

UNIVERSITY OF LEICESTER

DISSERTATION

CO3201

Avoiding Shoulder Surfing via Graphical Passwords

Author:
Adam BARNES

Supervisor:
Dr. James B HOEY

School of Computing and Mathematical Sciences

May 5, 2022



Declaration of Authorship

DECLARATION

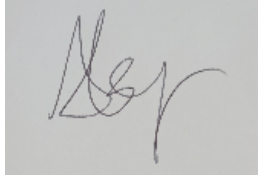
All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people's work have been specifically acknowledged by clear citation of the source, specifying author, work, date and page(s).

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 amounts to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Name: Adam Barnes

Signed:

A handwritten signature in black ink, appearing to read 'AB', is written over a light gray rectangular background.

Date: 27/04/2022

Abstract

Shoulder surfing is one of many attacks that can be carried out during authentication. This project explores Graphical Passwords, a form of authentication that uses images and colours, as a method of avoiding shoulder surfing. Numerous graphical password approaches were selected through a survey of the literature, especially those which incorporate shoulder surfing resistance into the methodology, and they were implemented into a piece of software that could be used in research. This was accomplished by using Java graphics libraries, connected to a MySQL server, which could allow multiple users to register a profile for each method. Each login page includes the graphical password methodology based on the paper behind it. The resistance of shoulder surfing was first tested through the creation of *mock* shoulder surfers, windows which appeared along the login process, and calculated the possible elements, that could be the password, on screen. Shoulder surfing was also tested through real participants in user testing. Multiple tests were performed on four participants, and the results showed that shoulder surfing resistance was generally high. However, some pitfalls were revealed with the methodologies, when evaluated, based on usability and security flaws. Alphanumeric passwords also showed a high shoulder surfing resistance against the participants. It was also shown that the unique usability of each graphical password could make them a good alternative to current forms of authentication in some cases, however, they were not seen as a clear replacement.

Contents

1	Introduction	1
1.1	Background	1
1.1.1	What is Shoulder Surfing and How Can it Pose a Risk?	1
1.1.2	What Is a Graphical Password?	1
1.1.3	How Can a Password Be Truly Secure?	1
1.2	Aims and Objectives of The Project	2
1.3	Challenges	2
2	Literature Review	3
2.1	Text and Colour Wheel	3
2.2	Coin Password	4
2.3	(Digraph substitution rules) Image Grid	4
2.4	Text and Colour Grid Password	5
2.5	Propositions not implemented	5
2.5.1	Free Drawing Password	5
2.5.2	PassPoints Password	6
2.5.3	Grid Image Based Method	6
2.6	Motivation	6
3	Software Requirements	7
3.1	Functional Requirements	7
3.1.1	Graphical Password Selection	7
3.1.2	Graphical Password Registration	7
3.1.3	Graphical Password Login	8
3.1.4	Graphical Password Analysis	9
3.1.5	Database Data Integrity and Security	9
3.2	Non-functional Requirements	9
3.2.1	Usability	9
3.2.2	Graphical Password Scalability	10
3.2.3	Software Performance	10
4	Specifications and Planning	11
4.1	Language and Architecture	11
4.2	Database Structure	12
4.3	Structure of The Requirements	13
4.4	Suite Navigation	14
4.5	Login Methods	15
4.5.1	Colour Wheel Method from 2.1	15
4.5.2	Coin Password Method from 2.2	16
4.5.3	(Digraph) Image Grid Method from 2.3	17
4.5.4	Colour Grid Method from 2.4	18
5	Implementation	19

5.1	Database	19
5.1.1	Database Setup	19
5.1.2	Database Runner	21
5.1.3	Database Queries	21
5.2	Graphical User Interface	22
5.2.1	Using Swing	22
5.2.2	Selecting a Graphical Password	22
5.2.3	Getting and Displaying Images	24
5.2.4	Graphics Package	25
5.3	Graphical Password Method Implementation	26
5.3.1	Colour Wheel Method	26
5.3.2	Coin Password Method	29
5.3.3	(Digraph) Image Grid Method	32
5.3.4	Colour Grid Method	36
6	Testing	37
6.1	Unit Tests	37
6.2	Integration Tests	37
6.3	User Testing and Feedback	38
7	Results and Evaluation	39
7.1	Security	39
7.1.1	Resistance Against Bystander Shoulder Surfers	39
7.1.2	Resistance Against Multiple Shoulder Surfing Attacks	39
7.1.3	Password Entropy and Brute Force Attacks	40
7.2	Usability	41
7.2.1	Word Processing and Retention Difficulties	41
7.2.2	Considerations For Colour	41
7.2.3	Complexity and Effort	42
7.3	Can Graphical Passwords Replace Other Authentication Methods?	43
8	Critical Appraisal and Conclusions	45
8.1	Summary	45
8.2	Improvements and Future work	46
8.2.1	Research Data	46
8.2.2	Database Additions	46
8.2.3	Front-end and Back-end Architecture	46
8.2.4	Approach Improvements	47
8.2.5	Mock Shoulder Surfer Improvements	47
8.3	Socio-economic Impact	47
8.4	Personal Development	48
A	Unit Tests	51
B	Integration Tests	53
C	User Tests	58
C.0.1	User Testing Participation Consent Form	58
C.0.2	User A	59

C.0.3	User B	60
C.0.4	User C	61
C.0.5	User D	62
C.0.6	Results	63
D	Time Management	65

List of Figures

4.1	A class diagram to show the structure of the database.	12
4.2	A use case diagram to visualise the requirements.	13
4.3	A flow chart to show the navigation process of the suite.	14
4.4	A mock-up of how the Colour Wheel Method login will work.	15
4.5	A coin password mock-up, and which order of coins the user should click. .	16
4.6	A mock-up of how the Image Grid Method login will work.	17
4.7	A mock-up of how the Colour Grid Method login will work.	18
5.1	A prompt to the user to enter mySQL details.	20
5.2	Examples of the DatabaseRunner being used.	21
5.3	How a PreparedStatement is constructed in java.	21
5.4	Welcome Page where the user can select a method to register or login. . . .	22
5.5	Registration form for the colour wheel method.	23
5.6	How a BufferedImage is converted to a byte stream.	24
5.7	Example of a paintComponent for the WheelCanvas class.	25
5.8	How the wheel was rotated using Collections functionality.	27
5.9	Code snippets within the drawCircleSector function.	27
5.10	The login page for Text and Colour Wheel Method.	28
5.11	Graphics2D package used to create each coin, using an Ellipse2D object. . .	29
5.12	The login page for the Coin Password method.	30
5.13	Shoulder surfing window whilst login is in process.	31
5.14	Shoulder surfing window after the login concludes.	31
5.15	The checks made for Scenarios A, B and C.	33
5.16	The login page for the (Digraph) Image Grid method.	34
5.17	Shoulder surfing window during the logging in phase.	34
5.18	Shoulder surfing window after the login phase.	35
5.19	The colour grid login process.	36
C.1	A line graph showing the logged times taken to login from longest to shortest.	64
D.1	A graph showing the list of issues created on GitLab over the duration of the project.	65
D.2	A screenshot showing some of the issues created, their labelling and naming.	65

List of Tables

A.2	Coin Password Unit Tests.	51
A.4	Colour Grid Unit Tests.	51
A.6	Colour Wheel Unit Tests	51
A.8	Image Grid Unit Tests	52
B.2	Integration Tests for the Colour Wheel Approach.	53
B.4	Integration Tests for the Image Grid Approach.	54
B.6	Integration Tests for the Coin Password Approach.	55
B.8	Integration Tests for the Colour Wheel Approach.	56
B.10	Integration Tests for general functions and the welcome page.	57
C.1	User A results for shoulder surfing whilst not knowing each mechanism. . .	59
C.3	User A approach feedback.	59
C.4	User A results for shoulder surfing after knowing each mechanism.	59
C.5	User B results for shoulder surfing whilst not knowing each mechanism. . .	60
C.7	User B approach feedback.	60
C.8	User B results for shoulder surfing after knowing each mechanism.	60
C.9	User C results for shoulder surfing whilst not knowing each mechanism. . .	61
C.11	User C approach feedback.	61
C.12	User C results for shoulder surfing after knowing each mechanism.	61
C.13	User D results for shoulder surfing whilst not knowing each mechanism. . .	62
C.15	User D approach feedback.	62
C.16	User D results for shoulder surfing after knowing each mechanism.	62
C.17	The shoulder surfing resistance-rate of each approach after a single shoulder surfing attempt.	63
C.18	The shoulder surfing resistance-rate of each approach after multiple shoulder surfing attempts.	63
C.19	The median time taken to login for each approach.	64

Chapter 1

Introduction

1.1 Background

This section introduces the main topics of discussion in this report.

1.1.1 What is Shoulder Surfing and How Can it Pose a Risk?

Shoulder surfing is the act of looking at the display and input of an individual's device, in the context that it is without consent. Most shoulder surfing incidents tend to be out of boredom and curiosity [1]. However, a user's security can be at risk if they enter a password while being shoulder surfed. A perpetrator may repeatedly observe a login process to gain access to the users password, whether through keyboard or touchscreen entry. There has been research made into the shoulder surfing of authentication methods [2]. However, the metrics surrounding the evaluation of shoulder surfing resistance in authentication could be improved [3].

1.1.2 What Is a Graphical Password?

Graphical passwords aim to reinvent the current forms of authentication, through the notion that humans are more likely to recall visual cues rather than text. Using an image as the basis for a password provides a much larger password entropy (in practice), as the number of data combinations can become categorically larger than that of textual passwords. This can be useful in the security against dictionary attacks [4], and even keystroke logging depending on the methodology used. There have been many graphical passwords proposed in literature, each with their own focus on different elements of security and usability.

1.1.3 How Can a Password Be Truly Secure?

Data protection has evolved dramatically over the last couple of decades, making passwords nearly impossible to crack using encryption techniques, and authentication a larger precedent with the introduction of bio-metric scan technologies [4]. Lip Yee POR et al [5] convey that a password that is unequivocally secure should contain a randomness attribute while also being memorable for the holder.

1.2 Aims and Objectives of The Project

This project aims to evaluate different graphical password methods in their ability to prevent shoulder surfing, whilst also being user-friendly: by creating a graphical password software testing suite. To achieve this aim, several objectives necessary to complete were outlined.

1. Research different graphical password concepts which aim to prevent shoulder surfing and choose those to implement.
2. Create an application to implement these approaches.
 - (a) Main page where the user can select an approach and choose to register or log in.
 - (b) Each registration and login page for each selected approach.
 - (c) Allow users to register a profile that will be linked to their graphical password approach details.
 - (d) If applicable, show how a malicious shoulder surfer could gain the users password through multiple logins or calculations given in the summary.
 - (e) Connect a database to the application so multiple user details can be stored for each approach.
3. Perform user testing to help with my final evaluation. Evaluation could include:
 - (a) Ability to stop a one-time shoulder surfer all of the time.
 - (b) Ability to continually deny a re-offending shoulder surfer with the same password.
 - (c) The time taken to login and the factors that affect this, such as password length or mechanism difficulty.
 - (d) Accessibility of each method.

1.3 Challenges

The project was started with very limited knowledge on the technologies needed to create graphical passwords, and the only knowledge to help with the construction of a desktop application is that of web designing. Another challenge would relate to the mechanisms within the graphical passwords themselves. For example, implementing the Image Grid Method (2.3) would require a lot of scenarios to be satisfied for the user to login, and a system developed which can keep hold of the possible images used as an imitation of a shoulder surfer. Also, the Text and Colour Wheel (2.1) would require graphic skills to create the wheel and move around the elements within it based on user input. Lots of small attributes will have to be considered to make an easily navigable system, with the relation between the software and database running as smoothly as possible.

Chapter 2

Literature Review

This chapter reports on the numerous graphical password approaches in literature.

2.1 Text and Colour Wheel

The text and colour based graphical password [6] is a proposition to directly oppose shoulder surfing. There have been several text-only based shoulder surfing models proposed, those of which the paper states are still insecure and inefficient. Instead, the paper proposes the use of colour together with text to provide better resistance to shoulder surfing.

A wheel of 8 sections include 8 characters and colour each. There are 64 characters in total, including all upper-case and lower-case letters, numbers and "." "/" symbols. The login mechanism works by rotating the text (or colour, for other similar approaches [7]) around the wheel until the current character of the password is in line with the colour selected in registration. There is a button to confirm this entry, which will either increment to the next character of the password or display a warning if the entry was incorrect. The login button is to be used when the user has (believed to have) entered all the characters in their password.

The paper states the password space (length) $L = 8$ as $\sum_{L=8}^{15} 8 * 64^L = 1.006 * 10^{28}$. The resistance to coincidental logins is also evaluated, and shown to be very effective past the length of 8.

The approach seems very effective against shoulder surfing, as when the user is entering the password only an action of pressing confirm is needed, for a random rendering of the wheel, with no indication of what letter the user is confirming for which colour. On the other hand, if the colour is known from registration, the shoulder surfer will have fewer variables to consider for the password. Instead, it will be the 8 letters of the chosen colour, in 8^L combinations. This is still very secure against both a passer-by or a malicious attacker proactively enacting observations.

The paper states that the method user friendly, due to its simple mechanism and its basis on the already familiar textual passwords. However, this may not be entirely accurate as the login method would take much longer than that of a textual password, especially once the length of the password becomes over half a dozen characters. The data provided to prove shoulder surfing resistance is small, and could be further assessed in this project. With a custom implementation, an analysis of the strength of the shoulder surfing resistance and how the usability falls short can be carried out.

2.2 Coin Password

The next approach by Fong et al [8] emphasises mobile devices, of which largely use a pin numerical password for authentication. This is deemed vulnerable to shoulder surfing, depending on factors such as observation angle, password size, phone size etc. [9]. The method is however a basis on which the researchers used to apply better shoulder surfing resistance.

The paper brings attention to different authentication techniques, such as cognitive biometrics, recognition and recall-based techniques, and hybrid schemes that have been developed to help oppose shoulder surfing attacks. The Coin Passcode which is proposed is a hybrid graphical password, as concentrates on the use of multi-elemental input, composed of icons (image-recall), numbers and colours.

Registration involves the user selecting at least two of each element, with a total length of 6. There are 10 different icons, numbers from 1 to 10 and 10 different colours available for selection. In the login phase, 10 coins are generated with each element randomized. The user must select the coin with the element included in their password in order, and after each entry the coins are regenerated. Once the user enters the password in order and without error, the user is authenticated.

This proposition is said to have higher complexity than the numerical passcode model that motivated it. An experiment showed that its shoulder surfing attack rate was 0%, and that its memorability success rate was the highest out of a range of password models at 88%, both in the papers Figures 13, 14 [8] respectively. The reasoning for the prior data is that a shoulder surfing attack is “meaningless” when there are multiple elements for each input. This is a big statement to make, and while this may be true for a stranger who would not be found shoulder surfing again, consideration must be made for the fact that malicious shoulder surfing attackers may watch a user login several times rather than once. The assumption could then be made that the shoulder surfer could keep a trace of these elements and remove those that do not appear again in succeeding logins. This project will explore these claims made and make comparisons to the other graphical password methods evaluated.

2.3 (Digraph substitution rules) Image Grid

The paper by Lip Yee Por et al [5] proposes a method which aims at making several different means of password entry for the user, based on the 2 “pass images” that they select in registration. For the login procedure, a grid of images, along with the 2 selected images are displayed. Rather than the user clicking on their selected images, which would be vulnerable to a shoulder surfing attack, the user can click other images that fit one of the mechanisms scenarios.

The paper lists scenarios A (if the pass images are horizontal to each other), B (if the pass images are on the same vertical vector) and C (if the pass image are on the same horizontal vector). In scenario A the user will click on the two images that are on opposite vectors (x_1, y_2 or x_2, y_1) to the pass images. For scenario B, the user clicks on the images that are below each pass image, and if the pass image is at the bottom, the image to click on loops back to the top. This is similar for Scenario C, where the images to click on are to the right

of the pass images.

With this mechanism in place, it makes it hard-to-impossible for the shoulder surfers to figure out the two pass images that the user selected, even if they are aware of the algorithm that is used. To gain these pass images, a malicious shoulder surfer would have to watch the login process numerous times to reduce the repeating images that appear within any scenario for each login.

Experiments in the paper show that participants login quicker over time, and that no shoulder surfers were able to identify the pass images used. These participants even knew about the algorithm being used. This demonstrates the layers of security to this method due to the numerous scenarios that the user can satisfy.

The implementation of this method would allow for some interesting experiments, for example simulating how long it would take for a shoulder surfer to know the two pass images even in the best-case time. User testing would also help evaluate the usability of this method in comparison with the other graphical password methods.

2.4 Text and Colour Grid Password

M. Thirunavukkarasu [10] proposed a graphical password to stop different attacks, such as dictionary attacks, spyware attacks, phishing attacks, and shoulder surfing; those of which conventional authentication is vulnerable to. The user enters a “pattern password”, which is a 6-letter textual password, in registration. In the login stage, a 6x6 grid is shown to the user, with a character in each cell. Each cell has a colour, with 6 different colours appearing 6 times in the grid. Every uppercase letter and number will populate the grid, each with their random colour.

To log in, the user must locate each character of their password, and enter the first letter of that colour. There is another phase, where another grid appears, and the user enters the letter of the colour through which his “imaginary pattern line” passes. This will not be implemented in the project.

No experiments were made in the paper to show the security of the software, including the shoulder surfing resistance. The concept of the user only needing to know a textual password rather than a graphical element for a graphical password made it an interesting choice for implementation. It also has a mechanism similar to that of the Text and Colour Wheel [6], so evaluating these two methods side-by-side would provide a good understanding of what elements are good to keep in a graphical password.

2.5 Propositions not implemented

Many propositions were under consideration, as described in the Interim Report [11] [12] [13]. The following subsections describe methods that were not implemented in the project.

2.5.1 Free Drawing Password

Lin et al [11] proposed a method where the user can draw several sample drawings, and this is compared with the user’s password drawing. The individuality of handwriting and

the act of drawing consistently motivated the idea for this password mechanism. For the experiment of their model, they had relative success which was larger with the number of draws. Shoulder surfing was not a factor considered in this paper and with this, a few issues had arisen. Should shoulder surfing occur, an attacker with good image-recognition would be able to easily remember the drawing, and would therefore be able to log into the user's account with several tries. The method was not considered for implementation due to these large issues, and that developing this method would take too long in the time frame needed to implement all these different methods.

2.5.2 PassPoints Password

Wiedenbeck et al [12] proposed a scheme where an image is provided by the system or the user, and the password relates to certain points of the image that the user clicked on during registration. These "pass points" can create a much larger password space than that of a textual password, as they can be placed anywhere within a huge number of pixels rather than a 64-character alphabet. These pass points can be securely stored via hashing, however, the nature of the password is again not suitable to avoid shoulder surfing. It is likely a user will pick memorable parts of the image, those that "stand out" and become easier for an attacker to memorise. In this case, a finite number of tries would be needed for a shoulder surfer to pick out all the pass points and authenticate themselves. To implement this method would be a big task compared to other methods, and with the issues already surrounding shoulder surfing, it would not be suitable for evaluation in my project.

2.5.3 Grid Image Based Method

The previous two propositions lacked shoulder surfing resistance, as the user input was direct and gave away the password immediately on screen. The proposition by Ariffin et al [13] follows this same suit. In registration the user is required to select cells within a 2D grid that overlays an image. During login, this same process happens and a comparison is made with the cells that are previously stored. With this process, the shoulder surfer can observe which cells are selected, in turn exposing the users password to the attacker. Overall, these vulnerabilities do not make the method worth exploring in this project.

2.6 Motivation

The methods selected for implementation all have something in common: the login process follows some process indirect to that of the registration, using pseudo-random elements and forcing the user to understand how the mechanism works, before knowing how these elements will be utilised for authentication. This will stop shoulder surfers from being able to use the input for the next login, as another pseudo-random set of elements will be displayed. However, layers of security possibly start to fade when the mechanism is made available to the shoulder surfer. All these different shoulder surfing scenarios should be evaluated along with the usability of the graphical password method in question to determine its value in authentication, and whether graphical passwords should at all be further researched.

Chapter 3

Software Requirements

This chapter magnifies the objectives laid out in section 1.2, setting out software development requirements that must be met for the project implementation to be successful.

3.1 Functional Requirements

3.1.1 Graphical Password Selection

1. A “hub” where the user can choose an approach
2. User can request to go straight to login, in the case that their details are already registered

3.1.2 Graphical Password Registration

For All Methods

1. Request username
2. Request password; this will be the main password connected to their profile
3. Store registered data in the database; in appropriate tables
4. Handling of already registered users; only one set of details for each approach for each user

Colour Grid Method

1. Request pattern password; must be 6 letters

(Digraph) Image Grid Method

1. Display images to be selected
2. Request user selects two of these images to be pass images

Coin Password Method

1. Display all elements to be selected; 10 images, 10 numbers, 10 colours
2. Request the user selects several elements, minimum of 2 elements each, as their coin password

Colour Wheel Method

1. Display the colours that can be selected
2. Request the user selects a colour and a password within the range of characters displayed on the wheel

3.1.3 Graphical Password Login**For All Methods**

1. Retrieve username and password, check against database
2. Fetch the selected graphical password method details to feed into it's login window; if the user exists and the password is correct

Colour Grid Method

1. Display a 6x6 grid, each cell with a random colour and characters
2. Each of the 6 colours should be distributed evenly (6 times) across the cells
3. Place all characters, alphabet and numbers, inside these cells
4. Retrieve the entry by the user, and compare this with the mapped pattern password

(Digraph) Image Grid Method

1. Display a grid of images, the size determined by the user (see 3.2.2)
2. Calculate the possible images the user can click on based on the scenario dependant on the positioning of the pass images
3. Retrieve the image clicked on by the user; allow the user to login if it is a correct possible image

Coin Password Method

1. Generate a random array of "coins" for each user entry, each coin including 1 of each random element – no duplicate elements should be displayed
2. Retrieve the coin that the user has clicked and check if one of the elements is the next element of the user password; if not give an incorrect message and reset the login
3. Check if the user has entered all the correct elements and confirm the login was successful

Colour Wheel Method

1. Display a wheel composed of 8 sections, each with a different colour and list of 8 characters

2. Allow the user to rotate the wheel, either the colours or the text, so that the user can combine their selected colour with a list of characters in which their next password character lies
3. Allow the user to click a confirm button, which will check if their next password character is within their selected colour; if so, increment to the next password character
4. Allow the user to click login where the current user entry is compared with the user's colour wheel password; give a correct or incorrect message

3.1.4 Graphical Password Analysis

For All Methods

1. A summary of what the shoulder surfer would be able to gather from the user's login
2. Time taken to login
3. Combinations the password can produce - if applicable

(Digraph) Image Grid Method

1. Imitate a shoulder surfer (who knows the mechanism) gathering all the images from each scenario for each login, until the two pass images are discovered

Coin Password Method

1. Imitate a shoulder surfer (who knows the mechanism) gathering each element from each coin that is input; with each incorrect input comparing the previous attempts and keeping the elements which reappear

3.1.5 Database Data Integrity and Security

1. No duplicate users; each user designated a unique public key
2. No duplicate graphical password login details for each user; users public key can only be linked to one set of login details for each method
3. No real passwords should be used; in case of this issue, main user passwords should be stored securely
4. User should be able to set the database address, name and password for their system

3.2 Non-functional Requirements

3.2.1 Usability

1. Details of each graphical password method must be explained for the users understanding
2. Easy navigation between parts of the suite

3. Program should be a suitable size, re-sizable if appropriate

3.2.2 Graphical Password Scalability

1. Allow users to change certain factors of the mechanisms where appropriate
 - (a) (Digraph) Image Grid Method: Grid size, images used
 - (b) Coin Password: Password length (minimum of 6)
 - (c) Colour Wheel Password: Password Length

3.2.3 Software Performance

1. Registration and login functionalities must behave similarly to how they do in the real world
2. Algorithms should be made with efficiency in mind, as this software will be used on different machines with different performance

Chapter 4

Specifications and Planning

This chapter gives an account of the design phase of the project. The use of diagrams help to illustrate the structure of the system and the functionality of the login phases for each researched approach.

4.1 Language and Architecture

The language of choice for this project was Java [14], due to its familiarity and common application to similar projects of this nature. It was important to choose a language that had built-in libraries to aid the production of graphical user interfaces: such as the Java `swing` and `graphics` packages. It also includes the JDBC (Java Database Connectivity) interface, which uses a driver to connect to databases and store data. Java is widely used for web and application design, so it would be the language likely used by the industry if they were to implement the graphical password seen here. Finally, the cross-platform capability brought by the JVM (Java Virtual Machine) would also allow for the project to be worked on multiple machines and operating systems. For storing the user registration data, `mysql` [15] was used also due to its familiarity and association with JDBC.

In terms of architecture for the project, the Java `swing` package uses the MVC (Model-View-Controller) architecture in a simplified capacity for its core design [16]. There is less focus on a MVC-like architecture for the overall software, instead, each `JFrame` will include the required functionality, and other classes will be used where appropriate. The `swing` package was an unfamiliar concept at the start of the project development, so focusing on learning the application of it was of more importance, rather than being limited to architectural constraints.

4.2 Database Structure

Figure 4.1 is a class diagram which illustrates how the database is structured.

This is a simple database structure which connects the user to one of each approach. These one-to-one relationships are to stop the user from creating multiple logins for each method, as only one username and password should be associated with each account. The different approaches do not have any association to one another.

Each approach has it's own table which includes a foreign key "userID". As this is unique, it removed the need for a primary key, and maintains the tables as atomic. Each table also stores the data relating to the individual registration and login requirements as seen under subsections 3.1.2 and 3.1.3.

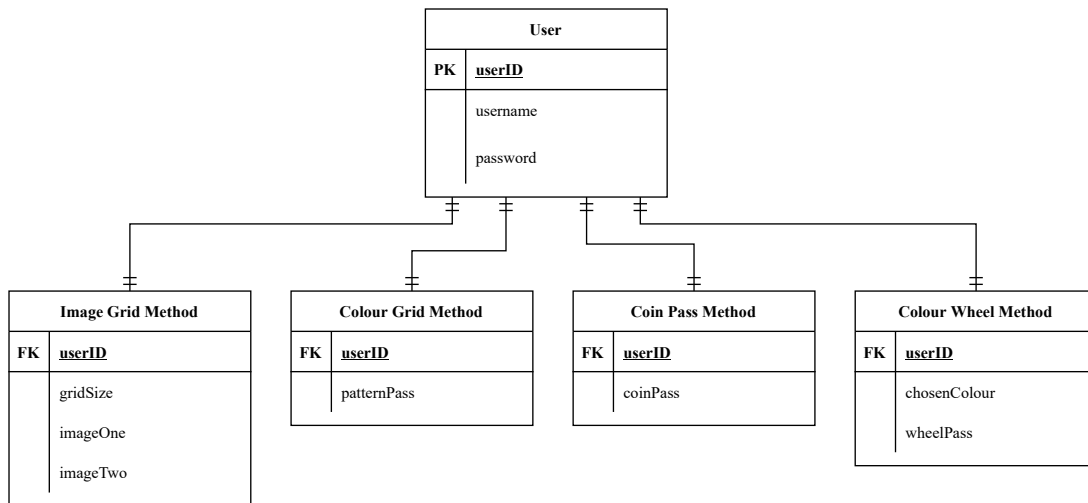


Figure 4.1: A class diagram to show the structure of the database.

4.3 Structure of The Requirements

The following Figure 4.2 is used to visualise the requirements by showing the different scenarios that the user can partake in and its relation to the system.

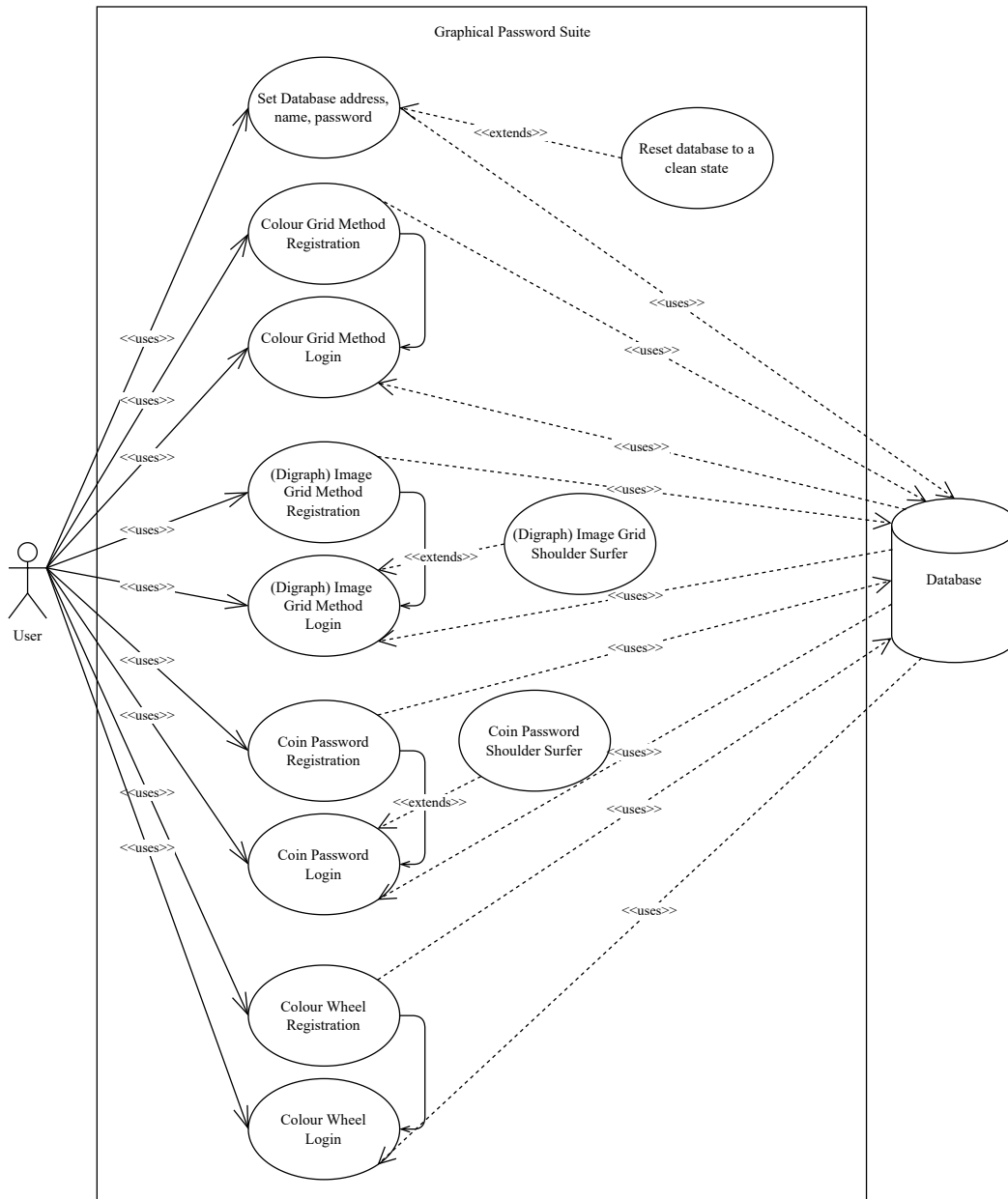


Figure 4.2: A use case diagram to visualise the requirements.

4.4 Suite Navigation

Figure 4.3 below shows how the different frames of the software will relate to one another and how the user can navigate them. The user starts by entering the details of the database they will be using. This includes the JDBC URL of the MySQL server, for example `jdbc:mysql://localhost:3306/`, the username and password.

The user will then be taken to the welcome page, which is the main source of navigation for all approaches. Each of these methods include a registration and login page, with the addition of a shoulder surfer or shoulder surfing analysis popup.

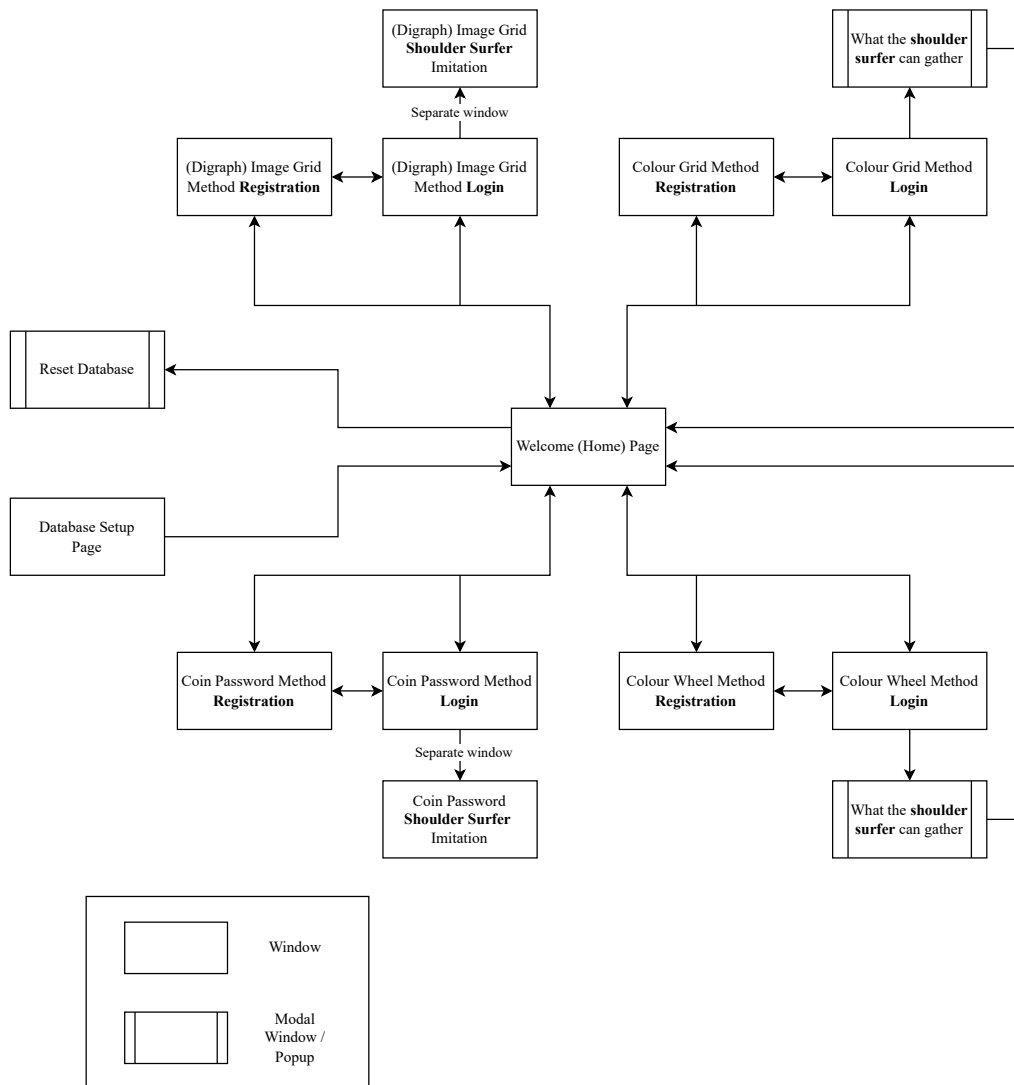


Figure 4.3: A flow chart to show the navigation process of the suite.

4.5 Login Methods

This section shows a demonstration of how the login process for each method will be implemented in my software.

4.5.1 Colour Wheel Method from 2.1

This section gives a demonstration of the user entering the first letter of their password. There are different steps needed to take, explained in Figure 4.4.

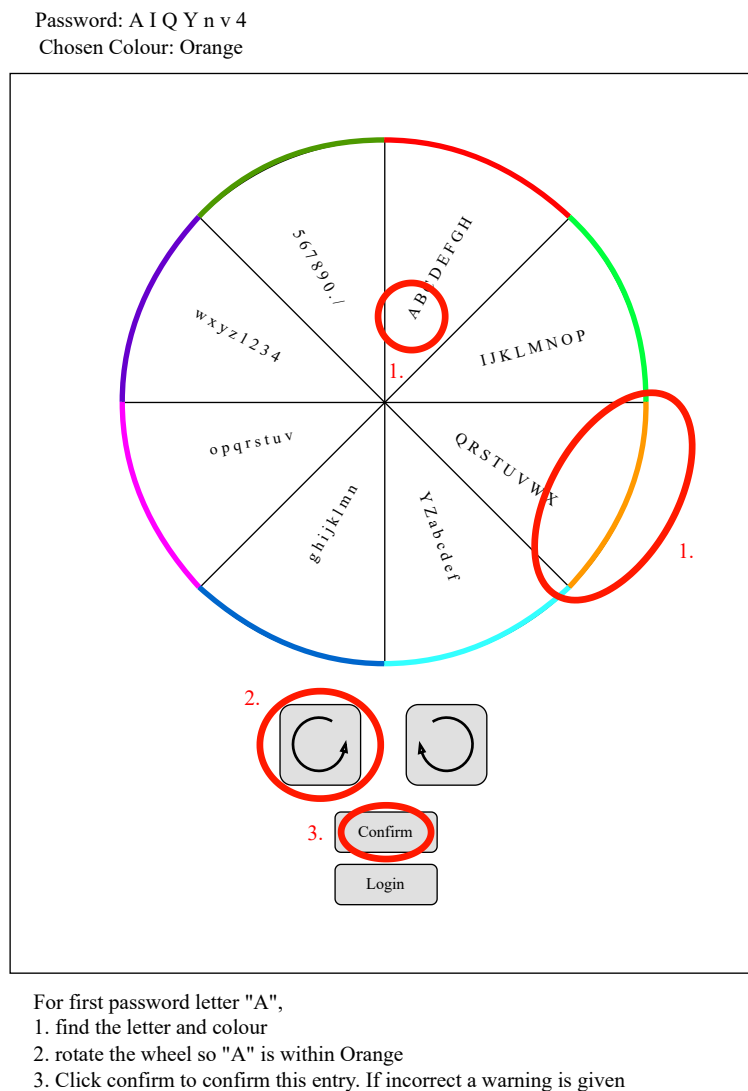


Figure 4.4: A mock-up of how the Colour Wheel Method login will work.

4.5.2 Coin Password Method from 2.2

Figure 4.5 shows a demonstration of a user attempting to login with a coin password. The order of which coins to click is displayed. To note, in practice, every time a coin is clicked a new array of randomly generated coins is displayed.

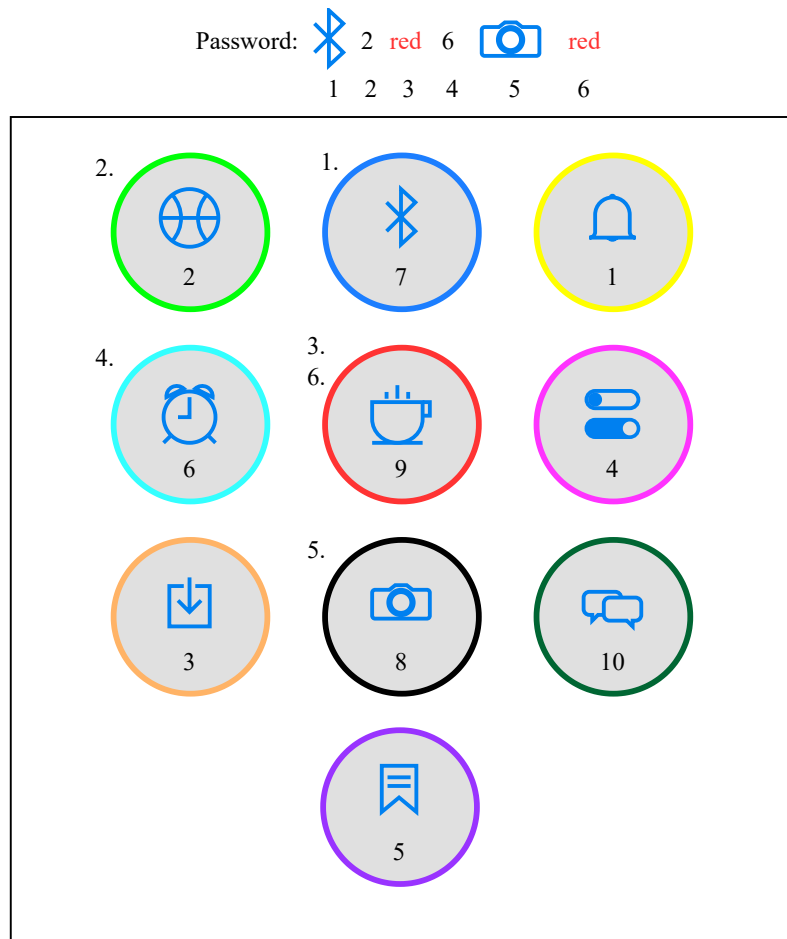


Figure 4.5: A coin password mock-up, and which order of coins the user should click.

4.5.3 (Digraph) Image Grid Method from 2.3

This section shows the process of logging in using the different scenarios that fit their randomly placed pass images, seen in Figure 4.6

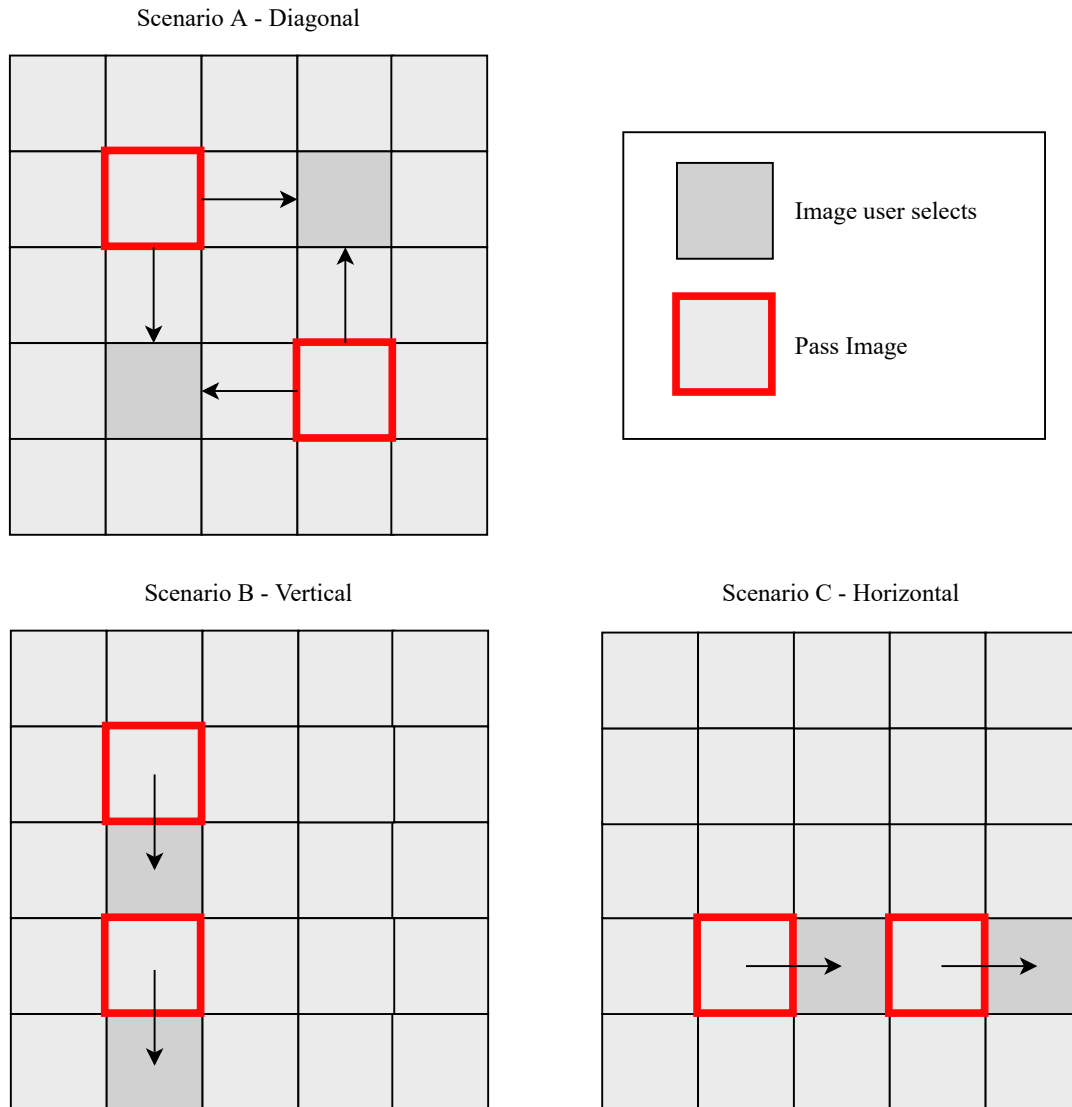


Figure 4.6: A mock-up of how the Image Grid Method login will work.

4.5.4 Colour Grid Method from 2.4

The Figure 4.7 shows how a user with the password "ABCDEF" would login with this pseudo-randomly generated grid.

The figure illustrates the Colour Grid Method login process. It consists of two main parts: the initial grid and the resulting password extraction.

Initial Grid: A 6x6 grid of colored squares. The colors are: Red, White, Red, Green, Yellow, Blue (Row 1); Green, Yellow, White, Pink, White, Pink (Row 2); Yellow, Blue, Green, White, Blue, Pink (Row 3); White, Pink, Yellow, White, Pink, Green (Row 4); Pink, Red, White, Red, Green, Blue (Row 5); White, Red, Blue, Red, Green, Blue (Row 6). The letters A-F are in the first row, G-L in the second, M-R in the third, S-X in the fourth, Y-Z in the fifth, and 0-9 in the sixth.

Password Input: Below the grid is a 'Password:' label and an input field. A 'Login' button is below the input field.

Resulting Password: An arrow points down to a second grid where the selected cells are highlighted in their original colors. The password 'JBH130' is entered in the input field.

Grid Details:

A	B	C	D	E	F
G	H	I	J	K	L
M	N	O	P	Q	R
S	T	U	V	W	X
Y	Z	0	1	2	3
4	5	6	7	8	9

Below the grid is a 'Password:' label and an input field. A 'Login' button is below the input field.

Below the grid is a 'Password:' label and an input field. A 'Login' button is below the input field.

Password is J- Pink, B- White, H- Yellow, 1- Red, 3- Blue,
0- White

Figure 4.7: A mock-up of how the Colour Grid Method login will work.

Chapter 5

Implementation

This chapter explains how the different parts of the solution were implemented. Section 5.1 describes how the database was set up alongside the rest of the software and how the user could utilise a database of their own. Section 5.2 describes how the GUI was formed using `java.Swing`, `awt.Graphics` and images from different sources. Section 5.2.2 describes the functions and variables used for the application of each approach.

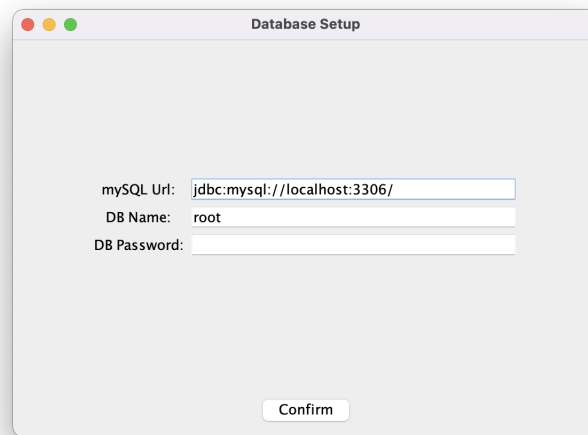
5.1 Database

The software was made using a local MySQL database, and in early development MySQL Workbench [17] was used to create the scripts and view the tables so that they were working as intended.

The details of this local database were hard coded, which had to be changed for any user to utilise their database. Therefore, an interface was created to fetch the database details given by the user and attempt to connect. Subsection 5.1.1 describes the steps taken to accomplish this. Subsection 5.1.2 shows the class which was used to store these details and run scripts.

5.1.1 Database Setup

Figure 5.1 shows the prompt given to the user when the software first runs. They enter the JDBC URL, name and password of the MySQL server used to host the database. Connections are only opened when necessary, so the details are passed throughout the system so they can be used when a connection wants to be made. This is stored in the `DatabaseRunner`, explained in subsection 5.1.2. Subsection 5.1.3 describes what steps were taken when a MySQL query needed to be made.



A screenshot of a macOS-style dialog box titled "Database Setup". The dialog has a light gray background and a title bar with three colored window control buttons (red, yellow, green) on the top left. Inside the dialog, there are three text input fields stacked vertically. The first field is labeled "mySQL Url:" and contains the text "jdbc:mysql://localhost:3306/". The second field is labeled "DB Name:" and contains the text "root". The third field is labeled "DB Password:" and is currently empty. At the bottom center of the dialog, there is a single button labeled "Confirm".

Figure 5.1: A prompt to the user to enter mySQL details.

5.1.2 Database Runner

The class `DatabaseRunner.java` is used to store the details of MySQL server and database in use, and perform functions, such as dropping the database (if it exists) or creating the database (if it does not exist). These are used in `Welcome.java` when the user clicks on the "Reset DB" button. This has the double functionality to reinitialize the database or initialise it, if it does not exist.

The details stay the same after they are entered, so it would have been suitable for different classes to get these via a static variable. However, these variables are first updated through user input. Therefore, the decision was made to pass this class throughout the program. Figure 5.2 shows an example of `ColourGridRegistration.java` using this class.

```

1      ...
2      // passed from previous window; Welcome.java, ColourGridLogin.java
3      public ColourGridRegistration(DatabaseRunner dbRunner) {
4          ...
5          this.dbRunner = dbRunner;
6      }
7      ...
8      if (btn.equals(registerBtn)) {
9          ...
10         // details inserted from DatabaseSetup.java
11         String url = dbRunner.getDburl();
12         String dbname = dbRunner.getDbname();
13         String dbpass = dbRunner.getDbpass();
14         ...
15     }

```

Figure 5.2: Examples of the DatabaseRunner being used.

5.1.3 Database Queries

During both registration and login, queries were made to authenticate the user. The `java.sql` library was applied for this problem, as the `PreparedStatement` object could be used to set up and execute a query with the given details from the user. Once executed, a `ResultSet` object would either return nothing (not authenticated) or the data selected (successful). Queries are made like so:

```

1      String query = "INSERT INTO user(username,password) values('" +
2          ↪ username + "','" + password.hashCode() + "')";
3      Statement statement = connection.createStatement();
4      int x = statement.executeUpdate(query);

```

Figure 5.3: How a PreparedStatement is constructed in java.

5.2 Graphical User Interface

This section describes the methods used to create the GUI. Subsection 5.2.1 describes how the `java.swing` library was used to create the different graphical components. 5.2.2 shows the main interface where the user can select a method, the registration forms and initial login form. 5.2.3 explains how local and via network images were fetched and used, and 5.2.4 describes how the `java.awt.Graphics` package was used in particular cases.

5.2.1 Using Swing

The majority of the classes created extend the `JFrame` class. It works as the main window where other swing components can be added, such as `JPanels` and `JLabels`. The route was taken to use a `JFrame` -> `JPanels` -> `JLabels`, `JButtons`, `JTextFields` architecture. This meant a main `JPanel` could be used for the form or login procedure, placed in the centre of the `JFrame` using `BorderLayout.CENTER`. For the navigation and functional buttons at the top and bottom of the window, these could be placed in a separate `JPanel` that is fixed to the north or south of the `BorderLayout`.

5.2.2 Selecting a Graphical Password

Welcome Page

The welcome interface is used as a hub for the user to see the instructions for each method, selecting one to register or login with. This selection is made using a drop-down box.

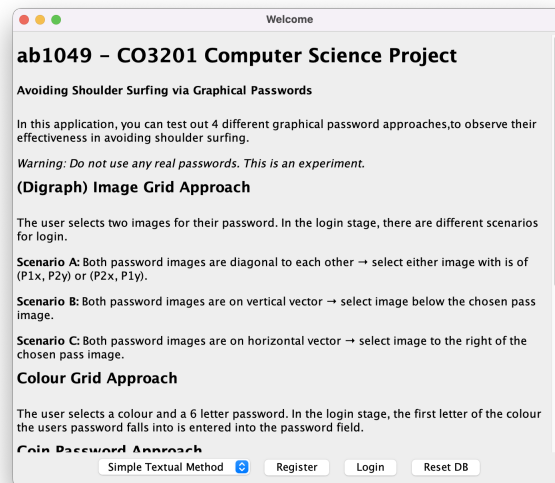


Figure 5.4: Welcome Page where the user can select a method to register or login.

Registration Forms

Each approach has its own registration form with the relevant input fields. Using the `GridBagLayout` manager, constraints could be added to these fields to organise the window effectively.

These classes implement the `ActionListener` class, which can receive action events (used for all windows with `JButtons`). Overriding the `actionPerformed` function allowed for the buttons in the interface to run a certain function once clicked. This was done by casting the `getSource` method to a `JButton`, using the `equals` method to see which `JButton` had been clicked and perform a function based on this. For the register button, a connection would be made to the database before an insertion function is used. Figure 5.5 shows an example of a registration form.

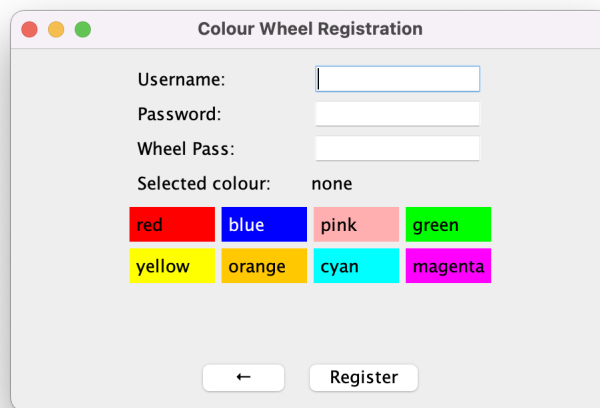


Figure 5.5: Registration form for the colour wheel method.

Initial Login Form

This form is used across all methods, used to authenticate the user's username and password, before taking the user to the specific graphical password login form. This was used to reduce the amount of repeating code across login forms. Using a `Method` enum helped the welcome or registration page inform the initial login page of which graphical password login it was to refer to.

The functionality and GUI of each specific graphical password login page will be discussed in their own subsection under section 5.3.

5.2.3 Getting and Displaying Images

The use of images in the Coin Password (2.2) and Image Grid (2.3) approaches meant that a method had to be found to get and store these images. The Image Grid method first used randomly generated images from the PicSum website [18], using `java.net.URL.openStream()`.

Due to circumstances explained later, an option for preset (local) images was provided. To allow these to be accessed by a runnable JAR file, the `class.getResource()` function was used.

To store the Image Grid *pass images*, the following approach is shown in Figure 5.6.

```
1      ...
2      // imageOne is a BufferedImage
3      ByteArrayOutputStream baosOne = new ByteArrayOutputStream();
4      ImageIO.write(imageOne, "jpg", baosOne);
5      InputStream isOne = new
        ↳ ByteArrayInputStream(baosOne.toByteArray());
6
7      // preparedStatement; set large amount of bytes to a blob
8      gridst.setBlob(3, isOne);
9      ...
```

Figure 5.6: How a `BufferedImage` is converted to a byte stream.

5.2.4 Graphics Package

No pre-made swing components were suitable for displaying the circular coins of the Coin Password, or the text and colour wheel. Therefore, new classes were made extending the `JComponent` class, which were “canvases” for the `paintComponent` method to utilise a `Graphics` object. This object was used to draw the component using various functions based on the parameters, such as the colour and text arrays (in their random order) for the colour wheel. Every time the wheel was rotated, the `repaint()` method could be used and the display would be updated.

```
1      ...
2      protected void paintComponent(Graphics g) {
3          // g2d has more functionality
4          Graphics2D g2d = (Graphics2D) g;
5          // smoothen the pixels
6          g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
7              ↪ RenderingHints.VALUE_ANTIALIAS_ON);
8          // for each circle section
9          for (int i = 0; i < 8; i++) {
10             // custom function to draw each sector
11             // including a colour and list of characters each
12             drawCircleSection(g2d, i, colourList.get(i),
13                 ↪ charLists.get(i));
14             }
15      }
```

Figure 5.7: Example of a `paintComponent` for the `WheelCanvas` class.

5.3 Graphical Password Method Implementation

The following subsections describe the implementation of the different .java files used to create each method.

5.3.1 Colour Wheel Method

ColourWheelRegistration.java

Along with the username and password fields, the wheelPass and chosenCol fields were also added. This was so that the user could set a separate password for the wheel within the boundaries of the characters used, and set a colour that they will be rotating their next character into for logging in. This colour selection was used by displaying the colours in a separate JLabel each, including a MouseListener which when clicked would select this colour and display it as the selected colour. The registration can be seen in Figure 5.5.

WheelCanvas.java

This class is tasked with painting the text and colour wheel itself, using the width, array of colours and nested array of characters passed into it. The Graphics2D package was used, explained in subsection 5.2.4, as it is suitable for drawing geometry such as Arc2D shapes filled with colours, and drawing text at certain rotations. Certain usability factors were brought into account, such as rotating the text 180° when it appears upside down, and setting any text within the blue sector to white, as black was not clear enough for the user.

ColourWheelLogin.java

The login page includes an object of the WheelCanvas.java class and JButtons which make use of the overridden actionPerformed function. The rotation buttons are given an anti-clockwise and clockwise icon respectively, and rotate the wheel using two simple commands shown in Figure 5.8.

The confirm button uses a nested for loop to find the sector of the user's chosen colour, and loop through each character, to see if any equal that of the current character of the user's password, indexed using passCount (which starts at 0). If correct, the passCount is incremented and the user can proceed with the login. If not, the user is given a warning via a JOptionPane.showMessageDialog window, with up to 3 warnings given as incremented by the warningCount variable.

The login button will compare the passCount with the passLength, and if equal give a successful result message to the user, explaining the shoulder surfing resistance whilst giving the time taken to login in seconds (using System.nanoTime()) and possible password combinations.

```
1     if (btn.equals(rotLftBtn)) {
2         Collections.rotate(collList, 1); // +1 to indicate
           ↪ anti-clockwise rotation
3         wc.repaint(); // this calls the WheelCanvas object to
           ↪ perform paintComponent again with the updated array
           ↪ of colours
4     } else if (btn.equals(rotRgtBtn)) {
5         Collections.rotate(collList, -1); // -1 to indicate
           ↪ clockwise rotation
6         wc.repaint();
7     }
```

Figure 5.8: How the wheel was rotated using Collections functionality.

```
1     ...
2     AffineTransform temp = g2d.getTransform();
3     g2d.setColor(colour);
4     g2d.fill(arc); // Arc2D shape
5     g2d.setColor(Color.black);
6     g2d.draw(arc); // draw black outline of arc
7     g2d.translate(rec.getCenterX(), rec.getCenterY());
8     g2d.rotate(-Math.toRadians(startAng)); //
           ↪ Math.toRadians used instead of degrees
9     g2d.translate(90, -50);
10    ...
```

Figure 5.9: Code snippets within the drawCircleSector function.

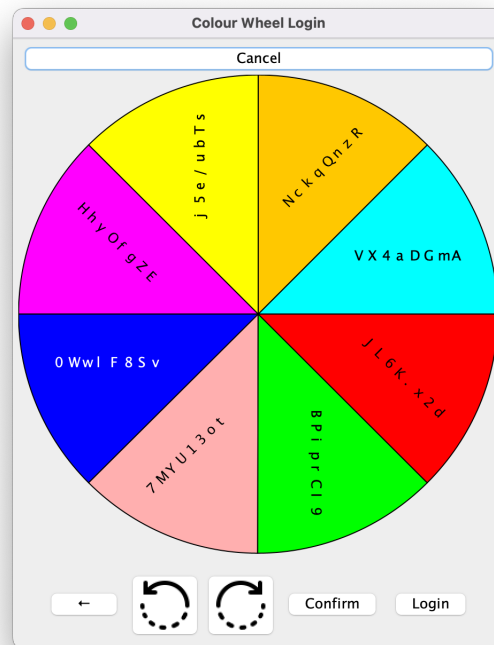


Figure 5.10: The login page for Text and Colour Wheel Method.

5.3.2 Coin Password Method

CoinPassRegistration.java

The mainPanel included a nested JPanel with all the elements displayed. Each element has a mouseClicked method which will use a map to get the string version, for icons and colours, or the string straight from the JLabel of the element for numbers. These entries are appended to a coinPassField, where its getText() string is passed into the database query. Other variables are used to count the amount of each type of element, so the suite can reject the user's password if it does not fall within the boundaries.

CoinCanvas.java

This was made to replace the square JLabels that were previously used to show the *coins*. The Graphics2D package was again used to draw a circle for each coin and each element based on what is passed into the constructor of the class.

```

1      Ellipse2D.Double circle = new Ellipse2D.Double(width, height, 80,
      ↪ 80);
2          g2d.setColor(Color.LIGHT_GRAY);
3          g2d.fill(circle);
4          g2d.setColor(col);
5          g2d.draw(circle);
6          g2d.drawImage(icon, (int)circle.getCenterX()-15,
      ↪ (int)circle.getCenterY()-17, (int)(width*1.5),
      ↪ (int)(height*1.5), null);
7          g2d.drawString(Integer.toString(num),
      ↪ (int)circle.getCenterX()-5,
      ↪ (int)circle.getCenterY()+25);

```

Figure 5.11: Graphics2D package used to create each coin, using an Ellipse2D object.

CoinPassLogin.java

The constructor calls the custom makeCoins method, which initialises the arrays to be randomised; including the icons which are fetched locally using a for loop. Each coin is created as a CoinCanvas object, which is given a MouseListener, so that once clicked a check will be performed to see if any of the 3 elements in the coin are equal to that of the first element being looked at in the user's coinPassElements array. If correct, the passEntry variable, which holds all of the current correct password entries, is updated and the first element of the coinPassElement is removed, making the next element the first in the array, and subsequently checked for the next entry when the coins are regenerated. If incorrect, the user is given a message, the coins and passEntry are reset so that the user has to start again. The CoinPassSurfer object is also updated with each user entry and will be discussed in the next subsection.

The login button will compare the passEntry to the coinPass, and if successful, a dialogue box is displayed. This gives the total time taken and combinations. The surferEvaluate

method is also run for the surfer object. If the login fails, the user is required to log in again, starting from the `InitialLogin` window.



Figure 5.12: The login page for the Coin Password method.

CoinPassSurfer.java

This is a separate window alongside the login, which updates as the user makes correct and incorrect entries. For correct entries, the `updateSurfer` method is used. The elements which have been viewed are stored in an array of local `Coin` objects. Multiple try and catch statements are used to remove any non-repeating elements if the coin at that index has already been viewed before. The remaining elements, if not all, are added to a `JPanel` which is then added to the `mainPane`.

For incorrect entries, the `restartSurfer` method is used, which removes all the displayed elements and resets the index of the current password element to 0 (the first element).

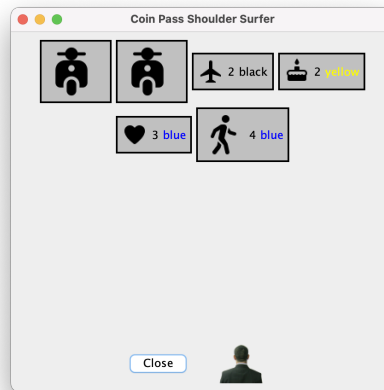


Figure 5.13: Shoulder surfing window whilst login is in process.

Once the user has completed their login, the `surferEvaluate` method is called, which removes all the elements displayed. The window is repainted with HTML text that discusses what the shoulder surfer could have gathered from that login, using info such as the length of the password, number of errors the user made and combinations.

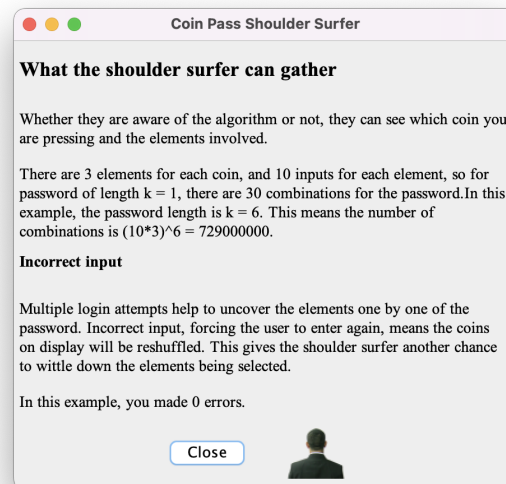


Figure 5.14: Shoulder surfing window after the login concludes.

5.3.3 (Digraph) Image Grid Method

ImageGridRegistration.java

The window includes a form which asks for the grid size and pass images as well as the username and password. The grid size uses a simple `TextField` where the user can enter a value between 4 and 8. To select the pass images, the user must click on two of the 6 generated images. If the user would like to select new images, clicking on another image will prompt the user if they would like to reset their pass images or not. When the register button is clicked, these details are inserted through a query. This process converts the `BufferedImage` pass images to bytes, shown in subsection 5.2.3.

In late development, it was found that using randomly generated images from PicSum [18] made it easy for the shoulder surfer to find the pass images within 2 logins, as they were the only images that kept reappearing. Therefore, an option was added in registration form for the user to continue using random images or use 64 preset images from a local folder. This was intended to make it harder for the shoulder surfer. The `randomOrPreset` column was added to the `image_grid_method` table in the database, and the queries in registration and login classes had this variable appended to them.

PassImage.java

This class is used to store the x and y values of the pass images and valid entry images used in the `ImageGridLogin.java`.

ImageGridLogin.java

The constructor runs a `makeGrid` method, which first fetches the images based on the random or preset option that the user selected in registration. It then uses a `for` loop from 0 to `gridSize2`, creating each `JLabel` with a random image and `MouseListener` set to it. It is then added to a `labelList` 2D array, while the image is set to an `imageList` 2D array both based on the x and y coordinates. Two `PassImage` objects are also created for their counterparts given in the parameters of the constructor, so that their specific x and y-axis can be checked up later. The `mouseClicked` method will run a nested `for` loop to search through each row and column of the grid, and when the x and y coordinates of the clicked `JLabel` are found, `IF` statements are used to perform the different scenario checks (as seen in Figure 5.15). The two possible images that the user could correctly click are temporarily stored as `PassImage` objects so that their x and y coordinates are saved. These coordinates are compared to those of the clicked `JLabel`, and if correct, the `runSurferScenarios` method is run.

The `runSurferScenarios` method calculates the possible images that the shoulder surfer could see as a pass image. If this list of images is larger than 2, a new grid of images is displayed and the user logs in again. These new possible images are compared with those before, and the repeating images remain. The `ImageGridSurfer` object is updated after every login with the new list of possible images. Once the size is 2 or smaller, the `surferEvaluate` method is called and the user is taken back to the `Welcome` page.

```

1      // get this label position
2      if (labelList[x][y] == label) {
3          // horizontal line
4          if (I1.x == I2.x) {
5              // at edge
6              if (I1.y+1 == gridSize) {
7                  P1 = new PassImage(I1.x, 0); P2 = new
9                  ↪ PassImage(I2.x, I2.y+1);
8              } else if (I2.y+1 == gridSize) {
9                  P1 = new PassImage(I1.x, I1.y+1); P2 = new
11                 ↪ PassImage(I2.x, 0);
10             } else {
11                 P1 = new PassImage(I1.x, I1.y+1); P2 = new
13                 ↪ PassImage(I2.x, I2.y+1);
12             }
13         }
14         // vertical line
15         else if (I1.y == I2.y) {
16             // at edge
17             if (I1.x+1 == gridSize) {
18                 P1 = new PassImage(0, I1.y); P2 = new
20                 ↪ PassImage(I2.x+1, I2.y);
19             } else if (I2.x+1 == gridSize) {
20                 P1 = new PassImage(I1.x+1, I1.y); P2 = new
22                 ↪ PassImage(0, I2.y);
21             } else {
22                 P1 = new PassImage(I1.x+1, I1.y); P2 = new
24                 ↪ PassImage(I2.x+1, I2.y);
23             }
24         }
25         // different x y coordinates
26         else {
27             P1 = new PassImage(I1.x, I2.y); P2 = new
28             ↪ PassImage(I2.x, I1.y);
29         }
30         // check if this image is a pass image
31         if ((x == P1.x) && (y == P1.y)) {
32             runSurferScenarios(P1.x, P1.y, label, recreate);
33         } else if ((x == P2.x) && (y == P2.y)) {
34             runSurferScenarios(P2.x, P2.y, label, recreate);
35         }
36     }

```

Figure 5.15: The checks made for Scenarios A, B and C.

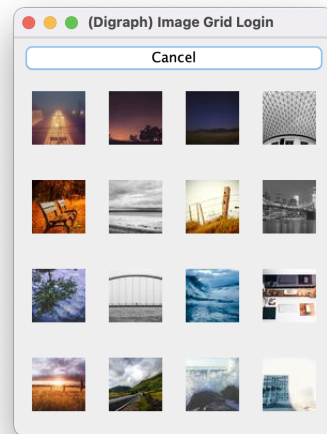


Figure 5.16: The login page for the (Digraph) Image Grid method.

ImageGridSurfer.java

This separate window displays the possible images passed into it by the `ImageGridLogin` with the use of the `updateSurfer` method. The `surferEvaluate` method repaints the window with HTML text that describes what the shoulder surfer could gather, accounting for the number of entries needed to narrow down the pass images, and whether the user selected to use random or preset images.

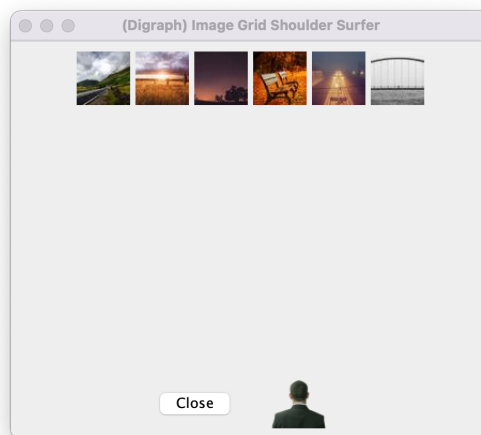


Figure 5.17: Shoulder surfing window during the logging in phase.

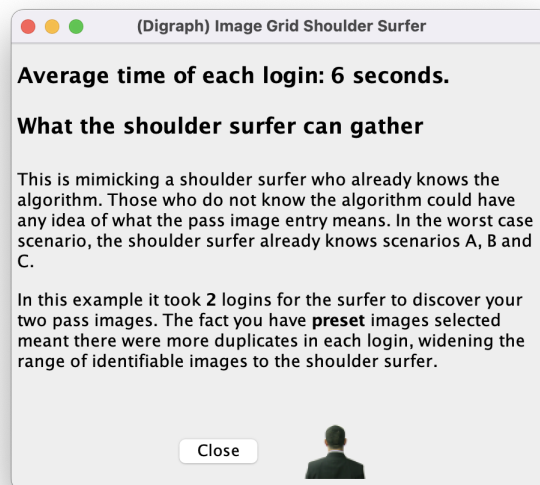


Figure 5.18: Shoulder surfing window after the login phase.

5.3.4 Colour Grid Method

ColourGridRegistration.java

This class performs a simple registration of the username and password along with the 6 character `patternPass`. This allows all letters and numbers.

ColourGridLogin.java

The constructor calls a `makeGrid` method which generates 36 `JLabels` with a random colour. Each character is set as the text in order of the grid as seen in Figure 5.19. The `ppResult` holds the correct order of colours in which the user is required to enter. This is calculated using a `Map` of characters and colours to store each character's colour, which is accessed using the index of the `patternPass` word character; this is then converted to a single letter string, which represents the first letter of the corresponding colour. To keep an even amount of colours on the grid, an array with 6 of each colour is created, so that when each `JLabel` is created, the colour that is selected, is then removed from this array. The user makes their entry of the `patternPass` colour letters via a `JPasswordField`. When they select to login, this input is compared with the `ppResult`. If successful, a `HTML` dialog is displayed, with a description of the what a shoulder surfer could gather, and in how many seconds. If unsuccessful, a incorrect dialog message is shown to the user.

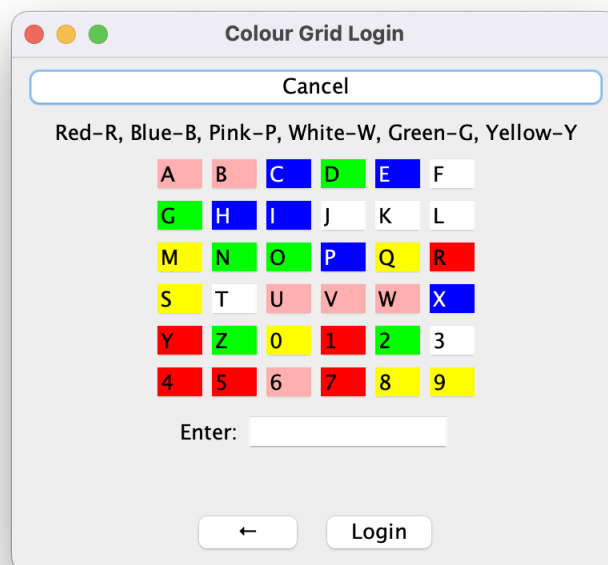


Figure 5.19: The colour grid login process.

Chapter 6

Testing

This chapter describes the various modes of testing used to assess the quality of code, and the user testing of each graphical password.

6.1 Unit Tests

Unit Testing is a powerful tool to ensure the quality of code during product development cycles. It helps speed up the debugging process by breaking down the software and testing the solutions to sub-problems. Once unit tests are written they can usually be kept the same for the duration of product development, making testing more time-efficient. [19]

JUnit is a framework for testing in Java and provides the developer with a simple way of integrating unit tests into their program [19]. It was an easy choice to use for this project as it is written in Java and is simple to use.

Each test class is stored within the `eclipse.testing` package. Using the `@Test` annotation, methods within each class tested valid inputs, invalid inputs and functions of the different registration and login classes for each method. The tests carried out used connections to the database directly, whereas it would've been better to use mock objects to showcase the different states of the database. This is so any real data gained would not be wiped by the testing classes. In any case, the tests still proved to be widely successful and there were no issues with data handling in the software. One failed result in `ColourGridTest` was due to the `makeGrid` trying to access an empty list, so the list was reinitialized every time the function was called.

An issue that became clear during this testing phase was that there were too many dependencies between the main logic and the GUI. The lack of separation made it hard to isolate tests on the background logic without calling for the use of GUI input, and even some of the functionality itself was baked into some GUI components; void methods which could not be used with assertion methods from the JUnit package.

The unit test results can be seen in appendix section A.

6.2 Integration Tests

The issue stated in section 6.1 meant that further testing was needed from the user and GUI perspectives. Black-box integration testing was selected for this, as less emphasis was placed on the internal logic of the system, rather the system requirements for each method and end-user. Test cases involved user navigation functionality via buttons, and the functionality of each registration and login window to make sure they act as intended by their original authors [6][8][5][10] and through the requirements.

The integration tests are included in the appendix section B.

6.3 User Testing and Feedback

To assist in the evaluation of the usability and shoulder surfing resistance of each graphical password approach, a few volunteers took part in a test (45 min approx). The user was first given 5 attempts to shoulder surf the conductor for each approach, without knowing the mechanism behind them. They were then explained each method and could perform registration and login themselves, providing comments on usability and security. For the final test, they were given 5 more attempts to shoulder surf the conductor, this time using the knowledge of each mechanism that they had learned. "Yes" was answered if the shoulder surfer was able to log in as the user after watching them, and "No" if they were incorrect, even if partially.

The structure of these tests were intended so that each approach could be evaluated from a security standpoint of a single by-stander attack, or through multiple attacks led by a malicious shoulder surfer. Letting the users try out each registration and login process allowed for better feedback on each method from a sample end-user point of view. It also solidified their understanding of each method before they attempted to shoulder surf again.

The full results of these user tests can be found in appendix section C. These results are discussed in Chapter 7.

Chapter 7

Results and Evaluation

This chapter goes into detail on the benefits and drawbacks of each approach, and how they compare through the aspects of security and usability. These observations are aided by the user testing and feedback as described in section 6.3.

7.1 Security

7.1.1 Resistance Against Bystander Shoulder Surfers

In the case of a one-time shoulder surfing attack, all approaches boasted near perfect results at preventing logins from unauthorised users. As seen in Table C.17, without knowing the mechanism, no users were able to crack the password on the first attempt. This is because each methodology has its password hidden behind functionality that only the user can work out for their password. By only looking at the login phase once, there is no prior information to compare to the current pseudo-random generation of the login phase.

After learning the methodology however, the Coin Password [8] method decreased in resistance. The attacker was able to guess the password once due to a series of errors committed by the user. This security flaw is due to the incorrect entry tactics in place, as for this implementation the user is brought back to the start of the login phase with a newly generated series of coins. Therefore, the shoulder surfer is instantly able to narrow down which elements had reappeared. With enough errors, as seen here, every element of the password can quickly be narrowed down. The shoulder surfing resistance of this methodology depends on the user being able to maintain a small number of errors during entry. When the conductor made no errors, there was an observed shoulder surfing resistance of 100%.

7.1.2 Resistance Against Multiple Shoulder Surfing Attacks

When protecting against multiple attacks from a shoulder surfer who does not know the methodology, all approaches continued to show upwards of 95% resistance. Table C.18 shows that every approach, bar the Image Grid [5], showed a resistance rate of 100%.

When User B was shoulder surfing the Image Grid method, they managed to make 3 successful attempts at guessing the conductors password. They explained that this was due to a informed guess they made that the images appeared in a certain way, and that the conductor was clicking a certain image a number of images below. This evidently is not how the functionality works, but it shows how an informed guess from someone who does not know the mechanism might be able to guess the password correctly. We can see that even once User B was explained the mechanism, their surfing success rate fell to 0% as shown in Table C.8.

The more obvious scenario for shoulder surfing is that the methodology is already known, especially if it is used for a common system such as mobile or cash machine authentication. Therefore, it is important to prioritise the worst-case scenario where a malicious shoulder surfer will already know the methodology in place. In this scenario, the complexity of the Image Grid methodology is still a "hindrance" to the shoulder surfer, showing high resistance during testing. However, some of the participants were also confident that after more attempts, they would still be able to eventually crack the password.

The Colour Wheel [6] methodology proved to be the most secure against shoulder surfing. Even when the users chosen colour was made available to the shoulder surfer, they could not make any progress in discovering the password after 5 attempts. As stated in Table C.16, user D could not shoulder surf due to "so many permutations", and users C and D made a similar statement regarding the need for "eye-tracking" to see what characters the user was focusing on. The Colour Grid [10] method is relatively similar in its tactics, and for a shoulder surfer to be successful, they would have to have already memorised the 6 letters for each colour that is entered by the user. Due to its pseudo-random nature, some logins may be easier to observe. For example, if every pattern pass character was a white character, the input would be "WWWWWW" and the shoulder surfer would only have to memorise these 6 characters.

The Coin Pass [8] methodology showed 100% and 95% resistance rates when the participants were unaware and aware of the methodology respectively. Despite the stats similar to that of the complex Colour Wheel security, participants were more confident at being able to guess the password over time, due to the decreasing password space as elements quickly become apparent in the user's password.

7.1.3 Password Entropy and Brute Force Attacks

The Colour Wheel [6] has an extremely large password entropy (mentioned in section 2.1) compared to the other approaches. The password length can be observed using the number of "Confirm" clicks, but the large password space of textual authentication is replicated here with the 64-character typeset, and its security is further increased due to the 8 colours used alongside. Even using a password length of 6, the number of combinations a Wheel Pass could produce is $8 * 64^6$. Thus, a brute force attack would not be a sufficient method for an attacker to gain access and the approach is very secure in this respect.

The Coin Pass [8] has many more combinations than that of its predecessor, the numerical pin password, with the combinations rising from 1,000,000 to $(3 * 10)^6 = 729,000,000$ for a password length of 6. The password length is easy to observe, however, by looking at each mouse click and coin regeneration on-screen. Furthermore, the number of combinations is still very low compared to that of an alphanumeric password, which for a length of 6 and 64 characters would produce $64^6 = 68,719,476,736$ combinations. The nature and space of this password would make it suitable for mobile authentication or even an ATM which requires a physical bank card, as that reduces the necessity for an extremely large password space¹.

The large number of bytes used to store an image makes it ideal for hashing in a database and preventing dictionary attacks. However, when physically logging in using the Image

¹It is difficult for an attacker to get hold of a users physical card. In a virtual environment, a 4 digit pin becomes much less secure as brute forcing attacks can be more efficient.

Grid [5] method, an attacker could gain access by simply clicking on a random image in the grid, with the chances increasing the smaller and more usable the grid gets (1 in 6 chance for a 3 by 3 grid). Therefore, a brute force attack would be feasible and there would certainly need to be more constraints to this methodology: for example requiring multiple logins at once, or more than 2 pass images to widen the pool of pseudo-random selection. As stated in subsection 5.3.3, decreasing the pool of images generated increased the number of attempts needed for a shoulder surfer to crack the password. This is because images separate from the pass images are more likely to reappear and continue to be considered possible pass images for a longer period.

The pattern password of the Colour Grid [10] itself has $36^6 = 2,176,782,336$ combinations. However, for the entry itself, each pattern pass character has 6 possible colours, meaning the number of combinations is $6^6 = 46656$. However, a brute force attack will not work in this way because for every login, the grid is completely randomised. Therefore the attacker could make a 1 in 46656 attempt, and even if they were authorised, the pattern password would still not be revealed, and would take a computationally long time to brute force.

7.2 Usability

7.2.1 Word Processing and Retention Difficulties

Current passwords as we know them emphasise being able to perfectly recall an alphanumeric string [20]. This can be challenging for those who are less able to quickly recognise characters, such as those with visual impairments or dyslexia [21]. The Colour Wheel and Colour Grid approaches use text pseudo-randomly in their methodologies together with colour. The cognitive ability needed to distinguish password characters from a memorised colour in a certain order would mean these methods would take much longer for some individuals, therefore becoming less accessible. The use of colour and images in the Coin Pass method would be better suited, however, the use of different modes of information (including numbers) would be difficult to memorise and recall, even for those without such disabilities. The affordances of the Image Grid methodology provide the most accessibility, as no alphanumeric input is required, only the memorisation of 2 images.

7.2.2 Considerations For Colour

Colour is a mode of information used heavily throughout the researched approaches. One of the participants in user testing was colour blind, which provided the ability to assess colour accessibility from a more accustomed point of view.

During registration of the Colour Wheel and Coin Password methods, the user is required to select colours. The naming convention is important, especially for the Colour Grid method, which requires an input of the first letter of the names of the colour it represents. This comes down to the discretion of the developer, and it may be worth using a proper naming convention, such as Definitive or Semantic naming [22]. To be accessible to different language speakers as well, language options will have to be provided.

With the Coin Password, a thin area around a grey circle is given a colour. This may be hard to differentiate for some users, and so thicker borders and colour labels would be

suitable for better accessibility.

For each method, it may also be wise to implement the choice to use colour blind-friendly palettes, which in turn can be designed for the different types of colour blindness, Protanopia, Deuteranopia and Tritanopia [23]. This can be provided alongside a colour coded legend, which can help the user further differentiate each colour.

For the Image Grid method, a pool of images must be chosen with gestalt notion of perception² [24] taken into account, for example, two pass images with high contrast compared to the rest of the pool will be more memorable for a shoulder surfer; images of different themes will also need to be considered, for example, if there are many images of sunsets used, this may confuse the user as to which one they selected.

7.2.3 Complexity and Effort

In Table C.19 we see the median time taken for the users to log in for each approach.

The Image Grid took 14 seconds, which is a fast time considering the high initial complexity of the methodology, but still slower than that of an alphanumeric or pin password. It took from 2 to 3 attempts for the user to grasp the 3 different scenarios and apply their pass images to these. With further use, its possible the time taken would not decrease much more, due to the time needed to find each pass image and in turn find a certain image to click on (Graph C.1 starts to plateau at around 10 seconds). The timings also depend on the grid size, and the largest grid size used in testing was 6. Figure C.1 shows that initially, the image grid took the least amount of time to log in and also has the shortest login time out of all the methods. The simple action of a click, compared to the multiple inputs needed in the other methodologies, also helps to make this approach quicker.

The Colour Grid also took 14 seconds, which is ≈ 2.3 seconds to make each entry based on the first letter for the colour of the character. The method was also the second-fastest in Figure C.1 to initially understand and login, and levelled out to 10-13 seconds, which is a small amount of time for the shoulder surfer to observe the process.

The Coin Password requires more concentration, as the user has to keep track of how many coins have been entered and scan 30 elements to find their next coin pass element. The median time taken was 30.5 seconds, which is double that of the Image Grid. Figure C.1 shows that despite taking over a minute initially, users were able to enter their coin password in under 20 seconds. This could be due to the repeating logins, side by side, allowing for much quicker recognition of the coin pass elements; how would login time suffer over hours to days?

The Colour Wheel had the longest login median time of 51s. Some users expressed frustration with this approach, it being "quite an involved method" which is "too time-consuming". This is an "annoyance if the user logs in several times a day". Figure C.1 shows a much longer time taken for users to work out the login phase when compared to the other methods, with one user taking 140 seconds to complete the login. The quickest login is at ≈ 30 seconds, which is still 10 seconds longer than the Coin Password. Despite its 100% shoulder surfing resistance, the *trade-off* in time is too large to

²Theorises the way the human brain analyses and interprets certain information about the hierarchy and relationships in an image, based on visual cues. Such examples are proximity, similarity and closure

consider this version of the methodology suitable for common use.

7.3 Can Graphical Passwords Replace Other Authentication Methods?

Some of the graphical passwords researched here, such as the Colour Grid [10] and Image Grid [5], present intuitive ideas which have high usability and possess strong shoulder surfing resistance. The affordances of graphical passwords allow methodologies to include high accessibility and easier password recognition through colours and images.

Participants were also tested in their ability to shoulder surf a login using the common textual form of authentication. Table C.18 exhibited a 100% shoulder surfing resistance rate for the alpha-numerical passwords via keyboard input. The reasoning for this was that users were not able to observe and memorise the key entries quick enough to solve the password. However, on occasion users guessed a couple of the password characters correctly, showing that despite the 100% resistance, further login observations may provide the shoulder surfer with the opportunity to guess more characters and reveal the password.

The form of input is also an important consideration, as participants expressed more difficulty shoulder surfing keyboard input rather than mouse clicks. The research by Tari et al [2] showed similar results when analysing the shoulder surfing resistance of the PassFaces™ graphical password using both mouse and keyboard input. Mouse clicks require the user to see what they're clicking on screen, and so a cursor is displayed, with the ramification that a shoulder surfer will be able to observe it. On the other hand, keyboard entry can be displayed in a password field that uses black circles to represent entered characters. Therefore, shoulder surfers will have to observe the physical input of keys, which depending on the input device may be hard, due to factors such as the user's hands blocking the line of vision. This is a big benefit to alphanumeric passwords when subsiding shoulder surfing.

The graphical password methodologies, on the other hand, stop the shoulder surfer from gaining information through the ambiguity of methodology itself, rather than the difficulty of observing input. For example, the Colour Wheel method [6] gives nothing away, as all possible colours and characters are visible on screen and the user makes entries via separate buttons. All the shoulder surfer would be able to observe is the rotation of the circle and how many times the user clicks to confirm entry. The Image Grid [5], perhaps less securely, as shown by its 15% less resistance in Table C.18, has multiple scenarios in place using digraph substitution rules, to conceal the pass images from a shoulder surfer through a large number of possibilities.

Most participants expressed the desire to continue using textual passwords, as they are most familiar with them and do not find the process tedious. While intuitive, the graphical passwords researched here present some issues considering usability, including time-taken to login, all of which are slower than an alphanumeric login. The *trade-offs* of the graphical approaches are generally much more unbalanced when compared to an alphanumeric approach, which requires a lot less time and effort for an approach which is still strong in its shoulder surfing resistance.

For a direct replacement to the alphanumeric password to further prevent shoulder surfing,

the Colour Grid [10] would be the best option researched in this project. This is due to its keyboard input, where even if the shoulder surfer is able to observe keyboard entry, there are still 6 different possible characters related to the each specific colour entered. Therefore, the login entry will be different every time and the shoulder surfer cannot just directly copy observed input over to the next. However, the password entropy is not as large as the alphanumeric password, it will take longer on average to login, and colour may be a usability issue to some users.

The research of graphical passwords may already be obsolete, with authentication techniques such as fingerprint scanning [25] and facial recognition [26] becoming more common in replacing alphanumeric methods; not that these new implementations come without their fair share of issues [27].

Chapter 8

Critical Appraisal and Conclusions

8.1 Summary

The main aim of the project was to “evaluate different graphical password methods in their ability to prevent shoulder surfing, whilst also being user-friendly: by creating a graphical password software testing suite”. Using the Java language and its GUI-based libraries, four approaches were implemented and combined into a “suite”. This suite contained a main page, which allowed the user to select and register a profile or log in with a method if they were already registered. These logins were enhanced with data gathering for testing, such as time taken, combinations and errors made; the Coin Password and Image Grid methods in particular included a separate window to show the user visually what the shoulder surfer (at worst case) can observe during the login or multiple logins.

All elements in the Graphical User Interface work as intended, with no function-less buttons and pop-ups. The navigation experience works smoothly, with checks in place to make sure no¹ errors corrupt the run-time of the system.

The user cannot proceed with the program unless there is a valid database connected. The database is implemented with no outstanding issues, using Java libraries to connect to an external MySQL database and run external queries.

Each registration page gathers all relevant details from the user and rejects invalid input, helping prevent errors from occurring during login. Every username constitutes a unique profile, and when making registrations for multiple approaches, the correct password must be used.

The login phases for each approach match the criteria for the research papers they were based on. There isn’t a *like-for-like* interface compared with the example diagrams shown in these papers, but this was not required to show off the functionality and make observations.

All approaches implemented were done so before user testing. User testing was performed successfully and the data gathered was useful in generating such statistics. User comments helped to explain the overall consensus regarding security and usability.

Multiple shoulder surfing factors were considered for evaluation, from single by-stander attacks to multiple observation attacks, and the shoulder surfing success rate when the mechanism was known or unknown. Usability was also analysed from user experience (7.2.3) and accessibility (7.2.1, 7.2.2) perspectives.

The graphical passwords were compared with other authentication counterparts, and the

¹As mentioned in subsection 8.2, the Image Grid contains a logic error based on an internet connection seen in Table B.4 test 1.3.

research efforts in this field were concluded to be possibly futile, despite their intuitive techniques to prevent attacks such as shoulder surfing.

8.2 Improvements and Future work

This subsection highlights where some areas of the project could be improved, and with more time what could be added to the project.

8.2.1 Research Data

While 160 login attempts may have been analysed with 4 volunteers, the accuracy of results could have been improved with a wider pool of users. This would challenge the researched methods to hold up against many users with different skills and recognition abilities. However, with the importance of shoulder surfing resistance listed in the creation of these methods, research with larger sample size is likely to replicate the results here.

If there was a longer time window for research, the success rate for user and shoulder surfer memorability could be researched over days and months, rather than logins alongside each other². This would challenge the user to have to remember their passwords, and potentially cause some usability restraints for the researched methods. On the other hand, this sort of research would be hard to implement in reality, as it would require users who are willing to partake in multiple surveys, and to remember their passwords as if they were consequences for not doing so.

With a wider pool of data, it would be useful to utilize the in-built database for this purpose, which follows onto the next section.

8.2.2 Database Additions

Having a database already been implemented, it would be useful to gather login data such as time taken, errors made, shoulder surfing attempts needed so this could be stored for research.

Using a database on the network would also allow research to be taken virtually, as users would be able to download the suite and connect to the online database and input data. This however would require a lot more time and serious consideration for data integrity and security.

The implementation of some approaches stored and fetched user details in plain text, such as the `wheelPass` for the Colour Wheel method. To increase security, these will have to be encrypted instead.

8.2.3 Front-end and Back-end Architecture

As previously mentioned in 6.1, much of the functionality is contained in methods relating to the graphical elements, so it was hard to test much of the code without intervention from

²In reality, once a user has logged in they are unlikely to be required to log in again until the next session.

the interface. At the start of the project, it would have been appropriate to separate these parts of the design, using a certain architecture, like MVC for example.

8.2.4 Approach Improvements

The login works correctly for each scenario [5], although there is no dialogue box if the user makes an incorrect entry. A user could effectively select every image until they login, but this would be an easy fix with the addition of a warnings system as seen in the Colour Wheel implementation. On a very small occasion, the `makeGrid` method generates duplicate images from the PicSum [18] website. The `buff_imgsEqual` method was made to mitigate this, however only works for preset local images. However, for research purposes, this implementation works to a satisfactory level to provide accurate shoulder surfing data. In future development, it would be wise to remove the dependencies of external websites for images, using them fully locally instead to prevent errors when fetching.

The Coins for the Coin Password method could be made more accessible, for example using thicker colour borders, as previously mentioned in 7.2.2, and a brighter background to account for the contrasts of different colours.

8.2.5 Mock Shoulder Surfer Improvements

The shoulder surfer windows could be improved, for example, the use of machine learning solutions that guess the users' passwords overtime. These results could be stored in a more in-depth database solution.

The Coin Password and Image Grid shoulder surfers encounter some issues when calculating possible elements, for example, the `ImageGridSurfer.java` will only display 1 of the pass images and a non-pass image that keeps appearing. This error may be because valid images may also be the pass images themselves if they are below or to the right of another pass image in scenarios B and C respectively.

8.3 Socio-economic Impact

This research drew some interesting points about how we approach the utilisation of authentication methodologies. If companies were to start implementing such graphical passwords for their trivial nature and/or shoulder surfing strength, they should first make sure that such methods:

- Are as quick and simple to use as possible; the user should not have to spend time working out the login process.
- The password is memorable over a long period of time and easily transferable across different devices and their input methods.
- Include accessibility options; this is important to accommodate early in design.

Shoulder surfing is not as prevalent as other attacks [1], such as phishing, credential stuffing and dictionary attacks. This alludes that companies should not focus on implementing a method to directly oppose shoulder surfing attacks.

The methodologies behind graphical passwords require more work on implementation compared to simple alphanumeric implementations. Consequently, graphical passwords would require more time, and therefore more cost than that of traditional methodologies, which already have many solutions widely available, and even free to use.

Graphical passwords may be a good alternative for some users. For example, those on mobile who do not wish to use facial or fingerprint recognition, but would like a shoulder surfing resistant solution, may wish to pick the Coin Password. The methodology is similar to that of a numeral pin-code, but introduces multiple elements to throw off shoulder surfers. Other users may decide to use graphical passwords based on their accessibility, say the Image Grid for its use of images, rather than security reasons.

Device-based attributes are also important. For example, it would be more important to consider shoulder surfing for smartphones, which are used in public more frequently, compared to a PC based in an secure office environment. As seen in section 7.3, the input methods are also of importance when considering shoulder surfing and directly relate to the device used itself.

Businesses must keep these things in mind when choosing authentication for their software, whilst users should be made aware of the alternatives, including benefits and downfalls, with the help of research like this.

8.4 Personal Development

The project has introduced many challenges to me over the course of it's duration. During the initial research and planning stages, I was tasked with finding suitable approaches to use throughout my project, and decide what to factor in when evaluating their ability to avoid shoulder surfing. This will help me facilitate software engineering projects in the real world, which will require deep thinking to provide a solution that satisfies many factors.

To implement these methods, I had to learn about the methods used in Java Application design, and apply the chosen methodologies to these to create a practical piece of software, that allowed me to experiment with the approaches and carry out user testing. Every methodology provided me with a different challenge. On one hand I had to provide a way of generating and storing Images, and calculating valid user input based on the different digraph scenarios of the Image Grid method. On another hand, I had to learn how to use graphics packages to create a fully functioning colour wheel that is based on certain variables, such as the placement of characters and rotation position of each colour. I also had to learn how to connect a database to this suite and run scripts based on user input, through the graphical elements. As well as this, the continual improvement of the program saw a large array of changes, those of which made the software bug-free and easily navigable to users.

While developing this project, I had to manage my tasks and time effectively. Figures D.1 and D.2 show the use of GitLab to label and prioritise tasks. I had to follow a strict personal schedule via my Gantt Chart, as seen on GitLab. Communication skills were also improved with meetings and email updates on a regular basis, presentations and interviews.

All these skills can be added to my professional portfolio, which will greatly influence my future career development.

Bibliography

- [1] M. Eiband, M. Khamis, E. von Zezschwitz, H. Hussmann, and F. Alt, "Understanding shoulder surfing in the wild: Stories from users and observers," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ser. CHI '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 4254–4265. [Online]. Available: <https://doi.org/10.1145/3025453.3025636>
- [2] F. Tari, A. Ozok, and S. Holden, "A comparison of perceived and real shoulder-surfing risks between alphanumeric and graphical passwords," vol. 149, 01 2006, pp. 56–66.
- [3] L. Bošnjak and B. Brumen, "Improving the evaluation of shoulder surfing attacks," 06 2018, pp. 1–2.
- [4] P. Jiang, Q. Wen, W. Li, Z. Jin, and H. Zhang, "An anonymous and efficient remote biometrics user authentication scheme in a multi server environment," *Front. Comput. Sci.*, vol. 9, no. 1, p. 142–156, feb 2015. [Online]. Available: <https://doi.org/10.1007/s11704-014-3125-7>
- [5] A. I. T. F. A. Lip Yee POR, Chin Soon KU, "Graphical password: prevent shoulder-surfing attack using digraph substitution rules," *Frontiers of Computer Science*, vol. 11, no. 6, p. 1098, 2017.
- [6] S. Sonkar, R. L. Paikrao, A. Kumar, and S. B. Deshmukh, "Minimizing shoulder surfing attack using text and color based graphical password scheme," *International journal of engineering research and technology*, vol. 3, 2014.
- [7] N. Projects. Graphical password to avoid shoulder surfing. [Online]. Available: <https://nevonprojects.com/graphical-password-to-avoid-shoulder-surfing/>
- [8] T. Fong, A. Abdullah, N. Zaman, and M. Supramaniam, "The coin passcode: A shoulder-surfing proof graphical password authentication model for mobile devices," *International Journal of Advanced Computer Science and Applications*, vol. 10, 01 2019.
- [9] A. J. Aviv, J. T. Davin, F. Wolf, and R. Kuber, "Towards baselines for shoulder surfing on mobile authentication," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, ser. ACSAC 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 486–498. [Online]. Available: <https://doi.org/10.1145/3134600.3134609>
- [10] M. Thirunavukkarasu, "An improving method of grid graphical password authentication system," *International Journal of Engineering Research and Applications*, vol. 07, pp. 40–43, 2017.
- [11] A. J. Lin and F. F. Cheng, "A free drawing graphical password scheme," *Computer-Aided Design and Applications*, vol. 6, no. 4, pp. 553–561, 2009. [Online]. Available: <https://www.tandfonline.com/doi/abs/10.3722/cadaps.2009.553-561>

- [12] S. Wiedenbeck, J. Waters, J.-C. Birget, A. Brodskiy, and N. Memon, "Passpoints: Design and longitudinal evaluation of a graphical password system," *Int. J. Hum.-Comput. Stud.*, vol. 63, no. 1–2, p. 102–127, jul 2005. [Online]. Available: <https://doi.org/10.1016/j.ijhcs.2005.04.010>
- [13] S. Ariffin, M. F. Abdul Kadir, A. Rose, M. A. Mohamed, and A. Mamat, "User authentication using grid based method," *International Journal of Engineering and Technology(UAE)*, vol. 7, pp. 97–100, 04 2018.
- [14] Oracle. (2020) Java. [Online]. Available: <https://www.oracle.com/uk/java/technologies/javase/jdk14-archive-downloads.html>
- [15] ——. (2021) Mysql. [Online]. Available: <https://www.mysql.com/>
- [16] J. Zukowski, *The Definitive Guide to Java Swing, Third Edition (Definitive Guide)*. USA: Apress, 2005.
- [17] Oracle. Mysql workbench. [Online]. Available: <https://dev.mysql.com/downloads/workbench/>
- [18] N. Y. David Marby. Picsum. [Online]. Available: <https://picsum.photos/>
- [19] P. Kua, "Unit testing." [Online]. Available: <https://www.thekua.com/publications/AppsUnitTesting.pdf>
- [20] K. Renaud, G. Johnson, and J. Ophoff, *Dyslexia and Password Usage: Accessibility in Authentication Design*, 08 2020, pp. 259–268.
- [21] E. Chisom, "Understanding dyslexia," 04 2016.
- [22] K. Muldoon, Mar 2022. [Online]. Available: <https://uxdesign.cc/how-should-you-name-your-colors-in-a-design-system-3086513476df>
- [23] R. Cravit, "How to use color blind friendly palettes to make your charts accessible," Nov 2021. [Online]. Available: <https://venngage.com/blog/color-blind-friendly-palette/>
- [24] J. Wagemans, "Historical and conceptual background," *Oxford Handbooks Online*, 2014.
- [25] N. Sulaiman and Q. Ariffin, *Overview on Fingerprinting Authentication Technology*, 03 2020, pp. 451–462.
- [26] M. Zulfiqar, F. Syed, M. J. Khan, and K. Khurshid, "Deep face recognition for biometric authentication," in *2019 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, 2019, pp. 1–6.
- [27] B. Joseph, "Our faces are replacing alphanumeric passwords. we should all be worried." Aug 2018. [Online]. Available: <https://medium.com/swlh/our-faces-are-replacing-alphanumeric-passwords-and-we-should-all-be-worried-433608d0ccc5>

Appendix A

Unit Tests

Test No.	Name	Description	Result
1	testValidRegistration	Test insertCoinPassDetails with valid input data	Passed
2	testInvalidRegistration	Duplicate user registration	Passed
3	testInvalidInitialLogin	Test checkCoinPassDetails with invalid userID	Passed
4	testValidInitialLogin	Test with a valid userID	Passed
5	testMakeCoins()	Test makeCoins method	Passed

Table A.2: Coin Password Unit Tests.

Test No.	Name	Description	Result
1	testValidRegistration	Test insertColourGridDetails with valid input data	Passed
2	testInvalidRegistration	Duplicate user registration	Passed
3	testInvalidInitialLogin	Test checkColourGridDetails with invalid userID	Passed
4	testValidInitialLogin	Test with a valid userID	Passed
5	testMakeGrid()	Test makeGrid method	Passed*

Table A.4: Colour Grid Unit Tests.

Test No.	Name	Description	Result
1	testValidRegistration	Test insertColourWheelDetails with valid input data	Passed
2	testInvalidRegistration	Duplicate user registration	Passed
3	testInvalidInitialLogin	Test checkColourWheelDetails with invalid userID	Passed
4	testValidInitialLogin	Test with a valid userID	Passed

Table A.6: Colour Wheel Unit Tests

Test No.	Name	Description	Result
1	testValidRegistration	Test insertImageGridDetails with valid input data	Passed
2	testInvalidRegistration	Duplicate user registration	Passed
3	testInvalidInitialLogin	Test checkImageGridDetails with invalid userID	Passed
4	testValidInitialLogin	Test with a valid userID	Passed
5	testMakeGrid()	Test makeGrid method	Passed

Table A.8: Image Grid Unit Tests

Appendix B

Integration Tests

Colour Grid Integration Tests				
Test No.	Description	Test	Expected Result	Actual Result
1	Make sure a 6x6 grid is displayed with an even distribution of colours in random locations	Login a user through Colour Grid Initial Login	Colour Grid Login window should display the grid as stated	As expected
2	Make every character, A-Z and 0-9, is visible in order	Login a user through Colour Grid Initial Login	Colour Grid Login window displays grid with all characters	As expected
3.1	Make sure the user can login by entering the first letters of the colours of the pattern pass characters	In Colour Grid Login, enter all correct first letters of colours and click login	Dialog box to show that the user successfully logged in, as well as giving the time taken and shoulder surfer information	As expected
3.2	Make sure the user can login by entering the first letters of the colours of the pattern pass characters	In Colour Grid Login, enter incorrect first letters of colours and click login	Dialog box to show incorrect login	As expected

Table B.2: Integration Tests for the Colour Wheel Approach.

Image Grid Integration Tests				
Test No.	Description	Test	Expected Result	Actual Result
1.1	Image Grid Method displays images based on random and preset selection in the login page " "	Go to Image Grid Method Initial Login; input correct details for a user with preset images	Image Grid Login should generate a grid of preset images	As expected
1.2		Go to Image Grid Method Initial Login; input correct details for a user with random images	Image Grid Login should generate a grid of random images	As expected
1.3		Same steps as 1.2, however the internet connection is disabled.	Image Grid Login should give an error	As expected - error
2	Make sure the user can login by selecting one of two correct pass images in the Image Grid Method	Go to Image Grid Login; click on a correct pass image based the given scenario (A,B or C)	A new grid should be made and the shoulder surfer window be updated, or shoulder surfer evaluation appear if the pass images have been uncovered	As expected
3.1	Make sure the Image Grid Surfer updates the correct possible images for every login the user completes	Go to Image Grid Login; a correct pass image is selected	All the images above, below, to the left and right are added to the shoulder surfer in the first login	As expected
3.2	Make sure the Image Grid Surfer updates the correct possible images for every login the user completes	3.1 completed; another correct pass image is selected	All the images above, below, to the left and right are compared with those from the first login, and the repeated images remain in the shoulder surfer window	As expected
4	Make sure the correct amount of images are displayed based on the users selected grid size	Register a user with the grid size of 5; go to login with this user	A 5x5 grid of images should be displayed	As expected

Table B.4: Integration Tests for the Image Grid Approach.

Coin Password Integration Tests				
Test No.	Description	Test	Expected Result	Actual Result
1	Make sure all elements appear in the registration and are select-able	Go to Coin Password Registration, click on "Bike" icon if it appears	All elements appear and the Bike when clicked adds ".1.png" to the "Coin Pass" field	As expected
2	Make sure all coins are generated randomly for each login and that each element appears once	Go through the Coin Password Initial Login; into the Coin Password Login window	Every element should appear at once, 10 coins altogether pseudo-randomly generated	As expected
3.1	When a user clicks on the correct coin including their current element, the login should regenerate the coins for the next coin pass element	Go through Initial Login; user clicks on the first correct coin that corresponds to their coin pass	Another grid of coins should be regenerated; shoulder surfer updated	As expected
3.2	When a user clicks on a incorrect coin, they should have to input their password again	Go through Initial Login; user clicks on an incorrect coin that does not correspond to their coin pass	Dialog box appears telling the user to login again. They are then taken back to the start of the coin pass; shoulder surfer is restarted	As expected
4	When the last coin has been correctly chosen, the user should be given an evaluation of what the shoulder surfer gathered	Go through initial login and coin password login; with the last correct coin clicked	The shoulder surfer window should change to show an evaluation including the correct password length and error counts	As expected
5	Make sure the Coin Password Surfer correctly updates the viewed elements based on errors	In Coin Password Login; user inputs 2 correct coins and then makes an incorrect input	The shoulder surfer window will update the first 2 (after their re-entries) coins removing the elements which do not appear again	As expected

Table B.6: Integration Tests for the Coin Password Approach.

Colour Wheel Integration Tests				
Test No.	Description	Test	Expected Result	Actual Result
1	Make sure the colours appear in registration and once a colour is picked this is displayed	Go to Colour Wheel Registration; click on the "pink" colour label	The label should be displayed alongside the "Selected Colour:"	As expected
2	Make sure that the wheel is displayed will all required characters pseudo-randomly distributed to each 8 sectors	Go through Colour Wheel Initial Login	A colour wheel with 8 sectors should be displayed, every required character should be placed once somewhere randomly within the wheel	As expected
3.1	Make sure clockwise wheel turning button is functional	In Colour Wheel Login; user clicks on clockwise button	All the colours move once clockwise, the text stays the same	As expected
3.2	Make sure anti-clockwise wheel turning button is functional	In Colour Wheel Login; user clicks on anti-clockwise button	All the colours move once anti-clockwise, the text stays the same	As expected
4.1	Make sure the confirm button works on correct and incorrect entry	In Colour Wheel Login; user turns their colour to correspond with their current wheel pass character; clicks "Confirm"	No dialog box, the user can continue confirming the next character	As expected
4.2	Make sure the confirm button works on correct and incorrect entry	In Colour Wheel Login; user turns their colour to not correspond with their current wheel pass character; clicks "Confirm"	Dialog box showing the current amount of warnings. On the third warning the user is taken out of the login screen	As expected
5.1	Make sure the login button works on correct and incorrect entry	In Colour Wheel Login; user confirms all correct characters of their password; clicks "Login"	The shoulder surfing evaluation is displayed via a dialog box, with time taken	As expected
5.2	" "	Without any characters confirmed the user clicks "Login"	Incorrect dialog box is displayed	As expected

Table B.8: Integration Tests for the Colour Wheel Approach.

General Integration Tests				
Test No.	Description	Test	Expected Result	Actual Result
1	Make sure dialog boxes appear and close on "Ok" button	Input database settings and click on register to show a dialog box; Click on "Ok" button within dialog box	The dialog box should close and Welcome page be opened	As expected
2	Make sure scrolling bar moves the page via user input	Click and drag scroll bar and use scrolling wheel on welcome page	The text on the page should move up and down	As expected
3	Make sure register button goes to correct page that is selected within the drop down box of the welcome page	Select "Colour Grid Method" in drop down box and click register	Window should change to a Colour Grid Registration page	As expected
4	Make sure login button goes to correct page that is selected within the drop down box of the welcome page	Select "Colour Wheel Method" in drop down box and click login	Window should change to a Colour Wheel Initial Login page	As expected
5	The "Reset DB" button should initialize the database if it does not exist, or reinitialize if it does	Click on "Reset DB" button	The database should be reinitialized	As expected
6	User can login via textual password and see the time taken	Go to Simple Textual Method Login; input correct details and click "Login" button	User should be given a dialog box which shows the time taken to login and what the shoulder surfer could gather	As expected

Table B.10: Integration Tests for general functions and the welcome page.

Appendix C

User Tests

C.0.1 User Testing Participation Consent Form

I agree to the participate in the research led by Adam Barnes for his CO3201 module. I understand:

- Once I participate, I can withdraw at any time
- I can refuse to answer questions or perform tests without any consequences
- All information I provide will be treated anonymously
- The purpose of this study has been explained to me and I have the opportunity to ask any questions during or after participation

Signatures for this form are available on the projects GitLab.

C.0.2 User A

Approach	Attempts				
	1	2	3	4	5
Colour Grid	No	No	No	No	No
Colour Wheel	No	No	No	No	No
Colour Wheel (knowing colour)	No	No	No	No	No
Coin Pass	No	No	No	No	No
Image Grid	No	No	No	No	No

Table C.1: User A results for shoulder surfing whilst not knowing each mechanism.

Approach	Times to Login	Comments
Colour Grid	29s, 17s, 10s	Quite easy to use; would be happy to use;
Colour Wheel	139s, 80s	Secure; takes a long time; would be happy to use; need a way of knowing if a letter is "locked in"; much harder to guess than the colour grid
Coin Pass	62s, 32s, 24s	Could learn to do it quickly; lots of changes every time; preferred method; ease of use; may be good for those with dyslexia, reading difficulty, children etc
Image Grid	14s avg, 11s avg	Would be easy to shoulder surf; probably not happy using this method

Table C.3: User A approach feedback.

Approach	Attempts				
	1	2	3	4	5
Textual Password	No	No	No	No	No
Colour Grid	No	No	No	No	No
Colour Wheel	No	No	No	No	No
Colour Wheel (knowing colour)	No	No	No	No	No
Coin Pass	No	No	No	No	No
Image Grid	No	No	No	No	No

Table C.4: User A results for shoulder surfing after knowing each mechanism.

C.0.3 User B

Approach	Attempts				
	1	2	3	4	5
Colour Grid	No	No	No	No	No
Colour Wheel	No	No	No	No	No
Colour Wheel (knowing colour)	No	No	No	No	No
Coin Pass	No	No	No	No	No
Image Grid	No	No	Yes	Yes	Yes

Table C.5: User B results for shoulder surfing whilst not knowing each mechanism.

Approach	Times to Login	Comments
Colour Grid	41s, 11s, 13s	Satisfactory time to login; hard to observe and calculate each first letter; may make a lot of mistakes
Colour Wheel	58s, 48s, 54s	Guessed the mechanism before they were told; quite secure; but too time consuming; would not be happy with this method
Coin Pass	61s, 35s, 17s	Got easier the more they used it; very secure; user friendly; would be able to guess someones password with multiple attempts; better balance of security and usability; preferred login
Image Grid	25s avg, 14s avg	Usability quite good; needs to be more secure; not happy with using this method

Table C.7: User B approach feedback.

Approach	Attempts				
	1	2	3	4	5
Textual Password	No	No	No	No	No
Colour Grid	No	No	No	No	No
Colour Wheel	No	No	No	No	No
Colour Wheel (knowing colour)	No	No	No	No	No
Coin Pass	Yes	No	No	No	No
Image Grid	No	No	No	No	No

Table C.8: User B results for shoulder surfing after knowing each mechanism.

C.0.4 User C

Approach	Attempts				
	1	2	3	4	5
Colour Grid	No	No	No	No	No
Colour Wheel	No	No	No	No	No
Colour Wheel (knowing colour)	No	No	No	No	No
Coin Pass	No	No	No	No	No
Image Grid	No	No	No	No	No

Table C.9: User C results for shoulder surfing whilst not knowing each mechanism.

Approach	Times to Login	Comments
Colour Grid	18s, 10s	Not accessibility-friendly due to colour; secure as it will take too long to observe each letters for each colours of entry
Colour Wheel	32s, 28s	Significantly longer to submit password, which is an annoyance if the user logs in several times a day; eye-tracking and screen watching could be used to crack; not accessibility friendly; easy to identify length of password by counting the times of confirmation
Coin Pass	45s, 29s	Not accessibility friendly; pass can be identified after several observations; easy to observe length of password
Image Grid	14s avg, 9s avg	Intuitive design, complexity of the methodology is a hindrance to the shoulder surfer; fairly easy for smaller grid size; multiple times for each login would be beneficial; pass images may be the only images to appear again

Table C.11: User C approach feedback.

Approach	Attempts				
	1	2	3	4	5
Textual Password	No	No	No	No	No
Colour Grid	No	No	No	No	No
Colour Wheel	No	No	No	No	No
Colour Wheel (knowing colour)	No	No	No	No	No
Coin Pass	No	No	No	No	No
Image Grid	No	No	No	No	No

Table C.12: User C results for shoulder surfing after knowing each mechanism.

C.0.5 User D

Approach	Attempts				
	1	2	3	4	5
Colour Grid	No	No	No	No	No
Colour Wheel	No	No	No	No	No
Colour Wheel (knowing colour)	No	No	No	No	No
Coin Pass	No	No	No	No	No
Image Grid	No	No	No	No	No

Table C.13: User D results for shoulder surfing whilst not knowing each mechanism.

Approach	Times to Login	Comments
Colour Grid	23s, 14s, 12s	a mind-boggle but would be happy to use; would not be able to shoulder surf this method; good login timing despite the complexity
Colour Wheel	64s, 29s, 27s	Quite an involved method; could not shoulder surf with so many permutations; 30 seconds is a long time to log in for a 5 letter password; could only shoulder surf by looking at users eyes, even then still need the colour;
Coin Pass	20s, 17s	Only problems with the rough implementation; Colour is a big focus which could be an issue; 3 different modes of information may be tough to remember; open to using; relatively intuitive
Image Grid	10s avg	Relatively intuitive, with initial knowledge barrier for shoulder surfers; easy to use once the mechanism is known; shoulder surfing is hard but possible after multiple tries

Table C.15: User D approach feedback.

Approach	Attempts				
	1	2	3	4	5
Textual Password	No	No	No	No	No
Colour Grid	No	No	No	No	No
Colour Wheel	No	No	No	No	No
Colour Wheel (knowing colour)	No	No	No	No	No
Coin Pass	No	No	No	No	No
Image Grid	No	No	No	No	No

Table C.16: User D results for shoulder surfing after knowing each mechanism.

C.0.6 Results

Approach	Resistance rate
Without knowing the mechanism	
Colour Grid	100%
Colour Wheel	100%
Colour Wheel (knowing colour)	100%
Coin Pass	100%
Image Grid	100%
After knowing the mechanism	
Textual Password	100%
Colour Grid	100%
Colour Wheel	100%
Colour Wheel (knowing colour)	100%
Coin Pass	75%
Image Grid	100%

Table C.17: The shoulder surfing resistance-rate of each approach after a single shoulder surfing attempt.

Approach	Resistance rate
Without knowing the mechanism	
Colour Grid	100%
Colour Wheel	100%
Colour Wheel (knowing colour)	100%
Coin Pass	100%
Image Grid	85%
After knowing the mechanism	
Textual Password	100%
Colour Grid	100%
Colour Wheel	100%
Colour Wheel (knowing colour)	100%
Coin Pass	95%
Image Grid	100%

Table C.18: The shoulder surfing resistance-rate of each approach after multiple shoulder surfing attempts.

Approach	Median Time Taken
Colour Grid	14s
Colour Wheel	51s
Coin Pass	30.5s
Image Grid	14s

Table C.19: The median time taken to login for each approach.

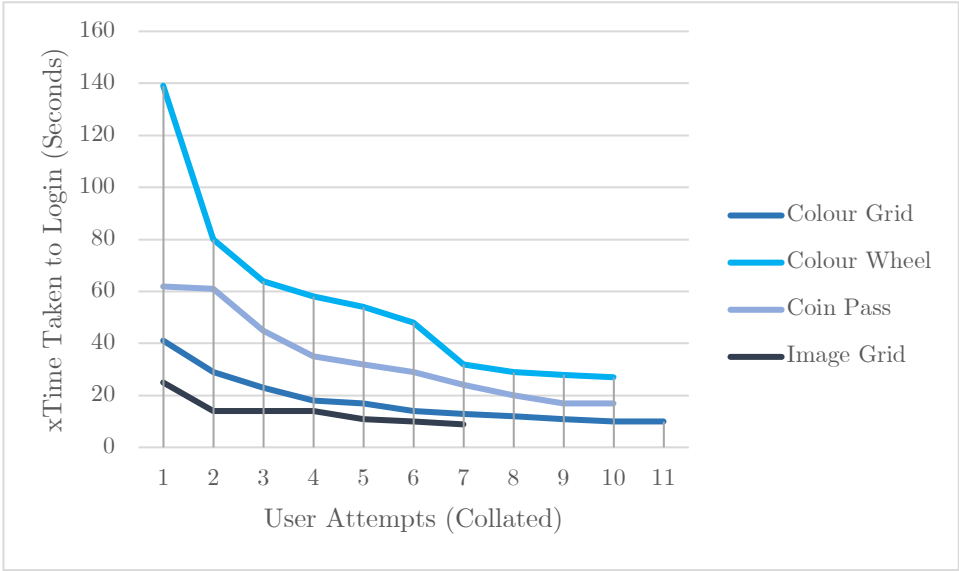


Figure C.1: A line graph showing the logged times taken to login from longest to shortest.

Appendix D

Time Management

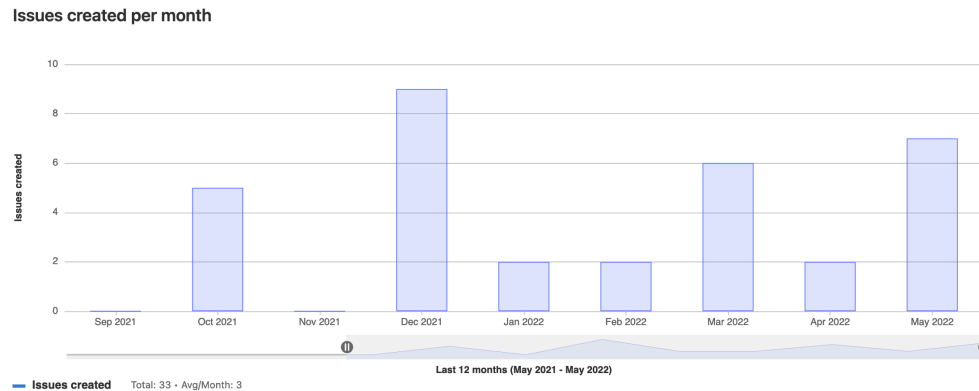


Figure D.1: A graph showing the list of issues created on GitLab over the duration of the project.

Colour Wheel Login #23 · created 1 month ago by A Barnes · Colour Wheel Graphical Password · Mar 14, 2022 · Software System · priority high · semester 2	CLOSED · updated 1 month ago
Colour Wheel Registration #22 · created 1 month ago by A Barnes · Colour Wheel Graphical Password · Mar 9, 2022 · Software System · priority high · semester 2	CLOSED · updated 1 month ago
Chapter 1 Introduction #21 · created 2 months ago by A Barnes · Dissertation · semester 2	CLOSED · updated 1 month ago
Coin Password Login #20 · created 2 months ago by A Barnes · Coin Password Implementation · Software System · priority high · semester 2	CLOSED · updated 1 month ago
Coin Password Registration #19 · created 2 months ago by A Barnes · Coin Password Implementation · Software System · priority high · semester 2	CLOSED · updated 1 month ago
Digraph Image Grid Login Form #18 · created 2 months ago by A Barnes · (Digraph) Image Grid Graphical Password · Feb 18, 2022 · Software System · priority high · semester 2	CLOSED · updated 2 months ago
Digraph Image Grid Registration Form #17 · created 2 months ago by A Barnes · (Digraph) Image Grid Graphical Password · Feb 14, 2022 · Software System · priority high · semester 2	CLOSED · updated 2 months ago
Grid Method Grids Selection and Login #16 · created 3 months ago by A Barnes · Colour Grid Method Graphical Password Implementation · Software System · priority high · semester 2	CLOSED · updated 2 months ago
Grid Method Registration #15 · created 3 months ago by A Barnes · Colour Grid Method Graphical Password Implementation · Software System · priority high · semester 2	CLOSED · updated 3 months ago
Add Grid Method to Welcome Drop Down & Register application #14 · created 4 months ago by A Barnes · Colour Grid Method Graphical Password Implementation · Jan 2, 2022 · Software System · priority high · semester 1	CLOSED · updated 3 months ago

Figure D.2: A screenshot showing some of the issues created, their labelling and naming.