COMP 3008 – Project 2

Quantitative Usability Evaluation

Team-2
Cheryl Dunn 100963953,
Tristan Cordeau 100906164
Savanna Endicott 100933116
Jagger De Leo 100938125

Submitted to: Dr. Robert Biddle
COMP 3008 Human Computer Interactions
School of Computer Science
Carleton University
April 6th 2017

Table of Contents

Introduction

In the world of software and systems development, gathering feedback about these systems is one of the most important aspects of development. The feedback can be used to discover major and minor flaws in the system as well as shortcomings, usability feedback, and statistics that can be analyzed in order to learn something about the user's behaviors and their success in using the system. In this project, we investigated this collection and use of descriptive statistics on password systems.

Our task was to analyze user's success entering passwords in the given password system, then create our own password system. This consisted of considering different ideas and aspects of the password problem in order to design the system, developing the system itself, putting in place a collection system for descriptive statistics of the user's experience and success or failure, and finally manipulating and organizing that information in ways that provide useful data analysis. This system was then compared to the system given in the first part of the project to generate conclusions.

Sample Data and Descriptive Statistics

Advantages and Disadvantages of Schemas

Three possible password schemas include *Text28*, *Blankpt28* and *Imagept28*. Each of these schemas were used to collect data on passwords. Users were asked to login to a website using one of the three schemas as a user name. The website would next generate a password based on the schema and data regarding how users interact and remember passwords, below is a description of each scheme along with their advantages and disadvantages.

Scheme Text28

When user's login using the *Text28* username and select create a password, the website creates a password that meets the requirements of 6 random lower-case letters. An example can be seen in figure 1 below. An advantage of this scheme is that the password is simple and easy to remember because it is within the magic number 7+/- 2 for item recall. Some disadvantages include that you are required to recall the password without any hints, even if you can't recall it. The human brain is better at recognition than recall, therefore a graphical password offers a higher chance of recognition.

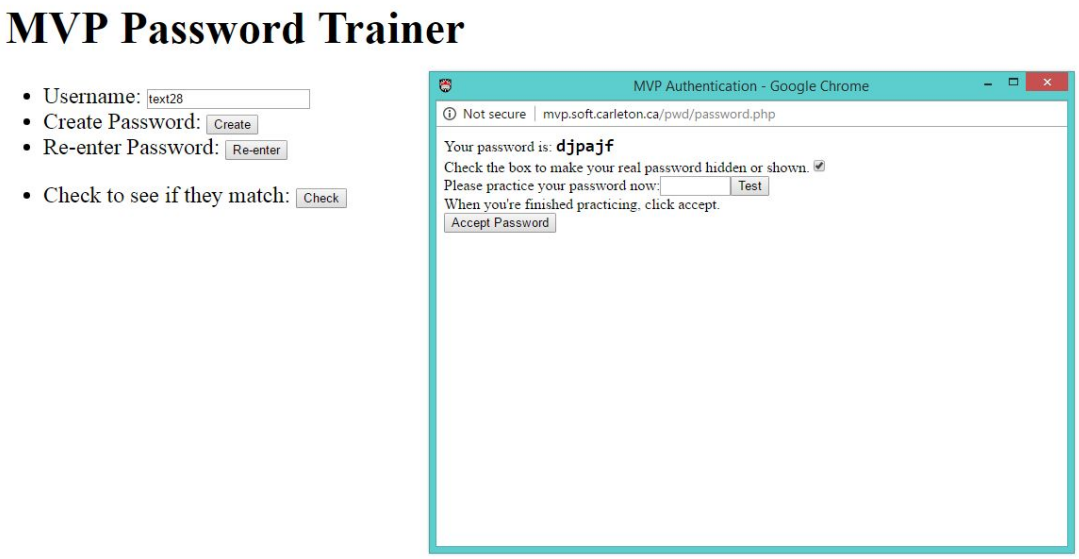


Figure 1 text28 Schema

Scheme Blankpt28

When users log in using the *Blankpt28* username and select *Create Password*, the website creates a password using 6 random tiles, chosen from a blank grid of 80 tiles. An example can be seen in figure 2. The advantages of this scheme is that it is graphical and therefore includes some recognition, though still requires recall of the password based on cues from the grid. Disadvantages include that it is more complex than a simple number and a user may not be accurate in their positioning. Therefore, the schema must adjust for a correct pattern that is one row down/up or one column left/right. It also requires shallow processing, in the form of positioning of the tiles, which requires maintenance rehearsal to encode information into memory.

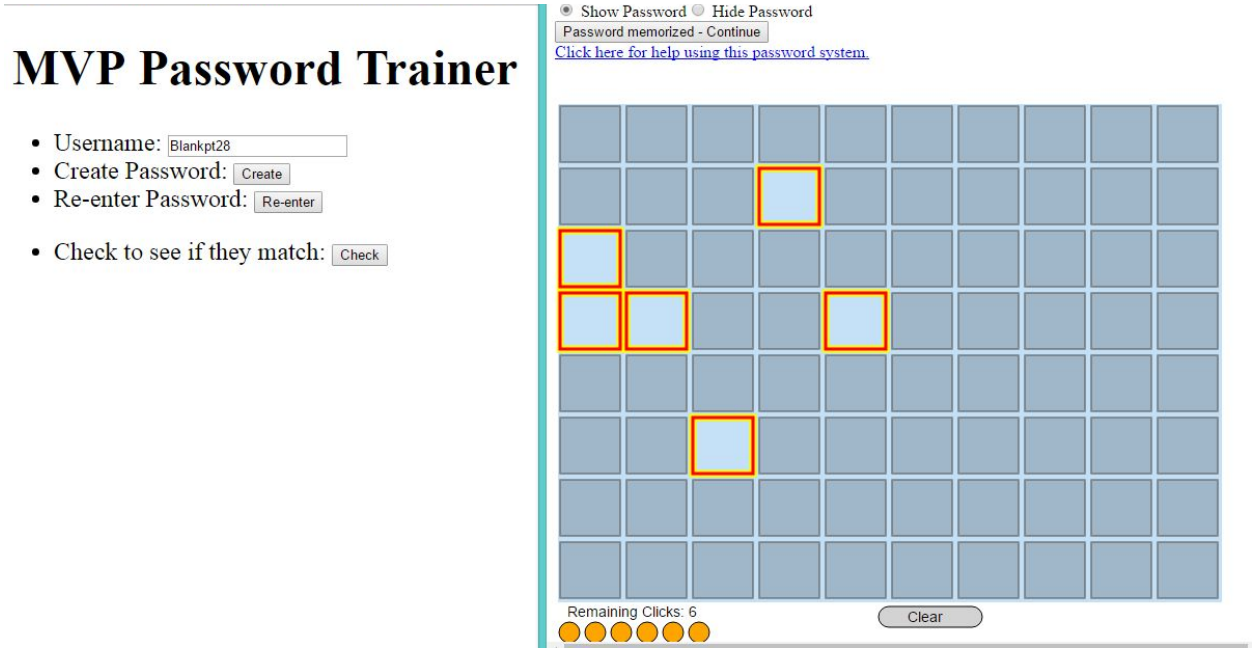


Figure 2 Blankpt28 Schema

Scheme Imagept28

When users login using the *Imagept28* username and select *Create Password*, the website creates a password using 6 random tiles, chosen from an image grid of 80 tiles. An example can be seen in figure 3. This schema is arguably the most memorable as the image creates cues for recognition that are highly distinguishable and increases the learnability of the password. It uses dual encoding theory by incorporating position as well as a descriptive label of the tile, increasing the chances of recall. A disadvantage is that it takes shallow processing which requires maintenance rehearsal to encode the information into memory.

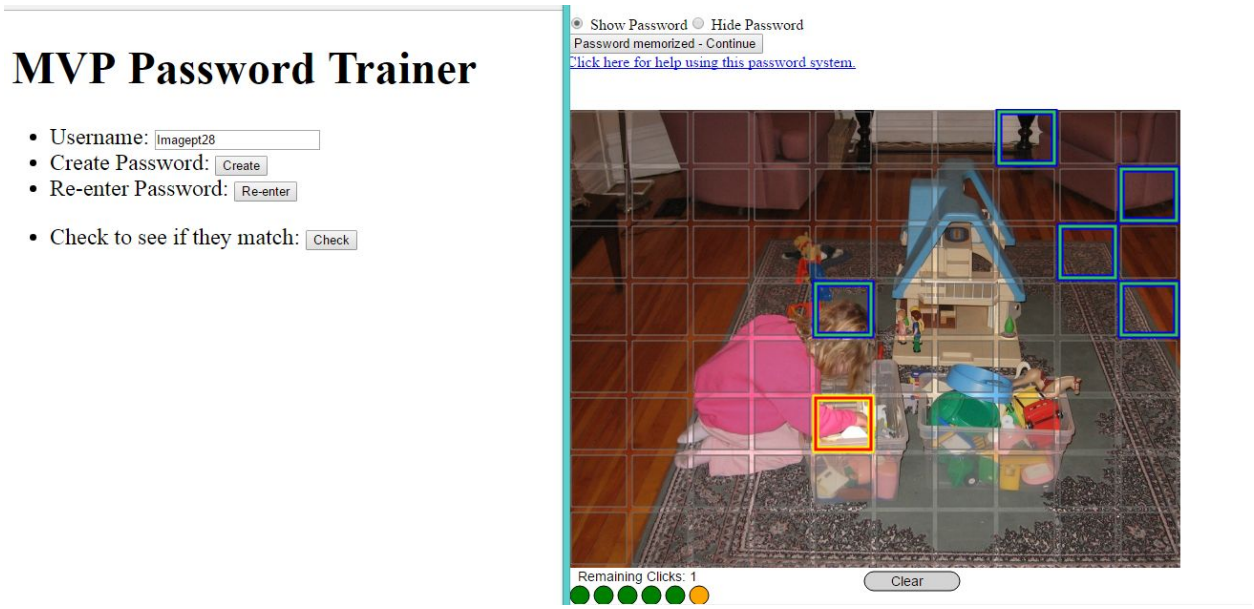


Figure 3 Imagept28 Schema

Processing Software

Within this project three log files corresponding to either a text, blank or image scheme were provided for analysis. Through the design and implementation of a software program to process the log data it was possible to format those log files into new log files that could be analyzed using R.

High-level Explanation

The software program was designed to read each row from the given log file and create a new csv file that was formatted to easily parse and analyze data. The initial format of the log file for each scheme contained a row for every event that occurred on the password trainer site, with a matching column to indicate the time, site, user, scheme, mode, event and data, and looked like figure 4, except with many more entries.

	A	B	C	D	E	F	G
1	time	site	user	scheme	mode	event	data
2	7/25/2011 11:38	HS-2011-1	ast403	unknown;	register	success	
3	7/25/2011 11:38	HS-2011-1	ast403	unknown;	verifytest	failure	sna
4	7/25/2011 11:39	wvacation	ast403	textrando	create	start	Mozilla/5.0 (V
5	7/25/2011 11:39	wvacation	ast403	textrando	create	pwtog	hide
6	7/25/2011 11:39	wvacation	ast403	textrando	create	pwtog	show
7	7/25/2011 11:39	wvacation	ast403	textrando	create	pwtog	hide
8	7/25/2011 11:39	wvacation	ast403	textrando	create	pwtog	show
9	7/25/2011 11:40	wvacation	ast403	textrando	create	pwtest	good
10	7/25/2011 11:40	wvacation	ast403	textrando	create	password:	pw:mwixvl
11	7/25/2011 11:40	wvacation	ast403	textrando	enter	start	Mozilla/5.0 (V
12	7/25/2011 11:40	wvacation	ast403	textrando	enter	password:	pwN:ast403
13	7/25/2011 11:40	wvacation	ast403	textrando	create	start	Mozilla/5.0 (V

Figure 4. Example Given Log File

	A	B	C	D	E
1		user	scheme	event	timeTaken_days
2	1	ast403	unknown;N/A	testing	0
3	2	ast403	unknown;N/A	testing	0
4	3	ast403	textrandom;az-6	Created P	0
5	4	ast403	textrandom;az-6	testing	7
6	5	ast403	textrandom;az-6	testing	7
7	6	ast403	textrandom;az-6	testing	22
8	7	ast403	textrandom;az-6	testing	23
9	8	ast403	textrandom;az-6	testing	31
10	9	ast403	textrandom;az-6	testing	32
11	10	ast403	textrandom;az-6	testing	36
12	11	ast403	textrandom;az-6	testing	39
13	12	ast403	textrandom;az-6	Created P	0

Figure 5 Outputted CSV Format

The software program sorts the data into user, scheme, event and time, the new format can be seen in figure 5. To see the exact output files from the software sort please reference TextScheme.csv, BlankScheme.csv and ImageScheme.csv. Potential events: password created, successful login, failed login or testing. The following table describes the transitions from the log file to the new csv.

Input from entry in log file	**Event in new csv file**	**Time(secs) in new csv file**
Event = create & Mode = start	Password created	∆Time = 0
Event = login & Mode = Successful	Successful login	∆Time = Final – Initial
Event = login & Mode = Failure	Failed login	∆Time = Final – Initial
Anything else	Test	∆Time = Final – initial

The data was organized in this format so that it would be possible to calculate the time difference between a “password created” event and a successful or failed login attempt. Please note that the time taken is recorded in seconds. On password created, the initial time is set to the logged entry time whereas for every other event the final time is set to the time of the logged entry. To calculated the difference in time (∆Time), the final time is the time of logged entry, and ∆Time = final time – initial time. This also implies that on a password created event the time will always be 0.

Once the Data is organized the new csv files need to be analyzed, within this section of the software entries per each scheme, each user's number of logins (successful and failed) as well as time taken to login (successful and failed) are used to create histograms and boxplots that can later be analyzed.

PseudoCode

In the next section you can find pseudo code for the GetData(), sortData() and AnalyzeData() functions. GetData() is a main function that when called, will pass in the proper csv files to the sort and analyze files. SortData() gets called from getData() to sort the data from a log file and output a new csv file following a more usable format. AnalzyeData() gets called from getData() to analyze the inputted files and output descriptive statistics. getGraphs is a function that takes the three dataframes from analyzeData and creates histograms and boxplots.

```
GetData():
        For each scheme  : sortData(scheme)
        For each scheme  : analyzeData(scheme)
        getGraphs(scheme1, scheme2, scheme2)

sortData(scheme):
        let users be a list of unique users in the scheme I
        let df_old be a data frame that stores information from I, user, scheme, mode, event, time
        let df be a data frame that stores user, scheme, event, time

        for each user:
                for each entry in df_old with user:
                if entry$mode  == "create" & entry$event == "start" :
                                set initial time to entry$time
                                set df$event = "Password Created"
                if entry$mode == "Login" & entry$event == "Successful":
                                set final time to entry$time
                                set df$event = "Login Success"
                if entry$mode == "Login" & df_entry$event == "Failure":
                                set final time to entry $time
                                set df$event = "Login Fail"
                set df$time = final time – initial time.
                Set df$user = entry$user
                Set df$scheme = entry$scheme

AnalyzeData (logFile ):
        let df be a data frame with the values user, scheme, event and time with the
corresponding entries from
        logfile

        for each user in the df:
                calculate the total number of logins (failed and successful) as well as the time
        taken for each type
                of login,  for as a row in a data frame totals.
```

calculate Descriptive statistics by taking
1) the mean number of login events, total, failed and successful
2) the mean time taken to login in successfully and failed logins
3) take a t-test for success vs failure

let success be a data frame that stores all successful logins with their time taken.
let failed be a data frame that stores all failed logins with their time taken.
calculate the statistics for each scheme's successful / failed logins
return a dataframe with all totals, success and failed.

GetGraph(scheme1, scheme2, scheme3):
Create a histogram from the sum of logins, failed and successful from total data frame for each scheme
Create a histogram and boxplot for the time taken to login (success or fail) from success and failed
dataframes for each scheme
Create a boxplot for time taken for each scheme.

Results

Within the following section, one can expect to find our analysis of the three scheme logs provided. Through the implementation and design of a software process it was possible to use the provided log files for descriptive analysis.

Number of Logins per scheme

Text 28

The following table describes the Mean, standard deviation and median for the number of attempted, total, successful and failed logins that occurred using the Text28 Scheme. This number take the average of all users' data. The average number of successful logins found is 87% while the average number of failed logins is 13%.

	Total # of Logins	# of Successful Logins	# of Failed Logins
Mean	16	14	2
Standard Deviation	4	1	4
Median	16	15	1

The frequency of all logins shows that on average 8 users attempted to login 10-20 times and 1 user attempted to login 30 times. The frequency of successful logins for the

greatest group of users was 15, and the majority of the users had a frequency of 0 failed logins. The histogram below shows the frequency of successful logins (green) and failed logins (red). It shows that failed logins occur early on and successful logins occur over time. This suggests that with the Text 28 scheme the majority of the time users were able to recall their password and successfully login. A t-test taken to check for significance between successful logins and fails showed no significance.

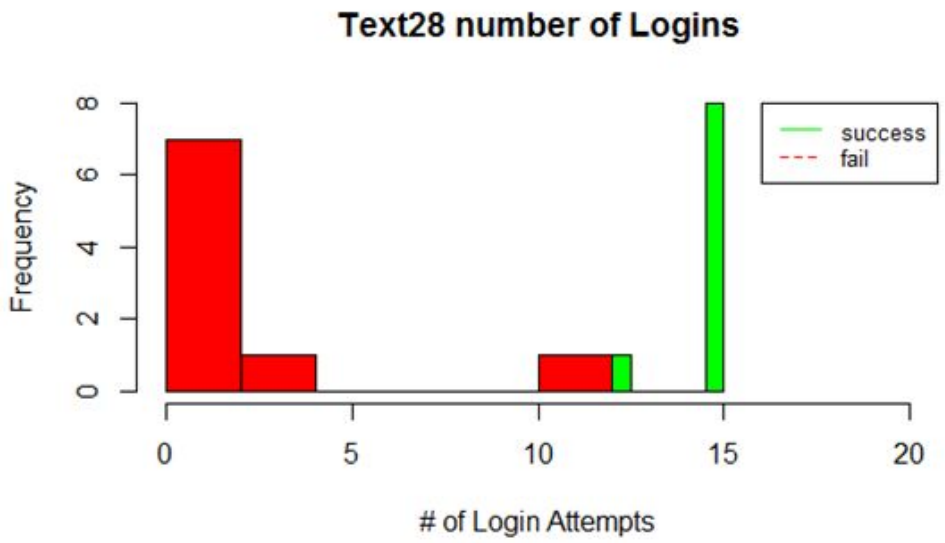


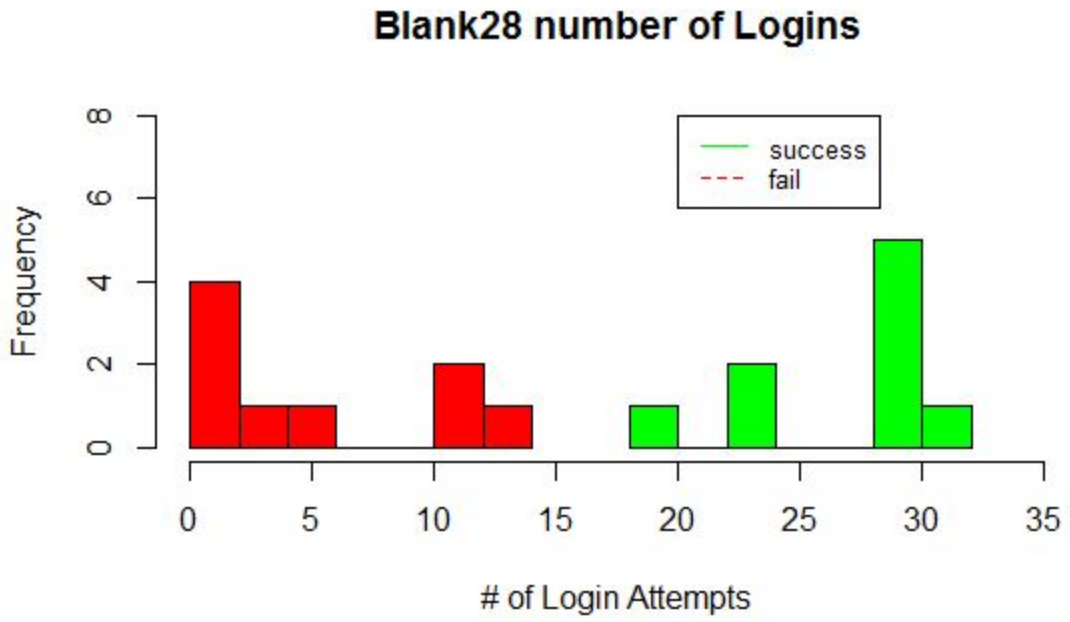
Figure 6 Histogram for Logins Text28

Blank 28

Similar to the table from text 28 the following table represents number of attempted logins, total, successful and failed. Where the average number of logins per user was 33, 90% of the logins with this scheme were successful and 5% of logins were failed. This is not statistically significant and therefore the difference between the Blnak28 and Text28 scheme regarding number of logins (total, successful and failed) cannot be assumed.

	# of all Logins	# of Successful Logins	# of Failed Logins
Mean	33	27	5
Standard Deviation	8	4	5
Median	34	30	4

The following histogram in figure 7 show the frequency of the average number of login attempts for successful logins and failed logins. The majority of users attempted to login between 20 and 45 times. All users had over 19 successful logins, and only 3 users had more than 10 failed logins. A t-test taken to check for significance between successful logins and fails showed no significance.



.

Figure 7 Histogram for total Logins Blank28.

Image28

The table below describes the number of attempted logins (successful and failed) for the Image29 scheme which had an average of 18 attempts. On average 83% of users login attempts were successful, and therefore 16% of user login attempts were failures. These numbers do not show statistical significance between either the Text28 or Blank28 schemes.

	# of all Logins	# of Successful Logins	# of Failed Logins
Mean	18	15	3
Standard Deviation	3	2	2
Median	17	15	2

Similar to the other schemes the following three histograms depict the frequency of login attempts on average. The Majority of users attempted 15-20 logins total. All

users took over 10 login attempts to login successfully and had only 8 users logged in successfully every time. The histogram in figure 8 shows again that failed logins occur initially but decrease over time with rehearsal. A t-test taken to check for significance between successful logins and fails showed no significance.

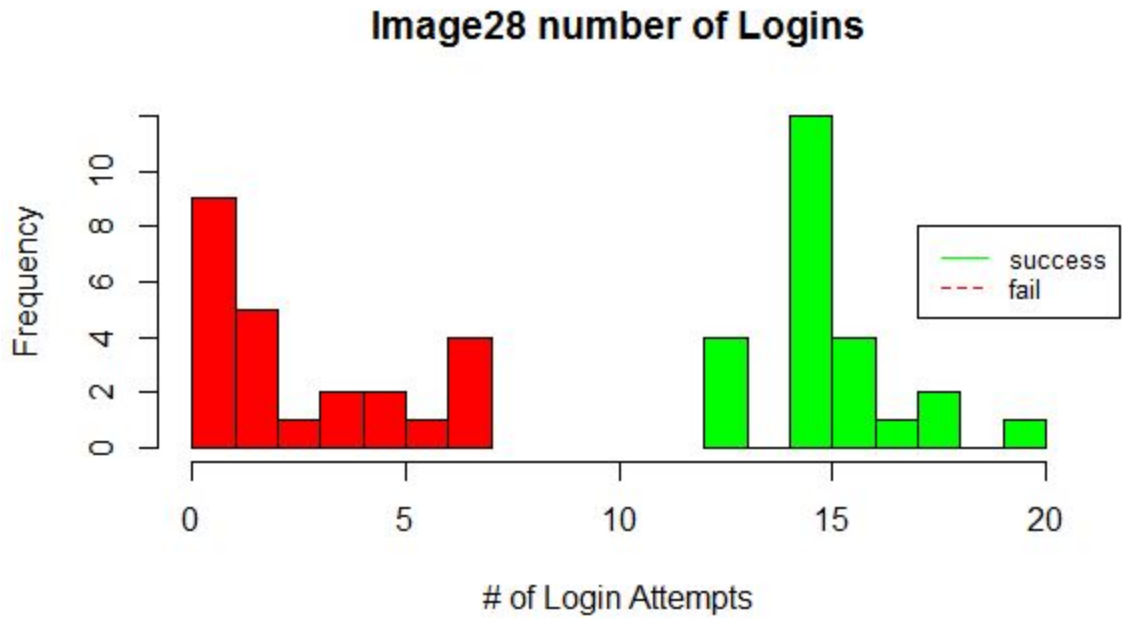


Figure 8 Histogram for Logins Imagee28

Login Time

The following section depicts how much time was taken for each users successful or failed logins for each scheme. The initial time occurs when a password is created, and time is taken for each successful or failed login.

The next Graph is a boxplot that shows the distributions of successful and failed login time per scheme. The text28 scheme distribution shows that failed logins occur early and decrease over time, as they become successful logins. The boxplot (figure 9) for successful logins for blank28 shows a distribution where the majority of users failed within the first two days of learning the password but were successful after three days. The box plot for failed logins for image28 shows a normal distribution, which suggest that an equal amount of success occurs before and after the mean time taken. This distribution suggest that the Image28 scheme is not easy to learn and there is no correlation between time and failed logins.

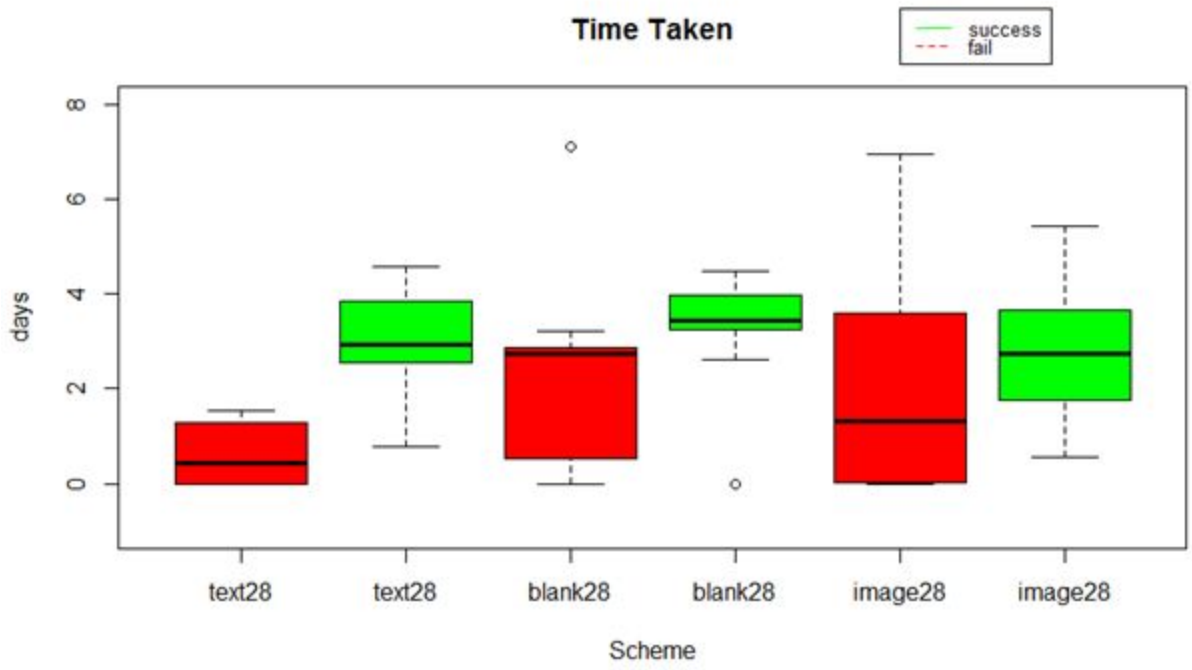


Figure 9 Boxplot for Login Time for all schemes

Text28

The next table shows the time taken, in days, for a user to have a successful or failed login. These statistics show that on average the failures occurred very early on in the testing. The number of successful logins increased over time, suggesting that once the password is learned it is successfully recalled.

	Time taken for Successful logins, (days)	Time taken for Failed logins (days)
Mean	2.90	0.65
Standard Deviation	1.31	0.69
Median	2.92	0.44

The following graph is a histogram that show the time taken for a user login success or fail. The frequency of successful(green) shows logins occurs between 2 to 4 days. The histogram for time taken for a failed login shows that a failure to login occurs in the first 2 days of learning the password, but as time increase failed logins decrease. The darker green sections are where there is overlap in the frequency of successful and failed time. Between zero and two days there are greater numbers of failed logins than

success, but as time increases there are more successful logins and no failed logins. Similar to the number of logins analysis this suggests that password recall increases over time and through rehearsal.

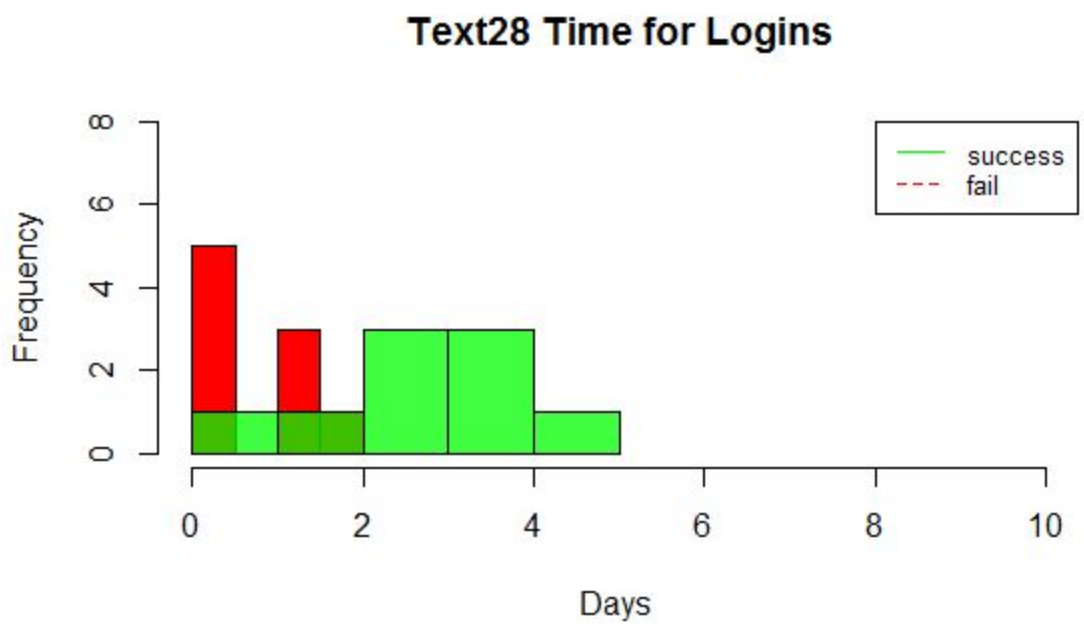


Figure 10 Histogram for Login Time Text28

Blank28

Next, is a table that describes the time taken on average for successful and failed logins for the Blank28 scheme. The average time taken for a successful login is 3.22 which is longer than the Text28 scheme, but not long enough to be statistically significant.

	Time taken for Successful logins (days)	Time taken for Failed logins (days)
Mean	3.22	2.21
Standard Deviation	1.32	2.23
Median	3.44	2.73

The following histogram show the Time taken to login successfully or not in terms of frequency. Time taken for a successful login occurs with the greatest frequency of 3 days. Failed logins show that failed logins occur right away but decrease over time. There is one occurrence where a user forgets after a longer period of time, but the box plot (discussed later) suggests that the occurrence is an outlier.

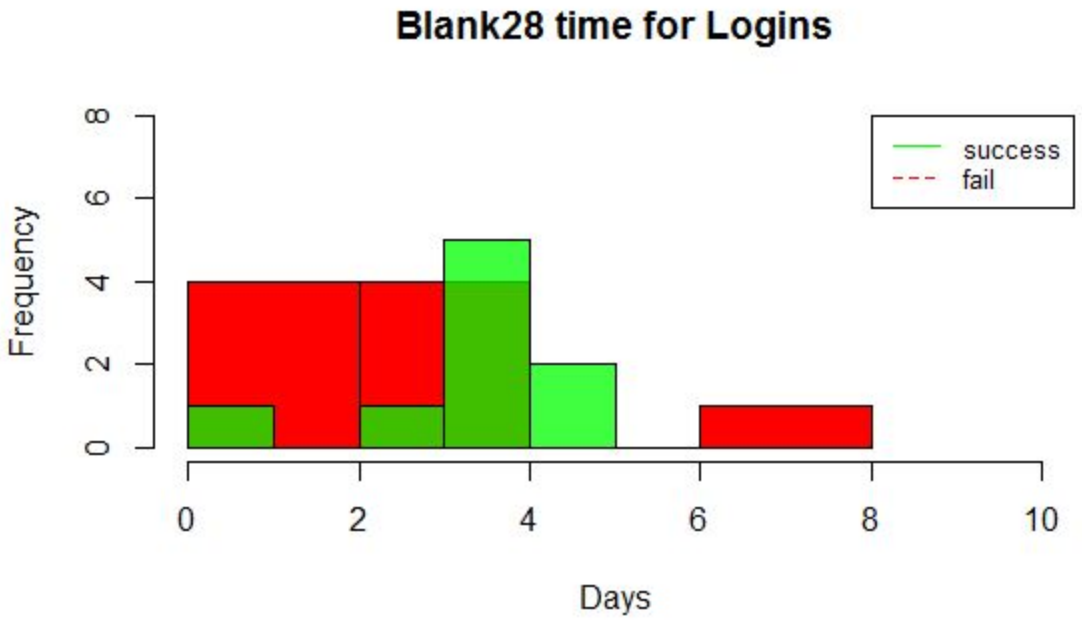


Figure 11 Histogram for Login Time Blank28

Image28

The next table shows time taken for successful and failed login times for the Image28 scheme. The average time is 2.77 which is slightly faster than the text28 and blank28 schemes, however the correlation is not statistically significant.

	Time taken for Successful logins (days)	Time taken for Failed logins (days)
Mean	2.77	1.96
Standard Deviation	1.24	2.18
Median	2.73	1.30

Next is the histogram of the image28 scheme regarding time taken for average success or fail login time. The histogram for failed login times, shows that a fail can

occur up to 7 days after the password is created, with the majority of fails falling after 2.7 days. The graphs suggest that this scheme is not easily recalled over time. This is shown as the frequency of failed logins decreases with the number of days, but never becomes zero as it does with the other two schemes. Successful logins peak at 3 days and then decrease over time.

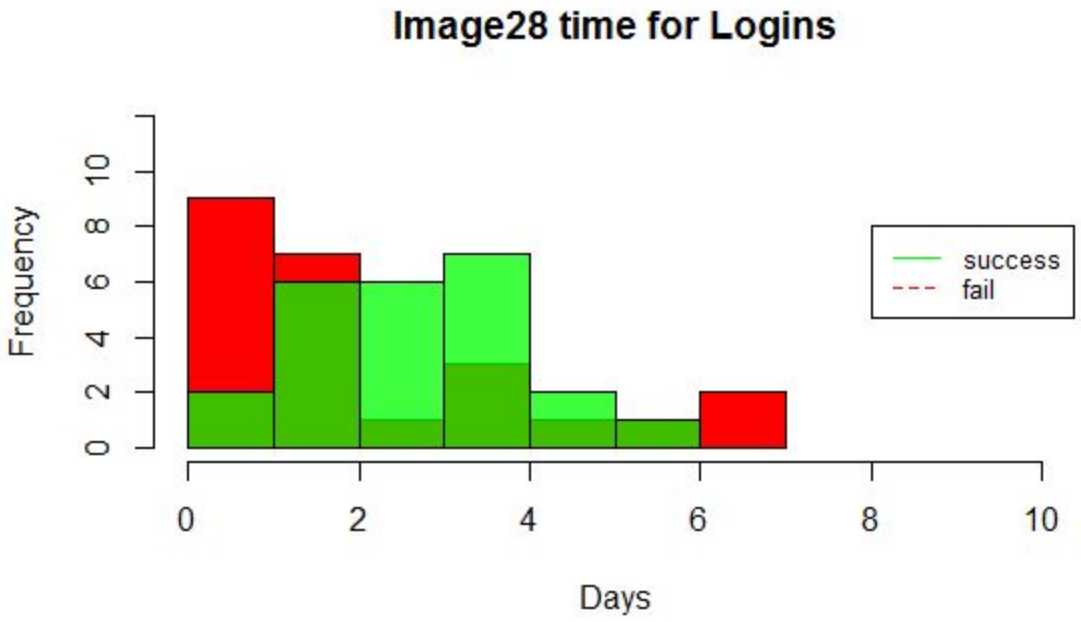


Figure 12 Histogram for Login Time Image28

Sites

Along with descriptive statistics for each scheme, the site used to login to the password trainer was looked at for each scheme. Each users could login to any of the schemes form one of three sites (wvacation, voteforyou and studentlife). Below are bar graphs that show the success and failure of each site per each scheme. There appears to be no statistical significance between the users site and their ability to login to the system.

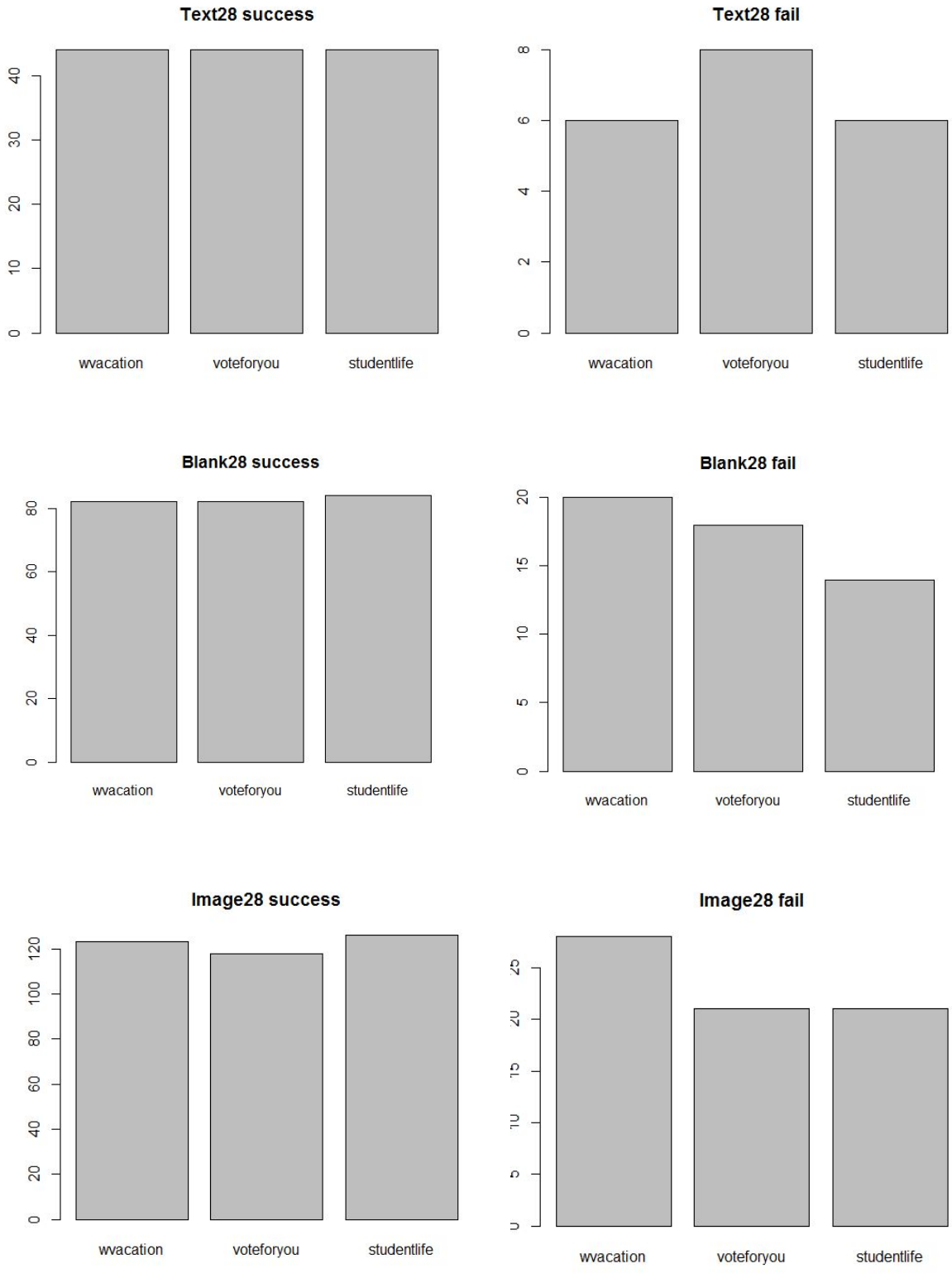


Figure 13 Login Sites Success and Fails

Conclusion

The Statistical descriptions and graphs all suggest that there is no significant difference between number of successful or failed logins. Comparing the results for each scheme, Blank28 had the highest number of logins total, as well as the highest percentages of successes and fails, but took slightly longer than the other two schemes. Each scheme showed a positive correlation between frequency and successful logins, along with a negative correlation between frequency and failed logins. Suggesting that repetition helps recall of passwords. The text28 and Blank28 schemes had similar correlations with successful logins over time. Whereas the Image28 scheme had a normal distribution for successful logins, implying there is no correlation between successful logins and time. Initially after learning a password users often have more failed than successful logins. Though over time successful logins begin to occur. Though the blank28 scheme does seem to take users slightly longer to login successfully, there is no significant difference between any of the schemes regarding time taken to login successfully. Regarding the site used there was no conclusive evidence that one was better from any of the others. From these findings, it can be concluded that no scheme is significantly different than the other, and there is no advantage or disadvantages to one over the other.

Design, Implementation, Statistical Inference

Design Rationale

Our design was carefully thought out based on a few ideas we found interesting. The use of emojis as the main visual component of our system was a clear, strong choice. The cute and poignant glyphs are ubiquitous on everyone's devices (barring a few exceptions, like emojis that were introduced with Unicode 9.0 that many older Android devices still don't support, and likely never will). Even if people don't make regular use of all ~2,000[1] emojis on a regular basis, they are easily identifiable and recognizable due to excellent design by Apple and others.

Part of our design process involved favoring a larger grid over a longer password. Naturally, a bigger grid gives us more entropy. We decided that we would roughly match the width of a typical mobile software keyboard. The iPhone's "home row" is 9 characters wide. At first we considered a 9x9 board, but hypothesized that using an even number of grid cells would open up some potential for mental chunking. For example, we could draw each quadrant of our 8x8 grid with a different background colour or simply with slight separation. We did not test this hypothesis, though it may be an interesting future research topic.

Having a larger board allowed us to request a shorter password from the user, only 4 emojis long. This preference derived from our belief that it would be easier to remember a password that required more searching for the answer, but only 4 pieces of an answer. The searching through the 8X8 grid for an emoji, although increasing the time needed to enter the password, was believed to also help you remember by having you focus on that emoji and the graphical space for a slightly longer period of time.

Emoji Selection

The process for picking emojis was to divide the total macOS/iOS set in half and pick a few from each category that were visually distinct. For example, we didn't pick the both male and female doctors, just one of each gender, alternating genders for things like age and profession. We also tried to avoid picking emojis that depend too much on colour. We picked a limited selection of flags, avoiding picking nearly indistinguishable flags like the Italian and Irish flags. Our initial pass over the Apple set yielded roughly 320. In our selections we made a conscious effort to choose happy, funny, and generally positive emojis. We took care to include hugely popular glyphs like - , ≈ , and, ƀ . Even though most passwords won't contain these characters, they may appear in close

proximity to the true password and could potentially provide additional queues for long term recognition. (i.e. the pink double hearts just below the fire emoji). This idea could backfire and make for more guessable passwords if users were to chose their own passwords, but since ours are randomly generated, it is irrelevant.

After we combined our favourite picks, we removed a significant number of accidental duplicates, removed a few that were visually similar between our divisions, filtered out particularly boring picks (scissors, a spoon, an open textbook, and some others we already forgot about), and removed some potentially profane glyphs, we were left with 234 emojis in our final list. Worth mentioning is that our demo only fully functions on up to date versions of macOS and iOS. Windows 10 and Android phones can only display about 80% of our master list. Substituting images or embedding a font that supports the entire charset would easily mitigate this issue, but it goes to show how hard Unicode support is, especially when the standard moves as quickly as it has in recent years.

Overall, we were very conservative in our choices. Another very interesting topic of study would be the maximum practical alphabet size. We only take advantage of about 10% of the total emoji space, and there's a chance we underestimated human memory and recognition capabilities. It would not be surprising if something on the order of 500 emojis would be just as easy to use to an unsuspecting user.

Password Scheme

The total "alphabet" of emojis is 233 glyphs. These were picked by hand to comb out any that might be too similar. We allow for repeats in the string, so right off the bat we have an upper bound of 233^4 combinations, right around 32 bits. Now, since we don't show the user a 16x16 grid, we have to approximate this level of entropy by randomly shuffling the alphabet for each user.

The grids are generated with a seeded PRNG where the seed is determined by:

seed = Hash(secret, user_id, x, y, grid_number)

Where *x*, *y* are the coordinates that were clicked and *grid_number* is which what grid the user wants. The first grid is represented by $x = y = grid\_number = 0$. Our implementation uses SHA256 as its hashing algorithm. The secret is a single shared value for all clients in our demo implementation, but in a real application it would be better to use a new random secret for each clients and generate a new secret for passwords resets.

We then shuffle a standard list of emojis using the Fisher-Yates shuffling algorithm and use the first 64 “cards of the deck”. Since we shuffle the data, there will

never be duplicate emojis on any single grid, which is important for usability. And since we use deterministic seeded PRNG, they will always be shuffled the same way.

Entropy Estimation

At first glance, it may seem that the total entropy would only be the alphabet size to the power of the string length. We chose an alphabet of roughly 233 visually distinct glyphs, which on its own provides sufficient entropy at a string length of just 4. $\log_2(233^4)$ = 31.45 bits.

But that's not all. When implementing the first iteration of the scheme it became apparent that we were not taking full advantage of the randomness and unpredictable property of the grids. Two users' passwords could possibly, though unlikely, share the exact same glyphs in the exact same order of entry. Even less likely still would be for each glyph to appear on the same coordinates in each user's grid. So, by encoding not only which emoji was clicked, but where in the grid it was for each "character" of the password we gain significant entropy.

Now, each character of a password can be 1 of 233 glyphs, and it can be in one of 64 places. For a password of length n, we get a password space of $(233 * 64)^n$, which for n=4, is 55.45 bits, 2^{24} times more than what was first anticipated and considerably more than the project requirements.

A length 4 password (ɐ ř Ļ ɨ) encoded as plaintext: 164:0,1;202:3,2;203:0,7;125:0,3;

164 is ɐ , 202 is ř , 203 is Ļ , and 125 is ɨ . The remaining numbers are coordinates and semicolons are separators.

While this result is impressive, it does not apply in practice. For one, *anyone* can see the first grid for *any* other user (that is, assuming the server stores the secret and there is no other form of authentication). This reduces the possibilities for the first character of a specific user's password to 1 in 64, which is not good, to say the least. What makes this approach secure is the *secret* portion of the hash calculation. Without it, all an attacker would need to drastically reduce the password space is the username/user_id. From there, the number of possible passwords is reduced to 64^n, where if n=4, the password entropy is only 24 bits. One other obvious weakness is the leaking of the first character of the password. Users would not be able to log in without seeing the first grid, which, by design, must contain the first emoji-glyph of their passphrase. However, simple standard practice rate limiting and attempt capping would severely reduce the feasibility of online attacks. **In short, don't leak the secret.** Without it, there is no way to predict the contents of the next grid. Strong time and/or memory hard hashing mitigates offline attacks as well, as is true for all password systems.

If we assume rate limiting is in place and works in such a way that an attacker can only attempt a miniscule number of passwords before being locked out, our final estimate of the effective password space is roughly 64 * (230 * 64)^3, which corresponds to **47.53 bits**, about as strong as an 8-character long alphanumeric (plus 14 symbols) password.

User Interface

The screenshots below display the user interface and steps of password authentication. Figure 14 displays a potential password entry, Figure 15 displays the login screen where the password is created upon username entry, and figure 16 shows the screen displayed to a user when practicing their password entry.

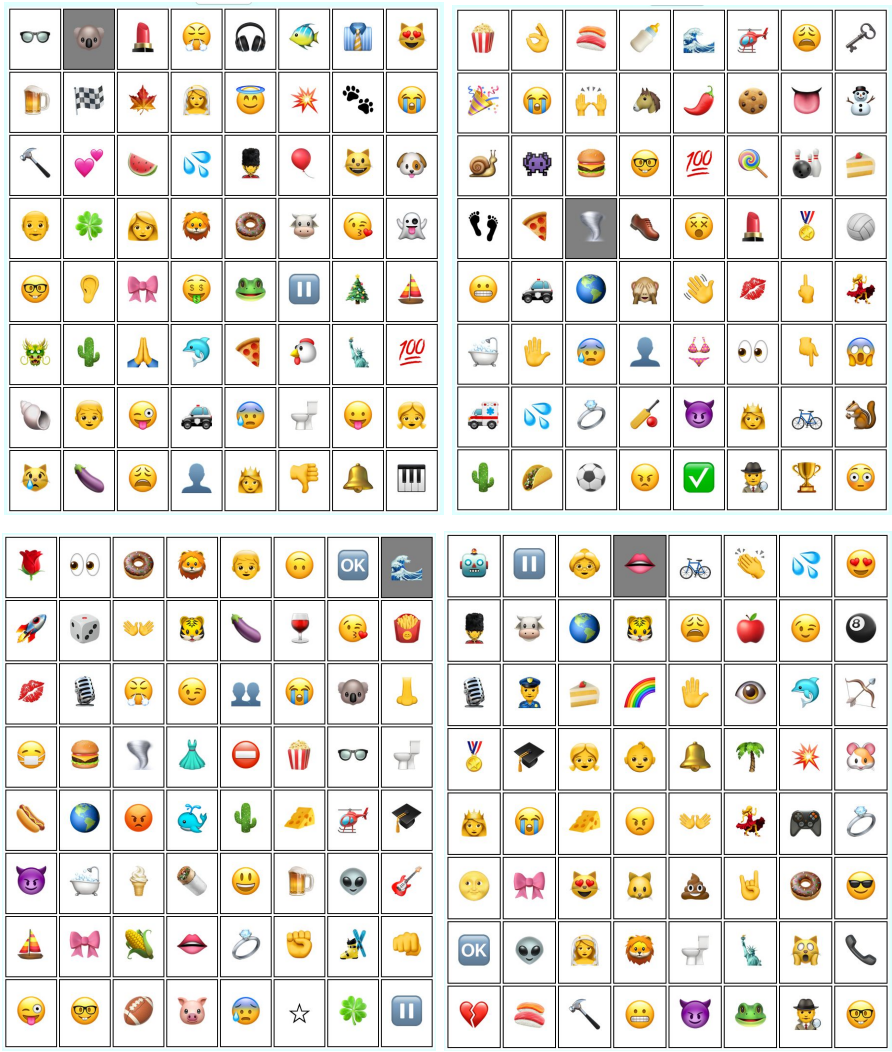


Figure 14, TopLeft, TopRight, Bottem Left, Bottem Right, shows the correct sequential entry of Grids 1- 4.

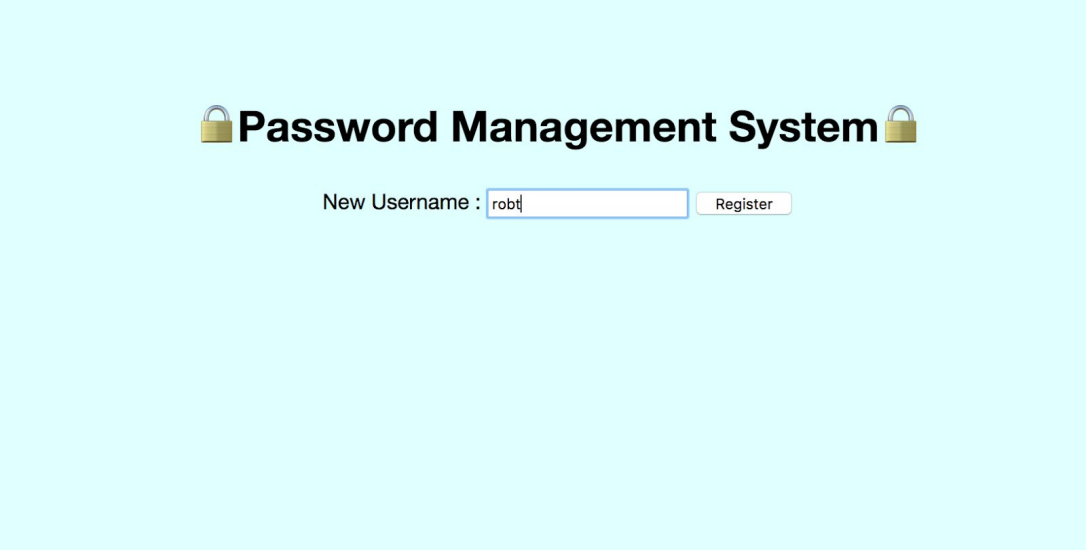


Figure 15 Initial registration screen

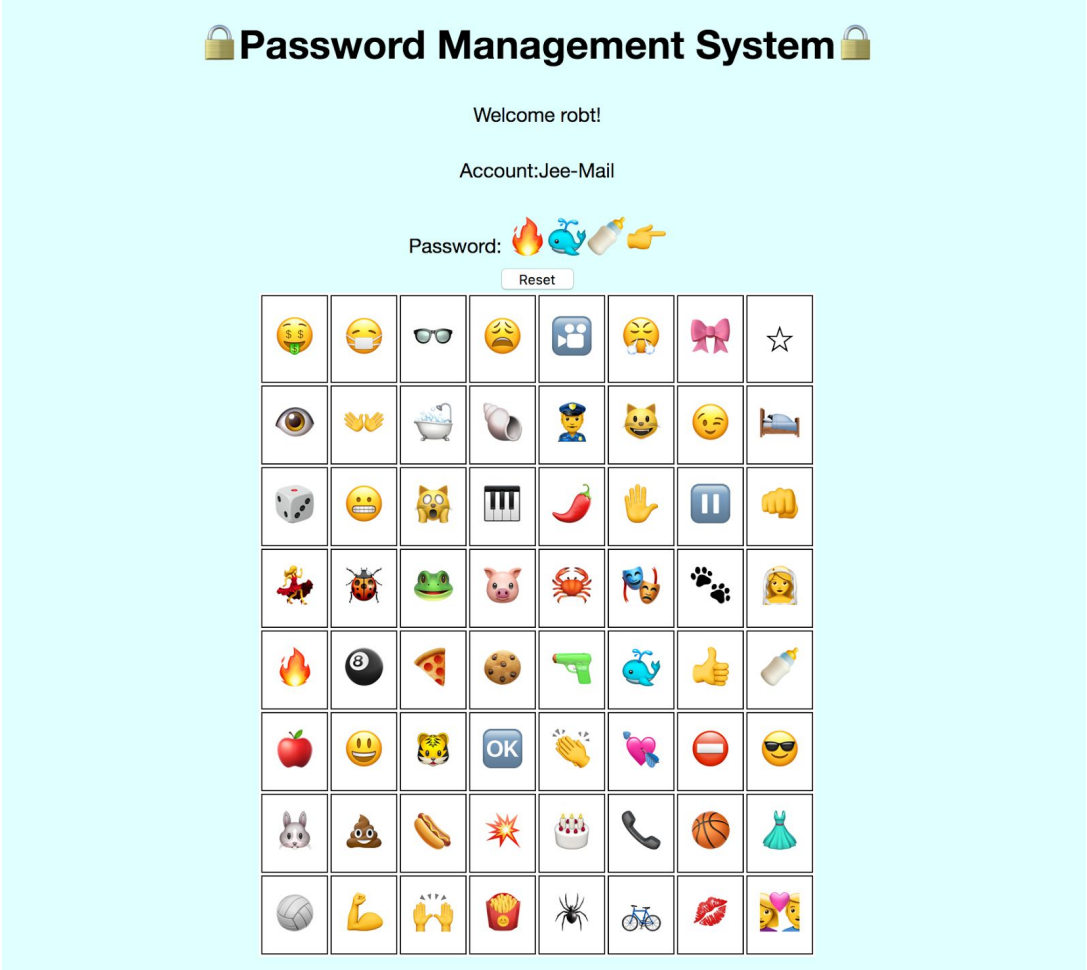


Figure 16 The screen you see to learn your password

The image below displays our choice of emojis, 233 emojis we found to be the most distinct and memorable.

```
var emojistr = ""; for(var i = 0; i < emojis.length; i++) { emojistr += emojis[i] + ", " }
```

```
emojis.length
233
```

fFigure 17, A list of All possible Emojis.

Password Testing Procedure for Data Analysis

The process followed by a user for password entry is as follows:

1. They enter a username to register or login with (behaves the same way for new and existing users)
2. 3 passwords are created for them, including the grid layout of each of the 4 portions of each password. The 3 passwords are uniquely identified by the username and website. The websites are hardcoded to be “Fakebook”, “Jee-mail”, and “Safety-Beeposit”
3. They are shown the website name and first password, and are asked to enter it as displayed. Once they’ve finished entering it, they can click “reset” to practice again, or “ok” to try it without the password being visible.
4. Once they’ve successfully entered it without the password being visible, they move on to the next website’s password, behaving exactly as the one before but with a completely new password and set of grids.
5. This is repeated for the last website’s password, the current website always remaining displayed at the top of the page.
6. Finally the testing round begins, which has the user enter the password for each website in a random order with no prompts. If they fail any of the websites passwords 3 times, they fail and are returned to the registration page. Otherwise they are shown a success message if they guess all three passwords within 3 tries each.

Simple framework for quantitative testing

Survey Questions

Below is a list of questions used to investigate the users perception of the scheme design, along with a link to the survey.

1. I was satisfied with my experience
2. I found the system unnecessarily complex.
3. I found the system intuitive and easy to learn
4. I think that I would like to use this system frequently.
5. I think that I would need the support of a technical person to be able to use this system.
6. I would imagine that most people would learn to use this system very quickly.
7. I felt very confident using the system.
8. I had trouble distinguishing the emoji's
9. I found it easy to select the emoji I was attempting to select
10. I thought my generated password was too long
11. I thought the time taken to login was too long
12. I had trouble remembering my generated password
13. I thought the emoji's made my password easier to remember
14. I found it easy to recover when I make a mistake using the system
15. I found that the emoji's did not help with recalling my password
16. I found that a multiple grid display was difficult and confusing to use
17. I found that the system provided good feedback regarding its status?
18. I found as a user that I had good control of the system?

Link : https://hotsoft.carleton.ca/comp3008limesurvey/index.php/924959?lang=en

Study Results

Survey analysis

The Survey Analysis contains a figure and statistics that represent responses to the survey regarding the Emoji Scheme. You can find a figure for each question that showed significant results as well as a brief description of the results.

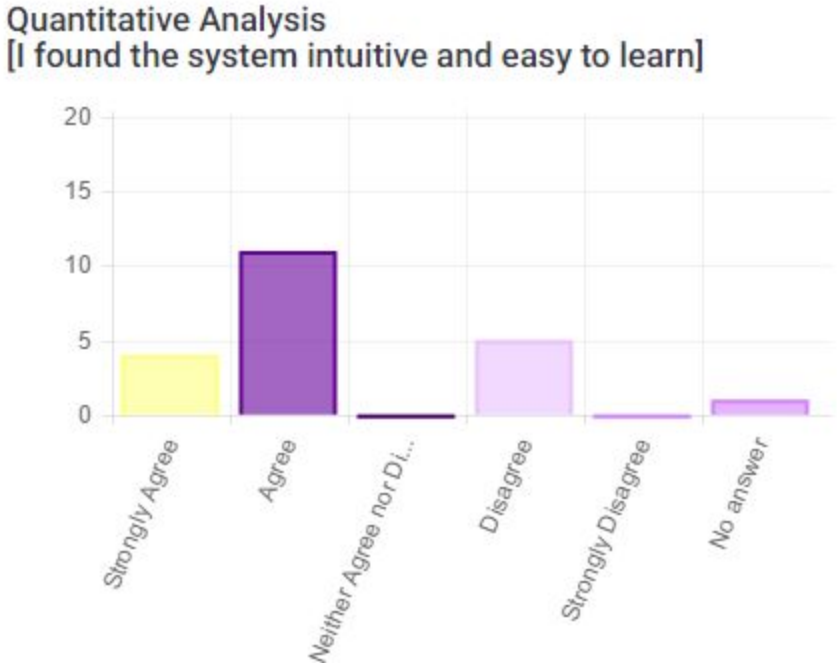


Figure 21 Quantitative Analysis- I found the system easy to learn

Surveys revealed that initial users didn't have trouble learning the system. 71% of users found the system intuitive and easy to learn, with 11 participants voting to agree with the statement, and 4 to strongly agree out of a total of 21 responses.

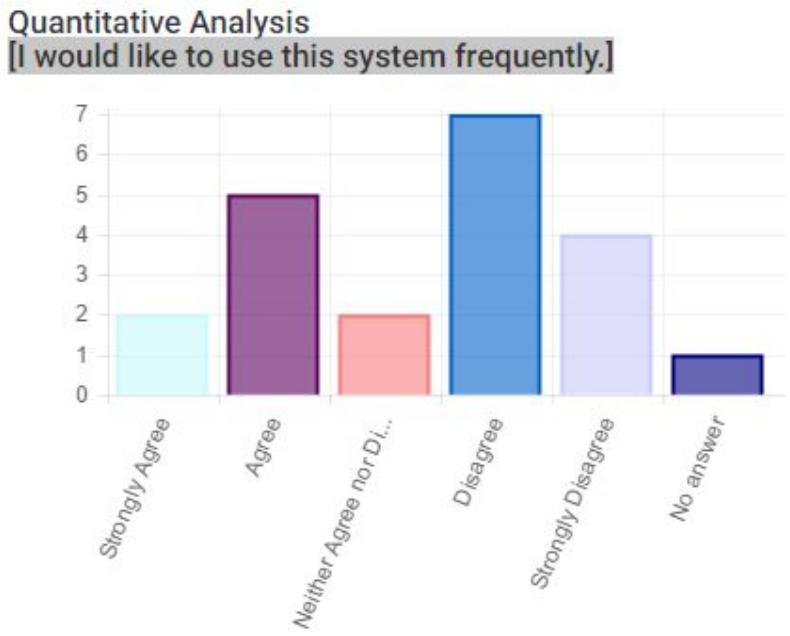


Figure 22 Quantitative Analysis- I would like to use this system frequently

Despite the fact that users found the system easy to learn, only 35% of users voted that they would like to use the system frequently, with 55%, an 11/20 majority voting that they would not like to use the system frequently

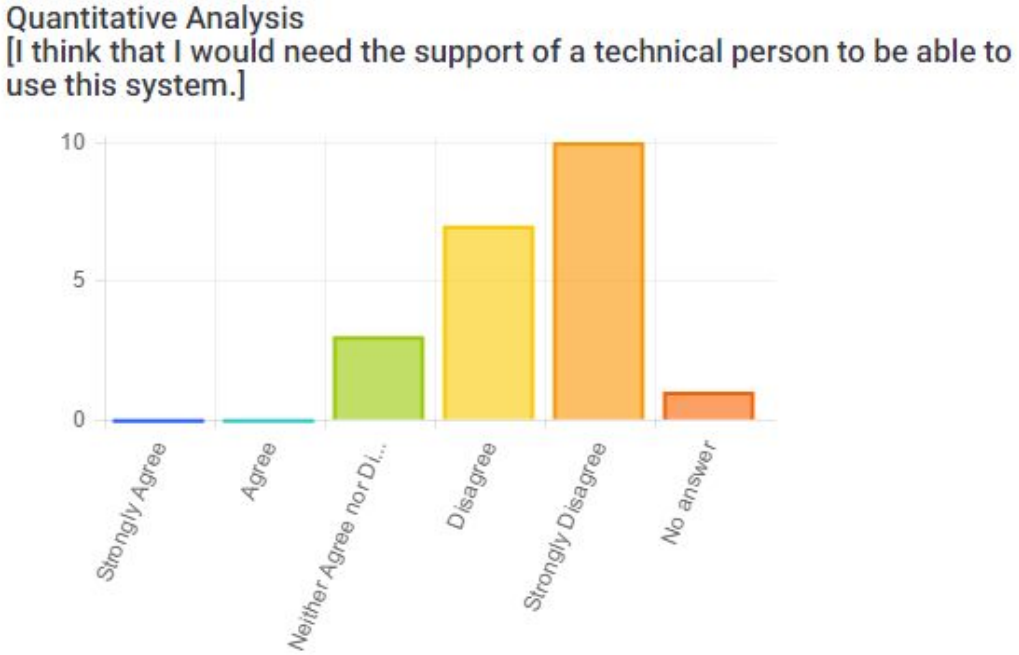


Figure 23 - Quantitative Analysis- I found the system easy to learn

17/21 users, or 80.9% stated that they would not need the support of a technical person to be able to use the system. which infers a strong usability for our system. this is reinforced with the following question which asks;

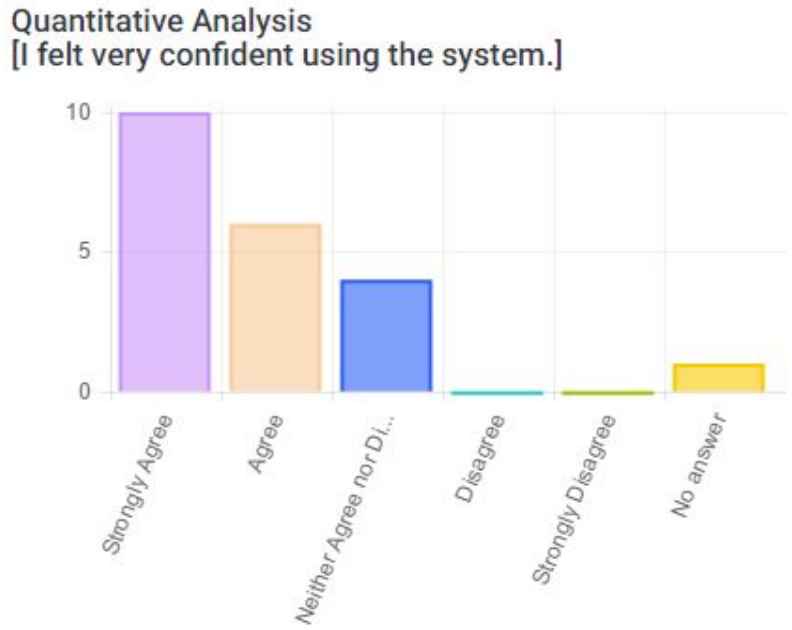


Figure 24 - Quantitative Analysis- I felt very confident using the system

Again, a strong majority of 80% or 16/20 voted that they felt confident using the system. with 0 users saying they lacked confidence in their ability to use the system.

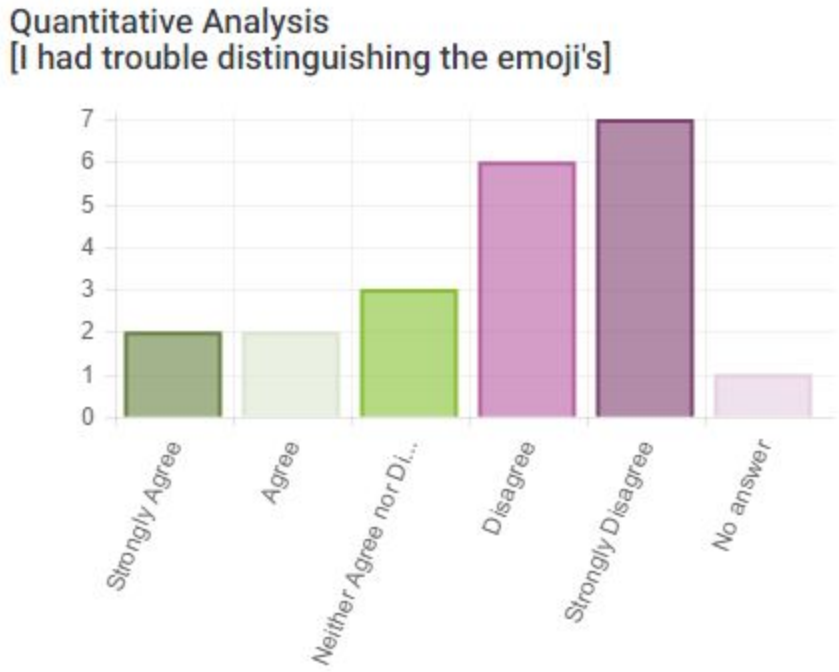


Figure 25- Quantitative Analysis- I had trouble distinguishing the emojis

The emoji design introduced by our system was received well. Users found the emoji system to be easily understood and distinguishable from one another. By comparison 65% of users found that they could easily distinguish between the emoji's while only 20% found it difficult.

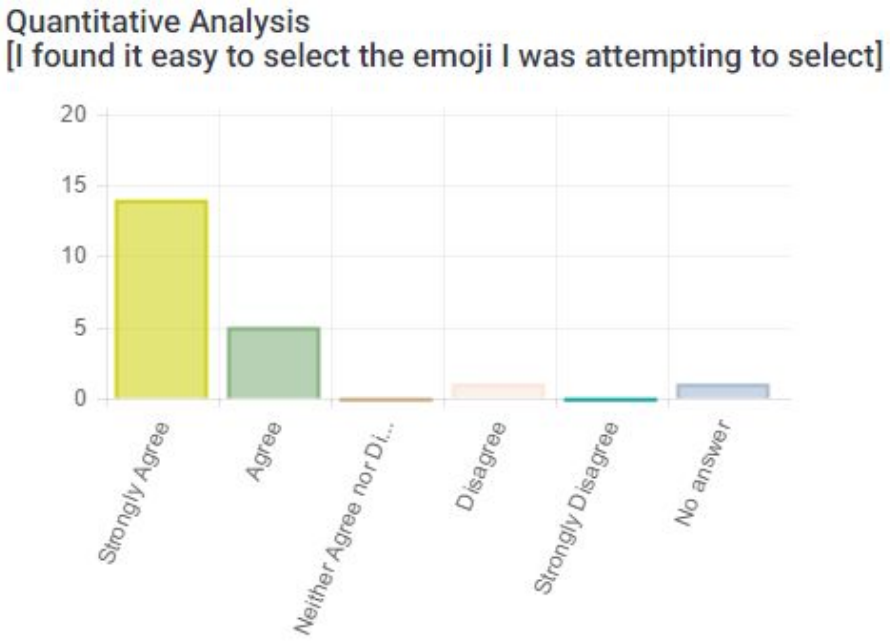


Figure 26 - Quantitative Analysis- I found it easy to select the emoji I was attempting to select

The second point of analysis for our emoji system was our 8x8 grid from which users could select their emoji's. A significant 94% or 19/20 users found it easy to select the emoji they were attempting to select. Which speaks to our 8x8 design being accessible for the vast majority of users.

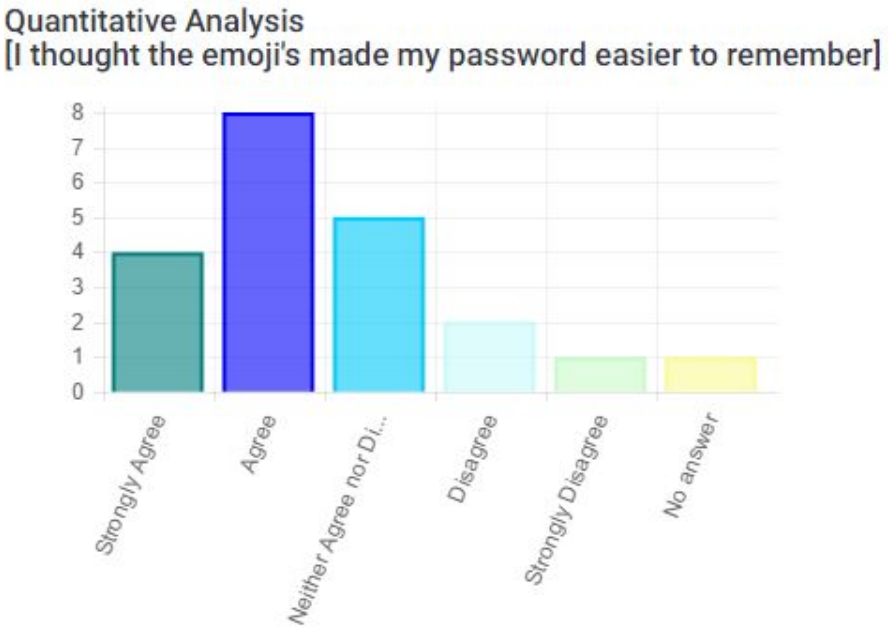


Figure 26 - Quantitative Analysis- I thought the emoji's made my password easier to remember

Only 3/20 users or 15% found that the emoji's didn't make the password easier to remember. with the rest either finding no difference or an improved experience as a result of the emoji's

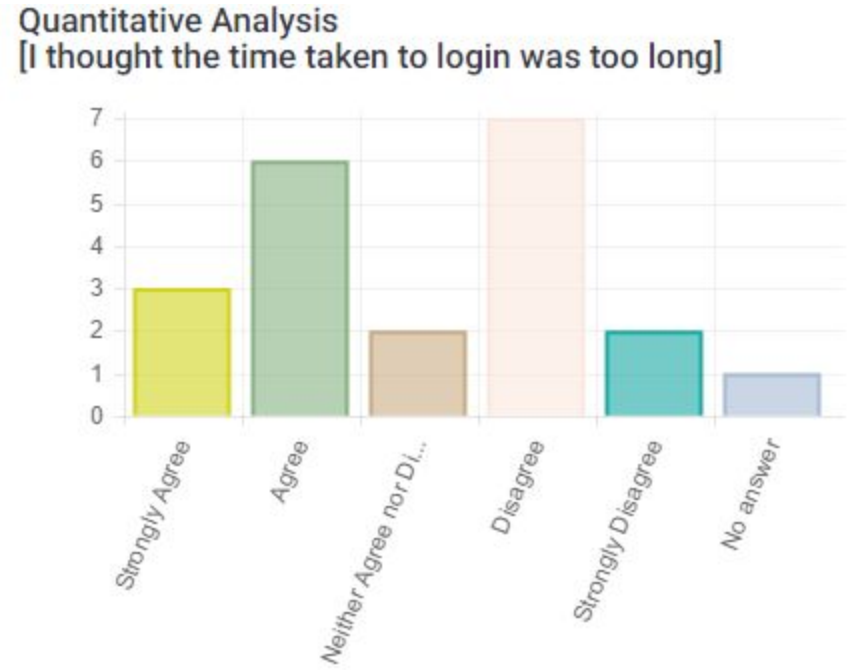


Figure 27- Quantitative Analysis- I thought the time taken was too long

When asked about time taken to log in there was no significant trend in either direction with 9 users voting that they though it took too long, and 9 voting that it didn't take too long.

In conclusion the survey portion of our analysis revealed a number of trends and opinions. The vast majority of users found the system easy to learn and use. Statistically speaking, the emojis were a strong addition. Users showed that they found it made their passwords easier to remember without a significant increase to the perceived time taken. The system we chose to use when implementing the emojis was also well received, with a majority of users stating that they could both differentiate between the emojis and select the one they desired easily. This all speaks to an improvement from a traditional password system. However only 35% of users stated that they would like to use the new system frequently, which suggests further testing and improvements are needed

Scheme Comparision

The Emoji Scheme that we implemented tested user's' ability to recall a randomly generated password for three passwords sites. The table below contains information regarding the number of times a user logged in with their password. On average each user logged in twice with a 50% chance of success or fail. Compared to the Test 28 scheme , the Emoji scheme is harder to recall. However this could also be due to the fact that users rehearsed and attempted the Text28 scheme more before the testing phase.

	# of all Logins	# of Successful Logins	# of Failed Logins
Mean	2	1	1
Standard Deviation	2	2	1
Median	1	0	1

Looking at the histogram for Number of Logins for the emoji scheme vs the Text28 Scheme (figure 6), there is contrast as the later is skewed the right and the former is skewed to the left. Which suggest that the Given more login attempts the frequency of success decreases and the number of fails increase.

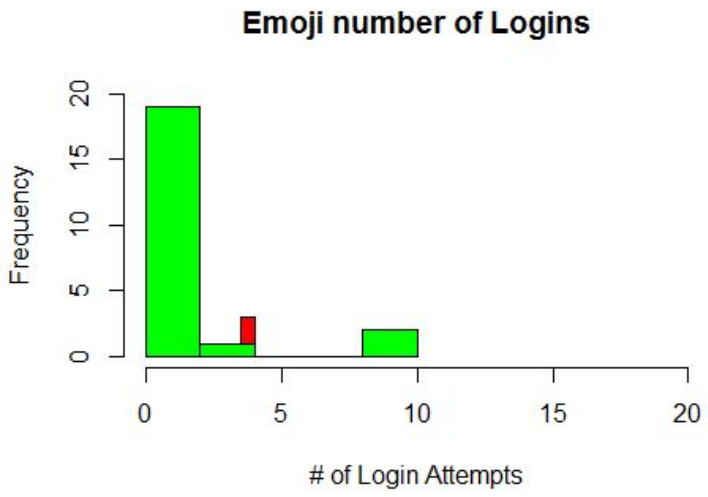


Figure 18 Number of Logins for The Emoji Scheme

The Time taken to login using the Emoji Scheme is shown in the next table. Given that for the emoji scheme all of the testing was done in a single day within a short time frame, the time is shown in seconds instead of days. In general users took longer to login successfully vs failing. With successful logins occurring earlier in time than then failed logins . In comparison to the text28 scheme, users have a harder time to login to the emoji scheme over time. The boxplot for the Emoji Scheme (Figure 20) shows several outliers for the time taken to fail a login, as well as a median of 0 for both failed and successful logins. These results suggest that more user testing is required, both of they system to collect more data and by the users when practicing their password entry.

	Time taken for Successful logins, (sec)	Time taken for Failed logins(sec)
Mean	43	25
Standard Deviation	75	69
Median	0	0

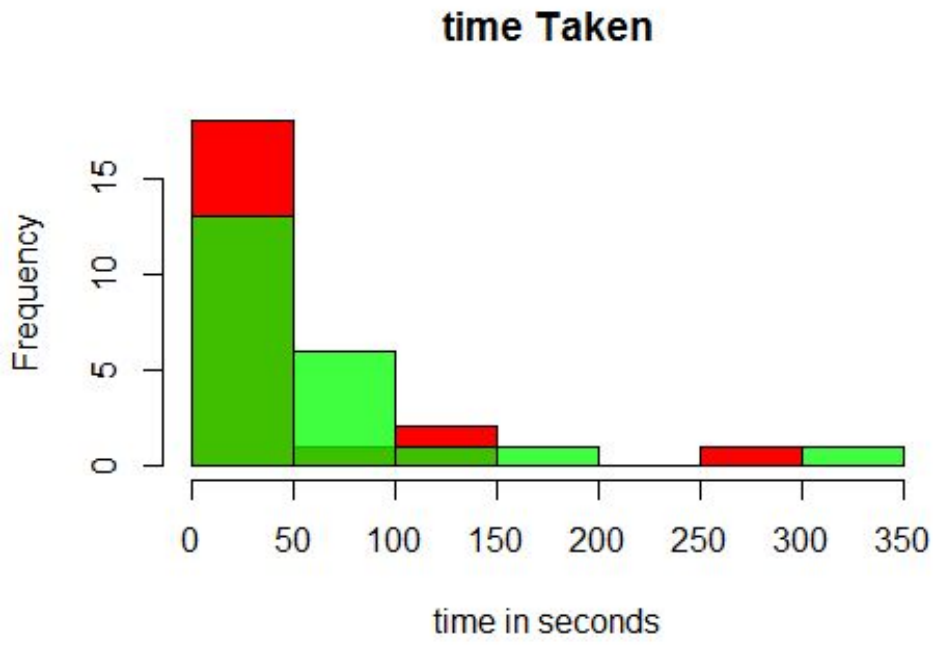


Figure 19 The Time taken for a successful or failed login. Green shows successful logins, green shows failed logins, and the darker green shows where there is an overlap.

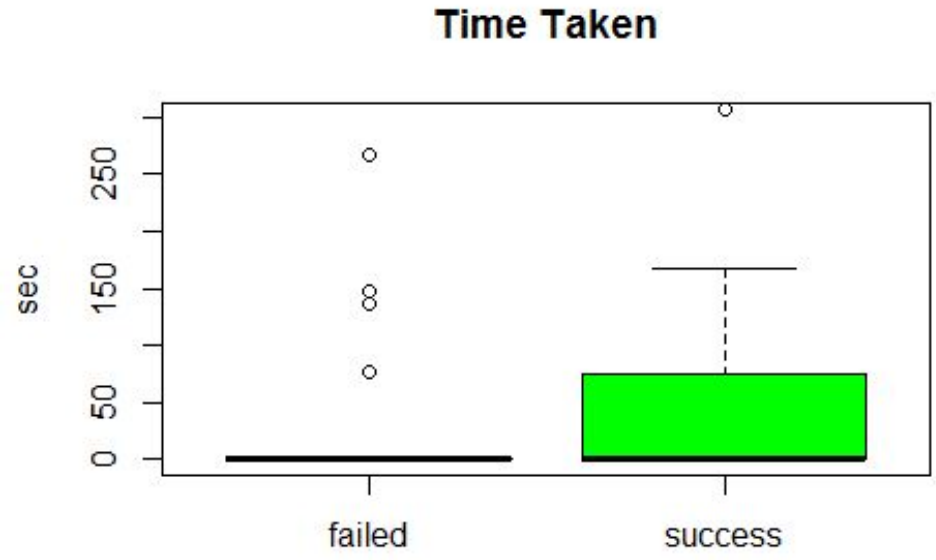


Figure 20 Box plot showing Time taken for successful and failed logins.

Comparing the Text28 scheme to the Emoji scheme, Text28 is superior as it provides a password that is easier to recall given time and rehearsal - which is the exact opposite of the Emoji scheme. However a comprehensible conclusion of the statistics cannot be made for the Emoji scheme as there are not enough login attempts. If you look at the Text28 histograms, it can be seen that early in the testing users had trouble logging into the system, it was over time with rehearsal that users were able to login successfully. Given more testing there is chance that the Emoji Scheme will exhibit a similar pattern.

The possibility of exhibiting a similar pattern becomes more apparent when looking that survey system results, which suggest that users like the Emoji scheme and find it easy to learn. The discrepancies between the logs and surveys could again be a result of limited time and rehearsal, as it is difficult to learn multiple passwords quickly. for more conclusive results it is suggested that more testing is done.

Conclusion

From the testing of our system, it is clear that a large benefit would come from the testing of users creating passwords with our system and rehearsing them over time. Perhaps our system is more oriented to long-term memory as opposed to short-term, which for a password system would be an advantage and not a disadvantage.

After modifying the logging system, collecting participants to complete the survey, overseeing the participants test the system, creating a program to analyse the data collected, then finding the data didn't provide very clear results, we developed a better understanding of how much time and effort goes into descriptive statistic analysis of a program or system. The amount of participants and testing was clearly not enough to provide desired information about the system, and this was something we were not expecting.

The quality of a descriptive data analysis is directly attributed to: the size of the testing domain (users, tests, length of tests or return of existing participants over time), the quality and quantity of inferential statistics for each run of the program, and the representation and computation of these statistics to provide a detailed description of the overall performance of the system.