Emoji Password Scheme

By: Bachelors in Paradise

Darren Pierre 101015833
Paradis Esmaeelzadeh Khadem 100797010
Henri Umba 101022562
Nem Zutkovic 101085982
Mohab Abdelkader 100981616

COMP 3008
April 13th, 2019

Table of Contents

1 – DESCRIPTIVE STATISTICS 3

1.1 – Exploring Password Schemes 3
1.2 – Downloading image21.csv & text21.csv 5
1.3 – Design and Implementation of a Program to Process Log Data 6
1.4 – Comparison of Descriptive Statistics: Text21 vs. Image21 14

2 – DESIGN, IMPLEMENTATION, STATISTICAL INFERENCE 19

2.1 – Design Rationale 19
2.2 – Implementation of Emoji-based Password Scheme 20
2.3 – Quantitative Testing of Emoji-based Password Scheme 23
2.4 – Questionnaire 25
2.6 – Usability Testing Comparison 27

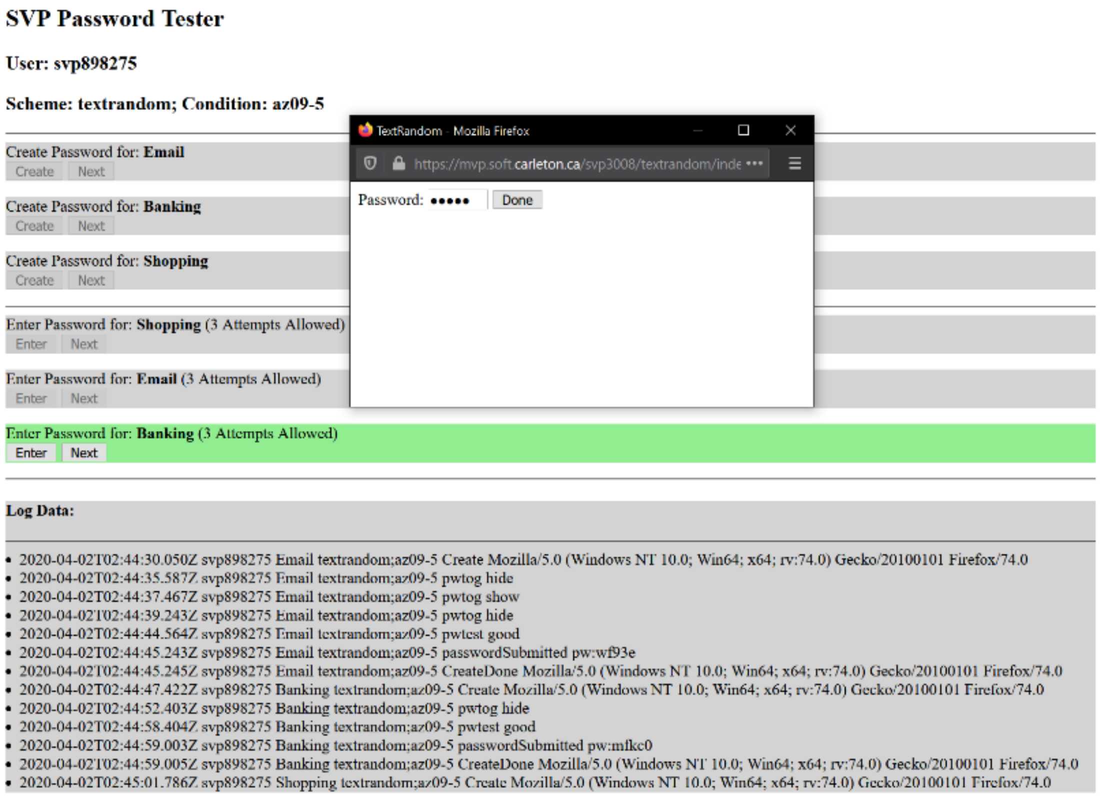
3 – WORKLOAD DISTRIBUTION AND SUMMARY 33

4 – APPENDIX 33

1 – Descriptive Statistics

1.1 – Exploring Password Schemes

For the first part of the descriptive statistics portion of this project, we looked at the advantages and disadvantages of each of the two password schemes. The first is the text-based password or *textrandom* and the second is the graphical password, otherwise known as *passtiles*. Both password schemes can be found in the Password Tester online.



SVP Password Tester

User: svp898275

Scheme: textrandom; Condition: az09-5

Create Password for: **Email**

Create Password for: **Banking**

Create Password for: **Shopping**

Enter Password for: **Shopping** (3 Attempts Allowed)

Enter Password for: **Email** (3 Attempts Allowed)

Enter Password for: **Banking** (3 Attempts Allowed)

TextRandom - Mozilla Firefox

https://mvp.soft.carleton.ca/svp3008/textrandom/inde

Password: ●●●●●

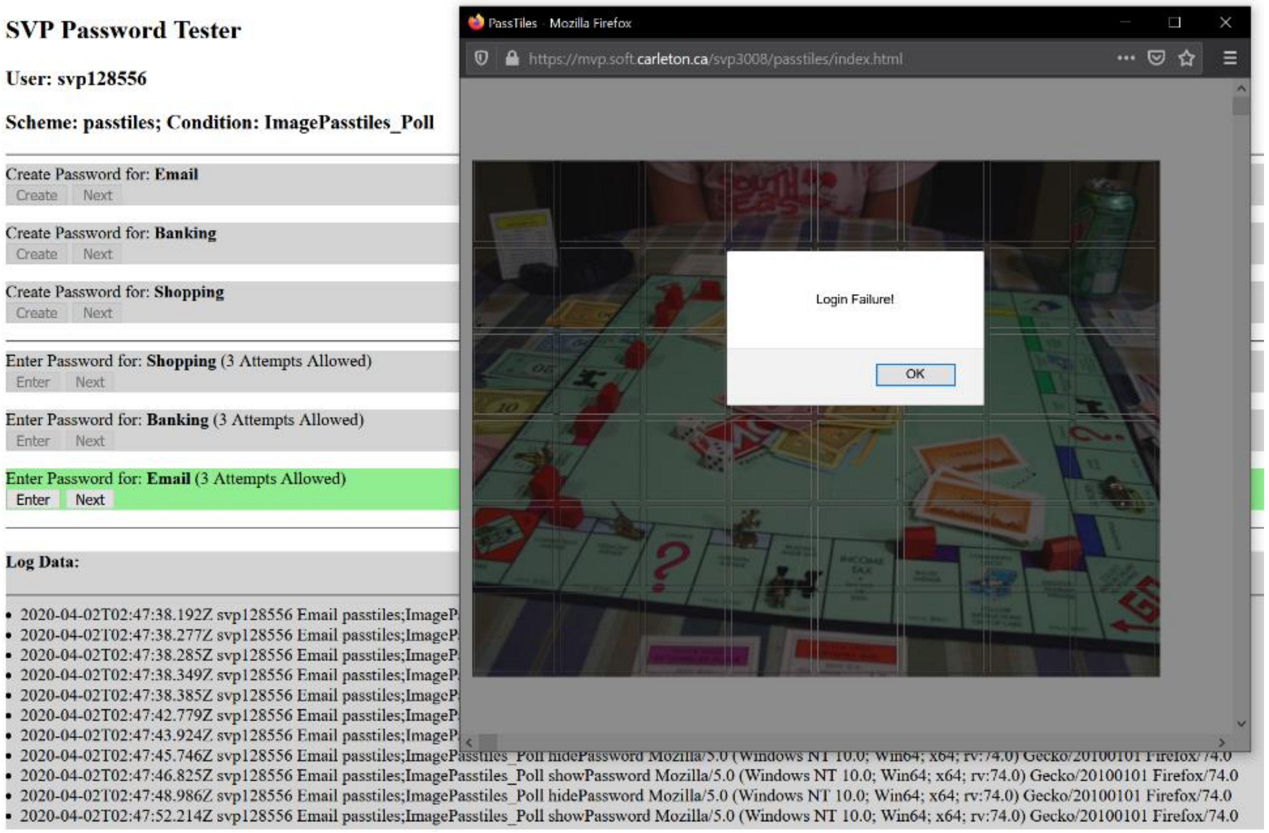
Log Data:

```
• 2020-04-02T02:44:30.050Z svp898275 Email textrandom;az09-5 Create Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0
• 2020-04-02T02:44:35.587Z svp898275 Email textrandom;az09-5 pwtog hide
• 2020-04-02T02:44:37.467Z svp898275 Email textrandom;az09-5 pwtog show
• 2020-04-02T02:44:39.243Z svp898275 Email textrandom;az09-5 pwtog hide
• 2020-04-02T02:44:44.564Z svp898275 Email textrandom;az09-5 pwtest good
• 2020-04-02T02:44:45.243Z svp898275 Email textrandom;az09-5 passwordSubmitted pw:wf93e
• 2020-04-02T02:44:45.245Z svp898275 Email textrandom;az09-5 CreateDone Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0
• 2020-04-02T02:44:47.422Z svp898275 Banking textrandom;az09-5 Create Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0
• 2020-04-02T02:44:52.403Z svp898275 Banking textrandom;az09-5 pwtog hide
• 2020-04-02T02:44:58.404Z svp898275 Banking textrandom;az09-5 pwtest good
• 2020-04-02T02:44:59.003Z svp898275 Banking textrandom;az09-5 passwordSubmitted pw:mfkc0
• 2020-04-02T02:44:59.005Z svp898275 Banking textrandom;az09-5 CreateDone Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0
• 2020-04-02T02:45:01.786Z svp898275 Shopping textrandom;az09-5 Create Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0
```

The text-based password is a very classical approach to user security and does have its merits. Text-based passwords offer a high level of security if a password is randomly generated from a large set of characters. *textrandom* does not offer the highest level of security despite it being a text-based password because upper case letters and symbols have been omitted. Also, the password length is 5 characters long which is fairly short in today's standards for password security. Despite this lack of character variability, the passwords are still hard to remember because they are randomly generated. On top of this, the user was given 3 randomly generated passwords to remember subsequently, that were each 5 characters long. That is essentially equivalent to giving the user a 15-character password to remember in one sitting. From our lecture notes and recent studies, it has been shown that users can typically remember between 5 to 9 items in their working memory. This means that the memorization of text-based passwords would be temporary and not remain in someone's long term memory. The one advantage of text-based passwords such as those offered in *textrandom* is that login times were faster because the user just had to enter a few strokes compared to guessing tiles.

Looking at graphical passwords or passtiles, it can be said they offer far superior memorability because there is a visual cue for the user. As we have learned, human memory can identify and memorize images better compared to text. The text does not stimulate the brain, whereas images do and this is one of the advantages of graphical passwords.

A disadvantage to graphical passwords, at least with *passtiles* in particular, is that it was less secure than the *textrandom* password scheme. This is because there were only 48 tiles per password and you cannot revisit tiles once selected, but the *textrandom* passwords had 26 lower-case letters and 10 digits, meaning 36 characters that could be repeated. The *textrandom* can generate about 40 million or more passwords, whereas in the graphical interface you just need to select 6 out of the 48 tiles. This is significantly less secure, to the point where usability might not matter because of the weak security. The second *passtiles* test, on the other hand, provides a twist where each tile needs to be clicked on in a certain order. This adds a significant level of complexity to the password scheme because the user needs to guess the tiles and the orders in which they are to be clicked. This is much better for security than the previous graphical password method, yet still not as secure as the text-based password approach. Another disadvantage we noted with graphical passwords is that the login times were a bit slower compared to text-based passwords, at least with the initial tests. We would suspect that if the graphical pattern were memorized over a longer period (i.e. not just in working memory), but the login times might be faster.


SVP Password Tester

User: svp128556

Scheme: passtiles; Condition: ImagePasstiles_Poll

Create Password for: **Email**

Create Password for: **Banking**

Create Password for: **Shopping**

Enter Password for: **Shopping** (3 Attempts Allowed)

Enter Password for: **Banking** (3 Attempts Allowed)

Enter Password for: **Email** (3 Attempts Allowed)

Log Data:

```
• 2020-04-02T02:47:38.192Z svp128556 Email passtiles;ImageP
• 2020-04-02T02:47:38.277Z svp128556 Email passtiles;ImageP
• 2020-04-02T02:47:38.285Z svp128556 Email passtiles;ImageP
• 2020-04-02T02:47:38.349Z svp128556 Email passtiles;ImageP
• 2020-04-02T02:47:38.385Z svp128556 Email passtiles;ImageP
• 2020-04-02T02:47:42.779Z svp128556 Email passtiles;ImageP
• 2020-04-02T02:47:43.924Z svp128556 Email passtiles;ImageP
• 2020-04-02T02:47:45.746Z svp128556 Email passtiles;ImagePasstiles_Poll hidePassword Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0
• 2020-04-02T02:47:46.825Z svp128556 Email passtiles;ImagePasstiles_Poll showPassword Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0
• 2020-04-02T02:47:48.986Z svp128556 Email passtiles;ImagePasstiles_Poll hidePassword Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0
• 2020-04-02T02:47:52.214Z svp128556 Email passtiles;ImagePasstiles_Poll showPassword Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0
```

PassTiles — Mozilla Firefox
https://mvp.soft.carleton.ca/svp3008/passtiles/index.html
Create
Next
Enter
Login Failure!
OK

In conclusion, graphical passwords are slower for users to adopt, but could be just as fast as text-based based passwords given enough time for the user to memorize the pattern. Graphical passwords also seem to have poor security but increasing the number of tiles and providing a sense of order with which tiles to select first could open-up the possibility for higher levels of security. Lastly, graphical passwords are easier to remember than text-based passwords simply by the nature of the human brain. With all this in mind, there is much to be said about the future of security and password authentication. Graphical passwords are certainly making some noise in the field of security and might overtake text-based passwords in some scenarios.

1.2 – Downloading image21.csv & text21.csv

No writeup required for this section

.

1.3 – Design and Implementation of a Program to Process Log Data

In this section of the report, we will be discussing the design and implementation of the log data processing program. The goal of this program was to process data from a completed user study into a state where it can be analyzed. The result of the data includes the following information:

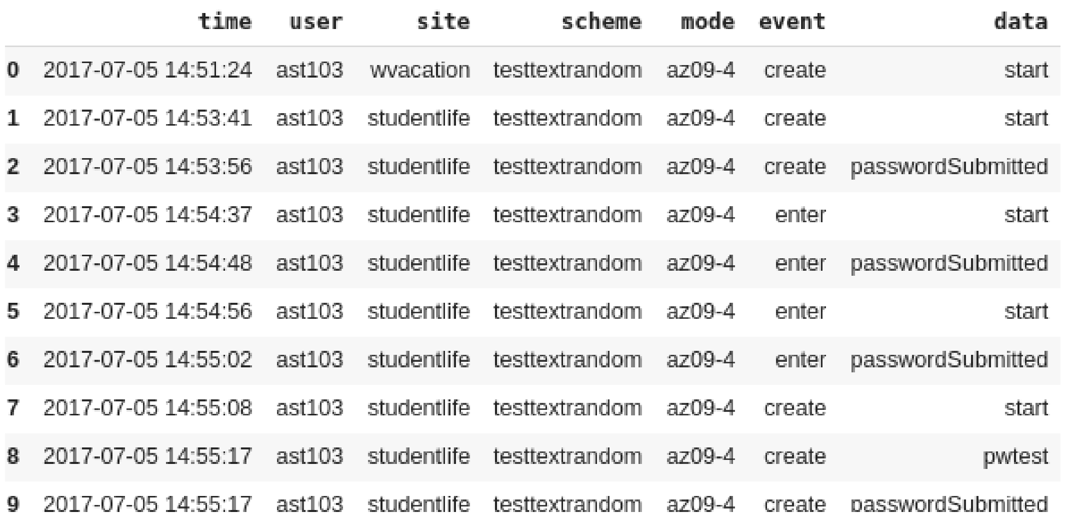
- The user-id
- The password schemes
- The number of logins,
- The number of successful logins
- The number of failed logins.
- The average time taken to enter a password for successful logins,
- The average time taken to enter a password for failed logins

We will first be discussing the dataset of the user study and relevant considerations that were kept in mind when developing the log data processing software. After we will be discussing a high-level explanation of how the log processing program works and provide the documented source code.

To process the data from the user study, we first needed to inspect and understand the datasets. The datasets come from a user study from two password schemes. One password scheme was a graphical password scheme named Image21. The other password scheme is a typical alphanumeric password scheme and is named Text21. The datasets had the columns representing the following fields:

- time: representing a timestamp of a user event
- user-id: a unique identifier for a user
- site: website identifier
- scheme: password scheme/sub-scheme identifier
- mode: usage mode identifier
- event: event identifier
- data: data associated with the password event

An example of how the dataset looks like is provided below in Figure 1.



	time	user	site	scheme	mode	event	data
0	2017-07-05 14:51:24	ast103	wvacation	testtextrandom	az09-4	create	start
1	2017-07-05 14:53:41	ast103	studentlife	testtextrandom	az09-4	create	start
2	2017-07-05 14:53:56	ast103	studentlife	testtextrandom	az09-4	create	passwordSubmitted
3	2017-07-05 14:54:37	ast103	studentlife	testtextrandom	az09-4	enter	start
4	2017-07-05 14:54:48	ast103	studentlife	testtextrandom	az09-4	enter	passwordSubmitted
5	2017-07-05 14:54:56	ast103	studentlife	testtextrandom	az09-4	enter	start
6	2017-07-05 14:55:02	ast103	studentlife	testtextrandom	az09-4	enter	passwordSubmitted
7	2017-07-05 14:55:08	ast103	studentlife	testtextrandom	az09-4	create	start
8	2017-07-05 14:55:17	ast103	studentlife	testtextrandom	az09-4	create	pwtest
9	2017-07-05 14:55:17	ast103	studentlife	testtextrandom	az09-4	create	passwordSubmitted

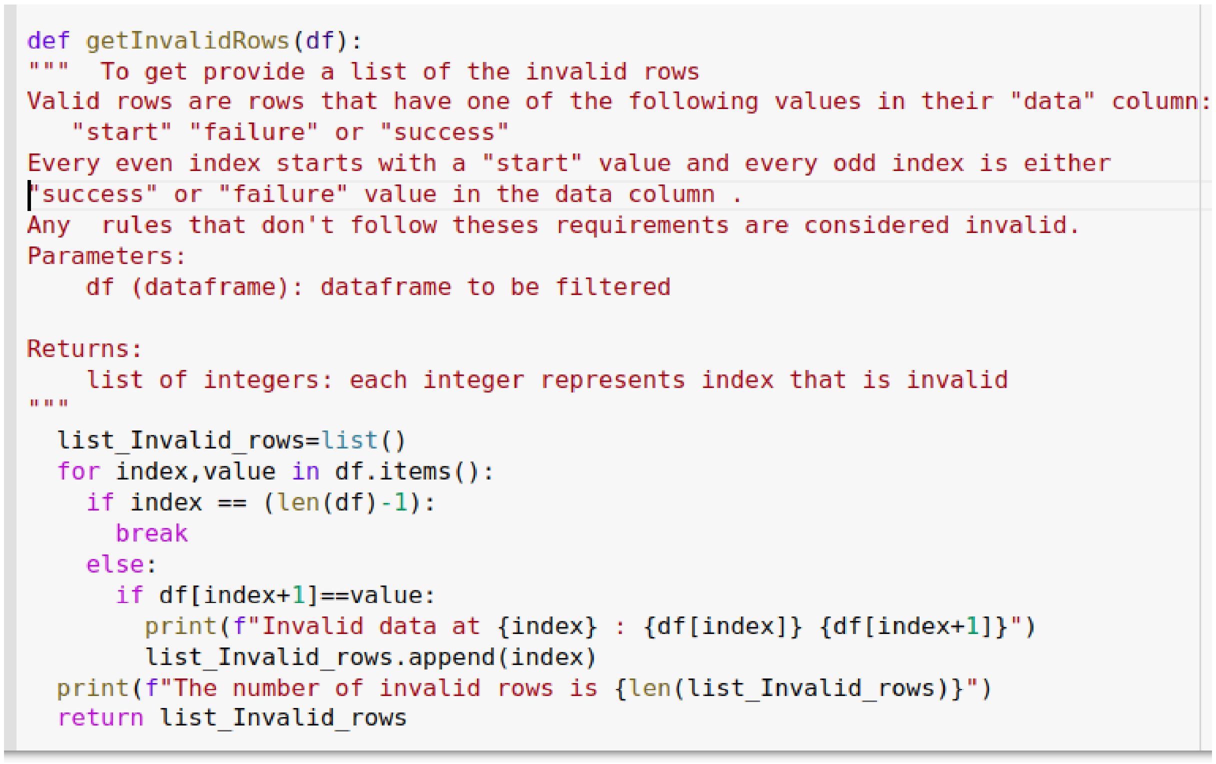
Figure 1: *First 10 rows of the Text21 raw dataset*

To process the data in the specified format we first needed to clean the data. Data cleaning is the process of detecting and removing irrelevant parts of data. From inspecting each of the datasets, there were no empty or missing values that we needed to handle. The datasets had some irrelevant rows for our task and not ready to compute the time taken for a login attempt.

Given the two issues previously mentioned, we first sought to remove any irrelevant rows that were not related to a user login time or login attempts. After that, further cleaned the data where we could compute the time taken for a login attempt. The data will be transformed into a form where every 2 rows the first row represent the start of a login attempt and the second row represents the end of a login attempt. Below shows the pseudocode and source code.

PseudoCode: Calculating the indexes of irrelevant rows

```
Function InvalidRow(data frame):
        Initialize list of invalid row indexes to an empty list
        Initialize data column to a list of the data frame data column
 For every value in the data frame data column:
                If the current value is equal to the element after the current value:
                        add the index of the current value to the list of invalid row indexes
        Return list of invalid row indexes
```

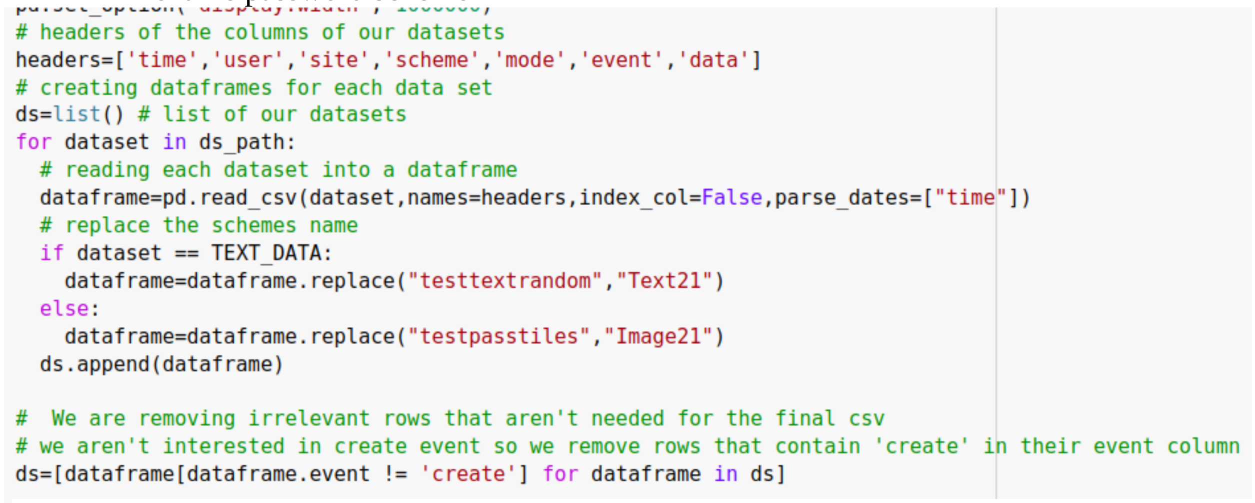


```
def getInvalidRows(df):
"""  To get provide a list of the invalid rows
Valid rows are rows that have one of the following values in their "data" column:
   "start" "failure" or "success"
Every even index starts with a "start" value and every odd index is either
"success" or "failure" value in the data column .
Any  rules that don't follow theses requirements are considered invalid.
Parameters:
    df (dataframe): dataframe to be filtered

Returns:
    list of integers: each integer represents index that is invalid
"""
  list_Invalid_rows=list()
  for index,value in df.items():
    if index == (len(df)-1):
      break
    else:
      if df[index+1]==value:
        print(f"Invalid data at {index} : {df[index]} {df[index+1]}")
        list_Invalid_rows.append(index)
  print(f"The number of invalid rows is {len(list_Invalid_rows)}")
  return list_Invalid_rows
```

Figure 2: *Source code for the function invalid rows*

Pseudocode: Creating data frames for our datasets and removing irrelevant rows

```
Initialize list of data frames to an empty list
For each data file:
  Create a data frame of the datafile
  Add the data frame to the list of data frames
For each data frame in the list of data frames:
        Remove rows that have a 'create' in their event column
        Rename password scheme
```



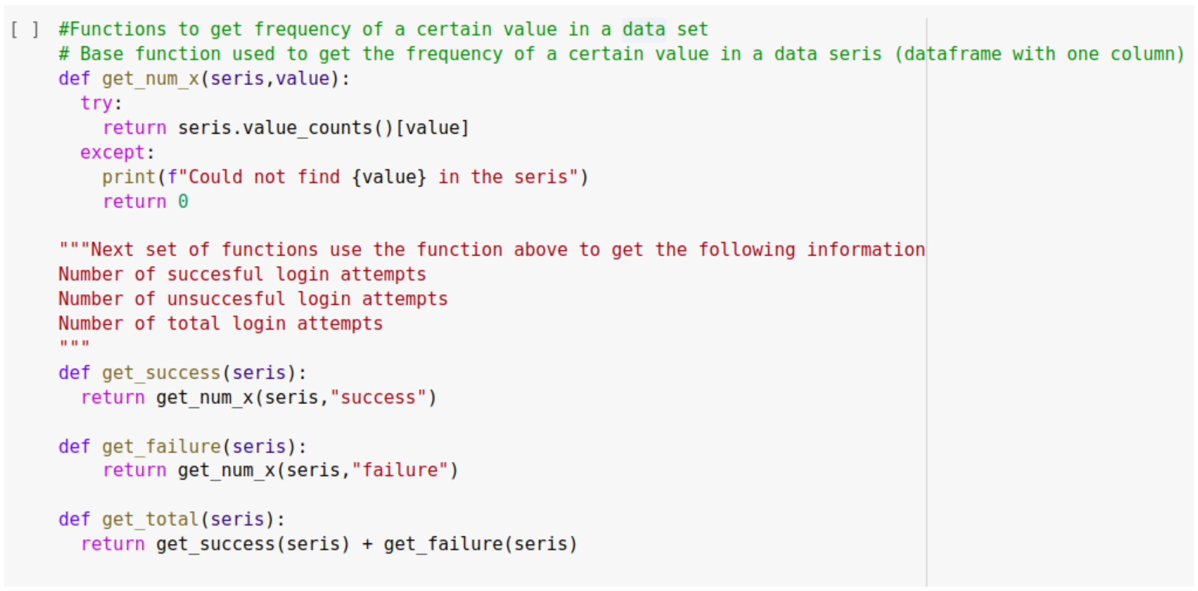
```
# headers of the columns of our datasets
headers=['time','user','site','scheme','mode','event','data']
# creating dataframes for each data set
ds=list() # list of our datasets
for dataset in ds_path:
  # reading each dataset into a dataframe
  dataframe=pd.read_csv(dataset,names=headers,index_col=False,parse_dates=["time"])
  # replace the schemes name
  if dataset == TEXT_DATA:
    dataframe=dataframe.replace("testtextrandom","Text21")
  else:
    dataframe=dataframe.replace("testpasstiles","Image21")
  ds.append(dataframe)

#  We are removing irrelevant rows that aren't needed for the final csv
# we aren't interested in create event so we remove rows that contain 'create' in their event column
ds=[dataframe[dataframe.event != 'create'] for dataframe in ds]
```

Figure 3: *Source code for creating data frames of our datasets and removing irrelevant rows*

After the first iteration of data cleaning, we needed to create aggregate functions that computed the values needed for the final CSV file. The following aggregate functions were created:

- Computing the total number of login attempts
- Computing the number of successful login attempts
- Computing the number of unsuccessful login attempts
- The average time for a successful login attempt
- The average time for an unsuccessful login attempt
- The average time for a login attempt

We do not provide the pseudocode for some aggregate functions because it is similar to the source code syntax.

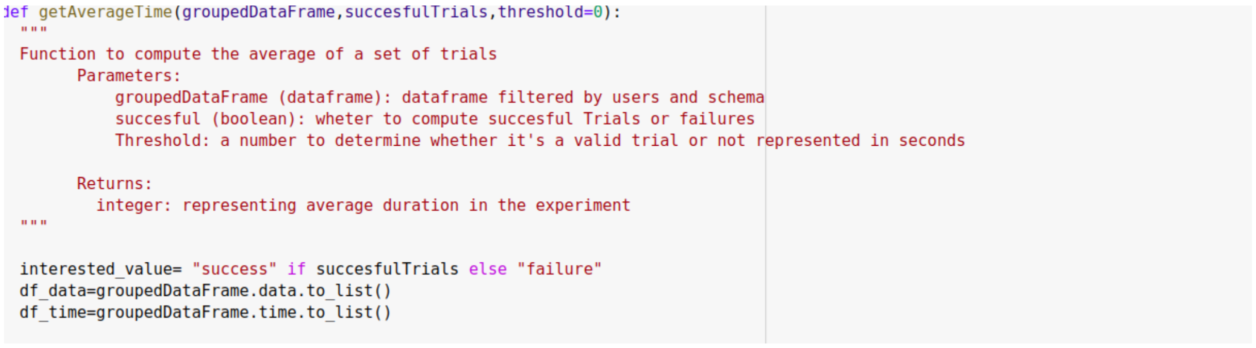


```
[ ]  #Functions to get frequency of a certain value in a data set
     # Base function used to get the frequency of a certain value in a data seris (dataframe with one column)
     def get_num_x(seris,value):
       try:
         return seris.value_counts()[value]
       except:
         print(f"Could not find {value} in the seris")
         return 0

     """Next set of functions use the function above to get the following information
     Number of succesful login attempts
     Number of unsuccessful login attempts
     Number of total login attempts
     """
     def get_success(seris):
       return get_num_x(seris,"success")

     def get_failure(seris):
         return get_num_x(seris,"failure")

     def get_total(seris):
       return get_success(seris) + get_failure(seris)
```

Figure 4: *Source code of simple aggregate functions*

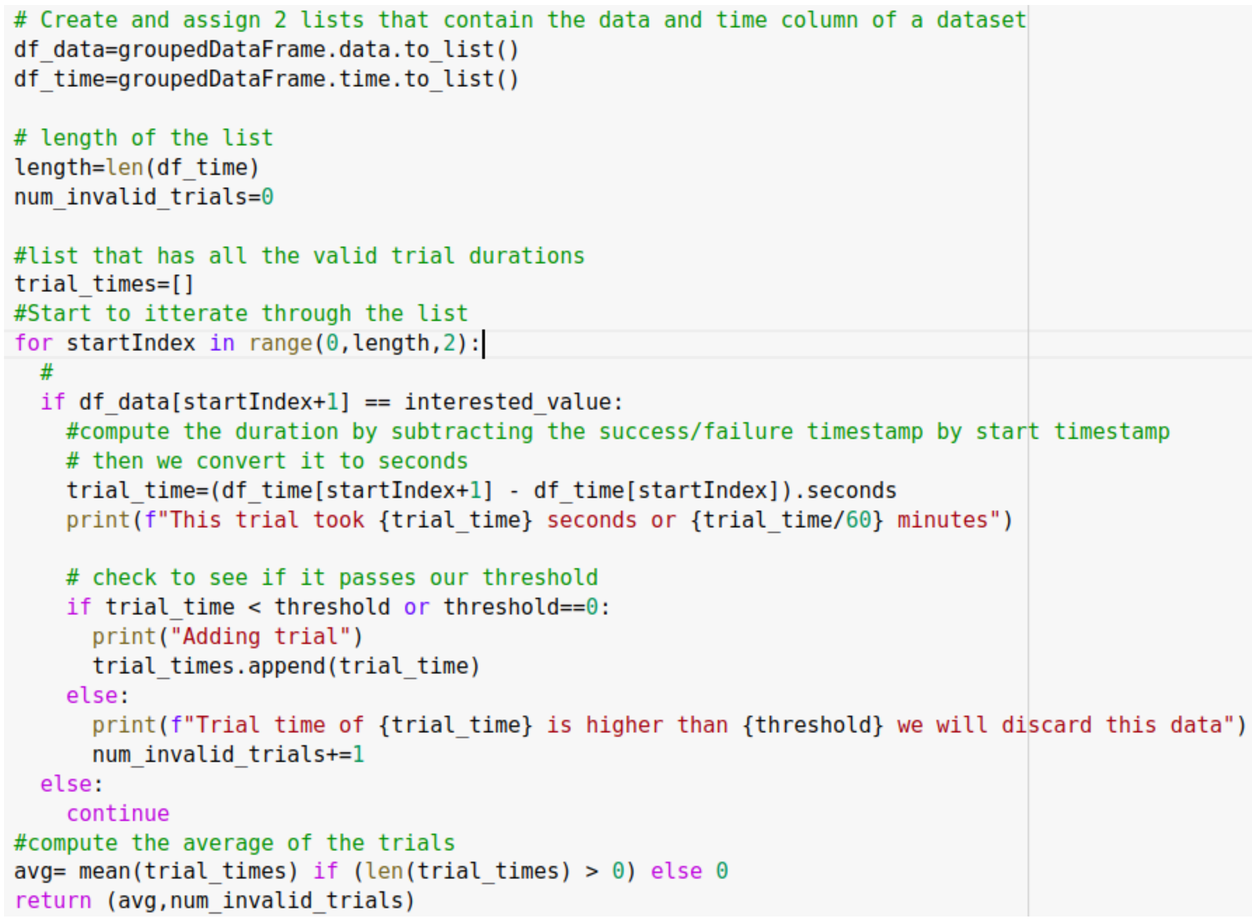
Pseudocode: Computing the average time for successful and unsuccessful trials

```
Function computeAverageTime(data frame , successful, threshold):
        Initialize list of trial times to an empty list
        Initialize number of invalid data entries to 0
        For every 2 rows  in the time column of the data frame:
            Initialize time taken to the difference of the following time value with the current time value
            If time_taken is less than threshold:
                Add it to trial times
            Else:
                Increment number of invalid data entries by 1

            Return average(trial_times) , numberof invalid
```



```
def getAverageTime(groupedDataFrame,succesfulTrials,threshold=0):
  """
  Function to compute the average of a set of trials
        Parameters:
            groupedDataFrame (dataframe): dataframe filtered by users and schema
            succesful (boolean): wheter to compute succesful Trials or failures
            Threshold: a number to determine whether it's a valid trial or not represented in seconds

        Returns:
          integer: representing average duration in the experiment
  """

  interested_value= "success" if succesfulTrials else "failure"
  df_data=groupedDataFrame.data.to_list()
  df_time=groupedDataFrame.time.to_list()
```

Figure 5: Source code of computing average time for successful or unsuccessful login attempts

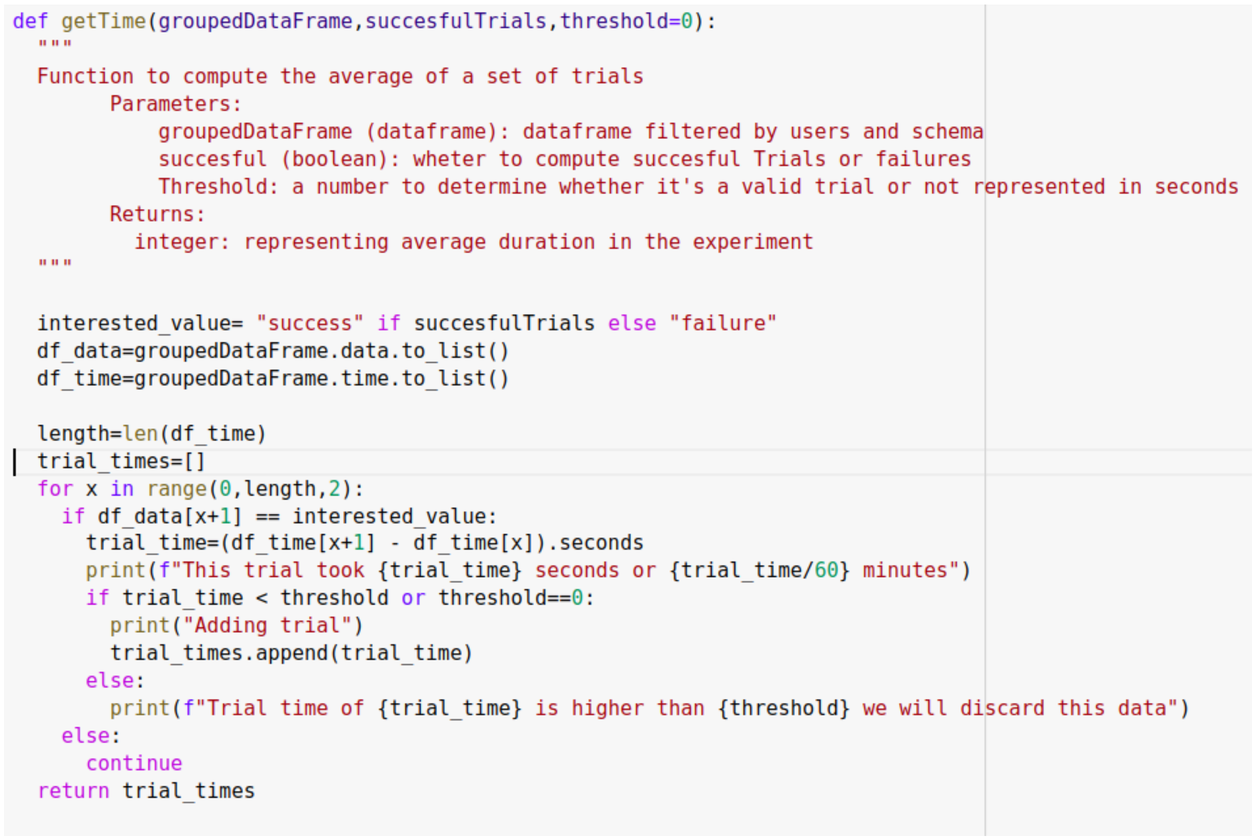


```
# Create and assign 2 lists that contain the data and time column of a dataset
df_data=groupedDataFrame.data.to_list()
df_time=groupedDataFrame.time.to_list()

# length of the list
length=len(df_time)
num_invalid_trials=0

#list that has all the valid trial durations
trial_times=[]
#Start to itterate through the list
for startIndex in range(0,length,2):
  #
  if df_data[startIndex+1] == interested_value:
    #compute the duration by subtracting the success/failure timestamp by start timestamp
    # then we convert it to seconds
    trial_time=(df_time[startIndex+1] - df_time[startIndex]).seconds
    print(f"This trial took {trial_time} seconds or {trial_time/60} minutes")

    # check to see if it passes our threshold
    if trial_time < threshold or threshold==0:
      print("Adding trial")
      trial_times.append(trial_time)
    else:
      print(f"Trial time of {trial_time} is higher than {threshold} we will discard this data")
      num_invalid_trials+=1
  else:
    continue
#compute the average of the trials
avg= mean(trial_times) if (len(trial_times) > 0) else 0
return (avg,num_invalid_trials)
```

Figure 6: Source code of computing average time for successful or unsuccessful login attempts

We are using a threshold value to filter out any invalid data entries for our time-related columns. We used the 78th percentile of average time taken for successful and unsuccessful login attempts as our threshold. Anything higher than the threshold will be considered invalid data and will not be used in the computation. We chose to use the 78th percentile because it was the value where it seems like a time where most users will complete the duration.

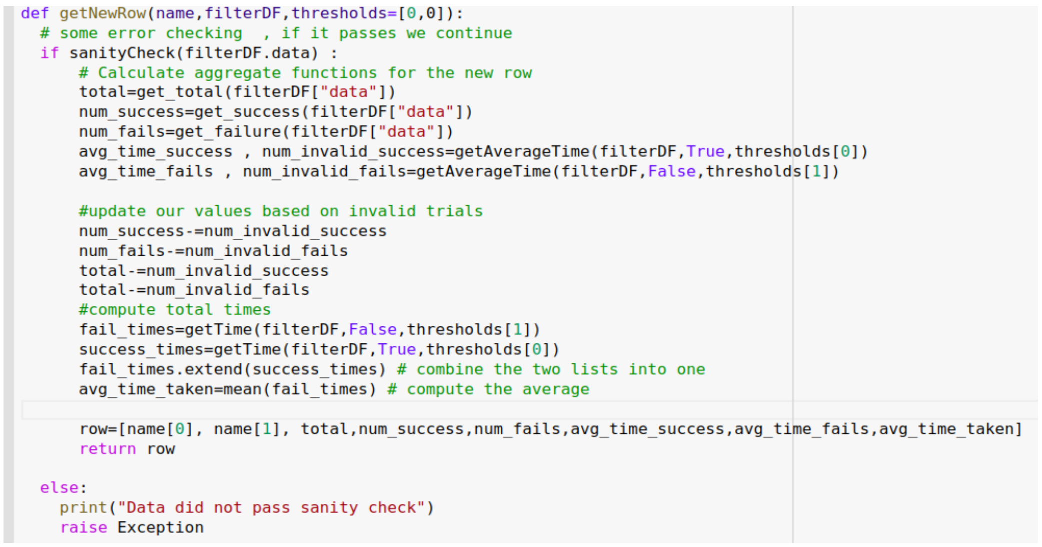
We created a function that is like the previous function to return trial times that will be used to compute the average time for a login attempt.



```python
def getTime(groupedDataFrame,succesfulTrials,threshold=0):
  """
  Function to compute the average of a set of trials
        Parameters:
            groupedDataFrame (dataframe): dataframe filtered by users and schema
            succesful (boolean): wheter to compute succesful Trials or failures
            Threshold: a number to determine whether it's a valid trial or not represented in seconds
        Returns:
          integer: representing average duration in the experiment
  """

  interested_value= "success" if succesfulTrials else "failure"
  df_data=groupedDataFrame.data.to_list()
  df_time=groupedDataFrame.time.to_list()

  length=len(df_time)
  trial_times=[]
  for x in range(0,length,2):
    if df_data[x+1] == interested_value:
      trial_time=(df_time[x+1] - df_time[x]).seconds
      print(f"This trial took {trial_time} seconds or {trial_time/60} minutes")
      if trial_time < threshold or threshold==0:
        print("Adding trial")
        trial_times.append(trial_time)
      else:
        print(f"Trial time of {trial_time} is higher than {threshold} we will discard this data")
    else:
      continue
  return trial_times
```

Figure 7: *Source code of the function to get trial times used to compute the average time*

After we were done creating aggregate functions for our program. We were ready to start processing the data for the required task. We first needed to split the data into groups based on user id and scheme. For each group, we compute our required information. After computing the required information for a user, we add the data to our final CSV file. Figure 7 depicts this process.



```
def getNewRow(name,filterDF,thresholds=[0,0]):
  # some error checking  , if it passes we continue
  if sanityCheck(filterDF.data) :
      # Calculate aggregate functions for the new row
      total=get_total(filterDF["data"])
      num_success=get_success(filterDF["data"])
      num_fails=get_failure(filterDF["data"])
      avg_time_success , num_invalid_success=getAverageTime(filterDF,True,thresholds[0])
      avg_time_fails , num_invalid_fails=getAverageTime(filterDF,False,thresholds[1])

      #update our values based on invalid trials
      num_success-=num_invalid_success
      num_fails-=num_invalid_fails
      total-=num_invalid_success
      total-=num_invalid_fails
      #compute total times
      fail_times=getTime(filterDF,False,thresholds[1])
      success_times=getTime(filterDF,True,thresholds[0])
      fail_times.extend(success_times) # combine the two lists into one
      avg_time_taken=mean(fail_times) # compute the average

      row=[name[0], name[1], total,num_success,num_fails,avg_time_success,avg_time_fails,avg_time_taken]
      return row

  else:
    print("Data did not pass sanity check")
    raise Exception
```

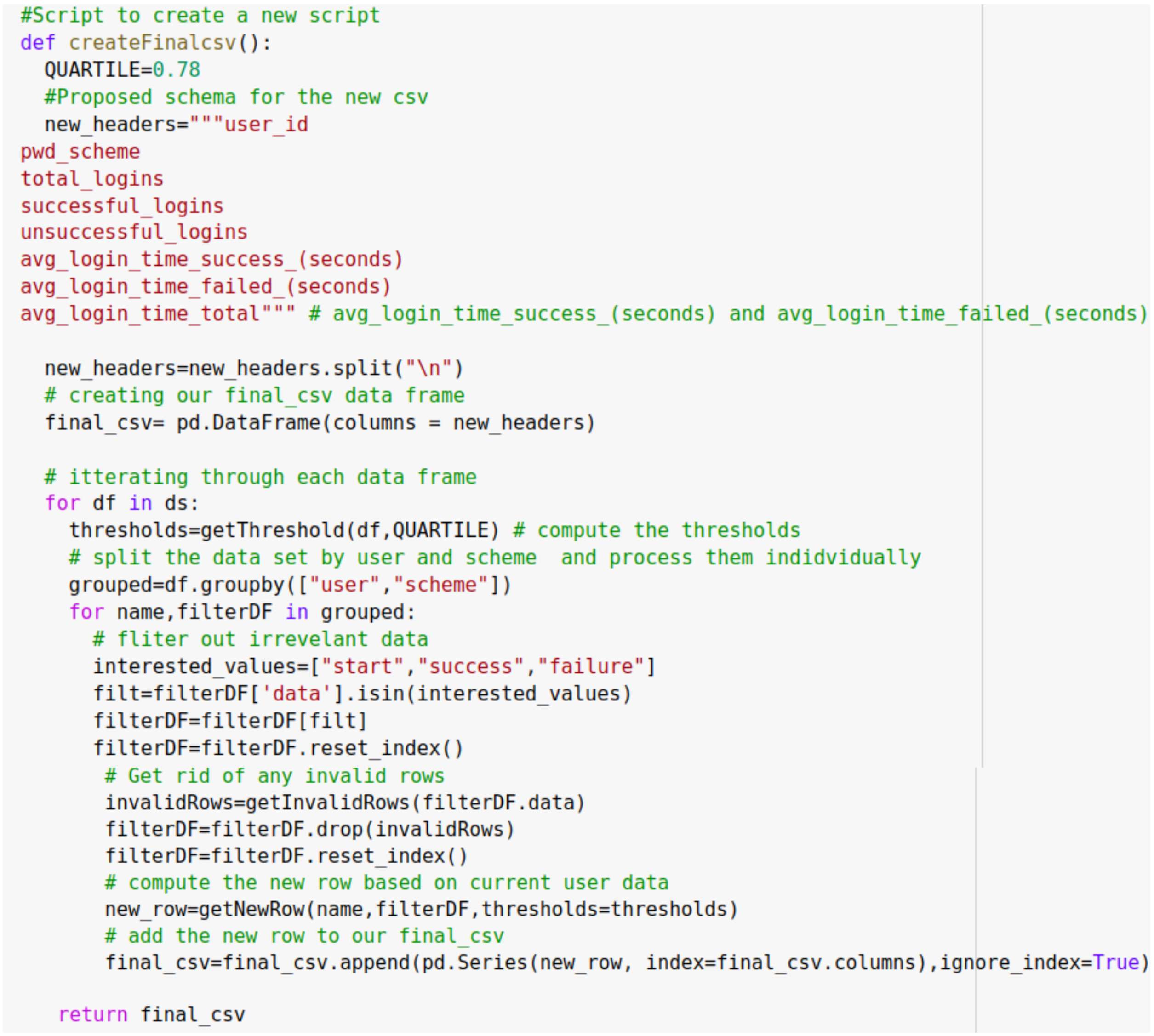
Figure 8: *Source code to compute the required information for a user*

Last step was to combine all the parts into one main program and our task of creating a log processing software will be done.

Pseudocode for the main function:

```
Create a data frame with the right headers called finalCsv
For every dataset :
        Compute the threshold:
        Split the dataset by users.
        For every user:
                Compute the invalid rows And filter the data
 We calculate values needed for our final CSV
```

Then we add that new set of values as a new row to our final CSV.

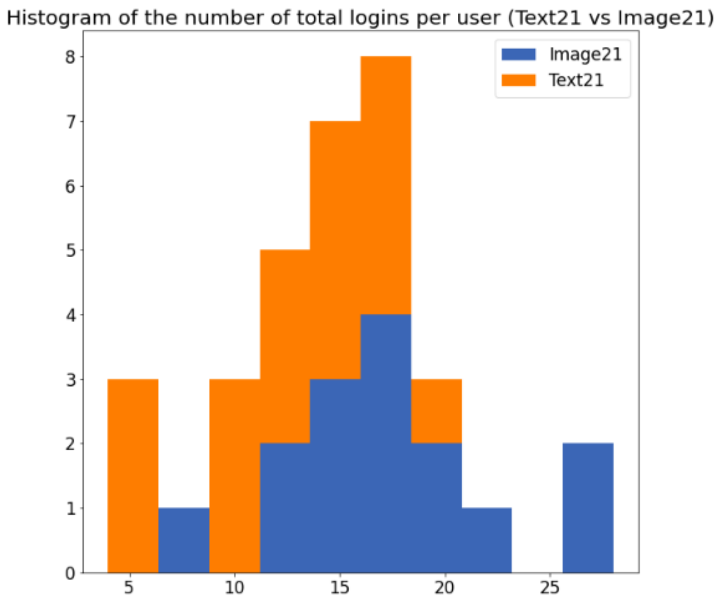


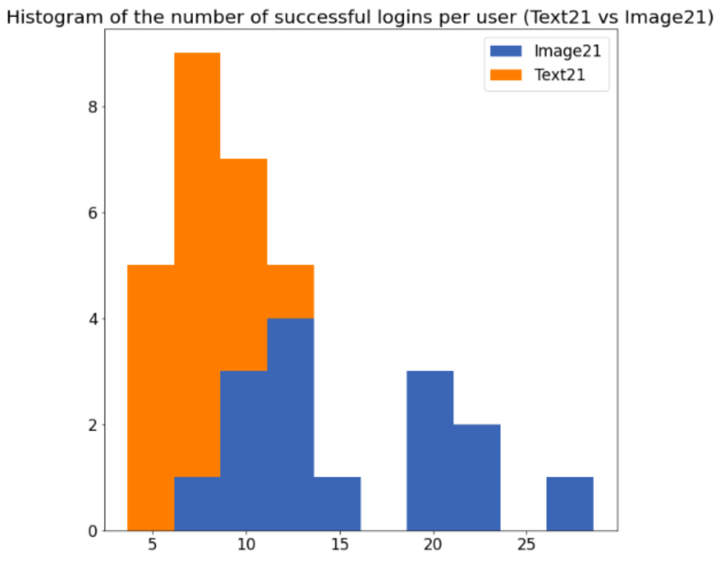
```
#Script to create a new script
def createFinalcsv():
  QUARTILE=0.78
  #Proposed schema for the new csv
  new_headers="""user_id
pwd_scheme
total_logins
successful_logins
unsuccessful_logins
avg_login_time_success_(seconds)
avg_login_time_failed_(seconds)
avg_login_time_total""" # avg_login_time_success_(seconds) and avg_login_time_failed_(seconds)

  new_headers=new_headers.split("\n")
  # creating our final_csv data frame
  final_csv= pd.DataFrame(columns = new_headers)

  # itterating through each data frame
  for df in ds:
    thresholds=getThreshold(df,QUARTILE) # compute the thresholds
    # split the data set by user and scheme  and process them indidvidually
    grouped=df.groupby(["user","scheme"])
    for name,filterDF in grouped:
      # fliter out irrevelant data
      interested_values=["start","success","failure"]
      filt=filterDF['data'].isin(interested_values)
      filterDF=filterDF[filt]
      filterDF=filterDF.reset_index()
       # Get rid of any invalid rows
       invalidRows=getInvalidRows(filterDF.data)
       filterDF=filterDF.drop(invalidRows)
       filterDF=filterDF.reset_index()
       # compute the new row based on current user data
       new_row=getNewRow(name,filterDF,thresholds=thresholds)
       # add the new row to our final_csv
       final_csv=final_csv.append(pd.Series(new_row, index=final_csv.columns),ignore_index=True)

   return final_csv
```

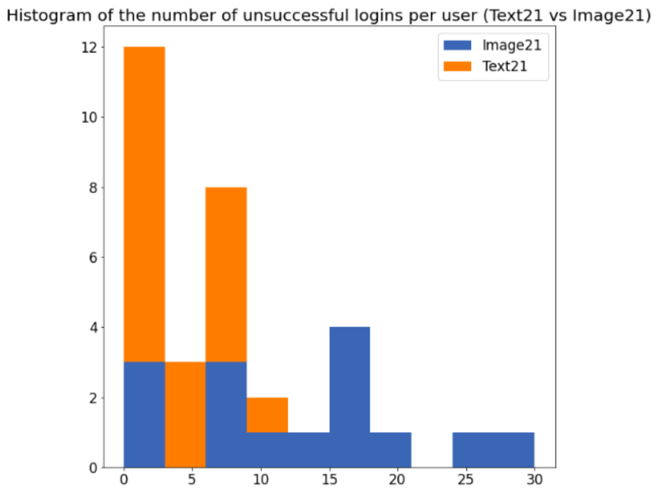
Figure 9*: Source code of the main program of log processing software*

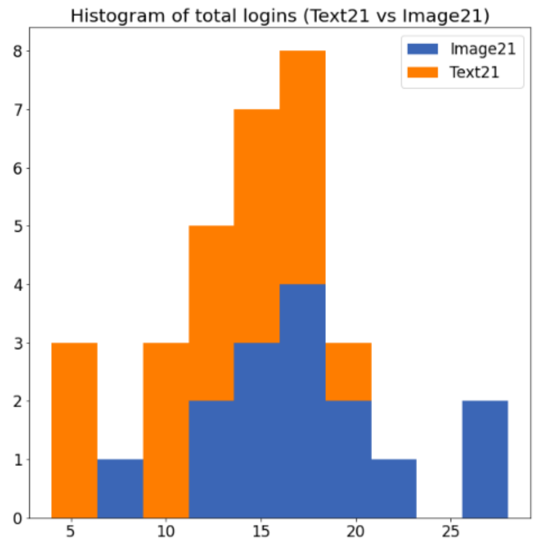
1.4 – Comparison of Descriptive Statistics: Text21 vs. Image21

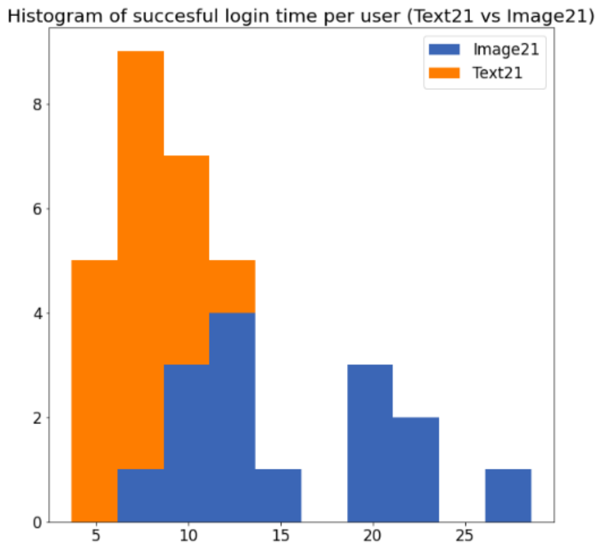
In this section of the report, we will discuss the findings from the data from the user study. We will first report the descriptive statistics and graphs from the data generated by the log processing data program. Then we will be discussing what the better usable password scheme would look like based on the data. Furthermore, we will be discussing the interpretation of the graphs and descriptive statistics. Theses interpretation will be used as arguments to argue which password scheme is better in terms of usability.

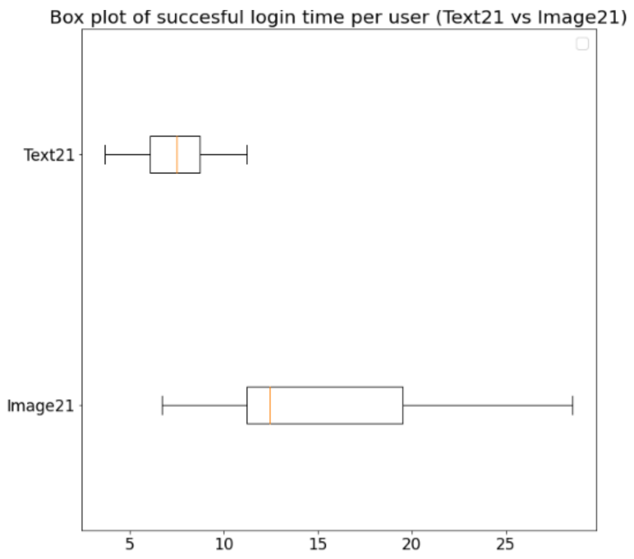


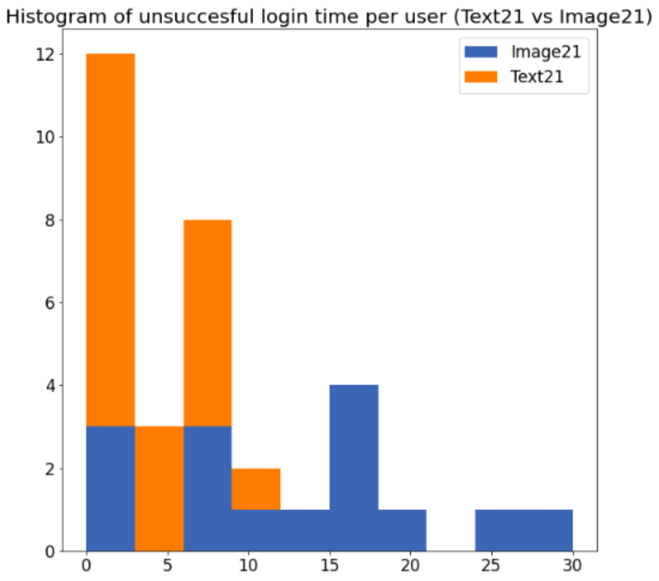


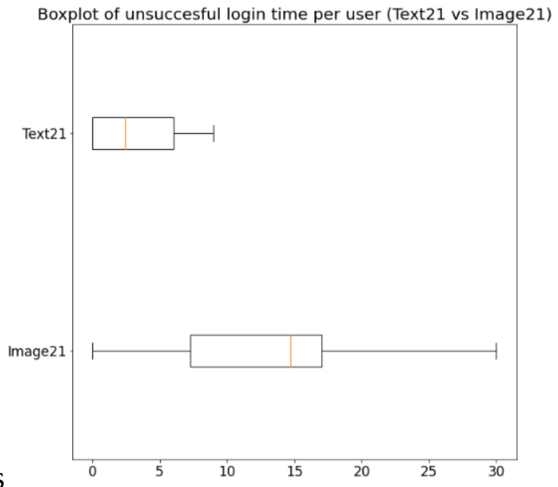

Histogram of the number of unsuccessful logins per user (Text21 vs Image21)
Image21
Text21
0
2
4
6
8
10
12
0
5
10
15
20
25
30


Histogram of total logins (Text21 vs Image21)
Image21
Text21
0
1
2
3
4
5
6
7
8
5
10
15
20
25

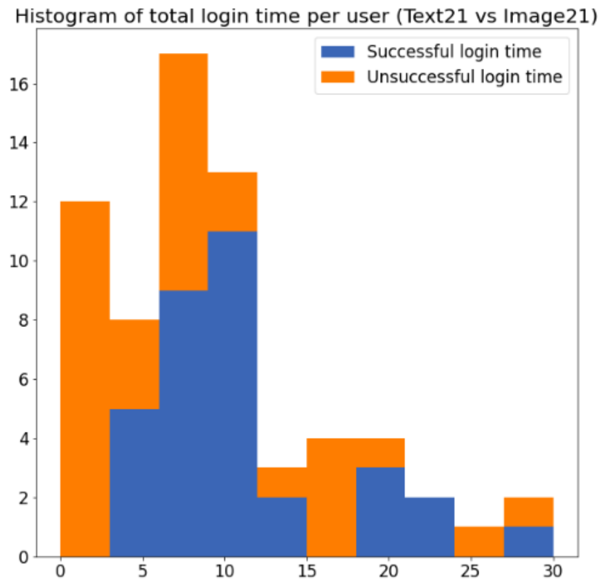

Histogram of succesful login time per user (Text21 vs Image21)
Image21
Text21
0
2
4
6
8
5
10
15
20
25

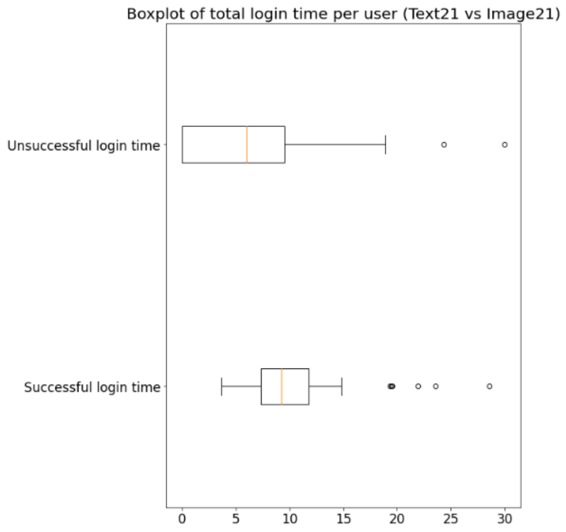

Box plot of succesful login time per user (Text21 vs Image21)
Text21
Image21
5
10
15
20
25


Histogram of unsuccesful login time per user (Text21 vs Image21)
Image21
Text21
0
2
4
6
8
10
12
0
5
10
15
20
25
30


Boxplot of unsuccesful login time per user (Text21 vs Image21)
Text21
Image21
0
5
10
15
20
25
30

s


Histogram of total login time per user (Text21 vs Image21)
Successful login time
Unsuccessful login time
0
2
4
6
8
10
12
14
16
0
5
10
15
20
25
30


Boxplot of total login time per user (Text21 vs Image21)
Unsuccessful login time
Successful login time
0
5
10
15
20
25
30

For this user study, the measures of usability were the memorability of passwords and the speed of password entry. The password scheme with better usability will be more memorable and users will have a faster speed of login compare to other users of password schemes. Also, the password with better usability will have a higher rate of successful logins and a lower rate of unsuccessful logins compare to other password schemes.

Based on the descriptive statistics, we believe that the Text21 password scheme is better in terms of usability. Based on the descriptive statistics, users of the Text21 password scheme on average had a faster login time in all time-related categories compare to users of the Image21 password scheme. Based on the measures of login attempts users of the Text21 scheme had a higher rate of successful login attempts than Image21 users on average. Also, users of the Text21 password scheme, had a lower rate of failed logins compare to Image21 users on average.
Based on all these findings from the descriptive statistics and graphs it seems that overall the Text21 password scheme is better than the Image21 password scheme based on usability. A further direction of this user study is to run inferential statistics to prove this hypothesis.

	Text21	**Image21**
Successful Number of Logins		
Mean	11.33	13.73
Standard Deviation	3.90	2.31
Median	12.00	14.00
Unsuccessful Number of Logins		
Mean	1.33	3.73
Standard Deviation	1.75	3.92
Median	0.50	3.00
Total Number of Logins		
Mean	12.67	17.47
Standard Deviation	4.47	5.46
Median	13.50	17.00

Table 1: *Comparison of descriptive statistics for the number of logins per user Text21 vs. Image21*

	Text21	**Image21**
Successful Login Time (seconds)		

Mean	7.40	15.49
Standard Deviation	2.02	6.20
Median	7.48	12.47
Unsuccessful Login Time (seconds)		
Mean	3.31	12.41
Standard Deviation	3.57	8.87
Median	2.40	14.71
Total Login Time (seconds)		
Mean	7.31	15.98
Standard Deviation	1.99	5.88
Median	7.30	14.21

Table 2: *Comparison of descriptive statistics for the login time per user Text21 vs. Image21*

2 – Design, Implementation, Statistical Inference

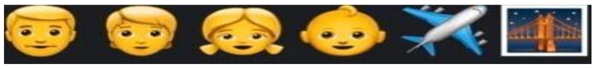
2.1 – Design Rationale

The password scheme we came up with consists of a randomly generated set of Emojis. Since our password scheme allowed for repetition, order mattered, and characters were independent of each other, the equation log2(charset^{length}) was used to calculate the password space (see below). We believe this password length would be enough for minimizing complexity for a user to remember while keeping the password scheme secure. In order to calculate the password space, we decided on a length of 6 Emoji characters and picked 21 distinct Emojis as the charset from Emojipedia. The reason 21 distinct Emojis were chosen for our password scheme is because we wanted to avoid having Emojis that are very similar. The purpose of this was so that we do not confuse our users when they are trying to remember their password. We picked Emojis that are commonly used by people and not too difficult to find on their Emoji keyboard.

Password Space Calculation: Log2(charset^{length}) = log2(21^6) ≈ 25,818,175

The reason we chose to do an emoji password scheme was because of the “Pictorial Superiority Effect”. The Pictorial Superiority Effect as presented in lectures states that images are more easily remembered than words. We thought that if people were given a selection of random Emojis, they would have an easier time of remembering and recalling it since they would be able to store in memory as the image and also whatever is associated with that emoji. For example, if the user is given soccer ball as part of their password, they could store that in their memory with the image as well as what it means. Emojis tend to be very expressive in the sense that they can express emotions, interests, things, and people.

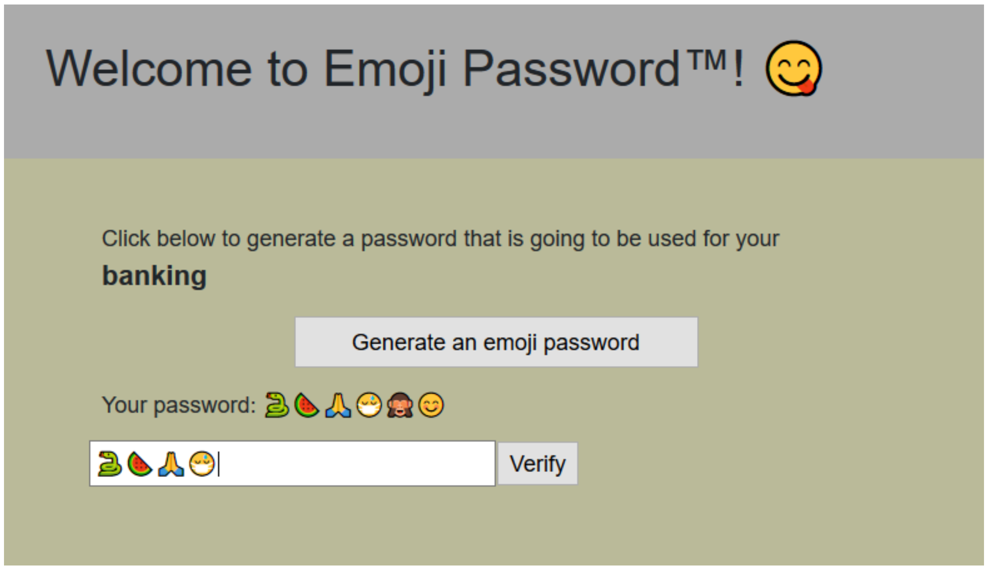
We believe that having an Emoji based password scheme where a user can pick the series of Emojis would be better than the traditional text-based password scheme because they are able to store it in their memory like a story. For example, someone might pick the password on the right for their banking. This could mean that there is a family of four people that love to travel. By associating the series of Emojis with their family and their interest, this password would then be easy to store in memory.



2.2 – Implementation of Emoji-based Password Scheme

Our password scheme was implemented using Node.js to run our Javascript webserver on a Carleton OpenStack instance or click here (works best on mobile device).

Our main page is the starting for a user's session. Starting by running through the first slide which is generating their password for their banking. The user can click on *Generate an emoji password* to give them their random password which they can then practice in the below box and verify if their practice was correct. They must do this at least once to be able to proceed to the email password and shopping password respectively, both of which have similar pages.



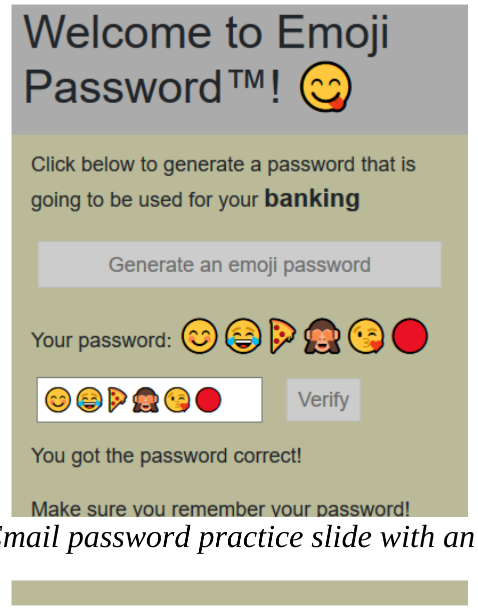
Welcome to Emoji Password™! 😋

Click below to generate a password that is going to be used for your **banking**

Your password: 🐍🍉🙏😷🙊😊

🐍🍉🙏😷

Figure 10: *Main Page on the banking practice slide with partially entered password.*



Welcome to Emoji Password™! 😋

Click below to generate a password that is going to be used for your **banking**

Your password: 😊😂🍕🙈😘🔴

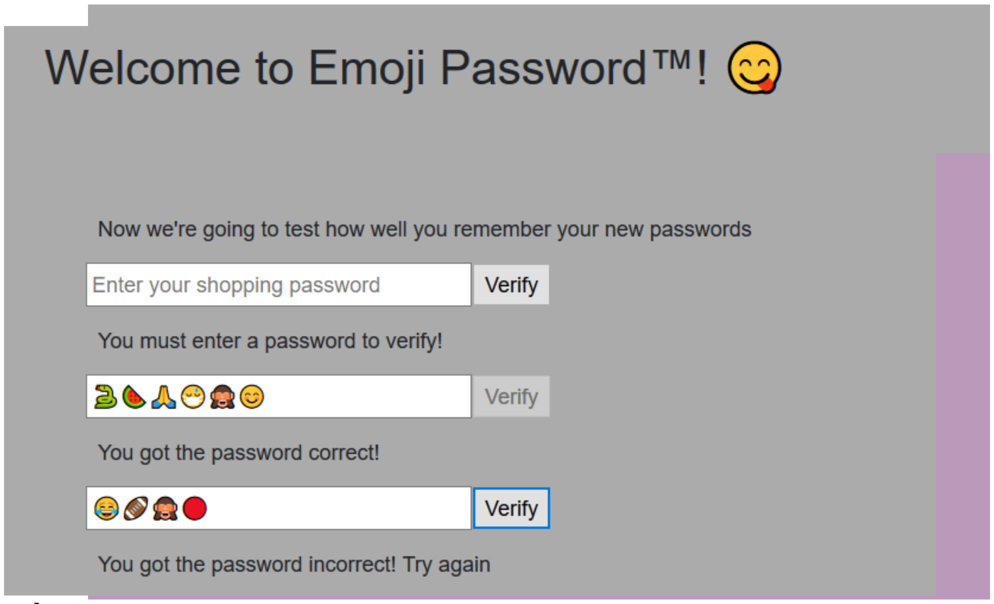
😊😂🍕🙈😘🔴

You got the password correct!

Make sure you remember your password!

Figure 11: *Mobile version of the main page with the banking*

Figure 12: *Email password practice slide with an incorrect password message.*



Welcome to Emoji Password™! 😋

Now we're going to test how well you remember your new passwords

Enter your shopping password | Verify

You must enter a password to verify!

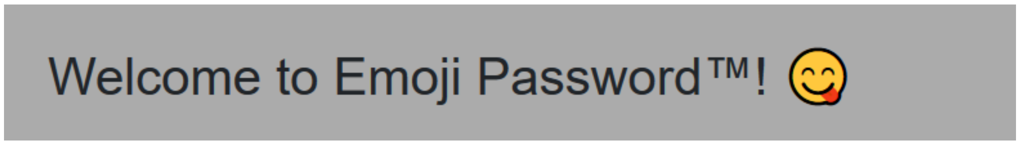
🐍🍉🙏😷🙈😊 | Verify

You got the password correct!

😂🏈🙈🔴 | Verify

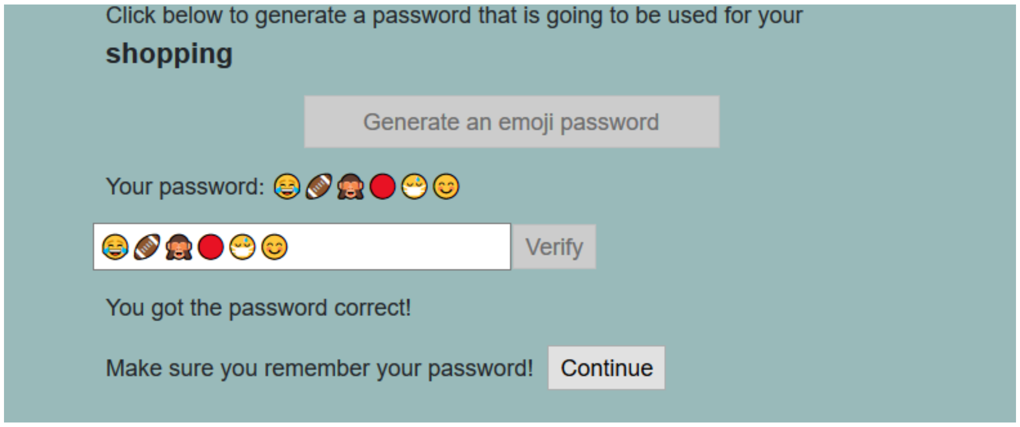
You got the password incorrect! Try again

After the user



Welcome to Emoji Password™! 😋

Figure 13: *Shopping password practice slide with a correct password and allowance to continue.*



Click below to generate a password that is going to be used for your **shopping**

Your password: 😂🏈🙈🔴😷😊

😂🏈🙈🔴😷😊 | Verify

You got the password correct!

Make sure you remember your password! Continue

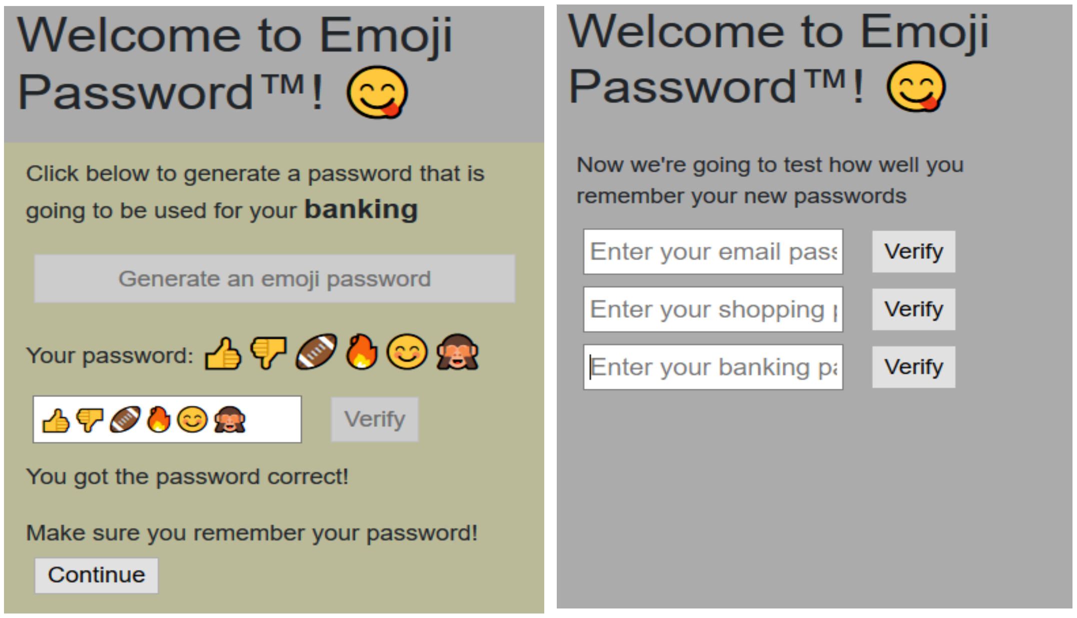
selected and practiced all the passwords, they were given the test slide which randomly ordered the passwords and tested them with the limit of 3 attempts per password. If the user didn't enter anything it was not considered an attempt and would tell give them feedback saying the must have something in the *Enter your [password]* text box to be verified.

Figure 14: *Test slide with the various possible feedback from verification.*

2.3 – Quantitative Testing of Emoji-based Password Scheme

We collected data as the user progressed through the password generation and practice slides as well as the test slide. This data was submitted after they either succeeded or failed on the test slide. We collected six different pieces of information, a list of all the passwords they generated, a list of the three passwords they chose, a list of all of their practice attempts, a list of all their test attempts, an array of three integers representing the number of fails for each respective password, and finally various timestamps from actions from start to finish.

We wanted to have as much information available as possible, so we collected timestamps when they arrive at on our testing page (denoted with "-S"), when the submit the final results (denoted with "-F"), and at every button click in between. This would tell us how long they took for the whole process, and by selecting specific timestamps for example, the last generate before a verify, we would be able to find out how long they took to enter the password in practice that they liked. Every time they generated a password the timestamp would be denoted with "-G". When they would verify in practice it would be denoted with "-V", while on the test slide is the verification would be denoted with "-T". Anytime they pressed continue for the next slide, it was denoted with "-C".



Welcome to Emoji Password™! 😋

Click below to generate a password that is going to be used for your **banking**

Your password: 👍👎🏈🔥😊🙈

👍👎🏈🔥😊🙈

You got the password correct!

Make sure you remember your password!

Welcome to Emoji Password™! 😋

Now we're going to test how well you remember your new passwords

Enter your email pass

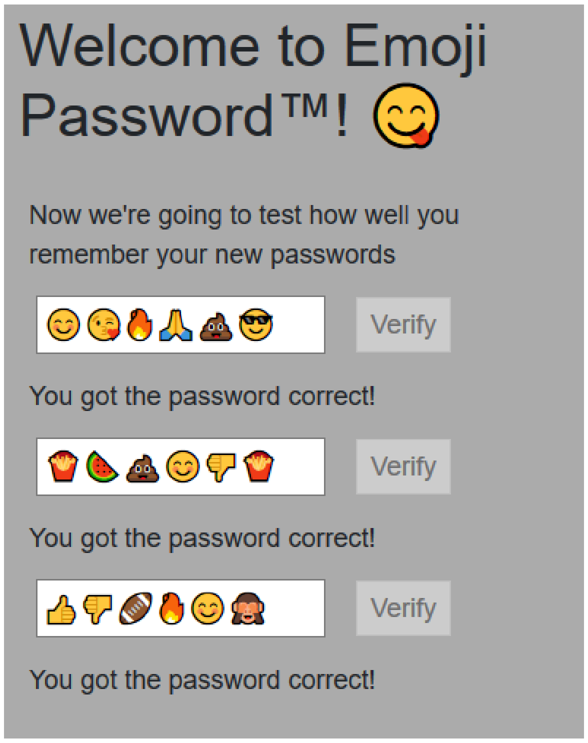
Enter your shopping

Enter your banking pa

Figure 15: *Mobile page version slides with the various buttons that cause timestamps.*

Given that we collected information on the number of failures, and the number of successes were limited, we were able to extrapolate the number of logins and the number of successes from the number of failures. For example, because of the limit of 3 attempts, if the user had a failure array of [3, 3, 3] then they had 9 logins, 0 successes, and 9 failures.

Users data was automatically submitted once the use had no more possible verifications to make. Either because they succeeded on all three passwords, had failed on all three attempts on all three passwords, or had a mixture of failure and success.



Welcome to Emoji Password™! 😋

Now we're going to test how well you remember your new passwords

😊😘🔥🙏💩😎 Verify

You got the password correct!

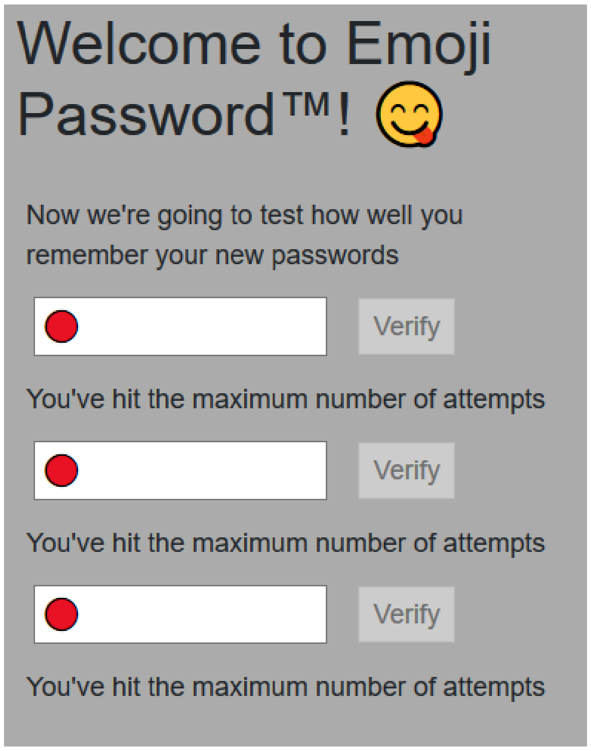
🍟🍉💩😊👎🍟 Verify

You got the password correct!

👍👎🏈🔥😊🙈 Verify

You got the password correct!

Figure 17: Mobile version of a completely successful test.



Welcome to Emoji Password™! 😋

Now we're going to test how well you remember your new passwords

🔴 Verify

You've hit the maximum number of attempts

🔴 Verify

You've hit the maximum number of attempts

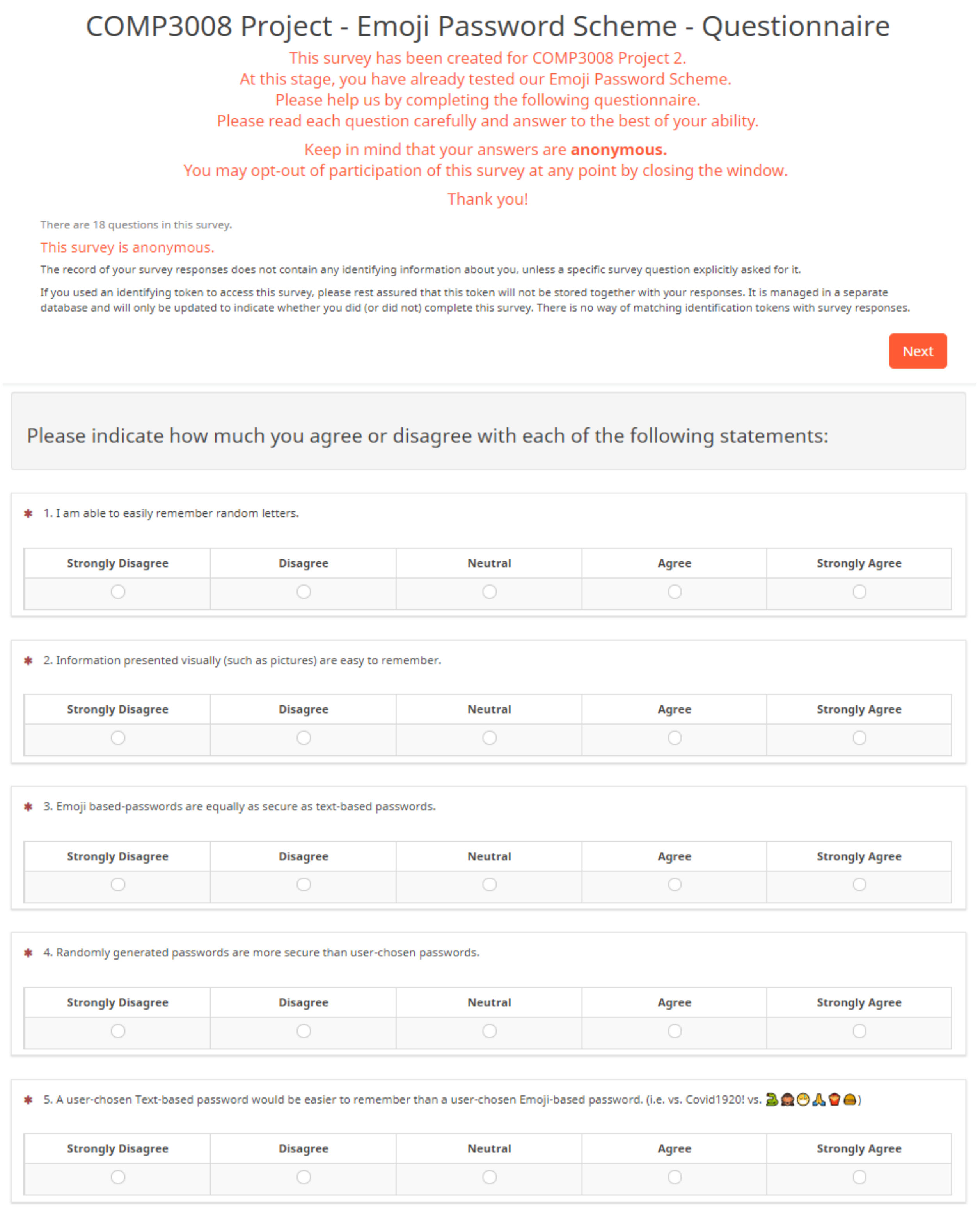
🔴 Verify

You've hit the maximum number of attempts

***Figure 16:** Mobile version of a completely failed test.*

2.4 – Questionnaire

The questionnaire can be found here. Note that there are some "dummy" questions used in the questionnaire such as 7, 9 and 10 just to give the tester a break.



COMP3008 Project - Emoji Password Scheme - Questionnaire

This survey has been created for COMP3008 Project 2.
At this stage, you have already tested our Emoji Password Scheme.
Please help us by completing the following questionnaire.
Please read each question carefully and answer to the best of your ability.

Keep in mind that your answers are **anonymous.**
You may opt-out of participation of this survey at any point by closing the window.

Thank you!

There are 18 questions in this survey.

This survey is anonymous.

The record of your survey responses does not contain any identifying information about you, unless a specific survey question explicitly asked for it.

If you used an identifying token to access this survey, please rest assured that this token will not be stored together with your responses. It is managed in a separate database and will only be updated to indicate whether you did (or did not) complete this survey. There is no way of matching identification tokens with survey responses.

Please indicate how much you agree or disagree with each of the following statements:

* 1. I am able to easily remember random letters.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○

* 2. Information presented visually (such as pictures) are easy to remember.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○

* 3. Emoji based-passwords are equally as secure as text-based passwords.

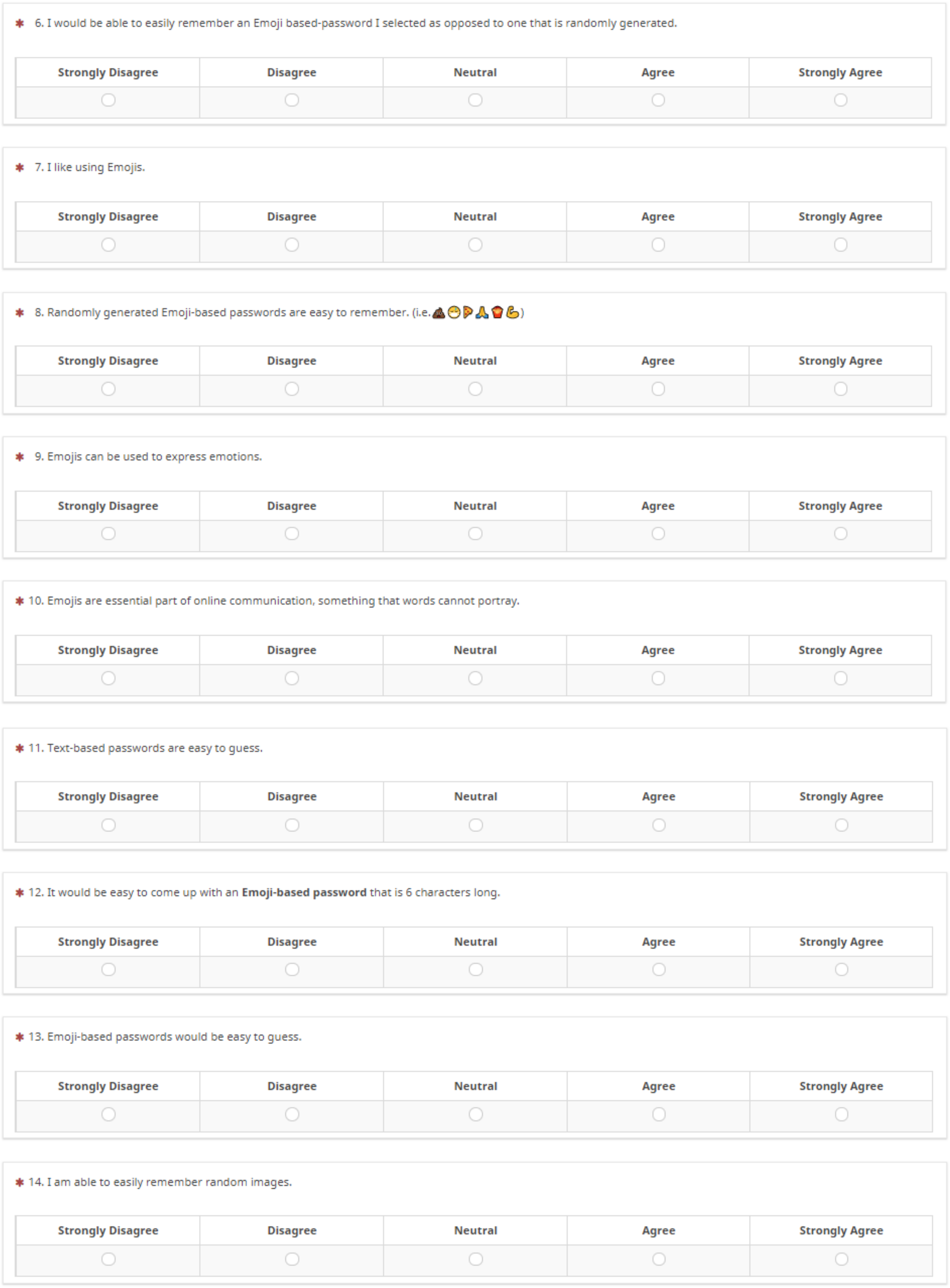
Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○

* 4. Randomly generated passwords are more secure than user-chosen passwords.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○

* 5. A user-chosen Text-based password would be easier to remember than a user-chosen Emoji-based password. (i.e. vs. Covid1920! vs. 🐍🙈😅🙏🍟🍔)

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○



* 6. I would be able to easily remember an Emoji based-password I selected as opposed to one that is randomly generated.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○

* 7. I like using Emojis.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○

* 8. Randomly generated Emoji-based passwords are easy to remember. (i.e.💩😅🍕🙏🍟💪)

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○

* 9. Emojis can be used to express emotions.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○

* 10. Emojis are essential part of online communication, something that words cannot portray.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○

* 11. Text-based passwords are easy to guess.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○

* 12. It would be easy to come up with an **Emoji-based password** that is 6 characters long.

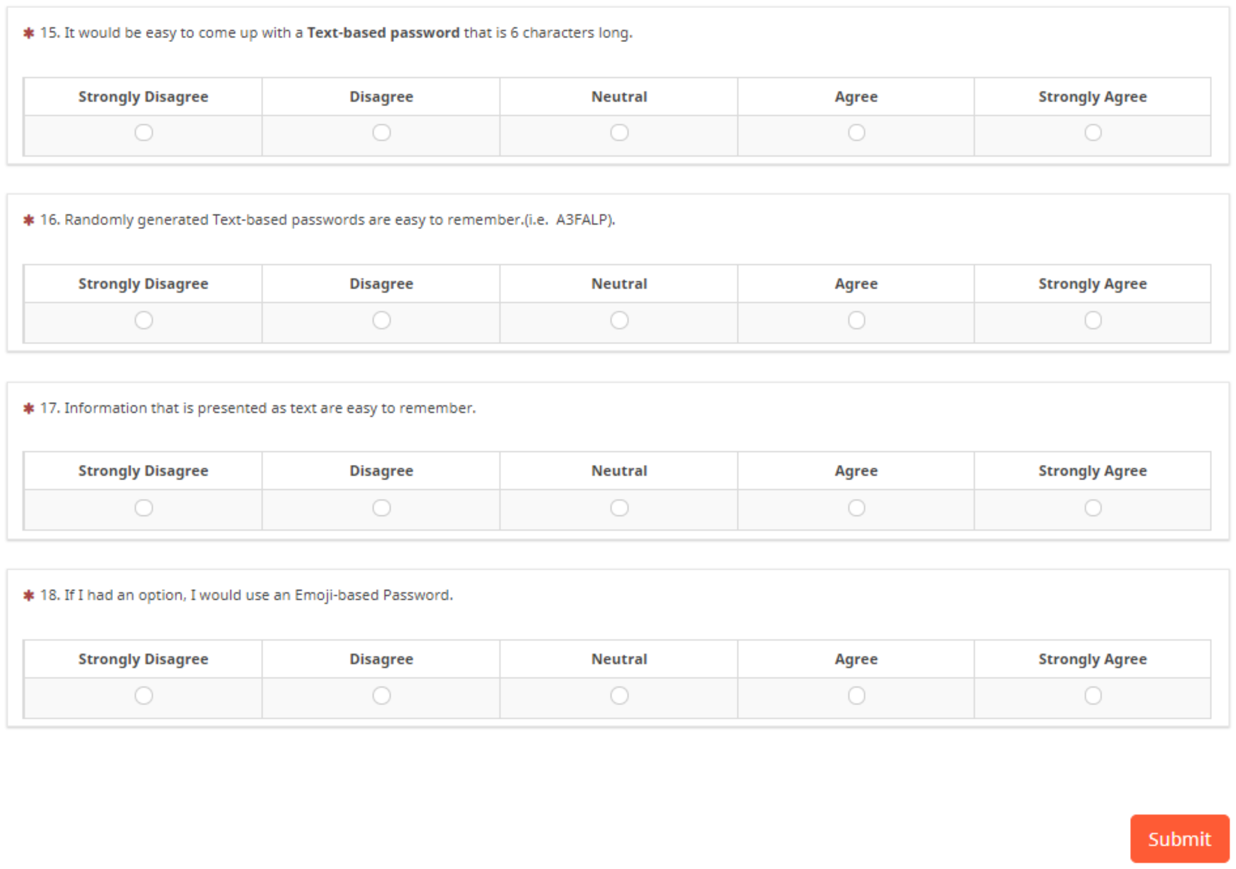
Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○

* 13. Emoji-based passwords would be easy to guess.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○

* 14. I am able to easily remember random images.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○



* 15. It would be easy to come up with a **Text-based password** that is 6 characters long.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○

* 16. Randomly generated Text-based passwords are easy to remember.(i.e. A3FALP).

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○

* 17. Information that is presented as text are easy to remember.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○

* 18. If I had an option, I would use an Emoji-based Password.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○

2.5 – User Testing

Our Emoji-based password scheme was tested with 18 people. Our consent form can be found here.

2.6 – Usability Testing Comparison

In order to compare the usability between Text21 and our Emoji password scheme, descriptive statistics as well as a questionnaire was used. In the questionnaire, users were asked what they think about the Emoji password scheme in comparison to Text-based password scheme using a variety of different questions. Some of the important results from the test data as well as the questionnaire are highlighted in this section.

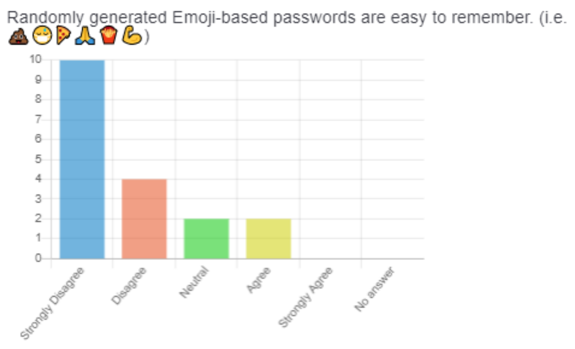
Overall, when comparing the descriptive statistics of our Emoji-based password scheme with Text21, it is evident that not many people had success with logins using the Emoji scheme (see Table 3). The average login for the randomly generated Emoji-based password scheme was 1. This result is significant as the standard deviation is 1.15 meaning that there is low variability between the results of successful login.

	Text21	**Emoji**
Successful Number of Logins		
Mean	11.33	1
Standard Deviation	3.90	1.15
Median	12.00	1
Unsuccessful Number of Logins		
Mean	1.33	5.89
Standard Deviation	1.75	3.49
Median	0.50	6
Total Number of Logins		
Mean	12.67	6.89
Standard Deviation	4.47	2.35
Median	13.50	7

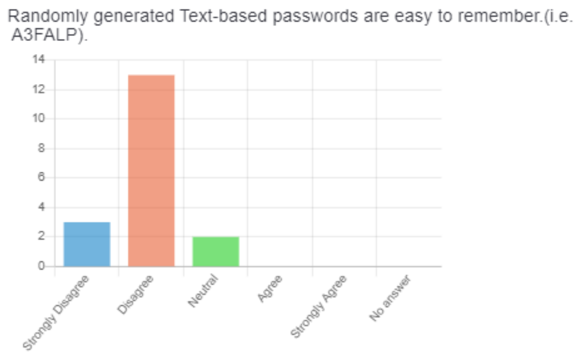
Table 3: *Comparison of descriptive statistics for the number of logins per user Text21 vs. Emoji*

The results of the questionnaire also showed similar results as most people strongly disagreed that remembering a randomly generated Emoji password was easy as presented below. It is also evident from the results that most people disagreed that a randomly generated Text-based password was easy to remember. There is not much we can conclude from this other than the fact that people do not find remembering a randomly generated password easy.

Emoji-based Password



Text-based Password



By testing our Emoji password scheme, we realized that most people had trouble re-entering their password and that the login time for the Emoji-based password scheme was also much higher than that of the Text21 password scheme as presented in Table 4. Looks like overall people not only had trouble remembering their password using the Emoji scheme but they also took much longer when recalling their password. The standard deviation for the login time of successful, unsuccessful, and total login time is very high which means that there is a great variance between the results. When analyzing the data, we noticed that some people completed the tasks very quickly, while others took much longer. As the testing for our password scheme was not completed in a controlled environment, the variance between the results is expected.

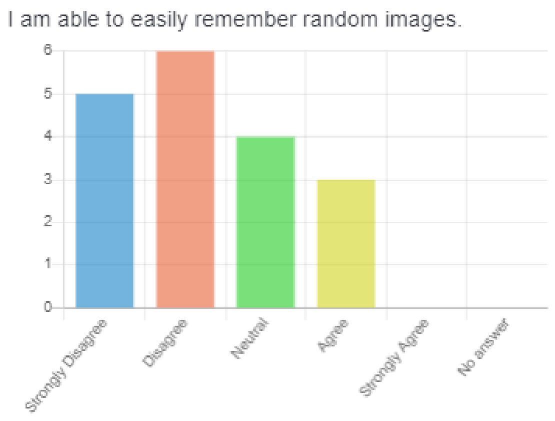
	Text21	**Emoji**
Successful Login Time (seconds)		
Mean	7.40	21.67
Standard Deviation	2.02	35.23
Median	7.48	1.5
Unsuccessful Login Time (seconds)		
Mean	3.31	17.33
Standard Deviation	3.57	17.95
Median	2.40	12
Total Login Time (seconds)		
Mean	7.31	25
Standard Deviation	1.99	37.11
Median	7.30	25

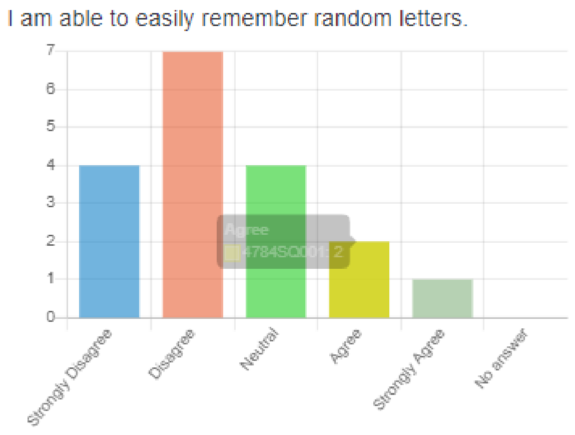
Table 4: Comparison of descriptive statistics for the login time per user Text21 vs. Emoji

From the results, we noticed that some people quickly entered in the correct password. We believe that these people took note of their password scheme or took screenshots with their phone and therefore were able to recall it later. Unfortunately, this is something we noticed after conducting all our tests. I believe that it would have been a good idea to let the testers know to not use anything that would help the recall the passwords for better and more accurate results.

In order to test memory, we asked questions regarding password memory and recall. For example, one question was whether people found it easy to remember random letters vs. random Emojis. The results varied for both Emoji and Text-based passwords with most people disagreeing for both types.

<u>Emoji-based Password</u> **<u>Text-based Password</u>**

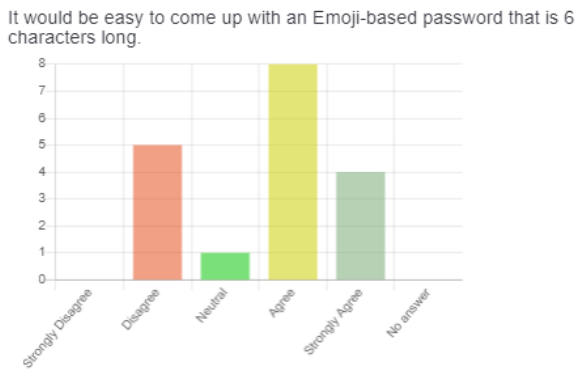


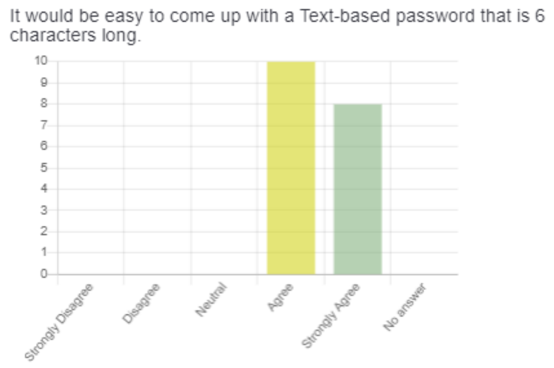


When asked which password scheme is easier to come up with as a user, overall, users stated that it is easier for them to come up with a Text-based password. We believe that the reason people were unsure about Emoji-based password would be because they have not been exposed to it as much as Text-based password.

Emoji-based Password

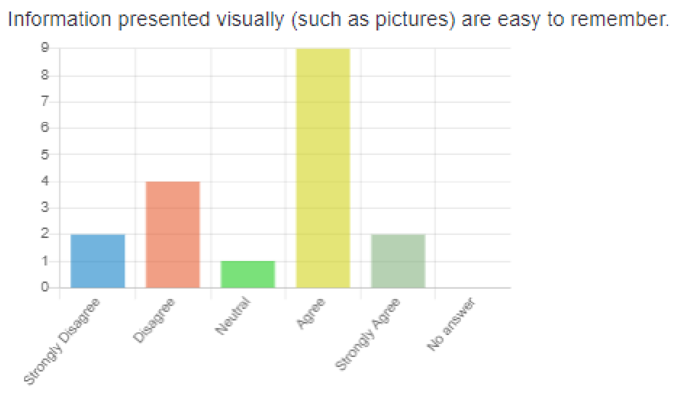
Text-based Password



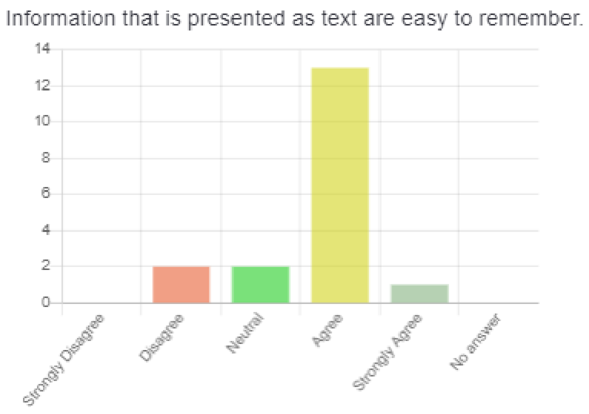


Users were asked whether they can remember information presented visually and in text. The results of this question were that about 83% (15 out of 18) people agreed that text-based information is easy to remember whereas only 56% (10 out of 18) people agreed that information presented visually are easy to remember. This is very interesting results as it goes a bit against the Pictorial Superiority Effect which states that images are more easily remembered than words. Again, it is hard to conclude anything from these results as they are based on a very small sample size and the test and questionnaire were not controlled. We believe that in a controlled environment with a larger sample size, there was potential of getting more accurate results.

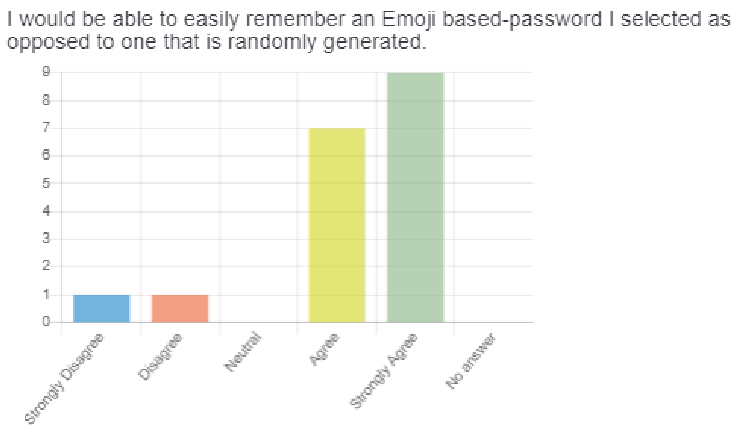
Emoji-based Password



Text-based Password

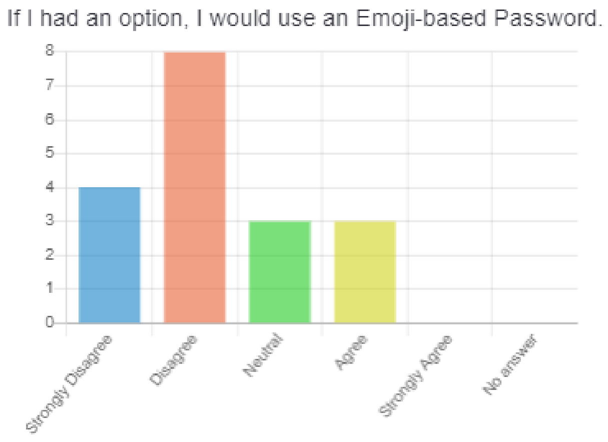


Our password scheme was based on a randomly generated, six-character long in length Emojis. The descriptive statistics shown in Table 3 and 4 indicate that people overall had a more difficult time remembering their password and therefore there were not too many successful logins. The total login time, successful and unsuccessful login times were much higher in comparison to Text21 password scheme. In our questionnaire, we asked the 18 testers various questions to see whether they find it easy to remember Emoji passwords as opposed to the traditional text-based password. Overall, the results showed that most people were comfortable and felt that the text-based password was something they could remember more easily in comparison to the Emoji scheme.



We were interested to know if their perception would change if they were given an option to pick their own set of Emojis. The results of this question showed that almost 89% (16 out of 18) people started that they would easily be able remember an Emoji-based password if they had the option to select them as opposed to a randomly generated one. This confirms our main logic behind coming up with this password scheme as we believe that this would be secure but also because Emojis carry a message whether that’s in terms of an emotion or interest, it would help people to remember and recall.

Lastly, we asked if people would use our Emoji-based password scheme if they had the option. As presented in the results below, more than half of the people stated that they would not use our password scheme. We believe that if people were given the option to select their own password scheme, they would be interested in using Emojis as they are a way to express feelings and interests which perhaps would make remembering and recalling the password easier than a randomly generated password.



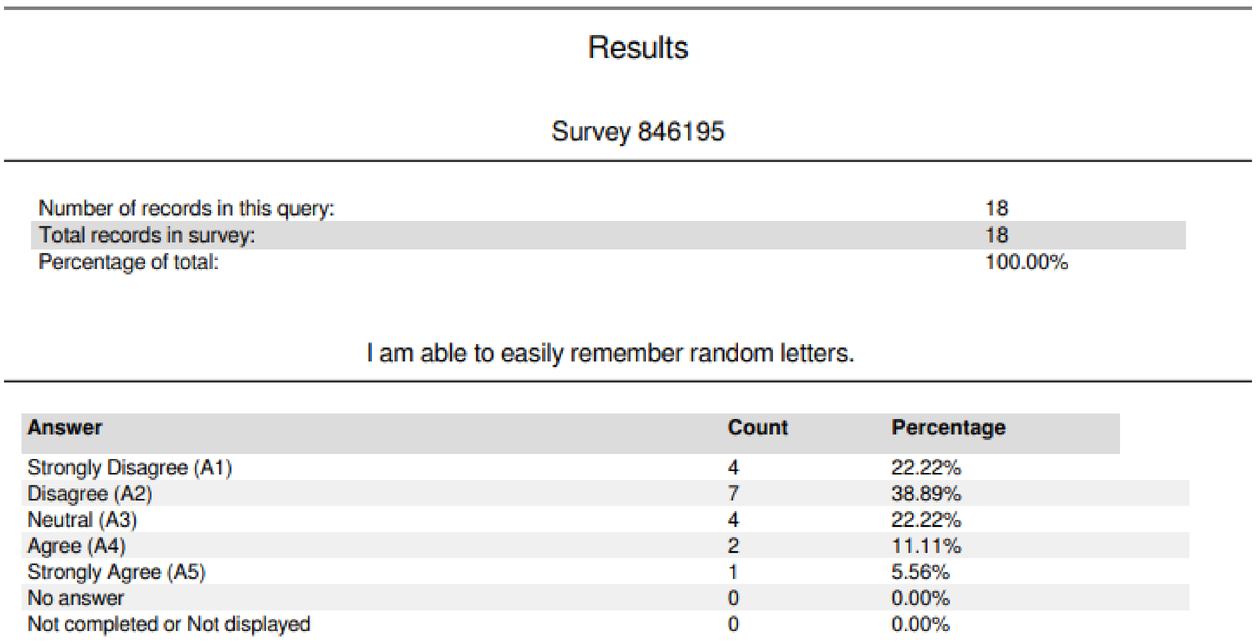
In conclusion, based on the descriptive statistics and the results of our questionnaire it is evident that Text21 password scheme is much better than our Emoji-based password scheme. Text21 password scheme had higher successful logins and login time in comparison to our password scheme. As discussed earlier, when asked if there was an option to use an Emoji-based password, about 17% (3 out of 18) people agreed. From this we can conclude that people are more in favour of using a normal text-based password. We believe the reason for this is that people have not had the exposure to Emoji-based passwords like they have for the normal text-based password

3 – Workload Distribution and Summary

Name	Student Number	Contributions
Mohab Abdelkader	100981616	• Password Scheme Front-end • Part 2.2 writeup • Part 2.3 writeup • Part 2.6 stats
Paradis Esmaeelzadeh Khadem	100797010	• Part 2.1 writeup • Part 2.4 writeup • Part 2.5 writeup • Part 2.6 stats & writeup
Darren Pierre	101015833	• Part 1 coding • Part 1 writeup
Henri Umba	101022562	• Password Scheme Back-end • Part 2.2 writeup
Nem Zutkovic	101085982	• Part 1 writeup

4 – Appendix

Results of questionnaire:



Results

Survey 846195

Number of records in this query:	18
Total records in survey:	18
Percentage of total:	100.00%

I am able to easily remember random letters.

Answer	Count	Percentage
Strongly Disagree (A1)	4	22.22%
Disagree (A2)	7	38.89%
Neutral (A3)	4	22.22%
Agree (A4)	2	11.11%
Strongly Agree (A5)	1	5.56%
No answer	0	0.00%
Not completed or Not displayed	0	0.00%

Information presented visually (such as pictures) are easy to remember.

Answer	Count	Percentage
Strongly Disagree (A1)	2	11.11%
Disagree (A2)	4	22.22%
Neutral (A3)	1	5.56%
Agree (A4)	9	50.00%
Strongly Agree (A5)	2	11.11%
No answer	0	0.00%
Not completed or Not displayed	0	0.00%

Emoji based-passwords are equally as secure as text-based passwords.

Answer	Count	Percentage
Strongly Disagree (A1)	1	5.56%
Disagree (A2)	4	22.22%
Neutral (A3)	8	44.44%
Agree (A4)	4	22.22%
Strongly Agree (A5)	1	5.56%
No answer	0	0.00%
Not completed or Not displayed	0	0.00%

Randomly generated passwords are more secure than user-chosen passwords.

Answer	Count	Percentage
Strongly Disagree (A1)	1	5.56%
Disagree (A2)	1	5.56%
Neutral (A3)	5	27.78%
Agree (A4)	5	27.78%
Strongly Agree (A5)	6	33.33%
No answer	0	0.00%
Not completed or Not displayed	0	0.00%

A user-chosen Text-based password would be easier to remember than a user-chosen Emoji-based password. (i.e. vs. Covid1920! vs. ⍰⍰⍰⍰⍰⍰⍰⍰⍰⍰⍰⍰)

Answer	Count	Percentage
Strongly Disagree (A1)	1	5.56%
Disagree (A2)	1	5.56%
Neutral (A3)	2	11.11%
Agree (A4)	7	38.89%
Strongly Agree (A5)	7	38.89%
No answer	0	0.00%
Not completed or Not displayed	0	0.00%

I would be able to easily remember an Emoji based-password I selected as opposed to one that is randomly generated.

Answer	Count	Percentage
Strongly Disagree (A1)	1	5.56%
Disagree (A2)	1	5.56%
Neutral (A3)	0	0.00%
Agree (A4)	7	38.89%
Strongly Agree (A5)	9	50.00%
No answer	0	0.00%
Not completed or Not displayed	0	0.00%

I like using Emojis.

Answer	Count	Percentage
Strongly Disagree (A1)	1	5.56%
Disagree (A2)	3	16.67%
Neutral (A3)	5	27.78%
Agree (A4)	6	33.33%
Strongly Agree (A5)	3	16.67%
No answer	0	0.00%
Not completed or Not displayed	0	0.00%

Randomly generated Emoji-based passwords are easy to remember. (i.e.⍰⍰⍰⍰⍰⍰⍰⍰⍰⍰⍰⍰)

Answer	Count	Percentage
Strongly Disagree (A1)	10	55.56%
Disagree (A2)	4	22.22%
Neutral (A3)	2	11.11%
Agree (A4)	2	11.11%
Strongly Agree (A5)	0	0.00%
No answer	0	0.00%
Not completed or Not displayed	0	0.00%

Emojis can be used to express emotions.

Answer	Count	Percentage
Strongly Disagree (A1)	0	0.00%
Disagree (A2)	0	0.00%
Neutral (A3)	0	0.00%
Agree (A4)	8	44.44%
Strongly Agree (A5)	10	55.56%
No answer	0	0.00%
Not completed or Not displayed	0	0.00%

Emojis are essential part of online communication, something that words cannot portray.

Answer	Count	Percentage
Strongly Disagree (A1)	1	5.56%
Disagree (A2)	0	0.00%
Neutral (A3)	2	11.11%
Agree (A4)	11	61.11%
Strongly Agree (A5)	4	22.22%
No answer	0	0.00%
Not completed or Not displayed	0	0.00%

Text-based passwords are easy to guess.

Answer	Count	Percentage
Strongly Disagree (A1)	2	11.11%
Disagree (A2)	5	27.78%
Neutral (A3)	8	44.44%
Agree (A4)	2	11.11%
Strongly Agree (A5)	1	5.56%
No answer	0	0.00%
Not completed or Not displayed	0	0.00%

It would be easy to come up with an Emoji-based password that is 6 characters long.

Answer	Count	Percentage
Strongly Disagree (A1)	0	0.00%
Disagree (A2)	5	27.78%
Neutral (A3)	1	5.56%
Agree (A4)	8	44.44%
Strongly Agree (A5)	4	22.22%
No answer	0	0.00%
Not completed or Not displayed	0	0.00%

Emoji-based passwords would be easy to guess.

Answer	Count	Percentage
Strongly Disagree (A1)	6	33.33%
Disagree (A2)	10	55.56%
Neutral (A3)	0	0.00%
Agree (A4)	2	11.11%
Strongly Agree (A5)	0	0.00%
No answer	0	0.00%
Not completed or Not displayed	0	0.00%

I am able to easily remember random images.

Answer	Count	Percentage
Strongly Disagree (A1)	5	27.78%
Disagree (A2)	6	33.33%
Neutral (A3)	4	22.22%
Agree (A4)	3	16.67%
Strongly Agree (A5)	0	0.00%
No answer	0	0.00%
Not completed or Not displayed	0	0.00%

It would be easy to come up with a Text-based password that is 6 characters long.

Answer	Count	Percentage
Strongly Disagree (A1)	0	0.00%
Disagree (A2)	0	0.00%
Neutral (A3)	0	0.00%
Agree (A4)	10	55.56%
Strongly Agree (A5)	8	44.44%
No answer	0	0.00%
Not completed or Not displayed	0	0.00%

Randomly generated Text-based passwords are easy to remember.(i.e. A3FALP).

Answer	Count	Percentage
Strongly Disagree (A1)	3	16.67%
Disagree (A2)	13	72.22%
Neutral (A3)	2	11.11%
Agree (A4)	0	0.00%
Strongly Agree (A5)	0	0.00%
No answer	0	0.00%
Not completed or Not displayed	0	0.00%

Information that is presented as text are easy to remember.

Answer	Count	Percentage
Strongly Disagree (A1)	0	0.00%
Disagree (A2)	2	11.11%
Neutral (A3)	2	11.11%
Agree (A4)	13	72.22%
Strongly Agree (A5)	1	5.56%
No answer	0	0.00%
Not completed or Not displayed	0	0.00%

If I had an option, I would use an Emoji-based Password.

Answer	Count	Percentage
Strongly Disagree (A1)	4	22.22%
Disagree (A2)	8	44.44%
Neutral (A3)	3	16.67%
Agree (A4)	3	16.67%
Strongly Agree (A5)	0	0.00%
No answer	0	0.00%
Not completed or Not displayed	0	0.00%