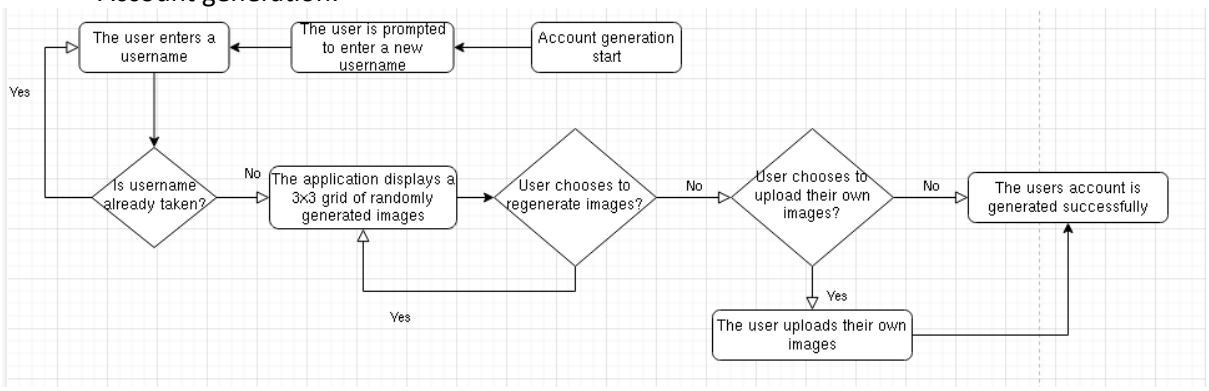
Toby Hothersall

Task 1

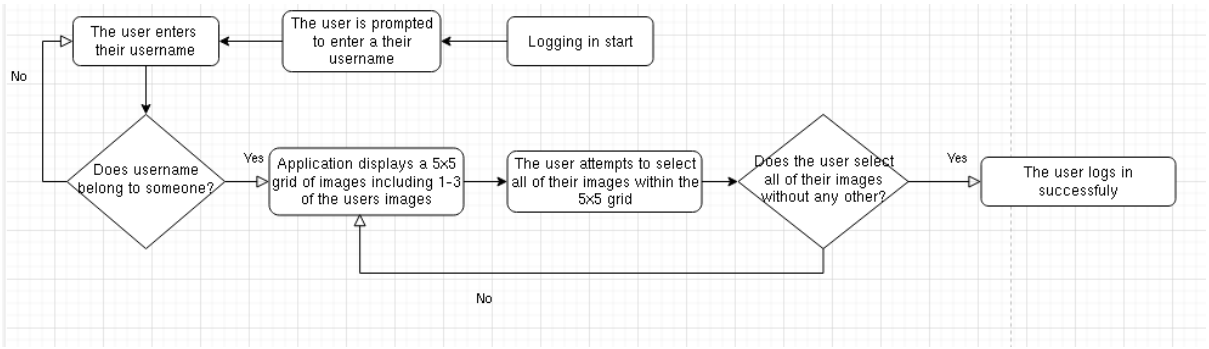
- I chose a recognition-based graphical password scheme where the user has to select 1-3 images from a 5x5 grid to log in.

i)

- Account generation:



- User login:



ii)

- Advantages:
  - Easy to remember: it is recognition based, and recognition is easier than recall so it is easier for the user to remember their password, in the long term, when logging in. Furthermore, by uploading your own images, you can tie emotion from said images to your password making it even easier to remember.
  - Fast login: It is faster to click on 1-3 images than to type out an alphanumeric password.

- Limitations:
  - Shoulder surfing: It is easier for malicious actors to shoulder surf, as you can't hide the entire screen, and it is easier to remember images than alphanumeric passwords.
  - Easy to guess: there is a good chance that a malicious actor could guess the password of the user in a 5x5 grid.
  - Storage space: takes up a large amount of storage space on the servers for the application using the scheme.

Toby Hothersall

Task 2

a) I stored the hash within cwk.txt and used ‘hashcat –identify cwk.txt’ to identify the possible hash functions.

```
toby@Toby-debian:~/.../labs/lab6$ echo 00324b23f59fec72b6cda8a53ea343bf >> cwk.txt
toby@Toby-debian:~/.../labs/lab6$ cat cwk.txt
00324b23f59fec72b6cda8a53ea343bf
toby@Toby-debian:~/.../labs/lab6$ hashcat --identify cwk.txt
The following 11 hash-modes match the structure of your input hash:

      # | Name                                                       | Category
  ======+============================================================+======================================
    900 | MD4                                                        | Raw Hash
      0 | MD5                                                        | Raw Hash
     70 | md5(utf16le($pass))                                        | Raw Hash
   2600 | md5(md5($pass))                                            | Raw Hash salted and/or iterated
   3500 | md5(md5(md5($pass)))                                       | Raw Hash salted and/or iterated
   4400 | md5(sha1($pass))                                           | Raw Hash salted and/or iterated
  20900 | md5(sha1($pass).md5($pass).sha1($pass))                    | Raw Hash salted and/or iterated
   4300 | md5(strtoupper(md5($pass)))                                | Raw Hash salted and/or iterated
   1000 | NTLM                                                       | Operating System
   9900 | Radmin2                                                    | Operating System
   8600 | Lotus Notes/Domino 5                                       | Enterprise Application Software (EAS)

toby@Toby-debian:~/.../labs/lab6$
```

b)

i. I tried MD5 first because it is the most likely out of the given list of hash algorithms from part a.

ii. I used ‘hashcat -m 0 -a 0 cwk.txt rockyou.txt’ in order to crack the hash. The flag -m specifies the hash mode, and I selected MD5, and -a specifies the attack mode and I chose the straight attack mode (a dictionary attack). cwk.txt stores the hashes that I want to crack (in this case just one), and rockyou.txt contains the dictionary to check the contents of cwk.txt against.

iii. The below screenshot shows the plaintext using the –show flag.

```
toby@Toby-debian:~/.../labs/lab6$ hashcat -m 0 -a 0 cwk.txt rockyou.txt --show
00324b23f59fec72b6cda8a53ea343bf:netgear1
toby@Toby-debian:~/.../labs/lab6$
```