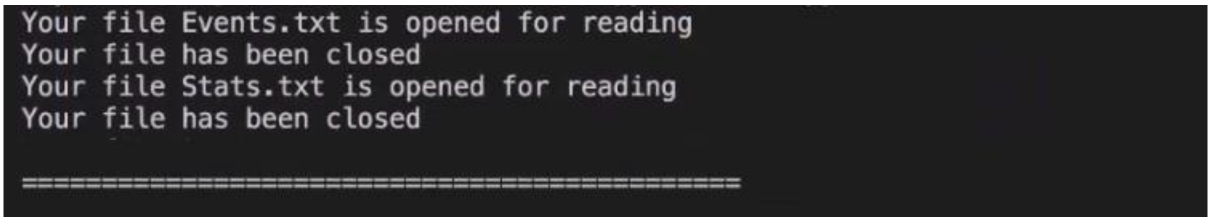
Contents

1. Initial Input

1.1 Storage of Information

The program, based on two sample data files, Events.txt and Stats.txt, simulates and provides the basis of our data. We contemplated various options to store the data but decided that the best option would an array. We think that an array would be the best fit for the scenario because it allows each individual value, including the event name to be stored. This is essential as towards the later part of the assignment, values such as mean and standard deviation must be computed and displayed by days.

Once the user has keyed in the commands for the program to read the files, a message will inform the user that both files has been opened for reading to extract the data out to be stored and closed successfully. (See Figure 1). We also included error validation for being unable to read said file.



```
Your file Events.txt is opened for reading
Your file has been closed
Your file Stats.txt is opened for reading
Your file has been closed

=============================================
```

Figure 1: Message informing the user that Events.txt and Stats.txt has been read and data has been stored

1.2 Potential inconsistencies

Inconsistencies can come from many aspects, in which one is logical and the other user input error. For logical inconsistencies, one of the first that we looked at and implemented code to validate error checking between the two files, namely Events.txt and stats.txt would be the types of events. A continuous event might have been wrongly identified as discrete event, and vice versa. Thus, to go about doing so, we checked that continuous events must be recorded to 2 decimal places and discrete events must take in integer.

Diving into the events, one possible consistency would be the time spent online. The mean (average) amount of time spent online should not exceed 1440 minutes logically as there is only a maximum of 1440 minutes per day. Thus, we implemented a condition to ensure that values there cannot be lower than 0 and more than 1440.

Other inconsistencies such as weights must always be positive as per requirements. One inconsistency that was made known but not detected was the number of events in both Events.txt and the parameters in Stats.txt stated could be different.

2. Activity simulation

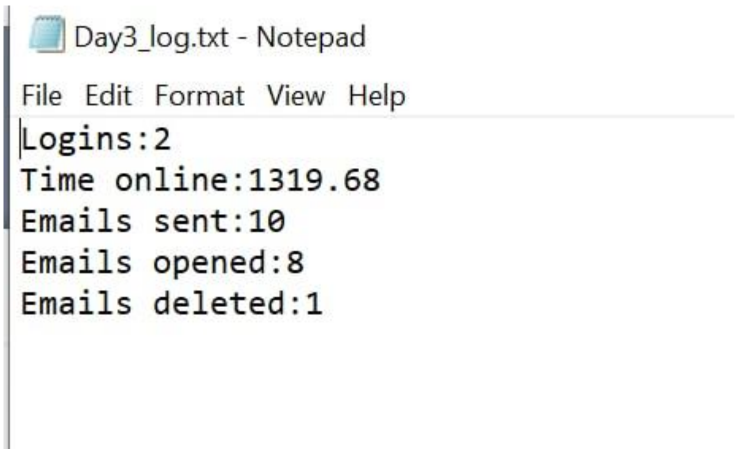
2.1 The process used to generate events

We did extensive research on activity simulation to generate random sets of data. To go about generating the data, that is similar to the baseline statistics, we needed parameters for the machine to reference.

Thus, we decided on the parameters required for our testing data were the minimum and maximum values from both Events.txt and stats.txt. By setting them, we are better able to simulate data that are more aligned to our requirements. This differs for discrete and continuous events as the generated data depends on the type of events. If the event is continuous, a figure is generated to two decimal places *(*See Figure 2*).* Once the logs have been successfully created, we store the information in arrays, like the data that was first initialized.

2.2 Name and format of the log file

As the naming and formatting of the log files requires further reading and must be human readable, we decided on a simple naming convention for the various days E.g. day1, day2 and the data in the text file would simply be Eventname:value so that it can be easily comprehended by both machine and human. We decided that such a format would be the best because it is consistent with the format of the data that has been initialized, which would allow the program to read the data easily by speeding up the reading process. This is vital as these log files create the baseline data for our Analysis Engine and Alert Engine respectively.



```
Logins:2
Time online:1319.68
Emails sent:10
Emails opened:8
Emails deleted:1
```

Figure 2: Sample data created in a text file

3. Analysis engine

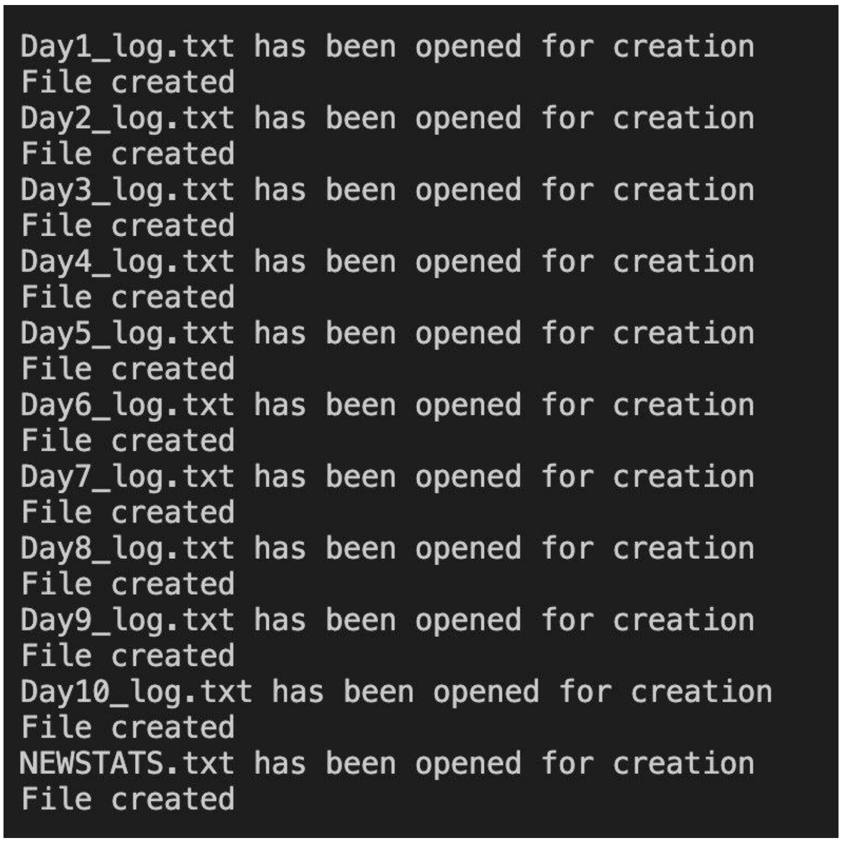
With the successfully generated data, we will begin analysis and one of the requirements included arithmetic operations which involved the calculation of the sum, the mean, and the standard deviation of each event.

In the background, the information that has been randomly generated was stored and calculated to derive the arithmetic operations accordingly.

Upon successful creation, there will be messages informing the user on the successful creation of files containing events and therefore prompt the start of analysis. The program informs the user that there has been log files with randomly generated data created (See Figure 3).

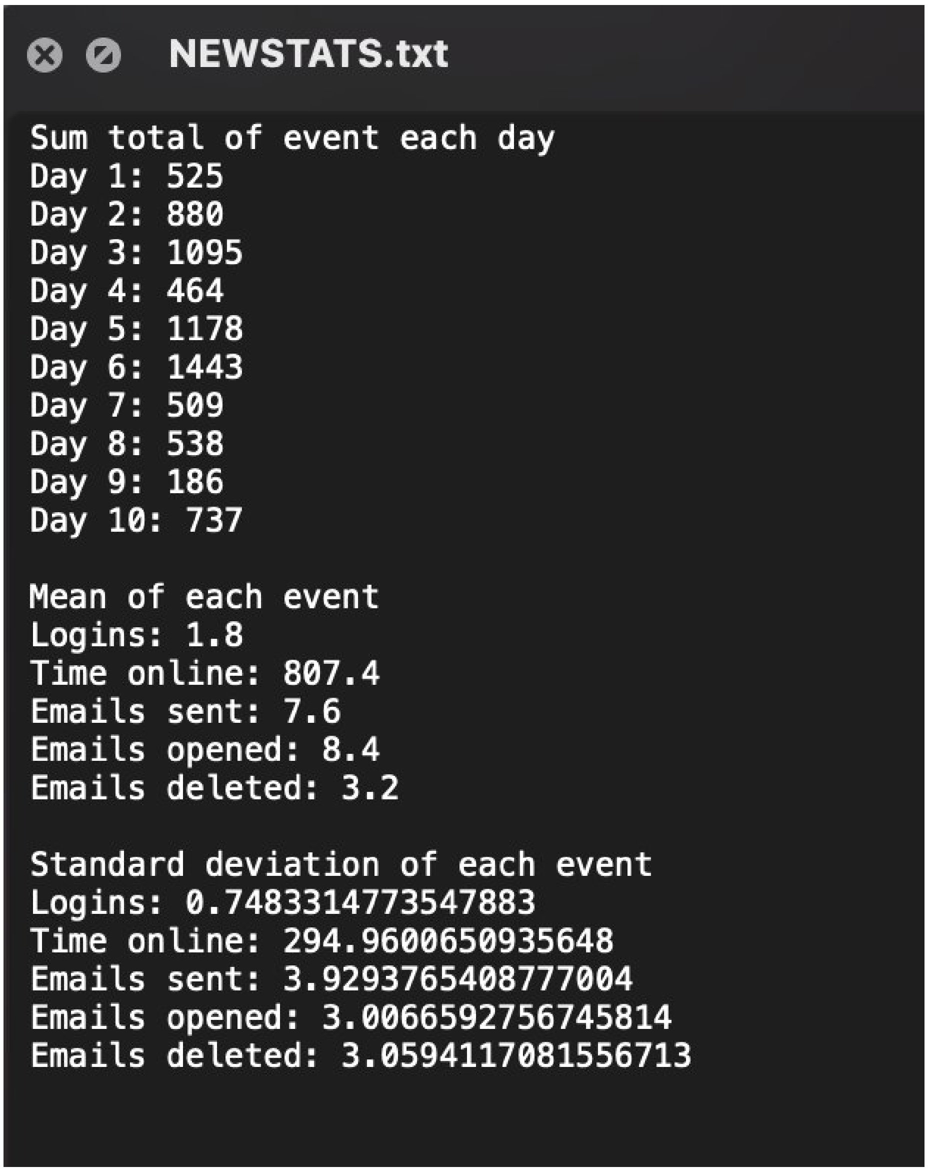
After the creation of log files, there will be a message displayed to inform the user that event generation is completed, and analysis will begin.

The total of each event, mean and standard deviation associated with that event will be stored in a text file (See Figure 4).



```
Day1_log.txt has been opened for creation
File created
Day2_log.txt has been opened for creation
File created
Day3_log.txt has been opened for creation
File created
Day4_log.txt has been opened for creation
File created
Day5_log.txt has been opened for creation
File created
Day6_log.txt has been opened for creation
File created
Day7_log.txt has been opened for creation
File created
Day8_log.txt has been opened for creation
File created
Day9_log.txt has been opened for creation
File created
Day10_log.txt has been opened for creation
File created
NEWSTATS.txt has been opened for creation
File created
```

Figure 3: Output showing successful creation of data



NEWSTATS.txt

```
Sum total of event each day
Day 1: 525
Day 2: 880
Day 3: 1095
Day 4: 464
Day 5: 1178
Day 6: 1443
Day 7: 509
Day 8: 538
Day 9: 186
Day 10: 737

Mean of each event
Logins: 1.8
Time online: 807.4
Emails sent: 7.6
Emails opened: 8.4
Emails deleted: 3.2

Standard deviation of each event
Logins: 0.7483314773547883
Time online: 294.9600650935648
Emails sent: 3.9293765408777004
Emails opened: 3.0066592756745814
Emails deleted: 3.0594117081556713
```

Figure 4: Statistical data containing arithmetic figures

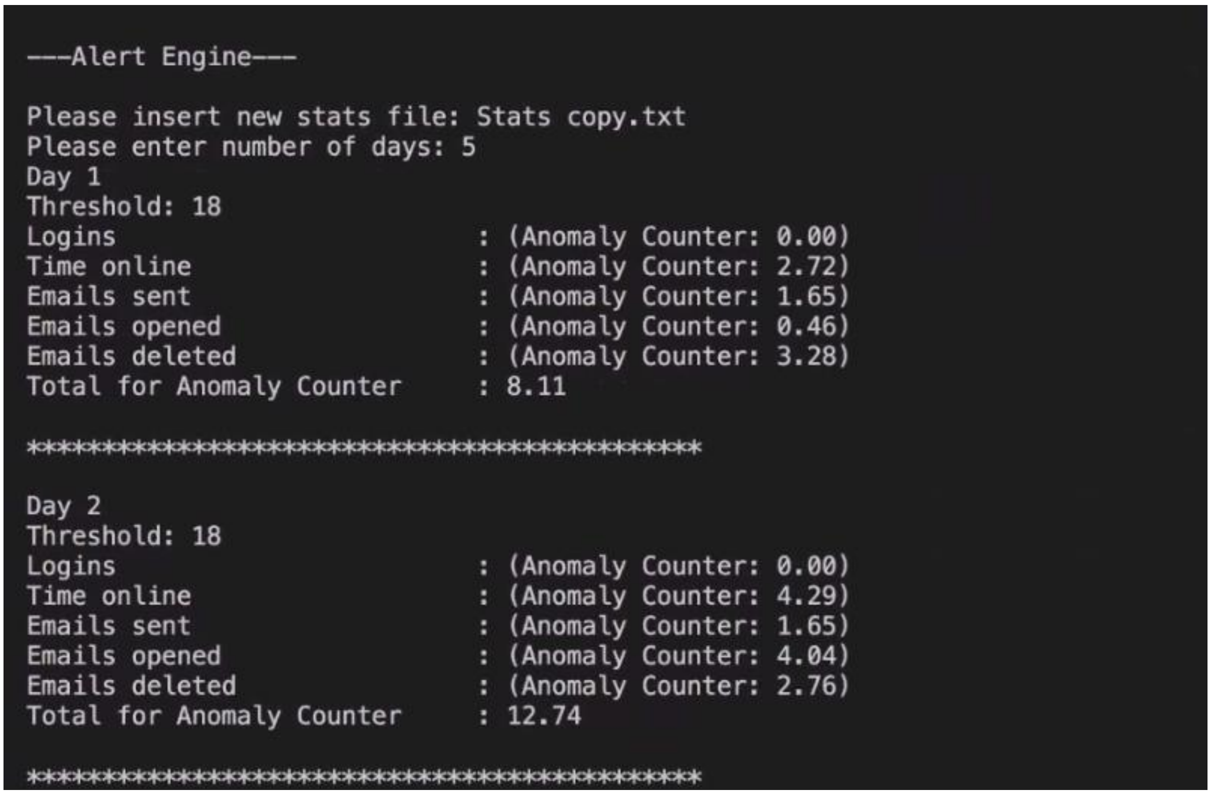
4. Alert Engine

The purpose of the Alert Engine is to check consistency between the live or given data and the baseline statistics. To process, we generated code to prompt the user for a new statistics file to be read and included basic error checking if the file cannot be read. Afterward, the user is being prompted on the number of days to be entered.

After the required information and data has been read and stored, based on the parameters passed in, the program will run the activity engine to provide the data based on the number of days input, as these values will be used to check against for alert detection.

We also did basic error validation such as checking if the number of days entered is an integer. For each day, the anomaly counter is added up as a total and compared against the threshold, in which the threshold is calculated based on the formula given.

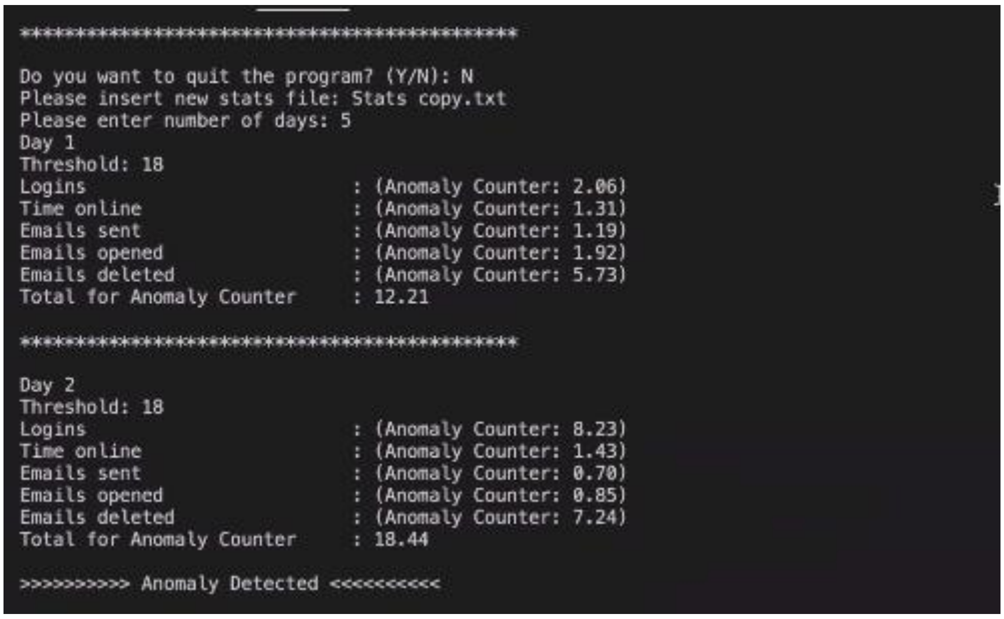
Once the calculations are done in the background, there will be an output that informs the user displayed per day if there are any abnormalities (See Figure 5).



```
---Alert Engine---

Please insert new stats file: Stats copy.txt
Please enter number of days: 5
Day 1
Threshold: 18
Logins                        : (Anomaly Counter: 0.00)
Time online                   : (Anomaly Counter: 2.72)
Emails sent                   : (Anomaly Counter: 1.65)
Emails opened                 : (Anomaly Counter: 0.46)
Emails deleted                : (Anomaly Counter: 3.28)
Total for Anomaly Counter     : 8.11

*********************************************

Day 2
Threshold: 18
Logins                        : (Anomaly Counter: 0.00)
Time online                   : (Anomaly Counter: 4.29)
Emails sent                   : (Anomaly Counter: 1.65)
Emails opened                 : (Anomaly Counter: 4.04)
Emails deleted                : (Anomaly Counter: 2.76)
Total for Anomaly Counter     : 12.74

*********************************************
```

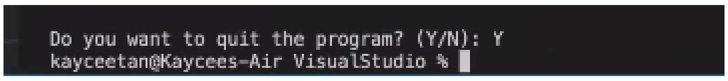
Figure 5: Output to show abnormalities

After the first run of the entire program, the user will be given an option to run the alert engine again or to quit the program. If the user enters ‘Y’, the program will prompt the user for another text file that contains another set of statistics, and the number of days to gather and generate information again *(*See Figure 6). However, if the user wishes to exit the program, he or she enters ‘N’ which would allow the user to quit the program (See Figure 7).



```
*********************************************

Do you want to quit the program? (Y/N): N
Please insert new stats file: Stats copy.txt
Please enter number of days: 5
Day 1
Threshold: 18
Logins                       : (Anomaly Counter: 2.06)
Time online                  : (Anomaly Counter: 1.31)
Emails sent                  : (Anomaly Counter: 1.19)
Emails opened                : (Anomaly Counter: 1.92)
Emails deleted               : (Anomaly Counter: 5.73)
Total for Anomaly Counter    : 12.21

*********************************************

Day 2
Threshold: 18
Logins                       : (Anomaly Counter: 8.23)
Time online                  : (Anomaly Counter: 1.43)
Emails sent                  : (Anomaly Counter: 0.70)
Emails opened                : (Anomaly Counter: 0.85)
Emails deleted               : (Anomaly Counter: 7.24)
Total for Anomaly Counter    : 18.44

>>>>>>>>>> Anomaly Detected <<<<<<<<<<
```

Figure 6: Option ‘N’: Rerun of Alert Engine



```
Do you want to quit the program? (Y/N): Y
kayceetan@Kaycees-Air VisualStudio % 
```

Figure 7: Option ‘Y’: Quit