# Abstract

*Computer security depends largely on passwords to authenticate human users from attackers. The most common computer authentication method is to use alphanumerical usernames and passwords. However, this method has been shown to have significant drawbacks. For example, users tend to pick passwords that can be easily guessed. On the other hand, if a password is hard to guess, then it is often hard to remember. To address this problem, some researchers have developed authentication methods that use pictures as passwords. In this paper, we conduct a comprehensive survey of the existing graphical password techniques and provide a possible theory of our own.*

The most common computer authentication method is

to use alphanumerical usernames and passwords. This

method has been shown to have significant drawbacks.

For example, users tend to pick passwords that can be

easily guessed. On the other hand, if a password is

hard to guess, then it is often hard to remember. To

address this problem, some researchers have

developed authentication methods that use pictures as

passwords. In this paper, we conduct a comprehensive

survey of the existing graphical password techniques.

We classify these techniques into two categories:

recognition-based and recall-based approaches. We

discuss the strengths and limitations of each method

and point out the future research directions in this

area. We also try to answer two important questions:

“Are graphical passwords as secure as text-based

passwords?”; “What are the major design and

implementation issues for graphical passwords?”

This survey will be useful for information security

researchers and practitioners who are interested in

finding an alternative to text-based authentication

method

The most common computer authentication method is

to use alphanumerical usernames and passwords. This

method has been shown to have significant drawbacks.

For example, users tend to pick passwords that can be

easily guessed. On the other hand, if a password is

hard to guess, then it is often hard to remember. To

address this problem, some researchers have

developed authentication methods that use pictures as

passwords. In this paper, we conduct a comprehensive

survey of the existing graphical password techniques.

We classify these techniques into two categories:

recognition-based and recall-based approaches. We

discuss the strengths and limitations of each method

and point out the future research directions in this

area. We also try to answer two important questions:

“Are graphical passwords as secure as text-based

passwords?”; “What are the major design and

implementation issues for graphical passwords?”

This survey will be useful for information security

researchers and practitioners who are interested in

finding an alternative to text-based authentication

method

The most common computer authentication method is

to use alphanumerical usernames and passwords. This

method has been shown to have significant drawbacks.

For example, users tend to pick passwords that can be

easily guessed. On the other hand, if a password is

hard to guess, then it is often hard to remember. To

address this problem, some researchers have

developed authentication methods that use pictures as

passwords. In this paper, we conduct a comprehensive

survey of the existing graphical password techniques.

We classify these techniques into two categories:

recognition-based and recall-based approaches. We

discuss the strengths and limitations of each method

and point out the future research directions in this

area. We also try to answer two important questions:

“Are graphical passwords as secure as text-based

passwords?”; “What are the major design and

implementation issues for graphical passwords?”

This survey will be useful for information security

researchers and practitioners who are interested in

finding an alternative to text-based authentication

method

The most common computer authentication method is

to use alphanumerical usernames and passwords. This

method has been shown to have significant drawbacks.

For example, users tend to pick passwords that can be

easily guessed. On the other hand, if a password is

hard to guess, then it is often hard to remember. To

address this problem, some researchers have

developed authentication methods that use pictures as

passwords. In this paper, we conduct a comprehensive

survey of the existing graphical password techniques.

We classify these techniques into two categories:

recognition-based and recall-based approaches. We

discuss the strengths and limitations of each method

and point out the future research directions in this

area. We also try to answer two important questions:

“Are graphical passwords as secure as text-based

passwords?”; “What are the major design and

implementation issues for graphical passwords?”

This survey will be useful for information security

researchers and practitioners who are interested in

finding an alternative to text-based authentication

method

The most common computer authentication method is

to use alphanumerical usernames and passwords. This

method has been shown to have significant drawbacks.

For example, users tend to pick passwords that can be

easily guessed. On the other hand, if a password is

hard to guess, then it is often hard to remember. To

address this problem, some researchers have

developed authentication methods that use pictures as

passwords. In this paper, we conduct a comprehensive

survey of the existing graphical password techniques.

We classify these techniques into two categories:

recognition-based and recall-based approaches. We

discuss the strengths and limitations of each method

and point out the future research directions in this

area. We also try to answer two important questions:

“Are graphical passwords as secure as text-based

passwords?”; “What are the major design and

implementation issues for graphical passwords?”

This survey will be useful for information security

researchers and practitioners who are interested in

finding an alternative to text-based authentication

method

*The most common computer authentication method is to use alphanumerical usernames and passwords. This method has been shown to have significant drawbacks. For example, users tend to pick passwords that can be easily guessed. On the other hand, if a password is hard to guess, then it is often hard to remember. To address this problem, some researchers have developed authentication methods that use pictures as passwords. In this paper, we conduct a comprehensive survey of the existing graphical password techniques. We classify these techniques into two categories: recognition-based and recall-based approaches. We discuss the strengths and limitations of eac*h method and point out the future research directions in this area. We also try to answer two important questions: “Are graphical passwords as secure as text-based passwords?”; “What are the major design and implementation issues for graphical passwords?” This survey will be useful for information security researchers and practitioners who are interested in finding an alternative to text-based authentication method.

*Graphical password authentication is a means of authentication that requires the recall and selection of images or sections of an image inputted during the registration phase in a graphical user interface. Today, access to computer systems is most often based on the use of alphanumeric passwords. Though, users have difficulty remembering a password that is long and rando mappearing. Instead, they create short, simple, and insecure passwords. Graphical passwords have been designed to try to make passwords more memorable and easier for people to use and, therefore, more secure. Using a graphical password, users click on images rather than type alphanumeric characters*

# Introduction

Human factors are often considered the weakest link in a computer security system. If we point out that there are three major areas where human-computer interaction is important: authentication, security operations, and developing secure systems.

Here we focus on the authentication problem. User authentication is a fundamental component in most computer security contexts. Studies showed that since user can only remember a limited number of passwords, they tend to write them down or will use the same passwords for different accounts. To address the problems with traditional username-password authentication, alternative authentication methods, such as biometrics, have been used. In this paper, however, we will focus on another alternative: using image as passwords.

Human factors are often considered the weakest link

in a computer security system. Patrick, et al. [1] point

out that there are three major areas where human-

computer interaction is important: authentication,

security operations, and developing secure systems.

Here we focus on th e authentication problem.

The most common computer authentication method is

for a user to submit a user name and a text password.

The vulnerabilities of this method have been well

known. One of the main problems is the difficulty of

remembering passwor ds. Studies have sh own that

users tend to pick short passwords or passwords that

are easy to remember [2]. Unfortunately, these

passwords can also be easily guessed or broke

Human factors are often considered the weakest link

in a computer security system. Patrick, et al. [1] point

out that there are three major areas where human-

computer interaction is important: authentication,

security operations, and developing secure systems.

Here we focus on th e authentication problem.

The most common computer authentication method is

for a user to submit a user name and a text password.

The vulnerabilities of this method have been well

known. One of the main problems is the difficulty of

remembering passwor ds. Studies have sh own that

users tend to pick short passwords or passwords that

are easy to remember [2]. Unfortunately, these

passwords can also be easily guessed or broke

Human factors are often considered the weakest link

in a computer security system. Patrick, et al. [1] point

out that there are three major areas where human-

computer interaction is important: authentication,

security operations, and developing secure systems.

Here we focus on th e authentication problem.

The most common computer authentication method is

for a user to submit a user name and a text password.

The vulnerabilities of this method have been well

known. One of the main problems is the difficulty of

remembering passwor ds. Studies have sh own that

users tend to pick short passwords or passwords that

are easy to remember [2]. Unfortunately, these

passwords can also be easily guessed or broke

Human factors are often considered the weakest link in a computer security system. Patrick, et al. [1] point out that there are three major areas where human-computer interaction is important: authentication, security operations, and developing secure systems. Here we focus on th e authentication problem. The most common computer authentication method is for a user to submit a user name and a text password. The vulnerabilities of this method have been well known. One of the main problems is the difficulty of remembering passwor ds. Studies have sh own that users tend to pick short passwords or passwords that are easy to remember. Unfortunately, these passwords can also be easily guessed or broke.

According to a recent Computerworld news article, the security team at a large company ran a network password cracker and within 30 seconds, they identified about 80% of the passwords. On the other hand, passwords that are hard to guess or break are often hard to remember. Studies showed that since user can only remember a limited number of passwords, they tend to write them down or will use the same passwords for different accounts.

To address the problems with traditional username-

password authentication, alternative authentication

methods, such as biometr ics [3, 7], have been used. In

this paper, however, we will focus on another

alternative: using pictures as passwords.

To address the problems with traditional username-password authentication, alternative authentication methods, such as biometr ics, have been used. In this paper, however, we will focus on another alternative: using pictures as passwords.

This paper will be particularly useful for researchers

who are interested in developing new graphical

password algorithms as well as industry practitioners

who are interested in deploying graphical password

techniques.

This paper will be particularly useful for researchers

who are interested in developing new graphical

password algorithms as well as industry practitioners

who are interested in deploying graphical password

techniques.

This paper will be particularly useful for researchers who are interested in developing new graphical password algorithms as well as industry practitioners who are interested in deploying graphical password techniques.

2. Overview of the Authentication

Methods

Current authentication methods can be divided into

three main areas:

• Token based authentication

• Biometric based authentication

• Knowledge based authentication

Token based techniques, such as key cards, bank cards

and smart cards are widel y used. Many token-based

authentication systems also use knowledge based

techniques to enhance security. For example, ATM

cards are generally used together with a PIN number.

Biometric based authentication techniques, such as

fingerprints, iris scan, or facial recognition, are not yet

widely adopted. The major drawback of this approach

is that such systems can be expensive, and the

identification process can be slow and often

unreliable. However, this type of technique provides

the highest level of security.

Knowledge based techniques are the most widely used

authentication techniques and include both text-based

and picture-based passwords. The picture-based

techniques can be further divided into two categories:

recognition-based and recall-based graphical

techniques. Using recognition-based techniques, a user

is presented with a set of images and the user passes

the authentication by recognizing and identifying the

images he or she selected during the registration stage.

Using recall-based techniques, a user is asked to

reproduce something that he or she created or selected

earlier during the registration stage.

1. **Overview of the Authentication Methods**

Current authentication methods can be divided into three main areas:

• Token based authentication

• Biometric based authentication

• Knowledge based authentication

Token based techniques, such as key cards, bank cards and smart cards are widel y used. Many token-based authentication systems also use knowledge based techniques to enhance security. For example, ATM cards are generally used together with a PIN number.

Biometric based authentication techniques, such as fingerprints, iris scan, or facial recognition, are not yet widely adopted. The major drawback of this approach is that such systems can be expensive, and the identification process can be slow and often unreliable. However, this type of technique provides the highest level of security.

Knowledge based techniques are the most widely used authentication techniques and include both text-based and picture-based passwords. The picture-based techniques can be further divided into two categories: recognition-based and recall-based graphical techniques. Using recognition-based techniques, a user is presented with a set of images and the user passes the authentication by recognizing and identifying the images he or she selected during the registration stage. Using recall-based techniques, a user is asked to reproduce something that he or she created or selected earlier during the registration stage.

# Graphical Password

Graphical passwords refer to using pictures (also drawings) as passwords. In theory, graphical passwords are easier to remember, since humans remember pictures better than words. Also, they should be more resistant to brute- force attacks, since the search space is practically infinite.

In general, graphical passwords techniques are classified into two main categories: recognition-based and recall- based graphical techniques.

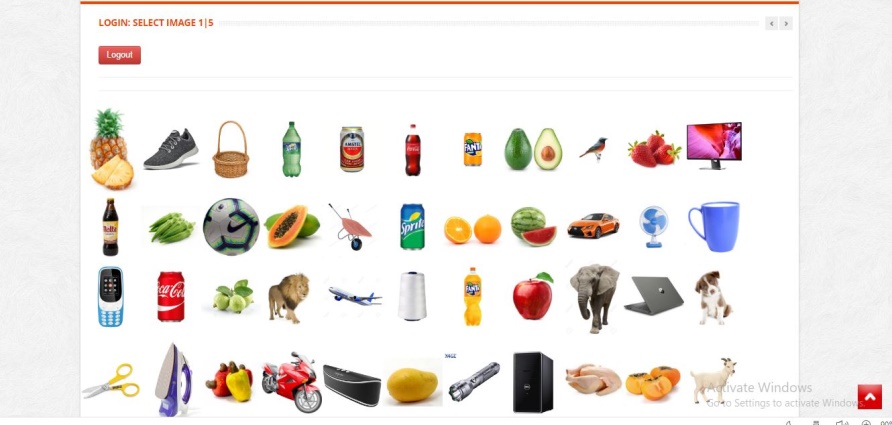
With increasing technical advancements the world is becoming digital at a high pace and everything is happening online. From paying your bills to ticket bookings to paying the person sitting next to you, you prefer to pay online. Not only payments but all activities, be it, communication through e-mails and messaging apps, keeping your documents in a digital locker, etc happen online.

With everything turning online, the risk of cybercrimes and privacy breaches is also increasing. Passwords play a huge role in keeping your data safe online as well as offline platforms. Passwords are the default method of authentication to get access to our accounts. There are various types of authentication available for users to secure their accounts.

Recognition based Authentication: A user is given a set of images and he has to identify the image he selected during registration.

For example, Passfaces is a graphical password scheme based on recognizing human faces. During password creation, users are given a large set of images to select from. To log in, users have to identify the pre-selected image from the several images presented to him.

Recall based Authentication: A user is asked to reproduce something that he created or selected at the registration stage. For example, in the Passpoint scheme, a user can click any point in an image to create the password and a tolerance around each pixel is calculated. During authentication, the user has to select the points within the tolerance in the correct sequence to login.

Cued Recall: Cued Click Points (CCP) is an alternative to the PassPoints technique. In CCP, users click one point on each image rather than on five points on one image (unlike PassPoints). It offers cued-recall and instantly alerts the users if they make a mistake while entering their latest click-point.

## Recognition Based System

In recognition-based techniques, a user is authenticated by challenging him/her to identify one or more images he or she chooses during the registration stage. Recognition- based systems, also known as cognometric systems [4] or searchmetric systems [3], generally require that users memorize a portfolio of images during password creation, and then to log in, must recognize their images from among decoys. Humans have exceptional ability to recognize images previously seen, even those viewed very briefly [8], [9]. From a security perspective, such systems are not suitable replacements for text password schemes, as they have password spaces comparable in cardinality to only 4 or 5 digit PINs (assuming a set of images whose cardinality remains reasonable, with respect to usability). Recognition based systems have been proposed using various types of images, most notably: faces, random art, everyday objects, and icons. Renaud [3] discusses specific security and usability considerations, and offers usability design guidelines focusing on recognition-based systems.

In some graphical password schemes, the system must retain knowledge of some details of the shared secret, i.e.,user specific profile data e.g. in recognition schemes, the system must know which images belong to a user’s portfolio in order to display them. This information must be stored such that its original form is available to the system (possibly under reversible encryption), and thus may be available to anyone gaining access to the stored information.

E.g. Phishing attack and shoulder surfing attack.

Dhamija and Perrig [4] proposed a graphical

authentication scheme based on the Hash

Visualization technique [9]. In their system, the user is

asked to select a certain number of images from a set

of random pictures generated by a program (figure 1).

Later, the user will be required to identify the pre-

selected images in order to be authenticated. Th

Dhamija and Perrig [4] proposed a graphical authentication scheme based on the Hash Visualization technique [9]. In their system, the user is asked to select a certain number of images from a set of random pictures generated by a program (figure 1). Later, the user will be required to identify the pre-selected images in order to be authenticated. The results showed that 90% of all participants succeeded in the authentication using this technique, while only 70% succeeded using text-based passwords and PINS. The average log-in time, however, is longer than the traditional approach. A weakness of this system is that the server needs to store the seeds of the portfolio images of each user in plain text. Also, the process of selecting a set of pictures from the picture database can be tedious and time consuming for the user.

Akula and Devisetty’s algorithm [10] is similar to the technique proposed by Dhamija and Perrig [4]. The difference is that by using hash function SHA-1, which produces a 20 byte output, the authentication is secure and require less memory. The authors suggested a possible future improvement by providing persistent storage and this could be deployed on the Internet, cell phones and PDA's.

Weinshall and Kirkpatrick [11] sketched several authentication schemes, such as picture recognition, object recognition, and pseudo word recognition, and conducted a number of user studies. In the picture recognition study, a user is trained to recognize a large set of images (100 – 200 images) selected from a database of 20,000 images. After one to three months, users in their study were able to recognize over 90% of the images in the training set. This study showed that pictures are the most effective among the three schemes tested. Pseudo codes can also be used, but require proper setting and training.

Sobrado and Birget [12] developed a graphical password technique that deals with the shoulder-surfing problem. In the first scheme, the system will display a number of pass-objects (pre-selected by user) among many other objects. To be authenticated, a user needs to recognize pass-objects and click inside the convex hull formed by all the pass-objects (figure 2). In order to make the password hard to guess, Sobrado and Birget suggested using 1000 objects, which makes the display very crowded and the objects almost indistinguishable, but using fewer objects may lead to a smaller password space, since the resulting convex hull can be large. In their second algorithm, a user moves a frame (and the objects within it) until the pass object on the frame lines up with the other two pass-objects. The authors also suggest repeating the process a few more times to minimize the likelihood of logging in by randomly clicking or rotating. The main drawback of these algorithms is that the log in process can be slow.

Man, et al. [14] proposed another shoulder -surfing resistant algorithm. In this algorithm, a user selects a number of pictures as pass-objects. Each pass-object has several variants and each variant is assigned a unique code. During authentication, the user is challenged with several scenes. Each scene contains several pass-objects (each in the form of a randomly chosen variant) and many decoy-objects. The user has to type in a string with the unique codes corresponding to the pass-object variants present in the scene as well as a code indicating the relative location of the pass-objects in reference to a pair of eyes. The argument is that it is very hard to crack this kind of password even if the whole authentication process is recorded on video because where is no mouse click to give away the pass-object information. However, this method still requires users to memorize the alphanumeric code for each pass-object variant. Hong, et al. [13] later extended this approach to allow the user to assign their own codes to pass-object variants. Figure 3 shows the log-in screen of this graphical passwor d scheme. However, this method still forces the user to memorize many text strings and therefore suffer from the many drawbacks of text-based passwords.

## Recall Based System

In recall-based techniques, a user is asked to reproduce something that he or she created or selected earlier during the registration stage. Recall-based graphical password systems are occasionally referred to as draw metric systems because users recall and reproduce a secret drawing. In these systems, users typically draw their password either on a blank canvas or on a grid (which may arguably act as a mild memory cue). Recall is a difficult memory task because retrieval is done without memory prompts or cues. Users sometimes devise ways of using the interface as a cue even though it is not intended as such, transforming the task into one of cued-recall, although one where the same cue is available to all users and to attackers.

Text passwords can also be categorized as using recall memory. With text passwords, there is evidence that users often include the name of the system as part of their passwords. Although there is currently no evidence of this happening with graphical passwords, it remains a plausible coping strategy if users can devise a way of relating a recall based graphical password to a corresponding account name.

These systems are generally susceptible to shoulder surfing to the extent that in many cases, the entire drawing is visible on the screen as it is being entered, and thus an attacker need accurately observe or record only one login for the entire password to be revealed.

Blonder designed a graphical password scheme in which a password is created by having the user click on several locations on an image. During authentication, the user must click on the approximate areas of those locations. The image can assist users to recall their passwords and therefore this method is considered more convenient than unassisted recall (as with a text-based password).

Passlogix has developed a graphical password system based on this idea. In their implementation, users must click on various items in the image in the correct sequence in order to be authenticated. Invisible boundaries are defined for each item in order to detect whether an item is clicked by mouse.

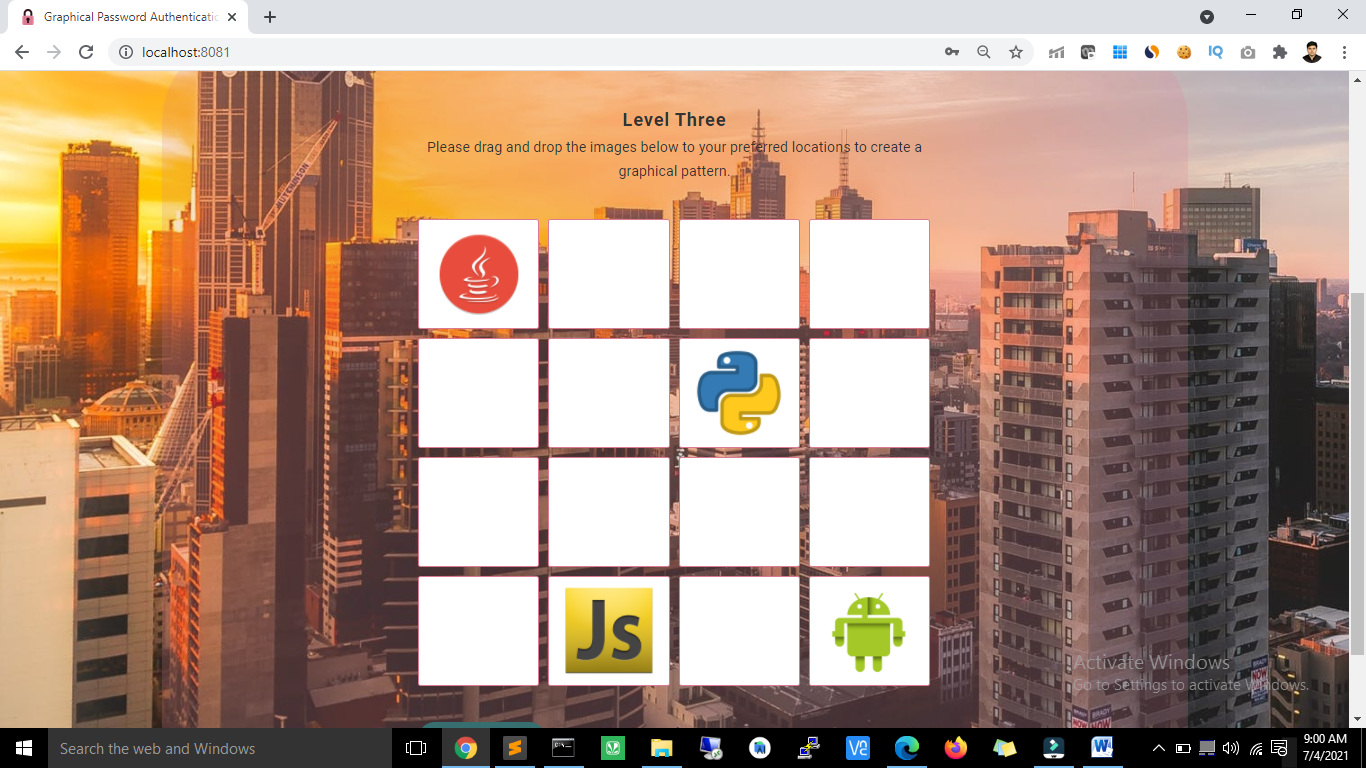
A similar technique has been developed by sfr. It was reported that Microsoft had also developed a similar graphical password technique where users are required to click on pre-selected areas of an image in a designated sequence. But details of this technique have not been available.

# Proposed System

In graphical authentication there are various techniques to secure your password. Here we are proposing a new algorithm of authentication using images. We used a grid based approach to authenticate by using image as a reference.

Shoulder surfing is a major drawback of graphical password authentication. To overcome this we have developed SSR (Shoulder Surfing Resistant) shield. The shield containing multiple fake mouse pointers are programed in such a way that it moves randomly in an image area and the original pointer will look exactly as fake mouse pointers. This shield provides a top layer for grid clicking as well as confusing other person.

At the time of registration, user will upload his/her image or set of images along with all details; then user selected image will appear on the page with transparent grid layer on it. So user will select certain grids to set his/her password as shown in the figure below.



### Grid approach.

1. **Implementation and Discussion**

The proposed system was implemented using PHP, CSS, JavaScript and Macromedia flash 2008(Action Script 2).

This Graphical Password can be implemented in authenticating several systems and websites. The implementation has few focuses:

* Login: Contains username, images, Graphical password and related methods.
* Grids: Contains unique grid values and grid clicking related methods.
* Password
* Contain image as reference & encryption algorithm.
* Color Code: Contains colored password with encryption algorithm.

As shown in the figure below researchers are trying to stabilize the goal in text based system. However, the text based approach is not able to achieve the goal because as the password strength increases usability decreases.

### 

### Usability vs Security.

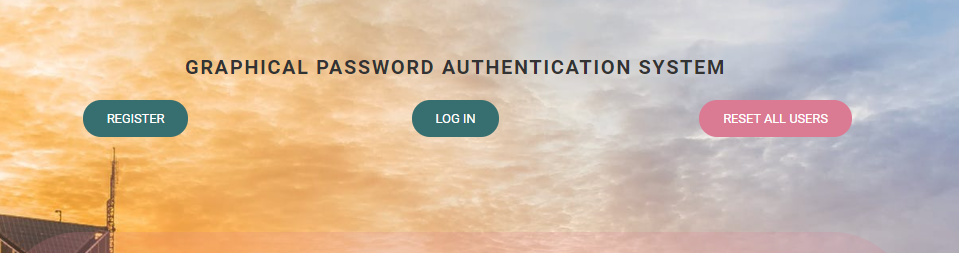
Our main aim is to achieve this goal. In which the usability as well as the security of the system is maintained in such a way that we don’t need to compromise on either of these constraints.

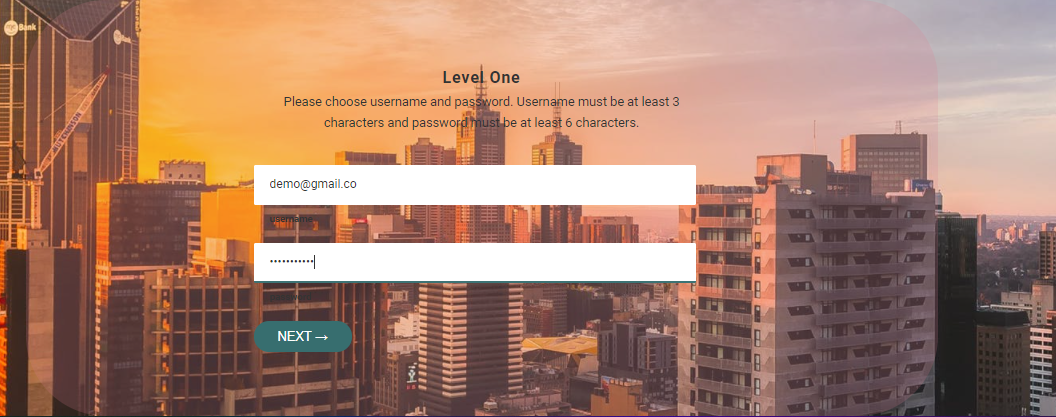
The working of our system is shown with the help of a flow graph in given figure:-

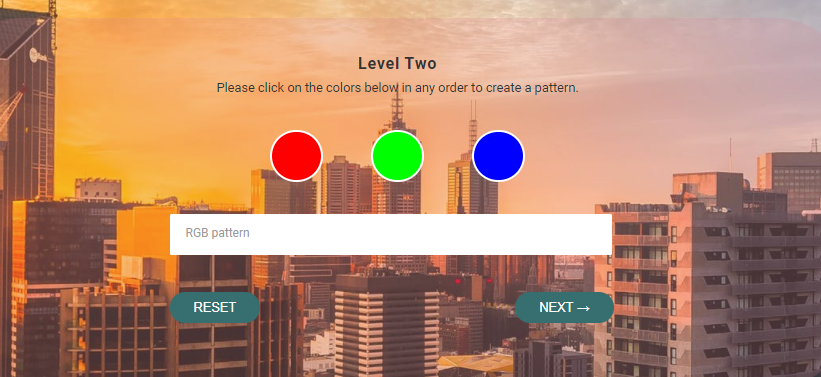
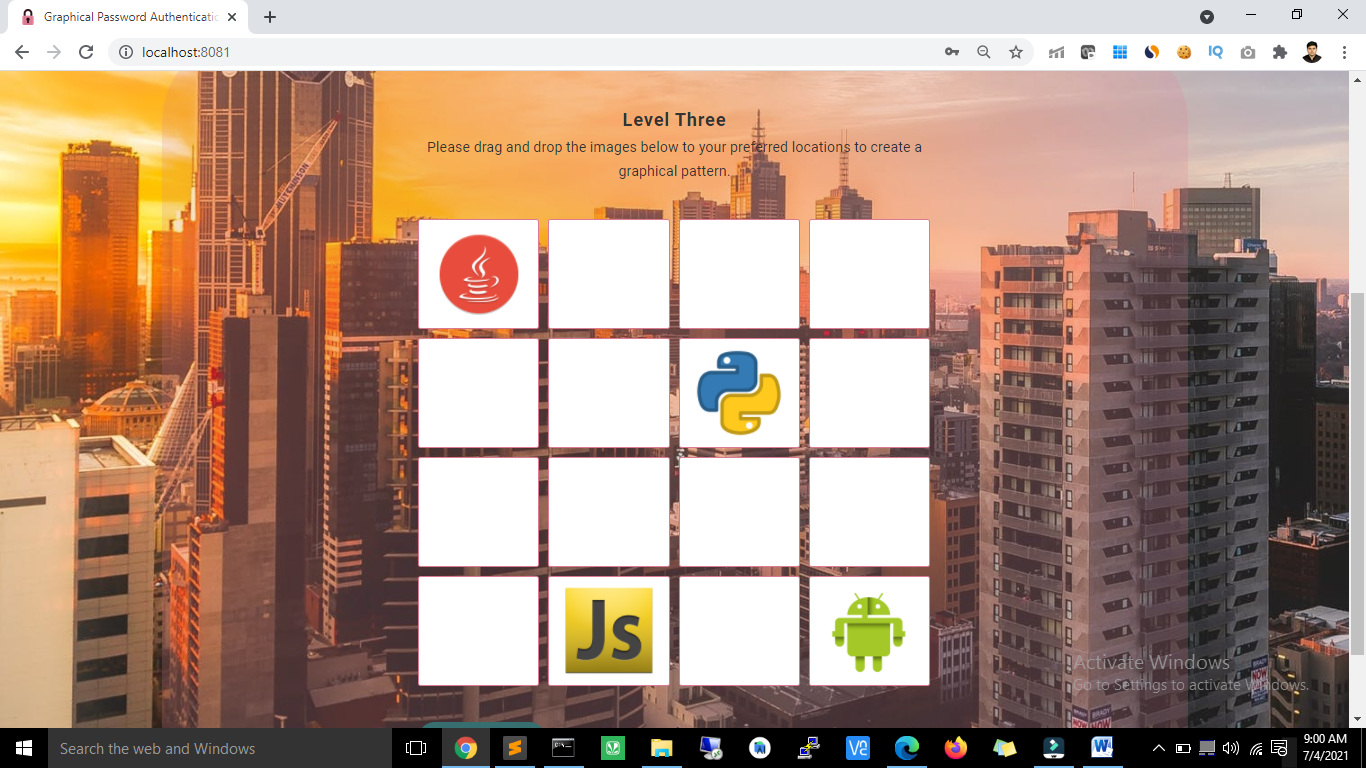


### Flow graph.

1. **Screenshots**

****

****

**** ****

**7.Advantages**

* It is user-friendly.
* It provides higher security than other traditional password schemes.
* Dictionary attacks are infeasible.
* CCP makes attacks based on hotspot analysis more challenging.

1. **Disadvantages**

* Registration and login take too long.
* It requires more storage space because of images.
* Shoulder surfing(Watching over people’s shoulders as they process information).

**9.Conclusions**

In this extended abstract we are trying to make our authentication system more user friendly and also we have tried to implement mature & fast Shoulder Surfing Resistant Mechanism. We have considered both methods: text based and graphical based systems and tried to reduce the efforts required by end-user to remember passwords. A look at the advancement in technology over the past few years tells us that the next era will have system security at its core. Thus Graphical Password may be adapted in future as a major authentication system.

Our system is a combination of recognition and recall based approach. It is more usable and secure as compare to previous graphical password authentication systems

As password space is very large it provides the security against brute force attack. It is easy to use. Passwords can be created and memorized easily

Randomization in both the authentication steps provides strong security against shoulder surfing. Overall our system is resistant to all other possible attacks also. This system can be used for highly secure systems.

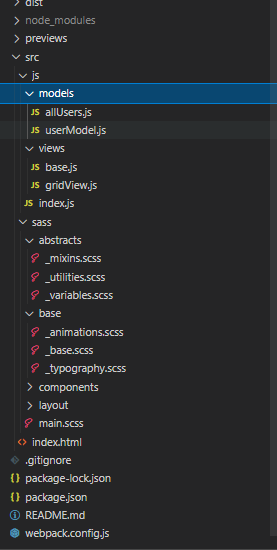
In future, one more addition possible to our system is, if the user forgets any password that password is mailed to user’s registered mail id and such a message will be sent to user’s registered mobile number also. So user can get the system updates although he is offline

Thus, in future, our system can be made more secure and easy to access

1. **References**

* [**https://idoc.pub/download/seminar-report-on-graphical-password-authentication-6nq8j11wppnw**](https://idoc.pub/download/seminar-report-on-graphical-password-authentication-6nq8j11wppnw)
* [**http://www.geeksforgeeks.com/graphical-password-authentication/**](http://www.geeksforgeeks.com/graphical-password-authentication/)
* [**https://ieeexplore.ieee.org/document/468240860000**](https://ieeexplore.ieee.org/document/468240860000)
* [**https://www.researchgate.net/publication/2210464r4286\_Graphical\_Passwords\_A\_Survey**](https://www.researchgate.net/publication/2210464r4286_Graphical_Passwords_A_Survey)
* [**https://ieeexplore.ieee.org/abstract/document/5749855000/**](https://ieeexplore.ieee.org/abstract/document/5749855000/)
* [**https://www.ijert.org/web-based-graphical-password-authentication-system**](https://www.ijert.org/web-based-graphical-password-authentication-system)

**9.Code**

****

Webpack.config

const path = require('path');

const HtmlWebpackPlugin = require('html-webpack-plugin');

module.exports = {

entry: './src/js/index.js',

output: {

path: path.resolve(\_\_dirname, 'dist'),

filename: 'js/bundle.js'

},

devServer: {

contentBase: './dist'

},

plugins: [

new HtmlWebpackPlugin({

filename: 'index.html',

template: './src/index.html'

})

]

};

Packages.json

{

"name": "authentication-system",

"version": "1.0.0",

"description": "A graphical password authentication system that uses three levels of authentication",

"main": "index.js",

"scripts": {

"watch:sass": "node-sass src/sass/main.scss dist/css/style.css -w",

"dev": "webpack --mode development",

"start": "webpack-dev-server --mode development --open"

},

"author": "Graphical Authentication System",

"license": "ISC",

"browser": {

"fs": false,

"child\_process": false

},

"devDependencies": {

"node-sass": "^5.0.0",

"html-webpack-plugin": "^3.0.7",

"webpack": "^4.2.0",

"webpack-cli": "^3.3.12",

"webpack-dev-server": "^3.1.1"

},

"dependencies": {

"sweetalert": "^2.1.2"

}

}

/Graphical-Password-Authentication/src/index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<link rel="shortcut icon" href="img/favicon.png" type="image/x-icon" />

<link rel="stylesheet" href="css/style.css" />

<link

href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap"

rel="stylesheet"

/>

<title>Graphical Password Authentication System</title>

</head>

<body style="background-image: url('https://images.pexels.com/photos/302769/pexels-photo-302769.jpeg?auto=compress&cs=tinysrgb&dpr=2&h=650&w=940'); background-repeat: no-repeat;

">

<header class="header">

<div class="row">

<div class="margin-bottom-s">

<h1 class="heading">Graphical Password Authentication System</h1>

</div>

<div class="col-1-of-3">

<a id="register" class="btn btn--primary">Register</a>

</div>

<div class="col-1-of-3">

<a id="login" class="btn btn--primary">Log In</a>

</div>

<div class="col-1-of-3">

<a

title="For the sake of convenience, this removes all the stored user information from the local storage."

id="remove"

class="btn btn--secondary"

>

reset all users

</a>

</div>

</div>

</header>

<div class="row register"></div>

<div class="row login"></div>

</body>

</html>

/Graphical-Password-Authentication/src/sass/main.scss

@import "abstracts/mixins";

@import "abstracts/variables";

@import "abstracts/utilities";

@import "base/animations";

@import "base/base";

@import "base/typography";

@import "components/button";

@import "components/circle";

@import "components/form";

@import "components/graphic";

@import "layout/container";

@import "layout/grid";

@import "layout/header";

/Graphical-Password-Authentication/src/sass/abstracts/\_mixins.scss

@mixin clearfix {

&:after {

content: "";

display: table;

clear: both;

}

}

@mixin centerHV {

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

}

/Graphical-Password-Authentication/src/sass/abstracts/\_utilities.scss

.center-text {

text-align: center !important;

}

.margin-bottom-s {

margin-bottom: 2rem !important;

}

.margin-bottom {

margin-bottom: 4rem !important;

}

/Graphical-Password-Authentication/src/sass/abstracts/\_variables.scss

//COLORS

$primary: #376e6f;

$primary-dark: #1c3334;

$secondary: #da7b93;

$secondary-dark: #2e151b;

$grey-dark-2: #999;

$grey-dark-1: #333;

$white: #fff;

$black: #000;

$red: #ff0000;

$green: #00ff00;

$blue: #0000ff;

//GRID

$grid-width: 114rem;

$gutter-vertical: 8rem;

$gutter-horizontal: 6rem;

$gutter-horizontal-s: 6rem;

//FONT

$default-font-size: 1.6rem; //16px

[/Graphical-Password-Authentication/src/sass/base/\_animations.scss](file:///D:/Habilelabs/Projects/Graphical-Password-Authentication/src/sass/base/_animations.scss)

@keyframes right {

0% {

opacity: 0;

transform: translateX(10rem);

}

100% {

opacity: 1;

transform: translate(0);

}

}

/Graphical-Password-Authentication/src/sass/base/\_base.scss

\*,

\*::after,

\*::before {

margin: 0;

padding: 0;

box-sizing: inherit;

}

html {

font-size: 62.5%;

}

/Graphical-Password-Authentication/src/sass/base/\_typography.scss

body {

font-family: "Roboto", sans-serif;

font-weight: 400;

line-height: 1.7;

color: $grey-dark-1;

background-color: $white;

}

.paragraph {

font-size: $default-font-size;

}

.heading {

font-size: 2.5rem;

text-transform: uppercase;

font-weight: 700;

display: block;

letter-spacing: 2px;

}

.sub-heading {

font-size: 2rem;

font-weight: 700;

display: block;

letter-spacing: 1px;

}

/Graphical-Password-Authentication/src/sass/components/\_button.scss

.btn {

&,

&:link,

&:visited {

text-transform: uppercase;

text-decoration: none;

padding: 1rem 3rem;

display: inline-block;

border-radius: 10rem;

transition: all 0.2s;

position: relative;

font-size: $default-font-size;

border: none;

cursor: pointer;

}

&:hover {

transform: translateY(-3px);

box-shadow: 0 1rem 2rem rgba($black, 0.2);

}

&:active,

&:focus {

outline: none;

transform: translateY(-1px);

box-shadow: 0 0.5rem 1rem rgba($black, 0.2);

}

&--primary {

background-color: $primary;

color: $white;

}

&--secondary {

background-color: $secondary;

color: $white;

}

&--right {

float: right;

margin-right: 1.5rem;

}

}

Graphical-Password-Authentication/src/sass/components/\_circle.scss

.circle {

width: 6rem;

height: 6rem;

display: inline-block;

border-radius: 50%;

border: 3px solid $white;

transition: all 0.2s;

margin: 0 auto;

cursor: pointer;

&:hover {

transform: translateY(-3px);

box-shadow: 0 1rem 2rem rgba($black, 0.2);

}

&:active,

&:focus {

outline: none;

transform: translateY(-1px);

box-shadow: 0 0.5rem 1rem rgba($black, 0.2);

}

&--red {

background-color: $red;

}

&--blue {

background-color: $blue;

}

&--green {

background-color: $green;

}

}

/Graphical-Password-Authentication/src/sass/components/\_form.scss

.form {

&\_\_group {

&:not(:last-child) {

margin-bottom: 2rem;

}

}

&\_\_input {

font-size: 1.5rem;

font-family: inherit;

color: inherit;

padding: 1.5rem 2rem;

border-radius: 2px;

background-color: $white;

border: none;

border-bottom: 3px solid transparent;

width: 90%;

display: block;

transition: all 0.3s;

&:focus {

outline: none;

box-shadow: 0 1rem 2rem rgba($black, 0.1);

border-bottom: 3px solid $primary;

&:invalid {

border-bottom: 3px solid $secondary;

}

}

&::-webkit-input-placeholder {

color: $grey-dark-2;

}

}

&\_\_input:placeholder-shown + &\_\_label {

opacity: 0;

visibility: hidden;

transform: translateY(-4rem);

}

&\_\_label {

font-size: 1.2rem;

font-weight: 700;

margin-left: 2rem;

margin-top: 0.7rem;

display: block;

transition: all 0.3s;

}

}

Graphical-Password-Authentication/src/sass/components/\_graphic.scss

.graphic {

width: 100%;

min-width: 50rem;

margin: 0 auto;

&\_\_row {

display: block;

&:not(:last-child) {

margin-bottom: 1rem;

}

@include clearfix;

&--drop {

float: left;

position: relative;

background-color: $white;

display: inline-block;

width: calc((100% - 4rem \* 3) / 4);

height: 10rem;

padding: 1rem;

border: 1px solid $secondary;

border-radius: 3px;

text-align: center;

&:not(:last-child) {

margin-right: 1rem;

}

}

&--drag {

height: 10rem;

@include centerHV;

}

}

}

Graphical-Password-Authentication/src/sass/layout/\_container.scss

.container {

padding: 4rem 0;

background-color: rgba($secondary, 0.2);

border-radius: 10rem;

margin: 0 auto;

animation-name: right;

animation-duration: .8s;

animation-timing-function: ease-out;

&\_\_form {

width: 50%;

padding: 4rem 2rem;

margin: 0 auto;

position: relative;

}

&\_\_icons {

width: 50%;

margin: 0 auto;

}

}

/Graphical-Password-Authentication/src/sass/layout/\_grid.scss

.row {

max-width: $grid-width; //1140px

margin: 0 auto;

@include clearfix;

[class^="col-"] {

float: left;

&:not(:last-child) {

margin-right: $gutter-horizontal;

}

}

.col {

&-1-of-2 {

width: calc((100% - #{$gutter-horizontal}) / 2);

}

&-1-of-3 {

width: calc((100% - #{$gutter-horizontal} \* 2) / 3);

}

&-1-of-4 {

width: calc((100% - #{$gutter-horizontal} \* 3) / 4);

}

&-1-of-5 {

width: calc((100% - #{$gutter-horizontal} \* 4) / 5);

text-align: center;

}

}

}

/Graphical-Password-Authentication/src/sass/layout/header.scss

.header {

position: relative;

padding: 10rem 0;

margin: 5rem auto;

height: 10vh;

text-align: center;

}

/Graphical-Password-Authentication/src/js/models/allUsers.js

export default class Users {

constructor() {

this.users = [];

}

getAllUsers() {

return this.users;

};

addUser(user) {

this.users.push(user);

localStorage.setItem('users', JSON.stringify(this.users));

};

restore() {

const storage = JSON.parse(localStorage.getItem('users'));

if(storage) {

this.users = storage;

}

};

reset() {

if(JSON.parse(localStorage.getItem('users'))) {

localStorage.removeItem('users');

}

}

}

/Graphical-Password-Authentication/src/js/models/userModel.js

import crypto from 'crypto';

export default class User {

constructor() {

}

addUsernameAndPassword(username, password, allUsers) {

const a = allUsers.map(el => {

if(el.username === username) {

return false;

}

});

if(a.includes(false)) {

return false;

}

this.username = username;

this.password = this.encrypt(password);

return true;

}

addPattern(pattern) {

this.pattern = this.encrypt(pattern);

}

addGrid(grid) {

this.grid = this.encrypt(grid);

}

comparePassword(password) {

return this.password === this.encrypt(password);

}

comparePattern(pattern) {

return this.pattern === this.encrypt(pattern);

}

compareGrid(grid) {

return this.grid === this.encrypt(grid);

}

encrypt(text) {

return crypto.createHash('sha256').update(text).digest('hex');

}

}

Graphical-Password-Authentication/src/js/views/base.js

import \* as Grid from './gridView';

export const elements = {

loginTop: document.querySelector('#login'),

registerTop: document.querySelector('#register'),

register: document.querySelector('.register'),

login: document.querySelector('.login'),

remove: document.querySelector('#remove')

};

export const elementStrings = {

formOne: '#form--1',

formTwo: '#form--2',

group: '.form\_\_group',

icon: '.circle',

username: '#username',

password: '#password',

pattern: '#pattern',

reset: '#reset',

nextOR: '#register--one',

nextTR: '#register--two',

nextHR: '#register--three',

nextOL: '#login--one',

nextTL: '#login--two',

nextHL: '#login--three',

drag: '.graphic\_\_row--drag',

drop: '.graphic\_\_row--drop',

}

const markups = {

one: `<div class="container"><div class="container\_\_form"><div class="margin-bottom center-text"><h2 class="sub-heading">Level One</h2><p class="paragraph">%\_DESCRIPTION%</p></div><form onsubmit="event.preventDefault();" class="form" id="form--1" autocomplete="off"><div class="form\_\_group"><input type="text" class="form\_\_input" placeholder="username" id="username" minlength="3" required /><label for="username" class="form\_\_label">username</label></div><div class="form\_\_group"><input type="password" class="form\_\_input" placeholder="password" id="password" minlength="6" required /><label for="password" class="form\_\_label">password</label></div><div class="form\_\_group"><button id="%\_BTNLEVEL%" class="btn btn--primary">Next &rarr;</button></div></form></div></div>`,

two: `<div class="container"><div class="container\_\_icons"><div class="margin-bottom center-text"><h2 class="sub-heading">Level Two</h2><p class="paragraph">%\_DESCRIPTION%</p></div><div class="row"><div class="col-1-of-5">&nbsp;</div><div class="col-1-of-5"><span id="red" class="circle circle--red">&nbsp;</span></div><div class="col-1-of-5"><span id="green" class="circle circle--green">&nbsp;</span></div><div class="col-1-of-5"><span id="blue" class="circle circle--blue">&nbsp;</span></div><div class="col-1-of-5">&nbsp;</div></div></div><div class="container\_\_form"><form class="form" id="form--2" autocomplete="off"><div class="form\_\_group"><input type="password" class="form\_\_input" placeholder="RGB pattern" id="pattern" required readonly /><label for="pattern" class="form\_\_label">pattern</label></div><div class="form\_\_group"><button type="button" id="reset" class="btn btn--primary">Reset</button><button type="button" id="%\_BTNLEVEL%" class="btn btn--primary btn--right">Next &rarr;</button></div></form></div></div></div>`,

three: `<div class="container"><div class="container\_\_form"><div class="margin-bottom center-text"><h2 class="sub-heading">Level Three</h2><p class="paragraph">%\_DESCRIPTION%</p></div><div class="graphic margin-bottom">%\_GRID%</div><div class="form\_\_group"><button type="button" id="%\_BTNLEVEL%" class="btn btn--primary">%\_BTNDESC%</button></div></div>`,

};

const replacements = {

oneLD: 'Please enter your username and password.',

oneRD: 'Please choose username and password. Username must be at least 3 characters and password must be at least 6 characters.',

twoLD: 'Please click on the colors below in the order you picked during registration.',

twoRD: 'Please click on the colors below in any order to create a pattern.',

threeLD: 'Please drag and drop the images below to the locations you specified during registration.',

threeRD: 'Please drag and drop the images below to your preferred locations to create a graphical pattern.'

};

const placeholders = {

description: '%\_DESCRIPTION%',

buttonLevel:'%\_BTNLEVEL%',

buttonDescription: '%\_BTNDESC%',

gridPlaceholder: '%\_GRID%',

};

export const clear = () => {

elements.register.innerHTML = '';

elements.login.innerHTML = '';

};

export const clearFields = () => {

const fields = document.querySelectorAll('input');

fields.forEach(el => el.value = '');

}

export const updatePattern = (color) => {

const cur = document.querySelector(elementStrings.pattern).value

const update = cur.concat(`${color}`);

document.querySelector(elementStrings.pattern).value = update;

}

export const renderOne = (type) => {

let markup = markups.one.replace(placeholders.buttonLevel, `${type}--one`);

if (type === 'login') {

markup = markup.replace(placeholders.description, replacements.oneLD);

elements.login.innerHTML = markup;

} else if (type === 'register') {

markup = markup.replace(placeholders.description, replacements.oneRD);

elements.register.innerHTML = markup;

}

};

export const renderTwo = (type) => {

let markup = markups.two.replace(placeholders.buttonLevel, `${type}--two`);

if (type === 'login') {

markup = markup.replace(placeholders.description, replacements.twoLD);

elements.login.innerHTML = markup;

} else if (type === 'register') {

markup = markup.replace(placeholders.description, replacements.twoRD);

elements.register.innerHTML = markup;

}

}

export const renderThree = (type) => {

let markup = markups.three.replace(placeholders.buttonLevel, `${type}--three`);

markup = markup.replace(placeholders.gridPlaceholder, Grid.create());

markup = markup.replace(placeholders.buttonDescription, type);

if (type === 'login') {

markup = markup.replace(placeholders.description, replacements.threeLD);

elements.login.innerHTML = markup;

} else if (type === 'register') {

markup = markup.replace(placeholders.description, replacements.threeRD);

elements.register.innerHTML = markup;

}

Grid.addImages();

}

/Graphical-Password-Authentication/src/js/views/gridView.js

export const create = () => {

return `<div class="graphic\_\_row">${addCells(1)}</div>

<div class="graphic\_\_row">${addCells(5)}</div>

<div class="graphic\_\_row">${addCells(9)}</div>

<div class="graphic\_\_row">${addCells(13)}</div>`;

}

export const addCells = (start) => {

let markup = ``;

for(let i=start; i<start+4; i++) {

markup+= `<div

class="graphic\_\_row--drop"

id="cell-${i}"></div>`;

}

return markup;

}

export const addImages = () => {

const images = ['car', 'dog', 'duck', 'flower'];

const cells = [];

while(cells.length < 4) {

let randomIndex = Math.floor(Math.random() \* 16) + 1;

if(!cells.includes(randomIndex)) {

cells.push(randomIndex);

}

}

cells.forEach((cur, i) => {

document.getElementById(`cell-${cur}`).insertAdjacentHTML('beforeend',

`<img

class="graphic\_\_row--drag"

id="${images[i]}"

src="img/${images[i]}.png"

draggable="true"/>`);

});

}

Graphical-Password-Authentication/src/js/index.js

import {elements, elementStrings, clear, clearFields, updatePattern, renderOne, renderTwo, renderThree} from './views/base';

import swal from 'sweetalert';

import User from './models/userModel';

import Users from './models/allusers';

// Reloads

const state = {};

window.addEventListener('load', () => {

state.users = new Users();

state.users.restore();

});

// Registeration Controller

elements.registerTop.addEventListener('click', () => {

clear();

renderOne('register');

state.current = new User();

});

elements.register.addEventListener('click', e => {

//1. Username and Password

if(e.target.matches(elementStrings.nextOR)) {

const created = rLevelOne(state.users);

if(created) {

clear();

renderTwo('register');

}

}

//2. RGB Pattern

if(e.target.matches(elementStrings.icon)) {

const color = e.target.closest(elementStrings.icon).id;

updatePattern(color);

} else if(e.target.matches(`${elementStrings.group}, ${elementStrings.group} \*`)) {

if(e.target.closest(elementStrings.reset)) {

clearFields();

} else if(e.target.closest(elementStrings.nextTR)) {

const patternAdded = rLevelTwo();

if(patternAdded) {

clear();

renderThree('register');

}

}

}

//3. Grid

if(e.target.matches(elementStrings.nextHR)) {

rLevelThree();

clear();

state.users.addUser(state.current);

swal('Registration successful!');

}

});

const rLevelOne = (users) => {

if(document.querySelector(elementStrings.formOne).checkValidity()) {

const unique = state.current.addUsernameAndPassword(

document.querySelector(elementStrings.username).value,

document.querySelector(elementStrings.password).value,

users.getAllUsers()

);

if(!unique) {

swal('A user with the same username already exists! Please choose another username.');

clearFields();

return false;

}

return true;

}

};

const rLevelTwo = () => {

if(document.querySelector(elementStrings.formTwo).checkValidity()) {

const pattern = document.querySelector(elementStrings.pattern).value;

if(pattern === '') {

swal('Color pattern cannot be empty! Please click on the color icons to choose a pattern.');

return false;

} else if(pattern.length < 6) {

swal('Color pattern must contain at least 2 colors!');

clearFields();

return false;

} else {

state.current.addPattern(pattern);

return true;

}

}

};

const rLevelThree = () => {

let grid = '';

const cells = document.querySelectorAll(elementStrings.drop)

const cellsArr = Array.prototype.slice.call(cells);

cellsArr.forEach(el => {

if(el.firstChild) {

grid += `${el.id}${el.firstChild.id}`

}

});

state.current.addGrid(grid);

};

// Log in Controller

elements.loginTop.addEventListener('click', () => {

clear();

renderOne('login');

});

elements.login.addEventListener('click', e => {

//1. Password

if(e.target.matches(elementStrings.nextOL)) {

const passwordMatch = oLevelOne(state.users);

if(passwordMatch) {

clear();

renderTwo('login');

}

}

//2. RGB Pattern

if(e.target.matches(elementStrings.icon)) {

const color = e.target.closest(elementStrings.icon).id;

updatePattern(color);

} else if(e.target.matches(`${elementStrings.group}, ${elementStrings.group} \*`)) {

if(e.target.closest(elementStrings.reset)) {

clearFields();

} else if(e.target.closest(elementStrings.nextTL)) {

const patternMatch = oLevelTwo();

if(patternMatch) {

clear();

renderThree('login');

}

}

}

//3. Grid

if(e.target.matches(elementStrings.nextHL)) {

const graphicMatch = oLevelThree();

if(graphicMatch) {

clear();

swal('Log in successful!');

}

}

});

const oLevelOne = (users) => {

if(document.querySelector(elementStrings.formOne).checkValidity()) {

const found = users.getAllUsers().find(el => el.username === document.querySelector(elementStrings.username).value);

if(!found) {

swal('There is no matching account for the username you entered!');

return false;

}

state.current = found;

Object.setPrototypeOf(state.current, User.prototype);

const match = state.current.comparePassword(document.querySelector(elementStrings.password).value);

if(!match) {

swal('Username and password do not match!');

return false;

}

return true;

}

};

const oLevelTwo = () => {

if(document.querySelector(elementStrings.formTwo).checkValidity()) {

const pattern = document.querySelector(elementStrings.pattern).value;

const match = state.current.comparePattern(pattern);

if(!match) {

swal('The color pattern you entered does not match that associated with the username!');

clearFields();

return false;

}

return true;

}

};

const oLevelThree = () => {

let grid = '';

const cells = document.querySelectorAll(elementStrings.drop)

const cellsArr = Array.prototype.slice.call(cells);

cellsArr.forEach(el => {

if(el.firstChild) {

grid += `${el.id}${el.firstChild.id}`

}

});

const match = state.current.compareGrid(grid);

if(!match) {

swal('The current locations of the images do not match those associated with the username!');

return false;

}

return true;

}

;

// Drag and Drop

document.addEventListener('dragover', e => {

e.preventDefault();

});

document.addEventListener('drop', e => {

e.preventDefault();

const data = e.dataTransfer.getData('image').split('.');

if (e.target.matches(`${elementStrings.drop}, ${elementStrings.drop} \*`)) {

const cell = e.target.closest(`${elementStrings.drop}`);

if (!cell.firstChild) {

cell.appendChild(document.getElementById(data[0]));

document.getElementById(data[1]).innerHTML = '';

}

}

});

document.addEventListener('dragstart', e => {

e.dataTransfer.setData('image', `${e.target.id}.${e.target.closest(`${elementStrings.drop}`).id}`);

});

// Reset all users

elements.remove.addEventListener('click', () => {

if(state.users) {

state.users.reset();

state.users = new Users();

}

clear();

})