CO3120: The Computing with Management Project

**Dissertation**

**Graphical Passwords: Prevent Shoulder Surfing**

Teddy Taylor

Student No: 179047462

May 2020

Supervisor: James Hoey

Second Marker: Hongji Yang

Word Count: 9584

Department of Informatics, University of Leicester

**DECLARATION**

All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people’s work have been specifically acknowledged by clear citation of the source, specifying author, work, date and page(s). Any part of my own written work, or software coding, which is substantially based upon other people’s work, is duly accompanied by clear citation of the source, specifying author, work, date, and page(s). I understand that failure to do this amount to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Name: Teddy Taylor

Signed: Teddy Jonathan Taylor

Date: 02/05/2020

**Abstract Page**

Businesses are constantly conscious on upgrading their security systems due to the developing methods of theft of personal information online, however through research I have discovered not many systems protect from shoulder surfing attempts. Businesses have evolved from using simple one time use string passwords and shifted towards more modernized alternatives with graphical passwords in order to add an extra layer of security for their consumers. Graphical passwords are proven to reduce the vulnerability of falling victim to shoulder surfing and poor memory as images are often easier to remember than string passwords that enforce special characters. The purpose of the graphical password to project was to make a unique password system which gives the benefits of being able to defend from shoulder surfing attempts while causing minimal hassle for the consumers experience as a complex graphical password system may counteract the purpose of the project as the user will struggle to login. Finding this balance between complete security from shoulder surfing and other methods while maintaining a friendly user experience is key to this project, while also beating the competition whose systems only use simple passwords with no graphical elements and awareness of shoulder surfing attempts. My research showed that while graphical passwords were being developed, they were not used as often despite their clear benefits, and large companies were still relying on email verification, which would also be easily breached by a shoulder surfing attempt. In order to reinforce the benefits of graphical passwords, I have created a graphical password system which generates a random 4x4 grid with similar interactive looking images which the user upon creating an account will have selected an image and a direction which when logging in the user will find their desired image, follow the direction (directions range from North, South, East, West, North West, South East) they chose either by 1 block or 2 depending on the option they have chosen and select that image following those instructions to verify their login attempt – the user’s email will be notified a successful login attempt has occurred with the timestamp – while all their information is kept encrypted and secured on the database. To conclude, this software when integrated into a company’s existing IT applications will be a useful tool to prevent shoulder surfing attempts.

**Table of Contents**

1. Introduction…………………………………………………………………………….
   1. Aims and Objectives……………………………………………………………..
   2. Challenges………………………………………………………………………...
2. Survey of Literature/Information Sources…………………………………………..

2.2 Background Study………………………………………………………………..

1. Structured Requirements…………………………………………………………….
   1. Essential Requirements Measurements……………………………………….
   2. Optional Requirements Measurements………………………………………..
2. Planning and Timescales…………………………………………………………….
3. System Design………………………………………………………………………...
   1. Architecture (MVC)……………………………………………………………….
   2. Technical Specification…………………………………………………………..
4. System Implementation………………………………………………………………
   1. Algorithms & Testing…………………………………………………………….
5. Critical Appraisal………………………………………………………………………
   1. Evaluation of Aims………………………………………………………………..
   2. Comparison of other Graphical Passwords……………………………………
   3. Commercial Impact……………………………………………………………….
   4. Personal Development…………………………………………………………..
6. Conclusion……………………………………………………………………………..
7. Bibliography and Citations…………………………………………………………...
8. **Introduction**

For this Project, I have chosen to focus online companies which use outdated password systems as my domain. I have decided I will create a graphical password system which will improve on these pre-existing methods of security and prevent any shoulder surfing attempts. I hope to build a functional software which balances both the graphical password elements while maintain a user-friendly experience. I will be using Java, JavaFX with MVC frame while using an SQL database as these tools will help me achieve my projects goals. The user will have a string password along with choosing an image and direction, upon logging in will use this image along with the direction and find the correct image in the 4x4 randomised grid to verify their login attempt, this software will also have several other features to maintain user friendliness such as a password recovery option and email notifications.

* 1. **Aims and Objectives**

For this segment I will now outline the original aims and objectives for my project that were set in the beginning that I set out to achieve.

**Aims**

The major aim of my system was to deliver a fully functioning graphical password system containing a random element which is impenetrable to shoulder surfing attacks by using a 4x4 grid and images which a user would select an image in a random grid along with a direction and confirm their login. As the project was being developed, I noticed my aims had updated as the project required more goals that needed to be reached in order to make a successful system. A new aim that developed was a password recovery system, I noticed that due to the nature of the project having two passwords (One String, One Graphical) consumers could not only forget the image they chose but also the direction and how many times the image moved (1 or 2). Another new aim that developed was a method to notify the user outside of the system via email, having this feature would enable users to know if their account had been accessed giving more security and awareness. My secondary aims of the system were to have all data encrypted so that passwords are secure in case of backend attacks or data manipulation attempts. Another important aim was to have the whole project user friendly and not hinder the experience of logging in with overcomplicating the login system, the goal was to have the system easily operational while also completely secure, to aid this aim I created a tutorial slideshow in the create account page to help anyone still confused on how the system operated. Another secondary aim was to make sure the software did not run into any bugs with information, users must be forced to select a image and password and cannot leave any of the fields blank, along with when closing a page and moving to the next one, the other page must close and the project must flow smoothly.

**Objectives**

These objectives will aid me in achieving a system which accomplishes the requirements and expectations of the project. Objectives are vital parts of any project as they measure the success of a system and can always be looked back on to make sure nothing is missed. In order to make my project successful, the system must meet these objectives:

* Create login page with suitable labels and working buttons in JavaFX.
* Create the ability for users to make an individual account with their selected images and directions saved in a database.
* Creating a complex algorithm to form a random grid for the graphical password in Eclipse.
* Create a view grid of images in JavaFX for users to select.
* Testing working login requirements in Java.
* Create a ‘create account’ page for users to make logins and unique passwords.
* Create the successful login page and the unsuccessful login page in JSPs.
* Create option for users to select directions for the random grid.
* Create a backend database which increments ID’s as accounts are created, along with storing usernames, passwords and direction chosen which information will be encrypted.
* JavaFX pages should load when selected by the user.
* All JavaFX graphics need to be clear and fitting for the pages – tutorial video must only play when the user selects so.
* Users must pass two security gates for a successful login – a normal string password and selecting the right image. If one of these is incorrect – user may not login.
* Learn more about JavaFX graphics/interactions.
* Learn more about password encryption.
* Develop sound programming practise so that functions and methods do not get too confusing from a developer perspective – thus defeating the purpose of having a secure program from attackers.

Over the course of development I realised that in order to complete my aims I was missing a few key objectives as I noticed the system was still open to backend attacks and the system was not as user friendly as I had hoped therefore I took initiative and created new objectives to help solve these issues. This list updates any changes made to my initial objectives:

* Encryption of passwords, as shoulder surfing will be useless if not protected within too.
* Cultivate understanding of what it means to create a user-friendly software – contrasting colours – spacing between graphics (considering as the project is working with a grid containing images) and resolutions of images being easy to read and easy to navigate while also maintaining a high level of security (balance in this trade off). With this comes a greater understanding of JavaFX.
* Full understanding about security from a software perspective.
* Some form of password recovery – security questions for user to reset either image or string password.
* Make the images clearer and look similar as my researched showcased that images that look the same are not easily distinguished at first sight, aiding the user while hindering the shoulder surfer.
* Develop an understanding of how this system will operate in a management environment.
* Comprehend how other graphical passwords operate and how my project compares, advantages and disadvantages.
  1. **Challenges**

I expected to be challenged by choosing to develop this project as my technical ability is at a postgraduate level, so developing a fully functioning system which takes users input and interaction into account while trying to make a bug free randomised grid did prove to rise some challenges. One of the major challenges that occurred was once I made the randomised grid I had to figure a way that the algorithm would load the users image and direction and implement it to the grid so it would work every time. Here is the list of challenges I originally came across while in development:

* Create a secure MVC layout that restricts any bugs or tampering so that the only way to login is through the graphical password layout.
* Create a completely randomly generated graphical password grid which implements a directional feature with the unique users saved decision inside the session – this will be the KEY challenge as most of the algorithm will be dedicated to making this feature work correctly with any user input while being completely random.
* Balance the trade off in user friendly experience and security – striving for the perfect balance in not becoming a hassle for user experience but also fully securing the account from attackers.
* Create a feature which if the user fails to select the correct image, the session is closed and redirected to the home screen.

**New Challenges:**

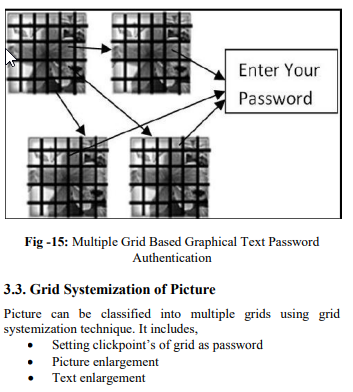
Throughout the course of the project I had to deal with unexpected challenges that grew in addition to the original challenges. Therefore, I had to invest extra time into research and understand in order to implement and solve these issues

* Feature which emails the user when a correct login has been made and the timestamp.
* Create a password recovery system which resets the user’s image or password based on their security pin.
* Creating a password recovery system that changes the image, not the basic string password.
* Testing the system on people who are not familiar with graphical passwords.
* Relating the project from a management perspective; how useful the system will be in the IT industry.
* Making the system more beneficial than existing graphical passwords already in the IT industry.

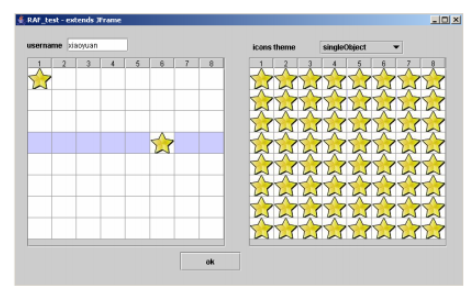
Another challenge which I came across was attempting to find a balance between a user friendly system and security as I found when testing the project on users who have not saw the system before seemed to be confused on how the grid operates, which I originally began to worry the system might increase login time by too much and the negatives outweigh the benefits. However, the possible solution was to limit the grid’s directions to 6 and spacing to 1 and 2, as if the user had too many options, it would become easier to forget which option you had chosen. I also implemented a way to check your options with your security pin and string password so users can check their login image and direction in case they had forgotten. Another new interesting challenge became notifiable when deciding the layout of the system and designing the icons in a user-friendly manner as images now light up when selected and if nothing is selected a warning appears notifying the user to select an icon. Linking the database to the system was a simple task, however the challenging aspect came when implementing a feature which emails the users when they login correctly using the Java Mail API which uses an email I setup as a host which will send a message to the users email with a message and timestamp. Extending my Java FX knowledge was a vital challenge I also had to overcome as going into the project I was limited on what software I should use to create the framework of the system. Research into how to link controllers using MVC (Model View Controller) and learning how to manipulate graphics with Java Classes so I could get the output I desired became a challenge in itself.

1. **Survey of Literature/Background Study:**

My project aims to create a system which implements a graphical password feature in order for a user’s account to be secure from shoulder surfing; in the beginning I researched other existing graphical password systems to get a grasp on what other companies were using to develop their graphical elements, doing this I found a paper on ‘Multiple Grid Based Graphical Text Password’ [**1**] which utilized many ideas on DAS (Draw a secret) and grid graphical passwords such as Man Algorithm, Jansen Algorithm and ‘Grid Systemization of Picture (Fig 1)’ which gave me the idea for the graphical password grid including images.

However, more research needed to be undertaken to decide what type of graphical element I wished to pursue for my own project, which is when I found a paper named ‘A Design and Analysis of Graphical Password’ **[2. Suo, X. (2006]** which showcased RAF (recall a formation – Fig 2) methodology where the user selects icons and places them into a grid and depending on placement of the icons the login will be correct. This premises gave me the idea to implement directional use in my project.

**FIG 1**



**FIG 2**

Now that I had established I wanted a grid with images incorporated into my system, I had to research what platform would best suit this particular project. I have had previous experience using Java and JavaFX and with the help of Oracle **JavaFX [3** Topley, K. (2020)**]** containing tutorials on elements such as VBoxs, HBoxs, labels and other elements needed to create a login page and graphical grid for the graphical element of the login interface.

The primary research was focused on MVC (Model View Controllers) so that the controllers loaded the correct JSP’s according to user ID. Fortunately, JavaPoint **[4]** allowed me to understand how MVC’s function by setting up controllers to write the behaviour of the elements while projecting them onto JSP’s to be viewed, all while connecting to a database.

To have consumers data be stored for my project, I researched methods on what database I should implement for my system. Further research was required to understand how to setup a database to store usernames, passwords, directions, and individual ID’s so that when logging in the correct user is called inside the program. Site **Grounds [5]** covered how to set this backend up with phpMyAdmin.

Specific research needed to be undertaken in order to arrange the JavaFX buttons into an array so that they can be placed at random into a 4x4 grid. Chris Hanley’s **[**6. Hanley, C. (2020)**]** video tutorials have taught me the foundation of the idea to store the buttons in the array.

When researching methods of encryption for the database to secure the users information, I researched the AES Algorithm [**7]** to help me to encrypt the data in the back end of the system.

After time had elapsed and the main project goals were met, the decision to implement the feature of having an email sent to the users account upon login was formed when researching on other existing logins which capitalise on email verification and make use of this feature, which provided me the idea of using classes such as SMTP and Java Mail API.

GitHub/**JavaFX [8],** provided vital information on how to link elements into a method inside Eclipse using ID’s by looking at past projects by other JavaFX users – specifically Minesweeper. Minesweeper and my projects graphical grid containing images had similar conditions, both needed to sort values into a grid at random, therefore research on how this game was coded helped develop the core algorithm I implemented in my graphical password grid to sort the image buttons at random through the 4x4 grid.

1. **Structured Requirements**

Requirements for my graphical password system were often altered during the development stages of my project as I gained better understanding of the area of study and software. Receiving feedback weekly also helped enormously as I was able to pinpoint mistakes and incorporate better ideas which shifted my requirements, these new essential requirements will be labelled below the original essential requirements.

**Essential Requirements:** These requirements have been completed fully and measure how successful my project has become.

Login system/page – A login system which verifies correct logins is an essential component for a graphical password project and must be clear what details to enter in which order.

Create Account page – Users must have the ability to setup their own accounts by picking a username, password, image, email, direction, and the ability to move 1 or 2 spaces in the grid.

Graphical Algorithm of 4x4 grid – The most essential requirement of the whole system relies on the graphical grid to always generate at random to prevent the system being easily predicted, along with remembering the users login details and granting a successful login on a correct selection.

Database to store encrypted user information – Information needs to be stored on a database so that login attempts can be verified, however another crucial requirement is that all data must be encrypted to prevent any backend attacks.

Reset Password Feature – Having the ability for consumers to reset their password is vital to the project as my system recommends to change your image every 3 login attempts in case of shoulder surfing attempts, without this feature the likelihood of a unauthorised login increases.

Tutorial Page – In case anyone is struggling to understand how the graphical password system operates, a tutorial page is an essential requirement which if not complete will fail to help new users and hinder the project’s success.

User friendly layout – Measuring this requirement proved difficult however after testing the design of grey/black and white with appropriate spacing and images which are highlighted on selection on various consumers, the feedback was overwhelming positive and all users were able to grasp the system at first chance.

Page flow – the system must operate smoothly as when a consumer clicks to go to the next page, all pages must redirect and close behind as this could cause confusion and make the system easier for shoulder surfing attempts as a user could forget to close a window and a unauthorised user may steal that information.

**New Essential Requirements:**

Realising that after monitoring the login attempt of a user more than a few dozen times, it becomes easier for shoulder surfing attempts to crack the solution as they monitor what image is always next to the one the user selects, therefore new requirements had to be applied in order to reduce this exact risk: KEY NEW ESSENTIAL REQUIREMENT: Double space feature – this feature enables users to select if they wish for the direction block to be increased not just once but twice, for example, instead of always selecting the image north by 1 space, you select the image north by 2 spaces. This new requirement drastically increased the likelihood that a shoulder surfing attempt would be successful.

Button to view information if a user forgets their details – this feature is located on the ‘forgot my password’ page and instead of having to change your password every time you forget, simply click this button after verifying your pin and you may view all the login details of that account.

While developing the system, a new essential requirement formed as the ability to notify users they have logged in at exactly what time became necessary in case of any breaches in the project, therefore the new requirement of having a feature which emailed users upon correct login was vital.

**Optional Requirements**

When selecting an image, not only does the image light up in red, but a text box appears with the description of that image for example ‘Red/Yellow’ in case you forget what the image looks like you can remember that line of text – this was an optional requirement as users can rely on remembering their image alone.

**Recommended Requirements**

A requirement I had thought considering was the ability to chance the grid size, however with more vital requirements such as the reset password feature and double space method, I decided a 4x4 grid had already accomplished the foundation of what I needed for my project and the ability to alter this was not needed, but would have been a nice addition if given extra time on the system.

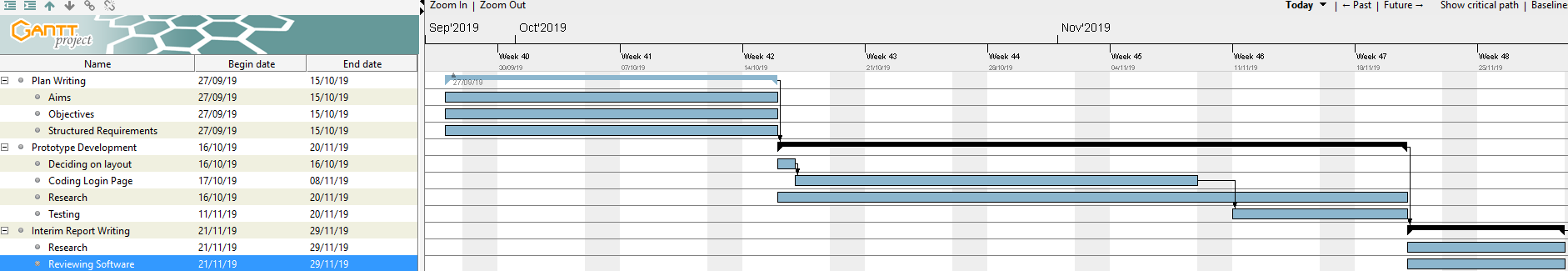
1. **Planning and Timescales:**

Revaluation the original schedule I can say with confidence I kept to my deadlines and adhered to all meetings on time, I did not make any altercations and managed to reach my milestones with enough time to develop new optional requirements such as the email system. Although what did greatly affect my performance for the better was the extra time over Christmas due to the report deadline being extended. Over the Christmas break due to the extension gave me extra to research on image theory and graphical passwords which gave me a new idea which consisted of changing the images of the grid to look similar besides slight colour changes, more on this decision in the Software Design segment.

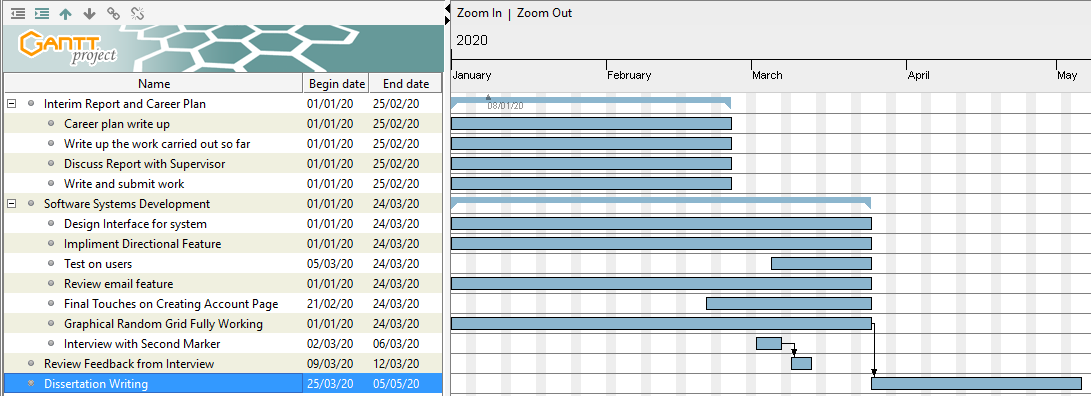
Another factor I did not take into consideration due to the unpredictability of the situation was COVID-19 that had caused my meetings since the 15th of March to be suspended and held online which during the first initial weeks of the lockdown was frustrating not being able to get the same benefits of face to face meetings and feedback with my supervisor, but later I adjusted to working at home and maintained a consistent work ethic and schedule by finishing my software side of the project by early March and using the remaining two months of time to fully focus on my dissertation write up. The deadline for the project was also extended which allowed me to become more adjusted to working from lockdown and allowed to so stay on schedule.

Outside of the virus causing drastic challenge for my work, I can say that it would have benefitted my workload if I had broken the work down to smaller subtasks and focused on each feature at a time, however there would be some days where I would get carried away working on one feature and feel the need to instantly complete the next feature straight away without reflecting, such as the login system and create account feature, which if I had cut the tasks up into segments my work could have been more efficient however I am still satisfied with the outcome of those features.

Here are the Gantt Charts for my project from first to second semester:

**Semester 1**

**Semester 2**



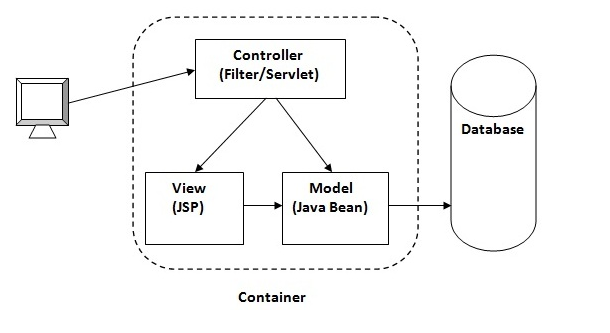
Here you can see the only updated graphic shown on the Gantt chart displays the extension of the dissertation deadline as due to COVID-19 I now had extra time to write the final dissertation until the 14th of May. The main goals of my project have been split into milestones making following the schedule easier to maintain. For a given deliverable I have set the completion of my work a week before the deadline in case of any interferences or deadline changes due to COVID 19 or if I wish to update any work I see fit.

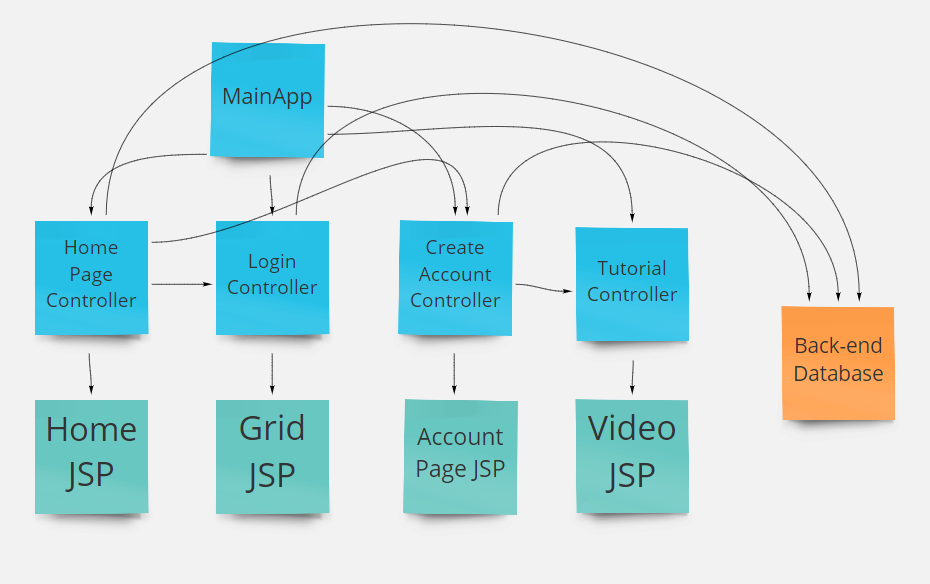
1. **System Design**

In this segment I will be reviewing the defining architecture, modules, interfaces, technical specifications, and software requirements for my system to measure the product that has been developed.

* 1. **Architecture (MVC)**

Below in diagrams Fig 3 and Fig 4 go over the main functions of my complete system displaying MVC (Model View Controller) technology.

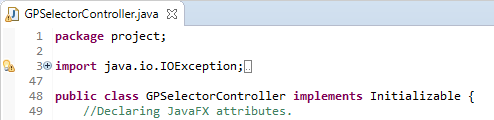




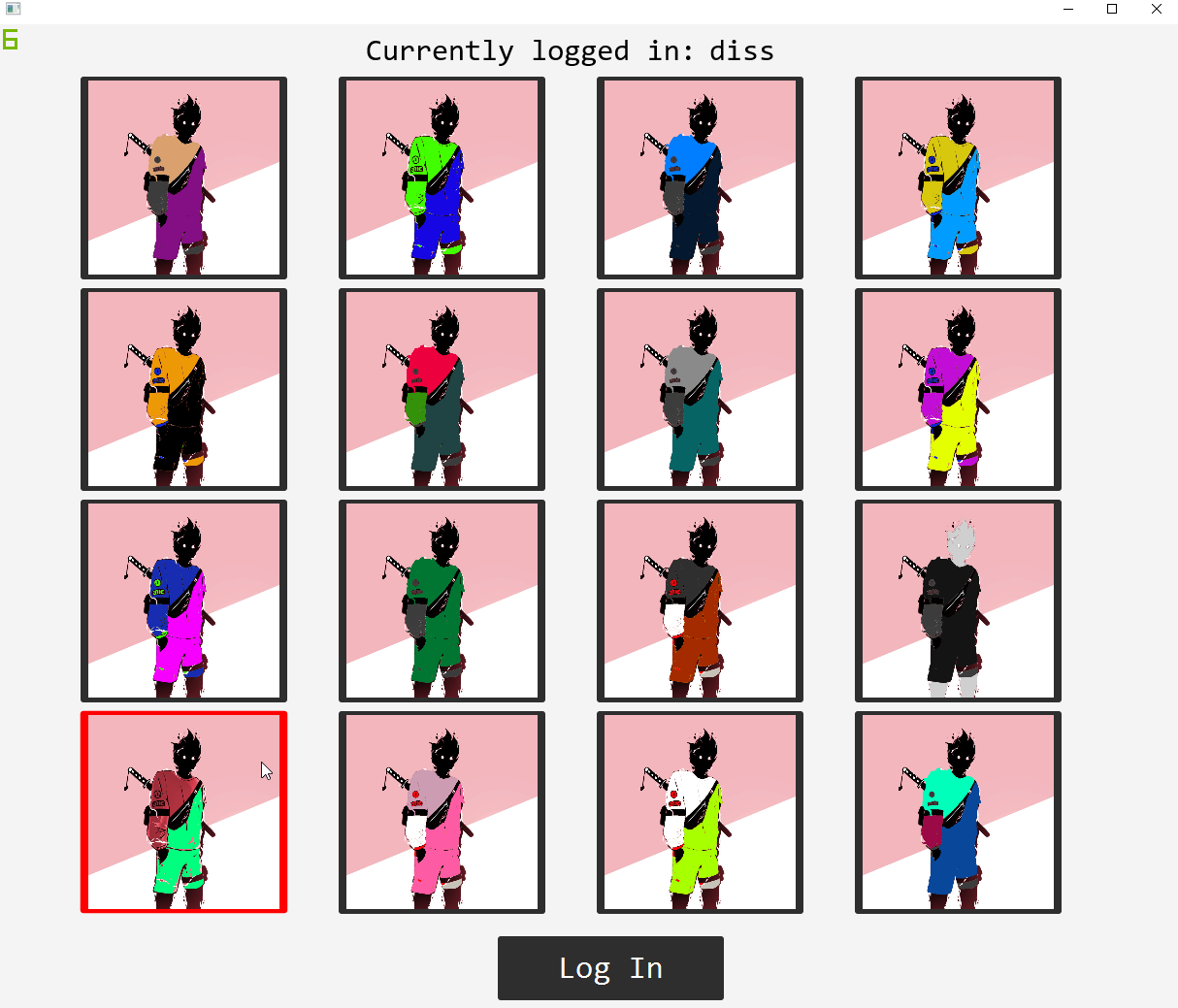
**FIG 4**

**FIG 3**

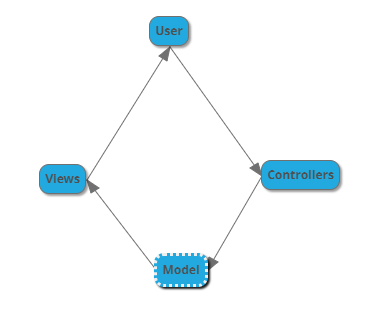
The central design for my final system is built on Eclipse IDE with Java FX, incorporating the architectural style of the Model View Controller (MVC) Architecture **(**[**Trygve Reenskaug**](https://en.wikipedia.org/wiki/Trygve_Reenskaug)**1970 [9]**). Looking at fig 3, the **Model** represents the data and logic of the system, the **View** being the user interface (JSP – Java Server Pages), and the **Controller** inside Eclipse that controls and handles the requests, the database relates to the SQL database on MyAdminPHP and finally the **Container** is “a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings” **[10].** The user will interact with the view (JavaFX interface using JSP’s) as a user interface – interactions within this user interface consist of; navigating pages, entering account information and selecting images on the grid – the view enables users to modify data such as entering their password details etc (Fig 2). Fig 3 displays how the controllers within my project react with the views and the backend. The reason I chose to integrate Model View Controller Architecture for my own project is that the framework is ideal for implementing a user interface which is vital for working with graphical password logins which require user input. An example of this framework in motion is the graphical password page itself, below you will see the View (Fig 5) and the Controller Class for the view (Fig 6). The View in Fig 5 will display the content on the JSP which is being manipulated by the controller in Fig 6, for instance if the user were to select a button the controller takes this action into account and runs it through the class to declare what happens next, in this case it will decide if the users input is correct when the submit button is clicked.



**FIG 6**



**FIG 5**

Additionally, another benefit of Model View Controller Architecture is that it allows the separation of the functionality between the model and the methods that interact with it **(Krasner et al, 1988 [11]),** making it much easier to control graphical elements for my own system. The final project consists of seven controllers; home page controller, login controller, create account controller, correct login controller, tutorial page controller, Graphical Password controller and reset password controller. The home page controller consists of a login menu with a username, password field and button to create an account. The controllers all interact with each other, for instance you may navigate from the home page to the create account page by clicking a button which will then open the view for the create account page. The GP controller consists of the most data structures as it has the main algorithm for the grid and many JavaFX dependencies. Entities in the model view controller do not have to depend on the GUI , this allows the user interface to be easily modified such as the random grid which easily manipulated the elements in the controller and displays them on the view and can be modified whenever for instance if a user accidently selects a wrong image, they can simply select another with no repercussions. Model View Controller systems also aid in interactive projects which take in user input and selecting options since I need the users input to gather their login details for the graphical password system to work appropriately. Fig 7 displays how the model view controller architecture works when entering your username to the system and then selecting the button to view your account details in the ‘Forgot My Password’ page. To break this flow of information transfer down, to start the user will interact with the view component displayed on screen by providing the inputs and receiving the outputs of information. The view complement will then transfer the data to the controller which will process the data and interact with it judging by the code written, then after the information is transferred to the model for storing the data into the database. The database then sends the data back to the view to be viewed by the user who has selected it.

**Submits Usernames**

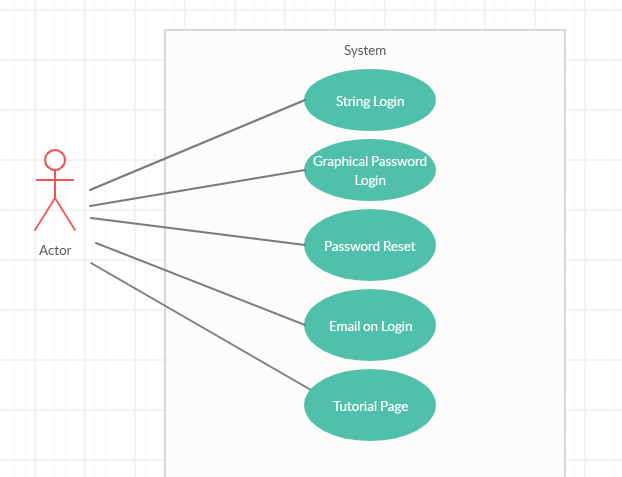
**Selects button to view information**

**Provides data**

**Sends to database**

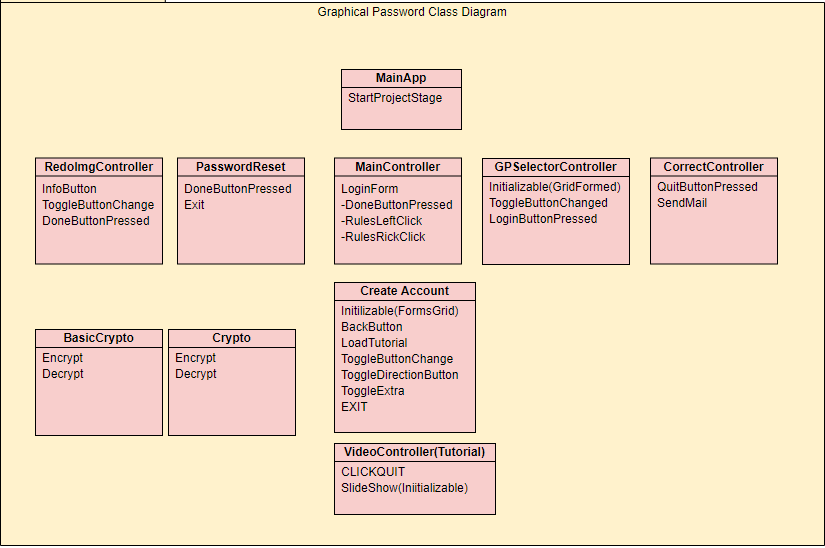
**FIG 7**

Below (Fig 8) I have created a use case diagram from my graphical password project which shows all the functionality a user can interact with on my software to display my systems behaviour.



**FIG 8**

These interactions shown in the use case diagram are only possible due to the support of the control view model.



**FIG 9**

Finally, above (Fig 9) you will see the layout of my final systems Class Diagram (some methods are not shown in full details such as connections to the database and JavaFX attributes as they are not under specific methods).

This class diagram shows the relations between my project’s classes, attributes, and all operations. All these controllers help manipulate elements of the GUI making model view controller architecture fitting for this project. The project is started when the MainApp is ran through Eclipse IDE or on the command line, the MainApp class loads the JavaFX stage which is manipulated by the MainController. The controllers that have links to the database are the following; MainController (for the login form to verify login string login attempts), GPSelectorController, (For verifying Graphical Password attempts), Create Account (for saving new information about a user’s account), PasswordReset (for verifying the security pin of a user) and finally RedoImgController (for resetting the users graphical password image). The class diagram represents my projects use of the entity subsystem for example changes inside RedoImgController class reflect on the GPSelectorController class. In order to simplify the system I made sure each page had a clear and direct motive, for example it would have been confusing for the user if both the tutorial page and the create account page were located under the same class and page, while also making backend work simpler for me while developing because if something were to be causing a bug I would simply locate the specific class controller related to that page with the bug, which would have been much harder if everything was written under one class. The class names are also appropriate to their functionality to ensure any IT firms that choose to develop this software can make any changes they desire and understand the project by reviewing the code and controllers.

* 1. **Technical Specification**

**Hardware Requirements:**

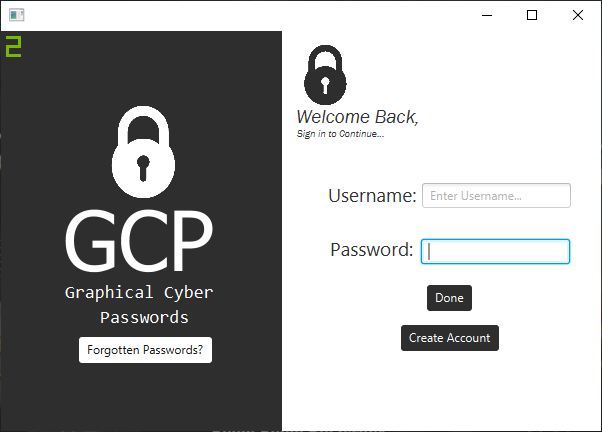
In order to run the system a computer or laptop is needed with Java installed. The system can be run by either through running the MainApp class through an IDE (Integrated development environment) such as Eclipse or try the command line (I have been running the system through Eclipse IDE by running MainApp. The user must have access to these resources to meet the hardware requirements:

* Keyboard – entering in your accounts data into the system.
* A mouse – navigating the system and selecting graphical elements.
* Internet connection – this is **VITAL** when operating the system as data needs to be loaded from the database on SQL MyAdminPHP, so the login works.
* A monitor – to display what is happening on the system.

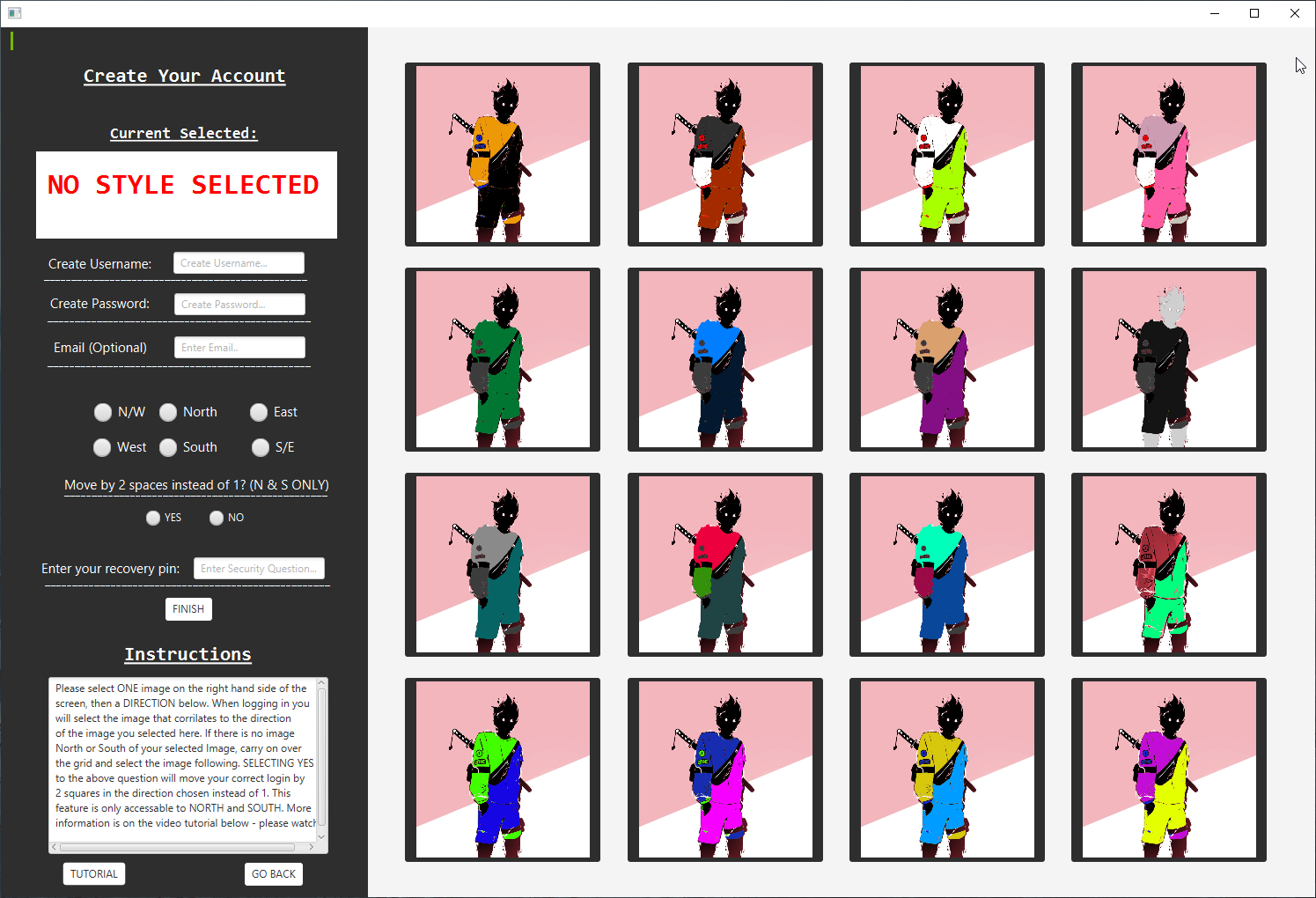
1. **System Implementation**

Here I will begin to list the primary functions that make up my complete system in detail. In order to have access to some functions I have included the java.mail.jar jar file so that the email feature works as intended. All images are also included inside the project to save hassle from loading them from a database. All graphical elements were created in Adobe Photoshop. The tools I used are as followed: Eclipse IDE, Java, JavaFX, SQL database and SceneBuilder.

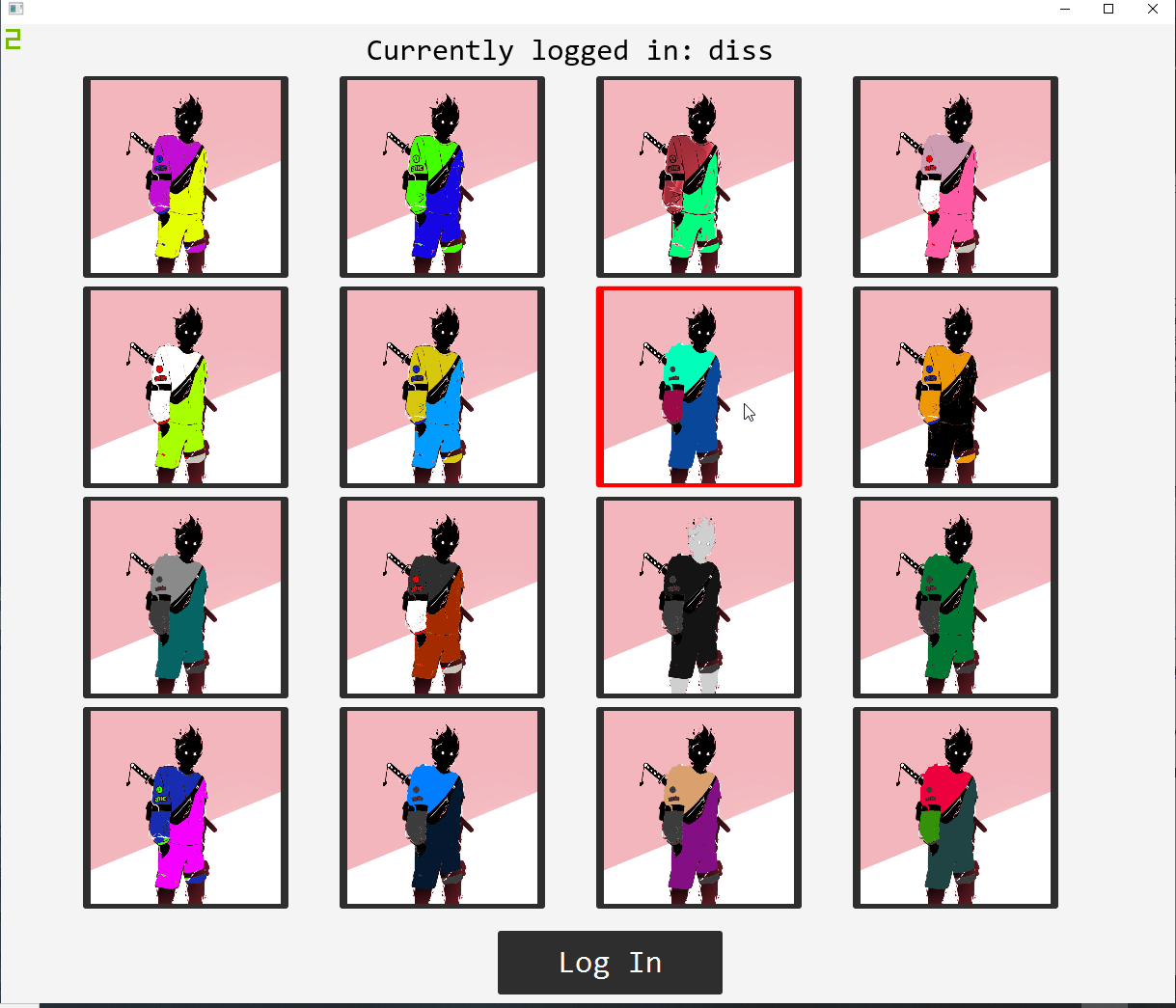
**Home Page:**



The home page consists of the screenshot above containing a username field, password field, recover account option and create account button. The idea behind the forgotten password feature is so that a user can reset their Image if they forget along with a feature to check their login information. The home page login once filled in correctly redirects you to the Graphical password login.

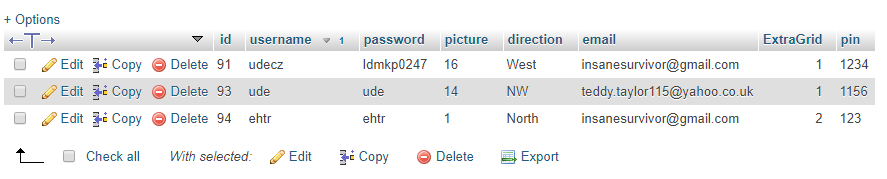
**Account Creation Page:**

Here is where the main bulk of the interactive user features take place, the user can select an image by clicking on any of the 16 images on the 4x4 grid panel. The user may also enter their desired username, password, email, security pin, image and select a direction for which option they must chose when it comes to logging in along with if they wish to have the direction to be moved by 1 or 2 blocks. Once the finished button has been selected (can only be selected if options are not null) the software will append the user’s information onto the database where it will be given a unique ID thus creating an account. The reason I chose similar looking images (as discussed in one of the objectives) is that my research discovered the ‘Picture Superiority effect’ **[12. Defeyter, M. (2009)],** specifically more to do with Nelsons ‘Sensory semantic theory’ **[13].** Picture superiority theory dictates that images are more easily remembered than images, thus making them easily to remember passwords. However, I did not want my images being easily remembered by a shoulder surfer therefore I implemented Nelsons sensory semantic theory which proves that “pictures’ with unique visual attributes make them easier to remember” and I do not wanting anyone but the user remembering the image they have chosen so I made all the images very similar, the trade-off for this is of course the user forgetting their image thus making it difficult to login and having to reset their password, but I believe it is up to the user to remember their image and maintain security.

**Login Page (When details are entered from home page):**

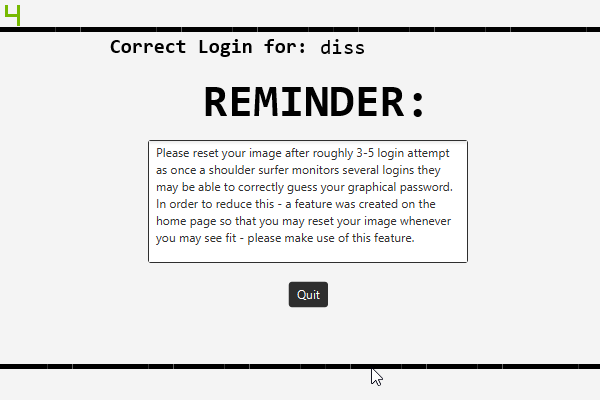
The login page works as intended, a 4x4 completely random grid of images which have the users saved choice in mind. Depending on the users selected direction and image, the user must first locate their image and then move by either one or two blocks in their chosen direction and select that image to correctly login. If the image chosen is incorrect, the page is closed and redirected to the home screen where they will have to re-enter the string password – this is to deny a brute force attack of just selecting and trying every single image inside one session. The username of who is logged in is displayed on the top of the header section. Images light up red when selected.

**Backend Database:**



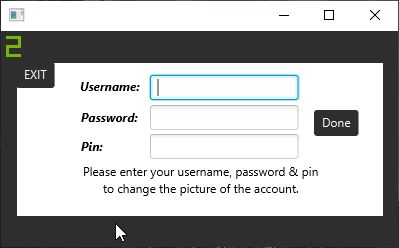
Here is the current backend, illustrating the ID, username, password, image, directional columns, email, extra grid, and pin which save the users data. You can see above that the username and password are encrypted so that a backend attack will be useless and add an extra layer of security to the system. Most classes in the system rely heavily on this database.

**Correct Login Page:**

****

Once the user has both correctly entered their string and graphical password this page is displayed with their login in the header. The reminds warns users that it is wise to reset their image after a few login attempts as a shoulder surfer could monitor several logins and guess the direction based off what images are always appearing in the direction to an image you chose. This was the main reason the change image feature was developed so that shoulder surfing attempts are near impossible. Selecting the Quit button returns the user to the home page while also sending an email to the accounts email address notifying them that a successful login attempt has been made with the time stamp included. I added this feature so users can remember what they were doing at that specific timeframe and know if the successful login attempt was legitimate or not so they can go reset their image with their security pin.

**Forgotten Password Page:**

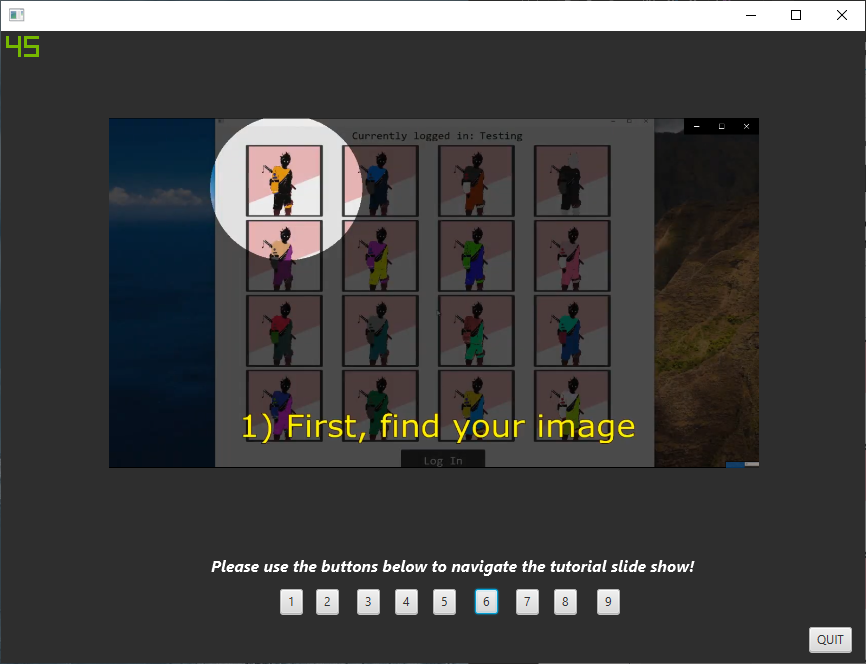
****

Once the button ‘Forgotten Password’ is selected from the home page, this window appears provoking the user to enter their Username, Password, and security pin to access the page to change their Graphical Password. Selecting the exit button returns the user to the home page.

**Reset Password Page:**

Once the forgotten password page is filled in correctly along with the correct security pin, the following Reset Password Page will load displaying the 4x4 graphical grid where users may select a new image for their account and click the done button to return to the home page with their updated image. Keep in mind the image may change but the direction and string password will stay the same. Users may also use this page to click the ‘Click here to see account info’ button which displays the user’s username, password, and direction in case they have forgotten any of their account information.

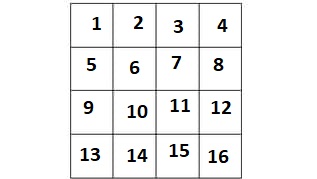
**Tutorial Page**

****

The tutorial page can be located in the create account page by selecting the button ‘Tutorial Page’ which loads a gallery showcasing the step by step actions required to operate my complete graphical password system. Users must select buttons 1-8 and have the opportunity to go back and forth in case anything is unclear. Selecting the ‘Quit’ button returns the user to the homepage.

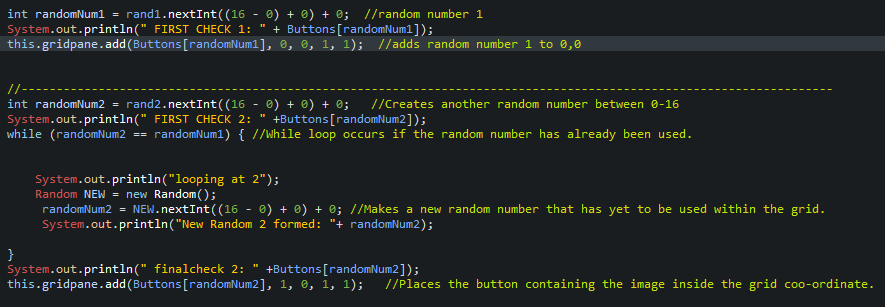
* 1. **Algorithms & Testing**

The most **challenging** component of the software which I spent the most time creating was the algorithm I produced to **completely generate a random 4x4 grid** (within the login page) containing button images which considered the users saved options. In order to do this, first I created a grid pane on JavaFX and then created 16 buttons with images inside them. I then put the list of buttons into an Array and created a random number generator between 0 and 15. These numbers represented the coordinates of the 4x4 grid (16 spaces) Fig 10.

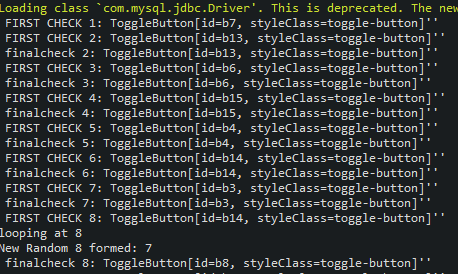


**FIG 10**



****

**FIG 11**

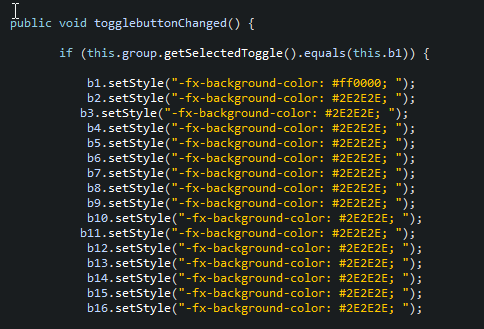
Inside this code snippet displays how the program decides what buttons will be placed into which grid co-ordinate. The random number is generated giving the number of the button e.g. 7 so button 7 will be placed into slot 1 on the grid (see fig 10). Now its grid slot number 2’s turns thus another random number is generated to find out which button will be placed here. Now what if slot number 2 also tried to add button number 7 which has already been used inside slot 1? If this occurs, a while loop continuously loops a random number between 0-15 until a unique button is selected, making sure all slots are given unique buttons at random. The issue with this current layout, despite never having this problem appear in testing, due to the nature of the random number layout there is the possibility that a login attempt could be unlucky and take longer due to landing on the same number each loop, and research will be done on this challenging aspect of the software to assure this issue is resolved and once a button is generated, the random number generator can no longer generate that specific number thus solving the possibility of any login delay or infinite loop. However, my testing has proven that this is never the case and the algorithm is efficient ****enough for my project and any lag time is due to the database connection, not the code.

**FIG 12**

Fig 12 - Here is the console displaying what has just been explained – the program is randomly generating which buttons to put inside the grid slots one by one. Slot 1 has generated button 7, slot 2 has generated 13 etc (Final check is a confirmation check). Slot 8 however has attempted to use button 14 which has already been predefined within slot 6, thus begins the while loop which has generated a non-used button 8 for slot 8.

However, as the project developed and this solution was realized, **another challenge** arose when testing my application on my friends. The way the graphical password works is by the user selecting NOT their image selected at account creation, but rather the image corresponding to the direction, e.g. Fig 10 – if your image is at 11 and you selected NORTH, you must select 7 to login correctly. The prototype system was flawed in this aspect as the user will never select their own image in any login attempt as it is impossible with the directional function, therefore if a shoulder surfer monitored the login attempts long enough and noted down which options were selected after at least 16 viewings (4x4 grid) or more the hacker could find out their private key (image chosen at account creation). In order to combat this challenge, a feature was developed allowing the user to change their image whenever they please along with another feature that increases the grid space by 2 blocks instead of the usual 1 making it even harder for shoulder surfing attempts to be carried out. Doing this keeps the private key changing so it is unpredictable to trace and eliminates any shoulder surfing attempt. However we now enter the debate in trade-off between user friendly interactivity and security as the user may find it stressful to reset their image every so often and may even forget their image, however due to the forgot my password feature users can check their direction and image whenever they please with their security pin.

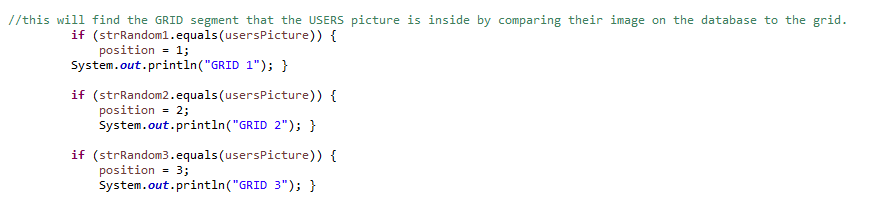
In order to make the software interactive and user friendly so that users do not get confused and select more than one image for the graphical password aspect was to implement strict code that forces the user to only be able to select one image. This feature was possible by using toggle buttons provided by JavaFX package.



Once the user selects their desired image, the image will highlight in a red colour and the others will be set to black – along with the image being saved as their desired choice when clicking the finish button which will save their choice onto the database – along with the username, password and directional option.

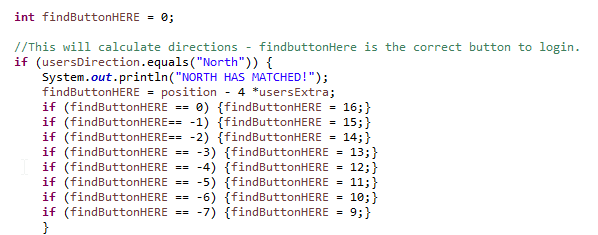


**FIG 13**

Once the grid is formed and the user has selected submit while selecting an image, this section of code is performed. The database finds the login of the user and fetches their picture and direction from the database (Fig 13), then using the random values in Fig 11 that were used to form the grid, those values are the buttons which are saved in string format (Image is cropped to save space but obviously there is more than 5 random values as 16 are needed for a full grid).

**FIG 14**

Once we have the user’s picture from the database and the string value of the button (Fig 10) we can compare them and find out which grid they are in. For example, if the user’s image were 4 and ‘strRandom1’ also equalled 4 we would know that button 4 is inside grid 1.

Once we know which grid our users’ image is positioned in, we must now take the users direction into account which we have already received from the database.

**FIG 15**

In Fig 15 we can see that if the users direction from the database matches with ‘North’ we must calculate on the grid where the correct login will be – we do this by subtracting 4 as you can see in Fig 10 subtracting 4 from 7 will find you in grid position 3 which is directly north of grid position 7. ‘userExtra’ is calculated by finding if the user has selected if they wish to have the grid move by 1 block north or 2, if they have selected yes then ‘userExtra’ will equal 2 and be 8 thus 7-8 equals -1. In this case, as you can see in Fig 15, -1 then equals grid 15 which as you can see in Fig 10 is exactly 2 blocks north of 7 if you continue along the grid. The reason in Fig 13, 14 and 15 you see messages which print out the grid’s location is because this helped me test the software as I would know by looking at the grid and images that they were performing correctly.

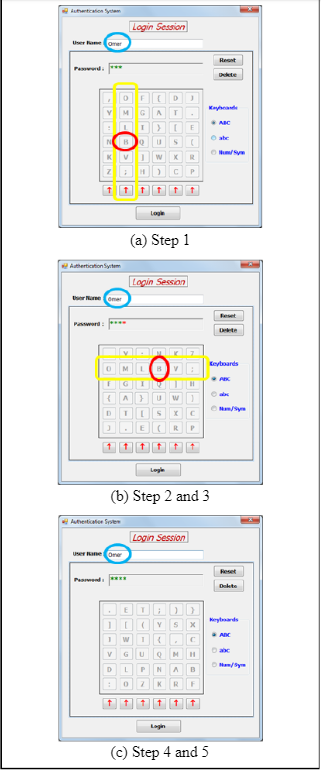
Once the correct grid login has been located it is saved as the correct button and any other attempts selected will close the window with a popup informing the user they have chosen incorrectly. The reason the window is closed after an incorrect login attempt is to deny a brute force attack by a hacker selecting every option in that one instance, instead the hacker would have to renter the string password and will find a completely new randomly generated grid.

1. **Critical Appraisal**
   1. **Evaluation of Aims**

Evaluating my complete systems results in consideration of my original aims I believe I have not only met them, but also managed to create new aims which I meet. Due to my past knowledge being of undergraduate level, it was difficult starting this project knowing how much research into MVC and JavaFX had to be undertaken. However, as the project progresses, and I set out my aims and system requirements I managed to develop new features on top of the original requirements and was able to balance a trade off in security and user friendliness efficiently. Having access to online papers showcasing pre-existing graphical passwords gave me insight to how to develop my own project. This research gave me the idea of the prototype which consisted of an 4x4 image grid where a password is selected on either north or south. I was able to expand on my knowledge of JavaFX and develop new features that solved the issue of shoulder surfing attempts being able to guess the password by watching the login several times by implementing a new feature that allowed users to reset their password; this idea is a convention of modern logins and aided my graphical password system. The experience of realising there was a fault in my system and undergoing research and solving the issue with a new feature not only gave me a newfound appreciation of coding but taught me that determination through uncertainty can lead to successful results. During my degree I was exposed to a taster of JavaFX in the form of making a dice game, building on this knowledge with research onto SceneBuilder I managed to do what I originally thought difficult which was to make a randomised grid which took into account the users image. Given the nature of my degree, I felt at an advantage from a management perspective as I was able to put myself in a the perspective of the consumer and have insight onto what business might look for with my project which helped with creating aims to making the system user friendly and easy to implement to my graphical password system to existing login systems without being too technical and difficult to understand. Furthermore, looking at my disadvantages due to being on a management degree I was not exposed to further Computer Science modules so my technical ability was hindered thus developing a complex login system became a challenge which I had to undergo more research more on Java and MVC than compared to other Computer Scientist students to grasp a better understanding and develop the system appropriately. An example of a feature which went extremely well during development was the ability to change block space from one to two. For example, due to how I developed the grid format for the random graphical grid, all the grid spaces had unique numbers listing from one to sixteen and to get a North direction the users image number (lets say seven) had to be misused by four to get the image above (see Fig 10). In order to add the plus block space two feature, all that was left to do was times the minus value by two thus increasing the space by two (See Fig 15). This showcases that due to making the layout of my randomised grid simple to understand and modify it allowed me to add features easily onto already existing features without having to dismantle or restructure any code, making the block spacing feature a new aim which was completed with ease despite thinking it was going to be challenging. In hindsight, the existing issue with the project is the database I used makes the projects load times long and can sometimes crash the project. When logging in, sometimes the user must be patient and wait for the database to fetch the information and the window may appear frozen however given enough time (no more than a few minutes) the graphical password page will load without fail. Another feature if given extra time to work on would be to enforce the user to change their password instead of recommending the password reset feature as some users may ignore the feature and not change their graphical password often enough and be susceptible to shoulder surfing attempts.

* 1. **Comparison of other Graphical Passwords**

Despite their clear advantages, graphical passwords are still not as often implemented as some may think as “perceptions of graphical passwords: 52% of people do not support the use of graphical passwords**[14]**” as larger companies opting to have secondary authentication through email however these methods are still vulnerable to shoulder surfing attacks as they may be viewing your screen while you login to your email to retrieve the randomly generated password code. Existing graphical passwords that are used commercially are systems such ATM’s and mobile phones showcased by a paper named “A smooth Textual Password Authentication Scheme against shoulder surfing attacks**[15]**” which displays ATM’s adopting graphical passwords for the benefits of reducing production costs of keypads and increasing in prevention of shoulder surfing; “some researchers tried another direction by proposing graphical password techniques [7, 8] as an alternative to textual password methods [9, 10]” showcasing this system (Fig 16) by saying “the proposed authentication system needs to strike a balance between security, usability, and ease implementation and improvement of the work procedures**[15]**”.

The system operates on these terms; the user does not select any of the keys on the grid but rather uses the arrows to navigate through the columns to enter their password, once the first character is selected, the displayed keyboard then performs a matrix transpose on the keyboard grid making a completely new keyboard (similar to my random image graphical grid) however not at random, but instead rotates the keyboard so each row now becomes a column (Fig 16 step 2). Then the user must select the arrow which points towards the column with the same character selecting previously, thus making it difficult for shoulder surfers as there are 6 elements on a row so the first character could be any of them. This process is repeated for all the rest of the characters until the password is complete. This system is easily understandable and operate and works well. However, what this system lacks compared to my own graphical password project is the aspect of random generation. Once the shoulder surfer has an understanding that the rows rotate they can observe the user login multiple times and see what letter remains on every selection and can note down and crack their password. My system in comparison is generated at random each time the page is loaded, and directions range from North, South, East, West, NW, SW including a one or two block space feature adding an extra layer of security. My system also includes the ability to change graphical passwords at any time making it more useful and secure.

**FIG 16**

* 1. **Commercial Impact**

The commercial impact of my system could have large benefits to online companies who wish to secure their login systems not only from outside attempts but also shoulder surfing attacks either through recordings on in person viewing. My system was made to balance complete security while maintaining a high level of user gratification as my project is easy to navigate and operate. From a management perspective, the simplicity of the system makes it easy for existing IT firms to implement this design across the company’s existing infrastructure without hindering cost or time. Companies are always looking for new developments in the cyber security field therefore I believe graphical passwords are the way forward.

* 1. **Personal Development**

Developing this graphical password project has challenged me in ways like no module coursework could ever do, I was left to research brand new topics and fields of IT which I had never viewed before and knew what I needed to produce but did not yet know how to achieve it. I believe looking from my prototype to the finished product, it is evident that I have illustrated sound project development skills as I have been able to look at my past mistakes and flaws in the system, develop new ideas and through determination research and reach my aims and objectives completely. I now grasp the Model View Controller layout at a high level and would have no issue implementing this software architecture style again for future projects. I have also learned new ways to style and manipulate JavaFX graphics due to the experience this project has given me. While also focusing on my system in terms of the management side of my degree, the experience helped me view my project from a business perspective and helped me understand the needs of the consumer and what a user would like and dislike from my system, and the benefits it could bring when implemented to an existing IT firm. In summary, this project on graphical passwords has deepened my understanding from a technological standpoint while also exposing me into thinking from a management perspective which I believe are the essential skills I will need to help propel my career in the IT industry.

1. **Conclusion**

In conclusion, I believe overall I have developed a comprehensive graphical password system which has not only reached my original aims and objectives but has managed to complete new aims through the development of this system. The project has also showcased the clear benefits to using graphical passwords for login systems and prevents shoulder surfing attempts better than some existing graphical password systems out in the world. The combination of my sound knowledge of Model View Controller architecture with my developed knowledge on Java, JavaFX and SQL I have surpassed my original requirements and goals for the system and developed a graphical password system that balances the trade-off between security and user accessibility at a sound and healthy equilibrium. In summary, I was able to advance my technical and management skills and develop a sound project that reached the final purpose of the dissertation which also allowed me to critically evaluate my project and ensure the result was successful while taking away key learning experiences.

1. **Bibliography and Citations**

1. Pdfs.semanticscholar.org. (2020). *MULTIPLE GRID BASED GRAPHICAL TEXT PASSWORD AUTHENTICATION*. [online] Available at: https://pdfs.semanticscholar.org/aaf7/d9b15967ce8877816432582542b3426a1407.pdf [Accessed 13 Feb. 2020].

2. Suo, X. (2006). *A Design and Analysis of Graphical Password*. [online] Scholarworks.gsu.edu. Available at: https://scholarworks.gsu.edu/cgi/viewcontent.cgi?referer=https://scholar.google.com/&httpsredir=1&article=1026&context=cs\_theses [Accessed 13 Feb. 2020].

3. Topley, K. (2020). *JavaFX Developer's Guide*. [online] Google Books. Available at: https://books.google.co.uk/books?hl=en&lr=&id=bnDCXcoZR9MC&oi=fnd&pg=PT34&dq=javafx+guide&ots=ha3VVGHLkb&sig=2m2YHkinXwnQpofcCuUrGodsaME&redir\_esc=y#v=onepage&q=javafx%20guide&f=false [Accessed 13 Feb. 2020].

4. www.javatpoint.com. (2020). *MVC in JSP - javatpoint*. [online] Available at: https://www.javatpoint.com/MVC-in-jsp [Accessed 13 Feb. 2020].

5. Website Tutorials - Find Out How to Use the Most Popular Web Apps. (2020). *phpMyAdmin Tutorials*. [online] Available at: https://www.siteground.com/tutorials/phpmyadmin/ [Accessed 13 Feb. 2020].

6. Hanley, C. (2020). *YoutTube Channel*. [online] YouTube. Available at: https://www.youtube.com/channel/UCQ-KjZ-dBpmaEoXp-1JyF3Q [Accessed 13 Feb. 2020].

7. Quickprogrammingtips.com. (2020). *How to Encrypt and Decrypt Data In Java Using AES Algorithm*. [online] Available at: https://www.quickprogrammingtips.com/java/how-to-encrypt-and-decrypt-data-in-java-using-aes-algorithm.html [Accessed 13 Feb. 2020].

8. GitHub. (2020). *mhrimaz/AwesomeJavaFX*. [online] Available at: https://github.com/mhrimaz/AwesomeJavaFX [Accessed 13 Feb. 2020].

9. Heim.ifi.uio.no. n.d. *Trygve/MVC*. [online] Available at: <http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html> [Accessed 13 May 2020].

10. Medium. (2020). *What are containers and how to containerizing an Asp.Net MVC application using Docker*. [online] Available at: https://medium.com/@somakdas/what-are-containers-and-how-to-containerizing-an-asp-net-mvc-application-using-docker-30931579f248 [Accessed 13 Feb. 2020].

11. ResearchGate. 1988. *(PDF) A Description Of The Model-View-Controller User Interface Paradigm In The Smalltalk80 System*. [online] Available at: <https://www.researchgate.net/publication/239452280\_A\_Description\_of\_the\_Model-View-Controller\_User\_Interface\_Paradigm\_in\_the\_Smalltalk80\_System> [Accessed 13 May 2020].

12. Defeyter, M. (2009). *Cognitive Development*. [online] https://www.sciencedirect.com/science/article/pii/S0885201409000471. Available at: https://www.sciencedirect.com/science/article/pii/S0885201409000471 [Accessed 13 Feb. 2020].

13. Alleydog.com. (2020). *Sensory Semantic Theory definition | Psychology Glossary | alleydog.com*. [online] Available at: https://www.alleydog.com/glossary/definition.php?term=Sensory+Semantic+Theory [Accessed 13 Feb. 2020].

14. n.d. *A Study Of Perceptions Of Graphical Passwords*. [online] Available at: <https://www.jstor.org/stable/26502733?seq=1> [Accessed 13 May 2020].

15. 2018. *A SMOOTH TEXTUAL PASSWORD AUTHENTICATION SCHEME AGAINST SHOULDER SURFING ATTACK*. [online] Available at: <https://www.researchgate.net/publication/325171260\_A\_Smooth\_Textual\_Password\_Authentication\_Scheme\_Against\_Shoulder\_Surfing\_Attack#pf2> [Accessed 13 May 2020].