difference between the main screen of the employee and customer windows is that the employee window will have an extra menu bar. If the user is a regular employee, then the menu bar will be the “Employee Options” bar. If the user is an admin user, then the menu bar will be the “Admin Options” bar. These two bars are similar, but the “Admin Options” bar has more options available. Currently, the “Employee Options” only has the “Launch Drink Profile Manager” option. The “Admin Options” bar contains the “Launch Drink Profile Manager”, “Display Log”, “Edit Configuration”, “Edit User Logins”, and “Show IP Address” options. The “Edit Configuration” option is the only that is not defined yet. The following figure shows the “Admin Options” menu bar options.



1. Admin Options Menu Bar

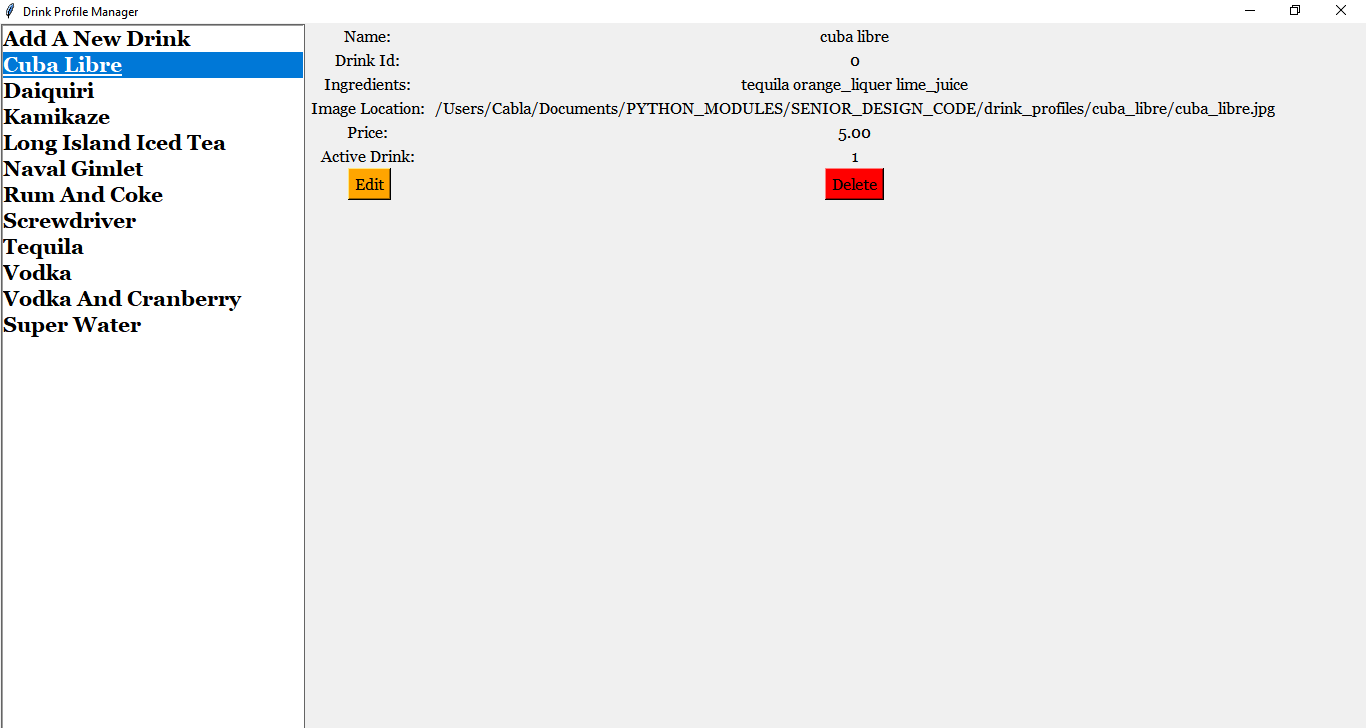
For the “Drink Profile Manager” option, the user is given the ability to change system information while inside the GUI. The goal of this option was to provide CRUD capabilities. CRUD is a software engineering abbreviation for “Create, Read, Update, and Delete”, and it describes the basic operations that should be offered for any application that deals with persistent storage. The following figure shows an image of this principle.



1. CRUD Principle

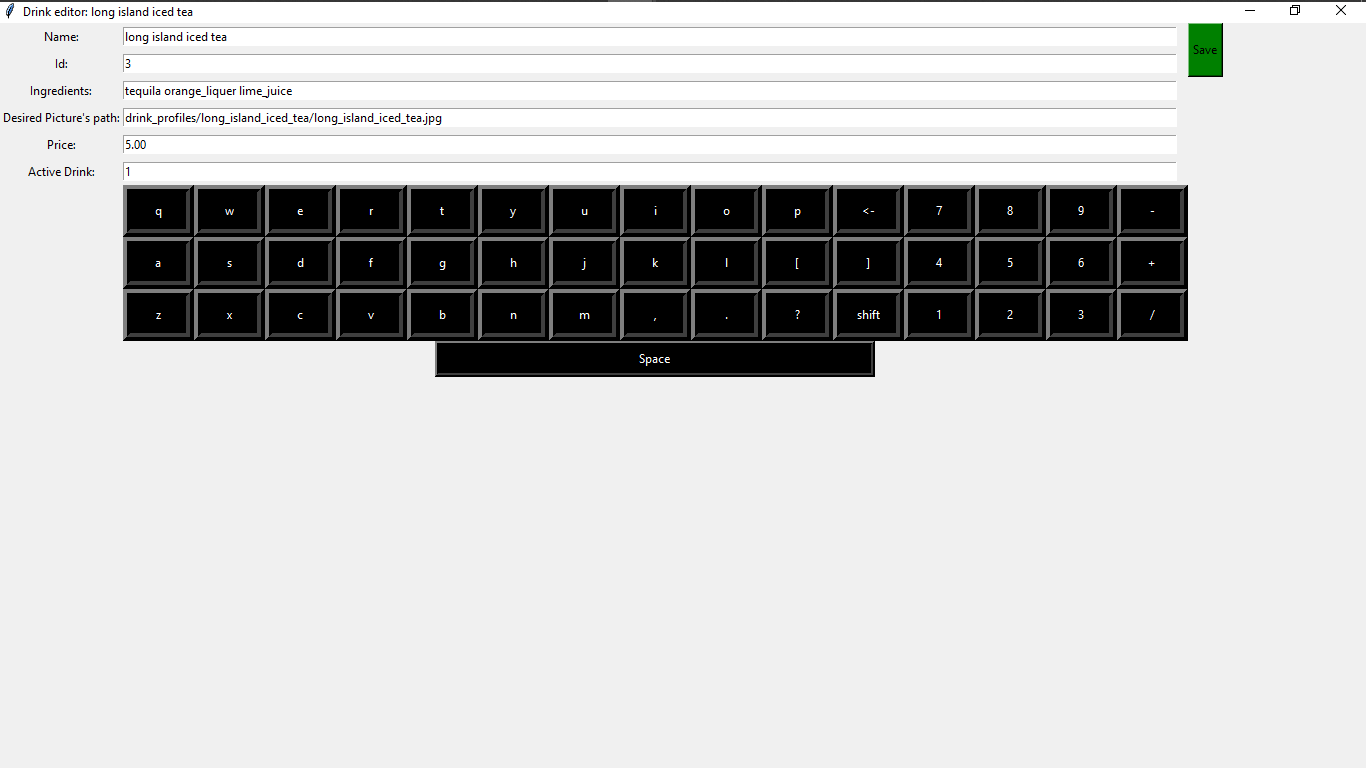
The DrinkProfile class is the class that is used internally for creating, reading, updating, and deleting drink profiles from the local file system. On startup of the application, DrinkProfile objects are created, so that they can store drink information, like the drink Id, drink name, picture location, and drink price. The information that is stored in these DrinkProfile objects are then extracted and displayed inside the GUI. In addition, each DrinkProfile object has methods that allow for creating new drink profiles that will get stored the local file system, deleting drink profiles, and editing drink profiles. This class is where the CRUD methods are implemented. Because CRUD methods must be available to the user of an application, these methods are only called within certain windows of the application.

The DrinkProfileManager class is a class made for editing drink profiles within the GUI. This class provides an interface in a top-level window that the user can interact with and access the CRUD capabilities. It will take DrinkProfile objects and display their information. This is not limited to active drinks, so inactive drink profiles will be loaded as well. This gives the user total control over current and past drink profiles. The user will be given the option to add a new drink, delete the current selected drink, or edit the current selected drink. This is shown in the next figure.



1. DrinkProfileManager Window

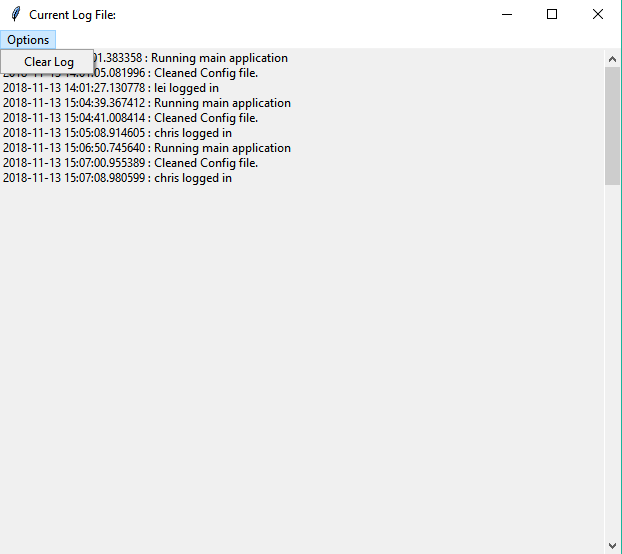
Depending on the option selected, the user will be taken to the drink editor window, which will allow the user to input their desired changes. The drink editor window makes use of the EmbeddedKeyboard class and entry fields as well. The entry fields are populated with the current drink profile information, and the drink profile information is only updated if the content in the entry field changes. The name, Id, ingredients, picture path, price, and active status are the parameters that can be changed. A save button widget was added as well.



1. Drink Editor Window

The “Add A New Drink Option” option in the Drink Profile Manager window brings up the same drink editor, but the entry fields are not populated and the save button is replaced by a create button (each button calls different methods from the DrinkProfile class). The delete button widget in the Drink Profile Manager window allows the user to permanently delete a drink profile from the local file system, and it displays a confirmation box before completing the request.

The next menu option for the admin user is the “Display Log” option. When clicked, this option displays a top-level window widget that shows the contents of the log file for the system. This is an important feature for the user because it will show information relating to the daily use of the machine and include a timestamp. Also, there is an option for clearing the log file if it is not needed. For the current GUI, only the today’s log can be displayed or cleared, but this will be changed in the future, so that multiple log files can be displayed. This window can be seen in the following figure.



1. Log Window with Clear Option

The next menu option for the admin user is the “Edit User Logins” option. When clicked, a top-level window will be instantiated that will show the current registered users for the application. This window is defined in the ApplicationUserEditor class. It simply decrypts the file that contains the user login credentials and displays them. Two button widgets are also placed inside the window. The ApplicationUserEditor window is displayed in the next figure.

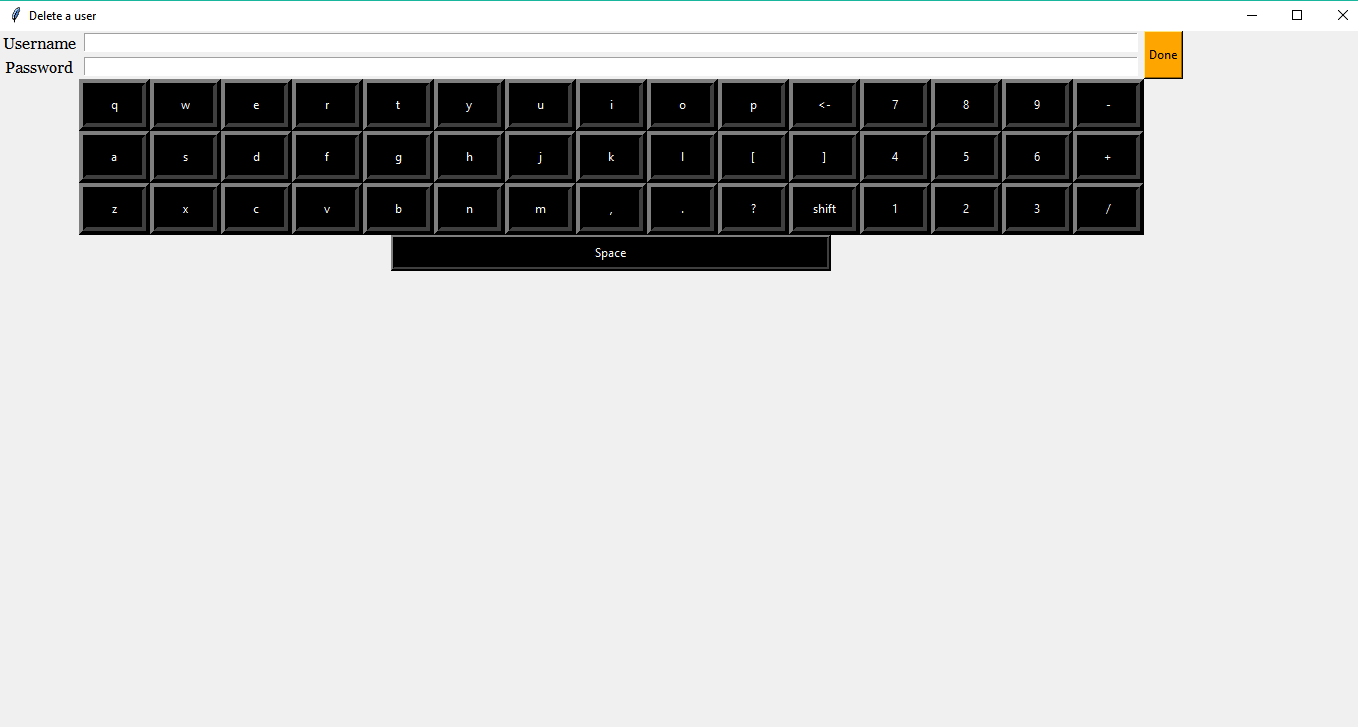


1. ApplicationUserEditor Window

The button widgets inside the ApplicationUserEditor window are for creating a new user login and for deleting an existing user login. These two buttons open the user editor top-level window. The “Delete a User” window has the following widgets: username label, username entry field, password label, password entry field, a done button, and a canvas that holds the embedded keyboard. The “Add a User” editor is slightly different because the user can click a checkbox for making the new user an Admin user. Both buttons will cause changes to the file that holds this information. This is shown in the next two figures.



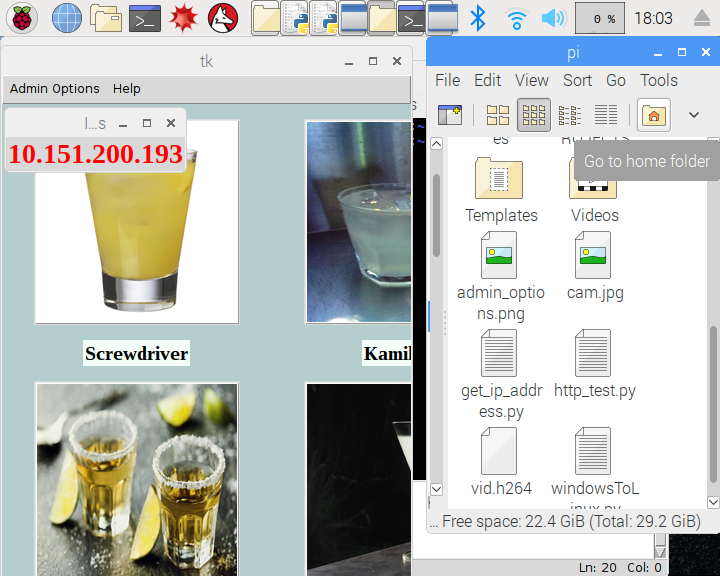
1. “Add a User” Window



1. “Delete a User” Window

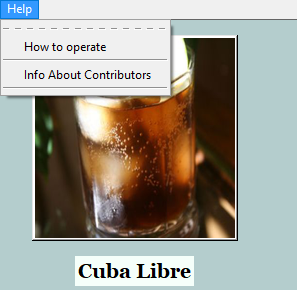
The final offered option for the admin user is the “Show IP Address” option. In order to make a desktop application that can share data with remote devices, a method must be used that can broadcast information over a network. The HyperText Transfer Protocol (HTTP) was used because it is a common, well supported method for serving or grabbing information from other computers on different networks. On startup, the desktop application will initialize a public HTTP server on port 80 of the Raspberry Pi. This server will broadcast files from a certain directory on the Raspberry Pi, so that other computers can retrieve the data from the files. For other computers to see the files, the IP address and port number of the host computer must be known. The user of the application will be able to see the IP address with this option and enter this information into the phone app, so that it can get the necessary files.

When the button for this option is clicked, a top-level window is displayed that contains a label widget. The label widget shows a string that is the wlan0 IP address number. This is an important option because the wlan0 IP address number can constantly change on public networks, so the user of the application must be able to view this information in order to properly share data between the main computer and the phone application. This top-level window is shown in the following figure.



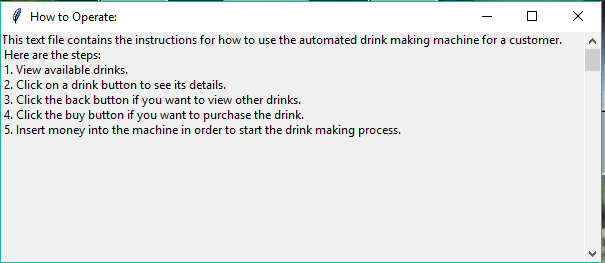
1. IP Address Window

For both the employee and customer windows, the “Help” menu bar is offered. This menu bar displays just two options,” How to Operate” and “Info About Contributors”. For a user to properly use the GUI, there has to be information inside the GUI about how to use the features. Providing this operation information inside a menu bar is standard practice for desktop applications. The “Info About Contributors” option is important because some users may want to contact the creators of the application to give feedback. The “Help” menu bar options can be seen in the following figure.



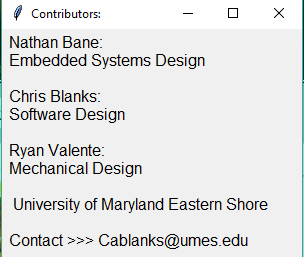
1. “Help” Menu Bar Options

The “How to Operate” option loads operation instructions from a text file into a top-level window. A different instruction file is used for both the employee and customer mode because they require different details about the operation. The employee mode’s instructions would be more about how to use all the features inside the employee window. The customer mode’s instructions would be about how to use the features inside the GUI to order a drink. This window can be seen in the next figure.



1. How to Operate Instructions Window for Customer

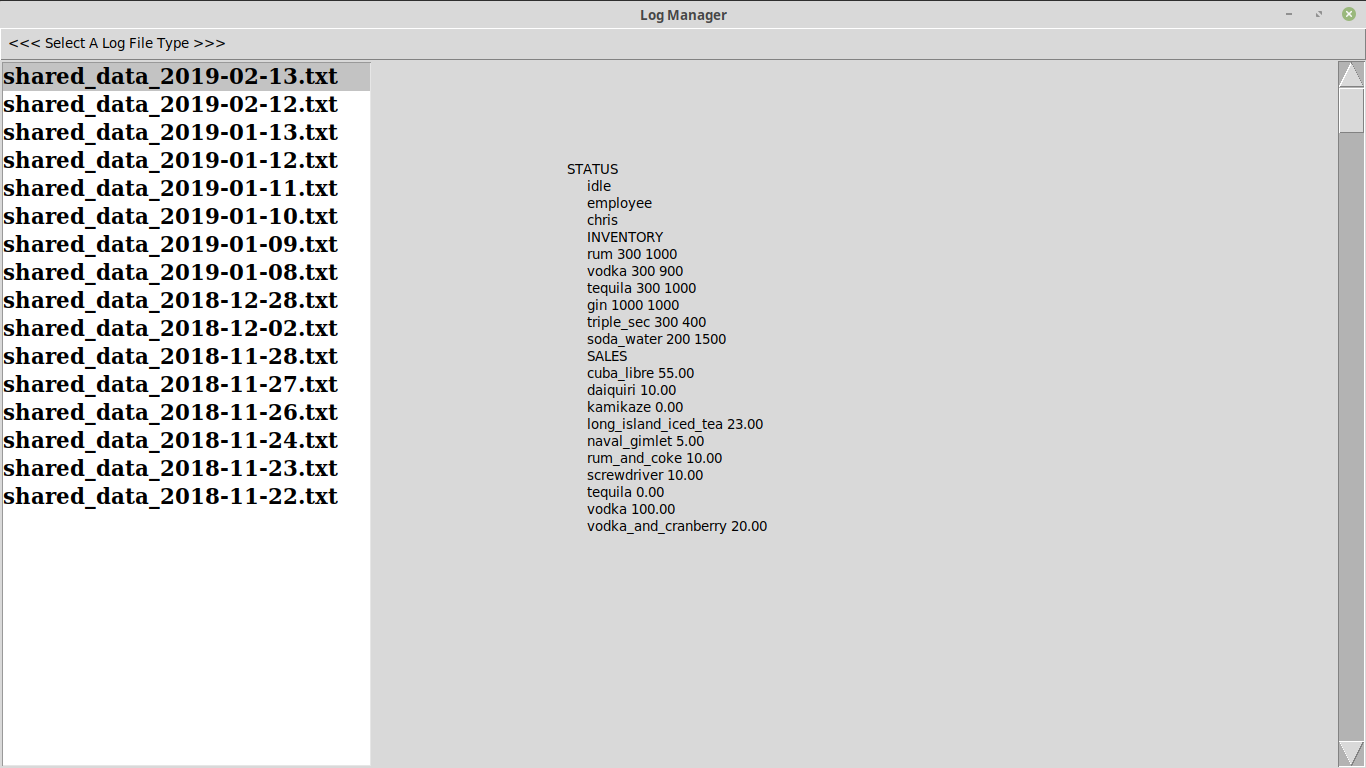
The “Contributors” option opens a top-level window that shows the names and roles of each person who contributed to the Automated Drink Dispensing Project. The name of our university is also displayed inside the window. Also contact information is provided inside this window. All of this text exists within a single label widget. This window can be seen in the following figure.



1. Contributors Window

In order to keep track of inventory items, an Inventory Manager was created. The Inventory Manager handles basic operations and displaying of registered inventory items. The following features are available within the manager: adding new inventory items, switching the valves of registered inventory items, editing current quantity, and editing original quantity. These features will allow a user to have full control of each inventory item.

Another feature for helping the user is a Log Manager. The log manager reads and displays log files that were created by the desktop application. There are three types of log files: shared data files, sales data files, and system log files. The shared data files are used to provide a system summary to devices that are using the phone application. The sales data files are for displaying daily sales within the desktop application. The system log files show what operations have been applied to the desktop application, which is good for tracking its daily use.

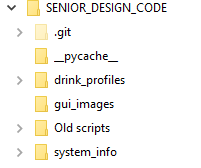


1. Log Manager

These are the main features that are packed into the desktop application. With the features that have been mentioned, the user of the desktop application should be able to control the Automated Drink Dispensing System and make updates within the GUI.

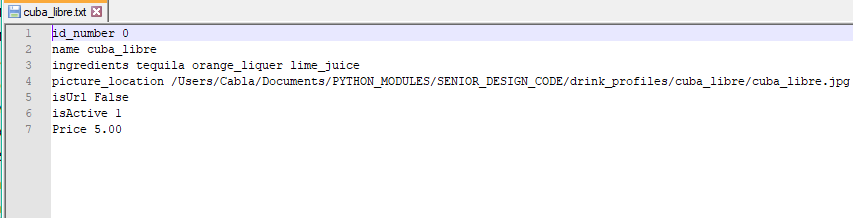
#### Subtask 2. Develop an efficient way to organize system & drink information

This subtask dealt with developing an efficient way to organize system and drink information that would be used by the main application, phone application, and peripheral devices. The need for an organized collection of data is crucial because mismanaged information can ruin the performance of the complete system and lower user satisfaction regarding the full product. All the system and drink information are stored in the local file system of the main computer. For easy access, this information is stored in child directories to the “SENIOR\_DESIGN\_CODE” parent directory. The system information is stored in its own directory, “system\_info”, and the drink information is stored in the “drink\_profiles” directory.



1. Parent and Child Directories

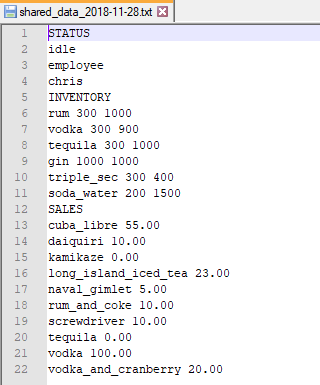
Within the drink profile directory, each drink that is offered on our current menu has its own child directory. Each of these children directories have their own image and text file. The images are all the JPEG format because that is the easiest format to work with when using Python’s native libraries for performing functions on images. The text file contains the information needed to build a drink profile inside the main application. On each line of the text file, there are space delimited values that become attributes for Drink objects. There are lines for the id number of the drink, the name, the ingredients, the location of the picture, and the price. The remaining lines are for internal use within the main application. By standardizing this format for the drink profiles, it allows the main application to load an arbitrary amount of drink profiles and contain all the relevant information in Drink objects that are used all over the code.



1. Example of the Text File Format of Each Drink Profile

Within the system\_info directory, there are a few files that are for internal use in the desktop application. These files are the config.txt, instructions\_4\_customer.txt, instructions\_4\_employee.txt, inventory\_info.txt, user\_login.txt, and key.txt. The config.txt is for keeping a persistent record of system information that can be referenced by the desktop application and phone application. The instructions text files were created for providing instructions about how to use the GUI for employee users and customer users. Each type of user will need different instructions because there are different privileges and features for each one. The inventory\_info.txt will be used for keeping track of the quantity of inventory items, which is very important. The user\_login.txt file is a very important file because it stores all the user login credentials. The desktop application will read from this file whenever an employee tries to log into the system. The contents of this file are encrypted using symmetric key encryption, so it looks just like a bunch of random characters put together if the user were to open the file normally. The key.txt file stores the symmetric key that is needed for decrypting the user\_login.txt file. The desktop application will attempt to retrieve the key from this file first before trying to read from the user\_login.txt file. If a key does not exist, then the desktop application will make a new one. This also means that the current user\_login.txt file cannot be decrypted if a backup key is not stored somewhere else.

There are three children directories located in the system\_info directory, and they are the drink\_sales, log\_files, and shared\_data directories. The drink\_sales directory holds the daily drink sales files that hold information about what drinks were sold throughout the day and the price of the bought drinks. Eventually, these files will be viewable in the log window option within the employee window. New sales are simply appended to the end of the current sales file. The log\_files directory contains the log files that describe actions that were executed throughout the day. These are already displayed inside the GUI, and when the user of the GUI chooses to clear a log file, the application deletes the current log file and makes a new one for the day. New actions are simply appended to the end of the current log file. The shared\_data directory contains shared data files for each day. The format of the shared data files is the same for all the files, and they are simply overwritten whenever changes are made to the data that is being shared. There are three sections for the data inside the files. The “STATUS” section contains three lines about the current system state, current mode for the application, and current logged in registered user. The “INVENTORY” section contains information about the current quantity and original quantity of the inventory of the machine. The “SALES” section contains information about the daily sales for each drink that was bought. For the current main application, the shared data is updated every single minute. Using a HTTP server, this directory is shared to remote devices that are aware of the main computer’s current IP address. Only this directory is broadcasted, so the other system information cannot be obtained. The phone application will be directed to the main computer’s IP address and use html parsing in order to find relevant shared data files. The format of the shared data files can be seen in the following figure.

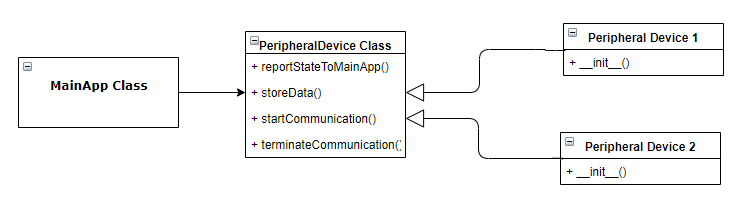


1. Shared Data File Format

#### Subtask 3. Add in capabilities for communication between main computer & peripheral devices

This subtask dealt with creating an interface between the main application and other peripheral devices. For this system work, there must be a standard interface for commands and data to be sent through. The main application will be sending out commands to peripheral devices and receiving data that will determine the next state of the whole system. The peripheral devices of the main computer are the camera, the payment collector, the embedded boards, and any other addition that will add useful capabilities to the full system. By defining a standard interface, adding even more peripheral devices will not take as much time as writing an individual interface for each new device.

This interface is in the form of the PeripheralDevice class. The PeripheralDevice class setups up basic attributes that the main application will use to command and interact with a peripheral device. Some of the attributes are the name of the device, state of the device, the data buffer, the communication method, and the GPIO pin numbers on the Raspberry Pi that the device occupies. With this information, the main application will have complete information about the device and will be able to effectively use that device. Some methods that the main application will call deal with reporting the state to the main app, sending data to the main app, starting communication with the device and the main computer, and terminating communication. With the use of inheritance, the Peripheral Device class will act as the base class for each individual device class. While each individual device class will have its own methods, the main application will only have to use the methods in the base class. This allows for easy extendibility and prevents unacceptable amounts of coupling between the devices and the main application. A diagram is shown in the following figure that displays this relationship.



1. PeripheralDevice Inheritance Diagram

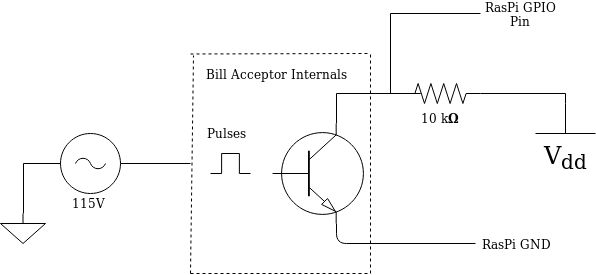
The first peripheral device that integrated into the system was the employee switch. The employee switch is a simple push button that is connected between a GPIO pin of the Master Computer and the Master Computer’s GND pin. The GPIO pin has an internal pull-up resistor connected to it, so that the pin remains at a logical high. When the push button is pressed, the GPIO pin is pulled to the GND voltage, which allows a user of the system to get to the login screen of the desktop application. Without being able to press the employee switch, the desktop application software will remain in customer mode. This GPIO pin is being polled every 2 seconds, so that it won’t disrupt the desktop application software.

The next peripheral device that was integrated into the system was a MEI bill acceptor. The bill acceptor can be configured to accept US bills (e.g. $1, $5, $10). The bill acceptor needs an input voltage of 115 VAC in order to turn on. When a bill is inserted into the bill acceptor, an input line sends a pulse for every dollar.



1. MEI Bill Acceptor

The bill acceptor input line was attached directly a GPIO pin on the Master Computer. The pin was waiting for the negative edge of a pulse in order to count the pulses that the bill acceptor emitted. The input line was connected to the collector of a NPN bipolar junction transistor (BJT). The emitter of the BJT was connected to the Master Computer’s GND pin. The gate of the BJT was assumed to be receiving pulses from some internal circuitry of the bill acceptor that counted the inserted money. For easier testing of the system, every drink option in the machine costs a dollar. There is no way to give change back to the user of the system, so either a change device will have to be integrated as well, every drink will have to be a certain dollar amount, or a new system for payment will have to be realized in order to make a full product.



1. MEI Bill Acceptor Interface Schematic

The next peripheral device to be integrated into the system was the Picamera. The Picamera can connect with a slot on the Master Computer’s PCB. The Picamera must be enabled through the “raspi-config” utility in the Raspbian OS. The Picamera can take pictures and videos. The role of the Picamera in the full system was for entertainment purposes. The Picamera would be used to collect live videos and draw animations on frames depicting users on the machine. The desktop software collected each edited frame from the Picamera and displayed it within the GUI. The Due to not having a full system setup with a mount for the Picamera, a 3D structure was printed that would hold up the Picamera for testing purposes.

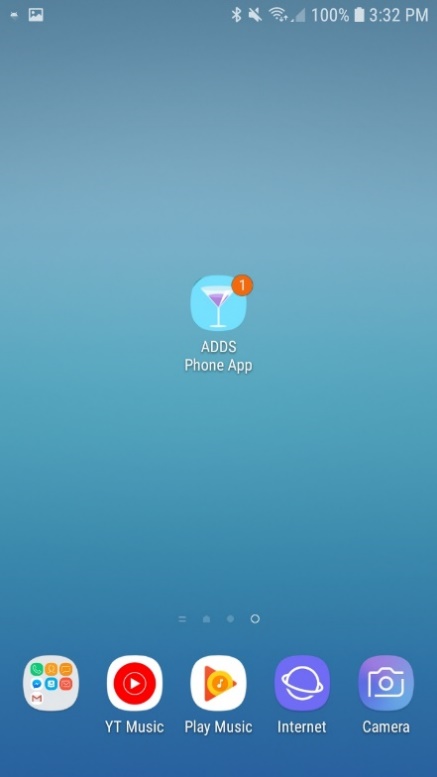


1. 3D Printed Picamera holder
   1. Implementation of Task 5.

#### Subtask 1. Design the user interface of the phone app with Android Studio’s SDK

The phone application for this project was built using Android Studio’s software development kit (SDK). Android Studio is a common tool used for making applications for phones running a version of the Android operating system. It allows you to build applications for many phones and different versions of the OS. For this project, the phone application can run on phones that have Ice Cream Sandwich (SDK version 15) up to Pie (SDK version 28), and it is compiled using SDK version 28. By building the application for these targets, there is at least 99.8% coverage of the android OS distribution of running devices (Android Developers, 2018). Android phone applications are usually written in the Java programming language and the Extensible Markup Language (XML). The Java portion will typically define the app behavior, and the XML portion will define the view/layout of the application. For testing the phone app, a J7 phone that is running Oreo (SDK version 26) was used.

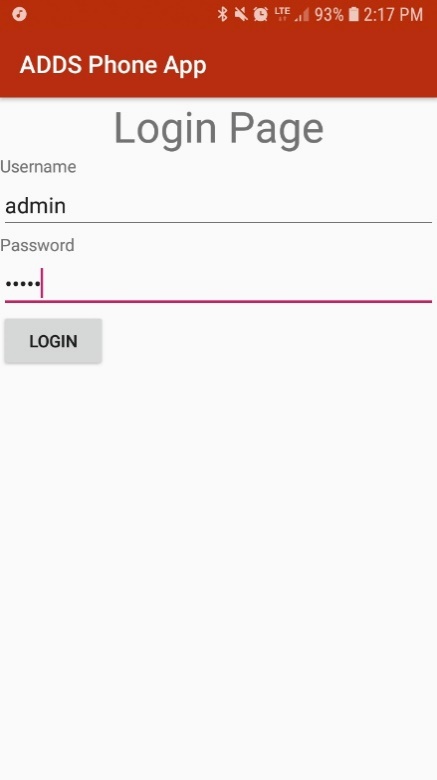
For this project’s phone application, the application had to provide monitoring features for an owner/manager of the Automated Drink Dispensing System (ADDS). The phone app’s icon was made to be a picture of a drink because that reflected the function of the ADDS. The application was named “ADDS Phone App”. The icon and name of the application can be seen in the following figure.



1. Phone app Icon with Notification Bubble

Because this phone app is supposed to be used by an employee of the establishment that purchases an ADDS, there is a login screen that is supposed to compare phone app user login credentials to the desktop application’s login credentials. This would require the phone app to acquire the user\_login.txt and key.txt file from the Raspberry Pi and find a Java library that can make a cipher for decrypting the user login information. For the current phone app, there is only one registered user, and the registered user is the “admin” user.

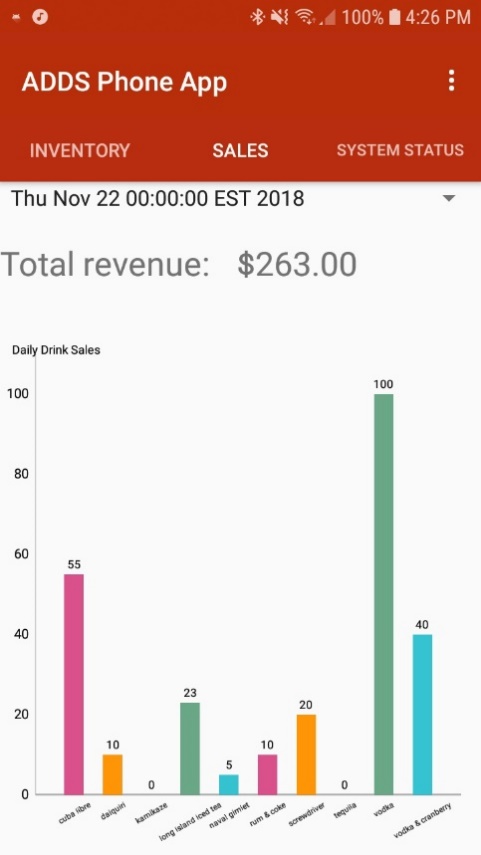
In order to make the user of the phone app login to the app, a login page had to be made. Using XML and Android Studio’s layout constraint specifications, a simple login page was made. The login page can be seen in the following figure.



1. Login Page of Phone App

The login page uses TextView, EditText, and Button widgets. The TextView widgets (Username and Password) display the text “Username” and “Password”, which gives the user a hint about what the following EditText fields are used for. The EditText fields are basically entry fields. The password EditText will hide the password from being shown. Once the user is done with entering their login credentials and presses the “LOGIN” button, the phone app will check to see if it matches the stored username and password sets. If the user fails to enter correct login credentials, then a Toast (a temporary On-screen message) will be displayed that says, “Failed login. Try again.”. When the user fails three times, the login button will be locked. The user will be forced to close the application and try again. If the user is successful, then a Toast will be displayed that says, “Successful login”. The phone application will then proceed to the next screen/activity.

The next activity allows the user to interact with the monitoring features of the phone app. There were three main features that the phone app had to at least have for the user, and they were a sales feature, inventory feature, and a system status feature. The sales feature had to at least show the total sales for the current day. The inventory feature had to at least show the amount of inventory left in the machine. The system status feature had to at least show basic system information. Because there were three features that had to be displayed and each had equal importance, a tabbed view was selected for the main menu of the phone app. A tabbed view allows the designer of an app to split features between multiple tabs, and each tab can be accessed by either swiping the screen or clicking on the tab head. The tabbed view can be seen in the next figure.



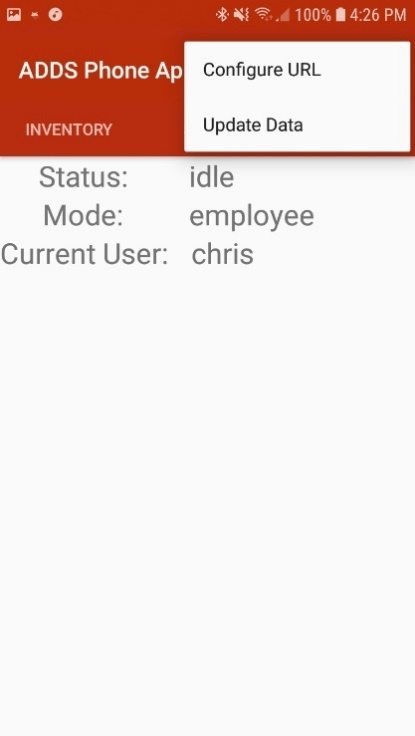
1. Tabbed View of Main Menu

A tabbed view allows each feature to take up the full screen. For Android Studio, tabbed views require a container and fragments. The container stores fragments and keeps track of the current fragment that is being displayed. Fragments are the different views that can be accessed by swiping or tapping. The container requires its own layout defined in XML, and each fragment requires its own layout described in XML. This allows for customization of each fragment, which is great if the designer wants to make each fragment handle different behavior and show different widgets. Because three different features were offered, three fragments were made. The fragments were named “Inventory”, “Sales”, and “System Status”, which can be seen in the previous figure. The data for each fragment is retrieved from the HTTP server that is being hosted by the Raspberry Pi. The user interface for each fragment is described in more detail in the following subtasks.

The user interface of the phone application was completed when all the main features were implemented and refined. There are still some features that can be added to the phone application in order to make it easier to use and more automated. That will be done in future work.

#### Subtask 2. Setup the HTTP client on the phone app & access the system information stored on the Raspberry Pi

For this subtask, the phone app needed to acquire the capability of accessing the files being broadcasted on the Raspberry Pi’s HTTP server. Naturally, this meant setting up a HTTP client on the phone application and guiding the client to the right Uniform Resource Locator (URL) and port number on the Raspberry Pi. The user can see the options for dealing with the HTTP server by clicking the settings icon in the bottom right corner of the action bar of the phone app. By clicking the “Update Data” option, the HTTP client will attempt to connect to the HTTP server and retrieve the specified data. A toast will be displayed with a message that is dependent on whether the HTTP client succeeds. These options are shown in the following figure.



1. URL Settings Options

When the “Configure URL” button is clicked, the phone app switches to a new activity, which is used for editing the URL link that the phone app’s client is being directed towards. The current URL will be displayed in a TextEdit widget. A TextView is located above the Text Edit, so that it show the user the TextEdit’s purpose. A save button widget is directly below the TextEdit widget, so that the user can save the new URL for future use. Whenever the URL is changed, the Android Preference value is changed. The new URL will show up every time the phone app is used until it is changed by the user again. The back button simply returns to the previous activity. These features are shown in the following figure.



1. URL Editor Layout

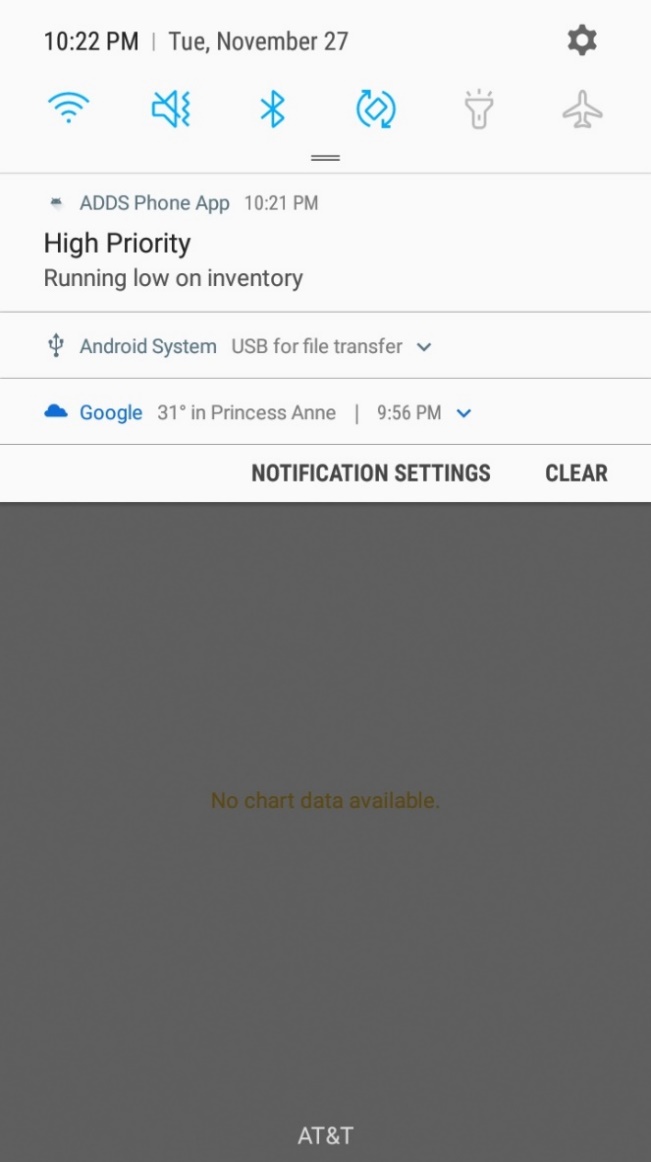
#### Subtask 3. Provide monitoring features within the phone app

The inventory fragment uses a TableLayout widget for displaying inventory information. The TableLayout widget splits the screen into a grid with columns and rows. A TableLayout widget can take TableRow widgets that help with making an organized layout. Within each cell in the grid, a single widget can be placed inside. For this fragment, TextViews were placed inside each cell of each row. Because the number of inventory items can be changed depending on the owner’s choice of drinks, the TableLayout is made dynamically. Usually, layouts are defined statically with XML, but that method is not very adaptable. Depending on the number of inventory items, TableRow widgets will be created and packed into the parent TableLayout widget. The only statically defined TableRows in the TableLayout widget are column titles, which are “Items”, “QTY Left”, and “Original QTY”. The “Items” column will hold the inventory item name. The “QTY Left” column will show the amount of fluid left in the inventory item’s container. The “Original QTY” column will show the original amount of fluid in the inventory item’s container. These details can be seen in the next figure.



1. Inventory Fragment View

By finding the ratio of quantity left by original quantity, a notification can be sent to the user of the Phone App that will indicate when the ratio is ½ or below. This notification currently manifests as a bubble on the app icon and as a banner in the notification center. The notification can be given a title and a message. The current title is “High Priority”. The current message is “Running low on inventory”. This notification can be seen in the following figure.



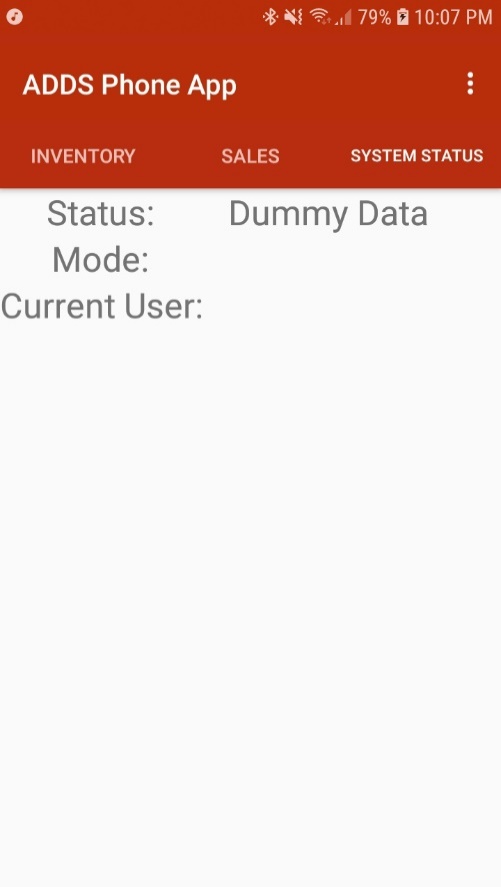
1. ADDS Phone App Notification

The sales fragment is meant to show the individual and total sales for a certain day. Each day of sales is displayed in the Spinner widget, which is essentially a dropdown menu with multiple items. The items are dynamically stored in the spinner because there is not a limit on how many shared data files the client will retrieve. For each shared data file, a item will be appended to the Spinner widget. The actual sales data will be plotted using a 3rd party Java library called “MPAndroidChart”. A bar graph was chosen as the graphical representation of the sales data because it was easy to implement and adapt for when a variable number of drinks are in the shared data file. The individual sales value is displayed at the top of each bar in the graph. The name of each drink is directly below their respective bars and the X-Axis. The bar graph is also capable of being zoomed in or out in the vertical direction, but not for the horizontal direction because it causes errors. The total sales are calculated by adding up the individual sales, and it is displayed with a TextView widget directly above the bar graph. This can be viewed in the following figure.



1. Sales Fragement View

The system status fragment uses just three TextView widgets to display the information gathered from the shared data file. The system status, system mode, and current user will be displayed next TextView labels. This information is important because it allows the employee to know how the machine is currently operating. This can be seen in the following figure.

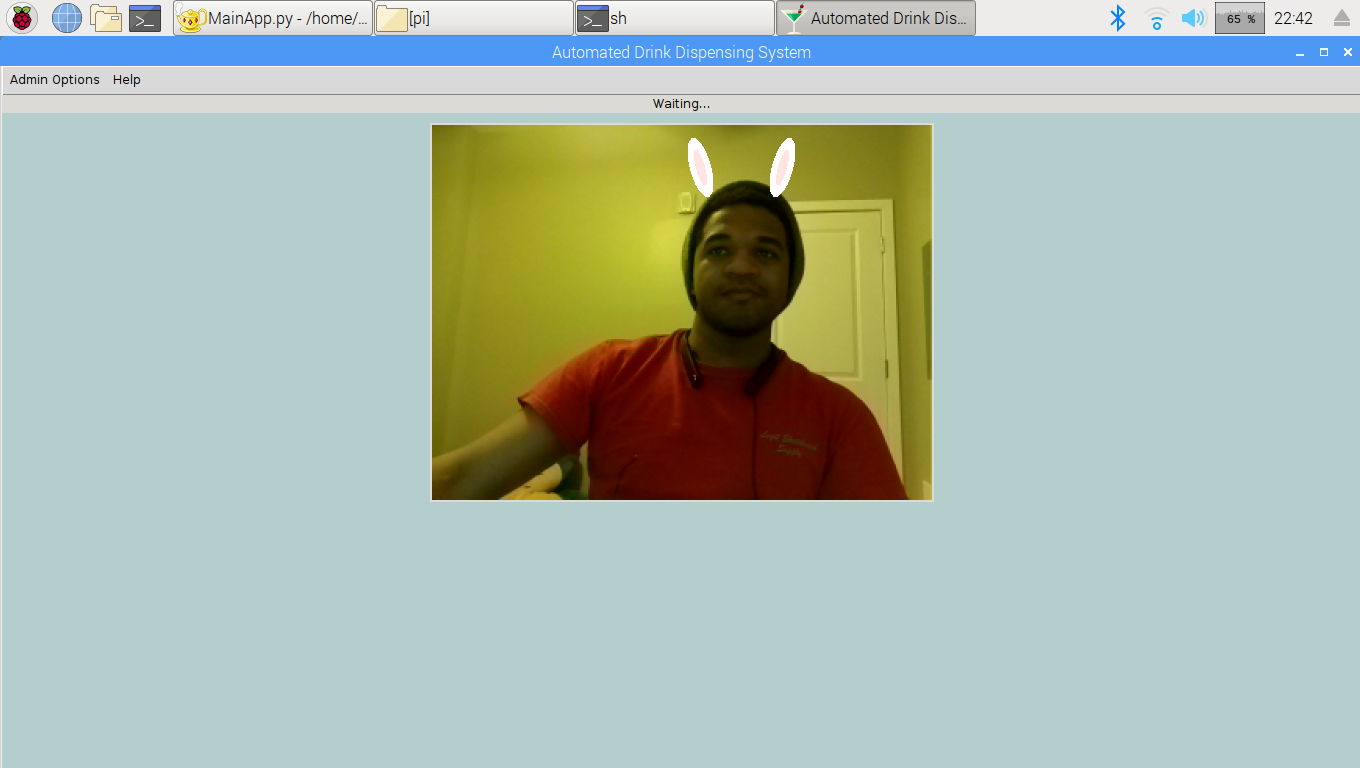


1. System Status Fragment View
   1. Implementation of Task 6

#### Subtask 1. Incorporate facial recognition into the desktop application

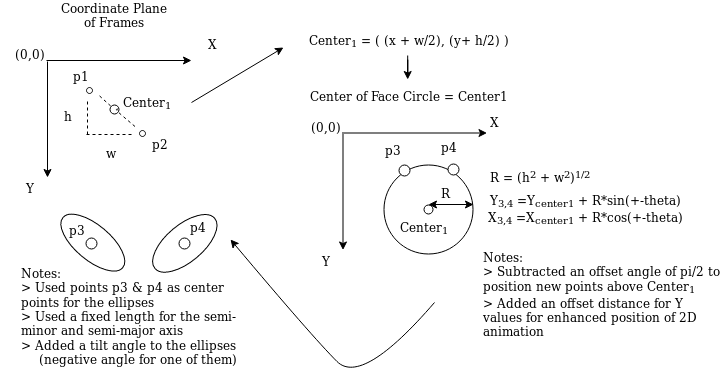
With the use of the Picamera peripheral device and the OpenCV (Open Computer Vision) library, facial recognition code was written to provide entertainment to users of the system. The Picamera acquired frames of images, and algorithms incorporating OpenCV facilities were used to edit the frames. The algorithms were based on first detecting a face within a frame and then drawing 2D animations onto the detected face. One animation even relied on detecting eyes within the subsection of the detected face.

The first 2D animation to be developed for the system was the bunny ears feature. The first step in this process was using a function in the OpenCV library that used Haar Cascade filters to detect faces within an image. The function returns likely candidates for a detected face within an image, which can be used to locate the center of a face and draw shapes onto it. The common drawing tools from OpenCV are circles, ellipses, lines, squares, and points. For the bunny ears, the center of the face was used to draw two concentric ellipses at the top of a user’s head.



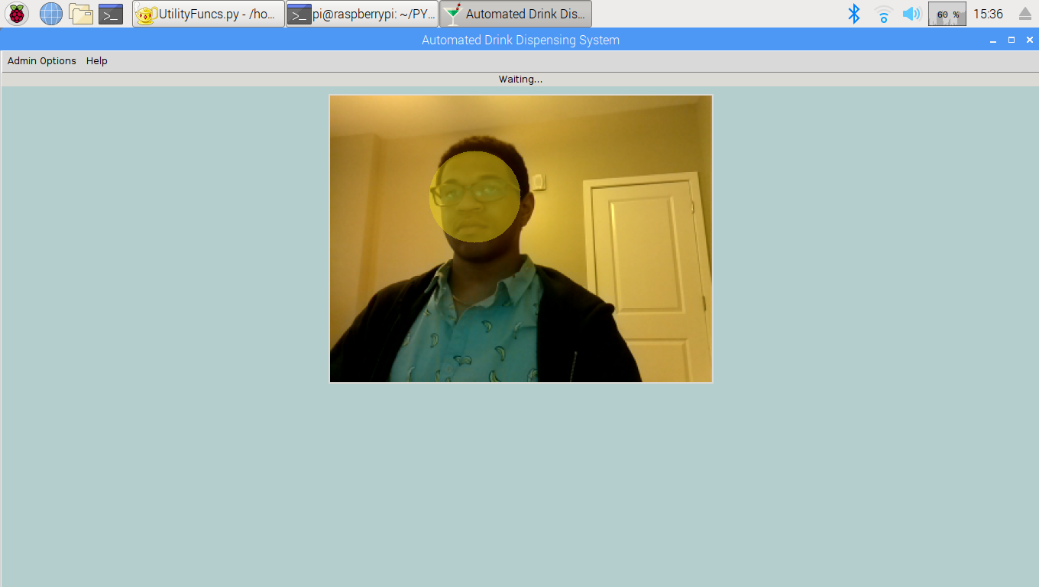
1. Bunny Ears Filter

The math for the bunny ears required doing planar math to calculate the correct position to place the ears at the head of the user. The plane of an image within the OpenCV library starts in the upper left corner of an image. The positive direction of the X and Y coordinates are in the south east direction. The math had to be done with respect to the unusual coordinate system. This is further explained and depicted in the following figure.



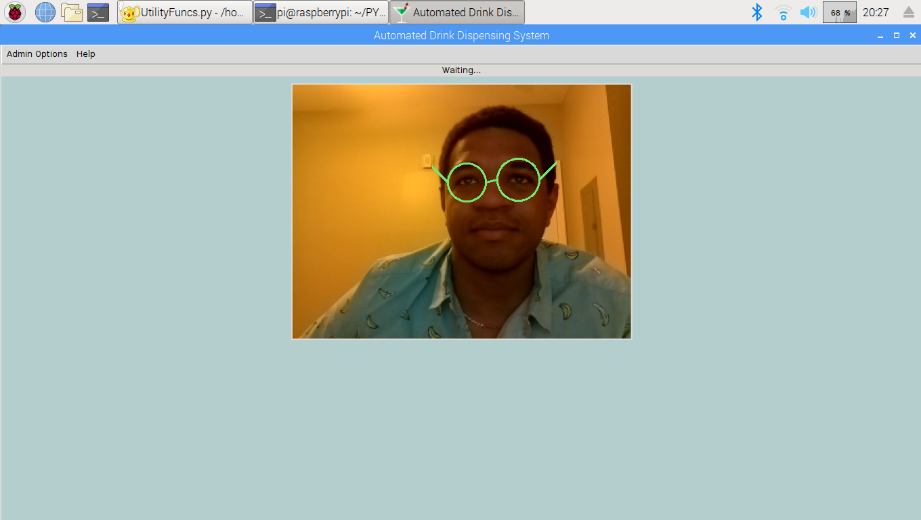
1. Bunny Ears Math

The next 2D animation was the rainbow mask. The rainbow mask required face detection within video frames as well. The filled circle function of the OpenCV library was used in order to shade the face of the user of the system. The color of the rainbow mask was randomized for every frame, so the mask can be any RGB color. For better animation effects, the rainbow mask was made to be transparent, so that the user can still see their face in the picture. One way to make a transparent mask was to perform a cross-dissolve operation on the original frame and the edited frame with the filled circle. The cross-dissolve operation was applied on every pixel in the image. The formula for the cross-dissolve is as follows: Pnew [i, j]= β\*P1[i, j] + ɑ\*P2[i, j]+ ɣ. The Pnew output value is the sum of scaled pixels from each frame plus an offset. To get the cross-dissolve effect with this formula, the gamma value should be 0 and the frame that contains the filled circle should have a scaled value that is one minus the other scaled value (i.e. β =1- ɑ ). This will weaken the influence of the filled circle image on the output pixel’s value, which makes it transparent like.



1. Rainbow Mask Filter

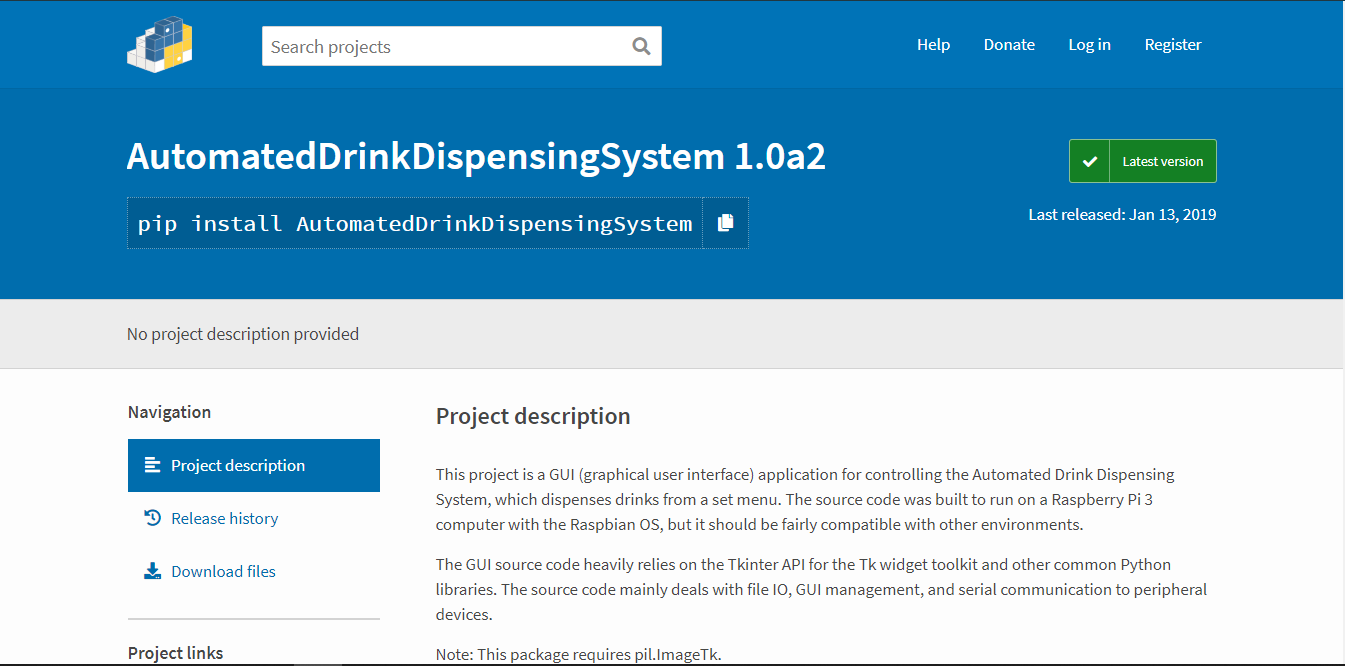
The final animation was the rainbow glasses. Like the rainbow mask, it relied on the face detection function of the OpenCV library. After faces were detected, the subregion of the face in the image frame was isolated used to detect eyes. This required a different Haar Cascade filter to recognize eyes. Once the eyes were recognized, circles were drawn to be concentric with the detected eyes. An algorithm was developed to figure out where to draw each piece of the frames of the drawn glasses. By using the distances between points at the circumference of each detected eye circle, the shortest distance between the right side of one circle and the left side of another circle would be denoted with a horizontal line, which would represent the bridge of the frame. The other points selected from the circle would help form the temples of the glasses (i.e. piece of a glasses frame that wraps around the ears). The temples were also lines, but they were drawn at an angle to represent hanging on the ears of the user.



1. Rainbow Glasses Filter

#### Subtask 2. Adapt the desktop application for compliance with Python Package Index

In order to make easy upgrades of the desktop software, the software was restructured to meet the requirements of the Python Package Index software repository. By storing current versions of the desktop application software in the Python Package Index, the desktop software can be downloaded by any computer in the world that has the PIP installer that comes with downloading the Python interpreter. The package is meant only for Python 3.5 and newer interpreters. The package is fully compatible with the Raspbian operating system, and it should be able to at least run on the Ubuntu operating system and its variants. In addition, when the package is downloaded with the PIP installer, all of the 3rd Party library dependencies will be installed too.



1. Desktop Application Package on PYPI
2. Conclusion

Throughout both semesters of our senior design project, our team has refined our initial ideas into a well-defined project that aims to solve a real-world problem. The final product that we created is a fully function prototype that can meet the requirements that we set out for it and adheres to the constraints that we defined. The system can successfully interact with an employee or customer user, take an order, make the order, and deliver the order to a customer. A user of the system can update their drink menu to fit varying needs. The Master Computer can effectively make sure that data cohesion is achieved between itself and the custom embedded boards.

Acknowledgement

Our team would like to acknowledge the following people and entities that supported our work. The Department of Engineering within the School of Business and Technology at University of Maryland Eastern Shore supported our work throughout the span of time that our team developed our ideas and proposed system. Our mentor for the project, Dr. Lei Zhang, guided our ideas, so that our team could make a well-thought out design and system. In addition, our team would like to acknowledge the help of Justin Derickson who helped with brainstorming the scope of the proposed system.

Appendix

1. Component Specs
2. Specs of the Raspberry Pi 3 Model B

A picture containing electronics

Description automatically generated

1. Raspberry Pi 3 Model B Specifications

A circuit board

Description automatically generated

1. Raspberry Pi 3 Model B General Purpose Input & Output Pins
2. Specs of the Embedded Board

* Voltage: 12 – 24V
* Current: 15 A
* Dimensions: 2” x 6.5” (l \* w)

1. Industry Standards

For this project, there were a few industry standards that were followed for the embedded system design and software design. The focus of the software design was on creating desktop applications and phone applications. Regarding the desktop application that was created, the Python language’s Python Enhancement Proposal (PEP) #8 was followed. PEP 8 specifies a style for writing Python scripts that encourages writing clean, maintainable code. This is very important for anybody who is making a large Python project because more readability and maintainability allows for easier testing, debugging, and adapting of a code base. Regarding the Android phone application, the best practices guide and quality guidelines on the Android Developer website were referenced, and it is maintained by the people who created the Android platform. The best practices guide gives tips about how to create Android phone applications that will be stable on multiple platforms with different hardware specifications. The quality guidelines specify how to handle features that deal with functionality, visual design, user interaction, compatibility, performance, security, and interacting with the google play store.

The embedded design followed several basic industry standards such as separating power and ground traces from the signal planes. The purpose of this is to reduce the electromagnetic interference that can occur as a result of the mixed power and ground rails among signal traces.

1. Source Code.
2. Source Code of the Desktop Application

#### MainApp.py

#!/usr/bin/env python3

"""

Programmer: Chris Blanks

Last Edited: 11/3/2018

Project: Automated Self-Serving System

Purpose: This script defines the MainApp class that runs everything.

"""

#Standard library imports

**import** tkinter **as** tk

**import** os

**import** time

**import** datetime

**from** threading **import** Timer

**from** cryptography**.**fernet **import** Fernet

**from** subprocess **import** check\_output

**import** subprocess

**import** \_thread **as** thread

**import** http**.**server **as** hs

**import** socketserver **as** ss

#My scripts

**from** CustomerWindow **import** CustomerWindow

**from** EmployeeWindow **import** EmployeeWindow

**import** DrinkProfile **as** dp\_class

**from** LoginWindow **import** LoginWindow

**from** KeyboardWindow **import** KeyboardWindow

**def** runMainApplication**():**

"""Basic run of application."""

**print(**os**.**getcwd**())**

root **=** tk**.**Tk**()**

main\_app **=** MainApp**(**master**=** root**)**

root**.**mainloop**()**

**class** **MainApp:**

#class member variables

isEmployeeMode**=** **False**

isValidLogin **=** **False**

drink\_profile\_directory **=** "/home/pi/PYTHON\_PROJECTS/automated-self-serving-system/drink\_profiles"

config\_file\_path **=** "/home/pi/PYTHON\_PROJECTS/automated-self-serving-system/system\_info/config.txt"

user\_login\_path**=** "/home/pi/PYTHON\_PROJECTS/automated-self-serving-system/system\_info/user\_login.txt"

encrypt\_key\_path **=** "/home/pi/PYTHON\_PROJECTS/automated-self-serving-system/system\_info/key.txt"

system\_info **=** "/home/pi/PYTHON\_PROJECTS/automated-self-serving-system/system\_info"

drink\_names **=** **[]**

isWithoutLogin **=** **False**

key **=** **True**

**def** \_\_init\_\_**(**self**,**master**):**

self**.**master **=** master

self**.**ip\_address **=** self**.**getIPAddress**()**

self**.**writeToLog**(**"Running main application"**)**

self**.**writeToSharedData**()**

self**.**startHTTPThread**()**

self**.**cipher\_suite **=** self**.**setupUserEncryption**()**

**if** self**.**isWithoutLogin **==** **True:**

self**.**createDefaultUserLoginFile**()**

self**.**drink\_objects **=** self**.**getDrinks**()**

self**.**createMainWindow**()**

self**.**selectWindow**()**

self**.**retrieveConfigurationInformation**()**

self**.**cleanOldDrinksFromConfig**()**

**def** selectWindow**(**self**):**

"""Determines what window is open."""

#input mode until GPIO pin is setup to trigger employee mode

selection **=** int**(**input**(**"Press 1 to enter employee mode."**))**

**if** selection **==** 1**:**

self**.**isEmployeeMode **=** **True**

self**.**launchLoginWindow**()**

**else:**

self**.**isEmployeeMode **=** **False**

self**.**master**.**withdraw**()**

self**.**createCustomerWindow**()**

**def** createMainWindow**(**self**):**

"""Displays main window elements. """

self**.**master**.**geometry**(**"200x200"**)**

self**.**master**.**geometry**(**"{0}x{1}+0+0"**.**format**(**self**.**master**.**winfo\_screenwidth**()**

**,**self**.**master**.**winfo\_screenheight**()))**

self**.**main\_title **=** tk**.**Label**(**self**.**master**,**text**=**"Root Window"**)**

self**.**main\_title**.**grid**()**

self**.**customer\_window\_btn **=** tk**.**Button**(**self**.**master**,**text**=**"customer window"

**,**command**=** **lambda** window**=**"customer"**:** self**.**relaunchWindow**(**window**))**

self**.**customer\_window\_btn**.**grid**()**

self**.**employee\_window\_btn **=** tk**.**Button**(**self**.**master**,**text**=**"employee window"

**,**command**=** **lambda** window**=**"employee"**:** self**.**relaunchWindow**(**window**))**

self**.**employee\_window\_btn**.**grid**()**

**def** relaunchWindow**(**self**,**window**):**

""" Relaunches the selected window."""

**if** window **==** "customer"**:**

self**.**isEmployeeMode **=** **False**

self**.**master**.**withdraw**()**

self**.**createCustomerWindow**()**

**elif** window **==** "employee"**:**

self**.**isEmployeeMode **=** **True**

self**.**master**.**withdraw**()**

self**.**launchLoginWindow**()**

**else:**

**print(**"What the heck?"**)**

**def** createCustomerWindow**(**self**):**

"""Creates separate customer window."""

self**.**customer\_top\_lvl **=** tk**.**Toplevel**(**self**.**master**)**

self**.**customer\_window **=** CustomerWindow**(**self**)**

**def** createEmployeeWindow**(**self**,**isAdminMode**):**

"""Creates separate employee window """

self**.**employee\_top\_lvl **=** tk**.**Toplevel**(**self**.**master**)**

self**.**employee\_window **=** EmployeeWindow**(**self**,**isAdminMode**)**

**def** launchLoginWindow**(**self**):**

"""Launches login window when employee mode is selected."""

self**.**login\_top\_lvl **=** tk**.**Toplevel**(**self**.**master**)**

self**.**login\_window **=** LoginWindow**(**self**)**

**def** launchKeyboardWindow**(**self**):**

"""Launches a top level window that contains a keyboard that can deliver

input to processes that need it."""

self**.**keyboard\_top\_lvl **=** tk**.**Toplevel**(**self**.**master**)**

self**.**keyboard\_window **=** KeyboardWindow**(**self**)**

**def** getDrinks**(**self**):**

"""Retrieves a list of active Drink objects."""

temp **=** **[]**

self**.**all\_defined\_drinks **=** **[]**

os**.**chdir**(**self**.**drink\_profile\_directory**)**

drink\_profile\_names **=** os**.**listdir**(**os**.**getcwd**())**

**for** name **in** drink\_profile\_names**:**

path\_builder **=** self**.**drink\_profile\_directory **+**"/"**+** name

os**.**chdir**(**path\_builder**)**

text\_file\_index **=** 0

**if** ".txt" **in** os**.**listdir**(**os**.**getcwd**())[**0**]:**

**pass**

**else:**

text\_file\_index**=** 1

drink **=** dp\_class**.**DrinkProfile**(**path\_builder **+**"/"**+** os**.**listdir**(**os**.**getcwd**())[**text\_file\_index**])**

**if** drink**.**isActive **==** "1"**:**

drink**.**name **=** **(**drink**.**name**).**replace**(**" "**,**"\_"**)**

drink**.**addDrinkToConfig**()**

temp**.**append**(**drink**)**

self**.**all\_defined\_drinks**.**append**(**drink**)**

#go back to SENIOR\_DESIGN\_CODE directory

os**.**chdir**(**".."**)**

os**.**chdir**(**".."**)**

**return** temp

**def** retrieveConfigurationInformation**(**self**):**

"""Retrieves configuration info (e.g. drink names) from config file """

f **=** open**(**self**.**config\_file\_path**,**'r+'**)**

lines **=** f**.**read**().**splitlines**()**

#print("'{}' contents:\n".format((f.name).split("/")[-1]),'\n'.join(lines))

line\_number **=** 1

**for** line **in** lines**:**

**if** line\_number **==** 1**:**

**if** line**.**split**()[**1**]** **==** '0'**:**

**print(**"Config file is not locked.\n\n"**)**

**else:**

self**.**isLocked **=** **True**

**print(**"Config file is locked.\n\n"**)**

**if** line\_number **==** 2**:**

drinks **=** line**.**split**(**" "**)**

**for** i **in** range**(**len**(**drinks**)-**1**):**

self**.**drink\_names**.**append**(**drinks**[**i**+**1**])**

line\_number**+=**1

f**.**truncate**()**

f**.**close**()**

**def** updateConfigurationFile**(**self**,**item\_to\_update**,**updated\_value**=** **None):**

""" """

f **=** open**(**self**.**config\_file\_path**,**"r+"**)**

lines **=** f**.**read**().**splitlines**()**

f**.**seek**(**0**)**

line\_headers **=** **[**"locked "**,**"active\_drink\_list "**,**"system\_status "**]**

line\_to\_edit **=** 0

**if** item\_to\_update **==** "data\_lock"**:**

line\_to\_edit **=** 1

**if** item\_to\_update **==** "drink\_list"**:**

line\_to\_edit **=** 2

**if** item\_to\_update **==** "system\_status"**:**

line\_to\_edit **=** 3

line\_number **=** 1

**for** line **in** lines**:**

**if** line\_number **==** line\_to\_edit **and** updated\_value **!=** **None:**

line **=** line\_headers**[**line\_to\_edit **-** 1**]** **+** updated\_value

f**.**write**(**line**+**"\n"**)**

**else:**

f**.**write**(**line**+**"\n"**)**

**if** line\_number **==** 3**:**

**break**

line\_number**+=**1

f**.**truncate**()**

f**.**close**()**

**def** cleanOldDrinksFromConfig**(**self**):**

"""Updates the active drinks in the config file."""

cleaned\_list\_of\_names **=** ""

loaded\_drink\_object\_names **=** **[]**

**for** drink **in** self**.**drink\_objects**:**

loaded\_drink\_object\_names**.**append**((**drink**.**name**).**replace**(**"\_"**,**" "**))**

**for** config\_name **in** self**.**drink\_names**:**

**if** config\_name**.**replace**(**"\_"**,**" "**)** **in** loaded\_drink\_object\_names**:**

cleaned\_list\_of\_names **=** cleaned\_list\_of\_names **+** config\_name **+** " "

self**.**updateConfigurationFile**(**"drink\_list"**,**cleaned\_list\_of\_names**)**

self**.**writeToLog**(**"Cleaned Config file."**)**

**def** writeToLog**(**self**,** message**):**

"""Writes messages into the log.txt file."""

self**.**todays\_log **=** self**.**system\_info**+**"/log\_files/log\_on\_"**+**str**(**datetime**.**date**.**today**())+**".txt"

log **=** open**(**self**.**todays\_log**,**"w+"**)**

full\_msg **=** str**(**datetime**.**datetime**.**now**())** **+**" : " **+** message

log**.**seek**(**2**)**

log**.**write**(**full\_msg **+** "\n"**)**

log**.**close**()**

**def** writeToDrinkSalesLog**(**self**,** message**):**

"""Writes time-stamped sales info into a log for each day."""

self**.**todays\_drink\_sales **=** self**.**system\_info**+**"/drink\_sales/drink\_sales\_"**+**str**(**datetime**.**date**.**today**())+**".txt"

log **=** open**(**self**.**todays\_drink\_sales**,**"a"**)**

full\_msg **=** str**(**datetime**.**datetime**.**now**())** **+**" : " **+** message

log**.**seek**(**2**)**

log**.**write**(**full\_msg **+** "\n"**)**

log**.**close**()**

**def** addUserToLogin**(**self**,**user\_type**,**username**,**password**):**

"""Creates a new user in the user\_login file. Ex: self.addUserToLogin("R","Lei","Zhang")"""

file **=** open**(**self**.**user\_login\_path**,**"rb+"**)**

lines **=** file**.**read**().**splitlines**()**

file**.**seek**(**0**)**

line\_num **=** 1

next\_line **=** **False**

**if** user\_type **==** "A"**:**

**for** line **in** lines**:**

line **=** str**((**self**.**cipher\_suite**.**decrypt**(**line**)).**decode**(**'ASCII'**))**

**if** next\_line **==** **True:**

file**.**write**(**temp**+**b"\n"**)**

next\_line **=** **False**

**if** "ADMIN USER" **in** line**:**

temp **=** username **+**" "**+** password

temp **=** self**.**cipher\_suite**.**encrypt**(**temp**.**encode**(**encoding**=**'UTF-8'**))**

next\_line **=** **True**

**if** "END" **in** line**:**

line **=** self**.**cipher\_suite**.**encrypt**(**line**.**encode**(**encoding**=**'UTF-8'**))**

file**.**write**(**line**+**b"\n"**)**

**break**

line **=** self**.**cipher\_suite**.**encrypt**(**line**.**encode**(**encoding**=**'UTF-8'**))**

file**.**write**(**line**+**b"\n"**)**

**else:**

**for** line **in** lines**:**

line **=** str**((**self**.**cipher\_suite**.**decrypt**(**line**)).**decode**(**'ASCII'**))**

**if** "REGULAR USER" **in** line**:**

line **=** line **+**"\n"**+** username **+**" "**+** password

**if** "END" **in** line**:**

line **=** self**.**cipher\_suite**.**encrypt**(**line**.**encode**(**encoding**=**'UTF-8'**))**

file**.**write**(**line**+**b"\n"**)**

**break**

line **=** self**.**cipher\_suite**.**encrypt**(**line**.**encode**(**encoding**=**'UTF-8'**))**

file**.**write**(**line**+**b"\n"**)**

file**.**flush**()**

file**.**truncate**()**

file**.**close**()**

msg **=** "Added "**+**username**+** " account to login."

self**.**writeToLog**(**msg**)**

**def** deleteUserFromLogin**(**self**,** username**,** password**):**

"""Deletes a user in the user\_login file (besides the original admin account)."""

file **=** open**(**self**.**user\_login\_path**,**"rb+"**)**

lines **=** file**.**read**().**splitlines**()**

file**.**seek**(**0**)**

login\_combo **=** username **+**" " **+** password

**print(**login\_combo**)**

**for** line **in** lines**:**

line **=** str**((**self**.**cipher\_suite**.**decrypt**(**line**)).**decode**(**'ASCII'**))**

**print(**"users "**+** line**)**

**if** "END" **in** line**:**

line **=** self**.**cipher\_suite**.**encrypt**(**line**.**encode**(**encoding**=**'UTF-8'**))**

file**.**write**(**line**+**b"\n"**)**

**break**

**if** "REGULAR" **in** line**:**

line **=** self**.**cipher\_suite**.**encrypt**(**"REGULAR USER"**.**encode**(**encoding**=**'UTF-8'**))**

file**.**write**(**line**+**b"\n"**)**

**elif** login\_combo **not** **in** str**(**line**):**

**print(**"rest:"**+**str**(**line**)** **)**

line **=** self**.**cipher\_suite**.**encrypt**(**line**.**encode**(**encoding**=**'UTF-8'**))**

file**.**write**(**line**+**b"\n"**)**

**else:**

**print(**line**)**

**pass**

file**.**truncate**()**

file**.**flush**()**

file**.**close**()**

msg **=** "Removed "**+**username**+** " account from login."

self**.**writeToLog**(**msg**)**

**def** setupUserEncryption**(**self**):**

"""Creates an encryption key if one is not already made """

**if** os**.**path**.**exists**(**self**.**encrypt\_key\_path**):**

file **=** open**(**self**.**encrypt\_key\_path**,**"rb"**)**

key **=** file**.**readline**()**

**if** os**.**path**.**exists**(**self**.**user\_login\_path**):**

**pass**

**else:**

self**.**isWithoutLogin **=** **True**

file**.**close**()**

**else:**

key **=** Fernet**.**generate\_key**()**

file **=** open**(**self**.**encrypt\_key\_path**,**"wb"**)**

file**.**write**(**key**)**

file**.**close**()**

self**.**isWithoutLogin **=** **True**

cipher\_suite **=** Fernet**(**key**)**

**return** cipher\_suite

**def** createDefaultUserLoginFile**(**self**):**

"""Creates a default user\_login.txt file that has the

default login credentials in an encrypted form. """

**print(**"Making new login file."**)**

new\_user\_login **=** open**(**self**.**user\_login\_path**,**'wb'**)**

admin\_str **=** "ADMIN USER"

default\_usr\_str**=**"admin admin"

reg\_usr\_str **=** "REGULAR USER"

end\_str **=** "END"

admin\_str **=** self**.**cipher\_suite**.**encrypt**(**admin\_str**.**encode**(**encoding**=**'UTF-8'**))**

default\_usr\_str **=** self**.**cipher\_suite**.**encrypt**(**default\_usr\_str**.**encode**(**encoding**=**'UTF-8'**))**

reg\_usr\_str **=** self**.**cipher\_suite**.**encrypt**(**reg\_usr\_str**.**encode**(**encoding**=**'UTF-8'**))**

end\_str **=** self**.**cipher\_suite**.**encrypt**(**end\_str**.**encode**(**encoding**=**'UTF-8'**))**

new\_user\_login**.**write**(**admin\_str**+**b"\n"**)**

new\_user\_login**.**write**(**default\_usr\_str**+**b"\n"**)**

new\_user\_login**.**write**(**reg\_usr\_str**+**b"\n"**)**

new\_user\_login**.**write**(**end\_str**+**b"\n"**)**

new\_user\_login**.**truncate**()**

new\_user\_login**.**close**()**

self**.**isWithoutLogin **=** **False**

#self.writeToLog("Made new login file")

**def** getIPAddress**(**self**):**

host\_info **=** check\_output**([**'hostname'**,**'-I'**])**

ip\_address **=** host\_info**.**split**()[**0**].**decode**(**'ascii'**)**

**print(**"IP: "**+**ip\_address**)**

**return** ip\_address

**def** writeToSharedData**(**self**):**

"""Updates the write to shared data every 5 minutes."""

MILLI **=** 1000

MIN\_IN\_SEC **=** 180 # 300 sec = 5 min

now **=** time**.**strftime**(**"%H:%M:%S"**)**

**print(**now**)** #keeps track of write time

self**.**todays\_shared\_data**=** self**.**system\_info**+**"/shared\_data/shared\_data\_"**+**str**(**datetime**.**date**.**today**())+**".txt"

log **=** open**(**self**.**todays\_shared\_data**,**"w"**)**

msg **=** """STATUS

idle

employee

chris

INVENTORY

rum 300 1000

vodka 300 900

tequila 300 1000

gin 1000 1000

triple\_sec 300 400

soda\_water 200 1500

SALES

cuba\_libre 55.00

daiquiri 10.00

kamikaze 0.00

long\_island\_iced\_tea 23.00

naval\_gimlet 5.00

rum\_and\_coke 10.00

screwdriver 10.00

tequila 0.00

vodka 100.00

vodka\_and\_cranberry 20.00"""

msg2 **=** """STATUS

mixing

employee

admin

INVENTORY

rum 300 1000

vodka 500 900

tequila 300 1000

gin 1000 1000

triple\_sec 300 400

soda\_water 200 1500

SALES

cuba\_libre 25.00

daiquiri 10.00

kamikaze 10.00

long\_island\_iced\_tea 23.00

naval\_gimlet 5.00

rum\_and\_coke 10.00

screwdriver 10.00

tequila 0.00

vodka 19.00

vodka\_and\_cranberry 20.00"""

**if** self**.**key **==** **True:**

log**.**write**(**msg**)**

**else:**

log**.**write**(**msg2**)**

log**.**truncate**()**

log**.**close**()**

self**.**key **=** **not** self**.**key

self**.**master**.**after**(**MIN\_IN\_SEC**\***MILLI**,**self**.**writeToSharedData**)** #recursively writes to shared data every 5 minutes

**def** startHTTPThread**(**self**):**

"""Creates a http server in a separate thread from the GUI."""

thread**.**start\_new\_thread**(**self**.**startHTTPServer**,**tuple**())**

**def** startHTTPServer**(**self**):**

"""Opens a http server in the shared data directory."""

**try:**

os**.**chdir**(**"/home/pi/PYTHON\_PROJECTS/automated-self-serving-system/system\_info/shared\_data"**)**

subprocess**.**call**([**"sudo"**,** "python"**,** "-m"**,** "SimpleHTTPServer"**,**"80"**])**

**except** PermissionError **as** err**:**

**print(**"Port is already open."**)** #printed in the abyss

os**.**chdir**(**".."**)**

os**.**chdir**(**".."**)**

**if** \_\_name\_\_ **==** "\_\_main\_\_"**:**

runMainApplication**()**

#### ApplicationUserEditor.py

#!/usr/bin/env python3

"""

Programmer: Chris Blanks

Last Edited: 11/3/2018

Project: Automated Self-Serving System

Purpose: This script defines the ApplicationUserEditor class that controls

user login information.

"""

**from** tkinter **import** messagebox

**import** tkinter **as** tk

**from** EmbeddedKeyboard **import** EmbeddedKeyboard

**class** **ApplicationUserEditor:**

**def** \_\_init\_\_**(**self**,**master**,**main\_app**,**size**):**

self**.**master **=** master

self**.**main\_app **=** main\_app

self**.**width **=** size**[**0**]**

self**.**height **=** size**[**1**]**

self**.**font **=** **(**"Georgia"**,**12**,**""**)**

self**.**users **=** self**.**getUsers**()**

self**.**configureWindow**()**

**def** configureWindow**(**self**):**

self**.**master**.**configure**(**width**=**self**.**width**,**height**=**self**.**height**)**

self**.**master**.**protocol**(**"WM\_DELETE\_WINDOW"**,**self**.**deployExitMessageBox**)**

self**.**addWindowElements**()**

**def** getUsers**(**self**):**

"""Gets a list of users."""

file **=** open**(**self**.**main\_app**.**user\_login\_path**,**"rb+"**)**

lines **=** file**.**read**().**splitlines**()**

file**.**close**()**

user\_list **=** ""

**for** line **in** lines**:**

line **=** **(**self**.**main\_app**.**cipher\_suite**.**decrypt**(**line**)).**decode**(**'ASCII'**)**

**if** "END" **not** **in** line**:**

user\_list **=** user\_list **+** line**+** "\n"

**return** user\_list

**def** addWindowElements**(**self**):**

self**.**user\_list\_label **=** tk**.**Label**(**self**.**master**,**text**=**"Current Users:"**,**

font**=**self**.**font**)**

self**.**user\_list\_label**.**grid**(**row**=**0**,**column**=**0**)**

self**.**users\_in\_gui **=** tk**.**Label**(**self**.**master**,**text**=**self**.**users**,**font**=**self**.**font**)**

self**.**users\_in\_gui**.**grid**(**row**=**1**,**column**=**0**)**

self**.**add\_button **=** tk**.**Button**(**self**.**master**,**text**=**"Add"**,**bg**=**"green"**,**

command**=lambda** x**=**"Add"**:** self**.**launchEditorWindow**(**x**,**"Add a user"**))**

self**.**add\_button**.**grid**(**row**=**0**,**column**=**1**)**

self**.**delete\_button **=** tk**.**Button**(**self**.**master**,**text**=**"Delete"**,**bg**=**"red"**,**

command**=lambda** x**=**"Delete"**:** self**.**launchEditorWindow**(**x**,**"Delete a user"**))**

self**.**delete\_button**.**grid**(**row**=**0**,**column**=**2**)**

**def** launchEditorWindow**(**self**,**button\_type**,**title**):**

self**.**top **=** tk**.**Toplevel**(**self**.**master**)**

self**.**top**.**title**(**title**)**

self**.**top**.**geometry**(**"{0}x{1}+0+0"**.**format**(**self**.**width**,**self**.**height**))**

username\_label **=** tk**.**Label**(**self**.**top**,**text**=**"Username"**,**font**=**self**.**font**)**

username\_label**.**grid**(**row**=**0**,**column **=** 0**)**

self**.**user\_name\_entry **=** tk**.**Entry**(**self**.**top**,**width**=**175**)**

self**.**user\_name\_entry**.**grid**(**row**=**0**,**column**=**1**)**

pass\_label **=** tk**.**Label**(**self**.**top**,**text**=**"Password"**,**font**=**self**.**font**)**

pass\_label**.**grid**(**row**=**1**,**column **=** 0**)**

self**.**pass\_entry **=** tk**.**Entry**(**self**.**top**,**width**=**175**)**

self**.**pass\_entry**.**grid**(**row**=**1**,**column**=**1**)**

**if** button\_type **==** "Delete"**:**

self**.**setupDeleteItems**()**

**elif** button\_type **==** "Add"**:**

self**.**setupAddItems**()**

**else:**

**print(**"How?"**)**

entries **=** **[**self**.**user\_name\_entry**,**self**.**pass\_entry**]**

keyboard\_canvas **=** tk**.**Canvas**(**self**.**top**,**width**=**350**,**height**=**350**)**

embed\_keyboard **=** EmbeddedKeyboard**(**keyboard\_canvas**,**entries**)**

keyboard\_canvas**.**grid**(**column**=**1**,**sticky**=**"s"**)**

**def** setupDeleteItems**(**self**):**

self**.**done **=** tk**.**Button**(**self**.**top**,**text**=**"Done"**,**bg**=**"orange"**,**

command**=lambda** x**=**"delete"**:** self**.**deployDoneMessageBox**(**x**))**

self**.**done**.**grid**(**row**=**0**,**column**=**2**,**sticky**=**"nsew"**,**rowspan**=**2**)**

**def** setupAddItems**(**self**):**

self**.**done **=** tk**.**Button**(**self**.**top**,**text**=**"Done"**,**bg**=**"orange"**,**

command**=lambda** x**=**"add"**:** self**.**deployDoneMessageBox**(**x**))**

self**.**checkbut\_state **=** tk**.**IntVar**()**

self**.**admin\_check **=** tk**.**Checkbutton**(**self**.**top**,**text**=**"Admin user?"**,**variable**=**self**.**checkbut\_state**)**

self**.**admin\_check**.**grid**(**row**=**0**,**column**=**2**)**

self**.**done**.**grid**(**row**=**1**,**column**=**2**,**sticky**=**"nsew"**,**rowspan**=**2**)**

**def** deployDoneMessageBox**(**self**,**button\_type**):**

"""Destroys window and brings back previous window."""

**if** messagebox**.**askokcancel**(**"Done"**,**"Are you sure?"**):**

**if** button\_type **==** "add"**:**

username **=** self**.**user\_name\_entry**.**get**()**

password **=** self**.**pass\_entry**.**get**()**

user\_type **=** ""

**if** self**.**checkbut\_state**.**get**()** **==** 1**:**

user\_type **=** "A"

self**.**main\_app**.**addUserToLogin**(**user\_type**,**username**,**password**)**

**elif** button\_type **==**"delete"**:**

username **=** self**.**user\_name\_entry**.**get**()**

password **=** self**.**pass\_entry**.**get**()**

self**.**main\_app**.**deleteUserFromLogin**(**username**,**password**)**

self**.**top**.**destroy**()**

self**.**master**.**destroy**()**

**def** deployExitMessageBox**(**self**):**

"""Destroys window and brings back previous window."""

**if** messagebox**.**askokcancel**(**"Quit"**,**"Are you sure?"**):**

self**.**master**.**destroy**()**

#### AppWindow.py

#!/usr/bin/env python3

"""

Programmer: Chris Blanks

Last Edited: 11/3/2018

Project: Automated Self-Serving System

Purpose: This script defines the AppWindow Class, which

the Employee and Customer windows inherit from.

Note:

-The spacing of the buttons is inconsistent when the drink profile is up for either

employee or customer mode. (seems to be dependent on columnspan of ingredient label)

"""

**from** tkinter **import** messagebox

**import** tkinter **as** tk

**from** PIL **import** Image

**from** PIL **import** ImageTk

**class** **AppWindow():**

background\_color **=** "LightCyan3"

bg\_color\_other **=** "mint cream"

var **=** 10

**def** \_\_init\_\_**(**self**,**main\_app**):**

"""Provides basic functionality to each window of the main application."""

self**.**main\_app\_instance **=** main\_app

**pass**

**def** displayDrinkOptionsInGUI**(**self**):**

"""Displays each drink button/image/label in the GUI."""

drink\_num **=** 0

column\_position **=** 0

row\_position **=** 0

self**.**drink\_option\_references **=** **[]**

**for** drink **in** self**.**main\_app**.**drink\_objects**:**

**if** column\_position **>** 4**:**

row\_position **=** 2

column\_position **=** 0 #resets column position to fit all buttons

drink\_img **=** Image**.**open**(**drink**.**pic\_location**)**

drink\_img **=** drink\_img**.**resize**((**200**,**200**),**Image**.**ANTIALIAS**)**

drink\_photo **=** ImageTk**.**PhotoImage**(**drink\_img**)**

self**.**drink\_button **=** tk**.**Button**(**self**.**frame**,**image**=**drink\_photo**,**bg**=**"white"

**,**command**=lambda** drink\_op**=** self**.**main\_app**.**drink\_objects**[**drink\_num**]:** self**.**setupDrinkEvent**(**drink\_op**))**

self**.**drink\_button**.**img\_ref **=** drink\_photo

self**.**drink\_button**.**grid**(**row **=**row\_position**,**column**=**column\_position**,** padx **=** 32

**,**pady **=** 15**)**

drink**.**name **=(**drink**.**name**).**replace**(**"\_"**,**" "**)**

self**.**drink\_label **=** tk**.**Label**(**self**.**frame**,**text**=(**drink**.**name**).**title**(),**

font**=(**"Georgia"**,**"15"**,**"bold"**),**bg**=**self**.**bg\_color\_other**)**

self**.**drink\_label**.**grid**(**row**=**row\_position**+**1**,**column**=**column\_position**)**

self**.**drink\_option\_references**.**append**(** **(**self**.**drink\_button**,**self**.**drink\_label**)** **)**

column\_position **=** column\_position **+** 1

drink\_num **=** drink\_num **+** 1

**def** setupDrinkEvent**(**self**,**drink\_option**):**

"""Changes current drink before initiating drink event."""

self**.**current\_drink **=** drink\_option

self**.**initiateDrinkEvent**()**

**def** initiateDrinkEvent**(**self**):**

"""Initiates drink event """

**print(**"Drink #"**,**int**(**self**.**current\_drink**.**id\_number**)+**1**,**": "**,(**self**.**current\_drink**.**name**).**replace**(**"\_"**,**" "**))**

self**.**clearDrinkOptionsFromGUI**()**

self**.**setupDrinkProfileInGUI**()**

**def** clearDrinkOptionsFromGUI**(**self**):**

"""Clears drink option items in GUI in order to make room for the next window."""

**for** item **in** self**.**drink\_option\_references**:**

item**[**0**].**grid\_forget**()**

item**[**1**].**grid\_forget**()**

**def** setupDrinkProfileInGUI**(**self**):**

"""Creates a drink profile for the current drink."""

self**.**drink\_profile\_elements **=** **[]**

img **=** Image**.**open**(**self**.**current\_drink**.**pic\_location**)**

img **=** img**.**resize**((**500**,**400**),**Image**.**ANTIALIAS**)**

tk\_photo **=** ImageTk**.**PhotoImage**(**img**)**

self**.**img\_item\_reference **=** tk\_photo #keeping a reference allows photo to display

img\_item **=** tk**.**Label**(**self**.**frame**,**image**=**tk\_photo**)**

img\_item**.**grid**(**row**=**0**,**column**=**0**)**

name\_of\_drink **=** tk**.**Label**(**self**.**frame**,**text**=(**self**.**current\_drink**.**name**).**title**(),**font**=(**"Georgia"**,**30**,**"bold"**))**

name\_of\_drink**.**grid**(**row**=**1**,**column**=**0**)**

text\_builder **=**" "**.**join**(**self**.**current\_drink**.**ingredients**).**replace**(**' '**,**', '**).**replace**(**'\_'**,**' '**)**

ingredient\_text **=** tk**.**Label**(**self**.**frame**,**text**=**"Ingredients: " **+** text\_builder**,**font**=(**"Georgia"**,**14**,**"bold"**))**

ingredient\_text**.**grid**(**row**=**0**,**column **=** 1**,**columnspan**=**10**,**sticky**=**"n"**)**

**if** self**.**main\_app**.**isEmployeeMode **==** **False:**

drink\_price\_str **=** "Price: $"**+**str**(**self**.**current\_drink**.**price**)**

drink\_price **=** tk**.**Label**(**self**.**frame**,**text**=** drink\_price\_str**,**font**=(**"Georgia"**,**16**,**"bold"**),**fg**=**"green"**)**

drink\_price**.**grid**(**row**=**1**,**column**=**2**)**

buy\_button **=** tk**.**Button**(**self**.**frame**,**text**=**"Buy?"**,**bg**=**"Green"**,**fg**=**"white"**,**command**=**self**.**startBuyEvent**)**

buy\_button**.**grid**(**row**=**2**,**column**=**2**,**sticky**=**"nsew"**)**

self**.**drink\_profile\_elements**.**extend**((**buy\_button**,**drink\_price**))**

**else:**

quantity\_label **=** tk**.**Label**(**self**.**frame**,**text**=**"Order Quantity:"**,**font**=(**"Georgia"**,**16**,**"bold"**))**

quantity\_label**.**grid**(**row**=**2**,**column**=**1**,**sticky**=**"n"**)**

self**.**drink\_profile\_elements**.**append**(**quantity\_label**)**

**for** i **in** range**(**5**):**

quantity\_btn **=** tk**.**Button**(**self**.**frame**,**text**=**str**(**i**+**1**),**bg**=**"white"**,**fg**=**"green"**,**font**=(**"Arial"**,**12**,**"bold"**),**

command**=** **lambda** x **=** i**+**1**:** self**.**startEmployeeOrderEvent**(**x**)** **)**

quantity\_btn**.**configure**(**height**=**2**,**width**=**2**)**

quantity\_btn**.**grid**(**row**=** 2**,**column**=**i**+**2**,**padx**=**6**,**sticky**=**"w"**)**

self**.**drink\_profile\_elements**.**append**(**quantity\_btn**)**

back\_button **=** tk**.**Button**(**self**.**frame**,** text**=**"Back"**,**bg**=**"white"**,**fg**=**"red"**,**command**=**self**.**resetDrinkOptions**)**

back\_button**.**grid**(**row**=**3**,**column**=**0**)**

self**.**drink\_profile\_elements**.**extend**((**img\_item**,**name\_of\_drink**,**ingredient\_text**,**back\_button**))**

**def** startBuyEvent**(**self**):**

"""Starts the buying process for the customer mode."""

self**.**isOrdered **=** self**.**displayConfirmationMessageBox**()**

**if** self**.**isOrdered**:**

**pass**

**def** startEmployeeOrderEvent**(**self**,**num\_of\_drinks**):**

"""Starts the ordering process for the employee mode."""

self**.**isOrdered **=** self**.**displayConfirmationMessageBox**(**"Employee"**,**num\_of\_drinks**)**

**if** self**.**isOrdered**:**

**pass**

**def** displayConfirmationMessageBox**(**self**,**mode**=**"Customer"**,**num\_of\_drinks**=**1**):**

"""Asks the user if they are sure about their drink selection """

**if** mode **==** "Customer"**:**

**if** messagebox**.**askokcancel**(**"Confirmation"**,**"Are you sure that you want a "**+**self**.**current\_drink**.**name**+**"?"**):**

**print(**"Order is confirmed."**)**

**print(**"One order of "**+**self**.**current\_drink**.**name **+**" on the way."**)**

msg **=** "1 "**+** self**.**current\_drink**.**name **+** " was ordered."

self**.**main\_app\_instance**.**writeToDrinkSalesLog**(**msg**)**

**return** **True**

**else:**

**return** **False**

**else:**

**if** messagebox**.**askokcancel**(**"Confirmation"**,**

"Are you sure that you want "**+**str**(**num\_of\_drinks**)+**" "**+**self**.**current\_drink**.**name**.**title**().**replace**(**"\_"**,**" "**)+**"(s) ?"**):**

**print(**"Order is confirmed."**)**

**print(** str**(**num\_of\_drinks**)+**" order(s) of "**+**self**.**current\_drink**.**name **+**" on the way."**)**

**if** num\_of\_drinks **==** 1**:**

msg **=** str**(**num\_of\_drinks**)+**" "**+** self**.**current\_drink**.**name **+** " was ordered."

**elif** num\_of\_drinks **>** 1**:**

msg **=** str**(**num\_of\_drinks**)+**" "**+** self**.**current\_drink**.**name **+** "s were ordered."

self**.**main\_app\_instance**.**writeToDrinkSalesLog**(**msg**)**

**return** **True**

**else:**

**return** **False**

**def** resetDrinkOptions**(**self**):**

**for** element **in** self**.**drink\_profile\_elements**:**

element**.**grid\_forget**()**

self**.**displayDrinkOptionsInGUI**()**

**def** createHelpMenu**(**self**):**

"""Defines a menu that offers information about the machine."""

info\_menu **=** tk**.**Menu**(**self**.**parent\_menu**)**

self**.**parent\_menu**.**add\_cascade**(**label**=**"Help"**,**menu**=** info\_menu**)**

#Will call the method defined in the child class

info\_menu**.**add\_separator**()**

info\_menu**.**add\_command**(**label**=**""**,** command**=** self**.**secret**)**

info\_menu**.**add\_separator**()**

info\_menu**.**add\_separator**()**

info\_menu**.**add\_command**(**label**=**"How to operate"**,** command**=** self**.**showOperationInstructions**)**

info\_menu**.**add\_separator**()**

info\_menu**.**add\_command**(**label**=**"Info About Contributors"**,**command**=**self**.**showContributors**)**

info\_menu**.**add\_separator**()**

**def** secret**(**self**):**

"""Does a secret action."""

self**.**var **=** self**.**var **-** 1

**if** self**.**var **==** 0**:**

self**.**master**.**destroy**()**

self**.**main\_app**.**master**.**deiconify**()**

**def** showContributors**(**self**):**

"""Lists contributors of the project in a top level window's message box."""

top **=** tk**.**Toplevel**()**

top**.**attributes**(**'-topmost'**,**'true'**)**

top**.**title**(**"Contributors:"**)**

top**.**geometry**(**"300x230"**)**

self**.**contributors\_msg **=** tk**.**Message**(**top**)**

msg**=** """Nathan Bane:\nEmbedded Systems Design\n\nChris Blanks:\nSoftware Design

\nRyan Valente:\nMechanical Design\n\n University of Maryland Eastern Shore\n\nContact >>> Cablanks@umes.edu"""

self**.**contributors\_msg**.**config**(**text**=**msg**,**font**=** **(**"Arial"**,**12**,**""**))**

self**.**contributors\_msg**.**grid**()**

**def** showOperationInstructions**(**self**):**

"""Instructs the user on how to order from the GUI."""

file **=** open**(**self**.**operation\_instructions\_file\_path**,**'r'**)**

lines **=** file**.**readlines**()**

file**.**close**()**

msg **=** " "**.**join**(**lines**)**

top **=** tk**.**Toplevel**()**

top**.**attributes**(**'-topmost'**,**'true'**)**

top**.**title**(**"How to Operate:"**)**

top**.**geometry**(**"600x230"**)**

scroll **=** tk**.**Scrollbar**(**top**,**orient**=** tk**.**VERTICAL**)**

scroll**.**grid**(**row**=**0**,**column**=**1**,**sticky**=**"ns"**)**

canvas **=** tk**.**Canvas**(**top**,**width**=**350**,**

height**=**230**,**

scrollregion**=(**0**,**0**,**2000**,**2000**))**

canvas**.**grid**(**row**=**0**,**column**=**0**,**sticky**=**"nsew"**)**

scroll**.**config**(**command**=**canvas**.**yview**)**

canvas**.**config**(**yscrollcommand **=** scroll**.**set**)**

canvas**.**create\_text**((**0**,**0**),**text**=**msg**,**anchor**=**"nw"**)** #top left and anchored to the right

top**.**rowconfigure**(**0**,**weight**=**1**)**

top**.**columnconfigure**(**0**,**weight**=**1**)**

#### CustomerWindow.py

"""

Programmer: Chris Blanks

Last Edited: 11/3/2018

Project: Automated Self-Serving System

Purpose: This script defines the CustomerWindow Class.

"""

**import** tkinter **as** tk

**from** tkinter **import** messagebox

#my scripts

**from** AppWindow **import** AppWindow

**class** **CustomerWindow(**AppWindow**):**

operation\_instructions\_file\_path **=** """/home/pi/PYTHON\_PROJECTS/automated-self-serving-system/system\_info/instructions\_4\_customer.txt"""

**def** \_\_init\_\_**(**self**,**main\_app\_instance**):**

AppWindow**.**\_\_init\_\_**(**self**,**main\_app\_instance**)**

self**.**main\_app **=** main\_app\_instance

self**.**master **=** self**.**main\_app**.**customer\_top\_lvl

self**.**master**.**configure**(**bg**=** AppWindow**.**background\_color**)**

self**.**frame **=** tk**.**Frame**(**self**.**master**)**

self**.**frame**.**grid**()**

self**.**frame**.**configure**(**bg**=** AppWindow**.**background\_color**)**

self**.**parent\_menu **=** tk**.**Menu**(**self**.**frame**)**

self**.**master**.**config**(**menu**=** self**.**parent\_menu**)**

self**.**configureWindow**()**

self**.**displayDrinkOptionsInGUI**()** #Method from AppWindow

self**.**createHelpMenu**()**

**def** configureWindow**(**self**):**

"""Sets window geometry and limits."""

self**.**master**.**geometry**(**"{0}x{1}+0+0"**.**format**(**self**.**master**.**winfo\_screenwidth**()**

**,**self**.**master**.**winfo\_screenheight**()))**

self**.**master**.**resizable**(**width**=False,** height**=False)**

self**.**master**.**attributes**(**'-fullscreen'**,True)**

#self.master.overrideredirect(True) #no window options (e.g. resizing)

self**.**master**.**protocol**(**"WM\_DELETE\_WINDOW"**,**self**.**deployExitMessageBox**)** #ALT + F4 will close the window in Windows

**def** deployExitMessageBox**(**self**):**

**pass**

#if messagebox.askokcancel("Quit","Are you sure?"):

#

#self.master.destroy()

#self.main\_app.master.deiconify()

#### DrinkProfile.py

#!/usr/bin/env python3

"""

Programmer: Chris Blanks

Last Edited: 10/27/2018

Project: Automated Self-Serving System

Purpose: This script defines the Drink Class.

Note:

- The current code can only handle JPGs, so have to make it capable of more

file types if the user will eventually be able to pull images from google images

- Possible additions:

\*a pop up window that displays current drink profiles and related ingredients

\*a method for retrieving images from url (or really from anywhere)

"""

**import** os

**import** pathlib

**import** shutil

**class** **DrinkProfile:**

drink\_profile\_directory **=** "/home/pi/PYTHON\_PROJECTS/automated-self-serving-system/drink\_profiles"

config\_file\_path **=** "/home/pi/PYTHON\_PROJECTS/automated-self-serving-system/system\_info/config.txt"

**def** \_\_init\_\_**(**self**,**drink\_txt\_file\_path **=** **None):**

self**.**drink\_txt\_file **=** drink\_txt\_file\_path

self**.**pic\_extension **=** **None**

self**.**isNewDrink **=** **False**

#drink attributes that can be set by GUI

self**.**id\_number **=** **None**

self**.**name **=** **None**

self**.**ingredients **=** **None**

self**.**pic\_location **=** **None**

self**.**isUrl **=** **False**

self**.**isActive **=** "1"

self**.**price **=** 0.0

"""

Note on self.edited\_attributes:

Changes in the values of this attribute will mean that the new value

will replace the previous value in the text file for the drink profile.

Each index corresponds to the drink attributes declared above in descending

order, so index 0 is id\_number and index 7 is price of the drinks

"""

self**.**edited\_attributes **=** **[**0**,**0**,**0**,**0**,**0**,**0**,**0**]**

self**.**checkIfNew**()**

**def** checkIfNew**(**self**):**

"""Checks to see if the drink object is a new drink option. If new then the isNewDrink boolean will be True

until the instance's attributes are defined and the createDrinkProfile method is called on the instance.S"""

**if** self**.**drink\_txt\_file **==** **None:**

self**.**isNewDrink **=** **True**

**else:**

self**.**getDrinkProfile**()**

**def** getDrinkProfile**(**self**):**

"""Retrieves drink profile information from a subdirectory"""

file **=** open**(**self**.**drink\_txt\_file**,**'r'**,**encoding**=**"ISO-8859-1"**)**

#print("what "+self.drink\_txt\_file)

line\_count **=** 1

**for** line **in** file**:**

line **=** line**.**encode**(**'utf8'**).**decode**(**'iso-8859-1'**)**

**if** line\_count **==** 1**:**

self**.**id\_number **=** line**.**split**()[**1**]**

**if** line\_count **==** 2**:**

**print(**line**.**split**()[**1**])**

self**.**name **=** line**.**split**()[**1**].**replace**(**'\_'**,**' '**)**

**if** line\_count **==** 3**:**

ingredient\_list **=** line**.**split**()**

self**.**ingredients **=** ingredient\_list**[**1**:**len**(**ingredient\_list**)]**

**if** line\_count **==** 4**:**

self**.**pic\_location **=** line**.**split**()[**1**]**

**if** line\_count **==** 5**:**

self**.**isUrl **=** line**.**split**()[**1**]**

**if** line\_count **==** 6**:**

self**.**isActive **=** line**.**split**()[**1**]**

**if** line\_count **==** 7**:**

self**.**price **=** line**.**split**()[**1**]**

line\_count **+=** 1

**if** self**.**isUrl **==** "False"**:**

self**.**pic\_extension **=** os**.**path**.**splitext**(**self**.**pic\_location**)[**1**]**

**def** createDrinkProfile**(**self**,**desired\_pic\_path**=None):**

"""Creates a new drink profile in the designated directory.

\*Functions as a callback for a GUI element after the instance's attributes are populated.

\*Drinks are by default active until changed to inactive in GUI."""

self**.**drink\_profile\_path **=** self**.**drink\_profile\_directory **+** "/" **+** self**.**name

self**.**pic\_location **=** self**.**drink\_profile\_path **+** "/" **+** self**.**name **+** self**.**pic\_extension

pathlib**.**Path**(**self**.**drink\_profile\_path**).**mkdir**(**exist\_ok **=** **True)**

os**.**chdir**(**self**.**drink\_profile\_path**)**

new\_name **=** self**.**name **+**".txt"

new\_text\_file **=** open**(**new\_name**,**"w"**,**encoding**=**"ISO-8859-1"**)**

new\_text\_file**.**write**(**"id\_number " **+** self**.**id\_number**+**"\n"**)**

new\_text\_file**.**write**(**"name " **+** self**.**name**+**"\n"**)**

new\_text\_file**.**write**(**"ingredients " **+** self**.**ingredients**+**"\n"**)**

new\_text\_file**.**write**(**"picture\_location " **+** self**.**pic\_location**+**"\n"**)**

new\_text\_file**.**write**(**"isUrl " **+** str**(**self**.**isUrl**)+**"\n"**)**

new\_text\_file**.**write**(**"isActive " **+** self**.**isActive**+**"\n"**)**

new\_text\_file**.**write**(**"Price "**+**str**(**self**.**price**)+** "\n"**)**

new\_text\_file**.**close**()**

**if** self**.**isUrl **!=** "False"**:**

**pass** #grab pic from url

**else:**

**if** desired\_pic\_path **==** **None:**

**print(**"No pic specified"**)**

**pass** #no pic

**elif** os**.**path**.**exists**(**desired\_pic\_path**):**

shutil**.**copyfile**(**desired\_pic\_path**,**self**.**pic\_location**)**

**else:**

**print(**"Desired path does not exist."**)**

self**.**isNewDrink **=** **False**

#go back to SENIOR\_DESIGN\_CODE directory

os**.**chdir**(**".."**)**

os**.**chdir**(**".."**)**

**def** editDrinkProfile**(**self**):**

"""Edits an existing drink profile with the value change that was packed into the instance's

edited\_attributes attribute."""

attrib\_indx **=** 0

changes **=** **[]**

**for** attrib\_change **in** self**.**edited\_attributes**:**

**print(**attrib\_change**)**

**if** attrib\_change **==** 0**:**

**pass**

**else:**

changes**.**append**((**attrib\_indx **+** 1**,**attrib\_change**))** #attrib\_indx must match line number

attrib\_indx **+=**1

self**.**changeValuesInTextFile**(**changes**)**

#reset edited\_attributes

**for** i **in** range**(**len**(**self**.**edited\_attributes**)):**

self**.**edited\_attributes**[**i**]** **=** 0

**def** changeValuesInTextFile**(**self**,**changes**):**

"""Takes a tuple as input. The first parameter is the row number, and the second parameter

is the new value."""

file **=** open**(**self**.**drink\_txt\_file**,**'r+'**,**encoding**=**"ISO-8859-1"**)**

lines **=** file**.**read**().**splitlines**()**

file**.**seek**(**0**)**

line\_headers **=** **[**"id\_number "**,**"name "**,**"ingredients "**,**"picture\_location "**,** "isUrl "**,**"isActive "**,**"Price "**]**

line\_count **=** 1

**for** line **in** lines**:**

**for** i **in** range**(**len**(**changes**)):**

**if** line\_count **==** changes**[**i**][**0**]:**

line **=** line\_headers**[**line\_count **-** 1**]+**str**(**changes**[**i**][**1**])**

**print(**line**)**

**if** changes**[**i**][**0**]** **==** 4**:**

self**.**acquireDesiredPic**(**changes**[**i**][**1**])** #change picture

file**.**write**(**line**+**"\n"**)**

line\_count **+=**1

file**.**close**()**

**def** deleteDrinkProfile**(**self**):**

"""Deletes an existing drink profile """

self**.**name **=** **(**self**.**name**).**replace**(**' '**,**'\_'**)**

drink\_profile\_path **=** self**.**drink\_profile\_directory **+** "/" **+** self**.**name

pic\_location **=** drink\_profile\_path **+** "/" **+** self**.**name **+** self**.**pic\_extension

txt\_file **=** drink\_profile\_path **+** "/" **+** self**.**name **+** ".txt"

os**.**remove**(**txt\_file**)**

os**.**remove**(**pic\_location**)**

os**.**rmdir**(**drink\_profile\_path**)**

**def** addDrinkToConfig**(**self**,** path**=** config\_file\_path**):**

"""Adds a drink to the configuration file for the main application if it is new."""

f **=** open**(**path**,**"r+"**,**encoding**=**"ISO-8859-1"**)**

lines **=** f**.**read**().**splitlines**()**

f**.**seek**(**0**)**

line\_number **=** 1

**for** line **in** lines**:**

**if** line\_number **==** 2**:**

occurences\_indx **=** **[]**

start **=** 0

**while** **True:**

index\_new **=** line**.**find**(**self**.**name**,**start**)**

**if** index\_new **==** **-**1**:**

**break**

start **=** index\_new **+** len**(**self**.**name**)**

occurences\_indx**.**append**(**index\_new**)**

**if** **not** occurences\_indx**:**

line **=** line **+**" "**+** self**.**name

**else:**

isNotARepeat **=** **True**

**for** sub\_indx **in** occurences\_indx**:**

**if** line**.**endswith**(**self**.**name**)** **or** line**[** sub\_indx **+** len**(**self**.**name**)]** **==** " "**:**

isNotARepeat **=** **False**

**if** isNotARepeat**:**

line **=** line **+**" "**+** self**.**name **+** " "

f**.**write**(**line**+**"\n"**)** #overwrites existing content

line\_number **+=** 1

f**.**close**()**

**def** acquireDesiredPic**(**self**,**desired\_pic\_path**):**

"""Acquires the desired pic and sets the pic\_location attribute of the drink object."""

**if** ".jpg" **in** desired\_pic\_path**:**

self**.**pic\_extension **=** ".jpg"

**if** " " **in** self**.**name**:**

self**.**name **=** **(**self**.**name**).**replace**(**" "**,**"\_"**)**

self**.**drink\_profile\_path **=** self**.**drink\_profile\_directory **+** "/" **+** self**.**name

self**.**pic\_location **=** self**.**drink\_profile\_path **+** "/" **+** self**.**name **+** self**.**pic\_extension

**if** desired\_pic\_path **==** self**.**pic\_location**:**

**pass** #nothing to change

**else:**

shutil**.**copyfile**(**desired\_pic\_path**,**self**.**pic\_location**)**

### Functions for testing DrinkProfile class' robustness

**def** testExistingDrink**():**

"""Tests viewing the attributes of an existing drink profile."""

test\_drink **=** DrinkProfile**(**self**.**drink\_profile\_directory**+**"/cuba\_libre/cuba\_libre.txt"**)**

**print(**test\_drink**.**name**,**"\nId:"**,**test\_drink**.**id\_number**,**"\n"**,**test\_drink**.**ingredients**)**

**print(**test\_drink**.**pic\_location**,**"\n"**,**test\_drink**.**isUrl**,**"\n"**,**test\_drink**.**pic\_extension**)**

**print(**test\_drink**.**price**)**

**def** testNewDrink**():**

"""Tests creating a drink profile."""

test\_drink2 **=** DrinkProfile**()**

test\_drink2**.**name **=** "test\_drink\_2"

test\_drink2**.**id\_number **=** "24"

test\_drink2**.**ingredients **=** "stuff ingredients nothing really"

test\_drink2**.**isUrl **=** "False"

test\_drink2**.**pic\_extension **=** ".jpg"

test\_drink2**.**price **=** 5.99

test\_drink2**.**createDrinkProfile**(**"/home/pi/Pictures/drink.jpg"**)**

**def** testAddingDrinkToConfig**():**

"""Tests adding a drink name to the config file for the system."""

test\_drink3 **=** DrinkProfile**()**

test\_drink3**.**name **=** "vodka"

test\_drink3**.**id\_number **=** "25"

test\_drink3**.**ingredients **=** "stuff ingredients nothing really"

test\_drink3**.**isUrl **=** "False"

test\_drink3**.**pic\_extension **=** ".jpg"

test\_drink3**.**addDrinkToConfig**(**"config\_copy.txt"**)**

**def** testDeletingADrinkProfile**():**

"""Tests deleting a drink profile."""

test\_drink4 **=** DrinkProfile**(**self**.**drink\_profile\_directory**+**"/Test\_drink\_2/test\_drink\_2.txt"**)**

test\_drink4**.**deleteDrinkProfile**()**

**def** testEditDrinkProfile**():**

"""Tests editing a drink profile."""

test\_drink5 **=** DrinkProfile**(**self**.**drink\_profile\_directory**+**"/Test\_drink\_2/test\_drink\_2.txt"**)**

test\_drink5**.**id\_number **=** "100"

test\_drink5**.**isActive **=** "0"

test\_drink5**.**price **=** 4.05

test\_drink5**.**edited\_attributes**[**0**]** **=** test\_drink5**.**id\_number

test\_drink5**.**edited\_attributes**[**6**]** **=** test\_drink5**.**isActive

test\_drink5**.**edited\_attributes**[**7**]** **=** test\_drink5**.**price

test\_drink5**.**editDrinkProfile**()**

**print(**test\_drink5**.**edited\_attributes**)**

**if** \_\_name\_\_ **==** "\_\_main\_\_"**:**

#testExistingDrink()

#testNewDrink()

#testAddingDrinkToConfig()

#testDeletingADrinkProfile()

#testEditDrinkProfile()

**pass**

#### DrinkProfileManager.py

#!/usr/bin/env python3

"""

Programmer: Chris Blanks

Last Edited: 11/10/2018

Project: Automated Self-Serving System

Purpose: This script defines the DrinkProfileManager class

that offers editing features to the regular employee and manager.

Note:

- The amount of file IO can be reduced greatly

"""

**from** tkinter **import** messagebox

**import** tkinter **as** tk

**import** os

#my classes

**from** DrinkProfile **import** DrinkProfile

**from** EmbeddedKeyboard **import** EmbeddedKeyboard

**class** **DrinkProfileManager:**

**def** \_\_init\_\_**(**self**,**master**,**main\_app**,**admin\_mode**,**size**):**

self**.**master **=** master

self**.**main\_app **=** main\_app

self**.**admin\_mode **=** admin\_mode

self**.**width **=** size**[**0**]**

self**.**height **=** size**[**1**]**

self**.**font1 **=** **(**"Georgia"**,**12**,**""**)**

self**.**delete\_button **=** **None**

self**.**edit\_button **=** **None**

self**.**configureWindow**()**

**def** configureWindow**(**self**):**

"""Sets window geometry and exit before launching profile manager."""

self**.**master**.**configure**(**width**=**self**.**width**,**height**=**self**.**height**)**

self**.**main\_app**.**employee\_window**.**top**.**protocol**(**"WM\_DELETE\_WINDOW"**,**self**.**deployExitMessageBox**)**

self**.**createProfileManagerMainWindow**()**

**def** createProfileManagerMainWindow**(**self**):**

"""Creates the main window of the profile manager."""

self**.**drinks **=** tk**.**Listbox**(**self**.**master**,**font**=(**"Georgia"**,**16**,**"bold"**))**

self**.**drinks**.**bind**(**'<<ListboxSelect>>'**,**self**.**listboxCallback**)**

self**.**names **=** **[]**

self**.**drinks**.**insert**(**1**,**"Add A New Drink"**)**

count **=** 2

**for** drink **in** self**.**main\_app**.**all\_defined\_drinks**:**

self**.**drinks**.**insert**(**count**,**drink**.**name**.**title**())**

self**.**names**.**append**(**drink**.**name**.**replace**(**"\_"**,**" "**).**title**())**

count **+=**1

self**.**master**.**add**(**self**.**drinks**)**

self**.**drink\_selected **=** **None**

self**.**canvas **=** tk**.**Canvas**(**self**.**master**)**

drink\_name **=** tk**.**Label**(**self**.**canvas**,**text**=**"Name: "**,**font**=**self**.**font1**)**

drink\_name**.**grid**(**row**=**0**,**column**=**0**)**

self**.**name\_var **=** tk**.**StringVar**()**

cur\_drink\_name **=** tk**.**Label**(**self**.**canvas**,**textvariable**=**self**.**name\_var**,**font**=**self**.**font1**)**

cur\_drink\_name**.**grid**(**row**=**0**,**column**=**1**)**

drink\_id **=** tk**.**Label**(**self**.**canvas**,**text**=**"Drink Id: "**,**font**=**self**.**font1**)**

drink\_id**.**grid**(**row**=**1**,**column**=**0**)**

self**.**id\_var **=** tk**.**StringVar**()**

cur\_id **=** tk**.**Label**(**self**.**canvas**,**textvariable**=**self**.**id\_var**,**font**=**self**.**font1**)**

cur\_id**.**grid**(**row**=**1**,**column**=**1**)**

drink\_ingredients**=** tk**.**Label**(**self**.**canvas**,**text**=**"Ingredients: "**,**font**=**self**.**font1**)**

drink\_ingredients**.**grid**(**row**=**2**,**column**=**0**)**

self**.**ingredients\_var **=** tk**.**StringVar**()**

cur\_ingredients **=** tk**.**Label**(**self**.**canvas**,**textvariable**=**self**.**ingredients\_var**,**font**=**self**.**font1**)**

cur\_ingredients**.**grid**(**row**=**2**,**column**=**1**)**

pic\_loc **=** tk**.**Label**(**self**.**canvas**,**text**=**"Image Location: "**,**font**=**self**.**font1**)**

pic\_loc**.**grid**(**row**=**3**,**column**=**0**)**

self**.**pic\_loc\_var **=** tk**.**StringVar**()**

cur\_loc **=** tk**.**Label**(**self**.**canvas**,**textvariable**=**self**.**pic\_loc\_var**,**font**=**self**.**font1**)**

cur\_loc**.**grid**(**row**=**3**,**column**=**1**)**

price **=** tk**.**Label**(**self**.**canvas**,**text**=**"Price: "**,**font**=**self**.**font1**)**

price**.**grid**(**row**=**4**,**column**=**0**)**

self**.**price\_var **=** tk**.**StringVar**()**

cur\_price **=** tk**.**Label**(**self**.**canvas**,**textvariable**=**self**.**price\_var**,**font**=**self**.**font1**)**

cur\_price**.**grid**(**row**=**4**,**column**=**1**)**

active **=** tk**.**Label**(**self**.**canvas**,**text**=**"Active Drink: "**,**font**=**self**.**font1**)**

active**.**grid**(**row**=**5**,**column**=**0**)**

self**.**active\_var **=** tk**.**StringVar**()**

cur\_active **=** tk**.**Label**(**self**.**canvas**,**textvariable**=**self**.**active\_var**,**font**=**self**.**font1**)**

cur\_active**.**grid**(**row**=**5**,**column**=**1**)**

self**.**master**.**add**(**self**.**canvas**)**

self**.**master**.**grid**()**

**def** listboxCallback**(**self**,**event**):**

"""What's selected in the listbox is displayed in main window"""

**if** self**.**edit\_button **!=** **None:**

self**.**edit\_button**.**grid\_forget**()**

**if** self**.**delete\_button **!=** **None:**

self**.**delete\_button**.**grid\_forget**()**

self**.**drink\_selected **=** self**.**drinks**.**get**(**self**.**drinks**.**curselection**())**

index **=** self**.**drinks**.**index**(**self**.**drinks**.**curselection**())**

**if** self**.**drink\_selected **==** "Add A New Drink"**:**

self**.**createNewDrink**()**

**elif** self**.**drink\_selected **in** self**.**names**:**

drink\_to\_display **=** **None**

**for** drink **in** self**.**main\_app**.**all\_defined\_drinks**:**

**if** self**.**drink\_selected **==** drink**.**name**.**title**():**

drink\_to\_display **=** drink

self**.**name\_var**.**set**(**drink\_to\_display**.**name**.**replace**(**"\_"**,**" "**))**

self**.**id\_var**.**set**(**str**(**drink\_to\_display**.**id\_number**))**

self**.**ingredients\_var**.**set**(**drink\_to\_display**.**ingredients**)**

self**.**pic\_loc\_var**.**set**(**drink\_to\_display**.**pic\_location**)**

self**.**active\_var**.**set**(**str**(**drink\_to\_display**.**isActive**))**

self**.**price\_var**.**set**(**str**(**drink\_to\_display**.**price**))**

self**.**edit\_button **=** tk**.**Button**(**self**.**canvas**,**text**=**"Edit"**,**font**=**self**.**font1**,**bg**=**"Orange"**,**

command**=lambda** x**=**drink\_to\_display**:** self**.**launchEditor**(**x**))**

self**.**edit\_button**.**grid**(**row**=**6**,**column**=**0**)**

**if** self**.**admin\_mode**:**

self**.**delete\_button **=** tk**.**Button**(**self**.**canvas**,**text**=**"Delete"**,**font**=**self**.**font1**,**bg**=**"red"**,**

command**=lambda** x**=**drink\_to\_display**:** self**.**deployDeleteMessageBox**(**x**,**index**))**

self**.**delete\_button**.**grid**(**row**=**6**,**column**=**1**)**

**else:**

**pass**

**def** createNewDrink**(**self**):**

"""A variation on the launchEditor() that creates a new drink."""

self**.**new\_drink **=** DrinkProfile**()**

self**.**launchEditor**(None,**"Make A New Drink"**)**

self**.**main\_app**.**employee\_window**.**top**.**withdraw**()**

**def** launchEditor**(**self**,**drink**=None,**title**=None):**

"""Launches an editor with an embedded keyboard"""

**if** drink **==** **None:**

title **=** "Make A New Drink"

**else:**

title **=** drink**.**name

self**.**top **=** tk**.**Toplevel**()**

self**.**top**.**title**(**"Drink editor: " **+** title**)**

self**.**top**.**geometry**(**"{0}x{1}+0+0"**.**format**(**self**.**master**.**winfo\_screenwidth**()**

**,**self**.**master**.**winfo\_screenheight**()))**

self**.**top**.**protocol**(**"WM\_DELETE\_WINDOW"**,**self**.**deployCancelMessageBox**)**

self**.**drinkToEdit **=** drink

self**.**configureEditor**(**title**)**

**def** configureEditor**(**self**,**title**):**

"""Sets up the window for editing drink profiles."""

name\_label **=** tk**.**Label**(**self**.**top**,**text**=**"Name:"**)**

name\_label**.**grid**(**row**=**0**,**column**=**0**)**

self**.**name\_entry **=** tk**.**Entry**(**self**.**top**,**width**=**175**)**

self**.**name\_entry**.**grid**(**row**=**0**,**column**=**1**,**sticky**=**"w"**,**pady**=**4**)**

**if** title **!=** "Make A New Drink"**:**

self**.**name\_entry**.**insert**(**0**,**self**.**drinkToEdit**.**name**)**

id\_label **=** tk**.**Label**(**self**.**top**,**text**=**"Id:"**)**

id\_label**.**grid**(**row**=**1**,**column**=**0**)**

self**.**id\_entry **=** tk**.**Entry**(**self**.**top**,**width**=**175**)**

self**.**id\_entry**.**grid**(**row**=**1**,**column**=**1**,**sticky**=**"w"**,**pady**=**4**)**

**if** title **!=** "Make A New Drink"**:**

self**.**id\_entry**.**insert**(**0**,**self**.**drinkToEdit**.**id\_number**)**

ingredients\_label **=** tk**.**Label**(**self**.**top**,**text**=**"Ingredients:"**,**pady**=**4**)**

ingredients\_label**.**grid**(**row**=**2**,**column**=**0**)**

self**.**ingredient\_entry **=** tk**.**Entry**(**self**.**top**,**width**=**175**)**

self**.**ingredient\_entry**.**grid**(**row**=**2**,**column**=**1**,**sticky**=**"w"**,**pady**=**4**)**

**if** title **!=** "Make A New Drink"**:**

self**.**ingredient\_entry**.**insert**(**0**,**self**.**drinkToEdit**.**ingredients**)**

pic\_loc\_label **=** tk**.**Label**(**self**.**top**,**text**=**"Desired Picture's path:"**)**

pic\_loc\_label**.**grid**(**row**=**3**,**column**=**0**)**

self**.**pic\_loc\_entry **=** tk**.**Entry**(**self**.**top**,**width**=**175**)**

self**.**pic\_loc\_entry**.**grid**(**row**=**3**,**column**=**1**,**sticky**=**"w"**,**pady**=**4**)**

**if** title **!=** "Make A New Drink"**:**

self**.**pic\_loc\_entry**.**insert**(**0**,**self**.**drinkToEdit**.**pic\_location**)**

price\_label **=** tk**.**Label**(**self**.**top**,**text**=**"Price:"**)**

price\_label**.**grid**(**row**=**4**,**column**=**0**)**

self**.**price\_entry **=** tk**.**Entry**(**self**.**top**,**width**=**175**)**

self**.**price\_entry**.**grid**(**row**=**4**,**column**=**1**,**sticky**=**"w"**,**pady**=**4**)**

**if** title **!=** "Make A New Drink"**:**

self**.**price\_entry**.**insert**(**0**,**self**.**drinkToEdit**.**price**)**

active\_label **=** tk**.**Label**(**self**.**top**,**text**=**"Active Drink:"**)**

active\_label**.**grid**(**row**=**5**,**column**=**0**)**

self**.**active\_entry **=** tk**.**Entry**(**self**.**top**,**width**=**175**)**

self**.**active\_entry**.**grid**(**row**=**5**,**column**=**1**,**sticky**=**"w"**,**pady**=**4**)**

**if** title **!=** "Make A New Drink"**:**

self**.**active\_entry**.**insert**(**0**,**self**.**drinkToEdit**.**isActive**)**

**if** title **==** "Make A New Drink"**:**

create\_button **=** tk**.**Button**(**self**.**top**,**text**=**"Create"**,**bg**=**"green"**,**fg**=**"black"**,**

command**=**self**.**createNewProfile**)**

create\_button**.**grid**(**row**=**0**,**column**=**2**,**sticky**=**"nsew"**,**rowspan**=**2**)**

**else:**

save\_button **=** tk**.**Button**(**self**.**top**,**text**=**"Save"**,**bg**=**"green"**,**fg**=**"black"**,**

command**=**self**.**saveChanges**)**

save\_button**.**grid**(**row**=**0**,**column**=**2**,**sticky**=**"nsew"**,**rowspan**=**2**)**

entries **=** **(**self**.**name\_entry**,**self**.**id\_entry**,**self**.**ingredient\_entry**,**self**.**pic\_loc\_entry**,**

self**.**price\_entry**,**self**.**active\_entry**)**

keyboard\_canvas **=** tk**.**Canvas**(**self**.**top**,**width**=**350**,**height**=**350**)**

embed\_keyboard **=** EmbeddedKeyboard**(**keyboard\_canvas**,**entries**)**

keyboard\_canvas**.**grid**(**column**=**1**,**sticky**=**"s"**)**

**def** createNewProfile**(**self**):**

"""Creates a drink profile from the given parameters"""

self**.**new\_drink**.**name **=** self**.**name\_entry**.**get**()**

self**.**new\_drink**.**id\_number **=** self**.**id\_entry**.**get**()**

self**.**new\_drink**.**ingredients **=** self**.**ingredient\_entry**.**get**()**

pic\_path **=** self**.**pic\_loc\_entry**.**get**()**

self**.**new\_drink**.**price **=** self**.**price\_entry**.**get**()**

self**.**new\_drink**.**isActive **=** self**.**active\_entry**.**get**()**

#limited for right now

**if** ".jpg" **in** pic\_path**:**

self**.**new\_drink**.**pic\_extension **=** ".jpg"

**elif** ".png" **in** pic\_path**:**

self**.**new\_drink**.**pic\_extension **=** ".png"

**else:**

#will give it a dummy pic

pic\_path **=** "/home/pi/Pictures/drink.jpg"

self**.**new\_drink**.**pic\_extension **=** ".jpg"

self**.**new\_drink**.**isUrl **=** "False"

self**.**new\_drink**.**createDrinkProfile**(**pic\_path**)**

self**.**main\_app**.**all\_defined\_drinks**.**append**(**self**.**new\_drink**)**

self**.**main\_app**.**writeToLog**(**"Created new drink: "**+**self**.**new\_drink**.**name**)**

self**.**deploySuccesfulMessageBox**()**

**def** deleteDrink**(**self**,**drink**,**index**):**

"""Deletes drink arg and cleans it from config file."""

drink**.**deleteDrinkProfile**()**

drinkTrash **=** drink

**for** el **in** self**.**main\_app**.**drink\_objects**:**

**if** el**.**name **==** drinkTrash**.**name**:**

self**.**main\_app**.**all\_defined\_drinks**.**remove**(**drink**)**

**del** drink

self**.**drinks**.**delete**(**index**)**

self**.**main\_app**.**cleanOldDrinksFromConfig**()**

**def** saveChanges**(**self**):**

"""Checks for changed fields and puts into effect the changes."""

new\_name **=** self**.**name\_entry**.**get**()**

new\_id **=** self**.**id\_entry**.**get**()**

new\_ingredients **=** **(**self**.**ingredient\_entry**.**get**())**

**if** "," **in** new\_ingredients**:**

new\_ingredients **=** **(**new\_ingredients**.**replace**(**","**,**""**)).**split**()**

**print(**new\_ingredients**)**

new\_pic\_loc **=** self**.**pic\_loc\_entry**.**get**()** #would have to parse and update extension here

new\_price **=** self**.**price\_entry**.**get**()**

new\_active\_condition **=** self**.**active\_entry**.**get**()**

**if** new\_id **!=** "" **and** new\_id **!=** self**.**drinkToEdit**.**id\_number**:**

self**.**drinkToEdit**.**id\_number **=** new\_id

self**.**changeIdNum**()**

**if** new\_name **!=** "" **and** new\_name **!=** self**.**drinkToEdit**.**name**:**

self**.**drinkToEdit**.**name **=** new\_name

self**.**changeName**()**

**if** new\_ingredients **!=** "" **and** new\_ingredients **!=** self**.**drinkToEdit**.**ingredients**:**

self**.**drinkToEdit**.**ingredients **=** new\_ingredients

self**.**changeIngredients**()**

**if** new\_pic\_loc **!=** "" **and** new\_pic\_loc **!=** self**.**drinkToEdit**.**pic\_location**:**

self**.**changeDrinkPicLocation**(**self**.**drinkToEdit**,**new\_pic\_loc**)**

**if** new\_price **!=** "" **and** new\_price **!=** self**.**drinkToEdit**.**price**:**

self**.**drinkToEdit**.**price **=** new\_price

self**.**changePrice**()**

**if** new\_active\_condition **!=** self**.**drinkToEdit**.**isActive**:**

**if** str**(**new\_active\_condition**)** **==** "1"**:**

**print(**"active"**)**

self**.**makeActive**(**self**.**drinkToEdit**)**

**elif** str**(**new\_active\_condition**)** **==** "0"**:**

**print(**"not active"**)**

self**.**deactivateDrink**(**self**.**drinkToEdit**)**

**else:**

**print(**"Incorrect state for isActive"**)**

self**.**main\_app**.**writeToLog**(**"Edited this drink: "**+** self**.**drinkToEdit**.**name**)**

self**.**top**.**destroy**()**

**def** changeIdNum**(**self**):**

"""Changes drink id in its respective text file."""

self**.**drinkToEdit**.**edited\_attributes**[**0**]** **=** self**.**drinkToEdit**.**id\_number

self**.**drinkToEdit**.**editDrinkProfile**()**

**def** changeName**(**self**):**

"""Changes drink name in its respective text file."""

self**.**drinkToEdit**.**edited\_attributes**[**1**]** **=** self**.**drinkToEdit**.**name

self**.**drinkToEdit**.**editDrinkProfile**()**

**def** changeIngredients**(**self**):**

"""Changes drink ingredients in its respective text file."""

**if** "," **in** self**.**drinkToEdit**.**ingredients**:**

self**.**drinkToEdit**.**ingredients **=** self**.**drinkToEdit**.**ingredients**.**replace**(**","**,**" "**)**

self**.**drinkToEdit**.**edited\_attributes**[**2**]** **=** self**.**drinkToEdit**.**ingredients

self**.**drinkToEdit**.**editDrinkProfile**()**

**def** changePrice**(**self**):**

"""Changes drink name in its respective text file."""

self**.**drinkToEdit**.**edited\_attributes**[**6**]** **=** self**.**drinkToEdit**.**price

self**.**drinkToEdit**.**editDrinkProfile**()**

**def** makeActive**(**self**,**drink**):**

"""Makes inactive drinks active."""

drink**.**isActive **=** "1"

drink**.**edited\_attributes**[**5**]** **=** drink**.**isActive

drink**.**editDrinkProfile**()**

**def** deactivateDrink**(**self**,**drink**):**

"""Makes active drinks inactive."""

drink**.**isActive **=** "0"

drink**.**edited\_attributes**[**5**]** **=** drink**.**isActive

drink**.**editDrinkProfile**()**

**def** changeDrinkPicLocation**(**self**,**drink**,**new\_pic\_path**):**

"""Updates the drink path/location"""

**if** os**.**path**.**exists**(**new\_pic\_path**):**

drink**.**pic\_location **=** new\_pic\_path

drink**.**edited\_attributes**[**3**]** **=** drink**.**pic\_location

drink**.**editDrinkProfile**()**

**else:**

**print(**"Path does not exist."**)**

**def** deploySuccesfulMessageBox**(**self**):**

"""Advises the user to finish making the new drink """

**if** messagebox**.**askokcancel**(**"Successful!"**,**"Press Ok to continue..."**):**

self**.**top**.**destroy**()**

self**.**main\_app**.**employee\_window**.**master**.**deiconify**()**

**def** deployDeleteMessageBox**(**self**,**drink**,**index**):**

"""Advises the user to finish making the new drink """

**if** messagebox**.**askokcancel**(**"Delete"**,**"Are you sure that you want to delete this drink?"**):**

self**.**main\_app**.**writeToLog**(**"Deleted this drink: "**+**drink**.**name**)**

self**.**deleteDrink**(**drink**,**index**)**

**def** deployIncompleteMessageBox**(**self**):**

"""Advises the user to finish making the new drink """

**if** messagebox**.**askokcancel**(**"Incomplete"**,**"Please fill all fields."**):**

**pass**

**def** deployCancelMessageBox**(**self**):**

"""Prompts user before closing editor."""

**if** messagebox**.**askokcancel**(**"Cancel"**,**"Are you sure? All progress will be lost."**):**

self**.**main\_app**.**employee\_window**.**top**.**deiconify**()**

self**.**top**.**destroy**()**

**def** deployExitMessageBox**(**self**):**

"""Prompts user before closing window."""

**if** messagebox**.**askokcancel**(**"Quit"**,**"Are you sure?"**):**

self**.**master**.**destroy**()**

self**.**main\_app**.**employee\_window**.**master**.**deiconify**()**

#### EmbeddedKeyboard.py

#!/usr/bin/env python3

"""

Programmer: Chris Blanks

Last Edited: 11/9/2018

Project: Automated Self-Serving System

Purpose: This script defines the EmbeddedKeyboard class. This

class creates a keyboard that takes a variable amount of entry fields

and can iterate through them with the "Next Field button"

"""

**import** tkinter **as** tk

**class** **EmbeddedKeyboard:**

buttons **=** **[**

'q'**,**'w'**,**'e'**,**'r'**,**'t'**,**'y'**,**'u'**,**'i'**,**'o'**,**'p'**,**'<-'**,**'7'**,**'8'**,**'9'**,**'-'**,**

'a'**,**'s'**,**'d'**,**'f'**,**'g'**,**'h'**,**'j'**,**'k'**,**'l'**,**'['**,**']'**,**'4'**,**'5'**,**'6'**,**'+'**,**

'z'**,**'x'**,**'c'**,**'v'**,**'b'**,**'n'**,**'m'**,**','**,**'.'**,**'?'**,**'shift'**,**'1'**,**'2'**,**'3'**,**'/'**,**' Space '**]**

upper **=** **[**

'Q'**,**'W'**,**'E'**,**'R'**,**'T'**,**'Y'**,**'U'**,**'I'**,**'O'**,**'P'**,**'<-'**,**'7'**,**'8'**,**'9'**,**'-'**,**

'A'**,**'S'**,**'D'**,**'F'**,**'G'**,**'H'**,**'J'**,**'K'**,**'L'**,**'['**,**']'**,**'4'**,**'5'**,**'6'**,**'+'**,**

'Z'**,**'X'**,**'C'**,**'V'**,**'B'**,**'N'**,**'M'**,**','**,**'.'**,**'?'**,**'shift'**,**'1'**,**'2'**,**'3'**,**'/'**,**' Space '**]**

**def** \_\_init\_\_**(**self**,**master**,**target\_entry\_fields**,**background**=None):**

self**.**master **=** master

self**.**background\_arg **=** background

self**.**entries **=** **[]**

self**.**entries**.**extend**(**target\_entry\_fields**)**

self**.**num\_of\_entries **=** len**(**self**.**entries**)**

self**.**current\_entry **=** 0 #starts at first given entry field

self**.**setBindings**()**

self**.**setBackgroundColorOfKeys**()**

self**.**current\_case **=** "lower"

self**.**initializeKeyboard**()**

**def** setBackgroundColorOfKeys**(**self**):**

"""Sets background color of keys in keyboard."""

**if** self**.**background\_arg **!=** **None:**

self**.**bg **=** self**.**background\_arg

**else:**

self**.**bg **=** "#000000"

**def** setBindings**(**self**):**

"""Sets the bindings for the entry fields. Six entries max."""

entry\_id **=** 0

**for** entry **in** self**.**entries**:**

**if** entry\_id **==** 0**:**

entry**.**bind**(**"<Button-1>"**,lambda** x**=**''**:** self**.**select**(**'Next Field'**,**0**))**

**if** entry\_id **==** 1**:**

entry**.**bind**(**"<Button-1>"**,lambda** x**=**''**:** self**.**select**(**'Next Field'**,**1**))**

**if** entry\_id **==** 2**:**

entry**.**bind**(**"<Button-1>"**,lambda** x**=**''**:** self**.**select**(**'Next Field'**,**2**))**

**if** entry\_id **==** 3**:**

entry**.**bind**(**"<Button-1>"**,lambda** x**=**''**:** self**.**select**(**'Next Field'**,**3**))**

**if** entry\_id **==** 4**:**

entry**.**bind**(**"<Button-1>"**,lambda** x**=**''**:** self**.**select**(**'Next Field'**,**4**))**

**if** entry\_id **==** 5**:**

entry**.**bind**(**"<Button-1>"**,lambda** x**=**''**:** self**.**select**(**'Next Field'**,**5**))**

**if** entry\_id **==** 6**:**

entry**.**bind**(**"<Button-1>"**,lambda** x**=**''**:** self**.**select**(**'Next Field'**,**6**))**

entry\_id**+=**1

**def** initializeKeyboard**(**self**):**

"""Creates the keyboard window."""

varRow **=** 1

varColumn **=** 0

self**.**button\_ref **=** **[]**

**for** button **in** self**.**buttons**:**

command **=** **lambda** x**=**button**:** self**.**select**(**x**)**

**if** button **!=** " Space "**:**

self**.**but **=** tk**.**Button**(**self**.**master**,** text**=** button**,** width **=** 5**,** bg**=**self**.**bg**,**fg**=**"#ffffff"**,**

activebackground**=**"#ffffff"**,**activeforeground**=**"#000000"**,** relief**=**"raised"**,**

padx**=** 8**,** pady**=**8**,** bd**=**8**,**command**=**command**)**

self**.**but**.**grid**(**row**=**varRow**,**column**=**varColumn**)**

self**.**button\_ref**.**append**(**self**.**but**)**

**else:**

self**.**but **=** tk**.**Button**(**self**.**master**,** text**=** button**,** width **=** 60**,** bg**=** self**.**bg**,**fg**=**"#ffffff"**,**

activebackground**=**"#ffffff"**,**activeforeground**=**"#000000"**,** relief**=**"raised"**,**

padx**=** 4**,** pady**=**4**,** bd**=**4**,**command**=**command**)**

self**.**but**.**grid**(**row**=**6**,**columnspan**=** 16**)**

self**.**button\_ref**.**append**(**self**.**but**)**

varColumn **+=** 1

**if** varColumn **>** 14 **and** varRow **==** 1**:**

varColumn **=** 0

varRow **+=** 1

**if** varColumn **>** 14 **and** varRow **==** 2**:**

varColumn **=** 0

varRow **+=** 1

**def** setupUpper**(**self**):**

"""Sets up an uppercase keyboard."""

varRow **=** 1

varColumn **=** 0

self**.**upper\_but\_ref **=** **[]**

**for** upper **in** self**.**upper**:**

command **=** **lambda** x**=**upper**:** self**.**select**(**x**)**

**if** upper **!=** " Space "**:**

self**.**but **=** tk**.**Button**(**self**.**master**,** text**=** upper**,** width **=** 5**,** bg**=**self**.**bg**,**fg**=**"#ffffff"**,**

activebackground**=**"#ffffff"**,**activeforeground**=**"#000000"**,** relief**=**"raised"**,**

padx**=** 8**,** pady**=**8**,** bd**=**8**,**command**=**command**)**

self**.**but**.**grid**(**row**=**varRow**,**column**=**varColumn**)**

self**.**upper\_but\_ref**.**append**(**self**.**but**)**

**else:**

self**.**but **=** tk**.**Button**(**self**.**master**,** text**=** upper**,** width **=** 60**,** bg**=** self**.**bg**,**fg**=**"#ffffff"**,**

activebackground**=**"#ffffff"**,**activeforeground**=**"#000000"**,** relief**=**"raised"**,**

padx**=** 4**,** pady**=**4**,** bd**=**4**,**command**=**command**)**

self**.**but**.**grid**(**row**=**6**,**columnspan**=** 16**)**

self**.**upper\_but\_ref**.**append**(**self**.**but**)**

varColumn **+=** 1

**if** varColumn **>** 14 **and** varRow **==** 1**:**

varColumn **=** 0

varRow **+=** 1

**if** varColumn **>** 14 **and** varRow **==** 2**:**

varColumn **=** 0

varRow **+=** 1

**def** shiftKeys**(**self**):**

"""Toggles between upper and lower case."""

**if** self**.**current\_case **==** "lower"**:**

**for** but **in** self**.**button\_ref**:**

but**.**grid\_forget**()**

self**.**current\_case **=** "upper"

self**.**setupUpper**()**

**else:**

**for** but **in** self**.**upper\_but\_ref**:**

but**.**grid\_forget**()**

self**.**current\_case **=** "lower"

self**.**initializeKeyboard**()**

**def** select**(**self**,**value**,**entry\_\_id **=** 0**):**

"""Defines the action for each button in the keyboard."""

pos**=** self**.**entries**[**self**.**current\_entry**].**index**(**tk**.**INSERT**)**

self**.**entries**[**self**.**current\_entry**].**icursor**(**pos**)**

**if** value **==** "<-"**:**

**if** pos **!=** 0**:**

self**.**entries**[**self**.**current\_entry**].**delete**(**pos**-**1**)**

**elif** value **==** " Space "**:**

self**.**entries**[**self**.**current\_entry**].**insert**(**pos**,**" "**)**

**elif** value **==** "Next Field"**:**

**if** entry\_\_id **!=** self**.**current\_entry**:**

self**.**current\_entry **=** entry\_\_id

**elif** value **==** "shift"**:**

self**.**shiftKeys**()**

**else:**

self**.**entries**[**self**.**current\_entry**].**insert**(**pos**,** value**)**

#### EmployeeWindow.py

#!/usr/bin/env python3

"""

Programmer: Chris Blanks

Last Edited: 11/3/2018

Project: Automated Self-Serving System

Purpose: This script defines the EmployeeWindow Class.

"""

**import** tkinter **as** tk

**from** tkinter **import** messagebox

**import** os

#My Scripts

**from** AppWindow **import** AppWindow

**from** DrinkProfileManager **import** DrinkProfileManager

**from** ApplicationUserEditor **import** ApplicationUserEditor

**class** **EmployeeWindow(**AppWindow**):**

operation\_instructions\_file\_path **=** """/home/pi/PYTHON\_PROJECTS/automated-self-serving-system/system\_info/instructions\_4\_employee.txt"""

**def** \_\_init\_\_**(**self**,**main\_app\_instance**,** isAdminMode **=** **False):**

AppWindow**.**\_\_init\_\_**(**self**,**main\_app\_instance**)**

self**.**master **=** main\_app\_instance**.**employee\_top\_lvl

self**.**master**.**configure**(**bg**=** AppWindow**.**background\_color**)**

self**.**frame **=** tk**.**Frame**(**self**.**master**)**

self**.**frame**.**configure**(**bg**=** AppWindow**.**background\_color**)**

self**.**size**=** **[**self**.**master**.**winfo\_screenwidth**()** **,** self**.**master**.**winfo\_screenheight**()** **]**

self**.**parent\_menu **=** tk**.**Menu**(**self**.**frame**)**

self**.**master**.**config**(**menu**=** self**.**parent\_menu**)**

self**.**main\_app **=** main\_app\_instance

self**.**isAdminMode **=** isAdminMode

self**.**configureWindow**()**

self**.**frame**.**grid**()**

self**.**createHelpMenu**()**

self**.**displayDrinkOptionsInGUI**()** #Method from AppWindow

**def** configureWindow**(**self**):**

"""Sets window geometry and limits."""

self**.**master**.**geometry**(**"{0}x{1}+0+0"**.**format**(**self**.**master**.**winfo\_screenwidth**()**

**,**self**.**master**.**winfo\_screenheight**()))**

self**.**master**.**protocol**(**"WM\_DELETE\_WINDOW"**,**self**.**deployExitMessageBox**)**

**if** self**.**isAdminMode**:**

self**.**setupAdminMenuBar**()**

**else:**

self**.**setupOptionsMenuBar**()**

**def** setupAdminMenuBar**(**self**):**

"""Provides extra features in the menu bar for full control of the app."""

**print(**"Admin status."**)**

self**.**admin\_menu **=** tk**.**Menu**(**self**.**parent\_menu**,**tearoff**=**0**)**

self**.**parent\_menu**.**add\_cascade**(**label**=**"Admin Options"**,**menu**=**self**.**admin\_menu**)**

self**.**admin\_menu**.**add\_command**(**label**=**"Launch Drink Profile Manager"**,**command**=** self**.**launchDrinkProfileManager**)**

self**.**admin\_menu**.**add\_separator**()**

self**.**admin\_menu**.**add\_command**(**label**=**"Display Log" **,**command**=** self**.**displayLogFile**)** #allow option to delete them

self**.**admin\_menu**.**add\_separator**()**

self**.**admin\_menu**.**add\_command**(**label**=**"Edit Configuration" **,**command**=** self**.**editConfigFile**)**

self**.**admin\_menu**.**add\_separator**()**

self**.**admin\_menu**.**add\_command**(**label**=**"Edit User Logins" **,**command**=** self**.**editUserLogins**)**

self**.**admin\_menu**.**add\_separator**()**

self**.**admin\_menu**.**add\_command**(**label**=**"Show IP Address" **,**command**=** self**.**showIPAddress**)**

**def** setupOptionsMenuBar**(**self**):**

"""Provides regular features in the menu bar for employees"""

**print(**"Employee status."**)**

self**.**options\_menu **=** tk**.**Menu**(**self**.**parent\_menu**,**tearoff**=**0**)**

self**.**parent\_menu**.**add\_cascade**(**label**=**"Employee Options"**,**menu**=** self**.**options\_menu**)**

self**.**options\_menu**.**add\_command**(**label**=**"Launch Drink Profile Manager" **,**command**=** self**.**launchDrinkProfileManager**)**

**def** launchDrinkProfileManager**(**self**):**

"""Allows the employee to add, edit, or delete drink profiles."""

self**.**top **=** tk**.**Toplevel**(**self**.**master**)**

self**.**top**.**title**(**"Drink Profile Manager"**)**

self**.**top**.**geometry**(**"{0}x{1}+0+0"**.**format**(**self**.**master**.**winfo\_screenwidth**()**

**,**self**.**master**.**winfo\_screenheight**()))**

profile\_manager\_win **=** tk**.**PanedWindow**(**self**.**top**,**orient**=** tk**.**HORIZONTAL**)**

profile\_manager\_win**.**pack**(**fill**=**tk**.**BOTH**,**expand**=**1**)**

self**.**drink\_profile\_manager **=** DrinkProfileManager**(**profile\_manager\_win

**,**self**.**main\_app**,**self**.**isAdminMode**,**self**.**size**)**

self**.**master**.**withdraw**()**

**def** displayLogFile**(**self**):**

"""Displays the most recent log file."""

file **=** open**(**self**.**main\_app**.**todays\_log**,**'r'**)**

lines **=** file**.**readlines**()**

file**.**close**()**

msg **=** " "**.**join**(**lines**)**

self**.**log\_display **=** tk**.**Toplevel**()**

self**.**log\_display**.**title**(**"Current Log File:"**)**

#top.geometry("350x230")

scroll **=** tk**.**Scrollbar**(**self**.**log\_display**,**orient**=** tk**.**VERTICAL**)**

scroll**.**grid**(**row**=**0**,**column**=**1**,**sticky**=**"ns"**)**

canvas **=** tk**.**Canvas**(**self**.**log\_display**,**width**=**600**,**

height**=**500**,**

scrollregion**=(**0**,**0**,**2000**,**2000**))**

canvas**.**grid**(**row**=**0**,**column**=**0**,**sticky**=**"nsew"**)**

scroll**.**config**(**command**=**canvas**.**yview**)**

canvas**.**config**(**yscrollcommand **=** scroll**.**set**)**

canvas**.**create\_text**((**0**,**0**),**text**=**msg**,**anchor**=**"nw"**)** #top left and anchored to the right

log\_menu **=** tk**.**Menu**(**self**.**log\_display**)**

log\_options **=** tk**.**Menu**(**log\_menu**,**tearoff**=**0**)**

log\_menu**.**add\_cascade**(**label**=**"Options"**,**menu**=**log\_options**)**

self**.**log\_display**.**config**(**menu**=**log\_menu**)**

log\_options**.**add\_command**(**label**=**"Clear Log"

**,**command**=** self**.**clearTodayLog**)**

self**.**log\_display**.**rowconfigure**(**0**,**weight**=**1**)**

self**.**log\_display**.**columnconfigure**(**0**,**weight**=**1**)**

**def** clearTodayLog**(**self**):**

"""Clears the current log."""

os**.**remove**(**self**.**main\_app**.**todays\_log**)**

self**.**deployClearedMessageBox**()**

self**.**main\_app**.**writeToLog**(**"Cleaned log file."**)**

**def** deployClearedMessageBox**(**self**):**

"""Deploys the message box for when the log file is cleared."""

**if** messagebox**.**askokcancel**(**"Cleared today's log"**,**"press Ok to continue."**):**

self**.**log\_display**.**destroy**()**

**def** editUserLogins**(**self**):**

"""Displays current registered users. Allows for adding or deleting users."""

user\_editor\_top **=** tk**.**Toplevel**()**

user\_editor\_top**.**title**(**"Application User Editor"**)**

#user\_editor\_top.geometry("{0}x{1}+0+0".format(self.size[0],self.size[1]))

self**.**user\_editor **=** ApplicationUserEditor**(**user\_editor\_top**,**self**.**main\_app**,**self**.**size**)**

**def** editConfigFile**(**self**):**

"""Allows editing of the contents in the configuration file. """

**pass** #Will be defined at a later time

**def** showIPAddress**(**self**):**

"""Shows the current IP address in a top level window. """

ip\_window **=** tk**.**Toplevel**()**

ip\_window**.**title**(**"IP Address"**)**

ip\_address **=** str**(**self**.**main\_app**.**ip\_address**)**

ip\_label **=** tk**.**Label**(**ip\_window**,**text**=**ip\_address**,**font**=(**"Georgia"**,**"20"**,**"bold"**),**fg**=**"red"**)**

ip\_label**.**grid**()**

**def** deployExitMessageBox**(**self**):**

"""Destroys employee window and brings up root window."""

**if** messagebox**.**askokcancel**(**"Quit"**,**"Are you sure?"**):**

self**.**isAdminMode **=** **False**

self**.**master**.**destroy**()**

self**.**main\_app**.**master**.**deiconify**()**

#### LoginWindow.py

#!/usr/bin/env python3

"""

Programmer: Chris Blanks

Last Edited: 11/3/2018

Project: Automated Self-Serving System

Purpose: This script defines the LoginWindow Class.

"""

**import** tkinter **as** tk

**from** tkinter **import** messagebox

**import** os

**from** PIL **import** Image**,** ImageTk

**from** EmbeddedKeyboard **import** EmbeddedKeyboard

**class** **LoginWindow:**

#class member variables

login\_file\_path **=** "/home/pi/PYTHON\_PROJECTS/automated-self-serving-system/system\_info/user\_login.txt"

failed\_attempt\_limit **=** 3

logo\_path **=** "/home/pi/PYTHON\_PROJECTS/automated-self-serving-system/gui\_images/Senior\_Design\_Logo.jpg"

background\_color **=** "LightCyan3"

**def** \_\_init\_\_**(**self**,** main\_app\_instance**):**

self**.**main\_app **=** main\_app\_instance

self**.**main\_app**.**master**.**withdraw**()**

self**.**master **=** self**.**main\_app**.**login\_top\_lvl

self**.**master**.**configure**(**bg**=** self**.**background\_color**)**

self**.**frame **=** tk**.**Frame**(**self**.**master**,**bg**=**self**.**background\_color**)**

self**.**frame**.**grid**()**

self**.**configureWindow**()**

self**.**attempt\_number **=** 1

self**.**isAdminAccount **=** **False**

self**.**isRegularUser **=** **False**

**def** configureWindow**(**self**):**

"""Creates the login window."""

self**.**master**.**protocol**(**"WM\_DELETE\_WINDOW"**,**self**.**deployExitMessageBox**)**

self**.**master**.**geometry**(**"1024x600"**)**

#screen\_height = self.master.winfo\_screenheight()

#screen\_width = self.master.winfo\_screenwidth()

#self.master.geometry("{0}x{1}+0+0".format(screen\_width,screen\_height))

self**.**login\_title**=** tk**.**Label**(**self**.**frame**,**text**=**"Employee Login"**)**

self**.**username\_label **=** tk**.**Label**(**self**.**frame**,**text**=**"Username:"**,**font**=(**"Times"**,**"12"**,**"bold italic"**),**

bg**=**self**.**background\_color**)**

self**.**username\_label**.**grid**(**row**=**1**,**column**=**1**,**sticky**=**"e"**)**

self**.**username\_entry **=** tk**.**Entry**(**self**.**frame**,**width**=**138**)**

self**.**username\_entry**.**grid**(**row**=**1**,**column**=**2**)**

self**.**password\_label **=** tk**.**Label**(**self**.**frame**,**text**=**"Password:"**,**font**=(**"Times"**,**"12"**,**"bold italic"**),**

bg**=**self**.**background\_color**)**

self**.**password\_label**.**grid**(**row**=**2**,**column**=**1**,**sticky**=**"e"**)**

self**.**password\_entry **=** tk**.**Entry**(**self**.**frame**,**width**=**138**)**

self**.**password\_entry**.**grid**(**row**=**2**,**column**=**2**)**

self**.**login\_btn **=** tk**.**Button**(**self**.**frame**,**text**=**"Login"**,**bg**=**"green"

**,**fg**=**"white"**,**command**=** self**.**testUserLogin**)**

self**.**login\_btn**.**grid**(**row**=**1**,**column**=** 3**,**rowspan**=**2**,**columnspan**=**2**,**padx**=**3**,**sticky**=**"nsew"**)**

entries **=** **[**self**.**username\_entry**,**self**.**password\_entry**]**

self**.**canvas **=** tk**.**Canvas**(**self**.**frame**,**width**=**350**,**height**=**350**,**bg**=**self**.**background\_color**)**

self**.**embed\_keyboard **=** EmbeddedKeyboard**(**self**.**canvas**,**entries**)**

self**.**canvas**.**grid**(**row**=**4**,**column**=**2**,**sticky**=**"s"**)**

self**.**master**.**grid\_rowconfigure**(**4**,**weight**=**0**)**

**def** testUserLogin**(**self**):**

"""Tests whether the user's login matches. Sends"""

self**.**searchForLoginMatch**()**

**if** self**.**main\_app**.**isValidLogin**:**

self**.**main\_app**.**isEmployeeMode **=** **True**

**if** self**.**isRegularUser **==** **False:**

self**.**isAdminAccount **=** **True**

self**.**main\_app**.**isValidLogin **=** **False** #value rest

self**.**master**.**destroy**()**

self**.**main\_app**.**createEmployeeWindow**(**self**.**isAdminAccount**)**

**else:**

**print(**"Invalid credentials! "**,**self**.**failed\_attempt\_limit**-**self**.**attempt\_number**,**

" attempt(s) left"**)**

self**.**attempt\_number **+=**1

**if** self**.**attempt\_number **>** 3**:**

self**.**main\_app**.**isEmployeeMode **=** **False**

self**.**main\_app**.**writeToLog**(**"Failed login"**)**

self**.**master**.**destroy**()**

self**.**main\_app**.**createCustomerWindow**()**

self**.**main\_app**.**master**.**deiconify**()**

**def** searchForLoginMatch**(**self**):**

"""Searches for a matching user login."""

username **=** self**.**username\_entry**.**get**()**

password **=** self**.**password\_entry**.**get**()**

file **=** open**(**self**.**login\_file\_path**,**'rb+'**)**

lines **=** file**.**read**().**splitlines**()**

**for** line **in** lines**:**

line**=** str**((**self**.**main\_app**.**cipher\_suite**.**decrypt**(**line**)).**decode**(**'UTF-8'**))**

**if** "REGULAR" **in** line**:**

self**.**isRegularUser **=** **True**

**if** **(**line**.**split**()[**0**]).**lower**()** **==** username**.**lower**()** **and** **(**line**.**split**()[**1**]).**lower**()** **==** password**.**lower**():**

self**.**main\_app**.**writeToLog**(**username **+** " logged in"**)**

self**.**main\_app**.**isValidLogin **=** **True**

**break**

file**.**close**()**

**def** deployExitMessageBox**(**self**):**

**if** messagebox**.**askokcancel**(**"Quit"**,**"Are you sure?"**):**

self**.**master**.**destroy**()**

self**.**main\_app**.**master**.**deiconify**()**

#### PeripheralDevice.py

#!/usr/bin/env python3

"""

Programmer: Chris Blanks

Last Edited: 10/24/2018

Project: Automated Self-Serving System

Purpose: This script defines the Peripheral Device class.

"""

**class** **PeripheralDevice:**

**def** \_\_init\_\_**(**self**,** main\_app\_instance**):**

self**.**main\_app **=** main\_app\_instance

self**.**name\_of\_device **=** **None**

self**.**state **=** **None**

self**.**buffer **=** **None**

self**.**communication\_method **=** **None**

self**.**pin\_number **=** **None**

**def** reportStateToMainApp**(**self**):**

**pass**

**def** sendDataToMainApp**(**self**):**

**pass**

**def** startCommunication**(**self**):**

**pass**

**def** terminateCommunication**(**self**):**

**pass**

1. Source Code of the Phone Application

#### MainActivity.java

package com**.**example**.**cabla**.**drinkmachinephoneapp**;**

**import** android**.**content**.**Context**;**

**import** android**.**content**.**SharedPreferences**;**

**import** android**.**preference**.**PreferenceManager**;**

**import** android**.**support**.**v7**.**app**.**AppCompatActivity**;**

**import** android**.**os**.**Bundle**;**

**import** android**.**util**.**Log**;**

**import** android**.**widget**.**EditText**;**

**import** android**.**view**.**View**;**

**import** android**.**widget**.**Button**;**

**import** android**.**widget**.**Toast**;**

**import** android**.**content**.**Intent**;**

**import** java**.**util**.**ArrayList**;**

public class MainActivity **extends** AppCompatActivity **{**

Button login\_but**;**

EditText username**;**

EditText password**;**

int attempts **=** 3**;**

public static final String MYPREFERENCES **=** "prefs"**;**

@Override

protected void onCreate**(**Bundle savedInstanceState**)** **{**

**super.**onCreate**(**savedInstanceState**);**

setContentView**(**R**.**layout**.**activity\_main**);**

//SharedPreferences prefs = getApplicationContext().getSharedPreferences(MYPREFERENCES, Context.MODE\_PRIVATE);

//SharedPreferences.Editor editor = prefs.edit();

//editor.putString("url","http://10.0.0.35:8000/shared\_data\_2018-11-22.txt");

//editor.apply();

LoginButton**();**

**}**

public void LoginButton**(){**

//Widgets defined in XML file

login\_but **=** findViewById**(**R**.**id**.**button1**);**

username **=** findViewById**(**R**.**id**.**user\_entry**);**

password **=** findViewById**(**R**.**id**.**pass\_entry**);**

login\_but**.**setOnClickListener**(new** View**.**OnClickListener**(){**

@Override

public void onClick**(**View v**){**

**if(**username**.**getText**().**toString**().**equals**(**"admin"**)**

**&&** password**.**getText**().**toString**().**equals**(**"admin"**)){**

//correct password

Toast**.**makeText**(**MainActivity**.this,**"Username and Password is correct"**,**

Toast**.**LENGTH\_SHORT**).**show**();**

Intent next\_screen **=** **new** Intent**(**"com.example.cabla.drinkmachinephoneapp.User"**);**

startActivity**(**next\_screen**);** //goes to tabbed screen

**}else{**

//incorrect password

Toast**.**makeText**(**MainActivity**.this,**"Username and Password is not correct"**,**

Toast**.**LENGTH\_SHORT**).**show**();**

**--**attempts**;**

**if(**attempts**==**0**){**

login\_but**.**setEnabled**(false);**

Toast**.**makeText**(**MainActivity**.this,**"Failure limit reached. Close app."**,**

Toast**.**LENGTH\_SHORT**).**show**();**

**}**

**}**

**}**

**}**

**);**

**}**

**}**

#### User.java

package com**.**example**.**cabla**.**drinkmachinephoneapp**;**

**import** android**.**content**.**Context**;**

**import** android**.**content**.**Intent**;**

**import** android**.**content**.**SharedPreferences**;**

**import** android**.**preference**.**PreferenceManager**;**

**import** android**.**support**.**design**.**widget**.**TabLayout**;**

**import** android**.**support**.**v7**.**app**.**AppCompatActivity**;**

**import** android**.**support**.**v7**.**widget**.**Toolbar**;**

**import** android**.**support**.**v4**.**app**.**Fragment**;**

**import** android**.**support**.**v4**.**app**.**FragmentManager**;**

**import** android**.**support**.**v4**.**app**.**FragmentPagerAdapter**;**

**import** android**.**support**.**v4**.**view**.**ViewPager**;**

**import** android**.**os**.**Bundle**;**

**import** android**.**util**.**Log**;**

**import** android**.**view**.**LayoutInflater**;**

**import** android**.**view**.**Menu**;**

**import** android**.**view**.**MenuItem**;**

**import** android**.**view**.**View**;**

**import** android**.**view**.**ViewGroup**;**

**import** android**.**widget**.**Toast**;**

**import** java**.**util**.**ArrayList**;**

public class User **extends** AppCompatActivity **{**

ArrayList**<**String**>** status\_package**;**

ArrayList**<**String**>** inventory\_package**;**

ArrayList**<**String**>** sales\_package**;**

ArrayList**<**String**>** dates**;**

String url **=** ""**;**

boolean noData **=** **true;**

//String data\_package;

private SectionsPagerAdapter mSectionsPagerAdapter**;**

// The {@link ViewPager} that will host the section contents.

private ViewPager mViewPager**;**

public static final String MYPREFERENCES **=** "prefs"**;**

SharedPreferences prefs**;**

@Override

protected void onCreate**(**Bundle savedInstanceState**)** **{**

**super.**onCreate**(**savedInstanceState**);**

setContentView**(**R**.**layout**.**activity\_user**);**

prefs **=** getApplicationContext**().**getSharedPreferences**(**MYPREFERENCES**,** Context**.**MODE\_PRIVATE**);**

/\*attemptDataTransfer();

if(noData){

Toast.makeText(this,"Could not retrieve data from: "+url,

Toast.LENGTH\_LONG).show(); //LENGTH\_LONG = 3.5 sec

}

\*/

Toolbar toolbar **=** **(**Toolbar**)** findViewById**(**R**.**id**.**toolbar**);**

setSupportActionBar**(**toolbar**);**

mSectionsPagerAdapter **=** **new** SectionsPagerAdapter**(**getSupportFragmentManager**());**

// Set up the ViewPager with the sections adapter.

mViewPager **=** **(**ViewPager**)** findViewById**(**R**.**id**.**container**);**

mViewPager**.**setAdapter**(**mSectionsPagerAdapter**);**

TabLayout tabLayout **=** **(**TabLayout**)** findViewById**(**R**.**id**.**tabs**);**

mViewPager**.**addOnPageChangeListener**(new** TabLayout**.**TabLayoutOnPageChangeListener**(**tabLayout**));**

tabLayout**.**addOnTabSelectedListener**(new** TabLayout**.**ViewPagerOnTabSelectedListener**(**mViewPager**));**

**}**

public void attemptDataTransfer**(){**

url **=** prefs**.**getString**(**"url"**,**"http://10.0.0.35:8000/shared\_data\_2018-11-22.txt"**);**

Log**.**i**(**"CHRIS\_TEST\_pref"**,**url**);**

DataRequester request **=** **new** DataRequester**(**url**);**

**try{**

Log**.**i**(**"CHRIS"**,**"TEST2"**);**

Thread t **=** **new** Thread**(**request**);**

t**.**start**();**

t**.**join**();**

**if(**request**.**getError**()){**

noData **=** **true;**

**}else** **{**

sales\_package **=** request**.**getSalesData**();**

inventory\_package **=** request**.**getInventoryData**();**

status\_package **=** request**.**getStatusData**();**

dates **=** request**.**getDates**();**

noData **=** **false;**

**}**

**}**

**catch** **(**InterruptedException i**){**

Log**.**i**(**"CHRIS"**,**"TEST3"**);**

Log**.**d**(**"ERROR"**,**"Thread issue"**);**

**}**

**}**

//This method is for attaching features to the menu icon

@Override

public boolean onCreateOptionsMenu**(**Menu menu**)** **{**

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater**().**inflate**(**R**.**menu**.**menu\_user**,** menu**);**

**return** **true;**

**}**

@Override

public boolean onOptionsItemSelected**(**MenuItem item**)** **{**

// Handle action bar item clicks here. The action bar will

// automatically handle clicks on the Home/Up button, so long

// as you specify a parent activity in AndroidManifest.xml.

int id **=** item**.**getItemId**();**

//noinspection SimplifiableIfStatement

**if** **(**id **==** R**.**id**.**ip\_config**)** **{**

Intent ip\_config\_screen **=** **new** Intent**(**"com.example.cabla.drinkmachinephoneapp.IP\_Config"**);**

startActivity**(**ip\_config\_screen**);**

**return** **true;**

**}**

**if(**id **==** R**.**id**.**request\_data**){**

Toast**.**makeText**(this,**"Attempting to update data"**,**

Toast**.**LENGTH\_SHORT**).**show**();**

attemptDataTransfer**();**

**if(**noData**){**

Toast**.**makeText**(this,**"Could not retrieve data from: "**+**url**,**

Toast**.**LENGTH\_LONG**).**show**();**

**}**

**return** **true;**

**}**

**return** **super.**onOptionsItemSelected**(**item**);**

**}**

/\*\*

\* A placeholder fragment containing a simple view.

\*/

public static class PlaceholderFragment **extends** Fragment **{**

/\*\*

\* The fragment argument representing the section number for this

\* fragment.

\*/

private static final String ARG\_SECTION\_NUMBER **=** "section\_number"**;**

public PlaceholderFragment**()** **{**

**}**

/\*\*

\* Returns a new instance of this fragment for the given section

\* number.

\*/

public static PlaceholderFragment newInstance**(**int sectionNumber**)** **{**

PlaceholderFragment fragment **=** **new** PlaceholderFragment**();**

Bundle args **=** **new** Bundle**();**

args**.**putInt**(**ARG\_SECTION\_NUMBER**,** sectionNumber**);**

fragment**.**setArguments**(**args**);**

**return** fragment**;**

**}**

@Override

public View onCreateView**(**LayoutInflater inflater**,** ViewGroup container**,**

Bundle savedInstanceState**)** **{**

View rootView **=** inflater**.**inflate**(**R**.**layout**.**fragment\_user**,** container**,** **false);**

//TextView textView = (TextView) rootView.findViewById(R.id.section\_label);

//textView.setText(getString(R.string.section\_format, getArguments().getInt(ARG\_SECTION\_NUMBER)));

**return** rootView**;**

**}**

**}**

/\*\*

\* A {@link FragmentPagerAdapter} that returns a fragment corresponding to

\* one of the sections/tabs/pages.

\*/

public class SectionsPagerAdapter **extends** FragmentPagerAdapter **{**

public SectionsPagerAdapter**(**FragmentManager fm**)** **{**

**super(**fm**);**

**}**

@Override // getItem is called to instantiate the fragment for the given page.

public Fragment getItem**(**int position**)** **{**

Fragment cur\_fragment **=** **null;**

**switch(**position**){**

**case** 0**:**

Bundle bundle\_0 **=** **new** Bundle**();**

bundle\_0**.**putStringArrayList**(**"inventory"**,**inventory\_package**);**

bundle\_0**.**putBoolean**(**"no\_data"**,**noData**);**

cur\_fragment **=** **new** InventoryFragment**()** **;**

cur\_fragment**.**setArguments**(**bundle\_0**);**

**break;**

**case** 1**:**

Bundle bundle\_1 **=** **new** Bundle**();**

bundle\_1**.**putStringArrayList**(**"sales"**,**sales\_package**);**

bundle\_1**.**putStringArrayList**(**"dates"**,**dates**);**

bundle\_1**.**putBoolean**(**"no\_data"**,**noData**);**

cur\_fragment **=** **new** SalesFragment**()** **;**

cur\_fragment**.**setArguments**(**bundle\_1**);**

**break;**

**case** 2**:**

Bundle bundle\_2 **=** **new** Bundle**();**

bundle\_2**.**putStringArrayList**(**"status"**,**status\_package**);**

bundle\_2**.**putBoolean**(**"no\_data"**,**noData**);**

cur\_fragment **=** **new** StatusFragment**();**

cur\_fragment**.**setArguments**(**bundle\_2**);**

**break;**

**default:**

//no match

**}**

**return** cur\_fragment**;**

**}**

@Override

public int getCount**()** **{**

// Show 3 total pages.

**return** 3**;**

**}**

**}**

**}**

#### InventoryFragment.java

package com**.**example**.**cabla**.**drinkmachinephoneapp**;**

**import** android**.**app**.**Notification**;**

**import** android**.**app**.**NotificationChannel**;**

**import** android**.**app**.**NotificationManager**;**

**import** android**.**content**.**Context**;**

**import** android**.**os**.**Bundle**;**

**import** android**.**support**.**v4**.**app**.**Fragment**;**

**import** android**.**support**.**v4**.**app**.**NotificationCompat**;**

**import** android**.**util**.**Log**;**

**import** android**.**view**.**Gravity**;**

**import** android**.**view**.**LayoutInflater**;**

**import** android**.**view**.**View**;**

**import** android**.**view**.**ViewGroup**;**

**import** android**.**widget**.**TableLayout**;**

**import** android**.**widget**.**TableRow**;**

**import** android**.**widget**.**TextView**;**

**import** java**.**util**.**ArrayList**;**

/\*\*

\* A simple {@link Fragment} subclass.

\*/

public class InventoryFragment **extends** Fragment **{**

boolean isLowLevel **=** **false;**

boolean hasAlerted **=** **false;**

public InventoryFragment**()** **{**

**}**

@Override

public View onCreateView**(**LayoutInflater inflater**,** ViewGroup container**,**

Bundle savedInstanceState**)** **{**

ArrayList**<**String**>** inventory\_info **=** getArguments**().**getStringArrayList**(**"inventory"**);**

boolean no\_status\_data **=** getArguments**().**getBoolean**(**"no\_data"**);**

Log**.**i**(**"CHRISTEST"**,**String**.**valueOf**(**no\_status\_data**));**

final View table\_view **=** inflater**.**inflate**(**R**.**layout**.**fragment\_inventory**,** container**,** **false);**

TableLayout table **=** table\_view**.**findViewById**(**R**.**id**.**inventory\_table**);**

ArrayList**<**ArrayList**<**String**>>** display\_data **=** **new** ArrayList**<**ArrayList**<**String**>>();**

**if** **(**no\_status\_data**)** **{**

display\_data **=** packDummyData**();**

**}** **else** **{**

display\_data **=** packRealData**(**inventory\_info**);**

**}**

int view\_index **=** 1**;**

int row\_index **=** 1**;**

int num\_of\_text\_views\_per\_row **=** 3**;**

int num\_of\_text\_views **=** display\_data**.**size**()** **\*** num\_of\_text\_views\_per\_row**;**

TextView**[]** data\_text\_views **=** **new** TextView**[**num\_of\_text\_views**];**

TableRow**[]** row\_inserts **=** **new** TableRow**[**display\_data**.**size**()];**

**for** **(**ArrayList**<**String**>** o **:** display\_data**)** **{**

row\_inserts**[**row\_index **-** 1**]** **=** **new** TableRow**(**getContext**());**

**for** **(**String str **:** o**)** **{**

**if** **(**view\_index **>** num\_of\_text\_views**)** **{**

**break;**

**}**

data\_text\_views**[**view\_index **-** 1**]** **=** **new** TextView**(**getContext**());**

data\_text\_views**[**view\_index **-** 1**].**setGravity**(**Gravity**.**CENTER**);**

data\_text\_views**[**view\_index **-** 1**].**setText**(**str**);**

data\_text\_views**[**view\_index **-** 1**].**setId**(**view\_index**);**

data\_text\_views**[**view\_index **-** 1**].**setLayoutParams**(new** TableRow**.**LayoutParams**(**TableRow**.**LayoutParams**.**FILL\_PARENT**,**

TableRow**.**LayoutParams**.**WRAP\_CONTENT**));**

row\_inserts**[**row\_index **-** 1**].**addView**(**data\_text\_views**[**view\_index **-** 1**]);**

view\_index**++;**

**}**

table**.**addView**(**row\_inserts**[**row\_index **-** 1**]);**

row\_index**++;**

**}**

**if** **(**isLowLevel **&&** **!**hasAlerted**)** **{**

notificationForLowLevel**();**

hasAlerted **=** **true;**

**}**

**return** table\_view**;**

**}**

public void notificationForLowLevel**()** **{**

NotificationCompat**.**Builder mBuilder **=** **new** NotificationCompat**.**Builder**(**getContext**(),** "default"**);**

NotificationManager mNotificationManager **=**

**(**NotificationManager**)** getContext**().**getSystemService**(**Context**.**NOTIFICATION\_SERVICE**);**

**if** **(**android**.**os**.**Build**.**VERSION**.**SDK\_INT **>=** android**.**os**.**Build**.**VERSION\_CODES**.**O**)** **{**

NotificationChannel channel **=** **new** NotificationChannel**(**"default"**,**

"YOUR\_CHANNEL\_NAME"**,**

NotificationManager**.**IMPORTANCE\_DEFAULT**);**

channel**.**setDescription**(**"YOUR\_NOTIFICATION\_CHANNEL\_DISCRIPTION"**);**

mNotificationManager**.**createNotificationChannel**(**channel**);**

**}**

mBuilder**.**setAutoCancel**(true)**

**.**setDefaults**(**Notification**.**DEFAULT\_ALL**)**

**.**setWhen**(**System**.**currentTimeMillis**())**

**.**setSmallIcon**(**R**.**drawable**.**ic\_launcher\_foreground**)**

**.**setTicker**(**"Hearty365"**)**

**.**setPriority**(**Notification**.**PRIORITY\_MAX**)** // this is deprecated in API 26 but you can still use for below 26. check below update for 26 API

**.**setContentTitle**(**"High Priority"**)**

**.**setContentText**(**"Running low on inventory"**)**

**.**setContentInfo**(**"Info"**);**

NotificationManager notificationManager **=** **(**NotificationManager**)** getContext**().**getSystemService**(**Context**.**NOTIFICATION\_SERVICE**);**

notificationManager**.**notify**(**0**,** mBuilder**.**build**());**

**}**

public ArrayList**<**ArrayList**<**String**>>** packDummyData**()** **{**

ArrayList**<**ArrayList**<**String**>>** data\_array **=** **new** ArrayList**<**ArrayList**<**String**>>();**

ArrayList**<**String**>** data **=** **new** ArrayList**<>();**

data**.**add**(**"light rum"**);**

data**.**add**(**"700 ml"**);**

data**.**add**(**"1000 ml"**);**

data\_array**.**add**(**data**);**

ArrayList**<**String**>** data2 **=** **new** ArrayList**<>();**

data2**.**add**(**"vodka"**);**

data2**.**add**(**"400 ml"**);**

data2**.**add**(**"1000 ml"**);**

data\_array**.**add**(**data2**);**

ArrayList**<**String**>** data3 **=** **new** ArrayList**<>();**

data3**.**add**(**"Tequila"**);**

data3**.**add**(**"600 ml"**);**

data3**.**add**(**"1000 ml"**);**

data\_array**.**add**(**data3**);**

ArrayList**<**String**>** data4 **=** **new** ArrayList**<>();**

data4**.**add**(**"Simple Syrup"**);**

data4**.**add**(**"800 ml"**);**

data4**.**add**(**"1000 ml"**);**

data\_array**.**add**(**data4**);**

ArrayList**<**String**>** data5 **=** **new** ArrayList**<>();**

data5**.**add**(**"Soda Water"**);**

data5**.**add**(**"1500 ml"**);**

data5**.**add**(**"2000 ml"**);**

data\_array**.**add**(**data5**);**

ArrayList**<**String**>** data6 **=** **new** ArrayList**<>();**

data6**.**add**(**"Triple Sec"**);**

data6**.**add**(**"300 ml"**);**

data6**.**add**(**"500 ml"**);**

data\_array**.**add**(**data6**);**

ArrayList**<**String**>** data7 **=** **new** ArrayList**<>();**

data7**.**add**(**"Lemon Juice"**);**

data7**.**add**(**"200 ml"**);**

data7**.**add**(**"300 ml"**);**

data\_array**.**add**(**data7**);**

ArrayList**<**String**>** data8 **=** **new** ArrayList**<>();**

data8**.**add**(**"Orange Juice"**);**

data8**.**add**(**"500 ml"**);**

data8**.**add**(**"800 ml"**);**

data\_array**.**add**(**data8**);**

ArrayList**<**String**>** data9 **=** **new** ArrayList**<>();**

data9**.**add**(**"Cranberry Juice"**);**

data9**.**add**(**"800 ml"**);**

data9**.**add**(**"1000 ml"**);**

data\_array**.**add**(**data9**);**

isLowLevel **=** **true;**

**return** data\_array**;**

**}**

public ArrayList**<**ArrayList**<**String**>>** packRealData**(**ArrayList**<**String**>** inventory\_data**)** **{**

ArrayList**<**ArrayList**<**String**>>** data\_array **=** **new** ArrayList**<**ArrayList**<**String**>>();**

ArrayList**<**String**>** data **=** **new** ArrayList**<>();**

Log**.**i**(**"inventory"**,** inventory\_data**.**toString**());**

**for** **(**String str **:** inventory\_data**)** **{**

String**[]** item\_parts **=** str**.**split**(**" "**);**

data**.**add**(**item\_parts**[**0**]);**

data**.**add**(**item\_parts**[**1**]);**

data**.**add**(**item\_parts**[**2**]);**

double ratio\_of\_leftover **=** Integer**.**valueOf**(**item\_parts**[**1**])/**Integer**.**valueOf**(**item\_parts**[**2**]);**

//sends a notification if any drink is below 50% fluid level

**if(**ratio\_of\_leftover **<** 0.5**){**

isLowLevel **=** **true;**

**}**

data\_array**.**add**(new** ArrayList**<**String**>(**data**));**

data**.**clear**();**

**}**

**return** data\_array**;**

**}**

**}**

#### SalesFragment.java

package com**.**example**.**cabla**.**drinkmachinephoneapp**;**

**import** android**.**os**.**Bundle**;**

**import** android**.**support**.**v4**.**app**.**Fragment**;**

**import** android**.**util**.**Log**;**

**import** android**.**view**.**LayoutInflater**;**

**import** android**.**view**.**View**;**

**import** android**.**view**.**ViewGroup**;**

**import** android**.**widget**.**AdapterView**;**

**import** android**.**widget**.**ArrayAdapter**;**

**import** android**.**widget**.**Spinner**;**

**import** android**.**widget**.**TextView**;**

**import** com**.**github**.**mikephil**.**charting**.**charts**.**BarChart**;**

**import** com**.**github**.**mikephil**.**charting**.**components**.**AxisBase**;**

**import** com**.**github**.**mikephil**.**charting**.**components**.**XAxis**;**

**import** com**.**github**.**mikephil**.**charting**.**components**.**YAxis**;**

**import** com**.**github**.**mikephil**.**charting**.**data**.**BarData**;**

**import** com**.**github**.**mikephil**.**charting**.**data**.**BarDataSet**;**

**import** com**.**github**.**mikephil**.**charting**.**data**.**BarEntry**;**

**import** com**.**github**.**mikephil**.**charting**.**formatter**.**AxisValueFormatter**;**

**import** com**.**github**.**mikephil**.**charting**.**utils**.**ColorTemplate**;**

**import** java**.**util**.**ArrayList**;**

/\*\*

\* A simple {@link Fragment} subclass.

\*/

public class SalesFragment **extends** Fragment **{**

boolean no\_status\_data**;**

ArrayList**<**String**>** sales\_info**;**

ArrayList**<**String**>** dates**;**

ArrayList**<**BarEntry**>** bar\_chart\_entries **=** **new** ArrayList**<>();**

String total\_sales **=** "$45"**;**

Float axis\_size **=** 10f**;** //placeholder value

String**[]** drink\_names**;**

Boolean initialDisplay **=** **true;**

String **[]** dates\_for\_spin**;**

String **[]** default\_values **=** **{**"Dates"**,**"11/17/2018"**};**

//maybe only load the latest 5-8 dates

public SalesFragment**()** **{**

// Required empty public constructor

**}**

@Override

public View onCreateView**(**LayoutInflater inflater**,** ViewGroup container**,**

Bundle savedInstanceState**)** **{**

// Inflate the layout for this fragment

sales\_info **=** getArguments**().**getStringArrayList**(**"sales"**);**

no\_status\_data **=** getArguments**().**getBoolean**(**"no\_data"**)** **;**

dates **=** getArguments**().**getStringArrayList**(**"dates"**);**

**if(**no\_status\_data **==** **false){**

**if(**dates **!=** **null)** **{**

dates\_for\_spin **=** **new** String**[**dates**.**size**()** **+** 1**];**

int counter **=** 0**;**

dates\_for\_spin**[**counter**]** **=** "-----"**;**

**for** **(**String temp **:** dates**)** **{**

dates\_for\_spin**[**counter **+** 1**]** **=** temp**;**

counter**++;**

**}**

**}else{** dates\_for\_spin **=** default\_values**;** **}**

**}**

**else{**

dates\_for\_spin **=** default\_values**;**

**}**

View view\_frag **=** inflater**.**inflate**(**R**.**layout**.**fragment\_sales2**,** container**,** **false);**

BarChart sales\_chart **=** view\_frag**.**findViewById**(**R**.**id**.**sales\_bar\_chart**);**

sales\_chart**.**invalidate**();**

TextView rev\_val **=** view\_frag**.**findViewById**(**R**.**id**.**total\_revenue\_value**);**

Spinner day\_of\_sales\_spinner **=** view\_frag**.**findViewById**(**R**.**id**.**day\_of\_sales\_spinner**);**

ArrayAdapter**<**String**>** adapter **=** **new** ArrayAdapter**<>(this.**getActivity**(),**

android**.**R**.**layout**.**simple\_spinner\_item**,**dates\_for\_spin**);**

adapter**.**setDropDownViewResource**(**android**.**R**.**layout**.**simple\_spinner\_dropdown\_item**);**

day\_of\_sales\_spinner**.**setAdapter**(**adapter**);**

**if(!**no\_status\_data**){**

prepareSalesInfo**();**

**}**

setupCallbacks**(**day\_of\_sales\_spinner**,**view\_frag**,**sales\_chart**,**rev\_val**);**

**return** view\_frag **;**

**}**

public void prepareSalesInfo**(){**

Log**.**i**(**"sales"**,** sales\_info**.**toString**());**

drink\_names **=** **new** String**[**sales\_info**.**size**()+**1**];**

drink\_names**[**0**]** **=** ""**;** //used to offset the first column on graph

int count **=** 1**;**

double sum **=** 0**;**

**for** **(**String str **:** sales\_info**)** **{**

String**[]** drink\_parts **=** str**.**split**(**" "**);**

drink\_names**[**count**]** **=** drink\_parts**[**0**].**replace**(**"\_and\_"**,**" & "**).**replace**(**"\_"**,**" "**);**

sum **=** sum **+** Double**.**valueOf**(**drink\_parts**[**1**]);**

bar\_chart\_entries**.**add**(new** BarEntry**(**count**,** Float**.**valueOf**(**drink\_parts**[**1**])));**

count**++;**

**}**

total\_sales **=** "$"**+** String**.**valueOf**(**sum**)+**"0"**;**

axis\_size **=** sales\_info**.**size**()+**1f**;**

**}**

public void setupCallbacks**(**Spinner spinner**,** View view**,** final BarChart sales\_chart**,**final TextView total**){**

spinner**.**setOnItemSelectedListener**(new** AdapterView**.**OnItemSelectedListener**()** **{**

@Override

public void onItemSelected**(**AdapterView**<?>** parent**,** View view**,** int position**,** long id**)** **{**

**if(**initialDisplay**){**

initialDisplay **=** **false;** //onItemSelected is called when everything is first initialized

// so this flag ignores that first call, but allows any call after

**}else{**

sales\_chart**.**invalidate**();**

sales\_chart**.**setVisibleXRangeMaximum**(**axis\_size**+**5**);**

total**.**setText**(**total\_sales**);** //default is "$45" if no other data is present

**if(**no\_status\_data**)** **{** axis\_size **=** **(**float**)** 7**;** **}**

**else{**

//set inside prepareSalesInfo

**}**

//axis configuration

XAxis x\_axis **=** sales\_chart**.**getXAxis**();**

**if(**no\_status\_data **==** **false){**

x\_axis**.**setLabelCount**(**sales\_info**.**size**()+**1**);**

**}**

x\_axis**.**setPosition**(**XAxis**.**XAxisPosition**.**BOTTOM**);**

x\_axis**.**setTextSize**(**2f**);**

x\_axis**.**setAxisMinValue**(**0f**);**

x\_axis**.**setAxisMaxValue**(**axis\_size**);**

x\_axis**.**setLabelRotationAngle**(-**30**);**

YAxis left\_axis **=** sales\_chart**.**getAxisLeft**();**

YAxis right\_axis **=** sales\_chart**.**getAxisRight**();**

left\_axis**.**setDrawZeroLine**(true);**

left\_axis**.**setAxisMinValue**(**0f**);**

right\_axis**.**setDrawLabels**(false);**

sales\_chart**.**getAxisRight**().**setEnabled**(false);**

**if(**no\_status\_data**)** **{**

bar\_chart\_entries**.**add**(new** BarEntry**(**1**,** 5**));**

bar\_chart\_entries**.**add**(new** BarEntry**(**2**,** 6**));**

bar\_chart\_entries**.**add**(new** BarEntry**(**3**,** 7**));**

bar\_chart\_entries**.**add**(new** BarEntry**(**4**,** 8**));**

bar\_chart\_entries**.**add**(new** BarEntry**(**5**,** 9**));**

bar\_chart\_entries**.**add**(new** BarEntry**(**6**,** 10**));**

**}else{**

//set inside prepareSalesInfo

**}**

BarDataSet data\_for\_display **=** **new** BarDataSet**(**bar\_chart\_entries**,**

"Revenue from drink menu"**);**

data\_for\_display**.**setColors**(**ColorTemplate**.**JOYFUL\_COLORS**);**

**if(**no\_status\_data**)** **{**

drink\_names **=** **new** String**[**7**];**

drink\_names**[**0**]** **=** ""**;**

drink\_names**[**1**]** **=** "Cuba Libre"**;**

drink\_names**[**2**]** **=** "Daiquiri"**;**

drink\_names**[**3**]** **=** "Screwdriver"**;**

drink\_names**[**4**]** **=** "Tequila"**;**

drink\_names**[**5**]** **=** "Vodka"**;**

drink\_names**[**6**]** **=** "Vodka & Cranberry"**;**

**}else{**

//set inside prepareSalesInfo

**}**

//needs to be an arraylist object before put on X-axis

final ArrayList**<**String**>** xEntrys **=** **new** ArrayList**<>();**

**for(**int i **=** 0**;** i **<** drink\_names**.**length**;** i**++){**

Log**.**i**(**"DRINKS"**,**drink\_names**[**i**]);**

xEntrys**.**add**(**drink\_names**[**i**]);**

**}**

//put drink names on X-axis

x\_axis**.**setValueFormatter**(new** AxisValueFormatter**()** **{**

@Override

public String getFormattedValue**(**float value**,** AxisBase axis**)** **{**

**return** xEntrys**.**get**((**int**)**value **%** xEntrys**.**size**());**

**}**

@Override

public int getDecimalDigits**()** **{**

**return** 0**;**

**}**

**});**

BarData data **=** **new** BarData**(**data\_for\_display**);**

data**.**setBarWidth**(**0.5f**);**

sales\_chart**.**setScaleXEnabled**(false);**

//sales\_chart.setScaleYEnabled(false);

sales\_chart**.**setPinchZoom**(false);**

sales\_chart**.**setDrawGridBackground**(false);**

sales\_chart**.**getAxisRight**().**setDrawGridLines**(false);**

sales\_chart**.**getAxisLeft**().**setDrawGridLines**(false);**

sales\_chart**.**getXAxis**().**setDrawGridLines**(false);**

sales\_chart**.**setDescription**(**"Daily Drink Sales"**);**

sales\_chart**.**setDescriptionPosition**(**150**,**30**);**

sales\_chart**.**getLegend**().**setEnabled**(false);**

sales\_chart**.**setData**(**data**);**

**}**

**}**

@Override

public void onNothingSelected**(**AdapterView**<?>** parent**)** **{}** //needed for proper use

**});**

**}**

**}**

#### StatusFragment.java

package com**.**example**.**cabla**.**drinkmachinephoneapp**;**

**import** android**.**os**.**Bundle**;**

**import** android**.**support**.**v4**.**app**.**Fragment**;**

**import** android**.**util**.**Log**;**

**import** android**.**view**.**LayoutInflater**;**

**import** android**.**view**.**View**;**

**import** android**.**view**.**ViewGroup**;**

**import** android**.**widget**.**TextView**;**

**import** java**.**util**.**ArrayList**;**

/\*\*

\* A simple {@link Fragment} subclass.

\*/

public class StatusFragment **extends** Fragment **{**

public StatusFragment**()** **{**

// Required empty public constructor

**}**

@Override

public View onCreateView**(**LayoutInflater inflater**,** ViewGroup container**,**

Bundle savedInstanceState**)** **{**

View view\_frag **=** inflater**.**inflate**(**R**.**layout**.**fragment\_status2**,** container**,** **false);**

TextView mode\_var **=** view\_frag**.**findViewById**(**R**.**id**.**mode\_variable**);**

TextView status\_var **=** view\_frag**.**findViewById**(**R**.**id**.**status\_variable**);**

TextView cur\_user\_var **=** view\_frag**.**findViewById**(**R**.**id**.**cur\_user\_variable**);**

ArrayList**<**String**>** status\_info **=** getArguments**().**getStringArrayList**(**"status"**);**

boolean no\_status\_data **=** getArguments**().**getBoolean**(**"no\_data"**)** **;**

**if(!**no\_status\_data**)** **{**

status\_var**.**setText**(**status\_info**.**get**(**0**));**

mode\_var**.**setText**(**status\_info**.**get**(**1**));**

cur\_user\_var**.**setText**(**status\_info**.**get**(**2**));**

**}**

**else{**

status\_var**.**setText**(**"Dummy Data"**);**

**}**

**return** view\_frag**;**

**}**

**}**

#### DataRequester.java

package com**.**example**.**cabla**.**drinkmachinephoneapp**;**

**import** android**.**util**.**Log**;**

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**InputStreamReader**;**

**import** java**.**net**.**HttpURLConnection**;**

**import** java**.**net**.**MalformedURLException**;**

**import** java**.**net**.**URL**;**

**import** java**.**text**.**ParseException**;**

**import** java**.**text**.**SimpleDateFormat**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Date**;**

**import** java**.**util**.**regex**.**Matcher**;**

**import** java**.**util**.**regex**.**Pattern**;**

public class DataRequester **implements** Runnable **{**

private volatile ArrayList**<**String**>** status\_store **=** **new** ArrayList**<>();**

private volatile ArrayList**<**String**>** inventory\_store **=** **new** ArrayList**<>();**

private volatile ArrayList**<**String**>** sales\_store **=** **new** ArrayList**<>();**

private volatile ArrayList**<**String**>** dates **=** **new** ArrayList**<>();**

private volatile boolean noConnection **=** **false;**

private volatile boolean isDone **=** **false;**

private String url **=** "" **;**

DataRequester**(**String desired\_url**){**

**if(**url **!=** **null){**

url **=** desired\_url**;**

**}**

**else{**

url **=** "http://10.0.0.35:8000/shared\_data\_2018-11-22.txt"**;**

**}**

**}**

@Override

public void run**(){**

retrieveFiles**();**

**}**

public boolean getError**(){**

**return** noConnection**;**

**}**

public ArrayList**<**String**>** getDates**(){**

**return** dates**;**

**}**

public ArrayList**<**String**>** getStatusData**(){**

**return** status\_store**;**

**}**

public ArrayList**<**String**>** getInventoryData**(){**

**return** inventory\_store**;**

**}**

public ArrayList**<**String**>** getSalesData**(){**

**return** sales\_store**;**

**}**

public void retrieveFiles**(){**

URL file\_location**;**

BufferedReader buffer**;**

**try** **{**

file\_location **=** **new** URL**(**url**);**

Log**.**i**(**"CHRIS1"**,**file\_location**.**getFile**());**

String name **=**file\_location**.**getFile**();**

String regex**;**

Matcher m **;**

Date date**;**

**try{**

String**[]** temp\_str **=** name**.**split**(**"/"**);**

int counter\_t **=** 0**;**

**for(**String i**:** temp\_str**){**

Log**.**i**(**"CHRIS\_DATE1"**,**temp\_str**[**counter\_t**]);**

counter\_t**++;**

Log**.**i**(**"CHRIS\_DATE2"**,**i**);**

**if(**i**.**contains**(**"shared\_data"**)){**

regex **=** "(\\d{4}-\\d{2}-\\d{2})"**;** //year month day

m **=** Pattern**.**compile**(**regex**).**matcher**(**i**);**

**if(**m**.**find**())** **{**

Log**.**i**(**"CHRIS\_DATE3"**,**m**.**group**(**1**));**

date **=** **new** SimpleDateFormat**(**"yyyy-MM-dd"**).**parse**(**m**.**group**(**1**));**

dates**.**add**(**date**.**toString**());**

Log**.**i**(**"CHRIS\_DATE4"**,**date**.**toString**());**

**}else{**

Log**.**i**(**"CHRIS\_DATE5"**,**"couldn't find"**);**

**}**

**}**

**}**

**}**

**catch(**ParseException e**){** Log**.**i**(**"CHRIS\_TEST\_REG"**,**e**.**toString**());** **}**

HttpURLConnection huc **=** **(**HttpURLConnection**)**file\_location**.**openConnection**()** **;**

HttpURLConnection**.**setFollowRedirects**(false);**

int ONE\_SEC **=** 1000**;**

huc**.**setConnectTimeout**(**3 **\*** ONE\_SEC**);**

huc**.**setReadTimeout**(**3 **\*** ONE\_SEC**);**

huc**.**setRequestMethod**(**"GET"**);**

huc**.**connect**();**

buffer **=** **new** BufferedReader**(new** InputStreamReader**(**huc**.**getInputStream**()));**

String line**;**

boolean statusSaveKey **=** **false;**

boolean inventorySaveKey **=** **false;**

boolean salesSaveKey **=** **false;**

**while((**line **=** buffer**.**readLine**())!=** **null)** **{**

**if** **(**statusSaveKey **&&** **!**line**.**contains**(**"INVENTORY"**))** **{**

Log**.**i**(**"CHRIS"**,**line**);**

status\_store**.**add**(**line**);**

**}**

**if** **(**inventorySaveKey **&&** **!**line**.**contains**(**"SALES"**))** **{**

Log**.**i**(**"CHRIS"**,**line**);**

inventory\_store**.**add**(**line**);**

**}**

**if** **(**salesSaveKey**)** **{**

Log**.**i**(**"CHRIS"**,**line**);**

sales\_store**.**add**(**line**);**

**}**

**if** **(**line**.**contains**(**"STATUS"**))** **{**

statusSaveKey **=** **true;**

**}**

**if** **(**line**.**contains**(**"INVENTORY"**))** **{**

statusSaveKey **=** **false;**

inventorySaveKey **=** **true;**

**}**

**if** **(**line**.**contains**(**"SALES"**))** **{**

inventorySaveKey **=** **false;**

salesSaveKey **=** **true;**

**}**

**}**

Log**.**i**(**"CHRIS\_STATUS"**,**status\_store**.**toString**());**

Log**.**i**(**"CHRIS\_INVENTORY"**,**inventory\_store**.**toString**());**

Log**.**i**(**"CHRIS\_SALES"**,**sales\_store**.**toString**());**

buffer**.**close**();**

**}**

**catch** **(**MalformedURLException bad\_url**)** **{**

Log**.**i**(**"CHRIS"**,** "bad url used;" **+** bad\_url**);**

noConnection **=** **true;**

**}**

**catch** **(**IOException io\_err**)** **{**

Log**.**i**(**"CHRIS"**,** "IO error?" **+** io\_err**.**toString**());**

noConnection **=** **true;**

**}**

**catch** **(**Error er**){**

noConnection **=** **true;**

**}**

**}**

**}**

#### IP\_Config.java

package com**.**example**.**cabla**.**drinkmachinephoneapp**;**

**import** android**.**content**.**Context**;**

**import** android**.**content**.**Intent**;**

**import** android**.**content**.**SharedPreferences**;**

**import** android**.**support**.**v7**.**app**.**AppCompatActivity**;**

**import** android**.**os**.**Bundle**;**

**import** android**.**util**.**Log**;**

**import** android**.**view**.**View**;**

**import** android**.**widget**.**Button**;**

**import** android**.**widget**.**EditText**;**

**import** android**.**widget**.**TextView**;**

**import** android**.**widget**.**Toast**;**

public class IP\_Config **extends** AppCompatActivity **{**

TextView ip\_entry**;**

Button save**;**

Button back**;**

EditText new\_url**;**

@Override

protected void onCreate**(**Bundle savedInstanceState**)** **{**

**super.**onCreate**(**savedInstanceState**);**

setContentView**(**R**.**layout**.**activity\_ip\_\_config**);**

final SharedPreferences prefs**;**

save **=** findViewById**(**R**.**id**.**save**);**

back **=** findViewById**(**R**.**id**.**back**);**

new\_url **=** findViewById**(**R**.**id**.**ip\_entry**);**

prefs **=** getApplicationContext**().**getSharedPreferences**(**"prefs"**,** Context**.**MODE\_PRIVATE**);**

ip\_entry **=** findViewById**(**R**.**id**.**ip\_entry**);**

ip\_entry**.**setText**(**prefs**.**getString**(**"url"**,**"http://10.0.0.35:8000/shared\_data\_2018-11-22.txt"**));**

final Intent previous **=** **new** Intent**(**"com.example.cabla.drinkmachinephoneapp.User"**);**

save**.**setOnClickListener**(new** View**.**OnClickListener**()** **{**

@Override

public void onClick**(**View v**)** **{**

SharedPreferences**.**Editor editor **=** prefs**.**edit**();**

editor**.**putString**(**"url"**,**new\_url**.**getText**().**toString**());**

editor**.**apply**();**

Toast**.**makeText**(**getApplicationContext**(),**"Updated URL"**,**Toast**.**LENGTH\_LONG**).**show**();**

startActivity**(**previous**);**

**}**

**});**

back**.**setOnClickListener**(new** View**.**OnClickListener**()** **{**

@Override

public void onClick**(**View v**)** **{**

startActivity**(**previous**);**

**}**

**});**

**}**

**}**

#### AndroidManifest.xml

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<manifest xmlns:android=**"http://schemas.android.com/apk/res/android"**

package=**"com.example.cabla.drinkmachinephoneapp"**>

<uses-permission android:name=**"android.permission.INTERNET"** />

<application

android:allowBackup=**"true"**

android:icon=**"@mipmap/adds\_icon"**

android:label=**"@string/app\_name"**

android:roundIcon=**"@mipmap/adds\_icon\_round"**

android:supportsRtl=**"true"**

android:theme=**"@style/AppTheme"**>

<activity android:name=**".MainActivity"**>

<intent-filter>

<action android:name=**"android.intent.action.MAIN"** />

<category android:name=**"android.intent.category.LAUNCHER"** />

</intent-filter>

</activity>

<activity

android:name=**".User"**

android:label=**"@string/title\_activity\_user"**

android:theme=**"@style/AppTheme.NoActionBar"**>

<intent-filter>

<action android:name=**"com.example.cabla.drinkmachinephoneapp.User"** />

<category android:name=**"android.intent.category.DEFAULT"** />

</intent-filter>

</activity>

<activity

android:name=**".IP\_Config"**

android:label=**"@string/title\_activity\_ip\_config"**

android:theme=**"@style/AppTheme.NoActionBar"**>

<intent-filter>

<action android:name=**"com.example.cabla.drinkmachinephoneapp.IP\_Config"** />

<category android:name=**"android.intent.category.DEFAULT"** />

</intent-filter>

</activity>

</application>

</manifest>

#### Menu\_user.xml

<menu xmlns:android=**"http://schemas.android.com/apk/res/android"**

xmlns:app=**"http://schemas.android.com/apk/res-auto"**

xmlns:tools=**"http://schemas.android.com/tools"**

tools:context=**"com.example.cabla.drinkmachinephoneapp.User"**>

<item

android:id=**"@+id/ip\_config"**

android:orderInCategory=**"100"**

android:title=**"@string/url\_config"**

app:showAsAction=**"never"** />

<item

android:id=**"@+id/request\_data"**

android:orderInCategory=**"101"**

android:title=**"@string/request\_data"**

app:showAsAction=**"never"** />

</menu>

#### Strings.xml

<resources>

<string name=**"app\_name"**>**ADDS Phone App**</string>

<string name=**"login\_page\_title"**>**Login Page**</string>

<string name=**"user\_name\_label"**>**Username**</string>

<string name=**"password\_label"**>**Password**</string>

<string name=**"login\_button"**>**Login**</string>

<string name=**"title\_activity\_user"**>**User**</string>

<string name=**"title\_activity\_ip\_config"**>**IPConfig**</string>

<string name=**"url\_config"**>**Configure URL**</string>

<string name=**"request\_data"**>**Update Data**</string>

<string name=**"section\_format"**>**Update Data**</string>

<string name=**"tab\_text\_1"**>**Inventory**</string>

<string name=**"tab\_text\_2"**>**Sales**</string>

<string name=**"tab\_text\_3"**>**System Status**</string>

<string name=**"inventory\_fragment"**>**Inventory fragment**</string>

<string name=**"subtitle1"**> **Items**</string>

<string name=**"subtitle2"**>**QTY Left**</string>

<string name=**"subtitle3"**>**Original QTY**</string>

<string name=**"total\_revenue"**>**Total revenue:**</string>

<string name=**"status\_label"**>**Status:**</string>

<string name=**"mode\_label"**>**Mode:**</string>

<string name=**"cur\_user\_label"**>**Current User:**</string>

</resources>

#### activity\_main.xml

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<android.support.constraint.ConstraintLayout xmlns:android=**"http://schemas.android.com/apk/res/android"**

xmlns:app=**"http://schemas.android.com/apk/res-auto"**

xmlns:tools=**"http://schemas.android.com/tools"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"match\_parent"**

tools:context=**".MainActivity"**>

<TextView

android:id=**"@+id/login\_title"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"wrap\_content"**

android:gravity=**"center"**

android:text=**"@string/login\_page\_title"**

android:textSize=**"35sp"** />

<TextView

android:id=**"@+id/user\_label"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:text=**"@string/user\_name\_label"**

app:layout\_constraintTop\_toBottomOf=**"@id/login\_title"**/>

<TextView

android:id=**"@+id/pass\_label"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:text=**"@string/password\_label"**

app:layout\_constraintTop\_toBottomOf=**"@id/user\_entry"**/>

<EditText

android:id=**"@+id/pass\_entry"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"wrap\_content"**

android:inputType=**"textPassword"**

app:layout\_constraintTop\_toBottomOf=**"@id/pass\_label"** />

<EditText

android:id=**"@+id/user\_entry"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"wrap\_content"**

android:inputType=**"text"**

app:layout\_constraintTop\_toBottomOf=**"@+id/user\_label"** />

<Button

android:id=**"@+id/button1"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:text=**"@string/login\_button"**

app:layout\_constraintTop\_toBottomOf=**"@id/pass\_entry"**/>

</android.support.constraint.ConstraintLayout>

#### activity\_user.xml

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<android.support.design.widget.CoordinatorLayout xmlns:android=**"http://schemas.android.com/apk/res/android"**

xmlns:app=**"http://schemas.android.com/apk/res-auto"**

xmlns:tools=**"http://schemas.android.com/tools"**

android:id=**"@+id/main\_content"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"match\_parent"**

android:fitsSystemWindows=**"true"**

tools:context=**".User"**>

<android.support.design.widget.AppBarLayout

android:id=**"@+id/appbar"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"wrap\_content"**

android:paddingTop=**"@dimen/appbar\_padding\_top"**

android:theme=**"@style/AppTheme.AppBarOverlay"**>

<android.support.v7.widget.Toolbar

android:id=**"@+id/toolbar"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"?attr/actionBarSize"**

android:layout\_weight=**"1"**

android:background=**"@color/colorPrimaryDark"**

app:layout\_scrollFlags=**"scroll|enterAlways"**

app:popupTheme=**"@style/AppTheme.PopupOverlay"**

app:title=**"@string/app\_name"**>

</android.support.v7.widget.Toolbar>

<android.support.design.widget.TabLayout

android:id=**"@+id/tabs"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"wrap\_content"**

app:tabIndicatorColor=**"@color/colorPrimary"**>

<android.support.design.widget.TabItem

android:id=**"@+id/tabItem"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:text=**"@string/tab\_text\_1"** />

<android.support.design.widget.TabItem

android:id=**"@+id/tabItem2"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:text=**"@string/tab\_text\_2"** />

<android.support.design.widget.TabItem

android:id=**"@+id/tabItem3"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:text=**"@string/tab\_text\_3"** />

</android.support.design.widget.TabLayout>

</android.support.design.widget.AppBarLayout>

<android.support.v4.view.ViewPager

android:id=**"@+id/container"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"match\_parent"**

app:layout\_behavior=**"@string/appbar\_scrolling\_view\_behavior"** />

</android.support.design.widget.CoordinatorLayout>

#### activity\_ip\_\_config

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<android.support.constraint.ConstraintLayout xmlns:android=**"http://schemas.android.com/apk/res/android"**

xmlns:app=**"http://schemas.android.com/apk/res-auto"**

xmlns:tools=**"http://schemas.android.com/tools"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"match\_parent"**

tools:context=**".IP\_Config"**>

<TextView

android:id=**"@+id/ip\_label"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

app:layout\_constraintTop\_toTopOf=**"parent"**

android:text=**"URL:"**>

</TextView>

<EditText

android:id=**"@+id/ip\_entry"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"wrap\_content"**

android:text=**""**

android:inputType=**"text"**

app:layout\_constraintTop\_toBottomOf=**"@id/ip\_label"**>

</EditText>

<Button

android:id=**"@+id/save"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:text=**"Save"**

app:layout\_constraintTop\_toBottomOf=**"@id/ip\_entry"** />

<Button

android:id=**"@+id/back"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:text=**"Back"**

app:layout\_constraintTop\_toBottomOf=**"@id/ip\_entry"**

app:layout\_constraintLeft\_toRightOf=**"@id/save"**

/>

</android.support.constraint.ConstraintLayout>

#### fragment\_inventory.xml

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<ScrollView

xmlns:android=**"http://schemas.android.com/apk/res/android"**

xmlns:tools=**"http://schemas.android.com/tools"**

android:scrollbars=**"none"**

android:layout\_width=**"fill\_parent"**

android:layout\_height=**"fill\_parent"**

android:layout\_weight=**"1"**>

<TableLayout

android:id=**"@+id/inventory\_table"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"wrap\_content"**

tools:context=**".InventoryFragment"**

android:stretchColumns=**"\*"**>

<TableRow

android:background=**"@drawable/cell\_border"**>

<TextView

android:id=**"@+id/Inventory\_title"**

android:text= **"@string/subtitle1"**

android:textSize=**"15sp"**

android:gravity=**"center"**>

</TextView>

<TextView

android:id=**"@+id/Quantity\_left\_title"**

android:text=**"@string/subtitle2"**

android:textSize=**"15sp"**

android:gravity=**"center"**>

</TextView>

<TextView

android:id=**"@+id/Quantity\_original\_title"**

android:text=**"@string/subtitle3"**

android:textSize=**"15sp"**

android:gravity=**"center"**

android:layout\_width=**"match\_parent"**>

</TextView>

</TableRow>

</TableLayout>

</ScrollView>

#### fragment\_sales2.xml

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<android.support.constraint.ConstraintLayout xmlns:android=**"http://schemas.android.com/apk/res/android"**

xmlns:app=**"http://schemas.android.com/apk/res-auto"**

xmlns:tools=**"http://schemas.android.com/tools"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"match\_parent"**

tools:context=**".MainActivity"**>

<Spinner

android:id=**"@+id/day\_of\_sales\_spinner"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"wrap\_content"**

/>

<TextView

android:id=**"@+id/total\_revenue\_label"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_marginTop=**"20sp"**

android:text=**"@string/total\_revenue"**

android:textSize=**"25sp"**

android:visibility=**"visible"**

app:layout\_constraintTop\_toBottomOf=**"@id/day\_of\_sales\_spinner"** />

<TextView

android:id=**"@+id/total\_revenue\_value"**

android:layout\_width=**"0dp"**

android:layout\_height=**"wrap\_content"**

android:layout\_marginLeft=**"20sp"**

android:layout\_marginTop=**"20dp"**

android:text=**"$0.00"**

android:textSize=**"25sp"**

android:visibility=**"visible"**

app:layout\_constraintHorizontal\_bias=**"1.0"**

app:layout\_constraintLeft\_toRightOf=**"@id/total\_revenue\_label"**

app:layout\_constraintRight\_toRightOf=**"parent"**

app:layout\_constraintTop\_toBottomOf=**"@id/day\_of\_sales\_spinner"** />

<com.github.mikephil.charting.charts.BarChart

android:id=**"@+id/sales\_bar\_chart"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"380dp"**

app:layout\_constraintTop\_toBottomOf=**"@id/total\_revenue\_label"**

app:layout\_constraintBottom\_toBottomOf=**"parent"**

android:layout\_marginBottom=**"40dp"**

android:layout\_marginTop=**"10dp"**

>

</com.github.mikephil.charting.charts.BarChart>

</android.support.constraint.ConstraintLayout>

#### fragment\_status2.xml

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<android.support.constraint.ConstraintLayout xmlns:android=**"http://schemas.android.com/apk/res/android"**

xmlns:app=**"http://schemas.android.com/apk/res-auto"**

xmlns:tools=**"http://schemas.android.com/tools"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"match\_parent"**

tools:context=**".MainActivity"**>

<TextView

android:id=**"@+id/status\_label"**

android:layout\_width=**"144dp"**

android:layout\_height=**"wrap\_content"**

android:text=**"@string/status\_label"**

android:textSize=**"25sp"**

android:gravity=**"center"**/>

<TextView

android:id=**"@+id/status\_variable"**

android:layout\_width=**"0dp"**

android:layout\_height=**"wrap\_content"**

android:text=**""**

android:textSize=**"25sp"**

app:layout\_constraintLeft\_toRightOf=**"@+id/status\_label"**

app:layout\_constraintRight\_toRightOf=**"parent"**

android:layout\_marginLeft=**"20dp"**/>

<TextView

android:id=**"@+id/mode\_label"**

android:layout\_width=**"144dp"**

android:layout\_height=**"wrap\_content"**

android:text=**"@string/mode\_label"**

android:textSize=**"25sp"**

app:layout\_constraintTop\_toBottomOf=**"@id/status\_label"**

android:gravity=**"center"**/>

<TextView

android:id=**"@+id/mode\_variable"**

android:layout\_width=**"0dp"**

android:layout\_height=**"wrap\_content"**

android:text=**""**

android:textSize=**"25sp"**

app:layout\_constraintLeft\_toRightOf=**"@+id/mode\_label"**

app:layout\_constraintRight\_toRightOf=**"parent"**

app:layout\_constraintTop\_toBottomOf=**"@id/status\_variable"**

android:layout\_marginLeft=**"20dp"**/>

<TextView

android:id=**"@+id/cur\_user\_label"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:text=**"@string/cur\_user\_label"**

android:textSize=**"25sp"**

android:gravity=**"center"**

app:layout\_constraintTop\_toBottomOf=**"@id/mode\_label"** />

<TextView

android:id=**"@+id/cur\_user\_variable"**

android:layout\_width=**"0dp"**

android:layout\_height=**"wrap\_content"**

android:text=**""**

android:textSize=**"25sp"**

app:layout\_constraintLeft\_toRightOf=**"@+id/cur\_user\_label"**

app:layout\_constraintRight\_toRightOf=**"parent"**

app:layout\_constraintTop\_toBottomOf=**"@id/mode\_variable"**

android:layout\_marginLeft=**"20dp"**/>

</android.support.constraint.ConstraintLayout>

#### fragment\_user.xml

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<android.support.constraint.ConstraintLayout xmlns:android=**"http://schemas.android.com/apk/res/android"**

xmlns:app=**"http://schemas.android.com/apk/res-auto"**

xmlns:tools=**"http://schemas.android.com/tools"**

android:id=**"@+id/constraintLayout"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"match\_parent"**

tools:context=**".User$PlaceholderFragment"**>

<TextView

android:id=**"@+id/section\_label"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_marginStart=**"@dimen/activity\_horizontal\_margin"**

android:layout\_marginTop=**"@dimen/activity\_vertical\_margin"**

android:layout\_marginEnd=**"@dimen/activity\_horizontal\_margin"**

android:layout\_marginBottom=**"@dimen/activity\_vertical\_margin"**

app:layout\_constraintLeft\_toLeftOf=**"parent"**

app:layout\_constraintTop\_toTopOf=**"@+id/constraintLayout"**

tools:layout\_constraintLeft\_creator=**"1"**

tools:layout\_constraintTop\_creator=**"1"** />

</android.support.constraint.ConstraintLayout>

# References

Altium. (2017, April 20). *Via*. Retrieved from Altium Documentation: https://www.altium.com/documentation/16.0/display/ADES/PCB\_Obj-Via((Via))\_AD

Android Developers. (2018, April 25). *Core App Quality*. Retrieved from Android Developers: https://developer.android.com/docs/quality-guidelines/core-app-quality

Android Developers. (2018, October 27th). *Distribution dashboard*. Retrieved from Developers: https://developer.android.com/about/dashboards/

BMOW. (2018, May 26th). *Raspberry Pi GPIO Programming in C*. Retrieved from BMOW: https://www.bigmessowires.com/2018/05/26/raspberry-pi-gpio-programming-in-c/

Duzin, Y. (2016, August 30th). *Raspberry Pi 3 Spec, GPIO*. Retrieved from OverIt: http://overit.tistory.com/entry/RaspberryPI-3-Spec-GPIO

Intellipaat. (2018). *Mongodb Crud Operations*. Retrieved from Intellipaat: https://intellipaat.com/tutorial/mongodb-tutorial/mongodb-crud-operations/

International Rectifier. (2004, July 16). IRF9953PbF - HEXFET Power MOSFET.

leon earl. (2018, June 8th). *MVC architecture*. Retrieved from MDN web docs: [1] https://developer.mozilla.org/en-US/docs/Web/Apps/Fundamentals/Modern\_web\_app\_architecture/MVC\_architecture (used for MVC)

Microchip Technology Inc. (2008, September). DS22103A - MCP23018/MCP23S18 16-Bit I/O Expander with Open-Drain Outputs.

Microchip Technology Inc. (2010). DS21935D - TCN75A 2-Wire Serial Temperature Sensor.

ON Semiconductor. (2014, February). *Using MOSFETs in Load Switch Applications.* Retrieved from On Semiconductor: https://www.onsemi.com/pub/Collateral/AND9093-D.PDF

O'Reilly. (2002). *The grid Geometry Manager*. Retrieved from Mastering Perl/TK: https://docstore.mik.ua/orelly/perl3/tk/ch02\_02.htm

Richtek Technology Corporation. (2016, February). DS9058-01 - RT9058 36V 100mA Linear Regulator.

ROHM Semiconductor. (2016, July 11). UM6K31N - Nch+Nch 60V 250mA Small Signal MOSFET.

Rossum, G. v. (2001, July 5th). *PEP 8 -- Style Guide for Python Code*. Retrieved from Python: https://www.python.org/dev/peps/pep-0008/

Texas Instruments. (2016, May 26). DRV8871 - 3.6A Brushed DC Motor Driver With Internal Current Sense. Dallas, Texas, United States of America.