Criteria C: Tools and Techniques:
Techniques

1. Python
    a. Imports
    b. Variables
    c. @app routes
    d. Functions

    JavaScript
    e. <scripts>
    f. document.getElementByID
    g. If statements

Python

A: Imports and modules

```
from flask import Flask, render_template, request, redirect
import os
import datetime
import csv
```

As you can see in the image above, I used various modules to make my program do what I wanted. The first modules is Flask. Flask is not installed by default and must be installed by the user. Flask is an easy to use Web Framework that allowed to make my program independent from my gui. Flask runs in the background, and all user input is through the ip and port number I set, in this case 127.0.0.1:5000. This means kids who close the browser window will not have to relaunch the python program, just reload the tab. The render_template function allows me to have the webpage load from a html file, and the request function allows me to get information to be used in the webpage back to the program, while the redirect function lets me send the user to different pages.

The os module lets me get the file directory I am in. This is used to set the location of the spreadsheets I use to keep track of who signs in and out. This was the most effective method to keep them all in a separate folder, still inside the current one.

```
path = os.getcwd()
f = path+'//signoutsheets//'+date_today+'signoutlog'+'.csv'
```

The datetime modules are used to get the current date and time through out my program. As the sign-out sheet is time based, it is important to use. The datetime module allows me to get the current date and time, as well as find the time in-between two dates and times. This is used to calculate the total time a student has left.

The csv module is what allows me to write to the file type I am using. Csv stands for comma separated value, and is easily converted to a spreadsheet, which can be sorted. Without the csv module, I would not be able to do this.

B: Variables

```
fname = "error"
lname = "error"
timeleft = datetime.datetime.now()
```

Variables are used throughout my program. I use it for the first name and last name of the student, dates and time, current path and files. An example of one of these are the two variables fname and lname. These both represent first name and lname respectively. They are both set outside of any functions, to allow them to be called at any time. They use the default value “error” to show when they have not been changed. The first time they are used is when the first and last name is entered from the webpage “home”. The request function gets the value for fname and lname from the webpage, and sets assigns them to the corresponding variable in the program. They are called globally to make sure the program is setting the value for all functions, as the same name will be used multiple times. It’s then used to write the name of the student into a csv file. It is used one last time, again to write to a csv file to show the student has returned. After a new student has signed in, the value is changed to match the new name.

```
global fname
global lname
global timeleft
fname = request.form['fname']
lname = request.form['lname']
lname_up = lname.upper()
fname_up = fname.upper()
```

C: @app.routes

```
@app.route('/back')
```

Routes correspond to the text after the backslash of the ip and port combo used to access the Flask app on a browser. This allows the program to understand what the user wants it to do, as well as switch between function and webpages. This separates the code into nice chunks, where it can be tested by accessing specific routes by typing different text following the ip/port.

(Source: https://palletsprojects.com/p/flask/)

D: Functions

```
def my_form_post():
 global fname
 global lname
 global timeleft
 fname = request.form['fname']
 lname = request.form['lname']
 lname_up = lname.upper()
 fname_up = fname.upper()
 date_today = datetime.datetime.now().strftime("%Y-%m-%d")
 date = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
 path = os.getcwd()
 f = path+'//signoutsheets//'+date_today+'signoutlog'+'.csv'
 timeleft = datetime.datetime.now()
 if fname != 'error' or lname != 'error':
  with open(f, mode='a+') as signout:
   signout_writer = csv.writer(signout, delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL)
   signout_writer.writerow([fname_up,lname_up,date,str("left")])

 return redirect('http://127.0.0.1:5000/back')
```

Functions allow large sections of code to be used multiple times, without typing out the code. Functions are used by Flask following routes to easily define what it wants it to do. This allows it to forget about indentation errors, as all errors will be handled by the function. If the function works, Flask can use it. Because it breaks up the code, it is much easier to test, as I can just use it at any point. Functions can also be found in my JavaScript, as it allows code to be written cleaner, while allowing easier testing

```
function teacher() {
  teachervar = true;
```

JavaScript

E: <script>

```
<script>
  // Set the date we're counting down to
  var countDownDate = new Date().getTime();
  // Update the count down every 1 second
  var x = setInterval(function () {
    // Get today's date and time
    var now = new Date().getTime();
    // Find the distance between now and the count down date
    var distance = now - timeStart;
    var colour = "white";
    var out = 0;
    document.body.style.backgroundColor = colour;
    // Time calculations for days, hours, minutes and seconds
    var days = Math.floor(distance / (1000 * 60 * 60 * 24));
    var hours = Math.floor((distance % (1000 * 60 * 60 * 24)) / (1000 * 60 * 60));
    var minutes = Math.floor((distance % (1000 * 60 * 60)) / (1000 * 60));
    var seconds = Math.floor((distance % (1000 * 60)) / 1000);
    // Display the result in the element with id="demo"
    document.getElementById("demo").innerHTML = hours + "h " +
      minutes + "m " + seconds + "s ";
    // if time is >10
    if (distance > 600000 && !teachervar) {
```

The <script> element allows me to write and run code in JavaScript on an html page. This code, unless it changes something is invisible to the user. Most of this code is a script I modified from W3schools. This script keeps track of the distance between two time values. However instead of counting down like the original timer did, it counts up to keep track of time. This is crucial to keeping track of time a student has been out. Without script elements, code would not run in html.

(source: https://www.w3schools.com/howto/howto_js_countdown.asp)

F: document.getElementByID

```
document.getElementById("demo").innerHTML = hours + "h " +
  minutes + "m " + seconds + "s ";
```

This function of JavaScript allows me to change parts of the visible html code without doing anything too crazy. When I create a visible section I want to change, I assign it an id, and when I want to change it, document.getElementByID lets me change that and only that value. This means in situation where I want to disable buttons or change a time displayed, I can do so without doing anything else.

G: If statements.

```
if (distance > 600000 && !teachervar) {
  setvariables()
  if (seconds % 2 == 0 && !teachervar)
  {
    var colour = "red"
    document.body.style.backgroundColor = colour;
  }
}
```

The largest decisions made in my program can be found in my JavaScript code. This code compares the time value and determines if it has been 10 minutes since the page loaded, and if the alarm has been turned off. This allows the page to alert the teacher if any student has been out for too long, without the teacher having to monitor it.

Word Count = 917