```
#!/usr/bin/env python
# Program to create a GUI to be used for brew recipe selection for Automatic Brewer
# Group 10: Automatic Brewer
# Group Members: Robert Bower (EE), Alonzo Ubilla (ME/EE), Kleber Valencia (EE), David Rodriguez (CE)

#imports
from Tkinter import*
import Tkinter
import tkMessageBox
from PIL import Image, ImageTk
import smbus
import time
import datetime
import subprocess
import sys
import smtplib
from email.mime.text import MIMEText
#import MySQLdb

bus = smbus.SMBus(1)
chip1_addr = 0x10
chip2_addr = 0x20

root = Tk()
#root.geometry("1300x700")
root.attributes('-fullscreen', True)
canvas = Tkinter.Canvas(root, height=700, width=1350,)
canvas.pack()

class Window(Frame):
   def __init__(self, master=NONE):
      Frame.__init__(self,master)
      self.master = master
      self.init_window()

   def init_window(self):
      self.master.title("Automatic Brewer Recipe Application")
```

```
    self.pack(fill=BOTH, expand=1)

    menu_bar = Menu(self.master)
    self.master.config(menu=menu_bar)
    # create the file_menu object to add to frame menu
    file_menu = Menu(menu_bar, tearoff=0)

    # create file menu & the three sub-menus
    file_menu.add_command(label="Open Recipe", command=self.open_recipe)
    file_menu.add_separator()
    file_menu.add_command(label="Exit", command=self.client_exit)
    menu_bar.add_cascade(label="File", menu=file_menu)

    # create help menu & one sub-menu
    help_menu = Menu(menu_bar, tearoff=0)
    help_menu.add_command(label="About", command=self.about_info)
    menu_bar.add_cascade(label="Help", menu=help_menu)

    # display the menu
    root.config(menu=menu_bar)

    welcome_label = Tkinter.Label(canvas, text='AUTO BREW RECIPE INPUT', font='Arial 25 bold').place(x=452,y=5)

    # Canvas object creation for better aesthetics
    canvas.create_rectangle(75, 390, 350, 650, fill='dark grey')
    canvas.create_rectangle(375, 390, 650, 650, fill='dark grey')
    canvas.create_rectangle(700, 390, 975, 650, fill='dark grey')
    canvas.create_rectangle(1000, 390, 1275, 650, fill='dark grey')
    self.create_mash_elements()
    self.create_boil_elements()
    self.create_hops_elements()
    self.create_cool_elements()
    self.create_op_mode()
    self.recipe_info()
    self.exit_start_btn()

def create_mash_elements(self):
```

```
    # Mash step input functionality, label creation and entry box
    mash_tun_box = Tkinter.Label(canvas, bg='dark grey', font='Arial 18 bold underline', text='MASH STEP')
    mash_tun_box.place(x=140, y=395)
    mash_temp_label = Tkinter.Label(canvas, bg='dark grey', font='Arial 12 bold', text='Mash Temperature(Celsius):')
    mash_temp_label.place(x=105, y=450)
    mash_temp_entry = Tkinter.Entry(canvas, textvariable=mash_temp)
    mash_temp_entry.place(x=105, y=475)
    mash_time_label = Tkinter.Label(canvas, bg='dark grey', font='Arial 12 bold', text='Mash Time(Minutes):')
    mash_time_label.place(x=105, y=520)
    mash_time_entry = Tkinter.Entry(canvas, textvariable=mash_time)
    mash_time_entry.place(x=105, y=545)

def create_boil_elements(self):
    # Boiling step input functionality, label creation and entry box
    boil_box = Tkinter.Label(canvas, bg='dark grey', font='Arial 18 bold underline', text='BOIL STEP')
    boil_box.place(x=450, y=395)
    weight_label = Tkinter.Label(canvas, bg='dark grey', font='Arial 12 bold', text='Boil Temperature(Celsius):')
    weight_label.place(x=410, y=450)
    weight_entry = Tkinter.Entry(canvas, textvariable=boil_temp)
    weight_entry.place(x=410, y=475)
    weight_label = Tkinter.Label(canvas, bg='dark grey', font='Arial 12 bold', text='Boil Time(Minutes):')
    weight_label.place(x=410, y=520)
    weight_entry = Tkinter.Entry(canvas, textvariable=boil_time)
    weight_entry.place(x=410, y=545)

def create_hops_elements(self):
    # Hops addition step input functionality, label creation and entry box for time
    hops_box = Tkinter.Label(canvas, bg='dark grey', font='Arial 18 bold underline', text='HOPS ADD')
    hops_box.place(x=775, y=395)
    hops_label = Tkinter.Label(canvas, bg='dark grey', font='Arial 12 bold', text='Addition of Hops\n(Minutes Remaining):')
    hops_label.place(x=760, y=450)
    hops_entry = Tkinter.Entry(canvas, textvariable=hop_addition)
    hops_entry.place(x=775, y=495)

def create_cool_elements(self):
    # Cooling temperature step input functionality, label creation and entry box for needed temperature control
    cooling_box = Tkinter.Label(canvas, bg='dark grey', font='Arial 18 bold underline', text='COOLING')
```

```
    cooling_box.place(x=1075, y=395)
    weight_label = Tkinter.Label(canvas, bg='dark grey', font='Arial 12 bold', text='Cooling Temperature(Celsius):')
    weight_label.place(x=1020, y=450)
    weight_entry = Tkinter.Entry(canvas, textvariable=cool_temp)
    weight_entry.place(x=1020, y=475)

def create_op_mode(self):
    # create the auto control button
    auto_mode = Tkinter.Button(canvas, width=15, text="AUTO MODE", font='Arial 10 italic bold', fg='blue', bg='grey', command=self.auto_selected)
    auto_mode.place(x=75, y=100)

    # create the manual control button
    manual_mode = Tkinter.Button(canvas, width=15, text="MANUAL MODE", font='Arial 10 italic bold', fg='red', bg='grey', command=self.manual_selected)
    manual_mode.place(x=75, y=150)

    op_mode = Tkinter.Label(root, text="SELECTED MODE: ", font='Arial 12 bold')
    op_mode.place(x=220, y=220)

def auto_selected(self):
    operation_mode = 'AUTO MODE'
    mode_label.config(text=operation_mode, font='Arial 12 bold', fg='blue')
    root.update()
    # send the auto mode start ack to MCU 0xF0
    bus.write_byte(chip1_addr, 0xF0)
    bus.write_byte(chip1_addr, 0x00)
    return

def manual_selected(self):
    operation_mode = 'MANUAL MODE'
    mode_label.config(text=operation_mode, font='Arial 12 bold', fg='red')
    root.update()
    # send the manual mode start ack to MCU 0xF1
    bus.write_byte(chip1_addr, 0xF1)
    bus.write_byte(chip1_addr, 0x00)
    return

def recipe_info(self):
```

```
    # create the entry box for the user to enter the recipe name in order for it to be saved more efficiently
    recipe_name_label = Tkinter.Label(canvas, font='Arial 12 bold', text='Enter the name of your recipe:')
    recipe_name_label.place(x=1000, y=75)
    recipe_name_entry = Tkinter.Entry(canvas, textvariable=recipe_name)
    recipe_name_entry.place(x=1000, y=100)

    # create the entry box for the user to enter the recipe name in order for it to be saved more efficiently
    grain_type_label = Tkinter.Label(canvas, font='Arial 12 bold', text='Enter the grain type:')
    grain_type_label.place(x=1000, y=125)
    grain_type_entry = Tkinter.Entry(canvas, textvariable=grain_type)
    grain_type_entry.place(x=1000, y=150)

    # create the entry box for the user to enter the recipe name in order for it to be saved more efficiently
    hop_type_label = Tkinter.Label(canvas, font='Arial 12 bold', text='Enter the hops type:')
    hop_type_label.place(x=1000, y=175)
    hop_type_entry = Tkinter.Entry(canvas, textvariable=hop_type)
    hop_type_entry.place(x=1000, y=200)

    # create the entry box for the user to enter the recipe name in order for it to be saved more efficiently
    yeast_type_label = Tkinter.Label(canvas, font='Arial 12 bold', text='Enter the yeast type:')
    yeast_type_label.place(x=1000, y=225)
    yeast_type_entry = Tkinter.Entry(canvas, textvariable=yeast_type)
    yeast_type_entry.place(x=1000, y=250)

def exit_start_btn(self):
    # create the main exit button that will be displayed in the bottom of window
    exit_btn = Tkinter.Button(self, width=15, text="EXIT", font='Arial 9 bold', bg='red',command=self.client_exit)
    exit_btn.pack(side=BOTTOM, pady=5)

    # create the send button that will save the recipe and send the information on to the brewing system
    send_btn = Tkinter.Button(self, width=15, text="START", font='Arial 9 bold', bg='light green',command=self.sys_start)
    send_btn.pack(side=BOTTOM)

# function to alert the user whenever they decide to exit the interface
def client_exit(self):
    answer = tkMessageBox.askquestion("CAUTION!", "Are you sure you want to EXIT?\nAll your data will be lost.",icon='warning')
    if answer == 'yes':
```

```
        root.destroy()
    else:
        return

# function to show the user a pop-up with some helpful information
def about_info(self):
    tkMessageBox.showinfo("Information", "This is an application intended for you, the home brewer, that will"
                                         " allow you to control the various input variables for your wort"
                                         " extraction process and/or select from previous saved recipes.")

# function to intiate the call to the system and pass all user variables to the MCUs
# also we are saving the user inputs to user_input.txt file to later use and read back from
def sys_start(self):
    file = open("user_input.txt", "a")
    file.write("%s\n" % datetime.datetime.today())                  # safe the date/time stamp
    file.write("recipe_name = %s\n" % recipe_name.get())            # safe the recipe name
    file.write("grain_type = %s\n" % grain_type.get())              # safe the grain type of the current recipe
    file.write("hops_type = %s\n" % hop_type.get())                 # safe the hops type of the current recipe
    file.write("yeast_type = %s\n" % yeast_type.get())              # safe the year type of the current recipe
    file.write("mash_temperature = %s\n" % mash_temp.get())         # safe the user input mash temp for recipe
    file.write("mash_time = %s\n" % mash_time.get())                # safe the user input mash time for recipe
    file.write("boil_temperature = %s\n" % boil_temp.get())         # safe the user input boil temp for recipe
    file.write("boil_time = %s\n" % boil_time.get())                # safe the user input boil time for recipe
    file.write("cooling_temperature = %s\n" % cool_temp.get())      # safe the user input cooling temperature
    file.close()
    bus.write_byte(chip2_addr,0xB3)
    bus.write_byte(chip2_addr,0x00)
    # Send mash temperature input from user to MCU
    bus.write_byte(chip1_addr, 0xA1)
    bus.write_byte(chip1_addr, int(mash_temp.get()))
    # Send mash time input from user to MCU
    bus.write_byte(chip1_addr, 0xA2)
    bus.write_byte(chip1_addr, int(mash_time.get()))
    # Send boil temperature input from user to MCU
    bus.write_byte(chip1_addr, 0xA3)
    bus.write_byte(chip1_addr, int(boil_temp.get()))
    # Send boil time input from user to MCU
```

```
        bus.write_byte(chip1_addr, 0xA4)
        bus.write_byte(chip1_addr, int(boil_time.get()))
        # Send the hops addition time left input from user to MCU
        bus.write_byte(chip1_addr, 0xA5)
        bus.write_byte(chip1_addr, int(hop_addition.get()))
        # Send the minimum cooling temperature input from user to MCU
        bus.write_byte(chip1_addr, 0xA6)
        bus.write_byte(chip1_addr, int(cool_temp.get()))
        # send an email to the Rpis email account
        msg = MIMEText('Recipe name: %s\nGrain type: %s\nHops type: %s\nYeast type: %s\nTimestamp: %s\nHave a nice
day!'%(recipe_name.get(),grain_type.get(),hop_type.get(), yeast_type.get(),datetime.datetime.today()))
        msg['Subject'] = ('MESSAGE FROM AUTO BREW: RECIPE INFO')
        msg['From'] = USERNAME
        msg['To'] = MAILTO
        server = smtplib.SMTP('smtp.gmail.com:587')
        server.ehlo_or_helo_if_needed()
        server.starttls()
        server.ehlo_or_helo_if_needed()
        server.login(USERNAME, PASSWORD)
        server.sendmail(USERNAME, MAILTO, msg.as_string())
        server.quit()
        # Call second GUI to initialize the control process depending on whether the user selected Auto Mode or Manual Mode
        import AutoModeSystem3
        print("gheesd")

    def open_recipe(self):
        #tkMessageBox.showinfo("Info", "this option will allow the user to open the recipe")
        openRecipe = Tk()
        openRecipe.title("Open Recipe Application")
        openRecipe.geometry("450x400")

    def test(self):
        print(operation_mode)
        return

# create the label object to display the image along with the welcome message on the window
background_image = ImageTk.PhotoImage(Image.open('background.png'))
```

```
image_label = Tkinter.Label(canvas, image=background_image).place(x=435,y=55)

mode_label = Tkinter.Label(canvas)
mode_label.place(x=80, y=250)

# declare the variables needed to get the SMS messages out
USERNAME = "autobrewucf@gmail.com"
PASSWORD = "autobrew2015"
MAILTO = "autobrewucf@gmail.com"

# variable declarations needed for the system
mash_temp = StringVar()
mash_time = StringVar()
boil_temp = StringVar()
boil_time = StringVar()
cool_temp = StringVar()
hop_addition = StringVar()
recipe_name = StringVar()
grain_type = StringVar()
hop_type = StringVar()
yeast_type = StringVar()
operation_mode = StringVar()

# display all of our window elements from root
app = Window(root)

#start the GUI
root.mainloop()
```